RESTful API dengan NodeJs, MySQL, dan Docker

oleh Ade Putra Prima Suhendri

A. RESTful API dengan NodeJs, MySQL, dan Docker

Pada project kali ini kita akan membuat RESTful API sederhana untuk data user menggunakan NodeJs dan Mysql.

B. Dokumentasi API

1. User - Get all user data

Method : **GET**

URL : **/user**

Parameter Body :

Field	**Type**	**Description**
status	Boolean	Status of response (true/ false)
message	String	Message of response
data	Array	Array contain user data :

Field	**Type**	**Description**
id	Integer	ID of the user
name	String	Name of the user
email	String	Email of the user

2. User - Create new user

Method : **POST**

URL : **/user**

Parameter Body :

Field	**Type**	**Description**
name	String	Name of the user
email	String	Email of the user
password	String	Password of the user
password_confirm	String	Confirm Password of the user

Response :

Field	**Description**
status	Status of response (true/ false)
message	Message of response
data	ID of the user

3. User - Update user

Method : **PUT**

URL : **/user**

Parameter Body :

Field	**Type**	**Description**
name	String	Name of the user
email	String	Email of the user
id	Integer	ID of the user

Response :

Field	**Description**
status	Status of response (true/ false)
message	Message of response

4. User - Delete user

Method : **DELETE**

URL : **/user**

Parameter Body :

Field	**Type**	**Description**
id	Integer	ID of the user

Response :

Field	**Description**
status	Status of response (true/ false)
message	Message of response

C. Struktur File Project

Berikut adalah struktur file project yang akan kita buat :

```
app-root/
│
├── app/
│      ├── package.json
│      ├── routes/
│      │     └── route.js
│      ├── .env
│      ├── db.js
│      └── server.js
├── docker-compose.yml
└── Dockerfile
```

package.json

File *package.json* adalah file yang berisi deskripsi project javascript kita, secara sederhana *package.json* adalah prosedur yang akan dijalankan oleh npm / yarn.

```
{
  "name": "node_mysql_docker",
  "version": "1.0.0",
  "description": "Restful API Node.js MySQL run on Docker",
  "author": "Ade Putra Prima Suhendri <admin@kasehat.co.id>",
  "main": "server.js",
  "scripts": {
     "start": "nodemon server.js"
  },
  "dependencies": {
     "bcrypt": "^5.0.0",
     "body-parser": "^1.19.0",
     "cors": "^2.8.5",
     "dotenv": "^8.1.0",
     "express": "^4.16.1",
     "mysql": "^2.18.1",
     "nodemon": "^1.19.2"
  }
}
```

db.js

File *db.js* akan digunakan untuk mengatur koneksi ke database mysql, berikut kode nya :

```
"use strict"
var mysql = require('mysql');
var connection;
module.exports = {
      DB: function () {
      connection = mysql.createConnection({
            host: process.env.DB_HOST,
            user: process.env.DB_USER,
            password: process.env.DB_PASS,
            database: process.env.DB_NAME,
            port: process.env.DB_PORT,
      });
      return connection;
      }
};
```

server.js

File *server.js* adalah aplikasi utama yang akan dijalankan pertama kali :

```
'use strict';
require('dotenv').config();
const express = require('express');
const app = express();
const BodyParser = require('body-parser');
const cors = require('cors');
const PORT = process.env.PORT;
const HOST = process.env.HOST;
app.use(cors());
app.use(BodyParser.json());
const Route = require('./routes/Route');
app.get('/', (req, res) => {
      res.send('RESTful API With NodeJs & Mysql on Docker');
  });
app.use('/user', Route);
app.listen(PORT, HOST);
```

