

## DevStudio

## Sumário

Sumário.....	2
Visão Geral do Curso .....	4
Objetivos do Curso.....	5
O que é o DevStudio?.....	6
Configuração .....	7
Preferências.....	8
Configurações .....	11
Criação e Edição de Programas .....	14
Novo Arquivo.....	15
Caracter ASCII .....	17
Documentação de Explicação .....	18
Documentação de Códigos .....	20
Documentação de Cabeçalhos .....	21
Localizar .....	22
Repetir Procura Abaixo .....	23
Repetir Procura Acima .....	23
Substituir.....	23
Indentar .....	24
Minúsculo.....	25
Maiúsculo .....	25
Duplicar Linha .....	26
Ir para a Linha .....	26
Ir para a Linha em Execução.....	27
Projetos, Compilação, Geração e Aplicação de Patch e Análise do RPO .....	28
Projetos.....	29
Compilar Tudo .....	33
Compilação de Projetos.....	35
Compilação de Pastas.....	35
Compilação de Arquivos .....	35
Compilação em Batch .....	35
Geração de Patches .....	36
Aplicações de Patches .....	38
Inspetor de Objetos .....	40
Log do Repositório .....	41
Gerente de Projetos .....	43
Execução dos Programas .....	44
Executar .....	45
Pausa da Execução .....	48
Parar a Execução.....	48
Ponto de Parada (BreakPoint).....	49
BookMarks .....	51
Percorrer Linha .....	52

Pular linha.....	53
Executar até o cursor .....	54
Animação .....	55
Para animação .....	55
Acelera animação .....	55
Desacelera animação .....	55
Ferramentas .....	57
Assistentes de códigos .....	58
Assistentes de conversões .....	61
Gerenciador de dados.....	64
Verificação de integridades.....	67
Desfragmentar repositório .....	68
Configuração de ferramentas.....	68
Análises de variáveis, tabelas e campos.....	70
Comandos.....	71
Watches .....	72
Break point´s .....	72
Pilha de chamadas (call stacks) .....	74
Variáveis .....	75
Tabelas e campos.....	76
Desenho de interface.....	77

---



---



---



---



---



---

## **Visão Geral do Curso**

---

Este curso foi elaborado para capacitar os analistas e programadores do Protheus a utilizarem os recursos da Ferramenta de Desenvolvimento DevStudio da MICROSIGA, para que seja possível o desenvolvimento de rotinas personalizadas referentes à customizações futuras dentro do Protheus.

## Objetivos do Curso

---

O objetivo deste curso é ensinar os futuros desenvolvedores a utilizarem por completo a Ferramenta de Desenvolvimento DevStudio, com todos os recursos oferecidos.

---

---

---

---

---

## O que é o DevStudio?

---

O DevStudio é um ambiente de desenvolvimento integrado que acompanha o Protheus, permitindo ao usuário editar, compilar e depurar programas escritos na Linguagem de Programação ADVPL.

Como Ferramenta de Edição, possui todos os recursos das ferramentas mais populares, como Cortar e Colar, Levar o Cursor até determinada linha do código, Localização e Substituição de Texto, etc., e recursos adicionais, como Indentação de Código, Inserção de Comentários de Documentação, etc.

Como Ferramenta de Debug, dispõe de ações de Debug como Percorrer Linha, Pular Linha, Executar, Seguir até o Retorno, Pausar Execução, Derrubar Client, etc., permitindo ao usuário executar e depurar suas rotinas de dentro de seu ambiente integrado, inspecionando o ambiente de execução de suas rotinas através de diversas janelas de informações, como variáveis (divididas entre variáveis locais, variáveis públicas, variáveis privadas e variáveis estáticas), expressões em Watch, Tabelas, Índices e Campos, Break Points, Programas Registrados (Inspetor de Objetos) e Pilha de Chamadas.

Além disso, os programas criados são compilados diretamente do DevStudio, em que são mantidos em Projetos e Grupos de Projetos.

Os Grupos de Projetos facilitam a compilação de um ou mais projetos de arquivos, utilizando conceitos de Repositórios e Diretivas de Compilação, possibilitando inclusive a manutenção de bibliotecas de rotinas do usuário.

## Configuração

---

Neste capítulo, aprenderemos como deverão ser feitas as configurações básicas do DevStudio para um melhor aproveitamento de seus recursos.

- Preferências;
- Configurações.

---

---

---

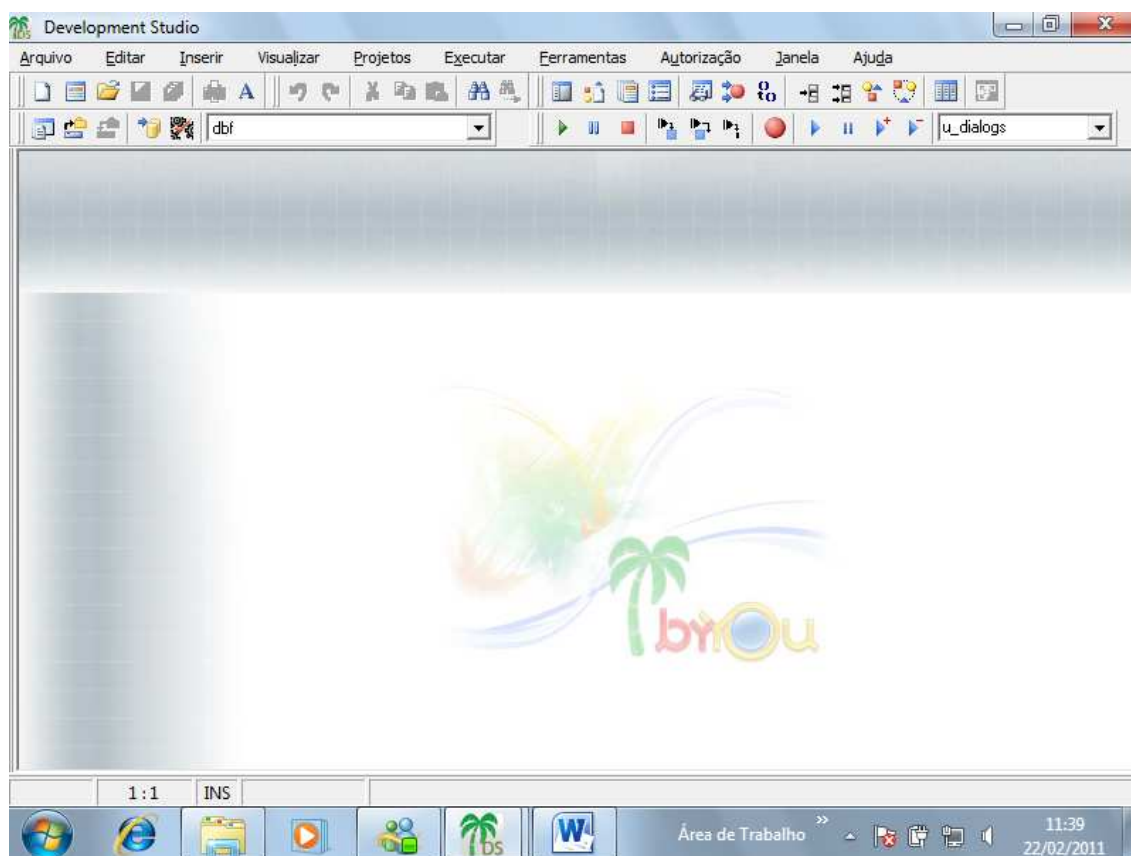
---

---

## Preferências

Na opção de Preferências é onde são definidas as informações que influenciam no comportamento do DevStudio, ou seja, o perfil de teclas utilizado, cores dos fontes, tabulação, etc.

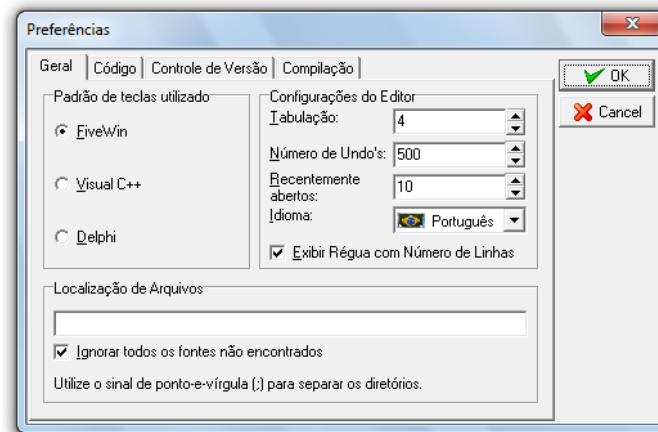
Para configurar Preferências do DevStudio, acesse a ferramenta de desenvolvimento DevStudio;



Selecione “Arquivo”, “Preferências”;

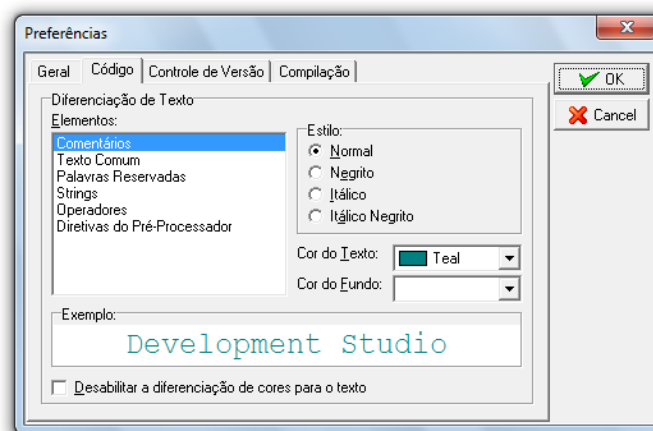
Na pasta “Geral”, na opção “Padrão de teclas utilizado”, selecione o padrão de telas desejado;





Marque a opção “Ignorar todos os fontes não encontrados”, que está localizada na parte inferior;

Na pasta “Código”, verifique as cores que serão utilizadas durante a “Digitação do Fonte”;




---

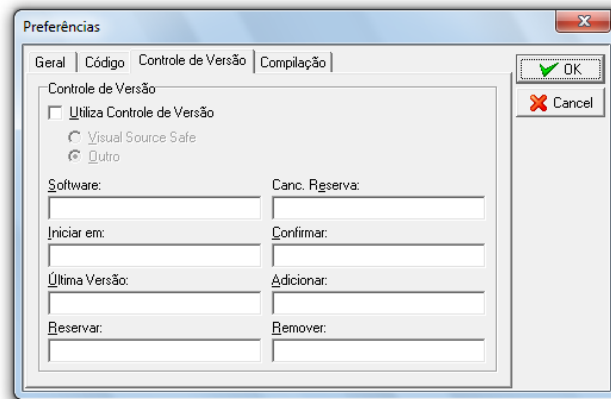
---

---

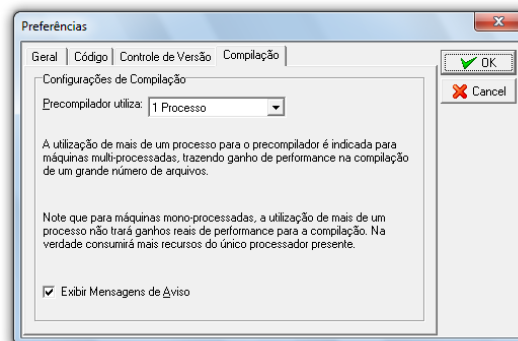
---

---

Na pasta “Controle de Versão”, informe o caminho do “Software”, utilizado para realizar o controle de versões dos fontes, caso seja utilizado algum;



Na pasta “Compilação”, deverá ser informado o número de processadores existentes no servidor de aplicação, para que haja ganhos de performance durante a compilação dos programas;



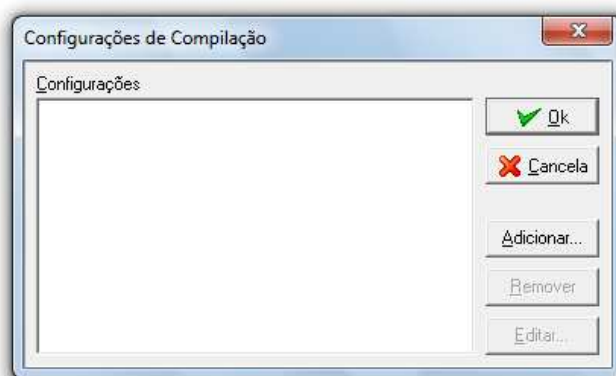
### ***Exercício***

1. Teste a padrão de teclas “Delphi”;
2. Marque a opção “Ignorar os fontes não encontrados”;
3. Volte a opção “Fivewin” para o padrão de teclas.

## Configurações

Após a definição de todas as Preferências que serão utilizadas, deveremos então configurar o Ambiente de Trabalho que será utilizado pelo compilador, ou seja, qual o caminho que deverá ser usado pela Ferramenta de Desenvolvimento DevStudio, para atualizar um determinado RPO durante a Compilação dos Fontes.

