# Top coisas que você não deveria fazer num código ABAP

*11/08/2014 09:00*

Nós já postamos diversas dicas de coisas que você não deveria fazer num código-fonte. Porém, em minhas andanças por ambientes de clientes, ainda vejo muitos códigos criados em 2013 /2014 , que têm os mesmos problemas de sempre.

Daí eu pensei: onde um desenvolvedor consegue um guia de coisas **BÁSICAS** ele deveria evitar, para que o código saia com uma qualidade um pouco melhor? Normalmente a consultoria/cliente tem algum guia de Best Practices, que ninguém nunca lê. A única preocupação dos devs é utilizar a nomenclatura que o cliente impõe, quando suas preocupações deveriam ir muito além disso.

Apresento aqui um checklist que você pode usar enquanto estiver codificando, com links e comentários explicando o porque de cada um dos pontos serem coisas que você deveria evitar. Se você sentir falta de algo, coloque nos comentários. Assim, aos poucos, iremos fazer um checklist bem mais robusto do que esses que existem perdidos e desatualizados nas empresas.

Em tempo: **A IDÉIA NÃO É APRESENTAR UM GUIDELINE PERFEITO. É LISTAR AS COISAS MAIS FÁCEIS DE SEREM EVITADAS.** Isto não é um guia de performance ABAP, obrigado.

Vamos lá!

---

- **Não crie constantes inúteis, tipo C_X, C_I. C_EQ, C_SPACE.**
  - Já expliquei o porque disso **neste post**. Você pode utilizar o *type-pools: abap* se quiser, para usar ABAP_TRUE, ABAP_FALSE e outras constantes que já existem e são bem mais descritivas do que o maldito C_X.
- **Comentários que não servem para nada.**
  - Não destrua **essa arte milenar**.
- **Variáveis com nomes repetidos e confusos, tipo T_AUX1, T_AUX2, V_TABIX1, V_TABIX2.**
  - No ABAP, as pessoas tem a mania de criar variáveis com nomes pequenos e repetidos, sabe-se lá porque. Não entendo porque criar uma T_ORDP ao invés de T_ORDEM_PRODUCAO. É medo de deixar o código mais claro? Claro que existe uma limitação no tamanho das variáveis, mas a única que limita em míseros 8 caracteres é a tela de seleção, o resto dá para deixar com nomes muito melhores. Quem já enfrentou um programa com T_MAT1, T_MAT2, T_MAT3, T_MAT4 e T_MAT5 sabe do que eu estou falando.
- **Mais de uma seleção na mesma tabela.**
  - Evite ao máximo fazer a mesma seleção duas vezes na mesma tabela em seu programa. Só faça isso se realmente não tiver mais nenhum jeito de evitar o acesso duplicado. Eu discuto com o funcional em busca de alternativas, sempre que encontro esse tipo de situação.
- **Não use LOOP WHERE em STANDARD TABLES.**
  - Sorted table existe há milhares de anos galera, porque não usá-las? Elas não mordem. LOOP WHERE em STANDARD TABLE é a mesma coisa que READ TABLE sem BINARY SEARCH em STANDARD TABLE (o que você também deveria evitar). Mas tentem usar SORTED TABLE, sério.. você não vai gastar nem 10 minutos aprendendo o básico de como utilizá-las.

