Dokumentasi Backend Laundry OFFLINE-UKK

Oleh :
Gagas Surya Laksana
XIIRPL1/21

REKAYASA PERANGKAT LUNAK
SMK TELKOM MALANG
JANUARI 2022

Persiapan Frontend

Ini adalah dokumentasi khusus untuk BACKEND UKK-OFFLINE LAUNDRY-APP. Jadi segala macam library dan dependensi yang ada akan diminimalkan mengingat codingan ini dikerjakan tanpa internet. Disini saya memakai **express, jsonwebtoken, mysql2, nodemon, sequelize.**

DEPENDENCIES

```
"dependencies": {
   "cors": "^2.8.5",
   "crypto-js": "^4.1.1",
   "dotenv": "^10.0.0",
   "express": "^4.17.2",
   "jsonwebtoken": "^8.5.1",
   "mysql2": "^2.3.3",
   "nodemon": "^2.0.15",
   "sequelize": "^7.0.0-alpha.2"
 }
```

SHORTCUT INSTALL

I. Persiapan awal dan instalasi dependencies

1. Membuat database dengan nama **laundry-app.**
2. Membuat folder **backend** dan **api** (nantinya ada 2 folder, backend dan frontend).
3. Masuk kedalam folder backend dan lakukan inisiasi `npm init --y`
4. Membuat file dengan nama **index.js**
5. Lakukan instalasi dependencies yang diperlukan. Disini saya menginstal, antara lain:

```
npm install sequelize mysql2 express nodemon cors jsonwebtoken
crypto-js dot-env
```

6. Atur *nodemon,* Masuk ke **package.json** dan tambahkan **"start": "nodemon index.js"** pada bagian **scripts**.

II. Create Migrations

1. Inisiasi sequelize dengan `sequelize init`**./ npx sequelize-cli init**
2. konfigurasi database pada **config \ config.json** seperti dibawah.

```
  "development": {
    "username": "root",
    "password": null,
    "database": "laundry-app",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
```

3. membuat migration model tabelnya. Sebagai berikut:

tb_member

```
sequelize model:create --name tb_member --attributes
nama:string,alamat:text,'jenis_kelamin:enum:{L,P}',tlp:string
```

tb_outlet

```
sequelize model:create --name tb_outlet --attributes
nama:string,alamat:text,tlp:string
```

tb_user

```
sequelize model:create --name tb_user --attributes
nama:string,username:string,password:text,id_outlet:integer,'role:
enum:{admin,kasir,owner}'
```

tb_paket

```
sequelize model:create --name tb_paket --attributes
id_outlet:integer,'jenis:enum:{kiloan,selimut,bed_cover,kaos,lainn
ya}',nama_paket:string,harga:integer
```

tb_transaksi

```
sequelize model:create --name tb_transaksi --attributes
id_outlet:integer,kode_invoice:string,id_member:integer,tgl:date,b
atas_waktu:date,tgl_bayar:date,biaya_tambahan:integer,diskon:doubl
e,pajak:integer,'status:enum:{baru,proses,selesai,diambil}','dibay
ar:enum:{dibayar,belum_dibayar}',id_user:integer
```

tb_detail_transaksi

```
sequelize model:create --name tb_detail_transaksi --attributes
id_transaksi:integer,id_paket:integer,qty:double,keterangan:text
```

III. Relation Migrations

Untuk membuat tabtel berelasi.

1. Mengubah data **migrations** sesuai kode dibawah:

20211225023949-create-tb-member.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_member', {
      id: {
```

```
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nama: {
        type: Sequelize.STRING
      },
      alamat: {
        type: Sequelize.TEXT
      },
      jenis_kelamin: {
        type: Sequelize.ENUM('L', 'P')
      },
      tlp: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_member');
  }
};
```

20211225024118-create-tb-outlet.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_outlet', {
      id: {
```

```
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nama: {
        type: Sequelize.STRING
      },
      alamat: {
        type: Sequelize.TEXT
      },
      tlp: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_outlet');
  }
};
```

20211225024300-create-tb-user.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_user', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
```

```
        type: Sequelize.INTEGER
      },
      nama: {
        type: Sequelize.STRING
      },
      username: {
        type: Sequelize.STRING
      },
      password: {
        type: Sequelize.TEXT
      },
      id_outlet: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_outlet",
          key: "id"
        }
      },
      role: {
        type: Sequelize.ENUM('admin', 'kasir', 'owner')
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_user');
  }
};
```

20211225024517-create-tb-paket.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_paket', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      id_outlet: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_outlet",
          key: "id"
        }
      },
      jenis: {
        type: Sequelize.ENUM('kiloan', 'selimut',
'bed_cover', 'kaos', 'lainnya')
      },
      nama_paket: {
        type: Sequelize.STRING
      },
      harga: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
```

```
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_paket');
  }
};
```

