Como gerar um sistema em 24H

P. Kantek

14 de julho de 2020

Sumário

1 Ambiente

A idéia desta iniciativa é prover know-how, ferramentas e dicas para projetar, criar e instalar sistemas em rápido processo, tipicamente 24 horas. O sistema vai rodar em Windows 64 bits (tipicamente windows 10 e sucessores) usando uma interface orientada a caractere – que é o mais indicado para sistemas baseados em bancos de dados textuais.

Por ser baseado em tecnologia Clipper, está limitado a telas de 24,80 – não há o que fazer. Entretanto, fez-se uma migração abandonando o Clipper, que foi descontinuado ainda no século passado e compilando o sistema no Harbour (https://sourceforge.net/projects/harbour-project/) software livre, e que aparentemente está no estado da arte em sistemas modernos.

1.1 Protocolo base

A interface básica do sistema aqui gerado é a de um menú de funções na horizontal na parte superior da tela principal, seguido de um drop-down (uma cortina) de subfunções que se abrem abaixo de cada função. A navegação aqui é pelas teclas de flechas do teclado, até chegar na subfunção desejada, como se pode ver nos dois exemplos abaixo

Ao teclar com ENTER aqui, abre-se uma tela nova que pode ser

- Na inclusão, uma tela em preto, com os campos esperados da inclusão. A inclusão é comandada pela tecla INS.
- Na alteração, uma visão tabular do banco de dados, na qual se navega usando HOME, END, PAGE-UP e PAGE-DOWN para grandes saltos e as flechas UP e DOWN para pequenos saltos. Achando o registro a alterar, um ENTER entre nele.
- Na exclusão, basta localizar o registro e teclar DEL.

1.2 Visão relacional

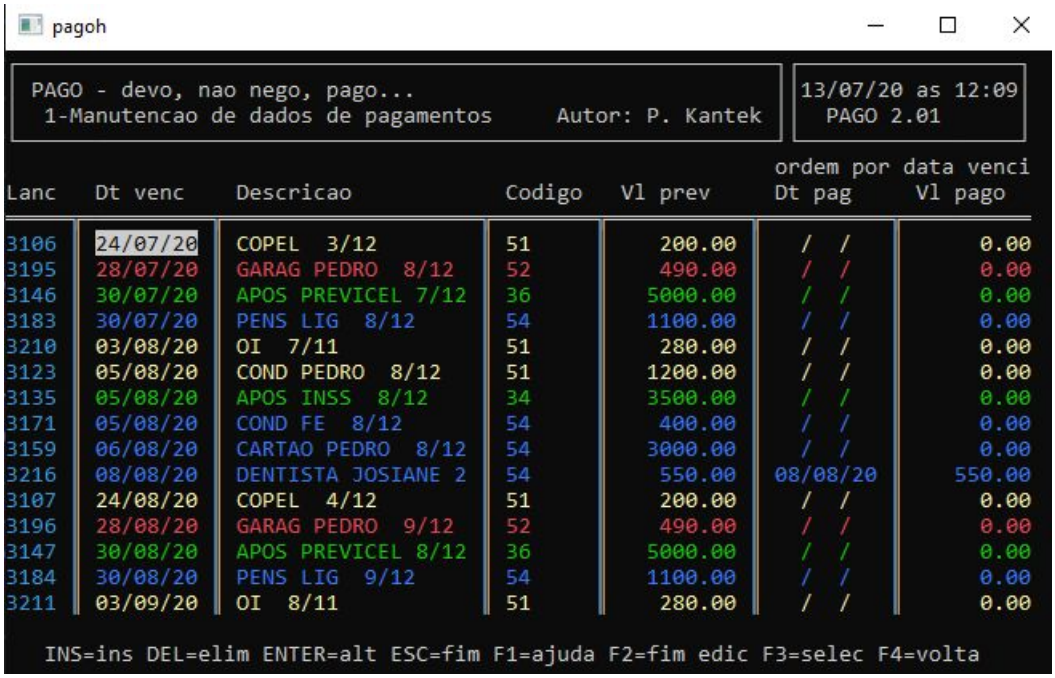
O banco de dados é aquele do Clipper, relacional, na forma de tabelas nas quais

linhas são os registros do banco de dados. Cada linha corresponde a uma ocorrência de dados.

colunas correspondem aos atributos (ou campos) do registro

Os tipos de dados empregados são os usuais (numéricos inteiros e de ponto flutuante – à gosto), textuais – limitados em princípio a 250 caracteres, mas por causa da interface do sistema, em geral tem um limite menor, tipicamente 70 caracteres ou algo próximo disso.

