M Reza Faisal
Ilmu Komputer, FMIPA, ULM

Full Stack Development – Bagian 1

Pada hands on labs ini akan diberikan contoh dan langkah-langkah untuk mengakses REST API pada backend dari aplikasi mobile yang dibangun dengan React Native. Penjelasan lengkap dari tentang hal ini dapat dilihat pada video berikut: https://youtu.be/LZFekwXY2pU.

Instruksi

1. Database

Buat database dengan nama media_social. Kemudian buat tabel posts dengan atribut:

- post_id.
- post_date.
- username.
- post.

Atau dapat dilihat pada Gambar 1.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**post_id**	int(11)			No	*None*		AUTO_INCREMENT
2	**post_date**	datetime			No	*None*		
3	**username**	varchar(100)	utf8mb4_general_ci		No	*None*		
4	**post**	text	utf8mb4_general_ci		No	*None*		

Gambar 1. Tabel posts.

Penjelasan detail langkah-langkah pembuatan pada database management tier ini dapat dilihat pada video berikut: https://youtu.be/rpzmw-w3-8Y.

2. Backend dengan Express JS

Backend service dibangun berbasis REST API dengan menggunakan bahasa pemrograman Node.js yang berbasis JavaScript. Untuk mempermudah pembangunan REST API digunakan framework Express JS. Berikut adalah kode yang digunakan sebagai backend service.

social_media_api.js

```
const express = require('express');
const app = express();
const mysql = require('mysql');
const port = 3000;

app.use(express.urlencoded({ extended: true }));
app.use(express.json());

// koneksi ke database
const db = mysql.createConnection({
```

```
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'media_social'
});

// menjalankan SQL - table posts
// create
app.post('/posts',(req, res) => {
    let sql = "INSERT INTO posts SET post_date=NOW()"
              +", username='"+req.body.username
              +"', post='"+req.body.post+"'";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
                  "message": "data berhasil disimpan",
                  "data":null});
    });
});

// retrieve all data
app.get('/posts',(req, res) => {
    let sql = "SELECT post_id, username, post, DATE_FORMAT(post_date, '%W %D %M %Y
%H:%i') as post_date FROM posts";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
                  "message": "data berhasil diambil",
                  "data":results});
    });
});

// retrieve a record by post_id
app.get('/posts/id/:id',(req, res) => {
    let sql = "SELECT post_id, username, post, DATE_FORMAT(post_date, '%W %D %M %Y
%H:%i') as post_date FROM posts WHERE post_id='"+req.params.id+"'";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
                  "message": "data berhasil diambil",
                  "data":results});
    });
});

// retrieve records by username
app.get('/posts/username/:username',(req, res) => {
    let sql = "SELECT post_id, username, post, DATE_FORMAT(post_date, '%W %D %M %Y
%H:%i') as post_date FROM posts WHERE username='"+req.params.username+"'";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
```

```
                    "message": "data berhasil diambil",
                    "data":results});
    });
});

// update
app.put('/posts/id/:id',(req, res) => {
    let sql = "UPDATE posts SET post='"+req.body.post+"' "
              +"WHERE post_id='"+req.params.id+"'";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
                    "message": "data berhasil diupdate",
                    "data":null});
    });
});

// delete
app.delete('/posts/id/:id',(req, res) => {
    let sql = "DELETE FROM posts WHERE post_id='"+req.params.id+"'";

    db.query(sql, (err, results) => {
        if(err) throw err;
        res.json({"status": 200,
                    "message": "data berhasil dihapus",
                    "data":null});
    });
});

app.use('/images', express.static('images'));

app.listen(port, () => {
    console.log(`cli-nodejs-api listening at http://localhost:${port}`)
});
```

Penjelasan kode di atas dapat dilihat pada beberapa video berikut:

- **REST API berbasis Express JS Part 1: Create Data** (https://youtu.be/8nm1tUBJtQM), video ini menjelaskan langkah-langkah pembuatan project `backend` dan installasi module `express` dan `mysql`. Kemudian dilanjutnya dengan pembuatan file social_media_api.js. pada video ini juga dijelaskan kode untuk koneksi database (baris ke-10 sampai ke-15) dan pembuatan endpoint untuk create (insert) data seperti yang dapat dilihat pada kode pada baris ke-19 sampai dengan 30.
- **REST API berbasis Express JS Part 2: Retrieve Data** (https://youtu.be/0P0Dr9U8WEU), video ini menjelaskan pembuatan endpoint untuk retrieve (mengambil) data dari database. Ada tiga endpoint yang dibuat pada video ini yaitu untuk mengambil seluruh record pada tabel posts (baris ke-33 sampai baris ke-42), mengambil sebuah record berdasarkan post_id (baris ke-45 sampai baris ke-54) dan endpoint untuk mengambil record berdasarkan username (baris ke-57 sampai ke-66).
- **REST API berbasis Express JS Part 3: Update & Delete Data** (https://youtu.be/ky1z5kewdIg), video ini menjelaskan pembuatan endpoint untuk update data (baris ke-69 sampai baris ke-79)

dan endpoint untuk menghapus data (baris ke-82 sampai dengan baris ke-91). Selain itu juga dijelaskan cara untuk mengelola file static (baris ke-93).

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.