```
var alfabeto = [ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
'j', 'k', 'l',
          'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
'w', 'x', 'y', 'z' ];

function servicetask12(attempt, message) {

     var indicesSaldoComissao =
retornaIndices("txt_cdSeguradora");
     var indicesProposta =
retornaIndices("txt_propostaSeguradora");

     var tipoInclusao = hAPI.getCardValue("rdTipoInclusao");
     log.info("### servicetask12 (Future) - Tipo: " +
tipoInclusao);

     if (tipoInclusao == "proposta") {
          for ( var i in indicesProposta) {

     incluiAtualizaRegistroPropostaFluig(indicesProposta[i]);
          }
     } else if (tipoInclusao == "saldo") {
          for ( var i in indicesSaldoComissao) {

     incluiAtualizaRegistroFluig(indicesSaldoComissao[i],
"saldo");
          }
     } else if (tipoInclusao == "resgate") {
          for ( var i in indicesSaldoComissao) {

     incluiAtualizaRegistroFluig(indicesSaldoComissao[i],
"resgate");
          }
     } else if (tipoInclusao == "comissao") {
          var contador = 0;

          if (indicesProposta.length == 0) {
               log
                         .warn("### ATENCAO: Nenhum item de
comissao encontrado para integrar.");
          }

          for ( var i in indicesProposta) {
               // Verifica se já não foi integrado antes
               if (hAPI.getCardValue("retornoProtheus___" +
indicesProposta[i]) != "OK") {
```

```
                    incluiTituloProtheus(indicesProposta[i],
contador);
               }
               contador++;
          }
     }
}

function incluiTituloProtheus(indice, contador) {
     try {
          var clientService =
fluigAPI.getAuthorizeClientService();
          var codigoTitulo = preencherZerosEsquerda(
                    parseInt(getValue("WKNumProces")) + "" +
alfabeto[contador], 7);

          var valorStr =
hAPI.getCardValue("txt_seguradoraComissao___" + indice);
          var valorFloat = 0;
          if (valorStr) {
               valorFloat =
parseFloat(valorStr.replaceAll("\\.", "").replaceAll(
                         ",", "."));
          }

          // Recupera data de vencimento com segurança
          var dtVencimento =
hAPI.getCardValue("txt_seguradoraVencimento___"
                    + indice);

          var data = {
               companyId : '1',
               serviceCode : "rest-protheus",
               endpoint : '/TITREC',
               method : 'post',
               timeoutService : '100',
               headers : {
                    "Accept-Charset" : "UTF-8",
                    "Content-Type" : "application/json",
               },
               params : {
                    "EMPRESA" : "" +
hAPI.getCardValue("cdEmpresaProtheus"),
                    "FILIAL" : "" +
hAPI.getCardValue("cdFilialProtheus"),
                    "E1_PREFIXO" : "FUT",
                    "E1_NUM" : "" + codigoTitulo,
```

```
                    "E1_PARCELA" : "A",
                    "E1_TIPO" : "BOL",
                    "E1_NATUREZ" :
hAPI.getCardValue("txt_cdNatureza___" + indice),
                    "E1_CLIENTE" :
hAPI.getCardValue("a1cod"),
                    "E1_LOJA" : hAPI.getCardValue("a1loja"),
                    "E1_EMISSAO" :
hAPI.getCardValue("creationDate"), // Usa data de criação do
formulário
                    "E1_VENCTO" :
dataFormatada(dtVencimento), // Usa função protegida
                    "E1_VALOR" : valorFloat,
                    "E1_HIST" :
hAPI.getCardValue("txtHistoricoComissao___"
                            + indice),
                    "E1_MOEDA" : 1,
                    "E1_TXMOEDA" : 1.0,
