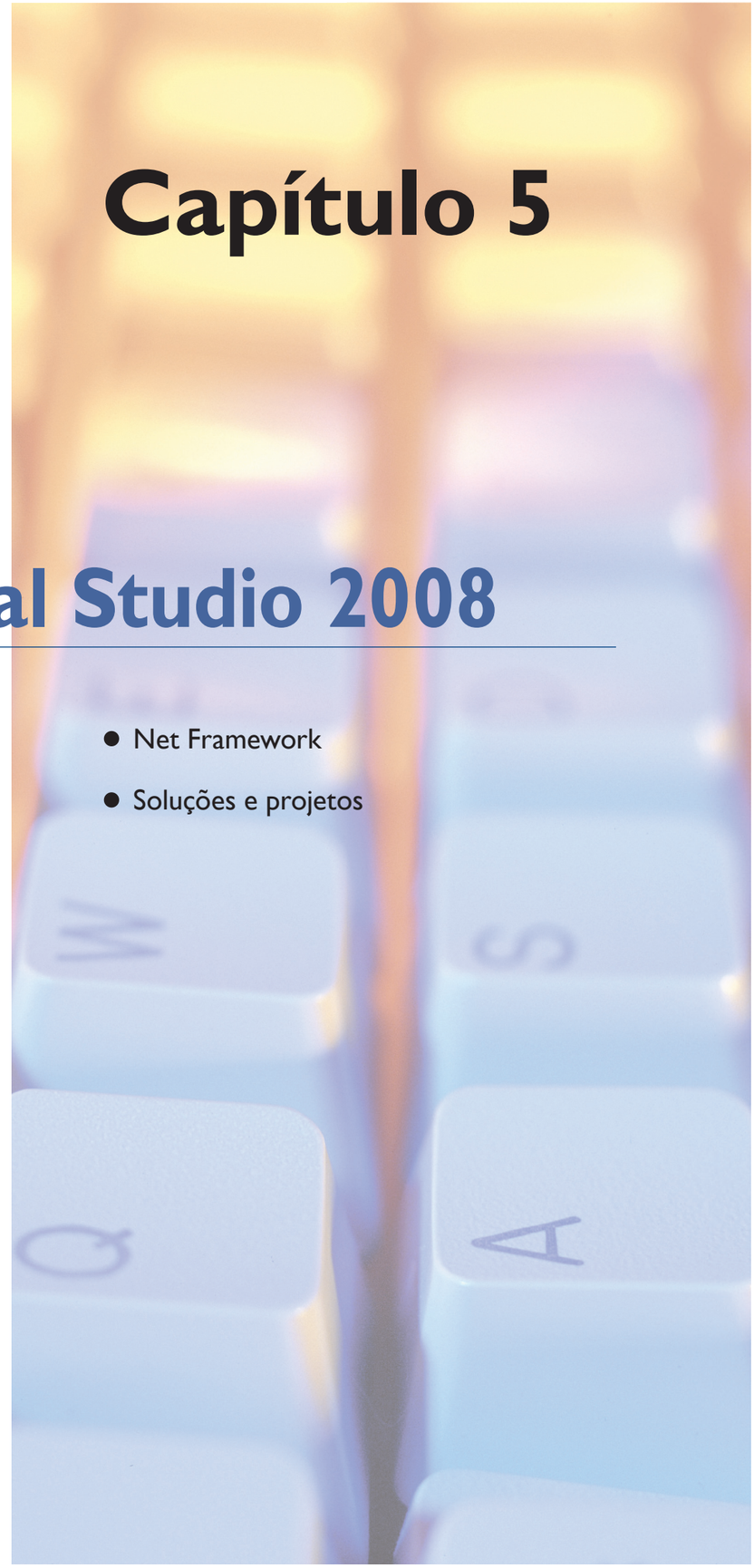
# Capítulo 5

# Visual Studio 2008

- Net Framework
- Soluções e projetos



**AJAX (acrônimo para a expressão em inglês Asynchronous Javascript And XML, que literalmente pode ser traduzido para Javascript e XML Assíncrono) é o nome dado à utilização metodológica de Javascript e XML para fazer com que as páginas web se tornem mais interativas.**

O Visual Studio é um conjunto de ferramentas de desenvolvimento que contém editores de códigos, IntelliSense, assistentes e diferentes linguagens em um mesmo ambiente de desenvolvimento integrado para principiantes e profissionais. Apresenta-se em diferentes plataformas: PC´s, servidores, aplicações web e móveis. Em uma visão mais abrangente, o Visual Studio permite o desenvolvimento rápido de aplicativos, recursos de depuração e banco de dados, sem depender dos recursos oferecidos pelo Framework 3.5. Auxilia no desenvolvimento Web habilitado para o **AJAX**, contando ainda com os recursos do **ASP.NET**.

