**PROJECT TOKO KOMPUTER**

**(SETTING SEQUELIZE)**

## Persiapan Database

1. Buatlah sebuah database MYSQL dengan nama “computer\_store” lalu untuk tabel dan isinya akan dibuat dengan menggunakan Library Sequelize.

## 

## Instalasi SEQUELIZE

Sequelize adalah promise-based ORM (Object Relational Mapping) untuk nodejs. ORM) adalah teknik yang memungkinkan kita untuk membuat kueri dan memanipulasi data dari database menggunakan paradigma berorientasi objek. Saat menggunakan sequelize, kita tidak harus menulis query SQL tetapi cukup menggunakan object dan function yang telah disediakan seperti findAll, findOne, create, update, destroy dan lain sebagainya.

1. Siapkan folder project bernama “**toko-komputer**”.
2. Lakukan inisiasi dengan menggunakan command **npm init --y**
3. Lakukan instal **nodemon** untuk mempermudah update setiap perubahan pada code, dengan command **npm i --save nodemon**
4. Lakukan konfigurasi **nodemon** di file **package.json**

| "scripts": {      "test": "echo \"Error: no test specified\" && exit 1",      "start": "nodemon index.js"    }, |
| --- |

1. Lakukan install Library Sequelize secara global dengan command

**npm install -g sequelize-cli**

1. Install library yang dibutuhkan yaitu sequelize, dan mysql2 dengan command

**npm i --save sequelize mysql2**

1. Eksekusi perintah berikut untuk mengijinkan pemberian perintah sequelize

**Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted**

1. Lakukan inisiasi sequelize dengan command **sequelize init**

## 

## Pembuatan Database Migration

1. Setting konfigurasi database di dalam folder “config\config.js”

| Config.js |
| --- |
| {    "development": {      "username": "root",      "password": null,      "database": "computer\_store",      "host": "127.0.0.1",      "dialect": "mysql"    },    "test": {      "username": "root",      "password": null,      "database": "database\_test",      "host": "127.0.0.1",      "dialect": "mysql"    },    "production": {      "username": "root",      "password": null,      "database": "database\_production",      "host": "127.0.0.1",      "dialect": "mysql"    }  } |

1. Selanjutnya, kita akan membuat tabel dengan menggunakan sequalize, ikutilah command dibawah ini:

* Buatlah tabel “product” pada folder migrations dengan ketik command

sequelize model:create --name product --attributes name:string,price:double,stock:double,image:string

* Buatlah tabel “admin” pada folder migrations dengan ketik command

sequelize model:create --name admin--attributes name:string,username:string,password:string

* Buatlah tabel “customer” pada folder migrations dengan ketik command sequelize model:create --name customer --attributes name:string,phone:string,address:string,image:string,username:string,password:string
* Buatlah tabel “transaksi” pada folder migrations dengan ketik command sequelize model:create --name transaksi --attributes customer\_id:integer,waktu:date
* Buatlah tabel “detail\_transaksi” pada folder migrations dengan command sequelize model:create --name detail\_transaksi --attributes transaksi\_id:integer,product\_id:integer,price:double,qty:double

1. Setelah file migration tabel terbuat, kita akan menambahkan detail tabel dan relasinya pada folder “migrations”, update code seperti berikut

| Create-products |
| --- |
| 'use strict';  module.exports = {    up: async (queryInterface, Sequelize) => {      await queryInterface.createTable('product', {        product\_id: {          allowNull: false,          autoIncrement: true,          primaryKey: true,          type: Sequelize.INTEGER        },        name: {          type: Sequelize.STRING        },        price: {          type: Sequelize.DOUBLE        },        stock: {          type: Sequelize.DOUBLE        },        image: {          type: Sequelize.STRING        },        createdAt: {          allowNull: false,          type: Sequelize.DATE        },        updatedAt: {          allowNull: false,          type: Sequelize.DATE        }      });    },    down: async (queryInterface, Sequelize) => {      await queryInterface.dropTable('product');    }  }; |

| Create-admin |
| --- |
| 'use strict';  module.exports = {    up: async (queryInterface, Sequelize) => {      await queryInterface.createTable('admin', {        admin\_id: {          allowNull: false,          autoIncrement: true,          primaryKey: true,          type: Sequelize.INTEGER        },        name: {          type: Sequelize.STRING        },        username: {          type: Sequelize.STRING        },        password: {          type: Sequelize.STRING        },        createdAt: {          allowNull: false,          type: Sequelize.DATE        },        updatedAt: {          allowNull: false,          type: Sequelize.DATE        }      });    },    down: async (queryInterface, Sequelize) => {      await queryInterface.dropTable('admin');    }  }; |

