Graphical user interface

Description automatically generated with medium confidence

Config.js

|  |
| --- |
| "scripts": {  "start": "node index.js",  "dev": "nodemon index.js"  }, |

Migration dan Model

sequelize model:create --name barang –attributes id\_member:integer

sequelize model:create --name petugas –attributes id\_member:integer

sequelize model:create --name masyarakat –attributes id\_member:integer

sequelize model:create --name lelang –attributes id\_member:integer

sequelize model:create --name history\_lelang –attributes id\_member:integer

Graphical user interface, text, application

Description automatically generated

* Migration file

|  |
| --- |
| "use strict"  module.exports = {  up: async (queryInterface, Sequelize) => {  await queryInterface.createTable("barang", {  id: {  allowNull: false,  autoIncrement: true,  primaryKey: true,  type: Sequelize.INTEGER,  },  nama: {  type: Sequelize.STRING,  allowNull: false  },  tgl: {  type: Sequelize.STRING,  },  hargaAwal: {  type: Sequelize.INTEGER,  defaultValue: 0  },  deskripsi: {  type: Sequelize.STRING,  },  image: {  type: Sequelize.STRING,  allowNull: false  },  createdAt: {  allowNull: false,  type: Sequelize.DATE,  },  updatedAt: {  allowNull: false,  type: Sequelize.DATE,  },  })  },  down: async (queryInterface, Sequelize) => {  await queryInterface.dropTable("barang")  },  } |
| "use strict"  module.exports = {  up: async(queryInterface, Sequelize) => {  await queryInterface.createTable("petugas", {  id: {  allowNull: false,  autoIncrement: true,  primaryKey: true,  type: Sequelize.INTEGER,  },  nama: {  type: Sequelize.STRING,  allowNull: false  },  username: {  type: Sequelize.STRING,  allowNull: false  },  password: {  type: Sequelize.STRING,  allowNull: false  },  level: {  type: Sequelize.ENUM("admin", "petugas"),  allowNull: false  },  createdAt: {  allowNull: false,  type: Sequelize.DATE,  },  updatedAt: {  allowNull: false,  type: Sequelize.DATE,  },  })  },  down: async(queryInterface, Sequelize) => {  await queryInterface.dropTable("petugas")  },  } |
| "use strict"  module.exports = {  up: async(queryInterface, Sequelize) => {  await queryInterface.createTable("masyarakat", {  id: {  allowNull: false,  autoIncrement: true,  primaryKey: true,  type: Sequelize.INTEGER,  },  nama: {  type: Sequelize.STRING,  allowNull: false  },  username: {  type: Sequelize.STRING,  allowNull: false  },  password: {  type: Sequelize.STRING,  allowNull: false  },  telp: {  type: Sequelize.STRING,  allowNull: false  },  createdAt: {  allowNull: false,  type: Sequelize.DATE,  },  updatedAt: {  allowNull: false,  type: Sequelize.DATE,  },  })  },  down: async(queryInterface, Sequelize) => {  await queryInterface.dropTable("masyarakat")  },  } |
| "use strict"  module.exports = {  up: async(queryInterface, Sequelize) => {  await queryInterface.createTable("lelang", {  id: {  allowNull: false,  autoIncrement: true,  primaryKey: true,  type: Sequelize.INTEGER,  },  idBarang: {  type: Sequelize.INTEGER,  allowNull: false,  references: {  model: "barang",  key: "id",  },  },  tglLelang: {  type: Sequelize.DATE,  allowNull:true  },  hargaAkhir: {  type: Sequelize.INTEGER,  allowNull:true  },  idMasyarakat: {  type: Sequelize.INTEGER,  allowNull: true,  references: {  model: "masyarakat",  key: "id",  },  },  idPetugas: {  type: Sequelize.INTEGER,  allowNull: false,  references: {  model: "petugas",  key: "id",  },  },  status: {  type: Sequelize.ENUM("Dibuka", "Ditutup"),  defaultValue: "Ditutup",  },  endTime: {  type: Sequelize.STRING,  allowNull: true  },  createdAt: {  allowNull: false,  type: Sequelize.DATE,  },  updatedAt: {  allowNull: false,  type: Sequelize.DATE,  },  })  },  down: async(queryInterface, Sequelize) => {  await queryInterface.dropTable("lelang")  },  } |
| "use strict"  module.exports = {  up: async(queryInterface, Sequelize) => {  await queryInterface.createTable("history\_lelang", {  id: {  allowNull: false,  autoIncrement: true,  primaryKey: true,  type: Sequelize.INTEGER,  },  idLelang: {  type: Sequelize.INTEGER,  allowNull: false,  references: {  model: "lelang",  key: "id",  },  },  idMasyarakat: {  type: Sequelize.INTEGER,  allowNull: false,  references: {  model: "masyarakat",  key: "id",  },  },    penawaranHarga: {  type: Sequelize.INTEGER,  allowNull: false  },  createdAt: {  allowNull: false,  type: Sequelize.DATE,  },  updatedAt: {  allowNull: false,  type: Sequelize.DATE,  },  })  },  down: async(queryInterface, Sequelize) => {  await queryInterface.dropTable("history\_lelang")  },  } |

