

**Aprendendo**

# **REPORT VIEWER**

**do design à implantação**



{  }

**Abc-dev**

**ANDRÉ ALVES DE LIMA**

## **PUBLICADO POR**

Abc-Dev

André Alves de Lima

Nenhuma parte desta publicação pode ser reproduzida, arquivada em um sistema de recuperação, ou transmitida em qualquer forma ou por quaisquer meios, eletrônico, mecânico, fotocópia, gravado ou de qualquer outra maneira sem a prévia e expressa autorização por escrito do autor.

Todas as outras marcas e nomes de produtos são marcas comerciais ou marcas registradas de suas respectivas empresas.

Este livro expressa o ponto de vista e opinião do autor. As informações contidas neste livro são fornecidas "como estão" e sem garantias de qualquer tipo, expressas, implícitas ou estatutariamente sugeridas.

O autor não se responsabiliza por quaisquer prejuízos ou danos materiais ou pessoais que possam advir direta ou indiretamente da utilização da informação disponibilizada, sendo o utilizador o único e exclusivo responsável por todas as decisões tomadas com base nessa informação.

Copyright © André Alves de Lima, setembro de 2015

Todos os direitos reservados

[contato@andrealveslima.com.br](mailto:contato@andrealveslima.com.br)

# Sumário

Introdução .....	1
Instalando o Report Viewer .....	2
Criando seu primeiro relatório .....	6
Acessando o preview de relatórios locais .....	11
Utilizando o Assistente de Relatório (Report Wizard) .....	14
Trabalhando com grupos de linhas .....	18
Trabalhando com grupos de colunas .....	21
Entendendo o layout do relatório .....	24
Data Region .....	24
Cabeçalho e rodapé .....	24
Tipos de controles .....	25
Propriedades do relatório .....	33
Exibindo a régua .....	36
Criando totalizadores .....	38
Classificação (ordenação) de dados .....	44
Classificação interativa .....	45
Filtrando os dados .....	48
Utilizando expressões .....	50
Estruturas de decisão .....	51
Totalizadores .....	52
Datas .....	52
Formatação de strings .....	52
Informações do relatório .....	54
Códigos customizados .....	55
Formatação de dados e formatação condicional .....	56
Trabalhando com grupos .....	60
Grupos de linhas .....	60
Classificação de grupos .....	63
Grupos de colunas .....	64

Adicionando parâmetros .....	68
Ajustando a expressão totalizadora no relatório filtrado .....	73
Mestre-detelhe, SubReports e Drill-down .....	76
SubReports.....	76
Drill-down.....	83
Exibindo imagens.....	86
Imagens incorporadas.....	86
Imagens externas .....	88
Imagens do Banco de Dados.....	91
Criando gráficos .....	96
Trabalhando com colunas (para gerar etiquetas) .....	104
Deployment do controle Report Viewer.....	109
Utilizando o Report Viewer Redistributable .....	109
Copiando as dlls no diretório de instalação .....	110
Publicação com ClickOnce.....	112
Sobre o autor.....	115

# Introdução

A geração de relatórios é uma das tarefas mais importantes no desenvolvimento de aplicações de negócios. Porém, apesar da sua importância, ela é uma atividade quase sempre ignorada, deixada de lado para o *"programador que cuida dos relatórios"*. Caso sua empresa trabalhe com o Report Viewer e você seja esse *"sortudo"* programador (ou talvez o seu substituto no período de férias), esse livro com certeza irá te ajudar, e muito.

Quando falamos em aplicações .NET, a ferramenta de relatórios mais utilizada é, de longe, o Crystal Reports. Se você trabalha com o Crystal Reports, você está acostumado a encontrar facilmente as respostas para suas questões na Internet. Porém, além do Crystal Reports, temos a alternativa gratuita da própria Microsoft, chamada Report Viewer.

Uma grande vantagem do Report Viewer, quando comparado ao Crystal Reports, é que ele pode ser utilizado nas edições gratuitas do Visual Studio (as chamadas edições *"Express"*). E, apesar de não parecer, o Report Viewer permite desenvolver relatórios tão complexos quanto os relatórios do Crystal Reports. O grande problema é que não existe muito material disponível na Internet sobre essa ferramenta (principalmente em português) e, quando existe, ele não está organizado de forma fácil de ser consumida. Foi justamente com o intuito de solucionar essa deficiência que este livro foi escrito.

## Capítulo 1

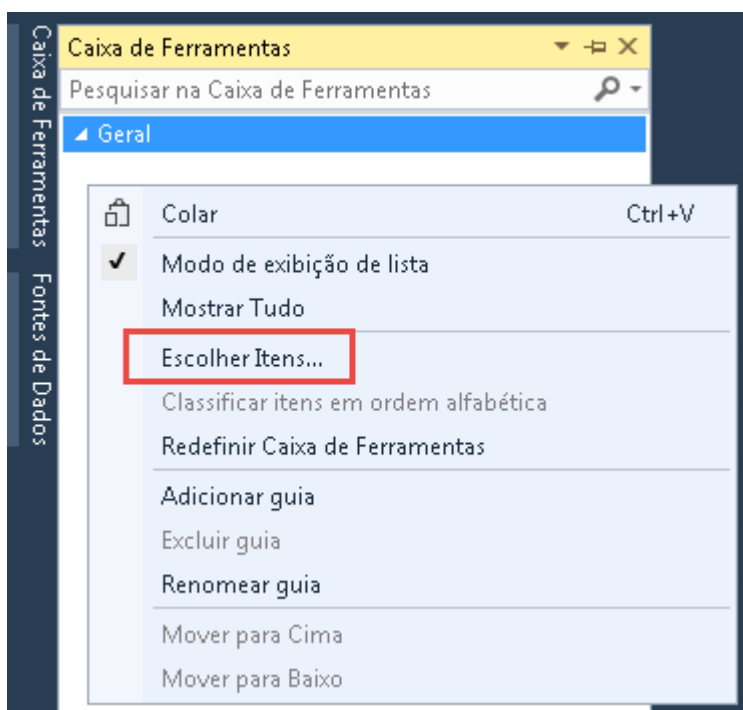
# Instalando o Report Viewer

O Report Viewer é instalado automaticamente com a edição Professional ou superior do Visual Studio. Além disso, a edição “Community” do Visual Studio também traz consigo as ferramentas de relatório do Report Viewer. Se você estiver utilizando uma dessas edições do Visual Studio, avance para o próximo capítulo.

Apesar de conseguirmos utilizar o Report Viewer com a edição “Express” do Visual Studio, ela não é completamente compatível com essa poderosa ferramenta de geração de relatórios. Nessa edição, é possível instalar o controle para exibição de relatórios, porém, não é possível desenhar os relatórios dentro do Visual Studio.

Para adicionar o controle de exibição de relatórios do Report Viewer no Visual Studio Express, baixe e instale a “Microsoft Report Viewer 2012 Runtime”, disponível no seguinte endereço: <http://www.andrealveslima.com.br/links/RVRuntime>.

Uma vez instalada a runtime, crie um novo projeto do tipo “Aplicativo Windows Forms” no Visual Studio Express. Feito isso, vá até a caixa de ferramentas, clique com o botão direito do mouse em uma área livre e clique na opção “Escolher Itens”:



**Figura 1.1. Escolhendo novos itens para a caixa de ferramentas**

Na aba “Componentes do .NET Framework”, clique em “Procurar”, vá até a pasta de instalação do Report Viewer e selecione o arquivo “Microsoft.ReportViewer.WinForms.dll” (uma vez que trabalharemos nesse livro somente com o Windows Forms – caso você deseje utilizar o Report Viewer no ASP.NET, basta selecionar a dll com final “WebForms”) ao invés da dll com final “WinForms”).

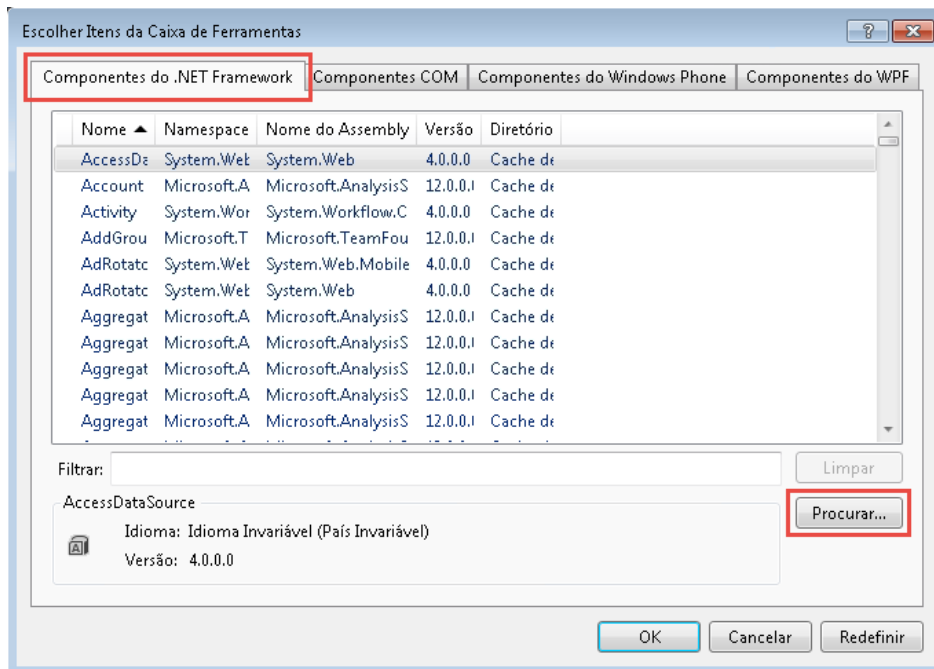


Figura 1.2. Procurando novos itens para adicionar à barra de ferramentas

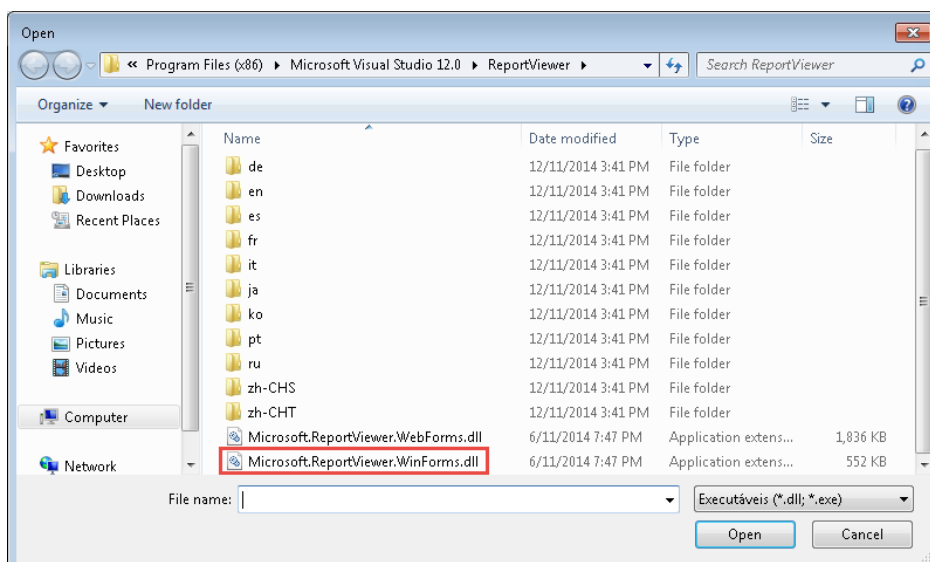
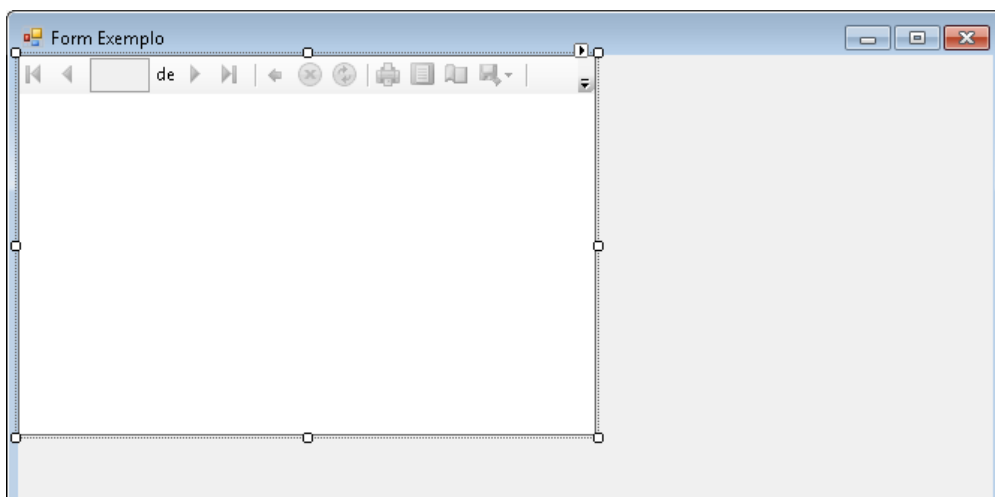


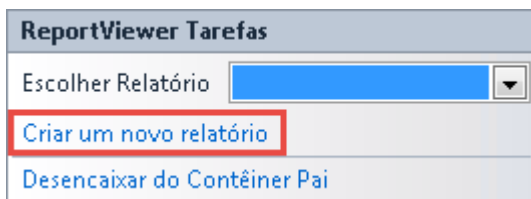
Figura 1.3. Escolhendo a dll do Report Viewer para ser adicionada à caixa de ferramentas

Após termos adicionado o controle, ele aparecerá na barra de ferramentas do Visual Studio dentro da categoria "Geral" com o nome "Report Viewer". Arraste esse controle para dentro do Form e você terá um resultado parecido com o da imagem abaixo:



**Figura 1.4. Controle do Report Viewer adicionado ao Form**

Como mencionado anteriormente, apesar de conseguirmos adicionar o controle visualizador do Report Viewer nas edições Express do Visual Studio, não é possível desenhar novos relatórios nem editar relatórios existentes diretamente dentro do Visual Studio Express. Caso você tente utilizar a smart tag do controle e clicar na opção "Criar um novo relatório", nada acontecerá.

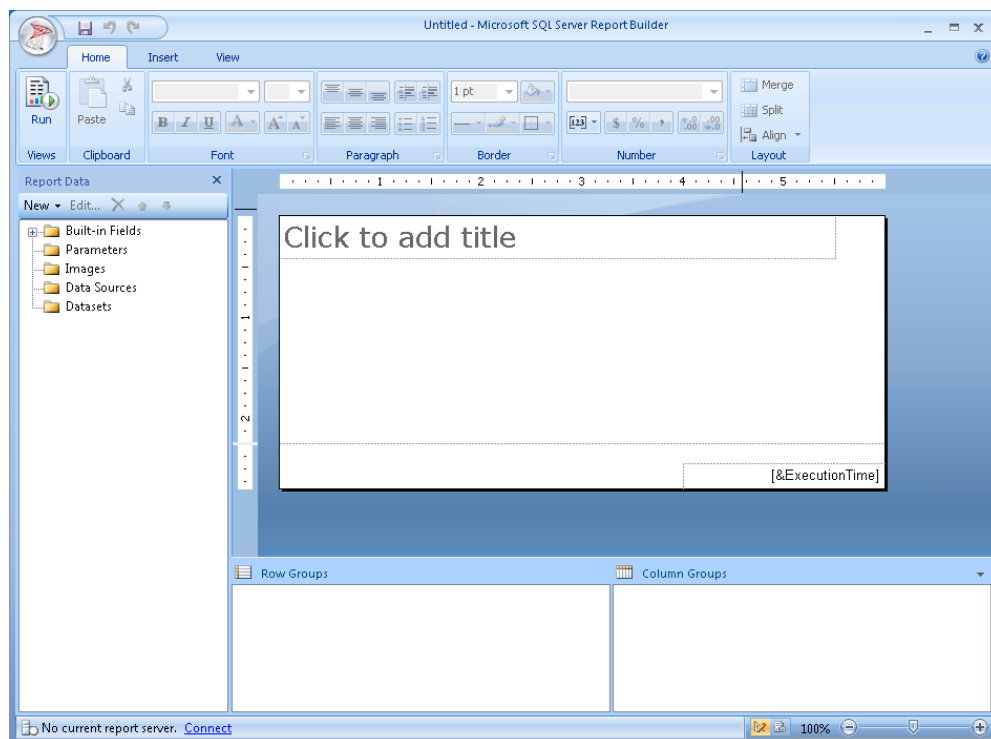


**Figura 1.5. Tentando criar um novo relatório através da smart tag do controle do Report Viewer**

Ao adotar o Visual Studio Express, será necessário utilizar uma ferramenta externa para desenhar novos relatórios ou editar relatórios existentes. A Microsoft oferece uma ferramenta gratuita para esse propósito, chamada "SQL Server 2012 Report Builder". Você pode fazer o download dessa ferramenta no seguinte endereço: <http://www.andrealveslima.com.br/links/ReportBuilder>.

Uma vez instalada, você conseguirá abrir, editar ou criar relatórios com as extensões rdl e rdcl, que são justamente as extensões utilizadas pelo visualizador do Report Viewer.





**Figura 1.6. Interface do SQL Server Report Builder**

Agora que já temos o Report Viewer instalado, vamos conferir como podemos criar o nosso primeiro relatório no próximo capítulo.

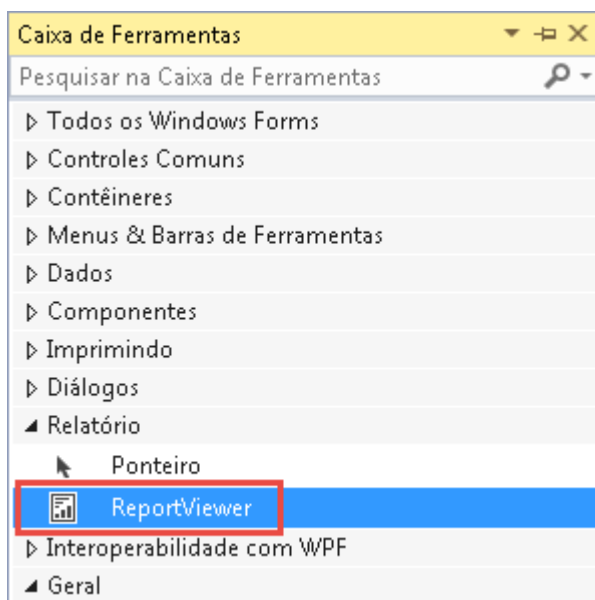
NOTA: COMO MENCIONADO NESSE CAPÍTULO, SÓ É POSSÍVEL CRIARMOS NOVOS RELATÓRIOS OU EDITARMOS RELATÓRIOS EXISTENTES COM O VISUAL STUDIO COMMUNITY OU SUPERIOR, JÁ QUE ESSE SUPORTE NÃO FOI ADICIONADO ÀS EDIÇÕES EXPRESS DO VISUAL STUDIO. PORTANTO, A PARTIR DESSE PONTO, UTILIZAREMOS A EDIÇÃO COMMUNITY DO VISUAL STUDIO PARA DEMONSTRAR A CRIAÇÃO E EDIÇÃO DOS RELATÓRIOS. DE QUALQUER FORMA, CASO VOCÊ TENHA QUE UTILIZAR O SQL SERVER REPORT BUILDER (ALTERNATIVA PARA EDITAR OS RELATÓRIOS PARA QUEM SÓ PODE UTILIZAR AS EDIÇÕES EXPRESS DO VISUAL STUDIO), A INTERFACE DESSA FERRAMENTA É BEM PARECIDA COM O EDITOR DE RELATÓRIOS DO VISUAL STUDIO.

## Capítulo 2

# Criando seu primeiro relatório

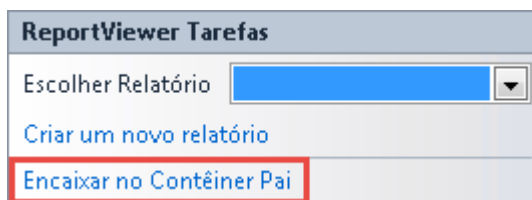
Para criar o seu primeiro relatório você primeiramente deverá criar um projeto do tipo “Aplicativo Windows Forms”. O Report Viewer também suporta projetos WPF, porém não de forma nativa. Dessa forma, todos os exemplos desse livro utilizarão um projeto do tipo “Aplicativo Windows Forms” como base.

Uma vez criado o projeto do tipo “Aplicativo Windows Forms”, vá até a seção “Relatório” da caixa de ferramentas e dê um duplo clique no controle “ReportViewer”:



**Figura 2.1. Controle do Report Viewer na categoria Relatório da Caixa de Ferramentas**

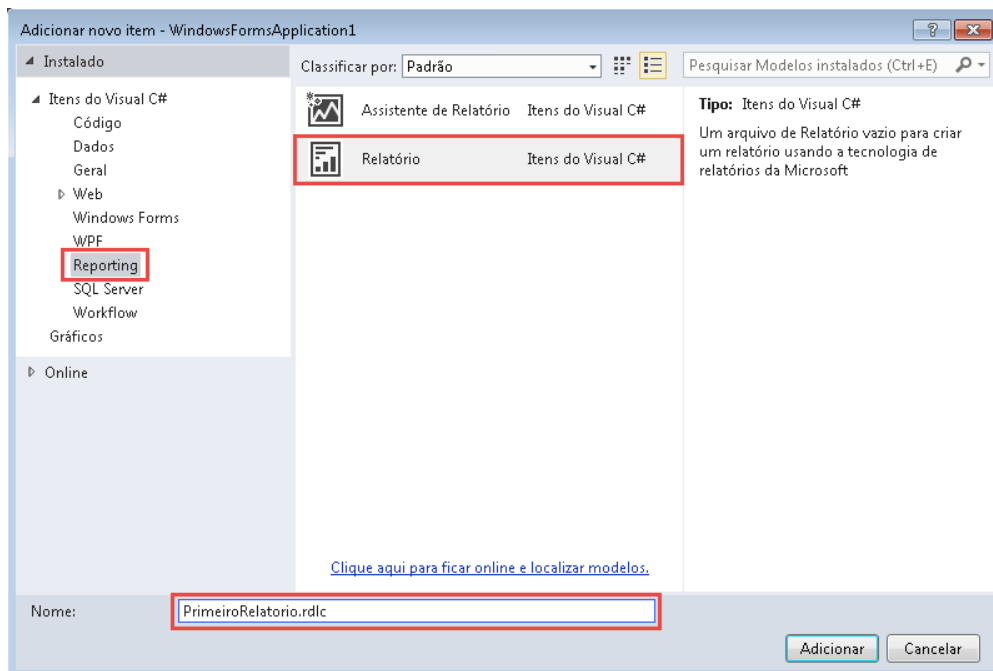
Feito isso, utilize a smart tag do controle de relatório para fazer o docking do controle, de forma que ele ocupe o formulário inteiro:



**Figura 2.2. Dockando o controle do Report Viewer no formulário**

Se desejar, você pode também aumentar o tamanho do formulário para que o relatório utilize um espaço maior da tela.

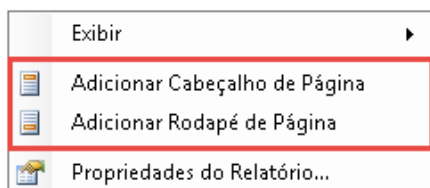
Agora que você tem um controle do Report Viewer no seu formulário, o próximo passo é criar o relatório. Para isso, utilize a opção “Adicionar Novo Item” no projeto e escolha o item do tipo “Relatório”, que se encontra dentro da categoria “Reporting”. Dê o nome “PrimeiroRelatorio” para o relatório a ser criado:



**Figura 2.3. Adicionando um novo relatório ao projeto**

NOTA: OUTRA OPÇÃO QUE VOCÊ PODERIA UTILIZAR PARA ADICIONAR UM NOVO RELATÓRIO AO PROJETO SERIA O ITEM “CRIAR UM NOVO RELATÓRIO” PRESENTE NA SMART TAG DO CONTROLE DO REPORT VIEWER. PORÉM, ESSA OPÇÃO NÃO ADICIONA UM RELATÓRIO EM BRANCO, MAS SIM, ACIONA O REPORT WIZARD, QUE SERÁ ABORDADO SOMENTE NO PRÓXIMO CAPÍTULO.

Ao confirmar o diálogo, o relatório será adicionado ao projeto. Note que a aparência inicial do relatório é mais simples do que outras ferramentas de geração de relatórios, como o Crystal Reports. Isso se deve ao fato de que, por padrão, os relatórios do Report Viewer aparecem somente com a área de detalhe. Para adicionar cabeçalho e rodapé, basta clicarmos com o botão direito do mouse em uma área externa ao relatório e escolhermos a opção “Adicionar Cabeçalho de Página” ou “Adicionar Rodapé de Página”.

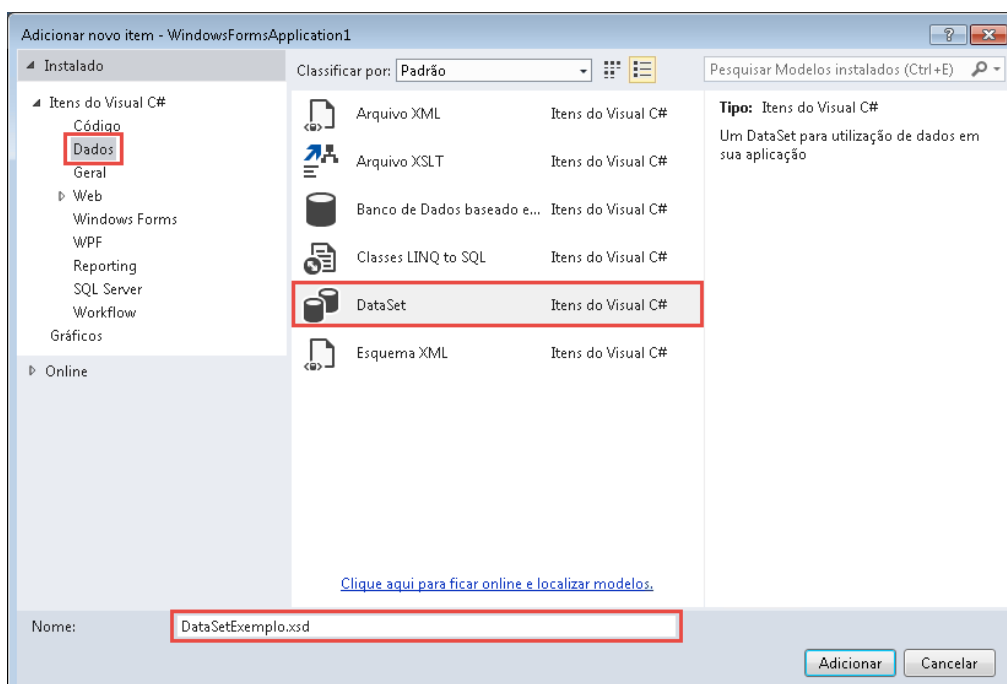


**Figura 2.4. Adicionando cabeçalho e rodapé**

Veremos uma explicação detalhada das diferentes áreas de design do relatório no **Capítulo 5 (Entendendo o layout do relatório)**.

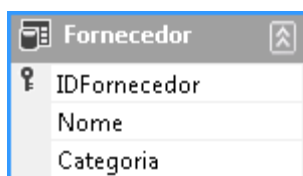
Todo relatório precisa de uma fonte de dados. A fim de simplificarmos o código e focarmos puramente no desenvolvimento do relatório, iremos utilizar DataSets tipados em todos os exemplos desse livro.

Para adicionar um novo DataSet ao projeto, utilize a opção “Adicionar Novo Item” e escolha o item do tipo “DataSet”, que se encontra dentro da categoria “Dados”. Dê o nome de “DataSetExemplo” para esse novo DataSet:



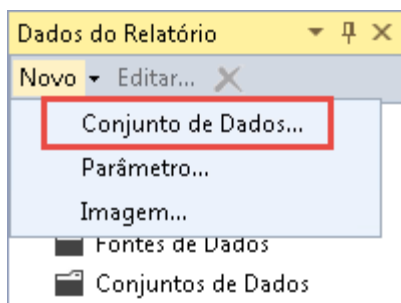
**Figura 2.5. Adicionando um novo DataSet ao projeto**

Dentro desse DataSet, crie uma DataTable chamada “Fornecedor”, com os campos “IDFornecedor”, “Nome” e “Categoria”. Configure a coluna “IDFornecedor” como chave primária auto incremento.



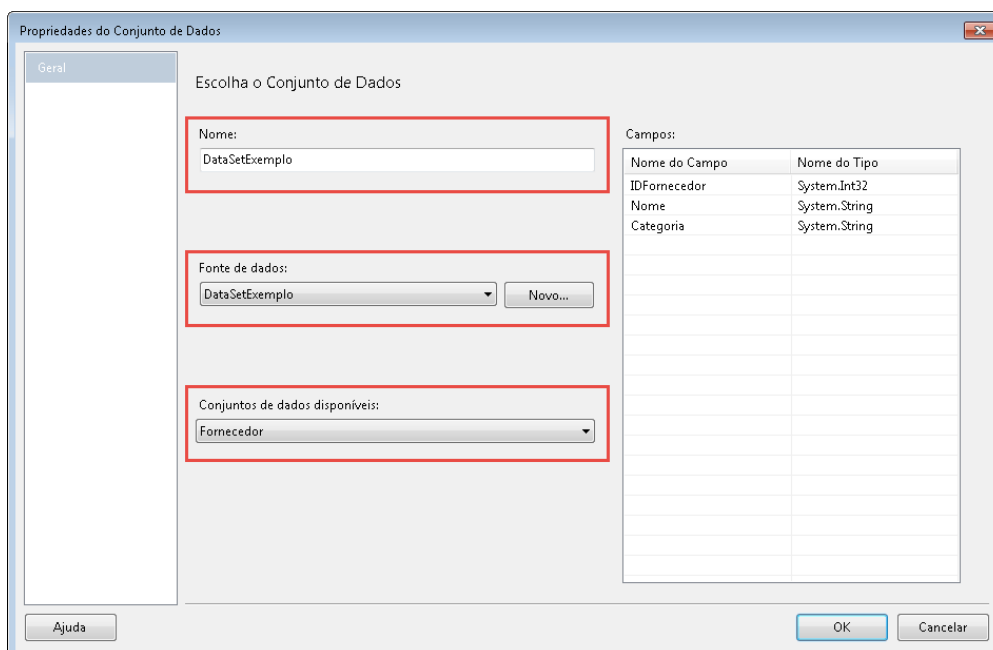
**Figura 2.6. Tabela Fornecedor adicionada ao DataSet**

Agora que já temos um DataSet no projeto, podemos criar uma nova fonte de dados para o relatório, utilizando esse DataSet como base. Para isso, vá até o relatório e escolha a opção *"Novo Conjunto de Dados"* na janela *"Dados do Relatório"*.



**Figura 2.7. Adicionando um novo conjunto de dados ao relatório**

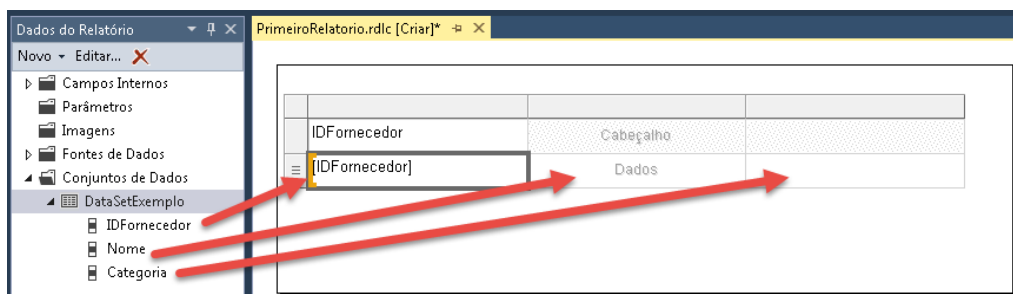
Na janela *"Propriedades do Conjunto de Dados"*, escolha o nome *"DataSetExemplo"* para o novo DataSet. Na caixa de opções *"Fonte de dados"*, escolha o DataSet criado anteriormente (DataSetExemplo); e na caixa de opções *"Conjuntos de dados disponíveis"*, escolha a tabela *"Fornecedor"*:



**Figura 2.8. Propriedades do novo DataSet que será adicionado ao relatório**

Com o Dataset adicionado no relatório, podemos agora utilizar os campos da tabela *"Fornecedor"* para desenharmos o nosso primeiro relatório. Esse primeiro relatório será uma simples listagem dos fornecedores.

Para alcançarmos esse objetivo, adicione um controle *"Table"* na área de detalhes do relatório e arraste os campos da área *"Dados do Relatório"* para as células dessa Table.



**Figura 2.9. Arrastando os campos do DataSet para dentro do controle Table**

NOTA: MUITAS VEZES A ÁREA “DADOS DO RELATÓRIO” NÃO ESTÁ VISÍVEL POR PADRÃO. CASO ELA NÃO ESTEJA ATIVA NO SEU VISUAL STUDIO, VOCÊ PODE ATIVÁ-LA CLICANDO NO MENU “EXIBIR” E DEPOIS ESCOLHENDO A OPÇÃO “DADOS DO RELATÓRIO” (NORMALMENTE A ÚLTIMA OPÇÃO DO MENU).

Feito isso, para que o relatório fique com uma melhor aparência, podemos customizar os títulos das colunas, fontes e cores de fundo das células da tabela, conforme a figura abaixo:

ID	Nome	Categoria
[IDFornecedor]	[Nome]	[Categoria]

**Figura 2.10. Melhorando a aparência da Table**

Uma vez criado o nosso primeiro relatório, precisamos testá-lo. No próximo capítulo iremos conferir como exibir o preview de relatórios locais com o Report Viewer.

NOTA: TODOS OS EXEMPLOS APRESENTADOS NESTE LIVRO ESTÃO DISPONÍVEIS NO PROJETO DE EXEMPLO, QUE É PARTE INTEGRANTE DESTA OBRA. VOCÊ PODE BAIXÁ-LO ATRAVÉS DESTA LINK: <http://www.andrealveslima.com.br/files/ebookreportviewer/projetoexemplo.zip>

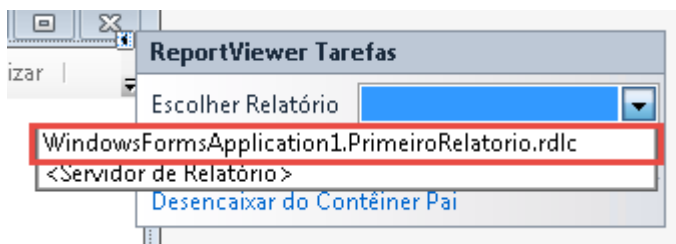
A COLEÇÃO DE EXEMPLOS CONTA COM 15 PROJETOS MOSTRANDO A EVOLUÇÃO DE TODO O CÓDIGO APRESENTADO. CADA PROJETO REPRESENTA UM CAPÍTULO DIFERENTE. ISSO QUER DIZER QUE VOCÊ PODE ACOMPANHAR TOTALMENTE O PROGRESSO QUE OS RELATÓRIOS FORAM SOFRENDO DURANTE O DESENVOLVIMENTO DO LIVRO.

## Capítulo 3

# Acessando o preview de relatórios locais

Como podemos observar, ao desenvolvermos relatórios locais, não é possível acessarmos o preview do relatório através do designer do Report Viewer. A única maneira de visualizarmos o preview de um relatório local é exibi-lo com a aplicação em execução.

Para exibir o relatório que criamos no **Capítulo 2 (Criando o seu primeiro relatório)**, vá até o formulário, clique na smart tag do controle do Report Viewer e escolha o relatório (PrimeiroRelatorio.rdlc) na caixa de opções:



**Figura 3.1. Escolhendo o relatório a ser exibido**

Note que, ao escolhermos o relatório na caixa de opções, o Visual Studio cria automaticamente uma instância do DataSet (DataSetExemplo) e um Binding Source no Form (FornecedorBindingSource). O Report Viewer utilizará esse DataSet para alimentar os dados do relatório:



**Figura 3.2. DataSet e Binding Source criados automaticamente no Form**

Em um cenário de aplicação real, os dados desse DataSet deveriam ser carregados de um banco de dados. Porém, a fim de simplificarmos o nosso exemplo e focarmos no que realmente importa (o relatório), iremos incluir manualmente novas linhas na tabela Fornecedor.

Vá até o code-behind do formulário e observe que o Visual Studio já criou um handler para o evento Load do Form. Nesse handler, o único código que temos até o momento é uma chamada do método *"RefreshReport"* do controle do Report Viewer. Antes dessa chamada, vamos incluir algumas novas linhas na tabela Fornecedor, conforme podemos observar na **Listagem 3.1** a seguir.

