```
var CarteiraPrevidencia = SuperWidget.extend({
    tableCarteiraPrevidencia: null,
    instanceIdCarteiraPrevidencia: null,

    init: function() {
        this.instanceIdCarteiraPrevidencia = this.instanceId;
        FLUIGC.calendar('.dateAtual');
        this.loadClientes();
        this.loadSeguradoras();
    },

    bindings: {
        local: {
            'consultaDados': ['click_consultaDados']
        }
    },

    consultaDados: function() {
        var self = this;
        var instanceId = this.instanceIdCarteiraPrevidencia;
        var cpfCnpj = $("#cpfCnpj_" + instanceId).val() ||
"";
        var cd_seguradora = $("#cd_seguradora_" +
instanceId).val() || "";
        var dataInicial = $("#periodoInicial_" +
instanceId).val();
        var dataFinal = $("#periodoFinal_" +
instanceId).val();

        if ((dataInicial != "" && dataFinal == "") ||
(dataInicial == "" && dataFinal != "")) {
            FLUIGC.toast({ message: "Preencha ambos os
períodos ou nenhum.", type: "warning" });
            return false;
        }

        Promise.all([
            this.recuperaDadosDatasetSaldo(cpfCnpj,
cd_seguradora, dataInicial, dataFinal),
            this.recuperaDadosDatasetProposta(cpfCnpj,
cd_seguradora, dataInicial, dataFinal)
        ]).then(function(resultados) {
            var dadosSaldo = resultados[0] || [];
            var dadosProposta = resultados[1] || [];

            if (dadosProposta.length === 0 &&
dadosSaldo.length === 0) {
```

```
                FLUIGC.toast({ message: "Nenhum registro
encontrado.", type: "warning" });
                return;
            }

            var processados =
self.processarDados(dadosProposta, dadosSaldo);
            self.montaTabela(processados);

            $(".tableCarteiraPrevidencia").removeClass("fs-
display-none");
            if (self.tableCarteiraPrevidencia) {
                setTimeout(function() {

self.tableCarteiraPrevidencia.columns.adjust().draw();
                }, 200);
            }

        }).catch function(err) {
            console.error(err);
            FLUIGC.toast({ message: "Erro ao recuperar
dados.", type: "danger" });
        });
    },

    processarDados: function(propostas, saldos) {
        var self = this;
        var saldosTratados = _.map(saldos, function(s) {
            return {
                nomeCompleto: s.nomeCompleto || "",
                txt_seguradora: s.txt_seguradora || "",
                txt_propostaSeguradora:
s.txt_propostaSeguradora || "",
                txt_seguradoraSaldo: s.txt_seguradoraSaldo ||
"0,00",
                txt_seguradoraRentabilidade:
s.txt_seguradoraRentabilidade || "0,00",
                txt_seguradoraResgate:
s.txt_seguradoraResgate || "0,00",
                txt_investMensal: s.txt_investMensal ||
"0,00",
                txt_investEsporadico: s.txt_investEsporadico
|| "0,00",
                dataCadastro: s.dataCadastro
            };
        });
```

```
        var saldosRecentes = saldosTratados;

        return _.map(propostas, function(p) {
            var percFundo =
self.parseMoeda(p.percentualFundo) || 0;
            var valorAporte = self.parseMoeda(p.txt_aporte)
|| 0;
            var valorMensal =
self.parseMoeda(p.txt_mensalidade) || 0;

            var saldoRelacionado = _.find(saldosRecentes,
function(s) {
                return s.txt_propostaSeguradora ===
p.txt_NewProposta || s.txt_propostaSeguradora ===
p.txt_propostaSeguradora;
            });

            var resgateFinal = saldoRelacionado ?
self.parseMoeda(saldoRelacionado.txt_seguradoraResgate) : 0;

            var obj = {
                nomeCompleto: p.nomeCompleto,
                txt_seguradora: p.txt_seguradora,
                txt_propostaSeguradora: p.txt_NewProposta ||
p.txt_propostaSeguradora,
                txt_mensalidade: ((valorMensal / 100) *
percFundo).toLocaleString('pt-BR', { minimumFractionDigits: 2
}),
                txt_aporte: ((valorAporte / 100) *
percFundo).toLocaleString('pt-BR', { minimumFractionDigits: 2
}),
                txt_investMensal: saldoRelacionado ?