* Models File

|  |
| --- |
| "use strict"  const { Model } = require("sequelize")  module.exports = (sequelize, DataTypes) => {  class barang 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.lelang, {  foreignKey: "idBarang",  })  }  }  barang.init({  id: {  type: DataTypes.INTEGER,  primaryKey: true,  autoIncrement: true,  },  nama: DataTypes.STRING,  tgl: DataTypes.DATE,  hargaAwal: DataTypes.INTEGER,  deskripsi: DataTypes.STRING,  image:DataTypes.STRING,  }, {  sequelize,  modelName: "barang",  tableName: "barang",  })  return barang  } |
| "use strict"  const { Model } = require("sequelize")  module.exports = (sequelize, DataTypes) => {  class history\_lelang 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.lelang, {  foreignKey: "id",  as: "lelang",  })  this.belongsTo(models.masyarakat, {  foreignKey: "id",  as: "masyarakat",  })  }  }  history\_lelang.init({  id: {  type: DataTypes.INTEGER,  primaryKey: true,  autoIncrement: true,  },  idLelang: DataTypes.INTEGER,  idMasyarakat: DataTypes.INTEGER,  penawaranHarga: DataTypes.INTEGER,  }, {  sequelize,  modelName: "history\_lelang",  tableName: "history\_lelang",  })  return history\_lelang  } |
| "use strict"  const { Model } = require("sequelize")  module.exports = (sequelize, DataTypes) => {  class lelang 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.barang, {  foreignKey: "id",  as: "barang",  })  this.belongsTo(models.masyarakat, {  foreignKey: "id",  as: "masyarakat",  })  this.belongsTo(models.petugas, {  foreignKey: "id",  as: "petugas",  })  }  }  lelang.init({  id: {  type: DataTypes.INTEGER,  primaryKey: true,  autoIncrement: true,  },  idBarang: DataTypes.INTEGER,  tglLelang: DataTypes.DATE,  hargaAkhir: DataTypes.INTEGER,  idMasyarakat: DataTypes.INTEGER,  idPetugas: DataTypes.INTEGER,  status: DataTypes.ENUM("Dibuka", "Ditutup"),  endTime: DataTypes.DATE  }, {  sequelize,  modelName: "lelang",  tableName: "lelang",  })  return lelang  } |
| "use strict"  const { Model } = require("sequelize")  module.exports = (sequelize, DataTypes) => {  class masyarakat 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.history\_lelang, {  foreignKey: "idMasyarakat",  })  this.hasMany(models.lelang, {  foreignKey: "idMasyarakat",  })  }  }  masyarakat.init({  id: {  type: DataTypes.INTEGER,  primaryKey: true,  autoIncrement: true,  },  nama: DataTypes.STRING,  username: DataTypes.STRING,  password: DataTypes.STRING,  telp: DataTypes.STRING,  }, {  sequelize,  modelName: "masyarakat",  tableName: "masyarakat",  })  return masyarakat  } |
| "use strict"  const { Model } = require("sequelize")  module.exports = (sequelize, DataTypes) => {  class petugas 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) {  }  }  petugas.init({  id: {  type: DataTypes.INTEGER,  primaryKey: true,  autoIncrement: true,  },  nama: DataTypes.STRING,  username: DataTypes.STRING,  password: DataTypes.STRING,  level: DataTypes.ENUM("admin", "petugas"),  }, {  sequelize,  modelName: "petugas",  tableName: "petugas",  })  return petugas  } |

