

► Aprenda a disciplina,  
busque a arte e contribua  
com idéias no:  
[www.ArchitectureJournal.net](http://www.ArchitectureJournal.net)  
Recursos nos quais você  
pode confiar.

# THE ARCHITECTURE JOURNAL™

Idéias para melhores resultados

Journal 14

## Arquitetura Móvel

Considerações Arquiteturais  
para um Mundo de Dispositivos

---

Melhores Práticas: Como  
Estender Aplicativos Corpora-  
tivos aos Dispositivos Móveis

---

Experiência para o  
Consumidor Conectado  
no Campo Automotivo

---

Perfil do The Architecture  
Journal: Faisal Waris

---

Arquitetura de Dados Móveis

---

Desenvolvimento Dirigido a  
Teste para Aplicativos Móveis

---

Estudo de Caso: Técnicos de  
Suporte de Campo





# Sumário

## Apresentação

1

por Simon Guest

## Considerações Arquiteturais para um Mundo de Dispositivos

2

por Atanu Banerjee

Explore os aspectos e as considerações arquiteturais do projeto de aplicativos para dispositivos móveis.



## Melhores Práticas: Como Estender Aplicativos Corporativos aos Dispositivos Móveis

10

por Kulathumani Hariharan

Descubra quais são as melhores práticas e recomendações para estender aplicativos corporativos a uma plataforma móvel.



## Experiência para o Consumidor Conectado no Campo Automotivo

17

por Christoph Schittko, Darryl Hogan e Jon Box

Como o setor automotivo pode ser ampliado para absorver capacidades de software mais avançadas? Explore um cenário e as considerações arquiteturais para criar experiências para o consumidor conectado.



## Perfil do The Architecture Journal: Faisal Waris

24

No nosso primeiro perfil de arquiteto externo, conversamos com Faisal Waris sobre a sua função, as suas idéias sobre dispositivos móveis e as tendências arquiteturais gerais.



## Arquitetura de Dados Móveis

26

por Rodney Guzman

Quais são os desafios impostos pelos dados, referentes a aplicativos móveis ocasionalmente conectados, e como é possível suplantá-los?



## Desenvolvimento Dirigido a Teste e Integração Contínua para Aplicativos Móveis

31

por Munjal Budhabhatti

Saiba como o desenvolvimento dirigido a teste e a contínua integração podem ajudar a aumentar a confiabilidade dos aplicativos e como estes podem ser utilizados nos aplicativos móveis.



## Estudo de Caso: Técnicos de Suporte de Campo

36

por András Velvárt e Peter Smulovics

Examine um estudo de caso que analisa como os técnicos de suporte de campo na Hungria se beneficiam com um aplicativo móvel, de última geração.



**Fundador**  
Arvindra Sehmi  
Microsoft Corporation

**Editor-chefe**  
Simon Guest  
Microsoft Corporation

**Conselho Editorial Microsoft**  
Gianpaolo Carraro  
John deVados  
John Evdemon  
Neil Hutson  
Eugenio Pace  
Javed Sikander  
Philip Teale

**Editor**  
Lisa Slouffman  
Microsoft Corporation

**Design, impressão e distribuição**  
CMP Technology – Editora  
Chris Harding, diretor administrativo  
Angela Duarte, gerente de publicação  
Lisa Broschitto, gerente de projeto  
Kellie Ferris, diretor de publicidade  
Jimmy Pizzo, diretor de produção

**Diagramação, finalização e impressão (BR)**  
Arthéria Comunicação & Multimídia  
[www.artheria.com.br](http://www.artheria.com.br)

## Microsoft®

As informações contidas neste The Architecture Journal ("Jornal") têm finalidade informativa, apenas. As matérias do Jornal não constituem a opinião da Microsoft Corporation ("Microsoft") nem da CMP Media LLC ("CMP") e, tampouco, são recomendações da Microsoft ou da CMP; assim sendo, você não deve confiar em nenhuma das matérias deste Jornal sem solicitar o aval de um consultor independente. A Microsoft e a CMP não fazem declarações nem oferecem garantias quanto à exatidão ou adequação para determinada finalidade de qualquer matéria deste Jornal e, em nenhuma circunstância, nem a Microsoft, nem a CMP aceitará responsabilidade de qualquer tipo, inclusive responsabilidade por negligência (salvo em caso de danos físicos ou morte) em relação a quaisquer tipos de perdas ou danos (incluindo, mas não se limitando, aos casos de perda de negócios, de receita, de lucros ou prejuízo imprevisto) resultantes do uso das informações deste Jornal. O Jornal pode ter imprecisões técnicas e erros tipográficos. O Jornal pode ser atualizado periodicamente e, às vezes, poderá estar desatualizado. A Microsoft e a CMP não se responsabilizam pela atualização das informações deste Jornal nem por deixar de fazê-lo. Este Jornal contém matérias encaminhadas e criadas por terceiros. Até o máximo permitido pela lei aplicável, a Microsoft e a CMP isentam-se de todas as responsabilidades por qualquer ilegalidade decorrente de erro, omissão ou imprecisão deste Jornal; além disso, a Microsoft e a CMP não se responsabilizam pelas matérias recebidas de terceiros.

As marcas comerciais a seguir são marcas comerciais registradas da Microsoft Corporation: ActiveSync, RoundTable, Silverlight, SQL Server, Virtual Earth, Visual Studio, Windows CardSpace, Windows Live, Windows Mobile, Windows Server, Windows Vista, Xbox 360, Xbox Live. Quaisquer outras marcas comerciais são de propriedade de seus respectivos proprietários.

Todos os direitos autorais e outros direitos de propriedade intelectual das matérias publicadas no Jornal pertencem à Microsoft Corporation ou estão a ela licenciados. Copiar, reproduzir, transmitir, armazenar, adaptar ou modificar o layout ou o conteúdo deste Jornal são ações proibidas, a não ser que haja autorização prévia, por escrito, da Microsoft Corporation e de cada um dos seus autores.

Copyright © 2008 Microsoft Corporation. Todos os direitos reservados.

# Apresentação

## Caro Arquiteto,

A cada dia vejo a difusão dos celulares e dispositivos móveis como algo incrível, especialmente quando analiso a taxa de crescimento projetada para os próximos anos. Com tal crescimento e com os rápidos avanços da tecnologia móvel, vejo que, muito provavelmente, meus filhos crescerão sem nunca conhecer o verdadeiro significado dos termos linha fixa, discagem e pulso! Em qualquer tecnologia, o software desempenha papel importante na complementação desse fenomenal crescimento do hardware, e este é o enfoque desta edição do *The Architecture Journal*.

Iniciando esta edição, Atanu Banerjee aborda muitas considerações e aspectos atuais dos aplicativos em dispositivos móveis. Em seguida, temos Kulathumani Hariharan, arquiteto da Tata Consultancy Services, compartilhando melhores práticas, dicas e recomendações que podem ser úteis, caso esteja pensando em instalar um aplicativo de linha de negócio em plataforma móvel.

Depois, temos Christoph Schittko, Darryl Hogan e Jon Box que nos apresentam o cenário de uma experiência de consumidor conectado em dispositivos automotivos. Vamos explorar como será o futuro do software nos automóveis e algumas das perspectivas arquiteturais que servem de suporte. Estritamente relacionado a este artigo, temos muito prazer em apresentar o nosso primeiro perfil de arquiteto externo no *The Architecture Journal*. Faisal Waris é consultor em arquitetura e trabalha na Ford Motor Company. Formulamos algumas perguntas para descobrirmos quais são os seus conceitos de arquitetura, especialmente no que concerne ao desenvolvimento móvel.

Depois da entrevista com Faisal, Rodney Guzman da InterKnowlogy nos conta o que pensa da arquitetura para dados móveis. Rodney explora alguns dos desafios impostos pelos dados com aplicativos ocasionalmente conectados e apresenta algumas idéias e conceitos para ajudar a abordar a resolução do conflito de dados. Em um mergulho mais profundo no desenvolvimento móvel, encontramos Munjal Budhabhatti da ThoughtWorks, que aborda a importância do desenvolvimento dirigido a teste e a integração contínua, práticas de engenharia comuns em muitas organizações e discute como esses procedimentos podem ser implementados nos aplicativos móveis.

Fechamos esta edição com uma viagem à Hungria, com András Velvárt e Peter Smulovics, para saber como a Monicomp, organização que instala e presta serviços de manutenção e reparos para sistemas em pontos de serviço, utiliza um aplicativo para PC, ultra-móvel, para dar suporte aos seus técnicos de campo.

E, assim, fechamos esta edição. Espero que alguns dos artigos e autores ajudem a inspirar o desenvolvimento dos aplicativos móveis em sua organização. Voltaremos no ano novo com o Jornal 15 que dissertará sobre o "papel de um arquiteto": focalizaremos com mais detalhes as pessoas que têm a nossa profissão e colocaremos o trabalho que executamos sob as lentes do microscópio!



Simon Guest



# Considerações Arquiteturais para um Mundo de Dispositivos

por Atanu Banerjee

## Resumo

O número de dispositivos móveis cresce rapidamente, mas a promessa de soluções que alavanquem as redes de dispositivos conectados permanece, em grande parte, não concretizada. Até há pouco, alguns dos itens necessários para construir ricas experiências conectadas não estavam disponíveis. Este artigo explora algumas das tendências econômicas, sociais e tecnológicas que movimentam a adoção dos dispositivos móveis; descreve os diferentes tipos de experiências de usuário que, agora, já são viáveis, e apresenta uma síntese das preocupações arquiteturais associadas a essas soluções de mobilidade, nos seguintes níveis: hardware, software, conectividade e capacidades de serviço.

## Os dispositivos oferecem novas oportunidades

Após dez anos de entusiasmo, as soluções de mobilidade estão, finalmente, alçando voo. Você poderia perguntar: "Por que só agora? O que mudou? Existem novas oportunidades a serem consideradas? Devo considerar dispositivos móveis em minhas soluções?" Acontece que tendências econômicas, sociais e tecnológicas estão acelerando a mudança para os dispositivos. Existe amplo espectro de dispositivos com diferentes fatores de forma, executando diferentes tipos de aplicativos, conforme ilustrado na Figura 1, associado a diferentes conjuntos de tendências.

## Tendências econômicas

Os mercados emergentes têm sido um condutor importante da adoção de celulares. Por exemplo, a revista *BusinessWeek* informa que em 2001 havia 500 mil linhas telefônicas na Nigéria; hoje, a telefonia celular do país tem mais de 30 milhões de assinantes. A estimativa atual revela que a telefonia celular mundial terá cinco bilhões de assinantes em 2015.

A adoção de celulares impulsiona a adoção de serviços para celulares. Na Ásia, muitos serviços aproveitam os dispositivos topo de linha para distribuir mídia rica e interativa. Esses serviços exigem smartphones de última geração e pocket PCs, como ilustrado pela Figura 1. Entretanto, muitas pessoas dos mercados emergentes não têm poder aquisitivo para tais dispositivos e, assim, serviços destinados a aparelhos para a massa de menor poder aquisitivo também estão sendo lançados. Em março de 2007, a Safaricom do Quênia, lançou um serviço baseado em SMS para pagamentos móveis denominado M-Pesa, ampla e rapidamente aceito. Como seria de se esperar, o acesso a comunicações aprimoradas também pode trazer muitos benefícios às economias locais. O Dr. Robert Jensen da Brown University dirigiu um estudo sobre pescadores indianos que começaram a usar celulares para encontrar os melhores mercados costeiros para o seu pescado. Os pescadores viram seus lucros aumentarem em 8% e, ao mesmo tempo, os preços ao consumidor, na prática, caíram 4% pois reduziu-se a perda de produto.

Atualmente, em Helsinque, na Finlândia, 57% dos bilhetes únicos do transporte público são pagos pelo celular. Na Croácia, paga-se mais da metade de todos os cartões de estacionamento pelo celular. Em Londres, 20% do movimento da taxa de congestionamento são pagos pelo celular. (Vide referências: *Mobile Phones As Mass Media* [Celulares como mídia de massa].)

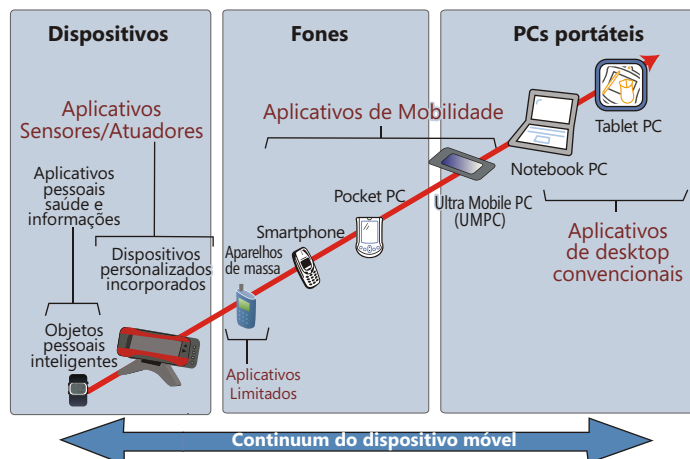
## Tendências sociais

Atualmente, o mundo tem duas vezes mais celulares do que PCs. Em março, o setor sem fio serviu-se da abertura da sua maior feira de negócios para expor as oportunidades para o mundo em "tela tripla" (PC, TV, celular) no qual os dispositivos móveis transformam-se nas principais avenidas para shows de TV, música, jogos e publicidade. Para muitos consumidores jovens, pode-se ainda argumentar que a do dispositivo móvel é a mais importante dessas três telas.

Acompanhando o crescimento dos dispositivos, a evolução da Internet nos encaminha a novos padrões de uso. As soluções atuais se diferenciam das antigas pelo seu alcance e escala globais, que levam a novos canais de participação do usuário. Por exemplo, a empresa Nike (material esportivo) vende o Nike+, pequeno sensor que se encaixa no tênis de corrida para acompanhar a atividade do atleta em um iPod que ele leva no bolso. Ao conectar o iPod da Apple ao PC, os detalhes da corrida desse usuário serão publicados no Nike+, site de rede de comunicação social para corredores, que permite a formação de grupos, em todo o mundo, destinado à troca de informações sobre itinerários e progressos dos corredores.

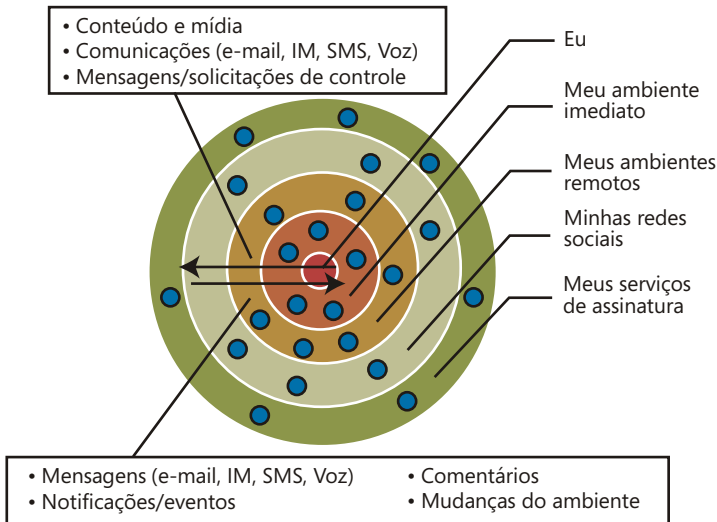
A Internet também está mudando a forma com que se cria conteúdo. A atividade dos blogueiros transforma a publicação de conteúdo menos impessoal, pois os leitores ficam mais próximos dos autores. O conteúdo também se torna interativo e social com jogos on-line, batepapo e com o advento das comunidades que se formam à volta de conteúdo gerado por usuários. Esta tendência será maior com a proliferação de dispositivos móveis que facilitam a captura de

Figura 1: Gama de dispositivos móveis e os tipos de aplicativos que executam





**Figura 2:** As experiências de usuários em um mundo de dispositivos conectados transpõem vários espaços físicos e virtuais. Cada ponto azul representa um ambiente físico (sala, casa, local de trabalho), um ambiente social (amigos, família, colegas), um ambiente virtual (páginas de perfis, mundo virtual, jogo on-line) ou um serviço on-line de assinatura.



conteúdo em dispositivo que ofereça edição direta desse conteúdo, anexar contexto ao conteúdo recém-criado e, depois, fazer o upload do conteúdo para armazenamento no PC ou na nuvem. Em alguns cenários P2P, também é possível compartilhar conteúdo diretamente, do próprio dispositivo.

Esta transformação dos pipelines de conteúdo faz com que seja mais provável haver criação de conteúdo acionada por eventos externos do que conforme programações fixas. A praticidade dos dispositivos aumenta a probabilidade de termos em mãos os meios de gravação de um evento, no momento em que acontece, levando à criação espontânea de conteúdo novo o qual, de outra forma, talvez não fosse capturado. Não é de surpreender, portanto, a explosão no volume de conteúdo gerado e armazenado. O aumento no volume de informações espontâneas dos cidadãos resultou, muitas vezes, em filmagens feitas com dispositivo móvel e numa dramática reação do público.

## Tendências tecnológicas

A primeira tendência que vem remodelando o setor é a da convergência. Atualmente, as pessoas utilizam grande variedade de dispositivos: smartphones, PDAs, laptops, PMPs (*personal media players*), câmeras e filmadoras portáteis. Espera-se que essas tecnologias convirjam em dispositivos de computação pessoal de uso geral, mais poderosos, que possam ser utilizados por ampla variedade de negócios e para tarefas orientadas ao consumidor. A convergência em redes significará tratamento transparente de voz e dados utilizando os mesmos protocolos.

A convergência traz uma segunda tendência: os dispositivos estão mais inteligentes. A nova geração de smartphones torna-se cada vez mais consciente dos ambientes do usuário e do contexto local por meio de sensores (como GPS e acelerômetro) e de melhor software instalado no dispositivo. Este contexto pode ser utilizado para marcar conteúdo (por exemplo, marcar uma foto com metadados de hora e local) para personalizar o comportamento do aplicativo (nenhum aplicativo informa quando o usuário está falando ao telefone, por exemplo), ou para controlar o ambiente local do usuário (como as

configurações de ajuste automático de um automóvel feitas com base na identidade do motorista, as quais foram recuperadas de um dispositivo do motorista). As redes também se ampliarão para abranger dispositivos e agentes distribuídos à volta do corpo, por meio de protocolos como o Bluetooth, conceito presente nas redes de área pessoal (PANs - *Personal Area Networks*). Tudo isso levará a novas arquiteturas nas quais os dispositivos desempenham um papel muito mais importante do que simples visores de informações: eles se tornarão, em si, plataformas de aplicativos de primeira classe. Não apenas isso, mas provavelmente haverá cenários (como as PANs) em que alguns dispositivos atuarão como servidores para outros dispositivos (em arquiteturas cliente-servidor) ou como superusuários/servidores de índice (nas arquiteturas P2P).

Isso é importante, pois os aplicativos para dispositivos móveis precisam proporcionar uma experiência de usuário muito diferente daquela de um desktop. A principal característica dos usuários móveis é que eles estão envolvidos em outras atividades primárias. Aplicativos baseados em dispositivo não devem impor concessões aos seus usuários; em lugar disso, devem se ajustar às vidas e aos estilos de vida das pessoas, estando cientes do contexto, sem causar obstruções, e prontos a proporcionar valor rapidamente, a curto prazo.

A terceira tendência é a Internet móvel: coleção de sites e serviços especificamente destinados aos dispositivos móveis e disponível por intermédio de protocolos de Internet. O crescimento da web móvel apressará o consumo dos aplicativos e serviços baseados na Internet em dispositivos móveis, o que hoje está restrito devido às limitações dos dispositivos e planos de acesso móvel.

A combinação dessas três tendências resultará em um movimento na direção da *pervasive computing*. Na medida em que os dispositivos proliferam e tornam-se mais inteligentes, mais poder de computação será incorporado nas extremidades da rede. Da mesma forma, ao melhorarem o tratamento do contexto do usuário, os dispositivos serão cada vez menos obstrutivos. Na medida em que os dispositivos tornam-se mais interconectados, e com a evolução da Internet móvel, os usuários terão à sua disposição rico conjunto de serviços que pode fazer uso desse contexto pessoal. Os limites entre os ambientes humanos e os dispositivos de computação ficarão gradualmente indefinidos e os usuários sentirão como se tivessem a ajuda de seus ambientes imediatos. Dispositivos de computação incorporados em grande número, exigirão novas arquiteturas de solução para tratar desafios que surgem com referência à experiência do usuário, ao gerenciamento de dispositivo, à segurança, ao gerenciamento de conteúdo e assim por diante. O acesso à rede precisa estar disponível de modo universal. Ainda que, obviamente, não estejamos no patamar da *pervasive computing*, estamos trilhando o caminho do crescimento de dispositivos e smartphones incorporados, da disseminação da conectividade sem fio em todos os nossos ambientes (dos locais de trabalho às residências e a alguns automóveis) e da ampla disponibilidade dos serviços de Internet para serem acessados pelos dispositivos.

## Como será a experiência do usuário?

Na medida em que os dispositivos tornam-se mais comuns, o software precisará cobrir uma gama de dispositivos conectados na web e adotar a crescente *pervasive computing* da Internet, um padrão básico da Web 2.0 descrito por Tim O'Reilly como "o software acima do nível de um único dispositivo". Os dispositivos são usados em múltiplos espaços físicos e virtuais (Figura 2, acima).

## Experiências relacionadas a mim

Os dispositivos nesses cenários são tipicamente utilizados para comunicações (telefone, e-mail), coleta de conteúdo (pesquisa móvel),

Figura 3: Rede de comunicação social em dispositivos



consumo de conteúdo (PMPs) e para monitoração de saúde (monitor cardíaco). O âmbito do aplicativo nesses cenários se concentra em informações reunidas sobre você, criadas para você ou aquelas que você consome diretamente. As informações relevantes incluem credenciais (ID ao vivo), contatos, mensagens, informações presenciais e conteúdo pessoal (áudio, vídeo). Os tipos de dispositivos importantes incluem celulares, smartphones, PDAs, UMPCs (*Ultra-Mobile PCs*), laptops e monitores de saúde. A conectividade necessária destina-se ao Bluetooth para PANs, sensores de saúde, etc.

### **Experiências referentes ao meu ambiente local**

Como descrito acima, a proliferação de dispositivos inteligentes levará a espaços nos quais os limites entre o ambiente imediato de uma pessoa e os dispositivos de computação desse ambiente ficam indefinidos. Isso será obtido por intermédio de dispositivos interconectados com sensores incorporados (GPS, acelerômetro, sensor de luz ambiente), que compreendem o contexto do usuário e não são obstrutivos em suas ações, permitindo aos usuários sentirem-se como se tivessem a ajuda do próprio ambiente.

Um exemplo de tal ambiente é a solução Microsoft Auto: conecta dispositivos de usuário (como celulares e MPs portáteis) em um único sistema automotivo, que pode ser operado pela voz do motorista ou por botões no volante. Em 2008, a Ford Motor Company lançará uma solução denominada Ford Sync para viabilizar a próxima geração de experiências de usuários móveis: por exemplo, usuários que entram no carro enquanto atendem ao celular poderão pressionar um botão no volante e a chamada estará conectada ao Sync, sem interrupção. Outro caso que amplia o ambiente automotivo com dispositivos é o OnStar, que fornece segurança e "assistência para manobra": dentro do carro, um dispositivo de comunicação está conectado ao rádio, a uma antena GPS e a um microfone, por intermédio de uma rede integrada (ou barramento).

As salas de conferência também estão se ampliando com dispositivos. O produto Microsoft Round Table combina câmera de videoconferência e microfone com detecção de som e movimento para automaticamente direcionar o foco para a pessoa que fala. Ao eliminar a necessidade de os oradores se virarem para ficar de frente para uma câmera fixa quando começam a falar é coerente com a ideia de que os dispositivos tornam-se menos obstrutivos em suas ações.

Em alguns casos, os dispositivos talvez precisem compartilhar informações com outros dispositivos também nas proximidades do usuário, assim como com serviços baseados na Internet, levantando

questões sobre descoberta, protocolo de comunicação, entendimento compartilhado da identidade, do contexto do usuário, etc.

### **Experiências referentes aos meus ambientes remotos**

Ambientes remotos são similares aos locais, pois são espaços em que os dispositivos reúnem informações e agem. Contudo, ambientes remotos não estão na vizinhança imediata do usuário do dispositivo. Melhor dizendo, os cenários e as experiências que se referem a ambientes remotos de usuários permitem à pessoa conectar-se, monitorar e trabalhar com dispositivos em outros locais. Os motivos para tal seriam monitorar ou até controlar o ambiente nos locais remotos como proteção contra atividades criminosas ou por outros motivos. Por exemplo, as pessoas podem estar interessadas em monitorar remotamente os seus lares ou locais de trabalho (como uma central de dados) ou até mesmo familiares (crianças e pais idosos). Um exemplo simples de dispositivo que realiza essa tarefa é o monitor de bebês. Os negócios talvez queiram monitorar locais remotos; nesta categoria também existem muitos cenários referentes à logística que utilizam dispositivos RFID (identificação por rádio frequência) - por exemplo, garantir o pedigree eletrônico de produtos farmacêuticos, durante o seu transporte, por intermédio de uma cadeia de fornecimento, para eliminar medicamentos falsificados.

### **Experiências referentes às minhas redes sociais**

A Figura 3 apresenta dispositivos móveis instalados em redes sociais, da mesma forma que fazem os PCs. Os usuários utilizam seus dispositivos para pesquisar e localizar pessoas e os respectivos conteúdos, para entrar em contato com amigos e conhecidos, e para compartilhar conteúdo com terceiros.

Contudo, o alcance e a escala dos dispositivos são muito mais amplos do que os PCs, e as interações sociais são mais espontâneas (quando o usuário está em atividade, o celular com câmera está sempre à mão). Os aplicativos e serviços precisam acomodar esses cenários, em que o alcance da atenção do usuário fica nitidamente reduzido.

### **Quais são as arquiteturas necessárias para se construir soluções ponta a ponta que dão suporte aos dispositivos?**

#### **Desafios da elaboração de experiências ricas em dispositivos**

Existem muitas diferenças entre dispositivos e PCs, e seria um erro considerar os dispositivos apenas como versões menores dos PCs. Para oferecer experiências ricas nos dispositivos, os arquitetos de soluções precisam considerar vários fatores restritivos, muito mais do que ao implementar aplicativos em um PC. Essas restrições incluem as capacidades de hardware dos próprios dispositivos, os sistemas operacionais e os tempos de execução do aplicativo instalado no dispositivo, as ferramentas de desenvolvimento, as opções de conectividade e, também, os serviços disponíveis executados na web. Alguns dos desafios a serem enfrentados na construção de experiências para dispositivos são:

1. Diferentemente dos PCs, as pessoas consideram que os dispositivos são acessórios: os usuários transportam os dispositivos consigo e, quase sempre, os consideram expressões do próprio estilo de vida ou até de sua auto-imagem. Fator de sofisticação, concepção avançada e experiência do usuário são itens críticos. **Implicações:** o dispositivo precisa ter concepção elaborada e capacidades de apresentação: tanto para o próprio dispositivo como para os aplicativos que executará.
2. Os recursos dos dispositivos são limitados. Na medida em que os dispositivos (e suas baterias) tornam-se menores e mais leves, os tamanhos das telas e os seus layouts ficam mais restritos e o poder aquisitivo para absorver esses aplicativos ricos, encolhe.

Dispositivos mais complexos tendem a ter bateria com menos autonomia do que os telefones puros e simples. A memória disponível dos dispositivos também é limitada, embora isso seja minimizado pelos aprimoramentos na tecnologia de armazenamento. A compatibilidade dos dispositivos com Wi-Fi, quase sempre, também custa o preço de uma vida útil de bateria menor. *Implicações:* a eficiência da energia é questão crítica. Os sistemas operacionais dos dispositivos devem ter controle refinado sobre a utilização do hardware. Em alguns casos, o processamento do aplicativo deveria ser descarregado do dispositivo para evitar consumo excessivo dos recursos. Isto leva a um equilíbrio com os dois primeiros itens desta lista.

3. Os dispositivos não são padronizados. Ao contrário dos PCs, os fatores de forma e os perfis de hardware/software dos dispositivos são muito menos padronizados. *Implicações:* desenvolvedores de aplicativos para dispositivos móveis precisam ter como objetivo o menor denominador comum para tamanho, formato e orientação da tela para implementar grande variedade de aparelhos. Esse fato leva a uma compensação implícita com a necessidade de se ter uma rica experiência do usuário; arquitetos de soluções terão de otimizar software e hardware, juntos, para obter a melhor experiência global. Os desenvolvedores de aplicativos para dispositivos incorporados têm um conjunto de desafios diferentes nesta área: falta de padronização em hardware dificulta pressupor o conjunto de recursos disponível para o aplicativo.

4. Os dispositivos precisam ser compatíveis com cenários offline e, ocasionalmente, com conexões lentas. Por muitos motivos, os dispositivos nem sempre estão conectados (por exemplo, as conexões talvez não sejam econômicas durante uma viagem internacional) ou as velocidades de acesso podem não ser as adequadas para dar suporte à tomada de decisão no ponto necessário (usar mapas on-line para tomar decisões de itinerários durante a viagem). *Implicações:* os dispositivos precisam ser mais do que visores de cliente magro: precisam ser, em si, plataformas de aplicativos. Processamento e armazenamento locais, sincronizados com PCs ou serviços de Internet, são requisitos-chave para essa capacidade.

