# Database Connectivity

Node JS

### Ade Nur Hidayat

#### About Me



#### Latest Work Experience:

Middleware Development Specialist, AXA Mandiri
 Java Developer, Difini Technology
 2021 - Present
 2018 - 2021

#### Education:

Andalas University
 Bachelor of Information System

#### Social Media:

https://www.linkedin.com/in/adenurhidayat/

2013 - 2018

#### **Ground Rules**



Give full attention in class



Keep your camera on



Use raise hand or chat to ask questions

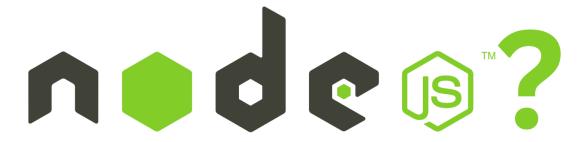


Make this room a safe place to learn and share

### Learning Objective

- Introduction Node.js dan Database
- Exploring Database Connectivity Libraries
- Establishing Database Connection
- Performing Database Operations
- Real-world Application Development

#### What is



#### Javascript as a Backend

- Express
- React
- Nest.js
- Koa.js
- Hapi.js
- Sails.js
- etc

### Database Connectivity Library

- MySql -> mysql, mysql2
- PostgreSql -> pg
- SQL Server -> mssql
- Mongodb -> mongoose
- Etc.

### Establishing Database Connection

Example of connecting to a PostgreSQL database without ORM



```
const { Client } = require('pg');
const connectionParams = {
    user: 'postgres', // Replace with your PostgreSQL username
    host: 'localhost', // Replace with your PostgreSQL host
    database: 'praktisi mengajar', // Replace with your PostgreSQL database name
    password: 'postgres', // Replace with your PostgreSQL password
    port: 5432 // Replace with your PostgreSQL port (default is 5432)
const client = new Client(connectionParams);
// Connect to the database
client.connect()
    .then(() => {
        console.log('Connected to PostgreSQL database');
        // Test the connection by running a sample query
        client.query('SELECT NOW()', (err, res) => {
            if (err) {
                console.error('Error executing query:', err);
                console.log('Current timestamp from the database:', res.rows[0].now);
            client.end();
    .catch(err => console.error('Error connecting to PostgreSQL database:', err));
```

### Establishing Database Connection using ORM

#### ORM offer several benefits:

- Increased Productivity
- Portability
- Reduced Development Time
- Improved Maintainability

### Database Connection using **ORM**

- Sequelize
- Prisma
- TypeORM
- LoopBack
- RxDB
- Mangoose

- Waterline
- CaminteJs
- BookShelf
- Objection.js
- Node-ORM2
- MikroORM

#### Command

- Using Sequelize
   npm install sequelize mysql2 -> support async/await
- Using Prisma
   npm install -g prisma
   prisma init
   prisma generate

#### Establishing Database Connection with ORM



```
Define a model representing a User table
const User = sequelize.define('User', {
 firstName:
   type: DataTypes.STRING,
   allowNull: false
 lastName: {
   type: DataTypes.STRING,
   allowNull: false
 email: {
   type: DataTypes.STRING,
   allowNull: false,
   unique: true
// Synchronize the model with the database
async function syncModel() {
   await sequelize.sync();
   console.log('User model synchronized with database.');
  } catch (error) {
   console.error('Error synchronizing model:', error);
```

Implementation

## Quiz



# Discussion

Terima Kasih