**DOKUMENTASI BACKEND UKK PEMBAYARAN SPP**

1. Buat folder project PEMBAYARAN SPP dan buat dua folder baru didalamnya PEMBAYARAN SPP/BACKEND + FRONTEND
2. Path CMD pada PEMBAYARAN/BACKEND
3. Jalankan **npm init –y** dan keluar **package.json**
4. Jalankan **npm install -g sequelize-cli**, jika gagal gunakan **npm** **install –save sequelize**.
5. Install packages berikut : **cors, express, jsonwebtoken, md5, multer, mysql2, nodemon, sequelize**.
6. Pada folder cofig/config.json pada bagian database isikan dengan database yang sudah dibuat

"development": {

    "username" : "root",

    "password" : null,

    "database" : "ukk\_spp",

    "host" : "127.0.0.1",

    "dialect" : "mysql"

  },

1. Buat migration dari setiap table. Dengan cara menjalankan **Sequelize model:create --name [nama tabel] --attributes [nama kolom:tipe, nama kolom:tipe]** Khusus untuk kolom yang menjadi primary key tidak perlu dituliskan pada attributes karena nanti akan ada primary key saat file migration sudah terbentuk.

* **Tabel Petugas**

sequelize model:create --name petugas --attributes username:string,password:string,nama\_petugas:string,level:string,image:string

* **Tabel Kelas**

sequelize model:create --name kelas --attributes nama\_kelas:string,kompetensi\_keahlian:string

* **Tabel Tahun**

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

* **Tabel Siswa**

sequelize model:create --name siswa --attributes nisn:string,nis:string,nama:string,id\_kelas:integer,alamat:text,no\_telp:string,id\_spp:integer,username:string,password:string,image:string

* **Tabel Pembayaran**

sequelize model:create --name pembayaran --attributes id\_petugas:integer,nisn:string,tgl\_bayar:date,bulan\_tahun:string,id\_spp:integer,jumlah\_bayar:integer

Graphical user interface, text, application

Description automatically generated

1. Rubah isi file pada folder **Models**

* **Kelas.js**

'use strict';

const {

  Model

} = require('sequelize');

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

  class kelas 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) {

      this.hasMany(models.siswa, {

        foreignKey: "id\_kelas",

        as: "kelas"

      })

    }

  };

  kelas.init({

    id\_kelas: {

      type: DataTypes.INTEGER,

      primaryKey: true

    },

    nama\_kelas: DataTypes.STRING,

    kompetensi\_keahlian: DataTypes.STRING

  }, {

    sequelize,

    modelName: 'kelas',

    tableName: "kelas"

  });

  return kelas;

};

* **Pembayaran.js**

'use strict';

const {

  Model

} = require('sequelize');

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

  class pembayaran extends Model {

    static associate(models) {

      // relation to petugas

      this.belongsTo(models.petugas, {

        foreignKey: "id\_petugas",

        as: "petugas"

      })

      // relation to siswa

      this.belongsTo(models.siswa, {

        foreignKey: "nisn",

        as: "siswa"

      })

      // relation to siswa

      this.belongsTo(models.siswa, {

        foreignKey: "id\_spp",

        as: "spp"

      })

    }

  };

  pembayaran.init({

    id\_petugas: DataTypes.INTEGER,

    nisn: DataTypes.STRING,

    tgl\_bayar: DataTypes.DATE,

    bulan\_tahun: DataTypes.STRING,

    id\_spp: DataTypes.INTEGER,

    jumlah\_bayar: DataTypes.INTEGER

  }, {

    sequelize,

    modelName: 'pembayaran',

    tableName: "pembayaran"

  });

  return pembayaran;

};

* **Petugas.js**

'use strict';

const {

  Model

} = require('sequelize');

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

  class pembayaran extends Model {

    static associate(models) {

      // relation to petugas

      this.belongsTo(models.petugas, {

        foreignKey: "id\_petugas",

        as: "petugas"

      })

      // relation to siswa

      this.belongsTo(models.siswa, {

        foreignKey: "nisn",

        as: "siswa"

      })

      // relation to siswa

      this.belongsTo(models.siswa, {

        foreignKey: "id\_spp",

        as: "spp"

      })

