# MODUL PERKULIAHAN

**Mata Kuliah** : Pengembangan Aplikasi Web  
**Materi** : Koneksi Database dan ORM dengan Sequelize  
**Pertemuan** : 04   
**Fakultas** : Teknik  
**Program Studi** : Teknologi Informasi  
**Kode MK** : AntaraTI501P  
**Disusun Oleh** : Ir.Asroni, S.T., M.Eng

## DAFTAR ISI

1. [PEMBAHASAN](#pembahasan)
2. [PREREQUISITES](#prerequisites)
3. [PRAKTIKUM](#praktikum)
   * [A. Tujuan Praktikum](#a-tujuan-praktikum)
   * [B. Materi](#b-materi)
   * [C. Metode Praktikum](#c-metode-praktikum)
   * [D. Langkah-langkah Praktikum](#d-langkah-langkah-praktikum)
4. [TUGAS](#tugas)

## PEMBAHASAN

1. Pengenalan Database dalam Pengembangan Web:
   * Jenis-jenis database (SQL vs NoSQL)
   * Pemilihan database untuk aplikasi web
2. Pengenalan ORM (Object-Relational Mapping):
   * Konsep dan manfaat ORM
   * Perbandingan ORM dengan query SQL langsung
3. Sequelize ORM:
   * Instalasi dan konfigurasi Sequelize
   * Koneksi ke database
4. Mendefinisikan Model dengan Sequelize:
   * Struktur model
   * Tipe data dan validasi
5. CRUD Operations dengan Sequelize:
   * Create (Inserting data)
   * Read (Querying data)
   * Update (Modifying data)
   * Delete (Removing data)
6. Associations dalam Sequelize:
   * One-to-One
   * One-to-Many
   * Many-to-Many
7. Migrations dengan Sequelize:
   * Membuat dan menjalankan migrations
   * Rollback migrations
8. Querying Lanjutan:
   * Filtering dan sorting
   * Pagination
   * Eager loading
9. Optimisasi Performa:
   * Indexing
   * Caching queries
10. Best Practices:
    * Penanganan error
    * Penggunaan transactions

## PREREQUISITES

Beberapa perangkat lunak yang digunakan untuk menunjang proses pembelajaran:

1. Node.js (versi LTS terbaru)
2. NPM (Node Package Manager)
3. MySQL atau PostgreSQL
4. Sequelize CLI
5. Text editor (Visual Studio Code, Sublime Text, atau Atom)
6. Postman atau tools API testing lainnya

## PRAKTIKUM

### A. Tujuan Praktikum

Praktikum ini bertujuan untuk memberikan pemahaman kepada mahasiswa tentang penggunaan database dalam pengembangan aplikasi web, khususnya dengan menggunakan ORM Sequelize. Mahasiswa akan belajar cara menghubungkan aplikasi Express.js dengan database, melakukan operasi CRUD, dan mengelola relasi antar tabel menggunakan Sequelize.

### B. Materi

1. Konfigurasi Sequelize dengan Express.js
2. Pembuatan model dan migrasi database
3. Implementasi operasi CRUD menggunakan Sequelize
4. Penggunaan associations untuk mengelola relasi antar tabel
5. Query lanjutan dengan Sequelize

### C. Metode Praktikum

1. Penjelasan materi oleh instruktur
2. Demonstrasi langsung penggunaan Sequelize dengan Express.js
3. Praktik mandiri oleh mahasiswa
4. Diskusi dan tanya jawab

### D. Langkah-langkah Praktikum

1. Lanjutkan proyek awal pertemuan sebelumnya
2. Buat sebuah database baru di sistem Anda (misalnya, **praktikum\_nim\_db** di MySQL)
3. Setup Database & Instalasi Sequelize
   * Instal *package* yang dibutuhkan di terminal:

npm install sequelize mysql2 sequelize-cli

A screenshot of a computer program

AI-generated content may be incorrect.

* + Inisialisasi Sequelize di proyek Anda:

npx sequelize-cli init

Perintah ini akan membuat folder config, models, migrations, dan seeders.

1. Konfigurasi koneksi database di file config/config.json dengan detail database Anda
   * Edit file config/config.json, hanya ubah yang development:
   * {  
      "development": {  
      "username": "root",  
      "password": "null",  
      "database": "praktikum\_nim\_db",  
      "host": "127.0.0.1",  
      "dialect": "mysql"  
      }  
     }
2. Membuat Model dan Migrasi
   * Gunakan sequelize-cli untuk membuat model Presensi beserta file migrasinya:

npx sequelize-cli model:generate --name Presensi --attributes userId:integer

* + Buka file models/presensi.js, kemudian hapus isinya dan ganti dengan kode di bawah ini:

'use strict';

const {

Model

} = require('sequelize');

module.exports = (sequelize, DataTypes) => {

class Presensi extends Model {

/\*\*

\* Helper method for defining associations.

\* This method is not a part of Sequelize lifecycle.

\* The `models/index` file will call this method automatically.

