const express = require('express');

const fs = require('fs');

const axios = require('axios');

const app = express();

const port = 3001;

app.use(express.json());

**//Função para ler os dados de 'db.funcionarios.json'**

function fetchDataFuncionarios() {

try {

const rawData = fs.readFileSync('db.funcionarios.json', 'utf8');

return JSON.parse(rawData);

} catch (err) {

console.error('Erro durante a leitura ou na analise dos dados dos funcionarios ', err);

return { funcionarios: [] };

}

}

**// Função para ler os dados de 'db.empresas.json'**

function fetchDataEmpresas() {

try {

const rawData = fs.readFileSync('db.empresas.json', 'utf8');

return JSON.parse(rawData);

} catch (err) {

console.error('Erro durante a leitura ou na analise dos dados das empresas ', err);

return { empresas: [] };

}

}

**// Função para salvar os dados de 'db.funcionarios.json'**

function saveDataFuncionarios(data) {

const jsonData = JSON.stringify(data, null, 2);

fs.writeFileSync('db.funcionarios.json', jsonData, 'utf8');

}

**// Função para salvar os dados de 'db.empresas.json'**

function saveDataEmpresas(data) {

const jsonData = JSON.stringify(data, null, 2);

fs.writeFileSync('db.empresas.json', jsonData, 'utf8');

}

**// Rota para obter uma lista com todos os funcionários**

app.get('/api/funcionarios', (req, res) => {

const funcionarios = fetchDataFuncionarios().funcionarios;

res.json(funcionarios);

});

**\*\*SEM PARÂMETROS NEM CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/funcionarios**

**// Rota para listar um os funcionarios desempregados**

app.get('/api/funcionarios/desempregados', (req, res) => {

const funcionarios = fetchDataFuncionarios().funcionarios;

const desempregados = funcionarios.filter(funcionario => funcionario.idEmpresa === 0);

res.json(desempregados);

});

**\*\*SEM PARÂMETROS NEM CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/funcionarios/desempregados**

**// Rota para obter um funcionario especifico atraves do ID**

app.get('/api/funcionarios/:idfuncionario', (req, res) => {

const dados = fetchDataFuncionarios().funcionarios;

const id = req.params.idfuncionario;

const funcionario = dados.find(item => item.id === parseInt(id));

if (funcionario) {

res.json(funcionario);

} else {

res.status(404).send('Funcionario não encontrado');

}

});

**\*\*REQUISICAO POR PARAMETROS**

**\*\*ENDPOINT localhost:3000/api/funcionarios/:idfuncionario**

**// Rota para obter informações do endereço do funcionário através do ID e de uma consulta externa no VIACEP**

app.get('/api/funcionarios/:idfuncionario/endereco', async (req, res) => {

try {

const dados = fetchDataFuncionarios().funcionarios;

const id = req.params.idfuncionario;

const funcionario = dados.find(item => item.id === parseInt(id));

if (!funcionario || !funcionario.cep) {

res.status(404).send('Funcionario não encontrado ou não possui CEP.');

return;

}

// Faz uma requisicao ao viacep.com usando o cep do funcionario

const cep = funcionario.cep;

const viacepResponse = await axios.get(`http://viacep.com.br/ws/${cep}/json/`);

res.json(viacepResponse.data);

} catch (error) {

console.error(error);

res.status(500).send('Erro ao buscar o endereço do funcionario.');

}

});

**\*\*REQUISICAO POR PARAMETROS**

**\*\*ENDPOINT localhost:3000/api/funcionarios/:idfuncionario/endereco**

**// Rota para adicionar um novo funcionário**

app.post('/api/funcionarios', express.json(), (req, res) => {

const dados = fetchDataFuncionarios();

const novoDado = req.body;

const funcionarios = dados.funcionarios;

novoDado.id = funcionarios.length > 0 ? Math.max(...funcionarios.map(f => f.id)) + 1 : 1;

funcionarios.push(novoDado);

saveDataFuncionarios(dados);

res.json(novoDado);

});