Para Configurar o DevStudio, selecione “Arquivo”, “Configurações”;



Clique na opção “Adicionar”;

No Campo “Descrição:”, informe o “Nome do Ambiente”, que será utilizado;

Na pasta “Compilação”, informe no Campo “Ambiente”, o “Nome do Ambiente”, que será utilizado pelo Compilador;

Deverá ser um ambiente válido, no arquivo de inicialização do AppServer (AppServer.INI);

No campo “Conexão:”, informe o “Protocolo Válido”, para a “Conexão com o AppServer”, que deverá ser “TCP”;

No campo “Diretório de Includes”, informe o “Caminho das Pastas”, onde se encontram os “Arquivos de Cabeçalhos de Programas (\*.CH)”;

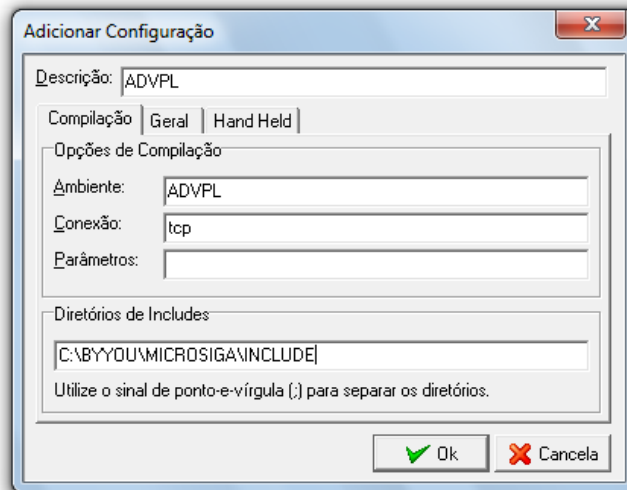
---

---

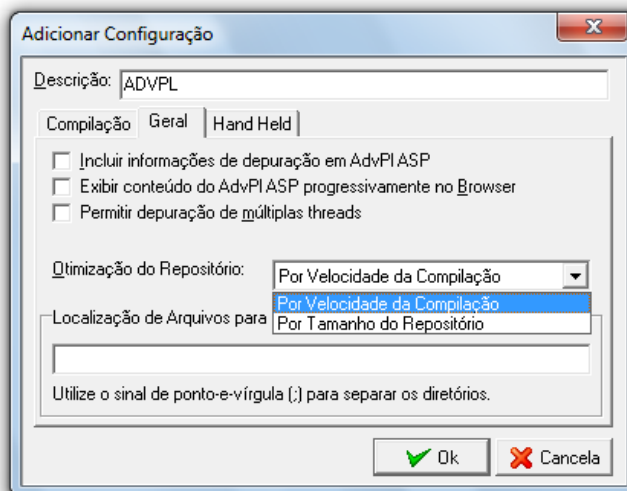
---

---

---



Na pasta “Geral”, posicione com o cursor sobre o campo “Otimização do Repositório” e selecione qual será a maneira que o compilador irá atualizar o repositório de objetos, as opções são: “Por Tamanho do Repositório” ou “Por Velocidade da Compilação”;

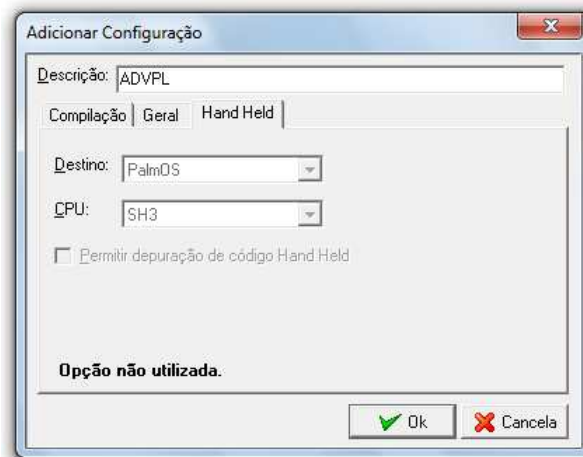


Se selecionada a opção “Por Tamanho do Repositório”, o mesmo ficará com um tamanho reduzido, pois o compilador irá alocar as “Funções” de acordo com os espaços existentes entre uma função e outra dentro do repositório, porém, a compilação irá demorar mais,

pois será necessário maior tempo para a indexação das mesmas dentro repositório;

Caso seja selecionada a opção “Por Velocidade de Compilação”, a compilação dos programas será bem mais rápida, porém o RPO ficará com um tamanho maior em relação à opção anterior, pois conforme as funções forem geradas, o compilador apenas irá inserí-las no repositório, normalmente no final deste, caso não encontre espaço suficiente para alocá-las entre uma função e outra, ou seja, o arquivo ficará fragmentado.

A pasta “Hand Held” não é mais utilizada.



### ***Exercício***

1. Configure um ambiente de compilação baseado nas definições do AppServer.ini.

---

---

---

---

---

## **Criação e Edição de Programas**

---

Neste capítulo, aprenderemos quais os procedimentos corretos, para a elaboração de novos programas e suas respectivas manutenções.

### **Ferramenta de Desenvolvimento DevStudio:**

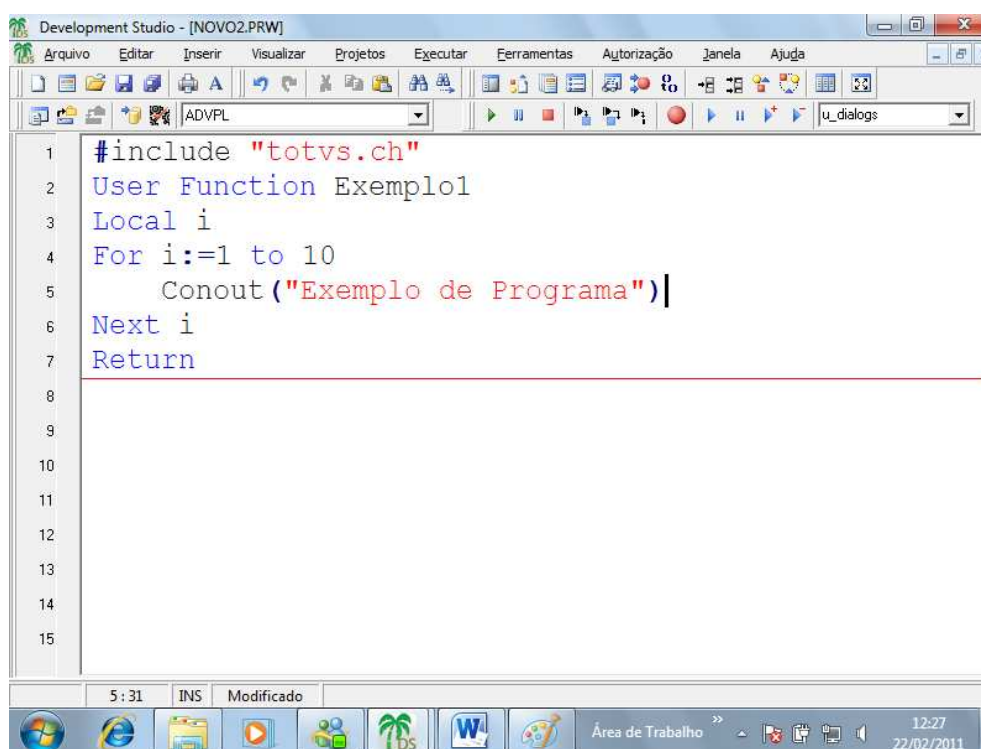
- Novo Arquivo;
- Character ASCII;
- Documentação de Explicação;
- Documentação de Código;
- Documentação de Cabeçalho;
- Localizar;
- Repetir Procura Abaixo;
- Repetir Procura Acima;
- Substituir;
- Indentação;
- Minúsculo;
- Maiúsculo;
- Duplicar Linha;
- Ir para a Linha;
- Ir para a Linha em Execução.

## Novo Arquivo

A opção Novo Arquivo irá possibilitar a edição de novos programas, dentro do DevStudio.

Para criar Novos Arquivos no DevStudio, selecione “Arquivo, “Novo Arquivo” ou clique no botão “Novo Arquivo”, disponível na barra de ferramentas;

Nesta página de edição é possível digitar o programa fonte;



```
1 #include "totvs.ch"
2 User Function Exemplo1
3 Local i
4 For i:=1 to 10
5     Conout ("Exemplo de Programa")|
6 Next i
7 Return
```

Selecione “Arquivo”, “Salvar” ou clique no botão “Salvar”, disponível na barra de ferramentas e salve no diretório previamente criado para guardar os programas fonte e projetos.

Note que após salvar o programa, automaticamente o mesmo será salvo com a extensão “PRW”, que significa que o fonte pertence ao ADVPL;

---

---

---

---

---

Procure sempre salvar seus programas na pasta “My Projects”, localizada no diretório raiz, essa pasta é exclusiva para este propósito.

### ***Exercício***

1. Crie um novo arquivo para o fonte abaixo:

```
#Include “totvs.ch”
```

```
User Function Exemplo1
```

```
Local nI
```

```
For nI:=1 to 10
```

```
    Conout(“Exemplo de Programa.”)
```

```
Next
```

```
Return
```

2. Salve o programa fonte como Exemplo1.prw.



Para uma melhor visualização do texto informado durante a elaboração de um programa, utilize a fonte “Ms Line Draw”, através das seguintes opções ”Arquivo” + ”Escolha de Fonte”, selecionando a fonte citada.



## Caracter ASCII

Tem como finalidade inserir um caracter no padrão ASCII, no fonte dos programas, caso isso seja necessário.

Para utilizar a Tabela ASCII, selecione “Arquivo”, “Novo Arquivo”. Clique em “Inserir” , ”Caracter ASCII” e selecione algum caracter da tabela;

Note que o caracter será inserido no fonte;

Car	Dec	Hex
	0	0
□	1	1
□	2	2
□	3	3
□	4	4
□	5	5
□	6	6
□	7	7
□	8	8
□	9	9
□	10	A
□	11	B
□	12	C

Para inserir dê um duplo clique no caracter desejado.

### *Exercício*

1. Inicie um novo fonte, inclua os caracteres ASCII 65, 167 e 166;
2. Feche o fonte, sem salvá-lo.

---

---

---

---

---

## Documentação de Explicação

---

Esta opção tem por finalidade criar um comentário explicativo que poderá ser utilizado como meio para detalhar as etapas do programa, também cria um Log referenciando os pontos do programa que foram documentados.

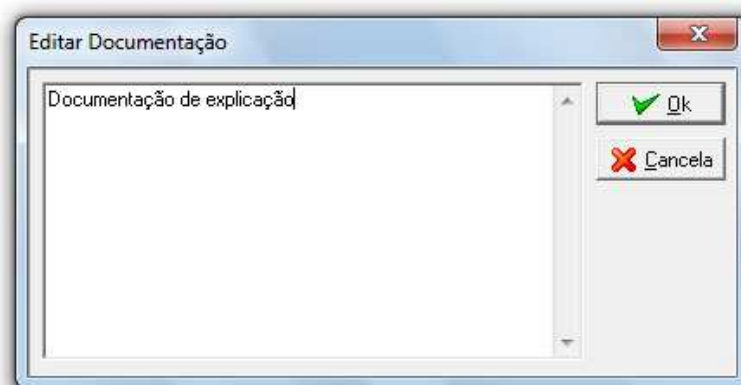
Para utilizar documentações de explicação, selecione “Arquivo”, “Novo Arquivo”

Ou clique no botão “Novo Arquivo”, disponível na barra de ferramentas;

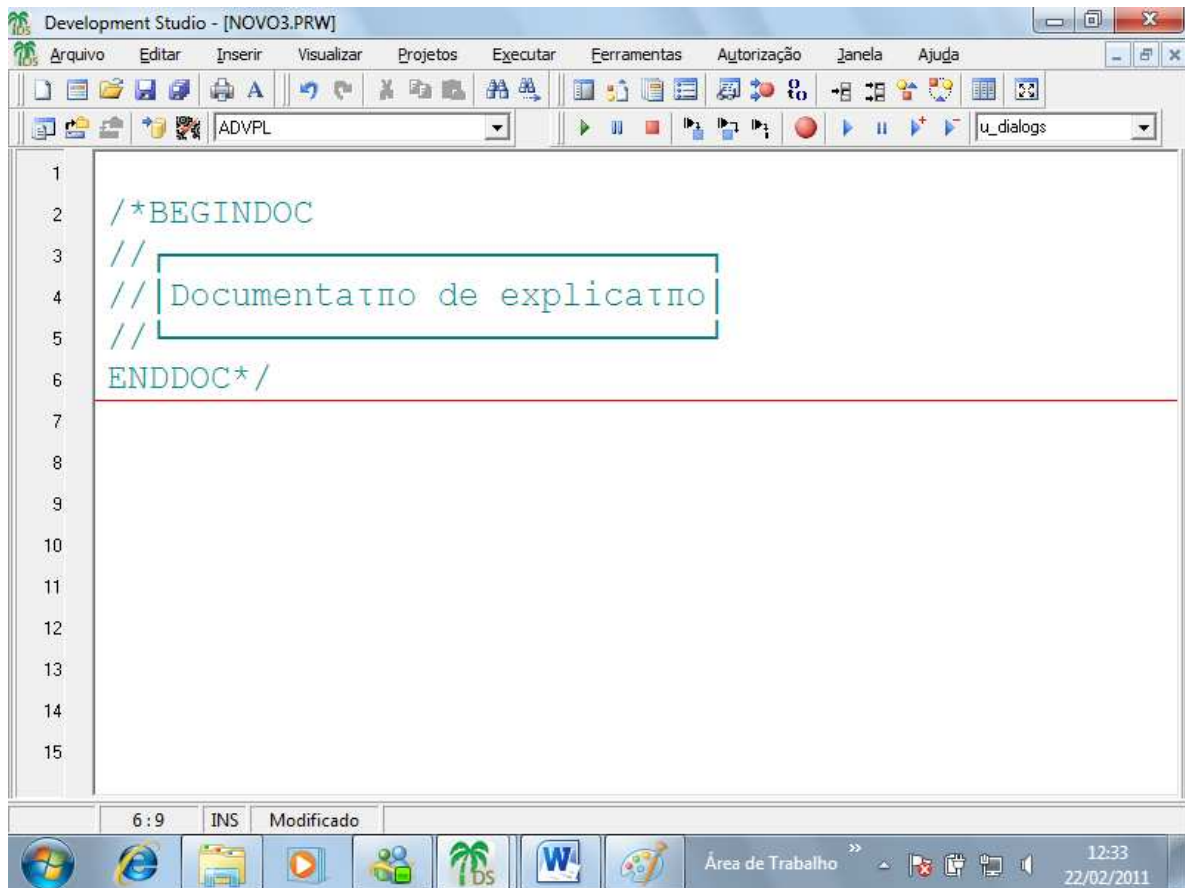
Clique em “Inserir” + ”Documentação de Explicação”;

No Box que foi exibido, informe a mensagem a seguir:

“Teste de programa, utilizando a Documentação de Explicação”.



Confira os dados e confirme;



Verifique que o comentário foi gravado no editor, iniciando com “BEGINDOC” e encerrando com “ENDDOC”.