5. A conectividade não é padronizada. Embora haja vários padrões de protocolos de rede, existem muitas formas para o usuário acessar informações e serviços na Internet, por meio dos seus dispositivos móveis. Dependendo das capacidades de cada dispositivo e do plano de serviço contratado com o operador de rede, os usuários podem desejar acessar informações e serviços na Internet móvel via voz (pelo reconhecimento de voz), por troca de mensagens (SMS, e-mail) ou por meio de protocolos de Internet (WiFi, conexão direta (*tethered*) ou plano de dados adequado). Em mercados emergentes, provavelmente os serviços precisarão ser

distribuídos por SMS ou por voz, pois os usuários terão aparelhos de massa, com capacidades limitadas. Para ricas experiências de mídia, talvez seja preciso ter um protocolo de Internet rápido.

*Implicações:* acesso personalizado ao tipo de experiência distribuído e ao mercado ao qual se destina. Os serviços precisarão, talvez, estar acessíveis para os dispositivos a partir de várias extremidades, cada qual compatível com endereço e método de acesso diferentes.

## Aplicação da abordagem Software + Serviços a dispositivos

Na última edição do *The Architecture Journal* foi abordado o paradigma emergente: Software + Serviços. No contexto de aplicativo móvel, a meta é combinar o melhor da web com os melhores aspectos dos dispositivos, embora sujeito às restrições descritas acima. Como ilustrado pela Figura 4, os arquitetos de soluções precisam projetar para um tipo específico de experiência de usuário e escolher o dispositivo apropriado com base nas capacidades mínimas necessárias do dispositivo (hardware e software instalados) e nas opções e serviços de conectividade disponíveis aos usuários do dispositivo.

### Fatores de forma de hardware do dispositivo

Assim como há uma ampla gama de cenários baseados em mobilidade, existe grande variedade de fatores de forma do dispositivo para dar suporte a esses cenários, conforme ilustra a Figura 1. A escolha do dispositivo depende da forma com que será utilizado.

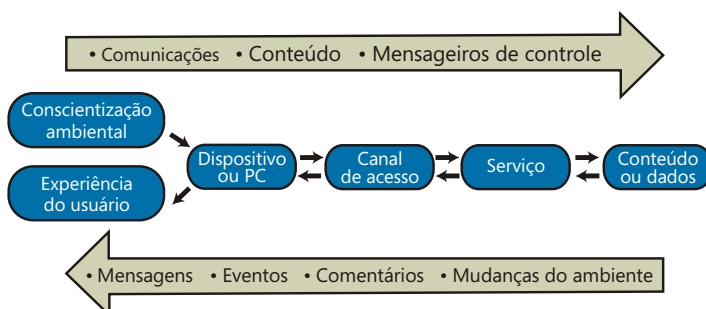
Por exemplo, no âmbito do consumidor de dispositivos pessoais, o espectro de hardware disponível varia, em termos gerais, dos dispositivos baseados em WM (*Windows Mobile*) de um lado, laptops do outro e dispositivos UMPC no meio. Quase sempre, os dispositivos WM são smartphones ou pocket PCs (em geral, com telas abaixo de 12,5 cm); os UMPCs são, em geral, companheiros digitais portáteis (telas que variam de 14,22 cm a 17,8 cm).

### Sensores para receber informações do ambiente imediato

Os dispositivos podem receber informações de grande variedade de elementos sensores, mencionados nesta seção. Alguns desses sensores fornecerão canais para que os usuários possam interagir com o software executado no dispositivo. Outros sensores do dispositivo fornecerão aplicativos, visualizando o contexto atual do usuário a qualquer tempo, para que o software possa ajustar-se adequadamente.

1. *Tecnologias de toque.* Alguns dispositivos utilizam tecnologias de toque para aprimorar a experiência do usuário. Dispositivos mais novos usam o "toque capacitivo" (não exige pressão para registrar o toque, diferentemente do "toque resistivo" encontrado em dispositivos antigos, que exigem o uso de uma caneta). Os dispositivos com "toque capacitivo" são mais fáceis de usar, mais precisos e ágeis. As antigas telas de toque nunca eram nítidas sob luz solar e dificultavam a visualização de mídia mais rica; as mais novas são, em geral, mais brilhantes, pois a sua superfície não está coberta com o fino filme necessário para o "toque resistivo". Outro avanço a ser mencionado é o multitouch (capacidade de manipular informações de mais de um dedo ao mesmo tempo), que permite aos usuários redimensionar uma janela abrindo ou fechando dois dedos sobre a tela. Em telas de toque, os usuários têm uma dificuldade comum: sentem falta de uma resposta tátil. Entretanto, os fabricantes de alguns aparelhos estão adicionando a tecnologia de feedback tátil encontrada nos controladores de jogos (por exemplo, produzir leve vibração quando se utiliza o teclado virtual em uma tela de toque). Assim, haverá uma resposta similar àquela que os usuários estão acostumados a receber dos teclados mecânicos convencionais.

**Figura 4:** Framework conceitual para aplicação da abordagem Software + Serviços a um grande número de dispositivos





2. **GPS.** Muitas soluções de mobilidade dependem do conhecimento da posição do usuário. Uma técnica comum para que o dispositivo determine a sua posição é o GPS (sistema de posicionamento global) que faz isso com base nas linhas de visão de três ou mais satélites (ou seja, o GPS não pode ser utilizado entre quatro paredes). Alguns serviços baseados em posicionamento na web utilizarão automaticamente as informações do GPS no dispositivo (quando disponível) para fornecer informações filtradas pela localização dos usuários (Live Search).

3. **Acelerômetro.** Alguns dispositivos têm acelerômetros internos. Uma utilização básica é a de detectar, automaticamente, a orientação do dispositivo, ou seja, modo retrato ou paisagem. Usos mais avançados do acelerômetro incluem reconhecimento do gesto, controle de mídia ou de jogos. No futuro, é bastante provável que os acelerômetros nos dispositivos poderão levar a cenários de controle sofisticados, similares àqueles do controle remoto do Nintendo Wii, construído com base em acelerômetro.

4. **Sensores para monitoração de saúde.** Como exemplos temos os sensores cardíacos e de glicemia.

5. **RFID.** Leitores, digitalizadores e impressoras RFID fazem parte de uma gama de dispositivos que utilizam tecnologias RFID em cenários principalmente corporativos como o gerenciamento da logística e da cadeia de suprimento. Os dispositivos com sensores RFID internos destinam-se a substituir uma geração de dispositivos mais antiga, que utiliza leitura de código de barras.

6. **Outros sensores.** Podemos citar, entre outros, os sensores de luz ambiente (para controlar o brilho da tela e conservar a vida da bateria) ou os sensores de proximidade (para desligar o visor quando o dispositivo estiver sendo utilizado como telefone).

## Software executado no dispositivo

Os produtos executados no dispositivo podem ser assim classificados:

1. Sistema operacional do dispositivo;
2. Plataforma de aplicativos: tempo de execução e ferramentas de projeto do aplicativo;
3. Navegador móvel: vem surgindo como uma plataforma de aplicativos em si, para dispositivos de consumo;
4. Aplicativos.

O sistema operacional deve ser escolhido com base na utilização do dispositivo, pois há uma compensação implícita entre gerenciar os recursos limitados do dispositivo e a riqueza dos aplicativos executados no dispositivo. Os dispositivos possuem diferentes necessidades: vamos analisar as pilhas de software de cada tipo de dispositivo representado na Figura 1.

**Pilha de software dos dispositivos internos.** O Windows Embedded CE é um kernel de sistema operacional permanente, de tempo real, 32 bits e memória protegida, compatível com ampla gama de arquiteturas de processador (ARM, MIPS, x86 ou SH4). É fornecido com um conjunto de aproximadamente 700 componentes, dos quais um subconjunto pode ser empacotado em imagens personalizadas. Por exemplo, uma única imagem do kernel pode ser montada para carregar o sistema operacional com aproximadamente 300 KB de memória RAM, mas também pode adicionar outras tecnologias à imagem, como servidor web, navegador, MP, suporte de rede, framework .Net Compact, os quais aumentam o tamanho da imagem do SO. Os dispositivos construídos com o Windows Embedded CE podem ser descentralizados ou ter algum tipo de visor. Além disso, os dispositivos podem ser abertos (ou seja, expõem as APIs do aplicativo) ou fechados (sem uma *developer story* (DS) de terceiros).

O Windows CE está disponível à comunidade geral de desenvolvimento de sistemas incorporados, para construírem os próprios dispositivos. Esse produto também é utilizado na Microsoft para construir soluções do Windows Mobile e do Microsoft Auto. O Windows Mobile é usado para dar potência a smartphones e PDAs; o Microsoft Auto é uma plataforma para o setor automotivo, para a construção de soluções avançadas para uso em veículos.

**Pilha de software para smartphones e PDAs.** O Windows Mobile escolhe o próprio conjunto de componentes do sistema operacional do Windows Embedded CE, com shell personalizado, tecnologias específicas do dispositivo (Connection Manager) e alguns aplicativos (Office Mobile). Os OEMs do Windows Mobile acrescentam aplicativos e serviços específicos próprios à imagem (plug-ins de tela, aplicativos como VoIP, jogos), mas não personalizam o conjunto de componentes na imagem WM básica. Um conjunto consistente de APIs resulta desse procedimento, oferecido em todos os dispositivos do WM: em teoria, os aplicativos escritos para um dispositivo do WM devem funcionar para todos os dispositivos desse ambiente. Na realidade, os dispositivos móveis apresentam grande variedade nas capacidades de hardware (opções de conectividade, tamanho da tela, resolução, orientação) e isso dificulta a construção de um aplicativo que funcione bem em todos os dispositivos, mesmo quando as APIs de base são as mesmas. A Figura 5 apresenta a variedade de opções de desenvolvimento para a construção de interfaces de aplicativos em smartphone Windows Mobile

**Pilha de software para Ultra-Mobile PCs.** Os dispositivos UMPC obtêm plena fidelidade da pilha de software: sistema operacional Windows Vista, .Net Framework como tempo de execução dos aplicativos gerenciados e IE7 como navegador. Os aplicativos para PC existentes não precisam ser reescritos para execução em UMPC, embora possam ser estendidos para compatibilidade com toque e tinta (*touch and ink*) - essas capacidades já estão incorporadas ao Windows Vista.

Entretanto, tudo isso tem custo: a vida útil da bateria (quase sempre, apenas 46 horas), pois os UMPCs não gerenciam os recursos do dispositivo no nível granular, da forma com que faz o Windows Mobile.

**Canais de acesso para dar suporte aos dispositivos.** Dispositivos móveis, como celulares, são principalmente utilizados para comunicações - em grande parte, hoje, para chamadas por voz - mas, também, para algumas outras formas de troca de

**Figura 5:** Opções de desenvolvimento para construção de interfaces de aplicativo em smartphone do Windows Mobile

Experiências de Smart Client		Ricas experiências interativas	Formas inteligentes	Formas básicas	
Experiência rica			Grande alcance		
Nativo	Gerenciado		Ajax	HTML	WAP
APIs Win32 (subconjunto das APIs Win32 do "desktop")	.Net Compact Framework & SQL Compact Edition	Silverlight (disponível para dispositivos em 2008)	AJAX/Atlas	Pocket IE ASP.NET	Pocket IE ASP.NET Controles móveis do ASP Net



mensagens como e-mail, mensagens instantâneas (IM) e SMS. Além desses serviços de comunicações básicas, espera-se que, no futuro, os dispositivos façam conexão com um conjunto muito mais rico de serviços de aplicativo. Embora exista um mercado relativamente grande para tais serviços, não é possível pressupor que os usuários desses serviços terão sempre dispositivos avançados preparados para a InHTML ternet, com plano de dados de operadora ou preparados para WiFi. Isto leva a distribuição do serviço a dispositivos por intermédio de outros canais também (como SMS ou voz, conforme anteriormente discutido sobre os serviços baseados em SMS para pagamentos pelo celular, no Quênia).

Abaixo estão alguns exemplos dos canais de rede pelos quais aplicativos e serviços podem ser distribuídos:

## 1. Voz

a. Reconhecimento de voz: esses serviços são acessados por meio de uma chamada telefônica e, na outra extremidade, há um software de reconhecimento de voz. A Figura 6 mostra como o Microsoft Office Communications Server pode ser usado para distribuir aplicativos preparados para o reconhecimento de voz, cujo acesso pode ser feito por meio dos serviços do aplicativo de telefonia ou de canais alternativos. Como exemplo de tal aplicativo móvel, veja os dispositivos do Live Search on Windows Mobile, que podem ser acessados por meio de uma interface de voz do U.S. Voice, software de reconhecimento que converte a voz dos usuários na sequência de consulta de pesquisa; os resultados são exibidos como habitualmente.

## 2. Troca de mensagens

a. SMS: atualmente, alguns serviços básicos são oferecidos pelo SMS (atualizações do mercado acionário, alertas e, agora, em alguns países, pesquisa na Internet). Por exemplo, a Microsoft Research realizou um projeto na Índia, construindo uma solução preparada para SMS para uma cooperativa de cana de açúcar. Os fazendeiros utilizam os seus telefones para obter informações (preço de mercado, por exemplo), fazendo os pedidos por mensagens de SMS. As respostas também são enviadas por SMS. A Microsoft Research disponibilizou o conjunto de ferramentas utilizado neste projeto como uma solução compartilhada no CodePlex. (Vide referências: SMS Server Toolkit.)

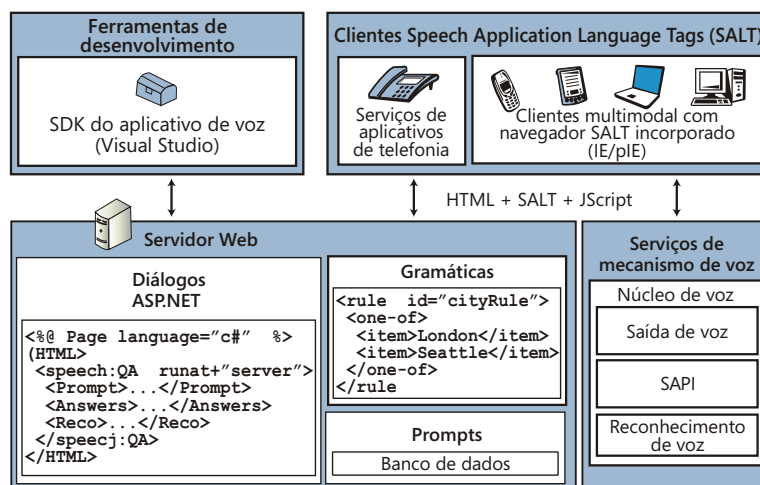
## 3. Internet

a. WiFi: este recurso trabalha bem com dispositivos equipados para WiFi, desde que nas proximidades de um ponto de acesso à Internet sem fio.  
b. Plano de dados móveis: em geral, trata-se de serviço prêmio, oferecido por operadoras de rede móvel, para acesso à Internet pela rede celular da própria operadora.

## 4. Conexão com outros dispositivos por meio das tecnologias P2P (ponto a ponto)

a. Alguns dispositivos podem trocar informações com outro dispositivo, diretamente, sem que seja preciso passar por servidor central. As considerações arquiteturais, como descoberta e protocolo de comunicação, podem ser realizadas de duas formas:  
i. Um servidor de índice central faz o trabalho de broker da conexão: por exemplo, considere o Xbox 360 e o Xbox Live. Quando o usuário se conecta no Xbox 360, pode juntar-se a um grupo de até 16 jogadores, participando de uma única sessão. Ainda que o serviço Live faça o acompanhamento de quem está on-line e, ainda, o trabalho de broker da conexão inicial no grupo, todas as mensagens posteriores entre os

**Figura 6:** Uso de tecnologias de voz no Microsoft Office Communications Server para distribuição de serviços pela Internet, preparados para voz.



consoles Xbox 360 serão direcionadas por meio das tecnologias P2P, e não por intermédio do servidor;

ii. Sem servidor de índice central: a descoberta de dispositivos próximos e o protocolo de comunicação entre eles ocorre diretamente, sem passar por servidor central. Por exemplo, o leitor de música Zune compartilha a música em execução com até três outros Zunes próximos, via tecnologias P2P.

## Serviços que dão suporte a dispositivos

Além dos serviços básicos de comunicação, espera-se que, no futuro, os dispositivos façam conexão com um rico conjunto de serviços na Internet. Muito provavelmente, esses serviços serão arquitetados em três camadas:

1. Serviços de aplicativos e soluções. Oferecem suporte específico a um conjunto de cenários (como saúde ou CRM).
2. Vinculados a serviços de terceiros. Esses serviços oferecidos por outros provedores ficam vinculados aos serviços de aplicativos; por exemplo, uma solução de mobilidade para provedores de assistência médica, que utiliza os serviços de e-mail, atualização ou colaboração fornecidos por terceiros.
3. Utilitários/infra-estrutura/serviços de blocos de construção.

Os serviços das segunda e terceira camadas representam capacidades comuns, horizontais, que cobrem muitos e diferentes serviços de aplicativos. Alguns deles estão descritos a seguir, no contexto das soluções de mobilidade. No futuro, esses serviços seriam, de modo geral, fornecidos pelo provedor de plataforma, como o Windows Live Platform ou até mesmo pela operadora de rede móvel.

## Gerenciamento e segurança do dispositivo

Os vetores de ataque dos dispositivos são similares àqueles dos PCs em rede, exceto que os dispositivos têm maior probabilidade de serem roubados ou perdidos, e - quase sempre - é mais difícil protegê-los fisicamente devido à sua mobilidade (em contraste com PC instalado em mesa ou central de dados). Assim sendo, existem três principais áreas a serem consideradas nas questões de segurança dos dispositivos. A primeira trata da proteção do próprio dispositivo. A segunda, da proteção da rede, ou seja, garantir a confidencialidade e a integridade das mensagens. Neste ponto, várias questões de segurança podem ser discutidas nas camadas da pilha da rede (por exemplo, técnicas de modulação de rádio para fornecer segurança na transmissão do sinal sem fio, IPsec, etc.) A terceira área de segurança

trata da proteção dos aplicativos executados no dispositivo (ou na web, mas que podem ser acessados pelos dispositivos), descrita na seção sobre gerenciamento de identidades e acesso.

Em alguns casos, o gerenciamento de dispositivos é como aquele dos PCs. Por exemplo, os dispositivos precisam ser atualizados com patches (firmware e software), mídia ou aplicativos. As comunicações com os dispositivos precisam ser protegidas e, em alguns casos, medidas e pagas (para downloads comerciais). Entretanto, o gerenciamento de dispositivos difere em alguns aspectos fundamentais pois, quase sempre, os dispositivos são muito mais difíceis de proteger: dispositivos móveis podem ser facilmente apoiados fora de lugar e, em muitos casos, o acesso a eles não pode ser limitado.

Os serviços de gerenciamento precisam ser fornecidos para dispositivos que se conectam em rede, para responderem os seguintes tipos de perguntas.

- Administradores de rede: "Neste momento, quantos dispositivos há na rede? Quanto de banda larga? Quanto tempo? Quais são os tipos de dispositivos existentes?"
- Suporte técnico: "Qual é o histórico do seu dispositivo? Foi atualizado? Quais são os dados do seu dispositivo?"
- Administradores de segurança: "Quais dispositivos não foram atualizados com o perfil 'minha segurança'? E se eu aplicar esta diretiva? Quantos dispositivos estão em conformidade?"
- Usuário: "Acabei de perder o meu dispositivo e gostaria de salvar as informações pessoais que há nele. É possível excluir os dados remotamente?"

Observe-se que bloquear um dispositivo não é o bastante, caso haja um cartão de armazenamento de 2 GB instalado, com informações confidenciais, no momento da perda. O Windows Mobile 6 trata desse problema criptografando os dados do cartão de armazenamento de modo que possa ser lido apenas pelo dispositivo que realizou a escrita.

### **Gerenciamento de identidades e acesso**

Os aplicativos executados nos vários dispositivos conectados em rede precisam de uma forma de compartilhar credenciais entre si, bem como com os serviços de back-end com os quais se conectam. Na medida em que os cenários para dispositivos tornam-se mais sofisticados, isso exigirá credenciais universalmente reconhecidas (por exemplo, a identidade do usuário e, em alguns cenários, também a identidade do dispositivo de origem). Atualmente, identificam-se os celulares pelo número de telefone. Embora os smartphones permitam ao usuário conectar-se com serviços de back-end on-line, tais serviços exigem, de modo geral, que o usuário faça a própria autenticação de várias outras formas - que nada têm a ver com o número do telefone, como o endereço de e-mail ou outras credenciais baseadas na web. No futuro, com a proliferação dos dispositivos, também crescerão os serviços de apoio a eles e um único identificador universal (talvez, conceitualmente similar do Live ID de hoje) poderá resolver o problema de autenticação. Entretanto, outras novas complicações surgirão: como será feita a conexão da identidade por meio dos dispositivos e serviços baseados na web? Em que ponto serão definidos os limites de confiança?

Ainda que muito tenha sido feito para proteger redes e dispositivos, é preciso ter um conjunto diferente de tecnologias para intermediar a confiança entre os aplicativos executados nesses dispositivos (e entre os serviços por eles conectados): tecnologias para identidades federadas. Essas tecnologias ajudam o usuário a gerenciar várias identidades digitais e controlar a quantidade de informações pessoais compartilhada com outros dispositivos e serviços. Cada uma dessas identidades seria construída em torno de um conjunto de declarações, expressões de confiança de uma parte certificadora - um ou mais

serviços de identidade na nuvem agindo como intermediários de confiança, emitindo declarações (ou expressões) de confiança incorporadas em tokens de segurança.

### **Ponto de reunião e presença**

Os serviços de presença móvel disponibilizam o contexto móvel dos usuários às suas redes sociais. Os dados de contexto, como local, inatividade, perfil do dispositivo (volume do toque, vibração) e informações de calendário deveriam estar disponíveis de outros dispositivos móveis ou por meio de gadget/widget/identificador incorporado no blog ou na página web do usuário. Pode ser exibido como lista (aumentando uma lista de contatos) ou mapa. Em resumo, a presença móvel deve permitir que as redes sociais do usuário saibam de sua disponibilidade e qual o modo preferido de comunicação, nesse momento. Essas informações não devem ser específicas da operadora móvel de determinado usuário, pois é improvável que todos os membros das redes sociais desse usuário sejam clientes da mesma operadora móvel. Essa informação presencial, de modo ideal, deveria conectar-se a um backbone de comunicações unificado, que combine as diferentes formas pelas quais as mensagens podem ser trocadas.

### **Serviços de base local**

No futuro, diferentemente dos PCs, é provável que a maior parte dos dispositivos dos consumidores informe a sua posição exata possivelmente por tecnologias similares ao GPS. Isso revela a possibilidade de ampla variedade de serviços na nuvem, os quais podem utilizar essas informações para apresentar ao usuário o conteúdo adequado daquele local. O crescimento dos sistemas de informações geográficas (GIS) on-line, com interfaces publicadas para armazenamento de dados e metadados espaciais associados, está viabilizando essa tendência. Alguns desses sistemas também estão marcados com conteúdo adicional gerado pelo usuário, identificado por local. Um serviço de base local faria uso das coordenadas geográficas do usuário como índice ou filtro do GIS, para recuperar as informações corretas. Esses serviços podem incluir pesquisa local, navegação e serviços emergenciais, rastreamento de crianças/animais de estimação/objetos de valor, jogos móveis para vários jogadores, localização de pessoas nas redes sociais do usuário, gerenciamento de logística/transporte, etc.

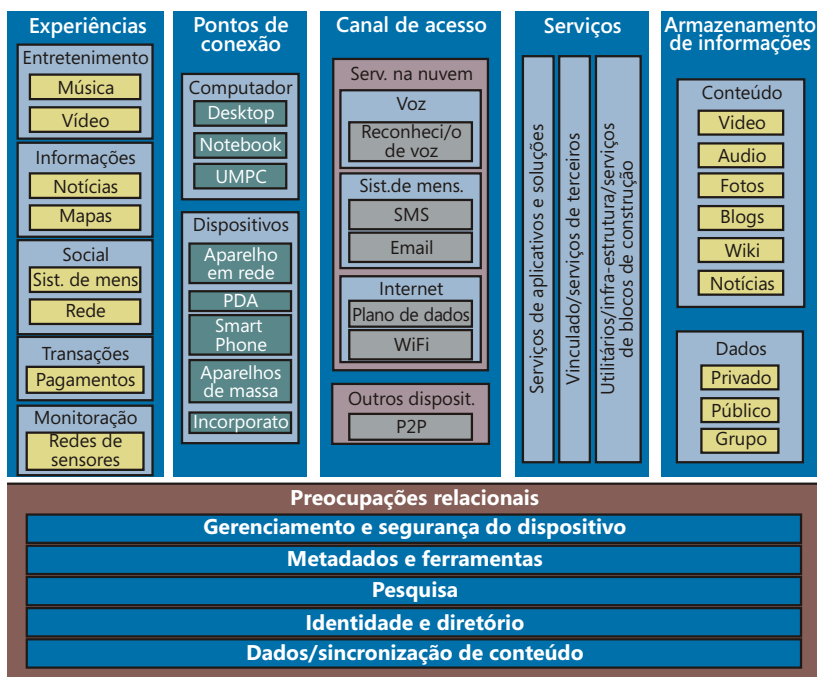
### **Pesquisa móvel e serviços de publicidade**

De determinadas formas, a pesquisa na web com base em dispositivo móvel não é muito diferente daquela feita com base em PC. Uma análise dos registros de pesquisa do Google apresentada em "*Deciphering Trends In Mobile Search*" (Decifrando tendências em pesquisa móvel) revela que apesar das limitações das técnicas de entrada, o número médio de palavras da consulta de uma pesquisa não mudou muito se comparado àquele dos celulares, PDAs e PCs. (Vide referências)

Entretanto, de outras formas, a pesquisa móvel tem potencial para ser mais dinâmica do que aquela feita em PC. Os dispositivos podem conhecer melhor o contexto atual do usuário (local). Atualmente, as ferramentas de busca fazem o processamento das consultas contra um índice construído pesquisando a web. O potencial da pesquisa móvel é que o contexto adicional disponível para o dispositivo poderia também ser usado pela ferramenta de busca no momento da formulação da resposta. Por exemplo, os resultados da pesquisa poderiam ser filtrados de acordo com a posição atual ou a lista de vínculos patrocinados poderia incluir um cupom de uma loja do local.

Como exemplo disso, agora, o Live Search do Windows Mobile inclui entrada de voz (ainda na versão beta), preços de combustível e horário de funcionamento do comércio. O serviço também pode usar dados de GPS em telefones preparados para tal, para fornecer pesquisa local, de acordo com a posição.

**Figura 7:** Visão resumida das opções de desenvolvimento para a construção de soluções de mobilidade



## Armazenamento, distribuição de conteúdo e serviços de gerenciamento de conteúdo.

Como mencionado na seção sobre tendências sociais, a proliferação de dispositivos está motivando uma explosão no volume de conteúdo gerado e este precisa ser armazenado fora do dispositivo. Em decorrência disso, precisamos ter serviços de armazenamento para fazer o backup dos dispositivos, para que as redes de distribuição de conteúdo possam movimentar os dados e, além disso, para que os serviços de gerenciamento de conteúdo possam organizar o material recém-adquirido para que este possa ser "encontrado".

A organização do conteúdo implica taxonomia – a qual talvez seja explicitamente definida, mas que provavelmente será mais emergente com base na marcação do conteúdo com um snapshot do contexto do usuário no momento em que conteúdo foi criado. Este contexto pode ser o local, a hora, um evento, etc. – qualquer coisa já informada ao dispositivo e automaticamente registrada.

## Serviços de alerta

Como o volume de conteúdo disponível on-line continua a crescer, os usuários precisam filtrar os sinais de informações que recebem. Uma forma de fazer isso é assinar um serviço de alerta particular, como o Windows Live Alerts. Esses alertas podem ser recebidos nos dispositivos móveis como mensagens SMS. Os usuários também podem incorporar um gadget para ler os alertas nos seus sites.

## Serviços de sincronização

Na medida em que os usuários se envolvem com um mundo de dispositivos, o conteúdo fica cada vez mais espalhado pelos PCs em casa, no trabalho, em serviços on-line e, agora, nos celulares. Uma parte importante das soluções de mobilidade ponta a ponta incluirá os serviços de sincronização, que resolvem esse problema de qualquer conteúdo, por intermédio de qualquer protocolo e em qualquer dispositivo ou PC. Esses serviços de sincronização precisariam ter condições de tratar questões sutis em cenários que envolvem o uso de cache, operação offline, compartilhamento e transferência entre áreas

de serviço (roaming). O Microsoft Sync Framework é uma forma de construir esses serviços: permite aos desenvolvedores de aplicativos acrescentar capacidades de sincronização, com facilidade, em qualquer aplicativo ou serviço. Isso é possível por meio de um modelo de provedor que pode ser estendido para dar suporte a cenários como sincronização de bancos de dados relacionais, sistemas de arquivo, listas, dispositivos, PIM, música, vídeo, etc.

## Resumo

A Figura 7 apresenta uma visão resumida de todas as opções de desenvolvimento disponíveis aos arquitetos de soluções de mobilidade, com exemplos de experiências específicas, pontos de conexão e canais de acesso. Existem várias preocupações relacionais para dispositivos, serviços e tecnologias de acesso como gerenciar identidades e confiança através dessas diferentes camadas.