                    "E1_XNOMSOL" : "",
                    "E1_XEMAILS" : "",
                    "E1_XNOME" : "",
                    "E1_XCPF" : new java.math.BigInteger(hAPI
                            .getCardValue("zoomcpfCnpj")),
                    "E1_XTPOPER" : "",
                    "E1_XDTFCH" : "",
                    "E1_XMOEDA" : "1",
                    "E1_XMOEDEX" : "",
                    "E1_XCONTRA" : "",
                    "E1_XTXCLI" : parseInt(0),
                    "E1_XTXSPOT" : parseInt(0),
                    "E1_XDESPBC" : parseInt(0),
                    "E1_XIOF" : parseInt(0),
                    "E1_XVLRTOT" : parseInt(0),
                    "E1_XSPREAD" : parseInt(0),
                    "E1_XVLRSPE" : parseInt(0),
                    "E1_XCOMIS" : parseInt(0),
                    "E1_XOBSERV" : "Mês de Competência: "
                            + hAPI

    .getCardValue("txt_seguradoraMesComp___"
                                    + indice),
                    "E1_XOPER" : ""
                            +
(hAPI.getCardValue("txt_cdSeguradora___" + indice) || "")
                }
        };
```

```
            log.info("JSON de integração " +
JSONUtil.toJSON(data));

            var retornoApi =
clientService.invoke(JSONUtil.toJSON(data));
            log.info("retornoApi.getResult() " +
retornoApi.getResult());

            var retornoApiGetResult = "" +
retornoApi.getResult();
            retornoApiGetResult = retornoApiGetResult.replace(
                        /[\u0000-\u001F\u007F-\u009F]/g, '')

            var retornojson = JSON.parse(retornoApiGetResult);
            var statusHttp = retornoApi.getHttpStatusResult();

            if (retornoApi.getResult() == null ||
retornoApi.getResult().isEmpty())
                  throw "Ocorreu uma falha no retorno da API!";

            if (statusHttp >= "400") {
                  throw retornoApi.getResult();
            }

            hAPI.setCardValue("retornoProtheus___" + indice,
retornojson.Mensagem);

            if (retornojson.Mensagem == "OK") {
                  hAPI.setCardValue("codigoTitulo___" + indice,
codigoTitulo);
            }

            if (retornojson.Mensagem != "OK") {
                  throw retornojson["Mensagem Detalhada"];
            }

      } catch (err) {
            log.error("Erro integration Protheus: " + err);
            throw err + " - Linha: " + (err.lineNumber ||
"N/A");
      }
}

function incluiAtualizaRegistroFluig(indice, rdTipoInclusao)
{

      var cpfCnpj = hAPI.getCardValue("zoomcpfCnpj");
```

```
    var zoomNomeCompleto =
hAPI.getCardValue("zoomNomeCompleto");
    var txt_cdSeguradora =
hAPI.getCardValue("txt_cdSeguradora___" + indice);
    var txt_seguradora =
hAPI.getCardValue("txt_seguradora___" + indice);
    var txt_seguradoraMesComp =
hAPI.getCardValue("txt_seguradoraMesComp___"
            + indice);

    var txt_seguradoraSaldo =
hAPI.getCardValue("txt_seguradoraSaldo___"
            + indice);
    var txt_seguradoraRentabilidade = hAPI
            .getCardValue("txt_seguradoraRentabilidade___"
+ indice);
    var txt_seguradoraResgate =
hAPI.getCardValue("txt_seguradoraResgate___"
            + indice);
    var txt_propostaSeguradora =
hAPI.getCardValue("txt_propostaSeguradora___"
            + indice);

    // Garante data
    var dataCadastro = retornaData();

    var txt_investEsporadico =
hAPI.getCardValue("txt_investEsporadico___"
            + indice);
    var txt_investMensal =
hAPI.getCardValue("txt_investMensal___" + indice);

    if (rdTipoInclusao == "saldo") {

        var retornoRegistroExistente =
