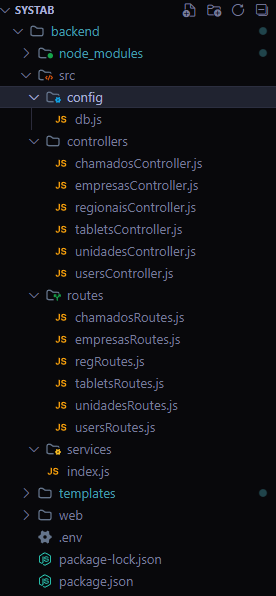
systab.docx

Esse documento contém toda a estrutura e códigos referentes ao projeto Systab, favor não perdê-lo. Guarde-o na memória.

Estruturação:



SysTab\backend :

- SysTab\backend\src\ : {

- src\config\ : { src\config\db.js }

- src\controllers\ : {

- src\controllers\chamadosController.js

- src\controllers\empresasController.js

- src\controllers\regionaisController.js

- src\controllers\tabletsController.js

- src\controllers\unidadesController.js

- src\controllers\usersController.js

}

- src\routes\ : {

- src\routes\chamadosRoutes.js

- src\routes\empresasRoutes.js

- src\routes\regRoutes.js

- src\routes\tabletsRoutes.js

- src\routes\unidadesRoutes.js

- src\routes\usersRoutes.js

}

- src\index.js

}

- SysTab\backend\.env

// A Seguir linhas de código de cada arquivo citado acima.

// Código:

* src\config\db.js :

const mysql = require("mysql2");

require("dotenv").config();

const db = mysql.createPool({

port: process.env.DB\_PORT,

host: process.env.DB\_HOST,

user: process.env.DB\_USER,

password: process.env.DB\_PASSWORD,

database: process.env.DB\_NAME,

});

db.getConnection((err, connection) => {

if (err) {

console.error("Erro ao conectar ao banco de dados:", err);

} else {

console.log("Conectado ao banco de dados MySQL!");

connection.release();

}

});

module.exports = db;

* src\controllers\chamadosController.js :

const db = require('../config/db');

const PDFDocument = require('pdfkit');

const { format } = require('date-fns');

const { ptBR } = require('date-fns/locale');

// Criar novo chamado

exports.criarChamado = async (req, res) => {

const { idTablet, problema, itens\_recebidos } = req.body;

try {

const [tablet] = await db.query('SELECT \* FROM tablets WHERE id = ?', [idTablet]);

if (tablet.length === 0) return res.status(404).json({ message: 'Tablet não encontrado.' });

const [result] = await db.query(

'INSERT INTO chamados (idTablet, problema, itens\_recebidos) VALUES (?, ?, ?)',

[idTablet, problema, itens\_recebidos]

);

res.status(201).json({ message: 'Chamado criado com sucesso.', id: result.insertId });

} catch (err) {

res.status(500).json({ message: 'Erro ao criar chamado.', error: err });

}

};

// Listar todos os chamados

exports.listarChamado = async (req, res) => {

try {

const [rows] = await db.query('SELECT \* FROM chamados');

res.json(rows);

} catch (err) {

res.status(500).json({ message: 'Erro ao buscar chamados.', error: err });

}

};

// Listar chamados por tablet

exports.listarChamadoPorTablet = async (req, res) => {

const { idTablet } = req.params;

try {

const [rows] = await db.query('SELECT \* FROM chamados WHERE idTablet = ?', [idTablet]);

res.json(rows);

} catch (err) {

res.status(500).json({ message: 'Erro ao buscar chamados por tablet.', error: err });

}

};

// Atualizar chamado

exports.atualizarChamado = async (req, res) => {

const { id } = req.params;

const { problema, itens\_recebidos } = req.body;

try {

await db.query(

'UPDATE chamados SET problema = ?, itens\_recebidos = ? WHERE id = ?',

[problema, itens\_recebidos, id]

);

res.json({ message: 'Chamado atualizado com sucesso.' });

} catch (err) {

res.status(500).json({ message: 'Erro ao atualizar chamado.', error: err });

}

};

