```
const ConsultaApiVital = require('../models/ConsultaAPI/ConsultaApiVital');
const ConsultaApiGTA = require('../models/ConsultaAPI/ConsultaApiGTA');
const CoberturaFornecedor = require('../models/Cobertura_Fornecedor');

class PesquisaAPIController {
    async index(req, res) {
        try {
            // Captura os dados do formulário
            const sigla = req.body.sigla || '';
            const logo = req.body.logo || '';
            const url = req.body.url || '';
            const login = req.body.login || '';
            const senha = req.body.senha || '';
            const fantasia = req.body.fantasia || '';

            // Obtém as coberturas do fornecedor
            const coberturas_fornecedor = await CoberturaFornecedor.Lista();

            let dadosApi = null;

            if (sigla === 'VIT') {

                let resultado = [];

                // Chamando API Vital
                dadosApi = await ConsultaApiVital.consultar(req, url, login, senha);

                // Processando retorno da API Vital e filtrando as coberturas
                const planosFiltrados = dadosApi.planos.map(plano => {

                    // Concatenando seguros e benefícios
                    const coberturasConcatenadas = plano.seguros.concat(plano.beneficios);

                    // Filtra apenas fornecedores da VIT na ordem de A065_ordenacao
                    const fornecedoresFiltrados = coberturas_fornecedor
                        .filter(fornecedor => fornecedor.A060_sigla_api === 'VIT')
                        .sort((a, b) => a.A065_ordenacao - b.A065_ordenacao);

                    // Cria um array para armazenar as coberturas finais
                    const coberturasFinalizadas = [];

                    // Itera sobre fornecedores filtrados
                    for (const fornecedor of fornecedoresFiltrados) {
                        // Encontra coberturas que correspondem ao A065_depara do
fornecedor
                        const coberturasCorrespondentes =
coberturasConcatenadas.filter(cobertura =>
                            cobertura.descricao_integracao === fornecedor.A065_depara
                        );

                        let coberturaFinal;

                        if (coberturasCorrespondentes.length > 0) {
                            // Se houver mais de uma cobertura, usa a primeira
                            //coberturaFinal = coberturasCorrespondentes[0];
                            coberturaFinal = {
                                ...coberturasCorrespondentes[0], // Inclui as propriedades
da cobertura original
                                ordenacao: fornecedor.A065_ordenacao // Adiciona a
ordenacao do fornecedor
```

```
                            };
                        } else {
                            // Se não encontrar nenhuma cobertura, cria uma cobertura
padrão
                            coberturaFinal = {
                                descricao: "n/a",
                                descricao_integracao: "n/a",
                                valor: "--",
                                ordenacao: fornecedor.A065_ordenacao
                            };
                        }

                        // Adiciona a cobertura finalizada ao array de resultados
                        coberturasFinalizadas.push(coberturaFinal);
                    }

                    return {
                        fantasia: fantasia,
                        logo: logo,
                        sigla: sigla,
                        codigo: plano.cod_plano,
                        nome: plano.nome,
                        valor: (parseFloat(plano.valor).toFixed(2)).replace(".", ","),
                        parcelas: plano.parcelas,
                        valor_parcela:
(parseFloat(plano.valor_parcela).toFixed(2)).replace(".", ","),
                        coberturas: coberturasFinalizadas
                    };
                });

                resultado = { planos: planosFiltrados };
                res.json(resultado);

            } else if (sigla === 'GTA') {

                var idades = req.body.idadePassageiro;
                idades = String(idades).split(',').map(Number);
                var idades_maior_89 = idades.filter(idade => idade >= 90).length;
                let resultado = [];
                // NA GTA, só vende para PAX com até 89 anos, se tiver algum pax com mais
de 89, não orçar.
                if(idades_maior_89 === 0) {

                    // NA GTA, só vende individualmente para pax com idades entre 86 e 89
                    // Podemos ter N PAX com idades entre 86 e 89, mas nesse caso não
podemos ter nenhum pax com idade fora deste range.
                    var idades_entre_85_89 = idades.filter(idade => idade >= 86 && idade <=
89).length;
                    if ((idades_entre_85_89 === 0) || (idades_entre_85_89 ===
idades.length)) {

                        // Chamando API GTA
                        dadosApi = await ConsultaApiGTA.consultar(req, url, login, senha);
                        const planosFiltrados = Object.values(dadosApi).map(plano => {

                            // Concatenando seguros e benefícios
                            const coberturasConcatenadas =
plano.seguros.concat(plano.beneficios);

                            // Filtra apenas fornecedores da GTA na ordem de
```

```
A065_ordenacao
                            const fornecedoresFiltrados = coberturas_fornecedor
                                .filter(fornecedor => fornecedor.A060_sigla_api === 'GTA')
                                .sort((a, b) => a.A065_ordenacao - b.A065_ordenacao);

                            //console.log("fornecedoresFiltrados", fornecedoresFiltrados)

                            // Cria um array para armazenar as coberturas finais
                            const coberturasFinalizadas = [];

                            // Itera sobre fornecedores filtrados
                            for (const fornecedor of fornecedoresFiltrados) {
                                // Encontra coberturas que correspondem ao A065_depara do
fornecedor
                                const coberturasCorrespondentes =
coberturasConcatenadas.filter(cobertura =>
                                    cobertura.descricao_integracao ===
fornecedor.A065_depara
                                );

                                //console.log("coberturasCorrespondentes",
coberturasCorrespondentes)

                                let coberturaFinal;

                                if (coberturasCorrespondentes.length > 0) {
                                    // Se houver mais de uma cobertura, usa a primeira
                                    //coberturaFinal = coberturasCorrespondentes[0]
                                    coberturaFinal = {
                                        ...coberturasCorrespondentes[0], // Inclui as
propriedades da cobertura original
                                        ordenacao: fornecedor.A065_ordenacao // Adiciona a
ordenacao do fornecedor
                                    };
                                } else {
                                    // Se não encontrar nenhuma cobertura, cria uma
cobertura padrão
                                    coberturaFinal = {
                                        descricao: "n/a",
                                        descricao_integracao: "n/a",
                                        valor: "--",
                                        ordenacao: fornecedor.A065_ordenacao
                                    };
                                }

                                // Adiciona a cobertura finalizada ao array de resultados
                                coberturasFinalizadas.push(coberturaFinal);
                            }

                            return {
                                fantasia: fantasia,
                                logo: logo,
                                sigla: sigla,
                                codigo: parseFloat(plano.planoId),
                                nome: plano.planoNome,
                                valor: (parseFloat(plano.valor).toFixed(2)).replace(".",
",",
                                parcelas: plano.parcelasFornecedor,
                                valor_parcela:
(parseFloat(plano.valor_parcelas).toFixed(2)).replace(".", ","),
```

```
                                coberturas: coberturasFinalizadas
                            };
                        });

                        resultado = { planos: planosFiltrados };
                    }
                }

                res.json(resultado);
            }

        } catch (error) {
            console.error('Erro no controlador:', error);
            res.status(500).send('Erro interno do servidor');
        }
    }
}

module.exports = new PesquisaAPIController();
```