



**DESENVOLVIMENTO DE APLICAÇÕES PARA
DISPOSITIVO MÓVEIS COM .NET**

TRABALHO DE GRADUAÇÃO

Aluno: Vanilson André de Arruda Burégio (vaab@cin.ufpe.br)

Orientador: André Luís Medeiros dos Santos (alms@cin.ufpe.br)

Agosto de 2003

Resumo

O uso de aplicações para dispositivos móveis, como celulares, PDAs, Handhelds tem se tornado cada vez maior entre pessoas e empresas que necessitam de grande flexibilidade no acesso e troca de informações. Porém, a grande variedade de dispositivos existentes, a falta de frameworks e ferramentas adequadas de desenvolvimento têm dificultado muito a construção dessas aplicações.

O objetivo deste trabalho é estudar e analisar características das possíveis arquiteturas (abordagens) que podem ser adotadas no desenvolvimento de aplicativos para dispositivos móveis utilizando a plataforma .Net da Microsoft, que visa resolver/minimizar os problemas de desenvolvimento citados.

Agradecimentos

Primeiramente quero aqui registrar meus agradecimentos a Deus por haver-me concedido a oportunidade de realizar este trabalho

Agradeço também à minha família pelo grande incentivo prestado no decorrer de todas as atividades envolvidas na confecção deste trabalho

Em particular, agradeço a meu orientador prof. André Luís Medeiros dos Santos pela ajuda prestada diante das dúvidas e pelo trabalho cooperativo realizado durante este período.

Agradeço ao CESAR, na pessoa de Luciana Valadares, por ter me prestado total apoio em momentos de difícil conciliação entre o trabalho e a vida acadêmica.

Agradeço aos amigos Marden, Renato Cezar e a todos do grupo Sharp Shooters pelas dicas e material de apoio fornecido.

Meus agradecimentos também são direcionados a todos que contribuíram direta ou indiretamente para a realização deste trabalho, através de críticas, sugestões e/ou cobranças.

Vanilson Burégio

Conteúdo

Resumo	2
Agradecimentos	3
Conteúdo	4
Índice de Figuras	6
Índice de Tabelas	6
1. Introdução	7
1.1 Objetivo	8
2. Introdução ao .NET	9
2.1 O que é .NET?	9
2.2 .NET Framework	9
2.3 Arquitetura do .NET Framework	10
2.3.1 Plataforma	10
2.3.2 Serviços de Aplicação	10
2.3.3 .NET Framework Class Library	11
2.3.4 Common Language Runtime (CLR)	11
2.3.5 Microsoft ADO.NET	11
2.3.6 ASP.NET	11
2.3.7 XML Web Services	12
2.3.8 User Interfaces	12
2.3.9 Linguagens	12
2.4 Desenvolvendo aplicações com .NET Framework	13
2.5 Vantagens em usar o .NET Framework	15
3. Aplicações e dispositivos móveis	17
3.1 Dispositivos móveis	17
3.1.1 Histórico de evolução	17
3.1.2 Considerações sobre o trabalho	20
3.2 Aplicações móveis e .NET	20
3.2.1 Ambiente de Desenvolvimento	21
3.2.2 Abordagens (Arquiteturas)	22
3.2.3 Ferramentas	22
4. Aplicações <i>Client-side</i>	24
4.1 .NET Compact Framework	24
4.1.1 O que é?	24
4.1.2 Plataformas suportadas	24
4.1.3 Biblioteca de classes	25
4.1.4 Benefícios	27

4.2	Principais Características	28
4.2.1	Aplicações executam no dispositivo.....	28
4.2.2	Aplicações suportadas por poucos dispositivos.....	28
4.2.3	Rica interface gráfica.....	28
4.2.4	Opções de trabalho off-line.....	28
4.3	Protótipos desenvolvidos	29
4.3.1	HelloWorld.....	29
4.3.2	Animação gráfica.....	32
4.3.3	Banco de dados em memória.....	36
4.3.4	Uso de arquivos XML.....	38
5.	Aplicações Web Móveis	41
5.1	ASP .NET e Mobile Internet Toolkit	41
5.1.1	Visão Geral de ASP .NET.....	41
5.2	Funcionamento de aplicações Web móveis	46
5.3	Características	Erro! Indicador não definido.
5.3.1	Aplicações localizadas no servidor.....	48
5.3.2	Suporte a múltiplos dispositivos.....	48
5.3.3	Suporte a múltiplos linguagens de marcação de texto.....	48
5.3.4	Necessidade de trabalho on-line.....	49
5.3.5	Menos poder de interface gráfica.....	49
5.3.6	Fácil deployment.....	50
5.4	Protótipos	50
5.4.1	HelloWorld.....	50
6.	Aplicações baseadas em Web Services	54
6.1	Introdução a Web-Services	54
6.1.1	O que são Web Services?.....	54
6.1.2	SOAP e Web Services.....	54
6.2	Principais Características	58
6.2.1	Comunicação entre aplicações (<i>application-to-application</i>).....	58
6.2.2	Independente de Linguagem.....	58
6.2.3	Independente de Protocolo.....	58
6.2.4	Independente de Plataforma.....	59
6.2.5	Arquitetura Stateless.....	59
6.3	Web Services e Dispositivos móveis	59
6.3.1	Consumo de web-services por aplicações client-side.....	59
6.3.2	Aplicações Web Móveis vs Web Services.....	63
7.	Conclusão	67
	Referências	68

Índice de Figuras

Figura 1 – Arquitetura do .NET Framework	10
Figura 2 – Processo de compilação de aplicações .NET	14
Figura 3 – Osborne 1 o primeiro computador portátil	18
Figura 4 - Newton Message Pad	19
Figura 5 – Dispositivos móveis mais recentes	20
Figura 6 – Plataforma de desenvolvimento para dispositivos móveis.....	23
Figura 7 – Layout da Biblioteca de Classes do .NET CF	26
Figura 8 – Desing no Visual Studio do protótipo HelloWorld	30
Figura 10 – HelloWorld executando no emulador para Pocket PC	31
Figura 11- Figura com seqüência de frames para animação (disquete.gif)	33
Figura 12 – Aplicação com animação executando no emulador.....	34
Figura 13 – Aplicação DataGridView.cs executando no emulador.....	38
Figura 16 – Aplicação com TreeView	40
Figura 17 – Componentes do ASP .NET	43
Figura 18 – Compilação de página ASPX.....	47
Figura 19 - Arquitetura de aplicações Web Móveis	47
Figura 23 - Estrutura de uma mensagem SOAP	56
Figura 24 - Proxy de um Web Service	57
Figura 25 – Ciclo de vida de uma aplicação cliente que acessa um Web Service ...	57
Figura 26 – Cliente consumindo o serviço de tradução BabelFishService	62
Figura 27 – Arquitetura de Web Service com ASP .NET	65

Índice de Tabelas

Tabela 1 - HelloWorld.aspx gerado para o Internet Explorer	51
Tabela 2 – HelloWorld.aspx gerado para o Browser OpenWave 4.1.1	52
Tabela 3 - HelloWorld.aspx gerado para o Browser OpenWave 5.2	52
Tabela 4 - HelloWorld.aspx gerado para o Browser OpenWave 6.2.2	53
Tabela 5 – Informações do BabelFishService.wsdl	62

1. Introdução

O uso de dispositivos móveis, como celulares, PDAs, Handhelds tem se tornado cada vez maior entre pessoas e empresas que necessitam de grande flexibilidade no acesso e troca de informações. Junto com a popularidade desses dispositivos surge a necessidade de implementar aplicações cada vez mais sofisticadas e que consigam atender de maneira satisfatória os desejos de seus usuários. As aplicações para dispositivos móveis podem variar desde uma simples aplicação stand-alone a sistemas que permitam a troca de dados remotamente através da internet utilizando tecnologias como, por exemplo, Web Services.

O desenvolvimento de aplicações para dispositivos móveis requer um cuidado especial, pois estes geralmente apresentam limitações de memória, processamento e resolução que devem ser levadas em consideração no projeto destas aplicações. Além disso, a grande variedade dos dispositivos existentes e a falta de ferramentas de desenvolvimento adequadas e frameworks tem dificultado muito a construção de aplicações portáteis e que utilizem, de maneira otimizada, os recursos específicos de determinados equipamentos. Pode-se perceber claramente que a maioria dos problemas acontece devido a fatores inerentes ao dispositivo em uso e que estes problemas são agravados atualmente pela ausência de ferramentas, técnicas e frameworks estruturados que permitam um desenvolvimento mais fácil das aplicações.

O surgimento de aplicações web móveis também tem gerado algumas questões que devem ser levadas em consideração no projeto de tais aplicações. Dentre elas pode-se citar:

- Variedade de dispositivos com diferentes capacidades
- Múltiplas linguagens de marcação de texto
- Múltiplas implementações de browsers
- Múltiplas implementações do padrão wap.
- Variações na capacidade de processamento client-side
- Falta de um ambiente sofisticado de desenvolvimento e debugging de aplicações web

Tendo em vista resolver/minimizar os problemas citados a Microsoft lançou a plataforma .Net que pode ser utilizada como uma tecnologia alternativa para o desenvolvimento de aplicações para dispositivos móveis. A plataforma .NET faz a promessa de que o desenvolvedor será capaz de implementar aplicações móveis utilizando o paradigma de programação orientado a objetos sem se preocupar com a diferença existente entre os diversos tipos dispositivos. Com .Net é possível implementar tanto aplicações client-side que executam em um dispositivo específico, como também aplicações web que serão acessadas via browsers dos dispositivos móveis. Cada abordagem apresenta suas próprias características e nuances que devem ser analisadas e consideradas na construção de diferentes tipos de aplicações.

1.1 Objetivo

Dentro do contexto apresentado, o objetivo deste trabalho é estudar e analisar as possíveis arquiteturas (abordagens) que podem ser adotadas no desenvolvimento de aplicativos para dispositivos móveis utilizando a plataforma .Net.

No estudo serão identificadas as características de cada abordagem e os possíveis cenários onde as mesmas poderão ser utilizadas. Na prática serão desenvolvidos protótipos que permitam validar os resultados obtidos, testar os simuladores e ferramentas disponíveis. Além disso, serão tratados também outros aspectos como, por exemplo, o consumo de Web Services por PDAs.

2. Introdução ao .NET

A plataforma de desenvolvimento .NET introduz novos conceitos, tecnologias e termos. O objetivo deste capítulo é explicar o significado do termo .NET, bem como oferecer uma visão geral do que consiste o .NET Framework.

Será apresentada a arquitetura do .NET framework e uma descrição de como desenvolver utilizando a tecnologia. Os conceitos introduzidos aqui serão de grande utilidade para um melhor entendimento das outras Seções deste documento.

2.1 O que é .NET?

Em linhas gerais, o termo .NET pode ser definido como um modelo de desenvolvimento, criado pela Microsoft, que visa a implementação de software independente de plataforma e dispositivo. Um dos principais objetivos desse modelo é permitir a integração entre aplicações através da troca de informações pela internet. O uso da plataforma .NET facilita o desenvolvimento de aplicações Web, mas pode ser aplicada na implementação de aplicações desktop também.

A plataforma .NET consiste de um conjunto de tecnologias/componentes que incluem desde o modelo de programação e ferramentas de desenvolvimento a servidores corporativos e serviços Web (Web Services) que podem ser utilizados por diversos tipos de aplicações. Um dos principais elementos da plataforma .NET é o .NET Framework, apresentado a seguir.

2.2 .NET Framework

O .NET Framework é a principal parte da plataforma .NET. Ele é a infraestrutura utilizada no modelo .NET e prove os serviços e componentes necessários para o desenvolvimento e execução de aplicações baseadas em .NET.

Os dois principais componentes do framework são:

- **CLR** (Common Language Runtime) – ambiente de execução das aplicações .NET.
- **Bibliotecas de Classes** (.NET Classe Library) – componentes de infraestrutura utilizados na implementação das aplicações.

2.3 Arquitetura do .NET Framework

Os componentes que formam a infra-estrutura da plataforma .NET estão organizados de acordo com o mostrado na Figura 1.

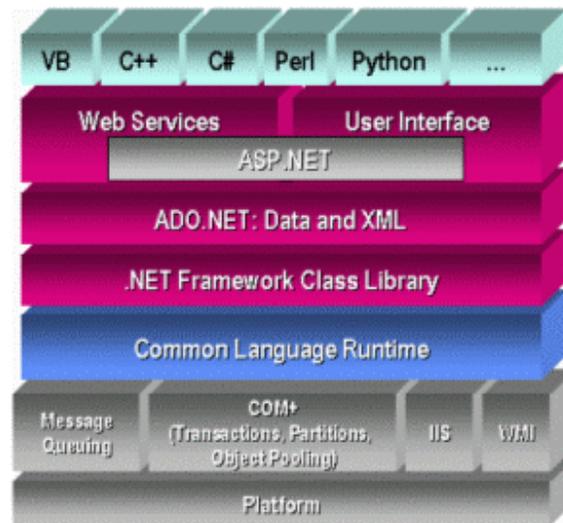


Figura 1 – Arquitetura do .NET Framework

2.3.1 Plataforma

Representa o sistema operacional onde o .NET Framework deve executar. Inicialmente apenas a família Windows (Win32) de sistemas operacionais, tais como, Windows 2000, Windows XP e Windows 98 suportavam a plataforma .NET. Atualmente o framework é suportado em outras plataformas, como o Windows CE (utilizado em dispositivos móveis), além disso, existem projetos para a execução do .NET framework em sistemas operacionais como o Linux, por exemplo.