A abordagem do arquiteto de soluções de mobilidade deve pensar em equilibrar a necessidade de alcance amplo e escala para o seu público-alvo contra a necessidade de distribuir ricas experiências com as quais os usuários podem se conectar. As soluções ideais combinarão o melhor da web com os melhores aspectos dos dispositivos.

## Referências

### Deciphering Trends in Mobile Search

<http://www.maryamkamvar.com/publications/KamvarBalujaComputerMagazine.pdf>

### "Mobile Phones As Mass Media: Models For Content Distribution",

por Alan Moore

<http://www.masternewmedia.org/media/mobile-phones/mobile-phones-as-mass-media-white-paper-part-2-20070711.htm>

### "Safaricom: On a Tear in Africa", por Jack Ewing, 27/08/2007,

BusinessWeek

[http://www.businessweek.com/globalbiz/content/aug2007/gb20070827\\_543072.htm](http://www.businessweek.com/globalbiz/content/aug2007/gb20070827_543072.htm)

### SMS Server Toolkit

<http://research.microsoft.com/research/downloads/details/9190f48f-6e3d-4ee8-b4a9-b346db76be1d/details.aspx>

### "Upwardly Mobile in Africa", por Jack Ewing, 13/09/2007,

BusinessWeek

[http://www.businessweek.com/globalbiz/content/sep2007/gb20070913\\_705733.htm](http://www.businessweek.com/globalbiz/content/sep2007/gb20070913_705733.htm)

## Sobre o autor

**Atanu Banerjee** é integrante da equipe de arquitetura de plataforma da Microsoft, onde trabalha em arquitetura para as soluções da próxima geração. Ingressou na Microsoft, vindo da i2 Technologies, onde desempenhou várias funções, por mais de sete anos, inclusive arquiteto-chefe de uma linha de produtos de gerenciamento de cadeia de suprimento, gerente de desenvolvimento, arquiteto de produtos, líder de equipe e desenvolvedor de software. Durante esse tempo, escreveu grande volume de código, projetou novas soluções e trabalhou com alguns clientes de grande porte do setor de manufatura. Antes da i2, Atanu trabalhou no grupo de sistemas de controle avançado da Aspen Technologies, na área de projetos e implementação de sistema de controle preditivo, com base no modelo para o setor de processo. Em 1996, Atanu recebeu o título de Ph.D. da Georgia Tech. Reside em Redmond, Washington com a sua mulher e o seu filho de seis anos.

# Melhores Práticas: como Estender Aplicativos Corporativos aos Dispositivos Móveis

por Kulathumani Hariharan



## Resumo

Estender os aplicativos corporativos aos dispositivos móveis torna-se, a cada dia, uma prioridade para que as organizações possam otimizar a sua força de trabalho. Para obter o resultado desejado de uma solução móvel, robusta, escalável, segura e ágil, com suporte de plataforma de vários dispositivos, muitos componentes precisam trabalhar juntos. O desafio está em estender, de modo transparente, vários tipos de aplicativos corporativos, muitos deles baseados em grande variedade de tecnologias e plataformas, aos dispositivos móveis. Este artigo descreve os componentes necessários para estender um aplicativo corporativo genérico aos dispositivos móveis, cobre algumas das melhores práticas e recomendações e, ainda, descreve um estudo de caso baseado em uma implementação do mundo real.

## Síntese da solução móvel

Ao estender aplicativos corporativos aos dispositivos móveis, muitas soluções exigem uma abordagem de três camadas: o aplicativo corporativo em si, o middleware móvel e o aplicativo cliente móvel.

**Aplicativo corporativo.** Existem muitos tipos de aplicativos corporativos que podem ser estendidos aos dispositivos móveis como CRM (gerenciamento de relações com clientes), ERP (planejamento de recursos empresariais) e BI (inteligência do negócio).

**Middleware móvel.** Considerando que a maior parte dos aplicativos corporativos não tem uma forma direta de trabalhar com dispositivos, o papel do middleware móvel (denominação utilizada neste artigo) é fundamental. Algumas das características importantes desta camada são segurança, sincronização de dados, gerenciamento de dispositivos e o necessário suporte para vários dispositivos.

**Aplicativo cliente móvel.** O aplicativo cliente móvel é, obviamente, o software que será executado no dispositivo. Nesta camada, existem várias considerações a serem feitas, inclusive disponibilidade de dados, comunicação com o middleware, utilização de recurso local e armazenamento local de dados. Além disso, muitos fatores do negócio precisam ser considerados. Por exemplo, quem são os usuários-alvo? Qual a importância de disponibilizar os dados mais recentes? Existem restrições para armazenar os dados no dispositivo? Quais provisões existem caso não haja conectividade de rede?

Ao selecionar a plataforma para o dispositivo, temos três opções principais:

- *Aplicativos on-line* (também conhecidos como thin client). Este é um software cliente, normalmente um navegador, utilizado quando a conectividade pode ser garantida. Sem conexão, o aplicativo móvel não funciona.
- *Aplicativos off-line* (também conhecidos como thick client). Este é o software cliente instalado localmente no dispositivo que mantém todos os dados necessários durante a duração da maioria das operações, com sincronização ao final de cada dia ou em intervalos pré-configurados.
- *Aplicativos ocasionalmente conectados* (também conhecidos como smart client). Este é o software cliente instalado localmente, similar ao modelo offline, mas o aplicativo pode atualizar e renovar os dados em qualquer momento. A frequência da renovação dos dados depende da criticalidade do aplicativo.

Agora, utilizando as três camadas acima como referência, vamos examinar o que isto significa para a automação da força de vendas/automação da força de campo (SFA/FFA).

## Como estender um aplicativo SFA/FFA baseado em produto ao dispositivo móvel

Em geral, a SFA/FFA baseada em produto faz parte de um aplicativo CRM ou ERP. Neste tipo de aplicativo, é comum não existir uma solução para dispositivos móveis. De modo típico, o front-end do lado do servidor do aplicativo baseia-se na web ou em aplicativo cliente rico, cujo suporte é feito por um banco de dados relacional com armazenamento de dados de grande porte que atenda a toda a organização. Talvez existam algumas restrições quanto ao acesso a este banco de dados, inclusive os seguintes desafios:

- modificações no esquema para dar suporte à extensão móvel – quase sempre é pouco o que se pode fazer (ou deveria ser feito) para modificar o esquema de suporte dos aplicativos móveis;
- acesso aos dados diretamente do banco de dados, e atualização das tabelas a partir do dispositivo móvel – freqüentemente, existem várias camadas de comunicação a serem atravessadas e não é possível acessar o banco de dados, diretamente do dispositivo;
- entendimento do esquema de armazenamento de dados – os esquemas desses tipos de aplicativos são projetados para serem estendidos e, por isso, podem ser de grande porte e pesados;
- projeto de uma área de passagem com uma estrutura de esquema similar àquela do back-end para que os dados sejam transferidos aos dispositivos móveis – criar um ambiente similar para desenvolvimento e passagem pode ser bastante difícil.



## Introdução à solução

Ao estender um aplicativo SFA/FFA baseado em produto a dispositivos móveis, os desafios mencionados na seção anterior precisam ser abordados efetivamente. A arquitetura precisa considerar os componentes que trabalham em tandem para abordar esses desafios.

A Figura 1 apresenta um modelo proposto para o aplicativo smart client; a Tabela 1 relaciona os componentes de middleware e aplicativo cliente, com descrição sucinta da sua função como parte da solução. Observe que a lista de componentes é um superset opcional e implementações específicas talvez não tenham todos os componentes.

## Melhores práticas

Pela prática, descobrimos que os seguintes itens representam as melhores práticas para a criação de aplicativos que se baseiam no modelo descrito na Figura 1 e na Tabela 1.

1. Usar os *stored procedures* do banco de dados para escrever código encapsulado (*wrapper code*) para acesso mais rápido aos dados.
2. Para dados específicos, estes deveriam ser preenchidos utilizando as visões do banco de dados para saída mais rápida para o dispositivo.
3. A infra-estrutura do banco de dados de passagem (*staging database*) poderia fazer parte do principal servidor de banco de dados, para uma resposta mais rápida aos dispositivos móveis (o benefício depende do número de usuários e da carga do servidor em qualquer momento).
4. Embora estendendo os dados do *back-end* ao banco de dados de passagem, incluir apenas as colunas e os campos necessários no dispositivo móvel, pois estes devem ser implantados no dispositivo. Isto ajudará a respeitar as suas restrições de tamanho.
5. O banco de dados de passagem só deveria ter dados por um período limitado (dois meses, por exemplo) com arquivos programados normalmente; restringir o tamanho do banco de dados reduzirá o tempo de pesquisa.
6. Usar o número da versão para rastrear os registros facilmente, quando for preciso fazer atualizações de delta durante a sincronização.

7. Usar tabelas de mapeamento no banco de dados de passagem para rastrear a versão do registro e facilitar a resolução de conflitos; por exemplo, para aplicar uma regra de conflito, anulando um registro de transação com uma mudança do lado do servidor, mesmo quando várias mudanças são feitas no lado do cliente. (Tabela de mapeamento é aquela do banco de dados de passagem que contém a chave principal da tabela do banco de dados de back-end e a chave principal do registro do banco de dados do dispositivo.)

8. O serviço de troca de dados deveria ser um processo recorrente e configurável no console do middleware para tratar mudanças contínuas no sistema de back-end e no banco de dados de passagem (acionado no lado do cliente), criando um método de trabalho assíncrono.

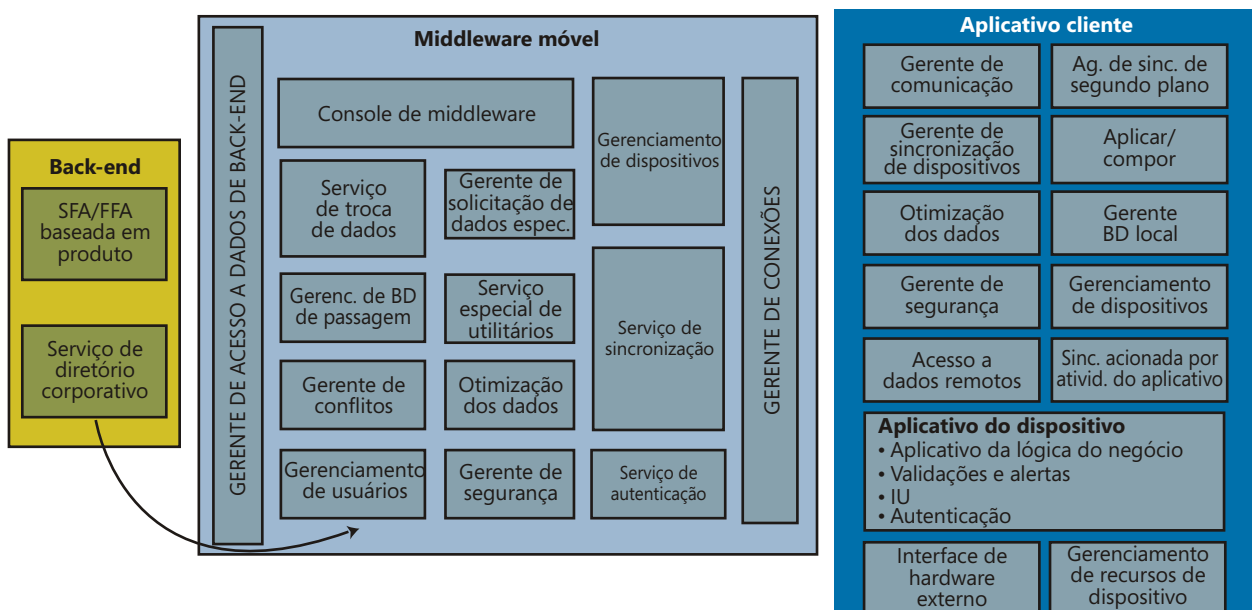
9. Atualizar apenas os detalhes necessários do usuário no middleware, vinculando-os ao serviço de diretório corporativo para autenticação e outros dados de usuário. Este procedimento reduziria problemas de sincronização das informações do usuário, entre o diretório corporativo e o middleware.

10. Não armazenar senhas no banco de dados de passagem; em lugar disso, consultar o serviço de diretório corporativo durante a autenticação. Este procedimento elimina problemas de sincronismo causados pela não-atualização da senha no lado do servidor no middleware.

11. Durante a sincronização, o aplicativo cliente deveria primeiramente verificar as atualizações do aplicativo enviando a sua versão atual e fazendo o download da versão mais recente, se aplicável; este é um mecanismo otimizado para gerenciamento de versão do aplicativo.

12. Armazenar o perfil do dispositivo do usuário (plataformas de dispositivo e versões de SO) no banco de dados do usuário e transferir as atualizações da versão ao dispositivo adequadamente, enviando diferentes compilações para diferentes usuários.

Figura 1: Componentes para estender uma SFA/FFA baseada em produto ao dispositivo móvel - abordagem Smart Client



13. Atualizar três tabelas: fila de entrada, fila de saída e fila de saída definida pelo usuário para gerenciamento de sincronização, simplificando o gerenciamento da fila e otimizando esse processo.

14. O gerente de comunicação pode ser criado para tentar tipos alternativos de conectividade quando o método principal não estiver disponível, para que a opção de conectividade de rede mais eficiente e disponível seja utilizada. Por exemplo, se não houver LAN sem fio disponível, o aplicativo tenta conexão com a rede GPRS (*General Packet Radio Service*); se esta também não estiver disponível, o cliente não sai de sincronia.

15. O intervalo de sincronização de segundo plano deve considerar o número de usuários/usuários simultâneos que o servidor pode absorver. Essas considerações ajudarão a reduzir a carga do servidor para aceitar a quantidade máxima de usuários móveis.

16. O gerente de sincronização de dispositivos precisa apenas enviar o nome do usuário, a versão do aplicativo do dispositivo, o tempo de intervalo da sincronização e as atualizações de delta durante a sincronização. Para reduzir o número de sincronizações simultâneas, o middleware deve responder se há atualização

**Tabela 1:** Componentes de middleware e aplicativo cliente

## Middleware

Componente	Descrição
Gerente de acesso a dados de back-end	<ul style="list-style-type: none"> <li>Compreende o código encapsulado (<i>wrapper code</i>) para chamar as APIs de back-end para inserir, atualizar e excluir dados de back-end.</li> </ul>
Gerente de solicitação de dados específicos	<ul style="list-style-type: none"> <li>Métodos pré-configurados que retornam dados específicos, em tempo real, para o dispositivo móvel.</li> </ul>
Gerenciamento de banco de dados de passagem	<ul style="list-style-type: none"> <li>Gerencia os dados no banco de dados de passagem;</li> <li>Armazena uma réplica de todas as tabelas de transação com colunas e dados específicos dos usuários móveis;</li> <li>Processa o arquivamento dos dados;</li> <li>Processa as filas de entrada/saída;</li> <li>Libera espaço no banco de dados.</li> </ul>
Gerente de conflitos	<ul style="list-style-type: none"> <li>Gerencia conflitos de dados e, ao mesmo tempo, faz a sincronização com o banco de dados de back-end;</li> <li>Monitora conflitos do tipo servidor vence x cliente vence, compartilhando registros entre vários usuários atualizando um registro no dispositivo quando já excluído do back-end e assim por diante.</li> </ul>
Serviço de troca de dados	<ul style="list-style-type: none"> <li>As composições mudam do back-end para um usuário específico, a serem enviada para o dispositivo móvel;</li> <li>Aplica mudanças do dispositivo móvel para o back-end utilizando o gerente de acesso a dados de back-end;</li> <li>Executa um serviço recorrente, regularmente executado após determinado intervalo (configurável);</li> <li>Envia dados compostos para a fila de saída;</li> <li>Coleta os dados da fila de entrada para aplicação ao back-end.</li> </ul>
Console de middleware	<ul style="list-style-type: none"> <li>Interface do usuário para configuração do middleware;</li> <li>Usada para configurar módulos como gerenciamento de usuários, subconjunto de dados, gerenciamento de sincronização, de dispositivo, etc.</li> </ul>
Serviço especial de utilitários	<ul style="list-style-type: none"> <li>Contém a lógica do negócio para realizar atividades específicas que não fazem parte do núcleo de middleware mas, sim, da solução móvel – por exemplo, um serviço de base local que obtém atualizações do local capturadas do dispositivo e usa um sistema de informações geográficas para mapear latitude e longitude exibindo os resultados em relatórios;</li> <li>Opcional, dirigida pelas exigências do negócio.</li> </ul>
Gerenciamento de usuários	<ul style="list-style-type: none"> <li>Gerencia os usuários móveis por meio dos dispositivos móveis;</li> <li>A lista de usuários pode estar vinculada a serviços de diretório corporativo, utilizando a mesma autenticação nos dispositivos móveis.</li> </ul>
Gerenciamento de dispositivos	<ul style="list-style-type: none"> <li>Gerencia os dispositivos do servidor;</li> <li>Envia novas atualizações do aplicativo para os dispositivos móveis;</li> <li>Visualiza os registros do aplicativo;</li> <li>Explora questões referentes ao dispositivo;</li> <li>Gerencia a aplicação das diretivas de segurança corporativas, como eliminar dados dos dispositivos não sincronizados por determinado número de dias.</li> </ul>
Otimização dos dados	<ul style="list-style-type: none"> <li>Otimiza a forma com que os dados são enviados ao dispositivo móvel;</li> <li>Compacta os dados e escolhe o melhor método para enviar os dados com base na velocidade da conexão ou no tipo de conectividade;</li> <li>Descompacta os dados provenientes do dispositivo móvel.</li> </ul>
Gerente de segurança	<ul style="list-style-type: none"> <li>Codifica os dados enviados ao dispositivo móvel;</li> <li>Decodifica os dados provenientes do dispositivo móvel.</li> </ul>
Serviço de sincronização	<ul style="list-style-type: none"> <li>Componente básico do middleware;</li> <li>Os dados de entrada provenientes do dispositivo móvel são recebidos na fila de entrada e o dados de saída são transferidos da fila de saída ao dispositivo móvel;</li> <li>O serviço de troca de dados compõe mudanças para um determinado usuário na fila de saída e aplica as mudanças da fila de entrada provenientes do dispositivo móvel para o back-end;</li> <li>Sincronização de segundo plano do dispositivo; o serviço mantém uma fila definida pelo usuário e verifica os novos registros a serem enviados ao dispositivo móvel.</li> </ul>
Serviço de autenticação	<ul style="list-style-type: none"> <li>Autentica o usuário móvel durante o processo de login e a sincronização.</li> </ul>
Gerente de conexões	<ul style="list-style-type: none"> <li>Manipula várias conexões ao mesmo tempo, dos usuários móveis até o número máximo de usuários simultâneos.</li> </ul>

disponível e qual o horário da próxima, para sincronização, caso nenhuma atualização seja necessária.

17. A coluna de estado de registro deveria ser atualizada, no nível de registro, para composição de dados mais rápida durante a sincronização.

18. Os aplicativos deveriam ser projetados de tal forma para que, quando a potência da bateria estiver baixa, a prioridade do *thread* de segundo plano também ficaria definida em "baixa", reduzindo a utilização da CPU e ampliando a vida da bateria.

19. Desenvolver o emulador (se já não estiver disponível) para o tipo de dispositivo. Isto reduz os esforços durante as fases de desenvolvimento e testes. Por exemplo, usar o construtor de

plataforma da Microsoft para desenvolver um emulador Win CE que possa suportar os recursos exigidos pelo aplicativo.

20. Durante o desenvolvimento do aplicativo, desenvolver também aplicativo(s) de simulação para testar os dados de entrada/saída por meio dos periféricos conectados ao dispositivo.

21. Dados dummy obtidos do fabricante do dispositivo ou criados utilizando manuais do dispositivo serão fundamentais para o aplicativo de simulação.

22. Ao projetar os módulos dos periféricos, colocar funções específicas de hardware e aplicativo nas respectivas bibliotecas para que as mudanças dos periféricos possam ser realizadas sem afetar a biblioteca do aplicativo.

## Aplicativo cliente (cont)

Componente	Descrição
Gerente de comunicação	<ul style="list-style-type: none"> <li>Estabelece a conexão com a rede.</li> </ul>
Gerente de sincronização de dispositivos	<ul style="list-style-type: none"> <li>Executa a autenticação com o serviço correspondente;</li> <li>Envia e recebe dados do serviço de sincronização no middleware;</li> <li>Faz o download das atualizações do aplicativo e dos comandos de gerenciamento de dispositivos.</li> </ul>
Otimização dos dados	<ul style="list-style-type: none"> <li>Otimiza a forma com que os dados são enviados ao middleware;</li> <li>Compacta os dados e escolhe o melhor método para enviar os dados com base na velocidade da conexão e no tipo de conectividade;</li> <li>Descompacta os dados de entrada provenientes do middleware.</li> </ul>
Gerente de segurança	<ul style="list-style-type: none"> <li>Codifica os dados enviados ao middleware;</li> <li>Decodifica os dados provenientes do middleware.</li> </ul>
Aplicar/compor	<ul style="list-style-type: none"> <li>Aplica os dados de entrada;</li> <li>Compõe mudanças a serem enviados ao back-end.</li> </ul>
Agendador de sincronização de segundo plano	<ul style="list-style-type: none"> <li>Componente configurável, programa sincronização de segundo plano a partir do dispositivo móvel;</li> <li>Envia dados ao servidor em intervalos pré-configurados sem a intervenção do usuário (automaticamente).</li> </ul>
Acesso a dados remotos	<ul style="list-style-type: none"> <li>Chama os métodos definidos no gerente de solicitação de dados específicos para ter dados e tempo real no dispositivo móvel; se a conectividade não estiver disponível, serão utilizados os dados existentes no dispositivo.</li> </ul>
Gerente de banco de dados local	<ul style="list-style-type: none"> <li>Gerencia os dados no banco de dados do dispositivo;</li> <li>Aplica e compõe dados;</li> <li>Elimina dados temporários do banco de dados;</li> <li>Gerencia o estado dos registros;</li> <li>Gerencia os detalhes de configuração do dispositivo.</li> </ul>
Gerenciamento de dispositivos	<ul style="list-style-type: none"> <li>Executa comandos do middleware;</li> <li>Aplica atualizações do aplicativo;</li> <li>Bloqueia o aplicativo se o usuário digitar senha errada em um número específico de vezes;</li> <li>Envia registros ao middleware.</li> </ul>
Sincronização acionada por atividade do aplicativo	<ul style="list-style-type: none"> <li>Aciona a sincronização ao final de um fluxo completo do negócio e confirma se o back-end do cliente móvel está sincronizado.</li> </ul>
Aplicativo da lógica do negócio	<ul style="list-style-type: none"> <li>"Contrapino" de toda a solução móvel (aplicativo real usado pelos usuários móveis).</li> </ul>
Validações e alertas	<ul style="list-style-type: none"> <li>Valida a entrada do usuário com algumas regras do negócio;</li> <li>No caso de não conformidade, exibe alertas adequados na interface do usuário.</li> </ul>
Interface de hardware externo	<ul style="list-style-type: none"> <li>Interage com interfaces de hardware externo, vinculados ao dispositivo</li> <li>Necessário para o fluxo de dados para aplicativos externos, como Scanner de código de barras.</li> </ul>
Gerenciamento de recursos de dispositivo	<ul style="list-style-type: none"> <li>Tem acesso ao estado do recurso e exibe alertas se for necessário chamar a atenção do usuário.</li> </ul>
Interface do usuário	<ul style="list-style-type: none"> <li>Interface da solução móvel;</li> <li>As informações entradas pelo usuário através da IU são validadas e processadas pela lógica do negócio.</li> </ul>
Autenticação	<ul style="list-style-type: none"> <li>Autentica o usuário para o middleware, se houver conectividade de rede; caso contrário, faz a autenticação com as credenciais disponíveis no local.</li> </ul>

23. Nos casos em que o aplicativo compreende várias telas com partes e funcionalidade comuns da IU, deve-se projetar uma forma básica que contenha elementos comuns.

24. Sempre que possível, usar frames em lugar de várias formas para obter resposta mais rápida da IU.

25. As mensagens (como as de erro e alerta) deverão ser configuradas no middleware e transferidas para os dispositivos. Outros arquivos de configuração também deveriam ser configuráveis no middleware e, então, transferidos para o aplicativo do dispositivo para uso.

26. Consultas específicas ao banco de dados não deveriam ser permanentes; em lugar disso, as consultas deveriam ser coletadas no middleware por meio de um arquivo de configuração.

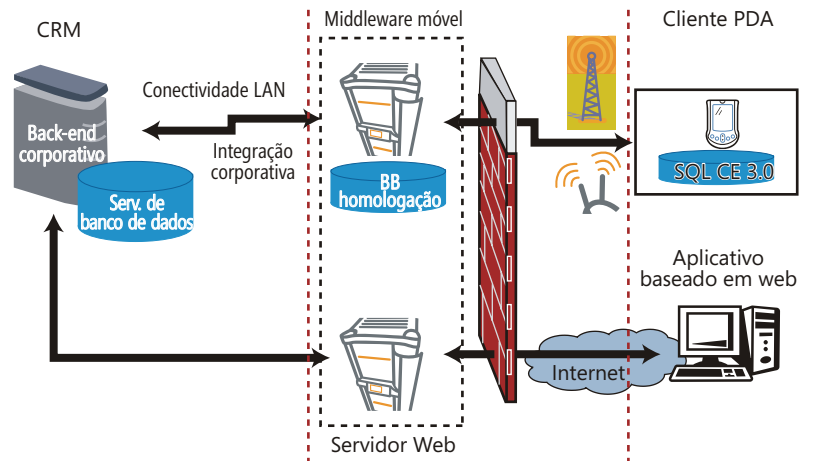
## Estudo de caso técnico: como estender um aplicativo CRM a dispositivos móveis de um sistema de gerenciamento de território

Uma indústria farmacêutica, de primeira linha, com presença em todo continente indiano, desejava aprimorar a eficiência de seus vendedores de campo. Os representantes comerciais visitam médicos em hospitais e clínicas para promover os medicamentos fabricados pela empresa, distribuir amostras e materiais promocionais e, ao final do dia, registram todos os detalhes por meio de um aplicativo CRM baseado na web.

A solução proposta, baseada em modelo portátil, tinha de cumprir as seguintes metas:

- aprimorar a eficiência e a eficácia dos vendedores;
- aumentar a produtividade em 10%;
- reduzir as despesas do processo manual (formulários e telefone);
- oferecer uma interface de usuário rápida e fácil, para ser prontamente adotada pelo usuário;
- transferir os dados mais recentes do dispositivo para o servidor, para que os gerentes possam acompanhar o trabalho dos vendedores;
- descarregar os dados mais recentes de estoques dos produtos no dispositivo, para receber os pedidos das farmácias (casos exista conectividade).

Figura 2: Diagrama de implantação do sistema de gerenciamento de território em dispositivo portátil



Esta solução baseada em modelo portátil foi projetada, desenvolvida e implementada em toda a Índia, sede da empresa farmacêutica.

### Definição do problema

Os representantes comerciais da empresa farmacêutica fazem cerca de dez visitas de campo/dia para encontrar médicos e promover os medicamentos, receber pedidos e distribuir amostras. Essas visitas demoram, em geral, de 2 a 10 minutos pois os médicos têm pouco tempo disponível. Nesse tempo limitado, o representante comercial precisa apresentar os medicamentos, obter comentários do médico e distribuir amostras, publicações e materiais promocionais. As informações coletadas são anotadas em papel pelo representante comercial o qual, ao final do dia, deve transferir todas as informações referentes às visitas para o aplicativo CRM baseado na web. O supervisor do representante de vendas visualiza os dados e aprova o trabalho do dia; a equipe de gerenciamento também pode analisar os dados e visualizar os relatórios.

Este processo manual de captura dos dados tem vários problemas. Os dados precisam ser anotados pelos representantes comerciais em papel e, depois, transferidos ao final do dia no sistema CRM pela web, o que resulta em erros e discrepâncias nos dados.

