UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

LÉO MORAES DE CARVALHO

DESENVOLVIMENTO DE UMA SOLUÇÃO WEB INTEGRADA COM O APLICATIVO DIABETES CONTROL PARA O CONTROLE E GERENCIAMENTO DE DIABETES

CRICIÚMA

2014

LÉO MORAES DE CARVALHO

DESENVOLVIMENTO DE UMA SOLUÇÃO WEB INTEGRADA COM O APLICATIVO DIABETES CONTROL PARA O CONTROLE E GERENCIAMENTO DE 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: Prof. MSc. Gustavo Bisognin

CRICIÚMA

2014

Léo Moraes de Carvalho

DESENVOLVIMENTO DE UMA SOLUÇÃO WEB INTEGRADA COM O APLICATIVO DIABETES CONTROL PARA O CONTROLE E GERENCIAMENTO DE 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.

Criciúma, 24 de Novembro de 2014.

BANCA EXAMINADORA

Prof. Gustavo Bisognin- Mestre - (Unesc) - Orientador

Prof. Paulo João Martins - Mestre - (Unesc)

Profa. Christine Vieira - Mestre - (Unesc)

Aos meus amigos e familiares.

AGRADECIMENTOS

Agradeço a minha família, aos meus amigos e todos os demais que me ajudaram a desenvolver e conseguir conquistar este objetivo.

“Quem nunca errou nunca experimentou nada novo”

Albert Einstein

RESUMO

A diabetes é considerada uma doença crítica que ataca a população nas mais diversas classes sociais. A utilização de sistemas automáticos de gerenciamento que auxiliem no controle da Diabetes pode proporcionar uma melhor qualidade de vida aos portadores. Na área da saúde é recomendado que os responsáveis sempre tenham em mãos as informações evolutivas sobre a doença e o paciente. Por isso esta pesquisa apresenta o desenvolvimento de um sistema *Web* para profissionais da saúde e pacientes, o qual possui foco no controle de indicadores da diabetes usando tecnologias como Java, Android e JAX-WS, assim foi feita a prototipação das telas necessárias para o protótipo, utilizando-se da arquitetura *Model View Controller* (MVC) foi realizado o desenvolvimento do protótipo juntamente com o desenvolvimento do *Web Service* para a integração entre o aplicativo Diabetes Control o sistema *web* Diabetes Control Web, onde permite que os dados de diabetes do paciente possam ser informados tanto por seu *smartphone* via *Web Service*, quanto por um sistema *Web*, assim aprimorando o controle e gerenciamento da diabetes por parte dos pacientes e melhorando a integração do paciente e médico que pode em tempo real analisar os índices de diabetes e realizar tratamentos simultaneamente com seus pacientes.

Palavras-chave: Java, Android, Diabetes.

ABSTRACT

Diabetes is considered a critical illness that attacks the population in various social classes. The use of automatic management systems to assist in controlling Diabetes can provide a better quality of life for patients. In health it is recommended that those responsible always have at hand the evolutionary information about the disease and the patient. Therefore this research presents the development of a Web system for health professionals and patients, which has focused on the control of diabetes indicators using technologies such as Java, Android and JAX -WS, so it was made prototyping the necessary screens for the prototype, using the Model-View-Controller (MVC) architecture was performed prototype development along with the development of the Web Service to the integration of web application diabetes Control and diabetes Control Web system, which allows the patient diabetes data can be informed by both your smartphone via Web Service, and by a Web system, thus improving the control and management of diabetes for patients and improving the integration of a patient and doctor with can analyze the real time rates of diabetes and perform treatment simultaneously with their patients.

Keywords: Java, Android, Diabetes.

LISTA DE ILUSTRAÇÕES

Figura 1 - Arquitetura Android 19
Figura 2 - Caminho dos dados na arquitetura Web Service 27
Figura 3 - Arquitetura dos Web Services 29
Figura 4 - Modelagem conceitual do aplicativo Diabetes Control 43
Figura 5 - Modelagem Conceitual do sistema 44
Figura 6 - Processo de desenvolvimento 44
Figura 7 - protótipo da tela de cadastro de paciente 46
Figura 8 - Prototipação da tela de Adicionar Registro 47
Figura 9 - Banco de dados do aplicativo *Diabetes Control* 48
Figura 10 - Banco de dados do aplicativo Diabetes Control Web 49
Figura 11 - Diferença entre os métodos salvar do *Web Service* e do *Diabetes Control Web* 50
Figura 12 - Anotações do JPA na entidade Registro 51
Figura 13 - Cadastro de pacientes 53
Figura 14 - Tela de adicionar paciente. 54
Figura 15 - Lista de pacientes 55
Figura 16 - Tela de listagem de registros 56
Figura 17 - Tela de cadastro de registros 57

LISTA DE TABELAS

Tabela 1 - Principais Versões do Android 18

LISTA DE ABREVIATURAS E SIGLAS

ADT	Android Development Tool
API	Application Programming Interface
CSV	Comma-separeted Values
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
DAO	Data Access Object
DM	Diabetes Mellitus
DVM	Dalvik Virtual Machine
JAX-WS	Java API for XML Web Services
JDK	Java Development Kit
JEE	Java Enterprise Edition
JPA	Java Persistence API
JSF	Java Server Faces
JTA	Java Transaction API
MVC	Model View Controller
OHA	Open Handset Alliance
PDF	Portabel Document Format
SDK	Software Development Kit
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UUDI	Universal Discovery, Description, and Integration
URL	Uniform Resource Locator
WSDL	Web Service Description Language
XLS	Extensão Excel
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVO GERAL 14
1.2 OBJETIVOS ESPECÍFICOS 14
1.3 JUSTIFICATIVA 14
1.4 ESTRUTURA DO TRABALHO 15
2 PLATAFORMA ANDROID 17
2.2 VERSÕES DO ANDROID 18
2.3 ARQUITETURA ANDROID 18
2.3.1 Framework 19
2.3.2 Android Runtime 20
2.3.3 Kernel 20
2.3.4 Libraries 20
2.3.5 Application 21
2.4 ANDROID SDK 21
2.5 SISTEMA DE ARMAZENAMENTO 22
2.6 DISPOSITIVOS MÓVEIS 22
2.6.1 Tablets 23
2.6.2 Smartphones 24
3 WEB SERVICES 26
3.1 PROTOCOLO SIMPLES DE ACESSO A OBJETOS (SOAP) 26
4 INFORMÁTICA E SAÚDE 30
4.1 TELEMEDICINA 30
4.2 DIABETES 31
4.3 TRATAMENTO DE DIABETES 32
5 APLICAÇÕES WEBS 33
5.1 CLIENT SIDE 33
5.1.1 HTML 33
5.1.2 CSS 34
5.1.3 JAVASCRIPT 34
5.1.4 Twitter Bootstrap 35

5.1.5 Primefaces 35
5.2 SERVER SIDE 36
5.2.1 Servidores de aplicação 36
5.2.2 Java 37
5.2.3 Java EE.....38
5.2.4 Java Persistence API– jpa 38
5.2.5 JAVA SERVER FACES – JSF 39
6 TRABALHOS CORRELATOS.....40
6.1 GOOGLE ANDROID: UM NOVO PARADIGMA DE DESENVOLVIMENTO MOBILE APLICADO AO GERENCIAMENTO MÉDICAS REFERENTE AO CONTROLE DO DIABETES 40
6.2 M-COMMERCE E A PLATAFORMA GOOGLE ANDROID 40
6.3 SISTEMA DE APOIO AO DIAGNÓSTICO MÉDICO UTILIZANDO TECNOLOGIAS MÓVEIS.....41
7 Diabetes Control web – interface de gerenciamento 42
7.1 METODOLOGIA.....42
7.1.1 Fluxo de Informações 43
7.2 MODELAGEM 45
7.2.1 Prototipação 45
7.2.2 Modelagem de banco de dados 47
7.2.3 Ambiente de Desenvolvimento 49
7.3 IMPLEMENTAÇÃO 50
7.5 RESULTADOS OBTIDOS.....57
8 CONCLUSÃO 59
REFERÊNCIAS.....60

1 INTRODUÇÃO

Dados estatísticos revelam que existem atualmente no mundo, em torno 860 milhões de pessoas com alguma doença crônica, estima-se que 25% desses pacientes poderiam se beneficiar imediatamente de soluções para o monitoramento de sua saúde em casa, porém a interação de recursos médicos em celulares beneficiaram cerca de 50% desses pacientes (MARTIN, 2013).

A crescente evolução da informática e dos aplicativos e soluções aplicadas a dispositivos móveis, tem proporcionado um aumento significativo da qualidade de vida de pacientes portadores de enfermidades que necessitam de controle diário. Uma doença bastante conhecida que, atualmente ataca uma grande parcela da população, é conhecida como diabetes.

A grande necessidade de controle associada a evolução da tecnologia, fomenta o desenvolvimento de soluções que tem como foco principal a melhoria da qualidade de vida dos pacientes portadores deste tipo de doença. Atualmente, muitos usuários de aplicativos móveis estão insatisfeitos com as produções de várias empresas, pois estas, ao invés de criarem aplicativos inovadores, que solucionem uma real necessidade relacionada ao mercado em que atuam, acabam apenas por desenvolver uma cópia de seus sites (NIEMIS, 2010, tradução nossa).

Atualmente o Diabetes Mellitus (DM) é considerado um dos mais importantes problemas de saúde pública, tanto pelo número de pessoas afetadas quanto pelos custos envolvidos no tratamento de suas complicações (PÉRES et al, 2007).

Segundo a Sociedade Brasileira de Diabetes (2007) com a informação gerada através dos sensores de glicose pode-se com a informação em tempo real e de forma continua, tomar decisões de aplicar insulina ou ingerir carboidratos para evitar hipoglicemia. Os quadros de hipoglicemia, por vezes intensos e muito sérios pode receber orientação imediata resultando muitas vezes em vidas salvas. Já que sabe-se que esta é uma das principais causas de morte de arritmias em diabéticos.

O trabalho consiste em desenvolver um sistema *Web* integrado com o aplicativo Diabetes Control, que permita a integração entre paciente e médico para o melhor controle e gerenciamento da diabetes.

1.1 OBJETIVO GERAL

Evoluir o aplicativo “Diabetes Control”, criando um novo sistema *web* para que seja possível controlar os níveis diabéticos assim como a sincronização entre aplicativo e sistema.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa são:

a) realizar estudo sobre diabetes;
b) realizar estudo sobre Android;
c) realizar estudo sobre plataformas de desenvolvimentos web;
d) implementar sistema web para controle dos dados;
e) implementar função que realize exportação de dados;
f) realizar integração do aplicativo *Diabetes Control com o aplicativo Diabetes Control Web*;

1.3 JUSTIFICATIVA

Em três anos, cerca de meio bilhão de pessoas no mundo irão receber algum tipo de cuidado ou controle médico através de aplicativos em seus smartphones (Martín, 2013).

Segundo Pereira e Silva (2009), ao longo dos anos, a quantidade de telefones celulares vem aumentando cada vez mais, atualmente o celular é o produto de consumo mais utilizado no mundo, sendo a quantidade existente correspondente à metade da população mundial (3,3 bilhões - 2007).

Acredita-se que até o final de 2013 este número chegará a 5,6 bilhões, comparativamente, a quantidade de televisores é em torno de 1,5 bilhões, a quantidade de celulares na Internet atinge mais que o dobro de computadores com acesso à Internet. Em conjunto com a Open Handset Alliance (OHA), o Android foi o primeiro projeto Open Source, permitindo o desenvolvimento de aplicativos utilizando Software Development Kit (SDK), tendo em vista a intenção de tirar o máximo de

proveito dos aparelhos móveis. A telemedicina surgida na década de 60 consiste no uso da tecnologia para possibilitar cuidados à saúde nas situações onde a distância é um fator crítico (WEN, 2013).

Um estudo em Manhatam descobriu que 71% dos médicos consideram smartphones essênciais para suas práticas e 84% disseram que Internet é fundamental em seu trabalho. Os médicos podem usar dispositivos como tablets para realizar o levantamento das informações do paciente durante uma consulta em seguida mostrar para o paciente como as doenças se espalham, ou como a curvatura da coluna vertebral ocorre. O uso de dispositivos móveis na área da saúde tem se expandido nos últimos 5 anos e continuará a crescer a medida que o uso dos smartphones forem aumentando. Mais de 70% da população dos Estados Unidos terá um smartphone até 2015 (O'CONNOR, 2013, Tradução nossa).

Foi escolhido o aplicativo "Diabetes Control" para ser utilizado como base para o desenvolvimento de um sistema web com o controle sobre os índices de diabetes.

1.4 ESTRUTURA DO TRABALHO

A presente pesquisa foi dívida em sete capítulos, no primeiro se encontra uma introdução sobre o tema proposto, os objetivos gerais e os objetivos específicos e a justificativa.

O segundo capitulo é apresentado a plataforma Android, assim como suas versões, arquitetura, sistema de armazenamento e seu kit de desenvolvimento.

Em seguida foi abordado a tecnologia móvel com smartphones e tablets, como também sobre conceito de sincronização com *Web Services.*

A diabetes e tecnologia em saúde é descrita no quarto capítulo, também mostrando a necessidade das tecnologias móveis na saúde.

Logo em seguida tem-se o capítulo referente a aplicações construídas na web e todas as suas tecnologias.

No sexto capítulo são descritos os trabalhos correlatos, levando em consideração uma maior abrangência nas tecnologias utilizadas.

O sétimo e último capítulo tem o desenvolvimento do sistema junto com a metodologia, modelagens e técnicas utilizadas.

2 PLATAFORMA ANDROID

Android é o primeiro projeto de uma plataforma Open Source (código aberto) para dispositivos móveis em conjunto com a OHA, que tem como intenção construir um sistema operacional de código aberto, onde qualquer software terá o mesmo privilégio que os aplicativos criados ou licenciados pelos fabricantes tem, criando uma integração e customização dos sistemas e aparelhos, com o objetivo de transformar telefones interessantes iguais aos computadores (PEREIRA; SILVA 2012).

Em 5 de novembro de 2007 a Google torna pública a primeira plataforma Open Source baseada na plataforma Java e com sistema operacional Linux, de desenvolvimento para dispositivos móveis, a qual foi chamada de Android (PEREIRA; SILVA 2009).

A plataforma foi construída para permitir que os desenvolvedores pudessem criar aplicativos móveis tirando proveito de tudo que um aparelho móvel tem a oferecer, podendo assim uma aplicação tirar vantagens de qualquer uma das principais funcionalidades do telefone como por exemplo: fazer chamadas, enviar mensagens de textos *Short Message Service* (SMS), utilizar as câmeras acopladas nos telefones entre outras (OPEN HANDSET ALLIANCE, 2013).

Hoje em dia o Android já está presente em centenas de milhares de dispositivos móveis pelo mundo em mais de 190 países espalhados pelo mundo, e cada dia vem crescendo cada vez mais, cerca de um milhão de novos aplicativos são ativados todos os dias (GOOGLE, 2013, tradução nossa).

Dentro do Android todas as aplicações são criadas iguais, o Android não diferencia entre aplicações construídas por desenvolvedores e funcionalidades do telefone, sendo assim aplicativos construídos por terceiros podem ser feitos para ter igualdade ao acesso as capacidades do telefone, com aplicativos construídos para Android os usuários são capazes de adaptar o telefone para seus interesses pessoais, podendo assim personalizar suas funcionalidades nativas alterando-as por aplicativos desenvolvidos por outros desenvolvedores (OPEN HANDSET ALLIANCE, 2013).

2.2 VERSÕES DO ANDROID

Desde seu lançamento o Android já teve diversas versões. A tabela 1 representa as versões já desenvolvidas até o momento.

Tabela 1 - Principais Versões do Android

Versão	Data de Lançamento	Nome
Android 1.5	04/2009	Cupcake
Android 1.6	09/2009	Donut
Android 2.0	10/2009	Eclair
Android 2.2	05/2010	Froyo
Android 2.3	12/2010	Gingerbread
Android 3.0	02/2011	Honeycomb
Android 3.1	05/2011	Honeycomb
Android 3.2	07/2011	Honeycomb
Android 4.0	10/2011	Ice Cream Sandwich
Android 4.1	07/2012	Jelly Bean
Android 4.2	10/2012	Jelly Bean
Android 4.3	07/2013	Jelly Bean
Android 4.4	2013	Kit Kat

Fonte: Google (2013).

