UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO ALVES NEUWALD

DIABETES CONTROL: UMA APLICAÇÃO MOBILE APLICADA AO GERENCIAMENTO DE INFORMAÇÕES MÉDICAS REFERENTES AO CONTROLE DO DIABETES

CRICIÚMA
2012

LEONARDO ALVES NEUWALD

DIABETES CONTROL: UMA APLICAÇÃO MOBILE APLICADA AO GERENCIAMENTO DE INFORMAÇÕES MÉDICAS REFERENTES AO CONTROLE DO DIABETES

Trabalho de Conclusão de Curso, apresentado para obtenção de grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: MSc. Gustavo Bisognin

Co-orientador: Esp. Fábio Bif Goularte

CRICIÚMA

2012

LEONARDO ALVES NEUWALD

DIABETES CONTROL: UMA APLICAÇÃO MOBILE APLICADA AO GERENCIAMENTO DE INFORMAÇÕES MÉDICAS REFERENTES AO CONTROLE DO DIABETES

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Desenvolvimento Mobile.

Criciúma, 26 de junho de 2012.

BANCA EXAMINADORA

Prof. MSc. Gustavo Bisognin (UNESC) - Orientador

Esp. Fábio Bif Goularte (UNESC) – Co-orientador

Prof. Esp. Fabrício Giordani (UNESC)

Profa. MSc. Christine Vieira Scarpato (UNESC)

A todos que torceram por mim durante toda minha jornada acadêmica.

AGRADECIMENTOS

Em primeiro lugar agradeço a minha família e namorada por compreender meus momentos de mau humor e ausência durante este período.

Agradeço também ao meu orientador Gustavo Bisognin e co-orientador Fábio B. Goulart por todo apoio e atenção dispensados ao longo do desenvolvimento do meu TCC.

Também tenho muito a agradecer a meus colegas de graduação e professores do curso de Ciência da Computação que participaram e contribuíram para meu crescimento acadêmico, em especial a professora Priscyla, que oportunizou meu primeiro projeto de pesquisa na área da saúde.

A meus colegas de trabalho que sempre me auxiliam e foram grande fonte de ajuda quando tive dúvidas.

Enfim, agradeço a todos que me auxiliam e que por uma questão de espaço não posso citá-los aqui.

“The two areas that are changing... are information technology and medical technology. Those are the things that the world will be very different 20 years from now than it is today.”

Bill Gates

RESUMO

O novo paradigma de computação móvel disponibilizada em *smartphones*, *tablets*, *pads* e dispositivos semelhantes, vêm se tornando maior e mais presente a cada dia. Sistemas *desktop* estão dando lugar a aplicações móveis nos mais diversos segmentos. Um segmento que está se destacando pela utilização da computação móvel é a saúde, onde a necessidade dos médicos possuírem informações sobre os pacientes é constante. A mobilidade e a facilidade de lidar com essas informações também é necessária, visto que profissionais de saúde estão sempre em movimento. Baseado nisto, essa pesquisa apresenta um modelo de computação móvel para a área da saúde, usando a tecnologia para dispositivos móveis Google Android. Esse modelo é aplicado no gerenciamento de informações de diabetes de paciente . A pesquisa buscou a utilização de tecnologias como JAX-WS, Java e Android para a criação de uma solução que possibilitasse o controle da diabetes da população. Desta forma, foi construído um modelo que permite a transmissão das informações da diabetes do paciente, geradas por um dispositivo móvel, para o dispositivo móvel do médico que está acompanhando esse tratamento através de um *Web Service*. Neste trabalho será apresentado como foi possível através de novas tecnologias a implementação de uma solução que possibilitasse melhor acompanhamento de médicos e pacientes no tratamento da diabetes.

Palavras-chave: Android, Java, Web Services, JAX-WS, Diabetes.

ABSTRACT

The new paradigm of mobile computing avaiable on smartphones, tablets and similar devices, are becoming bigger and more relevants each day. Desktop systems are giving way to mobile applications in several segments. One segment that is emerging by the use of mobile applications is health, where the need of having medical information about patients is constant. The mobility and facility of dealing with this information is also necessary because health professionals are always in motion. Based on this, this research presents a model of mobile computing for health care, using the Google Android technology for mobile devices. This model is applied to the management of information about patients diabetes levels. The survey sought the use of technologies such as JAX-WS, Java and Android to create a solution that would allow control of the population diabetes. This way, we built a model that allows the transmission of patients diabetes information, generated by a mobile device, to a mobile device of the doctor's who is leading this treatment through a Web Service. In this work will be presented how was possible through new technologies to implement a solution that would enable better monitoring of doctors and patients in the threatment of diabetes..

Keywords: Android, Java, Web Services, JAX-WS, Diabetes.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AVD	*Android Virtual Device*
CSS	*Cascading Style Sheets*
DM	Diabetes Mellitus
DVM	*Dalvik Vitual Machine*
HIV	*Human Immunodeficiency Virus*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IDE	*Integrated Development Environment*
IDF	*International Diabetes Federation*
ITU	*International Telecommunication Union*
J2ME	*Java 2 Micro Edition*
JPA	Java Persistence API
JVM	*Java Virtual Machine*
LED	*Light Emitting Diodes*
OHA	*Open Handset Alliance*
PDA	*Personal Digital Assistants*
PEP	Prontuário Eletrônico do Paciente
PSF	Programa de Saúde da Família
RMS	*Record Management System*
SAD	Sistemas de Apoio à Decisão
SBIS	Sociedade Brasileira de Informática em Saúde
SDK	*Software Development Kit*
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistemas de Informação
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
TI	Tecnologia da Informação
UDDI	*Universal Discovery, Description, and Integration*
UML	*Unified Modeling Language*
W3C	Worldwide Web Consortium
WHO	World Health Organization

WS	*Web Services*
WSDL	*Web Services Description Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Estima-se que existam atualmente no mundo, 1.500 milhões de televisores em uso, mais de 1 bilhão de pessoas estão conectados a Internet e quase 3 bilhões de pessoas têm um telefone celular, tornando-o um dos produtos de consumo de maior sucesso atualmente (OPEN HANDSET ALLIANCE, 2011).

O número de *smartphones* vendidos em 2010 foi 71% maior do que o vendido em 2009, algo em torno de 302 milhões de novos dispositivos. Segundo a mesma pesquisa, o número de aparelhos com sistema operacional Android era de 69 milhões de dispositivos em 2010 e possui uma tendência a aumentar proporcionalmente, sendo que em 2016 o Android deve estar em 45% dos *smartphones* ao redor do mundo, o que torna este mercado muito lucrativo (ABI RESEARCH, 2011, tradução nossa).

Segundo pesquisa da Gartner (2011), o lucro das lojas mundiais de aplicativos móveis irá crescer mais de 190% em 2011 em relação a 2010, ou seja, 15,1 bilhões de dólares em 2011. A loja de aplicativos da Apple App Store possui 9 de cada 10 *downloads* das lojas mundiais de aplicativos móveis. Ainda sim, lojas de aplicativos online como a Google Android Market, atual Google play e Microsoft Marketplace estão se tornando uma alternativa interessante de consumo de software *mobile* em relação a Apple App Store que atualmente lidera o segmento.

Ainda que a Apple App Store seja a maior loja de aplicativos, o desenvolvimento para o iOS, sistema operacional para dispositivos móveis da Apple, é fechado, não permitindo que sejam criados softwares fora de um computador Apple. Para realizar o *download* e possuir acesso ao conjunto de ferramentas para desenvolvimento no iOS, o site da Apple informa que é necessário possuir cadastro de desenvolvedor Apple, o que custa US$ 99,00 anuais, e ainda informa os requisitos técnicos para o desenvolvimento, que é possuir um Mac rodando Mac OS.

Hoje, um dos maiores desafios que a mobilidade dos softwares para saúde deve superar é na área de Telemedicina. O termo Telemedicina é apresentado na literatura desde a década de 60, e vem sendo adequada e aprimorada com o surgimento de novas tecnologias e necessidades relacionadas à saúde. Segundo Wen (2011) todas as definições de Telemedicina apontam para a possibilidade de proporcionar cuidados médicos em situações onde a distância é considerada um dos maiores fatores críticos.

Utilizando-se da sincronização de sistemas, deve ser possível ao médico, coletar informações de cuidados ao paciente em um dispositivo móvel mesmo que esteja fora de seu estabelecimento de saúde. Em atendimentos fora de seu estabelecimento, e até mesmo em algumas situações dentro do estabelecimento, o profissional de saúde pode não possuir acesso a uma rede de comunicação no momento em que estiver coletando informações do paciente. Nesta situação é necessário que o profissional possa coletar os dados utilizando um aplicativo que permita realizar a transmissão de dados para outro sistema somente quanto possuir acesso a rede, assim não prejudicando a coleta das informações.

Segundo Dolan (2011), foram procurados aplicativos móveis de saúde nas lojas da iPhone AppStore, Google Android Market, BlackBerry App World, App Catalog da Palm e da Nokia Ovi Store, e menos de 6 mil softwares foram encontrado. Sendo que destes 70% eram destinadas a usuários e 30% destinados aos profissionais da saúde. Na iPhone AppStore foi verificado que o maior número de softwares está disponível de forma grátis para pacientes, mas que conforme o preço aumenta, os aplicativos são mais voltados para os prestadores de saúde. Ainda com base nos dados apresentados se estima que a Google Android Market, atual Google Play atingiu mais de 3 milhões de *downloads* com seus aplicativos de saúde.

Aplicativos de saúde móveis terão papel importante na definição do futuro de sistemas de saúde pública e privada, tendo sido considerado uma das tendências *mobile* de 2010. Segundo Bedran (2011), aplicativos móveis podem conectar pacientes e prestadores de saúde, fazendo com que o histórico do paciente seja muito mais rico e de fácil acompanhamento, uma vez que o paciente possui o acompanhamento médico em suas mãos. Aplicativos como o *Diamedic* permitem que diabéticos gravem seus níveis de açúcar no sangue e suas doses de insulina, esses dados podem ser enviados diretamente a um prestador de saúde que poderá acompanhar diariamente as informações atualizadas de seu paciente.

Em 2008, a Prefeitura de São Paulo realizou uma parceria entre a Secretaria Municipal de Saúde e a Universidade Federal de São Paulo (Unifesp), onde foram instalados aparelhos de eletrocardiograma a distância em 126 ambulâncias e 4 pronto-socorros. O eletrocardiograma transmite os dados do exame do paciente via celular da ambulância para a equipe de cardiologistas que está nos pronto-socorros. Neste momento, dependendo da gravidade o tratamento começa dentro da própria ambulância. Os resultados desta ação podem ser considerados ótimos, antes a percentagem de mortalidade por infarto era de 21% e após o uso dos dispositivos móveis, esse número foi reduzido para 6% (BOUÇAS, 2011).

Diante disto, será realizado o desenvolvimento de uma solução móvel com tecnologia Android, em que seja possível controlar os níveis de diabetes de um paciente

independentemente do acesso a internet. No modelo, o sistema deve se comunicar com um módulo instalado no dispositivo de um profissional da saúde, assim possibilitando o envio e recebimento de informações através da sincronização de dados.

1.1 OBJETIVO GERAL

Implementar um aplicativo compatível com a tecnologia do sistema operacional Android, em que seja possível controlar os níveis de diabetes de pacientes, sincronizando informações entre dispositivos de pacientes e profissionais de saúde. O sistema deve operar de forma *stand-alone* caso não existe conexão a rede.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

a) realizar estudo sobre tecnologia Android;
b) realizar estudo sobre tecnologia de sincronização de sistemas;
c) implementar um aplicativo nativo para Android, que seja compatível com dispositivos móveis como *tablets* e *smartphones*.
d) implementar um WebService que sirva como ponte de conexão entre o aplicativo desenvolvido;
e) implementar um aplicativo que realize o controle de diabetes dos pacientes, possibilitando a troca de informações com outros dispositivos;
f) disponibilizar modelo na Google play.

1.3 JUSTIFICATIVA

Segundo Pereira e Silva (2009), o Android é o primeiro projeto de uma plataforma Open Source para dispositivos móveis em conjunto com a Open Handset Alliance (OHA), permitindo o desenvolvimento de aplicativos usando o Kit de Desenvolvimento de Software (SDK). A OHA desenvolveu o Android com a intenção de permitir aos desenvolvedores criarem aplicações que possam tirar o máximo de proveito do aparelho portátil, provendo sistema operacional, *middleware*, aplicativos e interface com o usuário.

Segundo Lecheta (2009), a força do Android se deve ao grupo Open Handset Alliance que tem como integrantes algumas das empresas mais consagradas da telefonia, semicondutores, softwares e fabricantes de aparelhos, como: HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, Sprint Nextel, China Mobile, T-Mobile, ASUS, Intel, Garmin eBay, Telefônica, Ascender e a Google como principal participante.

O Governo lançou no dia 23 de maio a medida provisória 534/11 que inclui os *tablets* na "Lei do Bem" (Lei nº 11.196), de 2005, que incentiva projetos e produtos de inovação tecnológica. Essa medida pode reduzir os impostos para os *tablets* em até 36%, o que tornaria esses dispositivos mais populares e também aqueceria o mercado de desenvolvimento (COLLETTA, 2011). Ainda segundo a Computer World (2011), as empresas Foxconn, AIOX, Compalead, Envision, Itautec, LG, Motorola, MXT, Positivo, Samnia, Samsung e Semp Toshiba já manifestaram interesse em fabricar *tablets* no Brasil. Dentre as empresas que manifestam o interesse, algumas participam do grupo OHA, o que podem favorecer ainda mais o Android.

A AirStrip Technologies é uma empresa especializada em desenvolvimento de softwares que gerem dados para celulares e *tablets* (AIRSTRIP TECHNOLOGIES, 2011, tradução nossa). Recentemente uma parceria entre as empresas AirStrip Technologies e GE Healthcare resultou na solução conhecida como AirStrip CARDIOLOGY™, onde, segundo o Business Wire (2011) o sistema desenvolvido permite que médicos analisem eletrocardiogramas (ECG) a qualquer momento e em qualquer local através de *tablets* e celulares. Atualmente a solução está disponível somente para iPhone e iPad, mas os demais programas da AirStrip Technologies estão disponíveis para diversas outras plataformas, incluindo a plataforma Android.

Outro exemplo de sucesso de dispositivos móveis voltados para área de saúde foi o Projeto Borboleta, desenvolvido no Instituto de Matemática e Estatística (IME) da USP. O projeto foi desenvolvido por um grupo de profissionais do Centro de Saúde-Escola Samuel Pessoa da Faculdade de Medicina da Universidade de São Paulo (FMUSP) e poderá elevar a qualidade dos serviços prestados aos 87,7 milhões de brasileiros atendidos pelo Programa Saúde da Família (PSF). O funcionamento do sistema é simples, antes de deixar o centro de saúde, o médico acessa a plataforma de informação do sistema que prepara um arquivo com os dados do prontuário dos pacientes que vão ser visitados naquele dia, assim o prestador de saúde realiza o *download* dos dados para o celular. Após realizar os atendimentos e registrar os mesmos no aparelho, o médico volta ao centro de saúde, e faz o *upload* dos dados,

atualizando o prontuário dos pacientes que foram atendidos (PRONTUÁRIO NO CELULAR, 2011).

Segundo Fred O'connor (2011), 71% dos médicos consideram *smartphones* essenciais para a prática médica e 84% disseram que a internet é fundamental. Alguns hospitais já possibilitam que pacientes realizem exames do sono em casa se utilizando de dispositivos sem fios para coletar e transmitir seus dados para médicos através da internet, reduzindo o custo de um exame de US$1,200 para US$300.

Uma das doenças que podem ser tratadas sem o contato com o médico é a diabetes. Segundo pesquisas da *International Diabetes Federation* (IDF) a diabetes já atinge no mundo mais de 280 milhões de pessoas com idade entre 20 e 79 anos. Somente no Brasil, a DM atinge mais de 7 milhões de pessoas (IDF, 2010).

Diante deste contexto, valendo-se da premissa que o sistema operacional Android será líder do mercado em poucos anos, e, aliado a necessidade crescente de utilização de sistemas voltados a área da saúde ligando médicos e pacientes, propõem-se a análise e desenvolvimento de um aplicativo para *tablets* e *smatphones* que possa estar disponível na Google Android Market atual Google Play para prestadores de saúde e pacientes.

1.4 ESTRUTURA DO TRABALHO

A presente pesquisa está dividida em seis capítulos. No primeiro capítulo é apresentada uma introdução ao tema proposto, os objetivos gerais e específicos e a justificativa para realização deste projeto.

O segundo capítulo é apresentado o esquema conceitual e seu kit de desenvolvimento do sistema operacional para dispositivos móveis Google Android. Neste capítulo também é possível compreender sobre os próprios dispositivos móveis e sua importância tecnológica.

O conceito de sincronização de sistemas via WebService e tecnologias disponíveis para desenvolvimento destes são apresentadas no terceiro capitulo.

O quarto capítulo é apresentado o conceito geral de informática em saúde, telemedicina e diabetes. Neste capítulo também é demonstrado como a tecnologia móvel vem se aliando a medicina e gerando bons resultados no tratamento a enfermidades.

Os trabalhos correlatos são descritos no quinto capítulo proporcionando uma visão mais abrangente sobre a utilização da tecnologia móvel na prática médica.

Por fim, todo desenvolvimento prático da aplicação, bem como a modelagem conceitual, técnicas e metodologia são apresentados no sexto capitulo.

2 A PLATAFORMA ANDROID

Em 2005 a Google, adquiriu uma *start-up*[1] com foco no desenvolvimento de aplicações para dispositivos móveis, a Android Inc. (STEELE; TO, 2011, tradução nossa). A compra da *start-up* fazia parte de uma estratégia da Google para entrar no mercado de dispositivos móveis (LEE, 2011, tradução nossa).

A Google queria que o Android fosse livre e aberto, e hoje grande parte do código fonte do Android se encontra sobre a permissão *open-source* Apache License[2], permitindo que qualquer um realize o *download* completo de seu código fonte. Diversas empresas de software e manufaturados adicionam extensões proprietárias e customizam o Android de diferentes maneiras, sendo está uma das especialidades de grandes empresas que foram afetadas pelo aparecimento do iPhone da Apple, que revolucionou o mercado de *smartphones* (LEE, 2011, tradução nossa).

2.1 VISÃO GERAL

O Android é um conjunto de sistema operacional, *middleware* e softwares aplicativos para dispositivos móveis. O *Software Development Kit* (SDK) da Android dispõe dentre diversas ferramentas de um *Application Programming Interface* (API) para que seja possível criar aplicações utilizando Java como linguagem de programação (GOOGLE, 2012, tradução nossa).

Antes do sistema operacional da Google, o desenvolvimento de aplicações móveis era muito fragmentado existindo uma variedade de plataformas para desenvolvimento voltado à dispositivos móveis. Existiam dispositivos rodando com diversos sistemas operacionais como Palm OS, RIM BlackBerry OS, Java Micro Edition (ME), Symbian OS, iPhone OS dentre outros, o que fragmentava ainda mais o desenvolvimento de aplicativos para dispositivos móveis, pois não era possível escolher somente uma plataforma de desenvolvimento (CONDER; DARCEY, 2010, tradução nossa).

O centro de pesquisa e desenvolvimento da Google, que já disponibiliza uma série de serviços gratuitos resolveu apostar nos dispositivos móveis, pregando dentro deste mercado a mesma filosofia que aplicada na *web*. Para entrar neste mercado, a Google se uniu a uma série de empresas interessadas na tecnologia móvel, e fez a pergunta: Como podemos

[1] Empresa jovem, normalmente em estado de desenvolvimento.
[2] Para mais informações, acesse: http://www.apache.org/licenses/LICENSE-2.0

construir o melhor aparelho celular? E, com isso, a Open Handset Alliance (OHA) foi criada em novembro de 2007 na intenção de responder a essa questão (CONDER; DARCEY, 2010, tradução nossa).

A OHA possui como integrantes muitas das maiores e mais bem sucedidas empresas das áreas de desenvolvimento de software, criação de aparelhos, provedores de serviços, operadoras de celulares e criadores de chips (CONDER; DARCEY, 2010, tradução nossa).

A OHA é composta 84 empresas, que juntas com o Google, desenvolvem o Android como uma plataforma aberta e livre (OPEN HANDSET ALLIANCE, 2011, tradução nossa). Entretanto a participação da Google no projeto do Android é tão extensa que aparentemente a responsabilidade da empresa é muito maior que o grupo como um todo. A empresa hospeda o projeto *open-source*, fornece documentação *online*, ferramentas, fóruns e o SDK para os desenvolvedores (CONDER; DARCEY, 2010, tradução nossa).

Desde seu lançamento na versão SDK 1.0, já foram desenvolvidas diversas versões do SO. A Tabela 1 apresenta as versões desenvolvidas até o momento:

Tabela 1 - Versões mais importantes do Android

Versão	Data de Lançamento	Apelido	API Level
Android 1.5	04/2009	*Cupcake*	3
Android 1.6	09/2009	*Donut*	4
Android 2.0	10/2009	*Eclair*	6
Android 2.0.1	12/2009	-	6
Android 2.1.x	01/2010	-	7
Android 2.2.x	05/2010	*Froyo*	8
Android 2.3	12/2010	*Gingerbread*	9
Android 3.0	02/2011	*Honeycomb*	11
Android 3.1	05/2011	-	12
Android 3.2	07/2011	-	13
Android 4.0	10/2011	*Ice Cream Sandwich*	14
Android 4.0.3	12/2011	-	15

Fonte: GOOGLE (2012)

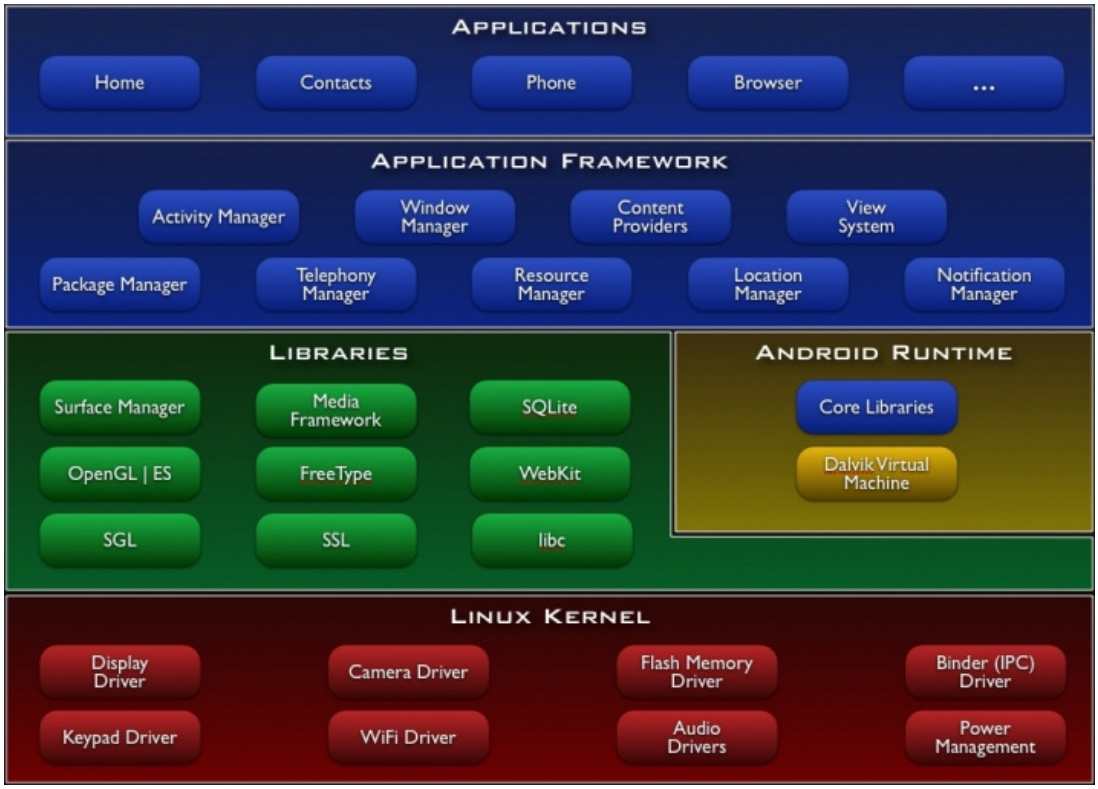
2.2 ARQUITETURA ANDROID

O sistema operacional Android foi construído dentro de um paradigma *mobile* totalmente voltado a *web* e integrado com diversos aplicativos e ferramentas da Google. Neste contexto, o SO apresenta-se para o mercado como um contêiner onde diversas aplicações podem ser facilmente baixadas da internet e instaladas, abordando os mais diversos domínios do conhecimento (MORRIS, 2011, tradução nossa).

Uma das principais características apresentadas pelo SO Android, é sua facilidade de uso, apresentando uma interface intuitiva com ícones grandes para suportar o formato *touch screen.* Outra característica importante é a alta portabilidade e desempenho apresentada pelo *kernel* podendo ser instalado em diversos aparelhos como *gadgets, smartfones, tablets,* eletrodomésticos e televisores (MORRIS, 2011, tradução nossa; STEELE; TO, 2011, tradução nossa).

A construção do Android é baseada no *kernel* do sistema operacional Linux 2.6, apresentando uma arquitetura robusta e estável, concedendo maior eficiência e confiabilidade as camadas de alto nível da aplicação. Uma representação gráfica das cinco camadas da arquitetura Android, pode ser observada na Figura 1 (GOOGLE, 2012, tradução nossa).

Figura 1 – Arquitetura Android



Fonte: GOOGLE (2012)

Conforme observado na Figura 1, o SO Android é subdividido nas seguintes partes básicas: *Linux Kernel*, *Libraries*, Android *Runtime*, *Aplication Framework* e *Applications*. Uma visão detalhada de cada uma destas partes é apresentada nos tópicos a seguir.

2.2.1 Aplicações

Na camada de aplicações, que fica acima das demais camadas, é onde se encontram todas as aplicações fundamentais do SO como cliente de e-mail, mapas navegadores, calendários, programas de envio de mensagens, agendas, contados. Nesta camada também são encontrados os aplicativos desenvolvidos pela comunidade Android, em Java (GOOGLE, 2012, tradução nossa; PEREIRA; SILVA 2009).

Está é a camada que mais chama a atenção dos usuários, e por este motivo empresas que distribuem o Android dentro de seus dispositivos, costumam realizar alterações na camada visual do sistema operacional, fazendo com que cada dispositivo tenha um sistema operacional Android com características visuais e aplicativos nativos diferentes das demais (MORRIS, 2011, tradução nossa).

2.2.2 Framework

Frameworks são estruturas voltadas ao desenvolvimento de softwares de larga escala. O principal objetivo dos *frameworks* é aumentar a produtividade durante o desenvolvimento do software e prover a reutilização de código, disponibilizando ao desenvolvedor uma série de recursos prontos, diminuindo assim a quantidade de código a ser escrito (RIEHLE, 2000, tradução nossa).

Por meio da camada de *framework* do Android, é disponibilizada a API e os recursos utilizados pelos aplicativos como as classes virtuais, provedores de conteúdo, gerenciadores de recursos, localização, notificação, pacotes, atividades entre outros. O *framework* foi criado de forma a permitir um desenvolvimento extremamente rico e inovador para que desenvolvedores pudessem retirar a máxima vantagem de dispositivos (GOOGLE, 2012, tradução nossa; PEREIRA; SILVA 2009).

Uma das principais características do *framework* é que ele possibilita aos desenvolvedores o reaproveitamento de componentes, sendo que uma aplicação pode

disponibilizar recursos, e outra aplicação pode acabar utilizando esses recursos disponibilizados, isso claro, se os níveis de segurança adotados pelo *framework* permitirem (GOOGLE, 2012, tradução nossa)

Conforme é possível verificar na Figura 2, existem diversos elementos na camada de *framework*. Abaixo da figura é possível verificar mais detalhadamente cada um destes itens.

Figura 2 – Camada de Framework do Android

Fonte: BRAY (2011)

a) *activity manager*: gerencia o ciclo de vida de todas as *activities*, quando iniciar e quando terminá-las;
b) *package manager*: é utilizado pelo *activity manager* para ler as informações dos pacotes de arquivos do Android. A *package manager* se comunica com o sistema informando sobre quais os pacotes estão sendo utilizados e quais são as suas capacidades;
c) *window manager*: se comunica com as janelas, gerenciando as que vão estar ativas e quais não estarão;
d) *content providers*: é quem possibilita a troca de informações entre aparelhos e aplicativos;
e) *view system*: possui todo o tratamento gráfico para os aplicativos como botões e frames.

Além dos recursos citados acima, também existem outros elementos não menos importantes como os *Locations Service*, *Bluettooth Service*, *Wi-Fi Service*, *USB Service*, e *Sensor Service* (PEREIRA; SILVA 2009).

2.2.3 Bibliotecas

Acima do *kernel* do Linux 2.6, o Android foi desenvolvido com diversas funcionalidades. Essas funcionalidades são disponibilizadas para os desenvolvedores por meio da estrutura de aplicação do Android (GOOGLE, 2012, tradução nossa).