    }

  };

  pembayaran.init({

    id\_pembayaran: {

      type: DataTypes.INTEGER,

      primaryKey: true

    },

    id\_petugas: DataTypes.INTEGER,

    nisn: DataTypes.STRING,

    tgl\_bayar: DataTypes.DATE,

    bulan\_tahun: DataTypes.STRING,

    id\_spp: DataTypes.INTEGER,

    jumlah\_bayar: DataTypes.INTEGER

  }, {

    sequelize,

    modelName: 'pembayaran',

    tableName: "pembayaran"

  });

  return pembayaran;

};

* **Siswa.js**

'use strict';

const {

  Model

} = require('sequelize');

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

  class siswa 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) {

      this.hasMany(models.pembayaran, {

        foreignKey: "nisn",

        as: "siswa"

      })

      // relation to pembayaran (id\_spp)

      this.hasMany(models.pembayaran, {

        foreignKey: "id\_spp",

        as: "spp\_pembayaran"

      })

      // relation to spp

      this.belongsTo(models.spp, {

        foreignKey: "id\_spp",

        as: "spp"

      })

      // relation to kelas

      this.belongsTo(models.kelas, {

        foreignKey: "id\_kelas",

        as: "kelas"

      })

    }

  };

  siswa.init({

    nisn: {

      type: DataTypes.STRING,

      primaryKey: true

    },

    nis: DataTypes.STRING,

    nama: DataTypes.STRING,

    id\_kelas: DataTypes.INTEGER,

    alamat: DataTypes.TEXT,

    no\_telp: DataTypes.STRING,

    id\_spp: DataTypes.INTEGER,

    username: DataTypes.STRING,

    password: DataTypes.STRING,

    image: DataTypes.STRING

  }, {

    sequelize,

    modelName: 'siswa',

    tableName: "siswa"

  });

  return siswa;

};

* **SPP.js**

'use strict';

const {

  Model

} = require('sequelize');

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

  class spp 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) {

      this.hasMany(models.siswa, {

        foreignKey: "id\_spp",

        as: "spp"

      })

    }

  };

  spp.i

  spp.init({

    id\_spp: {

      type: DataTypes.INTEGER,

      primaryKey: true

    },

    tahun: DataTypes.INTEGER,

    nominal: DataTypes.INTEGER

  }, {

    sequelize,

    modelName: 'spp',

    tableName: "spp"

  });

  return spp;

};

1. Folder **Migration**

* **Create-Petugas.js**

'use strict';

module.exports = {

  up: async (queryInterface, Sequelize) => {

    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.STRING

      },

      image: {

        type: Sequelize.STRING

      },

      createdAt: {

        allowNull: false,

        type: Sequelize.DATE

      },

      updatedAt: {

        allowNull: false,

        type: Sequelize.DATE

      }

    });

  },

  down: async (queryInterface, Sequelize) => {

    await queryInterface.dropTable('petugas');

  }

};

* **Create-Kelas.js**

'use strict';

module.exports = {

  up: async (queryInterface, Sequelize) => {

    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

      }

    });

  },

  down: async (queryInterface, Sequelize) => {

    await queryInterface.dropTable('kelas');

  }

};

* **Create-SPP.js**

'use strict';

module.exports = {

  up: async (queryInterface, Sequelize) => {

    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

      }

    });

  },

  down: async (queryInterface, Sequelize) => {

    await queryInterface.dropTable('spp');

  }

};

* **Create-Siswa.js**

'use strict';

module.exports = {

  up: async (queryInterface, Sequelize) => {

    await queryInterface.createTable('siswa', {

      nisn: {

        allowNull: false,

        primaryKey: true,

        type: Sequelize.INTEGER

      },

      nis: {

        type: Sequelize.STRING

      },

      nama: {

        type: Sequelize.STRING

      },

      id\_kelas: {

        type: Sequelize.INTEGER,

        references: {

          model: "kelas",

          key: "id\_kelas"

        }

      },

      alamat: {

        type: Sequelize.TEXT

      },

      no\_telp: {

        type: Sequelize.STRING

      },

      id\_spp: {

        type: Sequelize.INTEGER,

        references: {

          model: "spp",

          key: "id\_spp"

        }

      },

      username: {

        type: Sequelize.STRING

      },

      password: {

        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('siswa');

  }

};