| Create-customer |
| --- |
| 'use strict';  module.exports = {    up: async (queryInterface, Sequelize) => {      await queryInterface.createTable('customer', {        customer\_id: {          allowNull: false,          autoIncrement: true,          primaryKey: true,          type: Sequelize.INTEGER        },        name: {          type: Sequelize.STRING        },        phone: {          type: Sequelize.STRING        },        address: {          type: Sequelize.STRING        },        image: {          type: Sequelize.STRING        },        createdAt: {          allowNull: false,          type: Sequelize.DATE        },        updatedAt: {          allowNull: false,          type: Sequelize.DATE        }      });    },    down: async (queryInterface, Sequelize) => {      await queryInterface.dropTable('customer');    }  }; |

| Create-Transaksi |
| --- |
| 'use strict';  module.exports = {    up: async (queryInterface, Sequelize) => {      await queryInterface.createTable('transaksi', {        transaksi\_id: {          allowNull: false,          autoIncrement: true,          primaryKey: true,          type: Sequelize.INTEGER        },        customer\_id: {          type: Sequelize.INTEGER,          allowNull: false,          references: {            model: "customer",            key: "customer\_id"          }        },        waktu: {          type: Sequelize.DATE        },        createdAt: {          allowNull: false,          type: Sequelize.DATE        },        updatedAt: {          allowNull: false,          type: Sequelize.DATE        }      });    },    down: async (queryInterface, Sequelize) => {      await queryInterface.dropTable('transaksi');    }  }; |

| Create-detail-transaksi |
| --- |
| 'use strict';  module.exports = {    up: async (queryInterface, Sequelize) => {      await queryInterface.createTable('detail\_transaksi', {        transaksi\_id: {          type: Sequelize.INTEGER,          allowNull: false,          references: {            model: "transaksi",            key: "transaksi\_id"          }        },        product\_id: {          type: Sequelize.INTEGER,          allowNull: false,          references: {            model: "product",            key: "product\_id"          }        },        qty: {          type: Sequelize.DOUBLE        },        createdAt: {          allowNull: false,          type: Sequelize.DATE        },        updatedAt: {          allowNull: false,          type: Sequelize.DATE        }      });    },    down: async (queryInterface, Sequelize) => {      await queryInterface.dropTable('detail\_transaksi');    }  }; |

1. Selanjutnya lakukan eksekusi migrate untuk membuat struktur tabel dengan

command **sequelize db:migrate**

## 

## Pengaturan Relasi pada Model

1. Pada tahap ini, kita akan menambahkan relasi, kardinalitas pada model.
2. Update data seperti contoh berikut:

| product.js |
| --- |
| 'use strict';  const {    Model  } = require('sequelize');  module.exports = (sequelize, DataTypes) => {    class product 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        this.hasMany(models.detail\_transaksi,{          foreignKey: "product\_id",          as: "transaksi product"        })      }    };    product.init({      product\_id: {        type: DataTypes.INTEGER,        primaryKey: true      },      name: DataTypes.STRING,      price: DataTypes.DOUBLE,      stock: DataTypes.DOUBLE,      image: DataTypes.STRING    }, {      sequelize,      modelName: 'product',      tableName: "product"    });    return product;  }; |

