UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANTONIO WELLIGTON DOS SANTOS ABREU

APLICAÇÃO DE UMA ARQUITETURA BASEADA EM PROXY PARA APOIO À COMUNICAÇÃO E PROCESSAMENTO DE INFORMAÇÕES EM SISTEMAS MÓVEIS

QUIXADÁ

2012

ANTONIO WELLIGTON DOS SANTOS ABREU

APLICAÇÃO DE UMA ARQUITETURA BASEADA EM PROXY PARA APOIO À COMUNICAÇÃO E PROCESSAMENTO DE INFORMAÇÕES EM SISTEMAS MÓVEIS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: Computação

Orientador: Prof**. Vitor Almeida dos Santos**

QUIXADÁ

2012

A162a Abreu, AntonioWelligton dos Santos

Aplicação de uma arquitetura baseada em proxy para apoio a comunicação e processamento de informações em sistemas móveis / Antonio Welligton dos Santos Abreu. 2012.
43f. ; il. color. enc.

Orientador: Prof. MSc. Vitor Almeida dos Santos
Área de concentração: Computação

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Ceará, Campus Quixadá, Quixadá, 2012.

1. Computação Móvel 2. Web Service 3. Sincronização de DadosI. Santos, Vitor Almeida (Orient.) II. Universidade Federal do Ceará – Curso de Bacharelado em Sistemas de Informação III.Título

CDD 004

ANTONIO WELLIGTON DOS SANTOS ABREU

APLICAÇÃO DE UMA ARQUITETURA BASEADA EM PROXY PARA APOIO À COMUNICAÇÃO E PROCESSAMENTO DE INFORMAÇÕES EM SISTEMAS MÓVEIS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: Computação

Aprovado em: _____ / _____ / 2012.

BANCA EXAMINADORA

_____
Prof. MSc. Vitor Almeida dos Santos (Orientador)
Universidade Federal do Ceará-UFC

_____
Prof. MSc. Lincoln Souza Rocha
Universidade Federal do Ceará-UFC

_____
Prof. MSc. Camilo Camilo Almendra
Universidade Federal do Ceará-UFC

Dedico esta monografia a Rita e Aurea, que são as pessoas que mais me apoiaram durante esse período da minha vida. Eu sempre vou agradecer a Deus por ter me dado a oportunidade de ter duas mães, a de coração e a de sangue, pois elas duas me ensinaram a lutar sempre por meus objetivos de forma honesta e com respeito ao próximo.

AGRADECIMENTOS

Primeiramente a Deus, pois ele sempre me guiou pelos melhores caminhos da vida acadêmica e sem ele esse trabalho não poderia ser realizado.

Ao meu orientador, Vitor Almeida, por me apoiar durante todas as etapas desse trabalho de forma que resultou em um bom acompanhamento.

A todos os professores que sempre estavam dispostos para ajudar no que fosse possível.

Aos meus colegas Ismaily e Egberto que são meus eternos amigos e companheiros, que me ajudaram a enfrentar e superar todas as dificuldades que surgiram durante o curso.

A minha família que sempre estava disposta a fazer o impossível para eu concluir esse curso, ajudando em todos os sentidos para que fosse possível permanecer nessa cidade.

A Universidade Federal do Ceará, por me conceder a bolsa do Programa de Educação Tutorial.

Enfim a todos que contribuíram direta ou indiretamente para essa fase da minha vida.

“A grandeza não consiste em receber honras, mas em merecê-las.” (Aristóteles)

RESUMO

Aplicações para dispositivos móveis são cada vez mais utilizadas no mundo atual como meios para lidar com informações pessoais e organizacionais. Frequentemente, o uso de tais aplicações ocorre em ambientes que dispõem de uma rede local com computadores com maiores capacidades computacional e de comunicação se comparados com os dispositivos móveis. Este trabalho defende a hipótese de que uma eventual capacidade computacional ociosa destes computadores pode ser utilizada em benefício dos dispositivos móveis. Para tal, é apresentado um modelo de sistema distribuído onde computadores de uma rede local desempenham o papel de intermediários (ou *proxies*) entre aplicações clientes móveis e servidores, realizando algum processamento atribuído à aplicação de forma a contribuir com seu desempenho geral. Para avaliar a proposta, foi desenvolvido e utilizado um protótipo de uma aplicação móvel que auxilia o serviço de atendimento domiciliar. Os resultados alcançados foram satisfatórios e demonstraram a viabilidade da proposta.

Palavras-chave: Computação Móvel, Arquitetura Proxy, Sincronização de Dados.

ABSTRACT

Applications for mobile devices are increasingly used in today's world as a means to deal with personal and organizational information. Often, the use of such applications occurs in environments such as a local area network with greater computational capabilities and communication compared to mobile devices. This work supports the hypothesis that a computational resources of idle computers can be used for the benefit of mobile devices. To this end, we present a model of distributed system in which computers on a local network play the role of proxies between mobile clients and server applications, performing some processing assigned to the application in order to contribute to their overall performance. To evaluate the proposal it was developed and used a prototype of a mobile application that supports home care medical services. The obtained results were satisfactory and demonstrated the feasibility of the proposal.

Keywords: Mobile Computing, Proxy Architecture, Data Synchronization.

LISTA DE ILUSTRAÇÕES

Sumário

1 INTRODUÇÃO

Dispositivos móveis tais como telefones inteligentes e *tablets* têm se tornado cada vez mais diversos e acessíveis, o que torna o seu uso para atividades pessoais e profissionais cada vez mais frequente. Atividades profissionais que requerem mobilidade são as que mais podem se beneficiar destes dispositivos. Vendedores em visitas a possíveis compradores, médicos que realizam acompanhamento doméstico de pacientes e executivos em reuniões de negócios são exemplos potenciais de profissionais beneficiados com o uso das tecnologias móveis.

Muitas das aplicações móveis utilizadas requerem a interação com algum tipo de serviço remoto – tal como um serviço de banco de dados – e usualmente são implementadas seguindo uma arquitetura cliente-servidor. Esses sistemas móveis têm como requisito a capacidade de acesso a uma determinada rede para comunicação com o servidor. A interação entre o cliente móvel e o servidor acontece através da troca de informações. Essa troca deve ser gerenciada para lidar com possíveis gargalos na rede ou indisponibilidade de recursos quando, por exemplo, diversos nós (clientes móveis) acessam um recurso limitado (um único servidor).

Frequentemente, o uso destes dispositivos ocorre em ambientes domésticos e corporativos os quais contam com redes locais e computadores *desktop*. Ainda que os recursos computacionais e de comunicação dos dispositivos móveis estejam se tornando cada vez mais robustos – até comparáveis com os recursos de computadores pessoais – a capacidade ociosa de computadores *desktop* pertencentes a redes locais de ambientes domésticos e corporativos podem servir como um mecanismo para sincronizar o acesso a determinado recurso e proporcionar uma certa comodidade como, por exemplo, no momento de enviar dados, para às aplicações móveis. Desta forma, este trabalho sugere que é possível utilizar a capacidade de comunicação e computação ociosa de computadores da rede local do ambiente para auxiliar a troca de dados entre os dispositivos móveis e um servidor remoto, melhorando o desempenho geral da aplicação. Para tal, estes computadores devem desempenhar o papel de um *servidor proxy*, ficando responsáveis por tornar transparente a forma como os clientes móveis acessam e enviam dados para o servidor remoto.

O servidor *proxy* é uma das variações da arquitetura cliente-servidor e que representa um tipo de serviço que tem como objetivo receber requisições dos clientes (eles podem ser computadores fixos ou móveis) e, logo após, executar esses pedidos através de um processo de comunicação com outros servidores existentes na rede (COULOURIS; DOLLIMORE; KINDBERG, 2007). O *proxy* tem a característica de assumir o papel de cliente de outro

servidor, de forma que ele é um processo compartilhado por vários clientes que requisitam algum tipo de serviço. Ele também serve como *cache* para os recursos disponibilizados por outros servidores remotos (COULOURIS; DOLLIMORE; KINDBERG, 2007) e tem como principais funções reduzir o tempo de acesso e aumentar a disponibilidade. Também é utilizado para proteção, filtragem e adaptação.

Este trabalho propõe um modelo de sistema distribuído para clientes móveis em que computadores de uma rede local desempenham o papel de *proxies* das aplicações. Vários clientes móveis podem acessar de maneira periódica um determinado serviço. Em particular, serão enfatizados os serviços *web*, devido à sua ampla utilização e padronização. A aplicação *proxy* é empregada como mecanismo intermediário para a realização de computações e comunicações.

No intuito de avaliar a proposta, foi desenvolvido e utilizado um protótipo de uma aplicação móvel que auxilia o serviço de atendimento domiciliar (*home care*) realizado por profissionais de saúde.

No capítulo 2 estão detalhados os principais conceitos utilizados nesse trabalho. O capítulo 3 discute a arquitetura proposta, descrevendo os aspectos de cada camada que a formam. O capítulo 4 descreve os detalhes de implementação de cada componente da arquitetura. Por fim, o capítulo 5 contém as conclusões e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo trata da fundamentação teórica utilizada neste trabalho. São abordados os conceitos das áreas de sistemas distribuídos, *Web Services*, computação móvel e atendimento domiciliar, que serão utilizados para alcançar os objetivos deste trabalho. Para finalizar este capítulo contém uma descrição dos trabalhos relacionados.

2.1 Sistemas Distribuídos

Existem várias definições difundidas nos últimos anos sobre o conceito de sistemas distribuídos (SD), mas é possível dizer de certa maneira que o mesmo consiste em um conjunto de computadores independentes que cooperam entre si através de troca de informações e se apresentam a seus usuários como um único e coerente sistema, ou seja, "a principal meta de um sistema distribuído é facilitar aos usuários, e às aplicações, o acesso a recursos remotos e seu compartilhamento de maneira controlada e eficiente" (TANENBAUM; STEEN, 2007, p.2). Diante dessa definição são destacados alguns aspectos importantes para considerar um sistema distribuído: os computadores do sistema são autônomos; usuários acham que estão interagindo com um único sistema; os componentes (computadores) autônomos precisam cooperar uns com os outros; os computadores podem variar desde mainframes até pequenos nós em redes sensores; não tem nenhuma restrição de como os componentes são interconectados.

Segundo Tanenbaum e Steen (2007) podem-se destacar as seguintes características importantes de sistemas distribuídos:

- A diferença entre vários computadores e o modo como eles se comunicam estão, em grande parte, ocultas aos usuários;
- Usuários (podem ser pessoas ou programas) podem interagir com o sistema distribuído de maneira consistente e uniforme, independente de onde a interação ocorra;
- O sistema distribuído deve possuir escalabilidade;
- Ele deve estar disponível, mesmo que algumas partes possam estar temporariamente avariadas;
- Os usuários e aplicações não devem perceber quais são as partes que estão sendo substituídas ou consertadas, ou quais são as novas partes adicionadas para atender a mais usuários ou aplicações.

Um dos modelos mais usuais de sistemas distribuídos trata-se do cliente-servidor. Esse modelo tem dois papéis bem definidos e com uma divisão de responsabilidade. Abaixo segue a descrição de cada papel:

- Servidor: é um processo que implementa um serviço específico, onde o mesmo pode ser um fornecedor de um ou mais serviços;
- Cliente: é um processo que requisita um serviço de um servidor através de uma requisição, agindo como um consumidor de serviços disponibilizados pelos servidores.