**Listagem 3.1. Adicionando novas entradas na tabela Fornecedor**

```

1  private void FormExemplo_Load(object sender, EventArgs e)
2  {
3      DataSetExemplo.Fornecedor.AddFornecedorRow(
4          "Fornecedor 1", "Materiais de Escritório");
5      DataSetExemplo.Fornecedor.AddFornecedorRow(
6          "Fornecedor 2", "Materiais de Escritório");
7      DataSetExemplo.Fornecedor.AddFornecedorRow(
8          "Fornecedor 3", "Mão de Obra");
9      // Continua. Confira no projeto de exemplo
10     // a versão completa deste método.
11
12     this.reportViewer.RefreshReport();
13 }

```

Execute o projeto e observe o resultado:

ID	Nome	Categoria
-1	Fornecedor 1	Materiais de Escritório
-2	Fornecedor 2	Materiais de Escritório
-3	Fornecedor 3	Mão de Obra
-4	Fornecedor 4	Mão de Obra
-5	Fornecedor 5	Obra-Prima
-6	Fornecedor 6	Obra-Prima
-7	Fornecedor 7	Materiais de Escritório
-8	Fornecedor 8	Materiais de Escritório
-9	Fornecedor 9	Mão de Obra
-10	Fornecedor 10	Mão de Obra
-11	Fornecedor 11	Obra-Prima
-12	Fornecedor 12	Obra-Prima
-13	Fornecedor 13	Materiais de Escritório
-14	Fornecedor 14	Materiais de Escritório
-15	Fornecedor 15	Mão de Obra

**Figura 3.3. Preview de um relatório local com o aplicativo em execução**



E dessa forma conseguimos testar qualquer relatório local no Report Viewer. É um tanto quanto inconveniente (porque temos que criar um formulário para testar cada relatório), mas, infelizmente, é a única maneira de visualizarmos o resultado dos nossos relatórios locais.

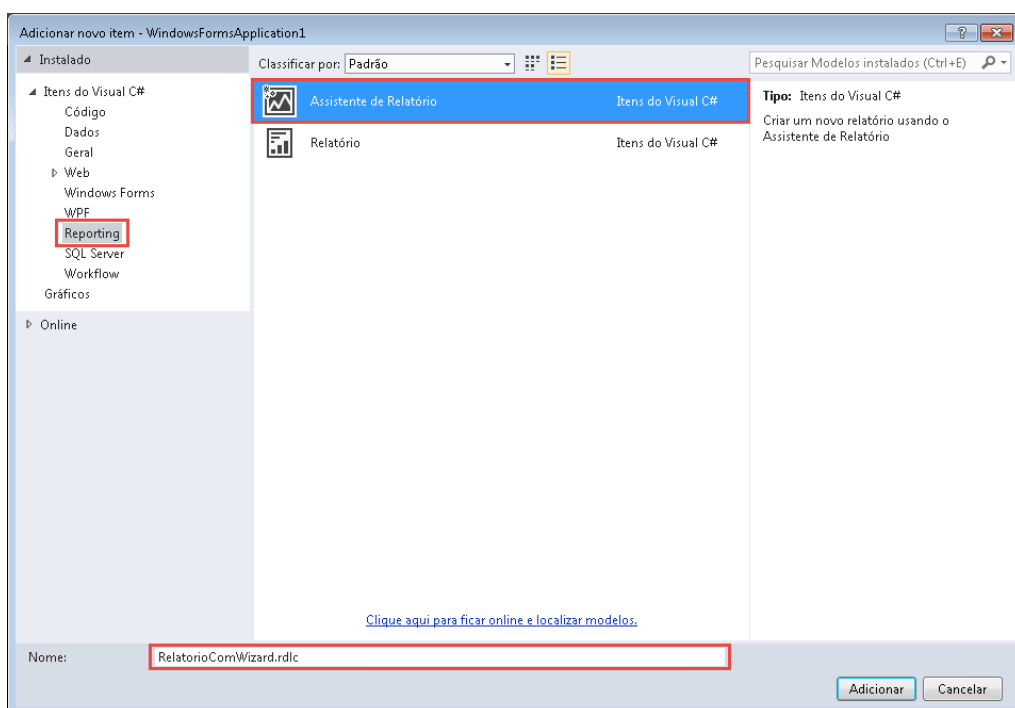
No próximo capítulo iremos explorar a criação de relatórios utilizando o Assistente de Relatório (ou Report Wizard).

## Capítulo 4

# Utilizando o Assistente de Relatório (Report Wizard)

Agora que você já viu como criar manualmente o seu primeiro relatório, vale a pena explorarmos o Assistente de Relatório. Nas situações em que você precisar gerar um relatório rapidamente, pode ser que o assistente de relatório te atenda.

Para acionarmos o Assistente de Relatório, vamos adicionar um novo item ao projeto, escolhendo o tipo “Assistente de Relatório”, que se encontra dentro da categoria “Reporting”. Dê o nome de “RelatorioComWizard.rdlc” ao relatório a ser criado:



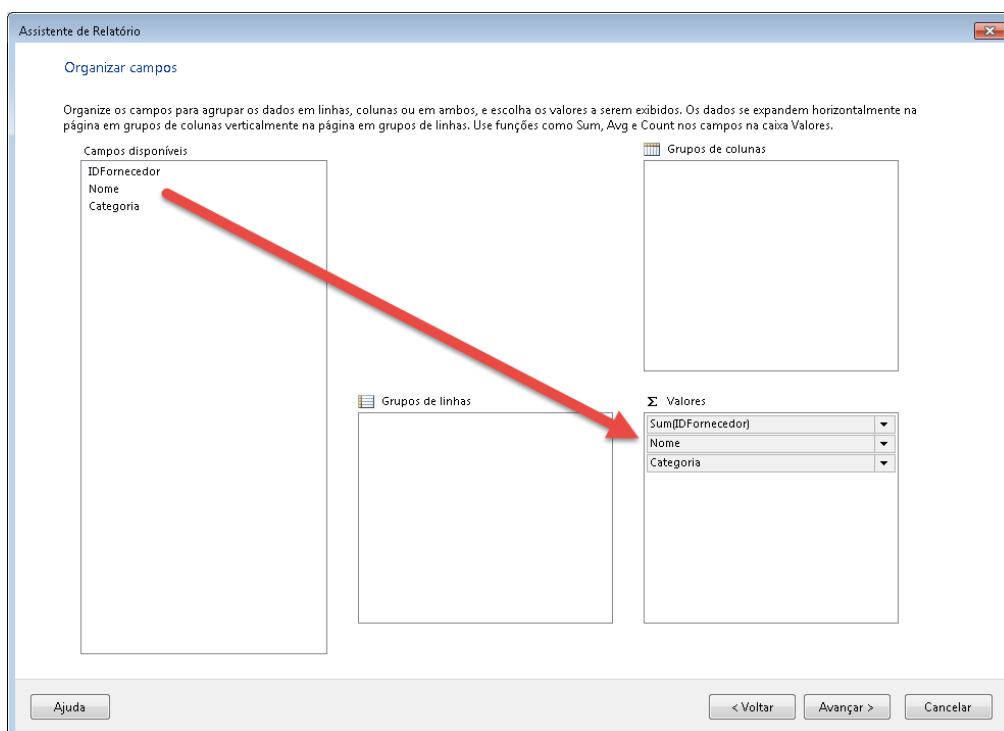
**Figura 4.1. Adicionando um novo relatório através do Assistente de Relatório**

Ao clicar no botão “Adicionar”, o Visual Studio exibirá a tela para selecionarmos a fonte de dados do relatório. Vamos utilizar o mesmo DataSet criado anteriormente, portanto, preencha as informações nessa tela da mesma forma que preenchamos anteriormente no **Capítulo 2 (Criando seu primeiro relatório)**, **Figura 2.8**.

A próxima tela do assistente (“Organizar campos”) serve para escolhermos os campos que queremos exibir no relatório. Fazemos isso arrastando os campos da lista “Cam-

pos disponíveis” para as outras listas. Como o próprio nome diz, a lista “Grupo de colunas” serve para criarmos grupos como colunas; e a lista “Grupo de linhas” serve para criarmos grupos como linhas (veremos logo a seguir o que significa cada um desses termos). Por fim, a lista “Valores” deverá conter todos os itens que queremos exibir na listagem do relatório.

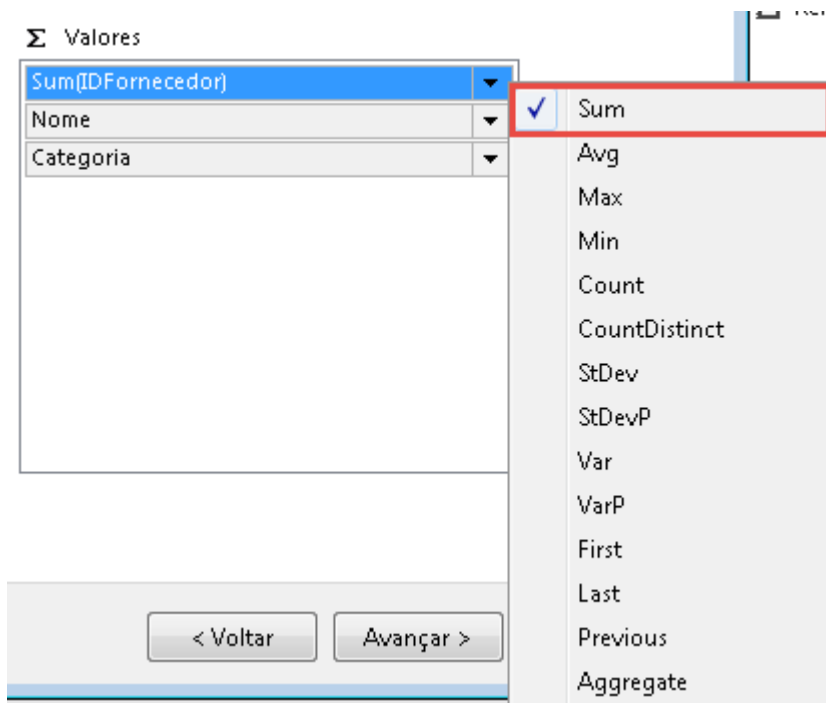
Para construirmos um relatório semelhante ao que desenhamos no **Capítulo 2 (Criando seu primeiro relatório)**, vamos adicionar todos os três campos da tabela Fornecedor na lista de “Valores”, conforme podemos conferir na imagem abaixo:



**Figura 4.2. Adicionando todos os campos da tabela na lista de “Valores”**

Um ponto importante a ser observado na **Figura 4.2** é que quando arrastamos um campo numérico para a lista de “Valores”, o assistente automaticamente configura o resultado desse campo como um somatório (“Sum(Campo)”). No nosso exemplo, como o campo “IDFornecedor” é numérico, note que o assistente configurou-o como “Sum(IDFornecedor)”.

É difícil entender a razão pela qual o assistente utiliza essa lógica para campos numéricos, pois esse provavelmente não é o comportamento que desejamos em um relatório de listagem. Para configurar corretamente esse tipo de campo, abra a listagem e desmarque a opção “Sum”:



**Figura 4.3. Desmarcando o totalizador do campo “IDFornecedor”**

Feito isso, podemos clicar no botão “Avançar” para seguirmos com a próxima etapa do assistente (“Escolha o layout”).

Na tela de escolha do layout, como o próprio nome diz, podemos escolher o layout do relatório. Como o nosso relatório não tem nenhum grupo de linhas, não é possível customizarmos nada nessa tela (todas as opções aparecerão desabilitadas). Logo mais iremos conferir como gerar relatórios com grupos utilizando o Assistente de Relatório. Por enquanto, escolha a opção “Avançar”.

A próxima (e última) tela do assistente é onde conseguimos selecionar o estilo do relatório. Basicamente, a única diferença entre os estilos são as cores e tipos de bordas da tabela. Escolha o que te agradar mais e clique em “Concluir”.

Ao finalizar o assistente, um relatório bem simples será gerado:

IDForneced	Nome	Categoria
[IDFornecedor]	[Nome]	[Categoria]

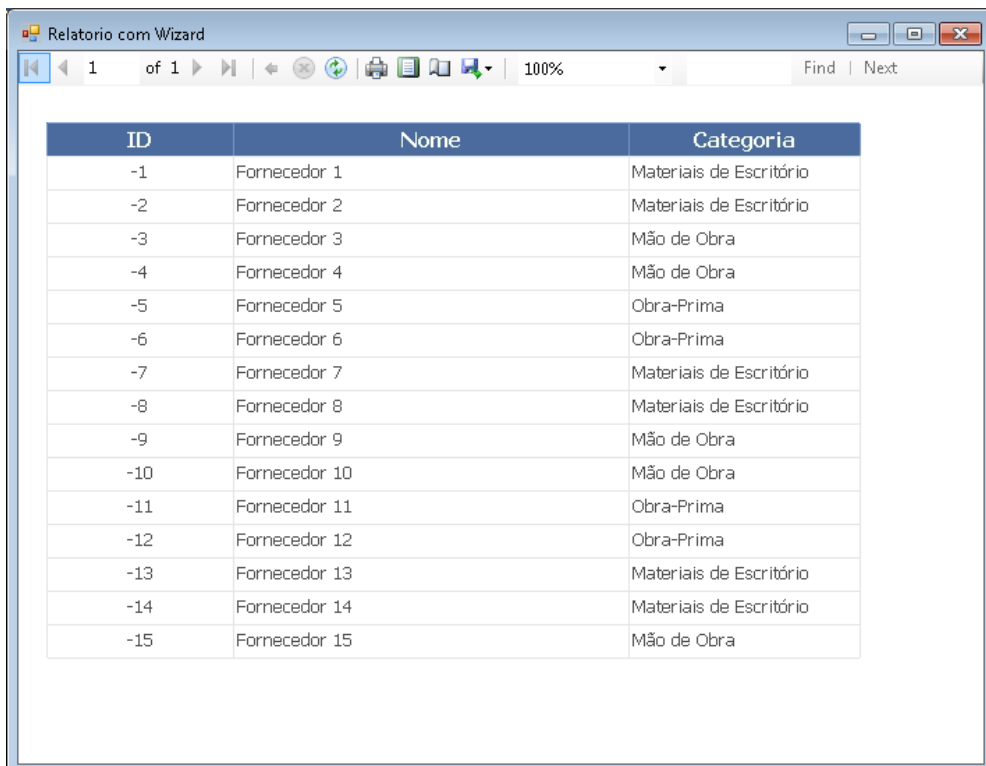
**Figura 4.4. Resultado do Assistente de Relatório**

Uma vez criado o relatório, podemos formatá-lo da maneira que quisermos. Por exemplo, podemos aumentar o tamanho das colunas, editar o texto do título das colunas e centralizar algumas células, como fizemos anteriormente no **Capítulo 2 (Criando seu primeiro relatório)**. Veja o resultado:

ID	Nome	Categoria
[IDFornecedor]	[Nome]	[Categoria]

**Figura 4.5. Resultado do Assistente de Relatório após formatação adicional**

Para testarmos o relatório, precisamos criar um novo formulário e seguir os passos apresentados no **Capítulo 3 (Acessando o preview de relatórios locais)**. O resultado será parecido como o apresentado na **Figura 4.6** a seguir.



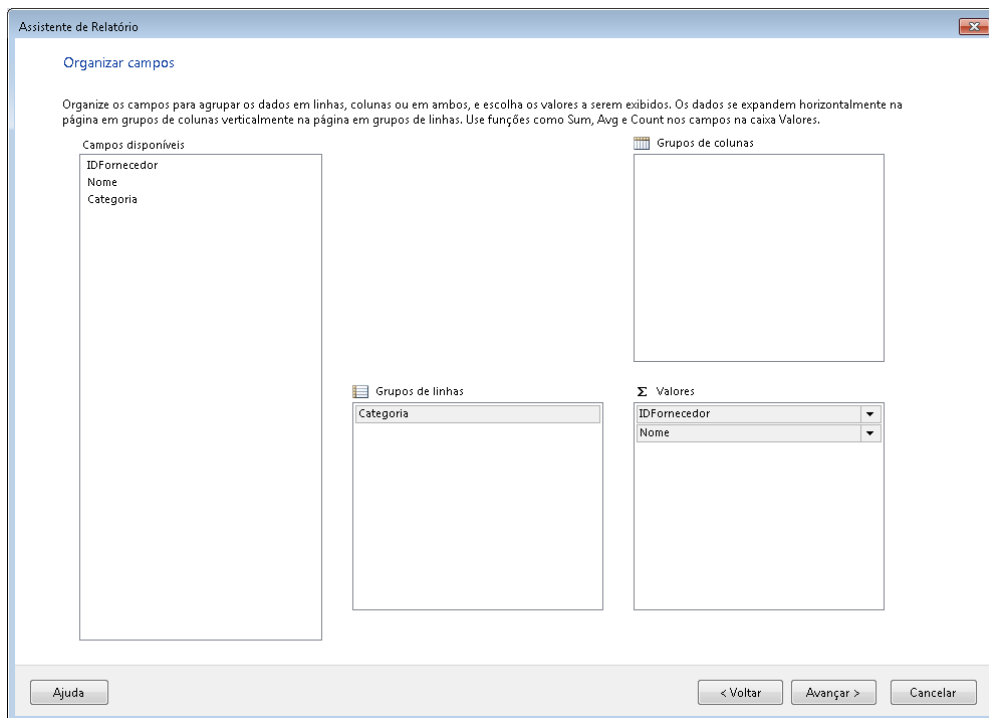
ID	Nome	Categoria
-1	Fornecedor 1	Materiais de Escritório
-2	Fornecedor 2	Materiais de Escritório
-3	Fornecedor 3	Mão de Obra
-4	Fornecedor 4	Mão de Obra
-5	Fornecedor 5	Obra-Prima
-6	Fornecedor 6	Obra-Prima
-7	Fornecedor 7	Materiais de Escritório
-8	Fornecedor 8	Materiais de Escritório
-9	Fornecedor 9	Mão de Obra
-10	Fornecedor 10	Mão de Obra
-11	Fornecedor 11	Obra-Prima
-12	Fornecedor 12	Obra-Prima
-13	Fornecedor 13	Materiais de Escritório
-14	Fornecedor 14	Materiais de Escritório
-15	Fornecedor 15	Mão de Obra

**Figura 4.6. Preview do relatório gerado anteriormente através do Assistente de Relatório**

O relatório gerado anteriormente não conta nenhum agrupamento. Porém, uma das funcionalidades mais úteis do Assistente de Relatório é a opção de gerarmos grupos de linhas ou grupos de colunas. Confira nas próximas seções como trabalhar com grupos de linhas e grupos de colunas com o Assistente de Relatório.

## Trabalhando com grupos de linhas

Infelizmente não é possível acionarmos o assistente novamente uma vez que ele tenha sido concluído. Portanto, para criar um relatório com grupos utilizando o assistente, siga os passos apresentados anteriormente e selecione o mesmo DataSet como fonte de dados. Porém, na etapa *“Organizar campos”*, ao invés de adicionar a coluna *“Categoria”* na lista *“Valores”*, adicione-a na lista *“Grupos de linhas”*, conforme a **Figura 4.7** a seguir.



**Figura 4.7. Criando um grupo de linhas com o Assistente de Relatório**

Uma vez criado um grupo de linhas, podemos utilizar as opções disponíveis na etapa "Escolha o layout". O primeiro bloco de opções diz respeito aos subtotais e totais gerais:

- **Bloqueado, subtotal abaixo:** a coluna com o grupo de linhas fica "bloqueada" (ou seja, ocupa o bloco inteiro da coluna) e o subtotal é apresentado na última linha do grupo

Categoria	IDForneced	Nome
[Categoria]	[IDForneced]	[Nome]
	[Sum(IDForneced)]	
Total	[Sum(IDForneced)]	

- **Bloqueado, subtotal acima:** a coluna com o grupo de linhas fica "bloqueada" e o subtotal é apresentado na primeira linha do grupo

Categoria	IDForneced	Nome
[Categoria]	[Sum(IDForneced)]	
	[IDForneced]	[Nome]
Total	[Sum(IDForneced)]	

- **Nível, subtotal acima:** a coluna com o grupo de linhas não fica “bloqueada” (ou seja, só aparece na primeira linha) e o subtotal é apresentado na primeira linha do grupo

Categoria	IDForneced	Nome
[Categoria]	[Sum(IDFornecedor)]	
	[IDFornecedor]	[Nome]
Total	[Sum(IDFornecedor)]	

A segunda opção dessa tela é relacionada à expansão e recolhimento dos grupos. Caso você deixe essa opção checada, os grupos aparecerão fechados por padrão (o usuário pode expandi-los clicando no botão “+” que será automaticamente adicionado na célula correspondente ao agrupamento).

Um inconveniente dos totalizadores gerados pelos agrupamentos do Assistente de Relatório é que, por padrão, eles realizam a soma dos valores de campos numéricos. Apesar de fazer sentido na maioria dos casos, algumas vezes a opção que faz mais sentido seria a contagem de valores. Por exemplo, no nosso caso, ao invés de computarmos a somatória de “IDFornecedor”, faz mais sentido exibirmos a contagem de Fornecedores por Categoria. Infelizmente não existe uma opção para alterarmos a função totalizadora diretamente no assistente, portanto, temos que alterá-la somente após o relatório ter sido gerado.

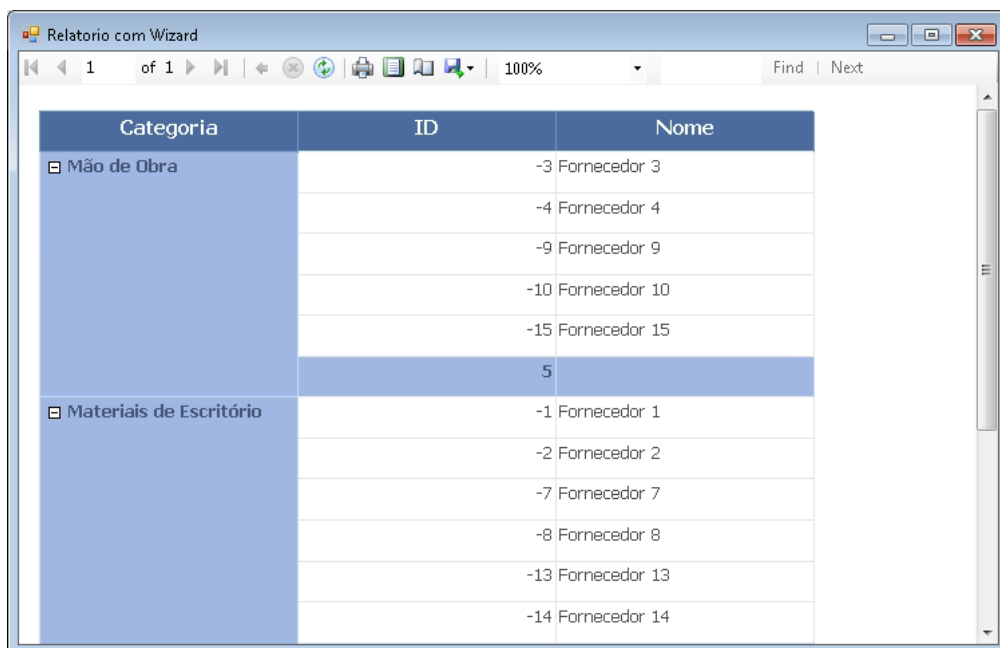
Na figura abaixo, note que, apesar do assistente ter gerado o totalizador como a somatória de “IDFornecedor”, podemos alterar para que uma contagem seja realizada ao invés da soma.

Categoria	ID	Nome
[Categoria]	[IDFornecedor]	[Nome]
	[Count(IDFornecedor)]	
Total	[Count(IDFornecedor)]	

**Figura 4.8. Totalizador do agrupamento por linha alterado de somatória para contagem**

Ao acessarmos o preview desse relatório, o resultado será parecido com a **Figura 4.9** apresentada a seguir.





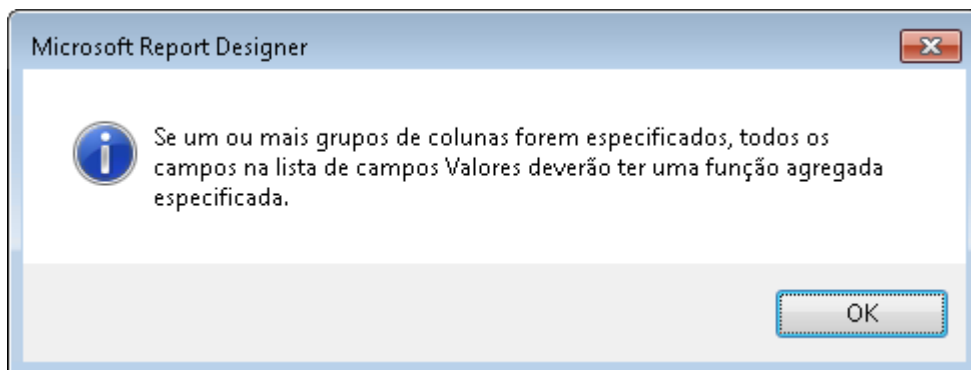
Categoria	ID	Nome
Mão de Obra	-3	Fornecedor 3
	-4	Fornecedor 4
	-9	Fornecedor 9
	-10	Fornecedor 10
	-15	Fornecedor 15
Materiais de Escritório	5	
	-1	Fornecedor 1
	-2	Fornecedor 2
	-7	Fornecedor 7
	-8	Fornecedor 8
	-13	Fornecedor 13
	-14	Fornecedor 14

**Figura 4.9. Resultado de um relatório gerado pelo assistente com grupo de linhas**

## Trabalhando com grupos de colunas

Outra funcionalidade extremamente interessante que está disponível no Report Viewer são os grupos de colunas. Com eles nós basicamente conseguimos transformar informações disponíveis nas linhas da tabela em colunas. Esse conceito é chamado de "traverse" e é muito difícil de ser implementado através de uma consulta SQL ou em outros geradores de relatórios.

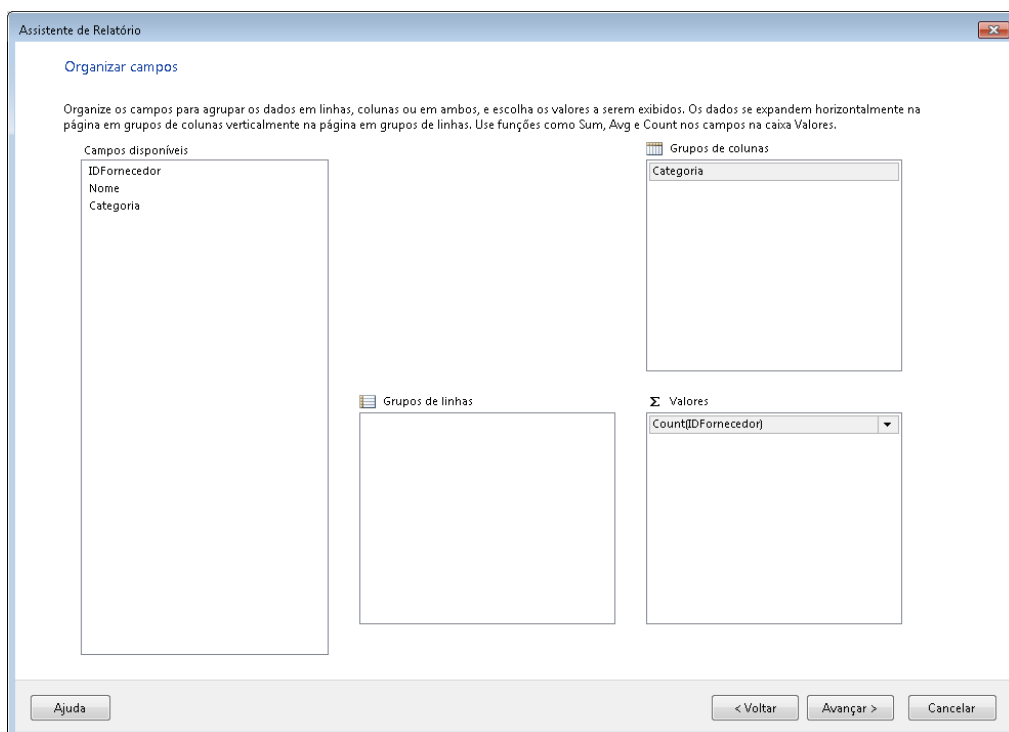
O Assistente de Relatório do Report Viewer tem a opção de criarmos grupos de colunas na etapa "Organizar campos". Porém, ao tentarmos adicionar a coluna "Categoria" na lista "Grupos de colunas", se mantivermos as outras colunas na lista "Valores", receberemos esta mensagem de erro:



**Figura 4.10. Erro ao tentarmos adicionar um grupo de colunas utilizando o assistente**

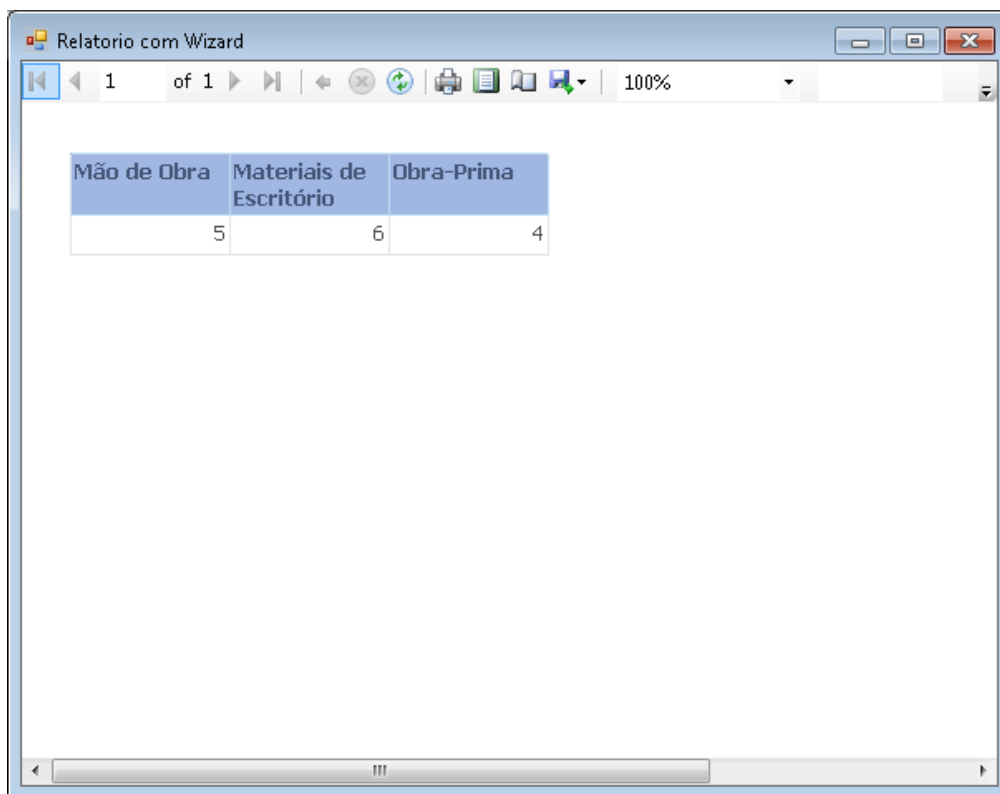
Como a própria mensagem diz, ao utilizarmos grupos de colunas, todas as colunas especificadas na lista “Valores” devem conter uma função totalizadora, já que não faz sentido mostrar uma lista de valores quando temos um grupo de colunas (você entenderá melhor ao conferir o resultado do relatório na **Figura 4.12**).

Portanto, para corrigirmos esse erro e conseguirmos gerar o relatório com êxito, remova a coluna “Nome” para fora da lista “Valores” e altere a função totalizadora do campo “IDFornecedor” para “Count”.



**Figura 4.11. Configurando um grupo de colunas com o Assistente de Relatório**

Feito isso, siga com as próximas etapas do assistente até que ele seja concluído. Confira o resultado do preview do relatório com grupos de colunas na **Figura 4.12**.



Mão de Obra	Materiais de Escritório	Obra-Prima
5	6	4

**Figura 4.12. Resultado do relatório com grupo de colunas gerado pelo assistente**

Note que cada uma das categorias é exibida como uma coluna da tabela e que a contagem de itens de cada categoria é exibida logo abaixo dela. Se tivéssemos mais ou menos categorias, o relatório se ajustaria automaticamente. Essa é uma funcionalidade muito poderosa e, como dito anteriormente, muito difícil de ser reproduzida em outros geradores de relatórios.

## Capítulo 5

# Entendendo o layout do relatório

Os relatórios do Report Viewer possuem três áreas específicas: a área de dados, o cabeçalho e o rodapé. Por padrão, ao criarmos um novo relatório, somente a área de dados é exibida. Nesse capítulo aprenderemos a ativar as outras regiões do relatório e veremos uma lista com os controles que podemos utilizar, além das propriedades que conseguimos alterar em cada relatório.

## Data Region

A área de dados (ou “*Data Region*”) dos relatórios no Report Viewer é o local onde exibiremos a lista de dados nos nossos relatórios. Essa área é ativada por padrão e é nela que colocaremos a maioria dos controles do relatório.

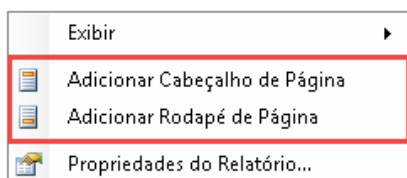
Dentro da área de dados, podemos incluir diversos componentes para realizar a listagens de dados, como, por exemplo, tabelas, matrizes, listas e gráficos (iremos explorar esses controles nos próximos capítulos). Cada componente de listagem de dados pode ser relacionado a um DataSet diferente e pode ser agrupado de maneira independente.

Uma observação importante é que não podemos utilizar expressões globais dentro da área de detalhes (como informações de paginação). Veremos maiores detalhes sobre expressões no **Capítulo 9 (Utilizando expressões)**.

## Cabeçalho e rodapé

O cabeçalho e rodapé de um relatório são informações que devem ser repetidas na margem superior ou inferior de cada página (por exemplo, o título do relatório e informações de paginação).

Ao criarmos novos relatórios no Report Viewer, as seções de cabeçalho e rodapé não são exibidas automaticamente. Para ativá-las temos que clicar com o botão direito na área externa do relatório e escolher uma das opções: “*Adicionar Cabeçalho de Página*” ou “*Adicionar Rodapé de Página*”:



**Figura 5.1. Adicionando cabeçalho e rodapé**

Veja na figura abaixo como fica um exemplo de relatório com um título no cabeçalho e a contagem de páginas no rodapé:

Lista de Fornecedores		
ID	Nome	Categoria
[IDFornecedor]	[Nome]	[Categoria]
«Expr.»		

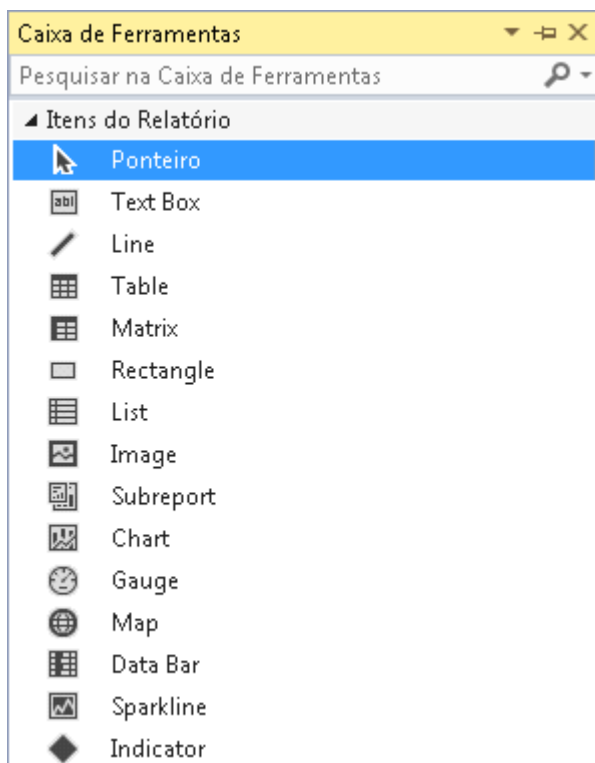
**Figura 5.2. Relatório com cabeçalho e rodapé**

Aprenderemos a criar a expressão necessária para indicar a paginação no **Capítulo 9 (Utilizando expressões)**.

## Tipos de controles

O Report Viewer conta com diversos controles que podemos utilizar dentro de nossos relatórios. Existem duas maneiras de adicionarmos controles no relatório: utilizando o menu de contexto ou utilizando a Caixa de Ferramentas.

Quando abrimos um relatório e acessamos a Caixa de Ferramentas do Visual Studio, conseguimos visualizar todos os controles disponíveis dentro da categoria "*Itens do Relatório*", conforme podemos conferir na **Figura 5.3** a seguir.