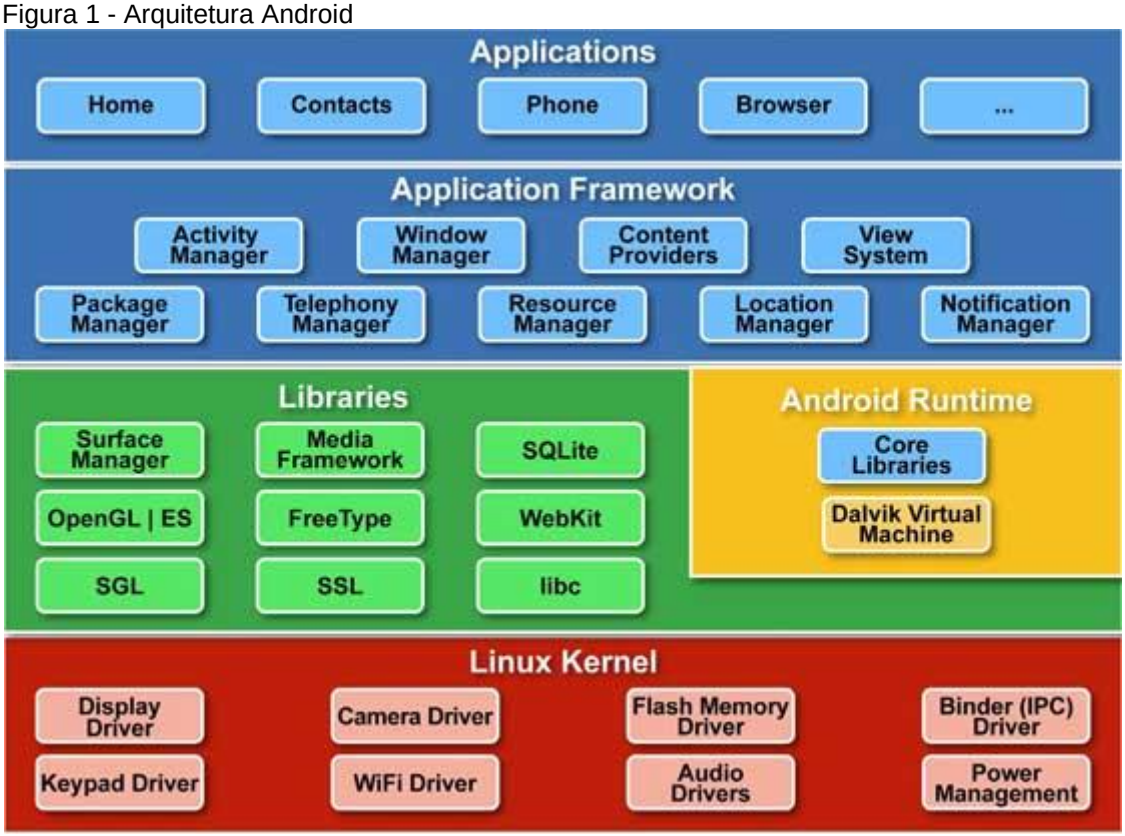
De acordo com o modelo apresentado tem-se o destaque da versão 4.4 que ainda não foi lançada e promete ser uma das melhores versões já lançadas (GOOGLE, 2013).

2.3 ARQUITETURA ANDROID

Android é mais que um sistema operacional, é um stack software (conjunto de programas que trabalham em conjunto para alcançar um objetivo em comum) composto por cinco camadas, a base do Android é uma versão do Kernel Linux 2.6 modificada que disponibiliza vários serviços como principais serviços de

segurança, gerenciamento de memória, processos e segurança, e uma camada de abstração de hardware para a camada de software (GOOGLE, 2013, tradução nossa).

Figura 1 - Arquitetura Android



Fonte: GOOGLE(2013)

A figura 1 demonstra arquitetura da plataforma android, em destaque a camada do Kernel que utiliza como base uma versão modificada do Linux, sendo assim podendo ter todas as funcionalidades do mesmo.

2.3.1 Framework

Na camada de Application Framework encontram-se todas as *Application Programming Interface* (API) e recursos necessários pelos aplicativos, como as classes virtuais, provedores de conteúdo, gerenciadores de recursos, localização, pacotes, atividades entre outros, foi criada para que os desenvolvedores possam tirar o máximo proveito para seus projetos, uma das principais características da

camada framework é que ele permite aos outros desenvolvedores o reaproveitamento de componentes (RABELLO, 2009).

2.3.2 Android Runtime

Ambiente de execução como é chamada a camada Android Runtime, é uma instância da Dalvik Virtual Machine (DVM), uma máquina virtual Java especialmente projetada e desenvolvida para o projeto Android, o DVM utiliza o núcleo do Linux para gerenciar memória e *multi-threading*, a DVM habilita que cada aplicativo Android possa executar seus próprios processos em sua instância do DVM (GOOGLE, 2013, tradução nossa).

A DVM executa arquivos *.DEX*, uma extensão do tipo *Dalvik Executable*, classes Javas convertidas para máquina virtual pela ferramenta DX juntamente com o SDK do Android (PEREIRA; SILVA 2009).

2.3.3 Kernel

O Android utiliza como Kernel uma modificação da versão 2.6 do Linux, que fornece as funcionalidades básicas para o sistema, como gerenciamento dos processos, memória, dispositivos de entrada e saída, câmera entre outros, o Kernel lida com todas as funcionalidades que são consideradas boas no Linux como a camada de rede e driver de dispositivos (RABELLO, 2009).

2.3.4 Libraries

A camada de bibliotecas é onde estão armazenadas todas as camadas do Android, bibliotecas C/C++ utilizadas pelo sistema, bibliotecas de áudio e vídeo, bibliotecas de visualização de imagens estáticas, imagens 2D e 3D, funções para navegadores web, funções para gráficos, funções de aceleração de hardwares, renderização 3D entre outros (GOOGLE, 2013, tradução nossa).

2.3.5 Application

A camada de aplicação é onde se encontram todas as aplicações fundamentais para o funcionamento do sistema operacional, é a camada onde se encontram todos os aplicativos desenvolvidos pela comunidade Android (PEREIRA; SILVA 2012).

Esta camada é onde as empresas que distribuem o Android em seus aparelhos alteram a camada visual, proporcionando a diferenciação de outros Androids, fazendo com que cada dispositivo tenha características próprias (MORRIS, 2011, tradução nossa).

2.4 ANDROID SDK

O SDK é o que disponibiliza as ferramentas e APIs necessárias para o desenvolvimento de aplicações para Android, a Google disponibiliza seu próprio SDK chamado de *Android Developers Tools* (ADT), para iniciantes é disponibilizado um download de um pacote com: eclipse com o plugin do ADT, ferramentas SDK Android, ferramentas de plataforma Android, a última plataforma Android, a última imagem do sistema Android para o emulador (GOOGLE, 2013, tradução nossa).

Uma das principais vantagens da SDK do Android que ao contrário de diversas plataformas proprietárias que exigem taxas para a obtenção do SDK, ele totalmente gratuito as ferramentas são totalmente disponíveis, para escrever um programa para Android é necessário toda a configuração do ambiente de desenvolvimento, os programas podem ser desenvolvidos sobre os sistemas operacionais: Windows, Macintosh ou Linux, os seguintes softwares são necessários estar instalados para começar a desenvolver uma aplicação para Android: Java *Development Kit* (JDK) versão 5 ou maior, Java *Integrated Development Environment* (IDE) como Eclipse, Android SDK, ADT (CONDER; DARCEY, 2010, tradução nossa)

2.5 SISTEMA DE ARMAZENAMENTO

Dispositivos móveis em geral sofrem com problemas de poucos recursos computacionais, a maioria como processamento e memória em comparação com sistemas desktops, os dispositivos com suporte a a *Structured Query Language* (SQL) já permitem consulta de alto nível, utilizando SQL como padrão, O Android também aceita banco de dados como: OracleLite e SQL Server Mobile Edition, só é possível utilizar a SQL pois uma biblioteca em C implementa o banco de dados que lê e escreve diretamente em arquivos do dispositivo, a vantagem do Sqlite é que ele já vem embutido na plataforma Android não havendo necessidade de instalá-lo, o Android tem suporte ao banco de dados através de uma API, oferecendo funções básicas de gerenciamento (criação, abertura e fechamento) (JOHNSON; JOHNSON, 2008).

2.6 DISPOSITIVOS MÓVEIS

A partir dos anos 90 ocorreu uma grande evolução e popularização dos dispositivos móveis como celulares e laptops, isso pelo grande crescimento no desenvolvimento de tecnologias para dispositivos móveis, uma das grandes vantagens dos dispositivos móveis é permitir acesso a informações aonde quer que o indivíduo esteja, esse ambiente proporciona a criação do conceito de computação móvel (FIGUEIREDO; NAKAMURA, 2003)

Para Lyytinen e Yoo (2012) a computação móvel tem que ser baseada na forma de mover consigo serviços computacionais, assim os computadores tornam-se dispositivos sempre presentes e com as mesmas funcionalidades que um computador tradicional oferece independentemente da localização atual do indivíduo, a computação móvel transformou a computação de uma maneira em que pode ser carregada para qualquer lugar.

Para Mateus e Loureiro(1998), a quarta revolução na computação foi a computação móvel que é considerada um paradigma de programação, essa revolução só fica atrás dos centros de processamento de dados na década de 60, a criação dos terminais nos anos 70 e redes de computadores nos anos 80, esse novo

paradigma proporciona ao usuário acesso a aplicativos, serviços e recursos sem a necessidade de grandes infraestruturas, tendo acesso aos recursos possibilitando a mobilidade de informações.

Várias empresas estão recrutando ou escalonando novos desenvolvedores para novos grupos de desenvolvedores para sistemas móveis, ou para adaptar serviços já existentes para suprimir a demanda do mercado, já se tornando um padrão de desenvolvimento onipresente nas empresas. (RABELLO, 2009).

Segundo Mateus e Loureiro (1998), a comparação entre as tecnologias pode ser feita a partir de quantos anos cada uma levou para adquirir um milhão de usuários, a televisão preta e branca demorou cerca de vinte anos, enquanto que os sistemas computacionais levaram cerca de seis anos, porém os dispositivos móveis demorou apenas um ano para conseguir tal marca.

Segundo Pereira e Silva (2009), ao longo dos anos, a quantidade de telefones celulares vem aumentando cada vez mais. Atualmente, o celular é o produto de consumo mais utilizado no mundo, sendo a quantidade existente correspondente à metade da população mundial (3,3 bilhões - 2007). Acredita-se que até o final de 2013 este número chegará a 5,6 bilhões, comparativamente, a quantidade de televisores é em torno de 1,5 bilhões, a quantidade de celulares na Internet atinge mais que o dobro de computadores com acesso à rede.

O primeiro celular teve seu protótipo desenvolvido na década de 70, desenvolvido pela Motorola teve seu lançamento somente na década de 80, chamado de DynaTAC 8000X, pesava cerca de 1kg e possuía uma memória para 30 números, após quase uma década de desenvolvimento quando finalmente foi lançado se tornou um símbolo para os ricos pois o mesmo custava entre $3995 - $8657 dólares (ABREU, 2004).

2.6.1 Tablets

Tablets são dispositivos móveis com telas sensíveis ao toque mais conhecida como *touch screen*, tem uma grande semelhança com computadores tradicionais, o *tablet* mais conhecido e mais comum é o *Ipad*, com a criação do *Ipad*

a apple abriu mercados para os demais desenvolvedores de dispositivos (GRUMAN, 2013).

A grande empresa de eletrônicos Samsung está atualmente liderando juntamente com a Apple o mercado de *Tablets*, a Samsung está a vários anos no mercado mais nos últimos anos vem ganhando destaque no setor de dispositivos móveis como *Tablets* e smartphones, com seus aparelhos equipados com sistemas operacional Android, relatórios apontam que a Samsung fechou o ano de 2012 com 22% da participação do mercado móvel e sozinha foi responsável pela venda de 42% de todos os dispositivos operando com Android no planeta (CANALTECH, 2013).

Hoje em dia os *Tablets* são utilizados de diversas maneiras, podem ser utilizados como material escolar para ajudar o aprendizado de crianças e principalmente na medicina. Uma grande vantagem dos dispositivos móveis é que estes aplicativos se encontram nas mãos dos médicos sempre que necessário, podendo estar em qualquer lugar do consultório ou até mesmo em mesas de cirurgias sem a necessidade de estarem conectados à Internet a grande vantagem é sobre a mobilidade que eles proporcionam.

2.6.2 Smartphones

Smartphone literalmente traduzindo significa ”Telefone Inteligente” pode ser considerado a evolução do celular, a função de receber e fazer chamadas é apenas um detalhe diante de tamanhas funcionalidades dispostas por ele, são considerados híbridos entre celulares e computadores, não são simplesmente telefones mais também não tem a capacidade computacional de um computador de mesa, um *smartphone* é capaz de fotografar, filmar e fazer vídeos, a quantidade de aplicativos para ele é enorme desde ferramentas para trabalhos até opções de jogos, as principais marcas de *smartphones* são: Apple, Samsung, Motorola, LG, Nokia, BlackBerry, e os sistemas operacionais são o da Apple o IOS e o da Google o Android, porém o Windows Phone da Microsoft tem ganhado bastante mercado no último ano (BARROS, 2013).

Em 2011 foi vendido cerca de 712,5 milhões de smartphones no mundo, representando um aumento de cerca de 11% referente ao ano de 2010, no terceiro trimestre de 2012 cerca de 258 milhões de smartphones foram vendidos no mundo, 38,8% maior que o mesmo período do ano de 2011, o maior aumento no setor foi da Apple que subiu de 26,9 milhões para 33,8 milhões, enquanto que a Samsung vendeu cerca de 81,2 milhões de smartphones (TELECO,2013).

3 WEB SERVICES

O crescimento de aplicações distribuídas e aplicações voltadas para web faz-se necessário o desenvolvimento independentemente da localização geográfica, surgindo assim a necessidade de disponibilizar aplicações sem a necessidade de estar conectado diretamente na Internet, os Web Services servem para acessarem as aplicações remotamente utiliza eXtensible Markup Language (XML) na sua estrutura e Uniform Resource Locator (URL) como outra página da Internet, a diferença entre Web Services e páginas da Internet é como é enviado os dados para o servidor enquanto as requisições são utilizadas na Internet, o método utilizado nos Web Service o cliente envia um arquivo XML para o servidor de acordo com a formatação das especificações do Simple Object Acces Protocol (SOAP). (POTTS, KOPACK, 2003).

Em 1998 o *Worldwide Web Consorthium* (W3C) começou a utilizar o XML como linguagem padrão para formatação de dados na Internet as principais razões para ser utilizado como padrão foram por sua linguagem neutra e armazenamento simples de um documento de dados complexo, a linguagem é baseadas em *tags* que significam o valor seguinte a ser apresentado, um texto escrito não significa nada dentro do XML, porém se estiver dentro de uma *tag* específica passa a ter algum significado (POTTS, KOPACK, 2003).

Os *Web Services* vieram como soluções para o problema de sincronização de sistemas.