Um sistema baseado no modelo cliente-servidor deve conter as seguintes características específicas segundo Bragança (1999):

- Recursos partilhados: um servidor pode servir vários clientes ao mesmo tempo e gerir os acessos a recursos partilhados;
- Protocolos assimétricos: existe uma relação de muitos-para-um entre clientes e servidor e os servidores esperam passivamente os pedidos dos clientes, pois os mesmos sempre iniciam o diálogo através da requisição de um serviço;
- Localização transparente: o servidor é um processo que pode residir na mesma máquina que o cliente ou numa máquina diferente que esteja ligada através de uma rede, ou seja, não é possível para o usuário saber qual máquina é o servidor;
- Independência: esse ponto é um conceito inerente às arquiteturas clientes-servidores, pois o software deve ser independente de hardware ou sistemas operativos;
- Baseado na transmissão de mensagens: sistemas baseados neste modelo são normalmente baseados em mensagens, ou seja, a mesma serve de mecanismo de transporte para os pedidos e respostas dos serviços;
- Encapsulamento de serviços: a forma de implementar os serviços é feita de maneira transparente em relação ao usuário;

- Integridade: o código e os dados do servidor devem ser mantidos centralmente, pois desta forma reduzem-se os custos de manutenção e aumenta-se a integridade dos dados.

Alguns servidores também podem assumir papel de clientes de outros servidores e essa é uma variação do estilo cliente-servidor conhecida como servidor *proxy* (Figura 1). O servido *proxy* atua como um processo compartilhado por vários clientes que requisitam algum tipo de serviço para outros servidores.

O servidor *proxy* é considerado um modelo arquitetônico (descreve os componentes do sistema do ponto de vista organizacional, tendo como foco a arquitetura da rede que é a representação da estrutura de alto nível do sistema, se preocupando em descrever os seus componentes e relacionamentos entre eles) de um SD. Esse modelo se preocupa com a divisão de responsabilidades entre os componentes do sistema e na alocação física desses componentes à infraestrutura de rede (COULOURIS; DOLLIMORE; KINDBERG, 2007), pois esse modelo tenta garantir que atributos de qualidade como desempenho, gerência, confiabilidade e adaptabilidade atenderão de maneira satisfatória a atual e futura demanda de pedidos de serviços.

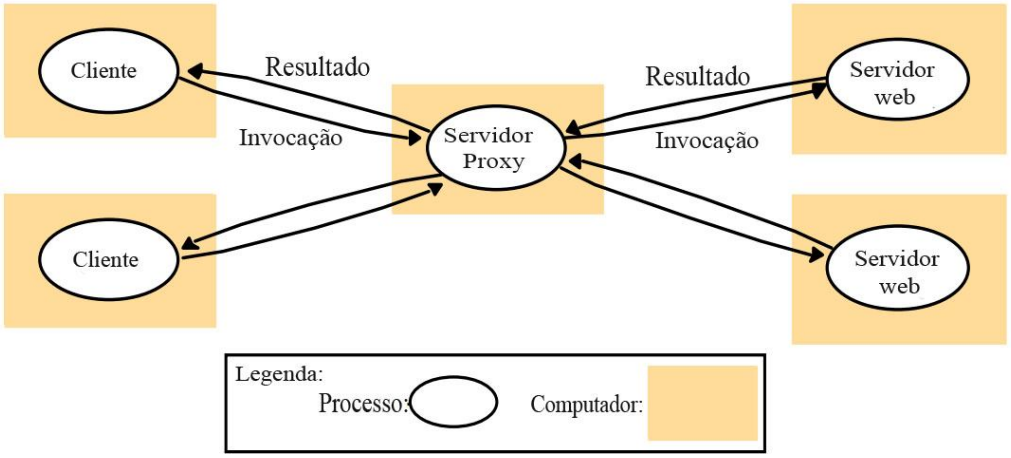

FIGURA 1 – Servidor *proxy* realizando pedidos a outros servidores. Fonte: adaptado de Coulouris (2007).

2.2 Web Services

Um serviço *web* (*web service*) é utilizado na integração de diferentes aplicações, portanto, ele é uma solução para comunicação entre aplicações desenvolvidas em plataformas diferentes, possibilitando a compatibilidade entre elas. Esse tipo de serviço fornece uma

interface que permite aos clientes interagirem com servidores *web* permitindo combinar as operações da interface com as de outros serviços para produzir nova funcionalidade (COULOURIS; DOLLIMORE; KINDBERG, 2007). Essa situação é ilustrada na Figura 2, onde um serviço de agente de viagens poderia usar suas operações para fornecer ao viajante uma combinação de serviços.

> Um *Web Service* é acessado através de protocolos e formatos de dados independentes de plataforma como o HTTP, XML e SOAP. A interface de um Web Service é acessível através de mensagens XML padronizadas e, portanto, em formato texto. São descritos utilizando um padrão formal chamado descrição de serviço que envolve os detalhes necessários para a interação com o serviço, incluindo o formato das mensagens, tipos de dados e localização. Um solicitante de um serviço descreve as características do serviço procurado e utiliza o provedor de registro para localizar um serviço apropriado. Uma vez localizado, a informação na descrição do serviço é utilizada para interação entre cliente e servidor. A descoberta, a invocação dinâmica de serviços e uma colaboração baseada em mensagens permitem o desenvolvimento de aplicações distribuídas com enorme grau de interoperabilidade. (TAMAE; LIMA, 2005, p.2).

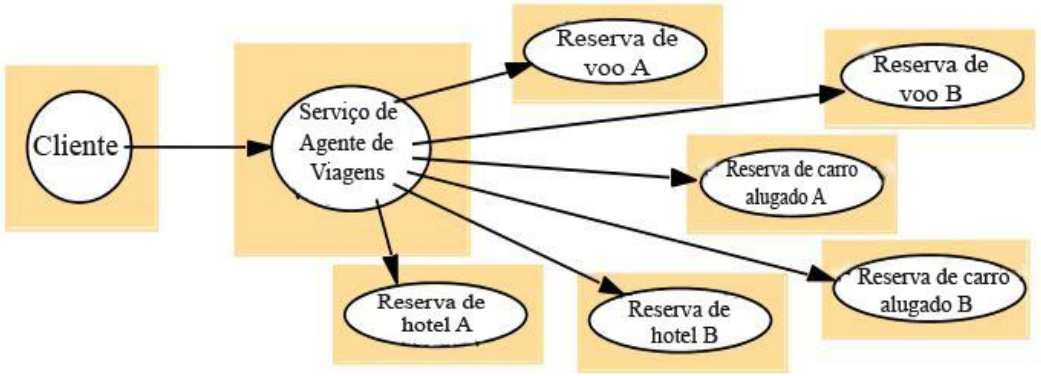

FIGURA 2 – O serviço de agente de viagens combina vários serviços *web*. Fonte: adaptado de Coulouris (2007).

Diversas plataformas de desenvolvimento fornecem facilidades para a construção de serviços *web*. A plataforma .NET disponibiliza a tecnologia *Windows Communication Foundation* (WCF) como solução para construção de serviços *web*. Ela é um modelo de programação criado para construir aplicações orientadas a serviços, onde o principal objetivo é permitir que desenvolvedores criem aplicações voltadas para área de sistemas distribuídos, através de um conjunto de bibliotecas disponibilizado pelo o framework (MCMURTRY et al., 2007).

Para desenvolver serviços *web* é necessária a criação e manipulação de três elementos fundamentais: especificação WSDL (Web Service Description Language) para a definição de serviços a serem utilizados por clientes, a especificação de *schemas* XSD (XML *schemas*)

para definição de dados a serem transportados e a definição de mensagens SOAP (Simple Object Access Protocol), que estabelecem as mensagens trocadas para a comunicação.

Uma especificação WSDL é um documento escrito em XML para definir a forma como se deve acessar um serviço *web*, que no WCF esse documento está fortemente vinculado a um contrato de serviço. O XSD é um documento que tem como objetivo definir uma estrutura de documento XML, em que no contexto de serviços *web*, esse documento XSD é comumente empregado para definir os tipos de dados de aplicação a serem transportados, já que os dados são escritos em arquivos XML, onde esses dados podem ser controlados por contratos de dados no WCF. Por fim, o SOAP é um protocolo de comunicação baseado em XML para a troca de informações utilizando HTTP, onde o mesmo é comumente utilizado para troca de mensagens por serviços *web*, pois se trata de um protocolo independente de plataforma, em que as mensagens podem se controladas por contratos de mensagens no WCF.

"Uma das grandiosidades do WCF é a possibilidade de utilizar qualquer tipo de aplicação como *host*, ou seja, ele não tem uma dependência de software" (ISRAEL, 2009, p.3) e ele pode expor serviços para serem acessados através dos mais diversos tipos de protocolos, como HTTP, TCP, IPC e MSMQ.

2.3 Computação Móvel e Ubíqua

A computação móvel e ubíqua surge devido aos avanços tecnológicos na miniaturização dos dispositivos e da conectividade sem fio. Do ponto de vista dos sistemas distribuídos, não há diferença básica entre computação móvel e ubíqua: o termo computação móvel refere-se à conexão de dispositivos que se movimentam no mundo físico cotidiano e a computação ubíqua diz respeito à exploração da integração cada vez maior dos dispositivos de computação com o nosso mundo físico. Os sistemas móveis e ubíquos são sistemas voláteis devido ao fato que eles são integrados com o nosso mundo físico cotidiano. (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Segundo Coulouris et al. (2007), essa classe de dispositivos tem alguns aspectos peculiares:

- Energia limitada: um dispositivo portátil ou incorporado no mundo físico normalmente precisa funcionar com baterias, e quanto menor e mais leve o dispositivo precisa ser, menor será a capacidade de sua bateria. Substituir ou recarregar essas baterias provavelmente será inconveniente, em termos de

tempo e acesso físico. Computação, acesso à memória e outras formas de armazenamento, tudo isso consome energia preciosa. A comunicação sem fio é particularmente dispendiosa em termos de energia. Assim, se um equipamento precisa permanecer, enquanto for possível, com um determinado nível mínimo de carga de bateria, os algoritmos precisam ser sensíveis à energia que eles consomem, especialmente em termos de complexidade de mensagem.

- Restrições de recurso: os dispositivos móveis e ubíquos têm recursos computacionais limitados, em termos de velocidade do processador, capacidade de armazenamento e largura de banda. Isso se dá, em parte, porque o consumo de energia aumenta quando essas características são melhoradas. Mas também acontece porque tornar os dispositivos portáteis, ou incorporá-los nos objetos físicos cotidianos, significa torná-los fisicamente pequenos, o que, dadas às limitações impostas pelos processos de fabricação, restringe o número de transistores nos microprocessadores e controladores.

De acordo com os pontos descritos em relação a essa classe de dispositivos é possível identificar dois problemas:

- Como projetar algoritmos que possam ser executados em tempo razoável.
- Como poupar os recursos dos dispositivos móveis usando os recursos do próprio ambiente.

Assim como ocorre com os *Web Services*, diversas plataformas de desenvolvimento estão disponíveis para o desenvolvimento de aplicações móveis. Em particular, será empregada a plataforma .NET para o desenvolvimento de aplicações para *Windows Phone* 7 *(*WP7*)*, que é um sistema operacional móvel desenvolvido pela *Microsoft.* A plataforma .NET disponibiliza o pacote de desenvolvimento de software (SDK – *Software Development Kit*), que permite os programadores elaborarem aplicativos para essa tecnologia, possibilitando o desenvolvimento de várias classe de aplicações, desde de jogos até projetos desenvolvidos para atender a necessidades de uma área específica. A programação para WP7 com o SDK da plataforma .NET é feita com alta produtividade, pois oferece soluções prontas para as possíveis necessidades que podem surgir para o desenvolvedor (MSDN, 2011).