consultaRegistroSaldoSeguradoraPorCliente(
                cpfCnpj, txt_cdSeguradora,
txt_seguradoraMesComp,
                txt_propostaSeguradora);
        log.info("## retornoRegistroExistente: " +
retornoRegistroExistente);
        var retornoIncluiAtualizaRegistro = "";

        if (retornoRegistroExistente != "") {
            retornoIncluiAtualizaRegistro =
atualizaRegistroFluig(
```

```
                        retornoRegistroExistente, cpfCnpj,
zoomNomeCompleto,
                        txt_cdSeguradora, txt_seguradora,
txt_seguradoraMesComp,
                        txt_seguradoraSaldo,
txt_seguradoraRentabilidade,
                        dataCadastro, rdTipoInclusao,
txt_propostaSeguradora,
                        txt_investEsporadico,
txt_investMensal);
            } else {
                retornoIncluiAtualizaRegistro =
incluiRegistroFluig(cpfCnpj,
                        zoomNomeCompleto, txt_cdSeguradora,
txt_seguradora,
                        txt_seguradoraMesComp,
txt_seguradoraSaldo,
                        txt_seguradoraRentabilidade,
dataCadastro, rdTipoInclusao,
                        txt_seguradoraResgate,
txt_propostaSeguradora,
                        txt_investEsporadico,
txt_investMensal);
            }

        } else {
            retornoIncluiAtualizaRegistro =
incluiRegistroFluig(cpfCnpj,
                    zoomNomeCompleto, txt_cdSeguradora,
txt_seguradora,
                    txt_seguradoraMesComp,
txt_seguradoraSaldo,
                    txt_seguradoraRentabilidade,
dataCadastro, rdTipoInclusao,
                    txt_seguradoraResgate,
txt_propostaSeguradora);
        }
    }

function incluiAtualizaRegistroPropostaFluig(indice) {

    var cpfCnpj = hAPI.getCardValue("zoomcpfCnpj");
    var zoomNomeCompleto =
hAPI.getCardValue("zoomNomeCompleto");
    var txt_cdSeguradora = hAPI
            .getCardValue("txt_cdSeguradoraProp___" +
indice);
```

```
    var txt_seguradora =
hAPI.getCardValue("txt_seguradoraProp___" + indice);
    var txt_NewProposta =
hAPI.getCardValue("txt_NewProposta___" + indice);

    var mensalidade = "";
    var aporte = "";

    if (hAPI.getCardValue("txt_mensAportNew___" + indice) ==
"aporte") {
        aporte =
hAPI.getCardValue("txt_inptMensalAportNew___" + indice);
    } else {
        mensalidade =
hAPI.getCardValue("txt_inptMensalAportNew___" + indice);
    }

    var txt_proReg = hAPI.getCardValue("txt_proReg___" +
indice);
    var txt_VgblPgblNew =
hAPI.getCardValue("txt_VgblPgblNew___" + indice);

    // Garante data
    var dataCadastro = retornaData();

    var jsonFundos = hAPI.getCardValue("jsonFundos___" +
indice);
    log.info("### jsonFundos:" + jsonFundos);

    var jsonFundosParse = [];
    try {
        jsonFundosParse = JSON.parse(jsonFundos);
    } catch (e) {
        log.warn("JSON Fundos vazio ou invalido: " + e);
    }

    var retornoRegistroExistente =
consultaRegistroPropostaSeguradoraPorCliente(
            cpfCnpj, txt_cdSeguradora, txt_NewProposta);
    log.info("## retornoRegistroExistente: " +
retornoRegistroExistente);

    if (jsonFundosParse.length > 0) {
        for (var i = 0; i < jsonFundosParse.length; i++) {
            var fundo = jsonFundosParse[i].fundo;
            var percentual =
jsonFundosParse[i].percentual;
```

```
                if (retornoRegistroExistente != "") {

    atualizaRegistroPropostaFluig(retornoRegistroExistente,
                                cpfCnpj, zoomNomeCompleto,
txt_cdSeguradora,
                                txt_seguradora,
txt_NewProposta, mensalidade, aporte,
                                dataCadastro, txt_proReg,