3.1 PROTOCOLO SIMPLES DE ACESSO A OBJETOS (SOAP)

O SOAP tem duas partes básicas para a mensagem o cabeçalho e o corpo (Header, Body), a mensagem fica dentro chamado de Envelope, o cabeçalho é opcional e pode conter todas as informações desde credenciais até instruções de processamento, no corpo também possui o payload que é o conteúdo a ser enviado

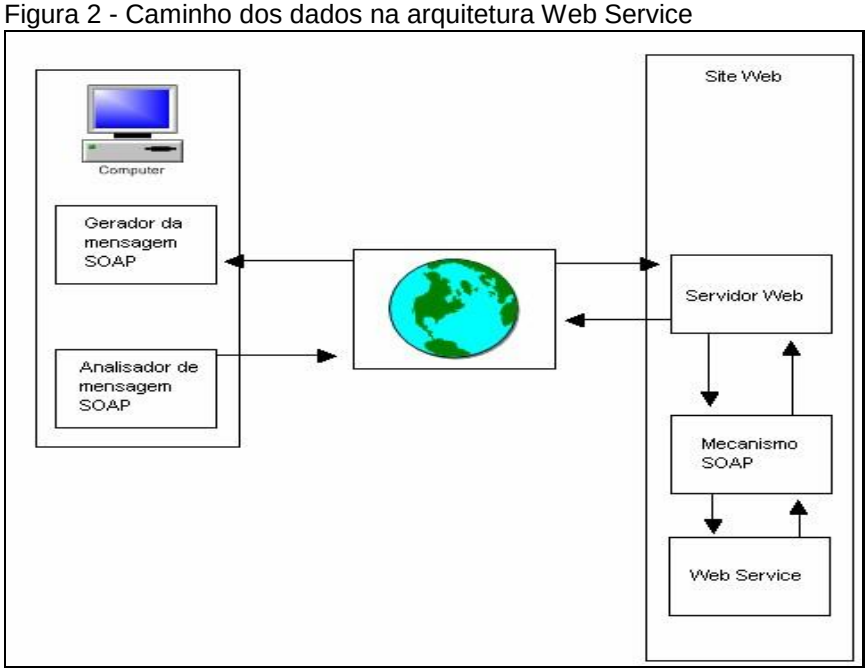
que pode variar de documentos XML até chamadas remotas (POTTS, KOPACK, 2003).

A troca de informações do SOAP com o Web Service é realizada pelo *HyperText Transfer Protocol* (HTTP), o cliente que utiliza esse tipo de padrão precisa de um arquivo chamado Web Service Description Language (WSDL), onde estão inclusa todas as regras para a comunicação, após receber as informações o desenvolvedor do lado do cliente irá montar as classes sobre o serviço que pretende acessar (PAULON, 2013).

O JAX-WS faz parte da plataforma Java EE, tem a funcionalidade de simplificar a tarefa do desenvolvimento de *Web Services* utilizando a tecnologia Java, ela fornece suporte a vários protocolos como SOAP 1.1 e SOAP 1.2, XML, e suporte a protocolos adicionais como HTTP, com suporte a anotações o JAX-WS simplifica o desenvolvimento e reduz o tamanho do arquivo .JAR (PAULON, 2013).

Os Web Services requerem atenção dobrada referente a segurança dos dados a serem disponibilizados. A figura 2 apresenta uma visão do funcionamento da arquitetura de um *Web Service*.

Figura 2 - Caminho dos dados na arquitetura Web Service



Fonte: Potts e Kopack (2003).

A WSDL usa como base para sua estrutura o XML e serve para a criação de documentos de descrição WS, fornecendo todo o tipo de informação para os Web Services, os principais elementos do WSDL são (POTTS, KOPACK, 2003):

a) tipos (*type*): são os tipos de dados que serão utilizados na troca de mensagens;
b) mensagem (*message)*: representa os dados a serem transferidos;
c) porta (*port*): endereço para ligações normalmente uma URL;
d) Tipo de porta (*portType*): são as operações disponíveis no Web Service representada pelo elemento de operação que faz referência a uma mensagem de entrada e uma de saída;
e) ligação (*binding*): especifica o protocolo e formatos para mensagens definidas.

O *Universal Discovery, Description, and Integration* (UDDI), é a especificação para integrar os WS entre as diferentes entidades, pode ser utilizado como uma ferramenta de busca onde organizações encontram outras organizações, pode ser comparado com uma lista telefônica que disponibiliza os consumidores dos serviços, o UDDI possui três divisões que são chamadas de páginas brancas, amarelas e azuis, onde na página branca é encontrado informações sobre a entidade, na página amarela as informações são referentes aos serviços e na verde como fazer um cliente solicitar serviços (ABINADER; LINS 2006).

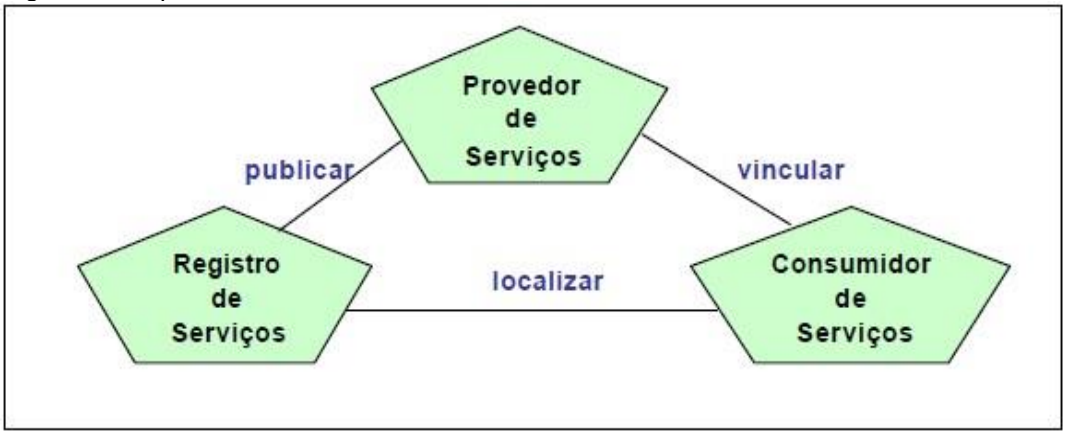
Todas as regras de funcionalidades são desenvolvidas do lado do servidor e também é onde fica os arquivos WSDL, no lado do cliente é onde o usuário terá acesso a todas as informações e recursos gerados pelo servidor seguem sempre aa especificações dos arquivos WSDL, pode ser escrita em qualquer linguagem de programação, para a sincronização de ambos os lados existe dois tipos de trocas: requisição/resposta à onde a iniciativa parte do cliente em realizar uma requisição com o servidor e solicitação/resposta onde o servidor solicita uma resposta do cliente (SILVA FILHO, 2004).

A arquitetura dos Web Services é baseada em três componentes, cada uma tem uma funcionalidade definida sendo elas (ABINADER; LINS, 2006):

a) provedor: pública e disponibiliza os meios para acesso aos serviços;

b) registro: mantêm os registros dos serviços locais;
c) consumidor: consome o serviço que é disponibilizado, realizando acessos e requisições, normalmente os consumidores são os clientes ativos que realizam requisições a serviços disponibilizados.

Figura 3 - Arquitetura dos Web Services



Fonte: Carvalho (2006).

Como pode se observar na figura 3 é mostrado como funciona arquitetura dos WS onde o registro de serviços públicos para o provedor e os serviços necessários para o consumidor realizar suas atividades, sendo assim podendo ser feito sincronização com serviços essenciais.

4 INFORMÁTICA E SAÚDE

A relação entre informática e saúde teve início na década de 70 onde foi criado o Núcleo de Tecnologia de Educação em Saúde, que originou a criação e desenvolvimento dos primeiros sistemas de monitoração fisiológica, porém a partir de 1983 onde ocorreu o grande avanço na relação entre informática e saúde onde foram criados novos grupos específicos para área de pesquisa e ensino e também o primeiro curso de informática para pós-graduandos de medicina (SABBATINI, 1995).

A informática em saúde abrange diversos campos por ela ser uma área em constante evolução, os principais campos são: prontuários eletrônicos do paciente, sistema de apoio a decisão, sistemas de informação em saúde, processamento de sinais biológicos, processamento de imagens médicas, telemedicina entre outros.

4.1 TELEMEDICINA

A telemedicina consiste em utilizar a medicina a distância, onde médico e paciente não mantem contatos físico mas sim por meios digitais utilizando recursos computacionais para transmissão e controle de dados, uma das principais utilização da telemedicina é a telecirurgia, onde os procedimentos cirúrgicos padrões são feitos a distância com a ajuda de aparelhos visuais (ALVES, 2002).

A telemedicina pode ajudar a atender à necessidade não supridas por outros métodos tradicionais de tratamentos a pacientes, um dos maiores problemas pode ser citado a mobilidade quanto aos pacientes que vivem em regiões remotas com difícil acesso a tratamentos médicos, é possível assim obter tratamentos de médicos especialistas através de videoconferências, evitando assim o problema de mobilidade que na maioria dos casos o custo de deslocamento é alto, a vantagem da telemedicina é o acompanhamento médico podendo até ser diários de pacientes com doenças crônicas como por exemplo a diabetes que é necessário o acompanhamento diário (WHO, 2010, tradução nossa).

4.2 DIABETES

O Diabetes Mellitus atinge cerca de 382 milhões de pessoas no mundo e se estima que esse número chegue a 592 milhões em 2035, a faixa etária que a DM mais atinge é entre 40 e 59 anos e 80% das pessoas atingidas são de países pobres (International Diabetes Federation, 2013, tradução nossa).

Segundo a Sociedade Brasileira de Diabetes (2007) no Brasil existe cerca de 12 milhões de pessoas com DM, devido ao grande número de complicações causada em 2011, o número de pessoas mortas por causa da DM chegou a 4,6 milhões no mundo inteiro.

No Brasil e no mundo a doença se destaca pelo problema de saúde pública, pela sua importância é refletida no aumento das taxas de morbidade e mortalidade, e nas sequelas de incapacidades como amputação de extremidades, cegueira, insuficiência renal e retinopatia diabética (GAMBA et al. 2004).

De acordo com a Sociedade Brasileira de Diabetes (2007) o DM é a elevação da glicose no sangue conhecido também como Hiperglicemia, decorrente do processo de digestões dos alimentos no intestino que é transformado em açúcar ou glicose, o controle é feito pela insulina substancia que é produzida pelo pâncreas, sendo os pacientes que contêm diabetes tem uma elevada taxa de insulina na corrente sanguínea decorrente da não absolvição adequado do organismo.

Existem quatro tipos de diabetes: A tipo 1 que é a diabetes que causa a destruição das células do pâncreas levando a uma insuficiência de insulinaa tipo 2 que é a diabetes que é a causa de um defeito secretor de insulina ou as células não reconhecem esse hormônio. a gestacional que ocorre durante a gravidez e outros tipos de DM específicos causados por problemas específicos como: defeitos na ação da insulina, defeitos nas células do pâncreas, tipos induzidos por drogas ou químicos, o tipo mais comum é o tipo 2 que atinge cerca de 90% das pessoas que possuem DM (INTERNATIONAL DIABETES FEDERATION, 2013, tradução nossa).

A diabetes 1 pode aparecer em qualquer faixa etária, porém está mais relacionada a indivíduos de no máximo 30 anos, os principais sintomas do diabetes são: indivíduo com frequente vontade de urinar, extrema vontade de tomar água, perda de peso em pouco tempo, as principais complicações da diabetes são:

complicações micro e macro vasculares, neuropatia, retinopatia, nefropatia e doenças periodontais (OLIVEIRA,2011).

O nível de glicose no sangue pode ser medido a qualquer hora do dia, caso o exame feito indique que a quantidade de glicose no sangue ultrapasse a marca de 200*mg/dl* o indivíduo é considerado com diabetes, entre 140*mg/dl* e 200*mg/dl* o indivíduo é considerado com pré-diabetes, entre 70*mg/dl* e 140*mg/dl* o indivíduo é considerado normal, e a baixo de 70*mg/dl* o indivíduo é considerado com hipoglicemia que é o baixo nível de glicose no sangue (OLIVEIRA,2011).

4.3 TRATAMENTO DE DIABETES

O tratamento da diabetes tipo 1 é relativamente simples: possuir uma boa dieta, praticar atividades físicas, realizar os testes de sangue diários, manter os exames médicos em dia e tomar insulina regularmente. Já o tratamento da diabetes tipo 2 é um pouco diferenciado, contando com os tratamentos da diabetes 1 como atividades físicas e boa alimentação, a única diferença é que o controle da insulina pode ser feito com a utilização de outros medicamentos que possam baixar a taxa de glicose no corpo. O tratamento para a diabetes tipo gestacional é feito através de uma alimentação com baixo teor de açúcar, gordura e carboidratos, exercícios físicos e aplicação da insulina com orientação médica (AMERICAN DIABETES ASSOCIATION, 1998).

5 APLICAÇÕES WEBS

As aplicações *web* funcionam de um jeito onde toda aplicação está localizada em um *web* server, que nada mais é que um computador ligado à Internet esperando pedidos de um navegador, que tem como função fazer a requisição ao servidor que retornar uma página *HyperText Markup Language* (HTML) (DUCKETT, 2010).

Cada navegador sabe a qual servidores eles estão fazendo requisições a partir da URL, que é o endereço do site onde se encontra o servidor que irá retornar a página HTML que foi requisitada pelo navegador, por exemplo quando um navegador solicita uma página HTML para a URL www.google.com.br será retornado a página padrão de busca da Google, os navegadores mais conhecidos são: Google Chrome, Mozilla FireFox e Internet Explore (DUCKETT, 2010).

As aplicações *web* são separadas em duas partes, a parte responsável pela exibição dos dados para o cliente que é chamada de *Client Side* e a parte responsável pelo processamento de dados no servidor que é chamada de *Server Side*.

5.1 CLIENT SIDE

A maioria das linguagens de programação utilizadas no lado do cliente são interpretadas pelos navegadores, a principal vantagem é a velocidade de execução já que são executadas no cliente sem a necessidade de ser feito requisições para o servidor, as principais linguagens utilizadas no lado do cliente são HTML, JavaScipt(JS) e Cascading Style Sheets (CSS) (DUCKETT, 2010).

5.1.1 HTML

O HTML é a linguagem de marcação de Hipertexto interpretada pelos navegadores, sua principal utilização é na criação de documentos e páginas *web*, a sua estrutura é baseada em *tag*, cada *tag* pode ser formada por diversos componentes como *links*, tabelas, imagens, vídeos entre outros, o inicio do HTML

deu-se a partir de 1992, a partir de 2000 a linguagem tornou-se uma norma internacional sendo controlada pela W3C

Ao longo dos anos o HTML já teve várias versões sendo a mais atual o HTML 5 sendo publicado suas especificações em janeiro de 2008 (SILVEIRA, 2013).

5.1.2 CSS

O CSS é a linguagem responsável pelos estilos visuais dos formulários presentes nas páginas *web*, a grande vantagem de se utilizar é a separação do estilo com a estrutura, sendo assim a estrutura da página fica a cargo do HTML e todos os estilos ficam a cargo do CSS, assim o HTML faz as estruturas de textos, formulários, botões, links entre outros e o CSS fica responsável pelas cores, posicionamento em tela, bordas e tudo o que for relacionado a apresentação para o usuário, por ser uma linguagem interpretada pelos navegadores cada navegador tem um jeito diferente de exibição dos comandos definidos pela linguagem, onde o mesmo comando pode ser visivelmente diferente em dois navegadores diferentes (DUCKETT, 2010).

5.1.3 JAVASCRIPT

O *JavaScript* é a linguagem do lado do cliente responsável pela interação do usuário tornando a aplicação mais amigável, ela faz com que trechos de códigos possam ser executados diretamente na parte do cliente sem a necessidade de se fazer uma requisição para o servidor, ela pode fazer com que a páginas *web* possam ter conteúdos mais dinâmicos é considerada uma linguagem orientada a objeto.

Com a necessidade de se criar grandes funções *JavaScript* para solucionar pequenos problemas e na dificuldade de programar recursos complexos foram criados *frameworks* de *JavaScript* considerando os mais importantes como JQuery e AnjugarJS (DUCKETT, 2010).