## 5.1. .NET Framework

**ASP.NET, sucessora da tecnologia ASP (de Active Server Pages ou páginas de servidor ativo) é a plataforma da Microsoft usada para o desenvolvimento de aplicações web.**

Desenvolvido pela Microsoft, o .NET Framework é um modelo de programação de código gerenciado para criar aplicativos cliente, servidores ou dispositivos móveis. É formado por um conjunto variado de bibliotecas que facilitam o desenvolvimento de aplicações, desde as mais simples até as mais complexas, bem como a instalação e distribuição de aplicações. Baseado em tecnologia de máquina virtual, o .NET Framework é totalmente orientado a objetos.

O **.NET Framework 3.5** incrementa as versões anteriores com novas implementações ASP.NET e AJAX e aumenta a integração com o LINQ (Language Integrated Query, ou consulta integrada de linguagem) que é uma nova ferramenta de pesquisas em base de dados, além de suporte total para Windows Workflow Foundation (WF), Windows Communication Foundation (WCF), Windows Presentation Foundation (WPF) e Windows CardSpace. Sua estrutura é composta por diferentes camadas, como podemos visualizar na figura 196.

**O Microsoft .NET Framework 3.5 é o modelo de programação do Windows Vista e, segundo a própria Microsoft ao anunciar o lançamento, "combina o poder do .NET Framework 2.0 com novas tecnologias para construção de aplicativos". Permite a realização de novas experiências, comunicação integrada e sem fronteiras, além de ser útil para vários processos corporativos.**

Na camada inferior, encontramos a Common Language Runtime (CLR) ou tempo de execução de linguagem comum. Sua finalidade é executar as aplicações, criando um ambiente de máquina virtual e compilando as linguagens de programação do .NET Framework em código nativo. O .NET Frameworks Base Class, na segunda camada de baixo para cima, representa as bibliotecas de classes disponíveis para o desenvolvimento de aplicativos (consulte o quadro *Recursos de classes disponíveis* na página 188). É o principal ponto de interatividade com o Runtime (tempo de execução).

Na terceira camada ascendente, está o ADO.NET (Data e XML). O ActiveX Data Objects (ADO) oferece todos os recursos necessários para a criação e manipulação de bancos de dados fornecidos por meio das classes System.Data, .Common, .OleDb, .SqlClients, SqlTypes, .Odbc e .Xml.



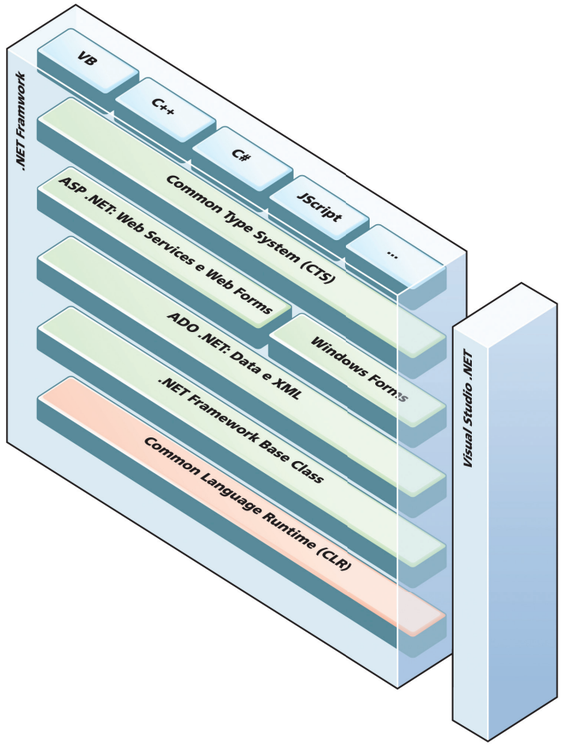



**Figura 196**
.NET Framework.

Na quarta camada temos Web Services e Web Forms. O Web Service representa a integração entre os dados de diferentes aplicações e plataformas, permitindo o envio e recepção de dados no formato XML. Tudo de maneira muito fácil. Para que isso ocorra, entra em cena o Web Forms, criando um ambiente de desenvolvimento semelhante às ferramentas que normalmente utilizamos, clicando e arrastando, assim como se faz no FrontPage da Microsoft. Já o Windows Form é uma evolução dos formulários utilizados para programação.