2.4 Atendimento Domiciliar

O termo atendimento domiciliar é empregado em um sentido amplo, “compreendendo uma gama de serviços realizados no domicílio e destinados ao suporte terapêutico do paciente” (FLORIANI; SCHRAMM, 2003, p.2). Neste cenário, o paciente é normalmente acompanhado por um profissional, tal como um auxiliar de enfermagem ou um cuidador, que tem como responsabilidade o registro das informações no dia-a-dia do paciente, podendo utilizar tais informações como um importante mecanismo para auxílio médico, ou seja, tal registro pode auxiliar em uma decisão médica à distância.

O *home care* tem como foco atender os pacientes que estão com o estado de saúde fragilizado por causa de alguma doença, ou seja, pessoas com “doenças incapacitantes como o Acidente Vascular Cerebral (AVC), infartos severos, demência, Parkinson, doenças pulmonares crônicas ou osteoarticulares são apenas alguns exemplos de enfermidades que implicam numa drástica limitação do indivíduo e acarretam a necessidade de um acompanhamento constante” (BRANDÃO, 2010, p.1).

Em alguns casos, após uma fase crítica que é geralmente tratada nos hospitais, os pacientes que começaram a apresentar um quadro de estabilização não necessitam mais da internação hospitalar, todavia apresentam um alto grau de dependência para as funções mais básicas. Em poucas palavras, o *home care* tem como proposta prestar vários tipos de serviços (assistência domiciliar de saúde, internamento domiciliar de saúde, atendimento domiciliar de saúde) de saúde a pacientes em sua residência de modo que melhore seu quadro clínico.

De acordo com Brandão (2010, p.2), as “despesas chegam a ser 60% menores do que as despesas com uma internação hospitalar, pois não envolve custos como lavanderia, alimentação e alguns tipos de medicamentos, por exemplo. Além da questão de economia de recursos, o tratamento domiciliar permite a recuperação do paciente num ambiente familiar, o que evidentemente traz benefícios à sua recuperação”.

Este ambiente foi utilizado como forma de validação deste trabalho por causa da sua característica de dinamismo, isto é, nesse cenário existe uma movimentação dos profissionais da área de saúde que, ou estão visitando os pacientes em suas casas ou em seus próprios consultórios. Isto torna esse cenário bastante propício para esse trabalho, pois o consultório é um candidato a possuir o *proxy*, porque esse é o local onde será enviado os dados de acompanhamento dos pacientes. Além dessas características desse ambiente, há poucos sistemas no mercado com a tecnologia WP7 direcionado para esse cenário.

2.5 Trabalhos Relacionados

Existem muitas aplicações que utilizam uma arquitetura baseada em *proxy*, mas poucas são aplicadas em ambientes móveis. Neste trabalho foram consideradas apenas as aplicações que estejam diretamente relacionadas a este trabalho. (BIEGEL; CAHILL; HAAHR, 2002) é um tipo de arquitetura que oferece suporte para aplicações cliente e servidor implementadas com RMI (*Remote Method Invocation*), e que atuam no ambiente móvel. Ela permite gerenciar as conectividades das aplicações no meio sem fio local, permitindo localizar e invocar corretamente objetos de servidores RMI hospedados em dispositivos móveis. Isso acontece através de *proxies* dinâmicos que foram desenvolvidos para fornecer redirecionamento de chamadas por meio de um *gateway* entre a *Wireless* e a *Ethernet* do ambiente, permitindo a exploração de servidores RMI móveis. Porém essa arquitetura se tornou obsoleta, pois as API das tecnologias utilizadas na mesma foram modificadas atualmente, com isso, essa solução não atende mais os objetivos esperados.

Ao contrário do trabalho anterior, existe o (ARTAIL; FAWAZ; GHANDOUR, 2012) que é bem mais atual e descreve uma nova abordagem para invocação de serviços *web* em dispositivos móveis. A proposta de arquitetura desse trabalho citado tem como objetivo minimizar o número de interações dos dispositivos com a rede e também reduzir o consumo de recursos de processamento, onde para isso acontecer, é definido que a descoberta dinâmica de serviços *web* deve ser passada para o servidor, aliviando assim o dispositivo móvel das tarefas que consomem energia, principalmente relacionados com a comunicação de servidores da *Internet* e análise de arquivos WSDL. Para acessar um serviço *web* de forma dinâmica foi definido que as aplicações utilizassem um *assembly* compilado a partir do arquivo WSDL localizado no servidor, isso em tempo de execução, onde o mesmo seria usado como um *proxy* pela aplicação móvel para interagir com um determinado serviço, assim se comportando como um processo local.

3 ARQUITETURA PROPOSTA

A arquitetura de sistema distribuído proposta neste trabalho tem o objetivo de auxiliar a troca de dados entre os dispositivos móveis e servidores remotos através dos computadores de redes locais, os quais desempenharão o papel de intermediadores (*proxy*) entre as aplicações móveis e tais servidores.

A arquitetura define três camadas. Cada camada representa um conjunto de componentes físicos vinculados a componentes de software executados neles. A Figura 3 apresenta a interação entre as três camadas da arquitetura proposta.

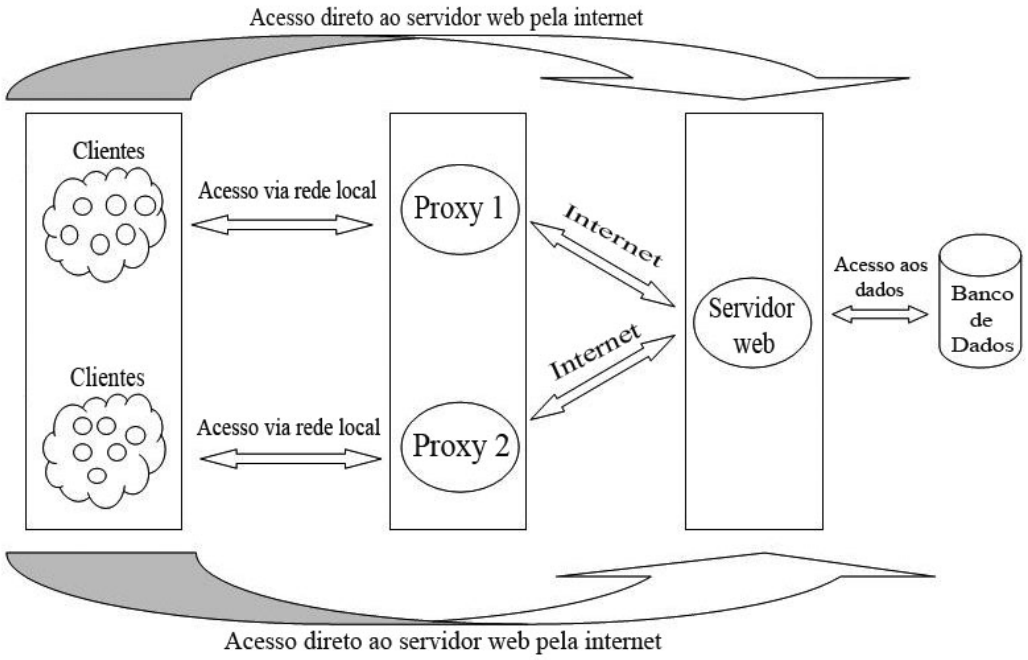

FIGURA 3 - Proposta de atuação de serviço *proxy* em redes locais. Fonte: o autor.

A seção 3.1 deste capítulo detalha a interação entre as camadas da arquitetura. As camadas de servidor *web* (serviços), de *proxy* e de cliente são descritas, respectivamente, nas seções 3.2, 3.3 e 3.4.

3.1 Interações Entre as Camadas da Arquitetura

Para ilustrar o funcionamento da arquitetura, foi definido um cenário de utilização, onde o mesmo se enquadra em qualquer ambiente que tenha as situações mencionadas no capítulo introdutório deste trabalho. O tipo de cliente utilizado neste cenário é móvel e tem acesso a um conjunto de funcionalidades fornecidas pela camada de serviços da arquitetura.

Cada dispositivo móvel quando entra na área de cobertura de uma determinada rede local tem a funcionalidade de reconhecer automaticamente o *proxy*, se este estiver disponível. Desta forma, ele poderia submeter um conjunto de dados para o servidor via *proxy*. No caso de o *proxy* não estar disponível, ele poderia também submeter diretamente ao servidor. A Figura 4 ilustra essas situações.

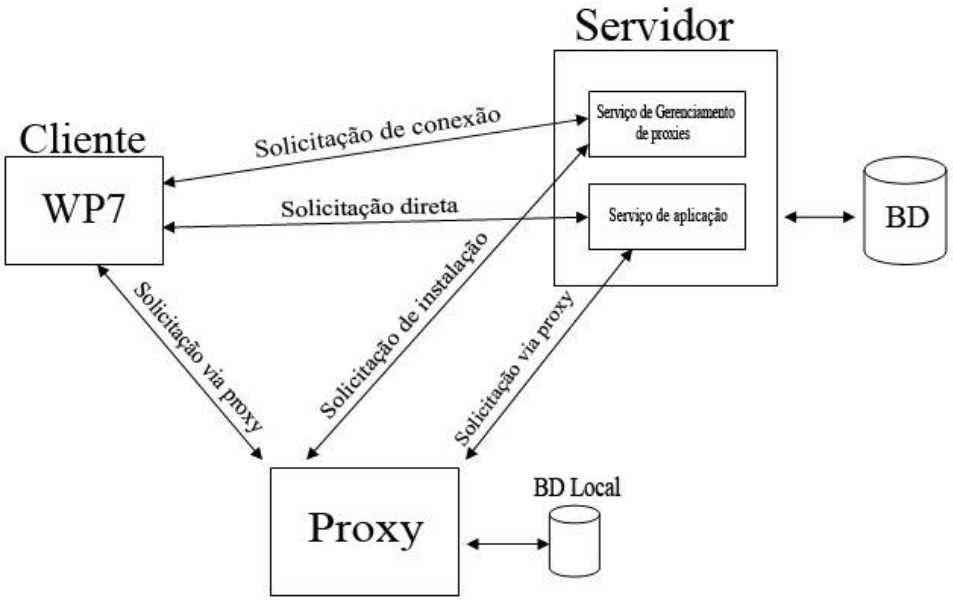

FIGURA 4 – Utilizando arquitetura proposta. Fonte: o autor.

A solicitação de conexão descrita na Figura 4 representa o momento em que o usuário entra na rede local, realizando a ação de conectar ao serviço *proxy* disponível naquele ambiente, operando em um computador *desktop*. Essa conexão é solicitada para que o usuário possa fazer uma solicitação via *proxy* no momento de realizar troca de dados com serviço de aplicação. Assim, o usuário poderia utilizar esse serviço para enviar determinados dados, considerando que esses dados podem ser enviados em outro momento se o servidor estiver indisponível.

Como se pode observar na Figura 4, o *proxy* tem uma base de dados local que tem como finalidade armazenar os resultados das submissões, possibilitando a ação de verificar posteriormente o resultado das mesmas. A solicitação de instalação descrito na figura representa o momento em que um novo *proxy* é inserido em uma rede. Ele solicita para o serviço de gerenciamento de *proxy* a sua instalação para poder estar disponível para os clientes móveis. Por fim, existe também a possibilidade do usuário estar em uma rede local que não tem nenhum serviço *proxy* operando no momento. Neste caso, ele poderia realizar uma solicitação direta ao serviço de aplicação.