Figura 3: arquitetura da solução do sistema de gerenciamento de território com base em dispositivo portátil

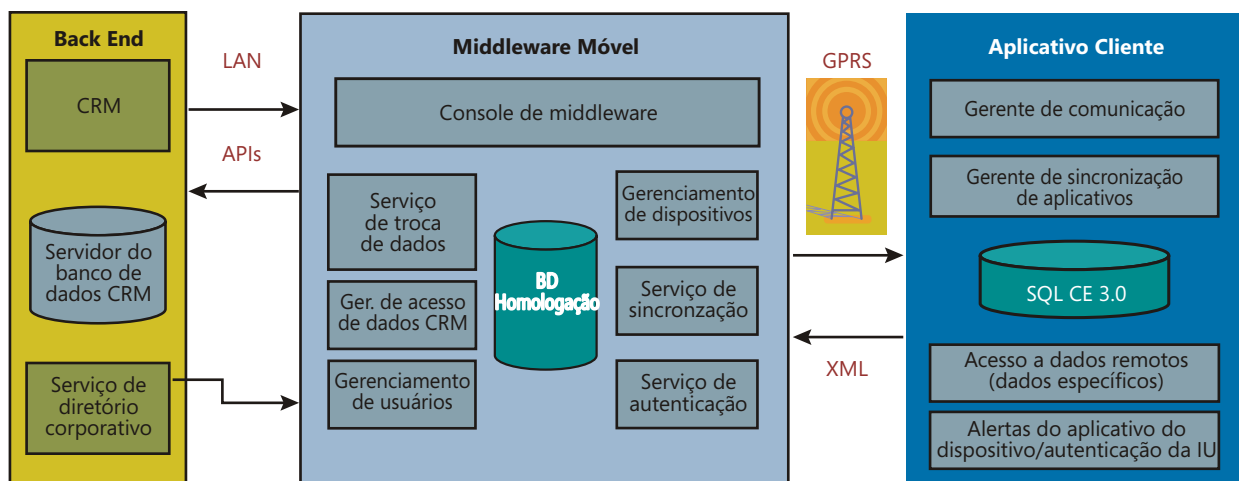




Tabela 2: síntese dos componentes middleware e aplicativo cliente na arquitetura de solução do estudo de caso do sistema de gerenciamento de território

## Middleware

Componente	Descrição
Gerente de acesso de dados CRM	<ul style="list-style-type: none"> <li>Executa as atualizações no banco de dados do sistema CRM (dados anotados pelo representante comercial durante as visitas em campo).</li> </ul>
Serviço de troca de dados	<ul style="list-style-type: none"> <li>Compõe as mudanças (fila de saída) do sistema de back-end do CRM, com base na subconfiguração (subsetting) dos dados de cada usuário (como agenda de visitas aos médicos/farmácias);</li> <li>É executado em intervalos de dois minutos após a conclusão do processo anterior;</li> <li>Aplica dados do banco de dados de passagem (fila de entrada) ao sistema de back-end do CRM utilizando o gerente de acesso a dados do CRM (dados anotados pelo representante comercial em campo).</li> </ul>
Gerenciamento de usuários	<ul style="list-style-type: none"> <li>Contém a lista de usuários vinculados ao serviço de diretório corporativo;</li> <li>Contém informações específicas do usuário, como as informações do dispositivo, último horário/data da sincronização, status de bloqueio do dispositivo, etc.</li> </ul>
Console de middleware	<ul style="list-style-type: none"> <li>IU do middleware;</li> <li>IU para gerenciamento do usuário e do dispositivo, permite visualizar registros de sincronização e registros do serviço de troca de dados.</li> </ul>
Gerenciamento de dispositivos	<ul style="list-style-type: none"> <li>Coloca novas compilações de aplicativos nos locais de compartilhamento predefinidos do middleware;</li> <li>Cria compilações para cuidar de diferentes tipos de dispositivos. (Observe que o componente correspondente no aplicativo cliente fica sob os cuidados do gerente de sincronização de dispositivo.)</li> </ul>
Serviço de sincronização	<ul style="list-style-type: none"> <li>Coloca os registros que chegam na fila de entrada;</li> <li>Transfere os registros a serem enviados da fila de saída;</li> <li>Administra a fila definida pelo usuário como uma tabela de índices para verificar se um determinado usuário tem muitos registros para fazer download.</li> </ul>
Serviço de autenticação	<ul style="list-style-type: none"> <li>Autentica o usuário conectando-o ao serviço de diretório corporativo.</li> </ul>

## Aplicativo cliente

Componente	Descrição
Gerente de comunicação	<ul style="list-style-type: none"> <li>Faz a conexão com o middleware para autenticação e sincronização de dados;</li> <li>Tenta fazer a conexão primeiramente com o middleware via Microsoft ActiveSync; se não disponível, tenta a conexão por GPRS/CDMA;</li> <li>Se nenhuma conexão estiver disponível, responde com mensagem informativa.</li> </ul>
Gerente de sincronização de dispositivos	<ul style="list-style-type: none"> <li>Compõe as mudanças no banco de dados do cliente em formato XML;</li> <li>Aplica dados de entrada ao banco de dados local do dispositivo;</li> <li>Coleta o arquivo de atualização do aplicativo cliente do local predefinido no middleware, se o arquivo não estiver disponível no dispositivo.</li> </ul>
Acesso a dados remotos	<ul style="list-style-type: none"> <li>Faz a conexão do middleware para obter listas atualizadas de visitas e campanhas.</li> </ul>
Aplicativo do dispositivo (alertas/IU/authenticação)	<ul style="list-style-type: none"> <li>Exibe alertas sobre detalhes de campanhas, visitas não realizadas, etc.;</li> <li>Faz a autenticação com o middleware se houver conectividade disponível ou autentica no local, caso contrário.</li> </ul>

As principais desvantagens do processo manual existentes eram:

- processo ineficiente na coleta dos dados;
- anotação de informações em papel: muito demorado;
- demora na obtenção das informações de campo.
- Suporte atrasado para consolidação e decisão em campo.
- Atraso no envio das últimas informações para o campo.
- Despesas incorridas com formulários, telefonemas, etc.
- Dados insuficientes para exposição adequada aos médicos e farmacêuticos.
- Representantes comerciais sem históricos das visitas.
- Fator de erro devido à entrada de dados com atraso.
- Falta de informações para o pessoal de campo sobre o status atual dos estoques de produtos.

## Solução

Considerando que a solução CRM já estava implementada, era preciso ter uma extensão para os dispositivos móveis, com o mesmo conjunto de recursos. O aplicativo portátil tinha de ser integrado ao back-end do CRM de modo transparente. Esta solução baseada em unidade portátil permite ao representante comercial anotar e transferir as informações do campo, com eficiência. O aplicativo móvel é executado na unidade portátil, levada pelo representante comercial quando em trabalho no campo, e com informações dos dados do cliente, do produto, das amostras, do histórico e agendas das visitas e estoques dos produtos. A Figura 2 (página anterior) apresenta a solução móvel implantada.

A solução apresenta quatro componentes principais: aplicativo portátil, banco de dados portátil, middleware e aplicativo CRM.

**Aplicativo portátil.** Este aplicativo é executado em dispositivos Windows Mobile e utilizado para anotar dados em campo. O aplicativo também tem um componente para sincronizar os dados da unidade portátil com o banco de dados do servidor no escritório.

**Banco de dados portátil.** Este é o banco de dados residente na unidade portátil. Contém dados específicos de cada representante comercial para possibilitar o seu trabalho em campo.

**Middleware.** Este componente, residente na extremidade corporativa, é utilizado para sincronizar os dados entre o banco de dados do CRM e o dispositivo portátil. O middleware utiliza um banco de dados de passagem que atua como o servidor do dispositivo portátil. Os dados de passagem são, então, sincronizados com o banco de dados do CRM utilizando APIs nativas, proporcionando uma integração transparente.

**Aplicativo CRM.** Este é o banco de dados de back-end, que armazena as informações da empresa. Os dados específicos de cada representante comercial são descarregados no banco de dados de passagem e, em seguida, no dispositivo do representante. Os dados atualizados do banco de dados da unidade portátil também são carregados primeiramente no banco de dados de passagem e, depois, atualizados no banco de dados do CRM.

A Figura 3 (página 14) apresenta a arquitetura da solução baseada nos componentes de nosso modelo genérico Smart Client (Figura 1). A Tabela 2 (página 15), relaciona os componentes do middleware e do aplicativo cliente com uma breve descrição de sua função como parte da extensão do aplicativo CRM ao móvel.

A arquitetura e o fluxo do processo podem ser assim resumidos:

- Os representantes comerciais possuem dispositivos portáteis com aplicativo móvel instalado.
- Os representantes comerciais criam uma programação semanal (pode também ser diária ou mensal) de visitas aos médicos, aprovada pelo supervisor.
- O representante faz a conexão com a rede corporativa por meio de GPRS e faz o download dos dados específicos do representante comercial no banco de dados da unidade portátil.
- O representante faz a visita aos médicos e anota as informações de vendas no aplicativo móvel. O representante também anota quais amostras ou materiais promocionais (se for o caso) foram entregues ao médico.
- Os dados são carregados e descarregados automaticamente, sem a intervenção do usuário; os dados mais recentes são colocados à disposição do usuário por meio da sincronização de segundo plano.
- O representante comercial acrescenta/atualiza as informações do médico, (substituição do profissional ou alteração das informações).
- O representante comercial reserva os produtos de novos pedidos de hospitais ou farmácias, com o auxílio da visualização em tempo real do status dos estoques.
- As despesas incorridas nas visitas a médicos e farmácias também são anotadas no aplicativo da unidade móvel.
- Os gerentes podem visualizar as atividades dos representantes comerciais por meio do componente que emite relatórios; também podem criar planos e estratégia do negócio após análise dos dados.
- Durante todo o dia, o gerente pode rastrear o comportamento no trabalho e os dados informados pelo representante comercial.

### *Abordagem dos principais desafios*

Durante o desenvolvimento, a equipe de projeto se defrontou com várias situações. Abaixo, estão descritos os desafios e a forma como foram abordados:

- Impossível estabelecer qualquer esquema de modificação para dar suporte à extensão móvel: este desafio foi resolvido criando-se uma área de passagem com estrutura de esquema similar ao banco de dados de back-end do CRM.
- Impossível atualizar diretamente nas tabelas a partir do dispositivo móvel: este desafio foi resolvido criando-se tabelas de mapeamento e utilizando código encapsulado para chamar as APIs do sistema de back-end do CRM.
- Entender a estrutura do esquema no qual os dados são armazenados: este desafio foi resolvido pela análise da documentação técnica do CRM e da estrutura de tabela do banco de dados do sistema CRM para entender cada campo e o seu uso.
- Projetar uma área de passagem com estrutura de esquema similar ao sistema CRM de back-end para que os dados sejam transferidos aos dispositivos móveis: este desafio foi resolvido da seguinte forma: primeiramente copiamos a estrutura do banco de dados de back-end do CRM para o banco de dados de passagem; depois, removemos os campos que não precisavam estar no dispositivo e, por último, criamos um serviço de troca de dados para que houvesse integração eficiente com o sistema de back-end do CRM.

**“OS REPRESENTANTES COMERCIAIS DA INDÚSTRIA FARMACÊUTICA FAZEM CERCA DE DEZ VISITAS DE CAMPO POR DIA A MÉDICOS PARA PROMOVER MEDICAMENTOS, PEGAR PEDIDOS E DISTRIBUIR AMOSTRAS. AS INFORMAÇÕES COLETADAS SÃO ANOTADAS EM PAPEL PELO REPRESENTANTE COMERCIAL; AO FINAL DO DIA, ESTE REPASSA TODAS AS INFORMAÇÕES REFERENTES ÀS VISITAS NO APLICATIVO CRM BASEADO NA WEB. ESTE PROCESSO MANUAL DE ANOTAR OS DADOS RESULTA EM ERROS E DISCREPÂNCIAS NOS DADOS”.**

### *Benefícios*

O sistema de automação da força de vendas tem sido bem-recebido pela indústria farmacêutica, especialmente pelos 2.000 representantes comerciais, usuários-alvo deste aplicativo.

A implementação do sistema definiu um processo de coleta de dados de campo eficiente e efetivo; aprimorou a comunicação entre os representantes comerciais e a gerência. Os representantes comerciais no campo e os gerentes no escritório têm acesso às informações necessárias - dados atualizados e de grande importância para eles - quer nas visitas de campo quer no planejamento de estratégias de marketing.

### *Sobre o autor*

**Kulathumani Hariharan** é um arquiteto de soluções sênior e trabalha para a Wireless & Pervasive Technologies Practice da TATA Consultancy Services. Especializa-se na arquitetura de soluções móveis corporativas e na definição da estratégia para adoção de middleware móvel para clientes corporativos móveis.

# Experiência para o Consumidor Conectado no Campo Automotivo

por Christoph Schittko, Darryl Hogan e Jon Box

## Resumo

O que diferencia um veículo de um monte de metal e quatro rodas? São os recursos. As montadoras estão sempre adicionando dispositivos de mídia mais novos, motores mais poderosos e assentos mais macios, tudo em nome do aprimoramento da experiência do motorista. No entanto, com todos esses aprimoramentos, as montadoras mal tocaram a superfície das capacidades do grande número de avanços em software e serviços colocados à disposição dos consumidores nos últimos anos.

Muito tem se falado dos serviços residentes na nuvem e dos respectivos aplicativos baseados no consumo de serviços. Apesar disso, os muitos artigos e arquiteturas de referência que procuraram demonstrar modelos para projetar esses sistemas assumiram uma abordagem idealística para o aplicativo, freqüentemente citando exemplos simples ou aplicando cenários não pragmáticos. Este artigo define uma arquitetura de solução prática, baseada em um cenário que se encontra na rotina diária. Pretendemos inspirar arquitetos a usarem a mesma abordagem na definição de soluções inovadoras para os problemas que enfrentam.

A arquitetura da solução aqui definida combina os serviços de plataforma reais, disponíveis neste momento, e os fabricados, que ajudam a refinar a solução. A solução demonstrará a aplicação de Software + Serviços (S+S) à arquitetura de aplicativos móveis, como forma de ampliar o estilo de vida digital além do desktop. Segurança, privacidade e arquitetura de dados serão abordadas em linhas gerais.

## Cenário

A família Woodson acabou de comprar um carro novo, com um laptop a bordo. Este PC atua como sistema de orientação e entretenimento, mas também já vem carregado com vários aplicativos de produtividade, como planejamento de viagens, lembretes e um gerente de lista. O software carregado no PC do automóvel não apenas executa processamento local, no dispositivo, mas também aproveita os serviços na nuvem para aprimorar as capacidades do dispositivo e para garantir que as informações apresentadas pelo software estão atualizadas.

Os Woodsons decidiram usar o carro novo na próxima viagem que a família fará. Mary Woodson além de mãe, é a coordenadora de eventos da turma. Mary usa o PC da família para planejar as suas aventuras utilizando uma versão baseada na web do aplicativo de planejamento de viagens instalado no computador fornecido com o carro. Mary planeja o itinerário que percorrerão e faz reservas para

hotéis e restaurantes nas paradas do caminho. Inadvertidamente, Mary usa um mashup que compreende vários serviços executados na nuvem. Todas as informações pertinentes à viagem ficam armazenadas remotamente em um local baseado na Internet, cujo acesso só é permitido à Mary e a quem ela designar.

**“A INTERNET ESTÁ NO INÍCIO DE SUA TRANSFORMAÇÃO PARA TORNAR-SE UMA PLATAFORMA DE SERVIÇOS NA NUVEM: PLATAFORMAS COMO O WINDOWS LIVE OFERECEM APIs DE GERENCIAMENTO DE PRESENÇA, ALERTAS, CONTATOS E CALENDÁRIO, ASSIM COMO MAPAS E INSTRUÇÕES; COMO OUTRO EXEMPLO, O BIZTALK SERVICES OFERECERÁ UMA SUBPLATAFORMA PÚBLICA, ALÉM DE ROTEAMENTO E DISTRIBUIÇÃO DE MENSAGENS”.**

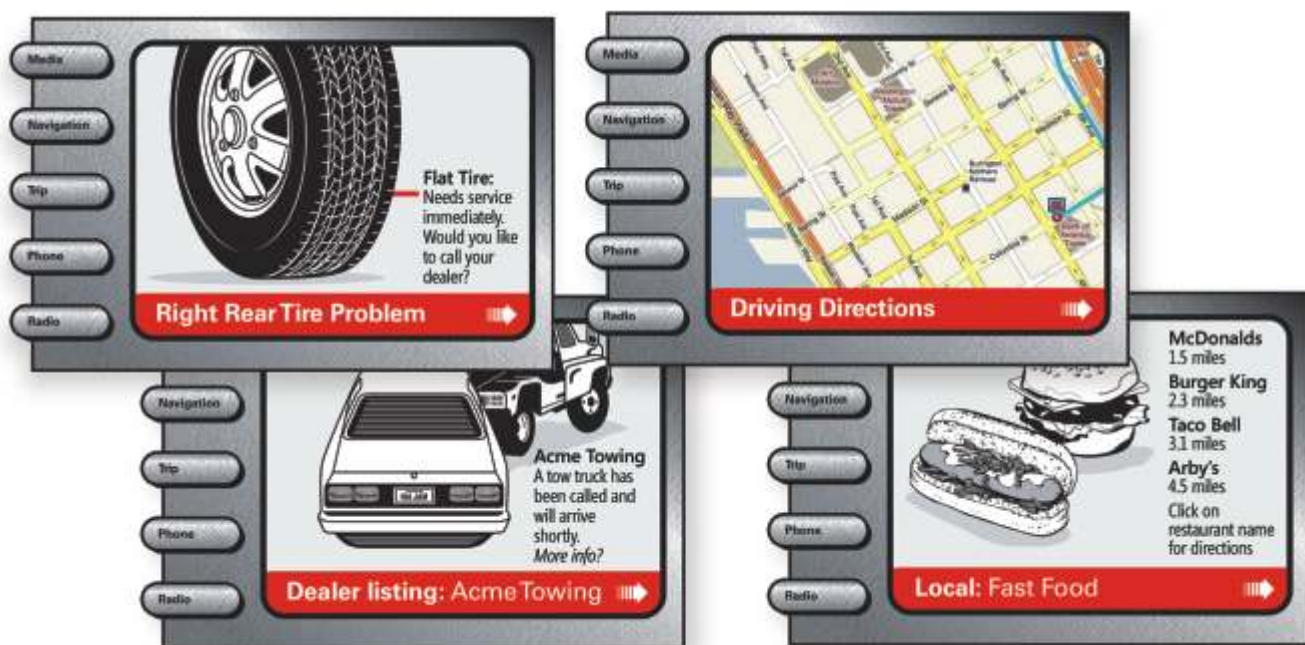
Ao chegar o momento de confirmar as reservas de hotel e pagar os ingressos de entretenimento, Mary hesita quando solicitam os seus dados do cartão de crédito. Como todos, ela ouviu que transmitir os dados do cartão de crédito em um site tem o potencial de acarretar o roubo de identidade. Mas, ela logo fica aliviada ao descobrir que poderá criar e usar um cartão de informações digitais, gerenciado pelo seu banco, para apresentar as informações de pagamento aos diversos fornecedores. Esta opção oferece uma medida de segurança melhor do que informar os dados do cartão de crédito pela Internet a cada um desses fornecedores, individualmente. Ela pode escolher o cartão apropriado, diretamente do seu desktop, como fonte de informações de pagamento, permitindo ao seu banco transmitir a ela informações seguras sobre o cartão de crédito aos fornecedores indicados, sem transmitir informações sigilosas do seu PC.

A viagem tem início sem qualquer acontecimento desagradável. As crianças escolheram levar apenas alguns de seus DVDs na viagem. Se decidirem, por capricho, que gostariam de ver algo diferente, poderão sempre fazer o download de um filme no laptop do carro. Quanto ao pai, o seu smartphone se conecta com o carro via Bluetooth e as suas chamadas, mensagens de texto e e-mails são, agora, direcionadas para o computador do veículo e não no telefone. Uma mensagem de texto, vinda do sistema de segurança da casa, indica que um dos detectores de movimento percebeu uma agitação no quintal. O pai liga para Bob Thomas, vizinho, e pede-lhe para verificar o ocorrido; ele descobre que foi uma das crianças do outro vizinho que procurava a bola.

Algumas horas depois de iniciada a viagem, acende a luz do console do carro. Os dados dessa ocorrência são imediatamente enviados a uma assistência técnica de diagnóstico indicada pelo fabricante do do veículo. A oficina descobre um problema de garantia que precisa de cuidados e envia uma mensagem para o PC de bordo com um relatório de diagnóstico simples e o nome/local da concessionária mais próxima do local em que está o carro. A mamãe clica sobre o



Figura 1: O PC do automóvel oferece aplicativos para gerenciar o veículo e a viagem.



endereço da concessionária, recebe instruções, passo a passo, e a família segue para o desvio obrigatório. A família utiliza a funcionalidade de pesquisa no console do computador do carro para encontrar um restaurante próximo da concessionária para fazer um lanche rápido. (A Figura 1 ilustra o cenário da lâmpada do motor.)

Depois da assistência técnica eles voltam para o seu caminho, com três horas de atraso. A mãe pega o itinerário da viagem, criado por ela em casa. A reserva do jantar no restaurante na próxima cidade precisa ser cancelada e novos planos para o jantar precisam ser feitos. Ela encomenda uma pizza, on-line, que será apanhada em uma pizzaria próxima e, mais uma vez, paga utilizando o seu cartão de informações bancárias. Depois da ótima refeição, eles finalmente partem para aquela viagem inesquecível.

## Arquitetura da solução

### Opções arquiteturais

Como podemos imaginar a partir deste cenário, a solução reúne serviços de inúmeros provedores e aplicativos que consomem esses serviços. Os aplicativos de consumo poderiam ser contêineres de aplicativos de acordo com o padrão da IU (interface de usuário) composta para permitir provisionamento dinâmico de novos serviços ou aplicativos personalizados para fins especiais, distribuídos como aplicativos cliente. A abordagem geral segue o modelo S+S para fornecer uma experiência de usuário simples, sem deixar de ser rica e intuitiva. A combinação de software cliente e serviços remotos é especialmente importante neste cenário móvel pois o software executado localmente pode aprimorar a experiência do usuário, mascarando a elevada latência das chamadas dos serviços OTA (over-the-air ou por radiodifusão) ou os problemas temporários de conexão de rede, comum até mesmo nas WANs sem fio mais modernas.

Os serviços classificam-se em três categorias gerais (Tabela 1):

- **Serviços comuns de plataforma**, serviços básicos publicados por provedores terceirizados. São tipos de serviços que outros

provedores utilizam como base para ampliar o próprio histórico, e também utilizar a maturidade e a estabilidade da infra-estrutura, da especialização e das atividades do negócio de terceiros (um dos princípios básicos do S+S). Muitos programas e negócios on-line já perceberam a oportunidade de construir uma plataforma "na nuvem". A oferta de serviços pela Microsoft sob a marca Windows Live é um exemplo de plataforma de Internet.

- **Serviços da montadora**, fornecidos pela montadora - no carro ou na nuvem - para acessar informações do veículo, da concessionária e da própria montadora.
- **Serviços de valor agregado**, fornecidos pela montadora de automóveis ou por terceiros para oferecer serviços que aumentam as vendas ou a fidelidade do cliente ou serviços oferecidos para compatibilidade com dispositivos para fins especiais, vendidos por terceiros.

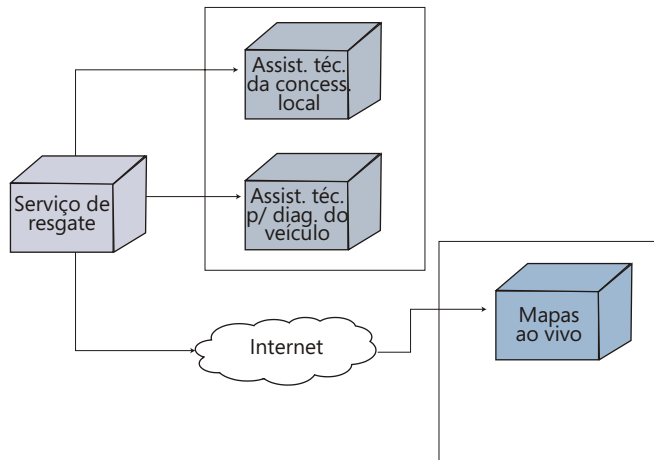
Essas categorias de serviço refinam a taxonomia de serviço mais genérica dos serviços do Windows Live mencionados no endereço: <http://www.microsoft.com/online/default.msp>

Tabela 1: Relações entre as categorias de serviço do consumidor conectado com a taxonomia do serviço Windows Live

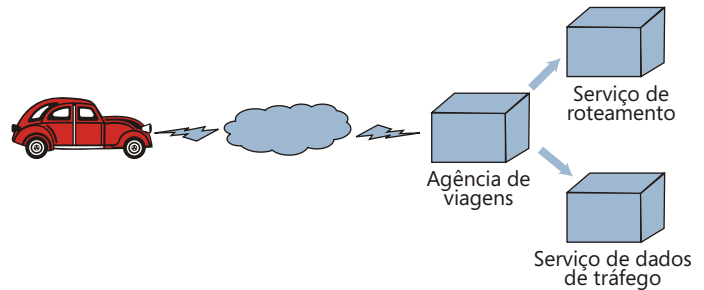
Serviço para o consumidor conectado	Categorias do serviço Windows Live
Serviços comuns de plataforma	Serviço de blocos de construção
Serviços da montadora	Serviços vinculados
Serviços de valor agregado	Serviço concluído ou serviço vinculado



**Figura 2:** Os serviços compostos de resgate agregam serviços da montadora de automóveis e de terceiros.



**Figura 3:** A arquitetura mais flexível consome apenas serviços remotos.



em que lhes falta conhecimento específico do domínio. Um exemplo de aplicativo composto poderia ser um serviço localizador da concessionária (Figura 2). Montadoras de automóveis devem saber o local de suas concessionárias, mas elas talvez não tenham os dados necessários para indicar o caminho para chegar a elas a partir de determinado ponto. Muito provavelmente, as montadoras confiariam nos serviços fornecidos pelo Windows Live Maps ou Mapquest.

Os serviços de valor agregado, por si mesmos, não seriam considerados produtos acabados. Esses serviços são blocos de construção específicos de domínio, para construir aplicativos completos. Podemos dizer que esses serviços fornecem os recursos utilizados para criar um item de software mais complexo e completo.

Grande parte da flexibilidade é obtida pela utilização de um modelo de implantação que compreende unicamente os serviços executados na nuvem, já que as modificações só exigem atualizações dos servidores que hospedam os serviços e não em cada carro que consome o serviço. Tipicamente, os serviços são baseados em HTTP, os quais podem ser chamados por um cliente: aplicativo ou outro serviço. O benefício desta abordagem é que os softwares executáveis são implantados de modo centralizado, em uma central de dados, facilitando o gerenciamento. A desvantagem é que o consumidor do serviço precisa ter uma conexão disponível para usar o serviço (Figura 3).

Ainda que seja possível executar a solução completa no cliente, esse padrão constitui um ambiente fechado em que o acesso a dados mais relevantes e atualizados não é possível e isso torna a solução menos útil. Este é o blueprint que existe hoje para muitos sistemas de computação móveis, especialmente aqueles de base automotiva. O caminho de aprimoramento desses dispositivos não existe e a funcionalidade limitada que fornecem é de benefício mínimo para o proprietário do dispositivo (Figura 4, página 20).

Um padrão preferível para uma solução de computação móvel aproveita a capacidade de acesso ao software e aos dados armazenados no dispositivo local. Neste caso, podemos implantar uma lógica de valor agregado no dispositivo e disponibilizar dados suficientes para o aplicativo, para que este pudesse ser utilizado mesmo quando a conexão está inativa. Este modelo descentraliza boa parte da lógica de integração e controle, além de introduzir desafios de manutenção e resolução de bugs, mas o aprimoramento na experiência do usuário, provavelmente, compensará o sacrifício (Figura 5, página 20).

## Plataforma tecnológica de serviços

A Internet está no início de sua transformação para tornar-se uma plataforma de serviços na nuvem: plataformas como o Windows Live

## Ambiente tecnológico (restrições e pressuposições)

A solução exige um ambiente em que o acesso à Internet esteja amplamente disponível, mas não necessariamente universal. De modo ideal, o carro tem meios de se conectar à Internet sem dispositivos adicionais - por exemplo, não exige um celular na base para possibilitar o acesso à rede para serviços oferecidos pelo carro. O acesso por rede ao carro permite cenários de aplicativos como início e diagnósticos remotos e notificações de eventos do carro.

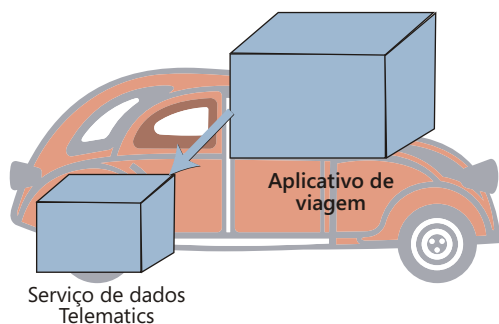
Alguns aplicativos que se beneficiam das informações do local talvez não precisem do contexto fornecido pelos dados específicos do veículo. Esses aplicativos também poderiam ser executados em dispositivos básicos, como smartphones preparados para GPS e PDAs, pois não há dependência dos dados específicos do veículo. Quase sempre, esses dispositivos estão equipados com plataformas modernas, como o .NET Compact Framework, e podem acessar serviços web baseados em SOAP, exatamente como um servidor ou PC desktop. Os produtos Visual Studio 2008 e .NET Compact Framework 3.5 acrescentam compatibilidade para o consumo de serviços web baseados em WCF e WS-\* para WS-Addressing e segurança de nível de mensagem baseadas no WS-Security.

## Serviços de valor agregado

No nosso cenário, a família Woodson interage com muitos tipos diferentes de software e serviços. Por exemplo, a montadora de carros forneceu um serviço para analisar os dados de diagnóstico do veículo e para notificar ao seu proprietário um problema com as informações sobre onde obter reparo para o problema. O serviço de planejamento da viagem armazenou o itinerário, disponibilizando-o pela Internet. Esses são serviços de valor agregado que podem ser completamente descritos em detalhes como serviços que fornecem uma experiência exclusiva ao consumidor. São diferentes dos serviços comuns de plataforma, pois não são considerados de uso geral ou amplamente disponíveis. Em lugar disso, muito provavelmente são desenvolvidos para estabelecer uma vantagem competitiva e, talvez, estender a utilidade de outro produto.

