# Dokumentasi Backend Pembayaran SPP

Oleh :
**Gagas Surya Laksana**
**XIRPL1/15**

**REKAYASA PERANGKAT LUNAK**
**SMK TELKOM MALANG**
**MARET 2021**

# Persiapan Backend

Pembuatan project **pembayaran_spp** saya awali dengan membuat backend-nya terlebih dahulu. Tujuannya adalah meminimalisir terjadinya error karena backend adalah pondasi utama dalam membuat program. Kali ini saya memakai **expressjs** sebagai backend, **mysql** sebagai database dan **sequelize** sebagai ORM untuk nodejs. Disini saya memakai MAMPP sebagai servernya.

## I. Persiapan awal dan instalasi dependencies

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

```
npm install sequelize mysql2 express nodemon
```

7. Atur *nodemon,* Masuk ke **package.json** dan tambahkan **"start": "nodemon server.js"** pada bagian **scripts**.
8. konfigurasi database pada **config \ config.js** seperti gambar dibawah.

```
{
  "development": {
    "username": "root",
    "password": "root",
    "database": "pembayaran_spp",
    "port": "8889",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
```

## II. Create Migrations

1. Inisiasi sequelize dengan `sequelize init`**.**
2. membuat migration model tabelnya. Sebagai berikut:

**spp**

```
sequelize model:create --name spp --attributes
tahun:integer,nominal:integer
```

**kelas**

```
sequelize model:create --name kelas --attributes
nama_kelas:string,kompetensi_keahlian:string
```

**petugas**

```
sequelize model:create --name petugas --attributes
username:string,password:string,nama_petugas:string,level:enum
```

**siswa**

```
sequelize model:create --name siswa --attributes
nis:char,nama:string,id_kelas:integer,alamat:text,no_telp:string,i
d_spp:integer
```

**pembayaran**

```
sequelize model:create --name pembayaran --attributes
id_petugas:integer,nisn:integer,tgl_bayar:date,bulan_dibayar:strin
g,tahun_dibayar:string,id_spp:integer,jumlah_bayar:integer
```

## III. Relation Migrations

Untuk membuat tabel berelasi.

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

**Create-spp.js**

```
await queryInterface.createTable('spp', {
      id_spp: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      tahun: {
        type: Sequelize.INTEGER
      },
      nominal: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

**create-kelas.js**

```
await queryInterface.createTable('kelas', {
      id_kelas: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nama_kelas: {
        type: Sequelize.STRING
      },
      kompetensi_keahlian: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

**create-petugas.js**

```
await queryInterface.createTable('petugas', {
      id_petugas: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      username: {
        type: Sequelize.STRING
      },
      password: {
        type: Sequelize.STRING
      },
      nama_petugas: {
        type: Sequelize.STRING
      },
      level: {
```

```
        type: Sequelize.ENUM('admin','petugas')
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

**create-siswa.js**

```
await queryInterface.createTable('siswa', {
      nisn: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nis: {
        type: Sequelize.CHAR
      },
      nama: {
        type: Sequelize.STRING
      },
      id_kelas: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "kelas",
          key: "id_kelas"
        }
      },
      alamat: {
        type: Sequelize.TEXT
      },
      no_telp: {
        type: Sequelize.STRING
      },
      id_spp: {
        type: Sequelize.INTEGER,
```

```
        allowNull: false,
        references: {
          model: "spp",
          key: "id_spp"
        }
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

**create-pembayaran.js**

```
await queryInterface.createTable('pembayaran', {
      id_pembayaran: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      id_petugas: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "petugas",
          key: "id_petugas"
        }
      },
      nisn: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "siswa",
          key: "nisn"
        }
      },
      tgl_bayar: {
        type: Sequelize.DATE
```

```
      },
      bulan_dibayar: {
        type: Sequelize.STRING
      },
      tahun_dibayar: {
        type: Sequelize.STRING
      },
      id_spp: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "spp",
          key: "id_spp"
        }
      },
      jumlah_bayar: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
```

2. Setelah selesai konfigurasi **relation.** Jalankan `sequelize db:migrate`
3. Jika berhasil akan muncul seperti ini:

```
Sequelize CLI [Node: 14.15.3, CLI: 6.2.0, ORM: 6.5.0]

Loaded configuration file "config/config.json".
Using environment "development".
== 20210309021520-create-spp: migrating =======
== 20210309021520-create-spp: migrated (0.044s)

== 20210309021708-create-kelas: migrating =======
== 20210309021708-create-kelas: migrated (0.035s)
```

```
== 20210309031817-create-petugas: migrating =======
== 20210309031817-create-petugas: migrated (0.049s)

== 20210309050217-create-siswa: migrating =======
== 20210309050217-create-siswa: migrated (0.026s)

== 20210309050434-create-pembayaran: migrating =======
== 20210309050434-create-pembayaran: migrated (0.024s)
```

## IV. Konfigurasi Models

Konfigurasi models ini bertujuan sebagai jembatan antara nodejs dengan database.