A seguir um exemplo desta interface

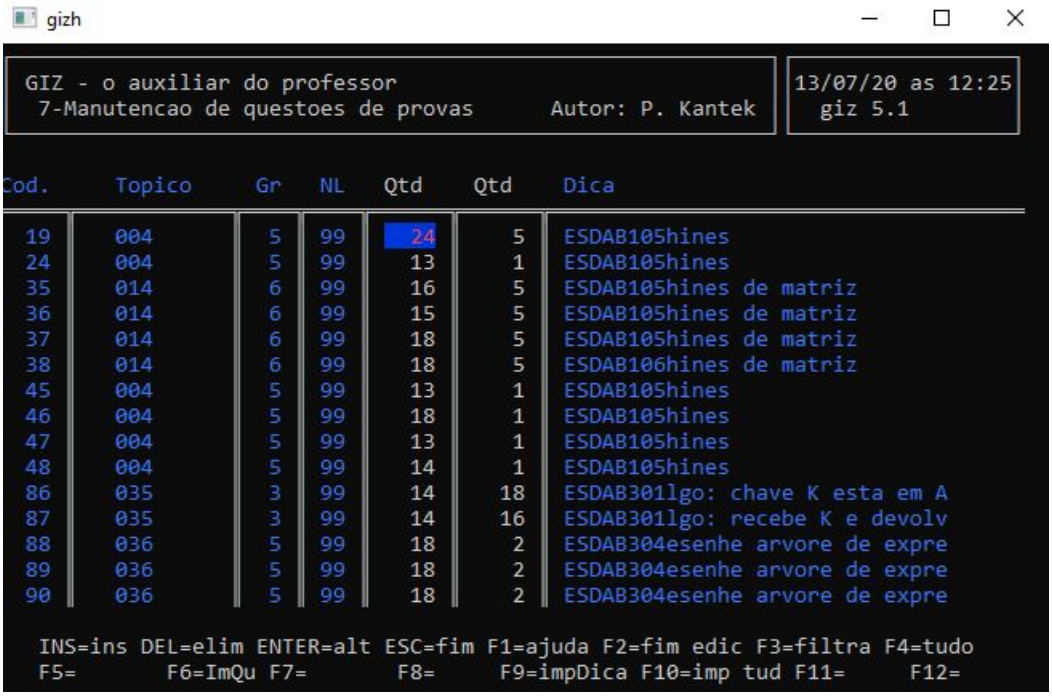


Mais um exemplo desta interface tabular

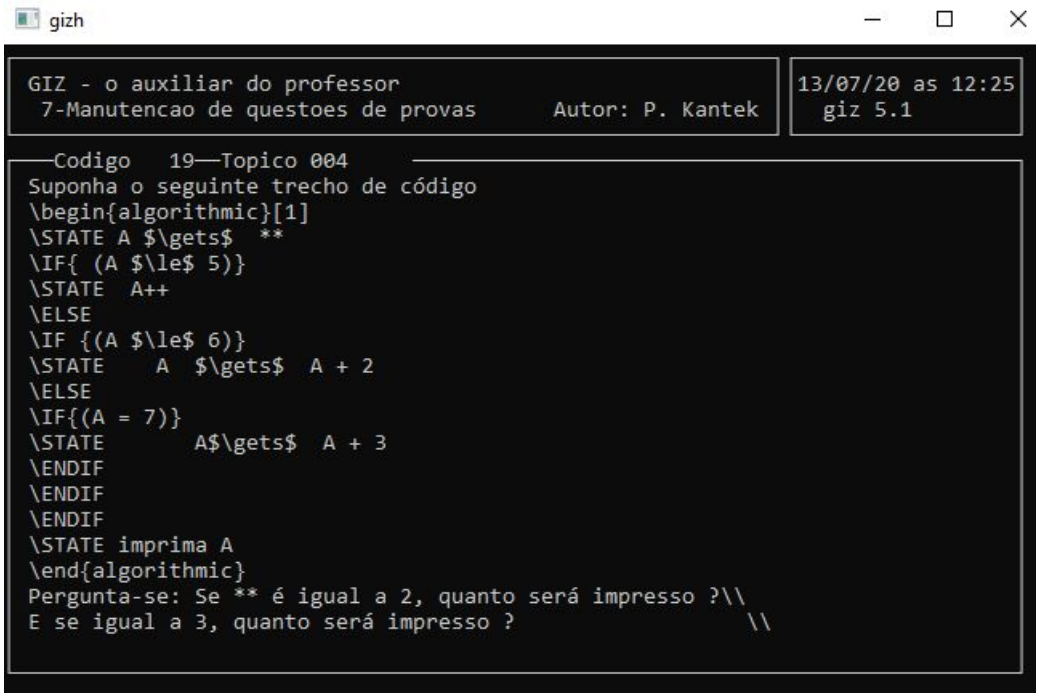
Note-se a existência também de campos do tipo DATA, que embora tratados internamento com anos de 4 dígitos, são mostrados na tela no formato DD/MM/AA, com ano contendo apenas 2 dígitos, para economia de espaço. Afinal o próximo bug do milênio só ocorrerá em 2999. podemos esperar relaxadamente.

1.2.1 Campos MEMO

A novidade aqui é este tipo de atributo que não pode ser manipulado na forma de tabela, já que seu comprimento pode chegar a 64Kb. Este campo é mostrado na tabela como a informação do número de linhas que ele tem. Quando clica-se aqui com ENTER, abre-se uma tela completa (24 linhas por 80 colunas) disponível para digitação e edição de textos livres. A edição se encerra com F2. Um exemplo do que se fala. Na tela a seguir, o quinto e o sexto campo são do tipo memorando.



Ao teclar ENTER com o cursor sobre o campo 5, comanda-se a edição deste campo, e abre-se a tela seguinte, que permite a edição:



A coleção de caracteres está limitada a ASCII, podendo-se usar páginas de código, mediante preparação prévia ou Unicode via UTF-8 (a conferir).

1.3 Teclas de função

Além deste protocolo, acima descrito, usam-se as seguintes PFs

F1 Auxílio, quando disponivel, ou não faz nada quando não disponível

F2 Padrão para terminar (com salvamento) qualquer edição em campo MEMORANDO

F3 Para criar um subconjunto de registros do banco de dados baseado em algum critério de seleção

F4 Para encerrar o subconjunto acima, passando a operar sobre todo o banco de dados

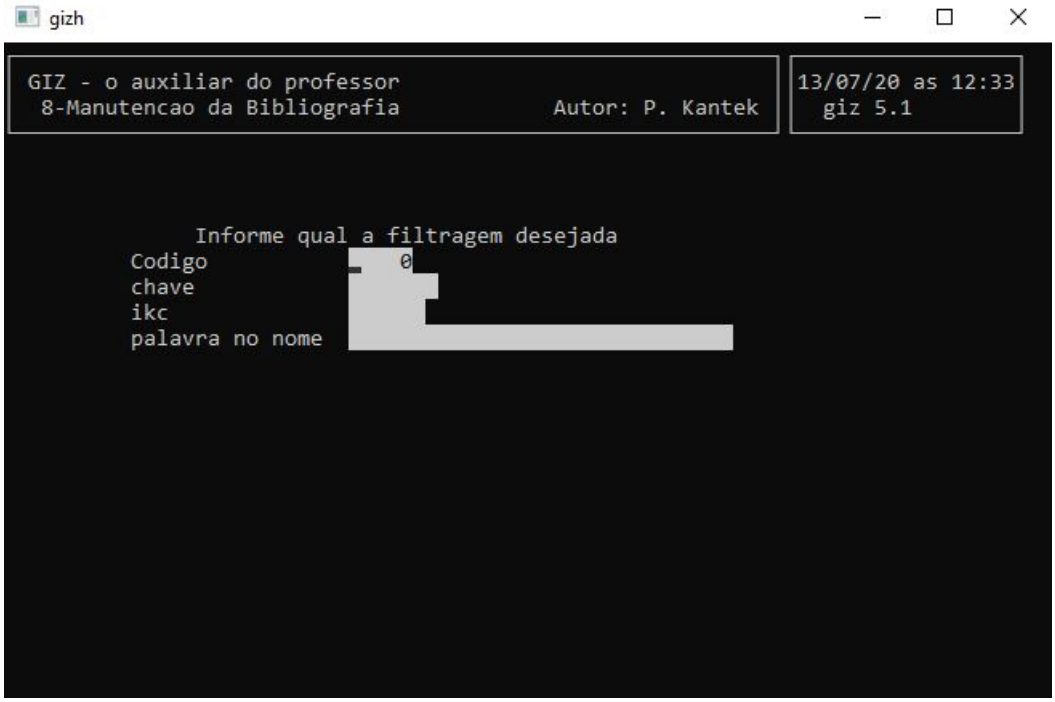
F5 Para modificar os critérios de ordenação do banco de dados, sempre em circularidade