A camada de bibliotecas traz consigo diversas bibliotecas *C/C++* utilizadas pelo sistema, incluídas nesse conjunto da biblioteca *C* padrão. Também existem uma série de bibliotecas para áreas multimídia, visualização de camadas 2D e 3D, funções para navegadores *web*, funções para gráficos, funções de aceleração de hardware, renderização 3D, fontes *bitmap* e vetorizadas e acesso ao banco SQLite (PEREIRA; SILVA 2009).

Figura 3 – Camada de Bibliotecas do Android

Fonte: BRAY (2011)

Podemos citar como sendo as principais bibliotecas disponíveis (GOOGLE, 2012, tradução nossa; PEREIRA; SILVA 2009):

a) *freetype*: renderização de fontes e bitmaps;
b) *system C library*: biblioteca *C bionic* que é uma biblioteca especialmente desenvolvida, customizada e otimizada para uso embutido no Android;

c) *webkit*: renderizador de páginas para navegadores, possuindo suporte a *Cascading Style Sheets* (CSS), javascript, DOOM e AJAX baseado no código aberto do navegador *webkit*;
d) SQLite: poderosa máquina de banco de dados relacional que está disponível para todas as aplicações de maneira leve e embutida;
e) SGL: responsável pelos gráficos 2D;
f) *surface Manager*: fornece acesso aos subsistemas de exibição para as camadas de aplicações 2D e 3D;
g) *media libraries*: conjunto de bibliotecas para suporte as os formatos mais populares de áudio e vídeo;
h) *LibWebCore*: navegador web moderno utilizado no Android;
i) 3D libraries: implementação baseada na API do *OpenGL* ES 1.0 para aceleração 3d via hardware ou o software de renderizaçao 3D.

Além das bibliotecas citadas acima, existem uma série de bibliotecas não tão conhecidas já estão disponíveis dentro do framework do Android conforme é possível verificar na Figura 3 (PEREIRA; SILVA 2009; GOOGLE, 2012, tradução nossa).

2.2.4 Android Runtime

Ao executar uma aplicação Android, é criada uma instancia da Dalvik Vitual Machine (DVM) em ambiente de execução. As aplicações escritas em Java são compiladas em *bytecodes* Dalvik e executadas dentro do DVM, assim sendo possível distribuir essas aplicações em qualquer dispositivo que possua Android e a DVM (PEREIRA; SILVA 2009).

As *Core Libraries* são um grupo de bibliotecas que fornecem grande parte das funcionalidades disponíveis nas bibliotecas da linguagem Java. Essas estruturas permitem aos desenvolvedores criar aplicações utilizando de um conjunto de bibliotecas de estruturação, acesso a arquivos, redes e gráficos já existentes no Java. Dentro das *Core Libraries* não se encontram classes que não fazem sentido para o desenvolvimento de aplicativos móveis (LEE, 2011, tradução nossa; FELKER; DOBBS, 2011, tradução nossa).

A DVM foi projetada para rodar em dispositivos com pouca memória e que fossem desenhados para permitir que várias instâncias de uma máquina virtual fossem executadas simultaneamente. Como os softwares desenvolvidos para o Android são escritos em Java, existe uma grande confusão, onde se refere a VM da Dalvik como uma *Java Virtual Machine* (JVM), quando isso não é verdade. O *bytecode* da DVM não é o mesmo da JVM do

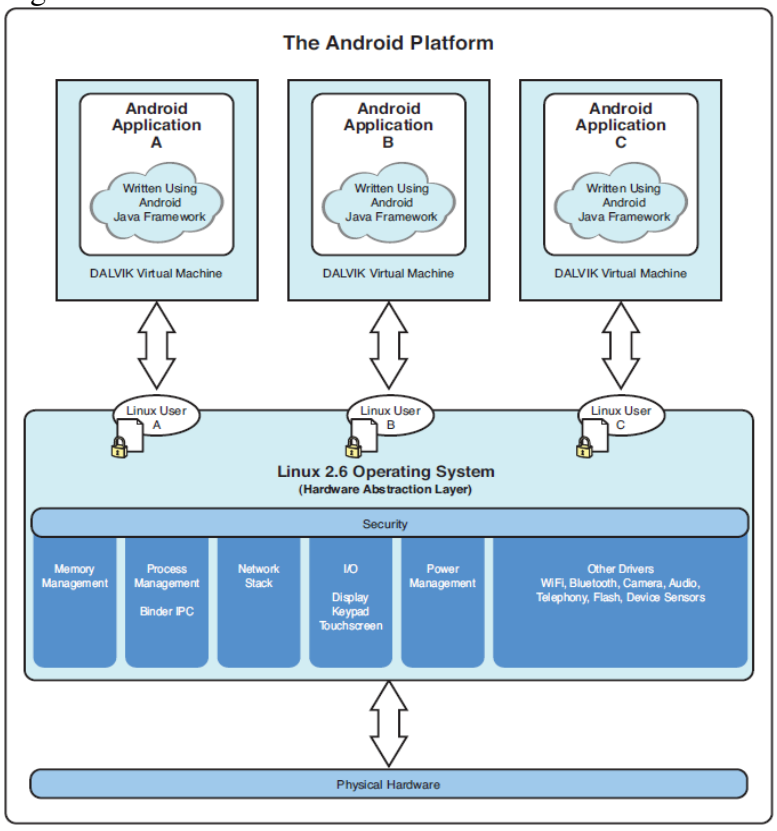
Java, sendo que uma ferramenta inclusa no SDK do Android converte os arquivos de classe Java comuns por arquivos em outro formato de classe, arquivos no formato “.dex” (DALVIKVM.COM, 2011, tradução nossa).

2.2.5 Linux Kernel

O Android foi criado sobre o código-fonte aberto do *kernel* 2.6 do Linux, confiando sobre esses pontos centrais do SO. Entretanto o Linux não fica embarcado, assim impossibilitando a utilização de utilitários como o X-windows e as bibliotecas GNU C (GOOGLE, 2012, tradução nossa; STEELE; TO, 2012, tradução nossa).

Segundo entrevista de Vaughan-nichols (2011) com Linus Torvalds, criador do Linux, existe uma grande distancia entre o Android e o Linux, principalmente pelo fato que parte do código do Linux existente dentro do Android, foi reescrita pela sua própria equipe de desenvolvimento, fugindo assim da vertente principal do sistema operacional.

Figura 4 – Android usando o *kernel* 2.6 do Linux



Fonte: CONDER; DARCEY (2010)

Conforme é possível verificar na Figura 4, o *kernel* atua como uma camada de abstração entre o hardware e o resto da pilha de software, cabendo ao mesmo realizar o gerenciamento de memória, gestão de processos, segurança, pilha de rede e *drivers*. (FELKER; DOBBS, 2011, tradução nossa; GOOGLE, 2012, tradução nossa)

Na figura 4 é possível verificar o processo realizado pela plataforma a cada requisição. Independentemente da maneira que a chamada for realizada pelo usuário, imediatamente esta chega ao *kernel* do Linux, que trata de criar uma DVM criada especialmente para executar o processamento (CONDER; DARCEY, 2010, tradução nossa).

2.3 SDK ANDROID

Ao realizar o *download* da SDK do Android diretamente em seu site, essa vem com as bibliotecas, depurador, documentação, código de exemplo, tutoriais e um emulador de dispositivos móveis. Entretanto para realizar a instalação da SDK do Android, é necessário já possuir instalada a JDK (CONDER; DARCEY, 2010, tradução nossa; GOOGLE, 2012, tradução nossa).

O kit de desenvolvimento pode ser classificado em dois grupos de ferramentas: ferramentas do SDK e ferramentas da plataforma. As ferramentas da plataforma são as ferramentas que dependem da versão do Android, conforme foram apresentadas na Tabela 1. Já as ferramentas do SDK são instaladas junto com o pacote principal e atualizadas posteriormente, não dependendo da versão escolhida para desenvolvimento (GOOGLE, 2012, tradução nossa).

As principais ferramentas já inclusas no kit de desenvolvimento são:

a) Android *Virtual Device* (AVD): simula diversas configurações de hardware de dispositivos móveis sem a necessidade de possuir cada um destes dispositivos;
b) Dalvik *Debug Server Monitor* (DDMS): depura aplicativos desenvolvidos;
c) Android *Debug Bridge* (ADB): permite executar comandos *shell* diretamente em um dispositivos ou então instalar uma aplicação manualmente;
d) Android *Hierarchy Viewer*: possibilita aos desenvolvedores verificar os relacionamentos entre componentes de layout, depurando e otimizando as telas do sistema.

Ainda existem diversas outras ferramentas que o SDK prove para gráficos, tamanhos de imagens entre outros (CONDER; DARCEY, 2010, tradução nossa; GOOGLE, 2011, tradução nossa).

2.4 ARMAZENAMENTO DE DADOS

Para gerenciar um grande número de informações como contatos, eventos, tarefas e outras informações geradas pelas mais diversas aplicações, os dispositivos móveis devem possuir maneiras de salvar essas informações, o que normalmente é uma tarefa custosa tanto ao nível de memória física quanto a processamento do dispositivo móvel (MEDNIEKS et al, 2011, tradução nossa).

Geralmente dispositivos móveis sofrem por possuir poucos recursos computacionais em comparação com desktops, servidores e até mesmo laptops, principalmente quando se fala em memória física e processamento. Além disto, existem restrições quanto ao suporte do tipo de aplicação pelo sistema operacional do dispositivo móvel (JOHNSON; JOHNSON, 2008).

Anteriormente, dispositivos móveis com a tecnologia J2ME, usavam recursos como *Record Management System* (RMS) para armazenamento de registros. Entretanto, esse recurso utilizava uma API de baixo nível, tornando a manipulação dos dados complexa. Para ler ou escrever dados, era necessário serializar e desserializar os dados (JOHNSON; JOHNSON, 2008).

Sistemas operacionais como o Android e iOS percebendo a dificuldade na manipulação de dados, acabaram por adotar o SQLite como o Sistema Gerenciador de Banco de Dados (SGBD) padrão para armazenamento de dados em seus dispositivos. O SQLite foi escolhido como SGBD dos principais sistemas operacionais para dispositivos móveis por apresentar características como alta portabilidade, utilizar pouco espaço em disco, ser eficiente e confiável. O SQLite é um banco de relacional, possuindo assim como linguagem padrão a *Structured Query Language* (SQL) (ALLEN; OWENS, 2010, tradução nossa).

A utilização da linguagem SQL somente é possível, pois uma biblioteca escrita em *C* implementa o banco de dados embutido que lê e escreve diretamente em arquivos do dispositivo. Mesmo o SQLite não fazendo parte do projeto do Android, ele permite sua manipulação por meio da linha de comando utilizando a ferramenta *sqlite3*, que permite a

execução dos comandos SQL e também de comandos próprios da ferramenta (JOHNSON; JOHNSON, 2008).

O SQLite se encontra sobre licença de *Public Domain*, ou seja, qualquer um pode contribuir com o aplicativo sendo que o mesmo pode ser até mesmo comercializado. Diferentemente de outros bancos de dados, o SQLite não possui um processo de servidor, pois ele apenas escreve e lê arquivos. Ainda sim, o SQLite possibilita a criação de *triggers*, índices e *views*, isso tudo dentro de um arquivo somente (SQLITE, 2011, tradução nossa).

Como resultado, dentro do Android é possível todos os bancos de dados no caminho */data/data/<nome-do-pacote>/databases* (JOHNSON; JOHNSON, 2008).

2.5 DISPOSITIVOS MÓVEIS

A partir dos anos 90, foi possível constatar um grande crescimento no desenvolvimento de tecnologias móveis, número de serviços prestados, quantidade de aplicações disponíveis e diversidade de dispositivos. Como retorno a esse crescimento da área, aconteceu também uma grande popularização destes dispositivos, serviços e aplicativos (FIGUEIREDO; NAKAMURA, 2003). Atualmente existem lojas especializadas na venda de aplicativos para os mais diversos dispositivos móveis, assim como empresas que somente trabalham nesta área.

Segundo Mateus e Loureiro (1998), a computação móvel representa um paradigma computacional, sendo considerada como a quarta revolução na computação, precedida pelo surgimento dos centros de processamento de dados nos anos sessenta, terminais nos anos setenta e das redes de computadores na década de oitenta. Esse novo paradigma disponibiliza aos usuários acesso a serviços, aplicativos e recursos independentemente de infra-estrutura, assim permitindo a mobilidade de acesso aos recursos.

Aparelhos com características que possibilitem a troca de informações através da rede e o transporte de maneira fácil pelo usuário podem ser considerados dispositivos móveis. Entretanto para satisfazer estas características é importante que o dispositivo móvel tenha tamanho reduzido, possua bateria de grande duração para não existir a necessidade de permanecer conectado a uma rede elétrica e, por ultimo, possua acesso à tecnologia de comunicação de dados sem fio (FIGUEIREDO; NAKAMURA, 2003).

Normalmente estes dispositivos móveis possuem capacidade de processamento limitada e um pequeno volume. Atualmente estão disponíveis sobre diversas formas, como

pagers, *personal digital assistants* (PDA), *handheld*, telefones celulares, *tablets* entre outros. No Brasil, a oferta e utilização destes dispositivos estão crescendo muito, tanto para utilização no lazer quanto no trabalho (SCHMITT JUNIOR, 2005).

Hoje, o dispositivo móvel com o maior número de usuários é o celular, que teve como origem os telefones fixos. Sua história teve início no ano de 1876, quando Alexandre Graham Bell criou um aparelho capaz de realizar a comunicação de voz entre dois pontos distintos. Neste mesmo ano, Graham Bell e seu assistente, patentearam este aparelho apenas duas horas antes da patente de um segundo inventor. Um pequeno período de tempo tornou Graham Bell como reconhecidamente o inventor do aparelho, tornando seu nome conhecido. Sua invenção que veio a se chamar telefone deu origem a uma série de evoluções, entre elas o próprio telefone celular (COMUNICAÇÕES, 2011).

Em 1947, a empresa Bell Company, desenvolveu um sistema capaz de realizar a comunicação de telefonia móvel dentro de uma determinada área de cobertura. Após 21 anos do desenvolvimento do sistema, as empresas AT&T e Bell, definiram um sistema de torres de transmissão de sinal, para que fosse possível encontrar sinais em diferentes locais, e somente em 1973 é que este sistema passou a ser utilizado em carros de policia (ABREU, 2004).

O primeiro telefone celular, chamado de DynaTAC 8000X, teve seu protótipo criado em 1973, pela Motorola, mas tendo sua primeira versão comercial desenvolvida em setembro de 1983. O celular, conforme é possível verificar na Figura 5 pesava cerca de 1 kg, possuindo memória para 30 números com um monitor de LED (*Light Emitting Diodes*) (ABREU, 2004).

Figura 5 – Motorola DynaTAC 8000X

Fonte: MOTOROLAMOBILITY (2011)

Uma das comparações feitas leva em conta o número de anos que as tecnologias levaram para possuir o numero de um milhão de usuários. Enquanto a televisão preta e branca levou algo em torno de vinte anos e os computadores pessoais levaram seis anos, os dispositivos móveis levaram apenas um ano para obterem o numero total de um milhão de usuários (MATEUS; LOUREIRO, 1998).

Segundo a *International Telecommunication Union* (ITU), em 2010, o mundo já possuía aproximadamente 5.2 bilhões de usuários de telefones celulares, algo em torno de 76 celulares para cada 100 habitantes. Países desenvolvidos possuem um número ligeiramente maior, são 116 celulares para cada 100 habitantes (ITU, 2010).

Atualmente uma grande variedade de dispositivos móveis são ofertados aos usuários, cada um com sua aplicação. Entre *pagers*, telefones celulares, *Personal Digital Assistant* (PDA), *tablets* e *smartphones*, atualmente os que mais se destacam são os *tablets* e *smartphones*, sejam pela variedade de utilizações para os mesmo, ou por englobarem outros dispositivos dentro dos mesmos.

2.5.1 Tablets

Tablets são dispositivos móveis com tamanho maior que *smartphones* possuindo uma tela sensível ao toque (*touch screen*) e que disponibilizam funcionalidades semelhantes a um computador pessoal. Atualmente diversos *tablets* estão disponíveis no mercado, dentre os quais se destaca o iPad. O iPad, *tablet* da Apple abriu as portas para esse novo conceito de dispositivo móvel, fazendo com que os demais concorrentes tivessem de criar dispositivos aos mondes do iPad (GRUMAN, 2011).

Embora a grande atenção para os *tablets* seja recente, esses dispositivos já estão disponíveis no mercado há algum tempo, com um conceito e funcionalidades um pouco diferentes do que vemos hoje no iPad. O GRiDpad Pen Computer pode ser considerado o primeiro *tablet* a chegar ao mercado. Lançado em 1989, pesava pouco mais de 2 kg com um processador 386 de 20 MHz, possuía uma versão do MS-DOS, com uma interface acessível por meio de uma caneta.

Atualmente como os *tablets* proporcionam um processamento rápido como um computador pessoal e a mobilidade de um dispositivo de mão, é possível identificar alguns nichos de mercados para o *tablet*. Grande parte desse nicho, antes ocupado por dispositivos como os Palms, vem sendo substituído por *tablets*.

O mercado de aplicativos para medicina é um exemplo. Uma grande vantagem dos aplicativos para *tablets* sobre os aplicativos *web* e *desktop*, é que estes aplicativos é que se encontram a disposição dos médicos em suas mãos onde for necessário, seja no consultório ou na mesa de cirurgia, sem a necessidade de conexão à internet e com uma grande mobilidade e facilidade de utilização. Grande parte dos aplicativos que existem hoje para dispositivos moves são para iPad (RAMOS; MEIRELLES, 2011).

2.5.2 Smartphones

Um s*martphone* é um celular com funções de PDA. Atualmente os *smartphones* possuem aplicativos nativos para acesso a internet, envio de mensagens, reprodução de musicas, bater fotos e gravar vídeos. Muitos dos dispositivos atualmente permitem aos desenvolvedores que criem seus próprios aplicativos e disponibilizem esses para todos os demais usuários (PC MAGAZINE, 2011).

O primeiro dispositivo a obter o conceito de *smartphone* foi desenvolvido em uma parceria entre a International Business Machines (IBM) e a BellSouth, em 1994, e se chamou Simon. Alguns anos depois, o primeiro *smartphone* com um número considerável de usuários foi produzido pela BlackBerry (PC MAGAZINE, 2011). Porém, foi em 2007 que aconteceu o maior acontecimento no mundo dos *smartphones* e dispositivos móveis. Com o surgimento do iPhone da Apple, foram criados além de um telefone celular, conceitos que estão disponíveis até hoje nos diversos aparelhos existentes como não existência de teclado, touch screen e multi touch e uma interface toda baseada em cliques do usuário com os dedos. Esses conceitos são aplicados até hoje em outros aparelhos, e dificilmente vemos um *smartphone* sem essas funcionalidades (MORIMOTO, 2011).

3 SINCRONIZAÇÃO DE SISTEMAS

A necessidade de integração e sincronização dos Sistemas de Informação (SI) está ligada a evolução das organizações, dos mercados e tecnologia. Atualmente é necessário compartilhar a informação existente nos SI para a Internet, bem como compartilhar informações entre sistemas diversos e iguais para geração de soluções e resultados (MARTINS, 2005).

O aparecimento da Internet criou uma necessidade maior de integrar informações, fazendo os gestores de Tecnologia da Informação (TI) optassem por estratégias e soluções globais para suprir esta necessidade. Tecnologias como *Service Oriented Architecture* (SOA) e *Web Services* (WS) apareceram como soluções para suportar a integração e sincronização dos SI, sendo válidas conforme os objetivos que a organização ou sistema propõem (MARTINS, 2005).

Os WS utilizam os protocolos HTTP e HTTPS, já conhecidos na web. Os serviços trafegam informações do cliente para o servidor utilizando documentos linguagem de marcação extensiva (XML), ou eXtensible Markup Language que é uma linguagem de descrição caracterizada por tags. A principal vantagem da utilização da linguagem XML é que SI podem se comunicar utilizando diferentes tecnologias (CERAMI, 2002, tradução nossa).

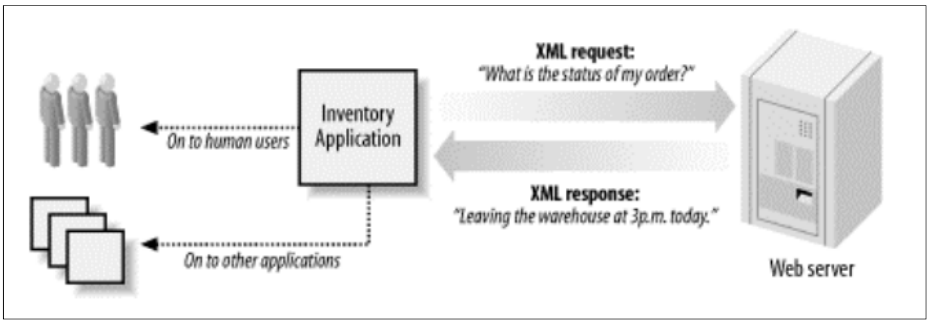
3.1 WEB SERVICES

A tecnologia dos WS é uma importante forma de integração de dados entre diferentes plataformas e programas. Por isso, um WS possui como características: estar disponível através da Internet ou uma rede, utilizar um sistema de troca de mensagem através de XML, não ser dependente de sistemas operacionais ou linguagens de programação, ser auto-descritivo através da linguagem XML e ser encontrado via mecanismos de busca (CERAMI, 2002, tradução nossa).

Serviços web promovem a troca de informações permitindo a publicação de rotinas e métodos acessíveis pela Internet. Por meio de uma interface transparente para o usuário é possível realizar a integração de dados entre aplicações distintas. Aplicações que utilizam a verificação de cartão de crédito, rastreamento de pacotes, acompanhamento de carteira, conversão de moeda e tradução de idiomas entre outros são exemplos de trocas de informações utilizando os serviços web (CERAMI, 2002, tradução nossa). Na Figura 6 é

realizada uma verificação da situação de compra em uma aplicação que busca informações em um WS através de XML.

Figura 6 – Exemplo de um aplicativo requisitando o Serviço Web

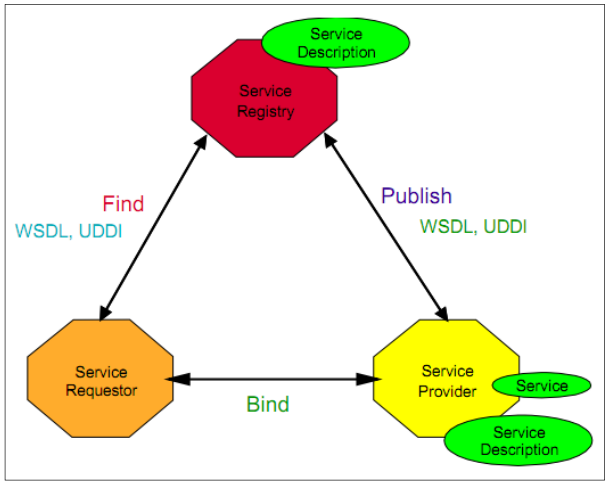


Fonte: CERAMI (2002)

Conforme é possível verificar na Figura 7, a arquitetura de um WS é baseada na interação entre três papéis: provedor de serviços, registrador de serviço e o solicitante do serviço. Os componentes do WS podem desempenhar os papeis descritos abaixo:

a) provedor do serviço: é o dono do serviço, publica e disponibiliza o acesso ao serviço, e por meio dele é possível obter o serviço;
b) corretor, o registrador do serviço: local onde se publica o serviço, sendo que por meio desta os serviços são anunciados como se estivessem em um catálogo;
c) consumidor, solicitante: pesquisa por um serviço no corretor, e caso encontrar o serviço vai ao provedor para consumi-lo (ABINADER; LINS, 2006).

Figura 7 – Componentes e operações de um *Web Service*



Fonte: KREGE (2001)

Segundo Krege (2001), os componentes executam ações de publicação (*publish*), localização (*find*) e vinculação (*bind*), conforme é possível verificar na Figura 7. Abaixo é possível verificar o comportamento realizado nestas operações:

a) publicar: torna o serviço acessível para que o serviço solicitante pode encontrá-lo;
b) localizar: o solicitante busca o serviço e recupera uma descrição de serviço diretamente ou consulta o registro de serviço para o tipo de serviço exigido;
c) vinculação: o serviço solicitante invoca ou inicia uma interação com o WS usando detalhes da vinculação para localizar, contatar e invocar o serviço.

Os WS são construídos sobre diversas especificações de tecnologias e padrões. Três componentes são considerados núcleos da arquitetura típica de um *Web Service* que utiliza o XML como linguagem base. Os componentes, *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) e *Universal Discovery, Description, and Integration* (UDDI), também referenciados como WUST (WSDL, UDDI, SOAP Techonologies) formam o núcleo da tecnologia (ABINADER; LINS, 2006; POTTS; KOPACK, 2003).

O XML foi adotado em fevereiro de 1998 pelo *Worldwide Web Consortium* (W3C) como linguagem padrão para formatação de dados na Internet por conter uma linguagem neutra e conseguir armazenar com a simplicidade de um documento dados altamente complexos (ABINADER; LINS, 2006). A linguagem é baseada em *tags* que designam o que significam os dados apresentados. Um texto simples como 7 3 2003 não significa nada, mas se endentados corretamente dentro das *tags* apresentam uma data (POTTS; KOPACK, 2003).

Figura 8 – Linguagem XML

```
<data>
    <dia>3</dia>
    <ano>2003</ano>
    <mes>7</mes>
</data>
```

Um dos componentes padrões de WS, o SOAP é uma especificação que define as regras para encapsular os dados que vão ser transferidos entre computadores, como dados específicos dos aplicativos como nomes e parâmetros de métodos, valores de retorno. Uma

mensagem SOAP é um documento XML para realizar a troca de informações entre computadores (POTTS; KOPACK, 2003).

O SOAP atua sobre o HTTP, proporcionando assim simplicidade. A definição do protocolo foca em um modo formal para que mensagens XML possam ser trocadas entre emissor e receptor. O emissor envia uma solicitação com uma mensagem SOAP, o servidor processa a solicitação e responde com outra mensagem SOAP. No servidor, o processamento do XML com a mensagem SOAP é realizado por um *parser* de XML que analisa o documento verificando se o mesmo está conforme as regras gramaticais e o vocabulário. Se a verificação obtiver sucesso, o processador SOAP chama o WS descrito no documento SOAP e passa os parâmetros que esse possua (ABINADER; LINS 2009).

Segundo Potts e Kopack (2003), o WSDL, tem como principal função permitir que o cliente interessado em utilizar o *Web Service* consiga realizar o mesmo sem a necessidade de entrar em contato com o autor do WS. Segundo Abinader e Lins (2009) qualquer programa que queria utilizar um WS, usa o WSDL para se vincular ao mesmo. Os autores de WS criam uma WSDL referente ao serviço disponibilizado, e em um serviço de diretório, publicam o mesmo por meio de uma URL.

Segundo Potts e Kopack (2003), um documento WSDL possui elementos que são básicos em sua estrutura interna. Esses elementos podem ser divididos de forma lógica em elementos concretos e abstratos. Os elementos concretos são orientados para vincular fisicamente os clientes ao serviço. Já os elementos abstratos são orientados para descrever as capacidades dos WS.

Os elementos XML abstratos que podem ser definidos em um WSDL são:

a) <wsdl:types> permite definir os tipos de dados utilizados na troca de informações;
b) <wsdl:message> define abstratamente um dado, antes da transmissão deste pelo WS;
c) <wsdl:operation> análogo a uma chamada de método Java, mas somente permite três mensagens (entrada, saída e falha);
d) <wsdl:portType> contém a definição das operações disponíveis em um WS, apresentando diversas mensagens de entrada e saída (CERAMI, 2002; POTTS; KOPACK, 2003).

Os elementos concretos em um WSDL são:

a) <wsdl:service> possui definições para agregar endereços ao invocar um serviço específico;

b) <wsdl:port> definir o endereço para uma porta, permitindo que um serviço possa ser acessado por SOAP e HTTP;
c) <wsdl:binding> possui a estrutura do canal de comunicação com o cliente, assim como os protocolos e os formatos dos dados utilizados em mensagens e operações (CERAMI, 2002).

Ainda segundo Potts e Kopack (2003), é possível encontrar mensagens SOAP e definições XML no WSDL.

O ultimo componente de um WS, o UDDI, é uma especificação para descrever, descobrir e integrar WS entre as diferentes entidades. O UDDI pode ser compreendido como mecanismo que possibilita as organizações encontrarem outras organizações, conduzindo transações de forma rápida (ABINADER; LINS 2009).