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

**spp.js**

```
class spp extends Model {
    static associate(models) {
      // define association here
      this.hasMany(models.siswa, {
        foreignKey: "id_spp",
        as: "siswa"
      })

      this.hasMany(models.pembayaran, {
        foreignKey: "id_spp",
        as: "pembayaran"
      })
    }
  };
  spp.init({
    id_spp: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true
    },
    tahun: DataTypes.INTEGER,
    nominal: DataTypes.INTEGER
  }, {
    sequelize,
    modelName: 'spp',
    tableName: 'spp'
  });
```

**kelas.js**

```
class kelas extends Model {
    static associate(models) {
      // define association here
      this.hasMany(models.siswa, {
        foreignKey: "id_kelas",
        as: "siswa"
      })
    }
  };
  kelas.init({
    id_kelas: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true
    },
    nama_kelas: DataTypes.STRING,
    kompetensi_keahlian: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'kelas',
    tableName: 'kelas'
  });
```

**petugas.js**

```
class petugas extends Model {
    static associate(models) {
      // define association here
      this.hasMany(models.pembayaran, {
        foreignKey: "id_petugas",
        as: "pembayaran"
      })
    }
  };
  petugas.init({
    id_petugas: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true
    },
```

```
    username: DataTypes.STRING,
    password: DataTypes.STRING,
    nama_petugas: DataTypes.STRING,
    level: DataTypes.ENUM('admin','petugas')
  }, {
    sequelize,
    modelName: 'petugas',
    tableName: 'petugas'
  });
```

**siswa.js**

```
class siswa extends Model {
    static associate(models) {
      // define association here
      this.belongsTo(models.spp, {
        foreignKey: "id_spp",
        as: "spp"
      })

      this.belongsTo(models.kelas, {
        foreignKey: "id_kelas",
        as: "kelas"
      })

      this.hasMany(models.pembayaran, {
        foreignKey: "nisn",
        as: "pembayaran"
      })
    }
  };
  siswa.init({
    nisn: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true
    },
    nis: DataTypes.CHAR,
    nama: DataTypes.STRING,
    id_kelas: DataTypes.INTEGER,
    alamat: DataTypes.TEXT,
    no_telp: DataTypes.STRING,
    id_spp: DataTypes.INTEGER
```

```
  }, {
    sequelize,
    modelName: 'siswa',
    tableName: 'siswa'
  });
```

**pembayaran.js**

```
class pembayaran extends Model {
    static associate(models) {
      // define association here
      this.belongsTo(models.petugas, {
        foreignKey: "id_petugas",
        as: "petugas"
      })

      this.belongsTo(models.siswa, {
        foreignKey: "nisn",
        as: "siswa"
      })

      this.belongsTo(models.spp, {
        foreignKey: "id_spp",
        as: "spp"
      })
    }
  };
  pembayaran.init({
    id_pembayaran: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true
    },
    id_petugas: DataTypes.INTEGER,
    nisn: DataTypes.STRING,
    tgl_bayar: DataTypes.DATE,
    bulan_dibayar: DataTypes.STRING,
    tahun_dibayar: DataTypes.STRING,
    id_spp: DataTypes.INTEGER,
    jumlah_bayar: DataTypes.INTEGER
  }, {
    sequelize,
```

```
    modelName: 'pembayaran',
    tableName: 'pembayaran'
  });
```

## V. Konfigurasi API NodeJS

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

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

```
MAC/LINUX:
touch kelas.js siswa.js petugas.js pembayaran.js spp.js auth.js
auth_verify.js
```

Untuk windows saya kurang mengerti command yang dipakai

### spp.js

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

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

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

// auth_verify
const verify = require("./auth_verify")
app.use(verify)

// get data
app.get("/", async(req,res) => {
    spp.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            message: "Data founded",
            spp: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
```

```
            found: true
        })
    })
})

// add data
app.post("/", async(req,res) => {
    // put data
    let data = {
        tahun: req.body.tahun,
        nominal: req.body.nominal
    }