F6 a F9 Livres para cada sistema em particular

F10 Impressão. Como a impressão tem variado ao longo dos últimos tempos (desde as impressoras matriciais, passando pelas lases/jato de tinta, até a visão mais moderna: PDF, esta opção gera aquivos em disco: o usuário que faça o que quiser depois.

ESC Tipicamente aborta o que se está fazendo.

A seguir, exemplos de critérios de seleção, chamados via F3, nos dois sistemas que estão sendo exemplificados



1.4 Cores

Como se viu nos exemplos tabulares acima, o sistema implementa a opção de colorir as linhas do banco de dados segundo critérios próprios da aplicação. Abre-se assim a possibilidade de uma dimensão adicional além das 2 (vertical e horizontal) do banco de dados textual.

2 As etapas do processo

2.1 1. O aplicativo

Seja individual ou parte de um todo maior o aplicativo deve ser identificado e contido definindo claramente o que fará e o que não fará parte dele. A saída deste bloco é um nome e uma declaração do que o aplicativo fará. Uma saída operacional

importante é uma sigla de 4 posições, que será usada adiante. Nesta documentação será AAAA.

2.2 2. O Banco de Dados

A primeira atividade em qualquer sistema é a modelagem dos dados do negócio. Há enorme literatura sobre como proceder, mas as boas práticas sugerem começar pelo modelo de dados, seguido de sua normalização e finalmente na sua transposição para um modelo físico compatível com aquele modelo lógico original. A saída desta etapa é um diagrama de entidades e relacionamentos. As entidades devem ser modeladas seguindo o padrão:

nome Uma sigla única de 4 posições. Neste exemplo será BBBB. O nome físico da entidade na implementação será AAAABBBB.

definição Uma informação sobre o papel desempenhando pela entidade

atributos Uma lista de atributos, onde para cada um deverá haver:

nome Um nome único de 4 caracteres do atributo. Aqui no exemplo é CCCC. Na implementação física, ele será BBBBCCCC.

chave A informação se esta entidade será ou não indexada, eventualmente fazendo parte de super ou sub chaves.

tipo O tipo do atributo, podendo ser N=numérico, A=alfanumérico, D=data, M=memorando

tamanho A quantidade máxima de bytes alocados a este atributo

outras informações quando aplicáveis

Esta etapa se encerra quando o modelo de entidades e relacionamentos estiver aprovado e for capaz de responder a todas as funcionalidades previstas para este sistema, bem como as interações com outros sistemas.

2.3 3. O diagrama de funções

A seguir, precisa-se definir a árvore de comandos para o sistema em 2 níveis:

- comandos de primeiro nível, com um limite de 6 ou 7 comandos.
- comandos de segundo nível, ligados a comandos de nível superior. O volume máximo deve estar por volta de até 9 subcomandos associados a cada comando.

2.4 4. funções atuariais

Identificar quais das funções e subfunções do programa são meramente atuariais (isto é, funções de inclusão, alteração, exclusão e listagem). Estas funções já estão resolvidas por este modelo, precisam apenas ser identificadas. Se forem atuariais com alguma especificidade, não há problemas, apenas este fato deve ser notado.

2.5 5. funções específicas

Identificar as funções que precisarão ser exaustivamente implementadas. Nos sistemas mais simples, quase não já o que fazer aqui, mas se houver, tais funções deverão ser exaustivamente estudadas e descritas.

2.6 6. Revisão e aprovação

Considerando que até agora só se produziu papel, quaisquer modificações terão custo irrisório. Até aqui nada se fez no computador, pelo que mudanças podem ser propostas e implentadas. A saída desta etapa é um relativo congelamento nas especificações, que a propósito, devem ser formalmente aprovadas.

3 Os programas

O primeiro programa no modelo é o supervisor da aplicação. Ele é a porta de entrada e a de saída da aplicação.

3.1 Tela inicial

Contendo 4 blocos, a saber

3.1.1 Bloco superior

4 linhas, dendo a primeira e a quarta ocupadas por um quadro. Idem para a primeira e a última coluna. Há dois blocos aqui: à esquerda a identificação: na linha 2 a sigla e nome do sistema e na linha 3: qualquer informação relevante. No bloco da direita, data e hora na linha 2 e sigla e versão na linha 3.