Um registro UDDI é muito parecido com os mecanismos de busca como Google, AltaVista entre outros. Ainda sim, podemos compará-lo com uma lista telefônica, que disponibiliza os potenciais consumidores dos serviços fornecidos pelas entidades. Como uma lista telefônica o registros UDDI possuem divisões, que atualmente são chamadas de páginas brancas, amarelas e verdes. As páginas brancas apresentam informações como o contato das entidades, nas páginas amarelas é possível encontrar classificações do serviço segundo diversas taxonomias e por ultimo as páginas verdes, que contêm os detalhes como é possível desenvolver um cliente para solicitar o serviço (ABINADER; LINS 2009).

4 INFORMÁTICA EM SAÚDE

Ao tomar uma decisão qualquer como definir um receituário, existem situações onde os médicos sentem a necessidade de trocar informações com outros colegas através da leitura de artigos, manuais, livros ou textos. Atualmente a informática médica prove os instrumentos para realizar a troca de informações entre colegas além de auxiliar na organização da consulta médica, na captura de informações do paciente, no receituário médico e no diagnóstico, provendo assim outra forma de busca de informações, tornando a decisão do profissional da saúde mais precisa (WECHSLER et al, 2003).

A área da informática em saúde é definida como um campo de rápido desenvolvimento científico atuando principalmente no armazenamento, recuperação e uso da informação, dados e conhecimentos biomédicos para a resolução de problemas e tomadas de decisões. Assuntos sobre informática médica estão se tornando mais importantes proporcionando a modernização e melhoria da prestação de cuidados de saúde, através de uma melhor gestão da informação de saúde como dos recursos associados (BLOIS; SHORTLIFFE, 1990, tradução nossa; ROCHA, 2011).

Dados e conhecimentos gerenciados por meio de métodos tradicionais como o papel se tornam obsoletos principalmente devido ao grande número de informações médicas que devem ser processadas para se chegar a uma decisão. Esse é considerado o principal motivo do crescimento da área da informática em saúde, que aliada aos avanços nas tecnologias da computação, comunicação e conhecimento médico fazem com que a Informática Médica desempenhe papel central na medicina moderna. Como retorno a esse crescimento ouve uma divisão dos sistemas de saúde em diversas áreas como: Sistemas de informação em saúde, Prontuário Eletrônico do Paciente (PEP), Sistemas de Apoio à Decisão (SAD), Processamento de Sinais Biológicos, Processamento de imagens médicas, Padronização da informação em saúde, Telemedicina entre outras (SADZINSKI, 2010; ROCHA, 2011; SBIS, 2011).

Atualmente uma das áreas da informática em saúde que mais se destaca é a Telemedicina, impulsionada pelas tecnologias que permitem a comunicação entre computadores, *smartphones*, *tablets* entre outros dispositivos. Essas tecnologias aliadas a Telemedicina possuem grande potencial para solucionar alguns dos problemas de saúde mundial (WHO, 2010, tradução nossa).

4.1 TELEMEDICINA

Segundo a WHO (2010), o termo Telemedicina, literalmente significando medicina a distância, possui mais de 104 definições no mundo. A Word Health Orgnization (WHO) define a Telemedicina como a disposição de serviços de saúde, à distância, por profissionais de saúde usando tecnologias e a comunicação para o intercâmbio de informações para diagnóstico, tratamento e prevenção à saúde da população.

Existem diversas aplicações possíveis dentro da área de Telemedicina, e em sua maioria podem ser dividas em dois tipos básicos segundo as pessoas envolvidas na transmissão, aplicações que conectam profissionais a profissionais ou profissionais a pacientes, e as segundo seu tempo de transmissão, seja elas síncronas ou assíncronas. Quando o paciente envia um e-mail ao profissional de saúde com algumas informações sobre seu caso médico, e o profissional de saúde retorna o e-mail com o diagnóstico ou opinião, possui um tipo de transmissão assíncrona entre paciente e profissional de saúde. Em outra situação onde profissionais em diferentes locais acompanham uma mesma cirurgia por meio de uma videoconferência e trocam informações, se entende como um tipo de transmissão síncrona de profissional de saúde para profissional de saúde (WHO, 2010, tradução nossa).

Em países onde a existem deficiências no cuidado a saúde, a Telemedicina pode ajudar a atender as necessidades não supridas por outros métodos tradicionais de cuidados a saúde de pacientes. Locais com difícil acesso é possível obter o atendimento de um médico especialista através de uma videoconferência, evitando a necessidade do deslocamento do profissional de saúde a áreas remotas, onde o custo de deslocamento é alto. Outra grande vantagem da utilização da Telemedicina é o acompanhamento de pacientes com condições crônicas, como a diabetes. Em situações de tratamento de enfermos com doenças crônicas o profissional de saúde pode acompanhar a evolução do tratamento diariamente, sem a necessidade de estar presencialmente com o paciente. Sem a necessidade de um acompanhamento presencial do profissional de saúde, é possível aumentar o controle no tratamento a enfermidade com uma redução significativa de consultas (WHO, 2010, tradução nossa).

Hoje existe um debate político sobre como a tecnologia poderia auxiliar países em desenvolvimento como solução de parte dos problemas de saúde para os mais diferentes setores da população. Uma das formas seria a utilização da telefonia móvel para criar uma interação entre pacientes e profissionais de saúde durante o tratamento. Porém são poucos os estudos sobre o tratamento de doenças crônicas não-transmissíveis como doenças

cardiovasculares, diabetes, depressão e para doenças crônicas transmissíveis como vírus da imunodeficiência humana (HIV) e tuberculose utilizando a tecnologia móvel no auxílio para tratamento dessas enfermidades (KAPLAN, 2006, tradução nossa).

As principais dificuldades na utilização dos dispositivos móveis no tratamento se referem aos valores pagos pelos aparelhos, valores pagos pelas ligações, limitações de acesso a rede, dificuldade de manuseio dos dispositivos móveis e analfabetismo da população que impossibilita a troca de informações com um médico (KAPLAN, 2006, tradução nossa).

Ainda sim, com a crescente evolução da tecnologia dos dispositivos móveis, que tornam os aparelhos mais acessíveis e populares, já é possível disponibilizar alguns serviços de Telemedicina através de dispositivos móveis para acompanhamento de diferentes tratamentos em saúde.

4.1.1 Mobilidade e Saúde

Dispositivos móveis já são utilizados como apoio no tratamento e prevenção e saúde de diversos pacientes, mas em sua maioria operados por profissionais de saúde e não pelos pacientes, desperdiçando um grande potencial destes dispositivos, que seria a atenção a saúde de pacientes usando a Telemedicina. Aplicativos que disponibilizam calculadoras desenvolvidas especialmente para medição de dosagem de medicamentos ou então aplicativos que possuem pesquisa de Código Internacional de Doenças (CID) possuem grande aceitação pelos médicos (CALIXTO, 2011).

Um exemplo de sucesso na utilização de sistemas para dispositivos móveis por profissionais da saúde é o Projeto Borboleta. O projeto foi iniciado em 2004 com objetivo de prover ferramentas e metodologias para realização de atendimentos domiciliares. O sistema é dividido em dois módulos distintos, o módulo servidor e o módulo para dispositivos móveis. Dentro do módulo servidor são armazenas as informações de todos os atendimentos domiciliares realizados pelos profissionais em saúde, enquanto dentro do módulo móvel, que é executado a parir de *smartphones*, PDAs e computadores de mão, estão armazenadas as informações sobre os pacientes que irão ser atendidos nas visitas realizadas. Também são armazenadas nestes dispositivos, as informações referentes aos últimos atendimentos do profissional. Após voltar para a central de saúde, as informações recolhidas pelos profissionais durantes os atendimentos domiciliares são sincronizadas com o módulo servidor do sistema, assim todo médico que for realizar um novo atendimento pode consultar

informações sobre seu paciente antes de iniciar o atendimento (PRONTUÁRIO NO CELULAR, 2011).

Empresas de desenvolvimento voltado para softwares para saúde já estão investindo pesado em mobilidade. A MV Sistemas, uma das empresas líderes de mercado para aplicativos de saúde no Brasil, está realizando o investimento de 2 milhões no desenvolvimento de aplicativos para dispositivos móveis para profissionais de saúde. O foco dos aplicativos que estão sendo desenvolvidos em parceria com as universidades de Passo Fundo (RS) e Federal de Pernambuco são à administração de medicamentos para pacientes e prontuário eletrônico para dispositivos móveis. A empresa prevê que os aplicativos vão dar mais mobilidade para os profissionais durantes os atendimentos dos pacientes, bem como vão ajudar no momento de realizar a dispensa de medicamentos, sendo uma fonte segura para realizar o tratamento dos pacientes (SOUZA, 2011).

Segundo Calixto (2011) ainda que a maior parte dos aplicativos para dispositivos móveis sejam voltados para os prestadores de saúde, existem aplicativos que voltados especialmente para pacientes. O aplicativo *Medicamentos de A a Z* para iPhone é um exemplo de aplicativo desenvolvimento essencialmente para pacientes. No aplicativo os pacientes podem consultar a lista de medicamentos, bem como ver suas indicações, contra indicações entre demais informações.

O laboratório Bayer disponibilizou um aplicativo voltado para seus pacientes do sexo feminino. O aplicativo batizado de Hora da Pílula desenvolvido para iPhone, iPod e iPad lembra os usuários que está no momento de tomar o medicamento anticoncepcional. Outros aplicativos para iPhone como o *RxmindMe* lembram os usuários de todos medicamentos que devem ser tomados diariamente. Entretanto, um grande potencial destes aplicativos é desperdiçando quando ele não gera informações para o profissional que prescreveu o tratamento, perdendo informações que poderiam ajudar o prestador de saúde nas tomadas de decisões no tratamento e acompanhamento do tratamento a saúde do paciente.

A maior parte dos aplicativos é voltada para pacientes ou prestadores de saúde, são poucos softwares que permitem uma integração entre informações geradas por pacientes e médicos. Aplicativos que conectem o médico e paciente podem permitir um melhor controle da saúde em tempo real do paciente pelo profissional de saúde. Um tratamento médico prescrito pelo profissional de saúde poderia ser acompanhando em tempo real pelo mesmo se houver um aplicativo que possa sincronizar informações entre o paciente e médico.

Além do controle de medicamentos, a tecnologia móvel com sincronização de informações entre pacientes e prestadores de saúde serviria muito bem para tratamento de

doenças crônicas, onde o médico pode acompanhar indicadores gerados pelo paciente diariamente. Aplicativos como o Diamedic, utilizado no controle de diabetes por pacientes com a doença crônica, poderiam ser sincronizados com dispositivos móveis ou bases de sistemas dos profissionais de saúde, possibilitando assim um acompanhamento diário sobre a evolução da doença do paciente. Desta forma, qualquer alteração das taxas de glicose do paciente seria enviada ao profissional de saúde que poderia alterar o tratamento do paciente apenas retornando um alerta para o paciente (BEDRAN, 2011).

Pesquisas apontam que o acompanhamento médico realizado junto ao paciente utilizando tecnologias móveis como SMS e a internet podem resultar em uma evolução no quadro de pacientes com doenças crônicas como diabetes. O estudo foi realizado em pacientes obesos com diabetes do tipo 2. Os pacientes realizavam seu controle de diabetes semanalmente atualizando dispositivos móveis como celulares ou a internet para atualizar seus níveis de glicoses em um site pessoal. Posteriormente um profissional de saúde acompanhava esses dados e realizava intervenções sobre o tratamento semanalmente, também via celulares e internet. Durante um ano, foi possível comprovar que os níveis de glicoses e outros indicares de diabetes de pacientes que foram acompanhados por meio da internet e dispositivos móveis eram melhores que pacientes cujo tratamento era convencional (KIM; KIM, 2007)

O desenvolvimento de estratégias que possibilitem um melhor acompanhamento da diabetes e demais doenças crônicas é de extrema importância. A Telemedicina utilizada como ferramenta no auxílio no tratamento da diabetes é considerada uma alternativa principalmente para pacientes com dificuldade no acesso aos cuidados de saúde.

4.1.2 Diabetes Mellitus (DM)

Relatos vindos de 1500 a.C. falam sobre uma doença que deixava a urina com gosto doce. Essa doença posteriormente viria a se chamar diabetes, nome de origem grega significando sifão, uma analogia as mangueiras utilizadas para transportar líquidos de um recipiente para outro. A origem do nome se refere ao sintomas clássicos, que são: excessivo consumo de líquidos e aumento do volume urinário (CANCELLIÉRI, 1999).

A nomenclatura Diabetes Mellitus (DM) é utilizada para diferenciar a DM da Diabetes Insipidus. As doenças apesar de compartilharem o nome diabetes, possuem origem e tratamentos diferenciados (CANCELLIÉRI, 1999).

Segundo pesquisas da IDF a diabetes mellitus já atinge no mundo mais de 280 milhões de pessoas com idade entre 20 e 79 anos. Somente no Brasil, a DM atinge mais de 7 milhões de pessoas. Na Tabela 2 é possível verificar os números de prevalência da DM por região na população mundial.

Tabela 2 - Prevalência da Diabetes Mellitus por região na população mundial

Região	População de 20 a 79 anos (Milhões)	Porcentagem com DM	Total
Região Africana	378.550	3,2%	12.089
Oriente Médio e Norte Africano	344.469	7,7%	26.646
Região Européia	646.367	8,6%	55.388
América do Norte e Região do Caribe	319.893	11,7%	37.362
América do Sul e Central	286.922	6,3%	17.958
Região do Sudeste da Ásia	837.732	7,0%	58.662
Região do Pacífico Ocidental	1.530.822	5,0%	76.709
Total	**4.344.755**	**6,6%**	**284.814**

Fonte: adaptado de IDF (2010)

O diabetes é uma disfunção metabólica, originada pelo comprometimento na produção e/ou utilização do hormônio da insulina. O grau de comprometimento de sua produção, de sua ação, do número ou da resposta dos receptores à insulina são indicadores dos dois tipos de diabetes mellitus: diabetes tipo 1 ou mellitus insulino-dependente e o diabetes tipo 2 ou mellitus não insulino-dependente (CANCELLIÉRI, 1999).

Na diabetes mellitus insulino-dependente ou tipo 1, o organismo produz baixa quantidade de insulina ou não chega a produzir insulina, assim obrigando o paciente a aplicação de doses de insulina para se manter saudável. A aplicação de insulina é necessária, pois sem ela a glicose não consegue penetrar em suas células transformando a glicose em energia. Caso essa transformação não aconteça, a glicose passa a se acumular em altas taxas na corrente sanguínea, afetando olhos, rins, nervos e coração (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

A diabetes do tipo 1 pode surgir em pessoas com qualquer idade, entretanto ela é apresentada com mais freqüência em pessoas com menos de 30 anos. Os principais sintomas

da diabetes tipo 1 são: micção freqüente, fome constante, sede constante, perda de peso, fraqueza, fadiga, nervosismo, mudanças de humor, náusea e vomito (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

Não existe um motivo ao certo por que as pessoas possuem diabetes do tipo 1. Normalmente existe uma predisposição à doença, entretanto muitas pessoas que possuem os mesmos genes da predisposição acabam não se tornando diabéticas. Em alguns casos, as pessoas que possuem diabetes desenvolvem uma grande quantidade de auto-anticorpos que circulam na corrente sanguínea, atacando os próprios tecidos do corpo, neste caso, atacando células que produzem insulina, bem como a própria insulina (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

Já na diabetes tipo 2 ou mellitus não insulino-dependente o organismo não produz insulina suficiente ou possui problemas para utilizar a insulina, desta forma sendo necessário que o paciente utilize insulina, mas diferentemente do tipo 1, nesta situação, o paciente não depende da insulina para se manter saudável (CANCELLIÉRI, 1999).

Outra diferença entre os diferentes tipos de diabetes é a idade de manifestação da doença. A diabetes mellitus não insulino-dependente costuma ser comum em pessoas com mais de 40 anos, possuindo como principais sintomas micção freqüente, sede constante, fome constante, perda de peso, pele ressecada ou com pruridos, visão embaçada, formigamento ou adormecimento dos pés e mãos, fadiga, fraqueza e infecções recorrentes (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

Normalmente, essa manifestação da diabetes costuma ocorrer em indivíduos que já possuem um histórico de doença pregressa do tipo 2. Assim, outros fatores influenciam no desenvolvimento da doença como ingestão de gorduras, alimentação não baseada em carboidratos e fibras e a falta de atividades físicas (AMERICAN DIABETES ASSOCIATION, 1998; CANCELLIÉRI, 1999).

4.1.2.1 Tratamento a Diabetes Mellitus

Para a diabetes do tipo 1, alguns procedimentos são recomendados no cuidado a saúde, destacando-se a necessidade de tomar insulina regulamente, possuir uma boa dieta alimentar, praticar atividades físicas, realizar auto-testes de exame de sangue e por ultimo manter e realizar exames médicos regularmente (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

Já o tratamento a diabetes do tipo 2 é semelhante ao aplicado a pacientes que possuem o outro tipo da doença, entretanto, em alguns casos, a insulina é dispensada, pois através de outros medicamentos é possível baixar a taxa de glicose do corpo. Dietas e exercícios físicos ajudam os medicamentos no tratamento, evitando assim ao máximo a necessidade da ingestão de insulina (AMERICAN DIABETES ASSOCIATION, 1998, tradução nossa).

Independentemente do tipo de diabetes, os cuidados médicos devem ser seguidos a risca com um controle diário. Para os pacientes que realizam a ingestão da insulina, o controle é maior ainda, sendo que existe uma necessidade de controlar diversos indicadores, como o peso, nível de glicose no sangue entre outros. Esses indicadores devem ter um acompanhamento de profissionais da saúde para que o tratamento seja monitorado constantemente.

5 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos relacionados que serviram de base para este projeto ou que possuem conteúdos semelhantes, com o objetivo de apresentar melhor a proposta e os conteúdos isolados que formam a mesma.

5.1 DESENVOLVIMENTO DE UM APLICATIVO PARA IPHONE E IPAD PARA ACESSO A INFORMAÇÕES MÉDICAS EM UM HOSPITAL PERVASIVO NO ÂMBITO DO PROJETO CLINICSPACES

Com o crescimento da necessidade da utilização de tecnologias no tratamento, prevenção e recuperação a saúde, aliado a uma popularidade de dispositivos da Apple, foi desenvolvido um aplicativo na área de saúde que sirva de modelo para uma futura análise na criação de interfaces de usuário para dispositivos móveis.

O software desenvolvido em objective-C, linguagem de desenvolvimento para iOS, onde o modelo proposto servirá de modelo para o projeto ClinicSpaces.

A solução apresentada permite o acesso a informações referentes a pacientes cadastrados em um sistema médico. Nela é possível realizar a edição e exclusão de compromissos de um médico em sua agenda por meio de iPhone e iPad. As informações das agendas médicas e dos pacientes são oriundas dos *Web Services* do Google Calendar e Google Health respectivamente (GUERRA, 2009).

5.2 M-COMMERCE E A PLATAFORMA GOOGLE ANDROID

A pesquisa apresenta um protótipo de sistema de comércio móvel (*m-commerce*) utilizando a plataforma Google Android e serviços de intercâmbio de dados (*Web Services*) para permitir a compra de produtos através de dispositivos móveis. Os principais objetivos e o incentivo do desenvolvimento foram a utilização de dispositivos móveis contendo software livre para utilizações diversas, além de apresentar uma tecnologia e um ambiente relativamente novo.

Para implementação do protótipo, foram utilizadas as tecnologias JAX-WS e MySQL para desenvolvimento do serviço que ficaria disponível para os dispositivos móveis. Por sua vez os dispositivos móveis utilizavam Java e XML, padrões para desenvolvimento de

aplicações Android, para se conectarem ao serviço. Algumas bibliotecas como a Ksoap2 também foram usadas na parte móbile do software.

Foram desenvolvidas duas aplicações de uso distinto, sendo um móvel e outra contendo o *Web Services*. Desta forma, foi possível alcançar o objetivo final que era realizar a implementação de um modelo de e-commerce para utilização em celulares usando a tecnologia Android (ANTONELI, 2011).

5.3 SISTEMA DE APOIO AO DIAGNÓSTICO MÉDICO UTILIZANDO TECNOLOGIAS MÓVEIS

Atualmente um dos problemas em municípios do interior do Brasil é a falta de médicos nos postos de saúde e hospitais da rede pública. Nessas situações, o enfermeiro fica sendo o único responsável pelo paciente em grande parte dos casos.

Visando solucionar esse problema, foi possível desenvolver um sistema baseando em tecnologias de telefonia móvel como meio para troca de dados entre os enfermeiros e médicos. Desta forma, se utilizando da Telemedicina, paciente e médico não necessitariam estar fisicamente no mesmo local para se consolidar um atendimento.

A tecnologia celular utilizada foi o Java 2 Micro Edition (J2ME) para os celulares, onde o enfermeiro disponibilizava os sintomas do paciente em um *Web Service*. O médico por sua vez baixava essas informações e formulava o diagnóstico, que posteriormente poderia ser consultado pelo enfermeiro.

Com essa tecnologia, foi criado um canal de comunicação entre profissionais médicos e enfermeiros de diferentes localidades seguindo um fluxo para que seja possível um diagnóstico mais preciso (REHM, 2005).

6 DIABETES CONTROL – SISTEMA DE CONTROLE DE DIABETES

A presente pesquisa tem como resultado um sistema que realiza o controle de vários indicadores sobre a diabetes dos pacientes. Esse controle é realizado por meio de um sistema *mobile* que recebe as informações inseridas pelos usuários, gerando e possibilitando controle sobre os mais diversos indicadores.

Com o sistema desenvolvido também é possível trocar informações entre dispositivos de médicos e pacientes, de forma que as informações geradas pelos pacientes são levadas ao médico que pode acompanhá-las, direcionando um melhor tratamento. Diversas tecnologias e ferramentas foram utilizadas para que o software pudesse alcançar seu propósito, sendo as principais: *Web Services*, Java, JAX-WS, Android e outras citadas durante a metodologia.

O trabalho foi realizado conforme as etapas metodológicas apresentadas a seguir:

6.1 METODOLOGIA

Com a necessidade de obter um referencial teórico, na primeira etapa deste trabalho, foi realizado um levantamento bibliográfico sobre tecnologias móveis aplicadas no tratamento à saúde. Em sua maior parcela, o estudo foi aplicado a tecnologia Android, bem como ferramentas e tecnologias necessárias para o desenvolvimento nesta plataforma. De forma sucinta foram apresentados neste trabalho temas como saúde e mobilidade, telemedicina e os principais conceitos sobre diabetes. Finalmente, com a necessidade de desenvolver uma ferramenta que pudesse realizar a sincronização de informações, foram estudados e descritos conceitos sobre *Web Services*.

Para que fosse possível realizar a criação da solução proposta, tanto no ambiente *mobile* quanto em um serviço da web, tornou-se necessário compreender e definir uma série de tecnologias e ferramentas que iriam auxiliar e firmar o desenvolvimento destas. Agrupando todas as tecnologias utilizadas, podemos citar: Google Android, Google Code, Java, JAX-WS, Eclipse IDE, SQLite, SubEclipse, NetBeans, Oracle, GlassFish e outras que foram usadas pontualmente.

A modelagem do software iniciou considerando o fluxo básico seguindo aplicativo de forma a garantir a troca correta de informações via WS. Dentro desta foram

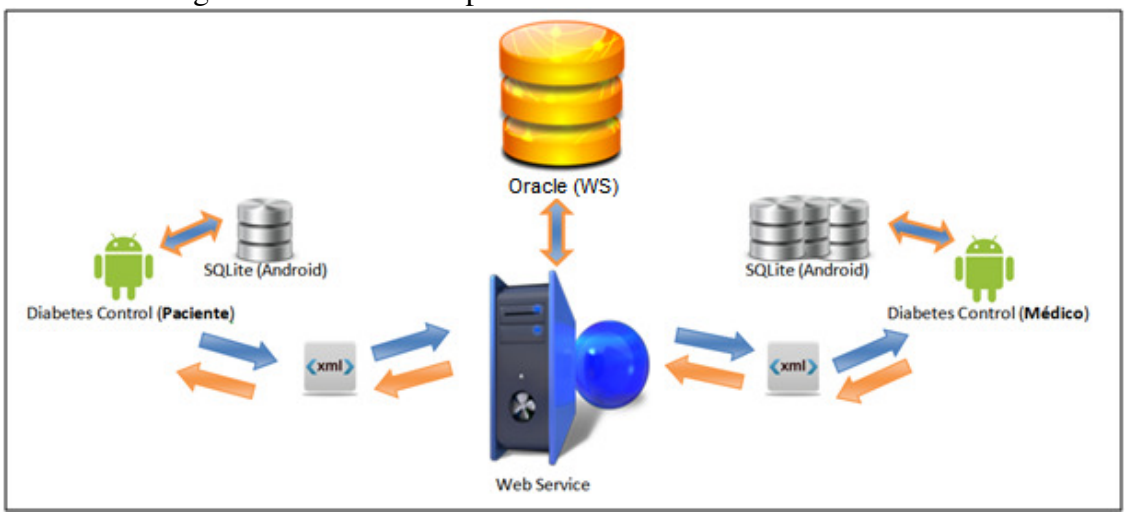
desenvolvidos alguns casos de uso e protótipos de tela, para que fosse possível visualizar um resultado.

Finalmente, após a codificação do aplicativo, foram realizados uma seqüência de testes sobre o aplicativo visando garantir o máximo de qualidade possível para que o mesmo fosse disponibilizado na loja de aplicativos da Google.

6.1.1 Fluxo de Informações

O sistema desenvolvido, nomeado de *Diabetes Control*, realiza uma série de trocas de informações entre dispositivos e serviços conforme exemplificado na Figura 9 em seu modelo conceitual:

Figura 9 – Modelagem Conceitual do Aplicativo



Conforme é possível visualizar na Figura 9, existem duas aplicações distintas. A aplicação Android na qual pacientes e médicos realizam suas marcações e anotações e o *Web Service* que sincroniza essas marcações.

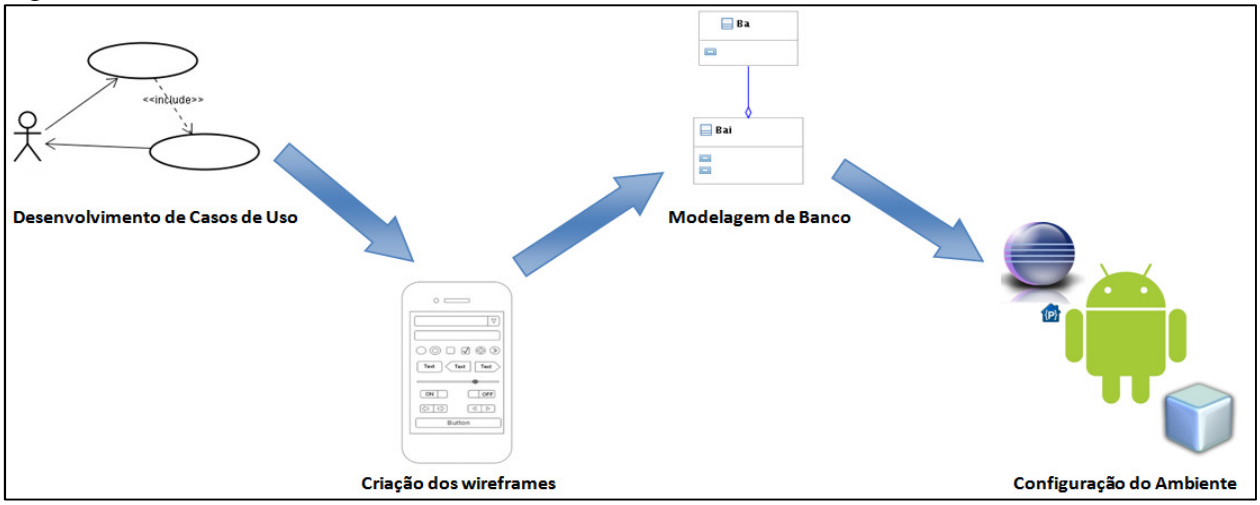
No modelo o WS atua como mediador do envio de informações do dispositivo do paciente para o dispositivo do médico. Essa configuração permite que os dados sejam sincronizados entre pacientes e médicos de forma praticamente instantânea, caso os dois dispositivos estejam conectados a internet. Caso um dos dispositivos não possua acesso a internet, o registro não será perdido e a sincronização acontecerá no próximo momento que o dispositivo esteja conectado a internet.

Os XML trocados possuem informações de peso, insulina, diabetes, notas médicas, dentre outros tipos de informações. Quando essas estão disponíveis nos dispositivos,

é possível realizar consultas, gerar gráficos e acompanhar médias. Tornando mais simples o acompanhamento da diabetes pelo paciente e médico.

Para realizar o desenvolvimento do modelo conceitual apresentado, foram criados diversos artefatos antes da codificação. Na Figura 10 é possível compreender o processo de desenvolvimento desde a criação dos casos de uso até a configuração do ambiente para desenvolvimento do modelo conceitual.

Figura 10 – Processo de desenvolvimento



Primeiramente foi realizada a criação dos cenários de casos de usos para as rotinas do sistema. Com a primeira etapa concluída, foram identificadas as telas e funcionalidades mais utilizadas, que foram desenhadas usando *wireframes*.