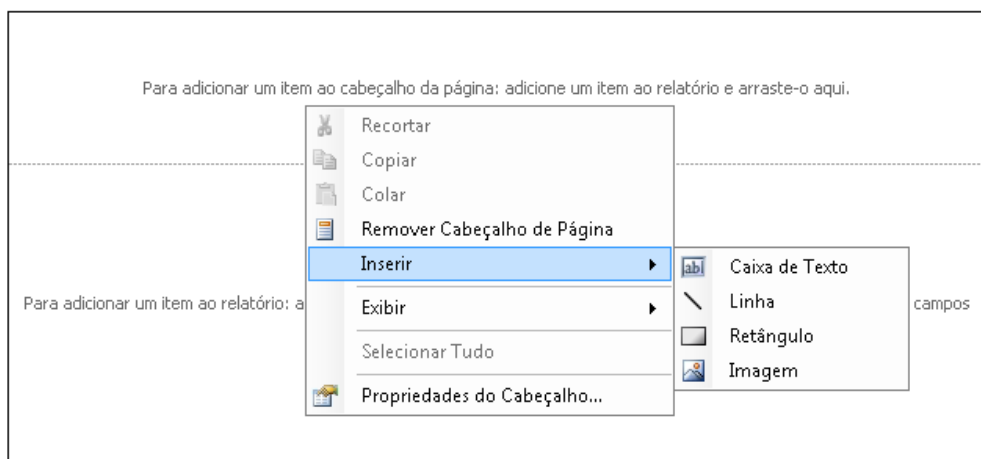
**Figura 5.3. Caixa de Ferramentas com os controles disponíveis para relatórios do Report Viewer**

Porém, nem todos os controles podem ser utilizados em todas as áreas do relatório. Por exemplo, tabelas, matrizes e listas (ou qualquer outro controle que exiba dados, como gráficos) não podem ser utilizados no cabeçalho ou rodapé do relatório. Esse tipo de controle só pode ser utilizado na área de dados do relatório.

Como conseguimos descobrir quais controles podemos utilizar em cada área do relatório? Simples, basta utilizarmos o menu de contexto do relatório.

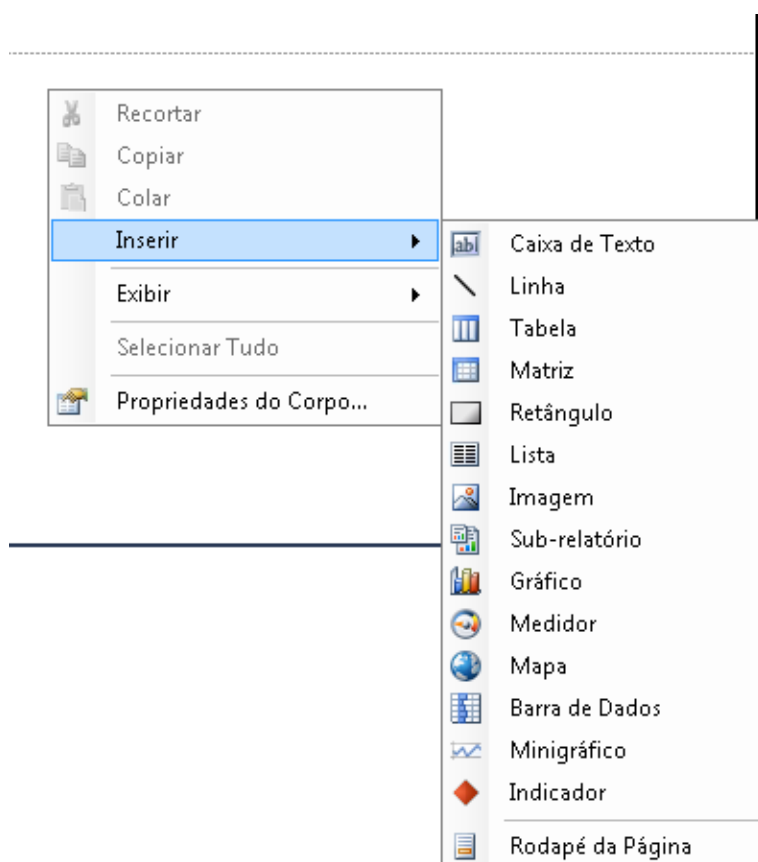
Ao clicarmos com o botão direito sobre uma área do relatório, teremos acesso ao menu de contexto. Dentro desse menu, temos um submenu chamado “Inserir”. É nesse submenu que encontraremos a lista de todos os controles que podem ser inseridos naquela área do relatório.

Por exemplo, ao acessarmos esse menu de contexto no cabeçalho ou rodapé do relatório, só teremos quatro controles disponíveis, conforme podemos observar na **Figura 5.4** a seguir.



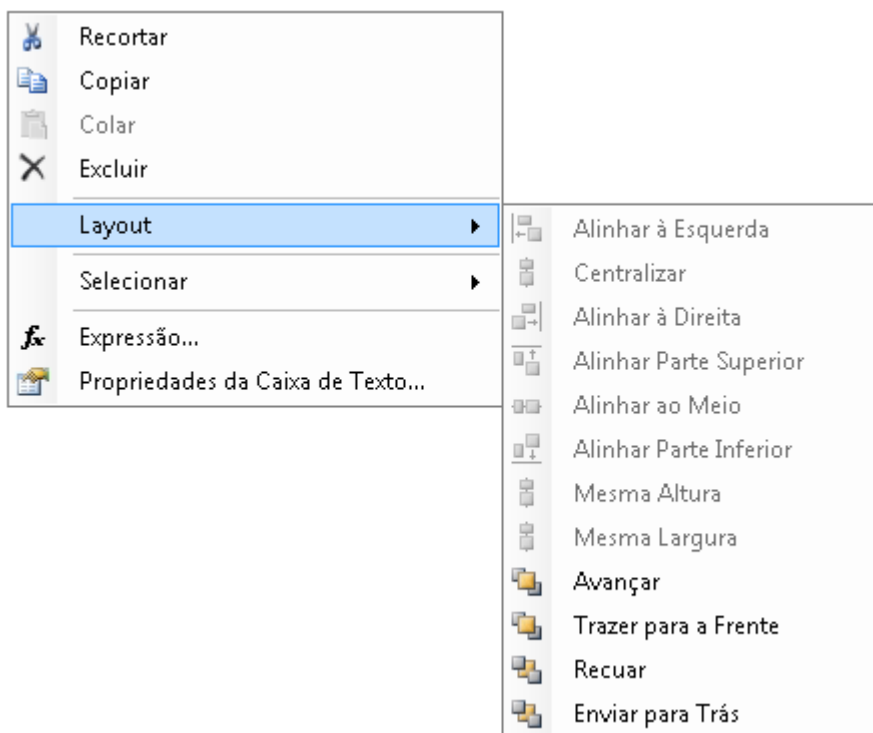
**Figura 5.4. Menu de contexto para inserir um controle no cabeçalho do relatório**

Por outro lado, se acessarmos o mesmo menu de contexto na área de dados do relatório, teremos uma lista muito maior de controles:



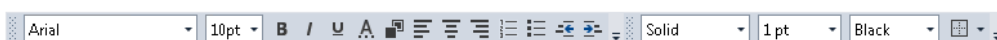
**Figura 5.5. Menu de contexto para inserir um controle no detalhe do relatório**

O alinhamento e ordem de posicionamento de cada controle pode ser alterado através do seu menu de contexto, no submenu “Layout”:



**Figura 5.6. Menu de contexto para configurar o layout de um controle**

Além disso, propriedades como fonte e bordas podem ser facilmente configuradas através da barra de ferramentas:



**Figura 5.7. Barra de ferramentas para formatação de controles**

Todas as outras propriedades podem ser configuradas na janela de propriedades, que é aberta pressionando a tecla “F4”.

## Caixa de Texto

A Caixa de Texto é o controle mais versátil do Report Viewer. Ela pode ser utilizada em qualquer área do relatório e, como o próprio nome diz, é simplesmente um container em que você pode colocar um texto ou uma expressão a ser calculada.

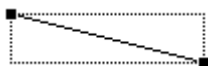


**Figura 5.8. Controle Caixa de Texto**



## Linha

Esse controle contém simplesmente uma linha, que você pode adicionar em qualquer área do seu relatório. É possível configurar a sua espessura, cor, e demais propriedades básicas. Ela é muito útil nas situações em que queremos criar uma separação entre uma área e outra do relatório (por exemplo, para separar o cabeçalho do resto do relatório).



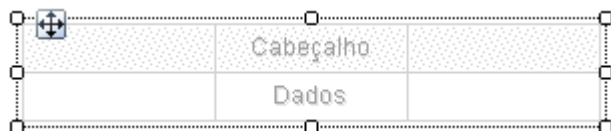
**Figura 5.9. Controle Linha**

## Tabela

Esse é talvez um dos controles mais utilizados em relatórios do Report Viewer. Como o próprio nome diz, esse controle representa uma tabela. Normalmente utilizamos o controle de tabela para exibirmos os dados do relatório de forma tabular.

É obrigatório que cada tabela esteja ligada a uma fonte de dados, portanto, logo que adicionamos uma tabela no nosso relatório, o Report Viewer pergunta qual é sua fonte de dados. Você pode conferir como criar uma fonte de dados no **Capítulo 2 (Criando o seu primeiro relatório)**.

Por padrão, o controle de tabela é criado com três colunas e duas linhas (uma de cabeçalho e outra de detalhe). Você consegue criar outras linhas ou colunas através do menu de contexto. Além disso, o controle de tabela suporta a criação de grupos, conceito que veremos em detalhes no **Capítulo 11 (Trabalhando com grupos)**.



**Figura 5.10. Controle Tabela**

## Matriz

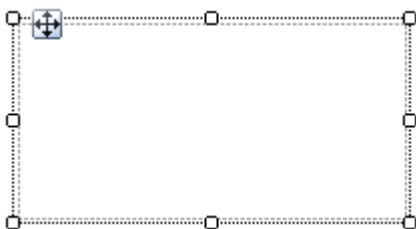
O controle de tabela suporta somente grupos de linhas. Já as matrizes suportam também grupos de colunas. As matrizes são basicamente uma extensão das tabelas com o suporte ao agrupamento por colunas. Todas as particularidades dos grupos de linhas e colunas serão apresentadas no **Capítulo 11 (Trabalhando com grupos)**.



**Figura 5.11. Controle Matriz**

## Retângulo

Utilizamos esse tipo de controle para adicionarmos uma figura geométrica em formato de retângulo nos nossos relatórios. Podemos trabalhar a formatação desse retângulo, escolhendo cores de fundo, cor da linha e propriedades da borda. Esse controle pode ser utilizado tanto na área de detalhes do relatório quanto no cabeçalho e rodapé. Normalmente utilizamos o controle Retângulo para criarmos bordas customizadas.



**Figura 5.12. Controle Retângulo**

## Lista

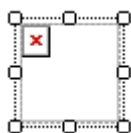
O controle de Lista é um misto entre o controle Tabela e o controle Matriz. O nome interno desse controle é Tablix (Table + Matrix = Tablix). Na realidade, os controles de Tabela e Matriz são controles Tablix com capacidades reduzidas (principalmente relacionadas a grupos de linhas e colunas). Tudo o que você consegue fazer com um controle de Tabela ou Matriz, você consegue reproduzir com o controle de Lista, o que o torna o mais poderoso e flexível dos três.



**Figura 5.13. Controle Lista**

## Imagem

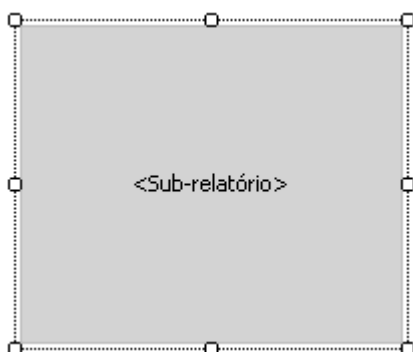
Com o controle Imagem podemos exibir figuras armazenadas em disco ou armazenadas no banco de dados (ou qualquer outro tipo de fonte de dados do relatório). Iremos abordar a fundo esse controle no **Capítulo 14 (Exibindo imagens)**.



**Figura 5.14. Controle Imagem**

## Sub-relatório

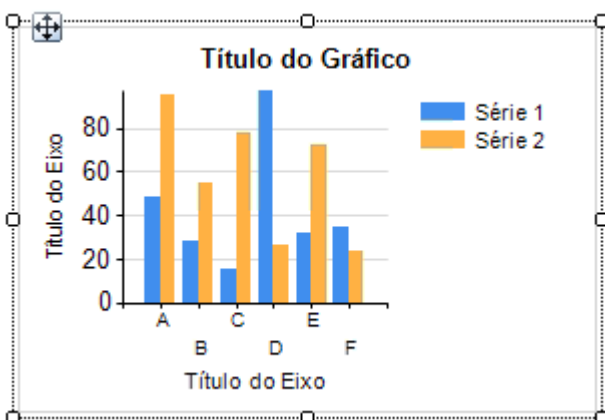
O conceito de sub-relatórios está presente em praticamente todas as ferramentas de geração de relatórios. No Report Viewer não poderia ser diferente. Em poucas palavras, com os sub-relatórios, conseguimos criar relatórios dentro de relatórios. Você encontrará uma explicação detalhada do controle de sub-relatórios no **Capítulo 13 (Mestre-detelhe, SubReports e Drill-down)**.



**Figura 5.15. Controle Sub-relatório**

## Gráfico

Com o controle de gráfico do Report Viewer, conseguimos adicionar centenas de tipos de gráficos aos nossos relatórios, desde gráficos de barras, colunas, pizza até gráficos mais complexos, como os gráficos polares. Abordaremos em detalhe o controle de gráficos no **Capítulo 15 (Criando gráficos)**.



**Figura 5.16. Controle Gráfico**

## Medidor

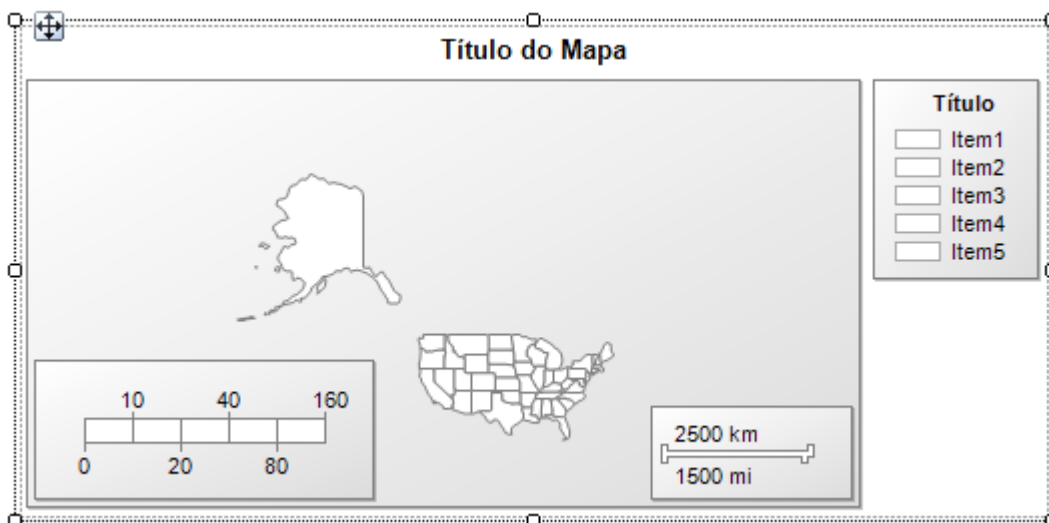
Os Medidores (também conhecidos pelo seu termo original em inglês, “*Gauges*”), são tipos de gráficos que mostram rapidamente a situação de um atributo. A forma visual dos medidores lembra os velocímetros que encontramos nos automóveis ou medidores de energia que encontramos nas casas.



**Figura 5.17. Controle Medidor**

## Mapa

O controle de mapas do Report Viewer é muito poderoso. Com ele conseguimos exibir dados geográficos de uma galeria padrão (somente mapas dos Estados Unidos estão disponíveis até o momento), arquivos de shape da ESRI (extensão shp, muito conhecida no mundo de sistemas de informação geográfica) ou consulta geoespacial do SQL Server.



**Figura 5.18. Controle Mapa**

## Barra de Dados

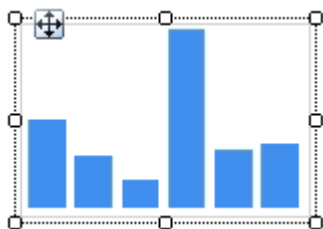
Barras de dados são um tipo simplificado de gráfico de barras ou colunas. São muito utilizados em colunas dentro dos controles de Tabela, Matriz ou Lista, a fim de mostrar o desempenho de algum indicador.



**Figura 5.19. Controle Barra de Dados**

## Minigráfico

Minigráficos são, como o próprio nome diz, gráficos em formato miniatura. O comportamento desse controle é o mesmo do controle de gráfico, mas, a escolha de tipo do gráfico é bem mais limitada e o resultado visual é muito mais simplificado.



**Figura 5.20. Controle Minigráfico**

## Indicador

Por fim, o controle Indicador possibilita a exibição de figuras pré-definidas baseadas em um valor do relatório. Com esse tipo de controle, o usuário consegue rapidamente entender se um valor do relatório está bom, médio ou ruim (ou baixo / alto, etc.). Assim como o controle de Barra de Dados, os Indicadores são normalmente utilizados dentro de colunas de Tabelas, Matrizes ou Listas.



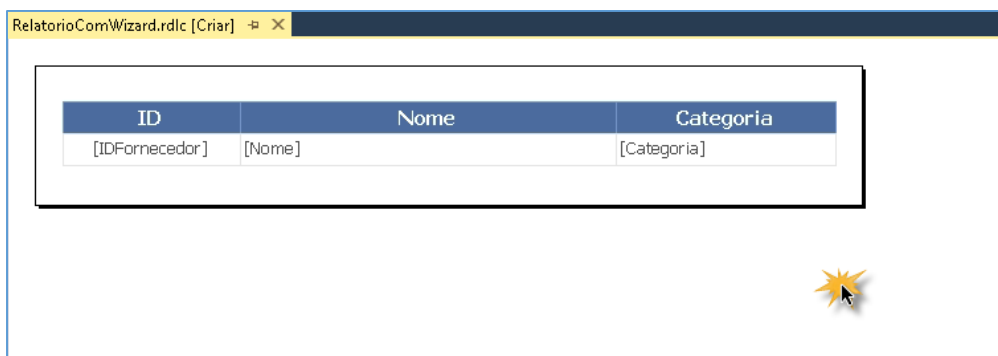
**Figura 5.21. Controle Indicador**

## Propriedades do relatório

No Report Viewer, as propriedades do relatório podem ser configuradas em dois locais: na tab de propriedades ou na janela de propriedades.

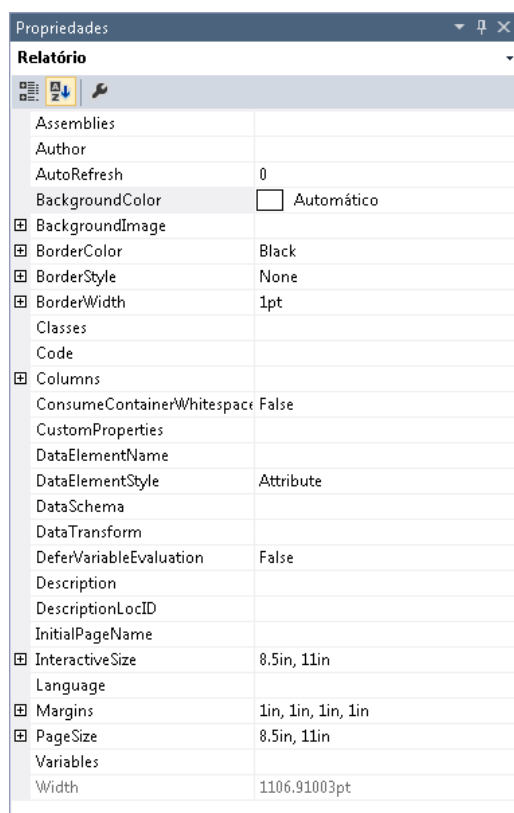
A tab de propriedades pode ser acessada ao clicarmos em alguma área fora do relatório e, em seguida, apertarmos a tecla F4. Atente-se para o fato que só será possível

acessarmos as propriedades do relatório se tivermos clicado na área correta (completamente fora do relatório, não no corpo):



**Figura 5.22. Onde devemos clicar para acessarmos as propriedades do relatório**

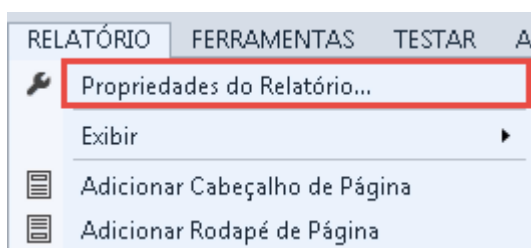
Ao clicarmos no lugar correto e apertarmos a tecla F4, o Visual Studio exibirá a tab de propriedades:



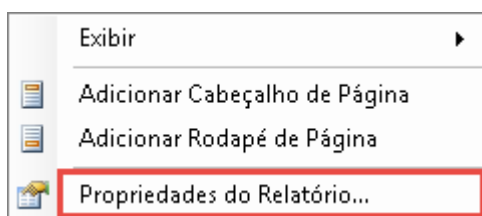
**Figura 5.23. Tab de propriedades do relatório**

A tab de propriedades do relatório contém todas as propriedades que podemos configurar nos relatórios do Report Viewer. Porém, caso você ache essa tabela muito complicada, também é possível acessar as principais propriedades do relatório através da janela de propriedades.

Para acessar a janela de propriedades, clique novamente na área apresentada na **Figura 5.22** e utilize o item “*Propriedades do relatório*” que está disponível tanto no menu principal do Visual Studio quanto no menu de contexto, como podemos conferir nas figuras abaixo:

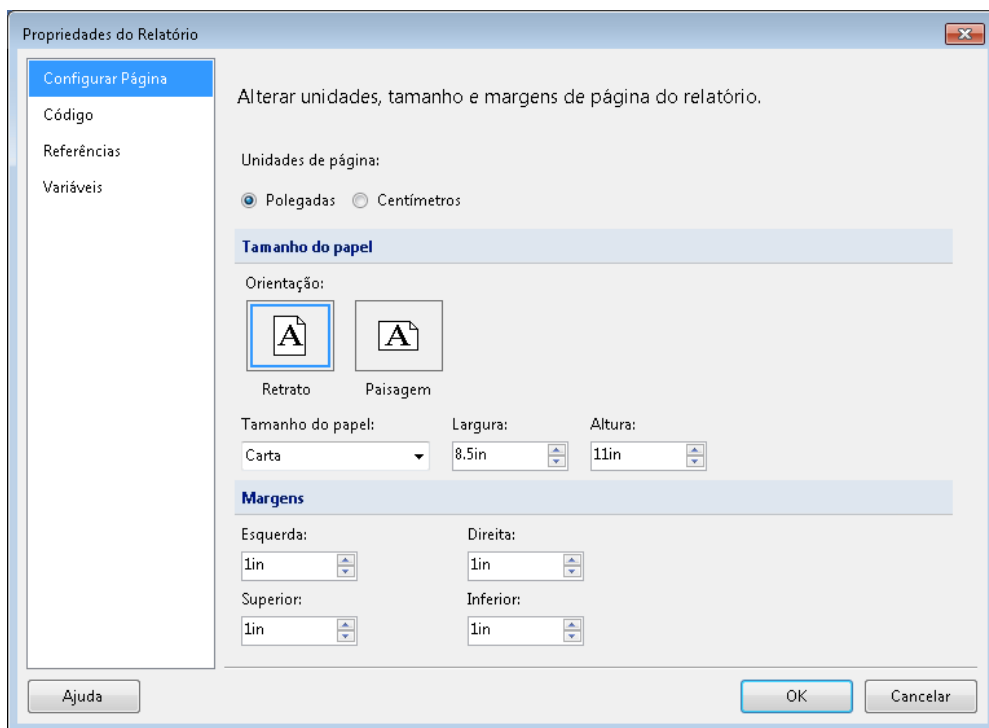


**Figura 5.24. Acessando as propriedades do relatório através do menu principal do Visual Studio**



**Figura 5.25. Acessando as propriedades do relatório através do menu de contexto**

A janela de propriedades do relatório, como mencionado anteriormente, apresenta as principais propriedades que podemos configurar nos nossos relatórios do Report Viewer. Dentre elas estão as configurações de página, códigos personalizados e variáveis, conforme demonstrado na **Figura 5.26** a seguir.

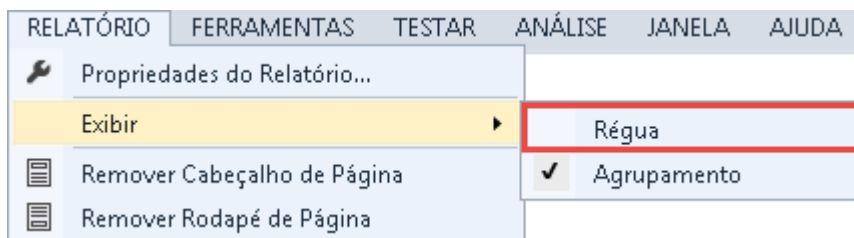


**Figura 5.26. Janela de propriedades do relatório**

## Exibindo a régua

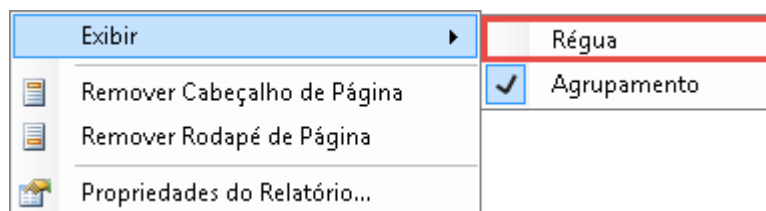
Em algumas situações precisamos de uma certa precisão ao posicionarmos os controles nos nossos relatórios. Por exemplo, quando desenvolvemos etiquetas ou quando estamos desenvolvendo relatórios que devem ser impressos em formulários pré-definidos, a precisão é essencial. Nesses casos, a régua do Report Viewer ajudará bastante.

Para ativarmos a régua (que vem desativada por padrão), podemos utilizar o menu do Report Viewer ou o menu de contexto do relatório:

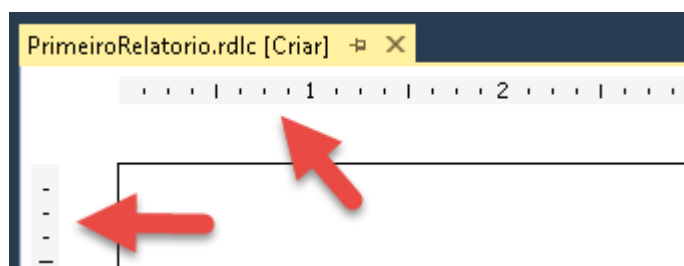


**Figura 5.27. Exibindo a régua através do menu do Report Viewer**





**Figura 5.28. Exibindo a régua através do menu de contexto**



**Figura 5.29. Resultado do relatório com a régua exibida**

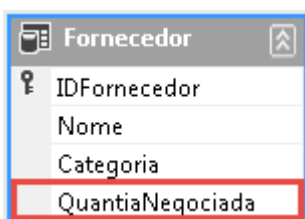
## Capítulo 6

# Criando totalizadores

A funcionalidade de totalizadores é essencial em qualquer relatório, por mais simples que ele seja. Sempre teremos que exibir, no mínimo, uma contagem de registros em um relatório de listagem.

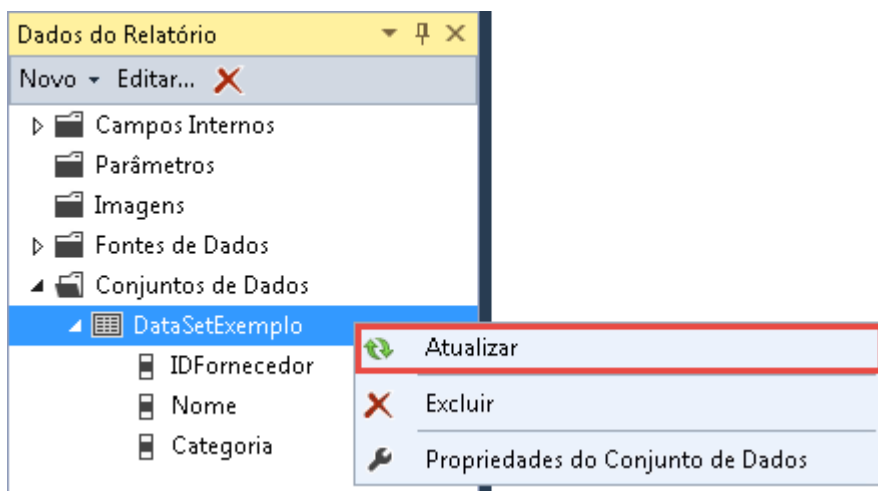
Para conferirmos como funciona a questão de totalizadores no Report Viewer, vamos evoluir o exemplo de relatório iniciado no **Capítulo 2 (Criando o seu primeiro relatório)**.

Abra o DataSetExemplo e, na tabela Fornecedor, adicione uma nova coluna do tipo Decimal chamada "QuantiaNegociada", conforme podemos conferir na figura abaixo:



**Figura 6.1. Novo campo "QuantiaNegociada" na tabela Fornecedor**

Feito isso, vá até o relatório que criamos no **Capítulo 2 (Criando o seu primeiro relatório)**, abra a janela de dados do relatório, clique com o botão direito em DataSetExemplo e escolha a opção "Atualizar":



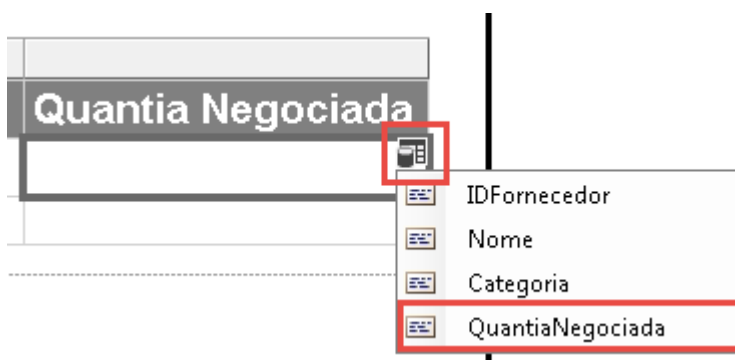
**Figura 6.2. Atualizando o DataSet do relatório**

NOTA: ALGUMAS VEZES, POR ALGUM MOTIVO DESCONHECIDO, A OPÇÃO "ATUALIZAR" NÃO SURTE EFEITO (O NOVO CAMPO NÃO APARECE NA LISTA DO DATASET). CASO ISSO ACONTEÇA COM VOCÊ, É

POSSÍVEL OBTERMOS O MESMO EFEITO CLICANDO NA OPÇÃO “PROPRIEDADES DO CONJUNTO DE DADOS” E APERTANDO O BOTÃO “OK” NA JANELA QUE SE ABRE.

Isso fará com que o relatório reconheça a nova coluna que criamos, de forma que possamos utilizá-la dentro do nosso relatório. Para isso, na tabela do relatório, clique com o botão direito na coluna “Categoria” e escolha a opção para inserirmos uma coluna à sua direita.

Com a coluna adicionada à tabela, configure o seu título para “Quantia Negociada”. Em seguida, clique no pequeno símbolo de dados na linha de detalhe e escolha a coluna “QuantiaNegociada” na lista de campos disponíveis:



**Figura 6.3. Escolhendo o campo “QuantiaNegociada” como fonte de dados para a nova coluna**

Ao fazer isso, note que o Report Viewer adicionou a totalização desse campo (**Figura 6.4**). Isso ocorreu porque o Report Viewer configura automaticamente um totalizador para todas as colunas numéricas adicionadas ao relatório:

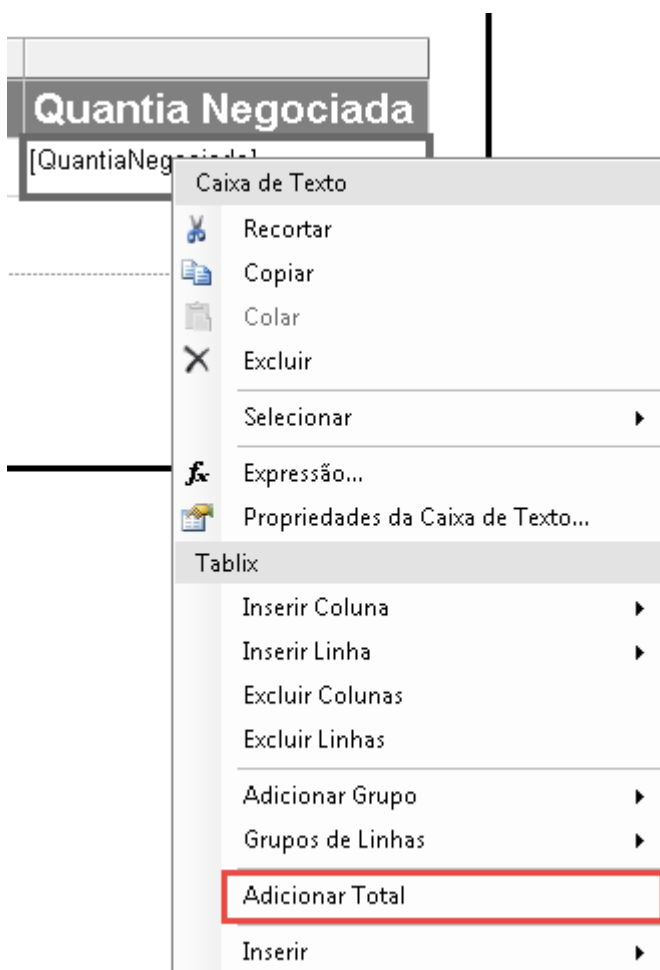
Quantia Negociada
[QuantiaNegociada]
[Sum(QuantiaNegociada)]

**Figura 6.4. Soma de “QuantiaNegociada”, adicionada automaticamente pelo Report Viewer**

Porém, essa adição automática da totalização só é efetuada pelo Report Viewer se o relatório já tiver uma linha de rodapé na tabela. Caso contrário, a totalização automática não será adicionada.

Para adicionarmos totalizadores nesses casos em que não temos a linha de rodapé no momento da configuração da coluna, basta clicarmos com o botão direito na célula correspondente à coluna que queremos criar o totalizador e escolhermos a opção “Adicionar Total”, conforme podemos conferir na **Figura 6.5**.

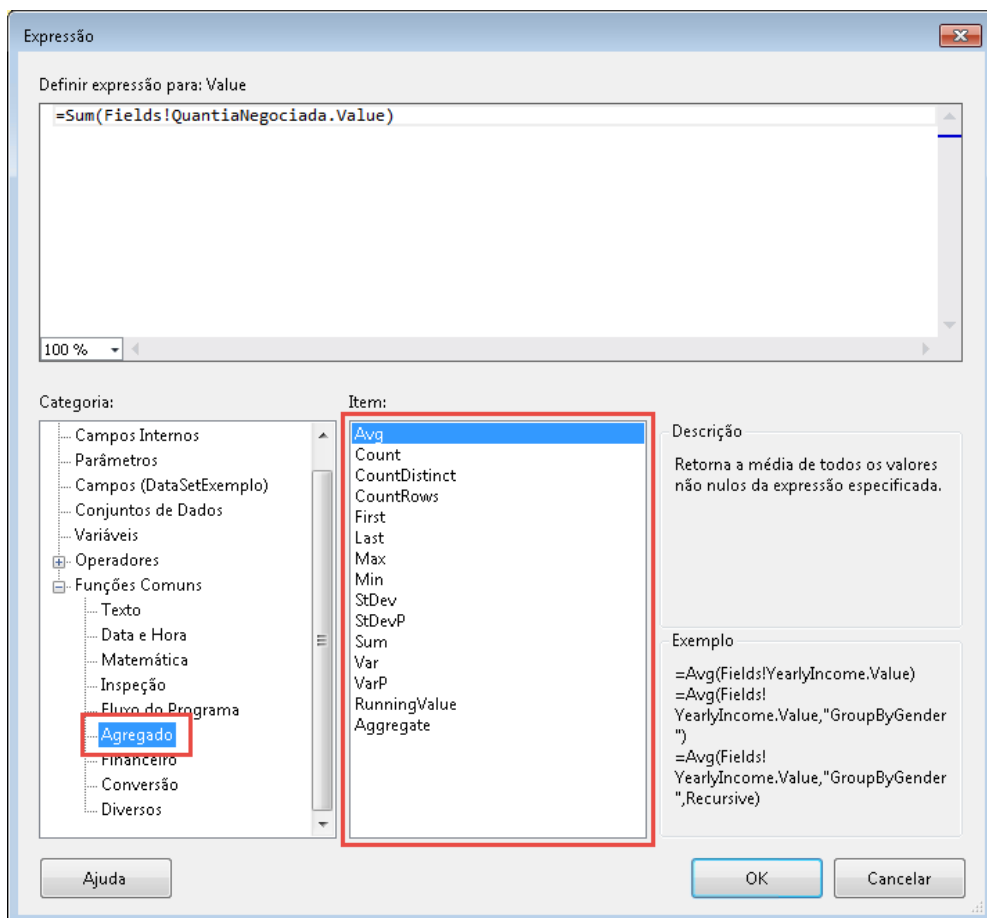
NOTA: ESTEJA ATENTO À FORMULA DE TOTALIZAÇÃO CASO O RELATÓRIO CONTENHA FILTROS, UMA VEZ QUE O REPORT VIEWER NÃO CONSIDERA OS FILTROS NO CÁLCULO DO TOTALIZADOR. DESSA FORMA, TEMOS QUE SEMPRE CONSIDERAR OS FILTROS AO MONTARMOS AS NOSSAS EXPRESSÕES TOTALIZADORAS. VOCÊ PODE CONFERIR MAIORES DETALHES SOBRE A CRIAÇÃO DE FILTROS E SUA CONSIDERAÇÃO NAS EXPRESSÕES TOTALIZADORAS NO CAPÍTULO 8 (FILTRANDO OS DADOS).



