```
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 servicetask11(attempt, message) {

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

     var tipoInclusao = hAPI.getCardValue("rdTipoInclusao");
     log.info("### servicetask11 - rdTipoInclusao: " +
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;
          for ( var i in indicesSaldoComissao) {
               if (hAPI.getCardValue("retornoProtheus___" +
indicesSaldoComissao[i]) != "OK") {

     incluiTituloProtheus(indicesSaldoComissao[i], contador);
               }
               contador++;
          }
     }
}

function incluiTituloProtheus(indice, contador) {
```

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

        // Tratamento para remover underscore do final do
email se existir
        var emailRaw = hAPI.getCardValue("emailSolicitante")
|| "";
        var emailClean = emailRaw.trim().replace(/_+$/, "");

          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" : "PRV",
                    "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" :
dataFormatada(hAPI.getCardValue("txt_dataEmissao")),
                    "E1_VENCTO" :
dataFormatada(hAPI.getCardValue("txt_seguradoraVencimento___"
+ indice)),
                    "E1_VALOR" :
parseFloat(hAPI.getCardValue("txt_seguradoraComissao___" +
indice).replaceAll("\\.", "").replaceAll(",", ".")),
                    "E1_HIST" :
hAPI.getCardValue("txtHistoricoComissao___" + indice),
```

```
                    "E1_MOEDA" : 1,
                    "E1_TXMOEDA" : 1.0,
                    "E1_XNOMSOL" : "",
                    "E1_XEMAILS" : emailClean, // Email
corrigido
                    "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 java.lang.Exception("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 na integração 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);
```

```
     // CORREÇÃO: Usar data atual do servidor para evitar
erro de campo vazio
     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);
```

```
     // CORREÇÃO: Usar data atual do servidor
    var dataCadastro = retornaData();

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

    // Proteção contra JSON vazio ou inválido
    var jsonFundosParse = [];
    try {
        jsonFundosParse = JSON.parse(jsonFundos);
    } catch(e) {
        log.warn("Erro ao parsear JSON de fundos ou vazio: "
+ e);
    }

     var retornoRegistroExistente =
consultaRegistroPropostaSeguradoraPorCliente(
                cpfCnpj, txt_cdSeguradora, txt_NewProposta);

    log.info("## retornoRegistroExistente: " +
retornoRegistroExistente);
     var retornoIncluiAtualizaRegistro = "";

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

            if (retornoRegistroExistente != "") {
                retornoIncluiAtualizaRegistro =
atualizaRegistroPropostaFluig(
                        retornoRegistroExistente, cpfCnpj,
zoomNomeCompleto,
                        txt_cdSeguradora, txt_seguradora,
txt_NewProposta,
                        mensalidade, aporte, dataCadastro,
txt_proReg,
                        txt_VgblPgblNew, fundo, percentual);
            } else {
                retornoIncluiAtualizaRegistro =
incluiRegistroPropostaFluig(
                        cpfCnpj, zoomNomeCompleto,
txt_cdSeguradora,
                        txt_seguradora, txt_NewProposta,
mensalidade, aporte,
```

```
                        dataCadastro, txt_proReg,
txt_VgblPgblNew, fundo,
                        percentual);
                }
            }
        } else {
            // Caso não tenha fundos (exceção), tenta salvar sem
fundos ou loga aviso
            log.warn("Nenhum fundo encontrado para a proposta " +
txt_NewProposta);
        }
    }

// -------------------------------------------------------------
// FUNÇÕES DE CRUD (Inclui/Atualiza) - Mantidas as originais,
apenas ajustada a formatação
// -------------------------------------------------------------

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");

        var fields = [
            { field: "cpfCnpj", value: cpfCnpj },
            { field: "nomeCompleto", value: nomeCompleto },
            { field: "txt_cdSeguradora", value:
txt_cdSeguradora },
            { field: "txt_seguradora", value: txt_seguradora
},
            { field: "txt_seguradoraMesComp", value:
txt_seguradoraMesComp },
```