3.1.2 Bloco de comandos

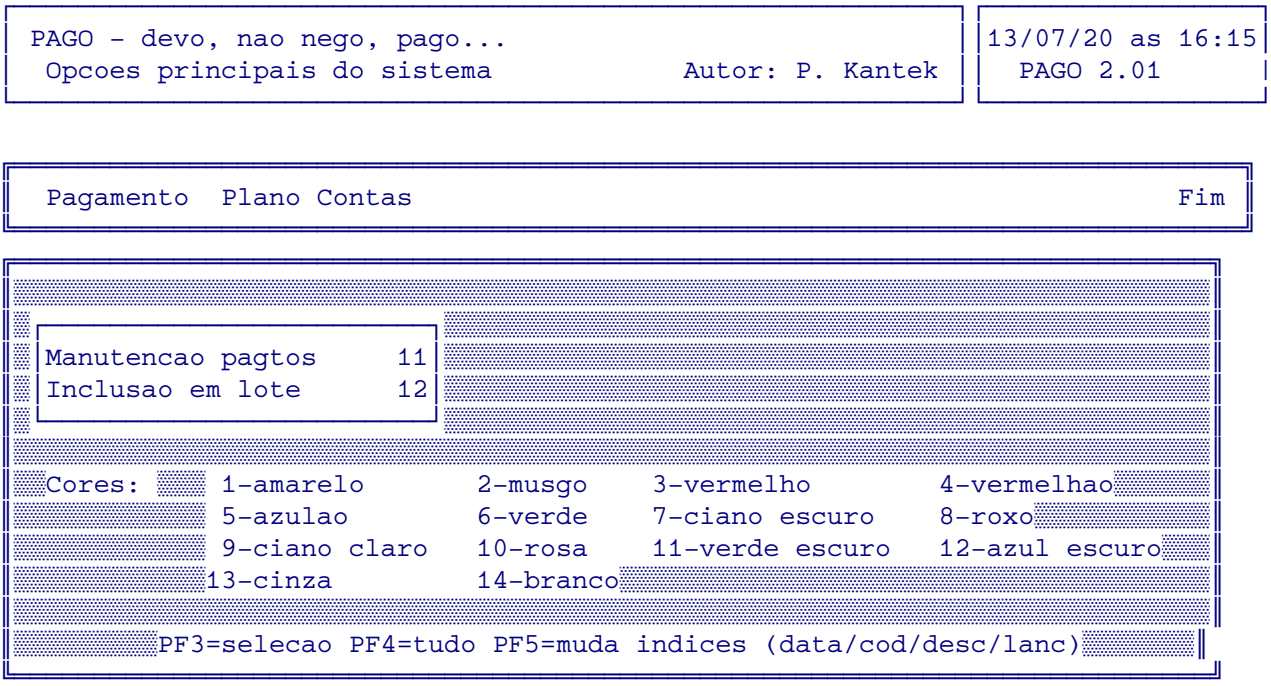
3 linhas sendo a primeiro e a terceira formada por quadro. Na única linha que fica estão os comandos da aplicação. O último é sempre FIM que sinaliza a saída do aplicativo. A localização (coluna) de cada comando precisa ser memorizada, pois será usada na escrita dos subcomandos mais abaixo.

3.1.3 Blocos de subcomandos

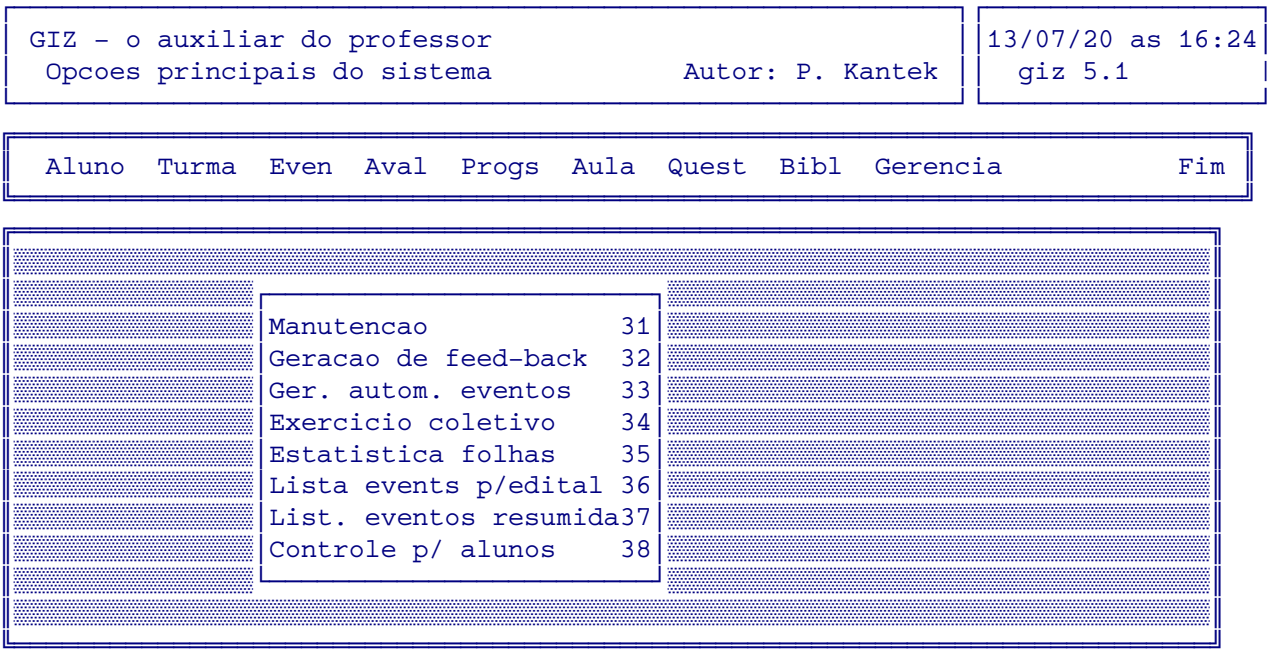
Um número variável de linhas, nas quais serão escritos os subcomandos associados a cada comando, quando o cursor estiver sobre cada comando. Quando o cursor sair daí, o bloco de subcomandos será apagado.

3.1.4 Rodapé padrão

Duas linhas com alguma informação relevante. Usualmente a lista de teclas de função ativas. Veja-se um exemplo:



Ou então outro exemplo



Manutencao de eventos que gerem notas

3.2 Arquivos

Todos os sistemas feitos de acordo com esta metodologia testam a existência dos arquivos de dados (tipos *.DBF e *.DBT) logo no início. Se os arquivos existirem eles serão usados normalmente. Se não existirem, serão criados vazios. Esta estratégia permite reinstalar o software com facilidade. Basta levar e executar o programa sem nenhuma outra infraestrutura. Ao final os arquivos estarão criados. O mesmo procedimento é feito quantos aos índices. Assim, havendo qualquer suspeita de corrupção dos índices, basta excluir o arquivo. Na próxima vez que o programa for executado, ele detectará a perda do índice e o recriará. A parte do programa que fará isto é

```
************************ ASSINALAARQ ************************
*                 Assinala o arquivo e o indice               *
*************************************************************
Function ASSINALAARQ()
   clear
   @ 11,35 say "PAG  (c) v.1"
   @ 02,10 clear to 06,70
   @ 02,11 to 06,69
   @ 18,10 clear to 22,70
   @ 18,11 to 22,69
   inkey()  // era inkey(2) para esperar 2 segundos

if .not. file ("pagocada.dbf")
   adbf :=
   aadd(adbf,"pagolanc","N",4,0)
   aadd(adbf,"pagodven","D",8,0)
   aadd(adbf,"pagodesc","C",18,0)
   aadd(adbf,"pagocodi","C",6,0)
   aadd(adbf,"pagovalp","N",9,2)
   aadd(adbf,"pagodpag","D",8,0)
   aadd(adbf,"pagovalr","N",9,2)
   aadd(adbf,"pagoobse","C",60,0)
   dbcreate("pagocada.dbf",adbf)
   sele a
   use pagocada
   delete file ("pagocad1.ntx")
endif
if file("pagocad1.ntx") .and. file("pagocad2.ntx") .and. file("pagocad3.ntx");
   .and. file("pagocad4.ntx")
   sele a
   use pagocada index pagocad1, pagocad2, pagocad3, pagocad4
else
   use
   use pagocada
   pack
   copy to lixo
   use
   erase "pagocada.dbf"
   erase "pagocad1.ntx"
   erase "pagocad2.ntx"
   erase "pagocad3.ntx"
   erase "pagocad4.ntx"
   rename "lixo.dbf" to "pagocada.dbf"
   use pagocada
   index on pagolanc to pagocad1
   index on pagodven to pagocad2
   index on pagocodi+dtos(pagodven) to pagocad3
   index on pagodesc to pagocad4
   sele a
   use pagocada index pagocad1, pagocad2, pagocad3, pagocad4
endif


if .not. file ("pagoplac.dbf")
```



```
   adbf:=
   aadd(adbf,"placcodi","C",6,0)
   aadd(adbf,"placdesc","C",30,0)
   aadd(adbf,"placcorx","N",2,0)
   aadd(adbf,"placvalp","N",9,2)
   aadd(adbf,"placvalr","N",9,2)
   aadd(adbf,"placqtde","N",4,0)
   dbcreate("pagoplac.dbf",adbf)
   sele b
   use pagoplac
   dele file ("pagopla1.ntx")
endif
if file("pagopla1.ntx")
   sele b
   use pagoplac index pagopla1
else
   sele b
   use pagoplac
   pack
   index on placcodi to pagopla1
   sele b
   use pagoplac index pagopla1
endif
return nil
```

3.3 Árvore de comandos

Os comandos e subcomandos serão exibidos por

```
Function main()
   SetMode(24,80)
   public auto := "P. Kantek      "
   public vers := "2.01"
   set epoch to 2000
   clear
   set date brit
   set deleted on
   assinalaarq()
   cabeca("Opcoes principais do sistema")
   declare niv1  [2]
      niv1  [1] = "Manutencao pagtos     11"
      niv1  [2] = "Inclusao em lote      12"
   declare niv2  [2]
      niv2  [1] = "Manutencao Pl Contas  21"
      niv2  [2] = "Balancete             22"
   sing = B_DOUBLE+chr(176)
   cho = 1
   nvez = 0
   while .t.
      nvez = nvez + 1
      @ 5,0 clear to 24,79
      set wrap on
      set message to 24 center
      @ 6,1 to 8,79 doub
      @ 10,2,23,78 box sing
      if nvez = 1
         @ 12,3 clear to 16,39
         @ 12,3 to 16,39
         @ 13,4 say "  Convertido para Harbour        "
         @ 14,43 say "Direitos autorais reservados.  "
         yyy = inkey(5)
         @ 6,1 to 8,79 doub
         @ 10,2,23,78 box sing
```

```
   endif
   declare col [5]   // numero de comandos
   declare chh [9]   // numero de subcomandos
   col[1]=4
   @ 7,col[1] promp "Pagamento" message "Manutencao de pagamentos"
   col[2] = col() + 2
   @ 7,col[2] promp "Plano Contas" message "Plano de contas"
   col[3] = col() + 2
   @ 7,74 promp " Fim " message "Para encerrar a execucao"
   keyboard chr(13)
   menu to cho
   @ 17,05 say "Cores: "  // conversa fiada
   @ 22,12 say "PF3=selecao PF4=tudo PF5=muda indices (data/cod/desc/lanc)"
   if cho =         3 // o ultimo ?
      resp = "N"
      @ 24,10 say "Deseja sair realmente ? - Tecle S para confirmar" get    resp
      read
      if upper(resp) = "S"
         clear
         @ 1,3 to 22,79
         @  2,5  say "msg na saída           "
         wait "Tecle algo para encerrar..."
      return nil
   else
      @ 24,10 clear
      cho = 1
      loop
   endif
endif
set key 4 to alfa
set key 19 to alfa
if cho = 1
   @ 12,col[1] to 15,col[1]+25 // 12+tamanho col1 + 1
   chh = achoice (13,col[1]+1,15,col[1]+24,niv1)
   set key 4 to
   set key 19 to
   if chh =         1
       pago11() // com=1 subc=1
   endif
   if chh = 2
       pago12() // com=1 subc=2
   endif
   set key 4 to alfa
   set key 19 to alfa
   cabeca("Opcoes principais do sistema")
   loop
endif
if cho = 2
   @ 12,col[2] to 15,col[2]+25
   chh = achoice (13,col[2]+1,15,col[2]+24,niv2)
   set key 4 to
   set key 19 to
   if chh =         1
     pago21()
   endif
   if chh =         2
      pago22()
   endif
   set key 4 to alfa
   set key 19 to alfa
   cabeca("Opcoes principais do sistema")
   loop
endif
```



```
   loop
   endif
enddo
//help()
return nil
```