3.2 Camada de Serviços

Essa camada representa o servidor remoto que é composto por dois serviços como mostra a Figura 5: *serviço de gerenciamento de proxies* e o *serviço de aplicação*. O *serviço de gerenciamento de proxies* é responsável por gerenciar os *proxies* existentes nas redes locais e o *serviço de aplicação* tem como finalidade prover suporte para as solicitações das aplicações móveis.

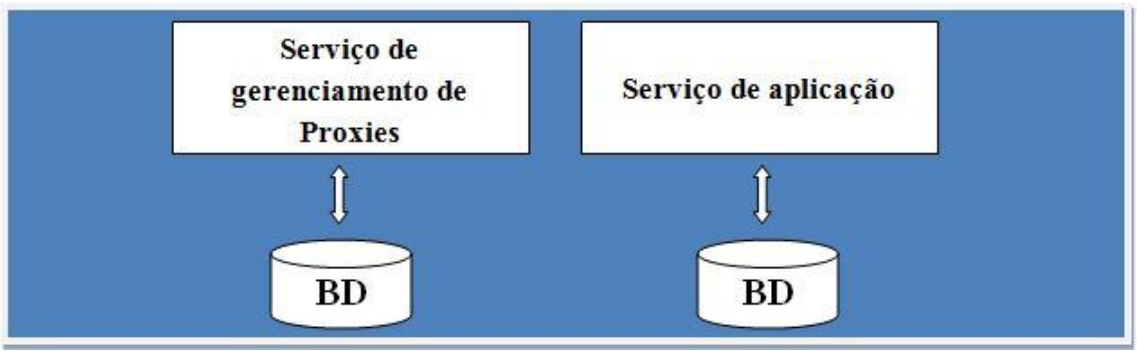

FIGURA 5 – Serviços que compõe a camada do servidor web. Fonte: o autor.

De acordo com a figura, pode-se perceber que cada serviço tem sua própria base de dados, pois cada um tem finalidades distintas, apesar de estarem executando no mesmo servidor. O *serviço de gerenciamento de proxies* tem como principal funcionalidade o registro de novos *proxies* em redes locais, possibilitar que um dispositivo móvel realize a conexão com um determinado *proxy* de maneira transparente assim que o mesmo entrar em uma rede local. Quando um novo *proxy* é implantado em uma determinada rede local, ele se identifica para o *serviço de gerenciamento de proxies*, que se encarregará de deixá-lo disponível para as aplicações móveis.

Em resumo, as principais funcionalidades desse serviço são:

- Adicionar um novo *proxy* na lista de *proxies* disponíveis para os clientes móveis.
- Retornar a lista de IP disponíveis para conexão em uma determinada rede local.

Por fim, o serviço de *aplicação* é responsável por disponibilizar as funcionalidades da aplicação que está utilizando a arquitetura proposta. Portanto, deve ser implementado pelo desenvolvedor usuário da arquitetura proposta.

Os dois serviços que compõe o servidor se comunicam e coordenam suas ações em relação aos outros componentes da arquitetura através de *Web Services*.

3.3 Camada de Proxy

Essa é a camada intermediária entre as camadas do servidor e cliente, formada pelo serviço *proxy* e um gerenciador de submissões para organizar o envio das requisições dos clientes para o servidor. A Figura 3 mostra com mais detalhes a interação entre essa camada e as outras. Nesta arquitetura, o *proxy* age em nome do cliente móvel intermediando a comunicação entre o mesmo e o servidor em determinados momentos como, por exemplo, na hora de enviar um conjunto dados de formulários para o servidor remoto. Esse serviço intermediário tem como objetivo melhorar o desempenho geral da aplicação e proporcionar certa latência no momento de enviar dados, para as aplicações móveis.

O *proxy* tem como funcionalidade sincronizar o acesso a determinado recurso e caso o mesmo não esteja disponível, o *proxy* ficaria responsável por enviar os dados quando fosse possível. O serviço *proxy* desenvolvido neste trabalho tem a capacidade de guardar os dados localmente em uma base dados e verifica periodicamente quando o servidor estará disponível para enviar os dados. Caso ocorra erro ou não na submissão dos dados, o *proxy* guarda o resultado da submissão e retorna o resultado para o dispositivo que solicitou a submissão, de maneira transparente para o cliente. De acordo com o que foi descrito, o usuário passaria a responsabilidade de enviar os dados para *proxy* no caso de o servidor estar indisponível, assim o usuário poderia sair da área de cobertura da rede local normalmente e, depois, verificar qual foi o resultado da submissão.

As principais funcionalidades do serviço *proxy* são:

- Recebe uma nova submissão a ser solicitada ao servidor quando possível.
- Retorna o resultado de uma ou mais submissões realizadas por um cliente móvel.
- Adiciona um novo *proxy* caso ele esteja executando pela primeira vez.

Assim como os demais serviços desse trabalho, o *proxy* também utiliza *Web Services* para se comunicar com outros componentes da arquitetura.

3.4 Camada de Cliente

Essa camada representa a interface de interação com o usuário, onde ocorre também uma interação direta ou indireta com um servidor remoto. Esta camada é a porta de entrada para utilizar e executar as funcionalidades do serviço de aplicação.

Além das funcionalidades do serviço de aplicação, a camada cliente utiliza também as funcionalidades do serviço *proxy* quando o usuário estiver em uma rede local.

4 IMPLEMENTAÇÃO DA ARQUITETURA PROPOSTA

Neste capítulo, a implementação da arquitetura proposta é detalhada. Uma vez que a arquitetura ainda está vinculada fortemente a um domínio, esta descrição também inclui tal domínio, que se trata do sistema para atendimento domiciliar.

O processo de desenvolvimento dos componentes da arquitetura adotado foi o iterativo e incremental, que "divide o desenvolvimento de um produto de software em ciclos, onde cada ciclo de desenvolvimento podem ser identificadas as fases de análise, projeto, implementação e testes do produto" (BEZERRA, 2007, p.33). Também foi utilizado o *Project Management Body of Knowledge* (PMBoK), que "define o gerenciamento e os conceitos relacionados, assim como também descreve o ciclo de vida do gerenciamento de projetos e os processos relacionados" (INSTITUTE, 2008, p.10). O gerenciamento de escopo (destacando os processos coletar requisitos, definir escopo e controlar o escopo) foi o conjunto principal de processos do PMBok utilizados neste trabalho.

A plataforma utilizada para desenvolver a aplicação móvel foi a *Windows Phone*7, juntamente com a *Integrated Development Environment* (IDE) Microsoft Visual Studio 2010 Express for Windows Phone e a IDE Microsoft Visual Web Developer 2010 Express para desenvolver a aplicação *web*, o serviço *proxy* e os demais serviços WCF auxiliares. O SGBD (sistema gerenciador de banco de dados) utilizado foi o Microsoft SQL Server 2008.

Como foi mostrada no capítulo 3, a arquitetura do sistema proposto divide-se em três camadas: camada de cliente, camada de Proxy e camada de serviços. A Figura 6 ilustra a interação entre os componentes dessa arquitetura.

FIGURA 6 – Relacionamento entre o serviço proxy e os demais componentes da arquitetura. Fonte: o autor.

A camada de cliente é uma aplicação móvel. A camada de *proxy* possui o *Web Service* que representa o *proxy* e um "Gerenciador de submissões" que trata-se de uma *thread* implementada no serviço *proxy* para realizar as submissões que foram delegadas pelos

clientes. A camada de serviços inclui o serviço da aplicação e o serviço gerenciador dos *proxies*. As camadas são descritas neste capítulo.

4.1 Camada de Serviços

Os componentes do servidor, ou seja, os serviços operando no mesmo, foram implementados utilizando a tecnologia WCF, que permite o desenvolvimento de serviços *web* com segurança e interoperabilidade bem como a facilidade de desenvolvimento inerente a plataforma utilizada na arquitetura proposta.

4.1.1 Serviço de Aplicação

Esse serviço tem a responsabilidade de prover as funcionalidades do ambiente de atendimento domiciliar para os dispositivos da camada cliente. A Figura 7 representa o diagrama de classes com as principais entidades do serviço e o relacionamento entre elas.

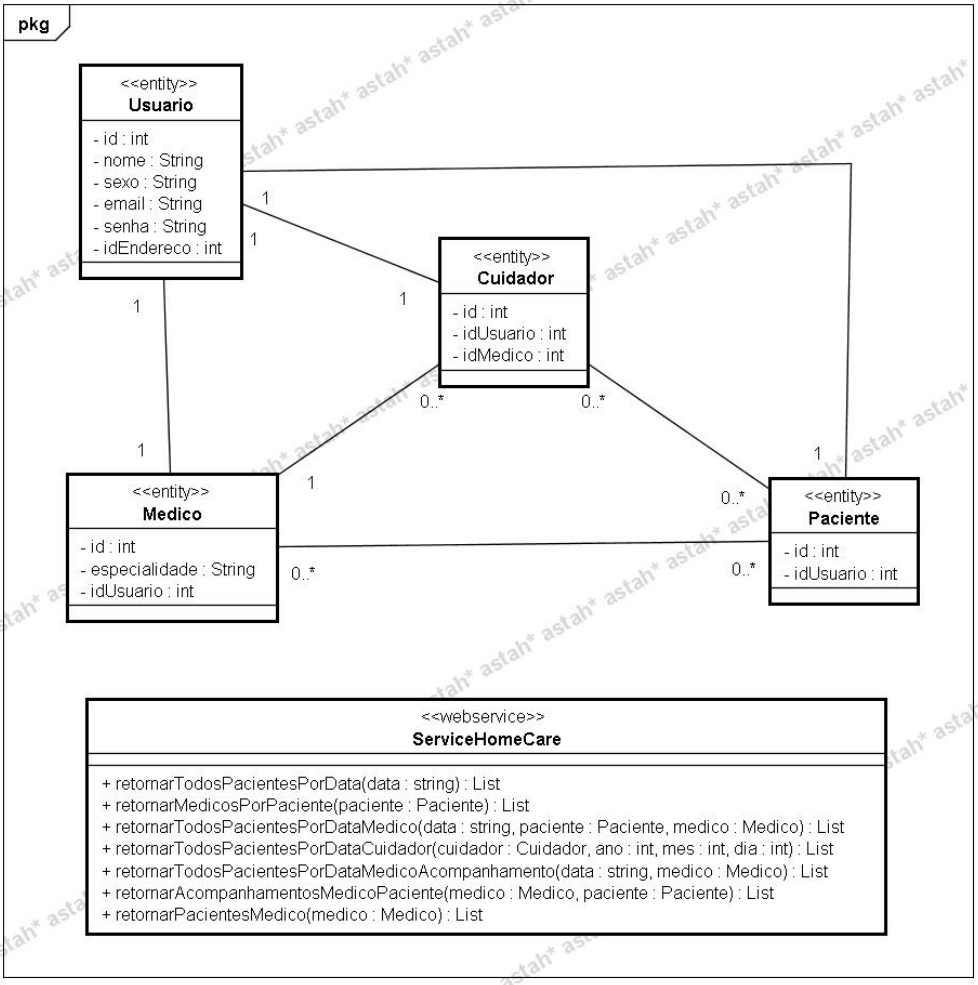

FIGURA 7 – Diagrama de classes com as principais entidades do serviço de aplicação. Fonte: o autor.

A classe “Usuario” representa todos os tipos de pessoas que irão utilizar o serviço, as quais podem ser médicos, cuidadores e pacientes, representados pelas classes de “Medico”, “Cuidador” e “Paciente” respectivamente.

A classe “ServiceHomeCare” é o *Web Service* responsável de fornecer as funcionalidades para realizar o acompanhamento dos pacientes no dia-dia da área de *home care*. A Tabela 1 descreve os principais métodos do serviço de aplicação.