* **Create-Pembayaran.js**

'use strict';

module.exports = {

  up: async (queryInterface, Sequelize) => {

    await queryInterface.createTable('pembayaran', {

      id\_pembayaran: {

        allowNull: false,

        autoIncrement: true,

        primaryKey: true,

        type: Sequelize.INTEGER

      },

      id\_petugas: {

        type: Sequelize.INTEGER,

        references: {

          model: "petugas",

          key: "id\_petugas"

        }

      },

      nisn: {

        type: Sequelize.STRING,

        references: {

          model: "siswa",

          key: "nisn"

        }

      },

      tgl\_bayar: {

        type: Sequelize.DATE

      },

      bulan\_tahun: {

        type: Sequelize.STRING

      },

      id\_spp: {

        type: Sequelize.INTEGER,

        references: {

          model: "siswa",

          key: "id\_spp"

        }

      },

      jumlah\_bayar: {

        type: Sequelize.INTEGER

      },

      createdAt: {

        allowNull: false,

        type: Sequelize.DATE

      },

      updatedAt: {

        allowNull: false,

        type: Sequelize.DATE

      }

    });

  },

  down: async (queryInterface, Sequelize) => {

    await queryInterface.dropTable('pembayaran');

  }

};

1. **sequelize db:migrate** pada terminal, lalu cek phpMyAdmin

**Graphical user interface

Description automatically generated**

1. Endpoint

* **Membuat server.js**

    const express = require("express")

    const app = express()

    const cors = require("cors");

    app.use(cors());

    const kelas = require("./router/endpoint-kelas")

    const petugas = require("./router/endpoint-petugas")

    const spp = require("./router/endpoint-spp")

    const siswa = require("./router/endpoint-siswa")

    const pembayaran = require("./router/endpoint-pembayaran")

    app.use("/pembayaran/spp/kelas", kelas)

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

    app.use("/pembayaran/spp/spp", spp)

    app.use("/pembayaran/spp/siswa", siswa)

    app.use("/pembayaran/spp/transaksi", pembayaran)

    app.use(express.static(\_\_dirname))

    app.listen(8000, ()=> {

        console.log("Server run on port 8000")

    })

* Membuat folder router dan isinya
* **Endpoint-kelas.js**

const express = require("express")

const models = require("../models/index")

const kelas = models.kelas

const app = express()

const auth = require("../auth")

app.use(auth)

app.use(express.json())

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

app.get("/", async(req,res)=>{

    let result = await kelas.findAll()

    res.json(result)

})

//CREATE

app.post("/", async(req,res)=>{

    let data = {

        nama\_kelas: req.body.nama\_kelas,

        kompetensi\_keahlian: req.body.kompetensi\_keahlian

    }

    kelas.create(data)

    .then(result => {

        res.json({message: "data inserted"})

    })

    .catch(error => {

        res.json({message:error.message})

    })

})

//UPDATE

app.put("/", async(req,res)=>{

    let param = await {id\_kelas:req.body.id\_kelas}

    let data = await {

        nama\_kelas: req.body.nama\_kelas,

        kompetensi\_keahlian: req.body.kompetensi\_keahlain

    }

    kelas.update(data, {where:param})

    .then(result => {

        res.json({message: "data updated"})

    })

    .catch(error =>{

        res.json({message: error.message})

    })

})

//DELETE

app.delete("/:id\_kelas", async(req,res)=>{

    let param = {id\_kelas:req.params.id\_kelas}

    kelas.destroy({where:param})

    .then(result=>{

        res.json({message: "data destroyed"})

    })

    .catch(error=>{

        res.json({message: error.message})

    })

})

module.exports = app

* **Endpoint-pembayaran.js**

const express = require("express")

const models = require("../models/index")

const pembayaran = models.pembayaran

const app = express()

const auth = require("../auth")

app.use(auth)

app.use(express.json())

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

app.get("/", async(req,res)=>{

    let result = await pembayaran.findAll()

    res.json(result)

})

app.get("/", async(req,res)=>{

    let result = await pembayaran.findAll()

    res.json(result)

})

app.post("/", async(req,res)=>{

    let data = {

        id\_petugas: req.body.id\_petugas,

        nisn: req.body.nisn,

        tgl\_bayar: req.body.tgl\_bayar,

        bulan\_tahun: req.body.bulan\_tahun,

        jumlah\_bayar: req.body.jumlah\_bayar

    }

    pembayaran.create(data)

    .then(result => {

        res.json({message: "data inserted"})