**Figura 6.5. Adicionando um totalizador através do menu de contexto da célula**

Note que o totalizador adicionado no relatório é do tipo “Sum”, ou seja, a soma de valores da coluna escolhida. Porém, obviamente esse não é o único tipo de totalizador suportado pelo Report Viewer. Para alterarmos o tipo do totalizador, basta clicarmos com o botão direito na fórmula do totalizador e escolhermos a opção “Expressão”.

Na janela “Expressão”, podemos encontrar dentro da categoria “Funções Comuns” / “Agregado” todas as opções de totalizadores disponíveis, como podemos conferir na **Figura 6.6** a seguir.



**Figura 6.6. Funções totalizadoras disponíveis no Report Viewer**

Na tabela abaixo, confira um pequeno descritivo de cada uma das funções agregadoras disponíveis no Report Viewer:

Função	Retorno
<b>Avg</b>	Média de todos os valores não nulos do conjunto
<b>Count</b>	Contagem de valores do conjunto
<b>CountDistinct</b>	Contagem de todos os valores distintos do conjunto
<b>CountRows</b>	Contagem de linhas do conjunto
<b>First</b>	Primeiro valor do conjunto
<b>Last</b>	Último valor do conjunto
<b>Max</b>	Valor máximo do conjunto
<b>Min</b>	Valor mínimo do conjunto
<b>StDev</b>	Desvio padrão de todos os valores não nulos do conjunto
<b>StDevP</b>	Desvio padrão da população de todos os valores não nulos do conjunto
<b>Sum</b>	Soma de todos os valores do conjunto
<b>Var</b>	Variação (variância) de todos os valores não nulos do conjunto

<b>VarP</b>	Variação (variância) da população de todos os valores não nulos do conjunto
<b>RunningValue</b>	Permite utilizar uma função de agregação específica em um conjunto
<b>Aggregate</b>	Agregação personalizada dependendo do provedor de dados

A fim de conseguirmos testar o relatório com essa nova coluna, temos que preenchê-la com alguns valores aleatórios. Para isso, vá até o code-behind do formulário de testes criado no **Capítulo 3 (Acessando o preview de relatórios locais)** e altere o código onde adicionamos as linhas à tabela *"Fornecedor"* conforme a listagem abaixo:

**Listagem 6.1. Ajustando o código do formulário de exemplo com valores para a coluna "QuantiaNegociada"**

```

1  DataSetExemplo.Fornecedor.AddFornecedorRow(
2      "Fornecedor 1", "Materiais de Escritório",
3      (decimal)125.34);
4  DataSetExemplo.Fornecedor.AddFornecedorRow(
5      "Fornecedor 2", "Materiais de Escritório",
6      (decimal)35.24);
7  DataSetExemplo.Fornecedor.AddFornecedorRow(
8      "Fornecedor 3", "Mão de Obra",
9      (decimal)58.56);
10 // Continua. Confira no projeto de exemplo
11 // a versão completa deste método.
```

Ao executarmos o aplicativo de exemplo, veremos que a nova coluna, bem como o seu total, é exibida com sucesso, conforme conferimos na **Figura 6.7** a seguir.



**Form Exemplo**

1 of 1 100% Find | Next

### Lista de Fornecedores

ID	Nome	Categoria	Quantia Negociada
-1	Fornecedor 1	Materiais de Escritório	125.34
-2	Fornecedor 2	Materiais de Escritório	35.24
-3	Fornecedor 3	Mão de Obra	58.56
-4	Fornecedor 4	Mão de Obra	26.24
-5	Fornecedor 5	Obra-Prima	835.73
-6	Fornecedor 6	Obra-Prima	2452.95
-7	Fornecedor 7	Materiais de Escritório	45.99
-8	Fornecedor 8	Materiais de Escritório	94.48
-9	Fornecedor 9	Mão de Obra	85.66
-10	Fornecedor 10	Mão de Obra	365.07
-11	Fornecedor 11	Obra-Prima	2.92
-12	Fornecedor 12	Obra-Prima	84.67
-13	Fornecedor 13	Materiais de Escritório	4672.8
-14	Fornecedor 14	Materiais de Escritório	34.65
-15	Fornecedor 15	Mão de Obra	92.56
			<b>9012.86</b>

**Figura 6.7. Relatório de Lista de Fornecedores com nova coluna (Quantia Negociada) e sua totalização**

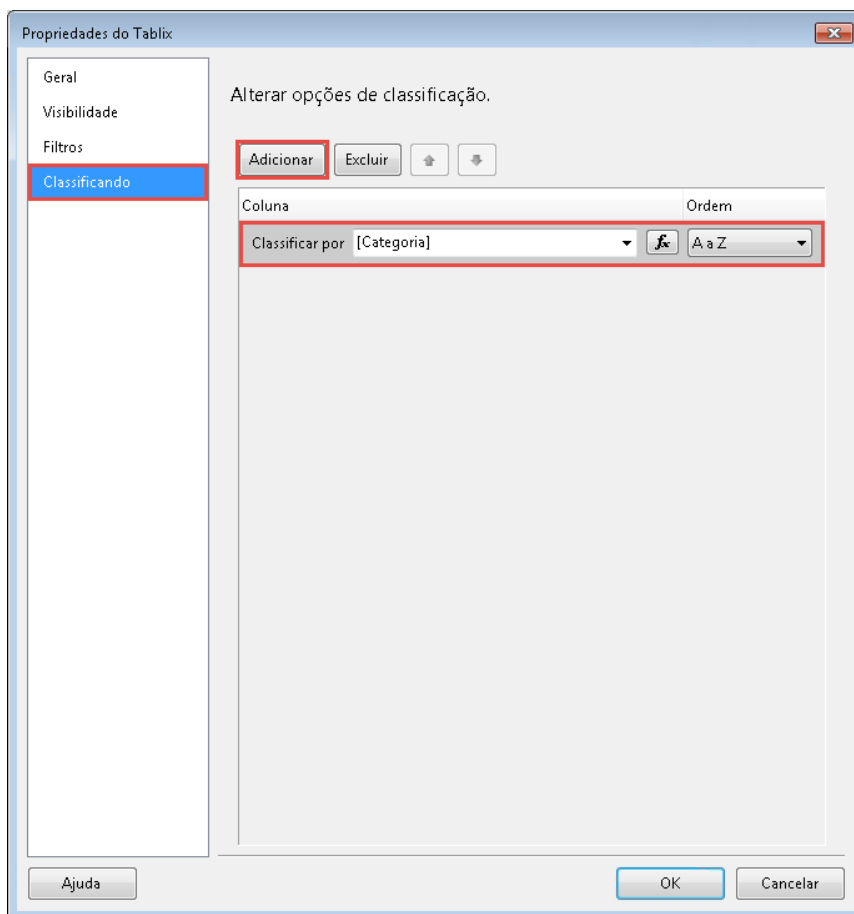
O Report Viewer comporta-se muito bem quando adicionamos totalizadores dentro de grupos (o que não é o caso de outras ferramentas geradoras de relatórios). Normalmente, o Report Viewer consegue detectar automaticamente qual grupo deve ser considerado na totalização dos registros.

## Capítulo 7

# Classificação (ordenação) de dados

Por padrão, o Report Viewer ordena os registros utilizando a ordem original dos dados no DataSet. Porém, é muito comum que queiramos configurar uma ordenação customizada para o relatório.

Os controles Tabela, Matriz e Tablix suportam a ordenação de seu conjunto de dados. Configurar a ordenação nesses controles é muito simples: basta acessarmos as suas propriedades e configurarmos os campos na página “Classificando”. Por exemplo, confira na figura abaixo como ficaria para ordenarmos o relatório construído no **Capítulo 6 (Criando totalizadores)** pela coluna “Categoria”:



**Figura 7.1. Ordenando o relatório pela coluna “Categoria”**



Note que é possível configurarmos a ordenação ascendente (A a Z) ou descendente (Z a A). Além disso, podemos adicionar várias colunas na lista de ordenação do controle.

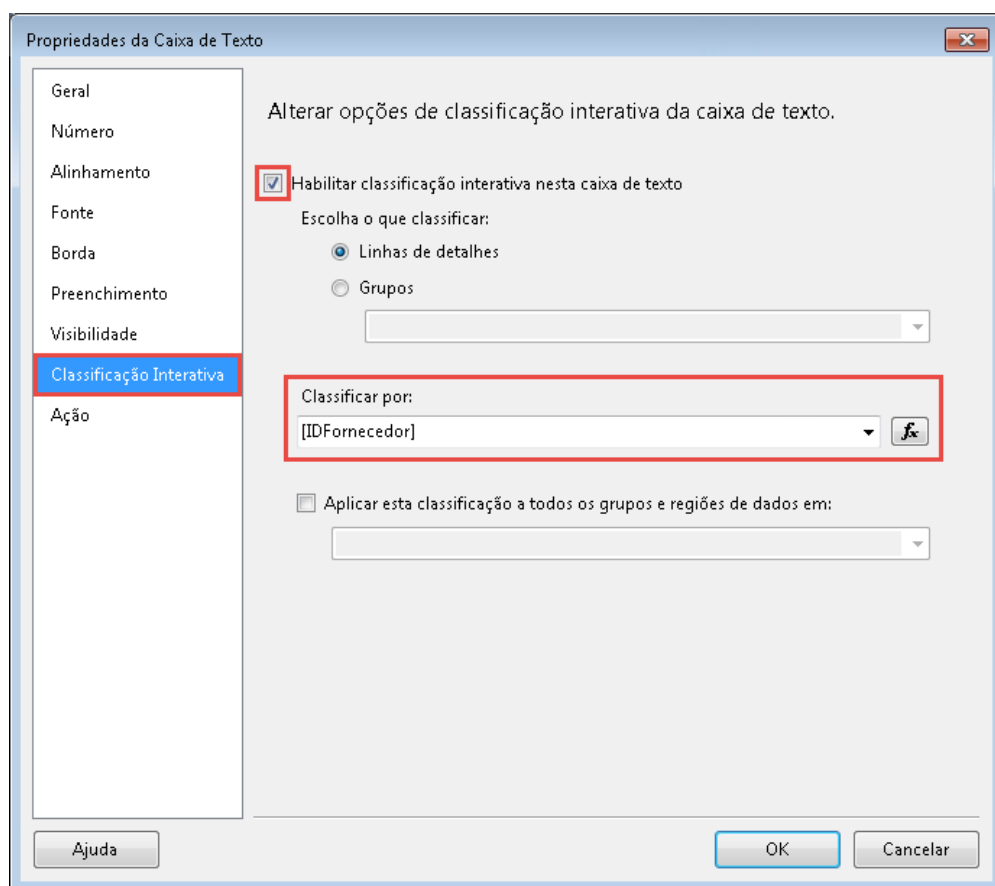
Outro ponto importante a ser ressaltado é que, através do botão “fx”, conseguimos utilizar uma expressão completamente customizada para fazer a ordenação da Tabela, Matriz ou Tablix.

## Classificação interativa

A ordenação (ou classificação) que conferimos até agora é a ordenação que será utilizada como padrão no momento da exibição do relatório. Porém, o Report Viewer conta com uma funcionalidade muito interessante chamada “*Classificação interativa*”. Com ela, o usuário pode alterar, em tempo de execução, a ordenação dos dados do controle.

Para adicionarmos a funcionalidade de classificação interativa, devemos clicar com o botão direito na célula do cabeçalho do controle e, no menu de contexto, devemos escolher a opção “*Propriedades da Caixa de Texto*”.

Na janela “*Propriedades da Caixa de Texto*”, vá até a categoria “*Classificação Interativa*”, cheque a opção “*Habilitar classificação interativa nesta caixa de texto*” e escolha o campo correspondente na caixa de opções “*Classificar por*”. Confira na **Figura 7.2** um exemplo em que configuramos a classificação interativa na caixa de texto do “*ID do Fornecedor*”.



**Figura 7.2. Adicionando classificação interativa na caixa de texto da coluna “ID do Fornecedor”**

Execute os mesmos procedimentos para as outras caixas de texto e, em tempo de execução, você terá um resultado similar ao apresentado na figura abaixo (note o destaque para os controles de classificação interativa nas células do cabeçalho da tabela):

### Lista de Fornecedores

ID	Nome	Categoria	Quantia Negociada
-3	Fornecedor 3	Mão de Obra	58.56
-4	Fornecedor 4	Mão de Obra	26.24
-9	Fornecedor 9	Mão de Obra	85.66

**Figura 7.3. Relatório em tempo de execução após adicionarmos a classificação interativa**

Por motivos óbvios, os controles da classificação interativa só aparecem no modo de visualização. Ao alterarmos o relatório para o modo de impressão, esses controles são

ocultados (porém, a ordenação configurada pelo usuário no modo de visualização é mantida quando alteramos para o modo impressão).

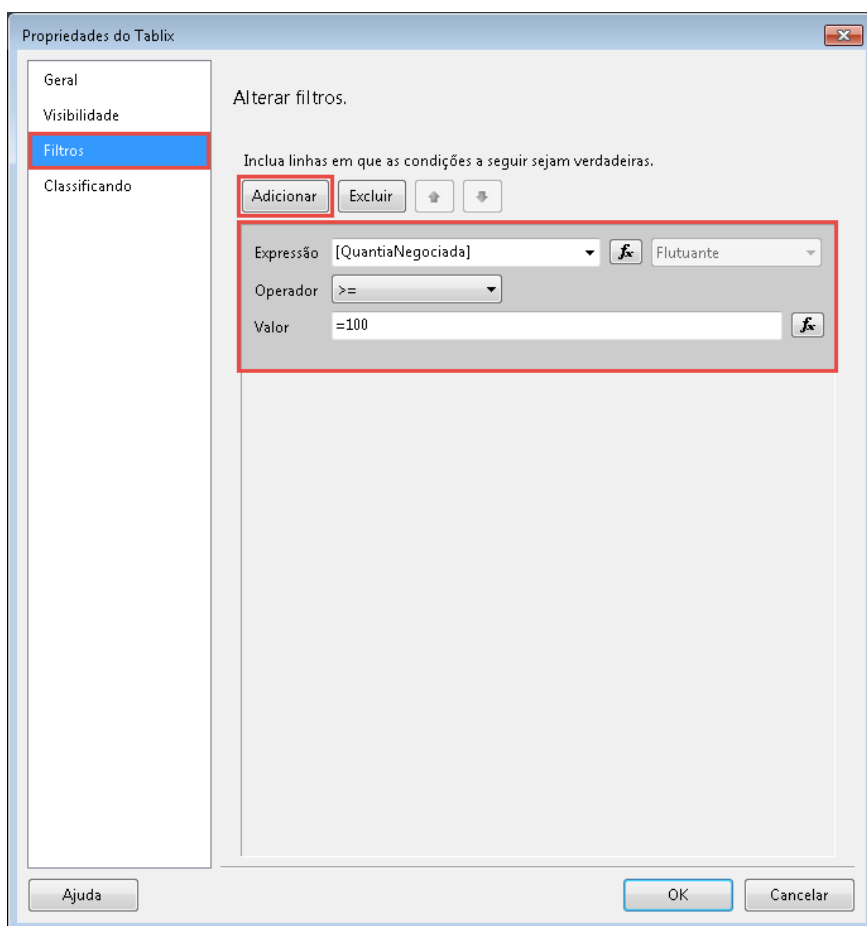
## Capítulo 8

# Filtrando os dados

Os controles de Tabela, Matriz e Tablix possibilitam que seus dados sejam filtrados. Dessa forma, mesmo que a fonte de dados tenha mais registros do que queremos exibir, podemos filtrá-los sem nenhuma dificuldade.

Na janela de propriedades desses controles, conseguimos adicionar expressões que serão avaliadas pelo mecanismo do Report Viewer em tempo de execução a fim de filtrar os dados exibidos.

Configurar filtros para os controles Tabela, Matriz e Tablix é muito simples. Basta acessar a janela de propriedades, ir até a página “Filtros” e adicionar o filtro desejado. Por exemplo, observe na **Figura 8.1** como podemos adicionar um filtro ao nosso relatório de forma que somente os fornecedores com quantia negociada maior ou igual a 100 sejam exibidos:



**Figura 8.1. Configurando um filtro no controle Tablix**

Note que, quando o campo utilizado no filtro tiver o tipo “decimal” ou “double” (como é o caso do nosso campo “QuantiaNegociada”), temos que especificar o valor em forma de expressão (“=100”, e não somente “100”). Caso contrário, o Report Viewer não conseguirá converter automaticamente o valor de um tipo para o outro e o relatório não será exibido.

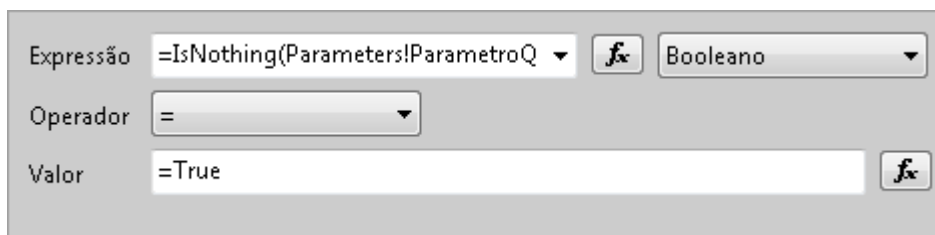
É possível adicionarmos mais de uma condição de filtro, porém, o operador entre as diversas condições sempre será o “e” (por exemplo, se condição 1 e condição 2 e condição 3 forem verdadeiras). Porém, como os filtros suportam expressões, também conseguimos criar filtros com múltiplas condições com o operador “ou” sem muita dificuldade.

Uma situação clássica em que precisamos utilizar o operador “ou” em condições do filtro é quando utilizamos parâmetros não obrigatórios no nosso relatório e só queremos filtrar os dados caso o parâmetro tenha sido fornecido. Nesse caso, configuramos o filtro com uma expressão conforme podemos conferir na **Listagem 8.1** abaixo:

#### Listagem 8.1. Configurando um filtro baseado em parâmetro não obrigatório

```
1 =IsNothing (Parameters!ParametroQuantiaNegociada.Value) Or  
2 Fields!QuantiaNegociada.Value >=  
3 Parameters!ParametroQuantiaNegociada.Value
```

Feito isso, basta configurarmos o filtro com o tipo booleano e valor “=True”:



**Figura 8.2. Condição de filtro do Tablix utilizando um parâmetro não obrigatório**

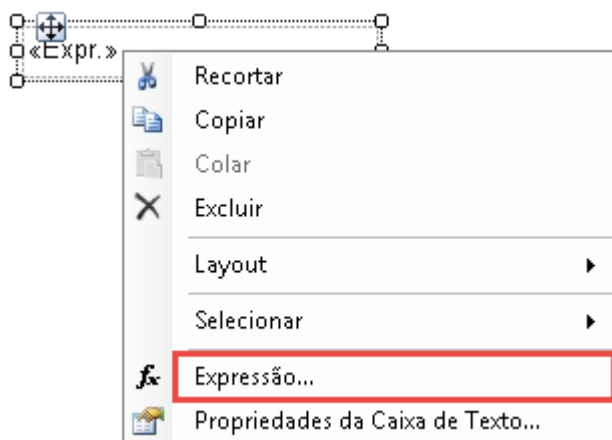
Abordaremos mais detalhes sobre a utilização de parâmetros no **Capítulo 12 (Adicionando parâmetros)**. Outra questão muito importante que será abordada nesse mesmo capítulo é a forma com que o Report Viewer lida com os somatórios de dados filtrados. Como o Report Viewer não considera os filtros ao aplicar a fórmula de sumarização, temos que ajustar as nossas expressões totalizadoras de forma que eles sejam considerados.

## Capítulo 9

# Utilizando expressões

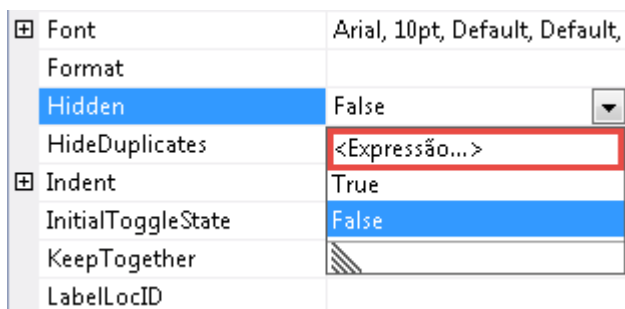
Praticamente todos os controles e propriedades dos relatórios desenvolvidos com o Report Viewer suportam expressões. Conseguimos utilizar expressões tanto em situações muito simples (como para fazer a concatenação de dois campos) quanto em situações mais complexas (como para configurar a cor ou visibilidade de um controle de forma dinâmica – formatação condicional).

Para adicionarmos expressões em um campo texto ou em uma célula de uma Tabela, Matriz ou Tablix, devemos clicar com o botão direito no local em que queremos adicionar a expressão e, no menu de contexto, devemos escolher a opção “Expressão”:



**Figura 9.1. Configurando uma expressão no controle Caixa de Texto**

Por outro lado, para configurarmos expressões em propriedades dos controles, basta acessarmos a janela de propriedades e abriremos a caixa de opções da propriedade desejada, escolhendo a opção “Expressão”. É muito comum utilizarmos expressões em propriedades para fazermos formatação condicional, onde escondemos ou não um campo dependendo de uma expressão (ou escolhemos a cor de um campo texto baseando-se em um outro valor). Veja na **Figura 9.2** o local onde conseguimos configurar uma expressão para a propriedade “Hidden” de um controle qualquer.



**Figura 9.2.** Configurando uma expressão na propriedade “Hidden” de um controle

## Estruturas de decisão

O grupo de expressões que implementa estruturas de decisão é um dos mais utilizados no desenvolvimento de relatórios (em qualquer plataforma). Dentro dessa categoria de expressões, a mais utilizada sem dúvida é a cláusula “*if*”.

No Report Viewer, ao invés dessa estrutura de decisão chamar-se “*if*”, ela chama “*IIf*” (dois “*is*” antes do “*f*” – provavelmente herança do Visual Basic, também presente no Excel). Sua utilização é muito simples: o primeiro parâmetro é uma condição, o segundo parâmetro é o valor que deve ser considerado caso a condição seja verdadeira e o terceiro parâmetro é o valor que deve ser considerado caso a condição seja falsa.

Também é possível utilizarmos um “*IIf*” dentro de outro “*IIf*”. Por exemplo, confira na **Listagem 9.1** como fica a expressão para adicionarmos um campo calculado no nosso relatório, baseado no campo “*QuantiaNegociada*”, em que seu valor seja “*Atenção*” caso a quantia negociada seja menor que 100, “*OK*” caso o seu valor esteja entre 100 e 1000 ou “*Ótimo*” caso a quantia negociada seja maior que 1000.

### Listagem 9.1. Expressão retornando os valores “Atenção”, “OK” ou “Ótimo” baseando-se no campo “QuantiaNegociada”

```
1  =IIf(Fields!QuantiaNegociada.Value < 100, "Atenção",
2      IIf(Fields!QuantiaNegociada.Value < 1000, "OK",
3          "Ótimo"))
```

Outra estrutura de decisão bastante comum é o “*Switch*”, muito utilizado nas mais diversas linguagens de programação. Com o “*Switch*” conseguimos avaliar diversas condições e indicar o valor de retorno que deve ser utilizado para cada uma das condições. Por exemplo, veja na **Listagem 9.2** como ficaria a expressão que retorna uma abreviação da coluna “*Categoria*”.

**Listagem 9.2. Expressão utilizando “Switch” para retornar uma abreviação da coluna “Categoria”**

```
1 =Switch(  
2 Fields!Categoria.Value = "Materiais de Escritório", "ME",  
3 Fields!Categoria.Value = "Mão de Obra", "MO",  
4 Fields!Categoria.Value = "Obra-Prima", "OP")
```

Um uso bastante recorrente de estruturas de decisão é para a criação de formatações condicionais. Esse tema será abordado no **Capítulo 10 (Formatação de dados e formatação condicional)**.

## Totalizadores

Outra categoria muito comum de expressões são os totalizadores. As expressões utilizadas para fazermos totalização de dados foram abordadas no **Capítulo 6 (Criando totalizadores)**.

## Datas

O Report Viewer contém uma infinidade de expressões relacionadas ao tópico data e hora. A mais comum de todas é a expressão “Now”, que retorna a data e hora atuais.

Além da expressão “Now”, outras expressões muito utilizadas são as que retornam partes específicas de um valor data/hora. Por exemplo, a expressão “Day” retorna o dia de uma data específica. A expressão “Month” retorna o mês de uma data específica, e assim por diante.

Para adicionarmos dias ou horas a uma data, podemos utilizar a expressão “DateAdd”. Essa expressão recebe o intervalo a ser utilizado (“Hour” para adicionar horas, “Day” para adicionar dias, e assim por diante), o valor para o intervalo e a data original. Por exemplo, para retornarmos a data atual + 30 dias, poderíamos utilizar esta expressão:

**Listagem 9.3. Expressão utilizada para adicionarmos 30 dias à data atual**

```
1 =DateAdd(DateInterval.Day, 30, Today)
```

## Formatação de strings

Além de ser possível concatenarmos strings através do operador “&”, também podemos formatá-las e ajustá-las com algumas outras expressões. Por exemplo, para formatar uma string utilizamos a expressão “Format”, sendo que o primeiro parâmetro é o valor a ser formatado e o segundo parâmetro é o formato. Para formatarmos o valor a ser exibido nas células da coluna “Quantia Negociada” aplicando o formato de moeda, utilizamos a expressão apresentada na **Listagem 9.4** a seguir.



### Listagem 9.4. Formatando o campo “Quantia Negociada” aplicando o formato de moeda

**1**     `=Format (Fields!QuantiaNegociada.Value, "c")`

NOTA: TAMBÉM CONSEGUIMOS CONFIGURAR O FORMATO DOS CAMPOS ATRAVÉS DAS PROPRIEDADES DO TEXTBOX. PARA ISSO, CLIQUE COM O BOTÃO DIREITO NO TEXTBOX (OU CÉLULA DO TABLIX), ESCOLHA A OPÇÃO “PROPRIEDADES DA CAIXA DE TEXTO” E, NA JANELA QUE SE ABRE, CONFIGURE O FORMATO DESEJADO NA ABA “NÚMERO”.

Confira a lista de formatos numéricos suportados na tabela abaixo:

Formato	Retorno
<b>C ou c</b>	Formato moeda (R\$ 15,74)
<b>D ou d</b>	Formato decimal (1234)
<b>E ou e</b>	Formato científico (1,34e+003)
<b>F ou f</b>	Formato de ponto flutuante (15,74)
<b>G ou g</b>	Formato geral (compara os formatos de ponto flutuante e científico, retornando o resultado mais compacto entre esses dois formatos)
<b>N ou n</b>	Formato numérico (formato de ponto flutuante, porém, com separadores de milhares, caso a cultura da aplicação os tenha)
<b>P ou p</b>	Formato percentual (17,74%)
<b>R ou r</b>	Formato round-trip (possibilita a conversão de um número em texto de forma que ele possa ser convertido de volta a número sem perder nenhuma precisão)
<b>X ou x</b>	Formato hexadecimal (00ff)

Para cada um dos formatos numéricos apresentados na tabela acima, podemos especificar também a quantidade de casas a serem consideradas. Por exemplo, para arredondarmos um valor em uma casa decimal utilizando o formato decimal, a expressão ficaria assim:

### Listagem 9.5. Informando o número de casas decimais com o formato decimal

**1**     `=Format (Fields!QuantiaNegociada.Value, "d2")`

Com essa expressão, conseguimos também formatarmos datas. Veja a lista completa de formatos suportados na tabela abaixo:

Formato	Retorno
<b>d</b>	Formato de data curto (31/12/2015)
<b>D</b>	Formato de data longo (quinta-feira, 31 de dezembro de 2015)
<b>t</b>	Formato de hora curto (15:02)
<b>T</b>	Formato de hora longo (15:02:35)

Formato	Retorno
<b>f</b>	Formato de data longo com hora curto (quinta-feira, 31 de dezembro de 2015 15:02)
<b>F</b>	Formato de data longo com hora longo (quinta-feira, 31 de dezembro de 2015 15:02:35)
<b>g</b>	Formato de data curto com hora curto (31/12/2015 15:02)
<b>G</b>	Formato de data curto com hora longo (31/12/2015 15:02:35)
<b>M ou m</b>	Número do mês (com dois dígitos forçados caso o formato seja o "M" maiúsculo)
<b>R ou r</b>	Formato RFC1123 (Thu, 31 Dec 2015 15:02:35 GMT)
<b>Y ou y</b>	Número do ano (com dois dígitos caso o formato seja "y" minúsculo ou quatro dígitos caso seja "Y" maiúsculo)

Além dos formatos padrão apresentados na tabela acima, também é possível utilizarmos formatos customizados (os mesmos que utilizamos no C#) como, por exemplo, "dd/MM/yyyy".

Por fim, temos também a expressão "Right", utilizada para retornarmos os valores de uma string de trás pra frente (por exemplo, os três últimos caracteres de uma string); a expressão "Left", para retornarmos os valores à esquerda da string (por exemplo, os três primeiros caracteres de uma string); a expressão "Len", utilizada para descobrirmos a quantidade de caracteres de uma string; e a expressão "InStr" que serve para descobrirmos a primeira posição de um caractere específico dentro de uma string.

## Informações do relatório

Outro tipo de expressão muito útil são as que nos proporcionam informações gerais do relatório. Por exemplo, podemos utilizar a expressão "ReportName" para descobrirmos o nome do relatório e a expressão "ExecutionTime" para descobrirmos a hora em que o relatório foi gerado.

Duas expressões que aparecem praticamente em todos os relatórios e se enquadram nessa categoria são as expressões "PageNumber" e "TotalPages". Elas retornam, respectivamente, o número da página atual e o número total de páginas. Portanto, para colocarmos a informação "Página X de Y" no rodapé do relatório, podemos utilizar a expressão abaixo:

### Listagem 9.6. Expressão para mostrarmos a página atual e a quantidade total de páginas

```
1  ="Página " & Globals!PageNumber &
2  " de " & Globals!TotalPages
```

É importante notar que essas expressões se encontram dentro do grupo de expressões globais (por isso elas começam com *"Globals."*). Por esse motivo, essas expressões só podem ser utilizadas no cabeçalho e rodapé do relatório, e não na área de detalhes.

Além das expressões apresentadas anteriormente, temos também duas outras expressões globais muito úteis, que são as expressões *"First"* e *"Last"*. Como o próprio nome diz, elas servem para retornar o primeiro e o último registro da fonte de dados.

## Códigos customizados

As expressões abordadas até agora muitas vezes são o suficiente para gerarmos um relatório de qualidade. Porém, algumas vezes precisamos de lógicas mais complexas. Pensando nisso, a Microsoft possibilita a utilização de métodos customizados nas expressões do Report Viewer (métodos estáticos implementados em .NET). O processo para utilizar códigos customizados é extenso e não será abordado neste livro.

Porém, algo muito interessante que irei apresentar é a utilização de métodos do namespace `Microsoft.VisualBasic` e a utilização das classes `System.Convert` e `System.Math`. Esse namespace e classes são referenciadas automaticamente pelo Report Viewer, portanto, é possível utilizá-las nas nossas expressões sem nenhum esforço. Por exemplo, suponha que queiramos retornar o valor absoluto do campo *"Quantia Negociada"* (valor absoluto significa que sempre teremos um valor positivo, ou seja, mesmo se o valor original for negativo, ele perderá o sinal e virará positivo). Nesse caso, podemos utilizar o método *"Abs"* da classe *"Math"*, conforme conferimos na **Listagem 9.7** abaixo:

### Listagem 9.7. Utilizando a classe Math do .NET dentro de expressões no relatório

```
1 =System.Math.Abs (Fields!QuantiaNegociada.Value)
```

## Capítulo 10

# Formatação de dados e formatação condicional

Os conceitos de totalizadores, classificação, filtro e expressões foram abordados nos capítulos anteriores. Neste capítulo, veremos outro tópico muito importante que certamente precisaremos dominar ao desenvolvermos relatórios com o Report Viewer: a formatação condicional.

Formatação condicional nada mais é que aplicar uma formatação específica em objetos do relatório caso uma certa condição seja atendida.

Por exemplo, no relatório que construímos nos capítulos anteriores, temos o campo *"Quantia Negociada"*. Aprendemos, no **Capítulo 9 (Utilizando expressões)**, como criar uma expressão que analise o campo *"Quantia Negociada"* de forma que ele retorne *"Ótimo"*, *"OK"* ou *"Atenção"*, dependendo do valor desse campo. E se quisermos alterar a cor das células da coluna *"Quantia Negociada"* dependendo do seu valor? É justamente aí que entra a formatação condicional.

A formatação condicional no Report Viewer se dá pela criação de expressões nas propriedades visuais dos controles (como Fonte, Cor, Visibilidade, etc). Para colorirmos as células da coluna *"Quantia Negociada"* dependendo do seu valor, devemos criar uma fórmula muito parecida com a apresentada no **Capítulo 9 (Utilizando expressões)**, quando criamos o campo *"Status"*. Porém, ao invés de retornarmos uma string contendo o *"Status"*, iremos retornar strings com a representação da cor que queremos para aquele determinado intervalo.

Por exemplo, para colorirmos as células da coluna *"Quantia Negociada"* de vermelho caso o seu valor seja menor do que 100, preto caso o seu valor esteja no intervalo entre 100 e 1000, e verde caso o seu valor seja maior do que 1000, vá até a célula de detalhe da coluna *"Quantia Negociada"*, abra a propriedade *"Color"* e clique em *"Expressão"*, conforme demonstrado na **Figura 10.1** a seguir.



**Figura 10.1.** Abrindo o editor de expressões para a propriedade Color

No editor de expressões aberto pelo Report Viewer, utilize a seguinte fórmula:

**Listagem 10.1. Fórmula para formatação condicional da cor das células da coluna “Quantia Negociada”**

```

1  =IIf (Fields!QuantiaNegociada.Value < 100, "Red",
2      IIf (Fields!QuantiaNegociada.Value < 1000, "Black",
3          "Green"))

```

Note que a expressão é praticamente idêntica à apresentada no **Capítulo 9 (Utilizando expressões)**, porém, nesse caso, retornamos os nomes das cores que queremos utilizar para cada intervalo. É importante observar que o nome das cores deve estar, obviamente, em inglês, além de obrigatoriamente ter que estar dentro de aspas duplas.

Poderíamos parar por aqui ao abordarmos o tema de formatação condicional, afinal de contas, basta aplicarmos o exemplo apresentado anteriormente com outras propriedades (como Hidden, Font, BorderStyle, BorderColor, etc) para criarmos outros tipos de formatação condicional.

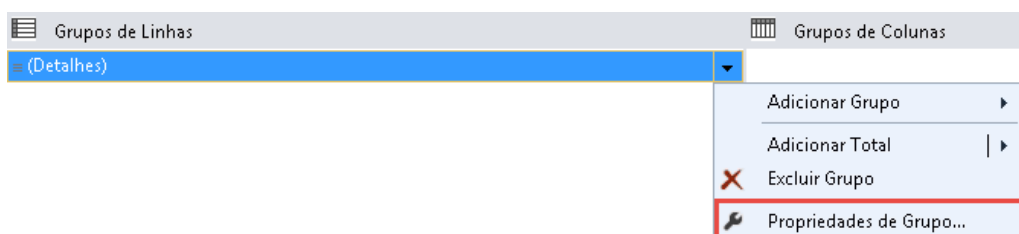
Porém, um ponto muito importante e pouco abordado na literatura sobre Report Viewer é a criação de variáveis para fazermos formatação condicional nos nossos relatórios.