* Helper File

|  |
| --- |
| date.js |
| const toIsoString = (date) => {  var tzo = -date.getTimezoneOffset(),  dif = tzo >= 0 ? '+' : '-',  pad = function(num) {  var norm = Math.floor(Math.abs(num));  return (norm < 10 ? '0' : '') + norm;  };    return date.getFullYear() +  '-' + pad(date.getMonth() + 1) +  '-' + pad(date.getDate()) +  'T' + pad(date.getHours()) +  ':' + pad(date.getMinutes()) +  ':' + pad(date.getSeconds()) +  '.' + pad(date.getMilliseconds()) + 'Z'    }  module.exports = {  toIsoString  } |
| endTime.js |
| const { CronJob } = require('cron')  const { lelangStatus } = require('./enum')  const { toIsoString } = require('./date')  const { lelang } = require('../models/index')  const runTime = (time, timestamp, id) => {  let job = new CronJob(time, async () => {  const date = new Date(Date.now())  const date7 = new Date(toIsoString(date))  const now = date7.getTime()  if (now > timestamp) {  await lelang.update({ status: lelangStatus.DITUTUP }, { where: { id: id } })  job.stop()  }  });  job.start()  }  module.exports = {  runTime  } |
| enum.js |
| const lelangStatus = {  DIBUKA : 'Dibuka',  DITUTUP :'Ditutup'  }  module.exports = {  lelangStatus  } |
| wrapper.js |
| const response = (res, type, result, message = '', code) => {  let status = true;  let data = result;  if (type === 'fail') {  status = false;  data = data || '';  message = message;  }  res.send(code, {  success: status,  data,  message,  code,  });  };  const logged = (res, type, result, token, message = '', code) => {  let status = true;  let data = result;  if (type === 'fail') {  status = false;  data = data || '';  message = message;  }  res.send(code, {  logged: status,  data,  token,  message,  code,  });  };  module.exports = {  response,  logged  } |