txt_VgblPgblNew, fundo,
                                percentual);
                } else {
                    incluiRegistroPropostaFluig(cpfCnpj,
zoomNomeCompleto,
                                txt_cdSeguradora,
txt_seguradora, txt_NewProposta,
                                mensalidade, aporte,
dataCadastro, txt_proReg,
                                txt_VgblPgblNew, fundo,
percentual);
                }
            }
        } else {
            log.info("Nenhum fundo para processar na proposta "
+ txt_NewProposta);
        }
}

function atualizaRegistroFluig(retornoRegistroExistente,
cpfCnpj, nomeCompleto,
        txt_cdSeguradora, txt_seguradora,
txt_seguradoraMesComp,
        txt_seguradoraSaldo, txt_seguradoraRentabilidade,
dataCadastro,
        rdTipoInclusao, txt_propostaSeguradora,
txt_investEsporadico,
        txt_investMensal) {
    log.info("## atualizaRegistroFluig ###");
    try {
        var config = recuperaDadosFixos();
        var serviceProvider = ServiceManager
                .getServiceInstance("ECMCardService");
        var wkServiceEngine = serviceProvider

    .instantiate("com.totvs.ECMCardServiceService");
        var service = wkServiceEngine.getCardServicePort();
        var cardFieldDtoArray = serviceProvider
```

```
    .instantiate("com.totvs.CardFieldDtoArray");

        // Helper para adicionar campos
        var addField = function(name, value) {
            var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
            dto.setField(name);
            dto.setValue(value || "");
            cardFieldDtoArray.getItem().add(dto);
        };

        addField("cpfCnpj", cpfCnpj);
        addField("nomeCompleto", nomeCompleto);
        addField("txt_cdSeguradora", txt_cdSeguradora);
        addField("txt_seguradora", txt_seguradora);
        addField("txt_seguradoraMesComp",
txt_seguradoraMesComp);
        addField("txt_seguradoraSaldo",
txt_seguradoraSaldo);
        addField("txt_seguradoraRentabilidade",
txt_seguradoraRentabilidade);
        addField("campoDescritor", nomeCompleto + " - " +
txt_seguradora
                + " - " + txt_propostaSeguradora + " - "
                + txt_seguradoraMesComp);
        addField("dataCadastro",
formataDataParaYYYYMMDD(dataCadastro));
        addField("rdTipoInclusao", rdTipoInclusao);
        addField("txt_propostaSeguradora",
txt_propostaSeguradora);
        addField("txt_investEsporadico",
txt_investEsporadico);
        addField("txt_investMensal", txt_investMensal);

        var result = service.updateCardData(1, "" +
config.usuarioFluig, ""
                + config.senhaFluig,
parseInt(retornoRegistroExistente),
                cardFieldDtoArray);

        var docId = result.getItem().get(0).documentId;
        if (docId > 0) {
            return docId;
        } else {
            log.error("Erro para salvar registro: "
```

```
                    +
result.getItem().get(0).webServiceMessage);
                return "nok";
            }
        } catch (e) {
            log.error("Erro excecao ao atualizar registro: " +
e.toString());
            return "nok";
        }
}

function incluiRegistroFluig(cpfCnpj, nomeCompleto,
txt_cdSeguradora,
            txt_seguradora, txt_seguradoraMesComp,
txt_seguradoraSaldo,
            txt_seguradoraRentabilidade, dataCadastro,
rdTipoInclusao,
            txt_seguradoraResgate, txt_propostaSeguradora,
txt_investEsporadico,
            txt_investMensal) {
    try {
            var config = recuperaDadosFixos();
            var serviceProvider = ServiceManager
                    .getServiceInstance("ECMCardService");