Localizado na penúltima camada de baixo para cima, o Common Type System (CTS), que pode ser literalmente traduzido como sistema de tipo comum, existe para que ocorra a integração entre as linguagens de programação. Define como os tipos de dados serão declarados, usados e gerenciados no momento da execução do aplicativo.

No topo, aparecem as linguagens de programação utilizadas para o desenvolvimento da aplicação, como VB, C++, C# e JScript. Assim, pode-se concluir que o conjunto de todas as camadas mencionadas representa o pacote do Visual Studio.Net.

### 5.1.1. Máquina virtual

A máquina virtual serve como uma camada entre o código e o sistema operacional. Todo código escrito no .NET (Visual Basic, C# ou qualquer outra lingua-

gem), é compilado para uma linguagem intermediária chamada CIL (Common Intermediate Language ou linguagem intermediária comum), que é distribuída e executada pelos diferentes clientes da aplicação.

### 5.1.2. Garbage collector (coletor de lixo)

Mecanismo interno que possibilita a retirada da memória de objetos que não estão sendo mais utilizados. A operação é feita sem a interferência do usuário, em intervalos de ociosidade da CPU.

## 5.2. Soluções e projetos

Quando iniciamos uma aplicação ou serviço no Visual Studio, temos um projeto que funciona como um repositório para gerenciamento dos códigos fonte, conexões com bancos, arquivos e referências. Ele representa uma pasta da Solução (Solution), que por sua vez poderá conter inúmeros projetos independentes

### Recursos de classes disponíveis

- **System:** entre os seus inúmeros recursos, está o suporte para programação, os tipos de bases (String, Int32, DateTime, Boolean etc.) e as funções matemáticas.
- **System.CodeDom:** para a criação e execução de código de maneira imediata.
- **System.Collections:** define containers como listas, filas, matrizes etc.
- **System.Diagnostics:** todas as classes necessárias para fazer diagnósticos.
- **System.Globalization:** suporte para a globalização, ou seja, essa classe integra toda a plataforma do Framework.
- **System.IO:** suporte para o FileSystem, usando classes de manipulação de arquivos e diretórios.
- **System.Resources:** usado para tradução do aplicativo em diferentes idiomas e também para retorno de mensagem de acordo com o idioma selecionado pelo usuário.
- **System.Text:** suporte para a codificação e ao StringBuilder, para manipulação de Strings.
- **System.Text.RegularExpressions:** suporte para expressões regulares.