- **Prefira INNER JOINS ao invés de FOR ALL ENTRIES, sempre que possível.**
  - FAE **não** são melhores do que Inner Joins. Se eu não te convenci, talvez **esse cara te convença**.
- **Alteração em tabela standard a partir de um programa Z**
  - Fazer isso daqui é pedir para dar problema. O que algumas pessoas não entendem, é que os programas standards muitas vezes não atualizam somente "a tabela que tem o número do documento". Quando uma ordem de venda é criada, não é só o registro na VBAK que é criado. Tem a VBAP, VBEP, VBUP, VBFA... E isso se repete para todo o sistema e outros tipos de documentos/registros. Alterar uma tabela standard é ignorar toda e qualquer consistência que o standard entrega para o banco de dados do cliente, e gerar problemas que você nem conseguirá imaginar.
- **CLEAR, REFRESH e FREE no começo de REPORTs, FORM ou MÉTODO para variáveis que acabaram de ser declaradas.**
  - Há alguns anos atrás eu falei que algumas pessoas sofrem da **síndrome do robô**. E parece que isso ainda continua acontecendo. Não faz o MENOR sentido limpar uma variável que acabou de ser declarada. É o seu atestado de que o seu medo de deixar "lixo" na variável é maior do que a vontade de entender como o sistema gerencia a memória dos programas. Ah, e o REFRESH está obsoleto, use o FREE no lugar.
- **COMMIT WORK em USER-EXIT, BADI, BTE, ENHANCEMENT, standard marretado**
  - Quer destruir o sistema? Então enfie um COMMIT-WORK dentro de uma USER-EXIT. As chances de você bagunçar a coisa toda é muito grande. Se você está em uma EXIT que está sendo feita dentro de um processo standard, é pq o processo standard ainda não acabou de ser executado, logo, como você pode dizer para o sistema commitar um processo que ainda não foi terminado? E se mais pra frente o standard resolver dar uma mensagem de erro? Cuidado.
- **MESSAGE em USER-EXIT**
  - Ah, então você não colocou COMMIT na EXIT, certo? De nada adianta se você foi lá e enfiou o comando MESSAGE para disparar uma mensagem do tipo I, E ou A. Saiba que o sistema pode disparar um COMMIT implícito, sem que você perceba. Sempre procure retornar mensagens com as ferramentas que a BADI/EXIT proporcionam. Aqui tem uns exemplos bem legais do que pode acontecer com o comando MESSAGE x COMMIT, em diferentes situações (sugestão por @jrnunes).
- **CALL TRANSACTION com constante/literal no MODE**
  - Sempre que você utilizar o CALL TRANSACTION, procure utilizar a extensão OPTIONS com uma workarea que não tenha nenhuma constante. Se você colocar o MODE com uma constante/literal e fixar um valor (por ex. "N" ), você está impossibilitando que um ABAPer altere parâmetros no debug do CALL TRANSACTION em QAS, para diagnosticar erros da execução da transação. Ajude o debug alheio: não deixe o MODE fixo. (sugestão por @fabiopagoti).
- **Não misture duas SM30s no mesmo grupo de funções**
  - Vai fazer duas SM30s (ou mais)? Então não tente economizar código colocando tudo no mesmo grupo de funções, mantenha o código de cada SM30 em seu próprio grupo. Isso vai evitar muita dor de cabeça quando o usuário invetar que quer alterar alguma coisa na tabela, e você tiver que regerar a SM30 (aquele monte de flags que você nunca lê, podem destruir tudo). Juntá-las num mesmo grupo também pode criar o caos quando você precisar transportar somente uma delas para PRD. A regra para não rolar stress é clara: cada SM30 no seu próprio grupo de função (sugestão por @lucattelli).

---

Por hora essas são as coisas principais. Evitando isso, seu código já vai melhorar bastante. A lista será atualizada constantemente, e este post ganhará destaque na home do site.

Se tiver alguma sugestão, comente! Mas lembre-se, daremos prioridade para coisas que são fáceis de arrumar, caso contrário, cairemos no mesmo problema dos guidelines de consultorias/cliente: documentos enormes e que ninguém lê.

Abraços a todos que revisam seu código antes de entregar!

---

## Comentários

**Lus Fabiano** — *01/11/2019 15:08*

Boa tarde. Qual a melhor forma de passar parâmetros para a transação “SOST?

**Carol** — *04/07/2017 15:00*

Olá, pode me informar se colocar ajuda de pesquisa para campo standard é uma prática correta?

**Haroldo** — *12/09/2016 10:42*

O importante é funcionar, não importa como...hahahaha

Ah, isso é usar no WHERE

LOOP AT t_fbukrs.
CONCATENATE ”” t_fbukrs-vkorg ”” INTO gbukrs.
if xlin = sy-tabix.
xvir = ”.
else.
xvir = ‘,’.
endif.

concatenate gbukrs xvir into gbukrs.
CONCATENATE xbukrs gbukrs INTO xbukrs.