2.3.2 Serviços de Aplicação

Representa os serviços que estão disponíveis para o desenvolvedor quando as aplicações executam na plataforma Windows. Como exemplos desses serviços tem-se o IIS (*Internet Information Services*), o WMI (*Windows Management Instrumentation*), o *Message Queuing*, entre outros. Os serviços de aplicação podem

ser acessados através de classes da biblioteca do Framework (*.NET Framework class library*).

2.3.3 .NET Framework Class Library

Conjunto de componentes de software que podem ser utilizados no desenvolvimento de aplicações baseadas no modelo.NET. É importante perceber que todas as linguagens compatíveis com o .NET acessam a mesma biblioteca de classes, além disso, é possível criar novos componentes e adicioná-los a biblioteca de classes existente.

2.3.4 Common Language Runtime (CLR)

Representa o ambiente de execução de aplicações .NET. Fazendo uma analogia com o desenvolvimento de aplicações Java [20] pode-se associar o .NET *Common Language Runtime* com a JVM (*Java Virtual Machine*), pois ambos desempenham funções similares, como por exemplo, o serviço de *garbage collection* (referente à liberação automática de memória).

A grande diferença é que o *Common Language Runtime* suporta a utilização de múltiplas linguagens.

2.3.5 Microsoft ADO.NET

É uma extensão da tecnologia ADO (*ActiveX Data Objects*) da Microsoft utilizada para acesso a dados.

Destaca-se por possuir suporte ao modelo de programação *offline*, isto é, o uso de objetos ADO .NET não exige uma conexão ativa com um banco de dados, o que torna possível o desenvolvimento de aplicações que se conectam temporariamente com uma fonte de dados, recuperam os dados necessários (antes da finalização da conexão) e manipulam os dados localmente já desconectados (*offline*).

2.3.6 ASP.NET

Pode ser visto como um framework de programação para aplicações que seguem o modelo cliente-servidor, tipicamente sistemas web. Com ASP .NET é possível desenvolver utilizando diferentes linguagens como C# (C Sharp) ou Visual basic .NET, por exemplo.

O uso de ASP .NET no desenvolvimento de aplicações móveis é apresentado na Seção 4.

2.3.7 XML Web Services

Em linhas gerais, XML Web Services representam componentes (serviços) que podem ser compartilhados por diferentes aplicações através da internet ou intranet.

A utilização de serviços web remotos por aplicações também é conhecida pela expressão “consumo de Web Services”. O .NET Framework prove ferramentas e classes que podem ser utilizadas para a construção, teste e distribuição de XML Web Services. A Seção 6 explora com mais detalhes o consumo de Web Services por dispositivos móveis.

2.3.8 User Interfaces

Os componentes de interface gráfica presentes no .NET Framework podem ser classificados em três categorias:

- **Web Forms** – interface gráfica utilizada por aplicações desenvolvidas com ASP .NET
- **Windows Forms** – interface para aplicações desktop que executam na plataforma Windows
- **Command Console** – interface de linha de comando

2.3.9 Linguagens

Representa o conjunto de linguagens suportadas pela plataforma .NET. Existe uma especificação, conhecida como CLS (Common Language Specification), que determina quando uma linguagem é compatível com .NET. Na prática, qualquer linguagem que esteja de acordo com a especificação CLS pode ser executada no ambiente de execução do .NET (CLR – Common Language Runtime).

A Microsoft oferece suporte para as linguagens Microsoft Visual Basic® .NET, Microsoft Visual C++® .NET, C#, e Microsoft JScript® .NET. Vale ressaltar que outras linguagens podem ser adicionadas ao conjunto de linguagens suportadas, desde que as mesmas sejam compatíveis com a especificação CLS.

2.4 Desenvolvendo aplicações com .NET Framework

A utilização do .NET Framework, no desenvolvimento de aplicações, envolve algumas decisões que devem ser levadas em consideração. Esta Seção tem por objetivo apresentar as principais decisões que devem ser tomadas na implementação de aplicações baseadas no modelo .NET e fornecer mais detalhes de como funciona o processo de compilação dessas aplicações.

O primeiro passo para se desenvolver com .NET Framework é determinar que tipo de aplicação ou componente será construído. Os principais tipos de aplicações/componentes que podem ser implementados com .NET são:

- **Aplicações Windows** – aplicações desktop que executam na plataforma Windows, como já mencionado na Seção 2.3.8 este tipo de aplicação utilizam como interface gráfica os Windows Forms
- **Biblioteca de Classes** (Classe Library) – componentes que são compartilhados e reusados por outras aplicações
- **Aplicações Web** – aplicações que utilizam o framework ASP .NET
- **Web Service** – serviço web a ser compartilhado e acessado remotamente por outras aplicações
- **Aplicações de Console** – aplicações que não possuem interface gráfica, isto é, são executadas através de linha de comando
- **Serviço Windows** (Windows Service) – representam um serviço do Windows, geralmente possuem um tempo de execução longo e executam em sua própria sessão do Windows

Uma vez selecionado o tipo de aplicação, o próximo passo é escolher a linguagem de programação a ser utilizada. No desenvolvimento típico de aplicações, essa escolha é uma tarefa que apresenta certa dificuldade, pois linguagens diferentes oferecem diferentes características. Como exemplo deste fato, pode-se citar linguagens como C e C++ que permitem ao programador gerenciar memória e implementar aplicações concorrentes com *threads*. Por outro lado, linguagens como Visual Basic e Delphi permitem a construção rápida de aplicações com interface

gráfica e possuem componentes de alto nível que executam, de maneira transparente ao desenvolvedor, tarefas como acesso a um banco de dados.

Com o uso do .NET Framework, o código de qualquer linguagem é compilado para um ambiente de execução único (CLR), sendo assim, as funcionalidades do CLR estão disponíveis para qualquer linguagem compatível com .NET, ou seja, se o ambiente de execução utiliza exceções para reportar erros, então todas as linguagens manipularão erros através de exceções. Caso o ambiente permita a criação de *threads*, então qualquer linguagem poderá criar uma thread. Sendo assim, todas as linguagens compatíveis com .NET possuem praticamente o mesmo poder e podem fazer uso dos mesmos componentes (.NET Class Library) disponibilizados pelo Framework. Isso significa que, com o uso do .NET, a escolha da linguagem poderá ser determinada simplesmente pelo grau de conhecimento ou familiaridade que o programador possui com a sintaxe da mesma, isto é, pela linguagem que o desenvolvedor julgar mais fácil de expressar sua intenção.

Na prática, em tempo de execução, o CLR não “conhece” que linguagem foi utilizada na escrita do código fonte, pois todas as linguagens compatíveis com .NET são compiladas para um código intermediário que utilizam uma linguagem única denominada MSIL (*Microsoft Intermediate Language*).

A Figura a seguir apresenta uma visão geral do processo de compilação e utilizado no .NET.

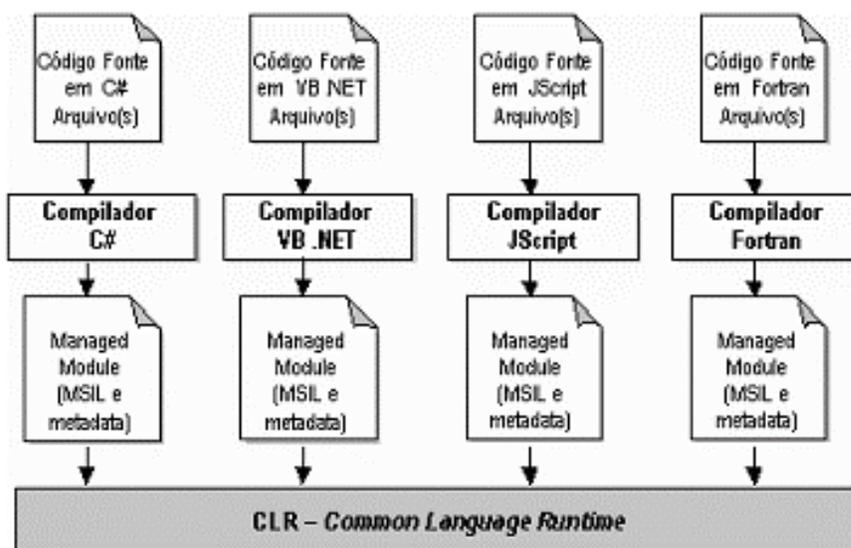


Figura 2 – Processo de compilação de aplicações .NET

Como pode ser visto na Figura apresentada, cada linguagem possui o seu próprio compilador. O processo de compilação de um arquivo ou conjunto de arquivos com código fonte em uma determinada linguagem resultará em um *Managed Module*.

Um *Managed Module* é um arquivo que, além do código em MSIL (*Microsoft Intermediate Language*) gerado pelo processo de compilação, possui um conjunto de meta-informações (*Metadata*) que descreve os tipos definidos e utilizados pelo código fonte.

Em tempo de execução, o CLR compila o código MSIL gerado, sendo este processo conhecido como *Just In-Time Compilation*. A idéia futura é que novos ambientes de execução (CLR) possam ser construídos para outros sistemas operacionais como o Linux, por exemplo, o que tornará o modelo .NET independente de plataforma.

Segundo a Microsoft, a diferença existente entre a execução de códigos em MSIL, gerados pela compilação de linguagens diferentes, é mínima e torna-se insignificante e imperceptível em fatores como, por exemplo, performance.

2.5 Vantagens em usar o .NET Framework

Em linhas gerais os principais benefícios ao se utilizar o .NET Framework são:

- **Independência de Linguagem** – com já mencionado, o modelo .NET suporta um conjunto razoável de linguagens que podem ser utilizadas no desenvolvimento de aplicações.
- **Modularidade** – no .NET Framework o código pode ser organizado hierarquicamente através do uso de *namespaces* ¹ o que permite a construção de códigos mais legíveis e sistemas de fácil manutenção devido a modularidade que pode ser obtida com este recurso.

¹ Equivalem a pacotes que podem ser utilizados para o agrupamento de classes

- **Sistema de tipos único** - o .NET Framework possui um sistema de tipos único que pode ser utilizado por todas as linguagens compatíveis com .NET. Todos os elementos do sistema de tipos são tratados como objetos o que permite a sua utilização em qualquer linguagem suportada pelo modelo.
- **Modelo de aplicação unificado** – as funcionalidades do .NET Framework estão disponíveis para qualquer linguagem ou modelo de programação compatíveis com .NET. Essa característica permite, por exemplo, que um mesmo trecho de código possa ser utilizado por aplicações desktop, Web ou até mesmo por Web services.
- **Suporte aos principais padrões Web** – o desenvolvimento de aplicações Web com .NET é facilitado pelo grande suporte que o mesmo possui aos padrões tecnológicos utilizados atualmente na internet. Como exemplos pode-se citar tecnologias como HTML, XML, SOAP (*Simple Object Access Protocol*), XPath (*XML Path Language*), XSLT (*Extensible Stylesheet Language Transformation*) entre outros padrões. Algumas dessas tecnologias citadas, como SOAP por exemplo, serão vistas com mais detalhes em Seções futuras deste documento.

3. Aplicações e dispositivos móveis

Até o momento, foi apresentado como funciona, de uma maneira geral, o processo de desenvolvimento de aplicações que utilizam o modelo .NET. Foi visto também quais os principais componentes envolvidos na sua plataforma e os tipos de aplicações que podem ser implementadas com o .NET Framework. Porém, nada foi dito com relação ao desenvolvimento de aplicações para dispositivos móveis, pois estas aplicações podem ser vistas como um caso a parte e serão tratadas durante o restante deste documento.

Esta Seção tem como principal objetivo oferecer uma visão geral do desenvolvimento de aplicações para dispositivos móveis utilizando o .NET.

3.1 Dispositivos móveis

Antes de iniciar o estudo de aplicações móveis é importante ter conhecimento sobre os dispositivos em que estas aplicações serão executadas, em termos práticos, isto também implica em conhecer que tipos de dispositivos serão tratados neste trabalho. A seguir, será apresentado um breve histórico que trata a evolução dos dispositivos computacionais móveis, desde o primeiro computador portátil aos dispositivos móveis mais modernos.

3.1.1 Histórico de evolução

A evolução dos dispositivos móveis pode ser vista na seguinte ordem:

3.1.1.1 Lançamento do 1º Computador portátil

O primeiro computador “portátil” foi o Osborne 1, lançado em 1981. Este computador tinha um design bem diferente dos dispositivos móveis que encontramos atualmente, sua tela era capaz de mostrar até 52 caracteres e se destacava por ser o primeiro computador a possuir um conjunto de softwares que incluía o BASIC, WordStar e SuperCalc.

Em 1982, a Compaq lançou o Compaq Portable que só era considerado compacto diante do seu principal concorrente: o IBM-PC. Ele apresentava algumas vantagens sobre o Osborn 1, pois executava o MS-DOS e era 100% compatível com programas escritos para IBM-PC. Muitos outros computadores portáteis surgiram

logo depois, dentre eles destaca-se o Radio Sharck TRS-80, que foi o primeiro computador com tamanho próximo ao de um notebook.



Figura 3 – Osborne 1 o primeiro computador portátil

3.1.1.2 Psion

O gênero dos dispositivos denominados PDA (*Personal Digital Assistant*) surgiu com o lançamento, em 1984, do *Psion Organizer I* pela empresa britânica Psion. O Psion I era baseado na tecnologia de 8 bits, possuía um banco de dados com funções de pesquisa e um pacote utilitário com funções matemáticas, além do relógio/calendário. Com o Psion 1 era possível programar com a linguagem POPL (*Psion Organizer Programming Language*).