            var wkServiceEngine = serviceProvider

    .instantiate("com.totvs.ECMCardServiceService");
            var service = wkServiceEngine.getCardServicePort();
            var cardDtos =
serviceProvider.instantiate("com.totvs.CardDtoArray");
            var cardDto =
serviceProvider.instantiate("com.totvs.CardDto");

            cardDto.setParentDocumentId(parseInt(""
                    + recuperaIdFormPaiSaldoSeguradora()));
            cardDto.setColleagueId("" + config.usuarioFluig);
            cardDto.setInheritSecurity(true);

            // Helper para adicionar campos
            var addField = function(name, value) {
                var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
                dto.setField(name);
                dto.setValue(value || "");
                cardDto.getCardData().add(dto);
            };
```

```
            addField("cpfCnpj", cpfCnpj);
            addField("nomeCompleto", nomeCompleto);
            addField("txt_cdSeguradora", txt_cdSeguradora);
            addField("txt_seguradora", txt_seguradora);
            addField("txt_seguradoraMesComp",
txt_seguradoraMesComp);
            addField("campoDescritor", nomeCompleto + " - " +
txt_seguradora
                    + " - " + txt_propostaSeguradora + " - "
                    + txt_seguradoraMesComp);
            addField("dataCadastro",
formataDataParaYYYYMMDD(dataCadastro));
            addField("rdTipoInclusao", rdTipoInclusao);
            addField("txt_propostaSeguradora",
txt_propostaSeguradora);

            if (rdTipoInclusao == "saldo") {
                addField("txt_seguradoraSaldo",
txt_seguradoraSaldo);
                addField("txt_seguradoraRentabilidade",
txt_seguradoraRentabilidade);
                addField("txt_investEsporadico",
txt_investEsporadico);
                addField("txt_investMensal",
txt_investMensal);
            } else if (rdTipoInclusao == "resgate") {
                addField("txt_seguradoraResgate",
txt_seguradoraResgate);
            }

            cardDtos.getItem().add(0, cardDto);
            var result = service.create("1", "" +
config.usuarioFluig, ""
                    + config.senhaFluig, cardDtos);

            var docId = result.getItem().get(0).documentId;
            if (docId > 0) {
                return docId;
            } else {
                log.error("Erro para salvar registro fluig: "
                        +
result.getItem().get(0).webServiceMessage);
                return "nok";
            }
        } catch (e) {
            log.error("Erro excecao ao incluir registro fluig:
" + e.toString());
```

```
            return "nok";
        }
    }

    function
    atualizaRegistroPropostaFluig(retornoRegistroExistente,
    cpfCnpj,
            zoomNomeCompleto, txt_cdSeguradora, txt_seguradora,
    txt_NewProposta,
            mensalidade, aporte, dataCadastro, txt_proReg,
    txt_VgblPgblNew, fundo,
            percentual) {
        log.info("## atualizaRegistroPropostaFluig ###");
        try {
            var config = recuperaDadosFixos();
            var serviceProvider = ServiceManager
                    .getServiceInstance("ECMCardService");
            var wkServiceEngine = serviceProvider

        .instantiate("com.totvs.ECMCardServiceService");
            var service = wkServiceEngine.getCardServicePort();
            var cardFieldDtoArray = serviceProvider

        .instantiate("com.totvs.CardFieldDtoArray");

            var addField = function(name, value) {
                var dto =
    serviceProvider.instantiate("com.totvs.CardFieldDto");
                dto.setField(name);
                dto.setValue(value || "");