* Router file

|  |
| --- |
| barang.js |
| const express = require("express")  const app = express()  const { response } = require('../helper/wrapper')  const { toIsoString } = require('../helper/date')  const multer = require('multer')  const fs = require('fs')  const path = require("path")  const auth = require('../auth')  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  const { barang } = require('../models/index')  const storage = multer.diskStorage({  destination: (req, file, cb) => {  cb(null, "./barangImage")  },  filename: (req, file, cb) => {  cb(null, "img-" + Date.now() + path.extname(file.originalname))  }  })  let upload = multer({ storage: storage })  app.get("/", auth("admin", "petugas", "masyarakat"),async (req, res) => {  await barang.findAll()  .then(result => {  return response(res, 'success', result, 'Success get data barang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed get data barang', 400)  })  })  app.get("/:id", auth("admin", "petugas", "masyarakat"), async (req, res) => {  const param = {  id: req.params.id  }  await barang.findOne({ where: param })  .then(result => {  return response(res, 'success', result, 'Success get data masyarakat', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed get data masyarakat', 400)  })  })  app.post("/", upload.single("image"), auth("admin", "petugas"),async (req, res) => {  const date = new Date(Date.now())  if (!req.file) {  return response(res, 'fail', '', 'Image is required', 400)  } else {  const data = {  nama: req.body.nama,  hargaAwal: req.body.hargaAwal,  deskripsi: req.body.deskripsi,  tgl: toIsoString(date),  image: req.file.filename  }  await barang.create(data)  .then(result => {  return response(res, 'success', result, 'Success create data barang', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed create data barang', 400)  })  }  })  app.put("/", upload.single("image"), auth("admin", "petugas"), async (req, res) => {  const date = new Date(Date.now())  const param = {  id: req.body.id  }  const data = {  nama: req.body.nama,  hargaAwal: req.body.hargaAwal,  deskripsi: req.body.deskripsi,  tgl: toIsoString(date)  }  if (req.file) {  const result = await barang.findOne({ where: param })  const oldFileName = result.image  *// delete old file*  const dir = await path.join(\_\_dirname, "../barangImage", oldFileName)  fs.unlink(dir, err => console.log(err))  *// set new filename*  data.image = req.file.filename  }  await barang.update(data, { where: param })  .then(result => {  return response(res, 'success', result, 'Success update data barang', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed create data barang', 400)  })  })  app.delete("/:id", auth("admin", "petugas"), async (req, res) => {  const param = {  id: req.params.id  }  let result = await barang.findOne({ where: param })  let oldFileName = result.image  *// delete old file*  let dir = path.join(\_\_dirname, "../barangImage", oldFileName)  fs.unlink(dir, err => console.log(err))  await barang.destroy({ where: param })  .then(result => {  return response(res, 'success', result, 'Success delete data barang', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed update data barang', 400)  })  })  module.exports = app |
| petugas.js |
| const express = require("express")  const app = express()  const { response } = require('../helper/wrapper')  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  const { petugas } = require('../models/index')  const { masyarakat } = require('../models/index')  const md5 = require("md5")  const auth = require('../auth')  app.get("/", auth("admin", "petugas"),async (req, res) => {  await petugas.findAll()  .then(result => {  return response(res, 'success', result, 'Success get data petugas', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed get data petugas', 400)  })  })  app.get("/:id", auth("admin", "petugas"),async (req, res) => {  const param = {  id: req.params.id  }  await petugas.findOne({ where: param })  .then(result => {  return response(res, 'success', result, 'Success get data petugas', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed get data petugas', 400)  })  })  app.post("/", auth("admin"),async (req, res) => {  const data = {  nama: req.body.nama,  password: md5(req.body.password),  level: req.body.level  }  const param = {  username: req.body.username  }  const resultMasyarakat = await masyarakat.findOne({ where: param })  const resultPetugas = await petugas.findOne({ where: param })  if (resultMasyarakat || resultPetugas) {  return response(res, 'fail', '', 'Username Already exist', 400)  } else {  data.username = param.username  await petugas.create(data)  .then(result => {  return response(res, 'success', result, 'Success create data petugas', 201)  })  .catch(err => {  return response(res, 'fail', err, 'Failed create data petugas', 400)  })  }  })  app.put("/", auth("admin", "petugas"),async (req, res) => {  const param = {  id: req.body.id  }  const payload = {  nama: req.body.nama,  level: req.body.level  }  const data = {}  if (req.body.password) {  payload.password = md5(req.body.password)  }  if (req.body.username) {  data.username = req.body.username  const resultMasyarakat = await masyarakat.findOne({ where: data })  const resultPetugas = await petugas.findOne({ where: data })  if (resultMasyarakat || resultPetugas) {  return response(res, 'fail', '', 'Username Already exist', 400)  } else {  payload.username = data.username  await petugas.update(payload, { where: param })  .then(result => {  return response(res, 'fail', result, 'Success update data petugas', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data petugas', 400)  })  }  } else {  payload.username = data.username  await petugas.update(payload, { where: param })  .then(result => {  return response(res, 'fail', result, 'Success update data petugas', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data petugas', 400)  })  }  })  app.delete("/:id", auth("admin"),async (req, res) => {  const param = {  id: req.params.id  }  petugas.destroy({ where: param })  .then(result => {  return response(res, 'success', result, 'Success delete data petugas', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data petugas', 400)  })  })  module.exports = app |
| masyarakat.js |
| const express = require("express")  const app = express()  const { response } = require('../helper/wrapper')  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  const { masyarakat } = require('../models/index')  const { petugas } = require('../models/index')  const md5 = require("md5")  app.get("/", auth("admin", "petugas", "masyarakat"), async (req, res) => {  await masyarakat.findAll()  .then(result => {  return response(res, 'success', result, 'Success get data petugas', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed get data petugas', 400)  })  })  app.get("/:id", auth("admin", "petugas", "masyarakat"), async (req, res) => {  const param = {  id: req.params.id  }  await masyarakat.findOne({ where: param })  .then(result => {  return response(res, 'success', result, 'Success get data masyarakat', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed get data masyarakat', 400)  })  })  app.post("/register", async (req, res) => {  const data = {  nama: req.body.nama,  password: md5(req.body.password),  telp: req.body.telp  }  const param = {  username: req.body.username,  }  const resultMasyarakat = await masyarakat.findOne({ where: param })  const resultPetugas = await petugas.findOne({ where: param })  if (resultMasyarakat || resultPetugas) {  return response(res, 'fail', '', 'Username Already exist', 400)  } else {  data.username = param.username  await masyarakat.create(data)  .then(result => {  return response(res, 'success', result, 'Success create data masyarakat', 201)  })  .catch(err => {  return response(res, 'fail', err, 'Failed create data masyarakat', 400)  })  }  })  app.put("/", auth("masyarakat"), async (req, res) => {  const param = {  id: req.body.id  }  const payload = {  nama: req.body.nama,  telp: req.body.telp  }  const data = {}  if (req.body.password) {  payload.password = md5(req.body.password)  }  if (req.body.username) {  data.username = req.body.username  const resultMasyarakat = await masyarakat.findOne({ where: data })  const resultPetugas = await petugas.findOne({ where: data })  if (resultMasyarakat || resultPetugas) {  return response(res, 'fail', '', 'Username Already exist', 400)  } else {  payload.username = data.username  await masyarakat.update(payload, { where: param })  .then(result => {  return response(res, 'fail', result, 'Success update data masyarakat', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data masyarakat', 400)  })  }  } else {  payload.username = data.username  await masyarakat.update(payload, { where: param })  .then(result => {  return response(res, 'fail', result, 'Success update data masyarakat', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data masyarakat', 400)  })  }  })  app.delete("/:id", auth("admin"), async (req, res) => {  const param = {  id: req.params.id  }  masyarakat.destroy({ where: param })  .then(result => {  return response(res, 'success', result, 'Success delete data masyarakat', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed delete data masyarakat', 400)  })  })  module.exports = app |
| lelang.js |
| const express = require("express")  const app = express()  const { response } = require('../helper/wrapper')  const { toIsoString } = require('../helper/date')  const { lelangStatus } = require('../helper/enum')  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  const { lelang } = require('../models/index')  const { barang } = require('../models/index')  const { history\_lelang } = require('../models/index')  const { runTime } = require('../helper/endTime')  app.get("/", auth("admin", "petugas", "masyarakat"), async (req, res) => {  await lelang.findAll()  .then(result => {  return response(res, 'success', result, 'Success get data lelang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed get data lelang', 400)  })  })  app.get("/:id", auth("admin", "petugas", "masyarakat"), async (req, res) => {  const param = {  id: req.params.id  }  await lelang.findOne({ where: param })  .then(result => {  return response(res, 'success', result, 'Success get data lelang', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed get data lelang', 400)  })  })  app.post("/", auth("admin", "petugas"),async (req, res) => {  const date = new Date(Date.now())  const idBarang = {  id: req.body.idBarang  }  const resultBarang = await barang.findOne({ where: idBarang })  const data = {  idBarang: req.body.idBarang,  tglLelang: toIsoString(date),  hargaAkhir: resultBarang.dataValues.hargaAwal,  idPetugas: req.body.idPetugas,  status: req.body.status  }  if (data.status === lelangStatus.DIBUKA) {  let end = new Date(req.body.endTime)  let timeStamp = end.getTime()  data.endTime = end  await lelang.create(data)  .then(result => {  runTime('\* \* \* \* \* \*', timeStamp, result.dataValues.id)  return response(res, 'success', result, 'Success create data lelang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed create data lelang', 400)  })  } else {  await lelang.create(data)  .then(result => {  return response(res, 'success', result, 'Success create data lelang', 201)  })  .catch(err => {  return response(res, 'fail', err, 'Failed create data lelang', 400)  })  }  })  app.put("/", auth("admin", "petugas"), async (req, res) => {  const date = new Date(Date.now())  const idBarang = {  id: req.body.idBarang  }  const resultBarang = await barang.findOne({ where: idBarang })  const param = {  id: req.body.id  }  const data = {  idBarang: req.body.idBarang,  tglLelang: toIsoString(date),  hargaAkhir: resultBarang.dataValues.hargaAwal,  idPetugas: req.body.idPetugas,  status: req.body.status  }  if (data.status === lelangStatus.DIBUKA) {  let end = new Date(req.body.endTime)  let timeStamp = end.getTime()  data.endTime = end  await lelang.update(data, {where: param})  .then(result => {  runTime('\* \* \* \* \* \*', timeStamp, param.id)  return response(res, 'success', result, 'Success update data lelang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data lelang', 400)  })  } else {  await lelang.create(data)  .then(result => {  return response(res, 'success', result, 'Success create data lelang', 201)  })  .catch(err => {  return response(res, 'fail', err, 'Failed create data lelang', 400)  })  }  })  app.post("/bid", auth("masyarakat"),async (req, res) => {  const idLelang = {  id: req.body.id  }  const resultLelang = await lelang.findOne({ where: idLelang })  const { hargaAkhir, status } = resultLelang.dataValues  const data = {  idLelang: idLelang.id,  idMasyarakat: req.body.idMasyarakat,  penawaranHarga: req.body.penawaranHarga  }  if (status === lelangStatus.DITUTUP) {  return response(res, 'fail', '', 'Status is closed', 400)  }  if (data.penawaranHarga <= hargaAkhir) {  return response(res, 'fail', '', 'bid must be higher', 400)  }  await history\_lelang.create(data)  await lelang.update({ hargaAkhir: data.penawaranHarga, idMasyarakat: data.idMasyarakat }, { where: idLelang })  .then(result => {  return response(res, 'success', result, 'Success bid', 201)  })  .catch(err => {  return response(res, 'fail', err, 'Failed bid', 400)  })  })  app.delete("/:id", async(req, res) => {  const param = {  id: req.params.id  }  lelang.destroy({ where: param })  .then(result => {  return response(res, 'success', result, 'Success delete data lelang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed update data lelang', 400)  })  })  module.exports = app |
| historyLelang.js |
| const express = require("express")  const app = express()  const { response } = require('../helper/wrapper')  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  const { history\_lelang } = require('../models/index')  app.get("/", auth("admin", "petugas", "masyarakat"), async(req, res) => {  await history\_lelang.findAll()  .then(result => {  return response(res, 'success', result, 'Success get data history lelang', 200)  })  .catch(err => {  return response(res, 'fail', err, 'Failed get data history lelang', 400)  })  })  app.get("/:id", auth("admin", "petugas", "masyarakat"), async(req, res) => {  const param = {  id: req.params.id  }  await history\_lelang.findOne({ where: param })  .then(result => {  return response(res, 'success', result, 'Success get data history lelang', 200)  }).catch(err => {  return response(res, 'fail', err, 'Failed get data history lelang', 400)  })  })  module.exports = app |
| login.js |
| const express = require("express")  const app = express()  app.use(express.json())  app.use(express.urlencoded({ extended: true }))  require('dotenv').config()  const SECRET\_KEY = process.env.SECRET\_KEY  const { masyarakat } = require('../models/index')  const { petugas } = require('../models/index')  const md5 = require("md5")  const jwt = require("jsonwebtoken")  const {logged} = require('../helper/wrapper')  app.post("/", async(req, res) => {  const param = {  username: req.body.username,  password: md5(req.body.password)  }  const resultMasyarakat = await masyarakat.findOne({ where: param })  const resultPetugas = await petugas.findOne({ where: param })  if(resultMasyarakat){  resultMasyarakat.dataValues.level = "masyarakat"  let payload = JSON.stringify(resultMasyarakat)  let token = jwt.sign(payload, SECRET\_KEY)  return logged(res, 'success', resultMasyarakat, token, 'Success login masyarakat', 200)  }else if(resultPetugas){  let payload = JSON.stringify(resultPetugas)  let token = jwt.sign(payload, SECRET\_KEY)  return logged(res, 'success', resultPetugas, token, 'Success login masyarakat', 200)  }else {  return logged(res, 'fail', '', '', 'Invalid Username or Password', 200)  }  })  module.exports = app |