5.1.4 Twitter Bootstrap

Twitter Bootstrap é um *framework* CSS e *JavaScript* que disponibiliza diversas ferramentas para a interface do usuário assim como *layouts*, plugins *JQuery* e elementos de entrada e saída de dados. O *Twitter Bootstrap* é considerado um dos códigos aberto mais populares do *GitHub,* foi lançado em 2011 e teve seu grande sucesso por ser um *framework* que integra CSS com *JavaScript* e por ter seu *layout* responsivo, as maiores vantagens do *Twitter Bootstrap* podem ser consideradas como (NISAKA, 2014):

a) Suporte entre os navegadores: o Bootstrap funciona em todas as últimas versões dos navegadores *desktop* e *mobiles*, assim como navegadores antigos podem apresentar pequenas mudanças em seus estilos ainda é totalmente funcional como no *Internet Explorer* 8.
b) Fácil de personalizar: ele é fácil de ser personalizado especialmente por utilizar apenas o que é necessário deixando utilizando apenas uma *grid* e deixando de fora todos os outros componentes.
c) Suporte a JQuery: o *Bootstrap* vem com diversos plugins de *JQuery*, que podem ser úteis em diversas situações, não são os melhores porém funcionam melhor se não forem personalizados.
d) Mobile-first: o Bootstrap tem o conceito de *Mobile-first* desde a versão 3.0, isso significa que a tela é totalmente responsiva quando a tela for aumentado os componentes também aumentam e quando for diminuindo os componentes também diminuem.

5.1.5 Primefaces

Primefaces é uma biblioteca de componentes de código aberto para JSF com mais de 100 componentes prontos, para criar aplicações *web*, de uma forma simples e eficiente, ela foi criada para trabalhar com AJAX por padrão, ou seja, não é necessário o esforço do desenvolvedor para trabalhar com requisições assíncronas ao servidor, uma das grandes vantagens é a simplicidade da utilização da biblioteca, basta apenas colocar o *jar* do primefaces dentro do projeto pois ela

não possui nenhuma dependência, não necessita nenhuma instalação (ÇALISKAN;VARAKSIN, 2013).

Para utilizar o primefaces é necessário que o desenvolvedor declare o seu *namespace*, na sua página sendo ele: “xmlns:p=http://primefaces.org/ui”, declarado o namespace como “p”, para utilizar os componentes é só utilizar a ta “<p:” e o componente, por exemplo o componente de entrada de texto: “<p:inputText />” (ÇALISKAN; VARAKSIN, 2013).

5.2 SERVER SIDE

Se o *Client Side* é responsável pela exibição dos dados para o cliente o *Server Side* é responsável pelo processamento dos dados, as principais vantagens da utilização do servidor de aplicação é a utilização dos softwares necessários para a aplicação abstraindo assim a necessidade do usuário saber quais destes sãos utilizados e sem precisar ser instalados na máquina do usuário (BURKE, 2010).

5.2.1 Servidores de aplicação

Servidor de aplicação é o servidor que disponibiliza o ambiente para execução das aplicações, o objetivo dos servidores é disponibilizar uma plataforma que não leve em consideração as complexidades de um sistema operacional, também fornecem a infraestrutura para execução de aplicações distribuídas, no desenvolvimento de aplicações comerciais deve-se ter o foco de resolver os problemas de negócios e não de infraestruturas, as principais características são (HEFFELFINGER, 2014):

a) Tolerância a falhas: é a propriedade que mantém o sistema operando mesmo após falhas na aplicação, esses erros podem ser *bugs* e erros de desenvolvimentos.
b) Balanceamento de cargas: *Load Balance* como é chamado promove a distribuição das cargas para o aumento de performance, ou seja, distribui o tráfego das chamadas para as demais máquinas que compõe o cluster para funcionarem como uma só.

c) Gerenciamento de componentes: funciona como gerenciador do servidor manipulando serviços, componentes, notificações e distribuição da lógica do negócio, assim como o gerenciamento de sessão.
d) Gerenciamento de transações: serve para manter a integridade entre as aplicações rodando dentro do servidor, assim como aplicativos e banco de dados.

Existem várias implementações para servidores de aplicação na sua maioria implementada na linguagem Java sendo elas *Red Hat Jboss, Sun GlassFish, Apache Tomcat*.

5.2.2 Java

A linguagem foi criada em 1991, sendo baseada na linguagem C++ e com o propósito de ser uma linguagem que oferecia a orientação a objeto, ser simples e pequena, a popularidade e a aceitação da linguagem foram pelo fato dela ser considerada uma linguagem multiplataforma, a linguagem é rodada em cima de uma *Java Virtual Machine* (JVM), ou seja, qualquer Sistema operacional que consiga executar a JVM pode utilizar os programas feitos em Java (ECKEL, 2006).

A primeira versão a ser lançada foi a 1.02 em 1996, não foi considerada uma linguagem de nível corporativo pois não havia nela alguns recursos básicos e por ser muito lenta, só foi considerada de um nível para ser utilizada no corporativismo a partir das versões 1.1 e 1.2 onde foram corrigidos vários erros e adicionados diversas funcionalidades (POTHU, 2008).

Atualmente Java é a segunda linguagem de programação mais utilizada do mundo estando na versão 1.7, Java possui três versões, *Java Standart Edition* versão para desktop, *Java Enterprise Edition (EE)* versão para *web* e *Java Micro Edition* para dispositivos móveis (TIOBE, 2012).

5.2.3 Java EE

Java EE teve o início em 1990 e trouxe para o desenvolvimento corporativo um grande leque de ferramentas para o desenvolvimento, é uma plataforma com base no desenvolvimento de aplicações *web*, essas aplicações são compostas por diversas camadas, uma camada de interface sendo definida por um *framework*, uma camada intermediária que é responsável por segurança e transações e uma camada para o controle de banco de dados (GUPTA, 2013).

O Java EE consiste em várias especificações que dizem como cada serviço deve se comportar, ou seja, podem haver vários serviços para a mesma especificação e a aplicação deverá funcionar independentemente de qual serviço for escolhido. A versão 7 do Java EE foi lançada em junho de 2013 e com ela veio uma melhora significativa quanto a facilidade e usabilidade de construir aplicações *web* (GUPTA, 2013).

5.2.4 JAVA PERSISTENCE API – JPA

A *Java Persistence* API (JPA) é a especificação para framework de Mapeamento Objeto-Relacional (ORM) do Java, para a implementação da especificação JPA as mais utilizadas são Hibernate e EclipseLink, onde o Hibernate foi criado antes da especificação porém a implementação oficial é o EclipseLink. A funcionalidade do Hibernate se dá ao abstrair o código SQL onde o mesmo é gerado em tempo de execução por isso é possível mudar o banco de dados sem ter que mudar o código da aplicação (POTHU, 2008).

A relação de tabelas SQL para o Java é feito através de entidades que são classes Java que representam as tabelas do banco onde cada atributo da classe representa uma coluna do banco, todas as configurações assim como dialeto do banco de dados, cache mapeamento de classes são feitas através de um arquivo XML chamado *persistence*, já os comandos do banco como, *insert, delete* e *update* são feitas através do *EntitManager* (POTHU, 2008).

5.2.5 JAVA SERVER FACES – JSF

O JSF é um *framework Model View Controller* (MVC) que faz parte das especificações do Java EE, baseado na linguagem Java, baseado na construção de componentes para aplicações *web*, utiliza um padrão programação a base de eventos, a versão mais atual é a 2.2 lançada em abril de 2013.

Ela trabalha no modelo de arquitetura MVC, ou seja, a aplicação é distribuída em três partes, o *Model* representa todas as entidades e regras de negócios da aplicação, o *View* representa todas as páginas de visualização e o Controller representam as classes intermediarias entre as entidades e as telas de visualizações que funcionam como uma forma de ligação entre as duas.

Quando o cliente faz uma requisição para uma página JSF, o servidor responde com uma página traduzida para HTML, o que acontece é que durante a requisição o JSF é divido em fases chamadas de ciclo de vida do JSF sendo elas (GEARY, HORSTMANN, 2010):

a) *Restore View:* primeira fase do JSF, responsável por construir a visualização da página.

b) *Apply Request Values:* fase onde os valores dos componentes são criados ou recuperados e convertidos.

c) *Procces Validation*: aqui é onde feito as validações dos componentes dispostos na tela.

d) *Update Model Values:* essa fase faz com que os valores do lado do servidor sejam atualizados com os da tela.

e) *Invoke Application:* fase responsável por executar as ações definidas, como por exemplo a navegação ao clicar em algum botão.

f) *Render Response:* fase que exibe os valores do lado do servidor na tela com os valores atual.

6 TRABALHOS CORRELATOS

A integração de saúde e informática está presente em qualquer segmento da saúde, nesse capítulo serão descritos alguns trabalhos que utilizam saúde e informática e que serviram de base para essa pesquisa.

6.1 GOOGLE ANDROID: UM NOVO PARADIGMA DE DESENVOLVIMENTO MOBILE APLICADO AO GERENCIAMENTO MÉDICAS REFERENTE AO CONTROLE DO DIABETES

Este foi um Trabalho de Conclusão de Curso em Ciência da Computação, foi utilizado como base para o desenvolvimento da pesquisa. Foi desenvolvido por Leonardo Alves Neuwald apresentado na Universidade do Extremo Sul Catarinense no ano de 2012, onde o objetivo era o desenvolvimento de uma aplicação para dispositivos móveis onde pudesse fazer o gerenciamento de pacientes com diabetes, realizando sincronização de serviços com médicos ou profissionais da saúde.

A solução encontrada foi o desenvolvimento de duas aplicações o WS que funciona para disponibilizar serviços entre pacientes e médicos e o aplicativo “Diabetes Control” que é separado em dois módulos, o módulo do paciente e do médico onde no módulo do paciente ocorre o cadastro de informações relacionados a diabetes e o módulo do médico onde recebe as informações do paciente e realiza os diagnósticos referente as informações do paciente (NEUWALD, 2012).

6.2 M-COMMERCE E A PLATAFORMA GOOGLE ANDROID

Este foi um Trabalho de Conclusão de Curso em Ciência da Computação desenvolvido por Diego Gomes Antoneli apresentada na Universidade do Extremo Sul Catarinense no ano de 2011. O objetivo era criar um protótipo de sistema de comércio utilizando dispositivos móveis Android e sincronização de serviços Web Services.

Para o desenvolvimento do protótipo foram utilizando diversas tecnologias para *Web Services.* Foi utilizado Java API for XML Web Services (JAX-WS) e o banco de dados MySQL. Já na parte dos dispositivos móveis foi utilizado a Linguagem de programação Java e XML, que são consideradas padrões de desenvolvimento para aplicativos Android.

Para o objetivo do trabalho foi desenvolvido duas aplicações, uma para dispositivos móveis e outra para sincronização de sistemas, no qual foi alcançado o objetivo final do trabalho que era a implementação de um modelo m-commerce para dispositivos móveis (ANTONELI, 2011).

6.3 SISTEMA DE APOIO AO DIAGNÓSTICO MÉDICO UTILIZANDO TECNOLOGIAS MÓVEIS

Este foi um Trabalho de Conclusão de Curso em Ciência da Computação, desenvolvido por Marcus Vinicius Galvão Rehm, apresentada na Universidade Federal da Bahia no ano de 2005. O objetivo do trabalho proposto foi o desenvolvimento de um aplicativo para dispositivos móveis que pudessem solucionar o problema de atendimento em municípios do interior do Brasil onde a falta de médicos é visível e na maioria dos casos os enfermeiros ficavam responsáveis pelos pacientes.

A solução encontrada foi os enfermeiros disponibilizarem os sintomas dos pacientes via Web Services em que do outro lado os médicos utilizavam essas informações para diagnosticar os pacientes e então os enfermeiros utilizariam o diagnóstico dos médicos para realizarem os procedimentos necessários, assim criando um canal de comunicação entre médico e enfermeiro de lugares distintos (REHM,2005).

7 DIABETES CONTROL WEB – INTERFACE DE GERENCIAMENTO

A presente pesquisa tem como resultado obtido um sistema em plataforma web que realiza o controle sobre os indicadores da diabetes dos pacientes. Junto com esse controle o sistema também prevê a integração com o aplicativo *Diabetes Control,* que por sua vez foi desenvolvido em versão *mobile*, e alterado para atender as necessidades do sistema. Sendo assim o sistema recebe dois tipos de informações, aquelas geradas pelo usuário diretamente pela interface, e aquelas que são cadastradas previamente no aplicativo mobile sendo sincronizada com o aplicativo web via *Web Service*.

Tanto no aplicativo *Diabetes Control* quanto no sistema *Diabetes Control Web* os dados cadastrados de pacientes e médicos se comunicam de tal forma que os médicos possam avaliar em tempo real os registros de indicadores de diabetes e tomarem as decisões necessárias em cada caso. Diversas ferramentas e tecnologias foram utilizadas para alcançar os objetivos propostos, como por exemplo: Java, Android, JPA, JSF, entre outras citadas posteriormente na metodologia.

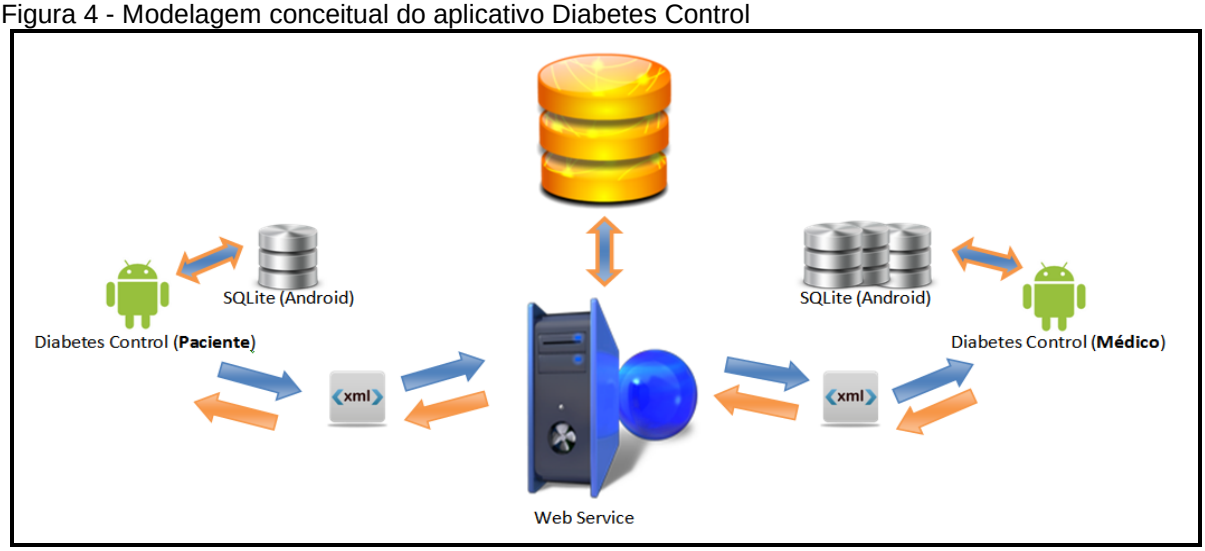
7.1 METODOLOGIA

O objeto de estudo deste trabalho, aborda a construção de um aplicativo e de um processo de integração computacional que aborda uma doença crítica a qual ataca a população nas mais diversas classes sociais. Esta doença é conhecida como diabetes. A utilização de sistemas automáticos de gerenciamento que auxiliem no controle dessa doença, proporciona uma melhor qualidade de vida aos portadores, bem como diminui a incidência de problemas mais graves, ocasionados pelo mau gerenciamento dos índices de glicemia dos pacientes diabéticos. Desta forma, este trabalho aborda como metodologia as fases descritas no fluxo de informações abaixo.

7.1.1 Fluxo de Informações

O fluxo de informações do aplicativo *Diabetes Control* consiste na troca de informações entre dispositivos e serviços e pode ser apresentado da seguinte forma.