20211225024926-create-tb-transaksi.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_transaksi', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      id_outlet: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_outlet",
          key: "id"
        }
      },
      kode_invoice: {
        type: Sequelize.STRING
      },
      id_member: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_member",
          key: "id"
        }
      },
      tgl: {
        type: Sequelize.DATE
      },
```

```
      batas_waktu: {
        type: Sequelize.DATE
      },
      tgl_bayar: {
        type: Sequelize.DATE
      },
      biaya_tambahan: {
        type: Sequelize.INTEGER
      },
      diskon: {
        type: Sequelize.DOUBLE
      },
      pajak: {
        type: Sequelize.INTEGER
      },
      status: {
        type: Sequelize.ENUM('baru', 'proses', 'selesai',
'diambil')
      },
      dibayar: {
        type: Sequelize.ENUM('dibayar', 'belum_dibayar')
      },
      id_user: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_user",
          key: "id"
        }
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

```
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_transaksi');
  }
};
```

20211225025642-create-tb-detail-transaksi.js

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('tb_detail_transaksi', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      id_transaksi: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references :{
          model: "tb_transaksi",
          key: "id"
        }
      },
      id_paket: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "tb_paket",
          key: "id"
        }
      },
      qty: {
        type: Sequelize.DOUBLE
      },
      keterangan: {
        type: Sequelize.TEXT
```

```
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('tb_detail_transaksi');
  }
};
```

2. Setelah selesai konfigurasi **relation.** Jalankan `sequelize db:migrate`

IV. Konfigurasi Models

Konfigurasi models ini bertujuan sebagai jembatan antara nodejs dengan database.

1. Mengubah data **models** seperti dibawah ini:

tb_detail_transaksi.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_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.tb_transaksi, {
        foreignKey: "id_transaksi",
        as: "tb_transaksi"
```

```
      })

      this.belongsTo(models.tb_paket, {
        foreignKey: "id_paket",
        as: "tb_paket"
      })
    }
  };
  tb_detail_transaksi.init({
    id_transaksi: DataTypes.INTEGER,
    id_paket: DataTypes.INTEGER,
    qty: DataTypes.DOUBLE,
    keterangan: DataTypes.TEXT
  }, {
    sequelize,
    modelName: 'tb_detail_transaksi',
    tableName: 'tb_detail_transaksi',
  });
  return tb_detail_transaksi;
};
```

tb_member.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_member 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.tb_transaksi, {
        foreignKey: "id_member",
        as: "tb_transaksi"
      })
    }
```

```
  };
  tb_member.init({
    nama: DataTypes.STRING,
    alamat: DataTypes.TEXT,
    jenis_kelamin: DataTypes.ENUM('L', 'P'),
    tlp: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'tb_member',
    tableName: 'tb_member',
  });
  return tb_member;
};
```

tb_outlet.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_outlet 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.tb_paket, {
        foreignKey: "id_outlet",
        as: "tb_paket"
      })

      this.hasMany(models.tb_transaksi, {
        foreignKey: "id_outlet",
        as: "tb_transaksi"
      })

      this.hasMany(models.tb_user, {
        foreignKey: "id_outlet",
```

```
        as: "tb_user"
      })
    }
  };
  tb_outlet.init({
    nama: DataTypes.STRING,
    alamat: DataTypes.TEXT,
    tlp: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'tb_outlet',
    tableName: 'tb_outlet',
  });
  return tb_outlet;
};
```

tb_paket.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_paket 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.tb_outlet, {
        foreignKey: "id_outlet",
        as: "tb_outlet"
      })

      this.hasMany(models.tb_detail_transaksi, {
        foreignKey: "id_paket",
        as: "tb_detail_transaksi"
      })
    }
```