* Auth.js

|  |
| --- |
| const jwt = require("jsonwebtoken")  require('dotenv').config()  const SECRET\_KEY = process.env.SECRET\_KEY  const {response} = require('./helper/wrapper')  auth = (role1, role2, role3) => {  let role = [role1, role2, role3]  return (req, res, next) => {  let header = req.headers.authorization  let token = header && header.split(" ")[1]  let jwtHeader = {  algorithm: "HS256"  }  if (token == null) {  return response(res, 'fail', '', 'Unauthorized', 401)  } else {  jwt.verify(token, SECRET\_KEY, jwtHeader, (error, user) => {  if (error) {  return response(res, 'fail', '', 'Invalid Token', 401)  } else {  if (role.includes(user.level)) {  console.log(role);  req.user = user  *// if(user.username == )*  next()  }else{  return response(res, 'fail', '', 'You dont have access', 401)  }  }  })  }  }  }  module.exports = auth |

* Index.js

|  |
| --- |
| const express = require("express")  const app = express()  const cors = require('cors')  const petugas = require('./router/petugas')  const masyarakat = require('./router/masyarakat')  const barang = require('./router/barang')  const lelang = require('./router/lelang')  const historyLelang = require('./router/historyLelang')  const login = require('./router/login')  require('dotenv').config()  const api = process.env.API  const port = process.env.PORT  app.use(cors())  app.use(`${api}/petugas`, petugas)  app.use(`${api}/masyarakat`, masyarakat)  app.use(`${api}/barang`, barang)  app.use(`${api}/lelang`, lelang)  app.use(`${api}/historyLelang`, historyLelang)  app.use(`${api}/login`, login)  app.listen(port, () => {  console.log(`server run on port ${port}`);  }) |
| .env |
| PORT = 8000  API = /api/v1  SECRET\_KEY = LELANG |