Figura 4 - Modelagem conceitual do aplicativo Diabetes Control

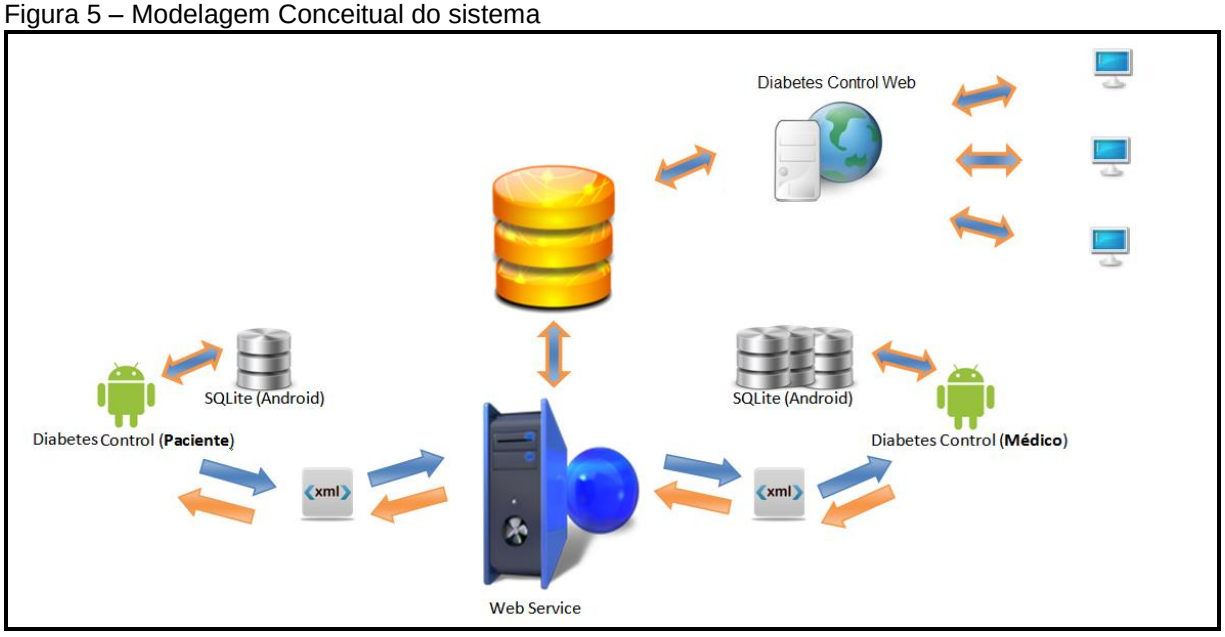


Fonte: Neuwald (2012).

Na figura 4, pode-se observar duas aplicações distintas: a aplicação para pacientes e para médicos. Ambas as aplicações são responsáveis por ler e gravar os registros inseridos através de sua interface. O *Web Service* que é responsável pelo processo de sincronização dos registros onde os pacientes e médicos informam seus dados e o *Web Service* tem a responsabilidade de sincronizar estes dados e, caso os registros que forem informados no momento onde o usuário se encontra fora da área de acesso à Internet, os registros são salvos e assim que ocorrer o acesso à Internet são sincronizados, apresentando uma característica assíncrona para a aplicação proposta. Caso o aplicativo esteja conectado à Internet, a sincronização dos dados do paciente com médico é realizada em tempo real.

O *Diabetes Control Web,* apresenta em seu fluxograma três aplicações distintas, duas já citadas anteriormente, e a terceira é apresentada como *Diabetes Control Web* que aborda a integração com o banco de dados do *Web Service* responsável pela sincronização dos registros de paciente e médico, fazendo assim o controle via banco de dados para o *Web Service* e depois do *xml* para as aplicações. uma visão do esquema proposto pode ser observada na figura 5.

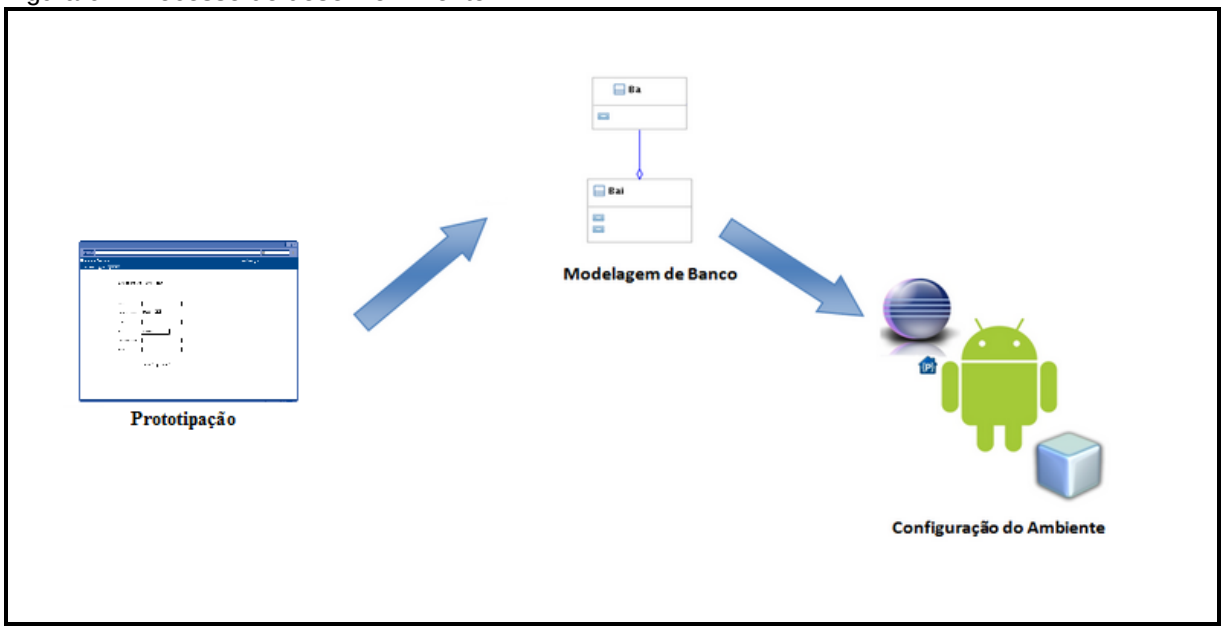
Figura 5 – Modelagem Conceitual do sistema



Fonte: Do autor.

O desenvolvimento do sistema deu-se após o término da modelagem conceitual e da prototipação, a figura 6 ilustra o fluxograma do processo de desenvolvimento.

Figura 6 – Processo de desenvolvimento



Fonte: Do autor.

Inicialmente foi feito a prototipação das telas do sistema levando em consideração o aplicativo *Diabetes Control*, foram prototipadas alguma das telas do sistema.

Logo após foi feito a modelagem do banco, que foi divido em dois, o do aplicativo *Diabetes Control* e o do sistema *Diabetes Control Web*, foi utilizado o mesmo modelo do aplicativo e alterado o modelo do Web Service para atender as necessidades do sistema desenvolvido.

A última etapa se tratou de configurar os ambientes de desenvolvimento, escolha das bibliotecas, ferramentas e frameworks utilizados. Cada etapa será descrita na modelagem.

7.2 MODELAGEM

Um requisito mínimo para o desenvolvimento do sistema é a modelagem, pela importância do potencial de detectar problemas no desenvolvimento futuro, além de atender os conceitos de engenharia de software, levando em consideração a qualidade do produto final.

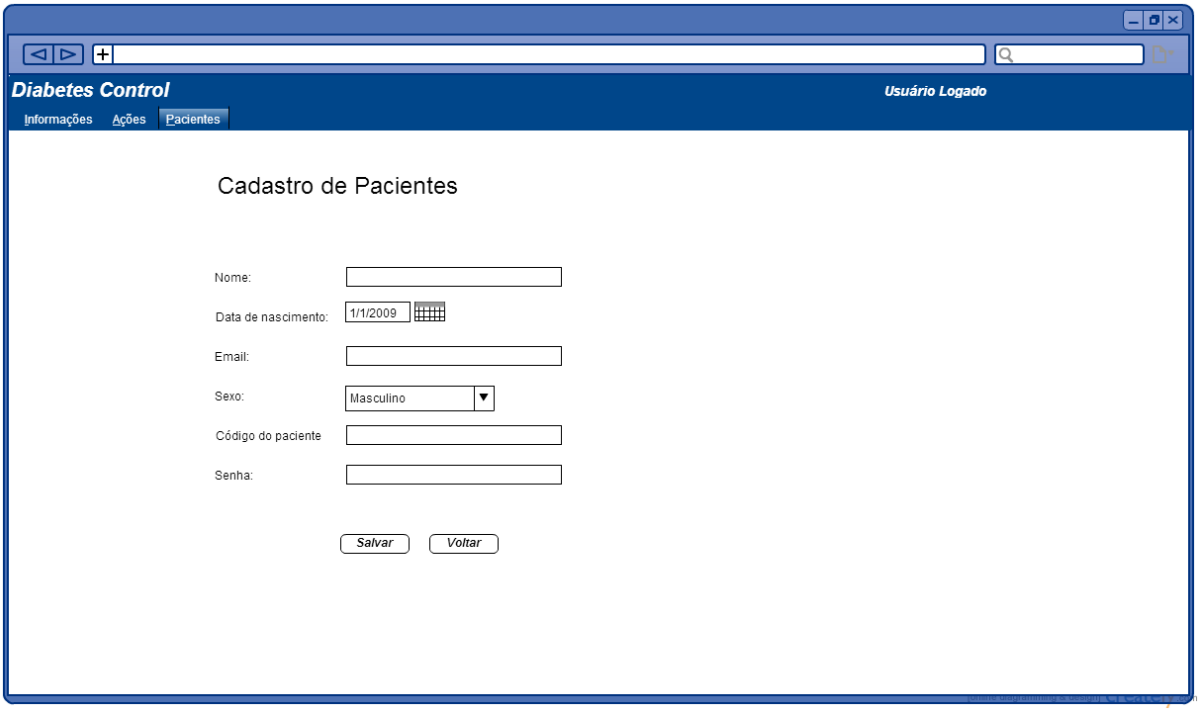
Para as modelagens do *Diabetes Control Web* foi escolhido utilizar a modelagem lógica para as tabelas do banco de dados e prototipação para as telas do sistema, para o melhor entendimento no fluxo e navegações entre as telas do sistema.

7.2.1 Prototipação

Para a modelagem de prototipação foi utilizado a ferramenta *creately*, está por sua vez disponibiliza ferramentas para modelagem de bancos e diagramas de engenharia de software, foi escolhida por ser gratuita e disponibilizar diversas ferramentas assim facilitando a prototipação do sistema, para utilizar o conceito de *mock-up* que trata-se de um conceito onde é feito uma apresentação visual do projeto a ser desenvolvido, podendo não receber todos os elementos finais do produto a ideia é do mock-up é de como ficará a parte gráfica do produto final.

A figura 7 representa a tela de cadastro de pacientes que foi a primeira a ser prototipada.

Figura 7 - protótipo da tela de cadastro de paciente

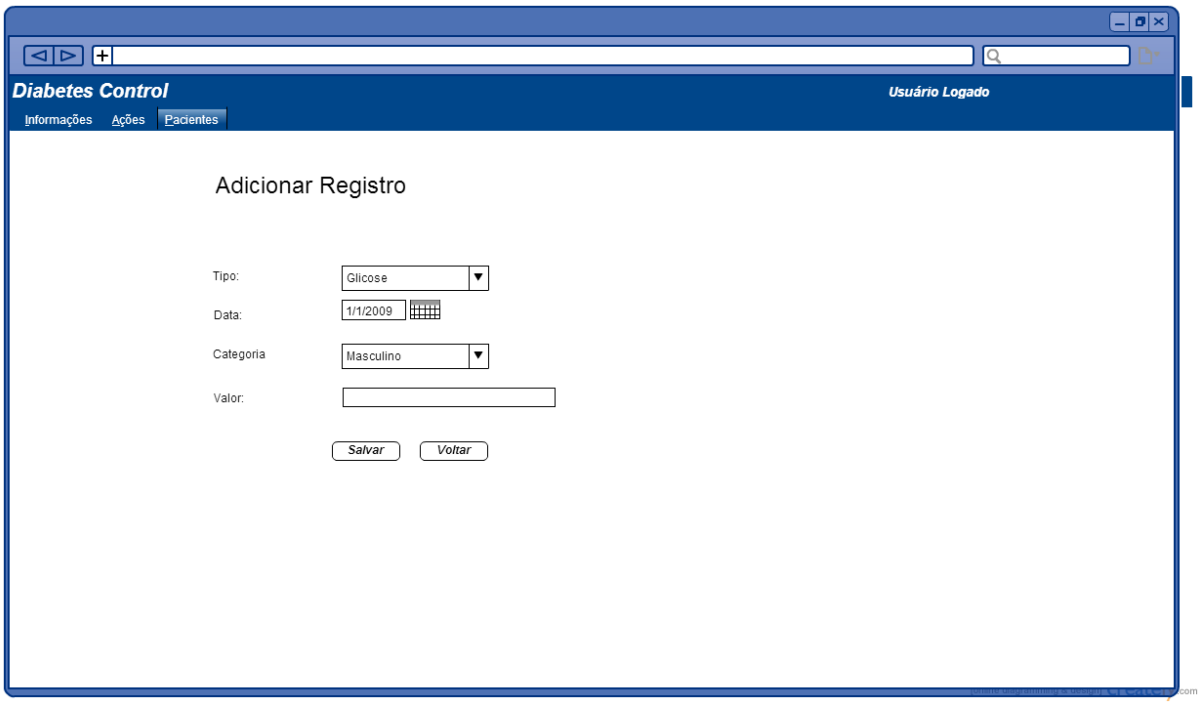


Fonte: Do autor.

Por meio desta tela é onde serão informados os dados do paciente a ser cadastrado no sistema. Por meio deste cadastrado são informados: nome, data de nascimento, email, sexo, código do paciente e senha, serão utilizados tanto na tela de login do sistema quanto para fazer a integração do médico com o paciente.

Com o paciente cadastrado pode-se cadastrar os registros onde é considerado uma das duas telas mais importantes do sistema, pois é onde o paciente cadastra todas as índices de diabetes, conforme figura 8.

Figura 8 – Prototipação da tela de Adicionar Registro



Fonte: Do autor.

A figura 8 demonstra a tela onde é feito todos os cadastros de indicativos da diabetes, o campo tipo que é uma *combobox* possui as opções de: medida de glicose, peso, pressão, pulso, aplicação de insulina e medicamentos ingeridos, os campos data e valor são obrigatórios para ser possível realizar o cadastro do registro, esta tela só acessada pelo paciente.

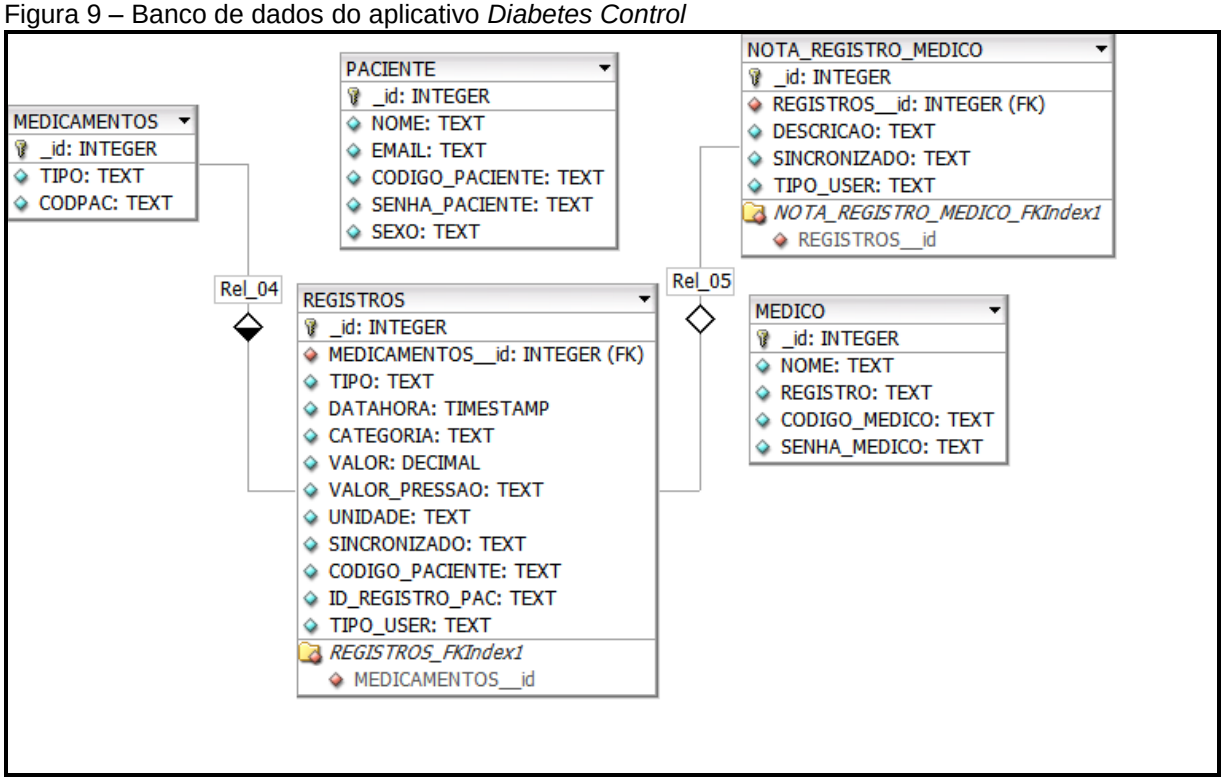
Logo após ser cadastrado, os registros podem ser visualizados na tela de listagem de registros. A tela de listagem de registros que pode ser visualizada tanto por médicos como por pacientes, nela pode se editar o registro ver e adicionar notas, com a visualização do protótipo e dos elementos contidos nele ficou mais claro e simples de como deveria ser feito todo o desenvolvimento das telas.