```
            { field: "txt_seguradoraSaldo", value:
txt_seguradoraSaldo },
            { field: "txt_seguradoraRentabilidade", value:
txt_seguradoraRentabilidade },
            { field: "campoDescritor", value: nomeCompleto +
" - " + txt_seguradora + " - " + txt_propostaSeguradora + " -
" + txt_seguradoraMesComp },
            { field: "dataCadastro", value:
formataDataParaYYYYMMDD(dataCadastro) },
            { field: "rdTipoInclusao", value: rdTipoInclusao
},
            { field: "txt_propostaSeguradora", value:
txt_propostaSeguradora },
            { field: "txt_investEsporadico", value:
txt_investEsporadico },
            { field: "txt_investMensal", value:
txt_investMensal }
        ];

        for var i = 0; i < fields.length; i++) {
            var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
            dto.setField(fields[i].field);
            dto.setValue(fields[i].value);
            cardFieldDtoArray.getItem().add(dto);
        }

          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 exceção 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(recuperaIdFormPaiSaldoSe
guradora()));
        cardDto.setColleagueId(config.usuarioFluig);
        cardDto.setInheritSecurity(true);

        var fields = [
            { field: "cpfCnpj", value: cpfCnpj },
            { field: "nomeCompleto", value: nomeCompleto },
            { field: "txt_cdSeguradora", value:
txt_cdSeguradora },
            { field: "txt_seguradora", value: txt_seguradora
},
            { field: "txt_seguradoraMesComp", value:
txt_seguradoraMesComp },
            { field: "campoDescritor", value: nomeCompleto +
" - " + txt_seguradora + " - " + txt_propostaSeguradora + " -
" + txt_seguradoraMesComp },
            { field: "dataCadastro", value:
formataDataParaYYYYMMDD(dataCadastro) },
            { field: "rdTipoInclusao", value: rdTipoInclusao
},
            { field: "txt_propostaSeguradora", value:
txt_propostaSeguradora }
        ];

        if (rdTipoInclusao == "saldo") {
            fields.push({ field: "txt_seguradoraSaldo",
value: txt_seguradoraSaldo });
```

```
            fields.push({ field:
"txt_seguradoraRentabilidade", value:
txt_seguradoraRentabilidade });
            fields.push({ field: "txt_investEsporadico",
value: txt_investEsporadico });
            fields.push({ field: "txt_investMensal", value:
txt_investMensal });
        } else if (rdTipoInclusao == "resgate") {
            fields.push({ field: "txt_seguradoraResgate",
value: txt_seguradoraResgate });
        }

        for var i = 0; i < fields.length; i++) {
            var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
            dto.setField(fields[i].field);
            dto.setValue(fields[i].value);
            cardDto.getCardData().add(dto);
        }

          cardDtos.getItem().add(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 ao incluir registro Fluig: " +
result.getItem().get(0).webServiceMessage);
            return "nok";
          }
      } catch (e) {
          log.error("Erro exceção 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) {
    // Mesma lógica de atualização, simplificada para
brevidade, mas mantendo a estrutura segura
    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 fields = [
            { field: "cpfCnpj", value: cpfCnpj },
            { field: "nomeCompleto", value: zoomNomeCompleto
},
            { field: "txt_cdSeguradora", value:
txt_cdSeguradora },
            { field: "txt_seguradora", value: txt_seguradora
},
            { field: "txt_propostaSeguradora", value:
txt_NewProposta },
            { field: "txt_mensalidade", value: mensalidade },
            { field: "txt_aporte", value: aporte },
            { field: "campoDescritor", value:
zoomNomeCompleto + " - " + txt_seguradora + " - " +
txt_NewProposta },
            { field: "dataCadastro", value:
formataDataParaYYYYMMDD(dataCadastro) },
            { field: "txt_proReg", value: txt_proReg },
            { field: "txt_VgblPgblNew", value:
txt_VgblPgblNew },
            { field: "nomeFundo", value: fundo },
            { field: "percentualFundo", value: percentual }
        ];

        for var i = 0; i < fields.length; i++) {
            var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
            dto.setField(fields[i].field);
            dto.setValue(fields[i].value);
            cardFieldDtoArray.getItem().add(dto);
        }