A realização da modelagem do banco foi iniciada após a diagramação dos casos de uso e desenho dos *wireframes*. O modelo criado foi dividido conforme a plataforma na qual o banco está disponível. Assim, foi modelado um esquema de banco para o *Web Service* e outro esquema de banco disponível no aplicativo móvel.

Finalmente, foi realizada a configuração do ambiente de desenvolvimento e escolha das bibliotecas e frameworks que fazem parte do projeto desenvolvido.

Cada um destes processos é apresentado detalhadamente na modelagem do software.

6.2 - MODELAGEM

Um aplicativo bem construído possui como um de seus requisitos mínimos a modelagem. Essa modelagem é necessária para que sejam previstos problemas que podem ocorrer durante a implementação, além de guiar o desenvolvimento do sistema e implementar conceitos de engenharia de software, melhorando assim a qualidade do produto desenvolvido.

Existem diversas técnicas de engenharia presentes no mercado para realizar modelagem de software. Para o sistema Diabetes Control, foram escolhidas aquelas que trariam maior valor agregado para o projeto, bem como melhor auxiliariam durante todo o ciclo de desenvolvimento. Desta forma, foi realizada a modelagem lógica das tabelas do banco de dados e a diagramação dos cenários de casos de uso. Alem disso, também foi utilizado o desenho dos *wireframes*, auxiliando assim na simulação do fluxo de navegação entre telas e ações do sistema desenvolvido.

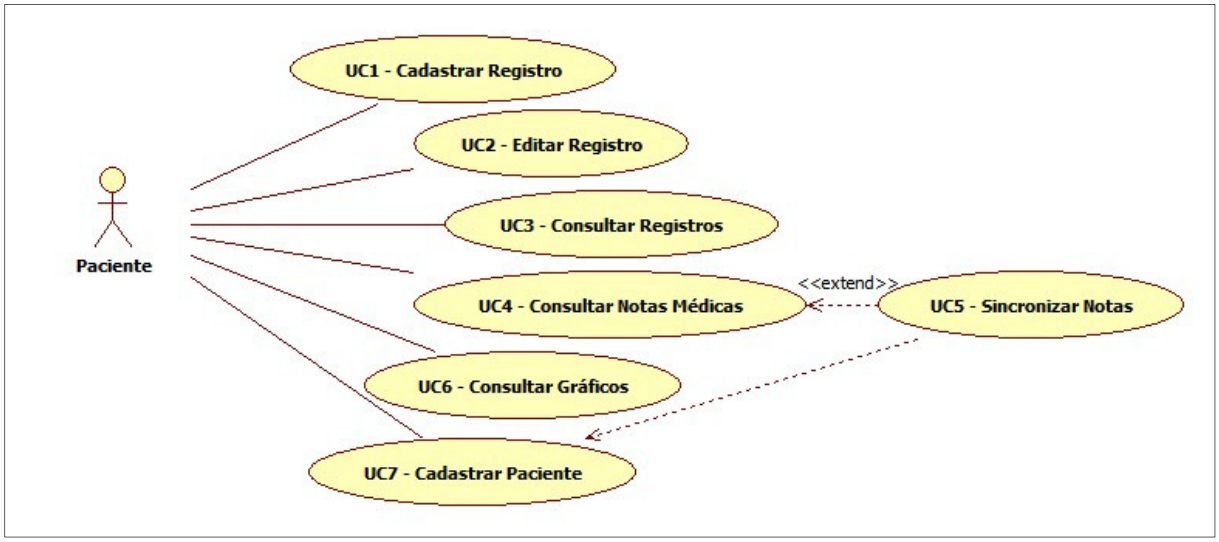
6.2.1 Modelagem de casos de uso

A modelagem de casos de uso possui o propósito, como as demais linguagens de desenvolvimento *Unified Modeling Language* (UML), de documentar, especificar e construir artefatos de forma totalmente visual para auxiliar no desenvolvimento de sistemas.

A diagramação dos casos de uso foi realizada por intermédio da ferramenta StarUML, por ser disponibilizada gratuitamente e possuir recursos avançados para o desenvolvimento dos diagramas necessários.

O primeiro diagrama disponível na Figura 11 mostra as funcionalidades cujo ator, representando o paciente, pode registrar no sistema do paciente.

Figura 11 – Diagrama de Casos de Uso do ator Paciente

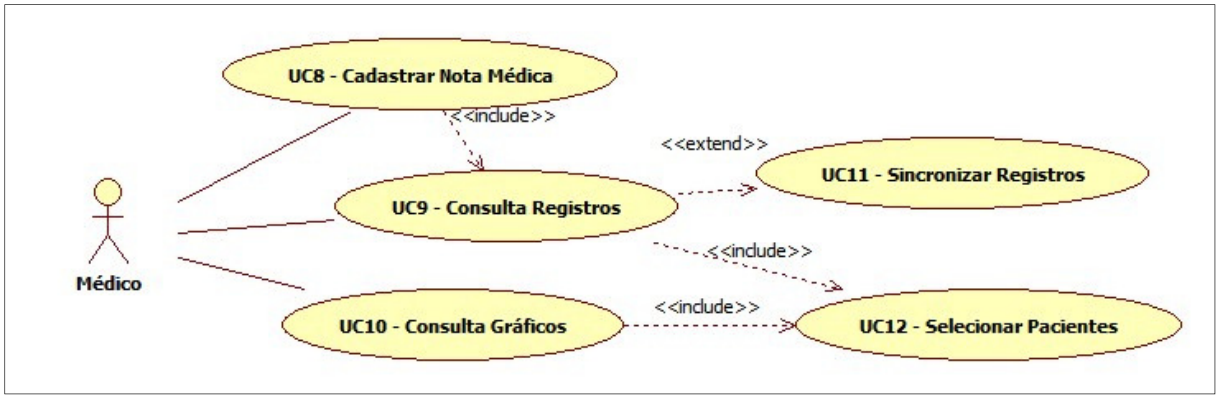


No diagrama, é possível notar que o caso de uso, *Consultar Notas Médicas*, estende o *Sincronizar Notas*, ou seja, ao consultar as notas, pode ser realizada a sincronização das notas com o serviço web. O diagrama apresenta a obrigatoriedade da realização da sincronização das notas, onde, primeiramente é necessário realizar o cadastramento do paciente.

Os casos de uso cadastrar registro, editar registro, consultar registro e consultar gráficos, se referem às informações de glicose, peso, insulina e outros indicadores. Já a informação de notas médicas diz respeito às anotações que o médico faz sobre os registros cadastrados pelo paciente e sincronizados com o dispositivo do médico, como é possível notar na Figura 12.

Nesse diagrama são apresentas as funções que o ator, representando o médico, pode realizar. No diagrama é possível verificar a necessidade de sincronização dos registros cadastrados pelo paciente a serem integrados ao dispositivo para que seja possível cadastrar as notas médicas.

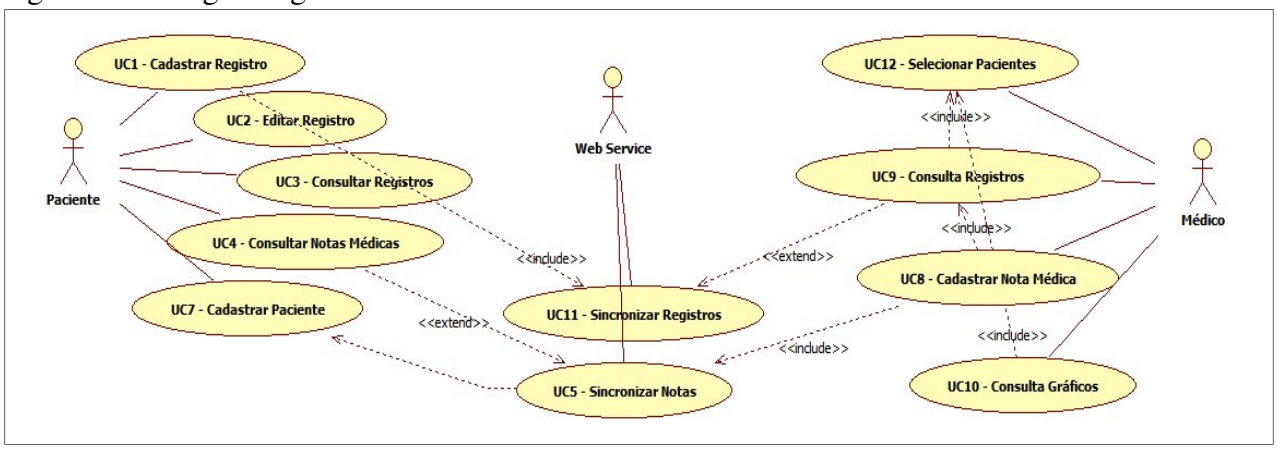
Figura 12 – Diagrama de Casos de Uso do ator Médico



O diagrama visualizado na Figura 12 apresenta o elemento de inclusão. O mesmo é exibido para a execução de determinadas ações, como por exemplo: para consultar um registro é necessária a seleção de um paciente. Isso ocorre, pois um médico poderá trabalhar com diversos pacientes ao mesmo tempo.

Um diagrama geral sobre todo fluxo de informações do aplicativo é apresentado na Figura 13, e apresenta o *Web Service* como um ator que executa ações de sincronização. Neste caso de uso geral, não são apresentados todos os casos de uso, somente aqueles que possuem uma maior importância.

Figura 13 – Diagrama geral de Casos de Uso do sistema



No diagrama da Figura 13, é possível notar a necessidade da sincronização por meio do WS para que seja possível a consulta das notas médicas no dispositivo do paciente, bem como a necessidade do cadastro dos registros para consulta destes no dispositivo do paciente. Como os casos de uso criados fica mais visível ao desenvolvedor ao regras a serem aplicadas ao software, possibilitando maior entendimento das regras aplicadas ao negócio do que as existentes no fluxo.

Com a diagramação de casos de uso realizada, o próximo passo é a realização da modelagem das tabelas para o sistema. Que nesta situação é dividida em duas partes.

6.2.2 Modelagem de banco de dados

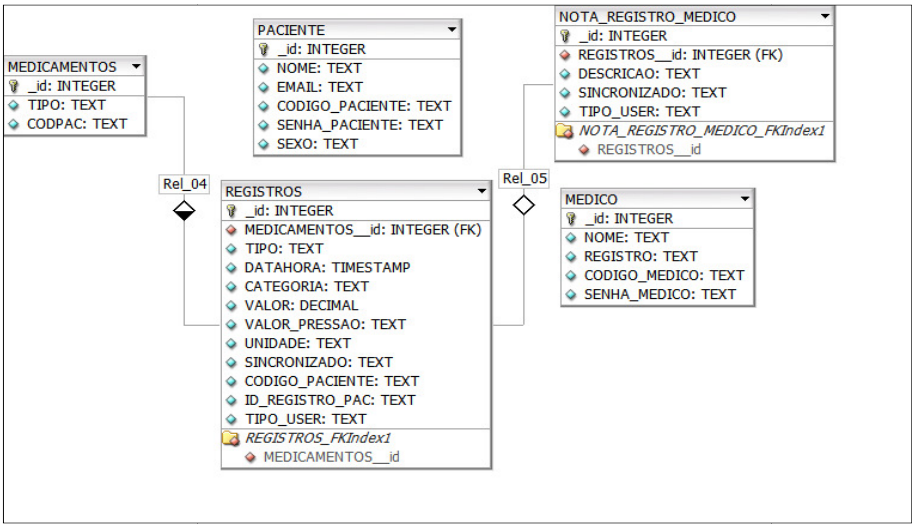
Com a necessidade de desenvolvimento em duas plataformas, tornou-se necessário a modelagem dos bancos de dados em dois esquemas de banco de dados diferentes. A modelagem é apresentada em seu modelo físico, tendo sido desenvolvida primeiramente para o banco disponível nos dispositivos móveis e posteriormente no *Web Service*.

Os modelos foram criados na ferramenta DBDesigner 4. A ferramenta foi escolhida por estar disponível de forma open-source e possuir uma modelagem eficiente com alta produtividade.

Na Figura 14 é possível verificar o modelo físico de banco de dados presente nos dispositivos móveis de médicos e pacientes. Algumas peculiaridades são notadas no banco utilizado no Android, como o fato de toda chave primária iniciar como *_id*. Outra diferença deste modelo para um modelo convencional é que algumas tabelas que necessitavam ser criadas para armazenamento de configurações, não foram necessárias devido ao fato do

Android possibilitar a criação das preferências do sistema por meio de um XML de configurações.

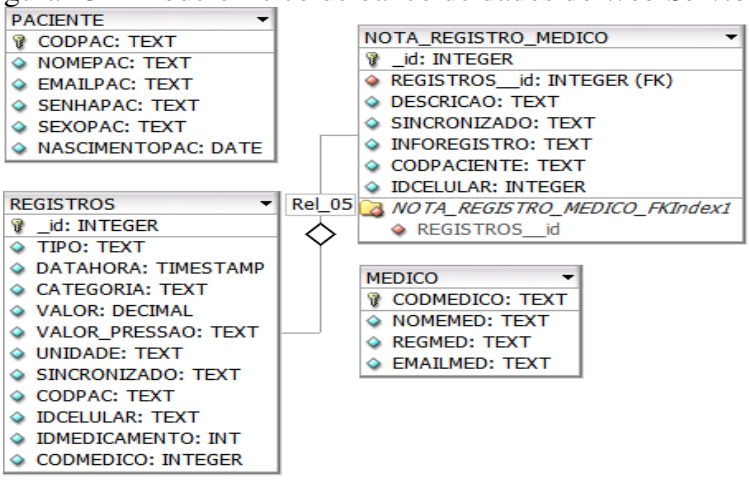
Figura 14 – Modelo físico de banco de dados do dispositivo móvel



A ausência de relacionamentos entre algumas tabelas como MEDICO e NOTA_REGISTRO_MEDICO e PACIENTE e REGISTROS é justificada pelo fato do banco ficar disponível tanto nos dispositivos dos pacientes quantos nos dispositivos dos médicos, e pelo relacionamento se dar como um relacionamento lógico com as tabelas disponíveis no WS.

Na Figura 15 é apresentado o modelo criado para ficar disponível no WS, e servir como armazenamento das informações enquanto essas trafegam entre os dispositivos de médicos e pacientes. As tabelas apresentadas possuem um modelo muito semelhante, possuindo apenas algumas colunas diferentes.

Figura 15 – Modelo físico de banco de dados do *Web Service*



Desta forma, os modelos são uma continuação um do outro. Cada informação disponível em um modelo é replicada para o próximo.

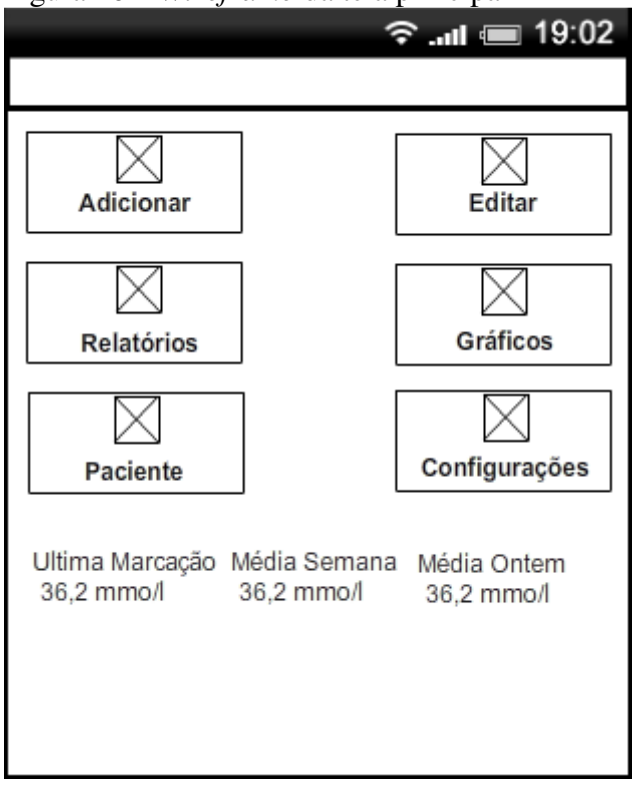
Com o modelo pronto, foi possível desenvolver os *wireframes* da aplicação, visto que já era possível definir a partir do modelo os campos que estariam disponíveis em cada tela.

6.2.3 Modelos de telas (*wireframes*)

Devido a complexidade de se construir um layout de tela e paralelamente já realizar o desenvolvimento da mesma utilizando o XML para criação de uma tela, as principais telas do sistema foram primeiramente concebidas utilizando *wireframes*. Como o objetivo não era modelar todas as telas do sistema, a ferramenta escolhida para criação dos *wireframes* foi a MockFlow, por permitir a criação de layouts sem a instalação de um aplicativo, sendo executado no navegador, e por possibilitar a criação de um projeto de forma gratuita.

Foram criadas três *wireframes* no MockFlow para servirem como base para o desenvolvimento dos XML layout das telas no Android. Na Figura 16 é possível verificar o desenho a página principal utilizando a modelagem da tela em *wireframe*.

Figura 16 – *Wireframe* da tela principal

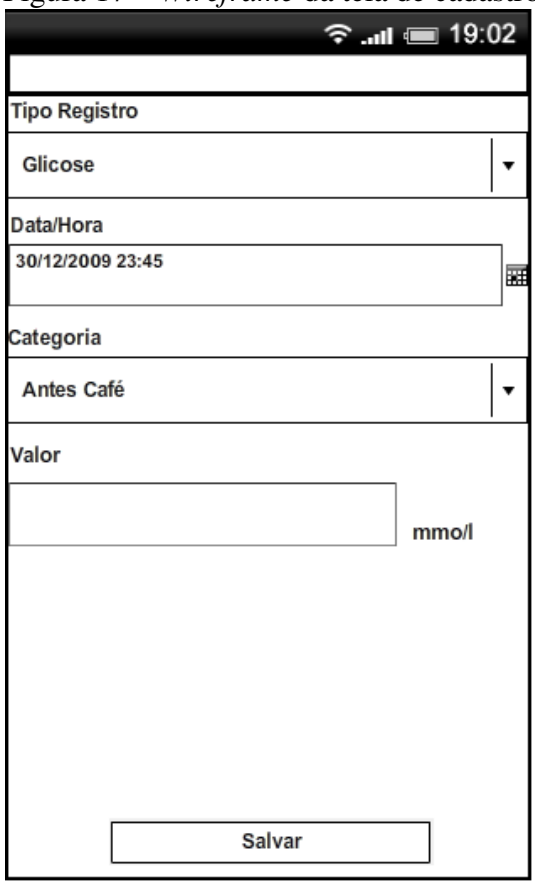


O *wireframe* da primeira tela apresenta a distribuição dos menus no aplicativo quando esse estivesse no modo paciente. Nesta tela é apresentado os botões de ação para as funcionalidades de adicionar, editar, alterar configurações, verificar gráficos e relatórios. Também é possível verificar que abaixo dos botões existe a informação com o resumo atual dos registros de glicose criados pelo paciente.

Com a visualização destas informações em um formato de tela, fica mais simples posterior implementação da tela.

Na Figura 17, é apresentado o wireframe da tela de cadastro de registros do paciente.

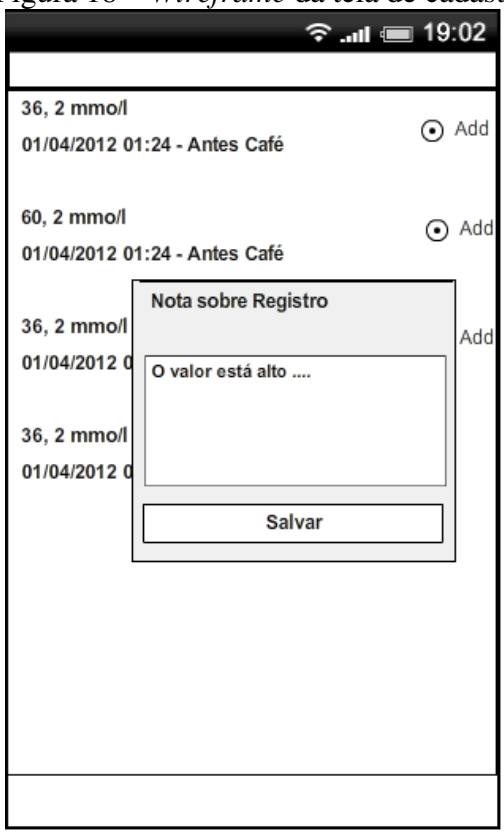
Figura 17 – *Wireframe* da tela de cadastro de registros



Por meio desta tela, é que são inseridas as informações de registros de aplicações de insulina, medidas de glicose, informações de peso e medicações tomadas. Esses registros possuem obrigatoriamente valor, data/hora e uma categoria. Esta é a principal tela de entrada de dados no aplicativo.

Outra tela identificada como uma das telas mais utilizadas do sistema é a tela de cadastro de notas médicas. Quando existe a sincronização das informações dos médicos e dos pacientes, o médico visualiza as informações dos registros gravados pelo paciente e pode adicionar notas a esses registros como é possível verificar na Figura 18.

Figura 18 – *Wireframe* da tela de cadastro de notas



Na Figura 18, é possível identificar que ao selecionar um registro cadastrado pelo paciente, será aberta um *alertdialog* para que o médico se informar alguma nota a ser enviada para o paciente que julgue necessária.

Por meio dos *wireframes*, foi possível identificar o fluxo de ações das principais telas do sistema, facilitando a criação de layouts e *activities* durante a implementação. A partir disto foi possível iniciar com a criação do ambiente de desenvolvimento e posterior desenvolvimento do aplicativo.

6.2.4 Ambiente de desenvolvimento

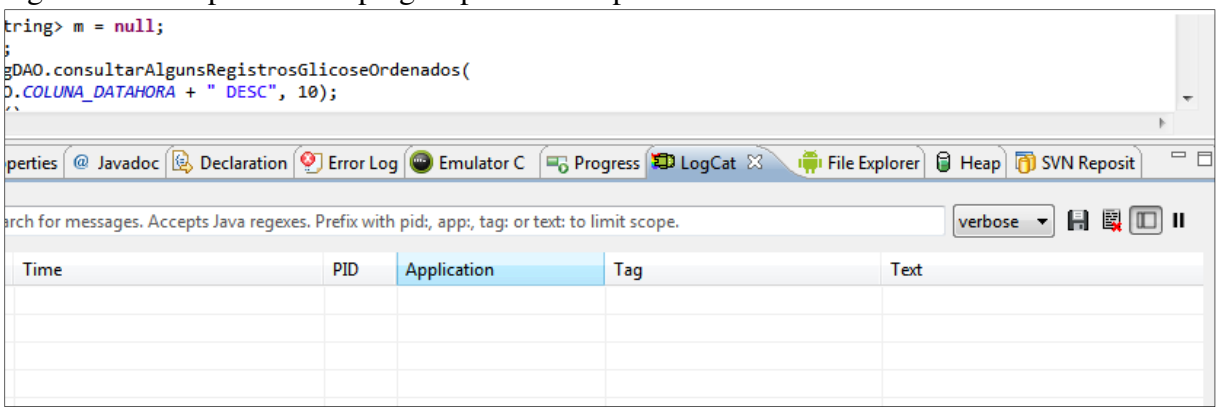
Para implementar o aplicativo foi necessário configurar o ambiente de desenvolvimento para que fosse possível obter os melhores resultados com maior

produtividade. Logo, foram selecionadas diferentes ferramentas de desenvolvimento, que por suas características auxiliaram na codificação da proposta.

Diante disto, foram criados dois ambientes de desenvolvimento distintos. O desenvolvimento *mobile* foi realizado no Eclipse IDE e o *Web Service* no NetBeans. O software gerenciador de banco de dados das duas aplicações seguiu a mesma regra, onde para o desenvolvimento no Android foi utilizado o SQLite e para o desenvolvimento do serviço foi usado o Oracle 10g Express Edition.

Para desenvolvimento Android, foram realizadas as instalações e configurações do Android SDK e o AVD manager, que são os elementos principais para o desenvolvimento *mobile*. Com o Eclipse como ferramenta de desenvolvimento foi possível usar *plugins* do *Android Development Tools* (ADT) e *SubClipse*. O Android ADT, como é possível verificar na Figura 19, adiciona ferramentas de depuração de erros, visualização de threads, alocação de memória, visualização de arquivos do sistema entre outras ferramentas que axiliam durante o desenvolvimento. Por sua vez, o SubClipse facilita versionamento dos arquivos de código fonte dentro da IDE.

Figura 19 – Eclipse com os plugins para SubClipe e Android ADT

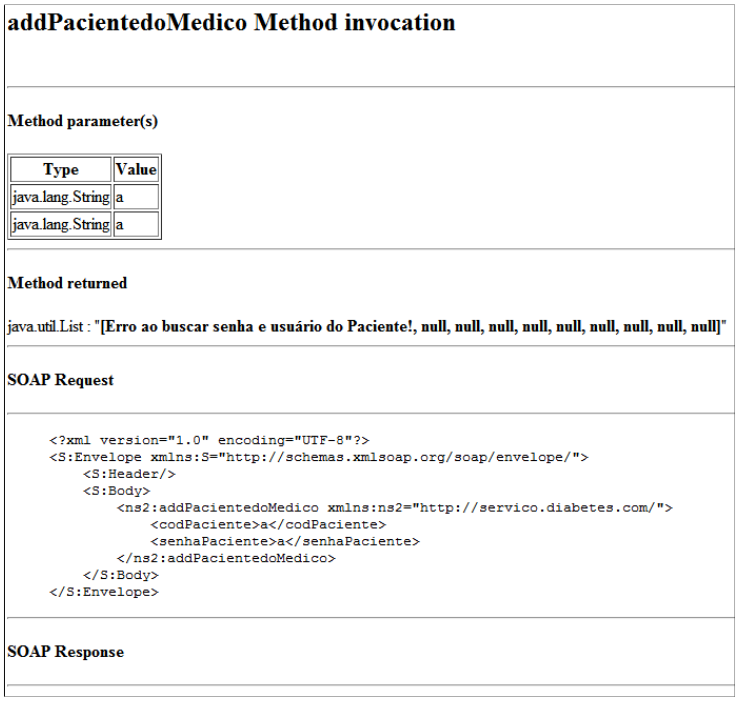


A IDE NetBeans, foi escolhida como ferramenta para o desenvolvimento do serviço web, por apresentar diversas configurações prontas e ferramentas que auxiliariam na codificação de um serviço web. Diferentemente do Eclipse, o NetBeans possui opção de ser instalado com um servidor de aplicações já configurado. Outra facilidade oferecida pelo ambiente de desenvolvimento foi a possibilidade de se testar os *Web Services* de maneira simples e isolada, sem a necessidade de utilização de outras ferramentas como é possível verificar na Figura 20.

Figura 20 – Página gerada pelo NetBeans para teste do *Web Service*

Dentro da pasta do serviço no NetBeans, ao clicar em 'Testar serviço Web' será aberta a página conforme a Figura 20, e ao informar os parâmetros do serviço e clicar no botão que possui o nome do serivço, é apresentada uma página como a Figura 21, onde é apresentado o XML de requisição e a resposta do mesmo.

Figura 21 – SOAP Request e SOAP Response no Netbeans



Outra facilidade da utilizando do NetBeans como IDE de desenvolvimento, também é a possibilidade de criar as operações dos *Web Services* de maneira gráfica. A ferramenta permite gerar os arquivos WSDL, XSD e todas classes JAX-WS de maneira simplificada.

O banco escolhido foi o Oracle. Para realizando do Mapeamento objeto-relacional, foi utilizada a biblioteca EclipseLink. Com o EclipseLink o controle para realização das persistências e a criação das tabelas com base nos objetos, é criada pela

biblioteca. O EclipseLink, que implementa a especificação Java Persistence API (JPA) e permitiu a implementação das persistências dentro das classes de modelo.

De forma semelhante ao Eclipse, no NetBeans também foi utilizado o versionamento do código fonte. Para as duas implementações, o código fonte foi versionado usando o serviço do Google Code, que possibilita o versionamento de projetos na web. Todos projetos hospedados no Google Code possuem código fonte aberto. Desta forma é possível encontrar os projetos desenvolvidos em **http://code.google.com/p/diabetes-control-ws/** e **http://code.google.com/p/diabetes-control/**.