O lançamento do Psion II aconteceu pouco tempo depois, com ele foram produzidas cerca de 500.000 unidades entre meados da década de 80 e início dos anos 90.

A terceira série desses dispositivos, lançada em 1993, foi inovadora, pois possuía a capacidade de se comunicar com um PC, com isto era possível transferir, converter e sincronizar dados entre os dois ambientes. O sucesso da Psion despertou interesse em outros grandes fabricantes o que ocasionou um aumento no mercado de PDAs.

3.1.1.3 Expansão do mercado de PDAs

Em agosto de 1993, a Apple lançou o *Newton Message Pad*, que trouxe a inovação da tecnologia de reconhecimento de texto escrito à mão. Com isso o usuário poderia interagir com o dispositivo através de uma “caneta”.

A tecnologia de reconhecimento de escrita (*handwriting recognition*) do *Newton Message Pad* era sofisticada, pois o dispositivo tentava “aprender” a escrita manual de um determinado usuário e convertê-la em texto.



Figura 4 - Newton Message Pad

Porém, a abordagem de reconhecimento automática do *Newton Message Pad* não se mostrou muito viável na época. Até que em 1996 a Palm Computing Inc. criou o seu primeiro PDA, o *Palm Pilot*. Este dispositivo possuía a sua própria “linguagem” de formatos de letra, chamada *Graffiti* que o usuário poderia aprender rapidamente e facilitava o processo de reconhecimento utilizado pelo dispositivo.

A Palm Computing se tornou líder de venda, em 1999 dominava cerca de 70% do mercado de PDAs e o número de desenvolvedores de aplicações para Palm cresceu significativamente.

3.1.1.4 Dispositivos Atuais

Em 1996 a Microsoft lançou seu primeiro sistema operacional para dispositivos móveis, o Windows CE (Compact Edition). Porém, as duas primeiras versões do Windows CE não tiveram muito sucesso, pois os dispositivos existentes na época ainda não eram adequados para suportar a interface gráfica proposta por este sistema. O lançamento do Pocket PC 2000, que é uma implementação específica da versão 3 do Windows CE, foi feito em abril de 2000 e passou a suportar um novo layout de interface gráfica, melhor que o utilizado nas versões anteriores do Windows CE.

Outras atualizações do sistema operacional Windows CE surgiram depois, tais como o Pocket PC 2002 (lançado no final de 2001) e o Windows CE .NET oficialmente lançado em janeiro de 2002. A família Pocket PC de sistemas operacionais representa uma versão especializada do Windows CE para dispositivos PIM (*Personal Information Management*), já o Windows CE .NET é uma plataforma mais genérica, suportada em uma vasta quantidade de dispositivos.

O dispositivo Pocket PC que se tornou rapidamente o maior concorrente do Palm foi o Compaq iPad, a Figura abaixo apresenta alguns dispositivos que utilizam sistemas derivados do Windows CE.



Figura 5 – Dispositivos móveis mais recentes

3.1.2 Considerações sobre o trabalho

Neste trabalho serão considerados os dispositivos móveis que utilizam a família Pocket PC de sistema operacional ou o Windows CE .NET. Além disso, serão abordados também os dispositivos que suportam acesso a aplicações web móveis, o que inclui alguns telefones celulares que utilizam browsers para acesso a internet.

Para a execução dos exemplos e realização de testes foram utilizados os emuladores existentes no próprio ambiente de desenvolvimento, que possui emuladores para as duas plataformas citadas (Pocket PC e Windows CE), e os simuladores da Openwave [18].

3.2 Aplicações móveis e .NET

No desenvolvimento de aplicações para dispositivos móveis com .NET, uma das características mais significativas para o desenvolvedor é o fato das aplicações .NET executarem sob o controle do CLR (ambiente de execução visto na Seção 2.3). Isto porque o CLR é o responsável por efetuar operações como o gerenciamento de memória, o que fornece ao desenvolvedor uma maior abstração.

No caso específico de memória, o CLR efetuará a alocação quando novos objetos forem criados e a liberação automática de memória (*garbage collection*) quando a mesma não for mais necessária. Essas operações (alocação/liberação) são totalmente efetuadas sem a influência do programador, o que representa um

grande ganho, pois estamos lidando com um ambiente onde os recursos são bastante limitados e que geralmente, no desenvolvimento tradicional de aplicações móveis, demanda um maior cuidado por parte do desenvolvedor. Estas e outras características do .NET o tornam uma boa opção de plataforma para o desenvolvimento de aplicações móveis.

Porém, outros fatores precisam ser levados em consideração no desenvolvimento de aplicações para dispositivos móveis utilizando o modelo .NET. Dentre estes fatores destacam-se:

- O Ambiente de Desenvolvimento
- Abordagens (arquiteturas) possíveis
- Ferramentas de apoio necessárias

A seguir temos a descrição de cada um dos fatores citados e algumas considerações sobre o que foi abordado neste trabalho.

3.2.1 Ambiente de Desenvolvimento

Na prática, é possível desenvolver aplicações .NET com um simples editor de texto, porém, para um desenvolvimento mais rápido, é fortemente aconselhável a utilização de um ambiente mais sofisticado. Atualmente o ambiente de desenvolvimento mais indicado e completo é o Visual Studio .NET.

O Visual Studio .NET é um ambiente de desenvolvimento que permite a implementação, depuração (*debugging*), execução e *deploy* de aplicações .NET. Pode ser utilizado tanto para a construção de aplicações *desktop* (aplicações para PCs) quanto para aplicações móveis (voltadas para dispositivos móveis), além disso suporta também o desenvolvimento de *Web Services* (abordado na Seção 6).

O Visual Studio .NET 2003 é a versão mais recente deste ambiente, como ele é possível desenvolver e integrar aplicações e/ou componentes escritos em diferentes linguagens, como, C#, Visual Basic .NET, J# e C++.

Os protótipos de aplicações desenvolvidos neste trabalho foram implementados com o Visual Studio .NET 2003.

3.2.2 Abordagens (Arquiteturas)

O desenvolvimento de aplicações para dispositivos móveis com .NET apresenta as seguintes abordagens:

- **Aplicações cliente** (client-side) – as aplicações cliente executam no próprio dispositivo e podem fazer uso de características (capacidades) específicas do mesmo, geralmente são aplicações mais difíceis de serem executadas em uma grande variedade de dispositivos e permitem a construção de interfaces gráficas ricas.
- **Aplicações web com ASP .NET** (server-side) – ao contrário das aplicações cliente, as aplicações web móveis executam no servidor e são “renderizadas” em um browser presente no dispositivo móvel. Com .NET é mais fácil desenvolver aplicações deste tipo que sejam compatíveis com uma grande variedade de dispositivos.
- **Aplicações baseadas em Web Services** (client-server) – representam serviços ou componentes que podem ser acessados remotamente através da internet. Em termos práticos, a utilização de um serviço remoto, também conhecida como consumo de Web Services, pode ser realizada tanto por aplicações móveis client-side quanto por aplicações web. Estes dois contextos serão abordados na Seção 6.

Para cada uma das abordagens apresentadas foi dedicada uma seção neste documento, onde serão detalhadas as principais características presentes em cada uma delas através da implementação e testes de protótipos de aplicações.

3.2.3 Ferramentas

Dependendo do tipo de abordagem a ser adotada será preciso utilizar um conjunto específico de ferramentas.

O desenvolvimento de aplicações cliente utiliza como infra-estrutura o .NET Compact Framework (.NET CF), que é uma versão simplificada do .NET Framework. A versão 2003 do Visual Studio .NET está integrada com o .NET CF e permite a implementação e emulação de aplicações para Pocket PC e dispositivos baseados no sistema operacional Windows CE .NET.

Já o desenvolvimento de aplicações web móveis utiliza o conjunto de ferramentas chamado Mobile Internet Toolkit, que possui os componentes necessários para a implementação de sistemas web móveis. Este conjunto de ferramentas também já está integrado com o Visual Studio .NET 2003.

A Figura a seguir apresenta uma visão geral da plataforma de desenvolvimento utilizada na implementação de aplicações para dispositivos móveis. Na figura pode-se perceber mais claramente a relação existente entre os elementos apresentados neste seção.

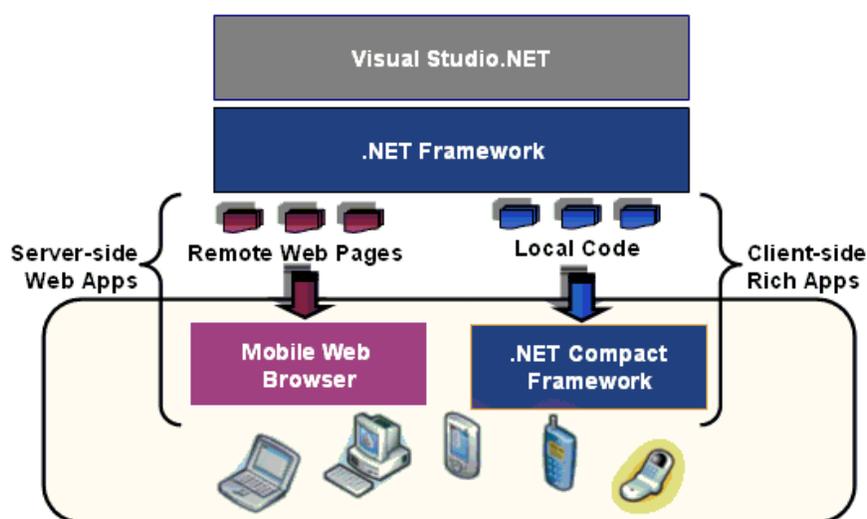


Figura 6 – Plataforma de desenvolvimento para dispositivos móveis

Como pode ser visto, cada abordagem possui propósitos específicos, logo, o cenário de desenvolvimento determinará que abordagem será mais apropriada. As seções a seguir apresentam com mais detalhes as características encontradas no desenvolvimento baseado em cada uma das abordagens.

4. Aplicações *Client-side*

Esta seção tem por objetivo apresentar as principais características encontradas no desenvolvimento de aplicações cliente para dispositivos móveis. Será apresentada uma visão introdutória a respeito da infra-estrutura de desenvolvimento, chamada .NET Compact Framework (.NET CF).

Além disso, serão mostrados alguns protótipos de aplicações, desenvolvidos durante o trabalho. Os protótipos apresentados visam analisar mais concretamente as principais características analisadas.

4.1 .NET Compact Framework

4.1.1 O que é?

O .NET Compact Framework é um subconjunto do .NET Framework (visto na Seção 2.2) especialmente desenvolvido para implementação de aplicações cliente em dispositivos móveis. O .NET Compact Framework possui uma nova implementação da CLR (*Common Language Runtime*) que foi modificada para suportar, de maneira mais eficiente, a execução de aplicações no contexto de pequenos dispositivos, isto é, com memória, poder de CPU e bateria limitados.

Por ser um subconjunto do .NET Framework, os desenvolvedores podem facilmente reusar a maioria dos conceitos e conhecimentos adquiridos na implementação de aplicações desktop, contando praticamente com a mesma plataforma de desenvolvimento.

4.1.2 Plataformas suportadas

A primeira versão do .NET Compact Framework está voltada para dispositivos Pocket PC 2000 e 2002 e para dispositivos que utilizam a versão 4.1 ou superior do Windows CE .NET. A seguir tem-se a lista das plataformas Microsoft suportadas pelo .Net Compact Framework.

4.1.2.1 Pocket PC 2000

Estes dispositivos podem usar processadores ARM, MIPS, SH3. O .NET Compact Framework é suportado em todos esses processadores.

4.1.2.2 Pocket PC 2002

É uma atualização do sistema operacional Pocket PC. O Pocket PC 2002 foi versionado apenas para processadores ARM.

4.1.2.3 Pocket PC 2002 Phone Edition

Equivale ao sistema operacional Pocket PC 2002 acrescido de algumas funcionalidades específicas de aparelhos telefônicos.

4.1.2.4 Pocket PC .NET

É a nova atualização do sistema operacional Pocket PC, foi construído com base no kernel do Windows CE .NET.

4.1.2.5 Windows CE .NET

Versão mais recente do sistema operacional Windows CE.

4.1.3 Biblioteca de classes

Sob o ponto de vista do desenvolvedor, a parte mais interessante do .NET Compact Framework é a biblioteca de classes, pois elas fornecem uma abstração dos serviços do sistema operacional, permitindo ao desenvolvedor adicionar funcionalidades aos programas de maneira consistente e independente da linguagem ou do sistema operacional em questão.

4.1.3.1 Estrutura

A biblioteca de classes do .NET Compact Framework possui menos de 50% das classes da versão completa do .NET Framework, mesmo assim, as funcionalidades do .NET CF não são limitadas se comparadas às da versão completa do .NET Framework. Isso se deve ao fato do .NET CF possuir todas as classes básicas do .NET Framework o que, em termos práticos, é suficiente para o desenvolvimento da maioria das aplicações.

Em linhas gerais, o layout básico da biblioteca de classes do .NET CF é similar ao layout encontrado no .NET Framework, com a diferença que alguns componentes Web não estão presentes na estrutura do .NET CF

A Figura a seguir apresenta a estrutura da biblioteca de classes do .NET Compact Framework.