Tabela 1 – Descrição dos principais métodos do serviço de aplicação.

Método	**Descrição**
`List<Paciente> RetornarTodosPacientesPorDataCuidador(Cuidador cuidador, int ano, int mes, int dia);`	Retorna todos os pacientes que devem ser atendidos pelo cuidador na data corrente.
`List<Acompanhamento> RetornarAcompanhamentosMedicoPaciente(Medico medico, Paciente paciente);`	Retornar a lista de todos os acompanhamentos realizados de um determinado paciente relacionado a um médico.
`List<Paciente> RetornarTodosPacientesPorDataMedicoAcompanhamento(string data, Medico medico);`	Retornar todos os pacientes que devem ser atendidos pelo médico na data corrente.
`List<Paciente> RetornarTodosPacientesPorData(string data);`	Retornar todos pacientes que foram atendidos na data corrente.
`List<Medico> RetornarMedicosPorPaciente(Paciente pac);`	Retorna todos os médicos que atenderam um determinado paciente.
`List<Paciente> RetornarPacientesMedico(Medico m);`	Retornar todos pacientes que estão relacionados a um médico.
`List<Acompanhamento> RetornarTodosPacientesPorDataMedico(string data, Paciente pac, Medico med);`	Retornar a lista de todos os acompanhamentos realizados de um determinado paciente relacionado a um médico na data corrente.

4.1.2 Serviço de Gerenciamento de Proxies

Esse serviço tem como propósito gerenciar os serviços *proxies* existentes nas redes locais e disponibilizar a lista de *proxies* ativos para os clientes móveis.

A forma de criação do WCF que representa esse serviço segue os mesmos passos do serviço de aplicação. A única diferença é que não foi preciso descrever os dados a serem transmitidos, pois esse *Web Service* define somente operações que não são necessárias a definição de tipos, ou seja, não foi preciso utilizar objetos específicos da lógica de negócios

do ambiente escolhido para avaliar a proposta desse trabalho. A Tabela 2 descreve os métodos desse serviço.

Tabela 2–Descrição dos métodos do serviço de gerenciamento de proxies.

Método	Descrição
`bool InstalarProxy(string ip, string nome);`	Adiciona um novo proxy na lista de proxies disponíveis para os clientes móveis.
`string[] ListarProxy();`	Retorna a lista de IP disponíveis para conexão em uma determinada rede local.

4.2 Camada de Proxy

O *proxy* implementado atende as requisições do clientes móveis que são repassadas para o serviço de aplicação. Essa delegação é feita de maneira transparente para usuário, pois no momento de enviar os dados o sistema verifica se existe um *proxy* local. Existe uma *thread* (gerenciador de submissão descrito na Figura 6 na camada de *proxy*) que recebe cada submissão e verifica se o serviço de aplicação está disponível para enviar os dados, e caso não esteja, ela ficará verificando qual o melhor momento para realizar essa ação e o usuário poderá sair da rede local. Após enviar os dados, o resultado da submissão é armazenado localmente em um banco de dados e quando o usuário realizar *login* novamente ele terá todos os resultados na sua tela inicial. A Tabela 3 descreve os métodos do gerenciador de submissões.

Tabela 3 – Descrição dos métodos do gerenciador de submissão.

Método	Descrição
`string GerenciarSubmissao( Acompanhamento acompanhamento);`	Gerenciar o envio de determinadas submissões solicitadas.
`List<int> ResultadoSubmissaoId( StreamReader idSubmissoes);`	Retorna uma lista com os ID das submissões realizadas por um cliente móvel.

O *Web Service* da camada de *proxy* (Figura 6) é o componente que realiza a passagem das requisições dos usuários do *proxy* para o serviço de aplicação, assim como também retorna os resultados desse serviço para os respectivos usuários que solicitaram, pois é ele que tem as configurações e ações necessários para acessar esse serviço. Nesse *Web Service* da

camada de *proxy* também foi definido uma ação de implantação do mesmo em uma rede local, onde essa ação seria realizada na sua primeira execução, ou seja, quando um novo *proxy* é inserido na rede ele solicita para o serviço *web* de gerenciamento de *proxies* (Figura 6) sua instalação, isso é possível através da classe "Global" derivada de "HttpAplication" existente no escopo desse tipo de aplicação, pois com ela é possível executar código na inicialização/finalização da aplicação.

O serviço *proxy* fornece um contrato para que os clientes possam acessar suas funcionalidades, já para ele ter acesso as ações dos *Web Services* "Gerenciamento de *proxies*" e "Serviço de Aplicação" ele assina os contratos fornecidos pelos mesmos. A Tabela 4 especifica os métodos do serviço de *proxy*.

Tabela 4 – Descrição dos métodos do serviço proxy.

Método	**Descrição**
`Void InserirSubmissao (object submissao);`	Adiciona uma nova submissão a ser solicitada ao servidor quando possível.
`List<Submissao> RetornarResultadoSubmissao( int[] idSubmissoes);`	Retorna o resultado de uma ou mais submissões realizadas por um cliente móvel.
`Protected void Application_Start(object sender, EventArgs e);`	Adiciona um novo *proxy* caso ele esteja executando pela primeira vez.

4.3 Desenvolvimento do Cliente WP7

Como forma de avaliar a proposta deste trabalho foi desenvolvido um protótipo de uma aplicação WP7 composta de alguns usuários com determinados papéis, descritos abaixo:

- Administrador: tem como responsabilidade o gerenciamento do sistema, ou seja, ele pode cadastrar/editar/excluir um médico, cuidador ou paciente, assim como também alocar cuidadores para um médico.
- Médico: responsável por gerenciar o tratamento dos pacientes durante o passar do tempo, ou seja, cadastrar medicamentos, realizar agendamentos e acompanhar o tratamento através de relatórios diversos.
- Cuidador: profissional tal como um auxiliar de enfermagem que tem como responsabilidade o registro das informações no dia-a-dia do paciente.
- Paciente: pessoa que vai ser acompanhada pelo médico e cuidador.

Para cada papel definido no sistema móvel, exceto o de paciente, foi desenvolvido um conjunto de funcionalidades. A Tabela 5 descreve as funcionalidades dos dois principais papéis do sistema móveis. O paciente apenas interage de forma indireta com o sistema, pois todas as funcionalidades relacionadas a ele são executadas por algum dos outros papéis.

Tabela 5 - Funcionalidade dos dois principais papéis do sistema móvel

Papel	**Funcionalidades**
Médico	• Agendar visita do paciente para o cuidador • Acompanhar visita agendada • Gerar relatório diário de uma visita de um paciente • Gerar relatório com todos pacientes cadastrados • Gerar relatório com todos cuidadores cadastrados • Gerar relatório de todas as visitas de um paciente
Cuidador	• Preencher formulário de visita de um paciente • Editar formulário de visita de um paciente

O sistema *home care* utiliza uma aplicação *web* para fins de administração dos dados do sistema, onde somente o papel de administrado tem acesso a esse ambiente. Nesta seção serão tratadas somente as principais telas do sistema móvel. A tela de login do sistema e a tela inicial do módulo do médico podem ser vistas na Figura 8.

O sistema móvel tem os módulos de cuidador e médico, onde cada módulo tem um conjunto de funcionalidades especificas. Na tela da Figura 8 o médico pode clicar no botão “GERENCIAR PACIENTES” para executar todas as ações relacionadas ao gerenciamento do paciente ou no botão “RELATÓRIOS GERAIS” para ver alguns relatórios do sistema, também pode ser visto na tela os resultados das submissões realizados pelo médico.

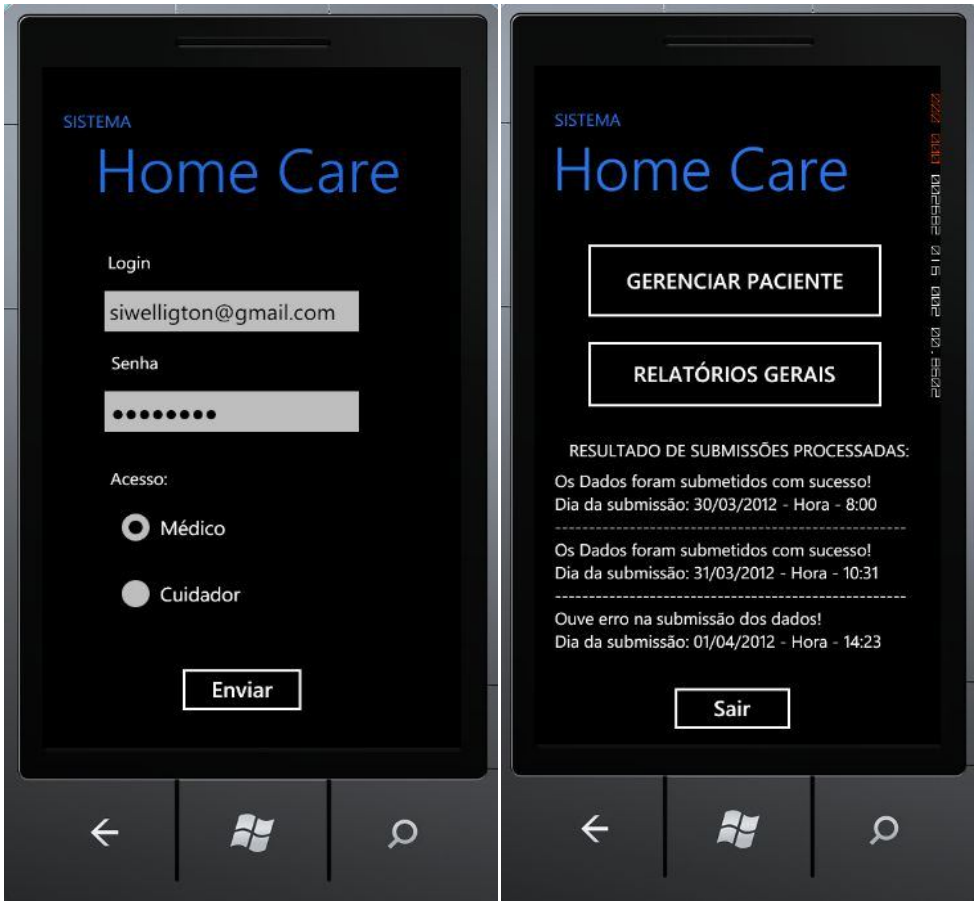

Figura 8 – Tela de login do sistema móvel/Tela inicial do módulo de médico.

Existem alguns relatórios gerais no módulo de médico. O primeiro deles é para consultar todos os pacientes cadastrados, ou seja, ele pode visualizar os pacientes que estão relacionados a ele e para verificar os dados cadastrados de um determinado paciente, basta selecionar um item da lista que corresponde à pessoal desejada. Semelhante ao primeiro relatório geral, também existe o relatório geral com todos cuidadores cadastrados, onde também pode visualizar os cuidadores que estão relacionados ao médico e também para verificar os dados pessoais do cuidador, basta clicar no item correspondente ao cuidador.

Os passos para relacionar os pacientes e cuidadores ao um médico são feitos na área de administração do sistema que foi mencionada anteriormente, onde o administrador realiza essas ações via *web*, pois é mais cômodo para ele realizar esse tipo ação em uma máquina, como por exemplo, um computador *desktop*, que ofereça mais espaço para visualização dos dados do sistema. Cada relatório está ilustrado nas Figuras 9 e 10 respectivamente.