6.3 - IMPLEMENTAÇÃO

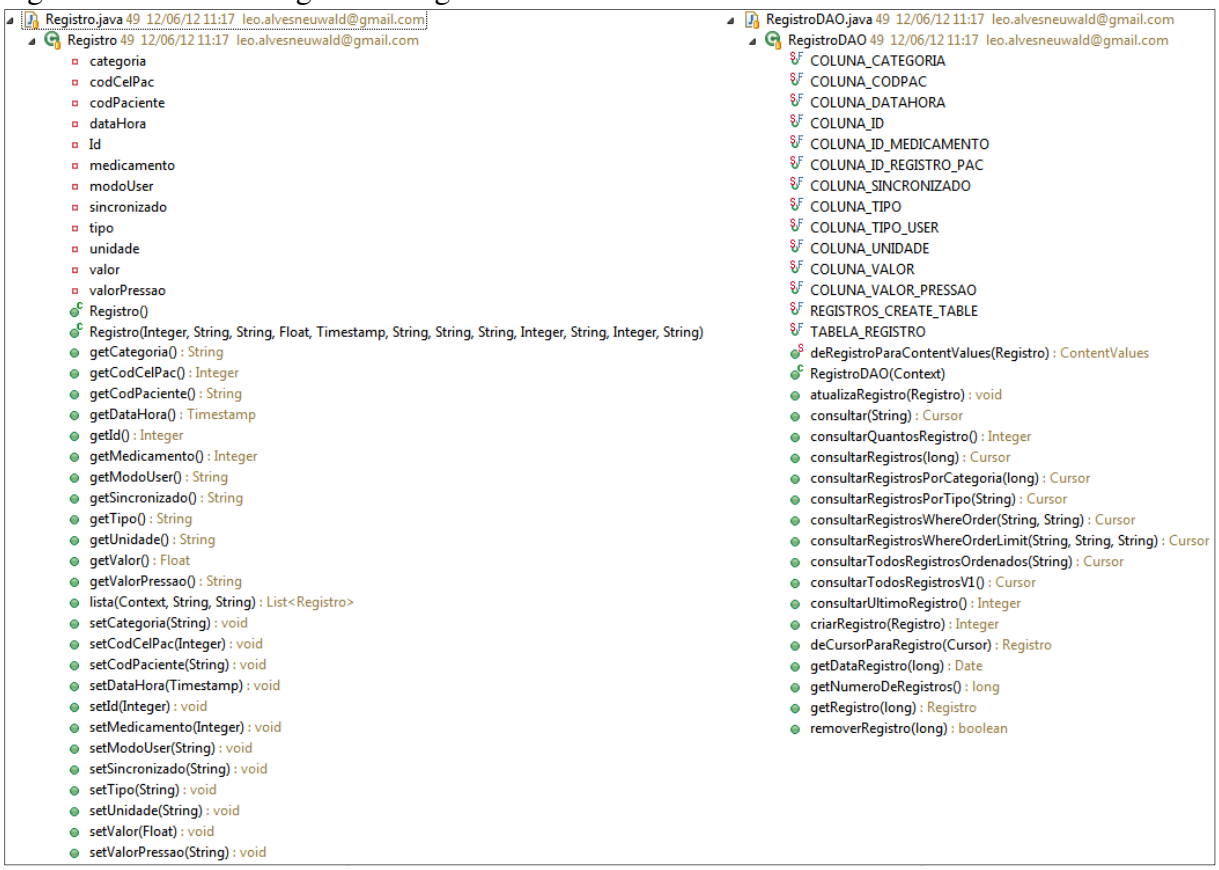
Definidos as tecnologias utilizadas e a modelagem, esta parte apresenta a implementação do aplicativo.

O desenvolvimento foi iniciado pelo aplicativo disponível para os dispositivos móveis, visto que independentemente de um serviço web, o aplicativo deveria sozinho ser capaz de gerenciar as informações relacionadas a diabetes do paciente. O projeto foi desenvolvido seguindo o padrão Model View Controller (MVC), onde os XMLs de configuração de layout representam a *view*, as *activities* representam o *controller* e o modelo é representado pelos objetos.

A versão do Android utilizada foi a versão 2.3 por ter o maior número de dispositivos usando essa versão e por possibilitar que todos usuários que utilizem o Android acima desta versão, também possam utilizar o Diabetes Control.

Foram criadas as classes modelos Medico, NotaRegistroMedico, Paciente e Registro. Esses objetos representam respectivamente, as informações do médico, as notas médicas que os médicos cadastram para os registros dos pacientes, as informações do paciente, o registro gerado pelo paciente sobre a diabetes e o registro médico. Com base nessas classes, foram criadas classes de Data Access Object (DAO), que encapsulam as funções do banco de dados. Na Figura 22 é apresentado um exemplo de um objeto e seu respectivo DAO.

Figura 22 – Classes Registro e RegistroDAO



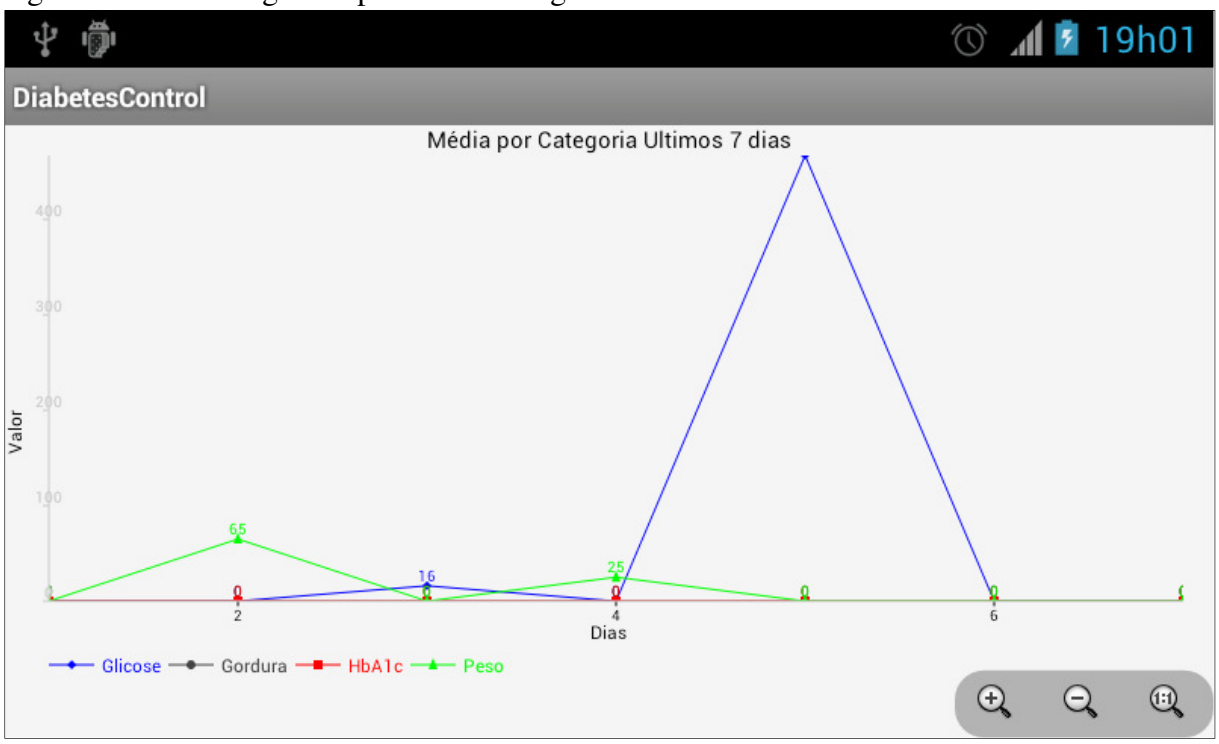
Todas as classes DAO criadas estendem a classe BasicoDAO. Na classe BasicoDAO é onde estão disponíveis os métodos de abertura e fechamento de conexão. Com o Android, existe a necessidade da criação de uma classe para gerenciar essa abertura e fechamento de conexão, porque não é possível que sejam abertas duas conexões de dados para execução de qualquer ação no banco no dispositivo.

Outra particularidade do desenvolvimento para Android é a criação e atualização de tabelas que é feita diretamente por uma classe que estende SQLiteOpenHelper. No projeto desenvolvido o nome da classe é ContextoDados, e inicialmente sempre que o aplicativo é instalado em um dispositivo, esse realiza a criação das tabelas conforme o script disponível no método *onCreate*. Quando o mesmo aplicativo é atualizado em um mesmo dispositivo, o método a ser chamado é o *onUpgrade*, que verifica se a versão do banco de dados foi atualizada, e caso tenha sido, roda o script de atualização disponível dentro do método.

Uma das necessidades para que seja possível controlar a diabetes é a necessidade de visualização destas informações em formato de gráficos. Diferentemente de outras funcionalidades, o Android não possui nenhum tipo de biblioteca nativa para o desenvolvimento de gráficos. Algumas outras soluções como o framework AChartEngine,

possibilitaram a criação de gráficos. Segundo 4viewsoft (2012), o AChartEngine é um framework open-source para geração de gráficos em dispositivos Android. Na Figura 23 Um exemplo de um gráfico gerado com o AChartEngine

Figura 23 – Gráfico gerado pelo AChartEngine



Existem ainda diversas outras possibilidades de criação de gráficos utilizando essa biblioteca, como será possível verificar mais adiante.

Na plataforma Google Android, a tela é criada em XML e manipulada em uma classe *activity*. Os elementos criados em um arquivo XML, ficam disponível na *activity* através da classe R do Android que é gerada automaticamente pelo ADT. Nessa classe estão disponíveis acesso a todos recursos disponíveis dentro da pasta *res*. Outro padrão para o desenvolvimento na plataforma, é a necessidade de declaração das *activities* e permissões dentro do arquivo *AndroidManifest*. Dentro deste arquivo XML, são colocadas todas as permissões que o aplicativo irá utilizar, bem como todas as *activities*. Na Figura 24 é demonstrado parte do *AndroidManifest* utilizando no desenvolvimento do projeto.

Figura 24 – Arquivo AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.diabetescontrol.activities"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/iv"
        android:label="@string/app_name" >
        <activity android:name="org.achartengine.GraphicalActivity" />
        <activity android:name=".GeneratedChartDemo" />
        <activity
            android:name=".GraficosPacienteActivity"
            android:theme="@style/MapBackground" />
    </application>
</manifest>
```

Como é possível identificar na Figura 24, estão sendo utilizadas as permissões para acesso a internet, acesso ao estado da conexão e acesso a localização atual. Essas permissões são necessárias para que seja possível realizar a conexão com o serviço web. Na classe Utils, existe um método que verifica se existe alguma conexão disponível com a internet, caso não exista, os dados cadastrados não são sincronizados.

Outra classe utilitária desenvolvida foi a DataUtil. Com ele é possível saber o primeiro dia da semana passada, o ultimo dia do mês passado, entre outras datas, necessárias para se apresentar os gráficos de diabetes e apresentar médias.

Um padrão muito interessante para o desenvolvimento em Android, é a possibilidade de criação de um arquivo de preferências do aplicativo. Esse arquivo é criado em XML, da mesma forma que é criada uma tela, e ele irá guardar todas as preferências utilizadas no restante do projeto. Para o projeto Diabetes Control, as unidades de medidas, a sincronização ou não das informações cadastradas e outros dados importantes e utilizados em diversos locais da aplicação, são salvos neste arquivo de preferências, diminuindo assim a necessidade de criação de algumas tabelas para guardar essas informações. Assim quando é necessário buscar qualquer informação salva basta utilizar um comando, sem a necessidade de abrir uma conexão com o banco e buscar em uma tabela de preferências.

Por fim, o desenvolvimento na plataforma Google Android, se deu com a criação das classes para conexão com o *Web Service*. Para criação destas classes, foi utilizada a biblioteca Ksoap2 para Android. Conforme Ksoap (2012), a biblioteca é um cliente Soap desenvolvida em Java 1.3 de forma Open-Source. A Figura 25 apresentada uma classe de conexão com o *Web Service* utilizando a biblioteca, aonde é enviado um SoapObject para o *Web Service*, que irá retornar um vetor de objetos.

Figura 25 – Classe de conexão com o Web Service usando Ksoap2

```
import java.util.Vector;

public class Service {
    private String soap_action = "";
    private String url = "";

    public Service(Context ctx, String soap_action) {
        this.soap_action = soap_action;
        this.url = Utils.URL_WS(ctx);
    }

    @SuppressWarnings("unchecked")
    public String[] execute(SoapObject request) {
        SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
                SoapEnvelope.VER11);
        MarshalDate md = new MarshalDate();
        md.register(envelope);
        envelope.setOutputSoapObject(request);
        try {
            HttpTransportSE androidHttpTransport = new HttpTransportSE(url);
            androidHttpTransport.call(soap_action, envelope);
            Vector<Object> rs = (java.util.Vector<Object>) envelope
                    .getResponse();
            String[] mensagens = new String[20];
            if (rs != null) {
                int i = 0;
                for (Object cs : rs) {
                    if (cs != null) {
                        mensagens[i] = cs.toString();
                    }
                    i++;
                }
            }
            return mensagens;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Conforme é possível verificar na Figura 25, com a implementação utilizando Ksoap2, a comunicação com o *Web Service* acontece de forma simples.

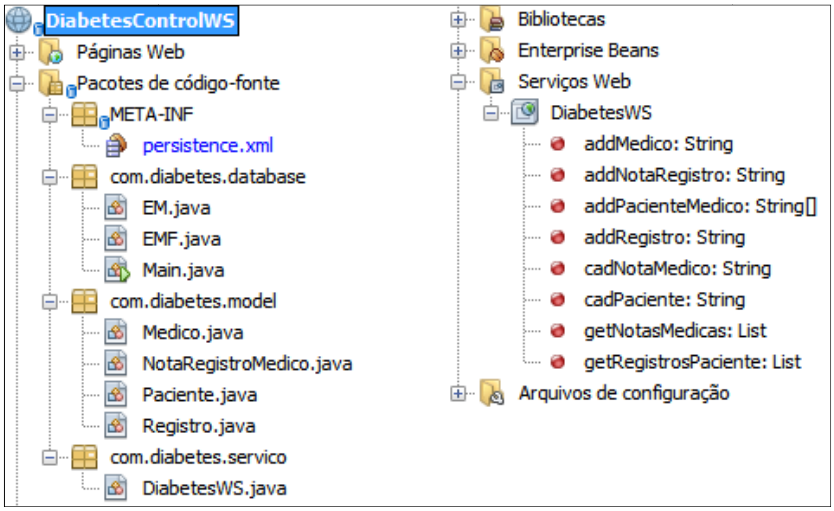
Finalizando do desenvolvimento da parte móvel do projeto, foi iniciada a criação do *Web Service* que iria receber as informações do aplicativo.

O WS foi concebido para receber informações dos registros de diabetes, pacientes cadastrados e notas médicas sobre os registros cadastrados. Como já citado anteriormente, o WS foi desenvolvimento utilizando o JPA EclipseLink, o que facilitou muito a implementação do aplicativo. Com o JPA, não foi necessária a criação de classes DAO, bastando realizar as anotações corretas dentro das classes de modelo para que fosse possível salvar, editar e excluir informações.

O projeto foi divido em três pacotes principais, database, model e serviço. Dentro do pacote database estão as classes que criam as persistências com o banco de dados usando o EclipseLink. As classes do pacote modelo possuem os objetos que são utilizados para persistir os dados e salvar as informações. Por último temos o pacote serviço, que é onde se encontra a classe que cria os métodos para conexão com o WS. O projeto também possui classes geradas pelo JAX-WS com os objetos do WS, bem como um arquivo XML com as configurações

necessárias para a criação das tabelas automaticamente utilizando o JPA. Na figura 26 é possível verificar toda estrutura desenvolvida no NetBeans para o serviço.

Figura 26 – Estrutura do *Web Service* desenvolvido



JAX-WS é a API do Java para criação dos serviços web utilizando XML. Essa API foi utilizando dentro do projeto com a finalidade de criação do serviço. Com essa API as configurações dos WS podem ser feitas através de anotações. Na Figura 27 é possível verificar como são realizadas essas anotações dentro do projeto.

Figura 27 – Anotações JAX-WS para criação de uma classe de serviço

```
import javax.jws.WebParam;
import javax.ejb.Stateless;

/**
 *
 * @author Leonardo
 */
@WebService(serviceName = "DiabetesWS")
@Stateless()
public class DiabetesWS {

    /**
     * Operação de serviço web
     */
    @WebMethod(operationName = "addRegistro")
    public String addRegistro(@WebParam(name = "tipo") String tipo) {
        //Code
        return "Sucess";
    }
```

Com a criação das classes do WS, é possível realizar a geração do WSDL e dos demais arquivos gerados pelo JAX-WS. O WSDL, arquivo que XML que descreve o serviço, gerado se encontra no Apêndice A.

Com as duas implementações prontas, bastou executar uma das rotinas do Android que realiza sincronização com o WS para verificar a sincronização dos dados entre os aplicativos. É possível encontrar a classe principal que contém os serviços web no Apêndice B. No Apêndice C é possível encontrar a classe que enviar os registros e recebe os registros do *Web Service* usando a biblioteca KSoap2 no Apêndice C.

Após a finalização da codificação, se iniciou a fase de testes do aplicativo.

6.4 – TESTES DE IMPLEMENTAÇÃO

Com o fim do desenvolvimento do aplicativo, se iniciou a faze de testes. O aplicativo móvel passou pelas fazes de teste em um aparelho Sansung Nexus S com 1 gb de memória RAM e tela de 4 polegadas rodando o Android 4.0.4 e por um emulador do Android 2.3 com a configuração de tela como QVGA (240x320 pixels) e 256Mb de memória RAM. Já o servidor web passou por testes em um servidor GlassFish local conectado ao banco Oracle 10g Express Edition.

Os testes foram realizados em uma rede local utilizando IP interno para configurar o acesso do celular ao WS. Os testes procederam com sucesso, sendo que os problemas encontrados que não foram corrigidos no mesmo momento foram cadastrados como *bugs* nas páginas dos projetos no Google Code.

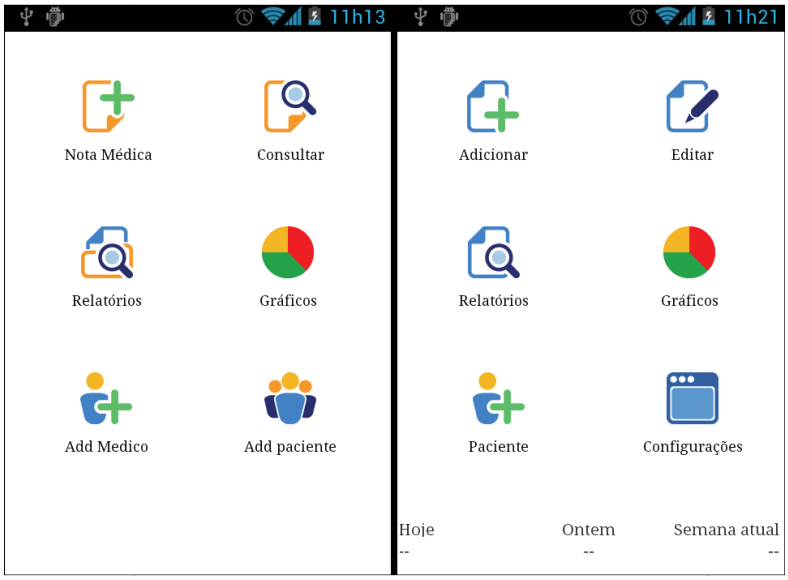
6.5 – RESULTADOS OBTIDOS

Como resultados foram criadas duas aplicações distintas. O WS, que regula as informações e transações entre dispositivos de paciente e médico e a própria implementação do dispositivo móvel se baseando na tecnologia Android. A aplicação móvel pode ser dividida como aplicação de dois módulos distintos. O módulo paciente, onde as informações são cadastradas pelo paciente e enviadas para o WS, e o módulo do médico, que recebe as informações do paciente e permite a realização de uma análise do médico que pode inclusive cadastrar notas médicas sobre os registros cadastrados pelo paciente.

Na aplicação móvel, está disponível para o paciente uma série de facilidades para o controle de diabetes. Gráficos, relatórios e indicadores em geral que facilitam a interpretação dos dados coletados. O sistema possui uma interface simples e intuitiva para que qualquer usuário independentemente do nível de conhecimento, consiga utilizar o software

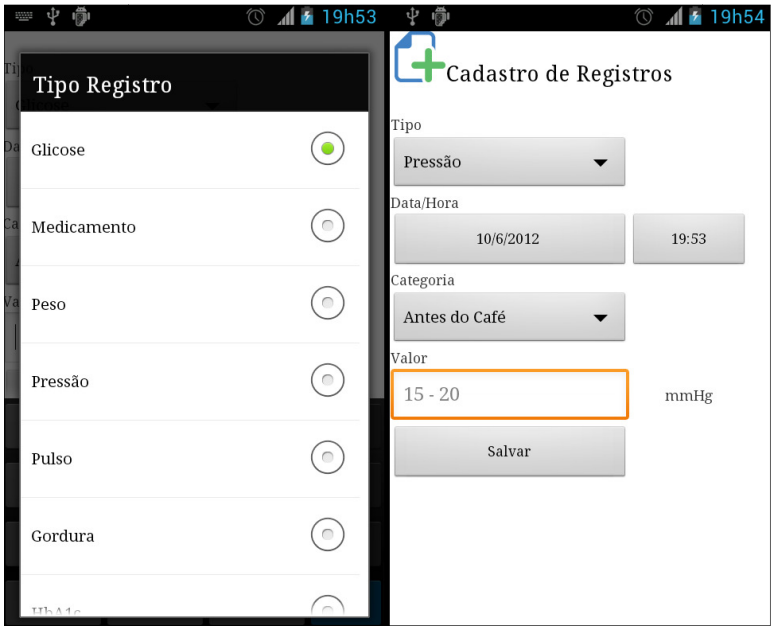
sem dificuldades, conseguindo tirar o máximo proveito das informações geradas. Na Figura 28, é possível verificar a tela principal do sistema no módulo paciente e médico.

Figura 28 – Telas iniciais dos módulos Médico e Paciente



Através das telas iniciais de Paciente e Medico, é possível acessar as principais funcionalidades do sistema. Os registros do paciente são cadastrados por meio da tela de "Cadastro de Registros" acessada através do botão "Adicionar" na tela inicial do módulo Paciente. Conforme é possível verificar na Figura 29, a tela permite o cadastramento de medicamentos, glicose, peso, pressão, pulso, gordura e HbA1c.

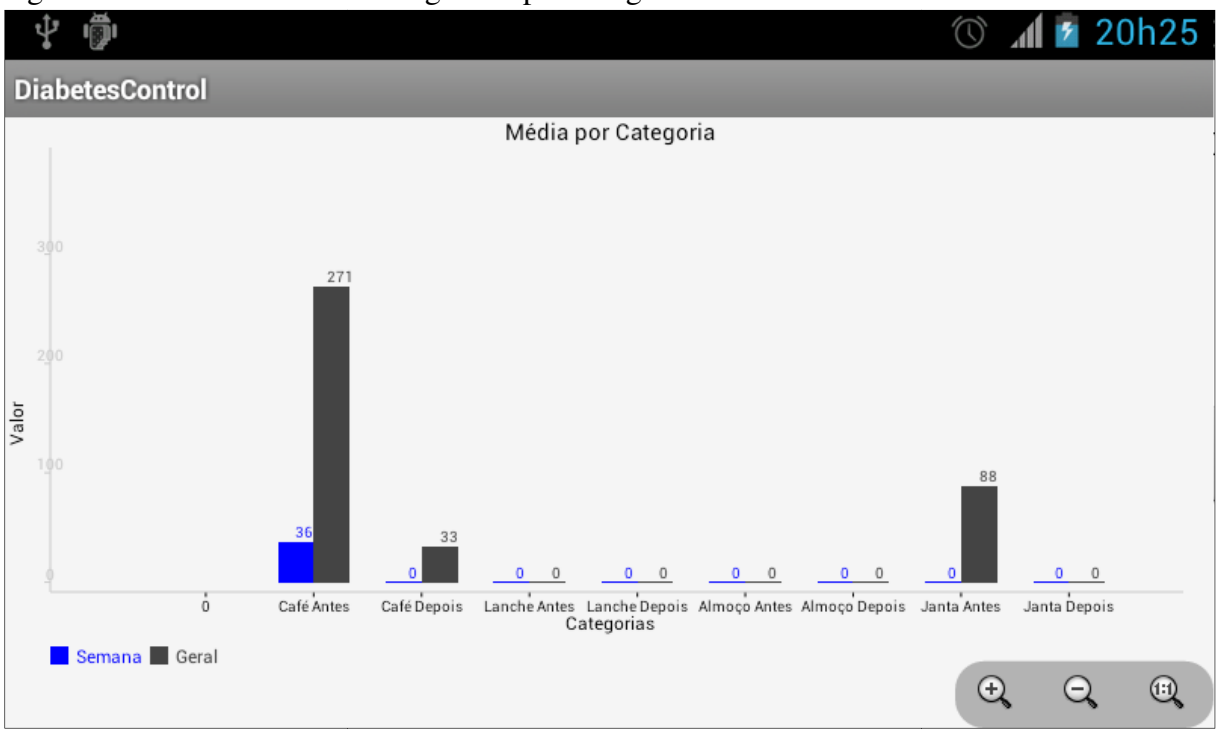
Figura 29 – Seleção de Tipo Registro e Cadastro de Registros



A Figura 29 demonstra as opções de seleção do tipo de registro. Na tela ao lado é apresentado o cadastro do registro com o tipo pressão selecionado. Caso o tipo selecionado fosse medicamento, seria possível selecionar uma opção com os diversos tipos de insulinas para que fosse escolhido um destes. Essa tela é a principal entrada de dados do sistema, sendo que com os dados gerados nesta tela são criados gráficos e relatórios.

Um dos gráficos gerados pelo sistema que auxilia no controle da diabetes é o gráfico de média de glicose por categoria. Conforme é possível verificar na Figura 30, são apresentas informações sobre a média da semana comparadas com a média geral de glicose por categoria. Baseado nestas informações o médico e paciente poderão comparar o tratamento atual com o tratamento que foi aplicado em períodos anteriores, podendo realizar alterações nos medicamentos.

Figura 30 – Gráfico de média de glicose por categoria



O Diabetes Control, também possibilita a emissão de relatórios que auxiliam no monitoramento de diversos indicadores da diabetes. Um desses indicadores que o sistema possibilita um acompanhamento por meio de relatórios é a média por período. Como é possível verificar na Figura 31, o sistema possibilita que sejam emitidos relatórios com a média do dia atual, dia anterior, semana atual, semana anterior, mês atual, mês anterior, anual

e o total geral de todas as marcações, possibilitando assim um acompanhamento sobre a evolução da doença em diversos períodos.

Figura 31 – Relatório de média por período

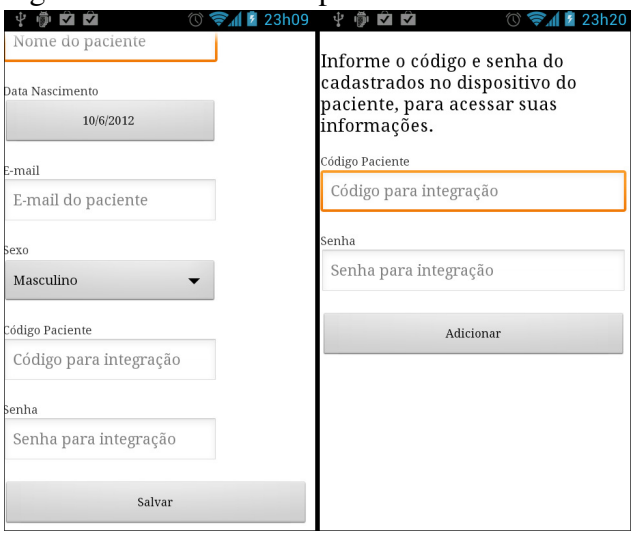
Além do relatório da Figura 31, estão disponíveis também, relatórios de aplicações de insulina, HbA1c médio, registros diários, ultimas marcações, maiores valores, entre outros relatórios que ajudam a manter uma inspeção diária sobre a diabetes.

Com o objetivo de aumentar ainda mais esse controle, foram criadas as rotinas para sincronização das informações cadastradas pelo paciente. Para que essa sincronização seja realizada, é necessário primeiramente, que o dispositivo do paciente esteja devidamente configurado. A configuração acontece primeiramente com o cadastro das informações do paciente no seu próprio celular. Isso é necessário, pois esse cadastro irá funcionar como um *login*. Quando o paciente realizar seu cadastro, todas suas informações que forem enviadas ao *Web Service*, irão possuir a sua identificação, o que permite que o médico consiga coletar somente as informações do seu paciente. Para que o médico obtenha as informações do seu paciente, será necessário informar o usuário e senha cadastrados no dispositivo móvel do paciente. Desta forma, é possível constatar que realmente o médico possui acesso as informações do paciente.

A Figura 32 apresenta as duas telas envolvidas na configuração da sincronização das informações de pacientes e médicos. Ao lado direito é apresentada a tela de cadastro do

paciente, cujas informações são preenchidas pelo paciente para identificar todos seus registros que irão ser sincronizados. Ao lado esquerdo, é apresentada a tela de acesso do médico as informações do paciente.

Figura 32 – Cadastro de pacientes e acesso as informações



Com essa configuração realizada, já é possível para o médico, acessar as informações dos registros de seus pacientes, fornecendo segurança ao paciente, que somente terá seus registros visualizados por médicos autorizados por ele.