**\*\*REQUISIÇÃO POR CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/funcionarios**

**{**

**“nome” : “nome”,**

**“sobrenome” : “sobrenome”,**

**“cpf” : “cpf”,**

**“cargo” : “cargo”,**

**“empresa” : “empresa”,**

**“cep” : 777777777**

**“idempresa” : 7**

**}**

**// Atualizar funcionarios atraves de seu ID**

app.put('/api/funcionarios/:id', express.json(), (req, res) => {

const dados = fetchDataFuncionarios();

const id = req.params.id;

const funcionarios = dados.funcionarios;

const index = funcionarios.findIndex(item => item.id === parseInt(id));

if (index !== -1) {

funcionarios[index] = req.body;

saveDataFuncionarios(dados);

res.json(funcionarios[index]);

} else {

res.status(404).send('Funcionario não encontrado');

}

});

**\*\*REQUISIÇÃO POR CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/funcionarios**

**{**

**“nome” : “nome”,**

**“sobrenome” : “sobrenome”,**

**“cpf” : “cpf”,**

**“cargo” : “cargo”,**

**“empresa” : “empresa”,**

**“cep” : 777777777,**

**“idempresa” : 7**

**}**

**//############### EMPRESAS ###################**

**//obter empresas lista de empresas**

app.get('/api/empresas', (req, res) => {

const empresas = fetchDataEmpresas().empresas;

res.json(empresas);

});

**\*\*SEM PARÂMETROS NEM CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/empresas**

**//Obter uma unica empresa**

app.get('/api/empresas/:idempresa', (req, res) => {

const dados = fetchDataEmpresas().empresas;

const id = req.params.idempresa;

const empresa = dados.find(item => item.id === parseInt(id));

if (empresa) {

res.json(empresa);

} else {

res.status(404).send('Empresa não encontrada');

}

});

**\*\*REQUISICAO POR PARAMETROS**

**\*\*ENDPOINT localhost:3000/ap/empresas/:idempresa**

**//obter funcionários vinculados a empresa**

app.get('/api/empresas/:idEmpresa/funcionarios', (req, res) => {

const idEmpresa = req.params.idEmpresa;

const funcionarios = fetchDataFuncionarios().funcionarios;

// filtrar funcionarios por idEmpresa

const funcionariosDaEmpresa = funcionarios.filter(funcionario => funcionario.idEmpresa == idEmpresa);

if (funcionariosDaEmpresa.length > 0) {

res.json(funcionariosDaEmpresa);

} else {

res.status(404).send('Nenhum funcionário encontrado para esta empresa.');

}

});

**\*\*REQUISICAO POR PARAMETROS**

**\*\*ENDPOINT localhost:3000/ap/empresas/:idempresa/funcionarios**

**// CNPJ JA**

const apiKey = 'e0a9cc03-3b9a-47f8-af1c-886b472dd1bd-b7d6e4a0-cb4f-41a1-a51d-14c277d58ace';

**// Rota para obter a informacao sobre uma empresa através do ID**

app.get('/api/empresas/:idempresa/informacoes', async (req, res) => {

try {

const dados = fetchDataEmpresas().empresas;

const id = req.params.idempresa;

const empresa = dados.find(item => item.id === parseInt(id));

if (!empresa || !empresa.cnpj) {

res.status(404).send('Empresa não encontrada ou não possui CNPJ.');

return;

}

// Fazer uma requisição ao CNPJA usando o CNPJ da empresa

const cnpj = empresa.cnpj;

const config = {

method: 'get',

url: `https://api.cnpja.com/office/${cnpj}`,

headers: {

'Authorization': apiKey

}

};

const cnpjResponse = await axios(config);

res.json(cnpjResponse.data);

} catch (error) {

console.error(error);

res.status(500).send('Erro ao buscar informações da empresa.');

}

});

**\*\*REQUISICAO POR PARAMETROS**

**\*\*ENDPOINT localhost:3000/ap/empresas/:idempresa/informacoes**

**//##########CONTABILIZADOR DE PASSAGENS ##################**

app.post('/api/calculo', async (req, res) => {

try {

// Checa se idempresa e idfuncionario estao no corpo de requisicao

if (!req.body.idempresa || !req.body.idfuncionario) {

res.status(400).send('idempresa and idfuncionario are required in the request body.');

return;

}

const dados = fetchDataEmpresas().empresas;

const idEmpresa = req.body.idempresa;

const idFuncionario = req.body.idfuncionario;

const empresa = dados.find(item => item.id === parseInt(idEmpresa));

if (!empresa || !empresa.cnpj) {

res.status(404).send('Empresa não encontrada ou não possui CNPJ.');

return;