// Fechar chamado

exports.atualizarStatus = async (req, res) => {

const { id } = req.params;

try {

await db.query(

'UPDATE chamados SET status = "Fechado", dataSaida = NOW() WHERE id = ?',

[id]

);

res.json({ message: 'Chamado fechado com sucesso.' });

} catch (err) {

res.status(500).json({ message: 'Erro ao fechar chamado.', error: err });

}

};

// Deletar chamado

exports.deletarChamado = async (req, res) => {

const { id } = req.params;

try {

await db.query('DELETE FROM chamados WHERE id = ?', [id]);

res.json({ message: 'Chamado deletado com sucesso.' });

} catch (err) {

res.status(500).json({ message: 'Erro ao deletar chamado.', error: err });

}

};

// Listar chamados abertos há mais de X dias

exports.listarChamadosAtrasados = async (req, res) => {

const dias = parseInt(req.query.dias) || 7;

try {

const [rows] = await db.query(

`SELECT \* FROM chamados

WHERE status = 'Aberto' AND dataEntrada < DATE\_SUB(NOW(), INTERVAL ? DAY)`,

[dias]

);

res.json(rows);

} catch (err) {

res.status(500).json({ message: 'Erro ao buscar chamados antigos.', error: err });

}

};

// Buscar chamado pelo ID do chamado

exports.buscarChamadoPorIdChamado = async (req, res) => {

const { id } = req.params;

try {

const [rows] = await db.query('SELECT \* FROM chamados WHERE id = ?', [id]);

if (rows.length === 0) {

return res.status(404).json({ message: 'Chamado não encontrado.' });

}

res.json(rows[0]);

} catch (err) {

res.status(500).json({ message: 'Erro ao buscar chamado.', error: err });

}

};

// Gerar Ordem de Serviço (PDF)

exports.gerarOS = async (req, res) => {

const { id } = req.params;

try {

const [chamadoRows] = await db.query('SELECT \* FROM chamados WHERE id = ?', [id]);

if (chamadoRows.length === 0) return res.status(404).json({ message: 'Chamado não encontrado.' });

const chamado = chamadoRows[0];

const [tabletRows] = await db.query(`

SELECT t.\*, u.nome AS nome\_unidade, r.nome AS nome\_regional, e.nome AS nome\_empresa

FROM tablets t

JOIN unidades u ON t.idUnidade = u.id

JOIN regionais r ON u.idRegional = r.id

JOIN empresas e ON t.idEmpresa = e.id

WHERE t.id = ?

`, [chamado.idTablet]);

if (tabletRows.length === 0) return res.status(404).json({ message: 'Tablet vinculado não encontrado.' });

const tablet = tabletRows[0];

const doc = new PDFDocument();

res.setHeader('Content-Type', 'application/pdf');

res.setHeader('Content-Disposition', `inline; filename="os\_chamado\_${id}.pdf"`);

doc.fontSize(20).text('Ordem de Serviço - Chamado de Tablet', { align: 'center' });

doc.moveDown();

const dataFormatada = format(new Date(chamado.dataEntrada), "dd 'de' MMMM 'de' yyyy", { locale: ptBR });

doc.fontSize(12).text(`ID do Chamado: ${chamado.id}`);

doc.text(`Data de Entrada: ${dataFormatada}`);

doc.text(`Status: ${chamado.status}`);

doc.text(`Problema: ${chamado.problema}`);

doc.text(`Itens Recebidos: ${chamado.itens\_recebidos ? 'Sim' : 'Não'}`);

doc.moveDown();

doc.text(`Tablet - Tombamento: ${tablet.tombamento}`);

doc.text(`Número de Série: ${tablet.ns}`);

doc.text(`Usuário: ${tablet.usuario}`);

doc.text(`Unidade: ${tablet.nome\_unidade}`);

doc.text(`Regional: ${tablet.nome\_regional}`);

doc.text(`Empresa: ${tablet.nome\_empresa}`);

doc.end();

doc.pipe(res);

} catch (err) {

console.error(err);

res.status(500).json({ message: 'Erro ao gerar Ordem de Serviço.', error: err });

}

};

* src\controllers\empresasController.js :

const db = require("../config/db");