### *Exercício*

1. Insira 3 documentações de explicação num novo fonte.

---

---

---

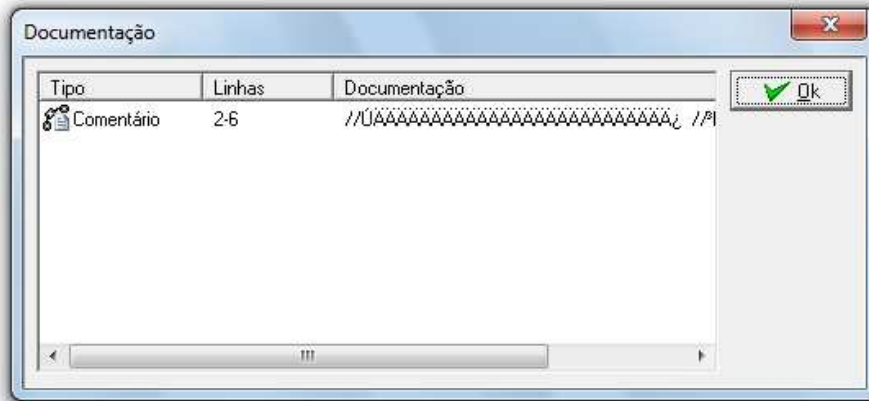
---

---

## Documentação de Códigos

A Documentação de Códigos é utilizada na sequência da Documentação de Explicação, pois nessa opção encontram-se gravados todos os pontos onde foram inseridos os comentários.

Para utilizar Documentações de Código, selecione “Ferramentas”, “Documentação de Código”;



Verifique que o comentário, digitado na documentação de explicação, foi gravado nesta opção, além disso, as linhas em que esse comentário foi inserido também está sendo informada.

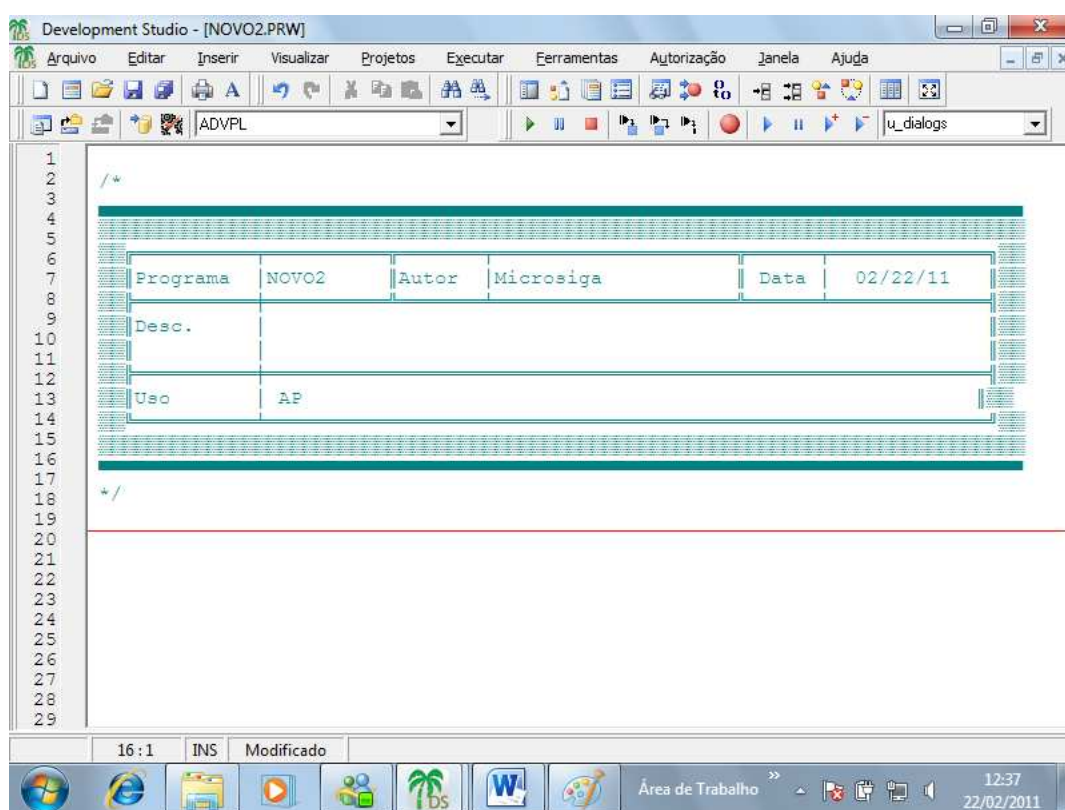
### *Exercício*

1. Visualize as documentações incluídas anteriormente.

## Documentação de Cabeçalhos

A opção “Documentação de Cabeçalhos” deverá ser utilizada quando for iniciado um programa, pois dessa maneira poderemos deixar documentado todos os detalhes daquele fonte, como Nome do Programa, Autor, Data e Finalidade.

Para utilizar clique em “Inserir”, ”Documentação de Cabeçalho”;



### *Exercício*

1. Num novo fonte, inclua uma documentação de cabeçalho;
2. Observe a alteração de fontes entre MsLineDraw e Courier New.

---

---

---

---

---

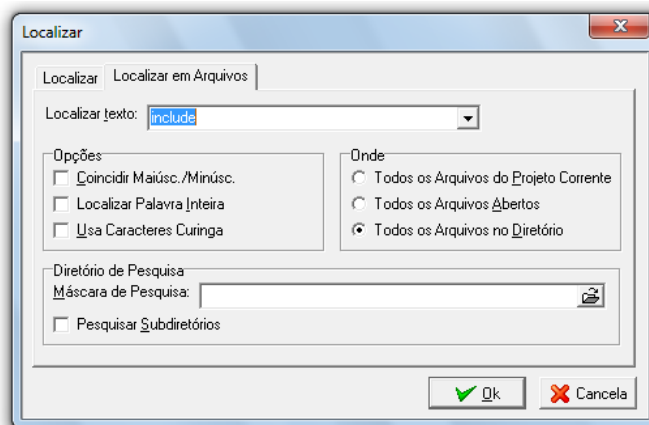
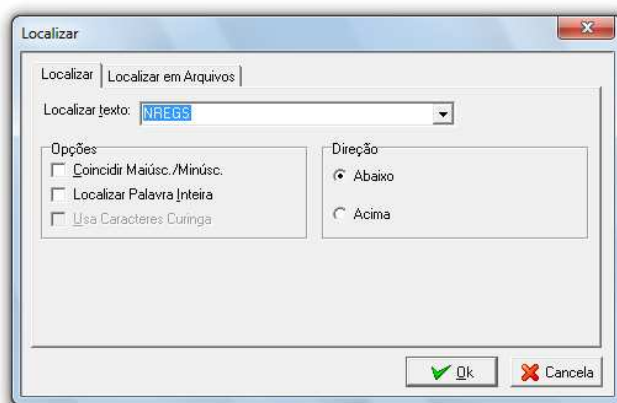
## Localizar

Para localizarmos uma determinada palavra dentro de um código-fonte aberto, poderemos utilizar esta opção.

Ela possui diversos tipos de pesquisas bastante criteriosas.

Como por exemplo coincidir as palavras maiúsculas e minúsculas, realizar a pesquisa abaixo ou acima do posicionamento do cursor, localização em pastas e subpastas.

Para utilizar este recurso, selecione “Editar”, “Localizar”.



### **Repetir Procura Abaixo**

Deverá ser utilizada quando houver a necessidade de se encontrar uma determinada palavra dentro do texto digitado, lembrando que essa pesquisa será feita à partir da linha onde localiza-se o Cursor para Baixo.

### **Repetir Procura Acima**

Deverá ser utilizada quando houver a necessidade de se encontrar uma determinada palavra dentro do texto digitado, lembrando que essa pesquisa será feita a partir da linha em que localiza-se o Cursor para Cima.

Para utilizar este recurso, selecione “Editar”, “Repetir Procura Abaixo” ou pressione “F3”.

## **Substituir**

Quando houver a necessidade por uma substituição de palavras no programa, poderemos optar por utilizar a opção substituir.

Aqui encontraremos, ao mesmo tempo, a opção Localizar e Substituir simultaneamente, sendo que os critérios para isto também serão bem detalhados, assim como na opção Localizar.

Para utilizar este recurso, selecione “Editar”, “Substituir”.



---

---

---

---

---

## Indentar

---

Quando quisermos que o fonte digitado seja Indentado (tabulado na estrutura de comandos) deveremos utilizar esta opção.

Para Indentar, selecione “Editar”, “Texto”, “Indentar”;

### *Exercício*

1. Digite o fonte abaixo:

```
#include "totvs.ch"
```

```
User Function Exemplo2
```

```
Local i
```

```
For i:=1 to 10
```

```
If i==3
```

```
MsgAlert("Estou no 3")
```

```
Elseif i==5
```

```
MsgAlert("Estou no 5")
```

```
EndIf
```

```
Next
```

```
Return
```

2. Execute a indentação automática.



Se a opção “Indentar” for utilizada em um fonte com estrutura de comandos incompleta, será exibido erro de estrutura



## Minúsculo

---

Esta opção tem por finalidade transformar um determinado texto marcado em Minúsculo, caso seja necessário.

Para utilizar este recurso, selecione o texto que se deseja que fique em letra Minúsculo, clique “Editar”, “Texto”, “Minúsculo”;

Note que o “Texto”, foi formatado todo com “Letras Minúsculas”.

### *Exercício*

1. Selecione parte do texto do fonte aberto e transforme em minúsculo.

## Maiúsculo

---

Esta opção tem por finalidade transformar um determinado texto marcado em Maiúsculo, caso seja necessário.

Para utilizar este recurso, selecione o texto que se deseja que fique em letra Maiúscula, clique “Editar”, “Texto”, “Maiúscula”;

Note que o texto será formatado com letras maiúsculas.

### *Exercício*

1. Selecione parte do texto do fonte aberto e transforme em maiúsculo.

---

---

---

---

---

## Duplicar Linha

---

Esta opção deverá ser utilizada quando quisermos que uma determinada linha do código seja duplicada logo abaixo, ou seja, terá o mesmo efeito que a opção copiar e colar.

Para utilizar este recurso, selecione “Editar”, “Texto”, “Duplicar Linha”;

A linha será duplicada logo abaixo.

### *Exercício*

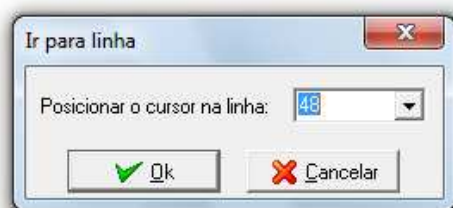
1. Clique numa linha qualquer do fonte e duplique-a;
2. Desfaça a alteração com CTRL+Z.

## Ir para a Linha

---

Deverá ser utilizada quando quisermos que o cursor se posicione em uma determinada Linha do Programa, sem que haja a necessidade do programador ficar navegando pelo texto.

Para utilizar este recurso, selecione “Editar”, “Ir para Linha”;



Informe o número da linha que deseja ir e clique em “OK”;

Verifique que na sequência, o cursor se posiciona na linha indicada.

### *Exercício*

1. Abra o fonte rddemo.prw e posicione na linha 100.

### **Ir para a Linha em Execução**

Deverá ser utilizada quando quisermos que o cursor se posicione na linha que estiver sendo executada pelo DevStudio.

Para utilizar este recurso, selecione “Editar”, “Ir para Linha em Execução”.

---

---

---

---

---

## **Projetos, Compilação, Geração e Aplicação de Patch e Análise do RPO**

---

Neste capítulo, aprenderemos como deverão ser compilados os programas, criação de projetos de trabalho, as possibilidades de compilação, geração e aplicação de patches, verificação e análises de log's do repositório.

### **Ferramenta de Desenvolvimento DevStudio:**

- Projetos;
- Compilar Tudo;
- Compilar Projetos;
- Compilar Pastas;
- Compilar Arquivos;
- Compilação em Batch;
- Geração de Patches;
- Aplicação de Patches;
- Inspetor de Objetos;
- Log's dos Repositórios;
- Templates.

## Projetos

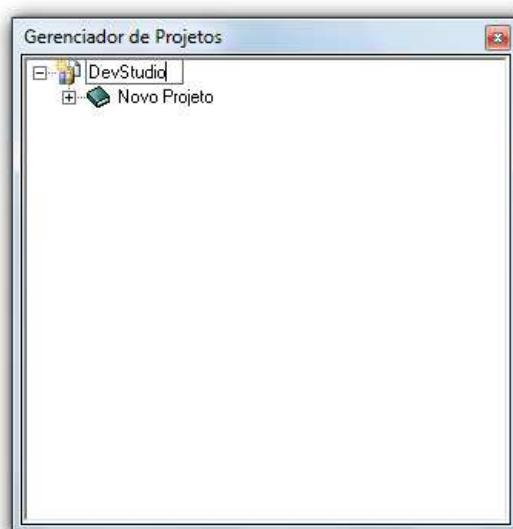
---

Tem por finalidade ajudar o desenvolvedor a administrar, de maneira prática e organizada, os programas envolvidos em uma determinada Customização.

Como por exemplo, um módulo que será customizado poderá ser representado por um Projeto, em que cada Tópico abordado pode ser classificado em uma Pasta diferente dentro do mesmo Projeto.