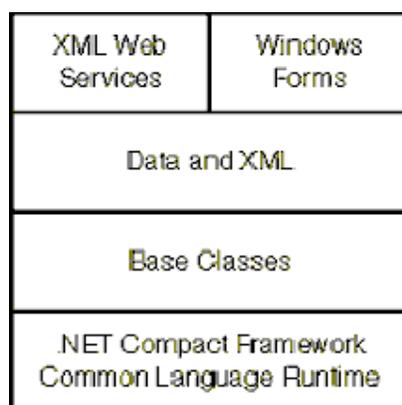


Figura 7 – Layout da Biblioteca de Classes do .NET CF

4.1.3.1.1 Base

Contém os tipos de dados, classes de coleções (arrays, filas, hash tables), classes que efetuam operações de entrada e saída, classes que dão suporte a segurança e globalização de aplicações.

4.1.3.1.2 Data and XML

Contém as classes do ADO .NET (visto na Seção 2.3.5) e classes voltadas para a formatação e *parsing* de dados em arquivos XML.

4.1.3.1.3 XML Web services

Contém classes que permitem a criação de clientes para Web Services.

4.1.3.1.4 Windows Forms

Possui as classes necessárias para a construção de aplicações com interface gráfica (GUI).

Algumas classes da biblioteca do .NET Framework foram excluídas da versão compacta pelas seguintes razões:

- algumas delas utilizam serviços do Windows que não existem no Windows CE. Como exemplo, tem-se as classes do ASP .NET que utilizam o IIS (Internet Information Server).
- outras implementam funcionalidades que não fazem sentido em uma aplicação presente em um dispositivo móvel, como por exemplo, impressão
- Por último, temos as aplicações que são computacionalmente custosas de se implementar e isso vai contra um dos objetivos do .NET Compact Framework que é minimizar e limitar o máximo possível a demanda pela CPU. Por essa razão, o .NET Compact Framework exclui funcionalidades como XML Path Language (XPath) e Extensible Stylesheet Language Transformations (XSLT) presentes na versão completa do .NET Framework.

4.1.4 Benefícios

Atualmente, uma das grandes barreiras na implementação de aplicações para dispositivos móveis é que muitos dispositivos exigem que os desenvolvedores aprendam diferentes APIs e usem diferentes ferramentas de programação, que geralmente diferem das utilizadas no desenvolvimento de aplicações tradicionais (para PCs).

O fato do .NET CF utilizar a mesma ferramenta de desenvolvimento (Visual Studio .NET) e o mesmo modelo de programação utilizados no .NET Framework é um dos grandes benefícios do uso dessa tecnologia, pois reduz significativamente o custo de desenvolvimento dessas aplicações e acarreta em um aumento de produtividade considerável.

4.2 Principais Características

As principais características presentes nas aplicações desenvolvidas com o .NET Compact Framework são:

4.2.1 Aplicações executam no dispositivo

As aplicações desenvolvidas com o .NET CF são por natureza aplicações *stand-alone* que só podem ser executadas no dispositivo móvel para o qual foram desenvolvidas.

4.2.2 Aplicações suportadas por poucos dispositivos

Geralmente as aplicações *client-side* usufruem as capacidades específicas do dispositivo em que a mesma está executando, sendo assim, na maioria das vezes temos aplicações que não são compatíveis ou não podem ser utilizadas com uma grande quantidade de diferentes dispositivos.

4.2.3 Rica interface gráfica

O .NET Compact Framework possui uma variedade muito grande de componentes visuais que podem ser utilizados na construção de aplicações *client-side*. Como exemplos desses componentes pode-se citar: botões, campos de texto, labels, calendários, imagens, entre outros. Além disso, também é possível fazer uso de funcionalidades que permitem a implementação de interfaces mais sofisticadas, como por exemplo, a construção de telas com animação.

A maioria dos protótipos (*client-side*) implementados neste trabalho teve como objetivo a construção de aplicações que utilizam recursos ou componentes visuais disponíveis no .NET Compact Framework.

4.2.4 Opções de trabalho off-line

É possível desenvolver aplicações que processam dados localmente sem a necessidade de uma conexão ativa. Na realidade, a aplicação só estabelece uma conexão para recuperar os dados e permanece desconectada durante o

processamento dos dados, o que só é possível graças ao uso de componentes ADO .NET.

Esta característica é bastante útil no desenvolvimento de aplicações onde o tempo de conexão é um fator crítico e onde conexões podem não estar disponíveis a qualquer momento, como é o caso de aplicações para dispositivos móveis. Outra característica importante do ADO .NET é alto nível de suporte à manipulação de arquivos XML, esta característica será detalhada mais adiante.

4.3 Protótipos desenvolvidos

A seguir, são apresentados alguns protótipos de aplicações que foram desenvolvidas durante o trabalho. Estes protótipos serviram para verificar algumas das características apresentadas na Seção anterior, além de possibilitar uma melhor análise do ambiente de desenvolvimento e modelo de programação utilizados.

Para a implementação dos protótipos foi utilizado o Visual Studio .NET 2003 como ambiente de desenvolvimento (IDE) e o .NET Compact Framework (.NET CF). Como já mencionado, o uso do .NET CF está voltado para dispositivos que utilizam os sistemas operacionais da família Pocket PC ou Windows CE, logo, os exemplos foram desenvolvidos para serem executados nestas plataformas. Para a execução dos exemplos e realização de testes foram utilizados os emuladores existentes no próprio ambiente de desenvolvimento, que possui emuladores para as duas plataformas citadas (Pocket PC e Windows CE). A linguagem de programação utilizada nos protótipos foi C#.

Por questões de legibilidade, algumas listagens de código aparecem com trechos “//...” representam código oculto e que não influenciará no entendimento da lógica.

As Seções a seguir apresentam mais detalhes a respeito das aplicações implementadas e fornece uma conclusão com comentários sobre cada uma delas.

4.3.1 HelloWorld

A primeira aplicação desenvolvida foi um simples programa HelloWorld, que serviu apenas para se familiarizar com o ambiente de desenvolvimento utilizado e verificar as opções de interface gráfica existentes.

O Visual Studio .NET possui um ambiente de design onde o layout das aplicações gráficas para dispositivos móveis pode ser criado facilmente. A Figura abaixo apresenta um formulário que representa a tela da aplicação sendo desenvolvida.

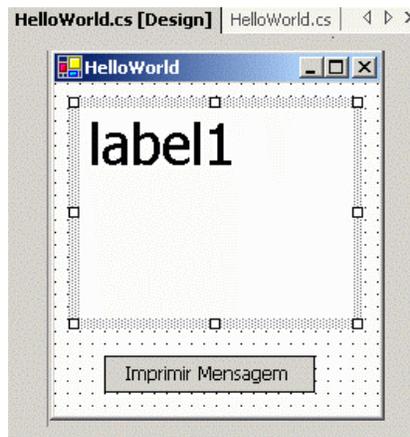


Figura 8 – Desing no Visual Studio do protótipo HelloWorld

Como pode ser visto na Figura, existem elementos de interface gráfica que podem ser utilizados nas aplicações móveis *client-side*, esses elementos possuem propriedades e eventos que tratam a interação com o usuário. No exemplo, foi criado um arquivo chamado HelloWorld.cs, que contém código em linguagem C# referente a aplicação, na Figura acima temos a visualização gráfica do arquivo. A Estrutura do código fonte da aplicação está mostrado na listagem a seguir:

```
public class HelloWorld : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.MainMenu mainMenu1;

    // ...
    private void button1_Click(object sender, System.EventArgs e)
    {
        // seta o texto no label...
        label1.Text = "Alo Mundo!!";
    }

    // ...
}
```

Figura 9 - Trecho de código do protótipo HelloWorld.cs

Como pode ser observado, os objetos de interface gráfica se encontram no *namespace* `System.Windows.Forms`. No exemplo, o método `button1_Click` é o responsável por tratar o evento de clique no botão e sua implementação simplesmente modifica a propriedade `Text` do objeto `Label` presente na aplicação.

A seguir temos o resultado da aplicação executando no emulador para Pocket PC:



Figura 10 – HelloWorld executando no emulador para Pocket PC

4.3.1.1 Conclusão

Embora simples, o desenvolvimento deste exemplo permitiu concluir de imediato que:

- O desenvolvimento de aplicações com o .NET Compact Framework utilizando o Visual Studio .NET é realmente muito similar com o de aplicações Windows desktop, onde se tem um formulário representando a janela da aplicação e elementos de interface gráfica são adicionados ao mesmo
- As opções de controles (componentes de representação visual) são muitas e tem-se componentes que variam desde um simples `Label` até

componentes mais sofisticados como Calendário e DataGrid (para a apresentação de dados em tabelas)

- O suporte a linguagens orientadas a objetos permite o desenvolvimento de aplicações mais legíveis e fáceis de manter
- O ambiente mostrou um bom suporte a testes e uma ótima integração com os emuladores existentes

4.3.2 Animação gráfica

Este protótipo foi baseado no exemplo mostrado no artigo intitulado *Creating a Microsoft .NET Compact Framework-based Animation Control* [16], que ensina como criar uma animação através de uma imagem (formato GIF86a) utilizando os recursos existentes no .NET Compact Framework (.NET CF), pois este não incorpora a classe `ImageAnimator` presente na versão completa do .NET Framework.

O artigo citado ensina como construir um objeto de controle (denominado `AnimateCtl`), que possui como principal funcionalidade o desenho em seqüência de partes (quadros) de uma imagem na tela. Neste contexto, um objeto de controle equivale a um componente de representação visual que pode ser reutilizado por outras aplicações. Em termos de implementação, os objetos de controle devem herdar da classe `System.Windows.Forms.Control`, suportada no .NET CF.

Na implementação do `AnimateCtl` foram utilizadas classes que se encontram nos *namespaces* `System.Drawing` e `System.Drawing.Imaging`, como exemplo dessas classes temos `Bitmap` e `Graphics` responsáveis respectivamente pela leitura de arquivos de imagem e desenho de imagens na tela. A seguir tem-se um trecho de código extraído da classe `AnimateCtl`, responsável direto pelo desenho da imagem na tela.



Ao método `Draw`, apenas foi adicionado o código para o cálculo da posição, na imagem, onde começa o frame a ser desenhado (variável `XLocation`), pois a funcionalidade de desenhar porções de uma imagem já é oferecida pelo .NET CF através do método `DrawImage` do objeto `Graphics`. Na prática, a variável `XLocation` serve como um apontador que percorre a imagem a ser desenhada.

Sendo assim, pode-se concluir que a imagem utilizada deve possuir uma série de quadros em seqüência. No protótipo foi utilizada a seguinte seqüência de imagem:



Figura 11- Figura com seqüência de frames para animação (disquete.gif)

A animação é obtida com sucessivas chamadas ao método `Draw` que, a cada intervalo de tempo estabelecido, desenha em seqüência os quadros da imagem. O resultado do protótipo é a aplicação mostrada na Figura a seguir:



Figura 12 – Aplicação com animação executando no emulador

Funcionamento: ao clicar no botão “Iniciar animação” a imagem presente na tela será animada, na realidade, um frame da imagem será mostrado a cada intervalo de tempo. O trecho de código a seguir apresenta o uso do controle `AnimateCtl` na classe (do tipo `Windows Form`) que representa a janela da aplicação mostrada na Figura anterior.

```

public class FormAnimacao : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button button1;
    private AnimateCtl animCtl;
    private System.Windows.Forms.Panel panel1;

    public FormAnimacao ()
    {
        InitializeComponent();

        // Cria uma instância do objeto AnimateCtl
        animCtl = new AnimateCtl();
        //Atribui o arquivo de imagem
        animCtl.Bitmap = new Bitmap(@"\My Documents\disquete.gif");
        //Seta a posição
        animCtl.Location = new Point(50, 50);
        //Adiciona o controle ao Form
        this.Controls.Add(animCtl);
    }
    private void button1_Click(object sender, System.EventArgs e)
    {
        //Inicia a animação da imagem
        animCtl.StartAnimation(90, 100, 3);
    }
    ...
}

```

Figura 12 – Trecho de Código da classe responsável por executar animações

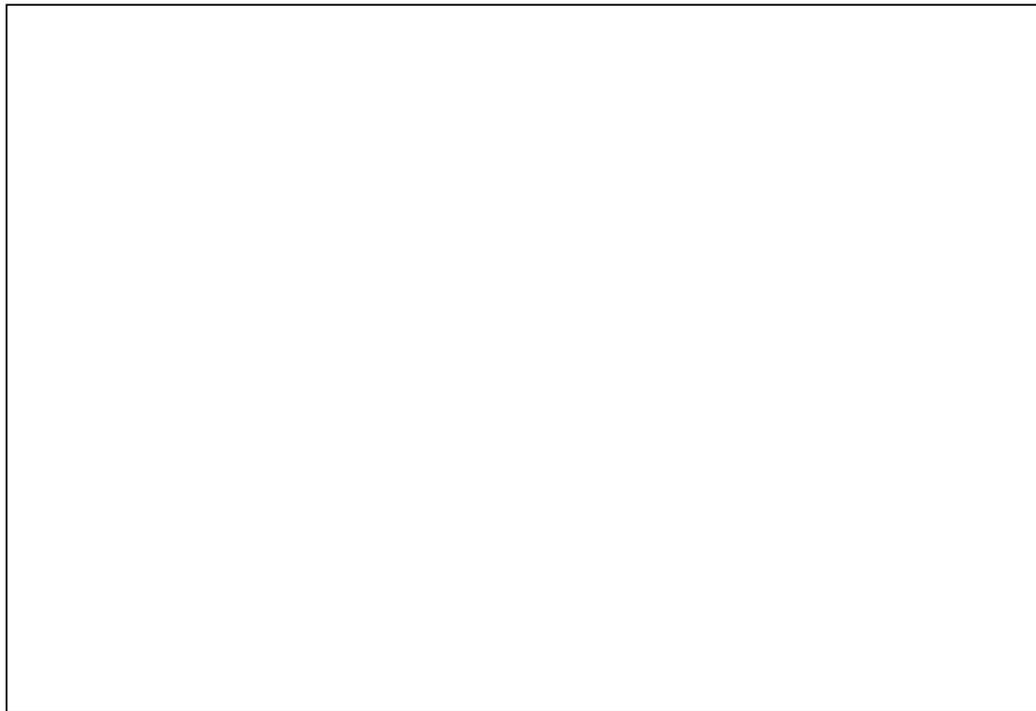
Note que o uso do controle (`AnimateCtl`) é simples e o método de ação do botão (`button1_Click`) faz apenas uma chamada ao método `StartAnimation` do controle e passa como parâmetros valores que representam respectivamente: a largura do frame (90 pixels), o intervalo entre a animação de cada quadro (100 ms) e o número de iterações a serem utilizadas (3) que corresponde a quantidade de vezes que a seqüência completa de quadros será mostrada.