route.js

File *route.js* digunakan untuk mengatur routing aplikasi kita :

```
const express = require('express');
const bcrypt = require('bcrypt');
const router = express.Router();
var salt = bcrypt.genSaltSync(10);
var connection = require('../db');
var response;

// GET ALL DATA USER
router.get('/', async (req, res) => {
     try {
     connection.DB().connect();
     connection.DB().query('SELECT id, name, email FROM users
ORDER BY id ASC', function (error, results) {
          if (error) {
               response = {
               status: false,
               message: error.sqlMessage,
               };
          } else {
               response = {
               status: true,
               message:"Success",
               data: results,
               };
          }
          res.json(response)
     });

     } catch (error) {
     response = {
          status: false,
          message:error.message
     };
     res.json(response)
     }

});
```

```
// CREATE USER
router.post('/', async (req, res) => {
      try {
      const name = req.body.name
      const email = req.body.email
      const password = bcrypt.hashSync(req.body.password, salt)
      if(req.body.password == req.body.confirm_password){
            connection.DB().connect()
            connection.DB().query('INSERT INTO users (name, email,
password) VALUES (?, ?, ?)', [name, email, password], (error,
results) => {
                  if (error) {
                        response = {
                        status: false,
                        message: error.sqlMessage,
                        };
                  } else {
                        response = {
                        status: true,
                        message:"Created",
                        data: results.insertId,
                        };
                  }
                  res.json(response)
            })
      }else{
            response = {
                  status: false,
                  message:"Password not match"
            };
            res.json(response)
      }
      } catch (error) {
      response = {
            status: false,
            message:error.message
      };
      res.json(response)
      }
});
```

```
// DELETE USER
router.delete('/', async (req, res) => {
      try {
      const id = req.body.id
      connection.DB().connect()
      connection.DB().query('DELETE FROM users WHERE id = ?', [id],
(error, results) => {
            if (error) {
                  response = {
                  status: false,
                  message: error.sqlMessage,
                  };
            } else {
                  response = {
                  status: true,
                  message:"Deleted",
                  };
            }
            res.json(response)
      })
      } catch (error) {
      response = {
            status: false,
            message:error.message
      };
      res.json(response)
      }
});
```

```
// UPDATE USER
router.put('/', async (req, res) => {
	try {
	const id = req.body.id
	const name = req.body.name
	const email = req.body.email
	connection.DB().connect()
	connection.DB().query(
		'UPDATE users SET name = ?, email = ? WHERE id = ?',
		[name, email, id],
		(error, results) => {
			if (error) {
				response = {
				status: false,
				message: error.sqlMessage,
			};
			} else {
				response = {
				status: true,
				message:"Updated"
			};
			}
			res.json(response)
		}
	)
	} catch (error) {
	response = {
		status: false,
		message:error.message
	};
	res.json(response)
	}

});

module.exports = router;
```

.env

file env atau Environment Variable merupakan variabel dinamis pada komputer yang dapat diakses oleh sebuah program.

```
HOST=127.0.0.1
PORT=3000

DB_HOST=localhost
```

```
DB_PORT=3306
DB_USER=root
DB_PASS=MYSQL_PASSWORD
DB_NAME=MYSQL_DATABASE_NAME
```

Pengaturan Docker

.env

```
HOST=0.0.0.0
PORT=3000

DB_HOST=localhost
DB_PORT=3306
DB_USER=root
DB_PASS=MYSQL_PASSWORD
DB_NAME=MYSQL_DATABASE_NAME
```

Dockerfile

```
FROM node:12
WORKDIR /app
COPY ./app/package.json /app
COPY ./app /app
RUN npm install
CMD ["npm", "start"]
EXPOSE 3000
```

docker-compose.yml

```
version: "3"
services:
  node:
     container_name: NODE_SERVER
     restart: always
     build: .
     volumes:
     - ./app:/app
     ports:
     - "80:3000"
     links:
     - database
     environment:
     DB_PORT: 3306
     DB_HOST: database
  database:
     container_name: DB_MYSQL
     image: mariadb
     environment:
     MYSQL_ROOT_PASSWORD: MYSQL_PASSWORD
     MYSQL_DATABASE: MYSQL_DATABASE_NAME
     ports:
     - "3306"
```

Menjalankan Docker

Ketikkan perintah berikut untuk mulai menjalankan docker pada server:

```
$ docker-compose up --build
```