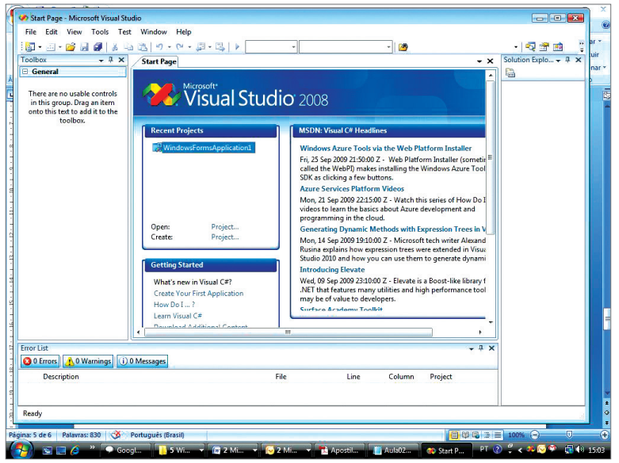



**Figura 197**
O Visual Studio.

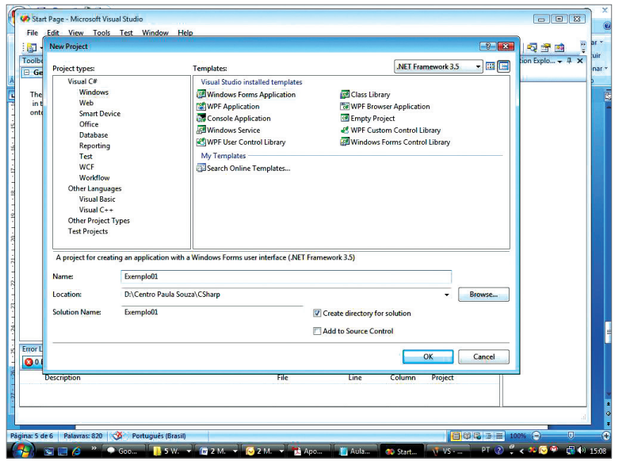



**Figura 198**
Janela de Projeto.

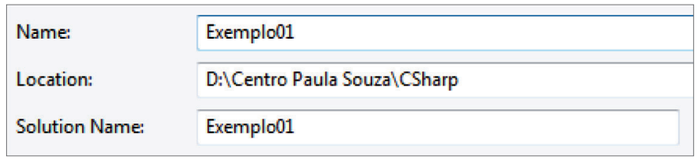
entre si, elaborados em diferentes linguagens e organizados no formato de pastas semelhante ao Windows Explorer. Os arquivos de projetos (.vbproj, .csproj etc.) e os arquivos de solução (.sln) estão no formato XML.

### 5.2.1. Iniciando o Visual Studio – Solution

Ao iniciar o Visual Studio pela primeira vez (figura 197), surge a tela de Start Page (ou página inicial). No menu File (arquivo), entre em New (novo) e clique em Project (projeto), como ilustra a figura 198.

Em templates (modelos), escolha o tipo de projeto a ser elaborado (Windows Form Application ou aplicação de formulário Windows, Console Application ou aplicação de painel de controle etc.). Lembre-se de nomear o seu projeto e

**Figura 199**
Definição da Solution.




indicar a localização, ou seja, onde os arquivos serão gravados e o nome da sua Solution (solução), como mostra a figura 199. Confirme os dados e teremos a Solution aberta para o desenvolvimento da aplicação.

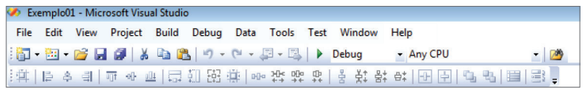
Caso o projeto seja elaborado utilizando o Visual Basic, o procedimento será o mesmo. Escolha em Project Types (tipos de projeto) a opção Others Languages (outras linguagens), depois Visual Basic e, por fim, a template na qual deseja trabalhar.

## 5.2.2. Conhecendo o Visual Studio

O Visual Studio apresenta uma série de janelas e guias que fazem parte da sua IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado). Confira algumas, a seguir.

**Barra de ferramentas** (toolbar), que disponibiliza os botões de comandos mais utilizados (figura 200).

**Figura 200**
Toolbar.




**Solution Explorer:** mostra os arquivos que fazem parte do seu projeto. É semelhante ao Explorer do Windows, pois permite criar, excluir e importar arquivos (figura 201).

Se analisarmos uma aplicação Windows Form Application em C#, dentro da janela Solution, podemos encontrar os seguintes arquivos: Form1.cs, para programação do formulário; Form1.Designer.cs, para programação visual do formulário; e Program.cs, o programa principal, no qual encontramos o método main(), que dará início à aplicação.

**Figura 201**
Solution Explorer.

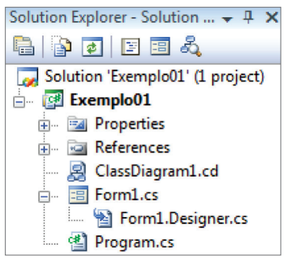
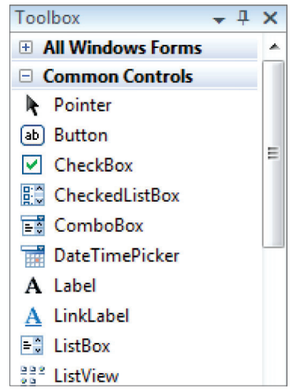





**Figura 202**
ToolBox.




Para aplicações em Visual Basic, será disponibilizado o arquivo Form1.vb, que possui a mesma função do Form1.cs. Confira, a seguir, janelas disponíveis e suas funções:

**ToolBox (caixa de ferramentas):** contém componentes para o desenvolvimento do projeto, os quais estão divididos em guias de acordo com o tipo de aplicação. Nesse caso, podemos verificar (figura 202) a aba de componentes da categoria Common Controls (controles comuns).

**Form (formulário):** essa janela (figura 203) receberá os componentes da toolbox e a programação correspondente. Os componentes serão "arrastados" sobre o Form para compor a interface do usuário e terão suas propriedades modificadas de acordo com o projeto. Para ativar a janela de código e realizar a programação, é preciso dar um duplo clique sobre o componente ou formulário.