                cardFieldDtoArray.getItem().add(dto);
            };

            addField("cpfCnpj", cpfCnpj);
            addField("nomeCompleto", zoomNomeCompleto);
            addField("txt_cdSeguradora", txt_cdSeguradora);
            addField("txt_seguradora", txt_seguradora);
            addField("txt_propostaSeguradora",
    txt_NewProposta);
            addField("txt_mensalidade", mensalidade);
            addField("txt_aporte", aporte);
            addField("campoDescritor", zoomNomeCompleto + " - "
    + txt_seguradora
                    + " - " + txt_NewProposta);
            addField("dataCadastro",
    formataDataParaYYYYMMDD(dataCadastro));
            addField("txt_proReg", txt_proReg);
```

```
            addField("txt_VgblPgblNew", txt_VgblPgblNew);
            addField("nomeFundo", fundo);
            addField("percentualFundo", percentual);

            var result = service.updateCardData(1, "" +
config.usuarioFluig, ""
                        + config.senhaFluig,
parseInt(retornoRegistroExistente),
                        cardFieldDtoArray);

            var docId = result.getItem().get(0).documentId;
            if (docId > 0) {
                return docId;
            } else {
                log.error("Erro para salvar registro Proposta:
"
                                +
result.getItem().get(0).webServiceMessage);
                return "nok";
            }
        } catch (e) {
            log.error("Erro excecao update proposta: " +
e.toString());
            return "nok";
        }
}

function incluiRegistroPropostaFluig(cpfCnpj,
zoomNomeCompleto,
            txt_cdSeguradora, txt_seguradora, txt_NewProposta,
mensalidade, aporte,
            dataCadastro, txt_proReg, txt_VgblPgblNew, fundo,
percentual) {
        log.info("### incluiRegistroPropostaFluig ###");
        try {
            var config = recuperaDadosFixos();
            var serviceProvider = ServiceManager
                        .getServiceInstance("ECMCardService");
            var wkServiceEngine = serviceProvider

        .instantiate("com.totvs.ECMCardServiceService");
            var service = wkServiceEngine.getCardServicePort();
            var cardDtos =
serviceProvider.instantiate("com.totvs.CardDtoArray");
            var cardDto =
serviceProvider.instantiate("com.totvs.CardDto");
```

```
          cardDto.setParentDocumentId(parseInt(""
                    +
recuperaIdFormPaiPropostaSeguradora()));
          cardDto.setColleagueId("" + config.usuarioFluig);
          cardDto.setInheritSecurity(true);

          var addField = function(name, value) {
               var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
               dto.setField(name);
               dto.setValue(value || "");
               cardDto.getCardData().add(dto);
          };

          addField("cpfCnpj", cpfCnpj);
          addField("nomeCompleto", zoomNomeCompleto);
          addField("txt_cdSeguradora", txt_cdSeguradora);
          addField("txt_seguradora", txt_seguradora);
          addField("txt_propostaSeguradora",
txt_NewProposta);
          addField("campoDescritor", zoomNomeCompleto + " - "
+ txt_seguradora
                    + " - " + txt_NewProposta);
          addField("dataCadastro",
formataDataParaYYYYMMDD(dataCadastro));
          addField("txt_aporte", aporte);
          addField("txt_mensalidade", mensalidade);
          addField("nomeFundo", fundo);
          addField("percentualFundo", percentual);
          addField("txt_proReg", txt_proReg);
          addField("txt_VgblPgblNew", txt_VgblPgblNew);

          cardDtos.getItem().add(0, cardDto);