Pense na seguinte situação: e se quisermos aplicar a mesma formatação condicional (apresentada na **Listagem 10.1**) nas células da coluna “Status”? Uma opção é copiarmos a mesma expressão e utilizarmos na expressão da propriedade “Color” da coluna “Status”. Infelizmente, apesar de ser a opção mais fácil, essa é a pior opção possível. Copiar e colar expressões em outros controles faz com que a manutenção de nossos relatórios se torne um pesadelo.

Para utilizarmos a mesma expressão em mais de uma coluna de um Tablix, podemos criar uma variável dentro do agrupamento. Essa variável conterá a expressão da formatação condicional, que então poderá ser utilizada em qualquer célula do agrupamento.

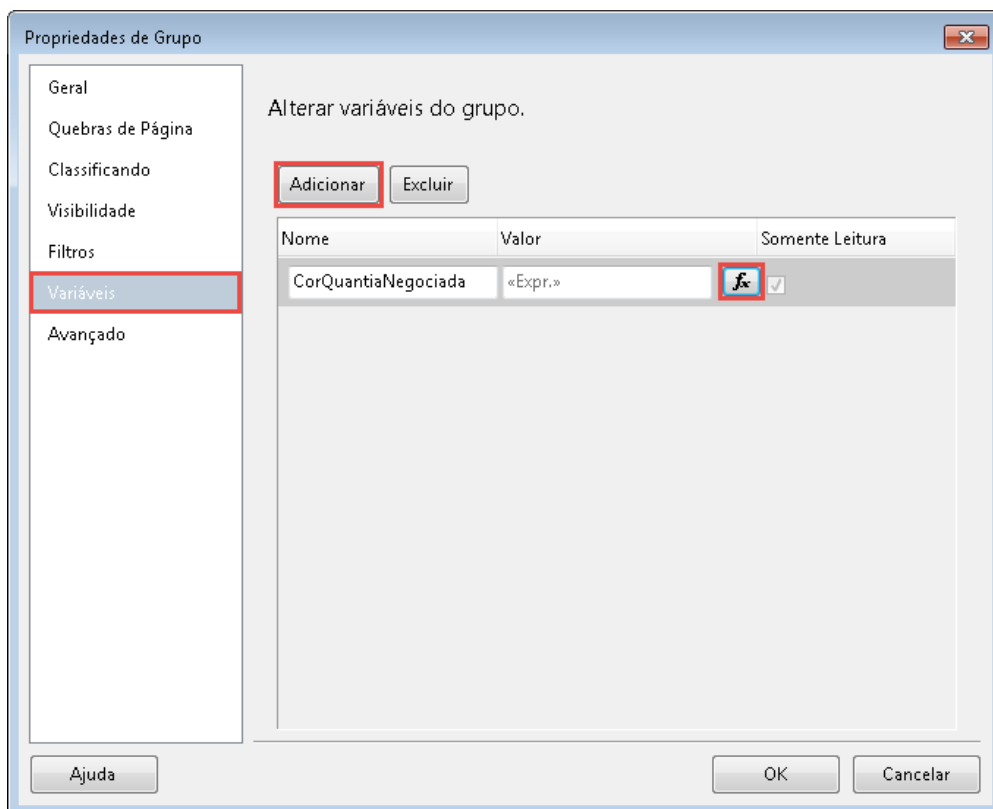
Mesmo que o Tablix não tenha nenhum grupo, o Report Viewer considera a área de detalhes como um grupo especial. Dessa forma, podemos criar variáveis dentro desse grupo especial e que serão disponibilizadas em qualquer célula da área de detalhes do Tablix.

Clique no item “(Detalhes)” na lista de “Grupos de Linhas” do Tablix e escolha a opção “Propriedades de Grupo”:



**Figura 10.2. Acessando as propriedades de um grupo do Tablix**

Na página de propriedades do grupo, vá até a categoria “Variáveis” e clique em “Adicionar”. Dê o nome de “CorQuantiaNegociada” para a nova variável que está sendo criada e, como demonstrado na **Figura 10.3** a seguir, clique no botão “fx” para escolher a expressão dessa nova variável. Utilize exatamente a mesma expressão apresentada anteriormente na **Listagem 10.1**.



**Figura 10.3. Adicionando uma variável a um grupo do Tablix**

Uma vez criada a variável, podemos utilizá-la para ambas as células tanto da coluna “*Quantia Negociada*” quanto da coluna “*Status*”. Para isso, repita o procedimento em que configuramos a expressão para a propriedade “*Color*” e, ao invés de utilizar a expressão completa, utilize a variável que acabamos de criar:

**Listagem 10.2. Acessando a variável “CorQuantiaNegociada” criada anteriormente**

**1**     `=Variables!CorQuantiaNegociada.Value`

Devemos utilizar variáveis para expressões repetidas não só para fazermos formatação condicional, mas sim, para qualquer outro tipo de expressão. Dessa forma, o relatório fica muito mais limpo e fácil de manter. Já pensou se tivérmos que alterar a cor utilizada na expressão e não estivermos trabalhando com variáveis? Teríamos que lembrar de todos os lugares em que utilizamos aquela expressão e alterarmos cada um desses lugares. Com a utilização de variáveis, bastaria alterar a expressão somente em um lugar. Muito mais fácil, não é mesmo?

## Capítulo 11

# Trabalhando com grupos

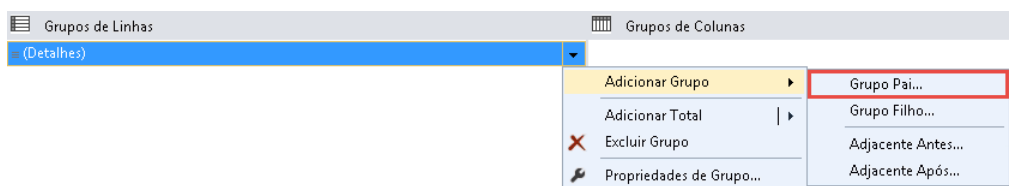
Já vimos um pouco da utilização de grupos no **Capítulo 4 (Utilizando o Assistente de Relatório)**. Porém, como o capítulo era relacionado ao Assistente de Relatório, os grupos foram criados automaticamente. Neste capítulo veremos como criar os mesmos tipos de grupos, só que manualmente.

## Grupos de linhas

O tipo de grupo mais utilizado em relatórios é o agrupamento de linhas. Com esse tipo de grupo, podemos agrupar linhas utilizando o valor de uma coluna específica. Dessa forma, as linhas do relatório são exibidas de forma agrupada por essa coluna.

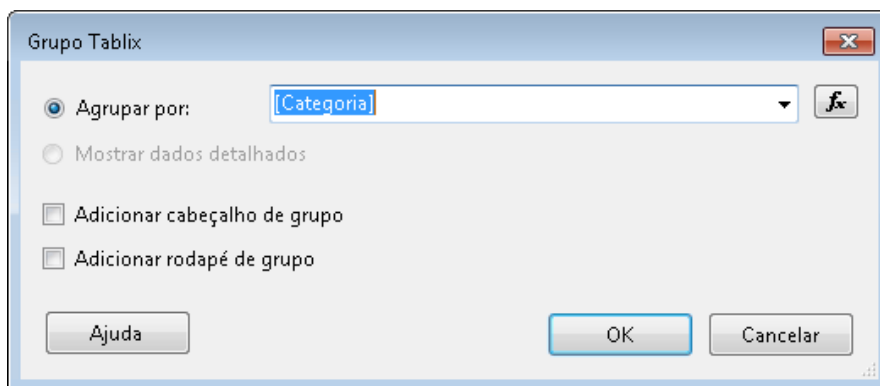
Para entendermos melhor o funcionamento dos grupos, vamos criar um agrupamento por “Categoria” no nosso relatório de Listagem de Fornecedores.

Clique em algum ponto do Tablix, vá até a seção “Grupos de Linhas”, abra a caixa de opções no item “(Detalhes)” e escolha a opção para adicionarmos um novo grupo pai:



**Figura 11.1. Adicionando um grupo de linhas “pai” ao Tablix**

Ao fazermos isso, o mecanismo do Report Viewer nos perguntará qual coluna queremos utilizar para agrupar os dados. No nosso caso, como mencionado anteriormente, agruparemos os dados utilizando a coluna “Categoria”, portanto, escolha a coluna “Categoria” na listagem “Agrupar por”:



**Figura 11.2. Escolhendo a coluna que será utilizada no agrupamento**



Como podemos observar também na **Figura 11.2**, nessa etapa conseguimos criar o cabeçalho e rodapé do grupo, caso seja necessário.

Com o grupo criado, a fim de evitarmos redundância, podemos excluir as colunas destinadas a informação da categoria que tínhamos criado anteriormente na primeira versão do relatório.

Ao executarmos novamente o nosso projeto, teremos agora um simples relatório com a Lista de Fornecedores, porém, agrupado por “Categoria”:

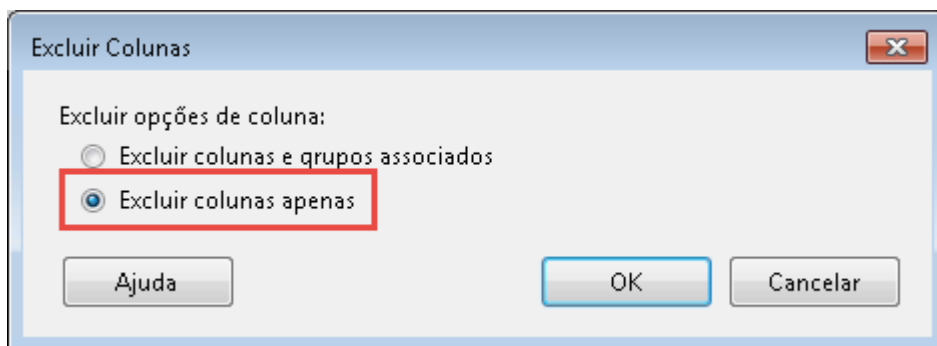
### Lista de Fornecedores

Categoria	ID	Nome	Quantia Negociada	Status
Mão de Obra	-3	Fornecedor 3	\$58.56	Atenção
	-4	Fornecedor 4	\$26.24	Atenção
	-9	Fornecedor 9	\$85.66	Atenção
	-10	Fornecedor 10	\$365.07	OK
	-15	Fornecedor 15	\$92.56	Atenção
Materiais de Escritório	-1	Fornecedor 1	\$125.34	OK
	-2	Fornecedor 2	\$35.24	Atenção
	-7	Fornecedor 7	\$45.99	Atenção
	-8	Fornecedor 8	\$94.48	Atenção
	-13	Fornecedor 13	\$4,672.80	Ótimo
	-14	Fornecedor 14	\$34.65	Atenção
Obra-Prima	-5	Fornecedor 5	\$835.73	OK
	-6	Fornecedor 6	\$2,452.95	Ótimo
	-11	Fornecedor 11	\$2.92	Atenção
	-12	Fornecedor 12	\$84.67	Atenção
			<b>\$9,012.86</b>	

**Figura 11.3. Relatório de Lista de Fornecedores agrupado por “Categoria”**

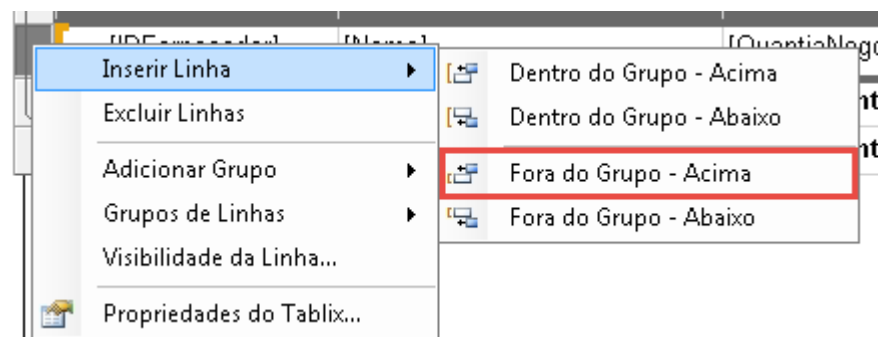
Muitas vezes não queremos que a coluna do agrupamento seja utilizada diretamente na tabela, mas sim, em um cabeçalho do grupo. Apesar de ser possível implementar esse visual, os passos necessários para atingir esse resultado ficaram mais complicados e menos visíveis ao usuário a partir do Visual Studio 2010.

Para fazermos com que a informação da “Categoria” seja exibida em um cabeçalho do grupo, temos que primeiramente excluir a coluna “Categoria” do Tablix, porém, mantendo o grupo. Clique com o botão direito na coluna “Categoria” e escolha a opção “Excluir Colunas”. Na janela “Excluir Colunas”, escolha a opção “Excluir colunas apenas”, conforme demonstrado na **Figura 11.4** a seguir.



**Figura 11.4. Excluindo apenas a coluna, mantendo o grupo criado anteriormente**

Feito isso, podemos adicionar linhas de cabeçalho para o grupo (o que antes não era possível porque coluna “Categoria” estava presente no Tablix). Insira três linhas clicando com o botão direito na linha de detalhes e escolhendo a opção “Inserir Linha Fora do Grupo – Acima”:



**Figura 11.5. Inserindo linhas fora do grupo (acima) para criar um cabeçalho de grupo**

Após termos adicionado as linhas, podemos reorganizar o relatório para que ele fique parecido com a figura abaixo:

	<b>Categoria:</b>	[Categoria]		
	<b>ID</b>	<b>Nome</b>	<b>Quantia Negociada</b>	<b>Status</b>
	[IDFornecedor]	[Nome]	«Expr.»	«Expr.»
			«Expr.»	

**Figura 11.6. Relatório com cabeçalho para o grupo de linhas**

Confira o resultado, que é um layout muito comum em relatórios de negócios com cabeçalhos para os agrupamentos, na **Figura 11.7** a seguir.

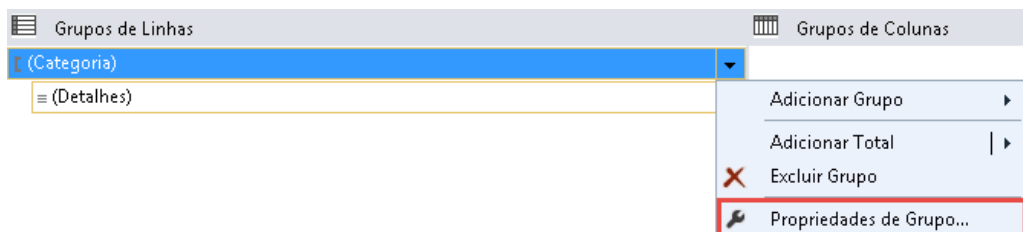
## Lista de Fornecedores

<b>Categoria:</b>	Mão de Obra		
ID	Nome	Quantia Negociada	Status
-3	Fornecedor 3	\$58.56	Atenção
-4	Fornecedor 4	\$26.24	Atenção
-9	Fornecedor 9	\$85.66	Atenção
-10	Fornecedor 10	\$365.07	OK
-15	Fornecedor 15	\$92.56	Atenção
		<b>\$628.09</b>	
<b>Categoria:</b>	Materiais de Escritório		
ID	Nome	Quantia Negociada	Status
-1	Fornecedor 1	\$125.34	OK
-2	Fornecedor 2	\$35.24	Atenção
-7	Fornecedor 7	\$45.99	Atenção

**Figura 11.7. Resultado do relatório com cabeçalho para grupos de linhas**

## Classificação de grupos

Por padrão, os grupos são classificados (ordenados) pelo campo de agrupamento de forma crescente (A-Z). Porém, conseguimos facilmente alterar essa ordem de classificação nas propriedades do grupo. Por exemplo, o relatório apresentado na seção anterior está ordenado pelo campo “Categoria” de forma crescente. Se quisermos alterar para que a classificação seja feita de forma decrescente (ou até mesmo utilizando uma outra lógica de ordenação), temos que primeiramente abrir as propriedades do grupo:



**Figura 11.8. Acessando as propriedades do grupo “Categoria”**

Dentro da página de propriedades do grupo, temos que ir até à categoria “Classificando” e lá conseguimos alterar a coluna utilizada na classificação (ou, se quisermos

utilizar uma expressão customizada, basta clicarmos no botão “fx”) e a ordem de classificação. Podemos até, caso necessário, adicionar múltiplos campos para classificação dos dados.

Todos os conceitos de classificação de dados podem ser aplicados ao ordenarmos os grupos. Para uma abordagem completa sobre esse assunto, confira o **Capítulo 7 (Classificação de dados)**.

## Grupos de colunas

Uma das ferramentas mais poderosas do Report Viewer é a possibilidade de criarmos grupos de colunas. Com essa funcionalidade, conseguimos criar relatórios estilo “*tabela dinâmica*”, o que é nada fácil de realizar em outras ferramentas de relatório.

Pensando no cenário apresentado até o momento, imagine que precisemos desenhar um relatório que sumarie a “*Quantia Negociada*” por “*Categoria*”. Normalmente, um relatório com esse tipo de informação teria mais ou menos este formato:

Categoria	Quantia Negociada
Mão de Obra	628,09
Materiais de Escritório	5008,50
Obra-Prima	3376,27

**Figura 11.9.** Sumarização de “*Quantia Negociada*” por “*Categoria*” através de grupos de linhas

Como podemos perceber, esse é um relatório com um grupo de linhas por “*Categoria*” retornando a soma de “*Quantia Negociada*” como uma coluna. Conseguimos desenhar esse relatório facilmente com o aprendizado da seção anterior juntamente com o que aprendemos no **Capítulo 6 (Criando totalizadores)**.

Porém, e se quisermos desenhar esse relatório de forma que as “*Categorias*” fossem colunas e a soma de “*Quantia Negociada*” fosse apresentada somente em uma linha? O resultado seria algo como esta tabela:

Mão de Obra	Materiais de Escritório	Obra-Prima
628,09	5008,50	3376,27

**Figura 11.10.** Sumarização de “*Quantia Negociada*” por “*Categoria*” através de grupos de colunas

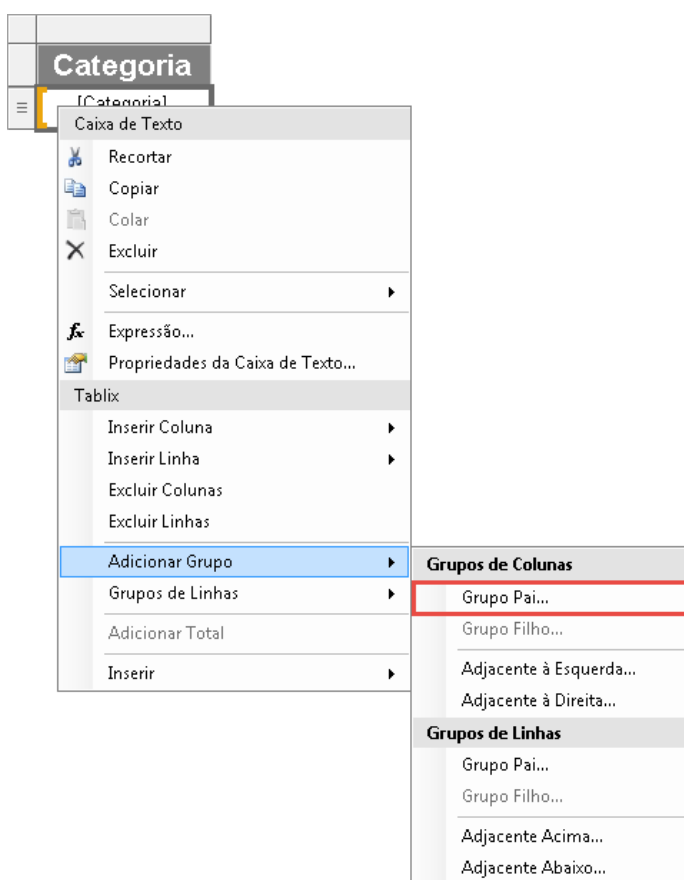
Note que a informação de cada “*Categoria*” é representada em uma coluna e a sua totalização é a apresentada na linha logo abaixo. Como podemos desenhar um relatório com esse layout no Report Viewer? Simples, basta utilizarmos o conceito de grupos de colunas.

Para demonstrar essa funcionalidade, crie uma cópia do relatório atual (agrupado por linhas), exclua o agrupamento por “Categoria” (somente o grupo) e altere o layout do Tablix de forma que ele fique parecido com esta figura:



**Figura 11.11. Layout do relatório somente com a coluna “Categoria”**

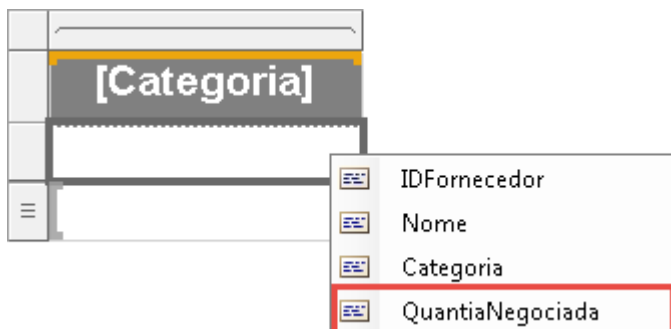
Feito isso, clique com o botão direito na célula “Categoria” na linha de detalhes e escolha a opção “Adicionar Grupo de Colunas Pai”:



**Figura 11.12. Adicionando um grupo de colunas “pai” ao Tablix**

A tela para escolha do campo de agrupamento é idêntica à apresentada anteriormente ao criarmos um grupo de linhas (**Figura 11.2**). Escolha o campo “Categoria” e clique em “OK”.

Com o grupo criado, temos que adicionar a totalização da “*Quantia Negociada*” na “*Categoria*”. Podemos fazer isso através do seletor de campos da célula do Tablix:

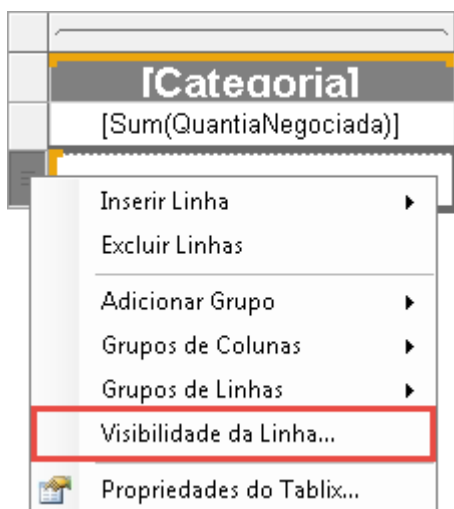


**Figura 11.13. Adicionando a sumarização de “Quantia Negociada” utilizando o seletor de campos**

Ao escolhermos o campo “*Quantia Negociada*”, o Report Viewer automaticamente faz a sumarização do campo, uma vez que estamos em uma célula fora da área de detalhes do Tablix.

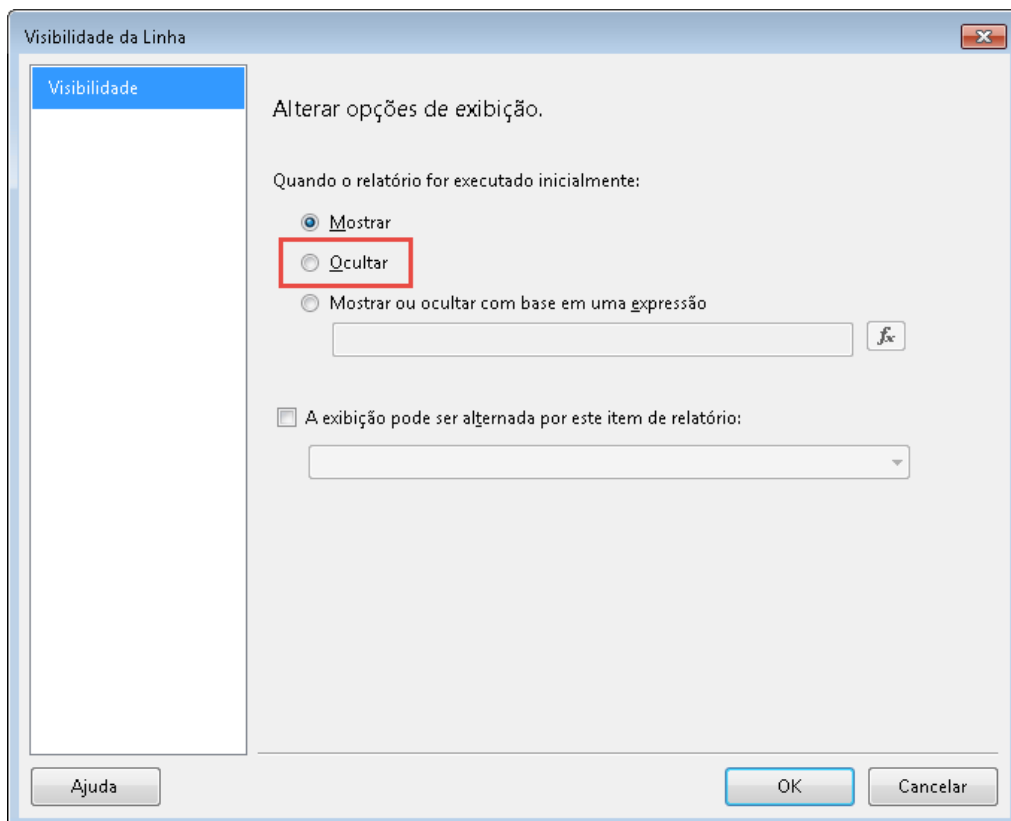
Finalmente, como queremos exibir somente a sumarização de cada “*Categoria*” (e não os detalhes da “*Categoria*”), temos que esconder a linha de detalhes. Caso contrário, o Report Viewer exibirá uma linha de detalhes vazia para cada registro do conjunto de dados, e esse não é o resultado que estamos esperando.

Para esconder a linha de detalhes, clique com o botão direito no seu cabeçalho e escolhermos a opção “*Visibilidade da Linha*”:



**Figura 11.14. Acessando as propriedades de visibilidade da linha de detalhes do Tablix**

Na janela de “*Visibilidade da Linha*”, para escondermos completamente a linha de detalhes, temos que escolher a opção “*Ocultar*”:



**Figura 11.15. Ocultando a linha de detalhes do Tablix**

Com isso, ao exibirmos o relatório, temos o resultado que estávamos esperando: uma sumarização da “*Quantia Negociada*” agrupada por “*Categoria*”, de forma que cada “*Categoria*” é representada em uma coluna. Confira o resultado do relatório:

## Lista de Fornecedores

Mão de Obra	Materiais de Escritório	Obra-Prima
628.09	5008.50	3376.27

**Figura 11.16. Resultado do relatório com agrupamento de colunas por “*Categoria*”**

Como mencionado anteriormente, essa funcionalidade do Report Viewer é muito poderosa, uma vez que o Tablix se ajustará automaticamente independentemente da quantidade de “*Categorias*” presentes no conjunto de dados.

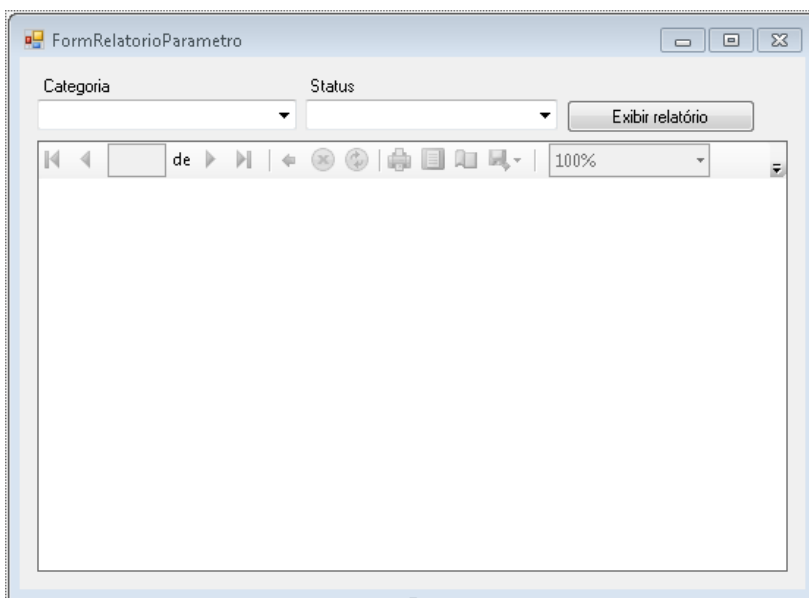
## Capítulo 12

# Adicionando parâmetros

A possibilidade de enviar parâmetros para relatórios do Report Viewer é uma funcionalidade muito utilizada no desenvolvimento de relatórios com essa ferramenta. Ao desenvolvermos nossos sistemas, é muito comum termos que enviar ao relatório algumas informações que o usuário tenha escolhido diretamente no aplicativo.

Um exemplo clássico é a aplicação de filtros através de campos dos nossos formulários. Aprendemos a filtrar os dados dos nossos relatórios no **Capítulo 8 (Filtrando os dados)**, mas, com a funcionalidade de parâmetros do Report Viewer, conseguimos aplicar filtros no relatório de forma dinâmica, onde o usuário seleciona os valores para o filtro no formulário e o relatório é filtrado de acordo com os valores escolhidos.

Para entendermos o funcionamento dos parâmetros no Report Viewer, vamos imaginar a situação em que queremos filtrar o nosso relatório por “Categoria” ou “Status”. Vamos criar dois ComboBoxes no nosso formulário onde o usuário poderá escolher por qual “Categoria” ou “Status” (ou ambos) ele quer filtrar o relatório. Além disso, ao invés de exibirmos o relatório automaticamente quando o formulário é aberto, vamos transferir essa responsabilidade para um botão no nosso formulário. O resultado final deverá ficar parecido com a figura abaixo:



**Figura 12.1.** Formulário com os ComboBoxes “Categoria” e “Status”, que serão enviados para o relatório através da funcionalidade de parâmetros



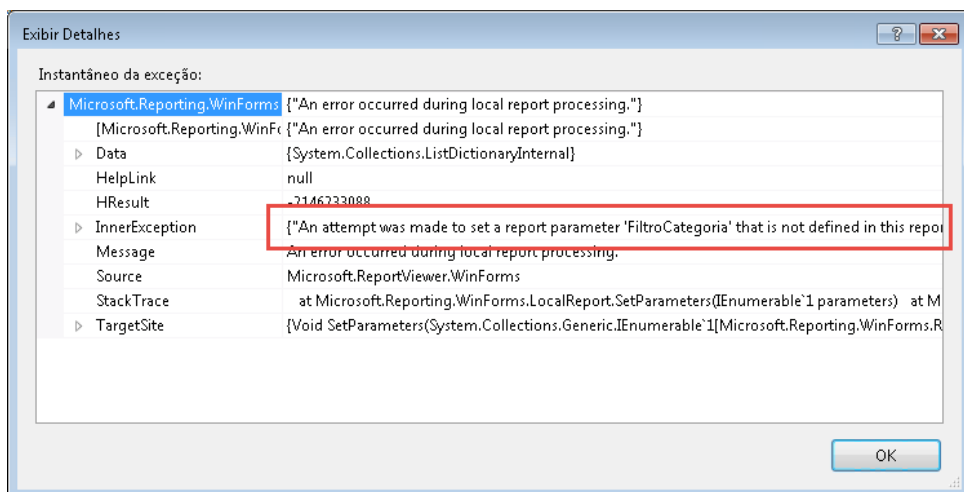
O ComboBox de "Categorias" tem o nome de "categoriaComboBox" com os itens "Materiais de Escritório", "Mão de Obra" e "Obra-Prima". O ComboBox de "Status" tem o nome de "statusComboBox" com os itens "Atenção", "OK" e "Ótimo", que são justamente os mesmos valores que utilizamos dentro do relatório.

No clique do botão "Exibir relatório", vamos criar os parâmetros e configurar seu nome e valor. Além disso, enviamos os parâmetros para o relatório e o atualizamos para conseguirmos ver o resultado. O código completo do evento "Click" do botão "Exibir relatório" está apresentado na **Listagem 12.1** abaixo.

#### Listagem 12.1. Código do evento "Click" do botão "Exibir relatório"

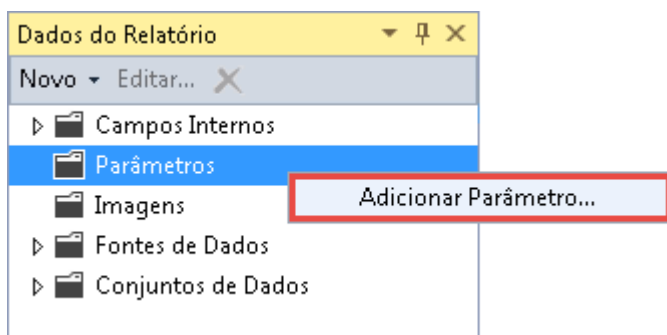
```
1  private void exibirRelatorioButton_Click(object sender,
2      EventArgs e)
3  {
4      var parametroCategoria =
5          new Microsoft.Reporting.WinForms
6              .ReportParameter();
7      var parametroStatus =
8          new Microsoft.Reporting.WinForms
9              .ReportParameter();
10
11      parametroCategoria.Name = "FiltroCategoria";
12      parametroCategoria.Values.Add(categoriaComboBox.Text);
13      parametroStatus.Name = "FiltroStatus";
14      parametroStatus.Values.Add(statusComboBox.Text);
15
16      reportViewer.LocalReport.SetParameters(
17          parametroCategoria);
18      reportViewer.LocalReport.SetParameters(
19          parametroStatus);
20
21      reportViewer.RefreshReport();
22  }
```

Fique alerta ao tentar enviar parâmetros a relatórios do Report Viewer pois, caso o parâmetro não exista no relatório, uma exceção será lançada, conforme demonstrado na **Figura 12.2** a seguir.



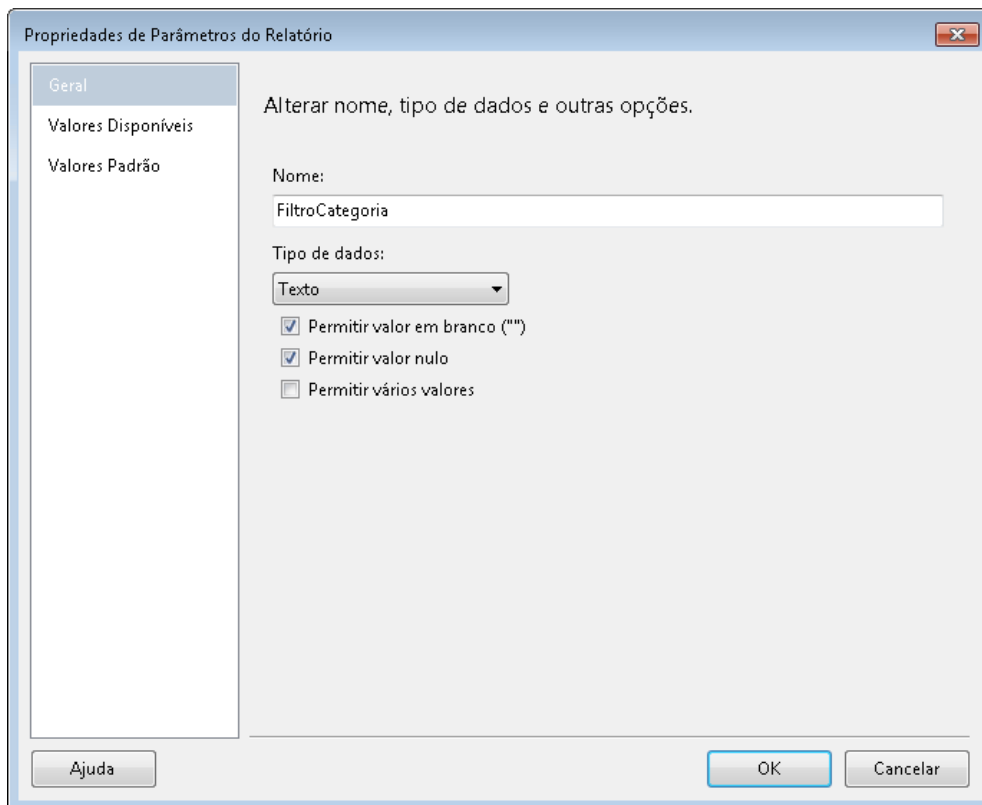
**Figura 12.2.** Exceção que é lançada ao tentarmos enviar um parâmetro que não existe no relatório

Portanto, para evitar que esse erro aconteça no nosso relatório, temos que criar os dois parâmetros. Para adicionar um parâmetro ao relatório, abra a janela “Dados do Relatório”, clique com o botão direito em “Parâmetros” e escolha a opção “Adicionar Parâmetro” no menu de contexto:



**Figura 12.3.** Adicionando um parâmetro ao relatório

Ao clicarmos nesse item, o Report Viewer exibirá a janela “Propriedades de Parâmetro do Relatório”. Nessa janela conseguimos determinar o nome do parâmetro (muito importante, uma vez que temos que utilizar exatamente esse mesmo nome ao enviarmos os parâmetros para o relatório), seu tipo e algumas outras propriedades (informações se o parâmetro é obrigatório e/ou multivalorado). A **Figura 12.4** a seguir mostra a criação do parâmetro “FiltroCategoria”, que iremos utilizar para filtrarmos as informações do relatório pela coluna “Categoria”.