Para criar Novos Projetos e Adicionar Arquivos, selecione “Projetos”, “Novo” ou clique no botão “Novo Grupo de Projetos”, disponível na barra de ferramentas;

Será apresentado uma tela “Gerenciados de Projetos”, onde deveremos definir o nome do projeto no quadro “Novo Grupo de Projetos”;



Para salvar, clique vá em “Projetos”, “Salvar”;

Logo a seguir, existe outro “Quadro”, em que poderemos detalhar ainda mais o nosso “Projeto”, daremos o nome de “IDE”, clicando com o botão direito do mouse e selecionando a opção “Renomear”;

---

---

---

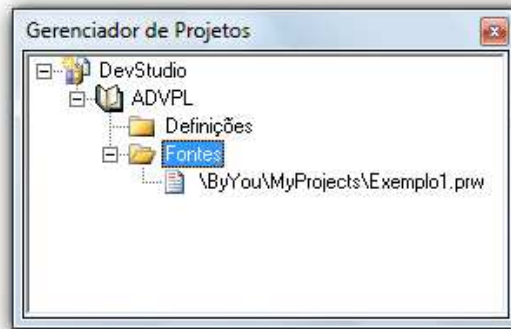
---

---

Agora, deveremos associar o nosso programa ao projeto, clicando com o botão direito do mouse sobre a pasta “Fontes”, selecionando a opção “Adicionar Arquivos”;

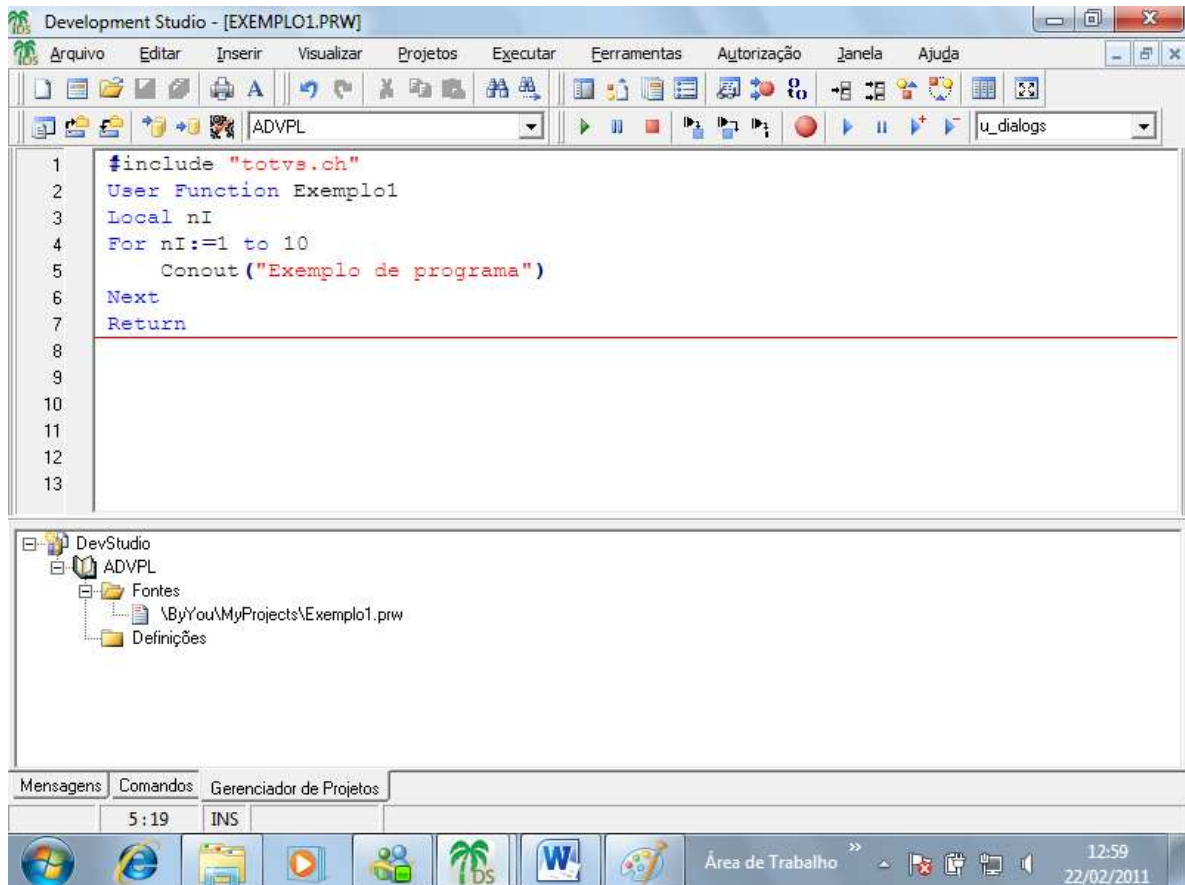
Marque o programa “Exemplo1.prw” e confirme;

Clique na opção “+”, da pasta “Fontes” e observe que ele acabou de ser inserido dentro da mesma;



Selecione “Visualizar”, “Mensagens”;

Clique com o botão esquerdo do mouse sobre a barra de títulos do gerenciador de projetos e arraste-o para o rodapé, até ele fixar-se com a guia “Mensagens” e “Comandos”.



---

---

---

---

---

### ***Exercício***

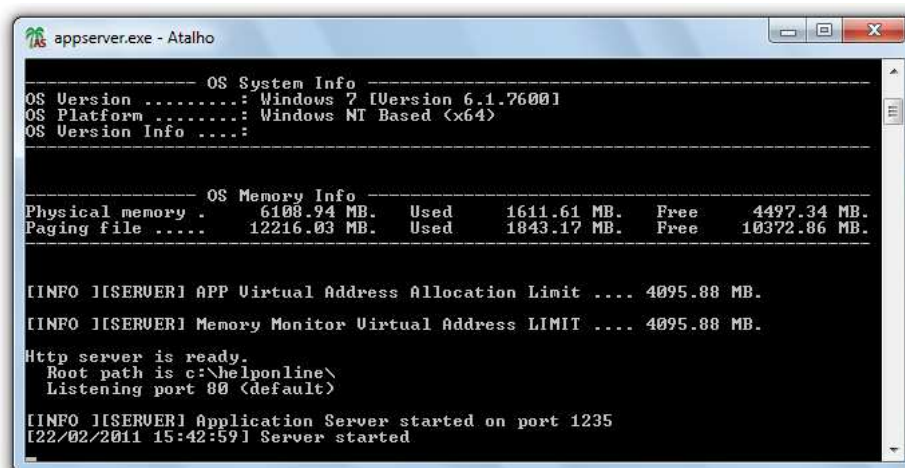
1. Com o programa “Exemplo1” aberto, selecione as seguintes opções: **“Projetos”, “Novo”**
2. Será apresentado o gerenciador de projetos, defina o nome “Curso DevStudio”;
3. Selecione as seguintes opções: **“Projetos”, ”Salvar”**
4. Salve o projeto, com o nome de DevStudio e confirme;
5. Defina o nome do grupo como “ADVPL” e confirme;
6. Selecione a palavra “Fontes”, clique novamente com o botão direito e mande “Adicionar arquivos”;
7. Selecione o programa “Exemplo1.prw” e confirme;
8. Clique na barra de títulos e arraste o gerenciador para a esquerda, até estacionar na lateral do DevStudio, depois pelo centro da tela e até a guia “Mensagens” até se agrupar a ela.



## Compilar Tudo

Utilizado para Compilar todos os programas que se encontram no Gerenciador de Projetos em aberto.

Para utilizar a opção “Compilar Tudo”, carregue o “Aplicativo” do “AppServer”;



Retorne para o DevStudio, clique com o botão direito do mouse sobre qualquer umas das pastas existentes no projeto aberto;

Selecione a opção “Compilar Tudo”, disponível na barra de ferramentas;

Somente administradores podem compilar projetos, no campo “ID do Usuário”, informe “Administrador” e confirme;



---

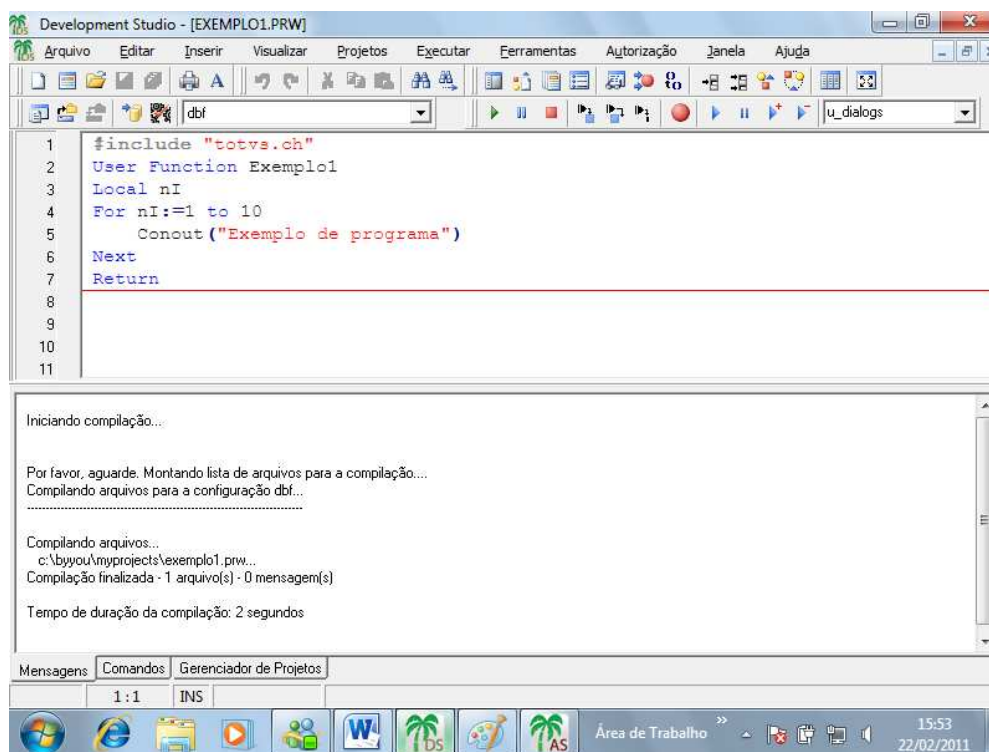
---

---

---

---

Observe que será informada na pasta “Mensagens” a conexão entre o DevStudio e o AppServer.



Caso a compilação não seja finalizada por motivos de erro no programa, este será informado na guia “Mensagens”, juntamente com o número da linha onde encontra-se o erro.

### ***Exercício***

1. Carregue o AppServer, no Desktop;
2. Retorne para o DevStudio, clique com o botão direito do mouse sobre qualquer umas das pastas existentes no projeto aberto;
3. Selecione a opção “Compilar Tudo”, disponível na barra de ferramentas;
4. Informe ID e senha do Administrador e confirme.

## **Compilação de Projetos**

---

Utilizado para compilar todos os programas contidos no projeto selecionado.

Os procedimentos para a compilação são os mesmos utilizados na opção “Compilar Tudo”, alterando apenas a opção “Compilar Tudo” por “Compilar Projeto”.

## **Compilação de Pastas**

---

Utilizado para compilar todos os programas contidos na Pasta selecionada.

Os Procedimentos para a compilação são os mesmos utilizados na opção “Compilar Projeto”, alterando apenas a opção “Compilar Projeto” por “Compilar Pasta”;

## **Compilação de Arquivos**

---

Utilizado para compilar um programa específico contido na Pasta selecionada.

Os Procedimentos para a compilação são os mesmos utilizados na opção “Compilar Projeto”, alterando apenas a opção “Compilar Projeto” por “Compilar Arquivo”, lembrando que é necessário que o cursor esteja posicionado exatamente sobre o programa que deverá ser compilado dentro da Pasta.

## **Compilação em Batch**

---

Utilizado para Compilar em vários ambientes simultaneamente existentes na configuração do DevStudio, todos os programas

---

---

---

---

---

contidos em um determinado Projetos, Pasta ou simplesmente um Arquivo selecionado.

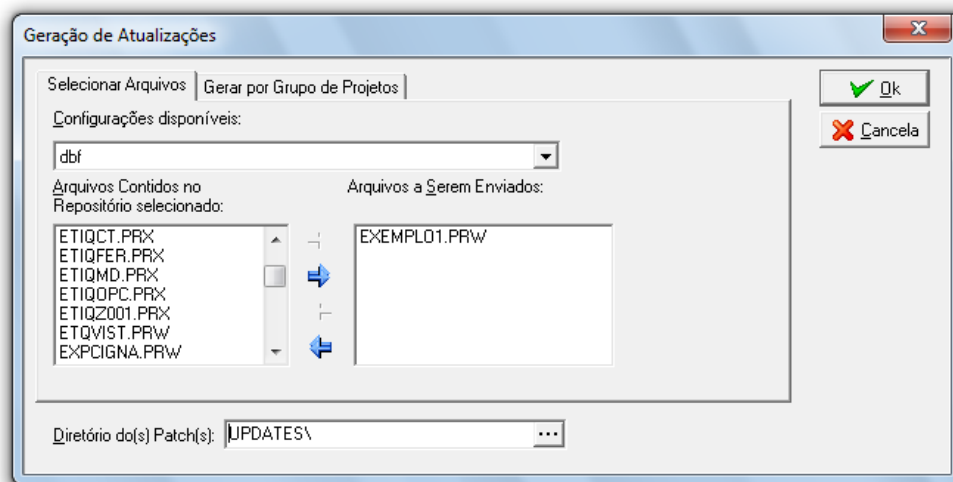
## Geração de Patches

Esta opção é utilizada para a geração de Patches, por meio de programas que já tenham sido compilados através do DevStudio e que se encontrem dentro do RPO utilizado.

Para utilizar a Geração de Patches, selecione “Ferramentas”, ”Geração de Atualizações”;

Será apresentada uma tela, contendo do lado direito todos os programas que se encontram compilados no RPO em uso;

Na pasta “Selecionar Arquivos”, na combobox “Configurações Disponíveis”, selecione o ambiente;

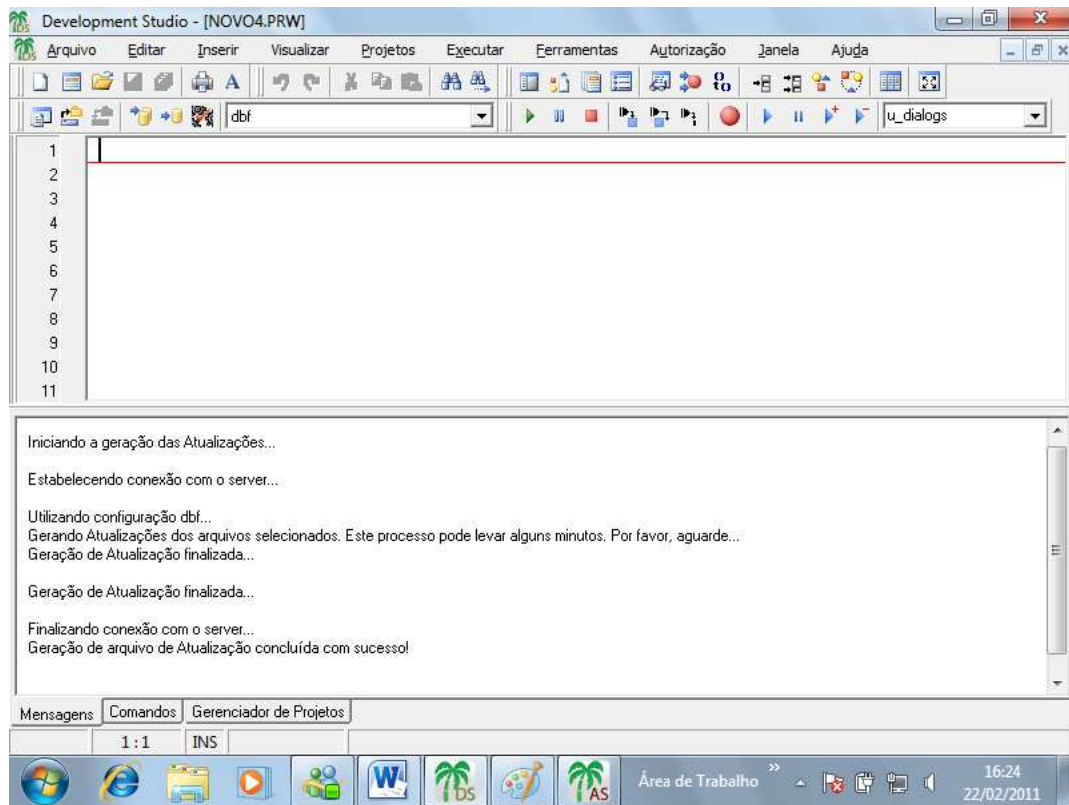


Logo abaixo, procure o programa “Exemplo1.prw” e adicione-o clicando na seta azul, á listbox do lado direito.