**Properties (propriedades):** permite alterar as propriedades dos componentes, que podem aparecer organizadas por categoria ou em ordem alfabética (figura 204).

A janela de propriedade traz, de acordo com cada componente toolbox, uma série de recursos para configuração. Por exemplo, um componente Label (rótulo), utilizado

**Figura 203**
Form.

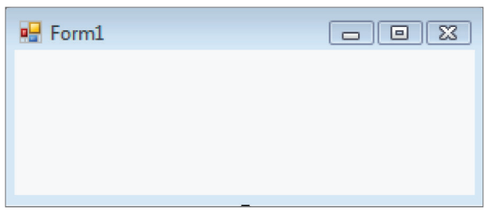
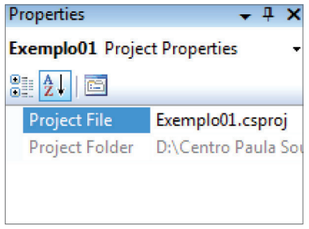


**Figura 204**
Properties.




para incluir expressões no projeto, possuindo diversas propriedades – tais como name (nome), text (texto), visible (visível), font (fonte), forecolor (cor) – e que poderão ser manipuladas diretamente na janela ou por meio de programação, utilizando a notação de "ponto", conforme o seguinte código, que oculta o Label:

```
Label1.Visible = false;
```

Code and Text Editor (editor de texto e de código) é a janela que utilizaremos para implementar os códigos de programação (figura 205). Para acessá-la, basta clicar duas vezes sobre ela ou em qualquer componente. Outra opção é usar os botões Code (código) e View (visualizar) na janela de Solution Explorer, como mostra a figura 206.

**Figura 205**
Code and text Editor.

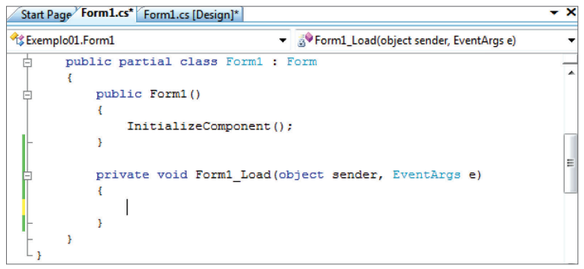
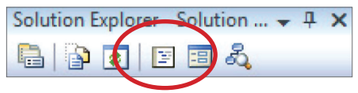



**Figura 206**
Em destaque, os botões Code e View.




### 5.2.3. Gerenciador de janelas

O Visual Studio apresenta muitas janelas e, para facilitar o seu gerenciamento, existem quatro recursos. Confira quais são, a seguir.

**Dockable:** coloca a janela aberta junto à janela principal do programa.

**Hide:** fecha aquela janela e, para abri-la novamente, usa o menu View.

**Floating:** a janela fica flutuante, podendo ser deslocada para qualquer parte do desktop.



**Figura 207**
Auto Hide.

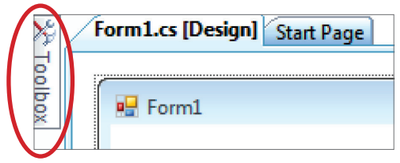
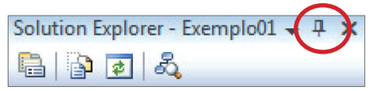


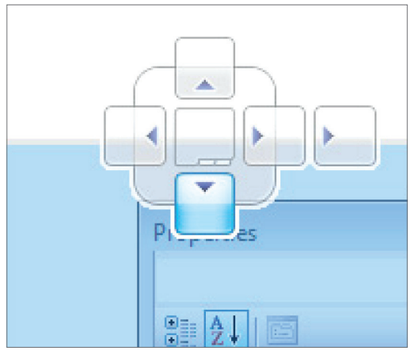

**Figura 208**
Fixar janela.




**Auto Hide:** um apontador indica a posição da janela, que ficará oculta. Basta um simples movimento do mouse sobre o título dessa janela (figura 207) para que ela seja aberta. Para fixá-la, utilize o ícone semelhante a um preguinho (figura 208), localizado na barra de título.

Podemos movimentar as janelas e colocá-las em qualquer lugar da aplicação com a ajuda dos guias. Para isso, clique na barra de título e mantenha o botão do mouse pressionado, arrastando a janela para qualquer área de seu desktop. Imediatamente, surgirão as indicações das guias (figura 209). Escolha uma delas e solte a janela.