**Figura 12.4. Criando o parâmetro “FiltroCategoria”, que será utilizado para filtrar os dados do relatório pela coluna “Categoria”**

As opções apresentadas nas seções “Valores Disponíveis” e “Valores Padrão” só são consideradas em relatórios vindos do servidor do Reporting Services. Como este livro só aborda relatórios locais, não entraremos em detalhes sobre essas configurações.

Após adicionar o parâmetro “FiltroCategoria”, siga os mesmos passos e adicione outro parâmetro com o nome “FiltroStatus”. Com isso, criamos os dois parâmetros que iremos utilizar para filtrar o relatório pelas colunas “Categoria” e “Status”.

Para filtrarmos o relatório pela coluna “Categoria”, vá até às propriedades do grupo “Categoria” e, na janela de “Propriedades do Grupo”, clique no item “Filtros” e adicione um novo filtro com a expressão detalhada na **Listagem 12.2** abaixo.

#### **Listagem 12.2. Expressão para o filtro de Categorias**

```

1  =IsNothing(Parameters!FiltroCategoria) Or
2  String.IsNullOrEmpty(
3      Parameters!FiltroCategoria.Value) Or
4  Fields!Categoria.Value = Parameters!FiltroCategoria.Value

```

Como você pode perceber, a lógica para a expressão do filtro é muito simples. Os registros do grupo de “*Categoria*” serão exibidos no relatório caso o parâmetro “*FiltroCategoria*” não tenha sido enviado ao relatório ou caso ele tenha sido enviado e o seu valor seja igual ao da “*Categoria*”.

Uma vez que adicionamos o filtro para a “*Categoria*”, temos ainda que filtrar o relatório pela informação de “*Status*”. Esse filtro é um pouco mais trabalhoso do que o outro, pois a coluna “*Status*” na verdade nem existe no nosso conjunto de dados (ela é uma coluna calculada com base em uma expressão, criada no **Capítulo 9 (Utilizando expressões)**).

Para filtrarmos o relatório pela coluna “*Status*”, temos que criar uma variável que calcule o seu valor (Atenção, OK ou Ótimo) baseado no valor da “*Quantia Negociada*”. Essa variável deve ser adicionada dentro do grupo de detalhes do Tablix. Para isso, acesse a página de variáveis dentro das propriedades do grupo de detalhes, adicione uma nova variável com o nome “*Status*” e utilize a formula apresentada abaixo:

#### Listagem 12.3. Expressão para a variável “*Status*”

```
1  =IIf(  
2      Fields!QuantiaNegociada.Value < 100,  
3      "Atenção",  
4      IIf(  
5          Fields!QuantiaNegociada.Value < 1000,  
6          "OK",  
7          "Ótimo"))
```

Para saber maiores detalhes sobre a criação de variáveis, consulte o exemplo demonstrado no **Capítulo 10 (Formatação de dados e formatação condicional)**.

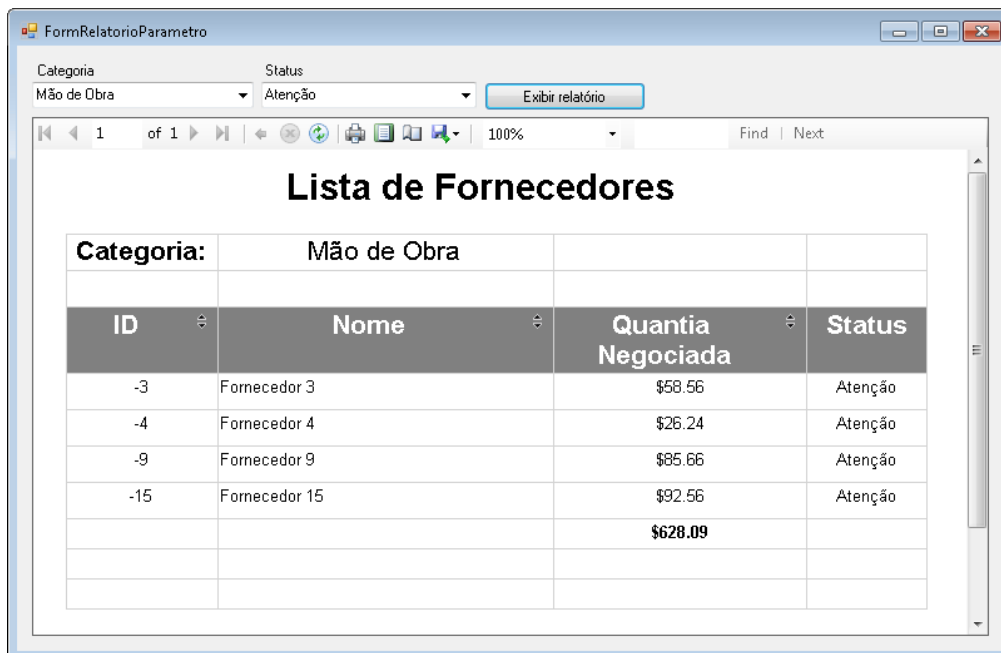
Após termos criado a variável que calculará o valor para a coluna “*Status*”, podemos utilizá-la para filtrarmos o relatório por “*Status*”. Para isso, ainda na janela de propriedades do grupo de detalhes, vá até a página de filtros e adicione um novo filtro contendo a expressão apresentada abaixo:

#### Listagem 12.4. Expressão para filtrarmos o relatório por “*Status*”

```
1  =IsNothing(Parameters!FiltroStatus) Or  
2      String.IsNullOrEmpty(  
3          Parameters!FiltroStatus.Value) Or  
4      Variables!Status.Value = Parameters!FiltroStatus.Value
```

Note que, dentro dessa expressão, estamos utilizando a variável “*Status*” criada no passo anterior.

Nesse ponto, ao tentarmos exibir o relatório filtrado pela categoria “*Mão de Obra*” e status “*Atenção*”, aparentemente o resultado será o esperado, conforme verificamos na **Figura 12.5** a seguir.



**Figura 12.5. Resultado do relatório filtrado por "Categoria" e "Status"**

Porém, você consegue detectar um problema no resultado acima? O somatório da categoria está errado.

## Ajustando a expressão totalizadora no relatório filtrado

Infelizmente, o Report Viewer não considera automaticamente a aplicação dos filtros no momento do cálculo das expressões totalizadoras. Isso faz com que, mesmo que o relatório tenha sido filtrado, os totalizadores serão calculados utilizando todo o universo de linhas do conjunto de dados.

É possível ver o efeito produzido por essa limitação na **Figura 12.5** apresentada anteriormente. Ao invés de termos uma sumarização correta de "Quantia Negociada" (considerando o filtro por "Status" aplicado ao relatório), o Report Viewer simplesmente calculou a sumarização considerando todas as linhas presentes na categoria. Dessa forma, ao invés de termos o resultado correto (\$263.02), temos um resultado que não faz o mínimo sentido com o filtro aplicado (\$628.09).

Para contornarmos essa limitação, temos que considerar o filtro na expressão totalizadora. Até então, a fórmula que utilizamos para calcular o total para "Quantia Negociada" é muito simples:

### Listagem 12.5. Expressão totalizadora simples, sem considerar o filtro por "Status"

```
1 =Format(Sum(Fields!QuantiaNegociada.Value), "c")
```

A expressão apresentada na **Listagem 12.5** representa um totalizador simples, que aprendemos no **Capítulo 6 (Criando totalizadores)**. Para ajustarmos essa expressão de forma que ela considere os filtros, ao invés de simplesmente considerarmos o campo “*Quantia Negociada*”, temos que utilizar a expressão condicional “*IIf*” para decidirmos se consideraremos o valor da “*Quantia Negociada*” ou não, baseando-se no parâmetro “*FiltroStatus*” disponibilizado ao relatório.

O resultado da expressão totalizadora ajustada para considerarmos o filtro por “*Status*” é um tanto quanto complexa. Confira na Listagem abaixo:

**Listagem 12.6. Expressão totalizadora ajustada para considerar o filtro por “*Status*”**

```

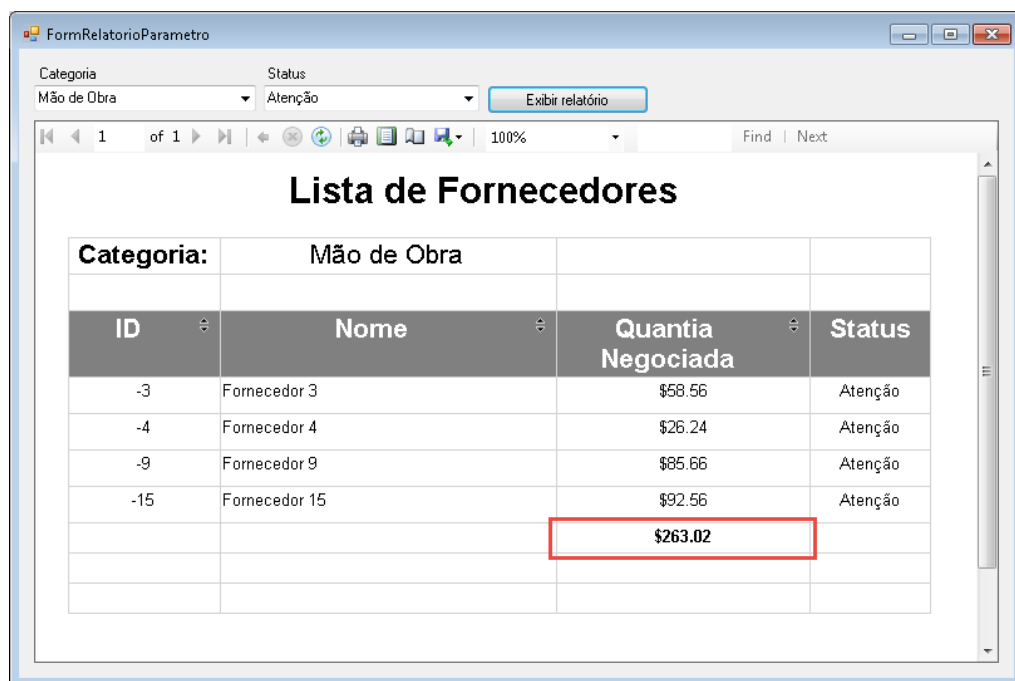
1  =Format (
2    Sum (
3      IIf (
4        IsNothing (Parameters!FiltroStatus) Or
5        String.IsNullOrEmpty (
6          Parameters!FiltroStatus.Value) Or
7        IIf (
8          Fields!QuantiaNegociada.Value < 100,
9          "Atenção",
10         IIf (
11           Fields!QuantiaNegociada.Value < 1000,
12           "OK",
13           "Ótimo")) = Parameters!FiltroStatus.Value,
14         Fields!QuantiaNegociada.Value,
15         CDec (0))) ,
16    "C")

```

Note que, devido a essa limitação, a complexidade da expressão aumentou exponencialmente. Isso porque, além de termos que considerar o filtro na expressão totalizadora, nós não podemos utilizar a variável “*Status*” criada anteriormente, pois o Report Viewer não permite a utilização de variáveis em expressões totalizadoras.

Dessa forma, se quebrarmos a expressão em partes, primeiramente temos a verificação se o parâmetro “*FiltroStatus*” foi disponibilizado ao relatório (linha 4-6). Caso ele tenha sido disponibilizado, comparamos o seu valor com o resultado de outra expressão que traduz o valor de “*Quantia Negociada*” nos valores de “*Status*” (7-13). Se o filtro não foi disponibilizado ou se o filtro condiz com o “*Status*” daquela linha, utilizamos o valor de “*Quantia Negociada*” (linha 14). Caso contrário, utilizamos o valor zero (linha 15), que, como você pode perceber, deve ser convertido para “*Decimal*” (se não fizemos a conversão para “*Decimal*”, o Report Viewer apresentará um erro na expressão totalizadora porque estaremos tentando sumarizar valores de tipos diferentes – “*Decimal*” e “*Integer*” – que é o tipo padrão do dígito zero), o que também não é permitido no Report Viewer.

Ao utilizarmos a expressão com os ajustes apresentados anteriormente, teremos a totalização funcionando perfeitamente mesmo quando o relatório estiver filtrado:



**FormRelatorioParametro**

Categoria: Mão de Obra Status: Atenção Exibir relatório

1 of 1 100% Find | Next

### Lista de Fornecedores

<b>Categoria:</b>		<b>Mão de Obra</b>	
<b>ID</b>	<b>Nome</b>	<b>Quantia Negociada</b>	<b>Status</b>
-3	Fornecedor 3	\$58.56	Atenção
-4	Fornecedor 4	\$26.24	Atenção
-9	Fornecedor 9	\$85.66	Atenção
-15	Fornecedor 15	\$92.56	Atenção
		<b>\$263.02</b>	

**Figura 12.6. Resultado do relatório filtrado com o somatório correto**

Como você pode perceber, o Report Viewer é uma ferramenta muito interessante e totalmente gratuita, mas, quando batemos em uma de suas limitações, temos que fazer certos malabarismos para obtermos o resultado esperado.

## Capítulo 13

# Mestre-detache, SubReports e Drill-down

Uma das principais dúvidas que surgem quando desenvolvemos relatórios em qualquer ferramenta é: como conseguimos exibir informações no estilo mestre-detache?

Na grande maioria das vezes, os dados que alimentam os nossos relatórios são originados de várias tabelas, normalmente seguindo uma hierarquia de mestre-detache, muitas vezes até mesmo com múltiplos níveis.

Caso consigamos mesclar os todos dados envolvidos no relatório em uma só DataTable, é possível construirmos uma hierarquia mestre-detache através da criação de agrupamentos. No **Capítulo 11 (Trabalhando com grupos)**, aprendemos a agrupar os dados de "Fornecedores" por "Categoria". Essa composição, apesar de simples, representa a estrutura em formato mestre-detache (onde a "Categoria" é o mestre e os "Fornecedores" são os detalhes).

Enquanto a estrutura mestre-detache for simples (como é o caso do agrupamento das "Categorias" mostrado anteriormente) e todos os dados estiverem presentes em uma única DataTable, conseguimos implementá-la através de agrupamentos. Porém, uma vez que a hierarquia mestre-detache atinge uma certa complexidade, ou uma vez que os dados da hierarquia se encontrem em tabelas diferentes, só conseguimos exibir essa hierarquia através de SubReports.

## SubReports

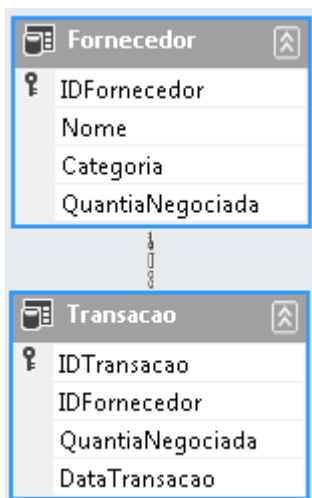
O conceito de SubReports é, como o próprio nome já diz, a possibilidade de colocarmos um relatório dentro de outro. Os SubReports são muito utilizados para a exibição de dados no formato mestre-detache.

Para exemplificarmos a utilização de SubReports, vamos estender o exemplo da Lista de Fornecedores criado anteriormente de forma que tenhamos mais um nível, onde listaremos as transações efetuadas com cada fornecedor. Como cada fornecedor pode ter nenhuma, uma ou várias transações, esse seria um típico exemplo de hierarquia mestre-detache (onde o fornecedor é o pai e as transações são os filhos).

O primeiro passo nesse processo é abrirmos o DataSetExemplo e criarmos a tabela "Transacao". Essa tabela deve ter uma chave primária auto-incremento ("IDTransacao"), a chave estrangeira para fazer a ligação com a tabela de fornecedores ("IDFor-



*necedor*”) e outras duas colunas que dão detalhes sobre a transação (“*QuantiaNegociada*” e “*DataTransacao*”). Não esqueça de adicionar o relacionamento entre as tabelas “*Fornecedor*” e “*Transacao*”. Veja na figura abaixo como deve ficar o resultado final do DataSetExemplo após adicionarmos essa nova tabela e o relacionamento:



**Figura 13.1. DataSetExemplo após adicionarmos a tabela Transacao e o relacionamento**

Uma vez adicionada a tabela “*Transacao*” ao DataSetExemplo, podemos construir o relatório que será exibido como SubReport. Esse relatório será muito simples e contará somente com um controle Tablix listando as colunas “*DataTransacao*” e “*QuantiaNegociada*” da tabela “*Transacao*”, conforme podemos conferir na figura abaixo:

Data	Transação
[DataTransacao]	[QuantiaNegociada]

**Figura 13.2. Layout do SubReport que fará a listagem das transações por fornecedor**

Para evitar repetições, não vou detalhar o passo-a-passo da criação desse relatório. Caso você tenha dúvidas em como chegar a esse resultado, confira o **Capítulo 2 (Criando o seu primeiro relatório)**. A única diferença é que, ao invés de utilizar a tabela “*Fornecedor*” como fonte de dados para o relatório, temos que utilizar a tabela “*Transacao*”.

Para fazermos a comunicação entre o relatório principal e os SubReports temos que utilizar a passagem de parâmetros. Com o Report Viewer, o relatório mestre consegue passar parâmetros ao SubReport. No nosso exemplo, o SubReport precisa saber de qual fornecedor ele deve listar as transações. Dessa forma, após termos construído o

layout do SubReport, precisamos adicionar um novo parâmetro nesse SubReport que receberá o "IDFornecedor" para fazer a filtragem dos dados.

Dito isso, adicione um novo parâmetro no SubReport, chamado "FiltroIDFornecedor". Para maiores detalhes sobre parâmetros no Report Viewer, confira o **Capítulo 12 (Adicionando Parâmetros)**. Com o novo parâmetro no SubReport, utilize-o para filtrar os dados do Tablix. Confira o **Capítulo 8 (Filtrando os dados)** caso surja alguma dúvida nesse processo.

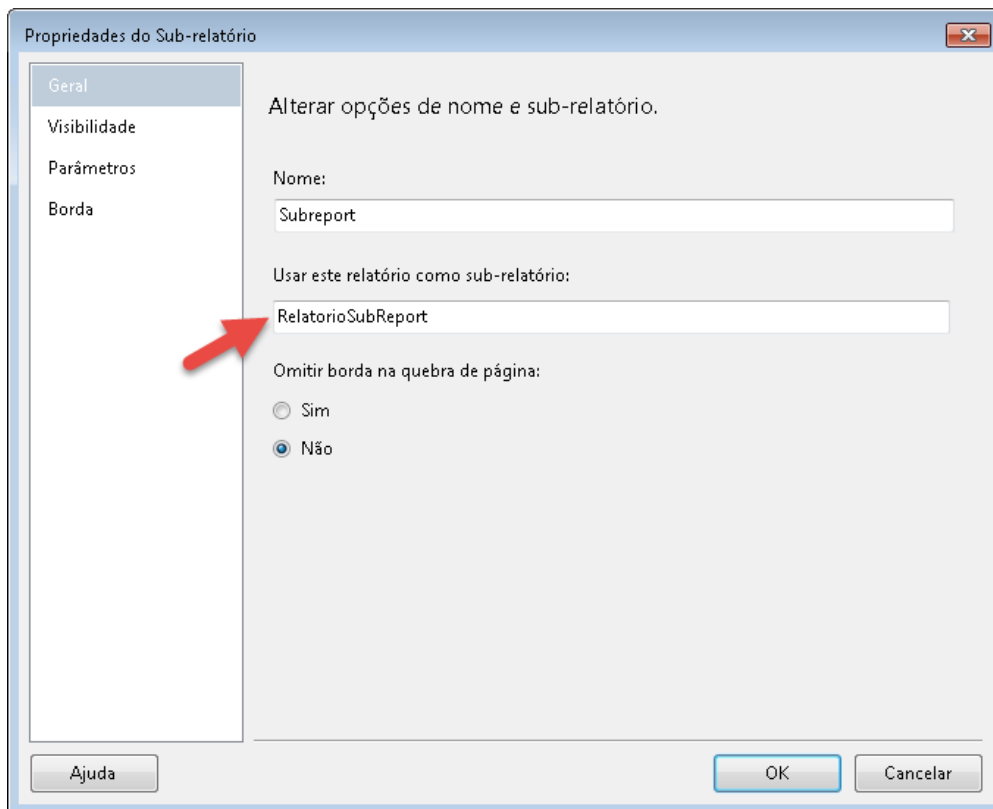
Uma vez configurado o filtro no SubReport utilizando o parâmetro "FiltroIDFornecedor", estamos prontos para seguir com o ajuste no relatório principal. Vá até o relatório principal que criamos anteriormente e adicione uma linha no nível de detalhe (dentro do grupo). Nessa nova linha de detalhe, clique com o botão direito na primeira célula e escolha a opção "Inserir Sub-relatório". Feito isso, formate o Tablix para que ele fique parecido com a figura abaixo:

Categoria:		[Categoria]		
ID	Nome	Quantia Negociada	Status	
[IDFornecedor]	[Nome]	«Expr.»	«Expr.»	
RelatorioSubReport				
		«Expr.»		

**Figura 13.3. Tablix com o SubReport adicionado**

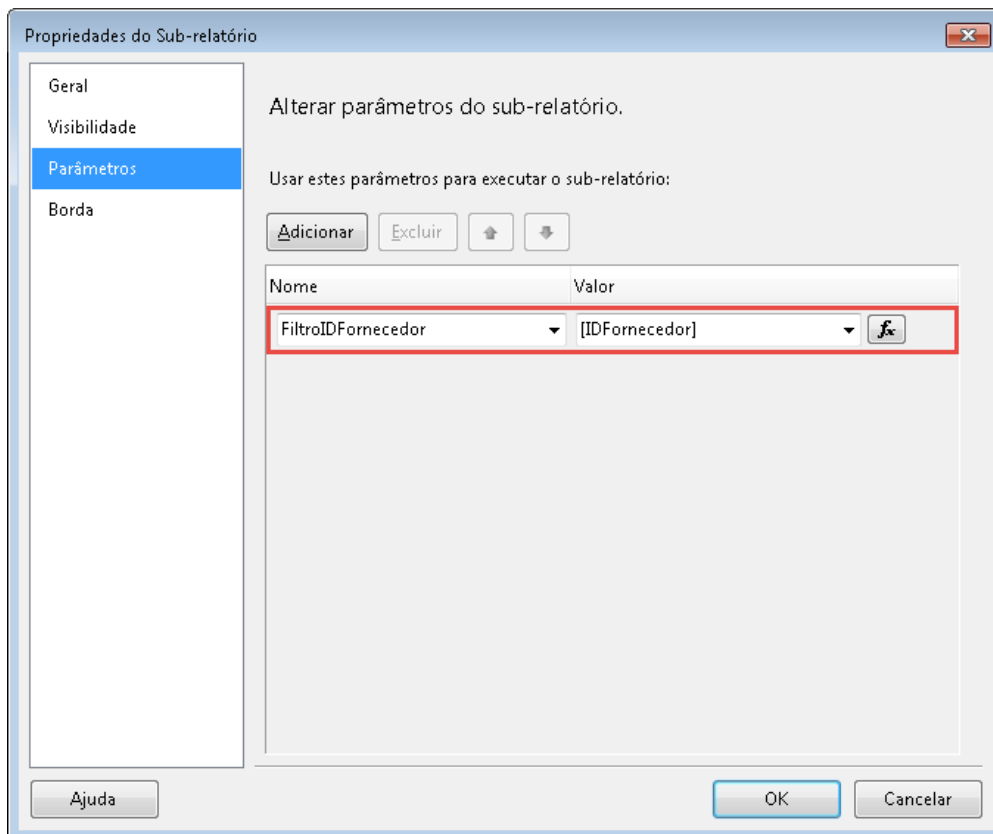
Duas propriedades muito importantes que devem ser configuradas no SubReport são o nome do SubReport e o parâmetro que fará a ligação entre os dois relatórios. Para acessar as propriedades do SubReport, clique com o botão direito sobre ele e escolha a opção "Propriedades do Sub-relatório".

Na página de propriedades do SubReport, temos que colocar exatamente o nome que utilizamos para o SubReport dentro da caixa de texto "Usar esse relatório como sub-relatório" (eliminando a extensão ".rdlc"). Por exemplo, se o nome do SubReport for "RelatorioSubReport.rdlc", temos que preencher essa caixa de texto com "RelatorioSubReport", como mostra a **Figura 13.4** a seguir.



**Figura 13.4.** Indicando o nome do SubReport nas propriedades do controle

Feito isso, temos que alternar para a página de parâmetros para adicionarmos o parâmetro de ligação entre os dois relatórios. O nome do parâmetro deve ser "*FiltroIDFornecedor*" (mesmo nome do parâmetro que criamos anteriormente dentro do SubReport) e o valor deve ser o campo "*IDFornecedor*", conforme podemos conferir na **Figura 13.5**.



**Figura 13.5. Parâmetro de ligação entre o relatório mestre e o SubReport**

Com esses ajustes realizados, terminamos a parte de layout dos relatórios. Agora temos que ajustar o código do formulário de forma que criemos informações na tabela de transações. Para isso, ajuste o código do evento Load do formulário e acrescente, para cada novo fornecedor, algumas transações, conforme podemos conferir na listagem abaixo:

**Listagem 13.1. Adicionando informações de transação para cada novo fornecedor**

```

1  var fornecedor =
2      DataSetExemplo.Fornecedor.AddFornecedorRow(
3          "Fornecedor 1",
4          "Materiais de Escritório",
5          (decimal)125.34);
6  DataSetExemplo.Transacao.AddTransacaoRow(
7      fornecedor,
8      100,
9      new DateTime(2015, 1, 1));
10 DataSetExemplo.Transacao.AddTransacaoRow(
11     fornecedor,
12     (decimal)25.34,
13     new DateTime(2015, 2, 1));

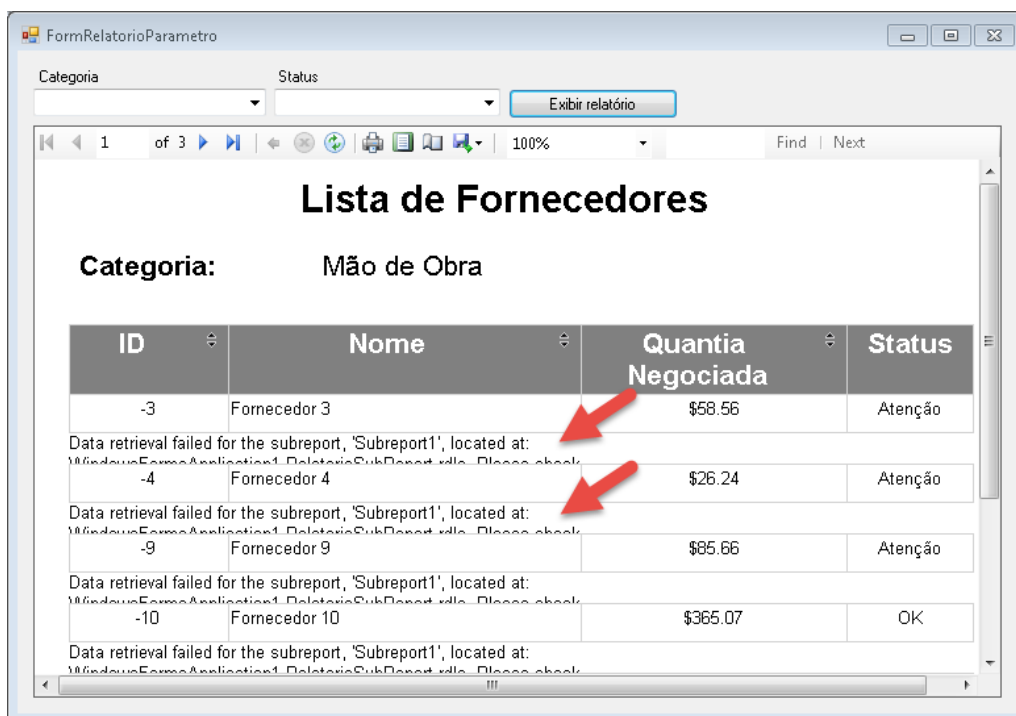
```

```

14 fornecedor =
15     DataSetExemplo.Fornecedor.AddFornecedorRow(
16         "Fornecedor 2",
17         "Materiais de Escritório",
18         (decimal)35.24);
19 DataSetExemplo.Transacao.AddTransacaoRow(
20     fornecedor,
21     30,
22     new DateTime(2015, 1, 1));
23 DataSetExemplo.Transacao.AddTransacaoRow(
24     fornecedor,
25     (decimal)5.24,
26     new DateTime(2015, 2, 1));
27 // Continua. Confira no projeto de exemplo
28 // a versão completa deste método.

```

Aparentemente temos tudo o que precisamos para exibirmos os dados do SubReport. Porém, se tentarmos executar o exemplo nesse momento, receberemos um erro em cada tentativa que o Report Viewer fizer para exibir o SubReport. Isso se deve ao fato que o Report Viewer não passa automaticamente o conjunto de dados de um relatório para o outro e, caso não indiquemos manualmente a fonte de dados para o SubReport, o Report Viewer não conseguirá exibi-lo, como podemos observar na figura abaixo:



**Figura 13.6. Erro ao carregar os dados do SubReport**

Para indicarmos a fonte de dados para o SubReport, temos que utilizar o evento "*SubReportProcessing*". Nesse evento, temos acesso à instância de "*SubreportProcessingEventArgs*", que possibilita informarmos a fonte de dados para o SubReport. Para utilizarmos esse evento, temos que adicionar o hook para o evento logo antes da chamada de "*RefreshReport*". Dessa forma, vá até a implementação do evento "*Load*" do formulário e adicione a linha para fazer esse hook:

### Listagem 13.2. Adicionando o "hook" para o evento SubReportProcessing

```
1 reportViewer.LocalReport.SubreportProcessing +=  
2     LocalReport_SubreportProcessing;
```

Além disso, temos que adicionar a implementação do evento, que será somente uma linha de código muito simples, onde adicionaremos uma nova fonte de dados ao SubReport, passando a tabela "*Transacao*" que preenchemos anteriormente:

### Listagem 13.3. Implementação do evento SubReportProcessing

```
1 void LocalReport_SubreportProcessing(  
2     object sender,  
3     Microsoft.Reporting.  
4     WinForms.SubreportProcessingEventArgs e)  
5 {  
6     e.DataSources.Add(  
7         new Microsoft.Reporting.  
8         WinForms.ReportDataSource(  
9             "DataSetExemplo",  
10            (DataTable)DataSetExemplo.Transacao));  
11 }
```

Com essa pequena alteração, ao tentarmos exibir novamente o relatório, teremos o resultado que esperamos: fornecedores (mestre) sendo listados com suas respectivas transações (detalhe), conforme podemos visualizar na **Figura 13.7** a seguir.

FormRelatorioParametro

Categoria:  Status:

1 of 3 100% Find | Next

### Lista de Fornecedores

**Categoria:** Mão de Obra

ID	Nome	Quantia Negociada	Status								
-3	Fornecedor 3	\$58.56	Atenção								
<table border="1"> <thead> <tr> <th>Data</th> <th>Transação</th> </tr> </thead> <tbody> <tr> <td>01/01/2015</td> <td>\$40.00</td> </tr> <tr> <td>01/02/2015</td> <td>\$10.00</td> </tr> <tr> <td>01/03/2015</td> <td>\$8.56</td> </tr> </tbody> </table>				Data	Transação	01/01/2015	\$40.00	01/02/2015	\$10.00	01/03/2015	\$8.56
Data	Transação										
01/01/2015	\$40.00										
01/02/2015	\$10.00										
01/03/2015	\$8.56										
-4	Fornecedor 4	\$26.24	Atenção								
<table border="1"> <thead> <tr> <th>Data</th> <th>Transação</th> </tr> </thead> <tbody> <tr> <td>01/02/2015</td> <td>\$26.24</td> </tr> </tbody> </table>				Data	Transação	01/02/2015	\$26.24				
Data	Transação										
01/02/2015	\$26.24										
-9	Fornecedor 9	\$85.66	Atenção								
<table border="1"> <thead> <tr> <th>Data</th> <th>Transação</th> </tr> </thead> <tbody> <tr> <td>01/01/2015</td> <td>\$40.00</td> </tr> <tr> <td>01/02/2015</td> <td>\$45.66</td> </tr> </tbody> </table>				Data	Transação	01/01/2015	\$40.00	01/02/2015	\$45.66		
Data	Transação										
01/01/2015	\$40.00										
01/02/2015	\$45.66										

**Figura 13.7. Resultado da implementação de mestre-detalle utilizando SubReports**

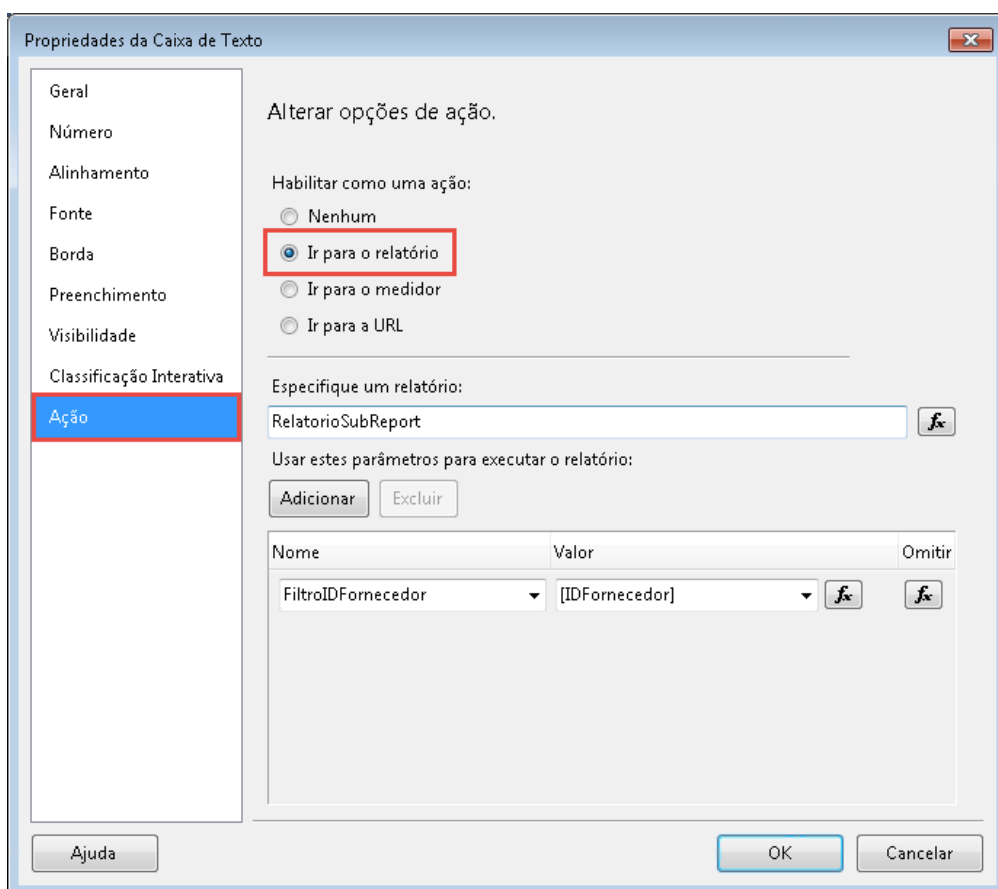
Como mencionado anteriormente, a utilização de SubReports para a implementação de mestre-detalle nos relatórios com o Report Viewer é muito utilizada. Essa é a sistemática que temos que utilizar para exibirmos dados caso a hierarquia esteja distribuída em múltiplas tabelas. Porém, vale a pena mencionar que os SubReports do Report Viewer não são muito performáticos. Dessa forma, tente evitar utilizá-los caso o mesmo resultado possa ser implementado através da funcionalidade de agrupamentos (que é mais otimizada).

## Drill-down

A funcionalidade de drill-down é uma alternativa aos SubReports. Algumas vezes, ao exibirmos a estrutura mestre-detalle nos nossos relatórios, eles acabam ficando muito poluídos. E se quisermos exibir somente a parte "mestre" da hierarquia no relatório principal e, ao clicarmos no "mestre", um outro relatório com os detalhes fosse aberto? É justamente com o drill-down que podemos implementar essa lógica nos nossos relatórios.

Para demonstrar o drill-down no Report Viewer, vamos alterar o nosso relatório de exemplo e fazer com que, ao clicarmos no ID do fornecedor, o relatório que lista as suas transações seja aberto separadamente. Para isso, clique com o botão direito na célula que exibe o ID do fornecedor e escolha a opção “*Propriedades da Caixa de Texto*”.