7.2.2 Modelagem de Banco de Dados

A modelagem de dois bancos de dados deu-se por haver duas aplicações diferentes o aplicativo *Diabetes Control* e o sistema *Diabetes Control Web* que utiliza o mesmo banco do *Web Service*, a modelagem física foi adaptada do banco já existente do *Web Service*, sendo o do aplicativo não modificado.

A figura 9 apresenta o modelo físico do banco de dados presente no aplicativo *Diabetes Control*, já o do Diabetes Control apresenta as chaves primárias todas como *_id*.

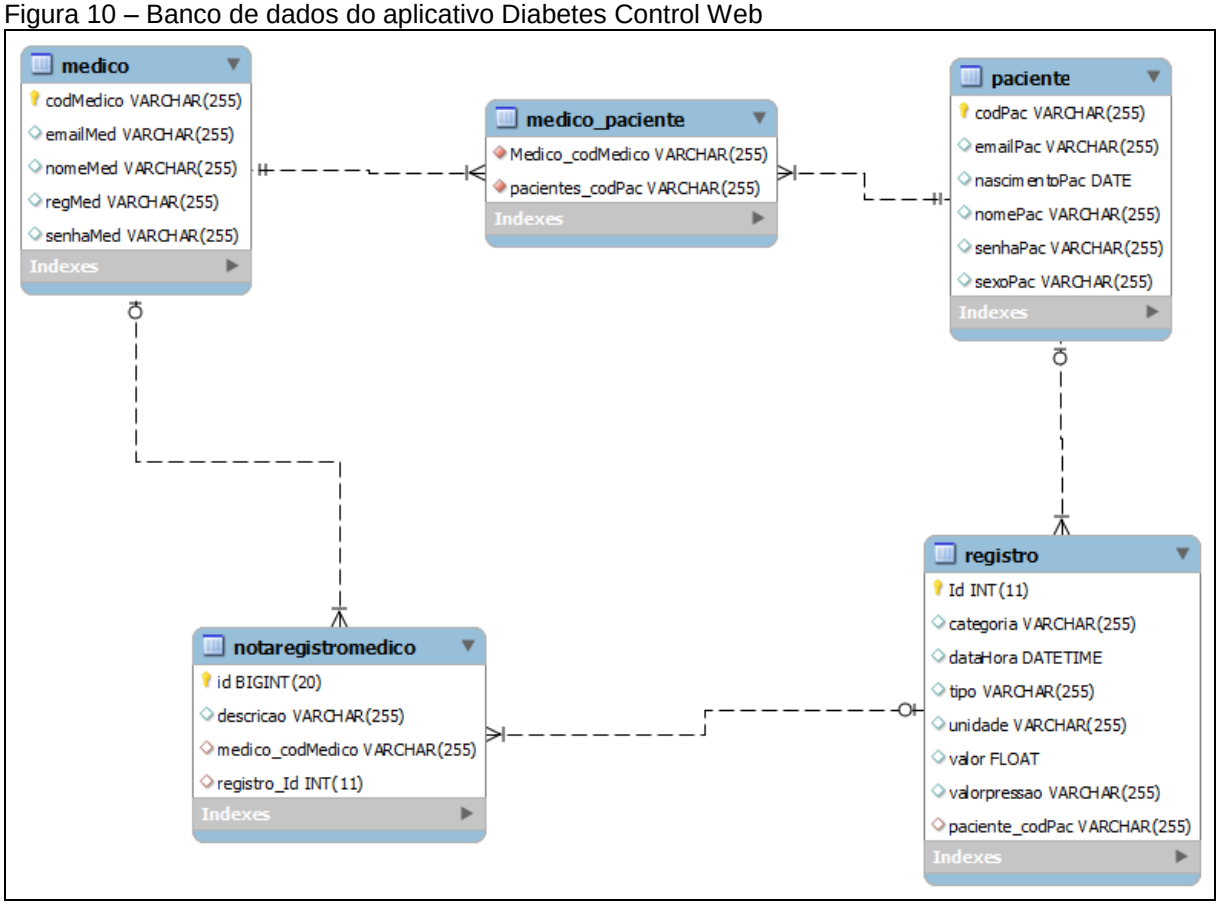
Figura 9 – Banco de dados do aplicativo *Diabetes Control*



Fonte: Neuwald (2012).

Para o banco de dados do *Web Service* e do *Diabetes Control Web*, foi feito o relacionamento de muitos para muitos da tabela Médico e Paciente, e diferenciado o nome das chaves primárias de cada tabela.

Figura 10 – Banco de dados do aplicativo Diabetes Control Web



Fonte: Do autor.

A figura 10 representa o modelo do banco de dados utilizado no sistema *Diabetes Control Web*, que diferente do Diabetes Control possui.

7.2.3 Ambiente de Desenvolvimento

Com o banco de dados modelado e com os protótipos desenvolvidos o próximo passo foi a configuração dos ambientes de desenvolvimento e ferramentas, para que com isso fosse possível obter os melhores resultados.

Foram criados dois ambientes distintos, um para o *Diabetes Control* utilizando o *ADT* no Eclipse e outro para o *Diabetes Control Web e Web Service* utilizando Netbeans, no *Diabetes Control* foi utilizado o SQLite como gerenciador de banco de dado e no *Web Service* e *Diabetes Control Web* foi utilizado MySQL.

Tanto no *Web Service* como no *Diabetes Control Web* foi necessário a configuração do servidor de aplicação. No *Web Service* foi utilizado *GlassFish* e no

Diabetes Control Web Jboss versão 7.0. Dentro do servidor Jboss foi necessário a configuração do data source para utilização do *Java Transaction API* (JTA), fazendo assim o servidor de aplicação se tornar responsável pelas transações feitas pelo JPA, fechando e abrindo transações automáticas sem a necessidade de serem feitas manualmente.

Figura 11 – Diferença entre os métodos salvar do *Web Service* e do *Diabetes Control Web*

```
public Paciente salvar() {
    EntityManager em = EM.getEntityManager();
    EntityTransaction transaction = em.getTransaction();
    transaction.begin();
    Paciente pac = em.merge(this);
    transaction.commit();
    em.close();
    return pac;
}
```

```
@TransactionAttribute(TransactionAttributeType.REQUIRED)
public void adicionarNovoPaciente(Paciente paciente) {
    manager.persist(paciente);
}

@TransactionAttribute(TransactionAttributeType.REQUIRED)
public void mergePaciente(Paciente paciente) {
    manager.merge(paciente);
}
```

Fonte: Do autor.

A figura 11 mostra a diferença dos métodos de salvar do *Web Service* e dos métodos adicionarNovoPaciente e mergePaciente do *Diabetes Control Web*. No método salvar é necessário todo o controle da transação para poder realizar o método merge do entityManager, já nos outros dois métodos só é informado com a anotação *@TransactionAttribute(TransactionAttributeType.Required*) que para realizar esse método é necessário uma transação, e o próprio servidor de aplicação se encarrega de todas as funcionalidades da transação.

7.3 IMPLEMENTAÇÃO

Após a definição da modelagem e todas as ferramentas a serem utilizadas, esta etapa apresenta a implementação do sistema.

O início do desenvolvimento deu-se a partir do sistema *Web,* sem a necessidade de integração do aplicativo *Diabetes Control* pelo *Web Service*. O sistema é capaz de gerenciar os dados referentes aos índices da diabetes do paciente. Para o desenvolvimento *Web* foi utilizado o padrão MVC, onde os arquivos *xhtml*, representam a camada de *view,* as entidades representam a camada de *model,* e os controllers representam a camada de *controller.*

Foram criadas as entidades Paciente, Médico, Registro, NotaRegistroMedico, que representam respectivamente, as informações dos

pacientes, as informações do médico, os indicativos de diabetes e as notas que os médicos cadastram para os registros dos pacientes. Com base nas entidades foram criados dois Data Acces Object (DAO), PacienteDAO, MedicoDAO que tem como função a conexão da aplicação com o banco de dados, sendo o PacienteDAO responsável pelas entidades Paciente e Registros e o MedicoDAO responsável pelas entidades Medico e NotaRegistroMedico.

Para o relacionamento das entidades foi utilizada as anotações do *JPA @ManyToMany* para o relacionamento de muitos pra muitos da tabela de Paciente para a tabela de Medico, foi utilizado o *@ManyToOne* da tabela de Registro para a tabela de Paciente e @oneToOne da tabela de Registro para a tabela de NotaRegistroMedico, *@Id* identifica a chave primária da entidade, @GeneratedValue represeta o modo que será automaticamente a chave primaria, utilizando o generator=”SEQ_REGISTRO” ele utiliza a tabela *SEQ_REGISTRO* para a criação da chave primaria e utilizando o *GenerationType.SEQUENCE* é utilizado o método de sequência que consiste em pegar o valor que será gravado na tabela *SEQ_REGISTRO* mais um, ou seja, cada vez que a entidade Registro for salva no banco o JPA irá adicionar mais um na tabela *SEQ_REGISTO*, a figura 12 mostra as anotações na entidade Registro.

Figura 12 – Anotações do JPA na entidade Registro

```
22        @Id
23        @GeneratedValue(generator = "SEQ_REGISTRO", strategy = GenerationType.SEQUENCE)
24        private Integer Id;
25        private String tipo;
26        private String categoria;
27        private Float valor;
28        @Temporal(javax.persistence.TemporalType.TIMESTAMP)
29        private Date dataHora;
30        private String unidade;
31        private String valorpressao;
32        @ManyToOne
33        private Paciente paciente;
34        @OneToOne(mappedBy = "registro")
35        private NotaRegistroMedico nota;
```

Fonte: Do autor.

Os controladores do *Diabetes Control Web* são: PacienteController e MedicoController. O PacienteController tem a responsabilidades de fazer a ligação entre as entidades Paciente e Registro com as telas de cadastros e listagens. Já o MedicoController faz a ligação das entidades Medico e NotaRegistroMedico com os

cadastros e listagens. O controle de sessão do usuário é feito através do controller userSession que é responsável pelo controle de sessão do sistema, ele tem as informações de login e do usuário logado no sistema.

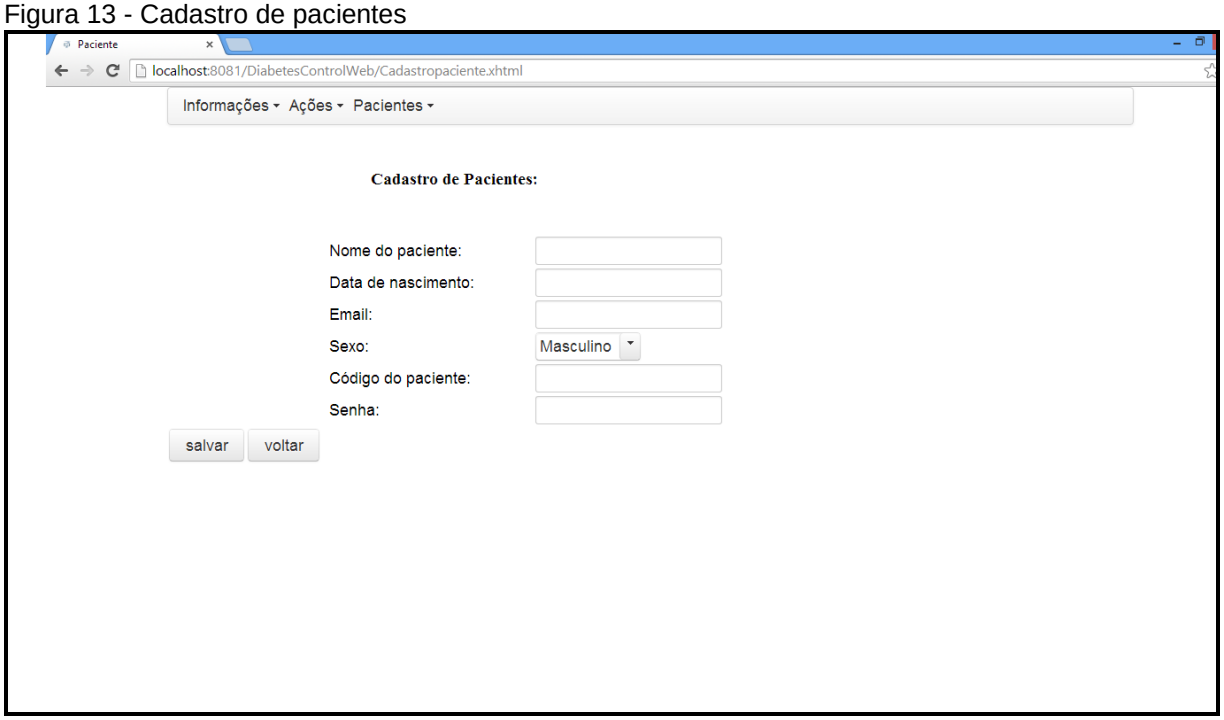
Para a parte gráfica do sistema utilizou-se a biblioteca *Primefaces* na versão *4.0*. Trata-se de uma biblioteca que é utilizada em conjunto com JSF para a evolução dos componentes que vão muito além das especificações, utilizou-se essa biblioteca por causa de sua facilidade de uso, por ser uma biblioteca leve, ter uma simples instalação e pela sua grande comunidade que proporciona um grande suporte para os problemas encontrados, todos os componentes utilizados no sistema são componentes da biblioteca.

Para o controle de *URL* foi implementado o LoginListener que verifica em cada página acessada se já existe um usuário logado caso não exista o usuário será direcionado para a página de login para poder acessar o sistema.

A integração do *Diabetes Control* com o *Diabetes Control Web*, deu-se através do *Web Service*. Este está em um projeto a parte e só foi modificado para poder ser integrado com o sistema desenvolvido. Foi alterado o provedor JPA de *EclipseLink* para *Hibernate*, adicionado as anotações para o controle no banco, alterado os métodos de busca e disponibilização do *entityManager* para o controle de transações.

A figura 13 apresenta a tela onde os médicos podem cadastrar novos pacientes para serem gerenciados pelo sistema, Nela é informado os campos: nome do paciente, data de nascimento, email, sexo, código do paciente e senha. Esta tela é acessada pelo menu Cadastrar paciente.