```
  };
  tb_paket.init({
    id_outlet: DataTypes.INTEGER,
    jenis: DataTypes.ENUM('kiloan', 'selimut', 'bed_cover',
'kaos', 'lainnya'),
    nama_paket: DataTypes.STRING,
    harga: DataTypes.INTEGER
  }, {
    sequelize,
    modelName: 'tb_paket',
    tableName: 'tb_paket',
  });
  return tb_paket;
};
```

tb_transaksi.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_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.tb_outlet, {
        foreignKey: "id_outlet",
        as: "tb_outlet"
      })

      this.belongsTo(models.tb_member, {
        foreignKey: "id_member",
        as: "tb_member"
      })

      this.belongsTo(models.tb_user, {
```

```
        foreignKey: "id_user",
        as: "tb_user"
      })

      this.hasMany(models.tb_detail_transaksi, {
        foreignKey: "id_transaksi",
        as: "tb_detail_transaksi"
      })
    }
  };
  tb_transaksi.init({
    id_outlet: DataTypes.INTEGER,
    kode_invoice: DataTypes.STRING,
    id_member: DataTypes.INTEGER,
    tgl: DataTypes.DATE,
    batas_waktu: DataTypes.DATE,
    tgl_bayar: DataTypes.DATE,
    biaya_tambahan: DataTypes.INTEGER,
    diskon: DataTypes.DOUBLE,
    pajak: DataTypes.INTEGER,
    status: DataTypes.ENUM('baru', 'proses', 'selesai',
'diambil'),
    dibayar: DataTypes.ENUM('dibayar', 'belum_dibayar'),
    id_user: DataTypes.INTEGER
  }, {
    sequelize,
    modelName: 'tb_transaksi',
    tableName: 'tb_transaksi',
  });
  return tb_transaksi;
};
```

tb_user.js

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class tb_user 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.tb_outlet, {
        foreignKey: "id_outlet",
        as: "tb_outlet"
      })

      this.hasMany(models.tb_transaksi, {
        foreignKey: "id_user",
        as: "tb_transaksi"
      })
    }
  };
  tb_user.init({
    nama: DataTypes.STRING,
    username: DataTypes.STRING,
    password: DataTypes.TEXT,
    id_outlet: DataTypes.INTEGER,
    role: DataTypes.ENUM('admin', 'kasir', 'owner')
  }, {
    sequelize,
    modelName: 'tb_user',
    tableName: 'tb_user',
  });
  return tb_user;
};
```

V. Konfigurasi API NodeJS

Konfigurasi API ini bertujuan sebagai tempat keluar masuknya data dari database ke bagian frontend.

1. Masuk ke dalam folder **api** yang telah dibuat tadi, dan buat file dengan copy command di bawah untuk menyingkat waktu:

```
WINDOWS:
wsl touch member.js outlet.js paket.js transaksi.js
transaksi_detail.js user.js
```

```
 MAC/LINUX:
```

```
touch member.js outlet.js paket.js transaksi.js
transaksi_detail.js user.js
```

2. Selanjutnya buat file **config.js** didalam folder **config** dan isi dengan script berikut. Ini berfungsi untuk menampung variable **port** dan **secretKey.**

config.js

```
const dotenv = require('dotenv');
dotenv.config();
module.exports = {
  port: 8000,
  secretKey: "helo"
};
```

PENTING!

Sebelum masuk ke langkah selanjutnya, buat folder **middleware** didalam folder **api** dan isi dengan 2 file bernama **auth_verify.js** dan **auth.js**

```
WINDOWS:
wsl touch auth_verify.js auth.js
```

auth_verify.js

```
const jwt = require("jsonwebtoken")
const { secretKey } = require('../../config/config');

auth_verify = (req, res, next) => {
    // get jwt from header
    let header = req.headers.authorization
    let token = null

    if(header != null){
        // get token from second side
        token = header.split(" ")[1]
    }

    if(token == null){
        res.json({
            message: "unauthorized"
        })
    } else {
        // jwt
        let jwtHeader = {
```

```
            algorithm: "HS256"
        }

        jwt.verify(token, secretKey, jwtHeader, err => {
            if(err){
                res.json({
                    message: "Invalid or expired token",
                    Token: token
                })
            }else{
                next()
            }
        })
    }
}

module.exports = auth_verify
```

auth.js

```
const express = require('express')
const app = express()
const jwt = require('jsonwebtoken')
var CryptoJS = require("crypto-js");

// Ambil konfig
const { secretKey } = require('../../config/config');

// Password Encryption dengan menggunakan library crypto-js
// Encrypt
const encrypt = (nakedText) => {
    return hash = CryptoJS.HmacSHA256(nakedText,
secretKey).toString()
}

// call model
const user = require("../../models/index").tb_user

// allow request body
app.use(express.urlencoded({extended:true}))
app.use(express.json())
```