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

// update data
app.put("/", async(req,res) => {
    // put data
    let data = {
        tahun: req.body.tahun,
        nominal: req.body.nominal
    }

    let param = {
        id_spp: req.body.id_spp
    }

    spp.update(data, {where: param})
    .then(result => {
        res.json({
```

```
            message: "Data updated",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

// delete data
app.delete("/:id_spp", async(req,res) => {
    // put data
    let param = {
        id_spp: req.params.id_spp
    }

    spp.destroy({where: param})
    .then(result => {
        res.json({
            message: "Data deleted",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

module.exports = app;
```

**kelas.js**

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

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

// allow request body
```

```
app.use(express.urlencoded({extended:true}))

// auth_verify
const verify = require("./auth_verify")
app.use(verify)

// get data
app.get("/", async(req,res) => {
    kelas.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            message: "Data founded",
            kelas: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: true
        })
    })
})

// add data
app.post("/", async(req,res) => {
    // put data
    let data = {
        nama_kelas: req.body.nama_kelas,
        kompetensi_keahlian: req.body.kompetensi_keahlian
    }

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

```
        })
    })
})

// update data
app.put("/", async(req,res) => {
    // put data
    let data = {
        nama_kelas: req.body.nama_kelas,
        kompetensi_keahlian: req.body.kompetensi_keahlian
    }

    let param = {
        id_kelas: req.body.id_kelas
    }

    kelas.update(data, {where: param})
    .then(result => {
        res.json({
            message: "Data updated",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

// delete data
app.delete("/:id_kelas", async(req,res) => {
    // put data
    let param = {
        id_kelas: req.params.id_kelas
    }

    kelas.destroy({where: param})
    .then(result => {
        res.json({
            message: "Data deleted",
            data: result
```

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

module.exports = app;
```

**petugas.js**

```
const express = require("express")
const app = express()
var md5 = require('md5');

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

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

// auth_verify
const verify = require("./auth_verify")
app.use(verify)

// get data
app.get("/", async(req,res) => {
    petugas.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            petugas: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})
```

```
// add data
app.post("/", async(req,res) => {
    // put data
    let data = {
        username: req.body.username,
        password: md5(req.body.password),
        nama_petugas: req.body.nama_petugas,
        level: req.body.level
    }

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

// update data
app.put("/", async(req,res) => {
    // put data
    let data = {
        username: req.body.username,
        password: md5(req.body.password),
        nama_petugas: req.body.nama_petugas,
        level: req.body.level
    }

    let param = {
        id_petugas: req.body.id_petugas
    }

    petugas.update(data, {where: param})
    .then(result => {
        res.json({
```

```
            message: "Data updated",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

// delete data
app.delete("/:id_petugas", async(req,res) => {
    // put data
    let param = {
        id_petugas: req.params.id_petugas
    }

    petugas.destroy({where: param})
    .then(result => {
        res.json({
            message: "Data deleted",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

module.exports = app;
```

**siswa.js**

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

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

// allow request body
```

```
app.use(express.urlencoded({extended:true}))

// get data by NISN
app.get("/:nisn", async(req,res) => {
    let nisn = {
        nisn: req.params.nisn
    }

    siswa.findOne({where: nisn, include:[{ all: true, nested: true
}]})
    .then(result => {
        if(result){
            res.json({
                message: "Data founded",
                data_siswa: result,
                found: true
            })
        } else {
            res.json({
                message: "Data not found",
                found: false
            })
        }
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

// auth_verify
const verify = require("./auth_verify")
app.use(verify)

// get data
app.get("/", async(req,res) => {
    siswa.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            message: "Data founded",
            siswa: result,
```

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

// add data
app.post("/", async(req,res) => {
    // put data
    let data = {
        nisn: req.body.nisn,
        nis: req.body.nis,
        nama: req.body.nama,
        id_kelas: req.body.id_kelas,
        alamat: req.body.alamat,
        no_telp: req.body.no_telp,
        id_spp: req.body.id_spp
    }

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

// update data
app.put("/", async(req,res) => {
    // put data
    let data = {
```

```
        nis: req.body.nis,
        nama: req.body.nama,
        id_kelas: req.body.id_kelas,
        alamat: req.body.alamat,
        no_telp: req.body.no_telp,
        id_spp: req.body.id_spp
    }

    let param = {
        nisn: req.body.nisn
    }

    siswa.update(data, {where: param})
    .then(result => {
        res.json({
            message: "Data updated",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

// delete data
app.delete("/:nisn", async(req,res) => {
    // put data
    let param = {
        nisn: req.params.nisn
    }

    siswa.destroy({where: param})
    .then(result => {
        res.json({
            message: "Data deleted",
            data: result
        })
    })
    .catch(error => {
        res.json({
```

```
            message: error.message
        })
    })
})

module.exports = app;
```