4.3.2.1 Conclusão

Com o protótipo apresentado, pode-se concluir que o uso do .NET Compact Framework permite a construção de interfaces gráficas ricas. No protótipo mostrado, por exemplo, foi possível simular uma animação, que é um recurso bastante útil quando se deseja construir interfaces gráficas mais atraentes. É importante reforçar também que a aplicação final obtida não apresenta grande complexidade, pois o .NET CF já oferece algumas funcionalidades que ajudaram no desenvolvimento deste exemplo.

4.3.3 Banco de dados em memória

Neste protótipo foi desenvolvida uma aplicação que utiliza os objetos existentes no modelo ADO .NET. No exemplo implementado uma base de dados é criada em memória e utilizada para preencher o conteúdo de um componente de interface gráfica, neste caso, um objeto do tipo `System.Windows.Forms.DataGrid` que faz parte do conjunto de controlos visuais do .NET Compact Framework. O trecho de código a seguir mostra a estrutura da classe (`DataGridForm.cs`) criada.



Código 2 – DataGridForm.cs

No código apresentado, da classe `DataGridForm`, a inicialização dos objetos ADO .NET (neste caso `DataSet` e `DataTable`) é realizada pelo método `criarBD` mostrado a seguir:

```

//...
private void criarBD()
{
    // Cria um DataSet.
    dataSet = new DataSet("BDEstoque");

    // Cria uma DataTable
    DataTable tProd = new DataTable("Produtos");

    // Adiciona três colunas a tabela criada.
    DataColumn cProdID = new DataColumn("ProdID", typeof(int));
    DataColumn cProdDesc = new DataColumn("ProdDesc");
    DataColumn cExiste = new DataColumn("Existe", typeof(bool));
    tProd.Columns.Add(cProdID);
    tProd.Columns.Add(cProdDesc);
    tProd.Columns.Add(cExiste);

    // Adiciona a tabela ao DataSet
    dataSet.Tables.Add(tProd);

    // Cria três Produtos na tabela de Produtos
    DataRow novaLinha;
    for(int i = 1; i < 4; i++)
    {
        novaLinha = tProd.NewRow();
        novaLinha["ProdID"] = i;
        // Adiciona a linha na tabela de Produtos
        tProd.Rows.Add(novaLinha);
    }
    // Fornece uma descrição para cada produto criado
    tProd.Rows[0]["ProdDesc"] = "Teclado";
    // ...

    // Informa se o produto existe no estoque
    tProd.Rows[0]["Existe"] = true;
    // ...
}

```

Código 3 – Método que cria uma base de dados em memória

É importante reforçar que, neste caso, a criação do banco de dados foi totalmente feita em memória. Porém, um objeto `DataSet` pode ser populado através de fontes de dados reais como um banco de dados, ou até mesmo um arquivo XML.

É essa característica que possibilita o trabalho *off-line* de um dispositivo móvel, de modo que a aplicação pode se conectar temporariamente com uma fonte de dados, recuperar os dados necessários e, já desconectada, manipulá-los localmente com o objeto `DataSet`. O protótipo seguinte apresenta como preencher um `DataSet` a partir de um arquivo XML.

Veja, na Figura a seguir, o resultado do uso do `DataGrid` em dispositivos móveis, mostrado neste exemplo.

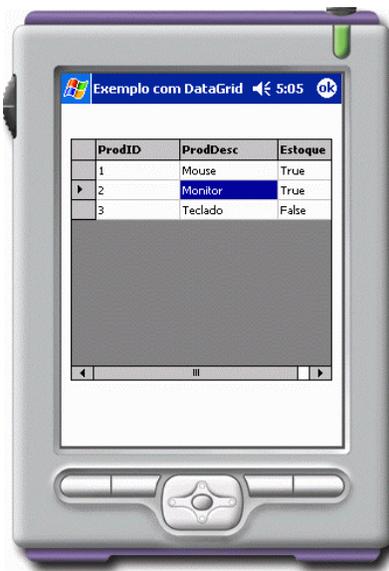


Figura 13 – Aplicação DataGridForm.cs executando no emulador

4.3.3.1 Conclusão

Com este protótipo foi possível perceber que a manipulação dos objetos ADO .NET em aplicações móveis é realizada da mesma maneira que a utilizada em aplicações windows desktop. Além disso, permitiu analisar como seria o trabalho *off-line* de dispositivos móveis que necessitam acessar dados de uma base de dados remota.

4.3.4 Uso de arquivos XML

Este exemplo implementa uma árvore de opções organizadas hierarquicamente (*TreeView*). Os elementos (opções) da árvore são obtidos a partir dos dados de um arquivo XML, o qual é carregado localmente em memória. Veja a seguir o formato do arquivo XML utilizado:

```

<?xml version="1.0" encoding="utf-8" ?>
<cin>
  <menu>
    <menuID>0</menuID>
    <nomeMenu>Servicos</nomeMenu>
    <opcao>
      <id>0</id>
      <menuID>0</menuID>
      <name>Reserva de Salas</name>
    </opcao>
    <opcao>
      <id>1</id>
      <menuID>0</menuID>
      <name>Reserva de laboratorios</name>
    </opcao>
    <opcao>
      <id>2</id>
      <menuID>0</menuID>
      <name>Solicitacao de Cracha</name>
    </opcao>
  </menu>
  ... outros menus
</cin>

```

Figura 14 - XML contendo dados que serão carregados em memória (MenuCin.xml)

O método, na classe Windows Form, responsável por carregar os dados do XML está mostrado abaixo:

```

private void Form1_Load(object sender, System.EventArgs e)
{
    // Cria um DataSet a partir de um arquivo XML
    DataSet ds = new DataSet();
    string path = Assembly.GetExecutingAssembly().GetName().CodeBase;
    ds.ReadXml(new FileInfo(path).DirectoryName + @"\MenuCin.xml");

    // Adiciona nos na raiz para cada menu na tabela de menus
    ListDictionary menuNodeLookup = new ListDictionary();
    foreach(DataRow row in ds.Tables["menu"].Rows) {
        // ...
    }

    // Adiciona opções de menu em cada nó de menu
    foreach(DataRow row in ds.Tables["opcao"].Rows) {
        // ...
    }

    treeView1.CheckBoxes = true;
    treeView1.ShowRootLines = true;
}
// ...

```

Figura 15 – Criando DataSet a partir de um arquivo XML

Resultado da Aplicação:



Figura 16 – Aplicação com TreeView

4.3.4.1 Conclusão

O exemplo apresentado serviu para comprovar quão fácil é obter dados a partir de arquivos XML.

5. Aplicações Web Móveis

Esta Seção tem como objetivo tratar o segundo tipo de aplicações que podem ser desenvolvidas para dispositivos móveis. Neste contexto, deve-se entender aplicações web móveis como sendo as aplicações para dispositivos móveis implementadas com ASP .NET em conjunto com o MIT (Mobile Internet Toolkit), ou seja, são aplicações web ou *web sites* desenvolvidos especialmente para serem apresentados em browsers de dispositivos móveis.

5.1 ASP .NET e Mobile Internet Toolkit

Nesta Seção serão exploradas as funcionalidades do ASP .NET, que é um dos componentes contidos no .NET Framework, explicado na Seção 2. Será examinado também as características do Mobile Internet Toolkit, que permite o suporte de ASP .NET para o desenvolvimento de aplicações voltadas para uma grande variedade de clientes móveis.

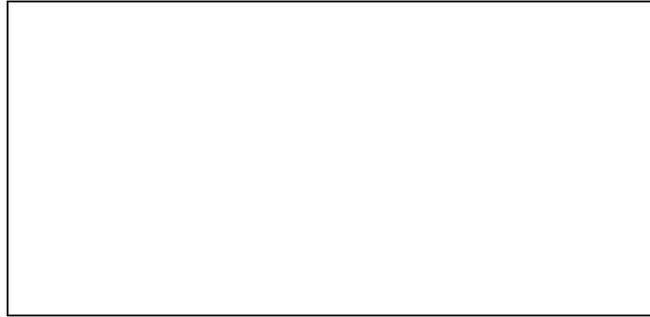
5.1.1 Visão Geral de ASP .NET

Podemos definir ASP .NET como sendo um framework de aplicação que fornece todo o suporte necessário para a construção de aplicações Web. Em termos práticos, é uma nova abordagem de desenvolvimento de sistemas Web baseado no modelo .NET.

O ASP .NET é derivado da tecnologia ASP, bastante utilizada na construção de Web Sites.

5.1.1.1 ASP

Uma aplicação em ASP consiste basicamente de arquivos escritos em uma linguagem de marcação de texto, como HTML ou WML, que possui seções especiais de código (scripts) delimitadas por tags `<%...%>`, de acordo com o mostrado no trecho de código a seguir.



Código 3 - Aplicação ASP HelloWorld

No exemplo, tem-se o código de uma página com extensão `.asp` (HelloWorld.asp), que possui uma seção simples de script (delimitada por `<%...%>`). Com ASP, as seções de script podem conter código em linguagens como JScript, VBScript (*Visual Basic Scripting Edition*), ou JavaScript.

5.1.1.2 ASP .NET

O ASP .NET é uma extensão da tecnologia ASP, porém oferece mais recursos que o seu predecessor, pois faz parte do .NET Framework.

Como características adicionais pode-se citar:

- Desenvolvimento de aplicações utilizando qualquer linguagem de programação compatível com .NET
- Código da aplicação compilado, que executa mais rápido que o código interpretado (abordagem utilizada em ASP)
- Suporta operações sem o uso de *cookies*
- Oferece acesso a dados através da utilização do modelo de objetos de dados do .NET (ADO .NET)
- Prove acesso, através da biblioteca de classes do .NET (.NET Class Library), a recursos como, por exemplo, serviços do sistema operacional

- Permite o gerenciamento de estado mais fácil, através de elementos chamados Web Forms

Uma aplicação Web desenvolvida com ASP .NET possui diferentes partes e componentes, de acordo com o apresentado na Figura a seguir.

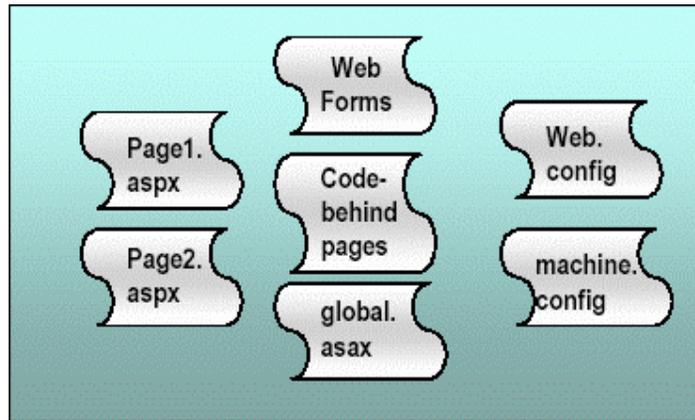


Figura 17 – Componentes do ASP .NET

5.1.1.2.1 Web Forms ou páginas .aspx

Componentes responsáveis pela interface gráfica com o usuário (GUI). Um Web Form é um componente visual que representa uma página ASP .NET (.aspx), ele serve como um *container* para os outros elementos de interface gráfica, chamados controles (*Web Controls*).

5.1.1.2.2 Web Controls

Como já mencionado, os web controls correspondem aos elementos de interface gráfica que podem ser utilizados em uma página ASP .NET (Web Form). Pode-se citar como exemplos de web controls, elementos visuais como caixa de texto, comboBox, Labels, entre outros.

5.1.1.2.3 Code-behind pages

Com os Web Forms do ASP .NET é possível separar os componentes da interface da lógica associada aos mesmos. Isto é possível através da associação de um Web Form com um arquivo, que contem o código a ser executado no servidor. O arquivo que contem a lógica de programação é chamado *code-behind page* e pode

ser escrito em linguagens de programação mais sofisticadas como C#, ao invés de linguagens de script com as usadas em ASP.

5.1.1.2.4 Arquivos de configuração

São arquivos XML que definem as configurações padrão a serem utilizadas na aplicação e no servidor Web. Toda aplicação Web possui um arquivo chamado *Web.config* e cada servidor web tem um arquivo chamado *Machine.config*.