// Criar empresa

exports.criarEmpresa = (req, res) => {

const { nome } = req.body;

if (!nome) {

return res.status(400).json({ error: "Nome da empresa é obrigatório." });

}

const sql = "INSERT INTO empresas (nome) VALUES (?)";

db.query(sql, [nome], (err, result) => {

if (err) {

console.error("Erro ao criar empresa:", err);

return res.status(500).json({ error: "Erro ao criar empresa." });

}

res.status(201).json({ message: "Empresa criada com sucesso.", idEmpresa: result.insertId });

});

};

// Listar empresas

exports.listarEmpresas = (req, res) => {

db.query("SELECT \* FROM empresas", (err, result) => {

if (err) {

return res.status(500).json({ error: "Erro ao listar empresas." });

}

res.json(result);

});

};

// Deletar empresa

exports.deletarEmpresa = (req, res) => {

const { id } = req.params;

db.query("DELETE FROM empresas WHERE idEmpresa = ?", [id], (err, result) => {

if (err) {

return res.status(500).json({ error: "Erro ao deletar empresa." });

}

res.json({ message: "Empresa deletada com sucesso." });

});

};

* src\controllers\regionaisController.js :

const db = require("../config/db");

exports.criarRegional = (req, res) => {

const { nome } = req.body;

if (!nome) return res.status(400).json({ error: "Nome da regional é obrigatório." });

const sql = "INSERT INTO regionais (nome) VALUES (?)";

db.query(sql, [nome], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao criar regional." });

res.status(201).json({ message: "Regional criada com sucesso.", idRegional: result.insertId });

});

};

exports.listarRegionais = (req, res) => {

db.query("SELECT \* FROM regionais", (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao listar regionais." });

res.json(result);

});

};

exports.editarRegional = (req, res) => {

const { id } = req.params;

const { nome } = req.body;

const sql = "UPDATE regionais SET nome = ? WHERE idRegional = ?";

db.query(sql, [nome, id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao atualizar regional." });

res.json({ message: "Regional atualizada com sucesso." });

});

};

exports.deletarRegional = (req, res) => {

const { id } = req.params;

db.query("DELETE FROM regionais WHERE idRegional = ?", [id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao deletar regional." });

res.json({ message: "Regional deletada com sucesso." });

});

};

* src\controllers\tabletsController.js :

const db = require("../config/db");

exports.criarTablet = (req, res) => {

const { tombamento, imei, idUsuario, idUnidade, idEmpresa } = req.body;

if (!tombamento || !imei || !idUsuario || !idUnidade || !idEmpresa) {

return res.status(400).json({ error: "Todos os campos são obrigatórios." });

}

// Verificar se o usuário já possui um tablet

const verificarSql = "SELECT \* FROM tablets WHERE idUsuario = ?";

db.query(verificarSql, [idUsuario], (verifErr, verifResult) => {

if (verifErr) return res.status(500).json({ error: "Erro ao verificar usuário." });

if (verifResult.length > 0) {

return res.status(400).json({ error: "Este usuário já possui um tablet vinculado." });

}

const sql = "INSERT INTO tablets (tombamento, imei, idUsuario, idUnidade, idEmpresa) VALUES (?, ?, ?, ?, ?)";

db.query(sql, [tombamento, imei, idUsuario, idUnidade, idEmpresa], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao criar tablet." });

res.status(201).json({ message: "Tablet criado com sucesso.", idTablet: result.insertId });

});

});

};

exports.listarTablets = (req, res) => {

const sql = `

SELECT t.\*, u.nome AS usuario, un.nome AS unidade, r.nome AS regional, e.nome AS empresa

FROM tablets t

JOIN usuarios u ON t.idUsuario = u.idUsuario

JOIN unidades un ON t.idUnidade = un.idUnidade

JOIN regionais r ON un.idRegional = r.idRegional

JOIN empresas e ON t.idEmpresa = e.idEmpresa

`;

db.query(sql, (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao listar tablets." });

res.json(result);

});

};

exports.buscarTablet = (req, res) => {

const { tombamento, imei } = req.query;

if (!tombamento && !imei) {

return res.status(400).json({ error: "Informe tombamento ou IMEI." });

}

const sql = `

SELECT t.\*, u.nome AS usuario, un.nome AS unidade, r.nome AS regional, e.nome AS empresa

FROM tablets t

JOIN usuarios u ON t.idUsuario = u.idUsuario

JOIN unidades un ON t.idUnidade = un.idUnidade

JOIN regionais r ON un.idRegional = r.idRegional

JOIN empresas e ON t.idEmpresa = e.idEmpresa

WHERE t.tombamento = ? OR t.imei = ?