**Figura 209**
Guias de janelas.



### 5.2.4. Nomenclatura de componentes

Cada componente recebe uma numeração automática quando inserido no projeto (Label1, Label2, Label3 etc.). Não há problemas em manter esses nomes, mas, para deixar o código mais legível e padronizado, o melhor é utilizar um prefixo relacionado ao tipo de componente e à sua identificação. Por exemplo:

```
Label1 = lblPergunta

Label2 = lblMensagemAlerta
```

A tabela 12 mostra alguns dos vários prefixos utilizados na plataforma .NET.

**Tabela 12**

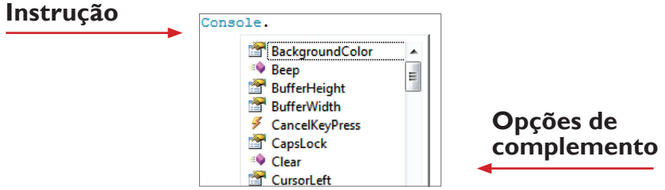
**PREFIXOS USADOS NA .NET**

| Componente | Prefixo | Componente | Prefixo |
|---|---|---|---|
| Label | lbl | ListBox | Lst |
| TextBox | txt | DataList | Dtl |
| DataGrid | dtg | Repeater | Rep |
| Button | btn | Checkbox | Chk |
| ImageButton | imb | CheckBoxList | Cbl |
| DropDownList | ddl | RadioButton | Rdo |
| RadioButtonList | rbl | PlaceHolder | PhD |
| Image | img | Table | Tbl |
| Panel | pnl | Validators | Val |

### 5.2.5. IntelliSense

Ao digitar alguma instrução, aparecerá uma série de complementos relacionados a ela. Quando escrevemos "Console", por exemplo, são disponibilizados vários métodos. Com a ajuda da tecla Tab ou da Barra de Espaço, a instrução  se compõe naturalmente (figura 210).

**Figura 210**
IntelliSense.




### 5.2.6. Executando a aplicação

Para executar a aplicação, pressione a tecla "F5", ou, na barra de menu, clique no item Debug. Escolha a opção Start Debugging ou utilize o botão da barra de ferramentas (figura 211).

**Figura 211**
Executando a aplicação.

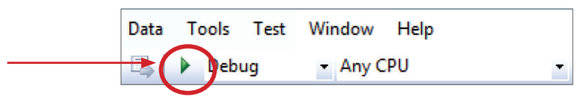



**Figura 212**
Controles de execução.





Durante essa atividade, podemos recorrer a alguns botões auxiliares (figura 212) como Break, Stop e Restart.

### 5.2.7. Identificação de erros

O Visual Studio nos ajuda a identificar ou interpretar alguns erros que podem ocorrer durante o processo de criação ou execução do código. Confira alguns, a seguir.

**Erro de sintaxe:** geralmente é identificado com uma linha em vermelho sublinhando a expressão. No exemplo mostrado na figura 213, estão faltando as aspas no fechamento da expressão.

```
Console.write("não sei o que isso faz);
```

**Figura 213**
Erro de sintaxe.

**Erro antes da execução:** quando o código apresenta algum erro e uma execução é forçada, uma caixa de diálogo solicita ao usuário que continue a execução do código, mesmo constando erro (figura 214). Isso faz com que a última versão correta seja executada, ignorando a atual.

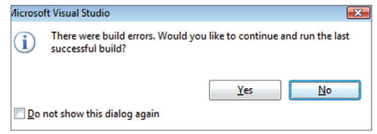



**Figura 214**
Janela de erro.

Na parte inferior do Visual Studio, podemos visualizar o painel (error list) de Erros, Warnings e Messages (figura 215).

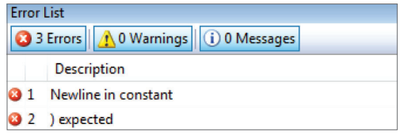



**Figura 215**
Painel de erros.

Clique sobre o erro identificado (1, 2 etc.) para que ele seja selecionado no código para visualização.

**Erro de classe ou método:** comumente identificado com uma linha em azul sublinhando a expressão (figura 216). No exemplo, a instrução está escrita de forma errada, pois o correto é WriteLine().

```
Console.WriteLn("não sei o que isso faz");
```

**Figura 216**
Erro de classe.