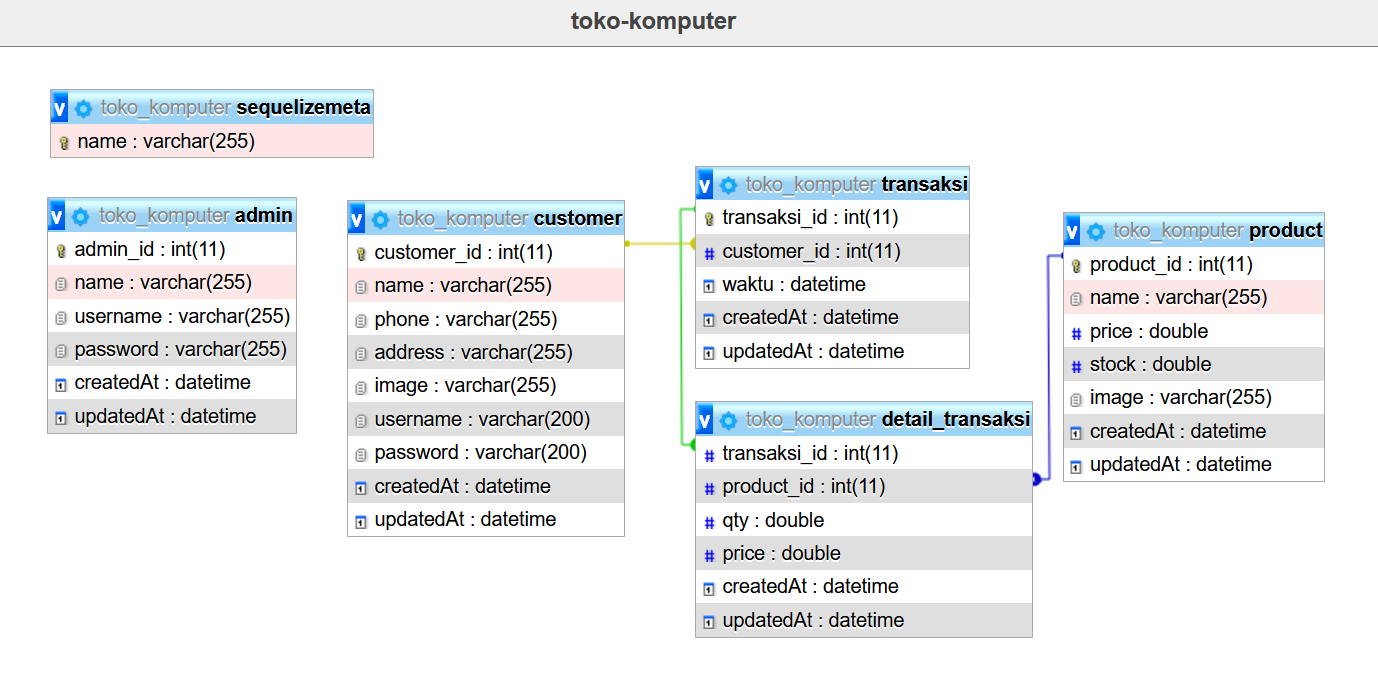
| admin.js |
| --- |
| 'use strict';  const {    Model  } = require('sequelize');  module.exports = (sequelize, DataTypes) => {    class admin extends Model {      /\*\*       \* Helper method for defining associations.       \* This method is not a part of Sequelize lifecycle.       \* The `models/index` file will call this method automatically.       \*/      static associate(models) {        // define association here      }    };    admin.init({      admin\_id: {        type:DataTypes.INTEGER,        primaryKey: true,        autoIncrement: true      },      name: DataTypes.STRING,      username: DataTypes.STRING,      password: DataTypes.STRING    }, {      sequelize,      modelName: 'admin',      tableName: 'admin'    });    return admin;  }; |

| customer.js |
| --- |
| 'use strict';  const {    Model  } = require('sequelize');  module.exports = (sequelize, DataTypes) => {    class customer 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        this.hasMany(models.transaksi,{          foreignKey: "customer\_id",          as: "transaksi customer"        })      }    };    customer.init({      customer\_id: {        type: DataTypes.INTEGER,        primaryKey: true      },      name: DataTypes.STRING,      phone: DataTypes.STRING,      address: DataTypes.STRING,      image: DataTypes.STRING,      username: DataTypes.STRING,      password: DataTypes.STRING    }, {      sequelize,      modelName: 'customer',      tableName: "customer"      });    return customer;  }; |

| transaksi.js |
| --- |
| 'use strict';  const {    Model  } = require('sequelize');  module.exports = (sequelize, DataTypes) => {    class transaksi 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        this.belongsTo(models.customer,{          foreignKey: "customer\_id",          as: "customer"        })        this.hasMany(models.detail\_transaksi,{          foreignKey: "transaksi\_id",          as: "detail\_transaksi"        })      }    };    transaksi.init({      transaksi\_id: {        type: DataTypes.INTEGER,        primaryKey: true,        autoIncrement: true      },      customer\_id: {        type: DataTypes.INTEGER,        allowNull: false      },      waktu: DataTypes.DATE    }, {      sequelize,      modelName: 'transaksi',      tableName: "transaksi",    });    return transaksi;  }; |

| detail\_transaksi.js |
| --- |
| 'use strict';  const {    Model  } = require('sequelize');  module.exports = (sequelize, DataTypes) => {    class detail\_transaksi 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  this.belongsTo(models.transaksi,{  foreignKey: "transaksi\_id",  as: "transaksi"  })  this.belongsTo(models.product,{  foreignKey: "product\_id",  as: "product"  })      }    };    detail\_transaksi.init({      transaksi\_id: {        type: DataTypes.INTEGER,        allowNull: false,        primaryKey: true      },      product\_id: {        type: DataTypes.INTEGER,        allowNull: false      },      qty: DataTypes.DOUBLE,      price: DataTypes.DOUBLE,    }, {      sequelize,      modelName: 'detail\_transaksi',      tableName: "detail\_transaksi",    });    return detail\_transaksi;  }; |

Hasil Praktikum adalah database seperti berikut



Catatan:

Jika hasil relasi tidak sesuai, maka silakan cek kembali kode dan sesuaikan seperti tahapan di atas.