`;

db.query(sql, [tombamento, imei], (err, result) => {

if (err) return res.status(500).json({ error: "Erro na busca." });

res.json(result);

});

};

exports.editarTablet = (req, res) => {

const { id } = req.params;

const { tombamento, imei, idUsuario, idUnidade, idEmpresa } = req.body;

// Verificar se o novo usuário já possui outro tablet (excluindo o atual)

const verificarSql = "SELECT \* FROM tablets WHERE idUsuario = ? AND idTablet != ?";

db.query(verificarSql, [idUsuario, id], (verifErr, verifResult) => {

if (verifErr) return res.status(500).json({ error: "Erro ao verificar usuário." });

if (verifResult.length > 0) {

return res.status(400).json({ error: "Este usuário já possui outro tablet vinculado." });

}

const sql = `

UPDATE tablets

SET tombamento = ?, imei = ?, idUsuario = ?, idUnidade = ?, idEmpresa = ?

WHERE idTablet = ?

`;

db.query(sql, [tombamento, imei, idUsuario, idUnidade, idEmpresa, id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao atualizar tablet." });

res.json({ message: "Tablet atualizado com sucesso." });

});

});

};

exports.deletarTablet = (req, res) => {

const { id } = req.params;

db.query("DELETE FROM tablets WHERE idTablet = ?", [id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao deletar tablet." });

res.json({ message: "Tablet deletado com sucesso." });

});

};

* src\controllers\unidadesController.js :

const db = require("../config/db");

exports.criarUnidade = (req, res) => {

const { nome, idRegional } = req.body;

if (!nome || !idRegional) return res.status(400).json({ error: "Nome e regional são obrigatórios." });

const sql = "INSERT INTO unidades (nome, idRegional) VALUES (?, ?)";

db.query(sql, [nome, idRegional], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao criar unidade." });

res.status(201).json({ message: "Unidade criada com sucesso.", idUnidade: result.insertId });

});

};

exports.listarUnidades = (req, res) => {

db.query("SELECT unidades.\*, regionais.nome AS regional FROM unidades JOIN regionais ON unidades.idRegional = regionais.idRegional", (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao listar unidades." });

res.json(result);

});

};

exports.editarUnidade = (req, res) => {

const { id } = req.params;

const { nome, idRegional } = req.body;

const sql = "UPDATE unidades SET nome = ?, idRegional = ? WHERE idUnidade = ?";

db.query(sql, [nome, idRegional, id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao atualizar unidade." });

res.json({ message: "Unidade atualizada com sucesso." });

});

};

exports.deletarUnidade = (req, res) => {

const { id } = req.params;

db.query("DELETE FROM unidades WHERE idUnidade = ?", [id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao deletar unidade." });

res.json({ message: "Unidade deletada com sucesso." });

});

};

* src\controllers\usersController.js :

const db = require("../config/db");

// Criar usuário

exports.criarUsuario = (req, res) => {

const { nome } = req.body;

if (!nome) return res.status(400).json({ error: "Nome do usuário é obrigatório." });

const sql = "INSERT INTO usuarios (nome) VALUES (?)";

db.query(sql, [nome], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao criar usuário." });

res.status(201).json({ message: "Usuário criado com sucesso.", idUsuario: result.insertId });

});

};

// Listar usuários

exports.listarUsuarios = (req, res) => {

db.query("SELECT \* FROM usuarios", (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao listar usuários." });

res.json(result);

});

};