Em muitos casos, esses serviços podem ser compostos de código personalizado e um ou mais serviços básicos de plataforma, permitindo ao provedor de serviço disponibilizar recursos nos locais

**Figura 4:** A arquitetura que oferece a menor flexibilidade consome apenas os serviços locais oferecidos pelo veículo.



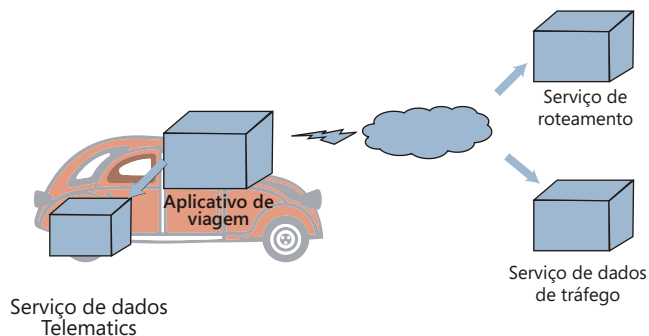
oferecem APIs de gerenciamento de presença, alertas, contatos e calendário, assim como mapas do Virtual Earth e orientações de caminhos. Outro exemplo é o BizTalk Services, que oferecerá uma subplataforma pública, além de roteamento e distribuição de mensagens. São todos ingredientes principais de aplicativos conectados ricos e robustos, mas os SLAs e conjuntos de recursos atuais nos deixam ver que esses serviços ainda estão começando seus ciclos de vida.

Serviços futuros poderão ampliar as configurações presenciais com a opção "dirigindo" que sempre permite alertas de tráfego e outros alertas configurados pelo motorista: a API de alerta poderia adicionar "carro" além de "aplicativo IM", "e-mail" e "SMS" à lista de pontos de notificação.

## Plataforma tecnológica de cliente móvel

Quando se trata de concretizar a experiência do consumidor conectado, existem várias plataformas à escolha e compensações entre elas. As prioridades e os limites ditados pela solução atual devem servir

**Figura 5:** A arquitetura S+S combina benefícios de serviços locais e remotos.



de base para a escolha da plataforma. Em geral, o Windows Vista Embedded oferece a experiência mais rica por meio do conjunto de recursos do sistema operacional e do .NET Framework completo, mas é também a plataforma com a maior projeção, a que mais exige recursos do processador, cuja licença é a mais cara. O Windows CE oferece uma alternativa de menor custo, que exige menos do hardware e oferece mais opções para personalizar o sistema operacional, mas com número menor de recursos. Uma plataforma baseada em Windows CE deve incluir o .NET Compact Framework para que seja possível aproveitar os benefícios de produtividade do desenvolvimento de código gerenciado e bibliotecas de classe de base. Por fim, o Windows Mobile fornece uma experiência mais rica que se aprimora constantemente. Os serviços de plataforma são fornecidos por meio do framework compacto, que oferece um modelo de programação consistente, com as habilidades de um amplo conjunto de desenvolvedores. A Tabela 2 da página anterior menciona os pontos fortes e as limitações de cada plataforma de desenvolvimento de aplicativos móveis.

**Tabela2:** Pontos para escolha da plataforma cliente

	Windows Embedded, .NET Framework	Windows CE, .NET Compact Framework	Windows Mobile, .NET Compact Framework 2.0 (3.5)
<b>Cenário-alvo</b>	Cenário rico em veículos - UMPC	Cenário de baixa fidelidade em veículos	Dispositivo de consumidor
<b>UX</b>	Conjunto de recursos WPF completo	Windows Forms, Silverlight, no Future Touch	Windows Forms, Silverlight, no Future Touch
<b>Comunicação</b>	WCF SOAP, REST e JSON	WS-I SOAP Web Services WCF Client (3.5)	WS-I SOAP Web Services WCF Client (3.5)
<b>Interação</b>	Voz com Vista ou suplemento de toque de terceiros	Voz, suplemento de toque de terceiros, se houver de hardware compatível	Voz, suplemento de toque de terceiros em dispositivos PocketPC
<b>Autenticação</b>	CardSpace	Nome de usuário	Certificados
<b>Armazenamento de dados</b>	Pleno acesso aos dispositivos locais de armazenamento em vários bancos de dados	Sistema local de arquivos, SQL CE (opcional)	Sistema local de arquivos, SQL CE
<b>Ferramentas de desenvolvimento</b>	Visual Studio	Visual Studio	Visual Studio
<b>Integração com carro</b>	Sim, via serial ou portas personalizadas	Sim, via serial ou portas personalizadas	Não

## Aplicativo cliente

Atualmente, temos à disposição várias opções para interação com sistemas computacionais. Aplicativos cliente ricos concretizam benefícios incriveis aos serviços de consumo, porque podem aproveitar os mecanismos para armazenamento de dados e o poder de processamento do local, dois itens indisponíveis nos mecanismos de distribuição baseados na web. Os aplicativos web têm o benefício de estarem hospedados e serem gerenciados de modo centralizado, mas, como dependem de conexão, não podem realizar plenamente a experiência do consumidor conectado. Um aplicativo cliente rico, além de fornecer a experiência de usuário mais abrangente, pode aproveitar prontamente os serviços que só estariam disponíveis no local. É mais prático consumir e avaliar os dados de desempenho do veículo localmente, no veículo, em lugar de transferir os dados para a nuvem, para processamento posterior. Além disso, evitamos quaisquer problemas de privacidade e segurança quando não transmitimos dados do carro a serviços remotos.

Por fim, um aplicativo cliente rico pode manipular identidade e sessão mais facilmente do que um aplicativo baseado na web. Armazenar cartões de informações localmente em um dispositivo e usar esses cartões para estabelecer identidade com um serviço ou outro aplicativo é um exercício trivial.

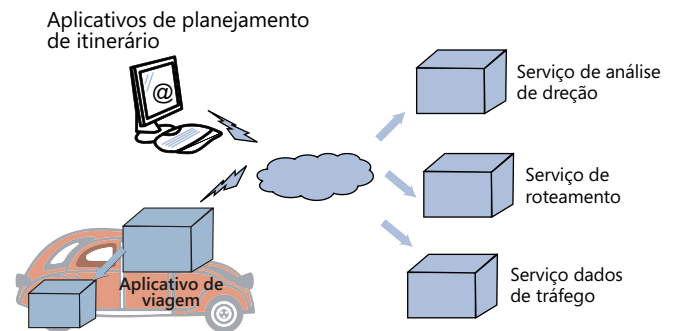
## Padrões de distribuição de serviço

Podemos contar com a conectividade amplamente disponível da Internet para a nossa solução, mas não podemos pressupor uma conectividade universal. Cada padrão de distribuição de serviço possui diferentes pontos fortes quanto à latência, flexibilidade e funcionalidade (Tabela 3). Se o cliente for rico, aceitará a adoção de um conjunto de padrões de projeto que nos permita não apenas aproveitar os períodos de desconexão da rede, como também usar as capacidades adicionais do cliente para aprimorar a experiência do usuário.

Muitos aplicativos ricos criados hoje em dia são apenas gabinetes que proporcionam entrada e saída para um conjunto de serviços subjacentes à interface de usuário. Neste padrão, o cliente é altamente dependente da conectividade para propiciar qualquer tipo de interação útil com o usuário final. A este cliente basta ter um cache para lidar com latências de rede ou condições em que a rede simplesmente não esteja disponível, porém sua utilidade não se estende para além daquilo que é fornecido pelos serviços que consome.

Podemos ampliar esse padrão para criar um segundo e mais útil padrão, no qual o cliente permanece como consumidor de serviços dependente, mas a infra-estrutura computacional é estendida ao cliente. Em outras palavras, aproveitamos a capacidade do aplicativo de executar tarefas de processamento no veículo, aliviando, assim, os serviços de parte da carga de processamento. Um resultado típico

**Figura 6:** Cabeças diferentes permitem uma melhor utilização dos serviços e uma experiência mais diferenciada para o usuário.



neste caso é um aplicativo com maior poder de resposta, embora com desempenho ainda não pleno.

Com a impossibilidade de fazer o download de toda a funcionalidade para o dispositivo cliente, podemos implementar um padrão no qual os serviços tornem-se simples provedores e consumidores de dados. A lógica de como utilizamos os dados está incorporada ao cliente, que torna-se o ponto focal da experiência do usuário. Este padrão nos permite lidar com a latência causada por conexões de rede lentas ou não existentes, bem como oferecer ao usuário uma experiência semelhante à que tem com o próprio PC em casa. Uma questão associada a esse padrão é o da capacidade de atualização. Como o aplicativo é desenvolvido para o dispositivo do cliente, torna-se mais difícil adicionar funcionalidade ou atualizar a existente. Serviços como o modelo de implantação de um clique do .NET Framework e a adoção de padrões de aplicativos de IU compostos ajudam a superar tal desafio. Novos serviços podem ser prontamente atualizados ou acrescentados, como no caso de serviços Web.

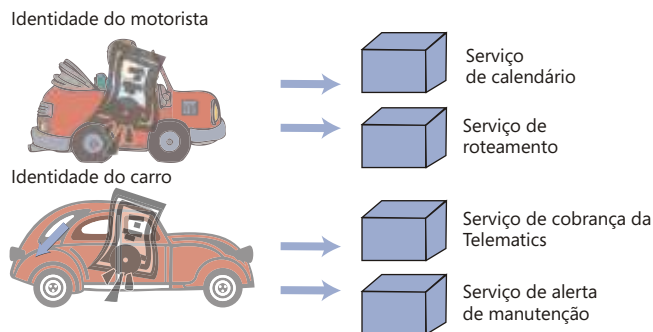
Os fornecedores de software e provedores de serviço podem oferecer várias versões de seus produtos, por exemplo: uma versão independente de veículo para dispositivos disponibilizados por meio de canais regulares de vendas e uma versão específica a um veículo, com funcionalidade adicional, disponibilizada pelo fabricante do veículo (Figura 6). Um modelo avançado "incrementaria" a experiência no dispositivo e no carro quando o cliente comprasse ambos os produtos. O aplicativo do dispositivo móvel também forneceria itens de valor agregado, como a exibição do local atual do carro, início remoto e sincronização automática de dados no próprio dispositivo quando levado ao carro. Ambos, o aplicativo do dispositivo e o aplicativo do veículo, poderiam acessar os serviços na nuvem do provedor de serviço, o que aumentaria a sua utilização pela abertura a um público-alvo mais vasto.

Com a arquitetura S+S, os aplicativos cliente não ficam ligados de forma indissociável ao veículo. Os serviços podem ser oferecidos através de uma miríade de dispositivos comerciais, como smartphones e PCs móveis, e com experiências criadas sob medida para essas plataformas. Como os dados armazenados na nuvem estão disponíveis em todo e qualquer dispositivo, é possível fazer a transição entre dispositivos de forma praticamente transparente. O princípio que deve ser mantido é que a interação com os dispositivos deve ser uma experiência natural para o usuário. A coleção crescente de dispositivos cada vez mais poderosos oferece várias opções para proporcionar um nível uniforme de interação entre cliente e computador. A interface de usuário pode ser alterada para adaptar-se ao fator forma, mas o nível de serviço permanecerá o mesmo.

**Tabela 3:** Comparação de arquiteturas cliente

Padrão	Latência	Flexibilidade	Funcionalidade
Consumidor de serviço magro	Alta	Alta	Média
Cliente mais rico	Média	Média	Média
Cliente inteligente	Baixa	Baixa	Alta

**Figura 7:** Carros e motoristas precisam de identidades digitais diferentes próprias para acessar os serviços.



### Segurança, identidade e o cliente

A identidade é um desafio em qualquer sistema, assim como autenticação, autorização e privacidade. Este é particularmente o caso dos aplicativos móveis. Quanto maior a separação entre o aplicativo básico e o dispositivo, mais difícil torna-se lidar com as questões de segurança. Um cliente mais rico permite a manutenção de tokens de segurança no dispositivo, criando um aplicativo comprovadamente mais seguro.

#### Comunicação segura

A privacidade é uma preocupação importante para todos que fazem intercâmbio de dados com serviços na nuvem. No passado, o foco da transmissão de dados criptografados recaía sobre o HTTPS e o SSL. O problema com o SSL é que trata-se de um protocolo ponto-a-ponto e não possibilita um intercâmbio seguro de dados entre várias extremidades. Os aplicativos compostos (que, quase por definição, constituem-se em diversos serviços) demandam uma abordagem ponta a ponta. Uma solução melhor seria ocultar os dados no dispositivo e permitir que fossem transportados por uma conexão criptografada ou clara. Esta é a abordagem adotada pelos criadores do protocolo WS-Security. O dispositivo do cliente possui uma chave pública utilizada para criptografar os dados antes da sua liberação para transporte. Apenas o serviço de destino é capaz de descriptografar a mensagem. Os dados da mensagem destinados a diferentes destinatários podem ser criptografados por diferentes chaves públicas, garantindo dessa maneira que só sejam compreensíveis pelo destinatário correto. As plataformas de desenvolvimento do lado cliente, como as várias versões do .NET Framework, implementam o WS-Security como parte do WCF.

As assinaturas digitais proporcionam uma medida adicional de confiança. O cliente poderia utilizar um identificador único, como uma chave privativa de um certificado X.509, para assinar a mensagem e assegurar que os dados tenham sido de fato enviados pelo remetente cuja identidade é exibido na mensagem. A assinatura digital também garantiria que os dados não foram adulterados durante o tráfego. A assinatura digital é basicamente uma combinação unidirecional dos dados de origem. Se a combinação não puder ser reproduzida pelo destinatário, a assinatura é considerada inválida: ou o par de chaves não é correspondente, o que invalida a identidade alegada, ou os dados foram adulterados durante o tráfego.

A abordagem S+S permite a utilização do WS-Security, mas oferece uma opção ainda mais segura para melhorar a privacidade: não transmitir nenhuma informação de segurança. A transferência de operações computacionais que incluem dados pessoais identificáveis

para o carro ou o dispositivo elimina a necessidade de transmitir as informações através de canais inseguros. Consideremos a autenticação do CardSpace como exemplo. Uma identidade CardSpace vinculada ao carro evita a transmissão de dados pessoais de identificação ou combinações fracas de nome de usuário e senha.

Para o bem da privacidade, os provedores de serviço são motivados a não solicitar nenhuma informação potencialmente confidencial. Em todos os casos excepcionais, o provedor do aplicativo deve solicitar a permissão do usuário para transmitir as informações para o serviço. Por padrão, todos os aplicativos devem seguir a diretriz da Microsoft para processamento seguro: não transmitir quaisquer informações potencialmente confidenciais sem a permissão explícita do usuário.

#### Autenticação e autorização

Os tokens de identidade podem ser utilizados para autenticar e autorizar os usuários para serviços que desejam acessar. Um cartão de informações, como o CardSpace, permite que o usuário apresente uma identidade a um serviço. A tarefa de verificar a identidade pode ser assumida pelo serviço ou uma parte confiável pode ser usada para a validação. A autorização ainda assim permaneceria sendo responsabilidade do serviço acessado.

A autenticação é muito importante no nosso cenário de mobilidade por diversas razões: precisamos restringir o acesso à aplicação a clientes pagantes, mas também precisamos garantir que todos os dados privativos de clientes sejam protegidos contra acesso não autorizado. No nosso cenário, há ainda uma outra preocupação: o acesso remoto ao carro. Existem preocupações de privacidade associadas ao acesso à localização do carro e ao histórico de viagens, mas também há preocupações de segurança. Ligar ou desligar o carro, por exemplo, é um recurso que deve ser muito bem protegido. Você não gostaria que um pirata desligasse o motor do seu carro no meio de uma via expressa.

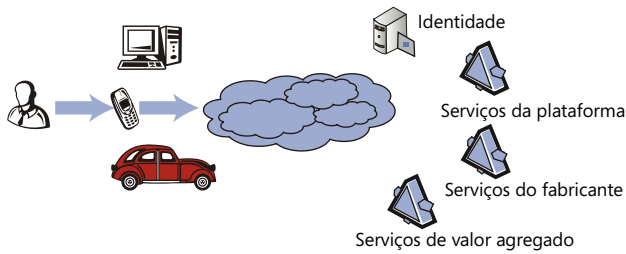
**“A ABORDAGEM S+S PERMITE O USO DO WS-SECURITY, MAS OFERECE UMA OPÇÃO AINDA MAIS SEGURA DE MELHORAR A PRIVACIDADE: NÃO TRANSMITIR NENHUMA INFORMAÇÃO DE SEGURANÇA. A TRANSFERÊNCIA DE OPERAÇÕES COMPUTACIONAIS QUE INCLUEM DADOS PESSOAIS IDENTIFICÁVEIS PARA O CARRO OU O DISPOSITIVO ELIMINA A NECESSIDADE DE TRANSMITIR AS INFORMAÇÕES ATRAVÉS DE CANAIS INSEGUROS”.**

Na Internet, a identidade do usuário é tipicamente estabelecida inserindo-se uma combinação de nome de usuário e senha, mas não é isso que esperamos quando entramos em um carro. A chave é o método tradicional de entrar no carro e acessar seus serviços. Podemos empregar um modelo de interação semelhante no cenário de serviços conectados com chaves inteligentes ou soluções baseadas no CardSpace.

Contudo, há algumas preocupações arquiteturais interessantes acerca de identidade no carro. Por exemplo, o carro em si é um aplicativo de vários inquilinos porque pode ter diversos motoristas com diferentes funções, como o proprietário, a filha adolescente do proprietário ou o mecânico, para citar alguns. Muitos carros atuais oferecem configurações de preferências de posições de assentos e direção para



**Figura 8:** A plataforma de serviços de Internet possibilita experiências ao consumidor conectado no carro, nos dispositivos e no PC.



diferentes motoristas, de acordo com a chave que utilizam. Esta experiência pode ser estendida aos dispositivos de processamento no veículo, disponibilizando aplicativos e dados de acordo com a chave utilizada. O acesso a todo o sistema pode ser limitado aos passageiros no carro.

O conceito de identidade é válido até mesmo para o veículo em si (Figura 7). Uma identidade digital poderia ser atribuída a carros individuais, permitindo acesso a serviços do fabricante específicos ao veículo. Os motoristas, por outro lado, necessitam de uma identidade portátil. A maioria dos usuários de computador hoje em dia tem pelo menos uma identidade digital associada a si. Estender essa identidade para uso no veículo não é uma tarefa trivial, mas é totalmente possível.

## Arquitetura de dados para vários inquilinos

Outro fator que deve ser considerado é o armazenamento e a privacidade dos dados. Para tornar este tipo de experiência computacional útil, uma grande quantidade de informações pessoais teria de ser armazenada na nuvem. Isto representa um desafio para o arquiteto de dados, que deve garantir que os dados sejam armazenados de forma eficiente e ao mesmo tempo sem comprometer a segurança e a privacidade de um consumidor que utilize o serviço.

A melhor solução, comprovadamente, para o armazenamento de dados de vários inquilinos é o banco de dados compartilhado, o método do esquema compartilhado. Neste caso, os dados de cada inquilino são armazenados em algumas tabelas com os dados associados a cada inquilino por meio de metadados. Este padrão delega a garantia de privacidade e segurança para qualquer aplicativo que acesse os dados e pode impor custos adicionais de desenvolvimento de software, mas a economia de longo prazo com a manutenção dos dados compensa com folga tais custos iniciais.

## Conclusão

As experiências de consumidores conectados, como a que foi descrita por este artigo, oferecem uma ótima oportunidade de agregar valor a produtos existentes. As gerações atuais e futuras de consumidores são conhecedoras de tecnologia e dependerão de auxílios digitais em todas as partes, não apenas nos computadores do seu trabalho. A ubiquidade dos dispositivos computacionais e a ampla disponibilidade de conectividade em rede representa uma oportunidade para que fabricantes e novos provedores de serviços conectem-se com seus clientes de formas novas e mais significativas (Figura 8).

O carro, em particular, é uma peça extremamente importante na vida de muitas pessoas. Você o usa para levar as crianças à escola, visitar clientes ou viajar nas férias. Você pode passar horas por semana dirigindo e certamente se encontrará em situações em que uma ajuda extra seria de grande valia.

A plataforma Microsoft é muito adequada para a criação de soluções S+S nos lados do cliente e do servidor. Tecnologias como o WCF e o .NET Framework são excelentes para o desenvolvimento de serviços baseados na nuvem, pois são compatíveis com protocolos de intercâmbio de mensagens (como JSON, POX/REST, SOAP e WS-\*) e garantem a interoperabilidade com todos os tipos de consumidores de serviços. Muitas vezes, os serviços não são criados desde o início, mas pela agregação de outros serviços. Tecnologias como o BizTalk Server ou o BizTalk Services baseados na nuvem são muito bem adaptados para a agregação de serviços "blocos de construção". A plataforma também oferece serviços "blocos de construção" Windows Live, como contatos, alertas ou fotos, que podem ser incluídos em serviços com valor agregado.

O modelo software + serviços representa um padrão ideal de serviços em diversas plataformas. Mecanismos flexíveis de fornecimento de serviços nos permitem adicionar novos recursos com rapidez e interrupções mínimas dos sistemas existentes. Os avanços nas tecnologias de apresentação e fatores de forma de dispositivos nos permitirá oferecer softwares aos usuários da forma mais apropriada aos seus contextos.

Já estamos vendo a combinação Software + Serviços aparecendo em muitas áreas. Os pioneiros na adoção desse modelo já estão comprovando o valor dessas soluções e definindo o padrão que deverá ser adotado pelos que estão chegando. As ferramentas e as plataformas estão aí. Agora só falta os provedores de aplicativos criarem soluções que atinjam os usuários das melhores maneiras possíveis.

## Sobre os autores

**Darryl Hogan** é arquiteto do Developer and Platform Evangelism Group da Microsoft. Darryl tem ampla experiência de arquitetura e implementação de inúmeros aplicativos corporativos durante cerca de 15 anos no setor de TI. Atualmente, a função de Darryl envolve o fornecimento de orientação e ensino a arquitetos que estejam implementando soluções e arquitetura corporativas em áreas de desenvolvimento de tecnologias Microsoft.

**Christoph Schittko** é arquiteto da Microsoft, no Texas, onde trabalha com clientes para a criação de soluções poderosas que combinam software + serviços, proporcionando experiências de ponta para consumidores e promovendo soluções de arquitetura orientada a serviço (SOA). Antes de entrar para a Microsoft, Christoph trabalhou com empresas na adoção modelos orientados a serviço e no fornecimento de soluções SaaS, ou software como serviço. Christoph tem mais de 14 anos de experiência como desenvolvedor e é arquiteto de soluções de software em um amplo leque de setores. Ele escreve e dá palestras sobre serviços web e XML em várias conferências. Christoph tem diploma avançado de engenharia elétrica pela Friedrich-Alexander University Erlangen-Nürnberg.

**Jon Box** é architect evangelist da Microsoft. Ele trabalha com clientes para que adotem as tecnologias da Microsoft na criação de soluções de alto impacto. Jon é programador profissional desde 1985. Ele trabalhou em uma grande variedade de ambientes e linguagens como: COBOL, Assembler, Clipper, C, C++ (Borland, ATL, MFC, Win32, COM/ DCOM), VB5/VB6 e .NET. Para conhecer melhor as idéias de Jon, visite o seu blog em <http://blogs.msdn.com/jonbox>.



# Perfil do Architecture Journal: Faisal Waris

Nesta edição, conheceremos melhor Faisal Waris, consultor de arquitetura na Ford. O *Architecture Journal* conversou com ele sobre a sua função, alguns dos seus desafios e seus pontos de vista sobre a arquitetura.

**AJ: Faisal, você poderia falar um pouco sobre si mesmo?**

FW: Sou consultor na Ford Motor Company, e trabalho para a Synova Corporation. Basicamente, minha função principal na Ford é a mesma que a de um arquiteto de SOA, e estou lá há quatro ou cinco anos. Também sou co-presidente de um grupo de trabalho AIAG que lida com sistemas de mensagens business-to-business (B2B). AIAG é a sigla de Automotive Industry Action Group, um organismo de padronização para a cadeia de suprimento automotiva. O grupo conta com uma base bastante grande de associados, como diversos OEMs e fornecedores.

**AJ: Quatro ou cinco anos na Ford como arquiteto SOA parece algo fascinante. Você poderia descrever o seu trabalho com mais detalhes?**

FW: A equipe a que pertenço faz parte do Enterprise Architecture Group, que assume diversas funções. Primeiramente, somos semelhantes em alguns aspectos a uma instituição de pesquisas: analisamos novas tecnologias, executamos provas de conceitos e testamos softwares e dispositivos para determinar se algum valor será agregado à Ford. Em seguida, planejamos e projetamos a tecnologia para as equipes de aplicativos principais. A outra função do grupo diz respeito ao estabelecimento de padrões, que pode incluir a definição de normas e melhores práticas acerca de arquitetura orientada a serviço (SOA) e outros tópicos arquiteturais. A governança de SOA é uma área essencial na qual estamos trabalhando no momento. Isto envolve a criação de processos, frameworks e orientações de uso para produtos e serviços de âmbito corporativo que estão sendo introduzidos na Ford. Nós também ajudamos as equipes de aplicativos a implementar serviços SOA na web e a diagnosticar problemas quando as coisas não dão certo.

**AJ: Muitos dos nossos leitores podem estar em processo de implementação da SOA. Quais recomendações você lhes daria?**

FW: Um dos pontos que discutimos aqui (e que provavelmente acontece o mesmo em muitos outros lugares) é: o que é um serviço? No que consiste um serviço? Se eu tenho um job de FTP, ele é um serviço? A palavra "serviço" descreve apenas um serviço web ou um serviço no AJAX também conta? Acho que muitas empresas, incluindo a nossa, enfrentam a dificuldade desta definição. Nós decidimos adotar uma forma muito pragmática de resolvê-la: utilizar a definição do OASIS. Você pode ir ao OASIS e obtê-la do seu site. Decidimos adotá-la como nossa definição porque muitas pessoas trabalharam nela, portanto trata-se de uma espécie de visão consensual sobre como definir a SOA. Ainda é uma definição bastante genérica, mas ela permite que você a ajuste para usos específicos. Segundo essa definição, quase tudo pode ser considerado um serviço, mas se você precisar de uma arquitetura que conecte todos esses serviços de uma maneira uniforme, respeitando a definição do OASIS, é preciso começar

a pensar no quê, de fato, você quer que seja um serviço. Descobrimos que para se ter uniformidade entre os serviços, a primeira etapa é estabelecer um determinado conjunto de padrões e protocolos.

**AJ: Você mencionou o foco na governança. Você acha que muitas organizações atuais enfrentam dificuldades com governança?**

FW: Acho que depende da maturidade da organização ou da maturidade da SOA na organização. Se você estiver começando agora, uma governança pesada não é o mais apropriado. Neste momento, o necessário é ajudar e nutrir a SOA incipiente na organização. Isto pode incluir o evangelismo da SOA, conversas com as equipes de aplicativos, explicação dos conceitos e de como realizar as tarefas. Após um certo ponto, quando todos estiverem confortáveis e os serviços estiverem sendo criados, pode-se voltar atrás e aplicar uma governança mais forte. Como organização, estamos no momento passando por esta transição. Já fizemos a evangelização e a SOA foi aceita. Estamos agora na fase de começar a pensar em como gerenciar isso tudo, garantindo que os serviços estejam sendo criados e evitando duplicações. Considerando que temos diferentes equipes de aplicativos com diferentes pontos de vista, temos de definir como fazer a intermediação do serviço certo para obter os consumidores certos, os provedores certos e o conjunto de interfaces certas.

**AJ: Você também mencionou a co-presidência de um grupo de trabalho do AIAG. Você daria aos leitores um pequeno histórico do AIAG, enfatizando em especial a missão do grupo?**

FW: O AIAG é mais que apenas um grupo de ação. Ele atua como organismo vertical na cadeia de suprimentos e também como definidor de padrões. É um grupo que reúne fornecedores e OEMs, e coordena o trabalho conjunto deles para resolver problemas comuns. Há padrões de engenharia e padrões de processo, mas cada vez mais estamos nos voltando para os padrões de eCommerce, que são os padrões relacionados com o intercâmbio de informações entre fornecedores e OEMs. Faço parte do grupo que trabalha com este tipo de padrões, e a nossa meta é possibilitar uma integração transparente entre os parceiros da cadeia de suprimento automotiva pela definição de processos de negócios padrão. Um processo padrão, por exemplo, poderia ser o gerenciamento de inventários (digamos Kanban, MinMax) qualidade ou garantia. Em seguida, pegamos esses processos e executamos provas de conceito, utilizando serviços web e tecnologias relacionadas (como a ebMS). Demonstramos que muitas implementações de um processo de negócios (com interfaces definidas por um conjunto de WSDLs) podem interoperar de forma segura e confiável. A nossa meta é promover isto no setor.

**AJ: A edição atual do Journal é sobre aplicativos e dispositivos móveis. Alguns leitores já ouviram falar do novo produto da Ford, o Sync. Você poderia falar um pouco mais sobre ele?**

FW: O Sync é produto de uma parceria entre a Microsoft e a Ford, e é um dos primeiros aplicativos do Microsoft Automotive PC já em produção. O Sync é muito interessante. No contexto atual, este aplicativo oferece serviços de entretenimento, reconhecimento de voz,

operação de telefone e dispositivos de mídia. Podemos considerá-lo uma plataforma computacional de finalidade geral no automóvel. Com este paradigma, o céu é o limite. Há infinitas possibilidades do que pode ser feito com ele. Claro, existem desafios relacionados ao que pode ser colocado em um carro da perspectiva do usuário, por uma série de fatores. Mas um novo mundo está se criando e estamos muito empolgados por fazer parte dele.

**AJ: Como você vê o papel do software na evolução dos automóveis no próximos anos?**

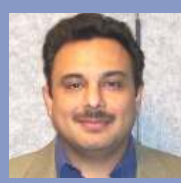
FW: A Ford está claramente voltada para diversas áreas, algumas delas impossíveis de serem mencionadas nesta entrevista por questões de confidencialidade, mas uma plataforma computacional de finalidade geral no carro permite muitos sonhos e possibilidades. Há o potencial para o fornecimento de diversos serviços dentro do carro que possam ser operados por reconhecimento de voz, em especial que apresente um alto desempenho no ambiente automotivo.