ENDLOOP.

CONCATENATE ‘VKORG’ ‘ IN’ ‘ (‘ xbukrs ‘)’ INTO xbukrs.

**Raju Shrestha** — *07/06/2016 10:57*

Very well document list. This is a very good reference for all ABAPers.

Regards,
Raju.

**Oldo** — *22/08/2014 07:19*

Bom dia.
Amigo, sugiro você colocar referências referente a essas dicas, referências oficiais da SAP.
Temos ai vários argumentos que podem mudar bastante o ambiente... Vejo no nosso ambiente vários problemas

que você relatou mas infelismente não posso argumentar de forma válida devido a falta de fontes oficiais. Não sou ABAP mas faço parte da equipe Basis.

> **Mauricio Cruz** — *25/08/2014 17:50*
>
> Amigo, tudo que as pessoas precisam é apertar o F1:
>
> http://help.sap.com/abapdocu_731/en/abenabap_pgl.htm
>
> Na lista compartilhada só organizei o que todo mundo já deveria estar careca de saber. Não é nenhuma novidade, nem por parte da comunidade, nem por parte da SAP.
>
> Abs!

**Custodio** — *11/08/2014 20:21*

Ai vai minha pequena contribuicao. Alguns desses itens vem de cabeca, outros colei dos “best practices” do meu cliente atual:

Nao reutilizar um include em multiplos programas (shared includes). Isso aumenta demais o consumo de memoria e cria problemas semanticos. Ao inves disso, prefira classes ou interfaces globais para declaracoes comuns e metodos para implementacoes comuns.

Nao crie metodos/subrotinas/modulos de funcao com 5000 linhas. nem 500 linhas. Seja “razoavel”. Eu tento mantes abaixo de 50 linhas de codigo, mas ate 100 ainda passa. mais que isso seu bloco esta fazendo mais do que deveria, entao voce deve procurar quebra-lo em 2 (ou mais).

evite incontaveis niveis de IF ... ELSEIF.... Prefira o uso de CASE, e sempre coloque a opcao mais provavel de ococrrer em primeiro. Ex:

CASE politico.
WHEN c_bandido.
...

WHEN c_coroinha_da_missa_de_domingo.
....
ENDCASE.

Sempre que possivel declare os tipos dos parametros em suas subrotinas/funcoes/metodos. Nao use type ANY apenas por usar. Nao faca dinamico apenas por que eh “legal”, considere os “trade offs” envolvidos.

Nao use TABLES como parametro. ao inves disso, use EXPORTING ou IMPORTING tipo table type.

Em metodos, use RETURNING ao inves de EXPORTING, sempre que possivel.

Use elementos de dados de acordo com a semantica e nao com as caractersiticas tecnicas. Se voce precisa declarar V_nome_da_pessoa, use type NAME1, e nao TYPE TEXT40 so porque voce precisa de 40 caracteres.

Use LOOP at internal_table ASSIGNING ao inves de INTO estrutura. A perfoirmance eh melhor em geral, e principalmente se voce for modificar o conteuro de internal_table.

NAO ATUALIZE VARIAVEIS DE SISTEMA!!!!

Toda excecao deve ser tratada ou propagada. Trate as que voce puder e/ou souber, e propague nas demais. Nao use CATCH cx_root apenas pro seu programa nao dar dump.

Abracos,
Custodio

**Guilherme Oenning** — *11/08/2014 19:30*

Olá,

Eu ainda tenho minhas dúvidas em relação ao uso do FAE e JOIN. Comecei a pouco tempo no ABAP e sempre fui acostumado com JOINS, mas confesso que gostei bastante da sintaxe e do resultado final de um programa que usa FAE.

Fiz cara feia ao olhar o trace, mas não fiquei tão assustado assim, existem coisas piores no SAP. Talvez eu ainda caia do cavalo o dia que encontrar um problema de performance no FAE, mas ainda não passei por esta experiência.