saldoRelacionado.txt_investMensal : "0,00",
                txt_investEsporadico: saldoRelacionado ?
saldoRelacionado.txt_investEsporadico : "0,00",
                txt_seguradoraSaldo: saldoRelacionado ?
saldoRelacionado.txt_seguradoraSaldo : "0,00",
                txt_seguradoraRentabilidade: saldoRelacionado
? saldoRelacionado.txt_seguradoraRentabilidade : "0,00",
                txt_seguradoraResgate: ((resgateFinal / 100)
* percFundo).toLocaleString('pt-BR', { minimumFractionDigits:
2 }),
                txt_VgblPgblNew: p.txt_VgblPgblNew || "",
                nomeFundo: p.nomeFundo || "",
                percentualFundo: p.percentualFundo + "%"
            };
```

```
            var sVal =
self.parseMoeda(obj.txt_seguradoraSaldo);
            var rVal =
self.parseMoeda(obj.txt_seguradoraRentabilidade);
            //obj.totalRentabilidadeSaldo = (sVal +
rVal).toLocaleString('pt-BR', { minimumFractionDigits: 2 });

            return obj;
        });
    },

    parseMoeda: function(valor) {
        if (!valor) return 0;
        var limpo = valor.toString().replace(/\./g,
"").replace(",", ".");
        var num = parseFloat(limpo);
        return isNaN(num) ? 0 : num;
    },

    montaTabela: function(dados) {
        if (this.tableCarteiraPrevidencia) {
            this.tableCarteiraPrevidencia.destroy();
            $('#tableCarteiraPrevidencia').empty();
        }

        this.tableCarteiraPrevidencia =
$('#tableCarteiraPrevidencia').DataTable({
            data: dados,
            columns: [
                { data: "nomeCompleto", title: "CLIENTE" },
                { data: "txt_seguradora", title: "SEGURADORA"
},
                { data: "txt_propostaSeguradora", title:
"PROPOSTA" },
                { data: "txt_mensalidade", title:
"MENSALIDADE FIXA" },
                { data: "txt_aporte", title: "APORTE FIXO" },
                { data: "txt_investMensal", title: "INVEST.
MENSAL" },
                { data: "txt_investEsporadico", title:
"APORTE ESPORÁDICO" },
                { data: "txt_seguradoraRentabilidade", title:
"RENTABILIDADE" },
                // { data: "totalRentabilidadeSaldo", title:
"SALDO TOTAL" },
                { data: "txt_seguradoraResgate", title:
"RESGATE" },
```

```
                { data: "txt_seguradoraSaldo", title: "SALDO
ACUMULADO" },
                { data: "txt_VgblPgblNew", title: "VGBL/PGBL"
},
                { data: "nomeFundo", title: "FUNDO" },
                { data: "percentualFundo", title:
"PARTICIPAÇÃO" }
            ],
            scrollX: true,
            responsive: false,
            autoWidth: false,
            // Tradução direta para evitar erro 404 de
carregamento de arquivo
              // externo
            language: {
                "sEmptyTable": "Nenhum registro encontrado",
                "sInfo": "Exibindo de _START_ até _END_ de
_TOTAL_ registros",
                "sInfoEmpty": "Exibindo 0 até 0 de 0
registros",
                "sInfoFiltered": "(Filtrados de _MAX_
registros)",
                "sInfoPostFix": "",
                "sInfoThousands": ".",
                "sLengthMenu": "_MENU_ resultados por
página",
                "sLoadingRecords": "Carregando...",
                "sProcessing": "Processando...",
                "sZeroRecords": "Nenhum registro encontrado",
                "sSearch": "Pesquisar",
                "oPaginate": {
                    "sNext": "Próximo",
                    "sPrevious": "Anterior",
                    "sFirst": "Primeiro",
                    "sLast": "Último"
                }
            },
            dom: 'Bfrtip',
            buttons: ['excel', 'pdf']
        });
    },

    filtrarDatasMaisRecentes: function(dados) {