\*/

static associate(models) {

// define association here

}

}

Presensi.init({

userId: {

type: DataTypes.INTEGER,

allowNull: false,

},

nama: {

type: DataTypes.STRING,

allowNull: false,

},

checkIn: {

type: DataTypes.DATE,

allowNull: false,

},

checkOut: {

type: DataTypes.DATE,

allowNull: true, // Boleh null

}

}, {

sequelize,

modelName: 'Presensi',

});

return Presensi;

};

* + Buka file baru yang muncul di folder migrations, hapus semua isinya dan ganti dengan kode di bawah ini:

'use strict';

/\*\* @type {import('sequelize-cli').Migration} \*/

module.exports = {

async up(queryInterface, Sequelize) {

await queryInterface.createTable('Presensis', {

id: {

allowNull: false,

autoIncrement: true,

primaryKey: true,

type: Sequelize.INTEGER

},

userId: {

type: Sequelize.INTEGER,

allowNull: false

},

nama: {

type: Sequelize.STRING,

allowNull: false

},

checkIn: {

allowNull: false,

type: Sequelize.DATE

},

checkOut: {

allowNull: true, // checkOut bisa kosong saat pertama kali check-in

type: Sequelize.DATE

},

createdAt: {

allowNull: false,

type: Sequelize.DATE

},

updatedAt: {

allowNull: false,

type: Sequelize.DATE

}

});

},

async down(queryInterface, Sequelize) {

await queryInterface.dropTable('Presensis');

}

};

* + Setelah kedua file tersebut Anda buat dan isi dengan benar, jalankan perintah migrasi di terminal untuk menerapkan perubahan ini ke database Anda:

npx sequelize-cli db:migrate

Sekarang, periksa database Anda. Seharusnya sudah ada tabel baru bernama Presensis

1. Refaktor presensiController.js
   * Kita tidak lagi mengimpor array presensiRecords, melainkan model Presensi dari Sequelize:
   * // 1. Ganti sumber data dari array ke model Sequelize
   * const { Presensi } = require("../models");
   * const { format } = require("date-fns-tz");
   * const timeZone = "Asia/Jakarta";
   * exports.CheckIn = async (req, res) => {
   * // 2. Gunakan try...catch untuk error handling
   * try {
   * const { id: userId, nama: userName } = req.user;
   * const waktuSekarang = new Date();
   * // 3. Ubah cara mencari data menggunakan 'findOne' dari Sequelize
   * const existingRecord = await Presensi.findOne({
   * where: { userId: userId, checkOut: null },
   * });
   * if (existingRecord) {
   * return res
   * .status(400)
   * .json({ message: "Anda sudah melakukan check-in hari ini." });
   * }
   * // 4. Ubah cara membuat data baru menggunakan 'create' dari Sequelize
   * const newRecord = await Presensi.create({
   * userId: userId,
   * nama: userName,
   * checkIn: waktuSekarang,
   * });
   * const formattedData = {
   * userId: newRecord.userId,
   * nama: newRecord.nama,
   * checkIn: format(newRecord.checkIn, "yyyy-MM-dd HH:mm:ssXXX", { timeZone }),
   * checkOut: null
   * };
   * res.status(201).json({
   * message: `Halo ${userName}, check-in Anda berhasil pada pukul ${format(
   * waktuSekarang,
   * "HH:mm:ss",
   * { timeZone }
   * )} WIB`,
   * data: formattedData,
   * });
   * } catch (error) {
   * res.status(500).json({ message: "Terjadi kesalahan pada server", error: error.message });
   * }
   * };
   * exports.CheckOut = async (req, res) => {
   * // Gunakan try...catch
   * try {
   * const { id: userId, nama: userName } = req.user;
   * const waktuSekarang = new Date();
   * // Cari data di database
   * const recordToUpdate = await Presensi.findOne({
   * where: { userId: userId, checkOut: null },
   * });
   * if (!recordToUpdate) {
   * return res.status(404).json({
   * message: "Tidak ditemukan catatan check-in yang aktif untuk Anda.",
   * });
   * }
   * // 5. Update dan simpan perubahan ke database
   * recordToUpdate.checkOut = waktuSekarang;
   * await recordToUpdate.save();
   * const formattedData = {
   * userId: recordToUpdate.userId,
   * nama: recordToUpdate.nama,
   * checkIn: format(recordToUpdate.checkIn, "yyyy-MM-dd HH:mm:ssXXX", { timeZone }),
   * checkOut: format(recordToUpdate.checkOut, "yyyy-MM-dd HH:mm:ssXXX", { timeZone }),
   * };
   * res.json({
   * message: `Selamat jalan ${userName}, check-out Anda berhasil pada pukul ${format(
   * waktuSekarang,
   * "HH:mm:ss",
   * { timeZone }
   * )} WIB`,
   * data: formattedData,
   * });
   * } catch (error) {
   * res.status(500).json({ message: "Terjadi kesalahan pada server", error: error.message });
   * }
   * };
2. Uji endpoint presensi check in dan check out
   * Jalankan request http di postman atau thunder client:

A screenshot of a computer

AI-generated content may be incorrect.

* + Pastikan data presensi tersimpan kedalam database:

A screenshot of a phone number

AI-generated content may be incorrect.

## TUGAS

1. Screenshot endpoint presensi checkin dan checkout.
2. Screenshot database setelah presensi:
3. Screenshot endpoint report.
4. Tempatkan gambar screenshot ke file Tugas4.md
5. Push project ke GitHub
6. **Kirim link Tugas4.md dari repository GitHub ke dalam MyKlass praktikum PAW!!!**