5.1.1.2.5 Arquivo Global.asax

Arquivo que contém o código necessário responsável por tratar eventos lançados por uma aplicação ASP .NET. Vale reforçar que esses eventos são gerais (em nível de aplicação) e não dizem respeito à uma página específica.

5.1.1.3 Mobile Internet Toolkit

Análogo ao .NET Compact Framework, que é a tecnologia .NET para a construção de aplicações *client-side*, o Mobile Internet Toolkit é a tecnologia .NET para o desenvolvimento de aplicações Web para dispositivos móveis, isto é, aplicações processadas no servidor e que executam no browser de um dispositivo móvel.

O Mobile Internet Toolkit (MIT) consiste de três componentes:

- **Mobile controls** – correspondem aos controles (Web Controls) do ASP .NET, porém foram desenvolvidos para dispositivos móveis. Como exemplo desses dispositivos tem-se: botões, labels, links, etc.
- **Mobile Web Form** – correspondem às páginas .aspx para browsers de dispositivos móveis. Um Mobile Web Form serve como um *container* para os Mobile Controls
- **Mecanismo de identificação de dispositivo** – componente que inclui o código para identificação do dispositivo que está solicitando a página ASP .NET.

5.2 Funcionamento de aplicações Web móveis

Esta Seção apresenta como funciona o processo de requisições de uma página web realizado por um dispositivo móvel.

Primeiramente o dispositivo móvel executa uma requisição HTTP ao servidor web que contém o Mobile Internet Toolkit e a requisição HTTP é processada no servidor em três principais estágios.

O primeiro estágio corresponde a identificação do dispositivo que realizou a requisição e a capacidade que o mesmo possui. Como exemplos de informações levantadas neste estágio, temos o tipo de browser que o dispositivo possui, a linguagem de formatação utilizada (*markup language*), e as capacidades de apresentação de imagem que o dispositivo possui. O responsável pela realização desse passo é o mecanismo identificador de dispositivos do MIT.

A requisição http possui três partes: *user Agent string*, *header information* e URL. O processo de identificação do dispositivo utiliza a *user agent string* para descobrir informações a respeito do dispositivo que está solicitando a página. Após isso, a URL da requisição é usada para localizar a página Web que foi solicitada (arquivo com extensão .aspx).

A página .aspx passa por um processo de compilação quando é acessada pela primeira vez. O processo de compilação consiste em enviar a página para um parser cujo resultado é, em seguida, processado por um compilador. A página compilada é então armazenada em uma cache (*assembly cache*) e o servidor por sua vez cria uma nova instância da página compilada e a usa para processar o request. Vale reforçar que os processos de *parsing* e compilação não são mais repetidos a partir da segunda requisição para uma página já compilada. Neste caso, o resultado da compilação, que se encontra em cache, será reusado o que aumenta significativamente a performance das próximas requisições.

A Figura a seguir ilustra o processo de compilação de uma página ASP .NET.

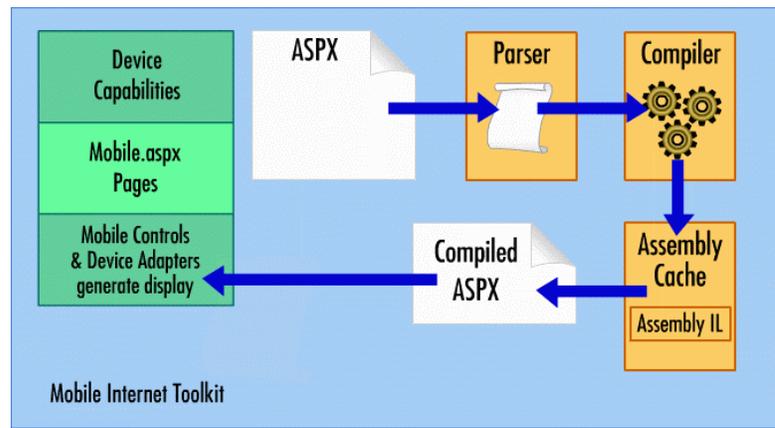


Figura 18 – Compilação de página ASPX

O processo de instanciação de uma nova página, realizado após a compilação, envolve a criação de todos os controles (*mobile web controls*) utilizados pela página. Toda a lógica (de negócios) contida na página também será executada. Em seguida, os adaptadores de dispositivos (*device adapters*), associados ao dispositivo que efetuou a requisição e aos controles utilizados na página, geram a resposta na linguagem de marcação de texto (*markup language*) adequada. A página de resposta então é encapsulada em um *HTTP response* e retornada ao dispositivo que solicitou a requisição.

A Figura abaixo ilustra a requisição a uma mesma página .aspx realizada por dispositivos que possuem capacidades diferentes.

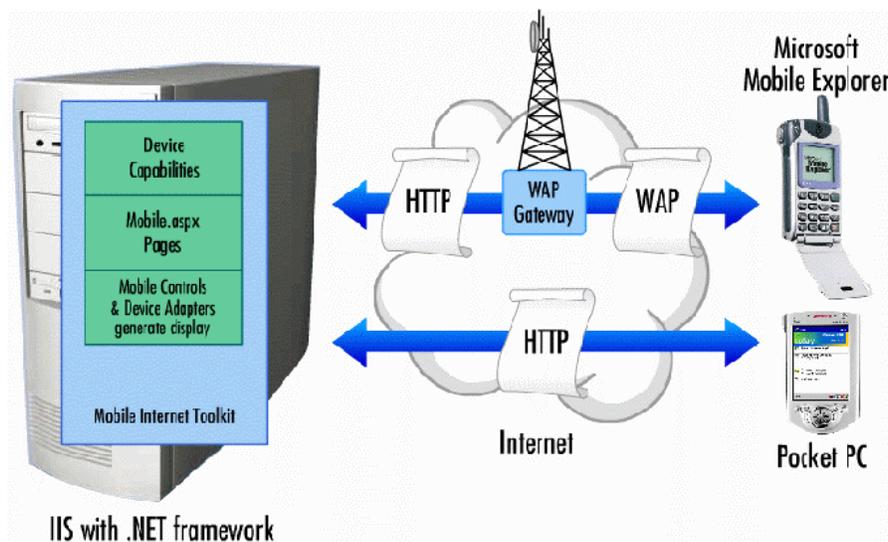


Figura 19 - Arquitetura de aplicações Web Móveis

Note que um browser WAP efetua sua requisição para um *Gateway* WAP, que por sua vez, “traduz” as requisições recebidas em requisições HTTP e as redireciona para o servidor Web através da internet.

É importante salientar que, quando um browser WML acessa uma aplicação web móvel, os mesmos passos de processamento da página serão executados, porém o arquivo .aspx só é compilado uma única vez (na primeira requisição) independente do dispositivo que efetuou a requisição. No esquema apresentado na Figura 19, por exemplo, se os dispositivos solicitarem a mesma página .aspx, a página será compilada uma única vez e não para cada dispositivo, pois na realidade o processo de geração de resposta é o responsável por gerar páginas em linguagens de marcação de texto diferentes, podendo gerá-las a partir de páginas que se encontram na cache.

Enfim, o WML gerado é retornado ao gateway como uma resposta HTTP, pois o gateway será responsável por processar a resposta em código WML e enviá-la de volta como uma resposta WAP (*WAP response*).

5.3 Características

5.3.1 Aplicações localizadas no servidor

Toda lógica de processamento e geração do conteúdo acontece no servidor. E no lado do cliente é necessário apenas o browser.

5.3.2 Suporte a múltiplos dispositivos

O conteúdo de uma única página (Web Form) pode ser distribuído para múltiplos dispositivos e o conteúdo gerado é otimizado de acordo com o dispositivo que fez a requisição. Uma outra vantagem obtido com o suporte a múltiplos dispositivos é que a aplicação não precisa ser recompilada quando novos dispositivos passam a acessá-la.

5.3.3 Suporte a múltiplos linguagens de marcação de texto

As linguagens de marcação de texto suportadas são:

- HTML 3.2 – versão básica de HTML. Suportada pelo Microsoft Internet Explorer existente em dispositivos que usam a família Pocket PC de sistema operacional (incluindo o Pocket Phone Edition) e Windows CE. NET.
- cHtml 1.0 – Compact Html (cHtml) possui conjunto de tags derivadas da versão do HTML 2.0 e HTML 3.2 apropriadas para dispositivos de tela pequena (small-screen devices) e que geralmente só possuem um tipo de fonte.
- WML – O MIT suporta as versões 1.1 e 1.2 de WML.

Como pode ser visto na Figura abaixo, uma página Web Form pode ser “renderizada” diferentemente dependendo do dispositivo que a solicitou.

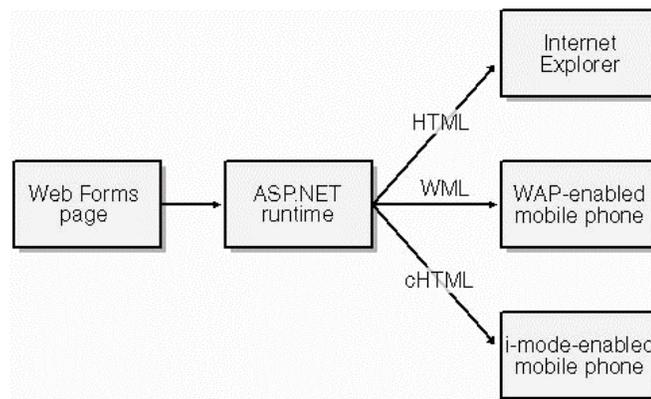


Figura 20 - [ASP .NET] Suporte a Múltiplas Linguagens

5.3.4 Necessidade de trabalho on-line

Uma desvantagem de se usar aplicações móveis *server-side* é o fato delas necessitarem de conectividade a um servidor web.

5.3.5 Menos poder de interface gráfica

Como a resposta gerada na interface da aplicação é em linguagem de marcação de texto, a interface gráfica de aplicações web não oferece recursos sofisticados se comparada com a interface de aplicações *client-side*.

5.3.6 Fácil deployment

Não é preciso, por exemplo, o uso de programas de setup, pois as aplicações executam em um browser sem a necessidade de instalação alguma.

5.4 Protótipos

Foram desenvolvidos alguns protótipos que tiveram como objetivo avaliar a independência de dispositivos e o suporte a múltiplas linguagens de marcação de texto. A seguir, será apresentado um protótipo HelloWorld de uma aplicação ASP .NET móvel.

Para testar a aplicação tratada foram utilizados, além do browser internet explorer, simuladores de dispositivos, que suportam diferentes tipos de *markup languages*. Os simuladores utilizados foram da OpenWave [18].

5.4.1 HelloWorld

A primeira aplicação utilizada é o exemplo de uma página HelloWorld bem simples.

O código utilizado na página aspx (WebForm) foi o seguinte:

```
<body Xmlns:mobile="http://schemas.microsoft.com/Mobile/WebForm">
  <mobile:Form id="Form1" runat="server">
    Hello World!!
  </mobile:Form>
</body>
```

Figura 21 – HelloWorld.aspx

Note que a página possui apenas um *mobile control* (<mobile:Form>) que será renderizado.

5.4.1.1.1 Resultados

Para a verificação da independência de dispositivo obtida com o uso de ASP .NET foram testados 4 clientes diferentes acessando a mesma página aspx apresentada, os clientes foram:

- Um browser internet Explorer

- Simulador de browser OpenWave versão 4.1.1
- Simulador de browser OpenWave versão 5.1
- Simulador de browser OpenWave versão 6.2.2

Os resultados obtidos foram os seguintes:

Internet Explorer

Código gerado:

Resultado no Browser	Código Fonte Gerado
Hello World!!	<pre> <html><body> <form id="Form1" name="Form1" method="post " action="MobileWebForm1.aspx?__ufps=684849"> <input type="hidden" name="__EVENTTARGET" value=""> <input type="hidden" name="__EVENTARGUMENT" value=""> <script language=javascript><!-- function __doPostBack(target, argument){ var theform = document.Form1 theform.__EVENTTARGET.value = target theform.__EVENTARGUMENT.value = argument theform.submit() } // --> </script> Hello World!! </form></body></html> </pre>

Tabela 1 - HelloWorld.aspx gerado para o Internet Explorer

Pode-se perceber que a página gerada possui código HTML e JavaScript, ambos suportados pelo Internet Explorer.

Browser OpenWave versão 4.1.1

Resultado no Browser	Código Fonte Gerado
 A silver OpenWave smartphone with a green screen displaying the text "Hello World!!". The phone has the OpenWave logo and name at the top.	<pre><wml> <head> <meta http-equiv="Cache-Control" content="max-age=0" /> </head> <card> <do type="accept"> <noop /> </do><p> Hello World!! </p> </card> </wml></pre>

Tabela 2 – HelloWorld.aspx gerado para o Browser OpenWave 4.1.1

Note que o código gerado está em linguagem WML compatível com o browser do dispositivo que acessou a página.

Browser OpenWave versão 5.2

Resultado no Browser	Código Fonte Gerado
 A silver OpenWave smartphone with a white screen displaying the text "Hello World!!". The phone has the OpenWave logo and name at the top.	<pre><wml> <head> <meta http-equiv="Cache-Control" content="max-age=0" /> </head> <card newcontext="false" ordered="true"> <do type="accept" optional="false"> <noop /> </do> <p align="left">Hello World!!</p> </card> </wml></pre>

Tabela 3 - HelloWorld.aspx gerado para o Browser OpenWave 5.2

Neste caso o código gerado também foi em WML, mas note que há uma pequena diferença com relação a versão 4.1.1. Pois algumas atributos de algumas tags foram informados. Por exemplo, na tag <p> o atributo `align` foi informado (`align="left"`). Em termos visuais não provocou nenhuma diferença, pois por *default* o texto já aparece alinhado a esquerda.