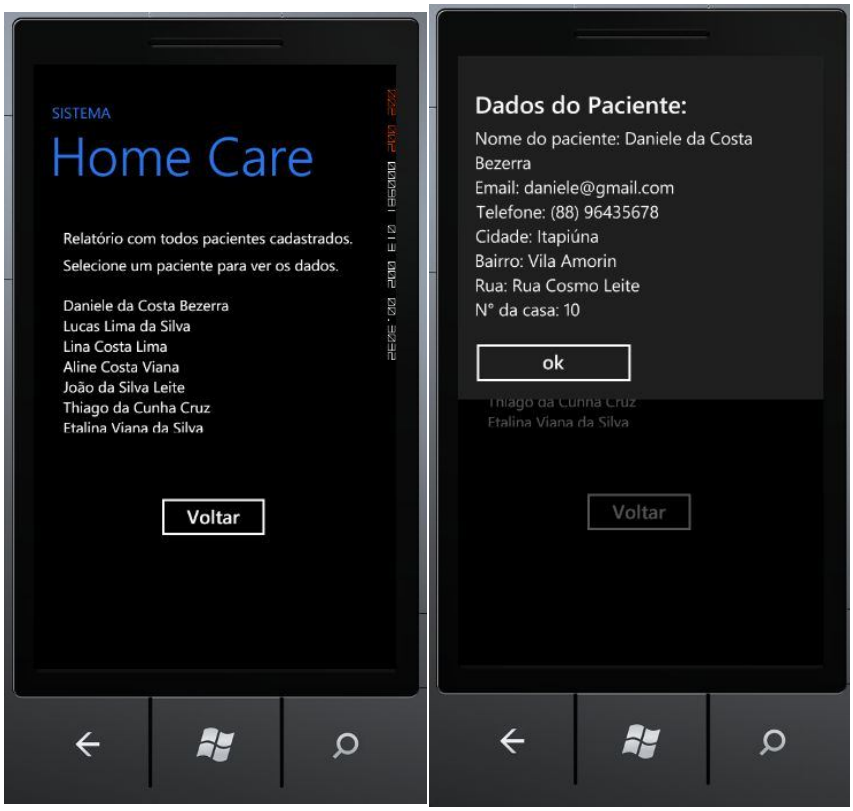

Figura 9 – Relatório de todos pacientes cadastrados/Dados de um paciente selecionado.

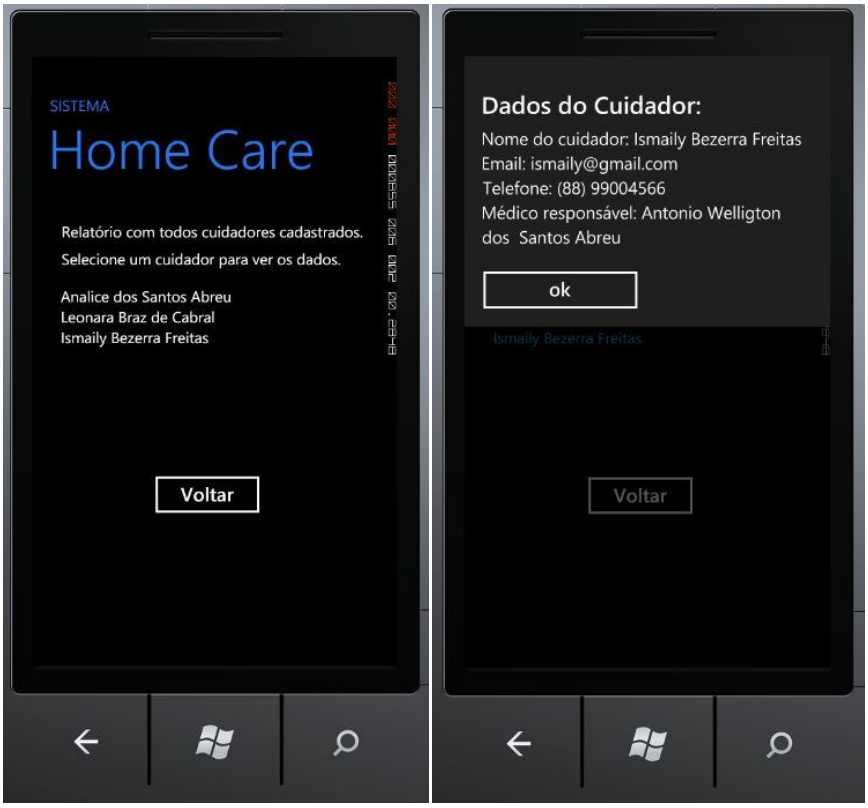

Figura 10 – Relatório de todos cuidadores cadastrados/Dados de um cuidador selecionado.

No gerenciamento de paciente o médico pode realizar o agendamento de visitas de um paciente, para isso, o primeiro passo é informar a data de início e fim do tratamento, onde durante esse tempo um ou mais cuidadores estarão realizando as visitas para realizar o acompanhamento do tratamento. O segundo passo é determinar quais medicamentos serão utilizados durante esse período de acompanhamento. Esses passos podem ser visualizados na Figura 11.

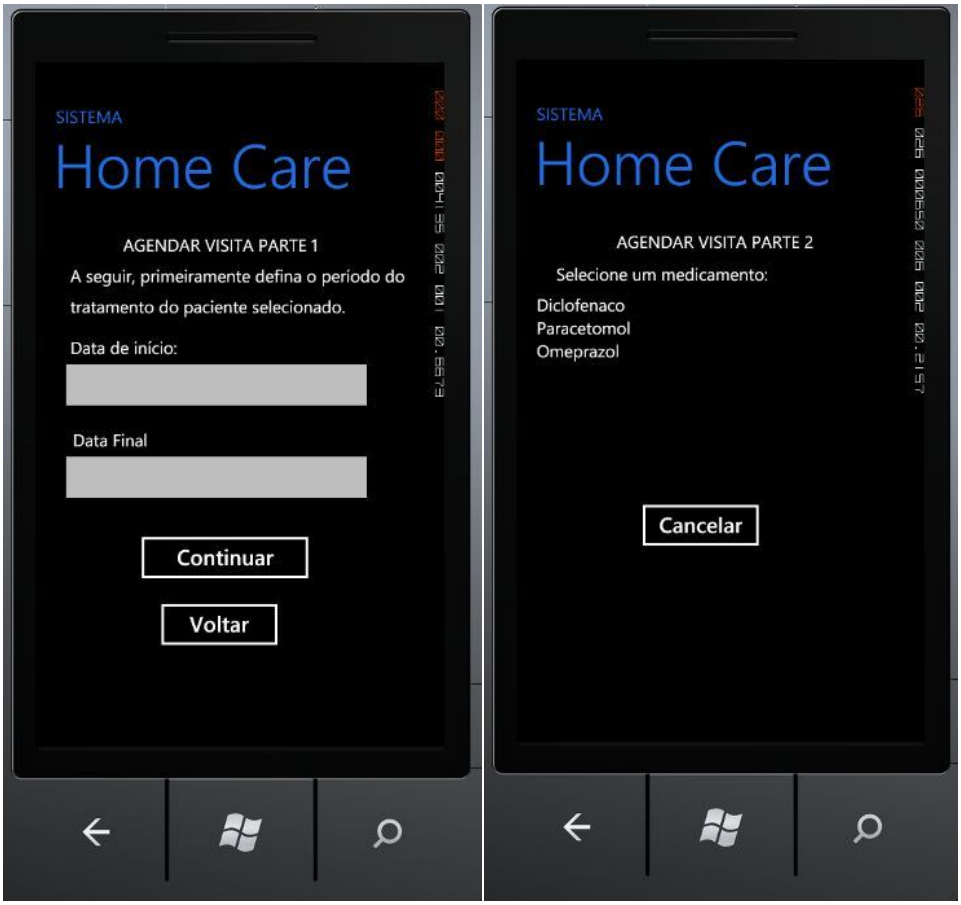

Figura 11 – Datas de início e fim do tratamento/Selecionar medicamentos utilizados.

Continuando a ação de realizar o agendamento de visitas, o terceiro passo é descrever a receita de cada medicamento selecionado, pois o médico pode escolher um ou mais remédios. O quarto passo, o profissional da área de saúde pode finalizar o agendamento, adicionar outro medicamento como foi dito anteriormente ou pode realizar o acompanhamento naquele momento caso ele queira, ou seja, necessário. Caso ele queira realizar o acompanhamento, isso geraria um quinto passo, ou seja, esse passo é opcional, pois quem irá realizar o acompanhamento é cuidador. Esses passos estão ilustrados na Figuras 12.

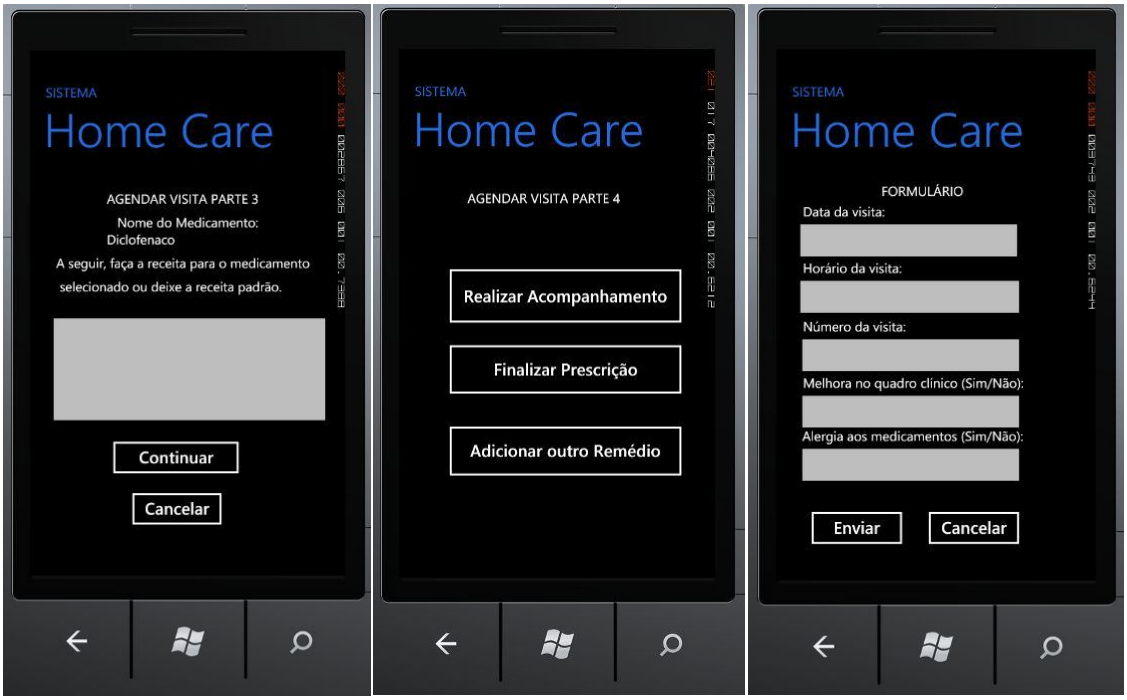

Figura 12 – Descrever receita/Escolher próxima ação/Realizar Acompanhamento.

O médico também pode visualizar todas as visitas (Figura 13) de um paciente ou visualizar os acompanhamentos (Figura 14) que foram realizados no dia, podendo verificar em cada ação os dados que foram registrados, para isso, basta selecionar uma visita da lista.