    })

    .catch(error => {

        res.json({message:error.message})

    })

})

app.put("/", async(req,res)=>{

    let param = await {id\_pembayaran:req.body.id\_pembayaran}

    let data = await {

        id\_petugas: req.body.id\_petugas,

        nisn: req.body.nisn,

        tgl\_bayar: req.body.tgl\_bayar,

        bulan\_tahun: req.body.bulan\_tahun,

        id\_spp: req.body.id\_spp,

        jumlah\_bayar: req.body.jumlah\_bayar

    }

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

    .then(result => {

        res.json({message: "data updated"})

    })

    .catch(error =>{

        res.json({message: error.message})

    })

})

app.delete("/:id\_pembayaran", async(req,res)=>{

    let param = {id\_pembayaran:req.params.id\_pembayaran}

    pembayaran.destroy({where:param})

    .then(result=>{

        res.json({message: "data destroyed"})

    })

    .catch(error=>{

        res.json({message: error.message})

    })

})

module.exports = app

* Endpoint-petugas.js

const express = require("express")

const models = require("../models/index")

const petugas = models.petugas

const app = express()

const md5 = require("md5")

app.use(express.json())

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

const multer = require("multer")

const path = require("path")

const fs = require("fs")

const jwt = require("jsonwebtoken")

const SECRET\_KEY\_ADMIN = "testing"

const authAdmin = require("../auth-admin")

const SECRET\_KEY\_PENGURUS = "cobain"

const authPetugas = require("../auth-petugas")

//storage

const storage = multer.diskStorage({

    destination: (req, file, cal) => {

        cal(null, "./image/admin\_img")

    },

    filename: (req, file, cal) => {

        cal(null, "img-" + Date.now() + path.extname(file.originalname))

    }

})

let upload = multer({ storage: storage })

//endpoint petugas

app.get("/for-petugas/:id\_petugas", authPetugas, async (req, res) => {

    let param = {

        id\_petugas: req.params.id\_petugas

    }

    let result = await petugas.findAll({ where: param })

    res.json(result)

    console.log(auth)

})

//enpoint admin

// end point for admin

app.get("/", authAdmin, async (req, res) => {

    let result = await petugas.findAll()

    res.json(result)

    console.log(auth)

})

app.post("/", upload.single("image"), async (req, res) => {

    if (!req.file) {

        res.json({

            message: "No file uploaded"

        })

    } else {

        let data = await {

            username: req.body.username,

            password: md5(req.body.password),

            nama\_petugas: req.body.nama\_petugas,

            level: req.body.level,

            image: req.file.filename

        }

        petugas.create(data)

            .then(result => {

                res.json({ message: "data inserted" })

            })

            .catch(error => {

                res.json({ message: error.message })

            })

    }

})

app.put("/", authAdmin, upload.single("image"), async (req, res) => {

    let param = await { id\_petugas: req.body.id\_petugas }

    let data = await {

        username: req.body.username,

        password: md5(req.body.password),

        nama\_petugas: req.body.nama\_petugas,

        level: req.body.level

    }

    if (req.file) {

        petugas.findOne({ where: param })

            .then(result => {

                let oldImageName = result.image

                let dir = path.join(\_\_dirname, "../image/admin\_img", oldImageName)

                fs.unlink(dir, err => console.log(err))

            })

            .catch(error => {

                console.log(error)

            })

        data.image = req.file.filename

    }

    petugas.update(data, { where: param })

        .then(result => {

            res.json({ message: "data updated" })

        })

        .catch(error => {

            res.json({ message: error.message })

        })

})

app.delete("/:id\_petugas", authAdmin, async (req, res) => {

    let param = await { id\_petugas: req.params.id\_petugas }

    let result = await petugas.findOne({ where: param })

    let oldImageName = result.image

    // delete old file

    let dir = path.join(\_\_dirname, "../image/admin\_img", oldImageName)

    fs.unlink(dir, err => console.log(err))

    petugas.destroy({ where: param })

        .then(result => {

            res.json({ message: "data destroyed" })

        })

        .catch(error => {

            res.json({ message: error.message })

        })

})

app.post("/login", async (req, res) => {

    let param = {

        username: req.body.username,

        password: md5(req.body.password)