Browser OpenWave versão 6.2.2

Resultado no Browser	Código Fonte Gerado
	<pre><html> <head> <meta http-equiv="Cache-Control" content="max-age=0" /> </head> <p:card> <p:do type="accept"> <p:noop/> </p:do> <p>Hello World!! </p> </p:card> </html></pre>

Tabela 4 - HelloWorld.aspx gerado para o Browser OpenWave 6.2.2

Neste caso o código gerado incluiu a tag <html> que não aparece nas versões anteriores do browser.

5.4.1.1.2 Conclusão

O exemplo mostrado comprova que o desenvolvimento com ASP .NET para dispositivos móveis suporta a geração de resposta para múltiplos dispositivos e múltiplas linguagens como já mencionado na seção 5.3.

6. Aplicações baseadas em Web Services

Esta Seção trata o desenvolvimento de aplicações baseadas em Web Services para dispositivos móveis. Esse tipo de aplicação foi tratado separadamente por não ser uma aplicação puramente cliente (*client-side*), nem uma aplicação puramente *server-side* como é o caso das aplicações Web móveis tratadas na Seção 5.

Na prática o conceito de web services está totalmente relacionado ao desenvolvimento de aplicações distribuídas e aplicações baseadas em Web Services se encaixam no modelo de arquitetura *client-server*.

Não é objetivo desta Seção tratar com detalhes o desenvolvimento de Web Services com .NET e sim como estes se encaixam no contexto de dispositivos móveis. As seções a seguir fornecem uma introdução a Web Services, em seguida é feita uma análise do seu uso no contexto de dispositivos móveis e finalmente alguns protótipos implementados serão mostrados.

6.1 Introdução a Web-Services

6.1.1 O que são Web Services?

De uma maneira geral, pode-se definir Web Services como componentes que representam serviços remotos possíveis de serem acessados via internet por outras aplicações. Em termos práticos, tem-se um “serviço de software” independente de plataforma, linguagem e localização que recebe requisições de um cliente e retorna respostas para o mesmo. A utilização de um Web Service por uma aplicação também é conhecida como consumo de Web Services.

6.1.2 SOAP e Web Services

Web Services basicamente utilizam os protocolos HTTP e SOAP para permitir que dados de negócio fiquem disponíveis na web.

O protocolo SOAP proporciona ao Web Service independência de plataforma, linguagem e localização, ou seja, não importa que linguagem de programação, sistema operacional ou plataforma são utilizadas na aplicação cliente que acessa o Web Service, assim como não importa onde o cliente está localizado.

As mensagens SOAP são representadas por documentos XML que podem ser enviadas por qualquer camada de transporte, mas a camada de transporte mais utilizada é HTTP.

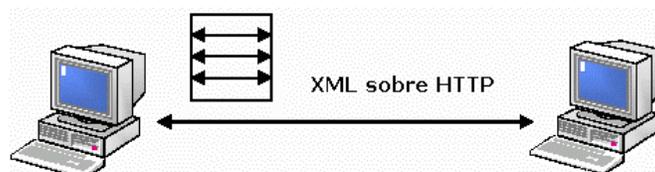


Figura 22 - Mensagens SOAP são documentos XML enviados sobre HTTP

Um Web Service é disponibilizado através de um servidor que precisa estar habilitado para o recebimento de chamadas, logo algum tipo de *listener*, que espera chamadas no formato SOAP, executa no servidor.

Quando uma chamada é aceita, o *listener* lê as informações da mensagem SOAP (no formato XML) e a envia para a aplicação correspondente que irá processá-la. Quando a mensagem é processada pela aplicação é gerada uma resposta que será enviada de volta ao cliente. A seção a seguir apresenta mais detalhes sobre o protocolo SOAP.

6.1.2.1 SOAP

O SOAP (*Simple Object Access Protocol*) é basicamente destinado para promover um mecanismo simples e leve para troca de informações estruturadas (XML) em um ambiente descentralizado e distribuído.

No processo de transmissão de mensagens SOAP, a mensagem original (documento XML) é “envolvida” por um envelope SOAP, que é constituído por diferentes partes: um cabeçalho (*header*) e um corpo (*body*).

As diferentes partes de uma mensagem SOAP contem diferentes tipos de informações.

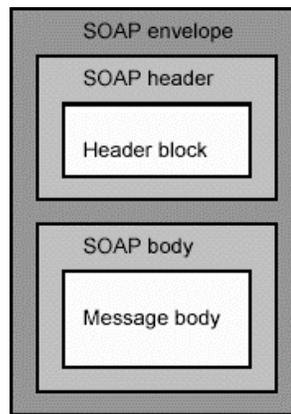


Figura 23 - Estrutura de uma mensagem SOAP

O *header* de uma mensagem SOAP é opcional e geralmente contém informações relacionadas a processos transacionais e autenticações.

O *body* é um elemento obrigatório de uma mensagem SOAP e contém informações relacionadas aos dados da requisição do cliente (*request*) ou então dados relativos a resposta retornada pelo Web Service.

6.1.2.2 Aplicações cliente para Web Services

Para desenvolver aplicações cliente para Web Services é preciso primeiramente conhecer a especificação dos serviços que o Web Service possui. Na prática, um Web Service está associado a um arquivo de descrição com extensão *.wsdl*, que possui a especificação dos serviços oferecidos pelo Web Service.

Um arquivo de descrição é acessado através de uma URL, existem ferramentas e sites onde o usuário pode procurar Web Services juntamente com seus arquivos de descrição. O arquivo de descrição possui informações a respeito dos tipos de parâmetros e retorno dos serviços oferecidos por um determinado Web Service, é através do arquivo de parâmetros que um *proxy* para o Web Service deve ser construído.

O *proxy* permite a interação da aplicação cliente com o Web Service, isto é, ele é o objeto ou classe que representa o Web Service para a aplicação, pois é através dele que a aplicação cliente acessa os serviços do Web Service. Através do arquivo de descrição, uma classe *proxy* pode ser gerada automaticamente por uma ferramenta.

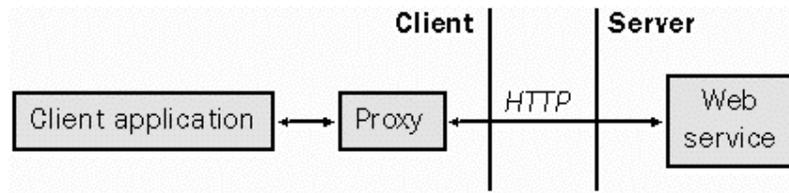


Figura 24 - Proxy de um Web Service

6.1.2.2.1 Ciclo de vida de uma aplicação cliente

A Figura a seguir ilustra o que acontece em tempo de execução no ciclo de vida de uma aplicação cliente utilizando um Web Service.

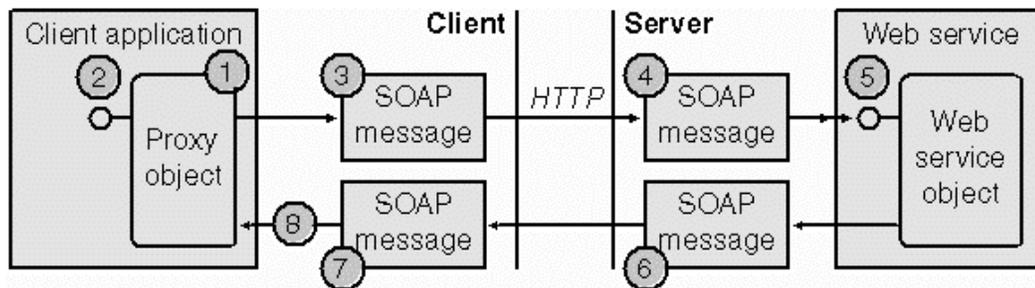


Figura 25 – Ciclo de vida de uma aplicação cliente que acessa um Web Service

1. A aplicação cliente cria uma instância de um *proxy* do Web Service
2. A aplicação cliente chama um dos métodos do proxy
3. A infra-estrutura de Web Service presente no cliente serializa os argumentos passados para o método do *proxy* em uma mensagem SOAP e a envia para o Web Service pela rede via HTTP
4. A infra-estrutura de Web Service no servidor recebe a mensagem SOAP e cria uma instância da classe que implementa o serviço solicitado do Web Service. É neste ponto que os argumentos, presentes na mensagem SOAP, são recuperados

5. Utilizando os argumentos recuperados, o método da classe que implementa o Web Service é executado, criando uma saída de valores a serem retornados
6. A infra-estrutura de Web Service no servidor cria a mensagem SOAP e salva os parâmetros com seus valores de retorno. Em seguida a mensagem é enviada pela rede.
7. A infra-estrutura de Web Service no cliente recebe a mensagem SOAP e recupera os parâmetros e valores retornados
8. O cliente recebe os parâmetros e valores retornados

6.2 Principais Características

A seguir tem-se uma lista das principais características encontradas ao se usar Web Services.

6.2.1 Comunicação entre aplicações (*application-to-application*)

Como foi visto, Web Services são projetados para interagir diretamente com outras aplicações através da internet. Logo, Web Services não possuem interface gráfica e sim interfaces chamadas “contratos” que descrevem os serviços que ele oferece

6.2.2 Independente de Linguagem

Uma aplicação cliente que consome um Web Service pode ser escrita em qualquer linguagem, desde que consiga interagir com o protocolo SOAP e documentos XML

6.2.3 Independente de Protocolo

Ao contrário de algumas tecnologias de componentes, Web Services não utilizam protocolos que são específicos de determinados modelos de objetos, tais como DCOM (*Distributed Component Object Model*). A comunicação com Web Services é feita através de protocolos e formato de dados padrões da Web, tais como HTTP, XML e SOAP. O que permite qualquer servidor, que suporta esses protocolos, acessar ou prover Web Services.

6.2.4 Independente de Plataforma

Como Web Services são acessados através de uma interface padrão, eles permitem que sistemas distintos trabalhem juntos.

6.2.5 Arquitetura Stateless

O modelo de Web Services assume uma arquitetura de serviços stateless, isto é, serviços que não armazenam dados (estado) em uma sessão com o usuário. Cada resposta de um Web Service equivale a um novo objeto, o que corresponde a um novo estado.

6.3 Web Services e Dispositivos móveis

No contexto de dispositivos móveis é possível criar aplicações que funcionam como clientes para Web Services, isto é, dispositivos móveis podem ser utilizados como aplicações que consomem serviços web remotos.

Em termos práticos, os dois tipos de aplicações, para dispositivos móveis, estudados (client-side e server-side) podem atuar como clientes de *Web Services*, porém o uso direto (no próprio dispositivo) só é realizado com a abordagem *client-side*.

Nesta Seção, será tratado o consumo de Web Services por dispositivos móveis através de aplicações *client-side* e em seguida algumas considerações serão feitas a respeito da utilização de aplicações web móveis (*server-side*) e o consumo de *Web Services*.

6.3.1 Consumo de web-services por aplicações client-side

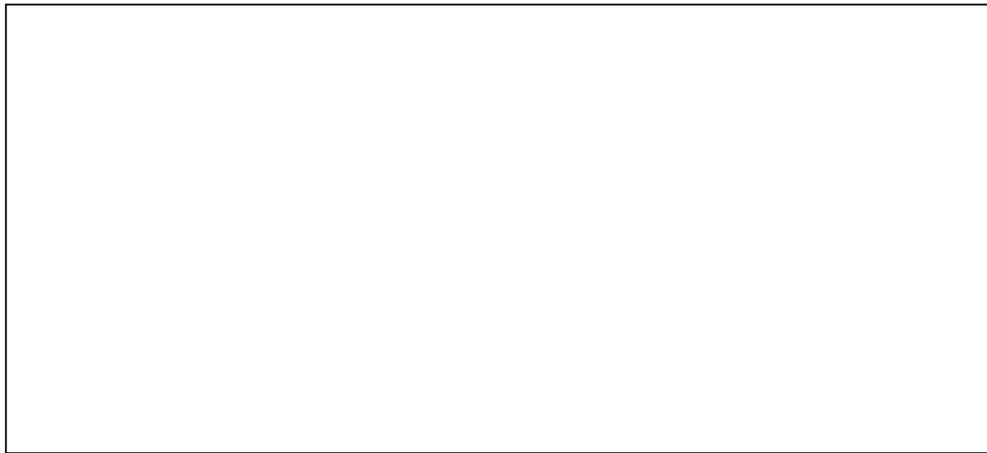
O uso do .NET Compact Framework permite a construção de aplicações em dispositivos móveis que funcionam como clientes de Web Services. Como os passos gerais para a construção de clientes Web Services foram exibidos na Seção 6.1.2.2, aqui será dado um foco mais prático através da apresentação de dois protótipos de aplicações que consomem um Web Service.

6.3.1.1 Protótipo 1 – HelloWorld

Neste exemplo foram implementados tanto o Web Service quanto a aplicação cliente que o consome. A implementação do Web Service serviu para analisar o suporte que o ambiente de desenvolvimento utilizado (Visual Studio 2003) possui para a construção de Web Services.

6.3.1.1.1 Web Service Hello World

Para a construção do Web Service foi preciso implementar a classe que implementa o serviço a ser oferecido remotamente. Como a linguagem de programação utilizada foi C# o arquivo com o código fonte possui a extensão .asmx.cs. Neste caso, um arquivo chamado HelloWorld.asmx.cs foi implementado.