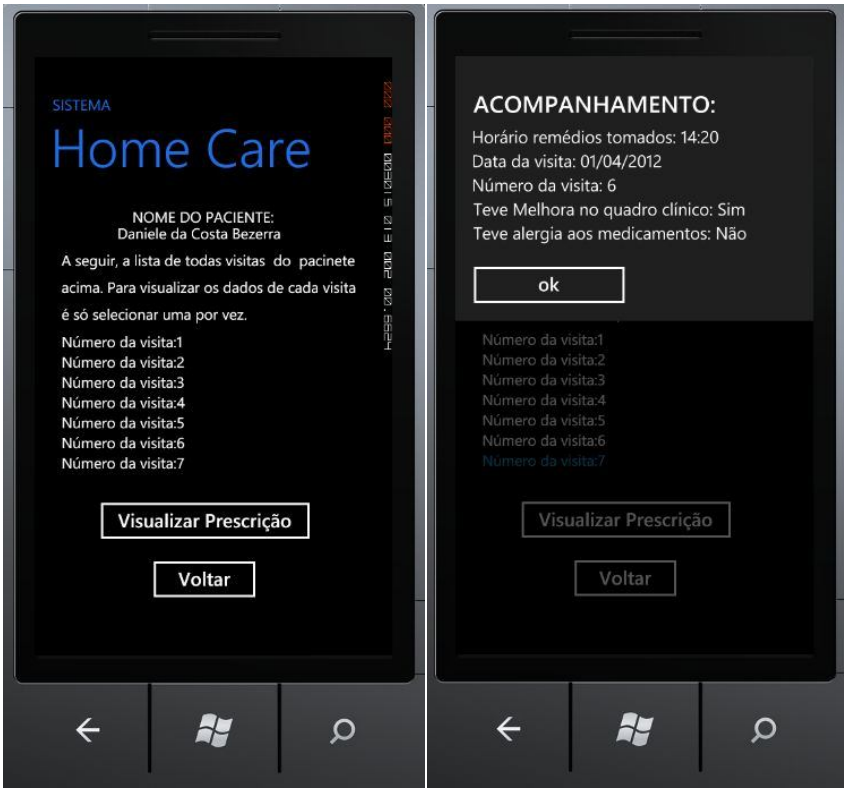

Figura 13 – Relatório com todas as visitas de um paciente/Dados de um acompanhamento.

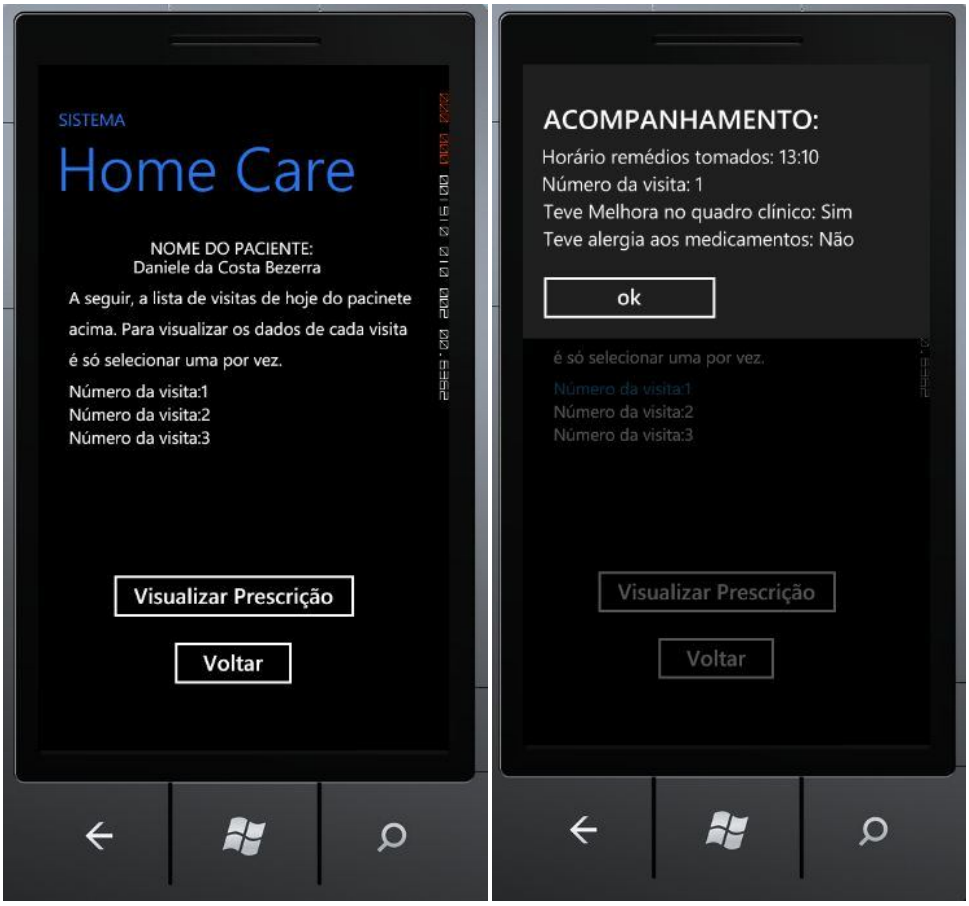

Figura 14 – Relatório das visitas diárias de um paciente/Dados de um acompanhamento.

Em relação ao módulo do cuidador a tela inicial pode ser vista na Figura 15. Nessa tela o cuidador pode visualizar os pacientes que devem ser atendidos no dia corrente. Semelhante ao módulo do médico, o administrador é quem define também quais pacientes estarão relacionados a um cuidador. O cuidador pode visualizar também na sua tela inicial os resultados das submissões realizadas por ele mesmo.

Para ver os dados de um paciente basta selecionar o item da lista correspondente ao mesmo. Esses dados podem ser muito úteis no caso de for preciso entrar em contato com um determinado paciente, ou seja, o cuidador pode ter dificuldades para chegar à residência do mesmo, assim ele poderia usar os dados de endereço ou simplesmente entrar contato por telefone para conseguir chegar ao local desejado.

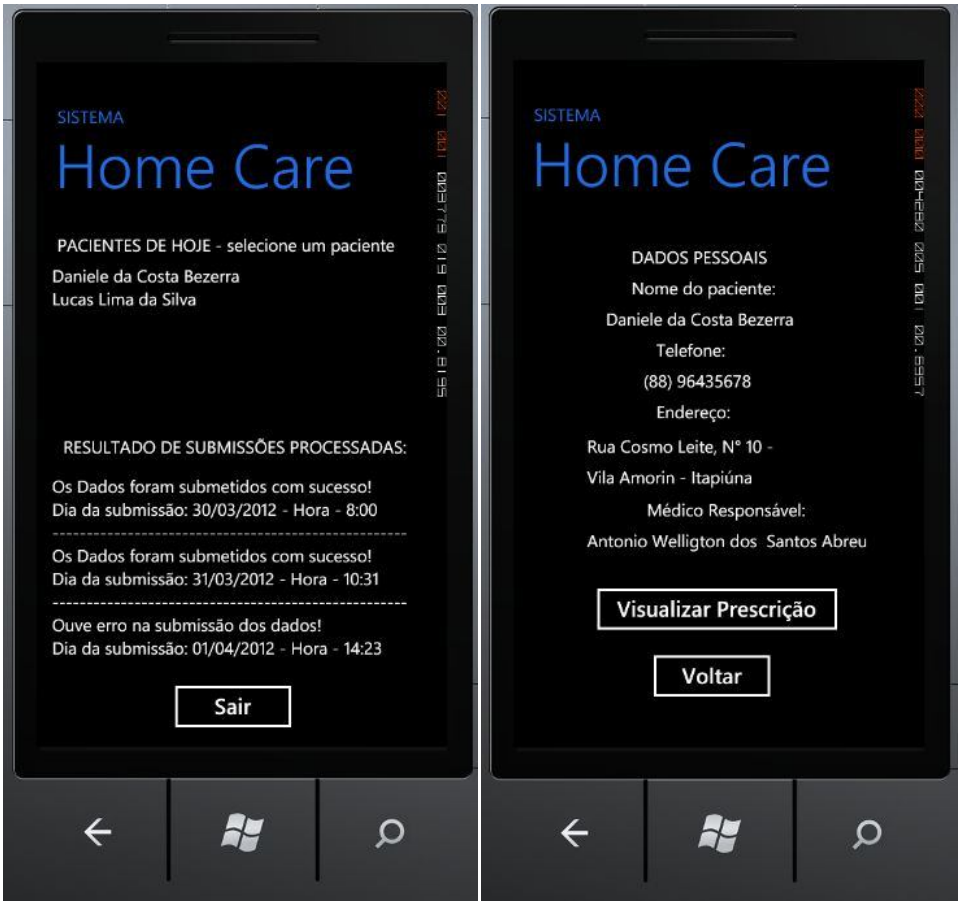

Figura 15 – Tela inicial do cuidador/Dados do paciente.

Na tela dos dados do paciente da Figura 15, pode-se iniciar o acompanhamento do paciente, pois basta o cuidador clicar no botão “Visualizar Prescrição”, onde irá aparecer a prescrição definida pelo médico (ilustrado na Figura 16). A prescrição é formada pelo período de tratamento e a lista de receitas definidas pelo médico. Cada receita tem um conjunto de informações que são formadas pelo nome do medicamento e como o mesmo deve ser administrado, para visualizar isso, basta selecionar uma determinada receita da lista que aparece na tela da Figura 16.

Nessa tela também aparecerá o botão “Preencher Formulário” que mostrará a tela responsável por realizar o acompanhamento do paciente. Após preencher as informações, o cuidador basta clicar no botão “Enviar” para salvar os dados do acompanhamento realizado naquele momento.

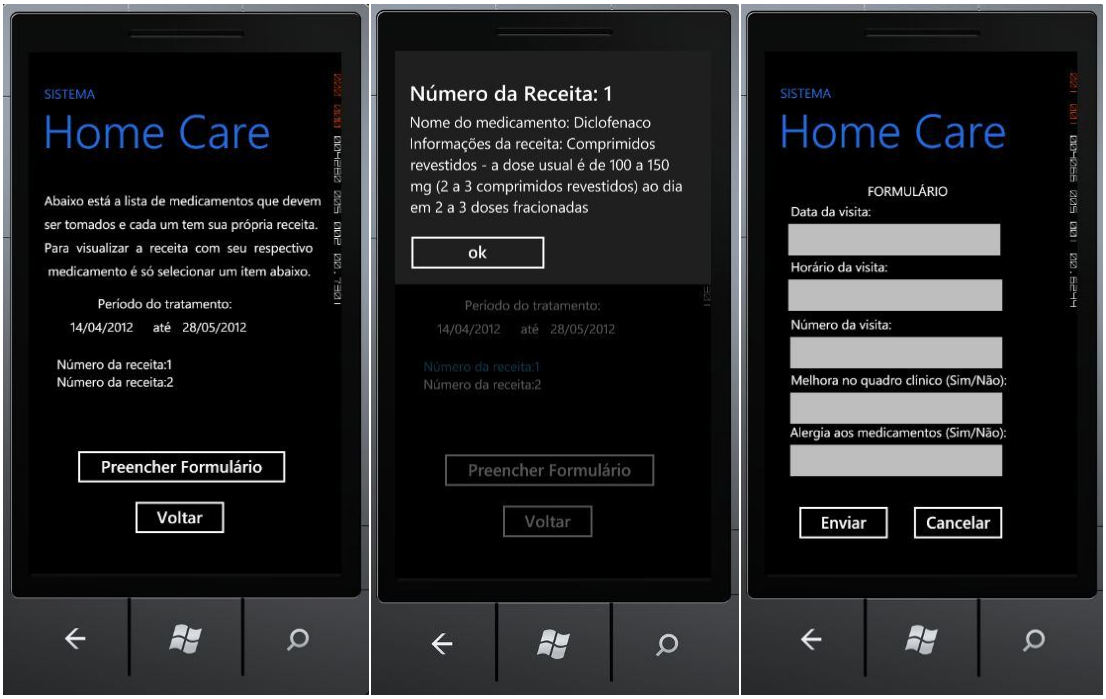

Figura 16 – Lista de Receitas/Dados de uma receita/Preencher formulário.