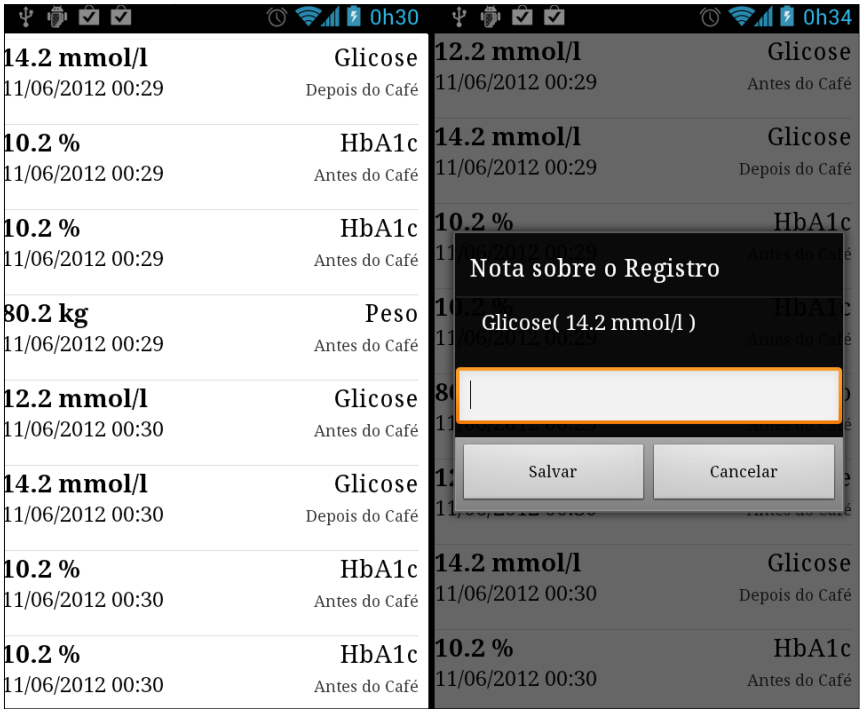
Ao acessar a tela de cadastro de notas médicas, primeiramente é apresenta a tela para seleção do paciente, conforme é possível notar na Figura 33. Qualquer relatório ou gráfico, antes de ser visualizado no módulo médico, necessita que seja selecionado o paciente de origem das informações.

Figura 33 – Lista de pacientes do médico

Após selecionar o paciente, o médico visualiza as informações sobre os registros cadastrados por um determinado paciente. A partir destes registros, o médico poderá cadastrar notas médicas para cada um deles. As notas, são sincronizadas com o dispositivo do paciente para que posteriormente o ele possa verificar as informações cadastradas pelo médico. Essa nota médica pode ser um lembrete de uma medicação que não está sendo tomada, uma mudança de tratamento ou qualquer informação que o médico julgue necessária.

Na Figura 34 é possível verificar a lista dos registros que foram sincronizados para o dispositivo do médico, e a nota que está sendo gerada para um destes registros.

Figura 34 – Lista de registros do paciente e nota médica sobre o registro



Após o cadastro da nota, esse registro também é enviado ao WS, permitindo que o paciente consulte essa informação.

Em ambos módulos do aplicativo ainda existe a *activity* de configuração. Através desta página é possível selecionar as unidades de medidas de peso, glicose, HbA1c e outros tipos de registros a serem cadastrados. Nesta *activity* também é possível selecionar se a sincronização irá ou não ocorrer, visto que o módulo paciente pode ser utilizado normalmente sem a sincronização das informações. Quando a opção de sincronização está marcada, ao salvar um cadastro ou executar alguma outra ação, essas informações são submetidas a

sincronização. Caso essa sincronização não aconteça instantaneamente, ocorrerá no próximo momento, sempre de forma assíncrona.

Pode-se concluir que todos objetivos propostos foram concluídos, tendo em vista a construção de um aplicativo para controle de diabetes do paciente, que gera indicadores desta doença para o paciente e para o médico, podendo ou não realizar a sincronização das informações via WS com o celular de um médico, utilizando para isso a tecnologia Android.

O modelo proposto está disponível na Google Play sobre o link: **https://play.google.com/store/apps/details?id=com.diabetescontrol.activities** .

6.6 – DIFICULDADES ENCONTRADAS

As principais dificuldades sugiram durante a implementação do *Web Service*. Inicialmente foi escolhida como plataforma de desenvolvimento do *Web Service*, a estrutura do Google App Engine. Entretanto, após inúmeros testes não foi possível obter sucesso na conexão do WS com o dispositivo móvel utilizando a biblioteca Ksoap2 para troca de informações utilizando SOAP. Devido a esses problemas, a implementação do WS, foi realizada no NetBeans, que permitiu a fácil e rápida configuração do mesmo.

Outro problema encontrado na troca de informações com o serviço web, foi a deficiência da biblioteca Ksoap2 de trabalhar com tipos de dados complexos como *Float* e *Date* pois não conseguia serializar os mesmos. Desta forma, foi necessário buscar uma solução para transmissão destes dados na web, sendo necessário implementar uma classe especialmente para cada um destes tipos de dados.

Por último, a dificuldade de se encontrar bibliotecas de ORM para o Android, e a imaturidade destas bibliotecas fez com que o projeto fosse desenvolvido utilizando JDBC. O problema gerado pelo JDBC é a manutenção aplicada a cada alteração de colunas no banco. Esse problema foi solucionado em parte, buscando-se utilizar um padrão para criação das classes DAO e utilizando constantes para nomear colunas e tabelas.

CONCLUSÃO

A utilização de dispositivos móveis pela população está aumentando gradativamente, tornando os dispositivos uma nova fonte para desenvolvimento de soluções nas mais diversas tecnologias. As soluções móveis já são utilizadas em áreas como a saúde, onde a informação deve estar disponível todo momento. Tecnologias que permitam o acompanhamento do paciente em tempo real pelo médico utilizando tecnologias móveis permitem um tratamento muito mais eficaz.

A fim de alcançar os objetivos deste trabalho, que consiste no desenvolvimento de um aplicativo móvel utilizando a plataforma Google Android como forma de controlar os níveis da diabetes dos pacientes, trocando informações com médicos via WS, foi realizado estudo sobre a plataforma Google Android, sincronização de dispositivos com WS, telemedicina e diabetes. O aplicativo móvel foi desenvolvido utilizando tecnologias como Google Android, a IDE Eclipse e todas ferramentas da SDK Android. O *Web Service* foi implementado utilizando o padrão JAX-WS, utilizando a IDE NetBeans como ferramenta de desenvolvimento.

Neste contexto, foi criada uma solução que sincroniza informações cadastradas em um dispositivo móvel com tecnologia Android por meio de um WS, com outro dispositivo móvel. As informações trocadas possuem informações de registros de diabetes dos pacientes e notas médicas sobre os registros de diabetes. Utilizando a solução desenvolvida, é possível que médicos acompanhem a evolução diária de seus pacientes direcionando o tratamento conforme indicadores gerados em tempo real.

Pode-se concluir que a plataforma Google Android possibilita um desenvolvimento de soluções robustas e com alto nível de complexidade. Usadas junto com tecnologias como os *Web Services*, permitem que informações possam ser compartilhadas e utilizadas nas mais diversas áreas do conhecimento.

Concluindo esta pesquisa, como trabalhos futuros sugere-se:

a) a utilização de segurança no trafego de dados entre o serviço web e o dispositivo móvel;

b) ampliação dos indicadores de diabetes, bem como a criação de uma calculadora, que automaticamente iria gerar informações sobre os registros de diabetes;

c) aumento das funcionalidades do *Web Service* e implementação da solução na plataforma iOS;

d) criação de uma página web para acesso de médico e paciente que possibilite a visualização e geração das informações;

e) exportação dos gráficos, relatórios e outras informações, possibilitando o envio destas informações via e-mail.

REFERÊNCIAS

4VIEWSOFT. **AChartEngine**. Disponível em: <http://www.achartengine.org/content/download.html>. Acesso em: 02 jun. 2012.

ABINADER, Jorge Abílio; LINS, Rafael Dueire. **Web Services em Java**. Rio de Janeiro: Brasport, 2006.

ABI RESEARCH (New York). Android **Will Seize 45% of Smartphone Market by 2016**.Disponível em: <http://www.abiresearch.com/press/3651-Android+Will+Seize+45%25+of+Smartphone+Market+by+2016,+Says+ABI+Research>. Acesso em: 17 maio 2011.

ABREU, Leonardo Marques de. **Usabilidade de Telefones Celulares com base em Critérios Ergonômicos**. 2004. 294 f. Dissertação (Mestrado) - Curso de Pós graduação em Design, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0310214_05_Indice.html>. Acesso em: 08 out. 2011.

AIRSTRIP TECHNOLOGIES. **Be There From Anywhere**: The AirStrip Story. Disponível em: <http://www.airstriptech.com/Portals/_default/Skins/AirstripSkin/Corporate/AboutUs/tabid/75/Default.aspx>. Acesso em: 23 maio 2011.

ALLEN, Grant; OWENS, Mike. **The Definitive Guide to SQLite**. 2. ed. New York: Apress, 2010.

AMERICAN DIABETES ASSOCIATION. **Diabetes de A a Z**: O que você precisa saber sobre diabetes explicado de maneira simples. São Paulo: JSN, 1998. Tradução de: Olavo Antonio Brito.

ANTONELI, Diego Gomes. **M-Commerce e a plataforma Google Android**. 2011. 106 f. Monografia (Graduação) - Universidade do Extremo Sul Catarinente, Criciúma, 2011.

BEDRAN, Diogo. **Top 10 Tendências Mobile 2010**: Saúde. Disponível em: <http://readwriteweb.com.br/2010/06/08/top-10-tendencias-mobile-2010-parte-3/>. Acesso em: 17 maio 2011.

Blois, M.S., Shortliffe, E.H. The Computer Meets Medicine: Emergence of a Discipline. **Medical Informatics**: Computer Applications in Health Care, New York, n. , p.3-36, 1990. Addison-Wesley Publishing.

BUSINESS WIRE. **Three HCA South Florida Hospitals First in Nation To Use Pioneering Mobile Technology for Cardiac Care**: AirStrip CARDIOLOGY Allows Medical Personnel to Review, Analyze and Manage Electrocardiograph Data on Mobile Phones and Tablets. Disponível em: <http://www.businesswire.com/news/home/20110427005387/en>. Acesso em: 23 maio 2011.

BRAY, Tim. **What Android Is**. Disponível em: <http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is>. Acesso em: 28 set. 2011.

CALIXTO, Bruno. **Aplicativos para celular ajudam médicos no diagnóstico e tratamento de doenças**. Revista Época. Disponível em: <http://revistaepoca.globo.com/Revista/Epoca/0,,EMI258871-15257,00.html>. Acesso em: 30 out. 2011.

CANCELLIÉRI, Cláudio. **Diabetes & Atividade Física**. Jundiaí: Fontoura, 1999.

CERAMI, Ethan. **Web Services Essentials**. California, USA: O'Reilly, 2002. 304p.

CIBELLE BOUÇAS. **Telemedicina em saúde pública ganha recursos adicionais**. Planejamento - Ministério do Planejamento, Orçamento e Gestão. Disponível em: <https://conteudoclippingmp.planejamento.gov.br/cadastros/noticias/2011/3/30/telemedicina-em-saude-publica-ganha-recursos-adicionais>. Acesso em: 29 maio 2011.

COLLETTA, Denise Dalla. **Redução de impostos de tablets poderia abrir a porta para outros eletrônicos, diz especialista**: Para professor, Medida Provisória mostra que governo está aberto para negociar. Revista Galileu. Disponível em: <http://revistagalileu.globo.com/Revista/Common/0,,EMI235559-17770,00-REDUCAO+DE+IMPOSTOS+DE+TABLETS+PODERIA+ABRIR+A+PORTA+PARA+OUTROS+ELETRONICO.html>. Acesso em: 29 maio 2011.

COMPUTER WORLD. **MCT cadastra 12 empresas para produção de tablets no País**: Nas próximas semanas, será publicada Portaria dos ministérios da Ciência e Tecnologia e do MDIC, definindo as regras do PPB do produto. Disponível em: <http://computerworld.uol.com.br/tecnologia/2011/05/23/mct-cadastra-12-empresas-para-producao-de-tablets-no-pais/>. Acesso em: 29 maio 2011.

COMUNICAÇÕES, Ministério Das (Org.). **História da Telefonia: Linha do Tempo**. Disponível em: <http://www.mc.gov.br/o-ministerio/historico/historia-da-telefonia>. Acesso em: 08 out. 2011.

CONDER, Shane; DARCEY, Lauren. **Android wireless application development**. 2. ed. Boston: Pearson Education, 2010.

DALVIKVM.COM. **Dalvik Virtual Machine**: Brief overview of the Dalvik virtual machine and its insights.. Disponível em: <http://www.dalvikvm.com/>. Acesso em: 02 out. 2011.

DIEGO, Rafael Hernández de; JIMÉNEZ, Francisco Martínez; GONZÁLEZ, Javier Rocamora. **Arquitectura genérica para telemedicina basada en servicios web y Google Health**: Aplicación práctica en diabetes. Madrid: Facultad de Informática, 2010. Disponível em: <http://eprints.ucm.es/13075/2/Memoria_SSII_201011.pdf>. Acesso em: 08 nov. 2011.

DIGIBARN COMPUTER MUSEUM. **GRiDpad Pen Computer**. Disponível em: <http://www.digibarn.com/collections/systems/gridpad/index.html>. Acesso em: 15 nov. 2011.

DOLAN, Brian. **3 million downloads for Android health apps**. Disponível em: <http://mobihealthnews.com/6908/3-million-downloads-for-android-health-apps/>. Acesso em: 17 maio 2011.

FIGUEIREDO, Carlos Maurício Seródio; NAKAMURA, Eduardo. Computação Móvel: Novas Oportunidades e Novos Desafios. **T&C Amazônia**, Manaus, n. 2, p.16-28, 01 jun. 2003. Semestral. Disponível em: <https://portal.fucapi.br/tec/imagens/revistas/ed02_completo.pdf>. Acesso em: 12 out. 2011.

FELKER, Donn; DOBBS, Joshua. **Android Application Development For Dummies**. Indianapolis: Wiley Publishing, 2011.

FRED O'CONNOR. **Smartphones, Tablets Seen Boosting Mobile Health**: Smartphones, tablet PCs and other wireless devices are poised to play a greater role in health care as doctors and patients embrace the mobile Internet, panelists at a mobile health technology conference in Boston said Thursday.. Disponível em:

<http://www.cio.com/article/601263/Smartphones_Tablets_Seen_Boosting_Mobile_Health>. Acesso em: 29 maio 2011.

GARTNER (Egham). **Worldwide Mobile Application Store Revenue Forecast to Surpass $15 Billion in 2011**: Revenue Between 2010 and 2014 is Forecast to Grow 1,000 Percent to Reach $58 Billion. Disponível em: <http://www.gartner.com/it/page.jsp?id=1529214>. Acesso em: 17 maio 2011.

GOOGLE. **Android Developers**. Disponível em: <http://developer.android.com/>. Acesso em: 09 jun. 2012.

GUERRA, Gustavo Pereira. **Desenvolvimento de um aplicativo para iPhone e iPad para acesso a informações médicas em um hospital pervasivo no âmbito do projeto ClinicSpaces**. 2010. 49 f. Monografia (Graduação) - Universidade Federal de Santa Maria, Santa Maria, 2010.

IDF. **IDF Diabetes Atlas**: Diabetes Estimates Excel Tables. 4. ed. Brussels: International Diabetes Federation, 2010. Disponível em: <http://www.idf.org/sites/default/files/DM%202010_7%20regions.xls>. Acesso em: 05 nov. 2011.

GRUMAN, Galen. **The iPad's victory in defining the tablet: What it means**: Apple's view of the tablet is now the accepted model, but one that most commodity competitors still haven't figured out . InfoWorld. Disponível em: <http://www.infoworld.com/d/mobile-technology/the-ipads-victory-in-defining-the-tablet-what-it-means-431>. Acesso em: 12 out. 2011.

ITU. **Key Global Telecom Indicators for the World Telecommunication Service Sector**. Switzerland: 2010. Disponível em: <http://www.itu.int/ITU-D/ict/statistics/at_glance/KeyTelecom.html>. Acesso em: 09 out. 2011.

JOHNSON, Thienne; JOHNSON, Paul. Gerenciamento de Banco de Dados no Android: Criando um aplicativo com uso do SQLite. **Java Magazine**, São Paulo, v. 59, n. , p.38-43, 01 maio 2008.

KAPLAN, Warren A.. Can the ubiquitous power of mobile phones be used to improve health outcomes in developing countries? **Global Health**, Boston, n. , p.2-9, 23 jun. 2006. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1524730>. Acesso em: 30 out. 2011.

KIM, Suk-il; KIM, Hee-seung. Effectiveness of mobile and internet intervention in patients with obese type 2 diabetes. **International Journal Of Medical Informatics**, Seoul, 18 set. 2007. p. 399-404. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1386505607001396>. Acesso em: 08 nov. 2011.

KREGE, Heather. **Web Services Conceptual Architecture (WSCA 1.0)**. IBM Software Group, 2001.

KSOAP. **Ksoap2-android**: A lightweight and efficient SOAP library for the Android platform. Disponível em: <http://code.google.com/p/ksoap2-android/>. Acesso em: 02 jun. 2012.

LEE, Wei-meng. **Beginning Android™ Application Development**. Indianapolis: Wiley Publishing, 2011.

LOPES, Carlos J. Feijó; RAMALHO, José Carlos. **Web Services**: Metodologias de Desenvolvimento. Braga: Universidade do Minho, 2004. Disponível em: <http://repositorium.sdum.uminho.pt/bitstream/1822/559/1/LR04.pdf>. Acesso em: 16 nov. 2011.

MATEUS, Geraldo Robson; LOUREIRO, Antonio Alfredo F.. **Introdução à Computação Móvel**. Rio de Janeiro: Nce/ufrj, 1998. Disponível em: <http://www-di.inf.puc-rio.br/~endler/courses/Mobile/mateus-loureiro-livro.ps>. Acesso em: 12 out. 2011.

MACHADO, Felipe Salles Neves et al. Utilização da telemedicina como estratégia de promoção de saúde em comunidades ribeirinhas da Amazônia: experiência de trabalho interdisciplinar, integrando as diretrizes do SUS. **Ciência & Saúde Coletiva**, Rio de Janeiro, v. 1, n. 15, p.247-254, 01 jan. 2010. Mensal. Disponível em: <http://www.scielo.br/pdf/csc/v15n1/a30v15n1.pdf>. Acesso em: 31 maio 2011.

MARTINS, Victor Manuel Moreira. **Integração de Sistemas de Informação**: Perspectivas, normas e abordagens. 2005. 201 f. Tese (Mestrado) - Universidade do Minho, Guimarães, 2005. Disponível em: <http://repositorium.sdum.uminho.pt/bitstream/1822/5657/3/tese_mestrado_victor_martins_2005.pdf> . Acesso em: 15 nov. 2011.

MEDNIEKS, Zigurd et al. **Programming Android**: Java Programming for the New Generation of Mobile Devices. Sebastopol: O'reilly Media, 2011.

MORIMOTO, Carlos E.. **Uma análise crítica do iPhone**. Disponível em: <http://www.hardware.com.br/analises/iphone/>. Acesso em: 13 out. 2011.

MORRIS, Jason. **Android User Interface Development**: Beginner's Guide. Birmingham: Packt Publishing, 2011.

MOTOROLAMOBILITY. **World's First Commercial Portable Cellular Phone**. Disponível em: <http://motorola-videoleadership.hosted.jivesoftware.com/community/motorola_heritage/timeline>. Acesso em: 08 out. 2011.

OPEN HANDSET ALLIANCE. **Overview**. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 17 maio 2011.

SADZINSKI, Osmani José. **Protótipo de um aplicativo para controle de medicamentos baseado nos padrões de estrutura da certificação de software da sociedade brasileira de informática e saúde e conselho federal de medicina**. 2010. 88 f. Dissertação (Graduação) - Curso de Ciência Da Computação, Universidade Do Extremo Sul Catarinense, Criciúma, 2010. Disponível em: <http://www.kiron.unesc.net/tcc/arquivos/trabalhos/244.pdf>. Acesso em: 26 out. 2011.

PC MAGAZINE. **Definition of: Smartphone**. San Francisco, 2011. Disponível em: <http://www.pcmag.com/encyclopedia_term/0%2C2542%2Ct%3DSmartphone&i%3D51537.asp>. Acesso em: 12 out. 2011.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android para desenvolvedores**. Rio de Janeiro: Brasport, 2009. 195 p.

POTTS, Stephen; KOPACK, Mike. **Aprenda em 24 horas Web Services**. Rio de Janeiro: Campus, 2003. Tradução de: Marcos Vieira.

PRONTUÁRIO NO CELULAR: Telefone armazena dados do paciente para visitas domiciliares. Pesquisa Fapesp: Fapesp, n. 181, 01 mar. 2011. Mensal. Disponível em: <http://www.revistapesquisa.fapesp.br/?ed=945&lg=>. Acesso em: 29 maio 2011.

RAMOS, Bernardo; MEIRELLES, Bruno. Um toque de precisão. **Revista DNA**: Hospital São Luiz, Morumbi, n. 17, p.1-4, 12 out. 2011. Disponível em: <http://www.saoluiz.com.br/sobre_o_sao_luiz/paginas/Revista_DNA/materia/Revista_DNA_017/Um_toque_de_precis%C3%A3o_-

_Tablets_como_o_iPad_p%C3%B5em_tecnologia_m%C3%A9dica_na_ponta_dos_dedos.aspx>. Acesso em: 12 out. 2011.

REHM, Marcus Vinicius Galvão. **Sistema de Apoio ao Diagnóstico Médico utilizando Tecnologias Móveis**. 2005. 59 f. Monografia (Graduação) - Universidade Federal da Bahia, Bahia, 2005.

ROCHA, Álvaro. **Reflexão sobre a Informática de Saúde**. GIMED - Grupo de I&D em Informática Médica, Universidade Fernando Pessoa. Disponível em: <http://www.i-gov.org/index.php?article=2720&visual=1&id=154&subject=213>. Acesso em: 26 out. 2011.

RIEHLE, Dirk. **Framework Design**: A Role Modeling Approach. 2000. 212 f. Dissertação (Phd) - Swiss Federal Institute Of Technology Zurich, Zürich, 2010. Cap. 1. Disponível em: <http://www.riehle.org/computer-science/research/dissertation/diss-a4.pdf>. Acesso em: 01 out. 2011.

SBIS. **O que é Informática em Saúde ?** Disponível em: <http://www.sbis.org.br/>. Acesso em: 26 out. 2011.

SCHMITT JUNIOR, Arno José. **Protótipo de FRONT END de controle de acesso usando J2ME**. 2005. 69 f. Dissertação (Bacharelado) - Curso de Ciências Da Computação, Universidade Regional De Blumenau, Blumenau, 2004. Disponível em: <http://www.bc.furb.br/docs/MO/2004/286677_1_1.pdf>. Acesso em: 09 out. 2011.

SOUZA, Verena. **MV Sistemas investe R$ 2 milhões em mobilidade para saúde**. Disponível em: <http://saudeweb.com.br/17044/mv-sistemas-investe-r-2-milhoes-em-mobilidade-para-saude/>. Acesso em: 30 out. 2011.

SQLITE. **About SQLite**. Disponível em: <http://www.sqlite.org/about.html>. Acesso em: 04 out. 2011.

STEELE, James; TO, Nelson. **The Android Developer's Cookbook**: Building Applications with the Android SDK. Boston: Pearson Education, 2011.

VAUGHAN-NICHOLS, Steven J. **Linus Torvalds on Android, the Linux fork**: There's still a lot of distance between Google's Android and its parent operating system Linux, but eventually, the gap will close… eventually.. Disponível em: <http://www.zdnet.com/blog/open-source/linus-torvalds-on-android-the-linux-fork/9426>. Acesso em: 02 out. 2011.

WECHSLER, Rudolf et al. A informática no consultório médico. **Jornal de Pediatria**, São Paulo, p. S3-S12. 1 jan. 2003. Disponível em: <www.scielo.br/pdf/jped/v79s1/v79s1a02.pdf>. Acesso em: 26 out. 2011.

WEN, Chao Lung. **Telemedicina e Telessaúde**: Um panorama no Brasil. Disponível em: <http://www.ip.pbh.gov.br/ANO10_N2_PDF/telemedicina_telesaude.pdf>. Acesso em: 23 maio 2011.

WHO. **Telemedicine**: Opportunities and developments in Member State. Switzerland: Who Library Cataloguing-in-publication Data, 2010. 2 v. (Global Observatory for eHealth series). Disponível em: <http://whqlibdoc.who.int/publications/2010/9789241564144_eng.pdf>. Acesso em: 27 out. 2011.