```
app.post('/', async (req,res) => {
    // put data
    let data = {
        username: req.body.username,
        password: encrypt(req.body.password),
        role: req.body.role
    }

    // put result
    let result = await user.findOne({where:data})

    if(result === null){
        res.json({
            message: "invalid username or password or level",
            isLogged: false
        })
    } else {
        // jwt
        let jwtHeader = {
            algorithm: "HS256",
            // expiresIn: exp.expToken // 1s 1h 1d 1w 1y
        }

        let payload = {
            data: result
        }

        let token = jwt.sign(payload, secretKey, jwtHeader)
        res.json({
            data: result,
            token: token,
            isLogged: true
        })
    }
})

module.exports = app
```

PENTING!

Langkah selanjutnya adalah main codingnya.

member.js

```
const express = require('express')
const app = express()

// Panggil Model dari sequelize db:migrate
const member = require("../models/index").tb_member

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)

// Bagian CRUD [Create, Read, Update, Delete]
// Get data
app.get('/', async(req, res) => {
    member.findAll()
    .then(result => {
        res.json({
            data_member: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database member
    let data = {
        nama: req.body.nama,
        alamat: req.body.alamat,
        jenis_kelamin: req.body.jenis_kelamin,
        tlp: req.body.tlp
```

```
    }

    member.create(data)
    .then(result => {
        res.json({
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        nama: req.body.nama,
        alamat: req.body.alamat,
        jenis_kelamin: req.body.jenis_kelamin,
        tlp: req.body.tlp
    }

    let id = {
        id: req.body.id
    }

    member.update(data, {where: id})
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
```

```
        })
    })
})

// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    member.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

outlet.js

```
const express = require('express')
const app = express()

// Panggil Model dari sequelize db:migrate
const outlet = require("../models/index").tb_outlet

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)
```

```
// Bagian CRUD [Create, Read, Update, Delete]
// Get data
app.get('/', async(req, res) => {
    outlet.findAll()
    .then(result => {
        res.json({
            data_outlet: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database outlet
    let data = {
        nama: req.body.nama,
        alamat: req.body.alamat,
        tlp: req.body.tlp
    }

    outlet.create(data)
    .then(result => {
        res.json({
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
```

```
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        nama: req.body.nama,
        alamat: req.body.alamat,
        tlp: req.body.tlp
    }

    let id = {
        id: req.body.id
    }

    outlet.update(data, {where: id})
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    outlet.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
```

```
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

paket.js

```
const express = require('express')
const app = express()

// Panggil Model dari sequelize db:migrate
const paket = require("../models/index").tb_paket

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)

// Bagian CRUD [Create, Read, Update, Delete]
// Get data
app.get('/', async(req, res) => {
    paket.findAll()
    .then(result => {
        res.json({
            data_paket: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
```

```
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database paket
    let data = {
        id_outlet: req.body.id_outlet,
        jenis: req.body.jenis,
        nama_paket: req.body.nama_paket,
        harga: req.body.harga
    }

    paket.create(data)
    .then(result => {
        res.json({
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        id_outlet: req.body.id_outlet,
        jenis: req.body.jenis,
        nama_paket: req.body.nama_paket,
        harga: req.body.harga
    }

    let id = {
        id: req.body.id
    }

    paket.update(data, {where: id})
```

```
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    paket.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

transaksi_detail.js

```
const express = require('express')
const app = express()
```

```
// Panggil Model dari sequelize db:migrate
const transaksi_detail =
require("../models/index").tb_detail_transaksi

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)

// Bagian CRUD [Create, Read, Update, Delete]
// Get data
app.get('/', async(req, res) => {
    transaksi_detail.findAll()
    .then(result => {
        res.json({
            data_transaksi: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database
transaksi_detail
    let data = {
        id_transaksi: req.body.id_transaksi,
        id_paket: req.body.id_paket,
        qty: req.body.qty,
        keterangan: req.body.keterangan || "tidak ada keterangan"
    }

    transaksi_detail.create(data)
```

```
    .then(result => {
        res.json({
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        id_transaksi: req.body.id_transaksi,
        id_paket: req.body.id_paket,
        qty: req.body.qty,
        keterangan: req.body.keterangan
    }

    let id = {
        id: req.body.id
    }

    transaksi_detail.update(data, {where: id})
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})
```

```
// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    transaksi_detail.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

transaksi.js

```
const express = require('express')
const app = express()

// Panggil Model dari sequelize db:migrate
const transaksi = require("../models/index").tb_transaksi