No campo “Diretório do(s) Patch(s)”, clique na opção ao lado direito e defina o diretório “UPDATES\”, como padrão para a criação dos arquivos de Patches;

Na sequência confirme a geração da atualização clicando no botão “OK”;

Verifique na pasta de “Mensagens” que o processo foi inicializado e finalizado com sucesso.



### *Exercício*

1. Gere um patch a partir do fonte Exemplo1.prw na pasta \Updates.

---

---

---

---

---

## Aplicações de Patches

A opção geração de patches é utilizada quando houver a necessidade da atualização dos programas existentes no RPO em uso pelo sistema.

Esse processo poderá ser utilizado, para atualizarmos os programas gerados pela MICROSIGA ou programas customizados.

Para utilizar aplicações de patches:

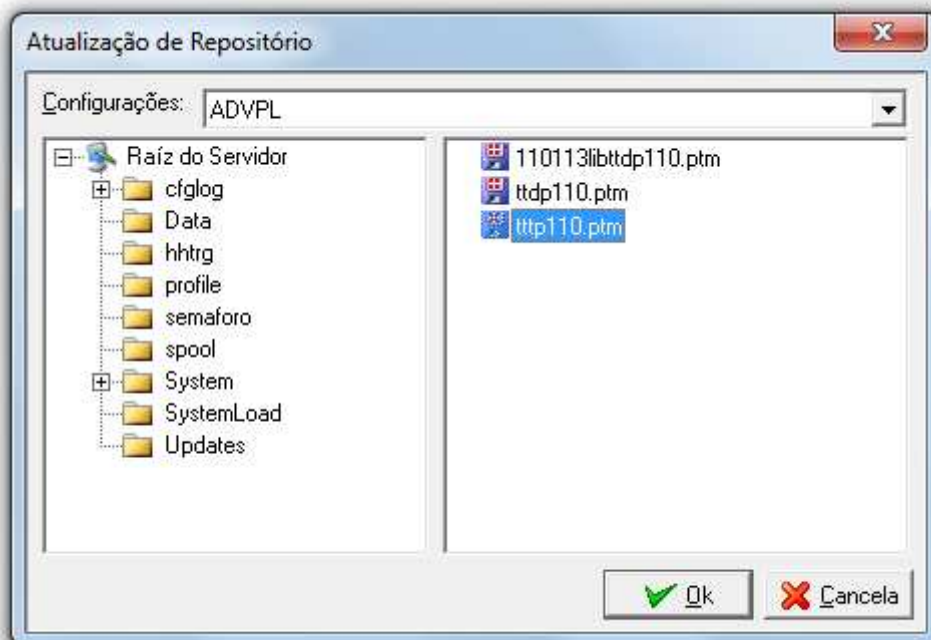
Selecione as seguintes opções: “Ferramentas” , ”Atualização de repositório”;

Será apresentada uma tela, onde devemos informar no campo “configurações” o ambiente desejado.

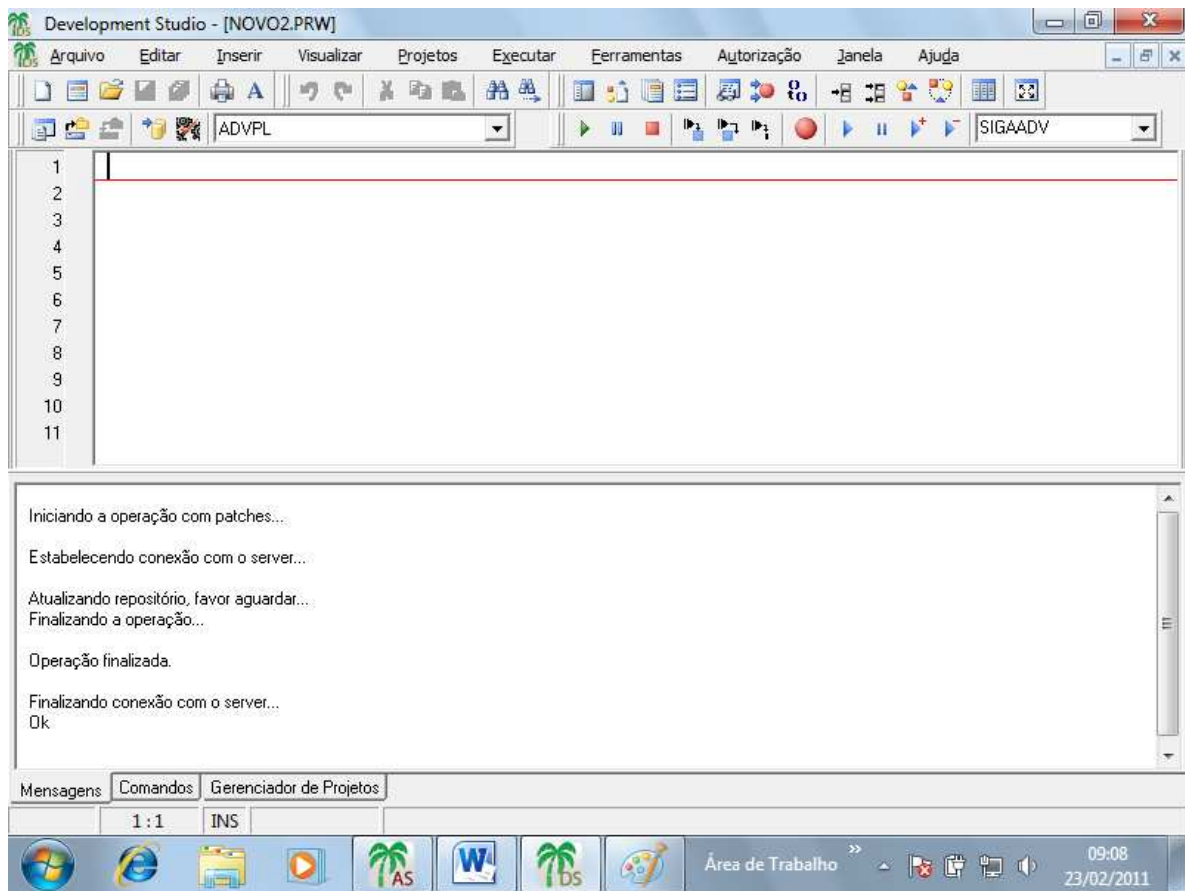
Logo abaixo do lado esquerdo, deveremos informar em que se encontra o patch a ser aplicada;

Selecione as seguintes opções: ”Raiz do servidor”, ”Updates”;

(os patches devem ser gravados dentro de uma pasta do rootpath, nunca no próprio rootpath)



Verifique na guia “Mensagens” que o processo foi inicializado.



### *Exercício*

1. Aplique o patch gerado anteriormente.

---

---

---

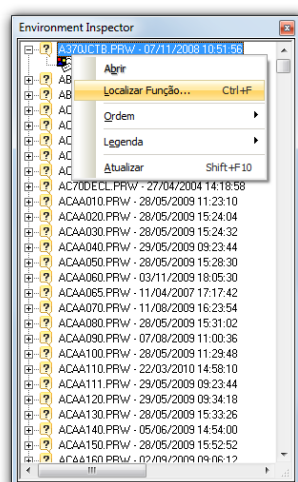
---

---

## Inspetor de Objetos

Utilizado para visualizar todos os Programas (Funções), que se encontram Compilados no RPO em uso.

Para utilizar Inspetores de Objetos, selecione “Visualizar”, ”Inspetor de Objetos”, ou clique no botão “Inspetor de Objetos”, disponível na barra de ferramentas. Verifique que será iniciado o carregamento do RPO;



Em seguida, será apresentada uma lista com todas os arquivos fontes contendo funções, arquivos HTML, ADVPL-ASP e imagens, que se encontram compilados no RPO em uso.

Clique com o botão direito do mouse sobre a tela do “Inspetor de Objetos” e verifique que será apresentada algumas opções de consultas, legendas das funções e tipos de pesquisas.

### *Exercício*

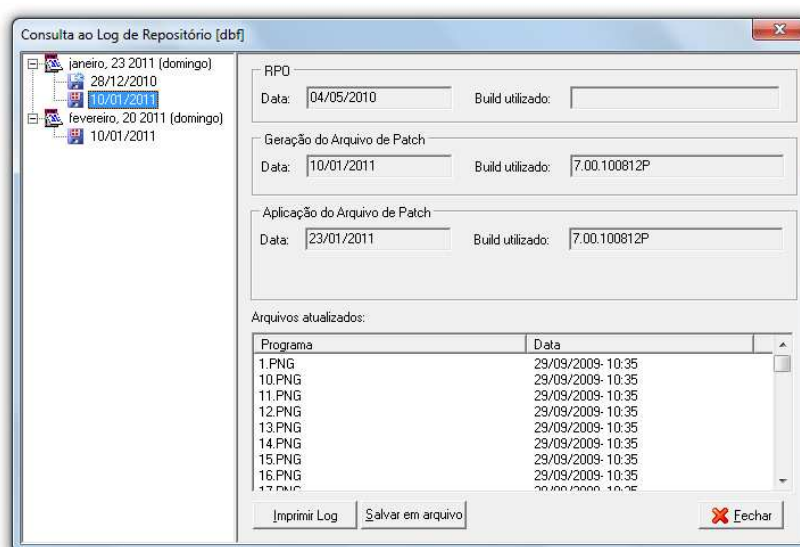
1. Utilize o inspetor de objetos para verificar a última compilação do fonte Exemplo1.prw.



## Log do Repositório

Utilizado para visualizar todo o histórico sobre o RPO em uso, poderemos através desta opção visualizar as datas dos últimos patches aplicados, o conteúdo dos patches e a data da build utilizada.

Para utilizar Log do repositório, selecione "Visualizar", "Log do Repositório" ou clique no botão "Log dos Repositórios", disponível na barra de ferramentas;



No lado esquerdo da tela, visualizamos em seguida posicione o cursor sobre o ícone "Disquete", que será apresentado;

Verifique que ao lado direito da tela será informada a data de geração e aplicação da patch, juntamente com o conteúdo dela e a data da build; caso seja necessário, poderemos imprimir este log, clicando na opção "Imprimir Log", que se encontra abaixo da tela;

---

---

---

---

---

### *Exercício*

1. Verifique quantos patches foram aplicados no RPO atual.

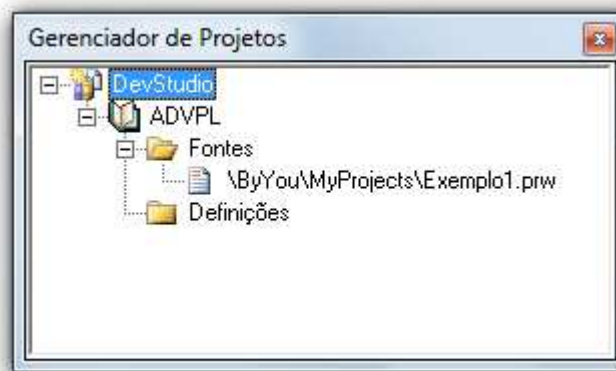
## Gerente de Projetos

---

Esta opção simplesmente habilita a pasta Gerenciador de Projetos e abre o Último Projeto, que estava sendo utilizado pelo DevStudio, caso não haja nenhum Projeto em uso.

Para tanto basta clicar nas seguintes opções:

**”Visualizar”, ”Gerente de Projetos”.**



---

---

---

---

---

## **Execução dos Programas**

---

Neste capítulo, aprenderemos a depurar os programas, executá-los, animá-los, adicionar BreakPoints, BookMarks, entre outros.