**AJ: Acreditamos que muitos leitores do The Architecture Journal aspiram ser arquitetos, são por exemplo desenvolvedores seniores buscando uma nova etapa em suas carreiras e pensando no que precisam para se tornar arquitetos. Como você já exerce esta função há um bom tempo, que conselho daria a um candidato a arquiteto hoje?**

FW: Eu acho que há um processo gradual na transição de um desenvolvedor para líder de desenvolvimento e depois para arquiteto. Há também muitos tipos diferentes de arquitetos. Na Ford, temos arquitetos de infra-estrutura, arquitetos de soluções e até mesmo arquitetos específicos, como os de SOA. Eu acredito que se você deseja ser um arquiteto, estar sempre atualizado com a literatura é essencial, porque ser capaz de compreender como pegar uma série de tecnologias e costurá-las em uma solução demanda um conhecimento bastante amplo e que inclua uma boa parte do que está acontecendo no setor. Tenho que ler muito só para me manter atualizado com o que está acontecendo. Eu também sou mais um arquiteto prático, então, embora eu esteja desempenhando uma função de arquitetura, eu gosto de arregaçar as mangas mexer com o código dos programas. Descobri que isto vale muito a pena. Muitos arquitetos param de fazer esse tipo de trabalho prático, o que pode ser um equívoco, pois não dá para ser um arquiteto de verdade se você não entender todo o quadro, e às vezes é necessário entender as coisas em um nível bem detalhado para ser capaz de tomar a decisão certa. Meu conselho é mexer bastante com novas tecnologias e fazer testes. Não estou dizendo que você deve escrever aplicativos para produção, mas pode fazer experimentos para ter uma idéia do potencial das novidades: o que dá para fazer, quais são os limites, os recursos... e essa visão geral lhe dará uma perspectiva muito melhor do que vai ou não funcionar.

**AJ: Uma das perguntas que sempre fazemos aos nossos entrevistados é: "qual é a coisa de que você mais se arrependeu na sua carreira e o que aprendeu com ela?"**

FW: Essa é difícil de responder! Não consigo me lembrar de algo de que tenha realmente me arrependido. Tive muitas oportunidades de me tornar um gerente puro e me afastei totalmente delas. Sempre tive um pé na área técnica. Acho que isso me ajudou a ser o que sou hoje. Nesta posição, eu de fato acredito que possa haver uma limitação que estou começando a enfrentar: como posso avançar desta posição? Devo passar para um cargo de gerência ou permanecer na arquitetura numa função de consultoria, enfim, como evoluir? Esta é uma dúvida que preciso resolver.



Faisal Waris  
Synova Corp.

Faisal Waris trabalha na Synova Corp. e é arquiteto-consultor de SOA na Ford Motor Company. Recentemente, tem atuado ativamente no programa Sync ([www.syncmyride.com](http://www.syncmyride.com)). O Sync é produto de uma parceria entre a Microsoft e a Ford, e é um dos primeiros aplicativos do Microsoft Automotive PC. Faisal é também co-presidente de um grupo de trabalho no Automotive Industry Action Group (AIAG, [www.aiag.com](http://www.aiag.com)), onde trabalha com padrões de comércio eletrônico do setor automotivo.

**AJ: Sobre a sua carreira, o que você espera conseguir nos próximos anos e a longo prazo?**

FW: Uma das minhas buscas pessoais é estabelecer uma maneira de lidar com o intercâmbio de mensagens B2B. Se você parar para pensar no que temos hoje em termos de intercâmbio de mensagens, a maioria é intercâmbio eletrônico de dados (EDI). Há alguma coisa de XML sendo implementada, mas se analisarmos bem, ainda baseia-se na maior parte em EDI por FTP. Minha busca pessoal é viabilizar serviços B2B na web, e acredito que temos todos os componentes. Temos as especificações WS-\* que funcionam bem no espaço B2B. Por exemplo, temos o WS-Addressing para o intercâmbio assíncrono de mensagens, o WS-ReliableMessaging para a entrega de mensagens robusta, o WS-Atomic-Transaction para transações, o WS-Security e o WS-SecureConversation para segurança e assim por diante. Muitos dos conjuntos de ferramentas estão agora implementando esses padrões, então, uma coisa com a qual estou trabalhando aqui na Ford e no AIAG é o estabelecimento desses novos padrões para o intercâmbio de mensagens B2B. Claro, isso não é nada fácil considerando a imensa base instalada de tecnologias existentes, mas, pouco a pouco, começamos a ver pessoas dando o valor ao intercâmbio de mensagens com XML, porque as possibilidades são muito maiores. Diferentemente do EDI, muitos aspectos do intercâmbio de mensagens XML podem ser gerenciados apenas com a adoção de metadados (como XML Schema, WSDL, WS-Policy etc.) Além disso, é possível criar aplicativos robustos (por meio do intercâmbio de mensagens confiável), o que é difícil com algo como o EDI. Esta é a minha causa pessoal, portanto, é muito empolgante e recompensador.

Outra área da tecnologia na qual me interessa são as tecnologias semânticas para web. Elas podem estar ficando um pouco para trás hoje em dia (acho que ela já passou do ponto de efervescência e pode até ter entrado na fase de desencanto), mas quando eu trabalho com informações, vejo como podem ser mais bem gerenciadas, e sempre volto para os recursos prometidos pela perspectiva semântica da web. Espero que essa necessidade seja reconhecida pelos outros e que haja um ressurgimento.

Neste momento, estou muito satisfeito com o que estou fazendo e muito, muito ocupado, então não tenho muito como pensar a longo prazo. Eu definitivamente gosto da função do arquiteto e não sei se vou querer abandoná-la tão cedo para passar para uma nova etapa da minha carreira.

**AJ: Faisal, obrigado por nos dar a oportunidade de conhecer seus pensamentos e idéias!**

Se você quiser sugerir um bom candidato para o perfil da próxima edição do Journal, escreva para os editores: [editors@architecturejournal.net](mailto:editors@architecturejournal.net).





# Arquitetura de Dados Móveis

por Rodney Guzman

## Resumo

Os aplicativos de conexão ocasional têm reputação de serem de difícil implementação. Os desafios são dispostos em camadas e o cerne do problema é como gerenciar os dados. Cada cliente de conexão ocasional conectado consome e produz dados em intervalos aleatórios. Durante a desconexão, são produzidas ilhas de dados que devem ser reconciliados quando a conexão voltar. Existem tecnologias para a transferência de dados de e para clientes; todavia, apenas as tarefas de reconciliação mais simples são processadas sem a necessidade de um código personalizado. Por exemplo, o SQL Replication é uma ferramenta fantástica para a passagem de novos dados de um sistema para outro. Porém, quando um registro é atualizado por mais de um cliente, a última atualização deve prevalecer sempre? Além disso, o que acontece quando novas entidades de dados são criadas em vários clientes, mas cada uma delas representa a mesma instância única de entidade? Infelizmente, os modelos de dados são normalmente muito complexos para serem manipulados por processos de resolução de conflitos genéricos.

Este artigo irá discutir como construir o seu modelo de dados para que seja compatível com um aplicativo de conexão ocasional. O modelo de dados deve ser compatível com o processo de resolução de conflitos complexo incorporado no .NET. Um exemplo de aplicativo de ponto de serviço com exigências estritas será utilizado: pontos de entrada para clientes inteligente e web; os clientes inteligentes serão de conexão ocasional; entidades de dados que envolvem várias tabelas; e um processo de resolução de conflitos complexo com capacidade para novos registros criados por instâncias de aplicativo web ou aplicativo de cliente inteligente.

## Cenário

Para os fins deste artigo, consideraremos um exemplo de aplicativo de gerenciamento de eventos. O aplicativo é um site hospedado na web com um modelo de dados robusto. As demandas de negócio incluem o aplicativo ser acessível em áreas remotas e desconectadas, onde os eventos serão realizados. Nos eventos, os produtos da empresa poderão ser comprados e as informações do participante atualizadas como em um site na Internet. Cada evento pode durar vários dias e receber milhares de participantes. Durante o evento, não há regularidade quanto a um participante pressupor que há conectividade de rede. Logo após o término do evento, os dados de cada cliente devem ser reconciliados. O diagrama de banco de dados

simplicado mostrado na Figura 1 é um exemplo de subconjunto do aplicativo.

O negócio demanda a reutilização de componentes existentes para minimizar o custo de criação de um aplicativo cliente inteligente. Além disso, não quer ter de recriar seus objetos de negócio que já existem em um nível de abstração apropriado no seu aplicativo ASP.NET. Eles querem que os mesmos objetos de negócio sejam desenvolvidos com o aplicativo cliente inteligente Windows Presentation Foundation (WPF). O SQL Express será utilizado no desktop e o esquema de dados reutilizado será o do aplicativo web.

## Quando houver conexão

Não é possível saber ao certo quando o aplicativo de conexão ocasional estará on-line, o que implica uma enorme carga ao aplicativo. Um aplicativo deve estar conectado em determinados momentos para executar funções básicas. No nosso cenário ilustrativo, para otimizar a experiência no evento para os participantes, as informações anteriores ao registro devem estar disponíveis no cliente desconectado durante o evento, portanto, o dispositivo é sincronizado com o depósito de dados central antes do evento. Independentemente de como as informações são intercambiadas, o dispositivo tem que estar conectado antes do evento para obtê-las.

As exigências iniciais do aplicativo incluíram um modelo que possibilitasse ao usuário executar qualquer função, a qualquer momento, sem que precisasse estar conectado. Ironicamente, o usuário tem que estar conectado em algum momento do ciclo de utilização do aplicativo para que haja a sincronização com o servidor. A falta de controle de quando o aplicativo estará conectado implica complexidades que não podem ser ignoradas. No caso deste aplicativo, as exigências foram modificadas para forçar a necessidade de o aplicativo estar conectado em alguns momentos-chave.

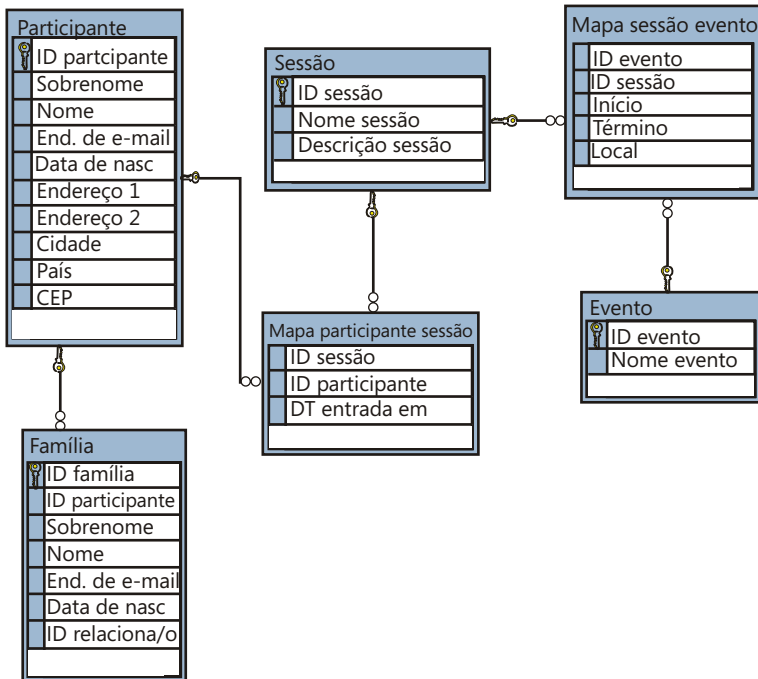
## A complexidade de nunca se saber quando ocorrerá a conexão

O SQL Replication é muito apropriado para trabalhar com o conceito de intercâmbio de dados e clientes de conexão ocasional. Contudo, se você estiver gerenciando um processo de várias etapas no qual cada etapa dependa de alguma forma da conectividade, saber onde se está no processo pode ser um desafio. No aplicativo de gerenciamento do evento, para depender exclusivamente do SQL Replication, o fluxo do processo deverá ser parecido com este:

1. A lista de eventos disponíveis seria replicada em cada dispositivo cliente. Se um novo evento for adicionado, ou um existente for modificado, todos os dispositivos devem ser sincronizados.
2. A autorização de quem pode ver cada evento deve ser replicada em cada dispositivo cliente. Se uma autorização for alterada, a alteração deve ser refletida também.
3. No dispositivo local, o usuário seleciona o evento a ser sincronizado, inserindo um registro na tabela EventDownload do banco de dados local, que inclui o nome do hospedeiro do computador cliente e o evento que deve ser replicado.



Figura 1: Uma versão extremamente reduzida das tabelas SQL



4. A tabela EventDownload é replicada de volta no servidor.

5. O SQL Replication filtrado transfere os eventos solicitados ao dispositivo do usuário.

Este fluxo de processo parece bastante simples, mas necessita de conectividade o tempo todo. Imagine ter uma conversa com um usuário não técnico sobre o porquê dele não poder ver o evento no seu dispositivo, pois a autorização que lhe confere o acesso mudou após ele ter sincronizado a sua máquina. Dependendo de um modelo em que você não tenha nunca o controle sobre a conectividade pode ser complicado para as operações de depuração e manutenção.

### Depender de momentos-chave de conexão

No final das contas, a conectividade é necessária em algum ponto. Diferente do modelo puro de replicação SQL, uma abordagem mais direta pode ser adotada para simplificar o processamento geral. Forçar a conectividade em pontos-chave evita muitas das operações de ida-e-volta descritas na seção anterior. Por exemplo, em vez de sincronizar a lista dos eventos com o banco de dados local, um serviço web pode ser acionado para recuperar uma lista de eventos autorizados com a qual o usuário pode sincronizar, alterando o fluxo:

1. O usuário inicia o aplicativo e recupera uma lista de eventos via chamada de serviço web. Apenas eventos autorizados são mostrados ao usuário.
2. O usuário escolhe um evento e chama o serviço web novamente. No servidor, é estabelecido um registro na tabela EventDownload com o nome do host do dispositivo do usuário.
3. As regras do SQL Replication filtrado associa eventos a nomes de host clientes, especificando quais eventos serão replicados para cada dispositivo de cliente (Figura 2).

Uma enorme quantidade de informações adicionais pode ser obtida através de serviços web, desde o número de registros que devem ser sincronizados até as atualizações de progresso de uma replicação (se ocorreu ou se está concluída).

### Resolução de conflitos

O SQL Replication e o ADO.NET não têm uma “bala de prata” para a resolução de conflitos. A maioria dos aplicativos é muito complicada

para ser processada por soluções prontas para utilização. Não dá para confiar sempre no esquema “o último prevalece”. Normalmente, precisamos esperar que os dados sejam acumulados de diversas fontes antes de tomar uma solução quanto a qual é “o melhor” registro.

As perspectivas são vertiginosas, mas existem técnicas para pegar atalhos nesta complexidade. No aplicativo de gerenciamento de eventos, dependemos dos pontos fortes do SQL Replication como pedra angular da solução. Um ponto forte conhecido do SQL Replication é que ele é fantástico para gerenciar inserções de SQL. No caso de atualizações, contudo, havia muitos cenários que demandavam um controle fino do processo de resolução de conflitos em intervalos programados.

O aplicativo de gerenciamento de eventos apresentou diversas complicações com este processo. Por exemplo, o registro de um participante não é representado por uma única tabela no banco de dados. Isto complica a replicação de informações que não foram alteradas, pois é a alteração que determina a replicação SQL. Um registro lógico dos participantes foi constituído, que é um registro que abrange diversas tabelas SQL. Quando ocorreu uma alteração em um componente do registro lógico, uma transação lógica foi produzida. Este conceito é importante, pois permite a transferência de informações ao depósito de dados, mesmo se uma alteração não ocorrer.

A maioria dos aplicativos padece dessa complexidade, e é justamente isto que torna a replicação bidirecional de dados tão difícil. Nós dividimos esta complexidade no aplicativo de gerenciamento de eventos, e o padrão que seguimos poderia ser implementado em qualquer aplicativo desconectado.

### Registros lógicos

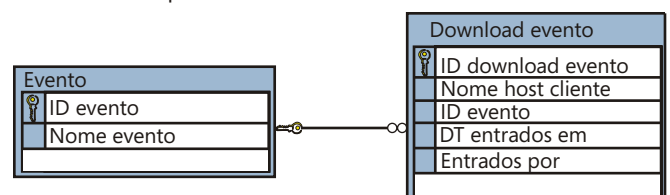
No contexto do aplicativo de gerenciamento de eventos, o registro lógico de um participante não contém apenas as informações na tabela de participantes, mas todas as informações relacionadas. Sessões selecionadas, compras de produtos e familiares constituem o registro lógico de participante. Se qualquer uma dessas informações for alterada, não importando de qual tabela a alteração for feita, então o registro lógico do participante foi alterado.

Um requisito básico para o aplicativo é que se as informações do participante forem manipuladas no cliente inteligente, a validação do registro lógico de participante inteiro ocorreu. Portanto, mesmo se uma única tabela for modificada, todos os dados da tabela que representam um registro lógico de participante foram validados como corretos no mesmo momento. Isto implica um dilema interessante, pois o processamento de replicação SQL padrão não é capaz de sincronizar todos os dados da tabela relacionados a um registro lógico se os dados não tiverem sido manipulados. Nós administramos este problema forçando a ocorrência de uma alteração no banco de dados para acionar o conjunto apropriado de eventos — chamados de “transações lógicas” — do SQL Replication.

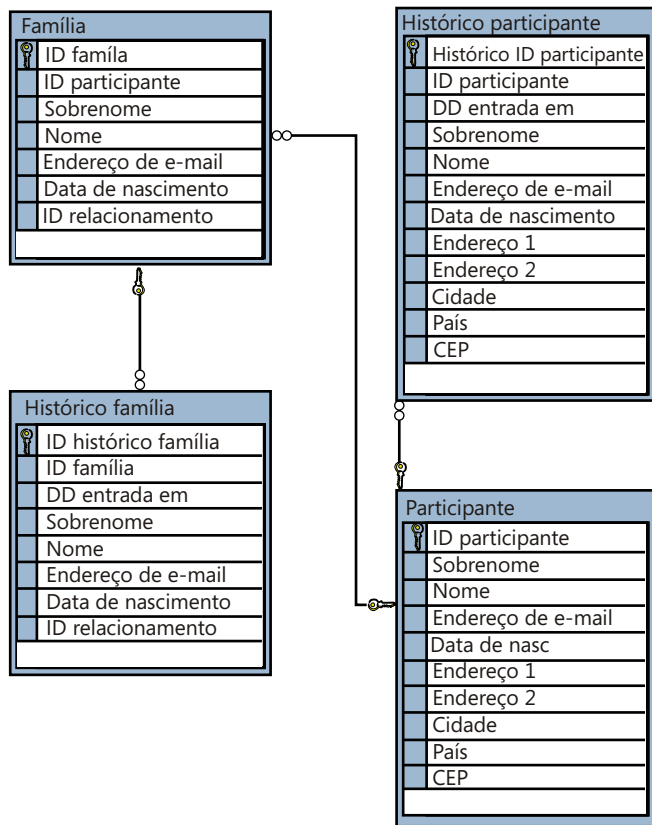
### Tabela de histórico

O banco de dados do aplicativo web tem uma tabela de histórico associada a cada tabela de usuário modificável no banco de dados. A

Figura 2: A tabela EventDownload especifica quais eventos serão baixados para cada host de cliente.



**Figura 3:** As tabelas de histórico nas tabelas-pai chaves permitem que as “transações” sejam mantidas.



finalidade dessas tabelas de histórico é manter o histórico de todas as alterações, quando foram feitas e quem as fez. Uma exigência do aplicativo cliente inteligente é que todas as alterações históricas locais sejam registradas e em seguida replicadas no depósito de dados central. Todas as tabelas de histórico são preenchidas por um gatilho na tabela pai respectiva. Toda vez que uma inserção ou atualização é feita na tabela-pai, um novo registro de histórico é gerado (Figura 3).

As tabelas de histórico constituem um histórico de transações de alterações de cada tabela, um fato que tornou-se importante para nós no momento de considerar como as atualizações de um registro poderiam ser replicadas novamente no depósito de dados central. Em vez de replicar uma alteração em um registro no depósito de dados central, como uma alteração na tabela-pai, poderíamos replicar os novos registros de histórico inseridos e reconciliar todas as informações de forma central e inteligente.

No nosso projeto, teríamos sempre pelo menos um registro em uma tabela de histórico. Se um registro for criado no cliente inteligente, o acionador dispararia e produziria um registro de histórico. Se um registro for replicado centralmente no cliente inteligente, o acionador dispararia, garantindo da mesma forma a presença de um registro de histórico.

Para aproveitar os registros históricos como transações, e promover o conceito de registros lógicos quando um registro for representado em diversas tabelas, é necessário vincular os registros de histórico entre as tabelas de histórico. Este requisito resultou no conceito de transações lógicas.

## Transações lógicas

Uma transação lógica é uma representação de um participante em todas as tabelas em um determinado ponto no tempo. O que estava

faltando nas tabelas de histórico foi uma maneira de associá-los. Quando uma alteração ocorreu na tabela-pai, o acionador será disparado para preencher a tabela de histórico respectiva. O acionador precisaria agora ser modificada para executar as seguintes funções (Figura 4):

- Fazer uma cópia da tabela-pai e inseri-la na tabela de histórico respectiva (como antes).
- Obter um novo ID de transação com registro de data e horário.
- Para cada tabela de histórico que representar um registro lógico, atualizar o último registro de histórico com o novo ID de transação.

O terceiro ponto é importante pois mesmo que tenhamos feito uma alteração em uma única tabela, e apenas um registro de histórico tiver sido gerado, neste momento todas as outras tabelas que representam o registro lógico foram validadas. A funcionalidade de obtenção de um novo ID de transação e atualização de todos os registros de histórico respectivos mais recentes é encapsulada em uma função SQL definida pelo usuário e inserida dentro de cada acionador.

As transações lógicas resolvem um problema chave que temos quando manipulamos informações que são atualizadas em vários dispositivos. Este problema deriva da complexidade da sincronização das mesmas informações, em vários dispositivos e em diferentes momentos. Considere este cenário:

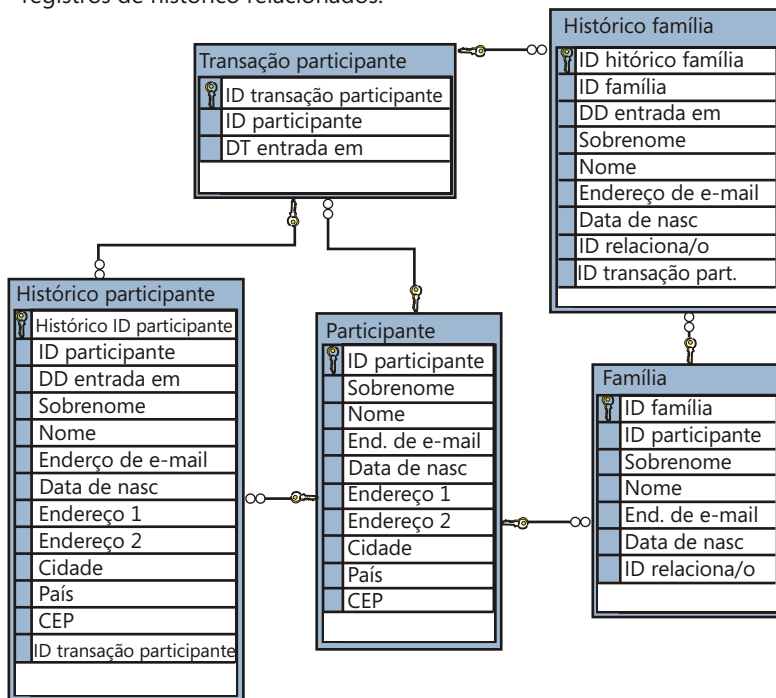
O dispositivo A atualiza o participante X na segunda-feira, e o dispositivo B atualiza exatamente as mesmas informações do participante X na terça-feira. Neste aplicativo, como as mesmas informações foram atualizadas, as informações atualizadas no dispositivo B são consideradas mais relevantes que as do A porque foram atualizadas depois. Contudo, o dispositivo B é sincronizado na quinta-feira e o A, na sexta-feira. No processo de resolução de conflitos da noite de quinta-feira, não se sabe de nenhuma atualização do dispositivo A, então as alterações do dispositivo B são aceitas. No processo de resolução de conflitos da noite de sexta-feira, as informações do dispositivo A são reconciliadas. Sem o registro de data e hora da transação lógica, é difícil saber quando a alteração ocorreu e se as informações do dispositivo A são mais ou menos relevantes que as do dispositivo B. As transações lógicas forneceram uma representação clara das informações de todos as entidades de dados no sistema.

## Registros atualizados

Quando um registro de tabela SQL é atualizado, o SQL Replication pode ser usado para transferir essas alterações de um banco de dados para outro. Parece muito simples quando dito dessa forma, todavia, saber se você deve sobrescrever as alterações que tem pelas de outro sistema não é tão trivial. A regra do último que chegar sempre prevalece não se aplica sempre. Pode haver partes do registro que devem ser mantidas e portanto a coisa não se resume a simplesmente sobrescrevê-lo.

Fazer um ajuste detalhado do que é exigido para que os registros atualizados possam fluir pelo aplicativo é importante para simplificar ao máximo o design geral. No caso de aplicativos móveis desconectados, o fluxo de informações sai de um servidor central, permanece e cresce temporariamente em cada dispositivo desconectado para finalmente retornar ao servidor central. Ao examinar este fluxo, fizemos algumas pressuposições e requisitos quanto ao modo de operação do aplicativo de gerenciamento de eventos.

**Figura 4:** Transações lógicas (AttendeeTransaction) mantêm os registros de histórico relacionados.



As informações do servidor central só foram transferidas uma vez para o dispositivo móvel. Por exemplo, quando um evento for solicitado para ser sincronizado e todas as informações de participantes forem replicadas no dispositivo, quaisquer atualizações feitas nas informações de participante no aplicativo web online não seriam replicadas no dispositivo. Mesmo que a intenção seja replicar as atualizações, não podemos confiar nos usuários para que re-sincronizem seus dispositivos durante o evento. Contudo, durante os eventos, os participantes podem atualizar suas informações online. O nosso projeto permite que vários dispositivos e o sistema online atualizem os mesmos registros e a reconciliação posterior pelo nosso processo de resolução de conflitos .NET. Isto é possível pela replicação apenas do que foi inserido, e não atualizado. Somente as inserções SQL (não as atualizações) são comunicadas entre os sistemas. Então podemos escolher um momento para reconciliar todas as informações centralmente.

No aplicativo de gerenciamento de eventos, após a replicação de um evento em um dispositivo, um participante replicado pode ser atualizado pelo aplicativo WPF. Quando o registro de participante é atualizado, um registro de histórico é produzido. Com o registro de histórico, uma nova transação lógica é criada. Ambos são inserções SQL. O que foi inserido é replicado novamente no servidor central. O registro-pai em si não é replicado, uma vez que isso sobrescreveria o registro do servidor. O processo de resolução de conflitos .NET examina a fila de transações lógicas e determina que novas alterações estão presentes. Examina também os registros de histórico que foram criados e determina a melhor abordagem para a integração dessas alterações nos registros-pai.

No aplicativo de gerenciamento de eventos, após a replicação de um evento em um dispositivo, um participante replicado pode ser atualizado pelo aplicativo WPF. Quando o registro de participante é atualizado, um registro de histórico é produzido. Com o registro de histórico, uma nova transação lógica é criada. Ambos são inserções SQL. O que foi inserido é replicado novamente no servidor central. O registro-pai em si não é replicado, uma vez que isso sobrescreveria o registro do servidor. O processo de resolução de conflitos .NET

examina a fila de transações lógicas e determina que novas alterações estão presentes. Examina também os registros de histórico que foram criados e determina a melhor abordagem para a integração dessas alterações nos registros-pai.

### Registros inseridos

As inserções SQL são muito mais simples de gerenciar com a replicação SQL da forma como estão, na superfície, sem resolução de conflitos para serem manipuladas. No entanto, o que acontece quando um novo participante é adicionado a um dispositivo desconectado que já existe? No nosso aplicativo, um participante poderia ter se registrado no aplicativo online depois que as informações de eventos tiverem sido replicadas; no evento, o participante poderia ser adicionado ao dispositivo móvel; posteriormente, o participante poderia estar trabalhando com outro participante em uma instância própria do aplicativo. Várias instâncias do mesmo registro de participante podem facilmente ser criados.

Começamos a considerar os registros inseridos no cliente como não sendo da mesma categoria que os registros inseridos no servidor central. Quando o registro inserido no cliente é replicado de volta no servidor, as informações contidas no registro devem ser validadas antes que fiquem acessíveis ao aplicativo. Para evitar que esses registros replicados sejam vistos pelo aplicativo ASP.NET, eles devem ser

marcados e excluídos das consultas SQL. O processo de resolução de conflitos .NET examinaria esses registros, determinaria como integrar as informações e decidiria se copiaria ou não as informações recém-inseridas em outro registro, retiraria o registro ou disponibilizaria um novo participante ao aplicativo ASP.NET (no nosso exemplo).