Na janela que se abre, clique na categoria “*Ação*”, escolha a opção “*Ir para o relatório*” e preencha as informações do relatório a ser aberto (nome do relatório sem a extensão “*rdlc*” e o parâmetro de ligação entre o relatório mestre e o relatório detalhe, como fizemos anteriormente para o SubReport). Confira o resultado dessa configuração na figura abaixo:



**Figura 13.8. Configurando o drill-down para abrir o relatório detalhe ao clicar no ID do fornecedor**

Após confirmarmos essa alteração e executarmos o relatório, o Report Viewer nos exibirá uma mensagem de erro dizendo que não foi possível carregar a fonte de dados para o relatório detalhe. Isso se deve ao fato que, da mesma forma que acontece com os SubReports, o Report Viewer não passa automaticamente a fonte de dados do re-



latório mestre para o relatório detalhe. Para indicarmos a fonte de dados para o relatório detalhe, temos que utilizar o evento Drillthrough. Primeiramente, adicione o hook para o evento logo antes da chamada de *"RefreshReport"*:

#### Listagem 13.4. Adicionando o "hook" para o evento Drillthrough

```
1 reportViewer.LocalReport.SubreportProcessing +=
2     LocalReport_SubreportProcessing;
3 reportViewer.Drillthrough +=
4     ReportViewer_Drillthrough;
5 reportViewer.RefreshReport();
```

Em seguida, adicione a implementação do evento Drillthrough, que é muito parecida com a implementação que fizemos anteriormente para o evento SubreportProcessing:

#### Listagem 13.5. Implementação do evento Drillthrough

```
1 void ReportViewer_Drillthrough(
2     object sender,
3     Microsoft.Reporting.
4     WinForms.DrillthroughEventArgs e)
5 {
6     var relatorio = ((Microsoft.Reporting.
7         WinForms.LocalReport)e.Report);
8     relatorio.DataSources.Add(
9         new Microsoft.Reporting.
10         WinForms.ReportDataSource(
11             "DataSetExemplo",
12             (DataTable)DataSetExemplo.Transacao));
13 }
```

Feito isso, ao executarmos o relatório e clicarmos no ID de algum fornecedor, o relatório detalhe será aberto com a lista de transações daquele fornecedor, e com isso concluímos a implementação do drill-down.

## Capítulo 14

# Exibindo imagens

**T**odo relatório que desenhamos muito provavelmente contará com pelo menos uma imagem. Por exemplo, é muito comum termos que adicionar o logotipo da empresa no cabeçalho do relatório ou termos que exibir imagens dos detalhes do relatório.

O Report Viewer possibilita a exibição de imagens em três modalidades diferentes: incorporada, externa ou carregada do banco de dados. Imagens incorporadas são, como o próprio nome diz, incorporadas diretamente no arquivo do relatório (rldc). As imagens externas são carregadas de um arquivo em disco e as imagens do banco de dados são as armazenadas em campos binários em tabelas no banco de dados.

Nas próximas seções vamos aprender a utilizar cada um desse tipo de imagem, bem como os cenários em que elas são geralmente utilizadas.

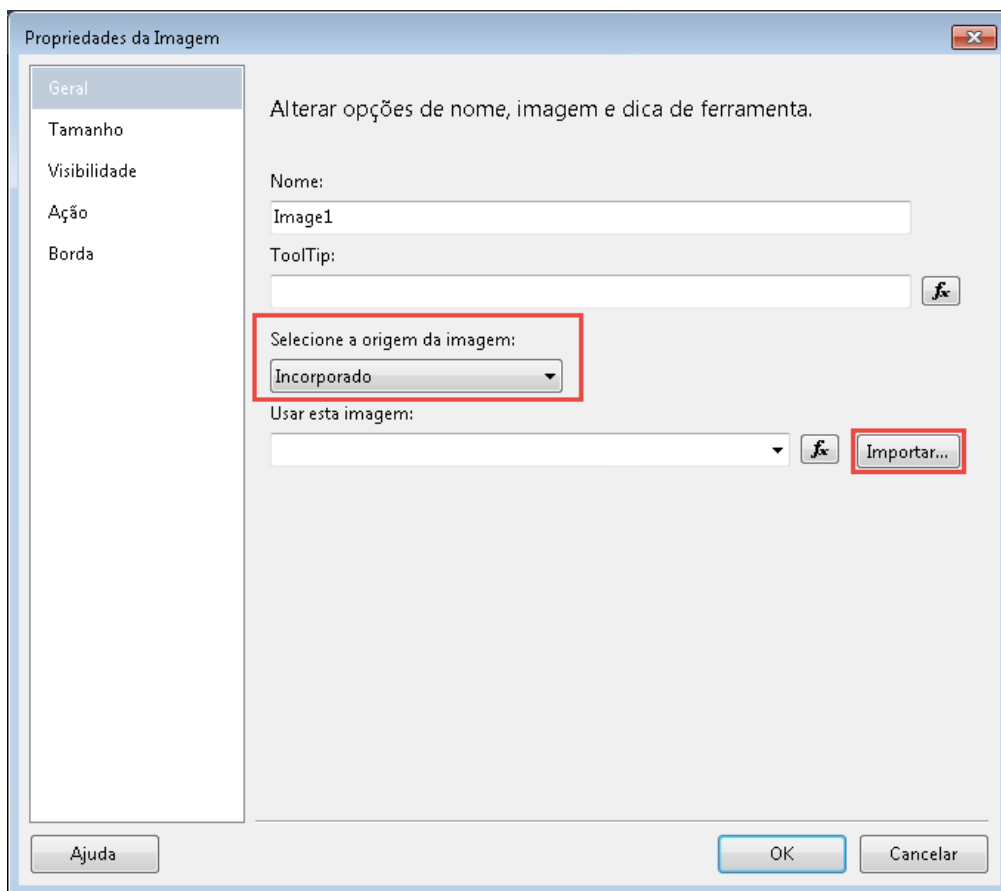
## Imagens incorporadas

As imagens incorporadas, como mencionado anteriormente, são anexadas juntamente com o arquivo do relatório (rdlc). Uma vez que escolhemos a imagem e a anexamos no relatório, não temos mais que nos preocupar em carregá-la, pois ela sempre estará corretamente integrada ao relatório.

Esse tipo de imagem é muito utilizado para exibirmos o logotipo da empresa (ou sistema) no cabeçalho do relatório. Se o mesmo relatório for por somente uma empresa (ou seja, não há a necessidade que o logotipo seja dinâmico), essa é uma boa alternativa.

Adicionar imagens incorporadas nos relatórios do Report Viewer é uma tarefa muito simples. Basta arrastarmos um controle do tipo *"Imagem"* onde desejamos exibir a imagem e o Report Viewer automaticamente nos exibirá a janela de *"Propriedades da Imagem"*.

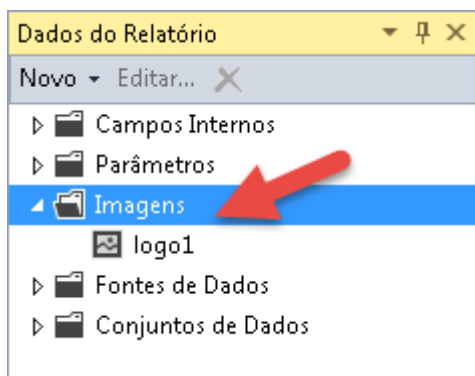
Nessa janela, escolha a opção *"Incorporado"* e clique no botão *"Importar"* para escolher a imagem que deverá ser utilizada, conforme demonstrado na **Figura 14.1** a seguir.



**Figura 14.1. Configurando uma Imagem incorporada**

É importante termos em mente que as imagens incorporadas ao relatório estarão realmente fazendo parte do arquivo do relatório. Dessa forma, quanto mais imagens forem incorporadas ao relatório, maior ficará o tamanho do arquivo rdlc.

Se precisarmos utilizar a mesma imagem incorporada em diferentes pontos do relatório, não é necessário importá-las múltiplas vezes. As imagens incorporadas podem ser reutilizadas em outros locais do relatório. Uma lista de todas as imagens que foram importadas anteriormente estará sempre disponível na janela “Dados do Relatório”, como podemos conferir na **Figura 14.2**. Nessa mesma janela conseguimos excluir potenciais imagens que não estejam mais sendo utilizadas no relatório.



**Figura 14.2.** Lista de imagens que foram incorporadas ao relatório anteriormente

## Imagens externas

As imagens externas são aquelas carregadas através de um caminho em disco. Ao contrário das imagens incorporadas, essas imagens não são adicionadas no arquivo do relatório, ou seja, elas não fazem com que o arquivo do relatório fique maior. Porém, temos que sempre ficar atentos para que o caminho da imagem seja um caminho válido, do contrário, por razões óbvias, a imagem não será carregada pelo Report Viewer.

Esse tipo de imagem só é recomendado se tivermos a certeza que as imagens desejadas estarão presentes em um determinado caminho em disco. Por exemplo, se você obrigatoriamente faz com que o sistema seja instalado em um caminho específico, sem dar a opção de escolha para o usuário no momento da instalação (como, por exemplo, é o caso do aplicativo de declaração de imposto de renda da Receita Federal do Brasil, entre outros), é seguro utilizar essa forma de carregamento de imagens.

Para demonstrar o carregamento de imagens externas no Report Viewer, vamos considerar que tenhamos alguns logotipos armazenados no caminho "C:\Logos" (logo.png, logo1.png, logo2.png e logo3.png). Se você quiser alterar o carregamento do logotipo para que ele utilize o carregamento externo ao invés de imagem incorporada, vá até as propriedades da imagem, altere o tipo de carregamento para "Externo", clique no botão "fx" e utilize a seguinte expressão:

### Listagem 14.1. Expressão para fazer o carregamento de uma imagem externa

```
1 = "File://c:\Logos\logo.png"
```

Note que a sintaxe para o carregamento de imagens externas em disco deve sempre começar com "File://", seguido do caminho da imagem.

Com esse tipo de carregamento também é possível carregarmos imagens remotas, ou seja, que estejam acessíveis através de uma URL. Por exemplo, para carregarmos o logotipo do Programa Microsoft MVP armazenado no meu site, podemos utilizar a expressão apresentada na **Listagem 14.2** a seguir.

**Listagem 14.2. Carregando uma imagem através de uma URL**

```

1  = "http://www.andrealveslima.com.br/
2  files/imagens/mvphorizontalmini.png"

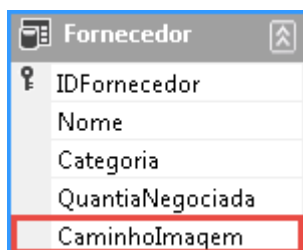
```

NOTA: PARA QUE RELATÓRIOS COM IMAGENS EXTERNAS SEJAM CARREGADOS COM SUCESSO, VOCÊ PRECISA CONFIGURAR A PROPRIEDADE `LOCALREPORT.ENABLEEXTERNALIMAGES` COMO `TRUE` NO SEU CONTROLE DE EXIBIÇÃO DO REPORT VIEWER. CASO ESSA PROPRIEDADE ESTEJA CONFIGURADA COMO `FALSE`, UMA EXCEÇÃO SERÁ LANÇADA E O RELATÓRIO NÃO SERÁ EXIBIDO. CONFIGURE ESSA PROPRIEDADE NA JANELA DE PROPRIEDADES OU ANTES DA CHAMADA DO MÉTODO `REFRESHREPORT`.

Por fim, outra maneira de fazer o carregamento de imagens externas é armazenando o caminho das imagens no banco de dados. Dessa forma, conseguimos utilizar uma expressão para fazer o carregamento das imagens para cada detalhe do relatório dinamicamente.

Para demonstrar esse cenário, vamos estender o exemplo da lista de fornecedores de forma que, para cada fornecedor, tenhamos uma coluna com o seu logotipo, que será carregado através do seu caminho em disco.

O primeiro passo que temos que seguir para implementar essa mudança é criarmos uma nova coluna do tipo texto na tabela `Fornecedor`, chamada `"CaminhoImagem"`:



**Figura 14.3. Nova coluna "CaminhoImagem" na tabela "Fornecedor"**

Feito isso, altere o carregamento do `DataSetExemplo` no formulário, caso contrário a aplicação deixará de compilar com sucesso. Vamos utilizar aleatoriamente os logotipos do diretório `"C:\Logos"` para cada um dos fornecedores:

**Listagem 14.3. Indicando o caminho do logotipo para cada um dos fornecedores**

```

1  var fornecedor =
2  DataSetExemplo.Fornecedor.AddFornecedorRow(
3  "Fornecedor 1",
4  "Materiais de Escritório",
5  (decimal)125.34,
6  @"c:\logos\logo1.png");
7  DataSetExemplo.Transacao.AddTransacaoRow(
8  fornecedor,
9  100,
10  new DateTime(2015, 1, 1));
11 DataSetExemplo.Transacao.AddTransacaoRow(

```

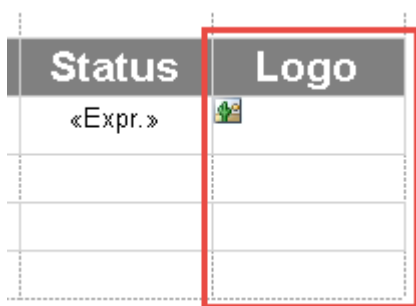
```

12     fornecedor,
13     (decimal)25.34,
14     new DateTime(2015, 2, 1));
15 fornecedor =
16     DataSetExemplo.Fornecedor.AddFornecedorRow(
17     "Fornecedor 2",
18     "Materiais de Escritório",
19     (decimal)35.24,
20     @"c:\logos\logo2.png");
21 DataSetExemplo.Transacao.AddTransacaoRow(
22     fornecedor,
23     30,
24     new DateTime(2015, 1, 1));
25 DataSetExemplo.Transacao.AddTransacaoRow(
26     fornecedor,
27     (decimal)5.24,
28     new DateTime(2015, 2, 1));
29 // Continua. Confira no projeto de exemplo
30 // a versão completa deste método.

```

Em seguida, temos que atualizar os campos do DataSet do relatório de forma que ele reconheça essa nova coluna. Para isso, clique com o botão direito no DataSet do relatório e escolha a opção “Atualizar”.

Para exibir o logotipo do fornecedor, crie uma nova coluna no Tablix com o título “Logo” e um controle do tipo Imagem na célula de detalhe:



**Figura 14.4. Coluna “Logo” criada no Tablix para exibirmos o logotipo do fornecedor**

O carregamento dos logotipos deverá ser feito através do caminho especificado na tabela Fornecedor, portanto, nas propriedades do controle Imagem selecione o tipo de carregamento “Externo” e utilize seguinte expressão para o caminho da imagem:

**Listagem 14.4. Expressão utilizada para carregar o logotipo através do caminho especificado na tabela Fornecedor**

```

1     ="File://" & Fields!CaminhoImagem.Value

```

Note que a expressão é similar à que utilizamos anteriormente para carregarmos a imagem a partir de um caminho em disco fixo, porém, dessa vez utilizamos o caminho armazenado na coluna “CaminhoImagem”.

Confira na **Figura 14.5** o resultado do relatório após essas alterações, que nada mais é do que esperamos: a listagem de fornecedores com a adição de uma nova coluna exibindo um logotipo fictício para cada fornecedor.

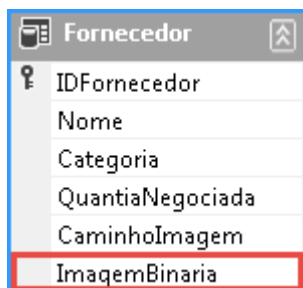


**Figura 14.5.** Resultado do relatório com a nova coluna “Logo”, exibindo o logotipo para cada fornecedor

## Imagens do Banco de Dados

A última modalidade de carregamento de imagens que temos à nossa disposição no Report Viewer é o carregamento de imagens da fonte de dados. Nesse tipo de carregamento, a imagem deve estar armazenada em uma coluna do tipo binário no conjunto de dados.

Para demonstrar esse tipo de carregamento, crie uma nova coluna do tipo “byte[]” na tabela Fornecedor. Dê o nome de “ImagemBinaria” a essa coluna:



**Figura 14.6. Nova coluna “ImagemBinaria” na tabela “Fornecedor”**

Feito isso, altere o código de carregamento do DataSet passando a representação binária dos logotipos para cada um dos fornecedores, conforme demonstrado na **Lis-tagem 14.5** a seguir.



**Listagem 14.5. Preenchimento da coluna “ImagemBinaria” no momento de criação dos fornecedores**

```
1  byte[] logo1, logo2, logo3;
2  using (System.IO.FileStream fileStream =
3      new System.IO.FileStream(
4          @"c:\logos\logo1.png", System.IO.FileMode.Open))
5      logo1 = StreamParaByteArray(fileStream);
6  using (System.IO.FileStream fileStream =
7      new System.IO.FileStream(
8          @"c:\logos\logo2.png", System.IO.FileMode.Open))
9      logo2 = StreamParaByteArray(fileStream);
10 using (System.IO.FileStream fileStream =
11     new System.IO.FileStream(
12         @"c:\logos\logo3.png", System.IO.FileMode.Open))
13     logo3 = StreamParaByteArray(fileStream);
14
15 var fornecedor =
16     DataSetExemplo.Fornecedor.AddFornecedorRow(
17         "Fornecedor 1",
18         "Materiais de Escritório",
19         (decimal)125.34,
20         @"c:\logos\logo1.png",
21         logo1);
22 DataSetExemplo.Transacao.AddTransacaoRow(
23     fornecedor,
24     100,
25     new DateTime(2015, 1, 1));
26 DataSetExemplo.Transacao.AddTransacaoRow(
27     fornecedor,
28     (decimal)25.34,
29     new DateTime(2015, 2, 1));
30 fornecedor =
31     DataSetExemplo.Fornecedor.AddFornecedorRow(
32         "Fornecedor 2",
33         "Materiais de Escritório",
34         (decimal)35.24,
35         @"c:\logos\logo2.png",
36         logo2);
37 DataSetExemplo.Transacao.AddTransacaoRow(
38     fornecedor,
39     30,
40     new DateTime(2015, 1, 1));
41 DataSetExemplo.Transacao.AddTransacaoRow(
42     fornecedor,
43     (decimal)5.24,
44     new DateTime(2015, 2, 1));
45 // Continua. Confira no projeto de exemplo
46 // a versão completa deste método.
```

Note que primeiramente criamos os arrays de bytes para cada um dos logotipos fictícios e utilizamos esses arrays de bytes no momento da criação dos fornecedores.

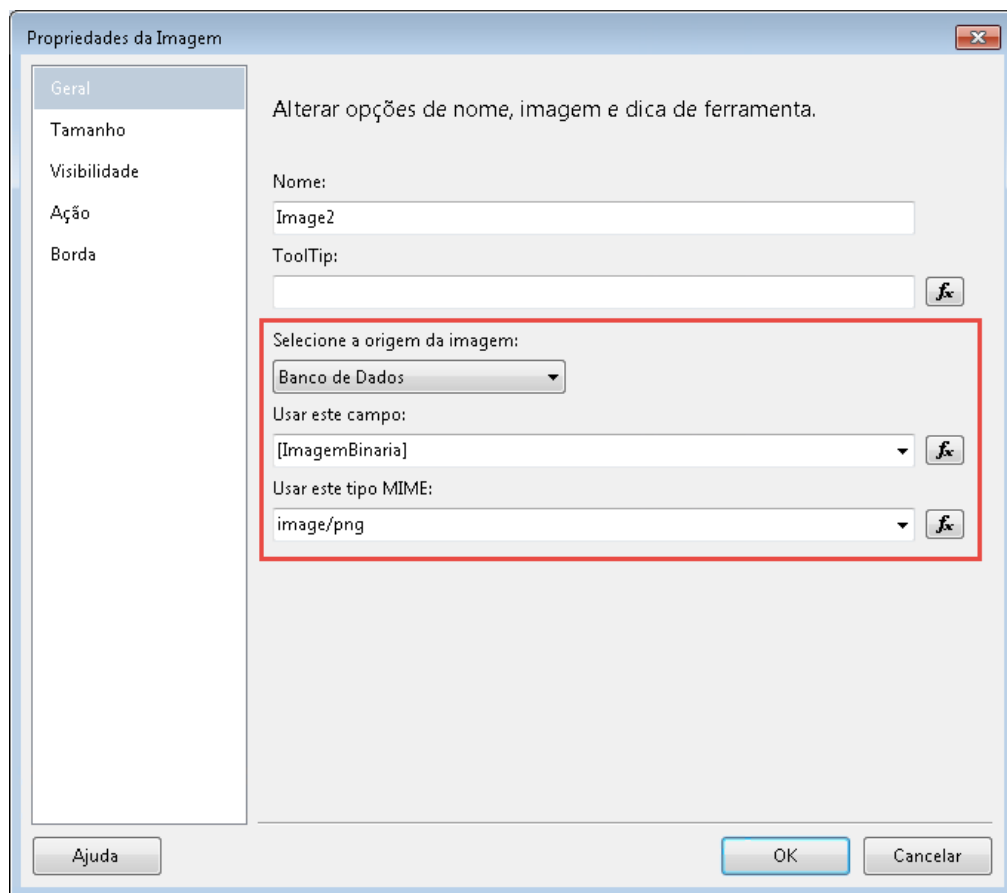
Para convertermos uma Stream em array de bytes, utilizamos o método apresentado abaixo na **Listagem 14.6**, que pode ser facilmente encontrado em fóruns, sites de discussões ou até mesmo na própria documentação da Microsoft.

**Listagem 14.6. Método utilizado para converter uma Stream para array de bytes**

```
1 public byte[] StreamParaByteArray(  
2     System.IO.Stream stream)  
3 {  
4     if (stream == null)  
5         return null;  
6  
7     stream.Seek(0, System.IO.SeekOrigin.Begin);  
8     byte[] toReturn = new byte[stream.Length];  
9     stream.Read(  
10         toReturn,  
11         0,  
12         System.Convert.ToInt32(stream.Length));  
13  
14     return toReturn;  
15 }
```

Uma vez que criamos e preenchemos a nova coluna com a representação binária da imagem, podemos utilizá-lo no nosso relatório. Para isso, atualize novamente os campos do DataSet do relatório clicando com o botão direito e escolhendo a opção “Atualizar”.

Com o novo campo reconhecido pelo relatório, altere o tipo de carregamento no controle imagem para “Banco de Dados” e escolha esse novo campo como fonte de dados para a imagem do logotipo de cada fornecedor. É importante notar que temos que escolher corretamente o tipo “MIME” da imagem, ou seja, a sua extensão. Caso contrário, a imagem não será corretamente carregada pelo Report Viewer. No nosso exemplo, a imagem está salva no formato “png”, portanto, temos que escolher o tipo “MIME” “image/png”, conforme conferimos na **Figura 14.7** a seguir.



**Figura 14.7. Propriedades da Imagem configurada para o carregamento através do Banco de Dados**

Ao executarmos o relatório novamente, vemos que as imagens são carregadas e exibidas exatamente como antes. Porém, nesse momento, ao invés das imagens estarem sendo carregadas de caminhos em disco, elas estão sendo carregadas diretamente da coluna "*ImagemBinaria*" da tabela *Fornecedor*.

## Capítulo 15

# Criando gráficos

Uma das principais maneiras de deixarmos os nossos relatórios mais ricos é utilizando gráficos. Normalmente, quando exibimos dados numéricos, principalmente dados estatísticos, a representação em forma de gráfico é a mais indicada para conseguirmos analisar as informações de uma maneira mais fácil.

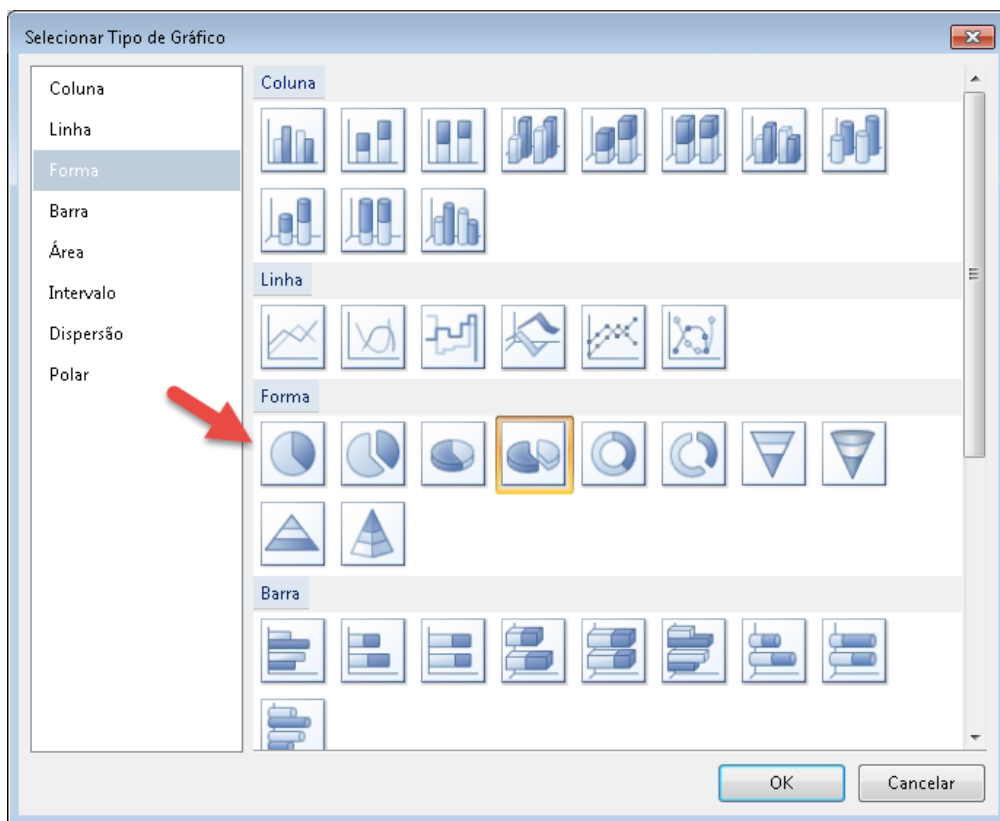
Os gráficos do Report Viewer, apesar de serem um pouco complexos (por terem inúmeras propriedades escondidas que só acabamos aprendendo a sua localização depois de utilizar o componente por muito tempo), atendem muito bem às principais necessidades dos relatórios de hoje em dia.

Para demonstrar a criação de gráficos no Report Viewer, vamos estender o relatório que desenhamos até agora, onde adicionaremos dois gráficos: um gráfico de pizza com a quantia negociada por categoria e um gráfico de coluna empilhada com a quantia negociada por categoria e status.

No relatório de listagem de fornecedores, abra um espaço logo abaixo do Tablix, que será o local onde colocaremos os nossos gráficos. Para adicionar um novo gráfico, clique com o botão direito em uma área vazia do relatório e escolha a opção *"Inserir Gráfico"*.

Ao escolher essa opção, o Report Viewer perguntará o tipo de gráfico. Note que o Report Viewer possibilita a criação dos mais diversos tipos de gráficos, desde os mais simples (como gráficos de coluna, barra e forma) até os mais complexos (como gráficos castiçal – muito utilizado para representar ações no mercado de valores –, gráficos de dispersão e polares). Como o processo de criação dos gráficos é independente do tipo de gráfico escolhido, neste livro aprenderemos somente a confecção de gráficos mais simples. De qualquer forma, os conceitos podem ser facilmente estendidos caso surja a necessidade da criação de um gráfico mais complexo.

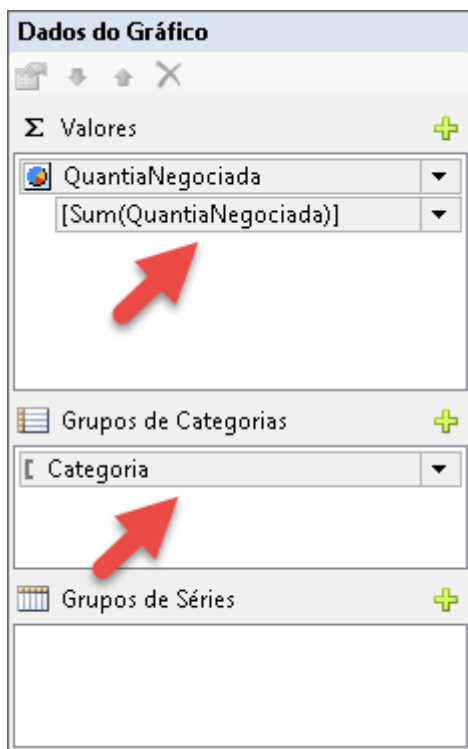
Como mencionado anteriormente, o primeiro gráfico que criaremos no nosso relatório será um gráfico de pizza com a quantia negociada por categoria. Dessa forma, na tela *"Selecionar Tipo de Gráfico"*, escolha um dos quatro tipos de gráfico de pizza disponíveis no Report Viewer como, por exemplo, o gráfico de pizza destacada 3D, que está selecionado na **Figura 15.1** a seguir.



**Figura 15.1. Escolhendo o tipo de gráfico – nesse exemplo, o tipo “Pizza Destacada 3D”**

A primeira configuração que temos que alterar é a fonte de dados do gráfico. Para isso, clique com o botão direito em uma área vazia do gráfico e escolha a opção “*Propriedades do Gráfico*”. Na janela de propriedades do gráfico, escolha o “*DataSetExemplo*” como conjunto de dados. Caso algum filtro tenha sido definido para outros controles do relatório, não podemos esquecer de definir os mesmos filtros para o gráfico, caso contrário ele considerará o conjunto de dados em sua totalidade. Relembre como fazer a filtragem de dados no **Capítulo 8 (Filtrando os dados)**.

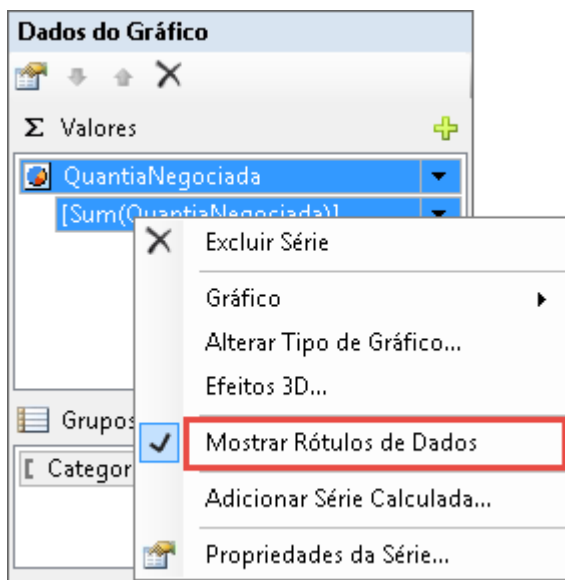
Com o gráfico adicionado ao relatório e a sua fonte de dados configurada, basta selecionarmos os valores que desejamos para as categorias, séries e valores. Como mencionado anteriormente, esse primeiro gráfico exibirá informações da quantia negociada por categoria. Portanto, a configuração nesse caso é muito simples: a coluna “*Categoria*” será utilizada nos “*Grupos de Categorias*” e a somatória de “*QuantiaNegociada*” será utilizada na parte de valores. Podemos conferir o resultado dessa configuração na **Figura 15.2** a seguir.



**Figura 15.2. Configuração do gráfico de “Quantia Negociada” por “Categoria”**

Para alterar o título do gráfico, clique sobre o ele e altere a propriedade “Caption” na janela de propriedades. Podemos também mover alguns controles internos do gráfico (como, por exemplo, a legenda) a fim de que o conteúdo do gráfico seja exibido de uma maneira mais organizada.

Em gráficos de pizza, como é o nosso caso, normalmente precisamos exibir os rótulos das categorias, caso contrário, fica difícil entender exatamente a porcentagem de cada uma das categorias (que é justamente o que queremos mostrar com um gráfico de pizza). Para fazermos isso, ative os rótulos dos valores clicando com o botão direito no item de somatório da quantia negociada e escolhendo a opção “Mostrar Rótulos de Dados”, como demonstrado na **Figura 15.3** a seguir.

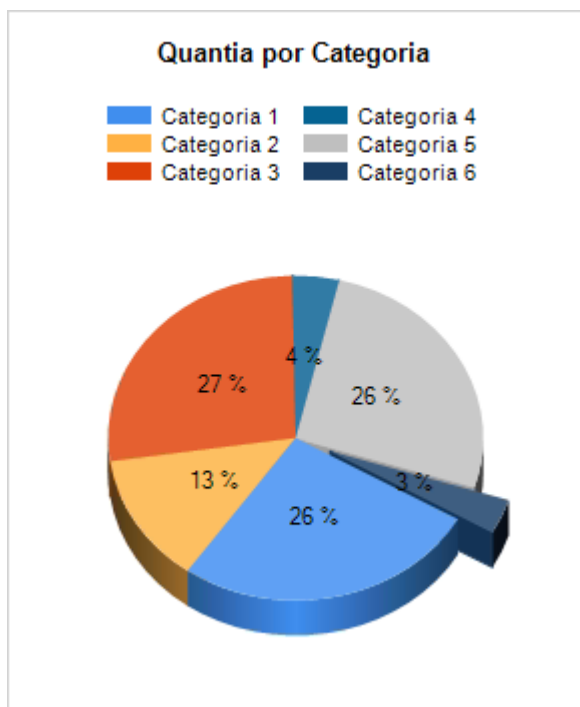


**Figura 15.3. Mostrando os rótulos de dados nos gráficos do Report Viewer**

Porém, ao ativarmos os rótulos de dados, a quantia absoluta será exibida, o que não é a melhor opção em gráficos de pizza. Felizmente, é possível exibirmos as porcentagens das categorias em gráficos de pizza no Report Viewer. Mas, infelizmente, isso foi implementado de forma tão obscura e escondida que são necessárias muitas pesquisas na Internet para encontrar a solução.

Para exibir as porcentagens nas totalizações do Report Viewer (ao invés do valor absoluto, que é o padrão), clique com o botão direito em um dos rótulos e escolha a opção "Propriedades do Rótulo de Série". Na janela que se abre, note que podemos utilizar uma expressão para os dados do rótulo. É justamente utilizando a expressão "#PERCENT" que conseguimos exibir as porcentagens nos rótulos da série. Essa expressão pode ser acompanhada de um formatador, caso você queira que a porcentagem seja exibida com uma quantidade específica de casas decimais. Por exemplo, para exibirmos a porcentagem sem casas decimais, utilizamos a expressão "#PERCENT{P0}". Para exibirmos a porcentagem com duas casas decimais, utilizamos a expressão "#PERCENT{P2}".

Confira o resultado final do gráfico após termos aplicado essa formatação na **Figura 15.4** a seguir.



**Figura 15.4. Gráfico de Quantia Negociada por Categoria**

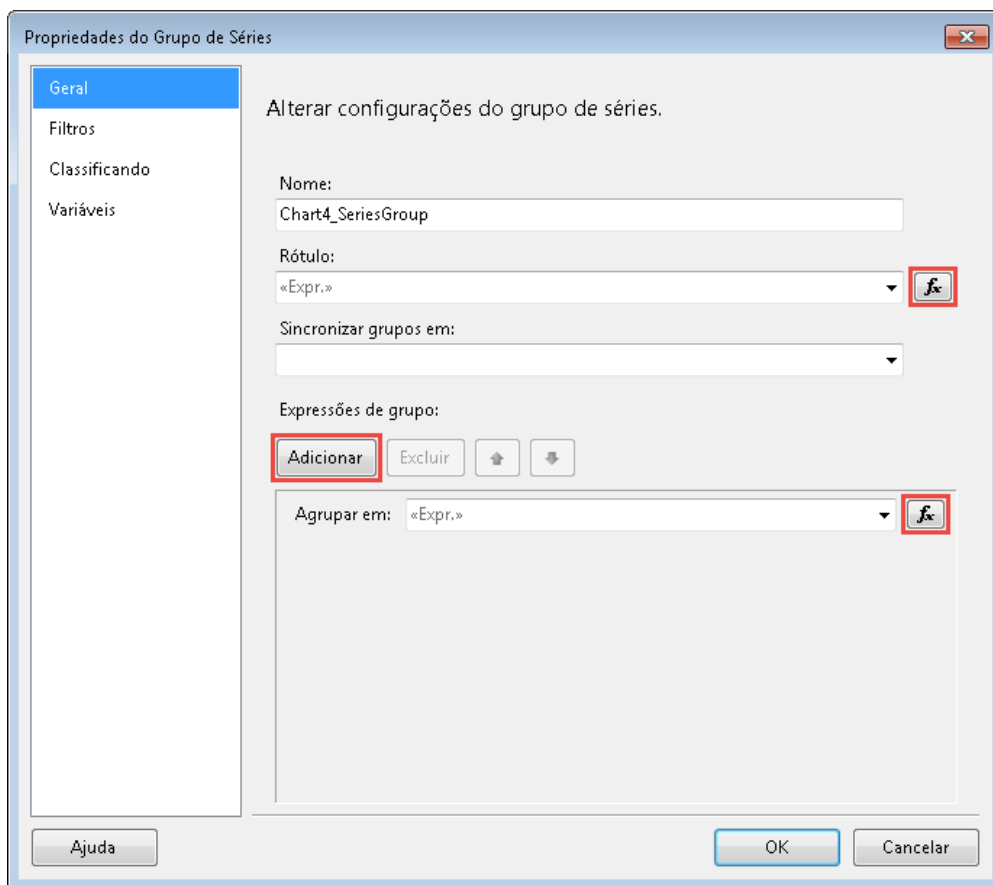
O segundo gráfico que vamos criar é um gráfico de coluna empilhada 3D. Com esse tipo de gráfico conseguimos exibir três séries de informações ao mesmo tempo (uma informação no eixo horizontal, uma informação no eixo vertical e outra informação nas colunas). Utilizaremos esse segundo gráfico para exibirmos a Quantia Negociada por Categoria e Status.