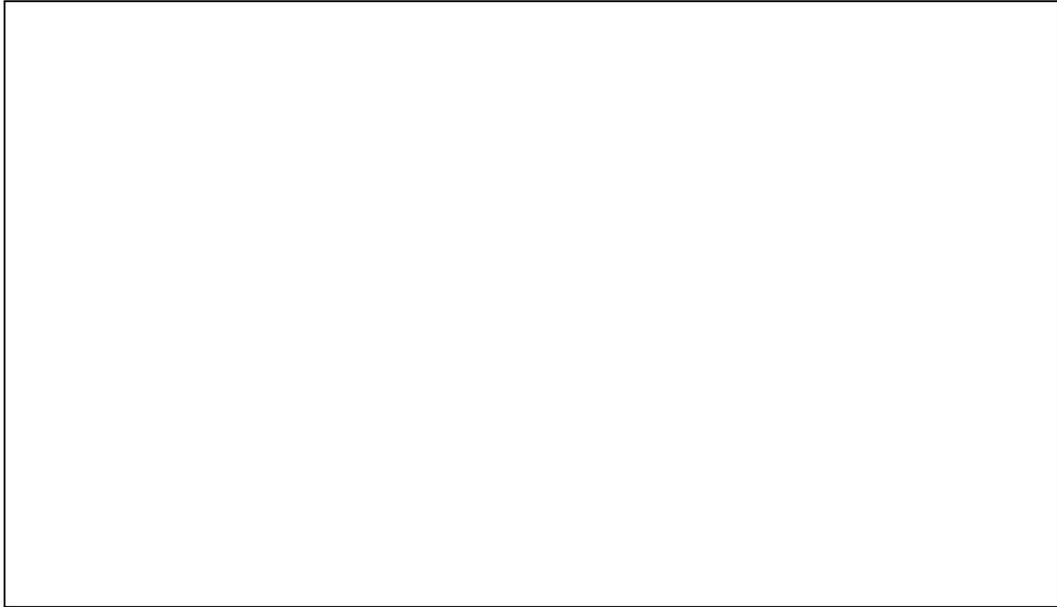


Note que o suporte a Web Services oferecido pelo .NET é considerável, pois, como pode ser visto no trecho de código, para implementar um serviço web remoto basta colocar a expressão `[WebMethod]` antes da assinatura do método de uma classe que herda de `System.Web.Services.WebService`

6.3.1.1.2 Aplicação Cliente

A implementação da aplicação cliente para consumir o Web Service criado utilizou o protótipo apresentado na Seção 4.3.1, porém foi preciso importar a classe referente ao *proxy* gerado para o Web Service HelloWorld, o *proxy* foi automaticamente gerado pela ferramenta de desenvolvimento (Visual Studio .NET neste caso).

O trecho de código a seguir mostra o resultado da implementação obtida.



Note que o uso do proxy é muito simples, na realidade é apenas uma chamada a um método de uma classe, todo o processo de criação e envio de mensagens SOAP é abstraído neste caso.

6.3.1.2 Protótipo 2 – Tradutor BabeFish

Neste protótipo foi desenvolvida uma aplicação para dispositivo móvel que consome um Web Service existente na Web. O Web Service utilizado foi o BabelFishService que é um serviço de tradução de texto da Altavista [19].

A URL do arquivo descritor (visto na Seção 6.1.2.2) do BabelFishService é:

- <http://www.xmethods.net/sd/2001/BabelFishService.wsdl>

O arquivo BabelFishService.wsdl contem as informações apresentadas na tabela a seguir:

BabelFishService.wsdl	
Method namespace URI	urn:xmethodsBabelFish
Input Parameters	<ul style="list-style-type: none">o translationmode (string)o sourcedata (string)
Output Parameters	return (string)
Endpoint URL	http://services.xmethods.net:80/perl/soaplite.cgi

Tabela 5 – Informações do BabelFishService.wsdl

O resultado do protótipo pode ser visto na Figura a seguir:



Figura 26 – Cliente consumindo o serviço de tradução BabelFishService

O trecho de código a seguir apresenta como foi implementado a chamada ao Web Service, quando o botão “Traduzir” é pressionado.



6.3.2 Aplicações Web Móveis vs Web Services

Em alguns cenários, dependendo do tipo de aplicação, a escolha entre utilizar uma determinada abordagem de desenvolvimento para aplicações móveis é direta. Por exemplo, em um cenário de aplicações distribuídas onde há a comunicação entre aplicações que foram implementadas em diferentes linguagens e/ou plataformas, muito provavelmente uma solução envolvendo o uso de Web Services será adotado.

Porém, em algumas situações, onde parte do processamento de informações precisa ser realizado no servidor, a escolha entre implementar uma aplicação web móvel utilizando ASP .NET pode se deparar com a opção de implementar uma aplicação *client-side* que utiliza serviços remotos na web (Web Service). Nestes casos uma análise mais detalhada das características de ambas abordagens deve ser feita. A seguir tem-se uma breve análise de alguns cenários de uso onde determinado tipo de aplicação pode ser mais adequado.

6.3.2.1 Cenário 1 – Rica interface gráfica

Este cenário representa as aplicações que possuem como principal requisito uma interface gráfica com o usuário (GUI) rica. Para estes casos a adoção de aplicações client-side utilizando um componente remoto (Web Service) pode ser mais interessante, pois dependendo dos requisitos estabelecidos na interface gráfica pode não ser possível implementar uma página ASP .NET que atenda às necessidades definidas. O uso de uma aplicação *client-side* seria bastante proveitoso neste caso, pois como visto na Seção 4.2.3 este tipo de aplicações oferece um suporte muito bom no que se diz respeito a componentes visuais que podem ser utilizados na interface de aplicações para dispositivos móveis.

Vale ressaltar que esta opção seria válida caso os dispositivos considerados utilizassem a família de sistemas operacionais Pocket PC ou Windows CE .NET, pois no caso de alguns celulares, por exemplo, que não suportam estas plataformas o uso de ASP .NET seria indiscutível, dado que foi considerado parte do processamento sendo feita em um servidor.

6.3.2.2 Cenário 2 – Serviços prontos

Um cenário a ser considerado também é quando um serviço, necessário na aplicação, já está implementado e disponível como um Web Service. Nestes casos o uso de aplicações *client-side* consumindo o Web Service na maioria das vezes é mais viável se comparado a uma aplicação puramente com ASP .NET, pois a reutilização do serviço como um Web Service evita a sua implementação caso a segunda abordagem fosse adotada.

Uma outra opção para este caso seria utilizar um misto das duas abordagens, onde uma aplicação móvel ASP .NET funcionaria como um cliente consumindo o Web Service. Isto seria válido, mais uma vez, para os dispositivos incompatíveis como a família de sistemas operacionais Pocket PC ou Windows CE .NET, pois estes não usufruem das aplicações *client-side* implementadas com o .NET Compact Framework. A arquitetura para esta solução está ilustrada na Figura a seguir:

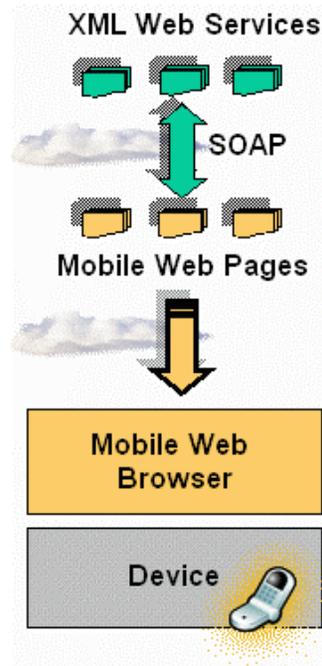


Figura 27 – Arquitetura de Web Service com ASP .NET

6.3.2.3 Cenário 3 – Serviços com retorno de dados

Como visto, o uso de Web Services permite a chamada remota de métodos que retornam dados como resultado, o que não é possível com a utilização de páginas ASP .NET. Nestes casos o uso de Web Service apresenta-se mais adequado, pois é possível implementar facilmente serviços que retornam dados.

Em algumas situações os dados retornados por um Web Service podem variar de uma simples string até um conjunto de dados que podem facilmente ser recuperados no cliente com objetos ADO .NET (veja exemplo na Seção 4.3.4), já que o retorno de um Web Service é um documento no formato XML.

6.3.2.4 Cenário 4 – Arquitetura Stateless

De acordo com as características de Web Services apresentadas na Seção 6.2, tem-se que o modelo de Web Services assume uma arquitetura de serviços que não armazenam dados (estado) em uma sessão com o usuário, pois cada resposta

de um Web Service equivale a um novo objeto, o que corresponde a um novo estado.

Sendo assim, o uso de Web Services em aplicações que necessitam armazenar dados em uma sessão com o usuário (arquitetura *statefull*) não se mostra viável, já que o estado é perdido. Logo, ao invés de um Web Service, pode-se optar por utilizar ASP .NET pois com este possui funcionalidades que permitem o gerenciamento e manutenção de estado entre requisições do usuário.

7. Conclusão

O uso de .NET no desenvolvimento de aplicações para dispositivos móveis pode ser considerado uma excelente opção, pois apresenta grande similaridade com o desenvolvimento de aplicações .NET desktop, o que proporciona uma curva de aprendizado bastante elevada se comparada com outras tecnologias para dispositivos móveis.

Como relação as abordagens de desenvolvimento oferecidas, não se pode fazer uma análise comparativa entre elas e indicar a melhor alternativa, pois cada uma das abordagens apresenta características próprias e são voltadas para tipos diferentes de aplicações.

De uma maneira geral pode-se concluir que a abordagem *client-side* deve ser utilizada para a implementação de aplicações *stand-alone* que possuem rica interface gráfica; a abordagem de desenvolvimento com ASP .NET apresenta-se como uma solução para aplicações web compatíveis com uma grande variedade de browsers; e por fim os Web Services surgem como uma nova solução para a implementação de aplicações ou componentes distribuídos acessados via internet.

Cada abordagem possui propósitos específicos, logo, o cenário de desenvolvimento e as características necessárias determinarão que abordagem será mais apropriada.

7.1 Trabalhos Futuros

Este trabalho analisou as características das diferentes abordagens de desenvolvimento baseado na construção de protótipos de aplicações testados em simuladores.

Uma possível extensão deste trabalho seria fazer uma análise mais refinada, das características estudadas, através de testes em dispositivos reais. Esta análise faria uso de aplicações mais complexas onde outros fatores, tais como performance, poderiam ser levados em consideração.

Além disso, um estudo relacionado a aspectos como concorrência e sincronização de dados poderia ser feito sobre aplicações móveis distribuídas, implementadas com Web Services.

8. Referências

- [1] Andy Wigley, Peter Roxburgh, **Building .NET Applications for Mobile**, Microsoft Press, 2002
- [2] Andy Wigley, Stephen Wheelwright, **Microsoft .Net Compact Framework**, Microsoft Press, 2002
- [3] **.NET Framework**
<http://msdn.microsoft.com/netframework/>, visitada em Maio, 2003
- [4] **Mobile Device Developers**,
<http://www.microsoft.com/mobile/developer/default.asp>, visitada em Maio, 2003
- [5] **Evolution of Mobile Web Services**,
http://www.fawcette.com/xmlmag/2002_08/magazine/columns/jjurvis/, visitada em Maio, 2003
- [6] **First Look at Smart Device Extensions**,
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnroad/html/road10242001.asp>, visitada em Maio, 2003
- [7] **Application Architecture for .NET: Designing Applications and Services**,
<http://msdn.microsoft.com/architecture/application/default.aspx?pull=/library/en-us/dnbd/html/distapp.asp>, visitada em Maio, 2003
- [8] **.NET Mobile Web SDK: Build and Test Wireless Web Applications for Phones and PDAs**,
<http://msdn.microsoft.com/msdnmag/issues/01/06/Mobile/default.aspx>, Junho, 2002
- [9] **Best Practices for Deploying Your Mobile Application and Services**,
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmitta/html/bestpracdeplmob.asp>, Janeiro, 2002
- [10] **Microsoft Mobile Internet Toolkit Lets Your Web Application Target Any Device Anywhere**, <http://msdn.microsoft.com/msdnmag/issues/02/11/MMIT/>, visitada em Junho, 2003
- [11] **An example of a working SOAP client**,
<http://hem.passagen.se/rebka/thesis13maj.pdf>
- [12] **Creating web services**,
<http://www.15seconds.com/Issue/010430.htm>
- [13] **Building Client Interfaces for .NET Web Services**,
<http://www.15seconds.com/issue/010530.htm>

- [14] **MSDN Training 2310B - Developing Microsoft ASP.NET Web Application,**
Microsoft Corporation

- [15] **Creating a Multiple Form Application Framework for the Microsoft .NET Compact Framework,**
<http://msdn.microsoft.com/library/en-us/dnnetcomp/html/netcfuiframework.asp>

- [16] **Creating a Microsoft .NET Compact Framework-based Animation Control,**
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/animationcontrol.asp>

- [17] **Creating ASP.NET Web Services,**
<http://www.ondotnet.com/lpt/a/2648>

- [11] **Developer. Support & Training – FAQ,**
<http://www.microsoft.com/mobile/developer/support/faq.asp#>, visitada em Maio, 2003

- [18] **OpenWave**
<http://www.openwave.com/>

- [19] **Altavista,**
<http://www.altavista.com>

- [20] **Java**
<http://java.sun.com/>