APÊNDICE A – ARQUIVO WSDL DO WEB SERVICE DIABETES CONTROL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.1.1-b09
(branches/2.1-6834; 2011-07-16T17:14:48+0000) JAXWS-RI/2.2.5-promoted-b04
JAXWS/2.2. -->
<definitions targetNamespace="http://servico.diabetes.com/" name="DiabetesWS"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:tns="http://servico.diabetes.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
 <types>
  <xsd:schema>
   <xsd:import namespace="http://servico.diabetes.com/"
schemaLocation="DiabetesWS_schema1.xsd"/>
  </xsd:schema>
  <xsd:schema>
   <xsd:import namespace="http://jaxb.dev.java.net/array"
schemaLocation="DiabetesWS_schema2.xsd"/>
  </xsd:schema>
 </types>
 <message name="addRegistro">
  <part name="parameters" element="tns:addRegistro"/>
 </message>
 <message name="addRegistroResponse">
  <part name="parameters" element="tns:addRegistroResponse"/>
 </message>
 <message name="cadNotaMedico">
  <part name="parameters" element="tns:cadNotaMedico"/>
 </message>
 <message name="cadNotaMedicoResponse">
  <part name="parameters" element="tns:cadNotaMedicoResponse"/>
 </message>
 <message name="addPacienteMedico">
  <part name="parameters" element="tns:addPacienteMedico"/>
 </message>
 <message name="addPacienteMedicoResponse">
  <part name="parameters" element="tns:addPacienteMedicoResponse"/>
 </message>
 <message name="cadPaciente">
  <part name="parameters" element="tns:cadPaciente"/>
 </message>
 <message name="cadPacienteResponse">
  <part name="parameters" element="tns:cadPacienteResponse"/>
 </message>
 <message name="addNotaRegistro">
  <part name="parameters" element="tns:addNotaRegistro"/>
```

```
 </message>
 <message name="addNotaRegistroResponse">
  <part name="parameters" element="tns:addNotaRegistroResponse"/>
 </message>
 <message name="addMedico">
  <part name="parameters" element="tns:addMedico"/>
 </message>
 <message name="addMedicoResponse">
  <part name="parameters" element="tns:addMedicoResponse"/>
 </message>
 <message name="getRegistrosPaciente">
  <part name="parameters" element="tns:getRegistrosPaciente"/>
 </message>
 <message name="getRegistrosPacienteResponse">
  <part name="parameters" element="tns:getRegistrosPacienteResponse"/>
 </message>
 <message name="getNotasMedicas">
  <part name="parameters" element="tns:getNotasMedicas"/>
 </message>
 <message name="getNotasMedicasResponse">
  <part name="parameters" element="tns:getNotasMedicasResponse"/>
 </message>
 <portType name="DiabetesWS">
  <operation name="addRegistro">
   <input wsam:Action="http://servico.diabetes.com/DiabetesWS/addRegistroRequest"
message="tns:addRegistro"/>
   <output wsam:Action="http://servico.diabetes.com/DiabetesWS/addRegistroResponse"
message="tns:addRegistroResponse"/>
  </operation>
  <operation name="cadNotaMedico">
   <input wsam:Action="http://servico.diabetes.com/DiabetesWS/cadNotaMedicoRequest"
message="tns:cadNotaMedico"/>
   <output
wsam:Action="http://servico.diabetes.com/DiabetesWS/cadNotaMedicoResponse"
message="tns:cadNotaMedicoResponse"/>
  </operation>
  <operation name="addPacienteMedico">
   <input
wsam:Action="http://servico.diabetes.com/DiabetesWS/addPacienteMedicoRequest"
message="tns:addPacienteMedico"/>
   <output
wsam:Action="http://servico.diabetes.com/DiabetesWS/addPacienteMedicoResponse"
message="tns:addPacienteMedicoResponse"/>
  </operation>
  <operation name="cadPaciente">
   <input wsam:Action="http://servico.diabetes.com/DiabetesWS/cadPacienteRequest"
message="tns:cadPaciente"/>
   <output wsam:Action="http://servico.diabetes.com/DiabetesWS/cadPacienteResponse"
message="tns:cadPacienteResponse"/>
  </operation>
```

```
   <operation name="addNotaRegistro">
     <input wsam:Action="http://servico.diabetes.com/DiabetesWS/addNotaRegistroRequest"
message="tns:addNotaRegistro"/>
     <output
wsam:Action="http://servico.diabetes.com/DiabetesWS/addNotaRegistroResponse"
message="tns:addNotaRegistroResponse"/>
   </operation>
   <operation name="addMedico">
     <input wsam:Action="http://servico.diabetes.com/DiabetesWS/addMedicoRequest"
message="tns:addMedico"/>
     <output wsam:Action="http://servico.diabetes.com/DiabetesWS/addMedicoResponse"
message="tns:addMedicoResponse"/>
   </operation>
   <operation name="getRegistrosPaciente">
     <input
wsam:Action="http://servico.diabetes.com/DiabetesWS/getRegistrosPacienteRequest"
message="tns:getRegistrosPaciente"/>
     <output
wsam:Action="http://servico.diabetes.com/DiabetesWS/getRegistrosPacienteResponse"
message="tns:getRegistrosPacienteResponse"/>
   </operation>
   <operation name="getNotasMedicas">
     <input wsam:Action="http://servico.diabetes.com/DiabetesWS/getNotasMedicasRequest"
message="tns:getNotasMedicas"/>
     <output
wsam:Action="http://servico.diabetes.com/DiabetesWS/getNotasMedicasResponse"
message="tns:getNotasMedicasResponse"/>
   </operation>
 </portType>
 <binding name="DiabetesWSPortBinding" type="tns:DiabetesWS">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
   <operation name="addRegistro">
     <soap:operation soapAction=""/>
     <input>
       <soap:body use="literal"/>
     </input>
     <output>
       <soap:body use="literal"/>
     </output>
   </operation>
   <operation name="cadNotaMedico">
     <soap:operation soapAction=""/>
     <input>
       <soap:body use="literal"/>
     </input>
     <output>
       <soap:body use="literal"/>
     </output>
   </operation>
   <operation name="addPacienteMedico">
```

```
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="cadPaciente">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="addNotaRegistro">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="addMedico">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getRegistrosPaciente">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="getNotasMedicas">
  <soap:operation soapAction=""/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
```

```
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="DiabetesWS">
    <port name="DiabetesWSPort" binding="tns:DiabetesWSPortBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
  </service>
</definitions>
```

APÊNDICE B – CLASSE DIABETESWS COM ANOTAÇÕES JAX-WS

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.diabetes.servico;

import com.diabetes.model.Medico;
import com.diabetes.model.NotaRegistroMedico;
import com.diabetes.model.Paciente;
import com.diabetes.model.Registro;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.ejb.Stateless;

/**
 *
 * @author Leonardo
 */
@WebService(serviceName = "DiabetesWS")
@Stateless()
public class DiabetesWS {

        /**
         * Operação de serviço web
         */
        @WebMethod(operationName = "addRegistro")
        public String addRegistro(@WebParam(name = "tipo") String tipo,
                        @WebParam(name = "categoria") String categoria,
                        @WebParam(name = "valor") Float valor,
                        @WebParam(name = "datahora") Date datahora,
                        @WebParam(name = "codpaciente") String codpaciente,
                        @WebParam(name = "unidade") String unidade,
                        @WebParam(name = "idcelular") Integer idcelular,
                        @WebParam(name = "idmedicamento") Integer idmedicamento,
                        @WebParam(name = "valorpressao") String valorpressao) {
                try {
                        new Registro(null, tipo, categoria, valor, datahora, null,
                                        codpaciente, unidade, idcelular, "N", valorpressao,
                                        idmedicamento).salvar();
                } catch (Exception e) {
                        e.getStackTrace();
                        return "error";
                }
                return "sucess";
```

```
}

/**
 * Operação de serviço web
 */
@WebMethod(operationName = "cadNotaMedico")
public String cadNotaMedico(@WebParam(name = "descricao") String descricao,
                    @WebParam(name = "infoRegistro") String infoRegistro,
                    @WebParam(name = "codPaciente") String codPaciente,
                    @WebParam(name = "idCelular") Integer idCelular) {
          try {
                    new NotaRegistroMedico(null, descricao, "N", infoRegistro,
                              codPaciente, idCelular).salvar();
          } catch (Exception e) {
                    e.getStackTrace();
                    return "error";
          }
          return "sucess";
}

/**
 * Operação de serviço web
 */
@WebMethod(operationName = "addPacienteMedico")
public String[] addPacienteMedico(
                    @WebParam(name = "codPaciente") String codPaciente,
                    @WebParam(name = "senhaPaciente") String senhaPaciente) {
          String[] mensagens = new String[10];
          try {
                    Paciente pac = new Paciente().getPacienteCodSenha(codPaciente,
                              senhaPaciente);
                    if (pac == null) {
                              mensagens[0] = "Verifique senha e usuário do Paciente!";
                    } else {
                              // Parâmetros passados para construção do Paciente no Android!
                              mensagens[0] = "sucess";
                              mensagens[1] = pac.getNomePac();
                              mensagens[2] = pac.getEmailPac();
                              mensagens[3] = pac.getCodPac();
                    }
                    return mensagens;
          } catch (Exception e) {
                    e.printStackTrace();
                    mensagens[0] = "Erro ao buscar senha e usuário do Paciente!";
                    return mensagens;
          }
}

/**
 * Operação de serviço web
```

```
     */
    @WebMethod(operationName = "cadPaciente")
    public String cadPaciente(@WebParam(name = "nomePac") String nomePac,
                    @WebParam(name = "emailPac") String emailPac,
                    @WebParam(name = "codPaciente") String codPaciente,
                    @WebParam(name = "sexoPac") String sexoPac,
                    @WebParam(name = "nascimentoPac") Date nascimentoPac,
                    @WebParam(name = "senhaPac") String senhaPac,
                    @WebParam(name = "update") String update) {
            try {
                    if ("S".equals(update)) {
                            new Paciente(codPaciente, nomePac, sexoPac, senhaPac,
emailPac,
                                            nascimentoPac).salvar();
                    } else {
                            Paciente pac = new Paciente();
                            if (pac.getPaciente(codPaciente) != null) {
                                    return "duplicado";
                            } else {
                                    new Paciente(codPaciente, nomePac, sexoPac, senhaPac,
                                                    emailPac, nascimentoPac).salvar();
                            }
                    }
            } catch (Exception ex) {
                    ex.printStackTrace();
                    return "error";
            }
            return "sucess";
    }

    /**
     * Operação de serviço web
     */
    @WebMethod(operationName = "addNotaRegistro")
    public String addNotaRegistro() {
            // TODO write your implementation code here:
            return null;
    }

    /**
     * Operação de serviço web
     */
    @WebMethod(operationName = "addMedico")
    public String addMedico(@WebParam(name = "nomeMed") String nomeMed,
                    @WebParam(name = "emailMed") String emailMed,
                    @WebParam(name = "codMedico") String codMedico,
                    @WebParam(name = "registroMed") String registroMed) {
            try {
                    new Medico(nomeMed, emailMed, nomeMed, codMedico).salvar();
            } catch (Exception e) {
```

```
                e.getStackTrace();
                return e.getMessage().toString();
            }
            return "Sucesso";
        }

        /**
         * Operação de serviço web
         */
        @WebMethod(operationName = "getRegistrosPaciente")
        public List<String[]> getRegistrosPaciente(
                @WebParam(name = "codPaciente") String codPaciente) {
            List<String[]> listString = new ArrayList<String[]>();
            String[] mensagens = null;
            try {
                List<Registro> list = new Registro().getRegistros(codPaciente);
                if (list == null) {
                    mensagens = new String[1];
                    mensagens[0] = "Sem Registros!";
                    listString.add(mensagens);
                } else {
                    for (Registro item : list) {
                        mensagens = new String[9];
                        mensagens[0] = "sucess";
                        if (item.getValor() != null) {
                            mensagens[1] = item.getValor().toString();
                        } else {
                            mensagens[1] = item.getValorpressao();
                        }
                        mensagens[2] = item.getCategoria();
                        mensagens[3] = item.getTipo();
                        mensagens[4] = item.getCodPaciente();
                        mensagens[5] = item.getUnidade();
                        mensagens[6] = item.getIdCelular().toString();
                        mensagens[7] =
String.valueOf(item.getDatahora().getTime());
                        if (item.getIdMedicamento() != null) {
                            mensagens[8] =
item.getIdMedicamento().toString();
                        } else {
                            mensagens[8] = null;
                        }
                        listString.add(mensagens);
                        item.setSincronizado("S");
                        new Registro().update(item);
                    }
                }
                return listString;
            } catch (Exception e) {
                e.getStackTrace();
```

```
                e.printStackTrace();
                mensagens[0] = "Erro ao buscar Registros!";
                listString.add(mensagens);
                return listString;
            }
        }

        @WebMethod(operationName = "getNotasMedicas")
        public List<String[]> getNotasMedicas(
                @WebParam(name = "codPaciente") String codPaciente) {
            List<String[]> listString = new ArrayList<String[]>();
            String[] mensagens = null;
            try {
                List<NotaRegistroMedico> list = new NotaRegistroMedico()
                        .getNotas(codPaciente);
                if (list == null) {
                    mensagens = new String[1];
                    mensagens[0] = "Sem Notas!";
                    listString.add(mensagens);
                } else {
                    for (NotaRegistroMedico item : list) {
                        mensagens = new String[5];
                        mensagens[0] = "sucess";
                        mensagens[1] = item.getDescricao();
                        mensagens[2] = item.getInfoRegistro();
                        mensagens[3] = item.getCodPaciente();
                        mensagens[4] = item.getIdCelular().toString();
                        listString.add(mensagens);
                        item.setSincronizado("S");
                        new NotaRegistroMedico().update(item);
                    }
                }
                return listString;
            } catch (Exception e) {
                e.getStackTrace();
                e.printStackTrace();
                mensagens[0] = "Erro ao buscar Notas!";
                listString.add(mensagens);
                return listString;
            }
        }
}
```

APÊNDICE C – CLASSE CADREGISTROWS USANDO KSOAP2

```
package com.diabetescontrol.webservice;

import java.util.Date;

import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;

import com.diabetescontrol.database.RegistroDAO;
import com.diabetescontrol.model.Registro;
import com.diabetescontrol.util.Utils;

public class CadRegistroWS {
	private String namespace = "http://servico.diabetes.com/";
	private String method_name = "";
	private Registro registro;
	private Context ctx;
	private SoapObject request;
	private String soap_action = "";
	private String url = "";

	public CadRegistroWS(Context ctx) {
		this.ctx = ctx;
		this.url = Utils.URL_WS(ctx);
	}

	public void sincRegistro(Registro reg) {
		Date datahora = new Date(reg.getDataHora().getTime());
		this.method_name = "addRegistro";
		this.request = new SoapObject(namespace, method_name);
		registro = reg;
		request.addProperty("tipo", reg.getTipo());
		request.addProperty("categoria", reg.getCategoria());
		request.addProperty("valor", reg.getValor());
		request.addProperty("datahora", datahora);
		request.addProperty("codpaciente", reg.getCodPaciente());
		request.addProperty("unidade", reg.getUnidade());
		request.addProperty("idcelular", reg.getId());
		request.addProperty("idmedicamento", reg.getMedicamento());
		request.addProperty("valorpressao", reg.getValorPressao());
		new servicoAsyncTask().execute();
	}

	class servicoAsyncTask extends AsyncTask<Void, Void, Void> {
		private ProgressDialog progressDialog;
```

```
		@Override
		protected Void doInBackground(Void... params) {
			String resultado = getRegistrosService();
			if ("sucess".equals(resultado)) {
				RegistroDAO regDao = new RegistroDAO(ctx);
				regDao.open();
				registro.setSincronizado("S");
				regDao.atualizaRegistro(registro);
				regDao.close();
			}
			return null;
		}

		@Override
		protected void onPreExecute() {
			progressDialog = new ProgressDialog(ctx);
			progressDialog.setMessage("Sincronizando");
			progressDialog.show();
		}

		@Override
		protected void onPostExecute(Void result) {
			progressDialog.cancel();
		}
	}

	private String getRegistrosService() {
		SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
				SoapEnvelope.VER11);
		MarshalDate md = new MarshalDate();
		MarshalFloat mf = new MarshalFloat();
		mf.register(envelope);
		md.register(envelope);
		envelope.setOutputSoapObject(request);
		try {
			HttpTransportSE androidHttpTransport = new HttpTransportSE(url);
			androidHttpTransport.call(soap_action, envelope);
			SoapPrimitive result = (SoapPrimitive) envelope.getResponse();
			return result.toString();
		} catch (Exception ex) {
			ex.printStackTrace();
		}
		return null;
	}
}
```

APÊNDICE D – ARTIGO

Diabetes Control: Uma aplicação mobile aplicada ao gerenciamento de informações médicas referentes ao controle do diabetes

Leonardo A. Neuwald[1], Gustavo Bisognin[2], Fábio B. Goularte[2]

[1]Acadêmico do Curso de Ciência da Computação – Departamento de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brazil

[2]Professor do Curso de Ciência da Computação – Departamento de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC - Brazil

{leo.alvesneuwald,gbisog}@gmail.com, fbg@unesc.net

Abstract. *This paper describes the conclusion work submitted to obtain the degree of bachelor in Computer Science at the UNESC. The work sought the use of technologies such as JAX-WS, Java and Android to create a solution that would allow control of the population diabetes. This way, we built a model that allows the transmission of patients diabetes information, generated by a mobile device, to a mobile device of the doctor's who is leading this treatment through a Web Service.*

Resumo. *O presente artigo descreve o trabalho de conclusão de curso apresentado para obtenção do grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense. O trabalho buscou a utilização de tecnologias como JAX-WS, Java e Android para a criação de uma solução que possibilitasse o controle da diabetes da população. Desta forma, foi construído um modelo que permite a transmissão das informações da diabetes do paciente, geradas por um dispositivo móvel Android, para o dispositivo móvel do médico que está acompanhando esse tratamento através de um Web Service.*

1. INTRODUÇÃO

Estima-se que existam atualmente no mundo, 1.500 milhões de televisores em uso, mais de 1 bilhão de pessoas estão conectados a Internet e quase 3 bilhões de pessoas têm um telefone celular, tornando-o um dos produtos de consumo de maior sucesso atualmente (OPEN HANDSET ALLIANCE, 2011).

O número de smartphones vendidos em 2010 foi 71% maior do que o vendido em 2009, algo em torno de 302 milhões de novos dispositivos. Segundo a mesma pesquisa, o número de aparelhos com sistema operacional Android era de 69 milhões de dispositivos em 2010 e possui uma tendência a aumentar proporcionalmente, sendo que em 2016 o Android deve estar em 45% dos smartphones ao redor do mundo, o que torna este mercado muito lucrativo (ABI RESEARCH, 2011, tradução nossa).

Hoje, um dos maiores desafios que a mobilidade dos softwares para saúde deve superar é na área de Telemedicina. O termo Telemedicina é apresentado na literatura desde a década de 60, e vem sendo adequada e aprimorada com o surgimento de novas tecnologias e necessidades relacionadas à saúde. Segundo Wen (2011) todas as definições de Telemedicina

apontam para a possibilidade de proporcionar cuidados médicos em situações onde a distância é considerada um dos maiores fatores críticos.

Utilizando-se da sincronização de sistemas, deve ser possível ao médico, coletar informações de cuidados ao paciente em um dispositivo móvel mesmo que esteja fora de seu estabelecimento de saúde. Em atendimentos fora de seu estabelecimento, e até mesmo em algumas situações dentro do estabelecimento, o profissional de saúde pode não possuir acesso a uma rede de comunicação no momento em que estiver coletando informações do paciente. Nesta situação é necessário que o profissional possa coletar os dados utilizando um aplicativo que permita realizar a transmissão de dados para outro sistema somente quanto possuir acesso a rede, assim não prejudicando a coleta das informações.

Diante disto, será realizado o desenvolvimento de uma solução móvel com tecnologia Android, em que seja possível controlar os níveis de diabetes de um paciente independentemente do acesso a internet. No modelo, o sistema deve se comunicar com um módulo instalado no dispositivo de um profissional da saúde, assim possibilitando o envio e recebimento de informações através da sincronização de dados.

2. Google Android

O Android é um conjunto de sistema operacional, middleware e softwares aplicativos para dispositivos móveis. O Software Development Kit (SDK) da Android dispõe dentre diversas ferramentas de um Application Programming Interface (API) para que seja possível criar aplicações utilizando Java como linguagem de programação (GOOGLE, 2012, tradução nossa).

Antes do sistema operacional da Google, o desenvolvimento de aplicações móveis era muito fragmentado existindo uma variedade de plataformas para desenvolvimento voltado à dispositivos móveis. Existiam dispositivos rodando com diversos sistemas operacionais como Palm OS, RIM BlackBerry OS, Java Micro Edition (ME), Symbian OS, iPhone OS dentre outros, o que fragmentava ainda mais o desenvolvimento de aplicativos para dispositivos móveis, pois não era possível escolher somente uma plataforma de desenvolvimento (CONDER; DARCEY, 2010, tradução nossa).

O sistema operacional Android foi construído dentro de um paradigma mobile totalmente voltado a web e integrado com diversos aplicativos e ferramentas da Google. Neste contexto, o SO apresenta-se para o mercado como um contêiner onde diversas aplicações podem ser facilmente baixadas da internet e instaladas, abordando os mais diversos domínios do conhecimento (MORRIS, 2011, tradução nossa).

Uma das principais características apresentadas pelo SO Android, é sua facilidade de uso, apresentando uma interface intuitiva com ícones grandes para suportar o formato touch screen. Outra característica importante é a alta portabilidade e desempenho apresentada pelo kernel podendo ser instalado em diversos aparelhos como gadgets, smartfones, tablets, eletrodomésticos e televisores (MORRIS, 2011, tradução nossa; STEELE; TO, 2011, tradução nossa).

A construção do Android é baseada no kernel do sistema operacional Linux 2.6, apresentando uma arquitetura robusta e estável, concedendo maior eficiência e confiabilidade as camadas de alto nível da aplicação. Uma representação gráfica das cinco camadas da arquitetura Android, pode ser observada na Figura 1 (GOOGLE, 2012, tradução nossa).

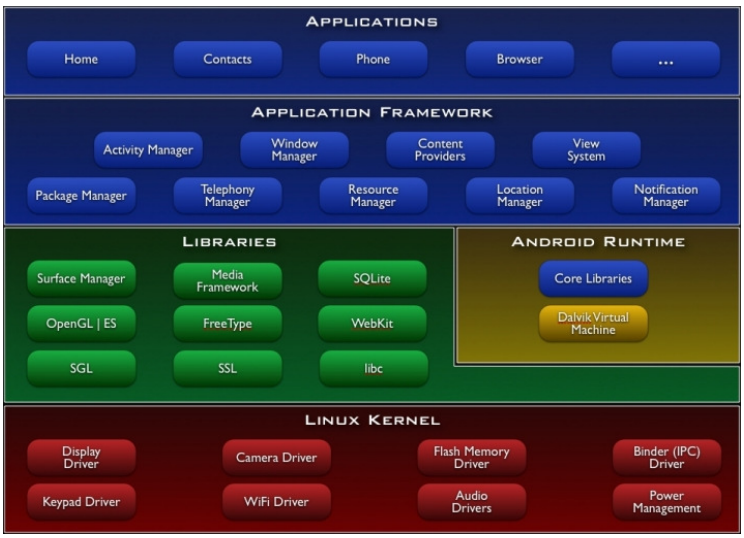


Figura 1 – Arquitetura Android

Conforme observado na Figura 1, o SO Android é subdividido nas seguintes partes básicas: Linux Kernel, Libraries, Android Runtime, Aplication Framework e Applications. Uma visão detalhada de cada uma destas partes é apresentada nos tópicos a seguir.

Na camada de aplicações é onde se encontram todas as aplicações fundamentais do SO. Já na camada de framework é disponibilizada a API e os recursos utilizados pelos aplicativos como as classes virtuais, provedores de conteúdos entre outros. Acima do kernel do Linux 2.6 se encontram diversas funcinalidades disponibilizadas a desenvolvedores, como bibliotecas C/C++ usadas para conexão com o SQLite ou criação de gráficos. Outra camada é a camada de runtime, onde ao executar uma aplicação Android, é criada uma instancia da Dalvik Vitual Machine (DVM) em ambiente de execução. Por fim, o Android se baseia no kernel do Linux 2.6 apresentando uma arquitetura robusta e estável, concedendo maior eficiência e confiabilidade as camadas de alto nível da aplicação.

3. Sincronização de Sistemas

A necessidade de integração e sincronização dos Sistemas de Informação (SI) está ligada a evolução das organizações, dos mercados e tecnologia. Atualmente é necessário compartilhar a informação existente nos SI para a Internet, bem como compartilhar informações entre sistemas diversos e iguais para geração de soluções e resultados (MARTINS, 2005).

O aparecimento da Internet criou uma necessidade maior de integrar informações, fazendo os gestores de Tecnologia da Informação (TI) optassem por estratégias e soluções globais para suprir esta necessidade. Tecnologias como Service Oriented Architecture (SOA) e Web Services (WS) apareceram como soluções para suportar a integração e sincronização dos SI, sendo válidas conforme os objetivos que a organização ou sistema propõem (MARTINS, 2005).

3.1. Web Services

A tecnologia dos WS é uma importante forma de integração de dados entre diferentes plataformas e programas. Por isso, um WS possui como características: estar disponível através da Internet ou uma rede, utilizar um sistema de troca de mensagem através de XML, não ser dependente de sistemas operacionais ou linguagens de programação, ser auto-descritivo através da linguagem XML e ser encontrado via mecanismos de busca (CERAMI, 2002, tradução nossa).

Os WS utilizam os protocolos HTTP e HTTPS, já conhecidos na web. Os serviços trafegam informações do cliente para o servidor utilizando documentos linguagem de marcação extensiva (XML), ou eXtensible Markup Language que é uma linguagem de descrição caracterizada por tags. A principal vantagem da utilização da linguagem XML é que SI podem se comunicar utilizando diferentes tecnologias (CERAMI, 2002, tradução nossa).

Serviços web promovem a troca de informações permitindo a publicação de rotinas e métodos acessíveis pela Internet. Por meio de uma interface transparente para o usuário é possível realizar a integração de dados entre aplicações distintas. Aplicações que utilizam a verificação de cartão de crédito, rastreamento de pacotes, acompanhamento de carteira, conversão de moeda e tradução de idiomas entre outros são exemplos de trocas de informações utilizando os serviços web (CERAMI, 2002, tradução nossa).

Três componentes são considerados núcleos da arquitetura típica de um Web Service que utiliza o XML como linguagem base. Os componentes, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) e Universal Discovery, Description, and Integration (UDDI), também referenciados como WUST (WSDL, UDDI, SOAP Techonologies) formam o núcleo da tecnologia (ABINADER; LINS, 2006; POTTS; KOPACK, 2003).

4. Informática em Saúde

A área da informática em saúde é definida como um campo de rápido desenvolvimento científico atuando principalmente no armazenamento, recuperação e uso da informação, dados e conhecimentos biomédicos para a resolução de problemas e tomadas de decisões. Assuntos sobre informática médica estão se tornando mais importantes proporcionando a modernização e melhoria da prestação de cuidados de saúde, através de uma melhor gestão da informação de saúde como dos recursos associados (BLOIS; SHORTLIFFE, 1990, tradução nossa; ROCHA, 2011).

Dados e conhecimentos gerenciados por meio de métodos tradicionais como o papel se tornam obsoletos principalmente devido ao grande número de informações médicas que devem ser processadas para se chegar a uma decisão. Esse é considerado o principal motivo do crescimento da área da informática em saúde, que aliada aos avanços nas tecnologias da computação, comunicação e conhecimento médico fazem com que a Informática Médica desempenhe papel central na medicina moderna. Como retorno a esse crescimento ouve uma divisão dos sistemas de saúde em diversas áreas como: Sistemas de informação em saúde, Prontuário Eletrônico do Paciente (PEP), Sistemas de Apoio à Decisão (SAD), Processamento de Sinais Biológicos, Processamento de imagens médicas, Padronização da informação em saúde, Telemedicina entre outras (SADZINSKI, 2010; ROCHA, 2011; SBIS, 2011).

Atualmente uma das áreas da informática em saúde que mais se destaca é a Telemedicina, impulsionada pelas tecnologias que permitem a comunicação entre computadores, smartphones, tablets entre outros dispositivos. Essas tecnologias aliadas a

Telemedicina possuem grande potencial para solucionar alguns dos problemas de saúde mundial (WHO, 2010, tradução nossa).

Section titles must be in boldface, 13pt, flush left. There should be an extra 12 pt of space before each title. Section numbering is optional. The first paragraph of each section should not be indented, while the first lines of subsequent paragraphs should be indented by 1.27 cm.

4.1. Telemedicina

Segundo a WHO (2010), o termo Telemedicina, literalmente significando medicina a distância, possui mais de 104 definições no mundo. A Word Health Orgnization (WHO) define a Telemedicina como a disposição de serviços de saúde, à distância, por profissionais de saúde usando tecnologias e a comunicação para o intercâmbio de informações para diagnóstico, tratamento e prevenção à saúde da população.

Em países onde a existem deficiências no cuidado a saúde, a Telemedicina pode ajudar a atender as necessidades não supridas por outros métodos tradicionais de cuidados a saúde de pacientes. Locais com difícil acesso é possível obter o atendimento de um médico especialista através de uma videoconferência, evitando a necessidade do deslocamento do profissional de saúde a áreas remotas, onde o custo de deslocamento é alto. Outra grande vantagem da utilização da Telemedicina é o acompanhamento de pacientes com condições crônicas, como a diabetes. Em situações de tratamento de enfermos com doenças crônicas o profissional de saúde pode acompanhar a evolução do tratamento diariamente, sem a necessidade de estar presencialmente com o paciente. Sem a necessidade de um acompanhamento presencial do profissional de saúde, é possível aumentar o controle no tratamento a enfermidade com uma redução significativa de consultas (WHO, 2010, tradução nossa).

4.2. Diabetes Mellitus (DM)

Relatos vindos de 1500 a.C. falam sobre uma doença que deixava a urina com gosto doce. Essa doença posteriormente viria a se chamar diabetes, nome de origem grega significando sifão, uma analogia as mangueiras utilizadas para transportar líquidos de um recipiente para outro. A origem do nome se refere ao sintomas clássicos, que são: excessivo consumo de líquidos e aumento do volume urinário (CANCELLIÉRI, 1999).

O diabetes é uma disfunção metabólica, originada pelo comprometimento na produção e/ou utilização do hormônio da insulina. O grau de comprometimento de sua produção, de sua ação, do número ou da resposta dos receptores à insulina são indicadores dos dois tipos de diabetes mellitus: diabetes tipo 1 ou mellitus insulino-dependente e o diabetes tipo 2 ou mellitus não insulino-dependente (CANCELLIÉRI, 1999).

Segundo pesquisas da IDF a diabetes mellitus já atinge no mundo mais de 280 milhões de pessoas com idade entre 20 e 79 anos. Somente no Brasil, a DM atinge mais de 7 milhões de pessoas.

5. Diabetes Control – Sistema de controle de diabetes

A presente pesquisa tem como resultado um sistema que realiza o controle de vários indicadores sobre a diabetes dos pacientes. Esse controle é realizado por meio de um sistema mobile que recebe as informações inseridas pelos usuários, gerando e possibilitando controle sobre os mais diversos indicadores.

Com o sistema desenvolvido também é possível trocar informações entre dispositivos de médicos e pacientes, de forma que as informações geradas pelos pacientes são levadas ao

médico que pode acompanhá-las, direcionando um melhor tratamento. Diversas tecnologias e ferramentas foram utilizadas para que o software pudesse alcançar seu propósito, sendo as principais: Web Services, Java, JAX-WS, Android e outras citadas durante a metodologia.

5.1. Metodologia

Para que fosse possível realizar a criação da solução proposta, tanto no ambiente mobile quanto em um serviço da web, tornou-se necessário compreender e definir uma série de tecnologias e ferramentas que iriam auxiliar e firmar o desenvolvimento destas. Agrupando todas as tecnologias utilizadas, podemos citar: Google Android, Google Code, Java, JAX-WS, Eclipse IDE, SQLite, SubEclipse, NetBeans, Oracle, GlassFish e outras que foram usadas pontualmente.

A modelagem do software iniciou considerando o fluxo básico seguindo aplicativo de forma a garantir a troca correta de informações via WS. Dentro desta foram desenvolvidos alguns casos de uso e protótipos de tela, para que fosse possível visualizar um resultado.

Finalmente, após a codificação do aplicativo, foram realizados uma seqüência de testes sobre o aplicativo visando garantir o máximo de qualidade possível para que o mesmo fosse disponibilizado na loja de aplicativos da Google.