    }

    let result = await petugas.findOne({ where: param })

    if (result) {

        let payload = JSON.stringify(result)

        let role = (result.level).toLowerCase()

        if (role === "admin") {

            let token = jwt.sign(payload, SECRET\_KEY\_ADMIN)

            res.json({

                logged: true,

                data: result,

                role: role,

                token: token

            })

        } else {

            let token = jwt.sign(payload, SECRET\_KEY\_PENGURUS)

            res.json({

                logged: true,

                data: result,

                role: role,

                token: token

            })

        }

    } else {

        res.json({

            logged: false,

            message: "invalid username or password"

        })

    }

})

module.exports = app

* Endpoint-siswa.js

const express = require("express")

const models = require("../models/index")

const siswa = models.siswa

const app = express()

const md5 = require("md5")

const SECRET\_KEY\_SISWA = "testing"

const authSiswa = require("../auth-siswa")

const jwt = require("jsonwebtoken")

const multer = require("multer")

const path = require("path")

const fs = require("fs")

app.use(express.json())

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

const storage = multer.diskStorage({

    destination: (req, file, cal) => {

        cal(null, "./image/siswa\_img")

    },

    filename: (req, file, cal) => {

        cal(null, "img-" + Date.now() + path.extname(file.originalname))

    }

})

let upload = multer({ storage: storage })

// end point for siswa

app.get("/for-siswa/:nisn", authSiswa, async (req, res) => {

    let param = {

        nisn: req.params.nisn

    }

    let result = await siswa.findAll(

        {

            where: param,

            include: ["kelas", "spp"]

        })

    res.json(result)

})

// end point for petugas

app.get("/for-petugas", authPetugas, async (req, res) => {

    let result = await siswa.findAll(

        {

            include: ["kelas", "spp"]

        })

    res.json(result)

})

app.get("/for-petugas/nisn/:nisn", authPetugas, async (req, res) => {

    let param = {

        nisn: req.params.nisn

    }

    let result = await siswa.findOne(

        {

            where: param,

            include: ["kelas", "spp"]

        })

    res.json(result)

})

app.get("/for-petugas/nama/:nama", authPetugas, async (req, res) => {

    let param = {

        nama: req.params.nama

    }

    let result = await siswa.findOne(

        {

            where: param,

            include: ["kelas", "spp"]

        })

    res.json(result)

})

// end point for admin

app.get("/for-admin", authAdmin, async (req, res) => {

    let result = await siswa.findAll(

        {

            include: ["kelas", "spp"]

        })

    res.json(result)

})

app.get("/for-admin/nisn/:nisn", authAdmin, async (req, res) => {

    let param = {

        nisn: req.params.nisn

    }

    let result = await siswa.findOne(

        {

            where: param,

            include: ["kelas", "spp"]

        })

    res.json(result)

})

app.get("/for-admin/nama/:nama", authAdmin, async (req, res) => {

    let param = {

        nama: req.params.nama

    }

    let result = await siswa.findOne(

        {

            where: param,

            include: ["kelas", "spp"]

        })

    res.json(result)

})

app.get("/", authAdmin, async (req, res) => {

    let result = await siswa.findAll()

    res.json(result)

})

app.post("/", authAdmin, upload.single("image"), async (req, res) => {

    if (!req.file) {

        res.json({

            message: "No file uploaded"

        })

    } else {

        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,

            username: req.body.username,

            password: md5(req.body.password),

            image: req.file.filename

        }

        siswa.create(data)

            .then(result => {

                res.json({ message: "data inserted" })

            })

            .catch(error => {

                res.json({ message: error.message })

            })

    }

})

app.put("/", authAdmin, upload.single("image"), async (req, res) => {

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

    let data = await {

        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,

        username: req.body.username,

        password: md5(req.body.password),

        image: req.file.filename

    }

    if (req.file) {

        const row = siswa.findOne({ where: param })

            .then(result => {

                let oldImageName = result.image

                let dir = path.join(\_\_dirname, "../image/siswa\_img", oldImageName)

                fs.unlink(dir, err => console.log(err))

            })

            .catch(error => {

                console.log(error.message)

            })

        data.image = req.file.filename

    }

    siswa.update(data, { where: param })

        .then(result => {

            res.json({ message: "data updated" })

        })

        .catch(error => {

            res.json({ message: error.message })

        })