          var result = service.create("1", "" +
config.usuarioFluig, ""
                    + config.senhaFluig, cardDtos);

          var docId = result.getItem().get(0).documentId;
          if (docId > 0) {
               return docId;
          } else {
               log.error("Erro para salvar registro proposta:
"
                         +
result.getItem().get(0).webServiceMessage);
               return "nok";
          }
```

```
    } catch (e) {
        log.error("Erro excecao insert proposta: " +
e.toString());
        return "nok";
    }
}

function consultaRegistroSaldoSeguradoraPorCliente(cpfCnpj,
txt_cdSeguradora,
        txt_seguradoraMesComp, txt_propostaSeguradora) {
    // BLINDAGEM DE DADOS
    var vCpf = String(cpfCnpj || "");
    var vSeg = String(txt_cdSeguradora || "");
    var vMes = String(txt_seguradoraMesComp || "");
    var vProp = String(txt_propostaSeguradora || "");

    var c1 = DatasetFactory.createConstraint('cpfCnpj',
vCpf, vCpf,
            ConstraintType.MUST);
    var c2 =
DatasetFactory.createConstraint('txt_cdSeguradora', vSeg,
vSeg,
            ConstraintType.MUST);
    var c3 =
DatasetFactory.createConstraint('txt_seguradoraMesComp',
vMes,
            vMes, ConstraintType.MUST);
    var c4 =
DatasetFactory.createConstraint('metadata#active', "true",
"true",
            ConstraintType.MUST);
    var c5 =
DatasetFactory.createConstraint('rdTipoInclusao', "saldo",
            "saldo", ConstraintType.MUST);
    // CORREÇÃO: Passando a constraint da proposta
    var c6 =
DatasetFactory.createConstraint('txt_propostaSeguradora',
vProp,
            vProp, ConstraintType.MUST);

    var constraints = new Array(c1, c2, c3, c4, c5, c6);

    // CORREÇÃO: Usando arrays vazios [] para evitar erro de
assinatura no Rhino
    var datasetDs_registrosSaldoSeguradoraCliente =
DatasetFactory.getDataset(
```

```
                'ds_registrosSaldoSeguradoraClienteFuture',
[], constraints, []);

    if (datasetDs_registrosSaldoSeguradoraCliente != null
                &&
datasetDs_registrosSaldoSeguradoraCliente.rowsCount > 0) {
        return
datasetDs_registrosSaldoSeguradoraCliente.getValue(0,
                    "metadata#id");
    } else {
        return "";
    }
}

function
consultaRegistroPropostaSeguradoraPorCliente(cpfCnpj,
        txt_cdSeguradora, txt_NewProposta) {
    // BLINDAGEM DE DADOS
    var vCpf = String(cpfCnpj || "");
    var vSeg = String(txt_cdSeguradora || "");
    var vProp = String(txt_NewProposta || "");

    var c1 = DatasetFactory.createConstraint('cpfCnpj',
vCpf, vCpf,
                ConstraintType.MUST);
    var c2 =
DatasetFactory.createConstraint('txt_cdSeguradora', vSeg,
vSeg,
                ConstraintType.MUST);
    var c3 =
DatasetFactory.createConstraint('txt_propostaSeguradora',
vProp,
                vProp, ConstraintType.MUST);
    var c4 =
DatasetFactory.createConstraint('metadata#active', "true",
"true",
                ConstraintType.MUST);

    var constraints = new Array(c1, c2, c3, c4);

    // CORREÇÃO: Usando arrays vazios []
    var ds_registrosPropostaSeguradoraCliente =
DatasetFactory.getDataset(
                'ds_registrosPropostaSeguradoraClienteFuture',
[], constraints, []);

    if (ds_registrosPropostaSeguradoraCliente != null
```

```
                &&
ds_registrosPropostaSeguradoraCliente.rowsCount > 0) {
            return
ds_registrosPropostaSeguradoraCliente.getValue(0,
"metadata#id");
        } else {
            return "";
        }
}

// ... (Demais funções auxiliares: recuperaDadosFixos,
recuperaIdFormPai..., retornaIndices, preencherZerosEsquerda,
retornaData) ...