3.4 Funções auxiliares

```
#include "INKEY.CH"
#include "FILEIO.CH"
#include "BOX.CH"
#command LIMPATELA  => @ 4,0 clear to 24,79
```

e mais

```
Function cabeca(fun)
set scoreboard off
// nao esquecer de alterar a funcao 81 (historico de versoes)
sipr = "pago"
nosi = "PAGO - devo, nao nego, pago..."
clear
@ 0,0 to 3,60
@ 1,2 say nosi
@ 2,3 say fun
@ 2,43 say "Autor: "+trim(auto)
@ 0,61 to 3,79
@ 1,62 say dtoc(date())+" as "+substr(time(),1,5)
@ 2,64 say "PAGO "+vers
return nil

Function alfa()
 if lastkey() = 4
  cho = cho + 1
 else
  cho = cho -1
 endif
keyboard chr(27) + chr(27)
set key 4 to
set key 19 to
return nil
```

3.5 Esquema geral de browse do banco de dados

O código a seguir mostra as linhas e colunas do banco de dados. Usando o objeto Tbrowse do Clipper, permite navegar pelos dados, implementando a lógica básica que determina

- Navegação: teclas PGUP, PGDN, ←, →, ↓, ↑, Home, End
- Inclusão: tecla INS
- Exclusão: tecla DEL
- Alteração: tecla ENTER