A cada tabela SQL que tiver registros que foram replicados do dispositivo no servidor central, uma coluna é adicionada para manter o status do registro. Cada registro poderia ter um dos seguintes status: criado no servidor, não reconciliado, mesclado, retirado ou reconciliado. Se o registro for criado com o aplicativo ASP.NET, ele seria marcado como "criado no servidor". Esta marca de estado ajuda na replicação filtrada de registros do servidor para o cliente. Se um registro for criado no cliente, então será "não reconciliado". Quando este registro for replicado de volta no servidor, então estará no mesmo estado "não reconciliado", o que o identificará para o processo de resolução de conflitos .NET como registro que precisa ser processado. O aplicativo web ASP.NET ignorará todos os registros em estado "não reconciliado". Depois de processado, dependendo da ação executada pelo processo de resolução de conflitos, o estado do registro passaria para "mesclado", "retirado", "reconciliado" ou "desconhecido".

### Resolução de conflitos com o .NET

Os requisitos do aplicativo móvel de gerenciamento de eventos para manipular a resolução de conflitos vão além do que as tecnologias prontas são capazes de fazer. Com as transações lógicas e estado associados aos registros inseridos, a nossa meta era obter os dados todos em um único ponto para ser capazes de tomar decisões mais inteligentes sobre como manipular as informações. Parte do quebra-cabeça é determinar quando de fato tomar essas decisões inteligentes. As tecnologias padrão prontas tomam decisões no momento em que os registros são movidos. Em vez disso, precisamos ter um quadro completo de todos os registros inseridos (com tabelas-pai e tabelas de histórico) antes de determinar como reconciliar as informações. Um processo de negócios foi colocado em vigor para executar o processo

de resolução de conflitos .NET à 1h00, no horário do servidor, para reconciliar as informações replicadas do dia anterior.

Quando o processo começar, ele obtém todas as transações lógicas desde o dia em que o processo foi executado com sucesso pela última vez, até o dia anterior à data atual. Um participante pode ser representado por várias transações lógicas uma vez que as informações poderiam ser atualizadas diversas vezes em um ou mais computadores. Para várias atualizações do mesmo participante no mesmo computador, apenas o último inserido é processado. Dos registros remanescentes, as transações lógicas são agrupadas por participante e classificadas por data e hora de inserção.

Quando cada transação lógica é processada, verifica-se se o registro do participante foi ou não replicado originalmente do servidor central ou inserida no computador do cliente. Em caso afirmativo, os registros de histórico foram replicados de volta e associados corretamente. Estamos agora livres para implementar o máximo de detalhes necessários na determinação de quais regras de negócio devem ser executadas nas informações. Cada registro lógico referenciada pela transação lógica tem uma ou mais tabelas associadas. Podemos decidir simplesmente sobrescrever as informações mais recentes em todas as tabelas, ou uma ou mais tabelas, ou colunas específicas de uma das tabelas. Isto tudo depende das regras de negócio e pode ser tão complexo quanto for necessário.

Se uma transação lógica for associada a um registro de participante criado pelo cliente, então uma determinação deve ser feita quanto ao participante existir ou não no sistema. Neste momento, ele não é detectado pelo aplicativo ASP.NET pois está em estado "não reconciliado". Uma chave composta é construída com os registros de participante para identificá-los de forma unívoca. Os participantes existentes são examinados para saber se há uma correspondência com

o novo registro. Um destes três caminhos é possível: o novo registro de participantes já existe, é novo ou não há informações suficientes para se tomar uma decisão. Se o participante já existir, todos os registros históricos associados ao registro do participante no estado "não reconciliado" são atualizados para que apontem para o registro de participante pré-existente. As mesmas regras de quando um registro já existente é atualizado são aplicadas. Se o registro do participante definitivamente é novo, o seu estado é alterado para "reconciliado" e ele é disponibilizado ao aplicativo ASP.NET. Se não puder ser feita nenhuma determinação, o estado do registro é alterado para "desconhecido" e um aplicativo web de gerenciamento de exceções é executado para se saber o que fazer com o registro.

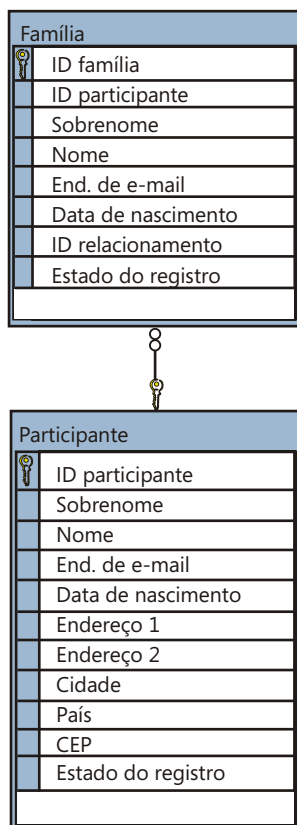
Este processo é repetido para cada transação lógica. Todo trabalho da preparação deste processo foi executado, almejando simplificá-lo o máximo possível e centralizar toda a complexidade em um único ponto.

### Conclusão

A construção de aplicativos móveis é um desafio. Não existe um framework mágico que podemos utilizar para habilitar cenários desconectados. Temos ferramentas poderosas, mas sem personalização, que servem apenas para os cenários mais simples. A abordagem descrita neste artigo é genérica o bastante para que seja aplicada em diversos tipos de aplicativos desconectados. Confiar no poder do SQL Replication para executar apenas as inserções nos permitiram focar a resolução de conflitos no processo.NET. Embora seja necessário muito trabalho com dados para aumentar o esquema do banco de dados para que viabilizar o aplicativo, acreditamos que o processo de resolução de conflitos tenha sido, o máximo possível, simplificado.

Esta abordagem não é nova e tem limitações que a tornam apropriada para apenas um subconjunto de aplicativos móveis. Muitos cenários de mobilidade, por exemplo, demandam a resolução de dados em tempo real no mesmo momento da replicação no servidor central, bem diferente das muitas horas de atraso propostas nesta solução. A replicação unidirecional apenas dos registros inseridos não seria viável, uma vez que a nossa opção aqui era centralizar quando e onde executar a resolução de conflitos. A perda de dados devida à resolução de conflitos também é relegada à intervenção manual nesta solução. Aproveitamos a demanda de reter e replicar todas as alterações de registros centralmente, que tornou-se as transações nas quais baseamos o nosso processo de resolução de conflitos. Se você não tiver esta demanda, a carga é transferida para a replicação de mais dados que você poderia estar disposto a ter ou ser capaz de processar. Finalmente, o processo de negócio que orienta a sua solução ditará o quão complacente você poderá ser com as suas opções.

**Figura 5:** A coluna RecordState (estado do registro) é adicionada para registrar de onde vem o registro e como foi processado.



### Sobre o autor

**Rodney Guzman** é o CTO e co-fundador da InterKnowlogy. Começou a trabalhar com sistemas de software em sonares submarinos durante a faculdade. Durante sete anos na SAIC, Rodney foi desenvolvedor líder e arquiteto de projetos como um grande portal na web baseado em Java SOA HTTP/XML para hospitais militares espalhados pelo país. Em 1998, Rodney passou para a Stellcom para trabalhar em projetos mais centrados na Microsoft, incluindo implementações de servidor de sites e uma framework de segurança corporativa para a Pacific Life, que permitiu a implantação de políticas personalizadas derivadas de grupos e atributos AD para dirigir a personalização e a segurança em sites da ASP na web. Na InterKnowlogy, Rodney define as direções tecnológicas, atuando como arquiteto líder de grandes projetos, como uma implementação de SOA de grande amplitude com framework de cliente inteligente, criando grandes propriedades web Microsoft (CommNet e Partner Campaign Builder) e grandes implementações MOSS. Rodney foi o arquiteto do aplicativo WPF/MOSS Scripps Cancer. Rodney deu palestras em diversos eventos da Microsoft e é autor de muitos artigos. Ele é membro da Commerce PAC e Microsoft Architectural Advisory Board, e é arquiteto de soluções MVP.



# Desenvolvimento Dirigido a Teste e Integração Contínua para Aplicativos Móveis

por Munjal Budhabhatti



## Resumo

Este artigo demonstra como o desenvolvimento dirigido a teste e a integração contínua enfrenta os desafios exclusivos da criação de aplicativos Windows Mobile.

## Estado atual do desenvolvimento de aplicativos móveis: questões e desafios

Globalmente, o número de assinantes de telefones celulares chega a 2,5 bilhões, com previsões de que atinja 4 bilhões até 2010. O dispositivo móvel é, hoje, uma plataforma mais rica para o oferecimento de aplicativos por causa deste crescimento exponencial e utilização ampla. O fator crítico, como sempre, é a experiência do usuário: usabilidade, confiabilidade e desempenho do aplicativo.

Para complicar ainda mais, o mundo do desenvolvimento de software está passando de ciclos de implantação semanais e mensais para a implantação contínua. Logo, como é possível garantir que um usuário tenha sempre a melhor experiência?

Muitos que já se familiarizaram com este ambiente de mobilidade devem conhecer duas práticas centrais de programação extrema: o desenvolvimento dirigido a testes automatizados, um estilo de desenvolvimento introduzido por Kent Beck, intitulado “desenvolvimento dirigido a teste”; e a prática de desenvolvimento de software de integração frequente de builds, batizada de integração contínua por Matthew Foemmel e Martin Fowler.

Estas práticas não são novas no mundo do software. Contudo, o desenvolvimento de aplicativos móveis demorou para aproveitar as vantagens do desenvolvimento dirigido a teste e da integração contínua endossados pela comunidade de software corporativo. Isto é consequência parcial do apoio limitado ou inexistente de uma plataforma móvel nos conjuntos de ferramentas existentes, como o NUnit/MSTest e o Cruisecontrol.net/Team Foundation Server.

Poucas ferramentas de testes de mobilidade permitem a gravação de interações com usuário via representação gráfica do dispositivo cliente, mas não proporciona o controle granular dos testes. Outras ferramentas demandam a presença de scripts nos dispositivos móveis ou dependem da execução de testes, manualmente, no dispositivo. Assim, os testes de aplicativos móveis são ineficientes e complexos, o que dificulta a produtividade.

## Desenvolvimento dirigido a teste

O desenvolvimento dirigido a teste (TDD) é uma abordagem evolucionária na qual o desenvolvimento do código é direcionado pela escrita inicial de um caso de teste automatizado, seguido pela escrita do código para atender ao teste e, por fim, a refatoração.

## O TDD é basicamente o desenvolvimento de um teste seguido da refatoração

Vermelho/verde/refatoração — o mantra do TDD — é a ordem da tarefa da programação:

1. *Vermelho*: escrever um pequeno teste de unidade automatizado malsucedido, que talvez nem mesmo seja compilado inicialmente (vide Figura 1, página 32).

2. *Verde*: escrever o código necessário para passar pelo teste malsucedido. Garantir que passe em outros testes também, se houver, no conjunto. Inserir o código no repositório de código-fonte (vide Figura 2, página 32).

3. *Refatoração*: aprimorar o código existente, em etapas curtas e incrementais, sem mudar a finalidade (vide Figura 3, página 33).

Portanto, esta técnica é inversa à da programação convencional: desenvolver o código e depois escrever um teste, executado manual ou automaticamente. Por que adotar uma mudança dessas, especialmente quando tendemos a considerá-la trabalho extra? Na realidade, o desenvolvimento dirigido a teste é uma programação a prova de riscos, que investe tempo no início para evitar falhas (ou ainda mais trabalho) no final. Kent Beck chamou-a de “uma maneira de administrar o medo durante a programação”.

**“O DESENVOLVIMENTO DIRIGIDO A TESTE É UMA MANEIRA DE ADMINISTRAR O MEDO DURANTE A PROGRAMAÇÃO — NÃO O MEDO NO MAU SENTIDO, MAS SIM O MEDO LEGÍTIMO. SE A DOR É UMA FORMA NATURAL DE DIZER, 'PARE!' O MEDO É UMA FORMA NATURAL DE DIZER 'CUIDADO'”.**

**—KENT BECK**

## Os benefícios do TDD

- *Aprimoramento de projeto*. Escrever um caso de teste completo força a criação de um código desacoplado (não vinculado estritamente a outro código), aumentando assim a sua coesão e diminuindo seu grau de vinculação.
- *Documentação*. Um caso de teste de unidade bem escrito proporciona uma especificação de trabalho e comunica a finalidade do código de forma clara. Além disso, sempre que o código for alterado, o caso de teste de unidade deve ser atualizado para passar pelo conjunto de testes. Portanto, um caso de teste de unidade permanece sincronizado com o código, naturalmente. Esta característica é um diferencial em relação aos casos de teste de unidade convencionais, desenvolvidos com o Microsoft PowerPoint ou o Microsoft Word. Embora documentos desse tipo comecem com boas intenções, com o tempo o resultado muitas vezes fica fora de sincronismo com a implementação subjacente.

**Figura 1:** Exemplos de casos de teste

```
namespace OutlookContacts.Tests
{
    //Ensure that OutlookContact compares correctly with other contacts.
    [TestFixture]
    public class OutlookContactTest
    {
        [Test]
        public void EnsureNullContactIsNotEqualToContact()
        {
            OutlookContact contact = new OutlookContact();
            Assert.AreNotEqual(null, contact);
        }
        [Test]
        public void EnsureContactsWithSameNameAreEqual()
        {
            OutlookContact contactMain = new OutlookContact("foo");
            OutlookContact contactOther = new OutlookContact("foo");
            Assert.AreEqual(contactMain, contactOther);
        }
    }
}
```

- *Mudança segura no sistema.* O TDD fornece feedback contínuo quanto ao resultado das mudanças do código em outras partes do sistema.
- *Recuperação rápida de falha.* O teste de unidade pode isolar problemas rapidamente, reduzindo as atividades de depuração e permitindo que o sistema recupere-se de falhas com agilidade.
- *Código nítido.* Código nítido significa que sua finalidade é expressa com clareza, pode ser alterado para receber novos recursos e não tem duplicação. A refatoração mantém a semântica comportamental do código, nunca adiciona ou remove funcionalidades. Trata-se de aperfeiçoar a qualidade do código, o que gera valor comercial.

A execução do teste de unidade automatizado é uma das exigências cruciais do TDD. Contudo, como as ferramentas de teste ainda estão evoluindo, a execução automática não é viável hoje para o desenvolvimento de aplicativos móveis. A implementação de TDD neste ambiente, portanto, é bastante desafiadora, se não impossível.

**Figura 2:** Exemplo de código

```
namespace OutlookContacts
{
    public class OutlookContact : Contact
    {
        public override bool Equals(object other)
        {
            if (other == null) return false;
            OutlookContact otherContract = (OutlookContact)other;
            return otherContract.AccountName == AccountName;
        }
    }
}
```

## Testes de unidade

Normalmente considera-se o processo de teste um processo metódico de comprovação da existência ou da ausência de falhas em um sistema. Quando um caso de teste é escrito antes da escrita do código, o caso de teste torna-se uma especificação, em vez de uma mera verificação do recurso.

Os testes também são uma forma de documentação dos defeitos encontrados. Vamos pressupor que um defeito tenha sido descoberto na fase de garantia de qualidade, durante o teste de alguns trechos de código recém-implantados. Mesmo que esse defeito seja muito simples de ser corrigido, o TDD exige um caso de teste. Primeiro, é necessário escrever um caso de teste que simule o comportamento de falha e, depois, o código para passar no teste. Esta prática garante que os defeitos, não importa o quão insignificantes, não se espalhem pelo sistema e que os testes de regressão tornem-se parte do conjunto de testes. A execução local do teste automático, antes de confirmar a aplicação das alterações no repositório de controle de fonte (SCR – Source Control Repository), mitigaria ainda mais o fenômeno de builds fragmentados.

É importante preparar o ambiente de teste em um emulador que seja o mais semelhante possível ao hardware almejado. O desenvolvimento e teste de aplicativos

Windows Mobile em um emulador x86 faz pouco sentido quando o hardware almejado é exclusivo para a arquitetura ARM, que predomina em equipamentos eletrônicos de baixo consumo de energia. Além disso, no mundo real, os componentes muitas vezes têm dependências de outros objetos, bancos de dados ou conexões de rede. É muito fácil cair na armadilha de pressupor que tais dependências funcionem perfeitamente. Portanto, se os testes forem escritos sem levar em conta as dependências, um feedback de erro poderá decorrer dos testes que falharem por falhas de dependências.

Uma forma de se proteger contra as dependências é construir o gráfico de objetos ou configurar o banco de dados em um estado predefinido antes da execução do caso de teste. Isto resolveria o nosso problema, mas aumentaria o tempo de execução do teste e de construção.

Uma abordagem mais elegante seria instanciar os objetos de teste e substituir a dependência do objeto pela implementação de imitadores ou stubs, que simulam o comportamento de objetos reais. Isto garante a execução de testes isolados e, assim, resultados confiáveis. Deve-se ter cuidado na simulação de objetos reais com imitadores ou stubs. É provável que todo o conjunto de testes de unidade seja executado sem falhas, e mesmo assim o produto não passe nos testes de garantia de qualidade. De acordo com a minha própria experiência, complementar objetos imitadores e stubs com testes de integração provê um sentido real de confiança.

## Integração contínua

Martin Fowler descreveu a integração contínua (CI) como uma prática de desenvolvimento de software de integração frequente de builds, não raramente, muitas vezes por dia. O fluxo de trabalho típico da CI, como mostra a Figura 4, seria:

### Desenvolvedor:

1. Escreve um novo caso de teste de unidade.
2. Executa o caso de teste de unidade localmente no emulador e confirma que o caso de teste falhou.

3. Adiciona código ou modifica o existente para que passe no caso de teste.
4. Executa o caso de teste de unidade localmente no emulador e confirma que o caso de teste passou.
5. Confirma o código no SCR.

#### *Servidor de CI:*

6. Faz o download do código-fonte sempre que ocorre uma mudança no SCR.
7. Compila e examina o código-fonte, e cria novos binários.
8. Define as dependências externas, como esquemas de banco de dados, e recursos (arquivos de configuração, conjuntos satélites).
9. Implanta os novos binários e executa os testes no emulador.
10. Faz o encapsulamento e implanta o produto em um ambiente de homologação.
11. Gera um feedback com base nos resultados do build.

#### **Repositório de código-fonte**

Todos os arquivos essenciais necessários para construir um produto residem no repositório de código-fonte (SCR). O SCR tem um papel importante no ciclo de vida do desenvolvimento de software e CI. Ferramentas do SCR, como o controle de fonte Subversion e Visual Studio Team Foundation, permitem que as equipes trabalhem em colaboração (simultaneamente com os mesmos artefatos ou artefatos diferentes), rastreiem alterações de código de forma simples e trabalhem em versões diferentes de arquivos, ao mesmo tempo.

O servidor de CI obtém a última versão do código-fonte do SCR, localiza todas as dependências necessárias e constrói o produto independentemente do build anterior. O SCR possibilita uma maior produtividade da equipe: um novo membro não precisa reconfigurar bibliotecas de terceiros, estruturas de projeto ou configurações IDE do projeto. Além disso, o tempo de depuração é reduzido, o que possibilita à equipe remover as alterações atuais, que seriam pequenas e incrementais se as práticas do TDD tiverem sido adotadas. O sistema poderia retomar a versão anterior do código, com segurança.

### **“A INTEGRAÇÃO CONTÍNUA É UMA PRÁTICA DE DESENVOLVIMENTO DE SOFTWARE NA QUAL OS MEMBROS DE UMA EQUIPE INTEGRAM O SEU**

Figura 3: Exemplo de refatoração

```
namespace OutlookContacts
{
    public class OutlookContact : Contact
    {
        public override bool Equals(object other)
        {
            if (other == null) return false;
            return CheckEqualityForOutlookContactObject(other);
        }
        private bool CheckEqualityForOutlookContactObject(object other)
        {
            OutlookContact otherContract = (OutlookContact) other;
            return otherContract.AccountName == AccountName;
        }
    }
}
```

**TRABALHO COM FREQUÊNCIA (NORMALMENTE CADA PESSOA FAZ A INTEGRAÇÃO PELO MENOS UMA VEZ POR DIA), RESULTANDO EM VÁRIAS INTEGRAÇÕES DIÁRIAS. CADA INTEGRAÇÃO É VERIFICADA POR UM BUILD AUTOMÁTICO (INCLUINDO TESTE) PARA DETECTAR ERROS DE INTEGRAÇÃO O MAIS RÁPIDO POSSÍVEL. MUITAS EQUIPES CONSIDERAM QUE ESTA ABORDAGEM LEVA A UM NÚMERO SIGNIFICATIVAMENTE MENOR DE PROBLEMAS DE INTEGRAÇÃO E PERMITE O DESENVOLVIMENTO DE UM SOFTWARE COESO COM MAIS RAPIDEZ”.**

**—MARTIN FOWLER**

É essencial incluir todas as dependências no SCR: o instalador de framework Windows Mobile SDK, .Net Compact, os drivers de serviços Virtual Machine Network e outros componentes e utilitários de terceiros.

#### *Build automático*

Ao contrário de algumas opiniões equivocadas, o build automático nos aplicativos móveis é muito mais que uma compilação de código. Ele monta o código-fonte do SCR, compila-o para a criação de binários e dependências (como objetos de configuração, montagens de recursos e assim por diante), examina e implanta os binários compilados em um dispositivo móvel ou emulador, carrega os esquemas de banco de dados e executa testes remotamente no dispositivo móvel.

Ferramentas de build como Nant e MSBuild não são compatíveis com a implantação em aplicativos móveis, em parte porque a compatibilidade com dispositivos móveis nessas ferramentas ainda é incipiente. Além disso, as ferramentas de teste de unidades como NUnit e MSTest apresentam um problema semelhante de execução de testes remotos em um dispositivo móvel. Para evitar tais limitações, uma nova ferramenta é fundamental.

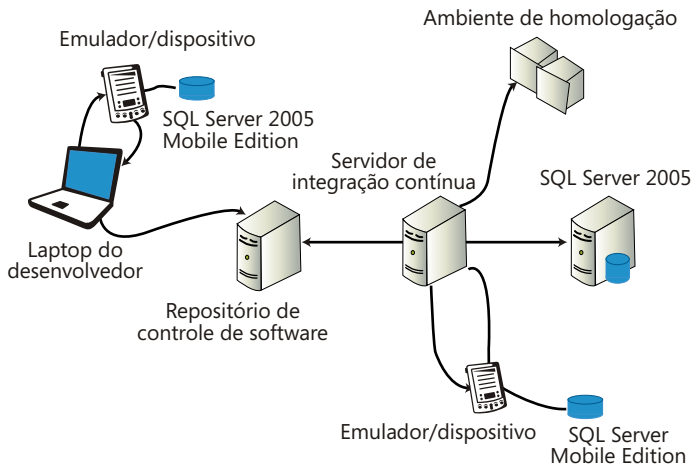
#### **wMobinium.net**

Para abordar esses problemas, criei uma ferramenta chamada wMobinium.net, que ajuda na implementação do TDD e da CI nos aplicativos .Net. A wMobinium.net é uma ferramenta de testes de unidade compatível com a implantação e a execução automática de testes de unidade remotos. Ela oferece um add-in do Visual Studio para a compatibilidade com o TDD nos aplicativos móveis .Net. É uma ferramenta disponibilizada livremente no site de código aberto CodePlex (vide Referências).

#### *Implantação automática*

Diferentemente do desenvolvimento de aplicativo para desktop, o desenvolvimento de aplicativos móveis enfrenta desafios únicos na implantação: primeiro, a implantação é necessária para testes de unidade no dispositivo; segundo, a implantação também é necessária após um build bem sucedido, para levar a solução em funcionamento para os testes de garantia de qualidade no ambiente de homologação.

Figura 4: A integração contínua no aplicativo .Net Windows Mobile



Para cada build, as dependências e binários recém-compilados devem ser copiados para uma pasta de programa no dispositivo. Quando um aplicativo demanda o teste em vários dispositivos, haverá outras complicações para o processo de implantação. Para evitar esses problemas, a wMobinium.net oferece uma ferramenta de implantação que implementa a Remote Application Programming Interface (RAPI) do Windows CE, e facilita a implantação de arquivo e pasta. Isto alivia o trabalho de implantações manuais.

### Confirmações frequentes

Um dos componentes-chave de uma implementação de CI são as confirmações frequentes e, em consequência, os builds frequentes. Com intervalos de confirmação mais longos, os membros da equipe tendem a trabalhar isolados uns dos outros e o build, por sua vez, tende a apresentar mais problemas de integração. Quando um membro da equipe dedica mais tempo a um recurso e encontra problemas de integração durante a confirmação do código, ele cria mais resistência em remover as alterações e reverter-lo à versão anterior. Esta relutância pode aumentar o tempo e os recursos investidos nas atividades de depuração. Em um cenário ideal, os membros da equipe inserem o código em intervalos de 30-60 minutos ou menos. O intervalo de inserção pode se prolongar por algumas horas, mas deve sempre ser inferior a um dia.

### Builds bem-sucedidos mais rápidos

Quando um desenvolvedor confirma o código no SCR, esperar pelo feedback da CI retarda o processo de desenvolvimento. Esperas maiores resultam em menor produtividade. Além disso, alguns dos subprocessos do build, como implantação e testes, são executados em um dispositivo/emulador, o que torna tais processos inerentemente mais lentos do que seriam em um desktop.

Para reduzir os tempos de build, é preciso concentrar-se no elo mais frágil: o componente que demora mais para ser executado. Com mais frequência, a causa seria uma dependência externa, como um banco de dados ou outro objeto. Acessar um banco de dados e configurar os dados para o caso de teste é uma operação que consome muitos recursos. Como mencionei antes, imitadores ou stubs devem dar conta do artifício. Se for impraticável fazer isso, deve-se passar o caso de teste para builds secundários ou noturnos (builds programados para serem executados à noite, ou quando a maioria dos recursos estiverem ociosos). Os casos de teste destinados a cenários em diversos

dispositivos também devem ser adicionados a builds secundários e noturnos.

Builds com erro são responsáveis pela pior frustração, como se toda a agitação da cadeia de produção do software recebesse um banho de água fria. O código no SCR não é mais confiável e toda a equipe fica paralisada diante do último código-fonte. A equipe precisa agir rapidamente e resolver a questão, consertando o build. Se o caso de teste for a causa do problema e a resolução for muito demorada, é mais seguro ignorar o caso de teste, temporariamente, para permitir que o build seja bem-sucedido. Contudo, é crucial manter o controle dos casos de teste ignorados em um painel de projeto de fácil acesso, um quadro branco ou um painel virtual usando um software colaborativo. Os testes ignorados devem ser corrigidos, posteriormente, e reintegrados à suíte de testes.

### Testes de unidade automáticos

É bastante comum verificar os mesmos defeitos voltando à tona após alguns builds, e muitas vezes observamos um analista de garantia de qualidade dizer: "mas este defeito já havia sido corrigido no build anterior". Os defeitos-bumerangue revelam a importância de escrever um teste para cada defeito encontrado antes de modificar a base do código. Depois de corrigido o caso de teste, todo o conjunto de testes deve ser executado e ser bem-sucedido antes de inserir o código modificado no SCR.

Participei de alguns projetos em que a equipe executou o conjunto de testes manualmente no dispositivo móvel. Imagine um desenvolvedor de aplicativo móvel que gasta alguns minutos modificando uma funcionalidade, mas o dobro do tempo para testá-la manualmente. Isto não apenas irá desestimular o desenvolvedor como também afetar a produtividade.

A wMobinium.net resolve esta chateação pela automatização de todo o fluxo de trabalho de testes de unidade. Diferentemente dos testes de unidade convencionais, a wMobinium.net apresenta a seleção do caso de teste no desktop, executa os testes no dispositivo e exibe os resultados no desktop. Essa ferramenta lida com complicações do tipo:

### Execução remota de casos de teste

Para executar os casos de teste remotamente em um dispositivo/emulador, a ferramenta serializa as informações de metadados dos casos de teste selecionados, inicia um processo condutor e executa os testes no dispositivo. Para fornecer os relatórios corretos ao servidor de CI, o processo remoto deve ser iniciado de forma sincronizada e monitorado continuamente, o que é um desafio e tanto.

### Serialização dos resultados do teste no desktop

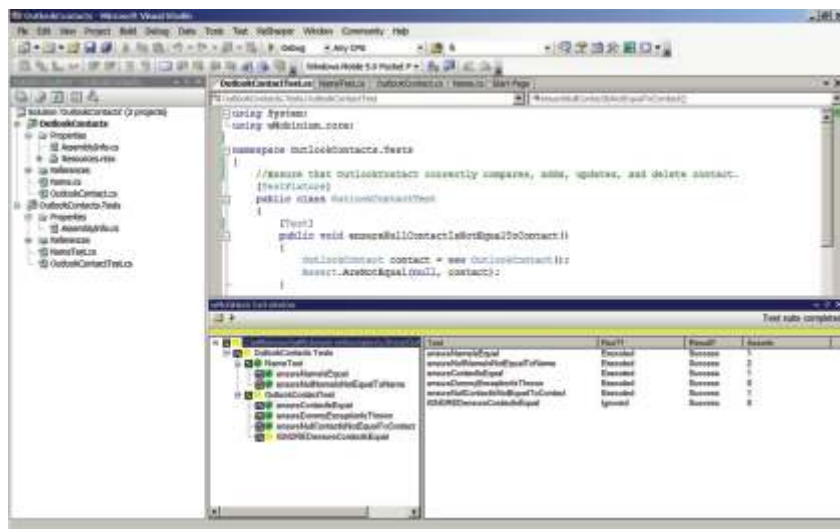
Na falta de compatibilidade com recursos remotos no .Net Compact Framework versão 2.0, o dispositivo deve se comunicar com o desktop por soquetes. Os eventos devem ser serializados, enviados ao desktop por um soquete, de-serializados e propagados aos ouvintes de eventos apropriados.