Para conseguirmos essa disposição, configure a seção de valores como o somatório do campo "*QuantiaNegociada*", o grupo de categorias pela coluna "*Categoria*" e o grupo de séries como uma expressão que retornará o "*Status*" dependendo do valor da quantia negociada.

Devido a uma falha no designer do controle do Report Viewer, só é possível acessarmos a tela de definição da expressão da série após termos selecionado uma coluna do DataSet. Dessa forma, primeiramente escolha qualquer uma das colunas no grupo de séries e depois clique novamente na lista e escolha a opção "*Propriedades do Grupo de Séries*".

Na janela "*Propriedades do Grupo de Séries*", temos que configurar uma expressão tanto para o rótulo da série como para o agrupamento da série. Para isso, clique em "*Adicionar*" e configure a expressão nesses dois campos através do botão "*fx*", conforme demonstrado na **Figura 15.5**.





**Figura 15.5. Configurando uma expressão no grupo de séries do gráfico**

A expressão que devemos utilizar nesses dois itens é a mesma que utilizamos anteriormente no **Capítulo 9 (Utilizando expressões)**, que simplesmente fará a conversão do valor da coluna "QuantiaNegociada" em "Atenção", "OK" ou "Ótimo", dependendo do seu valor:

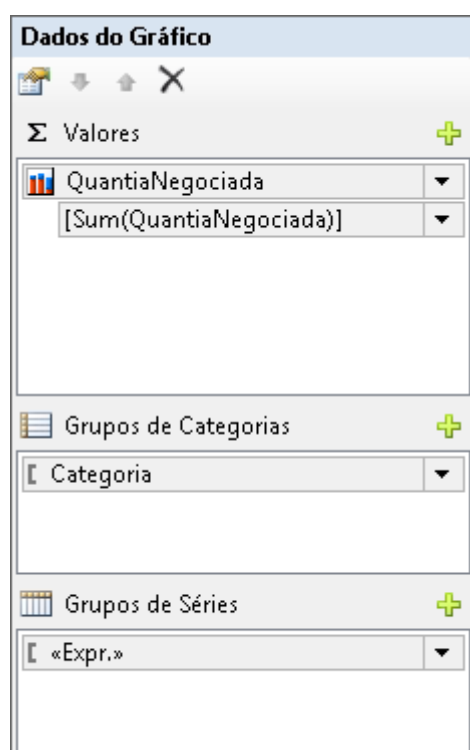
#### Listagem 15.1. Expressão para o grupo de séries do gráfico

```

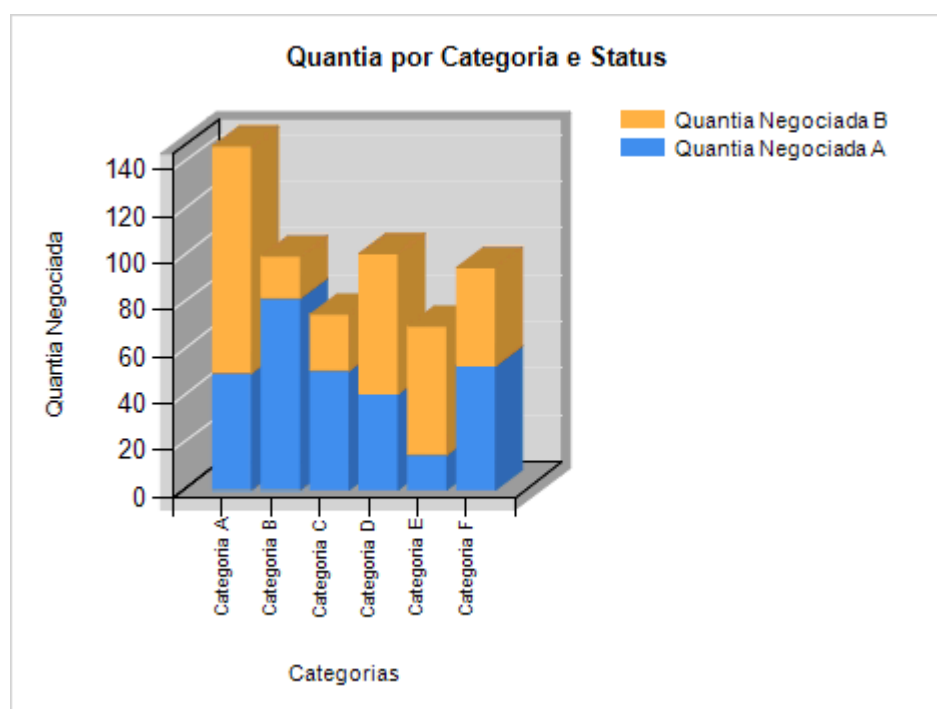
1  =IIf (Fields!QuantiaNegociada.Value < 100,
2      "Atenção",
3      IIf (Fields!QuantiaNegociada.Value < 1000,
4          "OK",
5          "Ótimo"))

```

Após seguir todos esses passos, as configurações de "Dados do Gráfico" e o designer do gráfico devem ficar parecidos com a **Figura 15.6** e **Figura 15.7** a seguir.

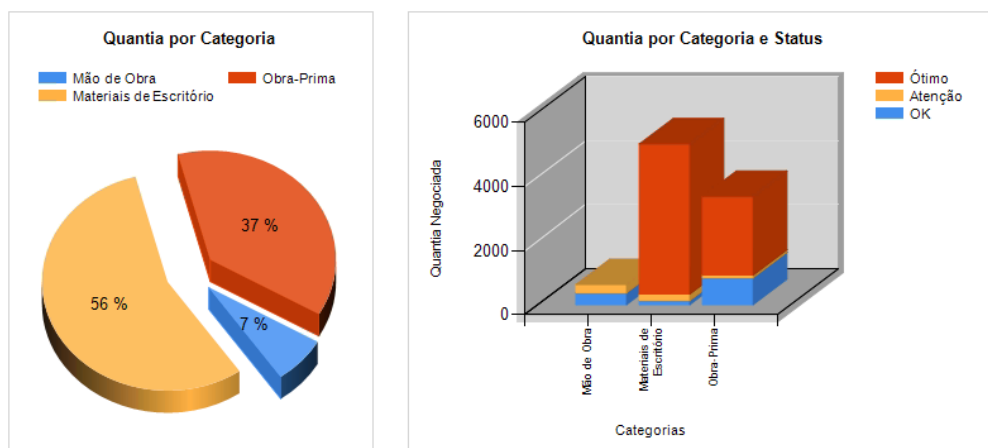


**Figura 15.6. Configuração dos dados do gráfico de colunas**



**Figura 15.7. Gráfico de Quantia Negociada por Categoria e Status**

Execute novamente a aplicação e verifique o resultado dos gráficos:



**Figura 15.8. Resultado dos gráficos sendo exibidos no relatório**

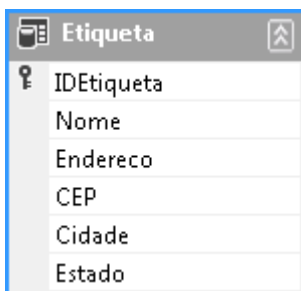
É importante ressaltar que os filtros do Tablix não são aplicados automaticamente nos controles de gráfico. Portanto, caso você queira que os filtros escolhidos também sejam aplicados nos gráficos, configure-os da mesma forma que fizemos para o Tablix no **Capítulo 8 (Filtrando os dados)**.

## Capítulo 16

# Trabalhando com colunas (para gerar etiquetas)

Em aplicativos que contém algum tipo de catálogo de endereços, é muito comum termos a necessidade de imprimir etiquetas (para envelopes) com os nomes e endereços das pessoas desse catálogo. Quando utilizamos o Report Viewer como ferramenta de relatórios, as etiquetas podem ser geradas através da funcionalidade de colunas no relatório.

Para demonstrarmos a geração de etiquetas com o Report Viewer, crie um outro DataSet que servirá de fonte de dados para esse novo exemplo. Chamaremos esse DataSet de *"DataSetEtiqueta"* e criaremos uma DataTable chamada *"Etiqueta"* dentro dele, com as seguintes colunas:



IDEtiqueta
Nome
Endereco
CEP
Cidade
Estado

**Figura 16.1. Definição da tabela que alimentará o relatório de etiquetas**

Todas as colunas dessa DataTable devem ser do tipo *"string"*, exceto a coluna *"IDEtiqueta"*, que deve ser do tipo auto incremento e deve ser configurada como chave primária.

Após termos criado esse novo DataSet, precisamos adicionar um novo relatório ao projeto. Dê o nome de *"RelatorioEtiqueta"* para esse novo relatório. Com o relatório criado, a primeira configuração que temos que alterar é a propriedade *"Columns"*. Nessa propriedade temos que definir o número de colunas que queremos utilizar no relatório. Para fins de exemplo, utilize duas colunas com espaçamento de 0,13 centímetros, conforme demonstrado na **Figura 16.2** a seguir.

Classes	
Code	
Columns	
Columns	2
ColumnSpacing	0.13cm
ConsumeContainerWhitesp	False
CustomProperties	

**Figura 16.2. Configurando a propriedade "Columns" com a quantidade de colunas desejadas e o espaçamento entre elas**

Normalmente, quando trabalhamos com etiquetas, é comum a utilização de folhas padronizadas. Para que as etiquetas sejam impressas corretamente no lugar certo, você terá que configurar as propriedades "Columns" e "ColumnSpacing" de acordo com a folha de etiquetas que você estiver utilizando.

Repare que, quando você altera a propriedade "Columns", o relatório apresentará a quantidade de colunas que você acabou de definir. No caso da configuração acima, conseguiremos visualizar duas colunas na área do relatório:

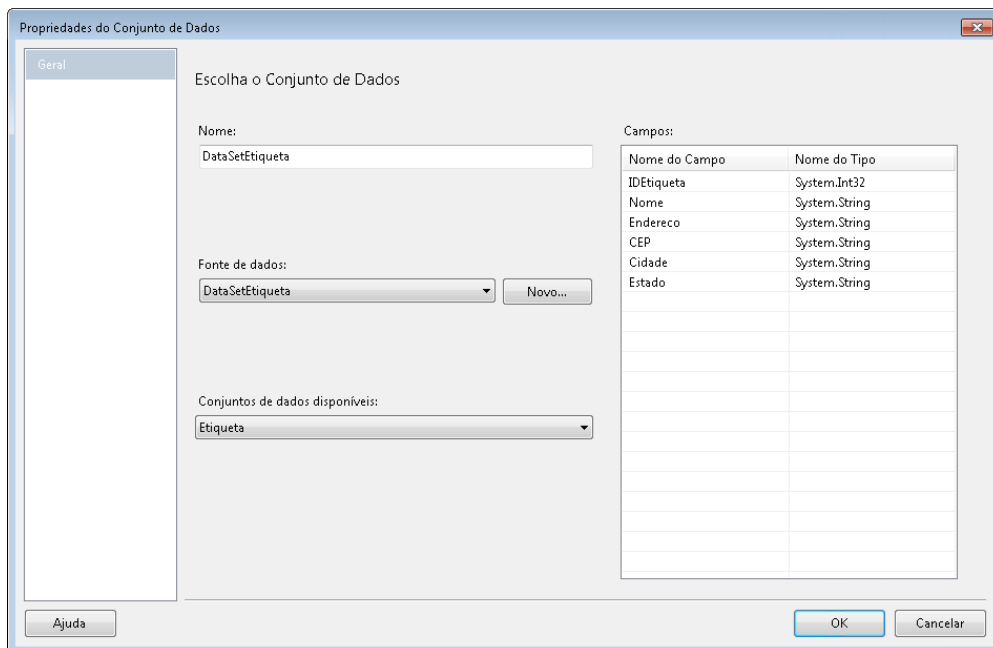
Para adicionar um item ao relatório: arraste um item da Caixa de Ferramentas para a superfície de design e arraste os campos de conjunto de dados para o item.	Coluna 2
--	----------

**Figura 16.3. Aparência de um relatório com duas colunas**

É possível configurarmos o tamanho horizontal das colunas utilizando o slider (a linha vertical entre uma coluna e outra). Para ter uma noção precisa do tamanho de cada coluna, recomendo que você ative a régua do relatório, conforme apresentado anteriormente no **Capítulo 5 (Entendendo o layout do relatório)**.

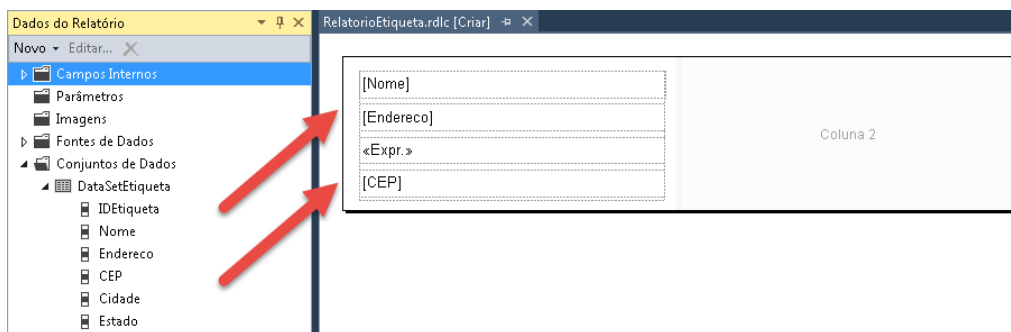
Apesar de conseguirmos visualizar as duas colunas que configuramos anteriormente, só é possível adicionarmos controles na primeira coluna da esquerda. O Report Viewer fará automaticamente a quebra dos dados de uma coluna para a outra, seguindo primeiramente na vertical (um registro abaixo do outro) e, quando a página tiver terminado, o Report Viewer começará a utilizar a segunda coluna.

Uma vez configurado o layout inicial do relatório, adicione um controle do tipo "List" dentro da primeira coluna do relatório e configure o DataSet desse controle, conforme apresentado na **Figura 16.4** a seguir.



**Figura 16.4. Configurando o DataSet do controle "List"**

Finalmente, arraste os campos do DataSet para dentro do controle "List":



**Figura 16.5. Arrastando os campos do DataSet para dentro do controle List**

Para o campo relacionado às colunas "Cidade" e "Estado", crie um TextBox com uma expressão customizada (já que queremos mostrar "Cidade / Estado"):

#### Listagem 16.1. Fórmula para concatenar os campos "Cidade" e "Estado"

```
1 =Fields!Cidade.Value & " / " & Fields!Estado.Value
```

Além de configurar corretamente o tamanho das colunas e seu espaçamento, é importante que você configure o tamanho da página e suas margens. Essas informações podem ser ajustadas nas propriedades do relatório, que foram previamente apresentadas no **Capítulo 5 (Entendendo o layout do relatório)**.

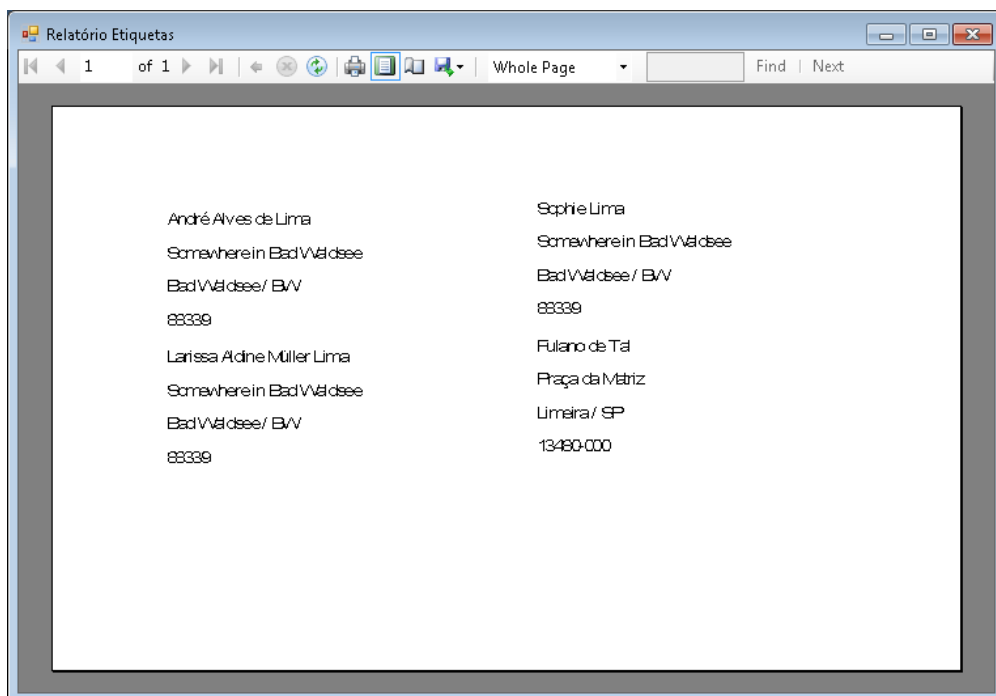
Uma vez criado o relatório, temos que exibi-lo. As etapas necessárias para criar um formulário que apresente o relatório foram detalhadas no **Capítulo 3 (Acessando o preview de relatórios locais)**. Siga essas mesmas etapas, só que dessa vez, escolha o novo relatório criado nesse capítulo (RelatorioEtiqueta).

No code-behind do formulário, encontre o método *"Load"* e adicione algumas linhas na tabela *"Etiqueta"*:

#### Listagem 16.2. Adicionando linhas fictícias na tabela *"Etiqueta"*

```
1 private void FormEtiqueta_Load(  
2     object sender, EventArgs e)  
3 {  
4     DataSetEtiqueta.Etiqueta.AddEtiquetaRow(  
5         "André Alves de Lima",  
6         "Somewhere in Bad Waldsee",  
7         "88339",  
8         "Bad Waldsee",  
9         "BW");  
10    DataSetEtiqueta.Etiqueta.AddEtiquetaRow(  
11        "Larissa Aldine Müller Lima",  
12        "Somewhere in Bad Waldsee",  
13        "88339",  
14        "Bad Waldsee",  
15        "BW");  
16    // Continua. Confira no projeto de exemplo  
17    // a versão completa deste método.
```

Ao executarmos a aplicação e exibirmos esse formulário, a primeira impressão é de que as colunas nem foram utilizadas. Na realidade elas foram. O que acontece é que, por padrão, o Report Viewer não exibe o preview em de impressão. Para vermos o resultado final, precisamos alterar o modo de visualização, clicando no botão *"Print Layout"*. Feito isso, o relatório será exibido com as colunas que configuramos anteriormente, conforme demonstrado na **Figura 16.6** a seguir.



**Figura 16.6. Relatório com etiquetas no modo "Print Layout"**

Caso você queira que o modo de impressão seja exibido por padrão, basta utilizar o método *"SetDisplayMode"* passando o *"PrintLayout"* antes da chamada de *"RefreshReport"* no método *"Load"* do formulário:

#### **Listagem 16.3. Configurando o Display Mode padrão para "Print Layout"**

```
1 this.reportViewer.SetDisplayMode(  
2     Microsoft.Reporting.WinForms.  
3     DisplayMode.PrintLayout);  
4 this.reportViewer.RefreshReport();
```

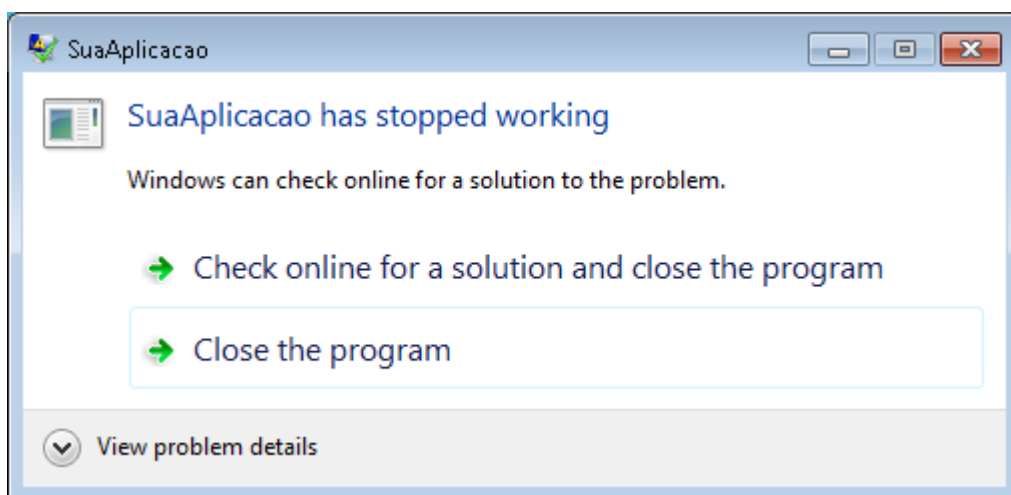


## Capítulo 17

# Deployment do controle Report Viewer

Sempre que desenvolvemos qualquer tipo de software, o último passo é fazer o deployment (distribuição) do aplicativo. Esse passo é muitas vezes ignorado pelos desenvolvedores, mas, não deveria. A experiência de instalação é o primeiro contato que o cliente terá com o sistema e, com certeza, você já ouviu falar que *“a primeira impressão é a que fica”*.

Quando trabalhamos com o Report Viewer, não basta simplesmente fazermos a distribuição do aplicativo, temos que distribuir também o Report Viewer, caso contrário, a aplicação não será executada e apresentará um erro parecido com este:



**Figura 17.1.** Erro ao tentarmos executar a aplicação sem a distribuição do Report Viewer

A distribuição do controle do Report Viewer pode ser feita de três maneiras: instalando o Report Viewer Redistributable manualmente (ou automaticamente de forma silenciosa), copiando as DLLs necessárias no diretório da aplicação ou publicando o aplicativo através do ClickOnce.

Nas próximas seções, veremos as diferenças, vantagens e desvantagens de cada um desses modelos de distribuição do Report Viewer.

## Utilizando o Report Viewer Redistributable

O Report Viewer Redistributable (ou Runtime) é um instalador à parte que serve para fazer a distribuição dos controles do Report Viewer. Se você está trabalhando com a

edição Express do Visual Studio, você já conhece essa runtime, já que foi através de sua instalação que você conseguiu ter acesso aos controles do Report Viewer no Visual Studio Express. Se quiser relembrar, confira novamente esse processo no **Capítulo 1 (Instalando o Report Viewer)**.

A runtime do Report Viewer está disponível para download no site da Microsoft. Para a sua conveniência, criei uma URL encurtada com o link onde ela pode ser baixada: <http://bit.ly/rv2012runtime>.

Sem sombra de dúvidas, a opção mais simples de todas é instalar manualmente essa runtime no computador onde o aplicativo será executado (ou no servidor, caso estejamos lidando com um website ASP.NET). Porém, como dito anteriormente, a primeira impressão é a que fica, e pedir para que o usuário instale manualmente a runtime do Report Viewer com certeza não causará uma primeira boa impressão.

A alternativa mais profissional é realizar a instalação silenciosa da runtime durante o processo de instalação do aplicativo. Se você está desenvolvendo um aplicativo profissional, muito provavelmente você terá que criar um instalador para a aplicação. Todo sistema de instalador possui a opção de chamarmos outros executáveis durante a instalação do aplicativo. Tudo o que temos que fazer é encontrar essa opção no gerador de instaladores e utilizar a linha de comando para instalar no Report Viewer de maneira silenciosa:

### Listagem 17.1. Linha de comando para instalar o Report Viewer Runtime de forma silenciosa

```
1 ReportViewer.exe /q:a /c:"install.exe /q"
```

Com essa linha de comando, o instalador da runtime do Report Viewer será executado, os controles do Report Viewer serão instalados e tudo isso sem que o usuário nem perceba (tudo de forma silenciosa).

Essa é a maneira mais fácil (e profissional) de fazer a instalação do Report Viewer. Se você estiver utilizando um instalador para distribuir a sua aplicação, nem pense duas vezes e utilize essa sistemática para distribuir o Report Viewer.

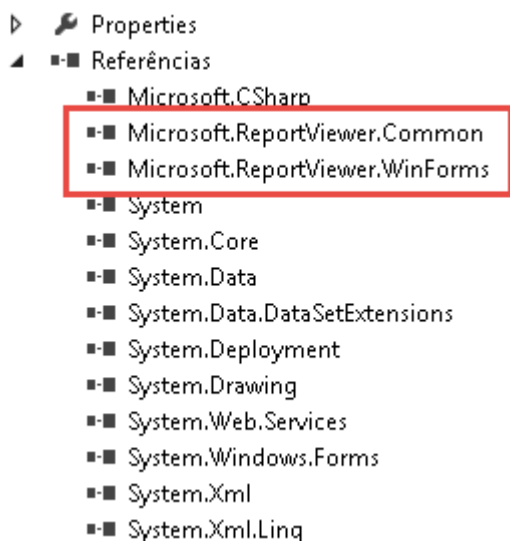
## Copiando as dlls no diretório de instalação

Algumas vezes, por algum motivo, não temos a possibilidade de criarmos instaladores para os nossos aplicativos e queremos copiar somente os arquivos necessários para a aplicação no computador cliente. Como fazemos com o Report Viewer nesse caso?

Por sorte, todas as funcionalidades do Report Viewer estão centralizadas em algumas poucas dlls (tanto que a runtime do Report Viewer tem somente 7.3Mb de tamanho). Uma vez que sabemos quais são essas dlls e onde elas estão localizadas, conseguimos

copiá-las e distribuí-las juntamente com o nosso aplicativo. Essa é também uma alternativa caso o seu sistema de instalador não dê suporte à execução de comandos customizados (de forma que a runtime não possa ser instalada automaticamente de forma silenciosa, como demonstrado na seção anterior).

Ao observarmos as referências no nosso projeto, já conseguimos identificar de cara dois assemblies utilizados pelo Report Viewer, o assembly com final “Common” e o assembly com final “WinForms”:



**Figura 17.2. Referências do Report Viewer no projeto Windows Forms**

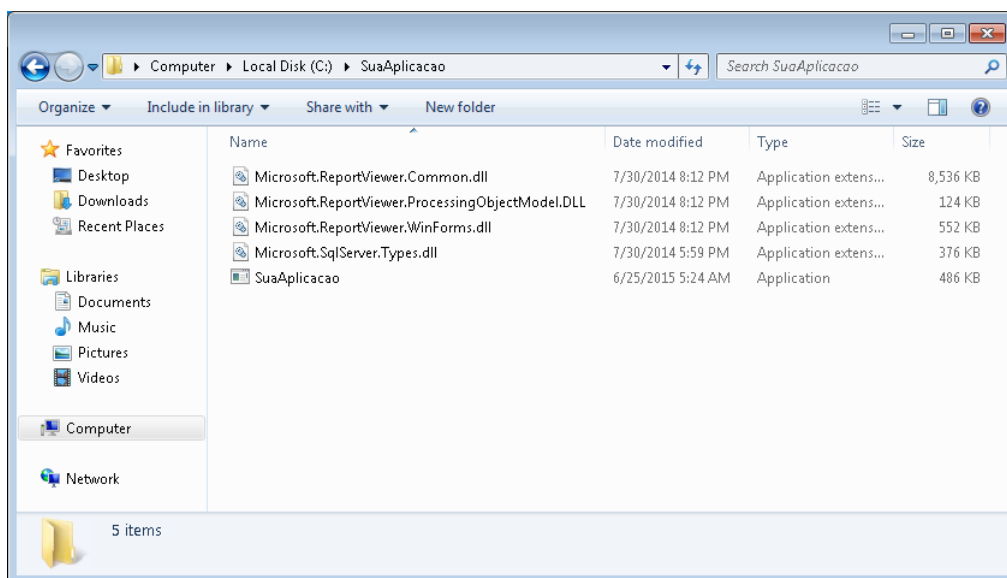
Se clicarmos nesses dois assemblies para verificarmos o caminho em disco onde eles estão armazenados, veremos que eles estão no GAC (C:\Windows\assembly\GAC\_MSIL). Se tentarmos acessar esse diretório a partir do Windows Explorer, receberemos uma mensagem dizendo que esse caminho não existe. Porém se tentarmos acessar qualquer um de seus subdiretórios (por exemplo, C:\Windows\assembly\GAC\_MSIL\Microsoft.Report Viewer.Common), conseguiremos acessá-lo normalmente (outra opção é acessar esse caminho via prompt de comando).

Dessa forma, para copiarmos as dlls necessárias para o Report Viewer, basta acessarmos os subdiretórios correspondentes aos assemblies do Report Viewer e copiarmos os arquivos para outra pasta. É importante notar que, caso mais de uma versão do Visual Studio esteja instalada no computador, as chances são grandes que mais de uma versão dos assemblies estejam disponíveis. Nesse caso, observe atentamente qual a versão do assembly utilizada no projeto para evitar conflitos na distribuição do aplicativo.

Além das duas dlls mencionadas acima, o Report Viewer necessita de outros dois assemblies, que também estão armazenados no GAC:

- Microsoft.ReportViewer.ProcessingObjectModel.dll
- Microsoft.SqlServer.Types.dll

Uma vez copiados esses quatro arquivos no mesmo diretório da aplicação, os relatórios serão carregados normalmente. Confira na figura a seguir um exemplo do diretório de instalação da aplicação com o executável e as dlls necessárias para o funcionamento do Report Viewer:

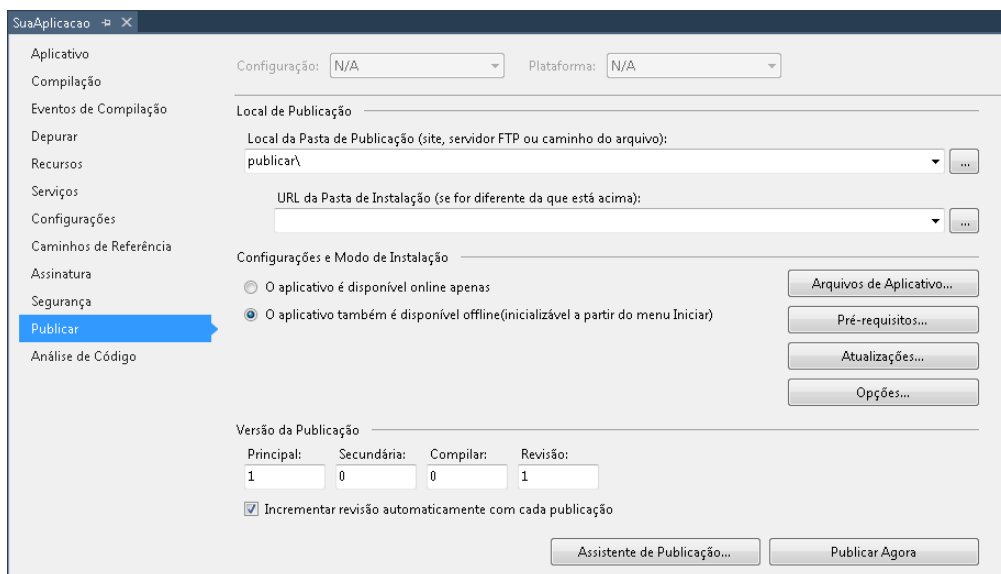


**Figura 17.3. Exemplo de diretório de instalação com as dlls demandadas pelo Report Viewer**

A vantagem dessa modalidade de distribuição é que ela é muito simples. Basta copiarmos esses arquivos em uma pasta no computador destino e a aplicação funcionará sem maiores problemas. A desvantagem é que ela não é nada profissional, uma vez que fazer com que o seu cliente copie arquivos de um lado para o outro definitivamente não causará uma boa impressão. Uma alternativa muito interessante é mesclarmos essa técnica com a utilização de um instalador (ou até mesmo um self-extract do WinRAR).

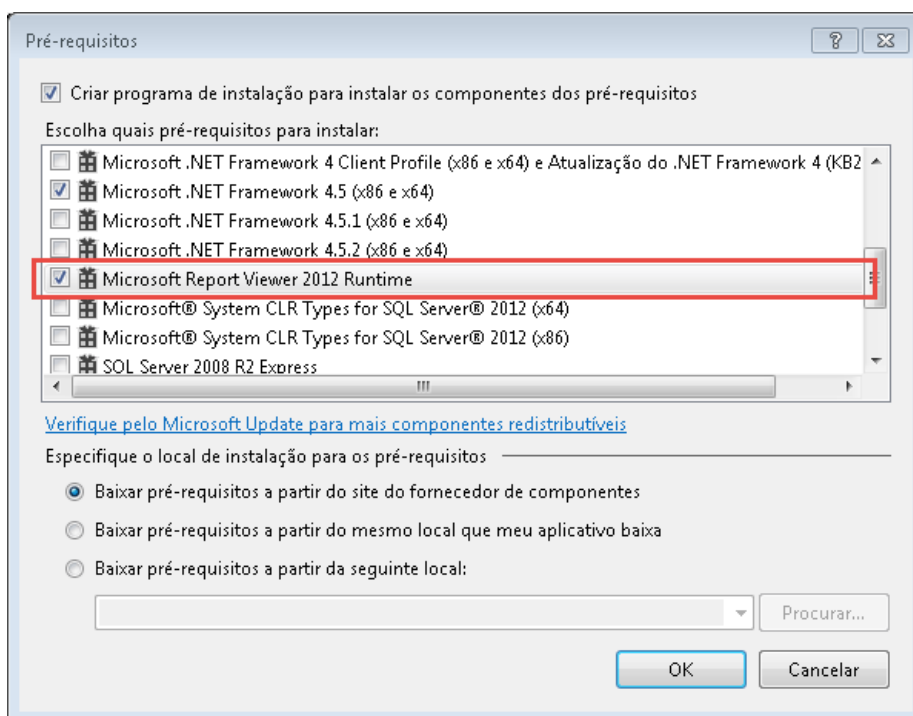
## Publicação com ClickOnce

Outra ferramenta muito utilizada na instalação de aplicações desktop é o ClickOnce. Essa opção está disponível em todas as edições do Visual Studio através da aba "Publicar" das propriedades do projeto, conforme demonstrado na **Figura 17.4**.



**Figura 17.4. Opções da publicação ClickOnce nas propriedades do projeto**

A distribuição do Report Viewer pelo ClickOnce é extremamente simples, uma vez que só temos que configurar o Report Viewer como pré-requisito da aplicação nas propriedades do ClickOnce. Para isso, clique no botão "Pré-requisitos", marque a opção "Microsoft Report Viewer 2012 Runtime" e escolha para que a runtime seja baixada do site do fornecedor de componentes:



**Figura 17.5. Configurando o Report Viewer como pré-requisito no ClickOnce**

Feito isso, o mecanismo do ClickOnce automaticamente baixará a runtime do Report Viewer e instalará juntamente com a aplicação.

Essa modalidade de instalação é a mais simples de todas no que diz respeito à distribuição do Report Viewer. Basta marcarmos uma caixa e tudo estará pronto. Porém, nesse caso temos a desvantagem que o ClickOnce demanda uma certa infraestrutura, uma vez que o pacote de distribuição deverá ser publicado em algum servidor IIS ou na rede local, e essa não é uma tarefa muito trivial. Além disso, os aplicativos instalados pelo ClickOnce não são armazenados na pasta *"Arquivos de Programas"* e não é possível escolhermos o seu local de instalação. Isso pode trazer problemas caso você esteja esperando que o aplicativo seja instalado em uma pasta específica ou dentro da pasta *"Arquivos de Programas"*.

## Sobre o autor

**A**ndré Alves de Lima trabalha desde 2005 com desenvolvimento, deployment e suporte de sistemas baseados no Microsoft .NET Framework. Fundador do Portal Abc-Dev e "Talking about Software Development, Technology and more" (<http://www.andrealves-lima.com.br>), escreve artigos técnicos desde 2006.

É MCPD e MCT Alumni, autor de vários artigos e vídeos para o portal Linha de Código / DevMedia e moderador de vários fóruns do portal MSDN, participando ativamente não só na moderação, mas também respondendo dúvidas nesses fóruns.

Em novembro de 2010 foi nomeado pela primeira vez Microsoft MVP (Most Valuable Professional) em Windows Platform Development e vem renovando o título todos os anos até a presente data (setembro de 2015).