}

// Retrieve the CEP of the funcionario from your database

const funcionarioDatabase = fetchDataFuncionarios();

const funcionario = funcionarioDatabase.funcionarios.find(item => item.id === parseInt(idFuncionario));

if (!funcionario || !funcionario.cep) {

res.status(404).send('Funcionário não encontrado ou não possui CEP.');

return;

}

// Fazer uma requisicao ao CNPJA usando o CNPJ da empresa

const cnpj = empresa.cnpj;

const config = {

method: 'get',

url: `https://api.cnpja.com/office/${cnpj}`,

headers: {

'Authorization': apiKey

}

};

const cnpjResponse = await axios(config);

// Extrair a cidade da empresa

const cidadeEmpresa = cnpjResponse.data.address.city;

// Pega o CEP do funcionario e usa para fazer uma solicitação ao ViaCEP

const cepFuncionario = funcionario.cep;

const viaCepResponse = await axios.get(`http://viacep.com.br/ws/${cepFuncionario}/json/`);

const cidadeFuncionario = viaCepResponse.data.localidade;

// Checa se as cidades sao a mesma

const isSameCity = cidadeEmpresa === cidadeFuncionario;

// Define uma mensagem baseado nas cidades da enpresa e do funcionario

let message = 'As cidades são diferentes.';

if (isSameCity) {

message = 'As cidades são iguais.';

}

res.json({

cidadeDaEmpresa: cidadeEmpresa,

cidadeDoFuncionario: cidadeFuncionario,

message: message

});

} catch (error) {

console.error(error);

res.status(500).send('Erro ao buscar informações da empresa ou funcionário.');

}

});

// ligar o servidor

app.listen(port, () => {

console.log(`Servidor rodando em http://localhost:${port}`);

});

**\*\*REQUISIÇÃO POR CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/calculo**

**{**

**“idfuncionario” : 1,**

**“Idempresa” : 1**

**}**

**Adicionar uma nova empresa**

app.post('/api/empresas', express.json(), (req, res) => {

try {

const dados = fetchDataEmpresas();

const novaEmpresa = req.body;

const empresas = dados.empresas;

// Gerar novo ID para a empresa

novaEmpresa.id = empresas.length > 0 ? Math.max(...empresas.map(e => e.id)) + 1 : 1;

empresas.push(novaEmpresa);

saveDataEmpresas(dados);

res.json(novaEmpresa);

} catch (error) {

console.error(error);

res.status(500).send('Erro ao adicionar a nova empresa.');

}

});

**\*\*REQUISIÇÃO POR CORPO DE REQUISIÇÃO**

**\*\*ENDPOINT localhost:3000/api/funcionarios**

**{**

**"nome" : "BRAZ CUBAS",**

**"cnpj" : "52556412000106"**

**}**

**EMPRESAS**

**1 - Braz Cubas : Mogi das Cruzes, SP**

**2 - Suzano S.A : Salvador, BA**

**3 - SANOFI MEDLEY : Suzano, SP**

**4 - BAUMINAS QUIMICA : Suzano, SP**

**5 - DENVER INDUSTRIA E COMERCIO : Suzano, SP**

**6 - CONEXAO WEB TELECOM : Suzano, SP**

**7 - AUTOMOB S.A. : Mogi das Cruzes, SP**

**8 - VISIVEL LIMPEZA AMBIENTAL LTDA : Mogi das Cruzes, SP**

**9 - PKV CONSULTORIA COMERCIO E PARTICIPACOES LTDA : Poa, SP**

**10 - TALARES EMPREENDIMENTOS, PARTICIPACOES E SERVICOS LTDA : POA, SP**

**11 - EDMA - EMPRESA DE DESENVOLVIMENTO E MECANIZACAO AGROPECUARIA , POA, SP**

**12 - MAHLE BEHR GERENCIAMENTO TERMICO BRASIL LTDA : Aruja, SP**

**FUNCIONARIOS**

**1 : Suzano, sp**

**2 : Mogi das Cruzes, SP**

**3 : Mogi das Cruzes, SP**

**4: Mogi das Cruzes, SP**

**5: Arujá, SP**

**6: Arujá, SP**

**7: Poa, SP**

**8 : Salvador, Bahia**

**9 : Salvador, Bahia**

**10 : Suzano, SP**

**11 : Suzano, SP**

**12 : Suzano, SP**