// (Inclua aqui todas as funções auxiliares que já estavam no
arquivo original, sem alterações)

function recuperaDadosFixos() {
    var constraintDs_config1 =
DatasetFactory.createConstraint(
            'metadata#active', 'true', 'true',
ConstraintType.MUST);
    var datasetDs_config =
DatasetFactory.getDataset('ds_configIntegracao', [
            'usuarioFluig', 'senhaFluig' ], new
Array(constraintDs_config1),
            null);

    try {
        if (datasetDs_config.rowsCount > 0) {
            return {
                "usuarioFluig" :
datasetDs_config.getValue(0, "usuarioFluig"),
                "senhaFluig" :
datasetDs_config.getValue(0, "senhaFluig"),
            }
        } else {
            log.error("Não há usuário e senha cadastrados
para integração...")
            throw "Não há usuário e senha cadastrados para
integração"
        }
    } catch (e) {
        log.error("Não há usuário e senha cadastrados para
integração..." + e)
        throw "Não há usuário e senha cadastrados para
integração" + e
    }
```

```
}

function recuperaIdFormPaiSaldoSeguradora() {
     var constraintDocument1 =
DatasetFactory.createConstraint('datasetName',
               'ds_registrosSaldoSeguradoraClienteFuture',
               'ds_registrosSaldoSeguradoraClienteFuture',
ConstraintType.MUST);
     var constraintDocument2 =
DatasetFactory.createConstraint('activeVersion',
               'true', 'true', ConstraintType.MUST);
     var datasetDocument =
DatasetFactory.getDataset('document',
               [ 'documentPK.documentId' ], new
Array(constraintDocument1,
                         constraintDocument2), null);
     return datasetDocument.getValue(0,
"documentPK.documentId")
}

function recuperaIdFormPaiPropostaSeguradora() {
     var constraintDocument1 =
DatasetFactory.createConstraint('datasetName',
               'ds_registrosPropostaSeguradoraClienteFuture',
               'ds_registrosPropostaSeguradoraClienteFuture',
ConstraintType.MUST);
     var constraintDocument2 =
DatasetFactory.createConstraint('activeVersion',
               'true', 'true', ConstraintType.MUST);
     var datasetDocument =
DatasetFactory.getDataset('document',
               [ 'documentPK.documentId' ], new
Array(constraintDocument1,
                         constraintDocument2), null);
     return datasetDocument.getValue(0,
"documentPK.documentId")
}

function retornaIndices(campoBase) {
     var indices = [];
     var campos = hAPI.getCardData(getValue("WKNumProces"));
     var chaves = campos.keySet().toArray();

     for ( var i in chaves) {
          if (chaves[i].indexOf(campoBase + "___") > -1) {
               indices.push(chaves[i].split("___")[1]);
          }
```

```
    }
    return indices;
}

function preencherZerosEsquerda(numero, tamanhoTotal) {
    var numeroString = String(numero);
    while (numeroString.length < tamanhoTotal) {
        numeroString = '0' + numeroString;
    }
    return numeroString;
}

function retornaData() {
    var data = new Date();
    var dia = data.getDate();
    var mes = data.getMonth() + 1;
    var ano = data.getFullYear();
    if (dia < 10) {
        dia = "0" + dia;
    }
    if (mes < 10) {
        mes = "0" + mes;
    }
    return dia + '/' + mes + '/' + ano;
}

function dataFormatada(data) {
    // PROTEÇÃO CONTRA NULL/UNDEFINED - Correção para o erro
"split of undefined"
    if (!data)
        return "";

    // Converte para string para garantir que possui o
método split
    var strData = String(data);

    var regexDDMMYYYY = /^\d{2}\/\d{2}\/\d{4}$/;
    var regexYYYYMMDD = /^\d{4}-\d{2}-\d{2}$/;

    if (regexYYYYMMDD.test(strData)) {
        var partes = strData.split('-');
        return partes[2] + '/' + partes[1] + '/' +
partes[0];
    } else if (regexDDMMYYYY.test(strData)) {
        return strData;
    }
    return null;
```

```
}

function formataDataParaYYYYMMDD(dataString) {
     // PROTEÇÃO CONTRA NULL/UNDEFINED
     if (!dataString)
          return "";

     var strData = String(dataString);
     var partesData = strData.split("/");

     if (partesData.length < 3)
          return strData; // Retorna original se formato
inválido

     return partesData[2] + "" + partesData[1] + "" +
partesData[0];
}
```