Quando comecei a estudar FAE escrevi este post http://blog.oenning.eti.br/2014/05/explorando-o-for-all-entries-in/

Deem uma olhada no exemplo. Como vocês fariam para preencher as duas estruturas com JOIN sem fazer duas leituras na tabela pai? Entendo que no JOIN você teria uma única tabela de destino com todos os campos, tanto do pai quanto do filho. As colunas do pai estariam replicadas N vezes. Imagina isto em um cenário de 3 níveis, VBAK, VBAP e VBEP. Ter uma tabela interna separada fica muito mais fácil de trabalhar. Qual a opinião de vocês?

Abraço!

**Custodio** — *11/08/2014 21:27*

Fala Guilherme,

O proprio Matt diz no pst original dele na SCN que o JOIn eh melhor “IN MOS CASES”. Eu rarissimas vezes uso FAE, somente tentando essa alternativa se nao estiver feliz com a performance do JOIN. Alias, nem me lembro a ultima vez que usei.

Nao sei se entendi sua pergunta. Alterei seu programa com o uso de JOIN, apenas excuindo algumas linhas aqui e ali. Postei la no seu blog. Nao analisei a performance, mas me parece igual.

O mais importante, acho eu, eh PENSAR antes de fazer, e me parece que voce faz isso.

Abraco,
Custodio

**Henrique Dias** — *11/08/2014 15:40*

Não usar SY-ABCDE+23(1), de resto pode fazer o que você quiser hahahahahhaaha

**Mauricio Cruz** — *11/08/2014 15:42*

NÃO MEU, NÃO PODE TER CÓDIGO VERDE

**Bruno** — *11/08/2014 15:29*

Sobre SM30:

1 – Não usar um mesmo grupo de funções para servir de SM30 para mais de uma tabela. Você evita dois problemas bastante comuns por onde passei: A) SYNTAX ERROR em PRD, mas em QAS funciona. Geralmente porque você subiu o grupo, mas não removeu a SM30 nova/alterada de uma ou mais tabelas que ainda não chegaram em PRD (geralmente de outros projetos/demandas já em QAS, mas ainda não validados). Seja porque esqueceu, seja porque nem sabia da existência dela (normalmente quando outro programador inseriu ela lá). B) Você não corre o risco de perder a sua SM30 porque o amiguinho mandou “re-gerar o grupo de funções” para a tabela dele, eliminando registros de todas as outras SM30 que existiam lá. Neste caso, boa sorte re-gerando tudo.

2 – Não customizar a SM30. Eu sei que o SAP permite que você insira FORMs que são disparados em eventos, e que até mesmo a tela da SM30 pode ser alterada. Eu sei disso tudo, mas não, não faça isso. O primeiro que “re-gerar” sua SM30 vai perder tudo, e só vai descobrir semanas depois, quando o “log de modificação” não funciona mais na PRD. Se a tabela precisa de QUALQUER OUTRA COISA além da SM30, desenvolva.

Abraços!

> **Mauricio Cruz** — *11/08/2014 15:44*
>
> Valeu Bruno! Esse negócio de colocar duas SM30s no mesmo grupo de funções é mto ruim. Tudo funciona até o cara precisar fazer uma alteração. Daí ele não sabe direito o que regerar, e dá um monte de problemas...
>
> Vou destacar este ponto, valeu!

> **Bruno** — *11/08/2014 15:36*
>
> Ah, e não faça isso jamais!

**Fábio** — *11/08/2014 12:41*

Complementando a lista do que ***não fazer***:

– Inserir comentários do tipo “Começo do chamado ABC-1234” e “Fim do chamado ABC-1234” e entre eles código.

– Colocar comandos do tipo CHECK em user-exits

– END-OF-SELECTION

– Colocar um Loop no form “SELECIONA_DADOS”

– Colocar um Select no form “PREPARA_DADOS”

– Não usar parâmetros em rotinas

– Definir rotinas

– Definir select-options com o prefixo p_

– Declarar work areas globais (99% de chance que esta work area poderia ser local)

– Colocar um literal ou uma constante na variação MODE do comando CALL TRANSACTION

– Ter 100% das tabelas internas do seu código declaradas como STANDARD (Fortes indícios que você não sabe usar os outros tipos.. ou até saber que existam outros!)