O código fica assim

```
function pago11()
   local obje,colu,tecl,z,mudou
   local res_filtro :=  || .t.
   mudou := 1
   set deleted on
   set date brit
   cls
   tabela_cores = "W/N,N/G,GR+/N,GR/N,R+/N,R/N,B+/N,G+/N,BG/N,RB/N,BG+/N,RB+/N,G/N,B/N,W/N,W+/
   w_dven = space(8) // campos da aplicação
   w_desc = space(20)
```



```
   w_codi = space(6)
   w_valp = 0.0
   w_dpag = space(8)
   w_valr = 0.0
   w_obse = space(60)

   cabeca("1-Manutencao de dados de pagamentos"
// ************** arma esquema de browse **************
   SETCURSOR(0)
   SELE A
   obje := TbrowseDB(5,0,21,79)
   obje:headsep := chr(205) + chr(203) + chr(205)
   obje:colsep := " " + chr(186) + " "
   obje:colorspec := tabela_cores

   colu := TbcolumnNew("Lanc",fieldblock("pagolanc"))
   colu:Colorblock :=  |y| escocor (y)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Dt venc",fieldblock("pagodven"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Descricao",fieldblock("pagodesc"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Codigo",fieldblock("pagocodi"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Vl prev",fieldblock("pagovalp"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Dt pag",fieldblock("pagodpag"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Vl pago",fieldblock("pagovalr"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   colu := TbcolumnNew("Observacao",fieldblock("pagoobse"))
   colu:Colorblock :=  |y| escocor (pagocodi)
   obje:addcolumn(colu)

   obje:freeze := 1
   @ 4,60 say "ordem por data vencim"
   sele a
   set order to 2
   set softseek on
   seek date()   começa a busca para apresentação por hoje

// *************** ciclo eterno do programa **********
   while .t.
     sele a
     @ 23,3 say "INS=ins DEL=elim ENTER=alt ESC=fim F1=ajuda F2=fim edic F3=selec F4=volta"
     @ 24,3 say "F5=     F6=      F7=       F8=     F9=      F10=imprime F11=     F12=    "
     if lastrec() == 0
        LIMPATELA
        @ 10,8 say " Nao existem pagamentos. Tecle ENTER para incluir o primeiro" // arq vazio
        tecl := inkey(0)
```



```
   if tecl == K_ESC
      EXIT
   endif
   inclue11(obje) // inclusão do primeiro
   loop
endif
if (obje:colpos <= obje:freeze)
    obje:colpos := obje:freeze + 1
 endif
 while (!obje:stabilize() )
    tecl := inkey()
    if (tecl != 0)
       exit
    endif
 end
 if (obje:stable)
    if (obje:hittop .or. obje:hitbottom)
       tone (87.3,1)
       tone (40,3.5)
    endif
    tecl :=inkey(0)
 endif
 IF ( tecl == K_DOWN )
    obje:down()
 ELSEIF ( tecl == K_UP )
    obje:up()
 ELSEIF ( tecl == K_PGDN )
    obje:pageDown()
 ELSEIF ( tecl == K_PGUP )
    obje:pageup()
 ELSEIF ( tecl == K_CTRL_PGUP )
    obje:gotop()
 ELSEIF ( tecl == K_CTRL_PGDN )
    obje:gobottom()
 ELSEIF ( tecl == K_RIGHT )
    obje:right()
 ELSEIF ( tecl == K_LEFT )
    obje:left()
 ELSEIF ( tecl == K_HOME )
    obje:home()
 ELSEIF ( tecl == K_END )
    obje:end()
 ELSEIF ( tecl == K_CTRL_LEFT )
    obje:panLeft()
 ELSEIF ( tecl == K_CTRL_RIGHT )
    obje:panRight()
 ELSEIF ( tecl == K_CTRL_HOME )
    obje:panHome()
 ELSEIF ( tecl == K_CTRL_END )
    obje:panEnd()
 ELSEIF ( tecl == K_ESC )
    EXIT
 ELSEIF ( tecl == K_ENTER )
    geteia11 (obje)
 elseif (tecl == K_INS )
    inclue11 (obje)
 elseif (tecl == K_DEL)
    exclue11 (obje)
 elseif (tecl == K_F3)
   res_filtro := filtrar11 (obje)  // operação de filtragem
 elseif (tecl == K_F4)
   res_filtro := desfilt11 (obje)  // desfiltragem
 elseif (tecl == K_F5)
```



```
        mudou:=mudakey11(mudou,obje)
      elseif (tecl == K_F10)            // impressão
         imprimir11 (res_filtro)
      ENDIF
   END
   CLS
RETURN NIL


function escocor (y)
   local manda :=
      sele B
      seek y
      if (b->placcorx == 0)
         xxx = 3
      else
         xxx = b->placcorx +
      endif
      manda := xxx,2
      sele A
return manda



// *************** processa o ENTER (alteracao) *************
Function geteia11(prop)
   local ncur := SETCURSOR(1)
   local colu,gete,tecl
   while (!prop:stabilize())
   end
   colu := prop:getcolumn(prop:colpos)
   gete := getnew( row(), col(), colu:block)
   readmodal(gete)
   prop:refreshall()  // coloca o registro no seu lugar JA
            // se fosse refreshcurrent colocaria so na proxima virada de tela
   SETCURSOR(0)
return nil


// ***************** Inclusao de novo registro *****************
Function inclue11 (prop)
   sele a
   set order to 1
   goto bott
   unum := a->pagolanc +
   goto top
   SETCURSOR(1)
   w_dven = space(8)
   w_desc = space(20)
   w_codi = space(6)
   w_valp = 0
   w_obse = space(60)
   w_obse1 = space(30)
   w_obse2 = space(30)
   w_dpag = space(8)
   w_valr = 0
   LIMPATELA
   @  5,22 say "INCLUSAO DE DADOS DE UM NOVO PAGAMENTO"
   @  7,13 say "Data de vencimento " get w_dven pict "@D"
   @  8,13 say "Descricao          " get w_desc pict "@!"
   @  9,13 say "Codigo (no PlanoC) " get w_codi
   @ 10,13 say "Valor previsto     " get w_valp pict "999999.99"
   @ 11,13 say "Obs                " get w_obse1
```



```
   @ 12,13 say "                    " get w_obse2
   @ 17,18 say "Data pagamento      " get w_dpag pict "@D"
   @ 18,18 say "Valor pago          " get w_valr pict "999999.99"

   @ 22,20 say "cola: 51-casa, 52-carro, 54-varejao"
   read
   w_obse = w_obse1 + w_obse2
   sele a
   set order to 1
   if w_dven <> space(8) .and. w_desc <> space(18)
      appe blank
      repl a->pagolanc with unum, a->pagodven with ctod(w_dven), a->pagodesc with w_desc
      repl a->pagocodi with w_codi, a->pagovalp with w_valp, a->pagodpag with ctod(w_dpag)
      repl a->pagovalr with w_valr, a->pagoobse with w_obse
   else
      wait "Vencimento ou descricao em branco"
   endif

   LIMPATELA
   prop:refreshall()  // coloca o registro no seu lugar JA
   SETCURSOR(0)
return nil

Function mudakey11 (quale,prop)
   sele a
   if quale == 1
      set order to 2
      @ 4,60 say "ordem por data venc "
      seek date()
//      goto top
      prop:refreshall()
      return 2
   endif
   if quale == 2
      set order to 3
      @ 4,60 say "ordem por codigo    "
      goto top
      prop:refreshall()
      return 3
   endif
   if quale == 3
      set order to 4
      @ 4,60 say "ordem por descricao "
      goto top
      prop:refreshall()
      return 4
   endif
   if quale == 4
      set order to 1
      goto top
      @ 4,60 say "ordem por lancamento"
      prop:refreshall()
      return 1
   endif
   return nil

// ******************** exclusao de registro *****************
Function exclue11 (prop)
local respo
respo = " "
SETCURSOR(1)
LIMPATELA
sim = "NAO"
```



```
@ 08,25 say "Codigo do lancamento " +str(a->pagolanc)
@ 10,25 say "Data de vencimento   " +dtoc(a->pagodven)
@ 12,25 say "Descricao            " +a->pagodesc
@ 14,25 say "Codigo               "  +a->pagocodi
@ 20,10 say "Para excluir, digite SIM " get sim
read
if upper(sim) = "SIM"
   if deleted()
      recall
   else
      delete
      pack
   endif
endif
SETCURSOR(0)
prop:refreshall()  // coloca o registro no seu lugar JA
LIMPATELA //
return nil


//************* funcao de selecao de acordo com condicoes *********
Function filtrar11(prop)
   local bfiltro
   public w_pala,w_dt1, w_dt2,w_vlac,w_vlab
   w_pala = space(20)
   w_dt1 = space(8)
   w_dt2 = space(8)
   w_vlac = 0
   w_vlab = 0
   SETCURSOR(1)
   LIMPATELA
   @ 6,15 say "Informe qual a filtragem desejada"
   @  9,20 say "descricao com a palavra  " get w_pala pict "@!"
   @ 10,20 say "entre duas datas venc    " get w_dt1 pict "@D"
   @ 10,55 say " e " get w_dt2 pict "@D"
   @ 11,20 say "acima de um valor prev   " get w_vlac
   @ 12,20 say "abaixo de um valor prev  " get w_vlab
   read
   if w_pala <> space(20)
      bfiltro :=  || (trim(w_pala) $ a->pagodesc)
      opcao_sel := 2
   elseif (w_dt1 <> space(8)) .and. (w_dt2 <> space(8))
      bfiltro :=  || ((ctod(w_dt1) <= a->pagodven) .and. (ctod(w_dt2) >= a->pagodven))
      opcao_sel := 3
   elseif w_vlac <> 0
      bfiltro :=  || w_vlac <= a->pagovalp
      opcao_sel := 4
   elseif w_vlab <> 0
      bfiltro :=  || w_vlab >= a->pagovalp
      opcao_sel := 5
   else
      bfiltro :=  || 1==1
      opcal_sel := 0
   endif
   prop:skipblock :=  |nregs| pulando11(nregs,bfiltro)
   prop:gotopblock :=  || vapcima11(bfiltro)
   prop:gobottomblock :=  || vapbaix11(bfiltro)
   prop:refreshall()  // coloca o registro no seu lugar JA
   setcursor(0)
   pulando11(0,nil)
return bfiltro
```



```
//************** desfazer o que a anterior fez ***************
Function desfilt11(prop)
   local filtro,nregs
   filtro :=  || .t.
   prop:skipblock :=  |nregs| pulando11(nregs,filtro)
   prop:gotopblock :=  || vaitop11()
   prop:gobottomblock :=  || vaibot11()
   pulando11(0,nil)
   vaitop11()
   prop:refreshall()  // coloca o registro no seu lugar JA
return filtro

Function vaitop11()
   go top
return nil

Function vaibot11()
   go bottom
return nil

//***************** implementa o salto condicional ****************
Function pulando11(ntojump,fifi)
   static lstartup := .t.
   static nrecnumtop, nrecnumbot
   local njumped := 0 // quantos registros foram saltados
   local nlastrec
   if lastrec() != 0
      if ntojump > 0
         while njumped < ntojump.and. recno() != nrecnumbot
            if (nlastrec := recno() ) != nrecnumbot
               skip
               while !eval(fifi) .and. !eof()
                  skip
               end
            endif
            if !eval(fifi) .or. eof()
               go nlastrec
               exit
            endif
            njumped++
         end
      elseif ntojump < 0
         while njumped > ntojump .and. recno() != nrecnumtop
            if (nlastrec := recno()) != nrecnumtop
               skip -1
               while !eval(fifi)
                  skip -1
                  if bof()
                     exit
                  endif
               end
            endif
            if !eval(fifi) .or. bof()
               go nlastrec
               exit
            endif
            njumped--
         end
      else  // ntojump = 0
         if fifi == nil
            lstartup := .t.
            return 0
         endif
```



```
         if lstartup
            lstartup := .f.
            while !eval(fifi) .and. !eof()
               skip
            end
            nrecnumtop := recno()
            go bottom
            while !eval(fifi)
               skip -1
               if bof()
                  exit
               endif
            end
            nrecnumbot := recno()
            go nrecnumtop
         else
            skip 0
         endif
      endif
   endif
return njumped



Function vapcima11(fifi)
   goto top
   while !eval(fifi)
      skip
   end
return nil

Function vapbaix11(fifi)
   go bottom
   while !eval (fifi)
      skip -1
   end
return nil


//********************* impressao ***********************
Function imprimir11(filx)
    public agora,clin,tudoc,tudod
    set printer to "listagem.fff"
//    set devi to printer
    clin := 80
    tudoc:=0
    tudod:=0
    dbeval (|| listara11(), filx,,,)
    set devi to printer
    @ clin,03 say "T O T A I S  ======  cred "
    @ clin,34 say transform(tudoc,"9,999,999.99")
    @ clin,50 say "debi "
    @ clin,57 say transform(tudod,"9,999,999.99")
    set devi to screen
    @ 22,10 say "Arquivo criado com o nome de LISTAGEM.FFF. Saia daqui com ESC"
//    CLS
return nil


Function listara11()
   local aa
   set devi to print
   aa := inkey()
```

```
   if aa <> 0 .and. lastkey() == K_ESC
       return
   endif
   if clin > 63
      @ 1,1 say "lancam"
      @ 1,10 say "data venc"
      @ 1,20 say "descricao"
      @ 1,40 say "cod"
      @ 1,48 say "vl pago"
      @ 1,60 say "dt pagam"
      clin := 3
   endif
   @ clin,1 say a->pagolanc
   @ clin,10 say a->pagodven
   @ clin,20 say a->pagodesc
   @ clin,40 say a->pagocodi
   @ clin,48 say a->pagovalr
   @ clin,60 say a->pagodpag
   if substr(a->pagocodi,1,1)<"4"
      tudoc := tudoc + a->pagovalr
   else
      tudod := tudod + a->pagovalr
   endif
   clin++
   set devi to screen
return nil
```