5.2. Fluxo de Informações

O sistema desenvolvido, nomeado de Diabetes Control, realiza uma série de trocas de informações entre dispositivos e serviços conforme exemplificado na Figura 2 em seu modelo conceitual:

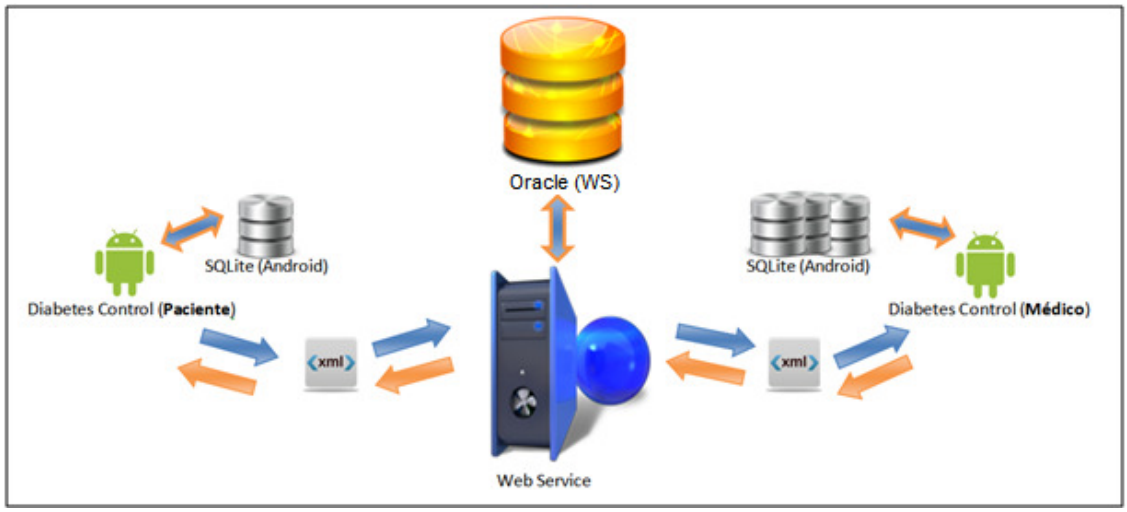


Figura 1 – Arquitetura Android

Conforme é possível visualizar na Figura 9, existem duas aplicações distintas. A aplicação Android na qual pacientes e médicos realizam suas marcações e anotações e o Web Service que sincroniza essas marcações.

No modelo o WS atua como mediador do envio de informações do dispositivo do paciente para o dispositivo do médico. Essa configuração permite que os dados sejam sincronizados entre pacientes e médicos de forma praticamente instantânea, caso os dois dispositivos estejam conectados a internet. Caso um dos dispositivos não possua acesso a internet, o registro não será perdido e a sincronização acontecerá no próximo momento que o dispositivo esteja conectado a internet.

Os XML trocados possuem informações de peso, insulina, diabetes, notas médicas, dentre outros tipos de informações. Quando essas estão disponíveis nos dispositivos, é possível realizar consultas, gerar gráficos e acompanhar médias. Tornando mais simples o acompanhamento da diabetes pelo paciente e médico.

5.3. Modelagem

Um aplicativo bem construído possui como um de seus requisitos mínimos a modelagem. Essa modelagem é necessária para que sejam previstos problemas que podem ocorrer durante a implementação, além de guiar o desenvolvimento do sistema e implementar conceitos de engenharia de software, melhorando assim a qualidade do produto desenvolvido.

Existem diversas técnicas de engenharia presentes no mercado para realizar modelagem de software. Para o sistema Diabetes Control, foram escolhidas aquelas que trariam maior valor agregado para o projeto, bem como melhor auxiliariam durante todo o ciclo de desenvolvimento. Desta forma, foi realizada a modelagem lógica das tabelas do banco de dados e a diagramação dos cenários de casos de uso. Alem disso, também foi utilizado o desenho dos wireframes, auxiliando assim na simulação do fluxo de navegação entre telas e ações do sistema desenvolvido.

A diagramação dos casos de uso foi realizada por intermédio da ferramenta StarUML, por ser disponibilizada gratuitamente e possuir recursos avançados para o desenvolvimento dos diagramas necessários.

O primeiro diagrama disponível na Figura 3 mostra as funcionalidades cujo ator, representando o paciente, pode registrar no sistema do paciente.

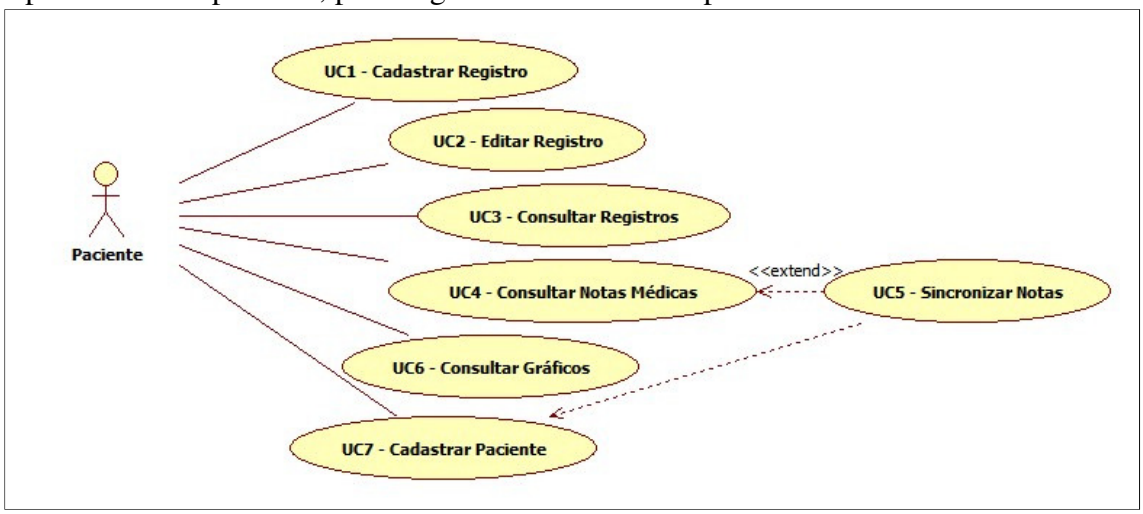


Figura 3 – Diagrama de Casos de Uso do ator Paciente

No diagrama, é possível notar que o caso de uso, Consultar Notas Médicas, estende o Sincronizar Notas, ou seja, ao consultar as notas, pode ser realizada a sincronização das notas com o serviço web. O diagrama apresenta a obrigatoriedade da realização da sincronização das notas, onde, primeiramente é necessário realizar o cadastramento do paciente.

Na Figura 4 é possível verificar o modelo físico de banco de dados presente nos dispositivos móveis de médicos e pacientes. Algumas peculiaridades são notadas no banco utilizado no Android, como o fato de toda chave primária iniciar como _id. Outra diferença deste modelo para um modelo convencional é que algumas tabelas que necessitavam ser criadas para armazenamento de configurações, não foram necessárias devido ao fato do

Android possibilitar a criação das preferências do sistema por meio de um XML de configurações.

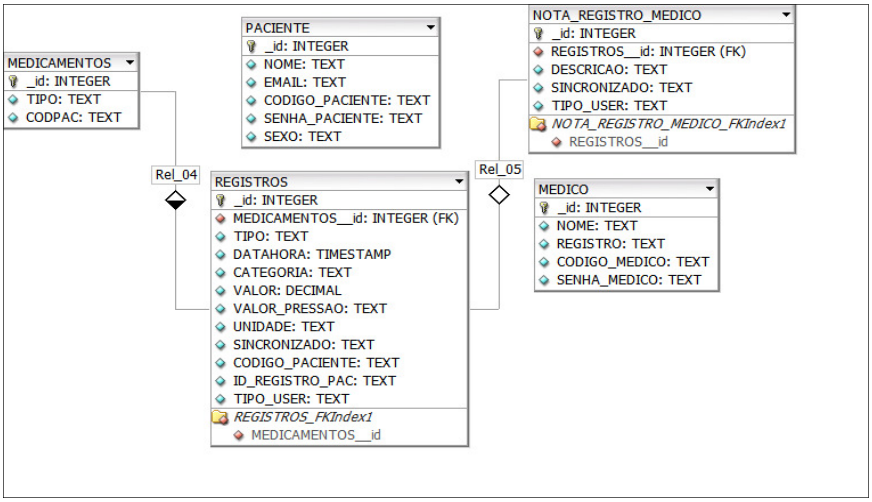


Figura 4 – Modelo físico de banco de dados do dispositivo móvel

A ausência de relacionamentos entre algumas tabelas como MEDICO e NOTA_REGISTRO_MEDICO e PACIENTE e REGISTROS é justificada pelo fato do banco ficar disponível tanto nos dispositivos dos pacientes quantos nos dispositivos dos médicos, e pelo relacionamento se dar como um relacionamento lógico com as tabelas disponíveis no WS.

Foram criadas três wireframes no MockFlow para servirem como base para o desenvolvimento dos XML layout das telas no Android. Na Figura 5 é possível verificar o desenho a página principal utilizando a modelagem da tela em wireframe.

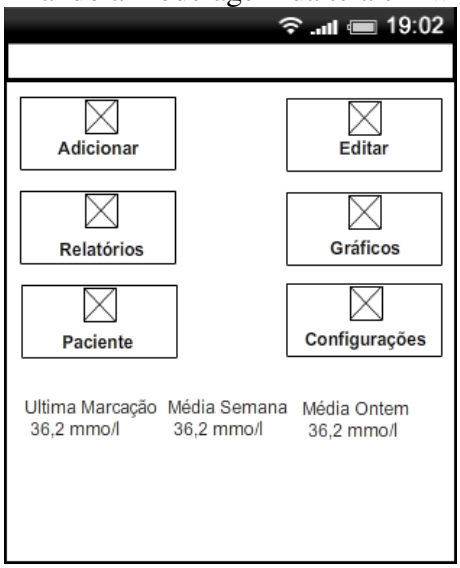


Figura 5 – Wireframe da tela principal

O wireframe da primeira tela apresenta a distribuição dos menus no aplicativo quando esse estivesse no modo paciente. Nesta tela é apresentado os botões de ação para as

funcionalidades de adicionar, editar, alterar configurações, verificar gráficos e relatórios. Também é possível verificar que abaixo dos botões existe a informação com o resumo atual dos registros de glicose criados pelo paciente.

5.4. Implementação

Definidos as tecnologias utilizadas e a modelagem, esta parte apresenta a implementação do aplicativo.

O desenvolvimento foi iniciado pelo aplicativo disponível para os dispositivos móveis, visto que independentemente de um serviço web, o aplicativo deveria sozinho ser capaz de gerenciar as informações relacionadas a diabetes do paciente. O projeto foi desenvolvido seguindo o padrão Model View Controller (MVC), onde os XMLs de configuração de layout representam a view, as activities representam o controller e o modelo é representado pelos objetos.

A versão do Android utilizada foi a versão 2.3 por ter o maior número de dispositivos usando essa versão e por possibilitar que todos usuários que utilizem o Android acima desta versão, também possam utilizar o Diabetes Control.

Foram criadas as classes modelos Medico, NotaRegistroMedico, Paciente e Registro. Esses objetos representam respectivamente, as informações do médico, as notas médicas que os médicos cadastram para os registros dos pacientes, as informações do paciente, o registro gerado pelo paciente sobre a diabetes e o registro médico. Com base nessas classes, foram criadas classes de Data Access Object (DAO), que encapsulam as funções do banco de dados.

Uma das necessidades para que seja possível controlar a diabetes é a necessidade de visualização destas informações em formato de gráficos. Diferentemente de outras funcionalidades, o Android não possui nenhum tipo de biblioteca nativa para o desenvolvimento de gráficos. Algumas outras soluções como o framework AChartEngine, possibilitaram a criação de gráficos. Segundo 4viewsoft (2012), o AChartEngine é um framework open-source para geração de gráficos em dispositivos Android.

Um padrão muito interessante para o desenvolvimento em Android, é a possibilidade de criação de um arquivo de preferências do aplicativo. Esse arquivo é criado em XML, da mesma forma que é criada uma tela, e ele irá guardar todas as preferências utilizadas no restante do projeto. Para o projeto Diabetes Control, as unidades de medidas, a sincronização ou não das informações cadastradas e outros dados importantes e utilizados em diversos locais da aplicação, são salvos neste arquivo de preferências, diminuindo assim a necessidade de criação de algumas tabelas para guardar essas informações. Assim quando é necessário buscar qualquer informação salva basta utilizar um comando, sem a necessidade de abrir uma conexão com o banco e buscar em uma tabela de preferências.

Por fim, o desenvolvimento na plataforma Google Android, se deu com a criação das classes para conexão com o Web Service. Para criação destas classes, foi utilizada a biblioteca Ksoap2 para Android.

Finalizando do desenvolvimento da parte móvel do projeto, foi iniciada a criação do Web Service que iria receber as informações do aplicativo.

O WS foi concebido para receber informações dos registros de diabetes, pacientes cadastrados e notas médicas sobre os registros cadastrados. Como já citado anteriormente, o WS foi desenvolvimento utilizando o JPA EclipseLink, o que facilitou muito a implementação do aplicativo. Com o JPA, não foi necessária a criação de classes DAO, bastando realizar as anotações corretas dentro das classes de modelo para que fosse possível salvar, editar e excluir informações.

O projeto foi divido em três pacotes principais, database, model e serviço. Dentro do pacote database estão as classes que criam as persistências com o banco de dados usando o EclipseLink. As classes do pacote modelo possuem os objetos que são utilizados para persistir os dados e salvar as informações. Por último temos o pacote serviço, que é onde se encontra a classe que cria os métodos para conexão com o WS. O projeto também possui classes geradas pelo JAX-WS com os objetos do WS, bem como um arquivo XML com as configurações necessárias para a criação das tabelas automaticamente utilizando o JPA

5.5. Trabalho desenvolvido

Como resultados foram criadas duas aplicações distintas. O WS, que regula as informações e transações entre dispositivos de paciente e médico e a própria implementação do dispositivo móvel se baseando na tecnologia Android. A aplicação móvel pode ser dividida como aplicação de dois módulos distintos. O módulo paciente, onde as informações são cadastradas pelo paciente e enviadas para o WS, e o módulo do médico, que recebe as informações do paciente e permite a realização de uma análise do médico que pode inclusive cadastrar notas médicas sobre os registros cadastrados pelo paciente.

Na aplicação móvel, está disponível para o paciente uma série de facilidades para o controle de diabetes. Gráficos, relatórios e indicadores em geral que facilitam a interpretação dos dados coletados. O sistema possui uma interface simples e intuitiva para que qualquer usuário independentemente do nível de conhecimento, consiga utilizar o software sem dificuldades, conseguindo tirar o máximo proveito das informações geradas. Na Figura 6, é possível verificar a tela principal do sistema no módulo paciente e médico.

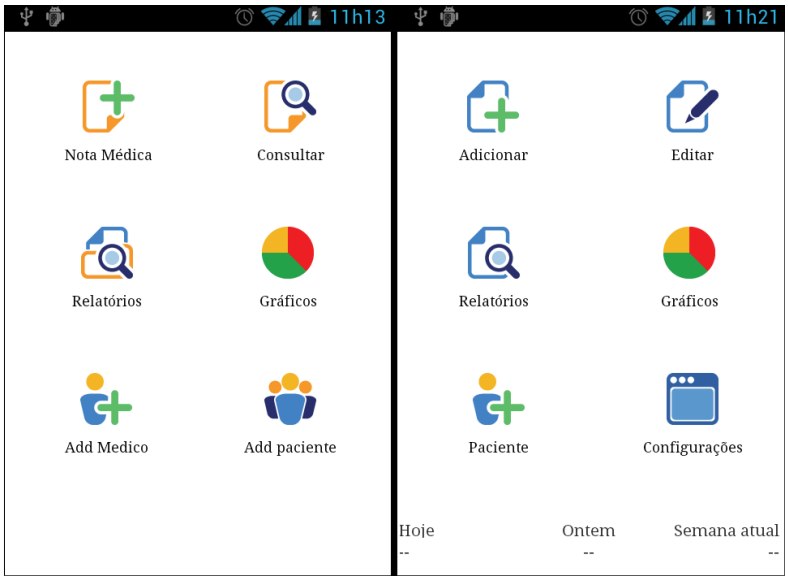


Figura 6 – Telas iniciais dos módulos Médico e Paciente

Através das telas iniciais de Paciente e Medico, é possível acessar as principais funcionalidades do sistema. Os registros do paciente são cadastrados por meio da tela de “Cadastro de Registros” acessada através do botão “Adicionar” na tela inicial do módulo Paciente. Conforme é possível verificar na Figura 7, a tela permite o cadastramento de medicamentos, glicose, peso, pressão, pulso, gordura e HbA1c.

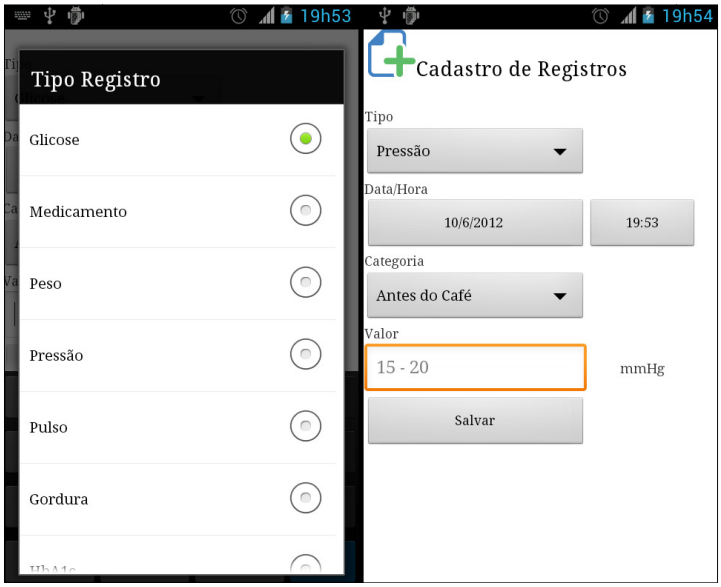


Figura 7 – Seleção de Tipo Registro e Cadastro de Registros

A Figura 7 demonstra as opções de seleção do tipo de registro. Na tela ao lado é apresentado o cadastro do registro com o tipo pressão selecionado. Caso o tipo selecionado fosse medicamento, seria possível selecionar uma opção com os diversos tipos de insulinas para que fosse escolhido um destes. Essa tela é a principal entrada de dados do sistema, sendo que com os dados gerados nesta tela são criados gráficos e relatórios.

Um dos gráficos gerados pelo sistema que auxilia no controle da diabetes é o gráfico de média de glicose por categoria. Conforme é possível verificar na Figura 8, são apresentas informações sobre a média da semana comparadas com a média geral de glicose por categoria. Baseado nestas informações o médico e paciente poderão comparar o tratamento atual com o tratamento que foi aplicado em períodos anteriores, podendo realizar alterações nos medicamentos.

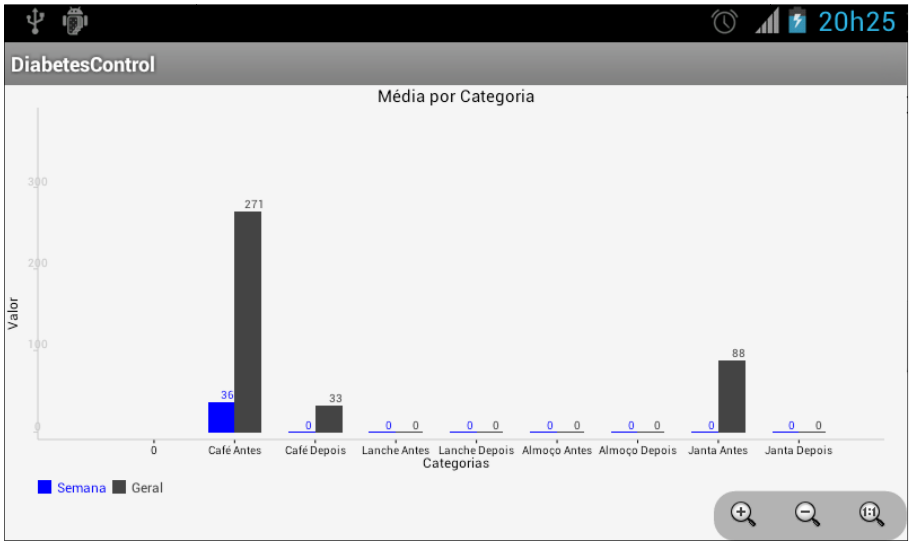


Figura 8 – Gráfico de média de glicose por categoria

O Diabetes Control, também possibilita a emissão de relatórios que auxiliam no monitoramento de diversos indicadores da diabetes. Um desses indicadores que o sistema possibilita um acompanhamento por meio de relatórios é a média por período. Como é possível verificar na Figura 9, o sistema possibilita que sejam emitidos relatórios com a média

do dia atual, dia anterior, semana atual, semana anterior, mês atual, mês anterior, anual e o total geral de todas as marcações, possibilitando assim um acompanhamento sobre a evolução da doença em diversos períodos.

Figura 9 – Relatório de média por período

Caso o medico tenha feito as cofigurações para sincronizar as informações do paciente, a cada tela que o medico utilizar o sistema, sera questionado sobre qual paciente ele irá ver os dados. Após selecionar o paciente, o médico visualiza as informações sobre os registros cadastrados por um determinado paciente. A partir destes registros, o médico poderá cadastrar notas médicas para cada um deles. As notas, são sincronizadas com o dispositivo do paciente para que posteriormente o ele possa verificar as informações cadastradas pelo médico. Essa nota médica pode ser um lembrete de uma medicação que não está sendo tomada, uma mudança de tratamento ou qualquer informação que o médico julgue necessária.

Na Figura 10 é possível verificar a lista dos registros que foram sincronizados para o dispositivo do médico, e a nota que está sendo gerada para um destes registros.

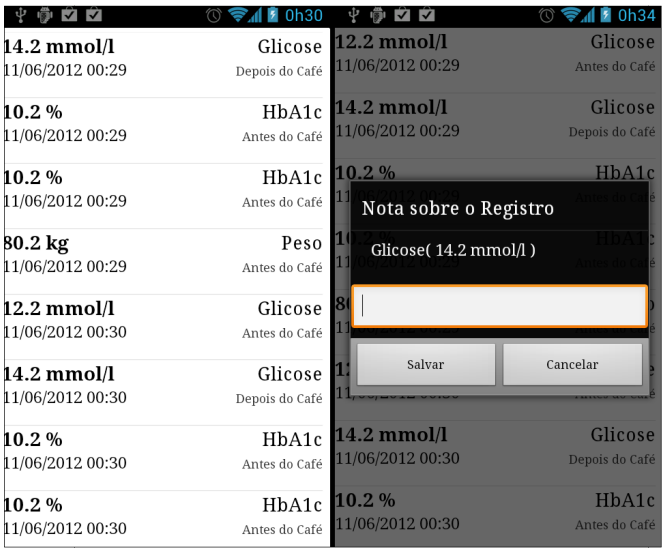


Figura 10 – Lista de registros do paciente e nota médica sobre o registro

Após o cadastro da nota, esse registro também é enviado ao WS, permitindo que o paciente consulte essa informação.

Em ambos módulos do aplicativo ainda existe a activity de configuração. Através desta página é possível selecionar as unidades de medidas de peso, glicose, HbA1c e outros tipos de registros a serem cadastrados. Nesta activity também é possível selecionar se a sincronização irá ou não ocorrer, visto que o módulo paciente pode ser utilizado normalmente sem a sincronização das informações. Quando a opção de sincronização está marcada, ao

salvar um cadastro ou executar alguma outra ação, essas informações são submetidas a sincronização. Caso essa sincronização não aconteça instantaneamente, ocorrerá no próximo momento, sempre de forma assíncrona.

Pode-se concluir que todos objetivos propostos foram concluídos, tendo em vista a construção de um aplicativo para controle de diabetes do paciente, que gera indicadores desta doença para o paciente e para o médico, podendo ou não realizar a sincronização das informações via WS com o celular de um médico, utilizando para isso a tecnologia Android.

O modelo proposto está disponível na Google Play sobre o link: https://play.google.com/store/apps/details?id=com.diabetescontrol.activities .

6. Conclusão

A utilização de dispositivos móveis pela população está aumentando gradativamente, tornando os dispositivos uma nova fonte para desenvolvimento de soluções nas mais diversas tecnologias. As soluções móveis já são utilizadas em áreas como a saúde, onde a informação deve estar disponível todo momento. Tecnologias que permitam o acompanhamento do paciente em tempo real pelo médico utilizando tecnologias móveis permitem um tratamento muito mais eficaz.

A fim de alcançar os objetivos deste trabalho, que consiste no desenvolvimento de um aplicativo móvel utilizando a plataforma Google Android como forma de controlar os níveis da diabetes dos pacientes, trocando informações com médicos via WS, foi realizado estudo sobre a plataforma Google Android, sincronização de dispositivos com WS, telemedicina e diabetes. O aplicativo móvel foi desenvolvido utilizando tecnologias como Google Android, a IDE Eclipse e todas ferramentas da SDK Android. O Web Service foi implementado utilizando o padrão JAX-WS, utilizando a IDE NetBeans como ferramenta de desenvolvimento.

Neste contexto, foi criada uma solução que sincroniza informações cadastradas em um dispositivo móvel com tecnologia Android por meio de um WS, com outro dispositivo móvel. As informações trocadas possuem informações de registros de diabetes dos pacientes e notas médicas sobre os registros de diabetes. Utilizando a solução desenvolvida, é possível que médicos acompanhem a evolução diária de seus pacientes direcionando o tratamento conforme indicadores gerados em tempo real.

Pode-se concluir que a plataforma Google Android possibilita um desenvolvimento de soluções robustas e com alto nível de complexidade. Usadas junto com tecnologias como os Web Services, permitem que informações possam ser compartilhadas e utilizadas nas mais diversas áreas do conhecimento.

REFERÊNCIAS

OPEN HANDSET ALLIANCE. Overview. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 17 maio 2011.

ABI RESEARCH (New York). Android Will Seize 45% of Smartphone Market by 2016.Disponível em: <http://www.abiresearch.com/press/3651-Android+Will+Seize+45%25+of+Smartphone+Market+by+2016,+Says+ABI+Research>. Acesso em: 17 maio 2011.

WEN, Chao Lung. Telemedicina e Telessaúde: Um panorama no Brasil. Disponível em: <http://www.ip.pbh.gov.br/ANO10_N2_PDF/telemedicina_telesaude.pdf>. Acesso em: 23 maio 2011.

GOOGLE. Android Developers. Disponível em: <http://developer.android.com/>. Acesso em: 09 jun. 2012.

CONDER, Shane; DARCEY, Lauren. Android wireless application development. 2. ed. Boston: Pearson Education, 2010.

MORRIS, Jason. Android User Interface Development: Beginner's Guide. Birmingham: Packt Publishing, 2011.

STEELE, James; TO, Nelson. The Android Developer's Cookbook: Building Applications with the Android SDK. Boston: Pearson Education, 2011.

MARTINS, Victor Manuel Moreira. Integração de Sistemas de Informação: Perspectivas, normas e abordagens. 2005. 201 f. Tese (Mestrado) - Universidade do Minho, Guimarães, 2005. Disponível em: <http://repositorium.sdum.uminho.pt/bitstream/1822/5657/3/tese_mestrado_victor_martins_2005.pdf>. Acesso em: 15 nov. 2011.

CERAMI, Ethan. Web Services Essentials. California, USA: O'Reilly, 2002. 304p.

ABINADER, Jorge Abílio; LINS, Rafael Dueire. Web Services em Java. Rio de Janeiro: Brasport, 2006.

POTTS, Stephen; KOPACK, Mike. Aprenda em 24 horas Web Services. Rio de Janeiro: Campus, 2003. Tradução de: Marcos Vieira.

Blois, M.S., Shortliffe, E.H. The Computer Meets Medicine: Emergence of a Discipline. Medical Informatics: Computer Applications in Health Care, New York, n. , p.3-36, 1990. Addison-Wesley Publishing.

ROCHA, Álvaro. Reflexão sobre a Informática de Saúde. GIMED - Grupo de I&D em Informática Médica, Universidade Fernando Pessoa. Disponível em: <http://www.i-gov.org/index.php?article=2720&visual=1&id=154&subject=213>. Acesso em: 26 out. 2011.

SADZINSKI, Osmani José. Protótipo de um aplicativo para controle de medicamentos baseado nos padrões de estrutura da certificação de software da sociedade brasileira de informática e saúde e conselho federal de medicina. 2010. 88 f. Dissertação (Graduação) - Curso de Ciência Da Computação, Universidade Do Extremo Sul Catarinense, Criciúma, 2010. Disponível em: <http://www.kiron.unesc.net/tcc/arquivos/trabalhos/244.pdf>. Acesso em: 26 out. 2011.

SBIS. O que é Informática em Saúde ? Disponível em: <http://www.sbis.org.br/>. Acesso em: 26 out. 2011.

WHO. Telemedicine: Opportunities and developments in Member State. Switzerland: Who Library Cataloguing-in-publication Data, 2010. 2 v. (Global Observatory for eHealth series). Disponível em: <http://whqlibdoc.who.int/publications/2010/9789241564144_eng.pdf>. Acesso em: 27 out. 2011.

CANCELLIÉRI, Cláudio. Diabetes & Atividade Física. Jundiaí: Fontoura, 1999.