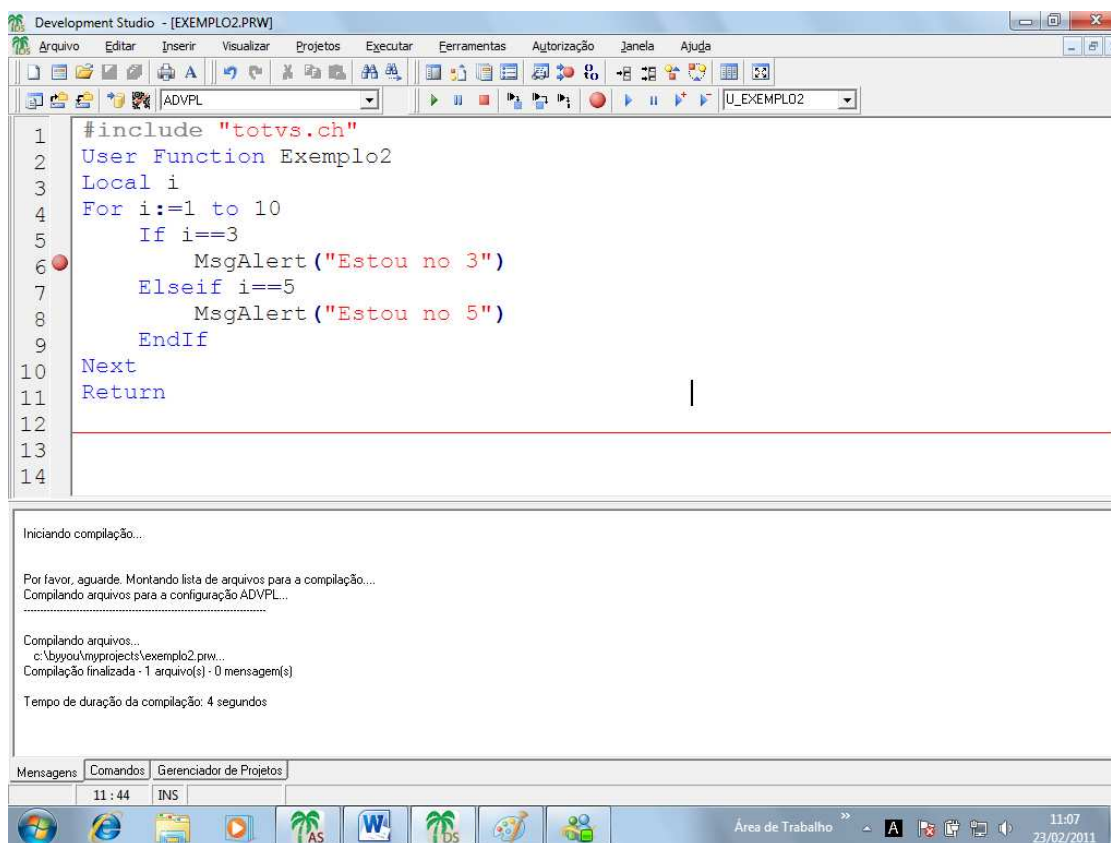
### **Ferramenta de Desenvolvimento DevStudio:**

- Executar;
- Pausa da Execução;
- Para Execução (Derrubar Client);
- Percorrer Linha;
- Pular Linha;
- Executar até o Cursor;
- Ponto de Parada (BreakPoint);
- BookMarks
- Animação;
- Para Animação;
- Desacelera Animação;
- Acelera Animação.

## Executar

Esta opção é utilizada para testar o programa após compilá-lo.

Primeiramente crie um fonte, salve-o, adicione-o ao projeto, salve o projeto e compile-o.



Na barra de ferramentas, ao lado direito da tela, informe o nome da função compilada”, neste caso “u\_Exemplo2”;

Nunca devemos esquecer que todo programa compilado pelo DevStudio deverá ser executado utilizando-se a letra “U\_” antes do nome da função, isso se deve ao fato de estarmos executando uma User Function.

---

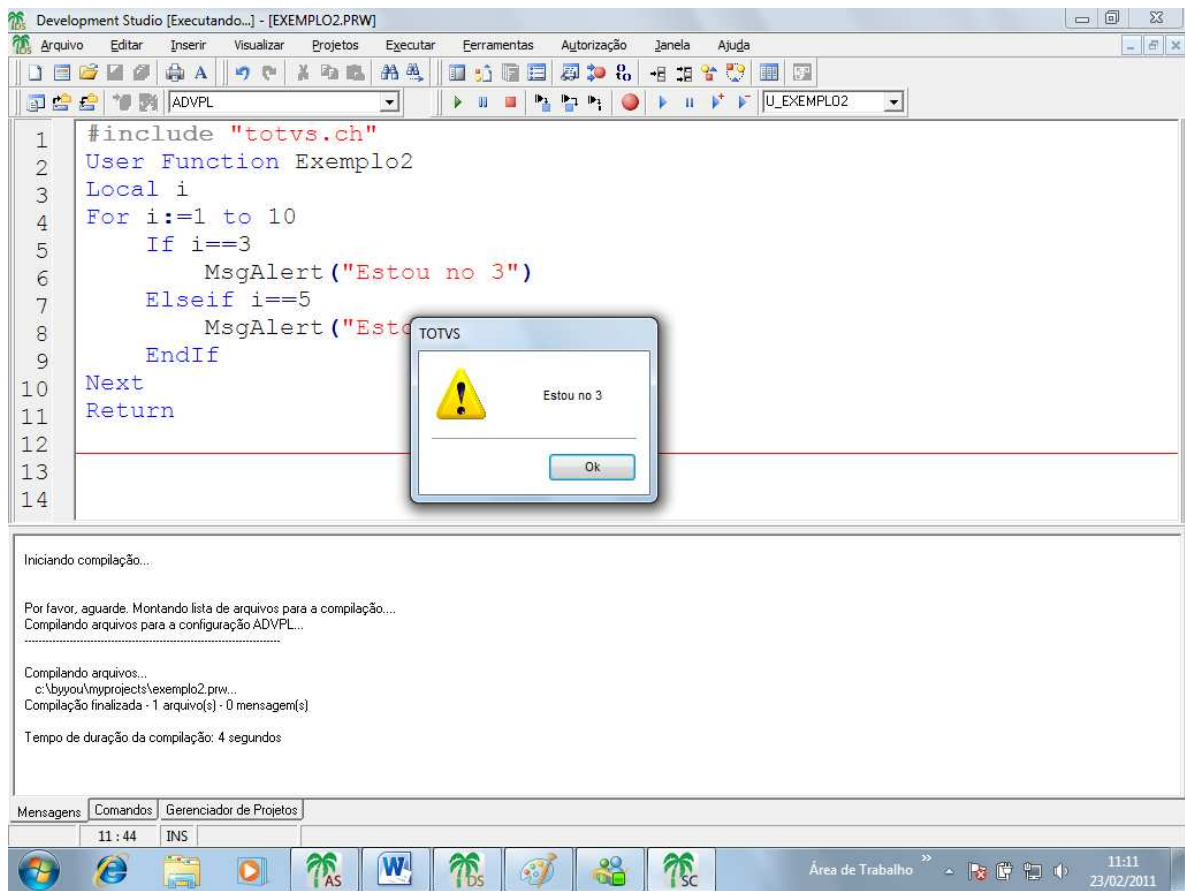
---

---

---

---

Para executar, selecione "Executar" no menu "Executar", o "Programa" será executado na tela.



Confirme as mensagens do programa até a sua finalização.

### *Exercício*

1. Compile o programa abaixo:

```
#Include "Rwmake.ch"
```

```
User Function Exemplo2
```

```
For nI:=1 to 10
```

```
    If nI==3
```

```
        MsgAlert("Estou no 3")
```

```
    ElseIf nI==5
```

```
        MsgAlert("Estou no 5")
```

```
    Endif
```

```
Next
```

```
Return
```

## Pausa da Execução

---

Deverá ser utilizado sempre que for necessário Pausar a Execução de um Programa.

Para tanto, basta iniciarmos a execução de uma determinada Função e selecionarmos: **“Executar”, ”Pausa”**

Ou clicarmos no botão Pausa de Execução, disponível na Barra de Ferramentas.

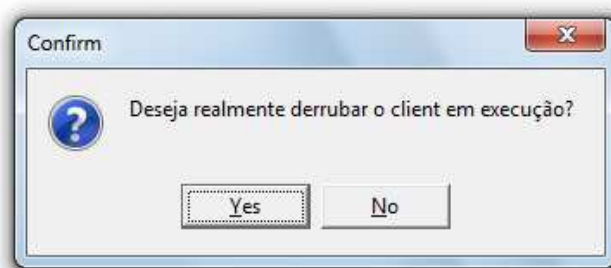
## Parar a Execução

---

Deverá ser utilizado sempre que for necessário “Derrubar a Execução de um Programa”.

Para utilizar a opção “Parar a Execução”: **“Executar”, ”Para Execução”**

Será exibida a mensagem de cancelamento da execução.



### *Exercício*

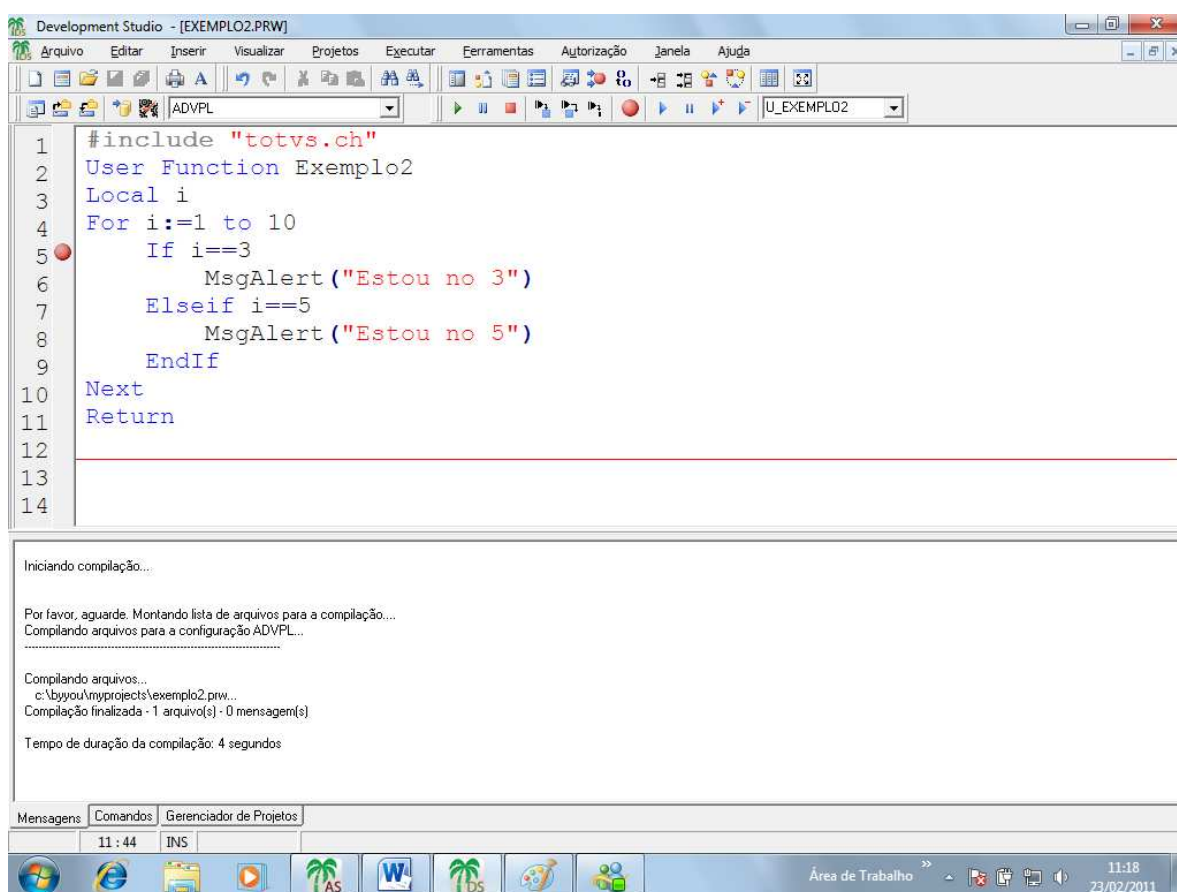
1. No programa anterior, pare a execução ao exibir a primeira mensagem.



## Ponto de Parada (BreakPoint)

Esta opção tem por finalidade definir um ponto de parada durante a execução de um determinado programa, ou seja, aonde quisermos que a execução do programa seja interrompida deveremos inserir um ponto de parada.

Para utilizar a opção ponto de parada (break point), com o cursor posicionado em alguma linha do programa, selecione “Executar”, ”liga/desliga ponto de parada”, ou clique no botão “Ponto de parada”;



Na Sequência, execute o programa;

---

---

---

---

---

Verifique que quando a “Execução” passar pela linha onde se encontra o ponto de parada, este será congelado;

```
1 #include "totvs.ch"
2 User Function Exemplo2
3 Local i
4 For i:=1 to 10
5     If i==3
6         MsgAlert("Estou no 3")
7     ElseIf i==5
8         MsgAlert("Estou no 5")
9     EndIf
10 Next
11 Return
12
13
```

Clique sobre o ponto de parada, removendo-o do programa e execute o programa novamente até a sua finalização.

### ***Exercício***


1. No fonte exemplo2.prw, adicione um ponto de parada na linha 5, execute-o observando a parada, remova o ponto e conclua a execução.

## BookMarks

---

Esta opção tem por finalidade marcar determinados pontos do programa, para facilitar a localização de pontos estratégicos do fonte, durante a sua manutenção.

Para adicionar bookmarks basta pressionar as teclas <ctrl> + <shift> + <número de 0 à 9>, em algum ponto do programa, isso fará a linha onde o cursor está posicionada seja marcada com um bookmark.

```
5      -- - - -
6          MsgAlert ("Estou no 3")
7   | Elseif i==5
8          MsgAlert ("Estou no 5")
9      EndIf
10 Next
11 Return
```

Para remover os bookmarks do programa, basta pressionar a combinação de teclas <Ctrl> + <Shift> + <Número do bookmark>.

### *Exercício*

1. Inclua um bookmark na linha 7 do fonte Exemplo2.prw.
2. Remova o bookmark incluído

---

---

---

---

---

## Percorrer Linha

---

Deverá ser utilizado para que o programa seja executado linha a linha pelo debugador, para isso deveremos adicionar um ponto de parada, no início do programa.

Para utilizar a opção percorrer linha, posicione com o cursor sobre qualquer “linha do programa” e clique em ”executar”, ”liga/desliga ponto de parada”;

Assim que a execução congelar no “ponto de parada”, selecione “executar”, ”executar”;

Clique nas seguintes opções:

“executar” > ”percorrer linha”

Ou clique no botão “percorrer linha”, disponível na barra de ferramentas, simultaneamente até o “fim do debug”.

### *Exercício*

1. Faça a depuração do fonte Exemplo2.prw.

## Pular linha

---

Deverá ser utilizado quando for necessário saltarmos linhas do programa, que estiver sendo executado.

Para utilizar a opção pular linha, posicione com o cursor sobre qualquer “linha do programa” e clique em ”executar”, ”liga/desliga ponto de parada”;

Assim que a execução congelar no “ponto de parada”, selecione: “executar”, ”executar”

Clique nas seguintes opções:

“executar” > ”executar”

Selecione as seguintes opções:

“executar” > ”pular linha”

Ou clique no botão “pula linha”, disponível na barra de ferramentas, simultaneamente até o “fim do debug”.

### *Exercício*

1. No fonte Exemplo2.prw, pule a linha que executaria a segunda mensagem.

---

---

---

---

---

## **Executar até o cursor**

---

Deverá ser utilizado quando quisermos que um determinado programa seja executado apenas até onde o cursor estiver posicionado.