        var agrupados = _.groupBy(dados, function(obj) {
            return obj.nomeCompleto + '|' +
obj.txt_propostaSeguradora;
        });
```

```
        return _.map(agrupados, function(grupo) {
            return _.max(grupo, function(obj) {
                return obj.dataCadastro;
            });
        });
    },

    loadClientes: function() {
        var self = this;
        FLUIGC.autocomplete("#nm_cliente_" +
this.instanceIdCarteiraPrevidencia, {
            displayKey: 'A1_NOME',
            type: 'tagAutocomplete',
            source: function(q, cb) {
                var c =
[DatasetFactory.createConstraint('A1_NOME', q.toLowerCase(),
q.toLowerCase(), ConstraintType.MUST)];

DatasetFactory.getDataset("ds_listaClientesProtheus", null,
c, null, {
                    success: function(res) { cb(res.values);
}
                });
            }
        }).on('fluig.autocomplete.selected', function(e, i) {
            $("#cpfCnpj_" +
self.instanceIdCarteiraPrevidencia).val(i.item.A1_CGC);
        });
    },

    loadSeguradoras: function() {
        var self = this;
        FLUIGC.autocomplete("#nm_seguradora_" +
this.instanceIdCarteiraPrevidencia, {
            displayKey: 'A1_NOME',
            type: 'tagAutocomplete',
            source: function(q, cb) {
                var c =
[DatasetFactory.createConstraint('A1_NOME', q.toLowerCase(),
q.toLowerCase(), ConstraintType.MUST)];

DatasetFactory.getDataset("ds_seguradorasProtheus", null, c,
null, {
                    success: function(res) { cb(res.values);
}
                });
```

```
            }
        }).on('fluig.autocomplete.selected', function(e, i) {
            $("#cd_seguradora_" +
self.instanceIdCarteiraPrevidencia).val(i.item.A1_COD);
        });
    },

    compararDatas: function() {
        var id = this.instanceIdCarteiraPrevidencia;
        var p1 = $("#periodoInicial_" + id).val().split("/");
        var p2 = $("#periodoFinal_" + id).val().split("/");
        return new Date(p1[2], p1[1] - 1, p1[0]) <= new
Date(p2[2], p2[1] - 1, p2[0]);
    },

    recuperaDadosDatasetSaldo: function(cpf, cd, dI, dF) {
        return new Promise(function(resolve) {
            var c = [];
            if (cpf)
c.push(DatasetFactory.createConstraint('cpfCnpj', cpf, cpf,
ConstraintType.MUST));
            if (cd)
c.push(DatasetFactory.createConstraint('txt_cdSeguradora',
cd, cd, ConstraintType.MUST));
            if (dI && dF)
c.push(DatasetFactory.createConstraint('dataCadastro',
formataDataParaYYYYMMDD(dI), formataDataParaYYYYMMDD(dF),
ConstraintType.MUST));

DatasetFactory.getDataset("ds_registrosSaldoSeguradoraCliente
", null, c, null, {
                success: function(d) { resolve(d &&
d.values.length > 0 ? d.values : null); },
                error: function() { resolve(null); }
            });
        });
    },

    recuperaDadosDatasetProposta: function(cpf, cd, dI, dF) {
        return new Promise(function(resolve) {
            var c = [];
            if (cpf)
c.push(DatasetFactory.createConstraint('cpfCnpj', cpf, cpf,
ConstraintType.MUST));
            if (cd)
c.push(DatasetFactory.createConstraint('txt_cdSeguradora',
cd, cd, ConstraintType.MUST));
```

```
            if (dI && dF)
c.push(DatasetFactory.createConstraint('dataCadastro',
formataDataParaYYYYMMDD(dI), formataDataParaYYYYMMDD(dF),
ConstraintType.MUST));

DatasetFactory.getDataset("ds_registrosPropostaSeguradoraClie
nte", null, c, null, {
                success: function(d) { resolve(d &&
d.values.length > 0 ? d.values : null); },
                error: function() { resolve(null); }
            });
        });
    }
});

function formataDataParaYYYYMMDD(ds) {
    if (!ds) return "";
    var p = ds.split("/");
    return p[2] + p[1] + p[0];
}
```