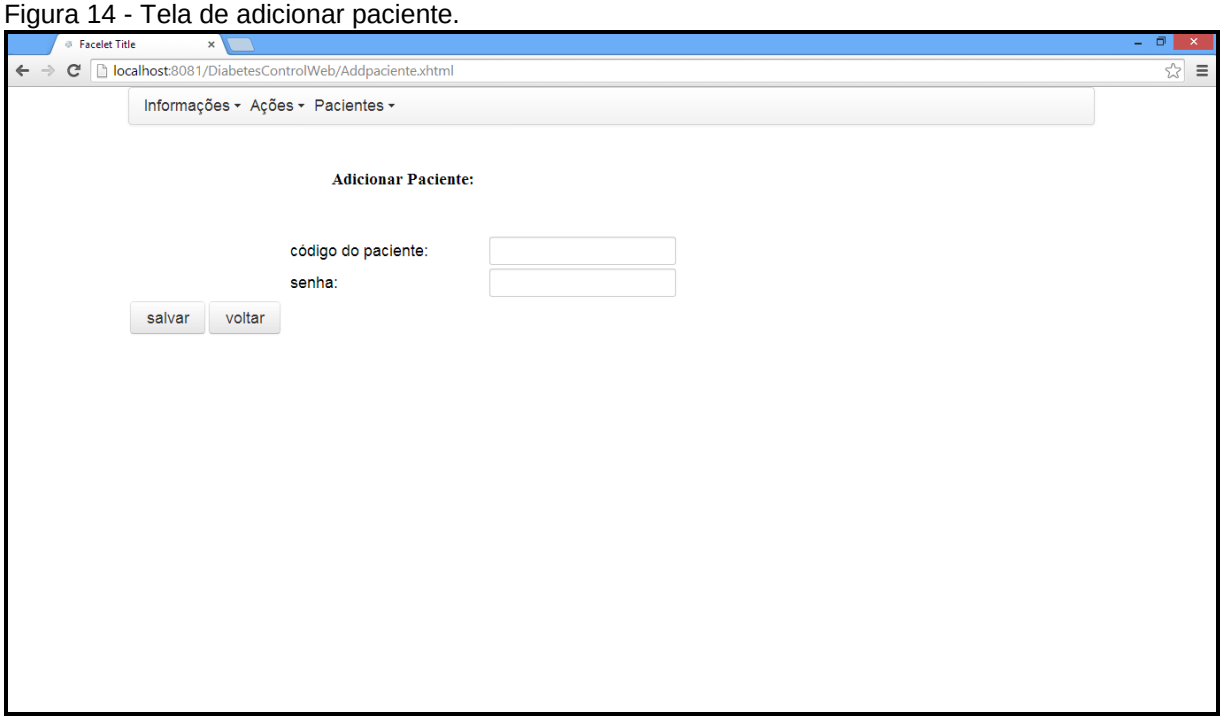
Figura 13 - Cadastro de pacientes



Fonte: Do autor.

A tela de adicionar paciente serve para o médico informar os dados do paciente que ele deseja gerenciar. Somente a partir deste momento o médico consegue visualizar no sistema os dados do paciente. A figura 14 mostra a tela onde o médico adiciona os pacientes.

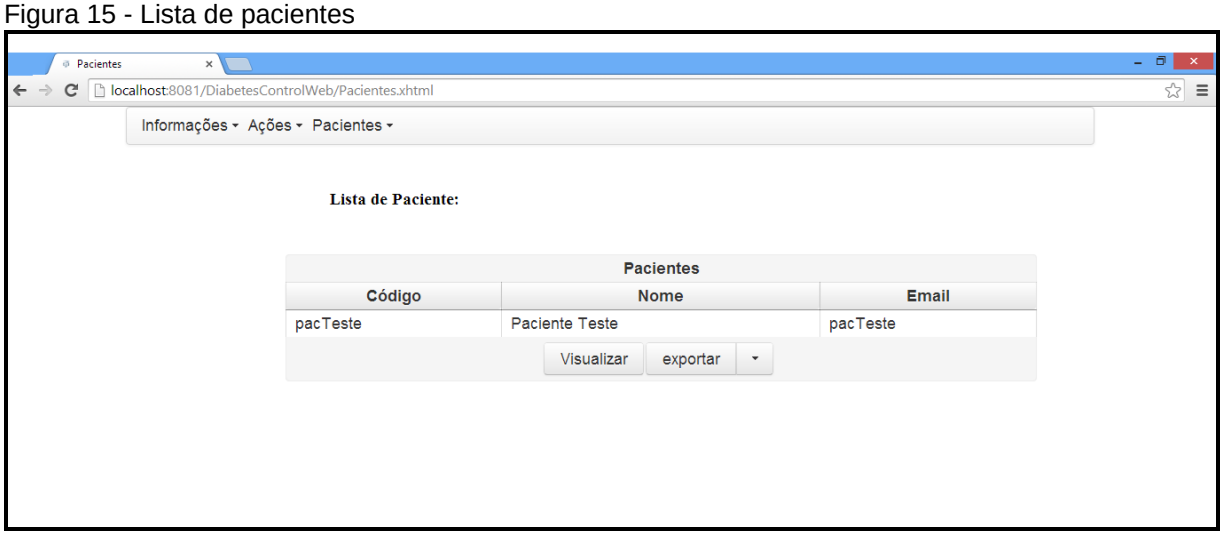
Figura 14 - Tela de adicionar paciente.



Fonte: Do autor.

Com o paciente adicionado ele então começa a ser gerenciado, agora o paciente está sendo listado na tela Pacientes, onde se encontram duas funcionalidades a de exportar, que exporta o conteúdo da tabela em arquivos do tipo Portabel Document Format (PDF), Excel(XLS), Comma-separeted Values (CSV) e XML, para assim os dados possam ser integrados com outros aplicativos, e a função de visualizar, que possibilita o médico visualizar ao paciente selecionado. A figura 15 representa a tela de listagem de pacientes.

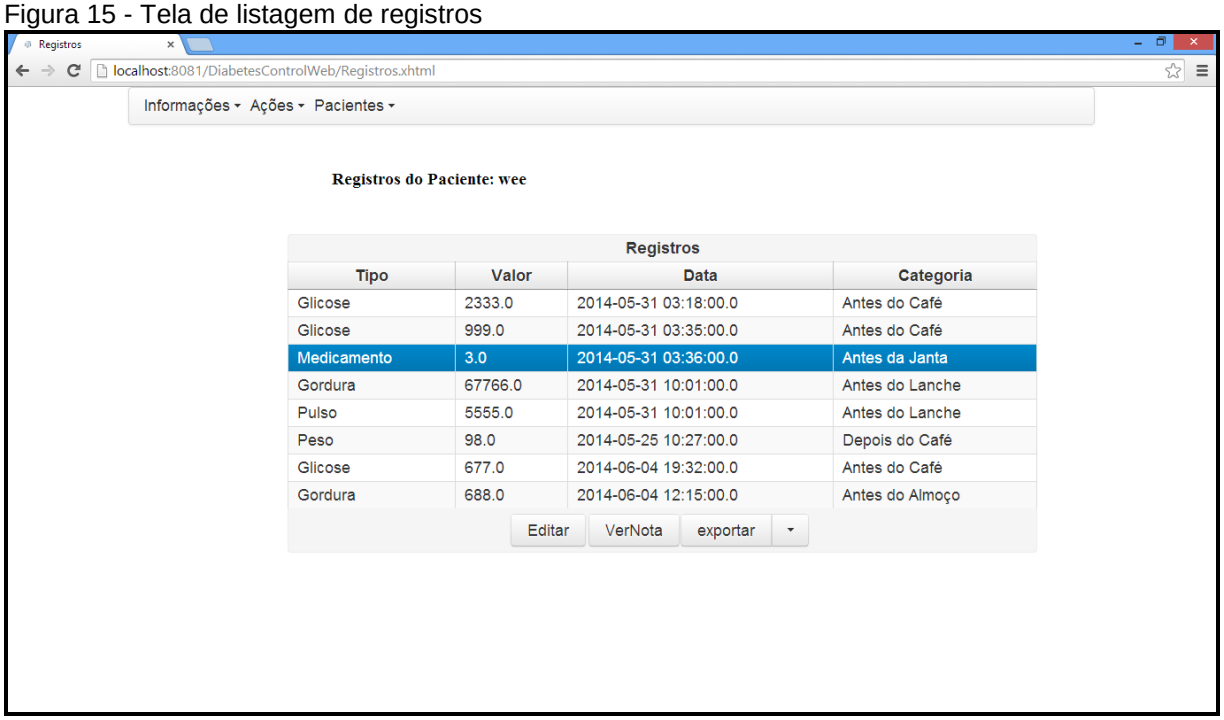
Figura 15 - Lista de pacientes



Fonte: Do autor.

Visualizando o paciente selecionado o médico abre a tela de visualização de pacientes, nesta tela é demonstrado todos os dados cadastrais do paciente: nome, data, email, sexo e código. Logo abaixo tem-se o botão de registros que redireciona para página de listagem de índices de diabetes do paciente selecionado, esta página pode ser acessada tanto por médico quanto por pacientes, por médicos a partir do fluxo descrito acima, e por pacientes pelo menu Lista de registros, nela contém as informações cadastradas na tela de cadastro de registros, estes registros são: tipo, valor, data e categoria. A listagem possui três funcionalidades a de exportar que exporta o conteúdo da tabela em arquivos do tipo PDF, XLS, CSV e XML, a de ver nota que possibilita ser visualizado as notas que o médico cadastrou para aquele registro do paciente selecionado em especifico, estas notas são visualizadas através de um *Dialog box* que é aberto quando o usuário seleciona a funcionalidade de ver nota. E por último a edição que leva o paciente para a tela gerenciamento dos indicadores de diabetes. Caso o usuário seja um médico é adicionado a funcionalidade de adicionar nota no lugar da de edição de registros, sendo redirecionado para a tela de adicionar notas. A figura 16 representa a tela de listagem de registros.

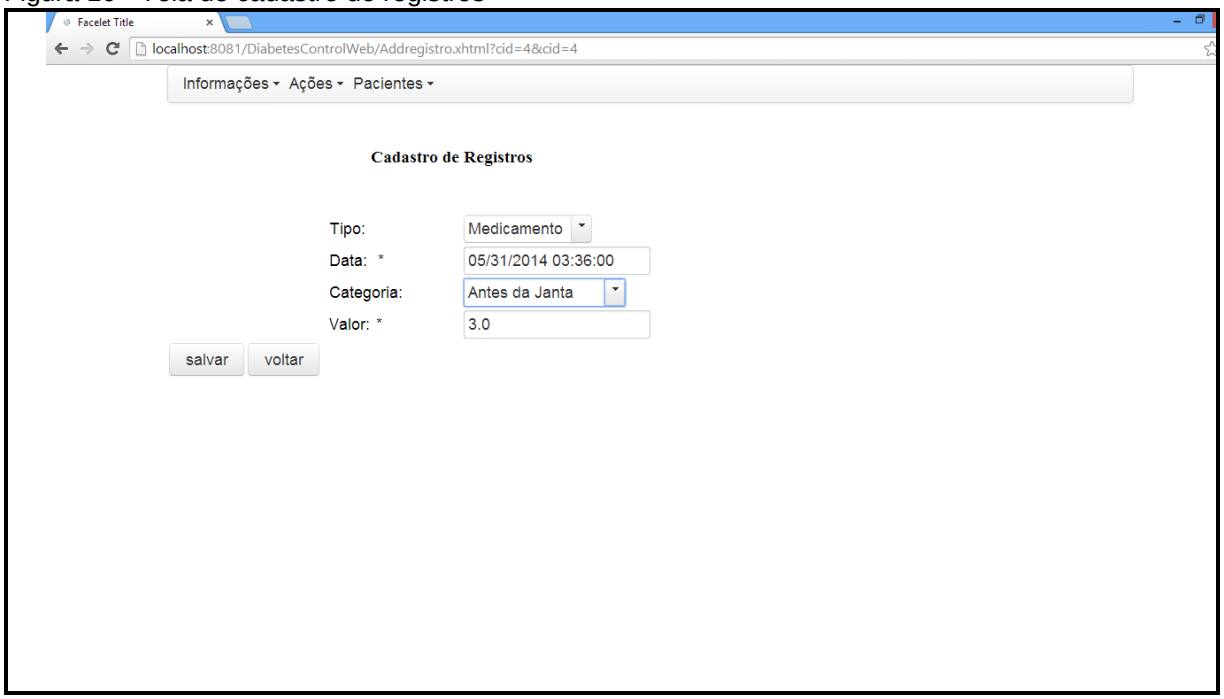
Figura 15 - Tela de listagem de registros



Fonte: Do autor.

Selecionado a opção de editar, o paciente é direcionado para a tela de adicionarRegistro. Esta tela pode ser acessada de duas maneiras a primeira pela tela de listagem de registros sendo assim o paciente seleciona o registro e clica em editar tornando o registro em modo de edição. Pelo menu Adicionar Registro onde toda a informação referente aos índices de diabetes do paciente seria um novo registro. Esta tela é responsável por cadastrar todos os dados de indicadores de diabetes, selecionando o tipo será possível escolher entre os indicadores: glicose, medicamento, peso, pressão, pulso, gordura e Hba1c, informando a data do registro, categoria e valor. Esta tela está entre as mais utilizadas do sistema, como pode ser observado na figura 17.

Figura 16 - Tela de cadastro de registros



Fonte: Do autor.

Com o registro cadastrado o médico pode adicionar a nota para ele. Na tela de listagem de registros o médico seleciona a opção de adicionar nota e então é redirecionada para página de adicionar nota onde contém somente o campo da descrição da nota, quando cadastrada o paciente já pode observar a nota que foi dada para o registro em questão, pela tela de listagem de registros na funcionalidade de ver notas.

Por tratar-se de duas aplicações distintas o *Web Service* foi disponibilizado em um projeto rodando GlassFish na porta 8080 e o *Diabetes Control Web* rodando na porta 8081.

7.5 RESULTADOS OBTIDOS

A grande demanda por aplicações Web associadas a aplicativos móveis, fomenta a construção de aplicações baseados na solução de problemas críticos do cotidiano da população Brasileira. Neste contexto, podemos destacar os problemas relacionados a saúde pública, onde foi identificado um vasto campo de estudo e

desenvolvimento de computação científica aplicada na melhoria da qualidade de vida da população.

Dentre os problemas críticos que acometem a população nos mais diversos níveis sociais, pode-se destacar a diabetes como sendo uma doença crítica e que demanda gerenciamento constante pelos órgãos responsáveis. Este controle é necessário, não somente para a melhoria da qualidade de vida do paciente, mas também para minimizar o impacto no Sistema Único de Saúde, uma vez que, um paciente bem controlado, diminui consideravelmente as chances de necessitar de tratamentos mais caros e complexos, como por exemplo a Hemodiálise.

Apesar do controle realizado nas unidades básicas de saúde, ainda é bastante comum encontrar pacientes desregrados e como um controle desorganizado, fazendo com que o Sistema Único de Saúde pague uma conta bastante elevada. Neste contexto, o presente trabalho apresenta como resultado principal a construção de uma solução web que pode ser acessada pelos profissionais de saúde responsáveis de qualquer lugar em qualquer dispositivo conectado a internet. Além disso, o trabalho também apresenta como resultado, uma metodologia de integração de dados do Aplicativo Diabetes Control, com a aplicação web, proporcionando um gerenciamento efetivo, mobile e totalmente On Line, sendo que, este sistema pode ser implantado na rede pública, e aplicado na melhoria do gerenciamento e coleta de dados referente aos pacientes diabéticos.

A integração dos dados ocorre através da manipulação de arquivos XML, associados a aplicação cliente e servidor, na qual possui características síncronas, caso os dois dispositivos estejam conectados à Internet, e assíncronas, caso o aplicativo Diabetes Control esteja em uma área Off Line. Esta característica, é apresentada como sendo um dos principais pontos do resultado obtido, uma vez que, no Brasil, ainda temos diversas regiões descobertas de internet. Contudo, estas regiões, também poderão ser beneficiadas pela tecnologia desenvolvida.

O sistema Web criado, permite o cadastro e controle dos indicadores de diabetes, assim como uma integração entre Diabetes Control e Diabetes Control Web onde é possível as informações serem cadastradas no Diabetes Control e lidas no Diabtes Control Web.

8 CONCLUSÃO

Com base no atendimento dos objetivos apresentados neste trabalho, que, resumidamente consiste em desenvolver um sistema Web associado a uma tecnologia de integração de dados com o aplicativo, com o *Diabetes Control.* Acredita-se que pode facilitar e aprimorar o gerenciamento de informações médico paciente, proporcionando uma melhor qualidade de vida a população que sofre desta doença, bem como a contribuição com o Sistema único de Saúde, no que tange a diminuição de tratamentos complexos, advindos de maus cuidados com pacientes diabéticos.