Para utilizar a opção executar até o cursor, posicione com o cursor sobre qualquer linha do programa e clique em "Executar", "liga/desliga ponto de parada";

Assim que a execução congelar no ponto de parada, selecione "Executar", "Executar";

Posicione o cursor numa linha posterior ao ponto de parada, não sendo a próxima;

Clique "executar até o cursor"

### ***Exercício***

1. Simule no fonte Exemplo2.prw o recurso de executar até o cursor.

## Animação

---

Deverá ser utilizado quando quisermos que o cursor acompanhe a execução do programa, dando assim, um efeito de animação, durante o debug do mesmo.

## Para animação

---

Deverá ser utilizado quando quisermos que o programa em animação seja pausado.

## Acelera animação

---

Utilizado para aumentar a velocidade da animação do programa durante o debug.

## Desacelera animação

---

Utilizado para diminuir a velocidade da animação do programa durante o debug.

### *Exercício*

1. Salve o fonte abaixo como Exemplo3.prw:

```
#include "totvs.ch"
```

```
User Function Exemplo3
```

```
Local i
```

```
For i:=1 to 100
```

```
    Conout(i)
```

---

---

---

---

---

Next

Return

2. Compile-o, coloque breakpoint na linha 3, na declaração “Local i”;
3. Execute a animação, parando, acelerando e desacelerando.



## Ferramentas

---

Neste capítulo, aprenderemos como utilizar os recursos do assistente de código, conversão de fontes, gerenciador de dados, verificar a integridade do rpo, patches e configurar atalhos para outros aplicativos.

### Ferramenta de desenvolvimento devstudio:

- Assistentes de códigos;
- Assistentes de conversões;
- Gerenciadores de dados;
- Checar integridades;
- Configurar ferramentas.

---

---

---

---

---

## Assistentes de códigos

Utilizado para ajudar o programador a desenvolver fontes de programas, como por exemplo: cadastros, relatórios, geração e importação de arquivos texto e processamentos genéricos.

Para utilizar assistentes de códigos para relatórios, selecione:

”ferramentas” > ”assistente de código” > ”relatório”



Clique na opção “avançar” em seguida informe o “alias do arquivo” (nome da tabela) e clique na opção “avançar”;



Na próxima tela, informe a chave para parâmetros de relatório e clique na opção “avançar”;



No campo “título”, e clique na opção “finalizar”;



---

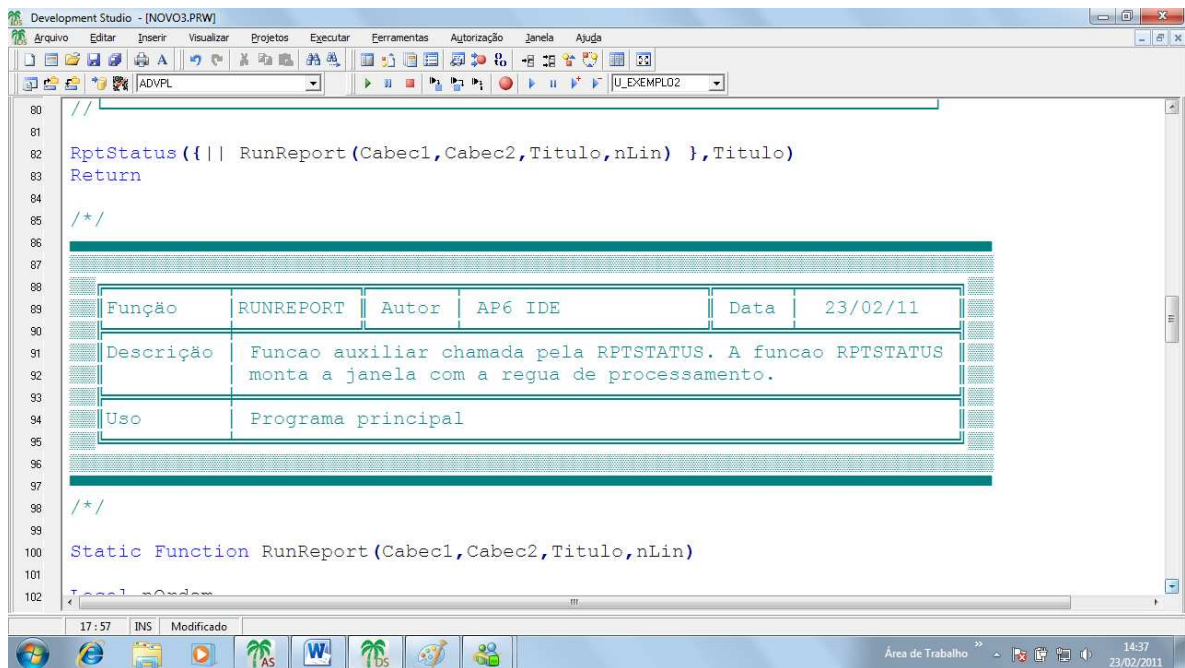
---

---

---

---

Verifique que o fonte do programa foi criado automaticamente com seus devidos comentários.



### *Exercício*

1. Utilize o Assistente de código para criar um relatório de Clientes, exibindo código, loja e nome dos clientes.
2. Utilize o assistente de código para criar um cadastro para a tabela SB1 (produtos).

## Assistentes de conversões

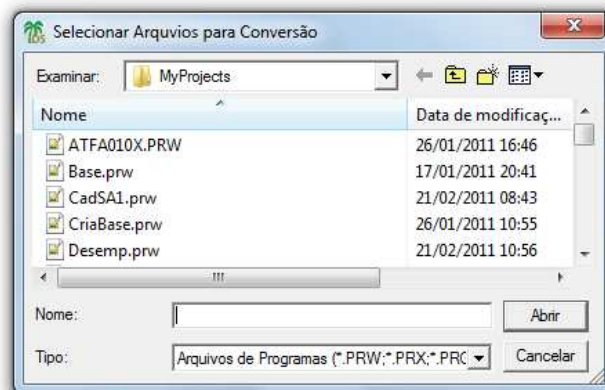
Utilizado para converter os fontes dos programas criados nas versões anteriores do sistema, como a versão siga advanced que possui diferenças em alguns pontos do programa.

Para utilizar assistentes de conversões, selecione: "Ferramentas", "Assistente de conversão". Será apresentada a tela abaixo:

Clique no botão "Adicionar Arquivo" no lado direito da tela:



Escolha o fonte a ser convertido;



---

---

---

---

---

O processo de conversão será iniciado;




Informe o diretório onde o fonte convertido será salvo e clique no botão “Avançar”;



Informe se deseja criar um novo projeto, clique em “Finalizar”;



O fonte convertido poderá ser compilado pelo DevStudio.

	Todos os fontes convertidos pelo assistente de conversão, receberão o suplemento “_ap” após a nome do arquivo.
---	--

---

---

---

---

---

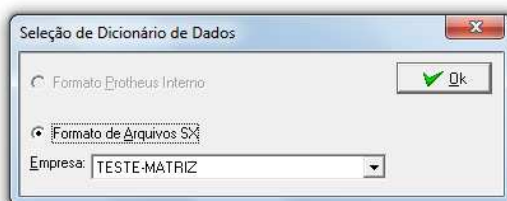
## Gerenciador de dados

Esta opção é utilizada como uma maneira de se obter informações sobre os arquivos e campos existentes no sistema, também será possível analisarmos as estruturas dos mesmos, suas validações e relacionamentos.

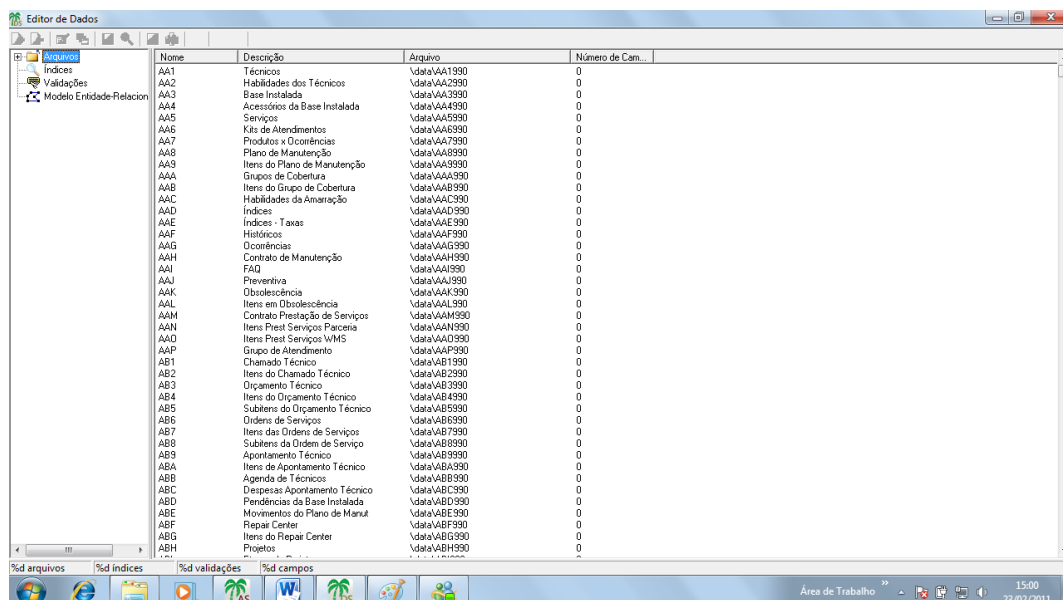
Para utilizar gerenciadores de dados, selecione as seguintes opções:

”Ferramentas”, ”Gerenciador de dados”

Selecione a empresa desejada e confirme;

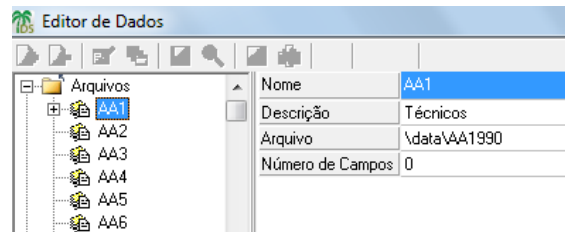


Serão lidos os arquivos SX (dicionário de dados) da empresa escolhida;

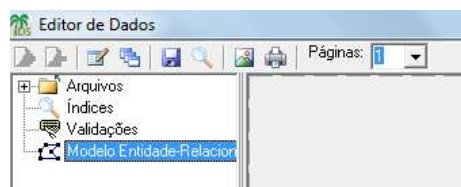




Clique no sinal de “+ Arquivos” à esquerda, a lista de tabelas será expandida abaixo. Selecione um Alias de tabela e será exibido à direita as informações sobre essa tabela.



Para utilizar modelos entidade-relacionamentos, selecione a opção “Modelo Entidade-Relacionamento”, que se encontra ao lado esquerdo da tela;



Clique na opção “Selecionar Arquivos”, localizado na barra de ferramentas, logo acima;

Será apresentada uma tela onde deveremos marcar os arquivos a relacionar;

Marque os seguintes arquivos que deseja e confirme;

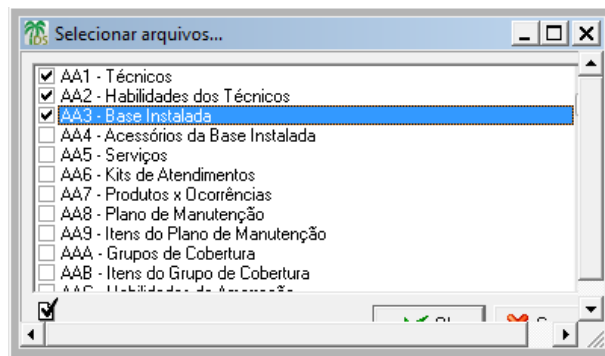
---

---

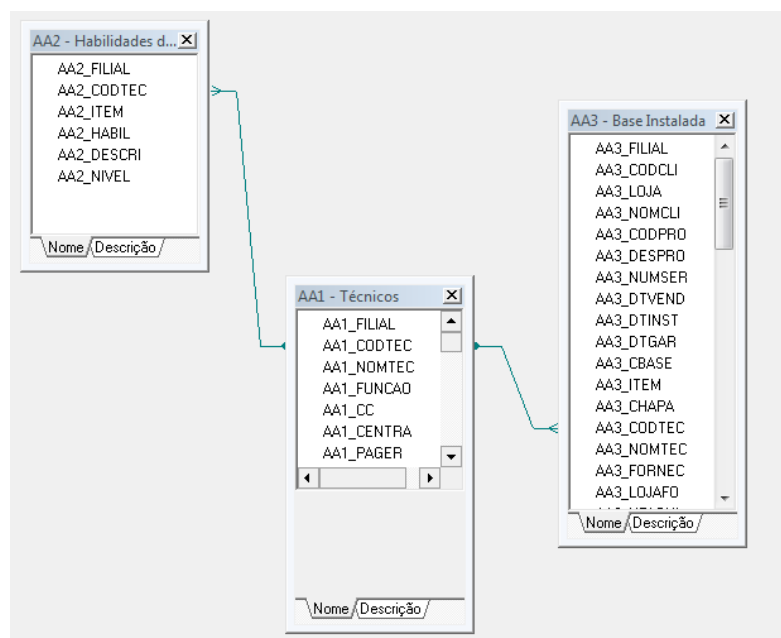
---

---

---



Serão apresentados os arquivos e seus relacionamentos, com um duplo clique sobre as pastas dos arquivos poderemos verificar também quais os campos que se relacionam entre as tabelas e o tipo de relacionamento (1xN, 1x1, etc);



### ***Exercício***

1. Faça um MER entre as tabelas SA1, SC5 e SE1.

## Verificação de integridades

---

Utilizado para verificar a integridade do rpo e patch que estão sendo utilizados pelo sistema.

Para tanto, basta selecionar as seguintes opções "Ferramentas" , "Checar integridade" e localize o arquivo de patch.



Utilize esta opção sempre que for atualizar um RPO ou antes de aplicar uma atualização ao mesmo.

---

---

---

---

---

## Desfragmentar repositório

---

Esta opção somente será válida se o DevStudio estiver configurado como “Por velocidade da compilação”, pois dessa maneira o RPO ficará fragmentado.

Para tanto basta selecionar as seguintes opções “Ferramentas”, ”Desfragmentar repositório”.