        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 update proposta: " +
result.getItem().get(0).webServiceMessage);
            return "nok";
        }
    } catch (e) {
        log.error("Erro exceção 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(recuperaIdFormPaiPropost
aSeguradora()));
        cardDto.setColleagueId(config.usuarioFluig);
        cardDto.setInheritSecurity(true);

        var fields = [
            { field: "cpfCnpj", value: cpfCnpj },
            { field: "nomeCompleto", value: zoomNomeCompleto
},
            { field: "txt_cdSeguradora", value:
txt_cdSeguradora },
            { field: "txt_seguradora", value: txt_seguradora
},
            { field: "txt_propostaSeguradora", value:
txt_NewProposta },
```

```
            { field: "campoDescritor", value:
zoomNomeCompleto + " - " + txt_seguradora + " - " +
txt_NewProposta },
            { field: "dataCadastro", value:
formataDataParaYYYYMMDD(dataCadastro) },
            { field: "txt_aporte", value: aporte },
            { field: "txt_mensalidade", value: mensalidade },
            { field: "nomeFundo", value: fundo },
            { field: "percentualFundo", value: percentual },
            { field: "txt_proReg", value: txt_proReg },
            { field: "txt_VgblPgblNew", value:
txt_VgblPgblNew }
        ];

        for var i = 0; i < fields.length; i++) {
            var dto =
serviceProvider.instantiate("com.totvs.CardFieldDto");
            dto.setField(fields[i].field);
            dto.setValue(fields[i].value);
            cardDto.getCardData().add(dto);
        }

        cardDtos.getItem().add(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 insert proposta: " +
result.getItem().get(0).webServiceMessage);
            return "nok";
        }
    } catch (e) {
        log.error("Erro exceção insert proposta: " +
e.toString());
        return "nok";
    }
}

// ---------------------------------------------------------
------------
// FUNÇÕES DE CONSULTA CORRIGIDAS (O Foco do Erro)
// ---------------------------------------------------------
------------
```

```
function consultaRegistroSaldoSeguradoraPorCliente(cpfCnpj,
txt_cdSeguradora, txt_seguradoraMesComp,
txt_propostaSeguradora) {
    // BLINDAGEM DE DADOS: Converter para string e tratar
nulos
    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);
     var c6 =
DatasetFactory.createConstraint('txt_propostaSeguradora',
vProp, vProp, ConstraintType.MUST);

    // CORREÇÃO CRÍTICA: Passando arrays vazios [] para
fields e sorting
    // E passando TODAS as constraints dentro do array
    var constraints = new Array(c1, c2, c3, c4, c5, c6);

     var dataset = DatasetFactory.getDataset(
                'ds_registrosSaldoSeguradoraCliente',
            [], // Campos (vazio)
           constraints, // Array de constraints correto
            [] // Ordenação (vazio)
    );

     if (dataset != null && dataset.rowsCount > 0) {
         return dataset.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);

     var dataset = DatasetFactory.getDataset(
               'ds_registrosPropostaSeguradoraCliente',
            [],
           constraints,
            []
    );

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

// ------------------------------------------------------------
// FUNÇÕES UTILITÁRIAS
// ------------------------------------------------------------

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("Erro config: " + e);
          throw e;
     }
}

function recuperaIdFormPaiSaldoSeguradora() {
     var constraintDocument1 =
DatasetFactory.createConstraint('datasetName',
'ds_registrosSaldoSeguradoraCliente',
'ds_registrosSaldoSeguradoraCliente', 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_registrosPropostaSeguradoraCliente',
'ds_registrosPropostaSeguradoraCliente',
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) {
	var regexDDMMYYYY = /^\d{2}\/\d{2}\/\d{4}$/;
	var regexYYYYMMDD = /^\d{4}-\d{2}-\d{2}$/;
	if (regexYYYYMMDD.test(data)) {
		var partes = data.split('-');
```

```
            return partes[2] + '/' + partes[1] + '/' +
partes[0];
        } else if (regexDDMMYYYY.test(data)) {
            return data;
        } else {
            return null;
        }
}

function formataDataParaYYYYMMDD(dataString) {
    if(!dataString) return "";
     var partesData = dataString.split("/");
    if(partesData.length < 3) return dataString;
     return partesData[2] + "" + partesData[1] + "" +
partesData[0];
}
```