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)

// Bagian CRUD [Create, Read, Update, Delete]
// Get data
```

```
app.get('/', async(req, res) => {
    transaksi.findAll()
    .then(result => {
        res.json({
            data_transaksi: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database transaksi
    let data = {
        id_outlet: req.body.id_outlet,
        kode_invoice: req.body.kode_invoice,
        id_member: req.body.id_member,
        tgl: req.body.tgl,
        batas_waktu: req.body.batas_waktu,
        tgl_bayar: req.body.tgl_bayar,
        biaya_tambahan: req.body.biaya_tambahan,
        diskon: req.body.diskon,
        pajak: req.body.pajak,
        status: req.body.status,
        dibayar: req.body.dibayar,
        id_user: req.body.id_user
    }

    transaksi.create(data)
    .then(result => {
        res.json({
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
```

```
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        id_outlet: req.body.id_outlet,
        kode_invoice: req.body.kode_invoice,
        id_member: req.body.id_member,
        tgl: req.body.tgl,
        batas_waktu: req.body.batas_waktu,
        tgl_bayar: req.body.tgl_bayar,
        biaya_tambahan: req.body.biaya_tambahan,
        diskon: req.body.diskon,
        pajak: req.body.pajak,
        status: req.body.status,
        dibayar: req.body.dibayar,
        id_user: req.body.id_user
    }

    let id = {
        id: req.body.id
    }

    transaksi.update(data, {where: id})
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
```

```
})

// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    transaksi.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

user.js

```
const express = require('express')
const app = express()
var CryptoJS = require("crypto-js");

// Ambil konfig
const { secretKey } = require('../config/config');

// Password Encryption dengan menggunakan library crypto-js
// Encrypt
const encrypt = (nakedText) => {
    return hash = CryptoJS.HmacSHA256(nakedText,
secretKey).toString()
}

// Panggil Model dari sequelize db:migrate
```

```
const user = require("../models/index").tb_user

// Berikan akses 'request-body'
app.use(express.urlencoded({extended:true}))
app.use(express.json())

// Middleware, Autentikasi user
const verify = require("./middleware/auth_verify")
app.use(verify)

// Bagian CRUD [Create, Read, Update, Delete]
// Get data
app.get('/', async(req, res) => {
    user.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            data_user: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// Add data
app.post('/', async(req,res) => {
    // Deklarasi semua variable dalam table database user
    let data = {
        nama: req.body.nama,
        username: req.body.username,
        password: encrypt(req.body.password),
        id_outlet: req.body.id_outlet,
        role: req.body.role
    }

    user.create(data)
    .then(result => {
        res.json({
```

```
            message: "Data inserted",
            isSuccess: true,
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

// Update data
app.put('/', async(req,res) => {
    let data = {
        nama: req.body.nama,
        username: req.body.username,
        password: encrypt(req.body.password),
        id_outlet: req.body.id_outlet,
        role: req.body.role
    }

    let id = {
        id: req.body.id
    }

    user.update(data, {where: id})
    .then(result => {
        res.json({
            message: "Data updated",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})
```

```
// Delete data
app.delete('/:id', async(req,res) => {
    let parameter = {
        id: req.params.id
    }

    user.destroy({where: parameter})
    .then(result => {
        res.json({
            message: "Data deleted",
            isSuccess: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            isSuccess: false
        })
    })
})

module.exports = app
```

VI. Done!

Selamat kamu sudah selesai membuat backend dari nodejs dibantu dengan framework **expressjs** dan **sequelize**. Pada tahap ini hal yang perlu dilakukan adalah mencobanya dengan menggunakan **POSTMAN.**

1. Sebelum melakukan test api. Pastikan untuk merubah command pada file **package.json.** Lakukan seperti pada code dibawah:

```
{
  "name": "backend-laundry-app",
  "version": "1.0.0",
  "description": "backend laundry-app",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index.js"
  },
  "license": "MIT",
  "dependencies": {
    "cors": "^2.8.5",
```

```
    "crypto-js": "^4.1.1",
    "dotenv": "^10.0.0",
    "express": "^4.17.2",
    "jsonwebtoken": "^8.5.1",
    "mysql2": "^2.3.3",
    "nodemon": "^2.0.15",
    "sequelize": "^7.0.0-alpha.2"
  }
}
```

2. Buka terminal/command prompt. Lakukan **npm start.** Jika berhasil akan muncul seperti ini:

```
[nodemon] starting `node index.js`
server run in port: 8000
```