})

app.delete("/:nisn", authAdmin, async (req, res) => {

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

    let result = await siswa.findOne({ where: param })

    let oldImageName = result.image

    let dir = path.join(\_\_dirname, "../image/siswa\_img", oldImageName)

    fs.unlink(dir, err => console.log(err))

    siswa.destroy({ where: param })

        .then(result => {

            res.json({ message: "data destroyed" })

        })

        .catch(error => {

            res.json({ message: error.message })

        })

})

app.post("/login", async (req, res) => {

    let param = {

        username: req.body.username,

        password: md5(req.body.password)

    }

    let result = await siswa.findOne({ where: param })

    if (result) {

        let payload = JSON.stringify(result)

        let token = jwt.sign(payload, SECRET\_KEY\_SISWA)

        res.json({

            logged: true,

            data: result,

            role: "siswa",

            token: token

        })

    } else {

        res.json({

            logged: false,

            message: "invalid username or password"

        })

    }

})

module.exports = app

* Endpoint-spp.js

const express = require("express")

const models = require("../models/index")

const spp = models.spp

const app = express()

const auth = require("../auth-admin")

app.use(auth)

app.use(express.json())

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

app.get("/", async(req,res)=>{

    let result = await spp.findAll()

    res.json(result)

})

app.post("/", async(req,res)=>{

    let data = {

        tahun: req.body.tahun,

        nominal: req.body.nominal

    }

    spp.create(data)

    .then(result => {

        res.json({message: "data inserted"})

    })

    .catch(error => {

        res.json({message:error.message})

    })

})

app.put("/", async(req,res)=>{

    let param = await {id\_spp:req.body.id\_spp}

    let data = await {

        tahun: req.body.tahun,

        nominal: req.body.nominal

    }

    spp.update(data, {where:param})

    .then(result => {

        res.json({message: "data updated"})

    })

    .catch(error =>{

        res.json({message: error.message})

    })

})

app.delete("/:id\_spp", async(req,res)=>{

    let param = {id\_spp:req.params.id\_spp}

    spp.destroy({where:param})

    .then(result=>{

        res.json({message: "data destroyed"})

    })

    .catch(error=>{

        res.json({message: error.message})

    })

})

module.exports = app

1. Membuat folder image/admin\_img dan image/siswa\_img
2. Membuat file auth-admin.js, auth-petugas.js, auth-siswa.js

* Auth-admin.js

const jwt = require("jsonwebtoken")

const SECRET\_KEY = "testing"

// function level\_1 ()

auth = (req,res,next)=>{

    let header = req.headers.authorization

    let token = header && header.split(" ")[1]

    let jwtheader = {

        algorithm: "HS256"

    }

    if(token===null){

        res.status(404).json({message:"unauthorized"})

    }else{

        jwt.verify(token, SECRET\_KEY, jwtheader, (error,user)=>{

            if(error){

                res

                .status(401)

                .json({

                    message:"invalid token"

                })

                console.log(error)

            } else {

                console.log(user)

                next()

            }

        })

    }

}

module.exports = auth

* Auth-petugas.js

const jwt = require("jsonwebtoken")

const SECRET\_KEY = "cobain"

// function level\_1 ()

auth = (req, res, next) => {

    let header = req.headers.authorization

    let token = header && header.split(" ")[1]

    let jwtheader = {

        algorithm: "HS256"

    }

    if (token === null) {

        res.status(404).json({ message: "unauthorized" })

    } else {

        jwt.verify(token, SECRET\_KEY, jwtheader, (error, user) => {

            if (error) {

                res

                    .status(401)

                    .json({

                        message: "invalid token"

                    })

                console.log(error)

            } else {

                console.log(user)

                next()

            }

        })

    }

}

module.exports = auth

* Auth-siswa.js

const jwt = require("jsonwebtoken")

const SECRET\_KEY = "cobaen"

// function level\_1 ()

auth = (req, res, next) => {

    let header = req.headers.authorization

    let token = header && header.split(" ")[1]

    let jwtheader = {

        algorithm: "HS256"

    }

    if (token === null) {

        res.status(404).json({ message: "unauthorized" })

    } else {

        jwt.verify(token, SECRET\_KEY, jwtheader, (error, user) => {

            if (error) {

                res

                    .status(401)

                    .json({

                        message: "invalid token"

                    })

                console.log(error)

            } else {

                console.log(user)

                next()

            }

        })

    }

}

module.exports = auth