– Copiar um programa standard

– Copiar um programa Z

– Colocar todo o código de uma BADi diretamente abaixo de um método em si ao invés de quebrar a lógica em métodos auxiliares na classe gerada automaticamente pela implementação da BADi.

– Usar mensagens genéricas do tipo & & &

– Definir elementos de dados sem usar domínios

– Definir tabelas transparente sem usar elementos de dados

Att,

> **Cris C.** — *13/09/2017 13:17*
>
> Qual o problema de colocar END-OF-SELECTION ??

> **Leo Schmidt** — *08/03/2018 14:57*
>
> Se o seu report não usa nenhum banco de dados lógico, simplesmente não faz sentido usar.
>
> Abraços!

> **Mauricio Cruz** — *11/08/2014 15:43*
>
> Fabio, valeu pelos inputs!
>
> Da sua lista, vou destacar um dos pontos que eu acho que atrapalha pra kct: o da constante no mode do call transaction. Custa colocar a pqp do options? haha.
>
> Valeu, abs!

> **Fábio** — *11/08/2014 16:15*
>
> O problema não é nem o MODE mas sim a constante nele. Se alguém colocar uma work area de constantes no OPTIONS dá no mesmo. O ruim é não poder alterar o MODE em tempo de execução via debugger.

> **Mauricio Cruz** — *11/08/2014 16:17*
>
> Acho que como direcionamento, o OPTIONS sempre funciona melhor do que o MODE. Mas entendi o seu ponto, e vou deixar mais claro no texto o lance da constante no OPTIONS tb.
>
> E shhhhhhhh não conta pra ninguém que dá pra fazer work area de cosntantes pô...

> **Fábio** — *12/08/2014 10:20*

> HAHAHAHA. Daqui a pouco tem gente criando uma work area com todas as contantes do programa.
>
> CONSTANTS: begin of c_all,
> c_a value ‘A’,
> c_b value ‘B’,
> …
> c_z value ‘Z’,
> end of c_all.

**Diogo** — *11/08/2014 11:13*

Outra coisa que me deixa p*&# da vida é colocar SELECT dentro de LOOP.
Já cansei de melhorar a performance de programas que o usuário diz que estão lentos.

> **Mauricio Cruz** — *11/08/2014 11:23*
>
> Ah, o velho e mau SELECT em LOOP. Eu até pensei em colocar, mas é algo que já é beeeeeeeeeeeeem batido em tudo qto é lugar, então preferi omitir. Se mais gente pedir, eu coloco. (de certa forma, esse comentário já faz parte do guia, enfim)

**Fawcs** — *11/08/2014 10:39*

Ah sim, NUNCA declare algo com header line, quer ver um for all entries dar pau? é você esquecer que a pessoa declarou a tabela com header line e você logo antes do select escrever:

if itab is not initial.

> **Mauricio Cruz** — *11/08/2014 11:25*
>
> Cara, você estava sem sorte nesse dia heim? Sem sorte e sem [] heoaheouah

**Fawcs** — *11/08/2014 10:21*

O ponto do refresh/clear/abracadabra: Já provei usando o debuguer com a analise de memoria que aquela merda de variável não fica com resquício de memória ..
Pior que isso é ver:
form abracadabra.
data: itab type tabe of sflight.

clear itab. refresh itab. free itab.
…

> **Mauricio Cruz** — *11/08/2014 10:28*
>
> E quando o cara coloca um CLEAR ou FREE numa variável local, antes do ENDFORM/ENDMETHOD?

> Eu entendo que a pessoa pode ter passado por coisas terríveis por conta de lixo na memória... Mas não custa nada testar e entender direito o problema, ao invés de sair tacando CLEAR e FREE em tudo qto é lugar...

> **Fábio** — *11/08/2014 12:27*
>
> Não custa nada garantir neh...
>
> Se fosse possível estas mesmas pessoas usariam MALLOC ao invés de DATA