### Add-in da wMobinium.net

As ferramentas descritas aqui ajudam o servidor de CI a continuamente construir, implantar e testar um aplicativo móvel .Net. Seria conveniente se o recurso de testes de unidade fosse compatível na forma de uma ferramenta integrada ao Visual Studio.



Figura 5: Add-in da wMobinium.net



O add-in da wMobinium.net, um suplemento do Visual Studio (Figura 5), faz parte do conjunto de ferramentas da wMobinium.net. Após ativar o add-in, todos os testes disponíveis na solução são exibidos na janela de ferramentas. Um workflow típico seria:

1. A ferramenta wMobinium.net exibe os casos de teste disponíveis em uma solução.
2. O usuário seleciona os casos de teste a serem executados.
3. Os casos de teste selecionados são serializados e enviados ao dispositivo/emulador conectado.
4. Os casos de teste são executados no dispositivo/emulador e os resultados, devolvidos ao desktop.
5. A ferramenta exibe os resultados no desktop.

### Os benefícios da utilização da CI

Partes interessadas e patrocinadores de projeto sempre dão preferência a resultados confiáveis, comunicações claras, visibilidade do projeto e qualidade superior de software. O desenvolvimento de software, todavia, raramente oferece essas qualidades sem os processos e práticas corretos.

Quando tudo parece estar dando certo, qualquer defeito poderia subitamente ameaçar o cronograma de desenvolvimento. Especialmente durante as integrações "big-bang", até mesmo os menores problemas — como perder entradas de configuração, banco de dados não sincronizado ou dependências perdidas — poderiam ser muito problemáticas quando encontradas em conjunto.

A integração contínua permite um feedback mais rápido. A cada alteração (adição de recursos novos ou modificação de existentes, não importa o tamanho deles), o servidor de CI integraria as novas partes, que passariam por todo o ciclo automático de build (compilação, testes, inspeção e implantação). Isto dá visibilidade do progresso do projeto, aumenta a qualidade do desenvolvimento de software e gera espírito de equipe.

A CI não proporciona essas funcionalidades prontas. É bastante possível implementar a CI sem incluir testes ou inspeções automáticos no processo de build. Contudo, esta opção seria a menos benéfica. Muitos, e eu me incluo entre eles, consideram que CI sem teste não é CI.

### Conclusão

Da perspectiva do usuário, a implementação do TDD e da CI é a mesma em um aplicativo tradicional de desktop e em um aplicativo móvel. O usuário cria um novo caso de teste automático de falha, escreve o código para passar no teste e refatora o código sem alterar sua finalidade. O servidor de CI pesquisa o código-fonte mais recente, cria novos binários, executa os testes e gera o feedback. No entanto, em um aplicativo móvel, a implementação difere quanto à execução remota dos casos de teste, à notificação dos resultados do teste e às implantações de build. Essas complexidades foram abordadas pela ferramenta wMobinium.

Minha experiência em uma das maiores organizações de microfinanças na África inclui, junto com a minha equipe, o desenvolvimento um aplicativo móvel .Net. Na falta de ferramentas de apoio, o desenvolvimento com a implementação do TDD e da CI, embora árduo, melhorou o design geral do aplicativo, a confiabilidade e o desempenho.

Com a liberação do .Net Compact Framework versão 2.0, o desempenho dos aplicativos móveis .Net aumentou radicalmente. A versão mais nova proporciona uma produtividade maior ao desenvolvedor, maior compatibilidade com todo o framework .Net e maior apoio para a depuração de dispositivo. A combinação do .Net Compact Framework com o TDD e a CI (usando a wMobinium.net) resultaria em grandes benefícios a uma organização e levaria a plataforma de aplicativos móveis a um outro patamar.

Com a proliferação atual de novos dispositivos móveis, o aplicativo móvel tornou-se parte essencial de uma oferta de produto corporativo mais ampla. É imperativo, mais do que nunca, tirar o desenvolvimento de aplicativos móveis do isolamento e incluí-lo em iniciativas de desenvolvimento dirigido a teste e integração contínua de âmbito corporativo.

### Referências

"Continuous Integration," Martin Fowler and Matthew Foemmel  
<http://www.martinfowler.com/articles/continuousIntegration.html>

Continuous Integration: Improved Software Quality and Reducing Risk, Paul Duvall, Steve Matyas, and Andrew Glover (Addison-Wesley Professional, 2007)

Test-Driven Development: By Example, Kent Beck (Addison-Wesley Professional, 2002)

### wMobinium.net

<http://www.codeplex.com/wMobinium>

### Sobre o autor

Munjal Budhabhatti é desenvolvedor sênior de soluções na ThoughtWorks. Ele tem mais de 10 anos de experiência no projeto de aplicativos corporativos de larga escala e implementou soluções inovadoras em algumas das maiores organizações de microfinanças, seguros e financeiras na África, Ásia, Europa, e América do Norte. Munjal ocupa a maior parte do seu tempo escrevendo aplicativos corporativos bem projetados, utilizando processos ágeis.



# Estudo de Caso: Técnicos de Suporte de Campo

por András Velvárt e Peter Smulovics

## Resumo

A Monicomp é a maior empresa de serviços de manutenção de equipamentos de bancos na Hungria. A organização instala, mantém e repara terminais de ponto de serviço (POS), caixas automáticos (ATM) e outros equipamentos bancários semelhantes, usando uma equipe de centenas de técnicos espalhados por todo o país. Toda empresa que precisa executar essas tarefas e tem de seguir a ISO 9000 necessita de todo o auxílio de TI possível. A organização tem que controlar o local onde cada metro de um cabo de dois quilômetros deve ser usado, de forma que, se houver qualquer problema com o cabo, seja possível voltar em cada loja por onde o cabo passa e executar os reparos antes que surjam mais problemas. Além disso, os clientes demandam informações atualizadas em tempo real — e isso pode incluir desde saber como estão as atualizações de software em âmbito nacional até o que está havendo com o terminal de POS da Pet Shop do sr. Silva.

Como era de se esperar, um método de trabalho baseado em papel não daria conta das exigências. Seria impraticável os técnicos irem a campo e enviar planilhas em papel para processamento no final da semana. É necessário uma solução do século 21. O sistema "MŰVÉSZ", elaborado por uma empresa de consultoria e desenvolvimento de software chamada Response, cobre toda a operação de serviços, desde o inventário até o faturamento; desde a geração de notificações de problemas, passando pela distribuição até a verificação do trabalho de mais de 100 técnicos executando tarefas de manutenção e instalação. Neste artigo, mostraremos como a arquitetura deste sistema é projetada e

## Workflow em planilhas

Antes que apresentemos a visão geral da arquitetura do aplicativo, vamos dar uma olhada nos importantes processos de workflow da organização. Para a Monicomp, a entidade mais importante de um workflow móvel é a planilha. A planilha é um item de trabalho que deve ser executada por um técnico de campo. Vamos examinar os conceitos-chave de um workflow simplificado:

**Criação da planilha.** A planilha normalmente é criada pelo expedidor. Quando uma planilha é criada, ela apresenta as seguintes características principais:

- O tipo de planilha, que categoriza o trabalho a ser executado: instalação, reparo ou desinstalação.

- Uma referência ou endereço do parceiro, onde o serviço deverá ser executado.
- Informações sobre a notificação de problema original que o expedidor registrou (se houver).
- Identificadores e dados adicionais dos dispositivos que serão instalados ou desinstalados.
- O identificador da planilha em si.
- Prazo do trabalho.

**Atribuição da planilha.** Para dar início ao workflow, o expedidor atribui uma planilha a um técnico após analisar a carga de trabalho e a posição geográfica dos técnicos disponíveis. Para facilitar essa tarefa, as posições geográficas dos técnicos são coletadas por dispositivos de rastreamento Global Positioning System (GPS) e exibidas em um mapa Virtual Earth, com ícones codificados por cores que indicam a carga de trabalho atual. (Vide Figura 1.)

**Download da planilha.** Após a atribuição de uma planilha ao técnico, uma mensagem Short Message Service (SMS) é enviada. A SMS notifica o técnico e o dispositivo Ultra-Mobile PC (UMPC) de que há um novo trabalho a ser feito. Com uma sincronização via General Packet Radio Service (GPRS), Edge, 3G ou uma rede sem fio (dependendo da disponibilidade na área em que se encontra o técnico), o download da nova planilha é feita no aplicativo de processamento de planilhas executado pelo UMPC.

**Figura 1:** Uso do Virtual Earth Services para selecionar o técnico que esteja mais próximo do destino e com menos carga de trabalho



**Preenchimento da planilha.** O técnico examina a planilha recebida, vai até o local onde o trabalho deve ser executado e faz o que tem de fazer. Durante o reparo, instalação, manutenção REST/POX ou trabalho de desinstalação, ele registra os detalhes do dispositivo em que está trabalhando (como o número de série) e atribui os equipamentos e materiais de seu inventário ao dispositivo. Ele também pode usar o aplicativo de processamento de planilhas para analisar as especificações técnicas do dispositivo em questão, se o reparo tornar-se complicado. Além disso, as especificações podem ser anotadas com o reconhecimento de escrita e compartilhadas em uma base de conhecimentos.

Outros dados que devem ser registrados incluem a hora de chegada ao local, o número de horas que durou o serviço executado pelo(s) técnico(s) e a distância de ida e de volta ao local. Você conhecerá mais detalhes do aplicativo de processamento de planilhas na seção "Interface de usuário com WPF", mais adiante neste artigo.

**Fechamento da planilha.** Depois de preenchida, a planilha é impressa em uma impressora móvel e assinada pelo cliente. A assinatura também poderia ser capturada pela tela sensível ao toque do UMPC com o reconhecimento de escrita (assim os técnicos não precisariam carregar as impressoras com eles); todavia, por questões jurídicas, esse recurso não é utilizado ainda.

**Upload da planilha.** O banco de dados do UMPC é sincronizado, e as alterações da planilha são carregadas no servidor central. A lógica do negócio no servidor registra o trabalho executado, remove os materiais utilizados do inventário do técnico e executa todas as outras tarefas de manutenção. O mecanismo de sincronização também é muito útil por criar backups atualizados dos bancos de dados dos técnicos. No caso de quebra ou perda de um UMPC, o técnico pode retomar rapidamente o seu trabalho porque o seu banco de dados pode ser recriado como era no momento da última sincronização.

## Dois tipos de sincronização

Os UMPC clientes podem executar dois tipos de sincronização. Para economizar largura de banda e vida útil da bateria, a sincronização rápida apenas transfere os dados diretamente relacionados com a planilha do técnico. A sincronização plena (docking) faz o download de um conjunto de dados completo para o UMPC (por exemplo, alterações na equipe de trabalho ou atualizações de software). Logo, é normalmente realizada apenas via Internet na casa do técnico.

## Visão geral da arquitetura — O quê e por quê?

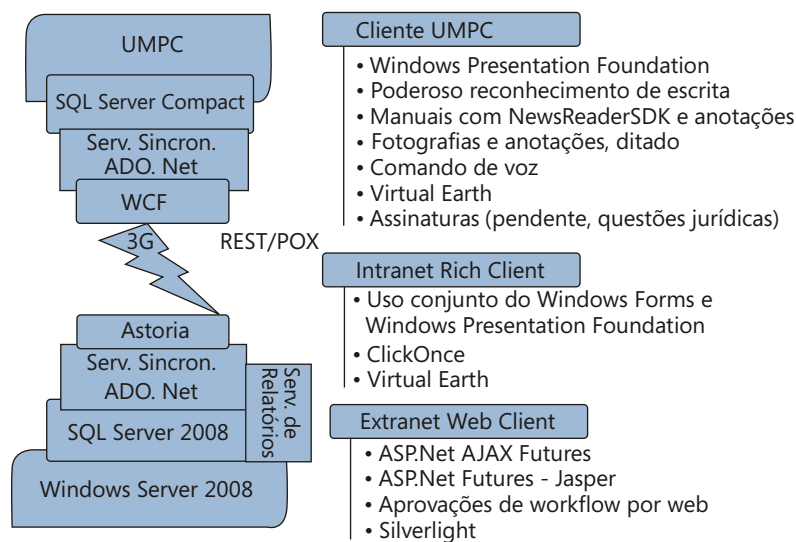
A arquitetura, independentemente de estar definindo conexões entre caixas brancas e preta, exibindo caracteres em uma linha de comando, acionando serviços na web ou utilizando um Office Business Application é, por natureza, determinada pelo tipo de qualidade da conexão. O processamento móvel, apesar da ideologia atual de conexão contínua, é, na realidade, de conexão ocasional. Assim, como deve ser planejada uma arquitetura para cenários de conexão ocasional?

Nesta seção, resumiremos algumas questões centrais que enfrentamos no planejamento e na implementação dessa arquitetura móvel e de alta complexidade. Para começar, vamos dar uma olhada na Figura 2, que mostra os componentes e links de comunicação principais de nossa arquitetura móvel.

## Conexão ocasional

A consideração mais importante em uma arquitetura de conexão

Figura 2: Visão geral dos principais componentes e seus links de comunicação



ocasional é a sincronização, seja ela unidirecional, bidirecional, lide com lógica de resolução de conflitos e apresente os dados em uma interface de usuário. Não abordar esses problemas pode causar problemas de uniformidade; resolver os problemas torna a lógica do negócio muito mais complicada que um cenário puramente off-line ou on-line. Para diminuir o tempo de chegada ao mercado, selecionamos os ADO.NET Synchronization Services, introduzidos com outras partes do conjunto de ferramentas, para criar uma solução automatizada, padronizada e menos propensa a erros.

## Componentes-chave do sistema

No coração do sistema está um servidor de aplicativo de múltiplas camadas, que executa o armazenamento de dados e a funcionalidade da lógica do negócio. Diversos tipos de clientes utilizam os seus serviços:

- Um aplicativo Windows Forms utilizado na matriz pelos responsáveis pelo estoque, expedidores e técnicos que executam serviços de reparo que não podem ser feitos no local. O aplicativo é implantado com o ClickOnce, que ajuda a garantir atualizações de versões de forma transparente.
- Um aplicativo web fora da rede, com interface fluida e ágil, utilizado pelos maiores clientes para a colocação de centenas de pedidos em um único lote, priorizar e rastrear o andamento desses pedidos.

Um aplicativo Windows Presentation Foundation (WPF) executado em centenas de UMPCs, utilizado por técnicos que executam tarefas de instalação, reparo e manutenção por todo o país.

## A configuração do servidor

Como servidor de back-end, consideramos que o Windows Server 2008 é uma solução ideal para os cenários de Software + Serviços como este (principalmente porque é capaz de hospedar serviços do Windows Communication Foundation (WCF) e atender e enfileirar solicitações HTTP no nível de kernel, e, ao mesmo tempo, gerenciar de uma perspectiva de infra-estrutura.



Utilizamos uma instância do SQL Server 2008 executando no servidor para fornecer recursos como armazenamento de dados espaciais e servir, à medida que os dados são armazenados, como solução de relatórios ao aplicativo de expedição e site externo na web. Os problemas de sincronização mencionados anteriormente foram resolvidos pelos ADO.NET Synchronization Services, com a solução "Database in the Cloud" (banco de dados na nuvem) da Microsoft, chamada Astoria, sobre a qual o leitor pode obter mais detalhes na edição 13 do Architecture Journal. Com o Astoria, criamos um serviço WCF e, por este serviço, utilizando os protocolos Representational State Transfer (REST) e Plain Old XML (POX), o banco de dados subjacente era lido e modificado. Para estabelecer a lógica do negócio necessária nos locais apropriados, consideramos que os serviços de sincronização, o Astoria e os acionadores de banco de dados trabalharam muito bem juntos.

### Aplicativo de expedição, armazenamento e faturamento

Para auxiliar o trabalho dos estoquistas, líderes de grupo e expedidores na matriz, era necessário um aplicativo fácil de usar e que disponibilizasse prontamente todas as informações necessárias. Este aplicativo de linha de negócios (LOB) integra-se com os sistemas de posicionamento (utilizando o armazenamento de dados espaciais), criando uma boa experiência para o usuário (com uma combinação do WPF e do Windows Forms) e exibindo os mapas (com os Virtual Earth Live Services).

### Site de web externo

Como essa era uma nova solução, tivemos alguma flexibilidade para buscar tecnologias sem correr um risco muito grande de ter de manter os aplicativos existentes. Implementamos uma etapa de workflow personalizado para podermos aprovar, recusar ou modificar os registros visíveis para o usuário final, principalmente para impedir a exibição de dados obsoletos. A solução para isso derivou do "Windows Workflow Foundation Web Workflow Approvals Starter Kit", que pudemos personalizar com um pequeno esforço. Outra oportunidade para ampliar o horizonte tecnológico foi a prototipagem rápida oferecida pela solução de estrutura "Jasper", fornecida com o ASP.NET Futures: começando com uma interface simples, achamos que ficou fácil passar para uma personalização profunda sem perder o investimento nos formulários originais. Os práticos recursos do botão "voltar" e do apoio a histórico do ASP.NET Ajax Futures fez com que as alterações de interface ocorressem de forma fluida.

### Aplicativo UMPC

Consideramos que o aplicativo criado para o dispositivo UMPC era a nossa oportunidade para acrescentar o raciocínio mais inovador. O dispositivo UMPC e o aplicativo de processamento de planilhas executam diversas tarefas:

- Registro do trabalho executado pelo técnico, com o preenchimento da planilha.
- Comunicação com o servidor via 3G, Edge ou WiFi, utilizando o ADO.NET Synchronization Services e uma instância local do SQL Server 2008 Compact Edition para o armazenamento de dados.
- Gravação de voz e câmera para documentação, com anotações adicionais com o reconhecimento de escrita.
- Exibição de manuais em formato XPS com o Microsoft News Reader SDK para os ATMs/POSS, incluindo recursos de anotação e colaboração, e navegação baseada em voz, soft-button ou caneta stylus.
- Exibição de mapa e informações de itinerário para o local da próxima visita.

Qual é a infra-estrutura que está por trás disso? Do lado do desenvolvedor, foi realmente uma questão de unir blocos de construção bem projetados, como Enterprise Library, WPF, ADO.NET Synchronization Services, SQL Server 2008 Compact Edition, WPF e o .NET Framework. Uma vez que o UMPC é um PC completo com processador compatível com x86, que executa sistemas operacionais familiares como o Windows XP Tablet PC Edition ou o Windows Vista, julgamos que o processo de desenvolvimento real é bastante parecido com o desenvolvimento de um aplicativo normal para desktop. No entanto, da perspectiva do projetista e de aspectos ergonômicos de software, as coisas são um tanto mais complexas. Na próxima seção discutiremos este ponto.

### Interface de usuário com WPF

Deixando a interface de usuário tradicional para trás

O Windows Presentation Foundation oferece aos projetistas de IU a chance de reconsiderar velhos hábitos de projeto de IU e a oportunidade para alguns pensamentos divergentes e multifacetados.

Se adotássemos a visão tradicional da IU, provavelmente teríamos enumerado as funções do software, criado grupos e exibido um menu

Figura 3: O usuário é levado diretamente à parte mais utilizada do aplicativo.



Figura 4: A abertura da pasta revela mais detalhes e acesso à edição de seu conteúdo.





Figura 5: O inventário fica sempre acessível no lado direito.



Figura 6: Menus em “pizza” são ideais para aproveitar a memória espacial. Áreas de alta sensibilidade ajudam na interação por toque com os dedos.



principal depois do início do programa, quase que certamente com as opções usadas com mais frequência. Em vez disso, decidimos eliminar completamente o menu principal: o software começa com a exibição mais utilizada, com a tela geral da planilha. A funcionalidade adicional de sincronização, atualizações de versão, repositório de referências e acesso ao inventário do usuário foram implementados em painéis complementares, de deslizamento lateral.

Na tela de visão geral de planilha, as planilhas são exibidas como pastas. As pastas já mostram os dados mais importantes no lado de fora (Figura 3), porém, mais detalhes são exibidos quando a pasta é aberta com um leve toque na tela sensível ao toque (Figura 4). Também colocamos o botão “Edit Worksheet” (Editar planilha) dentro da pasta, garantindo, dessa forma, que nenhuma planilha seja aberta sem antes o usuário ter tido a oportunidade de conhecer os seus principais detalhes.

#### Aproveitamento da memória espacial

De acordo com a Wikipedia, “a memória espacial é a parte da memória responsável pelo registro de informações sobre o próprio ambiente e suas orientações espaciais”. Por exemplo, a memória espacial ajuda a lembrar que a caneta que você está procurando foi vista pela última vez no canto direito da mesa.

Se você aproveitar as vantagens da memória espacial, e colocar sempre o mesmo item no mesmo lado da tela, os usuários sempre se lembrarão onde encontrá-lo. Para implementar isto no nosso caso, o painel inferior de deslizamento lateral contém materiais de referência, e o painel direito de deslizamento lateral, o inventário (todos os equipamentos existentes na caçamba dos seus caminhões, prontos para serem usados na instalação ou reparo de dispositivos (Figura 5).

O painel de inventário tem duas funções. Inicialmente, o técnico verifica se ele tem as peças de reposição de que precisa. Mais tarde, ao preencher a planilha, ele registra as peças que usou arrastando e soltando os itens do inventário para a planilha. Em ambos os casos, o inventário pode ser encontrado no lado direito, acessado, filtrado e classificado da mesma forma. Escolhemos o lado direito porque, durante os testes de usuário, descobrimos que a maioria dos usuários são destros, portanto, seguram o UMPC na mão esquerda e a caneta na mão direita. Sendo este o caso, a operação de arrastar e soltar para a adição de um item do inventário na planilha ficou mais prática.

Optamos por incluir um menu em forma de pizza na tela do editor de planilhas (Figura 6). Descobrimos que os usuários

iniciantes compreendiam rapidamente como o menu funcionava com a ajuda de animações sutis e ágeis durante a operação do menu, submenu ou seleção de um item do menu. As grandes áreas sensíveis do menu ajudam quando o menu é usado com os dedos — os dedos não são tão precisos quanto um mouse ou uma stylus, mas são sempre úteis. O menu em pizza também aproveita a memória espacial do usuário, uma vez que é muito mais fácil lembrar que o ícone “Close worksheet” (Fechar planilha) está à direita, do que lembrar que é o terceiro item de um menu suspenso.

Na vida real, o posicionamento dos itens é uma maneira de organizá-los e queríamos levar este conceito para toda a interface de usuário. Por exemplo, talvez você tenha o hábito de manter documentos importantes no lado esquerdo da sua mesa, e os menos importantes, à direita. Oferecemos a mesma possibilidades aos nossos técnicos na visão geral da planilha: as pastas podem ser movidas, giradas e até mesmo redimensionadas (Figura 7).

#### Interação natural

Embora eu e você estejamos acostumados com o mouse, não é uma maneira natural de interação. Temos que mover um objeto de forma estranha na mesa, enquanto olhamos para outra parte para ter o feedback e confirmar se o cursor atingiu o destino pretendido. Quando se trata de interações do tipo apontar e clicar,

Figura 7: O posicionamento personalizável das pastas permite que o agrupamento espacial ajuste-se aos hábitos individuais de cada usuário.



**Figura 8:** Na tela de desenho, o técnico pode registrar qualquer tipo de informação gráfica.



arrastar e soltar ou desenhar, usar uma tela sensível ao toque é muito mais intuitivo e produtivo, mesmo para usuários experientes de mouse. Isto posto, contudo, um dedo ou mesmo uma caneta ou stylus tende a ser muito menos preciso que um mouse. Para acomodar isto quando projetamos a interface do usuário, mantivemos os botões grandes e nos asseguramos de que não tocassem um nos outros. Pela mesma razão, quando utilizamos uma barra de rolagem, tivemos que garantir que fosse muito mais larga que o normal.

Durante os testes com usuário, confirmamos que a barra de rolagem é algo que não se deseja ter de forma alguma em um ambiente sensível ao toque. Se você pensar a respeito, usar a barra de rolagem não é natural: é necessário pressionar o botão com a seta para baixo ou arrastar uma mãozinha para baixo, quando se quer que o conteúdo se mova para cima. Quando você está rolando um documento, na verdade ele está se movendo na direção oposta à do mouse ou da esfera. Uma maneira muito mais natural de rolagem é arrastar o conteúdo em si na direção desejada. Deixar para trás esta maneira antinatural de rolagem é uma das razões que tornam a última geração de dispositivos móveis tão agradáveis de usar.

### **Uso do reconhecimento de escrita para inserir dados**

No que se refere a registrar informações, a versatilidade do papel e da caneta é imbatível do ponto de vista da usabilidade. A utilização do reconhecimento de escrita proporciona aos técnicos uma forma de desenhar diagramas, escrever textos ou fazer anotações livres em uma parte do documento. No mundo digital, Tablet PCs, UMPCs e Pocket PCs com telas sensíveis ao toque podem funcionar da mesma forma. Verificamos que o reconhecimento de escrita digital seria útil para fazer desenhos, anotações e registrar a assinatura do cliente como confirmação de que o trabalho foi executado.

Utilizar a API do reconhecimento de escrita no Windows Presentation Foundation (no namespace System.Windows.Ink) fez do uso da tinta digital uma brisa nos cenários acima. Em apenas duas horas, fomos capazes de adicionar uma tela de desenho com capacidade de armazenar, editar e redimensionar os traços para o reconhecimento de escrita — sendo que a maior parte do tempo foi gasta com a criação de ícones que alternam modos de desenho. (A Figura 8 mostra um exemplo de imagem anotada). O reconhecimento de texto (ou Ink Analysis) é também um processo muito direto e funciona para vários idiomas, como inglês (britânico ou americano), francês, espanhol, coreano, alemão, japonês ou chinês.

Sem o reconhecimento de texto, o método de preenchimento do formulário pelo técnico seria bastante complexo sem um mouse e um

teclado. Um bom exemplo é o campo em que registra o número de horas que demoraram para fazer o reparo. Verificamos que o usuário tinha de apontar (e clicar) no campo apropriado, selecionar o seu conteúdo (se houvesse algum), clicar para exibir um teclado na tela, inserir o número de horas e fechar o teclado. São cinco cliques em uma série de alterações de contexto na mente do usuário ("Eu quero inserir um número. Preciso selecionar o local aonde quero ir. Preciso exibir um teclado virtual. Preciso usar o teclado, fechar o teclado e voltar para a tarefa principal de preencher a planilha"). Compare essa sequência com o ato de escrever o número "3" no campo apropriado com a caneta (Figura 6), e você gostará da simplicidade natural deste novo-antigo modo de interação.

### **Conclusão**

A criação de aplicativos móveis de conexão ocasional implica vários desafios. Felizmente, tecnologias de hardware e software que viabilizam esses aplicativos estão começando a ficar mais comuns — UMPCs com recursos de hardware de PCs de alguns anos atrás, rastreadores GPS, conectividade móvel com Internet, bibliotecas sofisticadas de software, reconhecimento de escrita e voz incorporado ao sistema operacional e diversos serviços on-line. E a lista não pára de crescer. Com o sistema que colocamos em operação para a Monicomp, acreditamos que as peças estão se encaixando. E isto permitirá aos desenvolvedores criar soluções móveis que apenas os escritores de ficção científica ousavam imaginar há uma década.

### **Sobre os autores**

**András Velvart** tem o estilo de desenvolvimento de software de Lincoln: "Se eu tenho seis horas para cortar uma árvore, usarei quatro horas para afiar o machado". No passado, ele foi um dos poucos que ajudaram a disseminar a idéia da World Wide Web na Hungria. Ele sempre gostou de experimentar e implementar as mais novas tecnologias. Muitas das suas provas de conceito tornaram-se acréscimos úteis a projetos, desde as áreas de IU/usabilidade até o processo de desenvolvimento. András trabalha com tecnologias Microsoft há 12 anos. Ele foi desenvolvedor-chefe, arquiteto e gerente de projeto de muitos projetos Web, desktop e corporativos, até que fundou a própria empresa de consultoria e desenvolvimento de software, a Response. András pode ser contatado no endereço [andras.velvart@response.hu](mailto:andras.velvart@response.hu).

**Peter Smulovics** é um homem da Microsoft, que procura agulhas no palheiro há mais de dez anos. Ele trabalhou em projetos como SQL 2005 Analysis Services, Visual Studio.Net 2005, Microsoft Business Framework and Portal, ADO.Net Entity Framework, Dynamics Great Plains and Solomon, executando testes, desenvolvimento, como líder de equipes e arquiteto. Ao mesmo tempo, suas atividades diárias incluem participar de grupos de usuários, dar palestras em fóruns da Developer Days, TechEds e Microsoft, liderar a introdução do .NET no país, entre outras. Atualmente, está trabalhando no grupo de Development and Platform Adoption (desenvolvimento e adoção de plataforma) e prestando auxílio em arquitetura e workshops para clientes corporativos sobre muitos produtos diferentes. Peter pode ser contatado no endereço [smulovics.peter@microsoft.com](mailto:smulovics.peter@microsoft.com).

**Isenção de responsabilidade:** Embora a solução descrita neste artigo baseie-se no sistema de trabalho real da Monicomp, em razão da natureza do tópico, alguns detalhes foram omitidos para proteger os interesses da empresa, enquanto outros estão em fase de planejamento ou implementação.

ARC

**Microsoft®**

THE  
ARCHITECTURE  
JOURNAL



---

► Inscreva-se no endereço: [www.architecturejournal.net](http://www.architecturejournal.net)