3.6 Edição de campo MEMORANDO

O campo deve ser mostrado na estrutura tabular dos dados como

```
   colu := TbcolumnNew("Qtd",{ || str(mlcount(que3enun),4)})
   colu:defcolor := {1,2}
   obje:addcolumn(colu)

   colu := TbcolumnNew("Qtd",{ || str(mlcount(que3resp),4)})
   colu:defcolor := {1,2}
   obje:addcolumn(colu)
```

E depois no programa de atualização de dados (aquele associado ao ENTER), deve-se fazer

```
  if (prop:colpos == 5)  // numero do campo memo enunciado
      LIMPATELA
      @ 4,0 to 24,79
      @ 4,4 say "Codigo " + str(h->que3codi)
      @ 4,17 say "Topico " + h->que3topi
        repl que3enun with memoedit(h->que3enun,5,2,22,77,.t.)
   else
      if ((prop:colpos < 5) .or. (prop:colpos == 7))
         colu := prop:getcolumn(prop:colpos)
         gete := getnew( row(), col(), colu:block)
         readmodal({gete})
      endif
   endif
```

4 Compilação

Para transformar um sistema operacional em Clipper (que não pode rodar em ambiente de 64 bits) em um plenamente funcional neste ambiente e igualmente no de 32 bits, deve-se compilar o fonte original no ambiente Harbour.

Uma mudança importante, é incluir o comando `SetMode(24,80)` logo após a definição da função `main()` do aplicativo, ficando assim

```
Function main()
   SetMode(24,80)
   ...
```


sem isto, a aplicação vai compilar, mas os endereçamentos de mensagens e campos nas telas de entrada e saída vão ficar mal posicionados.

A seguir, deve-se instalar o Harbour de acordo com as instruções do site de desenvolvimento conforme visto no início deste documento.

Finda a instalação, deve-se criar um arquivo BAT dentro do desktop do windows, com o nome de `SETARHB.BAT` e colocando nele

```
@echo off
set path=C:\hb30\bin;C:\hb30\comp\mingw\bin;%path%
set HB_COMPILER=mingw
cd\
%SystemRoot%\system32\cmd.exe
```

Para compilar uma aplicação Harbour este arquivo de lotes deve ser executado, pois ele acerta o caminho para o compilador C. A propósito, o Harbour transforma um conjunto de fontes Clipper em um conjunto de fontes em C entregando estes últimos a um compilador C padrão, no caso o `mingw`.

Quando este arquivo de lote rodar, deve-se – na mesma janela – mudar o diretório para aquele que contenha os programas fonte.

Neste diretório, deve-se criar um arquivo de nome XXXX.HBP contendo – como exemplo

```
-onomeaplic
-inc
-compr=yes
-quiet
-lxhb
-lhbwin
-lhbct
-gui
prog1.prg
prog2.prg
...
```

É claro que os arquivos `prog1.prg`, `prog2.prg` e etc são os programas fornte originalmente em Clipper e que vão gerar o aplicativo.

Tendo criado este arquivo é hora da mágica. Ainda na janela aberta acima, executar

```
hbmk2 XXXX.HBP
```

E, se tudo tiver ocorrido bem, o Harbour vai construir um programa chamado `nomeaplic.EXE` que será o executável desta aplicação.

Finalmente, é conveniente abrir um ícone qualquer no desktop apontando para este aplicativo, podendo-se agora chamar o aplicativo com um duplo clique no ícone.

Este texto está em /p/pago/sis24.tex