// Editar usuário

exports.editarUsuario = (req, res) => {

const { id } = req.params;

const { nome } = req.body;

const sql = "UPDATE usuarios SET nome = ? WHERE idUsuario = ?";

db.query(sql, [nome, id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao atualizar usuário." });

res.json({ message: "Usuário atualizado com sucesso." });

});

};

// Deletar usuário

exports.deletarUsuario = (req, res) => {

const { id } = req.params;

db.query("DELETE FROM usuarios WHERE idUsuario = ?", [id], (err, result) => {

if (err) return res.status(500).json({ error: "Erro ao deletar usuário." });

res.json({ message: "Usuário deletado com sucesso." });

});

};

* src\routes\chamadosRoutes.js :

const express = require('express');

const router = express.Router();

const chamadosController = require('../controllers/chamadosController');

router.post('/', chamadosController.criarChamado);

router.get('/atrasados', chamadosController.listarChamadosAtrasados);

router.get('/', chamadosController.listarChamado);

router.get('/:id/pdf', chamadosController.gerarOS);

router.get('/:id', chamadosController.listarChamadoPorTablet);

router.get('/id/:id', chamadosController.buscarChamadoPorIdChamado);

router.put('/:id', chamadosController.atualizarChamado);

router.delete('/:id', chamadosController.deletarChamado);

router.patch('/:id/fechar', chamadosController.atualizarStatus);

//router.get('/relatorio', chamadosController.gerarRelatorio);

console.log('Chamados routes loaded!');

module.exports = router;

* src\routes\empresasRoutes.js :

const express = require("express");

const router = express.Router();

const empresasController = require("../controllers/empresasController");

router.post("/", empresasController.criarEmpresa);

router.get("/", empresasController.listarEmpresa);

router.delete("/:id", empresasController.deletarEmpresa);

module.exports = router;

* src\routes\regRoutes.js :

const express = require("express");

const router = express.Router();

const regionaisController = require("../controllers/regionaisController");

router.post("/", regionaisController.criarRegional);

router.get("/", regionaisController.listarRegionais);

router.put("/:id", regionaisController.editarRegional);

router.delete("/:id", regionaisController.deletarRegional);

module.exports = router;

* src\routes\tabletsRoutes.js :

const express = require("express");

const router = express.Router();

const tabletsController = require("../controllers/tabletsController");

router.post("/", tabletsController.criarTablet);

router.get("/", tabletsController.listarTablets);

router.get("/busca", tabletsController.buscarTablet);

router.put("/:id", tabletsController.editarTablet);

router.delete("/:id", tabletsController.deletarTablet);

module.exports = router;

* src\routes\unidadesRoutes.js :

const express = require("express");

const router = express.Router();

const unidadesController = require("../controllers/unidadesController");

// router.post("/", unidadesController.cadastrarUnidade);

router.get("/", unidadesController.listarUnidades);

//router.put("/:id", unidadesController.editarUnidade);

// router.delete("/:id", unidadesController.deletarUnidade);

console.log('Unidades routes loaded!');

module.exports = router;

* src\routes\usersRoutes.js :

const express = require("express");

const router = express.Router();

const usuariosController = require("../controllers/usuariosController");

router.post("/", usuariosController.criarUsuario);

router.get("/", usuariosController.listarUsuarios);

router.put("/:id", usuariosController.editarUsuario);

router.delete("/:id", usuariosController.deletarUsuario);

console.log('Usuarios routes loaded!');

module.exports = router;

* src\index.js :

require('dotenv').config();

const express = require('express');

const cors = require('cors');

const tabletsRoutes = require('./routes/tabletsRoutes.js');

const chamadosRoutes = require('./routes/chamadosRoutes.js');

const empresasRoutes = require("./routes/empresasRoutes");

const app = express();

app.use(cors({

origin: 'http://localhost:3000', //frontend url

credentials: true,

}));

app.use(express.json());

//rotas

app.use('/tablets', tabletsRoutes);

app.use('/chamados', chamadosRoutes);

app.use("/empresas", empresasRoutes);

const PORT = process.env.PORT || 3001;

app.listen(PORT, () => {

console.log(`Servidor rodando na porta ${PORT}`);

});