Para desenvolver à aplicação *Windows Phone* foi utilizada à linguagem C#, que no caso desse tipo de aplicação, adota o mesmo conceito de *code-behind* utilizado pela tecnologia ASP.NET. A plataforma *Silverlight* que utiliza a linguagem XAML foi usada para desenvolver a interface de usuário semelhante as das Figuras 8 a 16. Existe vários templates para se desenvolver uma aplicação *Windows Phone*, conforme mostra a Figura 17.

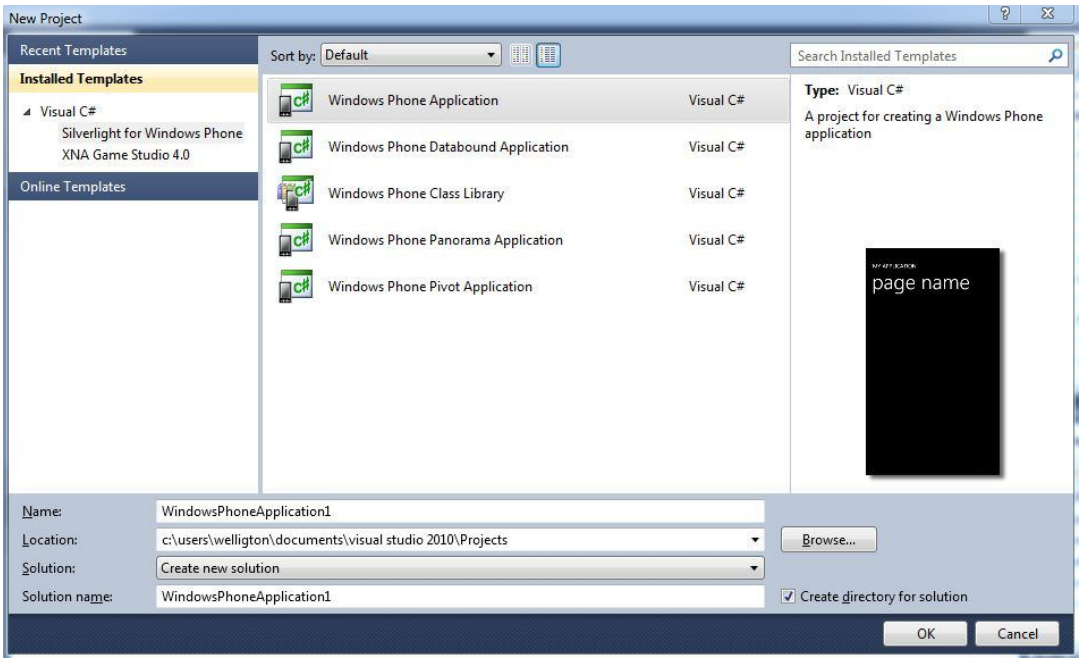

Figura 17 – Templates de aplicações Windows Phone.

Para a aplicação utilizada nesse trabalho foi escolhido o template *Windows Phone Application*. Os arquivos criados para essa aplicação são mostrados na janela *Solution Explorer*, conforme a Figura 18.

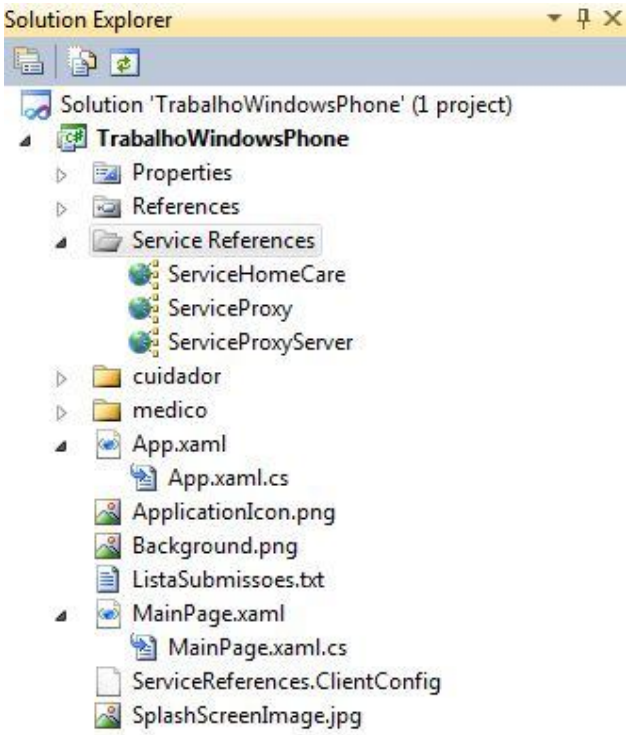

Figura 18 – Arquivos da aplicação da aplicação Windows Phone.

As figuras ApplicationIcon.png, Background.png e SplashScreenImage.jpg mostradas na Figura 18 representam respectivamente um ícone que é utilizado, à imagem de plano de fundo e a imagem de início da aplicação. O arquivo App.xaml, define uma classe do tipo "Application"que é responsável por inicializar e executar a aplicação *Windows Phone*. Outro arquivo XAML utilizado é o MainPage.xaml que define a tela principal da aplicação e utiliza uma classe do tipo "PhoneApplicationPage", onde também as demais telas que executam as funcionalidades dos módulos do cuidador e médico definidas respectivamente nas pastas "cuidador" e "medico" no *Solution Explorer* da aplicação, derivam desse tipo. Todos os arquivos XAML possuem um arquivo xaml.cs que correspondem ao seu *code-behind* para utilizar à linguagem C#.

O código XAML trata-se de um código XML com elementos visuais definidos hierarquicamente onde contém o elemento raiz "phone:PhoneApplicationPage", dentro do qual define um elemento "Grid" que tem como objetivo organizar os seu elementos internos

em linhas e colunas, semelhante uma tabela. Cada elemento visual definido em uma tela possui um objeto correspondente que pode ser manipulado através do C#.

A pasta Service References que aparece na Figura 18 armazena as referências dos WCF “ServiceHomeCare”, “ServiceProxyServer”e “ServiceProxy” que são os serviços *web* utilizados na aplicação e representam respectivamente o serviço de aplicação, serviço de gerenciamento de *proxies* e o serviço *proxy*. Essas referências são acessadas através do contrato fornecido por cada serviço *web*. O arquivo ListaSubmissoes.txt tem como objetivo armazenar os “ID” das submissões realizadas pelo cuidador ou médico, para que seja possível a recuperação do resultado da submissão posteriormente. Em relação ao arquivo ServiceReferences.ClientConfig presente no *Solution Explorer*da aplicação, ele configura os dados de cada serviço para seja possível acessar as informações disponibilizadas pelos WCF através da aplicação. A pasta “cuidador” representa todas as funcionalidades do módulo de cuidador, enquanto a pasta “medico” armazena todas as ações do módulo de médico. Os demais arquivos são propriedades e referências utilizadas para o tipo de template escolhido.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou um modelo de arquitetura para o ambiente móvel que tem como objetivo servir como um mecanismo para sincronizar o acesso a determinado recurso. Por meio dessa solução, os usuários podem ter certa comodidade no momento de enviar um conjunto dados para um servidor remoto.

5.1 Testes realizados

A avaliação deste trabalho foi realizada por meio de um estudo de caso, que simula a utilização do cenário descrito na Figura 4. Esses testes consistem em executar cada componente da arquitetura e verificar se atende o comportamento esperado. Utilizamos uma única rede local e instanciamos alguns clientes móveis através do módulo simulador do WP7 que a IDE Microsoft Visual Studio oferece para os desenvolvedores. Esse módulo pode simular o uso do WP7 em um celular que suporta sua tecnologia. Cada ação da Figura 4 foi testada, baseado no domínio que foi definido – o sistema de atendimento domiciliar. O tipo de submissões testadas foram o de enviar dados de formulários de acompanhamento de pacientes.

A avaliação dessa arquitetura mostrou que a comunicação e a organização dos componentes que formam a mesma foram implementadas de maneira satisfatória. Cada camada atendeu os objetivos que foram definidos e forneceram as funcionalidades necessárias para cada nível realizar suas ações.

5.2 Trabalhos futuros

Como trabalhos futuros, pretendemos avaliar a arquitetura em um ambiente real de atendimento domiciliar, utilizando um dispositivo móvel que suporta a tecnologia WP7, assim iremos verificar o comportamento de cada componente da arquitetura no ambiente real. Pretendemos também adicionar novas funcionalidades a arquitetura, tais como permitir que o *proxy* seja genérico, ou seja, que ele posso ser utilizado por qualquer tipo de aplicação, pois nesse trabalho ele está definido para o ambiente de atendimento domiciliar, e que os clientes móveis possam realizar a descoberta de qualquer tipo de *proxy*, pois o serviço *proxy* deste trabalho também está vinculado a esse tipo de ambiente.

REFERÊNCIAS

ARTAIL, Hassan;FAWAZ,Kassem;GHANDOUR, Ali.A Proxy-Based Architecture for Dynamic Discovery and Invocation of Web Services from Mobile Devices.IEEE Transactions on Services Computing, vol. 5, no. 1.EUA, january-march 2012.

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. 2.ed. Rio de Janeiro: Elsevier, 2007.

BIEGEL, Greg; CAHILL, Vinny; HAAHR, Mads. A Dynamic Proxy Based Architecture to Support Distributed Java Objects in a Mobile Environment. Departmentof Computer Science UniversityofDublin.Irlanda, 2002.

BRANDÃO, L. C. B. Considerações sobre a cobertura do sistema de Home Care pelos planos de saúde. São Paulo, 2010.

BRAGANÇA, A. Desenvolvimento Cliente-Servidor. Instituto Superior de Engenharia do Porto, Portugal, 1999.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim.Sistemas distribuídos. 4.ed. São Paulo: Pearson, 2007.

FLORIANI, Ciro Augusto; SCHRAMM, Fermin Roland. Atendimento Domiciliar ao Idoso. Rio de Janeiro, p. 2, jul./ago. 2003.

INSTITUTE, Project Management. Um Guia do Conhecimento em Gerenciamento de Projetos. 4ª Ed. EUA: Project Management Inst-id, 2008. 10p.

ISRAEL, Aece. WCF – Introdução. Janeiro de 2009. Disponível em: <http://www.israelaece.com/post/WCF-Introducao.aspx>.Acesso em 17 out. 2011.

MCMURTRY, Craig; MERCURI ,Marc; WATLING, Nigel; WINKLER, Matt. Windows Communication Foundation.Indiana, USA:Sams Publishing, 2007.

Microsoft Developer Network (MSDN). What's New in the Windows Phone SDK.23 Set. 2011. Disponível em: <http://msdn.microsoft.com/en-us/library/ff637516(v=vs.92).aspx>. Acesso em: 19 out. 2011.

TAMAE, R. Y.; LIMA, P. R. Web Services: uma nova visão da arquitetura de aplicações distribuídas na internet. Revista Científica Eletrônica de Sistemas de Informação, São Paulo, p.2, fev 2005.

TANENBAUM, Andrew Stuart; STEEN, Maarten Van.Sistemas Distribuídos. 2.ed. São Paulo: Pearson, 2007.