---



Deve-se ter acesso exclusivo ao repositório para desfragmentá-lo. Esse processo irá reduzir o tamanho do repositório reescrevendo-o e eliminando espaço ocupado por rotinas antigas..

## Configuração de ferramentas

---

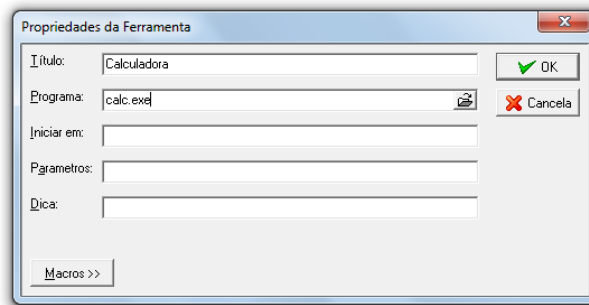
Quando quisermos criar um atalho para algum executável dentro do DevStudio poderemos utilizar esta opção. Ela simplesmente cria uma nova opção no menu de ferramentas, logo abaixo da opção “configurar ferramentas”.

Selecione as seguintes opções: ”Ferramentas”, ”Configurar ferramentas”



Clique em “Adicionar”;

Informe o título a ser exibido no menu, o caminho completo para o executável do aplicativo e demais parâmetros necessários;



### ***Exercício***

1. Faça um atalho para a calculadora (calc.exe) e mapa de caracteres (charmap.exe)

---

---

---

---

---

Neste capítulo, aprenderemos como utilizar o watches, analisar variáveis dos programas, pilha de chamadas, verificar os pontos onde estão localizados os break points, verificar as tabelas e campos utilizados pelo programa e o gerenciador de dados.

### **Ferramenta de desenvolvimento ide:**

- Comandos;
- Watches;
- Break points;
- Pilhas de chamadas;
- Variáveis locais;
- Variáveis privadas;
- Variáveis públicas;
- Variáveis estáticas;
- Tabelas e campos.

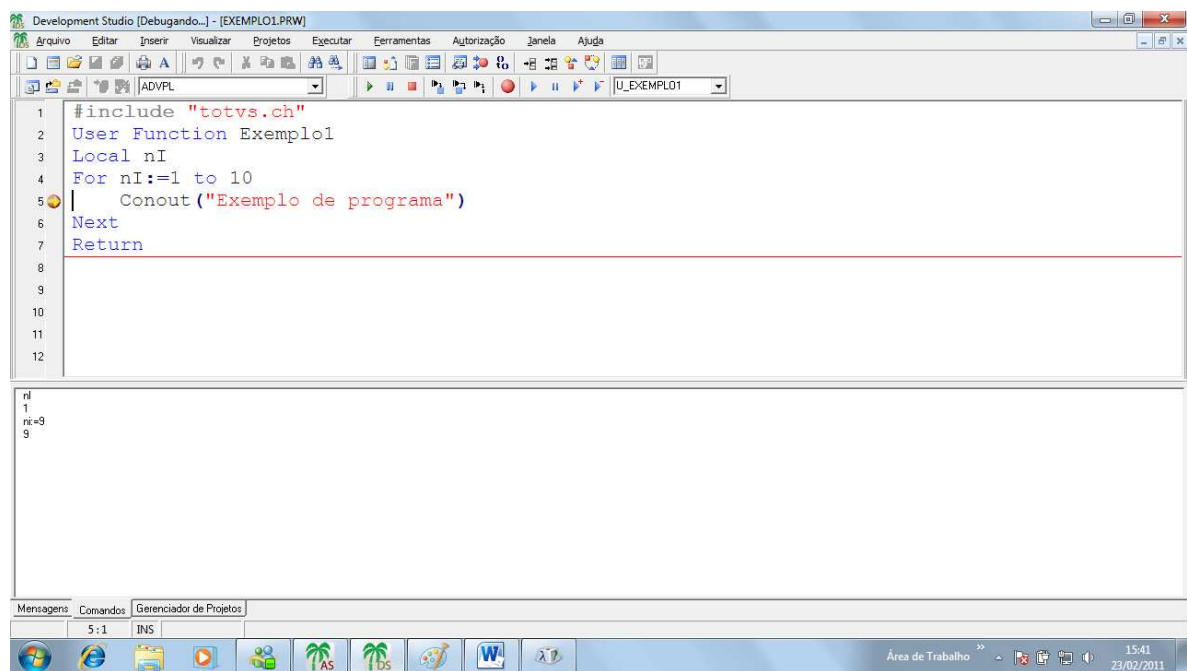
## Comandos

Utilizado para que seja possível analisarmos e alterarmos os valores das variáveis durante o debug dos programas.

Para utilizar a pasta comandos, crie um ponto de parada;

Na pasta comandos digite o valor da variável e pressione <enter>;

Observe que o valor da variável será retornado.



### *Exercício*

1. No fonte Exemplo1.prw, coloque um breakpoint na linha 5. Verifique o valor da variável nI e altere-a à vontade.

---

---

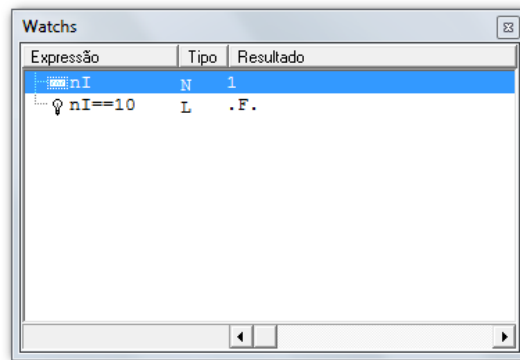
---

---

---

## Watches

Através dessa opção poderemos verificar o valor de variáveis, não permitindo alterá-las, e também incluir expressões ADVPL sem ter que necessariamente declarar uma variável.



### *Exercício*

1. No fonte Exemplo1.prw coloque um ponto de parada, execute o programa, exiba a janela de Watches e verifique o valor da variável nI e das expressões nI==10 e Time().

## Break point's

Esta opção deverá ser utilizada quando quisermos saber onde se encontram todos os pontos de parada que estão marcados no programa em uso.

Poderemos também inserir condições nos pontos de parada, dessa maneira, sempre que o programa estiver em modo de debug, o mesmo somente terá sua execução parada se atender a condição do ponto de parada.



### ***Exercício***

Como utilizar a opção break point:

1. Utilizando o programa “exemplo2”, posicione o cursor na “linha 4 do programa”;

2. Selecione as seguintes opções:

**“executar” > ”liga/desliga ponto de parada”**

3. Selecione as seguintes opções:

**“visualizar” > ”janelas de debug” > ”break points”**

5. No campo “condição”, digite “ni==3” e confirme;

6. Insira um “novo ponto de parada” na “linha 3 do programa”, posicionando o cursor na mesma linha e selecionando as opções:

**“executar” > ”liga/desliga ponto de parada”**

7. Execute o programa por meio das opções:

**“executar” > ”executar”**

8. Assim que a “execução do programa” for congelada no “ponto de parada”, localizado na “linha 3”, remova-o de lá, clicando sobre o “break point” daquela linha;

9. Selecione as seguintes opções:

**“executar” > ”animação”**

10. Observe que quando a “variável – ni” estiver com o “valor – 3”, a “execução” será congelada, atendendo a “condição do ponto de parada condicional”;

11. Encerre a “execução do programa”, selecionando as opções:

**“executar” > ”para execução”**

---

---

---

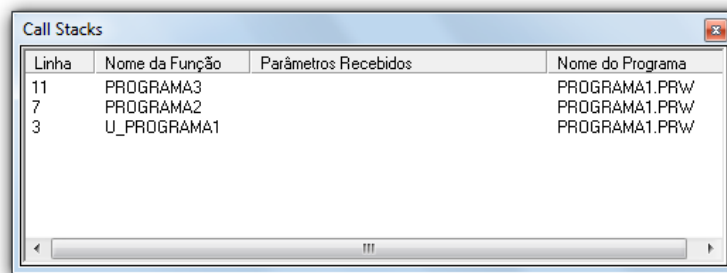
---

---

## Pilha de chamadas (call stacks)

Deverá ser utilizado sempre que quisermos analisar a seqüência das execuções dos programas, com a finalidade de identificar algum erro de lógica dos mesmos.

Serão exibidas as chamadas de rotina, por ordem descentente, ou seja, qual programa foi chamado por qual programa, começando do último.”.



### *Exercício*

1. Copie e compile o fonte abaixo:

```
#include "totvs.ch"
```

```
User Function programa1
```

```
programa2()
```

```
return
```

```
static function programa2
```

```
programa3()
```

```
return
```

```
static function programa3
```

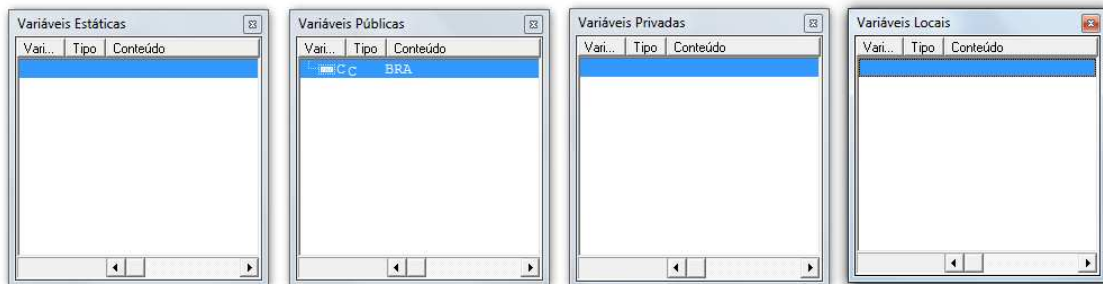
```
msgalert("oi")
```

```
return
```

2. Coloque um breakpoint na linha do msgalert() e verifique a pilha de chamadas.

## Variáveis

Esta opção deverá ser utilizada para demonstrar todas as variáveis do tipo local, privada, pública e estática, que se encontram no programa debugado, no momento.



### *Exercício*

1. Crie um fonte com o código abaixo:

```
#include "totvs.ch"
```

```
User Function Variaveis
```

```
Local cNome:="Sergio"
```

```
Private dDia:=Date()
```

```
Public nSalario:=10000
```

```
Static lCasado:=.T.
```

```
Return
```

2. Compile, coloque um breakpoint no Return e visualize as variáveis.

---

---

---

---

---

## Tabelas e campos

---

Nesta opção teremos a possibilidade de visualizar todos os arquivos (tabelas) e campos utilizados em um determinado programa, será possível também analisarmos as chaves de índices, utilizadas pelos arquivos, número do registro corrente e path do arquivo.



### *Exercício*

1. Copie o fonte abaixo:

```
#include "totvs.ch"
```

```
User Function Mt010Inc
```

```
Return
```

2. Coloque um breakpoint na linha do return, escolha SIGAFAT na combobox de execução na barra de ferramentas, inclua um produto e confirme;

3. Utilize a opção para verificar as tabelas abertas até o momento da execução.

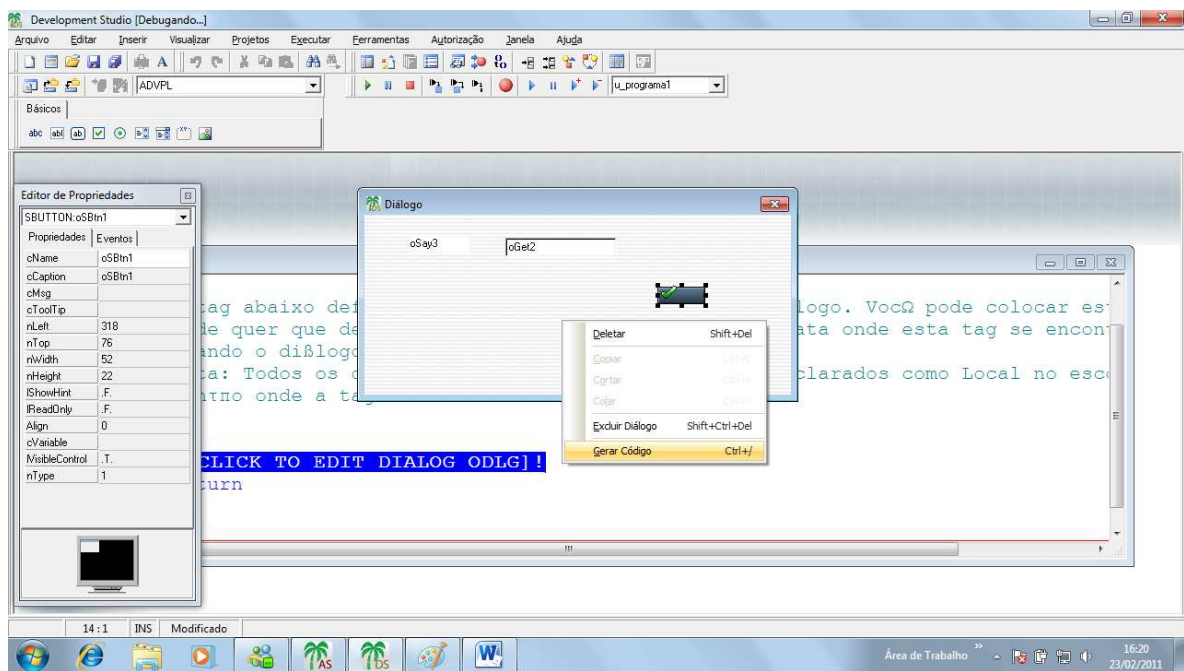
## Desenho de interface

Neste capítulo, aprenderemos como utilizar os recursos do desenhador de telas, utilizado para facilitar as telas de entrada de dados, que trabalham com objetos e diálogos.

Para utilizar o assistente de interface, clique em “Arquivo”, “Novo Dialogo”.

Observe os objetos disponíveis para inclusão na caixa de diálogo na barra “Básicos”. Os nomes dos objetos podem ser trocados, e os métodos executados em cada evento podem ser alterados pelo “Editor de propriedades”.

Ao concluir o layout da DialogBox, clique sobre ela com o botão direito e mande “Gerar código”.



### ***Exercício***

1. Crie uma caixa de diálogo, inclua uma textbox, um label e um button, gere o código, salve o fonte como exemploX.prw, adicione-o ao projeto e compile e execute-o.