**pembayaran.js**

```
const express = require("express")
const app = express()
var md5 = require('md5');

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

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

// auth_verify
const verify = require("./auth_verify")
app.use(verify)

// get data
app.get("/", async(req,res) => {
    pembayaran.findAll({include:[{ all: true, nested: true }]})
    .then(result => {
        res.json({
            pembayaran: result,
            found: true
        })
    })
    .catch(error => {
        res.json({
            message: error.message,
            found: false
        })
    })
})

// add data
app.post("/", async(req,res) => {
    // put data
```

```
    let data = {
        id_petugas: req.body.id_petugas,
        nisn: req.body.nisn,
        tgl_bayar: req.body.tgl_bayar,
        bulan_dibayar: req.body.bulan_dibayar,
        tahun_dibayar: req.body.tahun_dibayar,
        id_spp: req.body.id_spp,
        jumlah_bayar: req.body.jumlah_bayar
    }

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

// update data
app.put("/", async(req,res) => {
    // put data
    let data = {
        id_petugas: req.body.id_petugas,
        nisn: req.body.nisn,
        tgl_bayar: req.body.tgl_bayar,
        bulan_dibayar: req.body.bulan_dibayar,
        tahun_dibayar: req.body.tahun_dibayar,
        id_spp: req.body.id_spp,
        jumlah_bayar: req.body.jumlah_bayar
    }

    let param = {
        id_pembayaran: req.body.id_pembayaran
    }

    pembayaran.update(data, {where: param})
```

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

// delete data
app.delete("/:id_pembayaran", async(req,res) => {
    // put data
    let param = {
        id_pembayaran: req.params.id_pembayaran
    }

    pembayaran.destroy({where: param})
    .then(result => {
        res.json({
            message: "Data deleted",
            data: result
        })
    })
    .catch(error => {
        res.json({
            message: error.message
        })
    })
})

module.exports = app;
```

**auth.js**

```
const express = require('express')
const app = express()
const jwt = require('jsonwebtoken')
const md5 = require('md5')
```

```
// call model
const petugas = require("../models/index").petugas

// 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: md5(req.body.password),
        level: req.body.level
    }

    // let exp = {
    //     expToken: req.body.expToken
    // }

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

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

        let payload = {
            data: result
        }

        let secretKey = "koala"

        let token = jwt.sign(payload, secretKey, jwtHeader)
        res.json({
```

```
            data: result,
            token: token,
            logged: true
        })
    }
})

module.exports = app
```

**auth_verify.js**

```
const jwt = require("jsonwebtoken")

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"
        }

        let secretKey = "koala"

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

```
            }
        })
    }
}

module.exports = auth_verify
```

2. Setelah selesai membuat Router API-nya. Sekarang saatnya membuat gerbangnya untuk dapat dijalankan sesuai dengan kebutuhan user dari frontend. Buka file **server.js** dan isikan script dibawah ini:

**server.js**

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

/*
Access to XMLHttpRequest at 'http://localhost:8000/auth' from
origin
'http://localhost:3000' has been blocked by CORS policy: Response
to
preflight request doesn't pass access control check: No
'Access-Control-Allow-Origin' header is present on the requested
resource.
*/
var cors = require('cors')
app.use(cors())

app.use(express.static(__dirname))

// router
const kelas = require("./router/kelas")
const spp = require("./router/spp")
const siswa = require("./router/siswa")
const petugas = require("./router/petugas")
const pembayaran = require("./router/pembayaran")
const auth = require("./router/auth")

app.use("/auth", auth)
app.use("/kelas", kelas)
app.use("/spp", spp)
app.use("/siswa", siswa)
```

```
app.use("/petugas", petugas)
app.use("/pembayaran", pembayaran)

app.listen(8000, () => {
    console.log("Server run on 8000")
})
```

## 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",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "md5": "^2.3.0",
    "mysql2": "^2.2.5",
    "nodemon": "^2.0.7",
    "sequelize": "^6.5.0"
  }
}
```

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

```
[nodemon] starting `node server.js`
Server run on 8000
```

3. Cek pada document **“dokumentasi-api-pembayaran-spp”** untuk lebih lengkapnya.

```
https://docs.google.com/document/d/1b_Gl8H6yqPAl9bZt_2Ot52I0f36kelPNSEVf4ds3O4Y/edit?usp=sharing
```