Neste contexto, pode-se abordar que a troca de informações entre o médico e o paciente que é alimentada pelo responsável, podendo ser o próprio paciente, é feito pelo aplicativo *Diabetes Control*, na qual, o médico pode acompanhar remotamente os dados e tomar uma decisão com relação aso níveis de glicemia apresentados, podendo evitar a complicação dos quadros clínicos em diversas situações. O sistema desenvolvido, apresenta integração via *Web Service*, o qual foi adaptado utilizando o padrão JAX-WS. A aplicação web foi desenvolvida utilizando tecnologias como Java, JSF, JPA e *Web Services*.

Com a Utilização da solução desenvolvida é possível que os médicos acompanhem em tempo real a evolução dos seus pacientes referentes a diabetes e gerenciem o tratamento de acordo com os indicadores apresentados.

Como sugestões para trabalhos futuros, considerando o projeto de pesquisa realizado, podem ser listados os seguintes tópicos:

a) Implementar alguma biblioteca de segurança como Apache Shiro, ou JAAS.
b) Implementar um leitor de diabetes automático;
c) Acoplar o *Web Service* dentro do Diabetes Control Web;
d) Implantar o sistema em uma unidade Piloto para validar a solução proposta e evoluí-la.

REFERÊNCIAS

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

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: 04 out. 2013.

ALVES, Andréa Cristina Oliveira. **Proposta de um Modelo para a Implementação de um Ambiente Inteligente para o Ensino de Informática Médica**. 2002. 114p. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis.

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 Catarinense, Criciúma, 2011.

BARROS, Thiago, **O que é smartphone e para que serve**. 2012. Disponivel em: <www.techtudo.com.br/artigos/noticia/2011/12/o-que-e-smartphone-e-para-que-serve.html>. Acesso em: 04 nov. 2013.

BURKE, Bill. RESTful Java with JAX-RS. Sebastopol: O'Reilly Media, 2010. 312 p.

CARVALHO, José Alécio. **MISW**: Uma metodologia de integração com serviços web. 2006.63 f. Monografia – Curso de Informática, Universidade de Fortaleza, Fortaleza, BR – CE.Disponível em: <http://sites.google.com/site/alecio/MISW.pdf>. Acesso em: 25 nov. 2013.

ÇALISKAN, Mert; VARAKSIN, Oleg, **PrimeFaces CookBook**, 1. Ed. Birmingham, 2013.

CANALTECH. Disponível em: <http://canaltech.com.br/noticia/samsung/Conheca-um-pouco-da-historia-dos-smartphones-e-tablets-Samsung/>. Acesso em: 04 nov. 2013.

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

DUCKETT, Jon. Beginning HTML, XHTML, CSS, and JavaScript. Indianapolis: Wiley Publishing, 2010. 866 p.

ECKEL, Bruce. Thinking in Java. 4. ed. Stoughton: Prentice Hall, 2006.

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: 04 out. 2013.

GAMBA, Monica Antar et al. **Amputação de extremidades inferiores por diabetes mellitus:** estudo caso-controle. Revista de saúde pública. v. 38, n. 3, p. 99-404, 2004.

GEARY, David; HORSTMANN, Cay. **Core JavaServer faces**. 3ed. Redwood Shores: Prentice Hall, 2010.

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: 24 out. 2013.

GUPTA, Arun. Java EE 7 Essentials. Sebastopol: O'Reilly Media, 2013. 362 p.

GOOGLE. **Android Developers**. Disponível em: <http://developer.android.com/>. Acesso em: 10 abr. 2013.

HEFFELFINGER, Daivid, Java EE 7 with GlassFish 4 Application Servicer, Terceira edição, 2014.

INTERNATIONAL DIABETES FEDERATION. Bruxelas: Diabetes Atlas 2013 . em: <http://www.idf.org/diabetesatlas>. Acesso em : 31/10/2013.

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.

Lyytinen, K. e Yoo, Y (2012). **Issues and Challenges in Ubiquitous Computing. Communications of the ACM.** vol.45, no. 12, 2002.

MARTÍN,Javier. Aplicativos para celulares permitem exames em casa. Disponível em: <http://noticias.uol.com.br/saude/ultimas-noticias/redacao/2013/02/12/aplicativos-para-celular-permitem-fazer-exames-em-casa.html>. Acesso em :29 abr. 2013.

Mateus, G.R e Loureiro, A.F (1998). **Introdução a Computação Móvel.** Segunda edição, 1998, Minas Gerais.

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

NEUWALD, Leonardo Alves. **Google Android: Um novo paradigma de desenvolvimento mobile aplicado ao gerenciamento de informações médicas referente ao controle do diabetes**. 2012. 79 f. Monografia (Graduação) – Universidade do Extremo Sul Catarinense, Criciúma, 2012.

NIEMI, **V. LTE Security Architeture**. Nokia Research Center – Eurolab, Lausanne, Switzerland, 2010.

NISKA, Christoffer. **Extending Bootstrap**: Packt Publishing,England 2014.

O'CONNOR, Fred. **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>. Acessado em: 28 abr 2013.

OLIVEIRA, Kelly Cristina Silva de; ZANETTI, Maria Lucia. **Conheciemento e atitude de usuários com Diabetes Mellitus em um serviço de atenção básica a saúde**. Rev Esc Enferm USP. v. 45, n. 4, p. 862-8, 2011.

OPEN HANDSET ALLIANCE. **Overview**. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 09 maio 2013.

PAULON, Renan. **Web Services in JAVA**. Disponível em: <http://www.imasters.com.br/artigo/1863/>. Acesso em 11 nov. 2013.

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

PERES, L.A.B. et al. Aumento na Prevalência de Diabete Mellito Como Causa de Insuficiência Renal Crônica Dialítica – Análise de 20 Anos na Região Oeste do Paraná. **Arquivos Brasileiros de Endocrinologia e Metabologia**, 51(1): 111-115, 2007.

POTTS,Stephen; KOPACK,Mike. **Aprenda Web Services em 24 Horas**.Campus, Rio de Janeiro, RJ. 2003

POTHU, Sudhakar. **A Comparative Analysis of Object-relational mappings for Java**. 2008. 108 f. Dissertação (Mestrado) - University Of Applied Sciences, Braunschweig, 2008

RABELLO, Ramon Ribeiro (2009). **Android: Um Novo paradigma de desenvolvimento móvel**. Revista WebMobile Magazine, Edição18, 2009. Disponivel em : <http://www.devmedia.com.br/artigo-webmobile-18-android-um-novo-paradigma-de-desenvolvimento-movel/9350> . Acesso em: 10 out. 2013

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

SABBATINI. Renato M. E. **O Computador no Processamento de Sinais Biológicos**. InforMédica, São Paulo, 1995.

SILVA FILHO, Jadir Cirilo da. **Segurança em Web Services**. Trabalho de Conclusão de Curso (Especialização), Universidade Federal do Rio Grande do Sul. Porto Alegre, RS 2004.

SILVEIRA, Rodrigo. Learn HTML5 by Creating Fun Games. Birmingham: Packt Publishing, 2013. 374 p

SOCIEDADE BRASILEIRA DE DIABETES. **Tratamento e acompanhamento do Diabetes Mellitus:** diretrizes da sociedade brasileira de diabetes mellitus: Diagraph. Rio de Janeiro, 2007.

TELECO, Inteligência em Telecomunicações. **SMARTPHONE**. Disponível em: <www.teleco.com.br/smartphone.asp>. Acesso em : 04 nov. 2013.

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: 25 abr. 2013.

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: 20 mar. 2013.

APÊNDICE(S)

APÊNDICE A – Artigo Científico

Desenvolvimento de uma solução Web integrada com o aplicativo Diabetes Control para controle e gerenciamento de diabetes

Léo Moraes de Carvalho, Gustavo Bisognin

Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 – Caixa Postal 3167 – CEP 88806-000 – Santa Catarina – SC – Brasil

leomoraes@engeplus.com.br, gbisog@gmail.com

Abstract. *Diabetes is considered a critical illness that attacks the population in various social classes. The use of automatic management systems to assist in controlling Diabetes can provide a better quality of life for patients. In health it is recommended that those responsible always have at hand the evolutionary information about the disease and the patient. so it was made prototyping the necessary screens for the prototype, using the Model-View-Controller (MVC) architecture was performed prototype development along with the development of the Web Service to the integration of web application diabetes Control and diabetes Control Web system*

Resumo. *A diabetes é considerada uma doença crítica que ataca a população nas mais diversas classes sociais. A utilização de sistemas automáticos de gerenciamento que auxiliem no controle da Diabetes pode proporcionar uma melhor qualidade de vida aos portadores. Na área da saúde é recomendado que os responsáveis sempre tenham em mãos as informações evolutivas sobre a doença e o paciente. Assim foi feita a prototipação das telas necessárias para o protótipo, utilizando-se da arquitetura* Model View Controller *(MVC) foi realizado o desenvolvimento do protótipo juntamente com o desenvolvimento do* Web Service *para a integração entre o aplicativo Diabetes Control o sistema* web *Diabetes Control Web.*

1. Introdução

Atualmente o Diabetes Mellitus (DM) é considerado um dos mais importantes problemas de saúde pública, tanto pelo número de pessoas afetadas quanto pelos custos envolvidos no tratamento de suas complicações (PÉRES et al, 2007).

Segundo a Sociedade Brasileira de Diabetes (2007) com a informação gerada através dos sensores de glicose pode-se com a informação em tempo real e de forma continua, tomar decisões de aplicar insulina ou ingerir carboidratos para evitar hipoglicemia. Os quadros de hipoglicemia, por vezes intensos e muito sérios pode receber orientação imediata resultando muitas vezes em vidas salvas. Já que sabe-se que esta é uma das principais causas de morte de arritmias em diabéticos.

O trabalho consiste em desenvolver um sistema *Web* integrado com o aplicativo Diabetes Control que permita a integração entre paciente e médico para o melhor controle e gerenciamento da diabetes.

2. Aplicações WEB

As aplicações *web* funcionam de um jeito onde toda aplicação está localizada em um *web* server, que nada mais é que um computador ligado à Internet esperando pedidos de um navegador, que tem como função fazer a requisição ao servidor que retornar uma página *HyperText Markup Language* (HTML) (DUCKETT, 2010).

Cada navegador sabe a qual servidores eles estão fazendo requisições a partir da URL, que é o endereço do site onde se encontra o servidor que irá retornar a página HTML que foi requisitada pelo navegador, por exemplo quando um navegador solicita uma página HTML para a URL www.google.com.br será retornado a página padrão de busca da Google, os navegadores mais conhecidos são: Google Chrome, Mozilla FireFox e Internet Explore (DUCKETT, 2010).

As aplicações *web* são separadas em duas partes, a parte responsável pela exibição dos dados para o cliente que é chamada de *Client Side* e a parte responsável pelo processamento de dados no servidor que é chamada de *Server Side*.

2.1 Client Side

A maioria das linguagens de programação utilizadas no lado do cliente são interpretadas pelos navegadores, a principal vantagem é a velocidade de execução já que são executadas no cliente sem a necessidade de ser feito requisições para o servidor, as principais linguagens utilizadas no lado do cliente são HTML, JavaScipt(JS) e Cascading Style Sheets (CSS) (DUCKETT, 2010).

2.2 Server Side

A maioria das linguagens de programação utilizadas no lado do cliente são interpretadas pelos navegadores, a principal vantagem é a velocidade de execução já que são executadas no cliente sem a necessidade de ser feito requisições para o servidor, as principais linguagens utilizadas no lado do cliente são HTML, JavaScipt(JS) e Cascading Style Sheets (CSS) (DUCKETT, 2010).

3. Diabetes Control Web – Interface de Gerenciamento

O trabalho demonstra as etapas do desenvolvimento de um sistema para o controle e gerenciamento de diabetes, assim como as tecnologias utilizadas para o desenvolvimento.

O sistema em plataforma web que realiza o controle sobre os indicadores da diabetes dos pacientes. Junto com esse controle o sistema também prevê a integração com o aplicativo *Diabetes Control,* que por sua vez foi desenvolvido em versão *mobile*, e alterado para atender as necessidades do sistema. Sendo assim o sistema recebe dois tipos de informações, aquelas geradas pelo usuário diretamente pela interface, e aquelas que são cadastradas previamente no aplicativo mobile sendo sincronizada com o aplicativo web via *Web Service*.

Tanto no aplicativo *Diabetes Control* quanto no sistema *Diabetes Control Web* os dados cadastrados de pacientes e médicos se comunicam de tal forma que os médicos possam avaliar em tempo real os registros de indicadores de diabetes e tomarem as decisões necessárias em cada caso. Diversas ferramentas e tecnologias foram utilizadas para alcançar

os objetivos propostos, como por exemplo: Java, Android, JPA, JSF, entre outras citadas posteriormente na metodologia.

3.1 Implementação

O início do desenvolvimento deu-se a partir do sistema *Web,* sem a necessidade de integração do aplicativo *Diabetes Control* pelo *Web Service*. O sistema é capaz de gerenciar os dados referentes aos índices da diabetes do paciente. Para o desenvolvimento *Web* foi utilizado o padrão MVC, onde os arquivos *xhtml*, representam a camada de *view,* as entidades representam a camada de *model,* e os controllers representam a camada de *controller.*

Foram criadas as entidades Paciente, Médico, Registro, NotaRegistroMedico, que representam respectivamente, as informações dos pacientes, as informações do médico, os indicativos de diabetes e as notas que os médicos cadastram para os registros dos pacientes. Com base nas entidades foram criados dois Data Acces Object (DAO), PacienteDAO, MedicoDAO que tem como função a conexão da aplicação com o banco de dados, sendo o PacienteDAO responsável pelas entidades Paciente e Registros e o MedicoDAO responsável pelas entidades Medico e NotaRegistroMedico.

Para o relacionamento das entidades foi utilizada as anotações do *JPA @ManyToMany* para o relacionamento de muitos pra muitos da tabela de Paciente para a tabela de Medico, foi utilizado o *@ManyToOne* da tabela de Registro para a tabela de Paciente e @oneToOne da tabela de Registro para a tabela de NotaRegistroMedico, *@Id* identifica a chave primária da entidade, @GeneratedValue represeta o modo que será automaticamente a chave primaria, utilizando o generator=”SEQ_REGISTRO” ele utiliza a tabela *SEQ_REGISTRO* para a criação da chave primaria e utilizando o *GenerationType.SEQUENCE* é utilizado o método de sequência que consiste em pegar o valor que será gravado na tabela *SEQ_REGISTRO* mais um, ou seja, cada vez que a entidade Registro for salva no banco o JPA irá adicionar mais um na tabela *SEQ_REGISTO*, a figura 1 mostra as anotações na entidade Registro.

```
22        @Id
23        @GeneratedValue(generator = "SEQ_REGISTRO", strategy = GenerationType.SEQUENCE)
24        private Integer Id;
25        private String tipo;
26        private String categoria;
27        private Float valor;
28        @Temporal(javax.persistence.TemporalType.TIMESTAMP)
29        private Date dataHora;
30        private String unidade;
31        private String valorpressao;
32        @ManyToOne
33        private Paciente paciente;
34        @OneToOne(mappedBy = "registro")
35        private NotaRegistroMedico nota;
```

Figura 1 – Anotações do JPA na entidade Registro

Os controladores do *Diabetes Control Web* são: PacienteController e MedicoController. O PacienteController tem a responsabilidades de fazer a ligação entre as entidades Paciente e Registro com as telas de cadastros e listagens. Já o MedicoController faz a ligação das entidades Medico e NotaRegistroMedico com os cadastros e listagens. O controle de sessão do usuário é feito através do controller userSession que é responsável pelo controle de sessão do sistema, ele tem as informações de login e do usuário logado no sistema.

Para a parte gráfica do sistema utilizou-se a biblioteca *Primefaces* na versão *4.0*. Trata-se de uma biblioteca que é utilizada em conjunto com JSF para a evolução dos componentes que vão muito além das especificações, utilizou-se essa biblioteca por causa de sua facilidade de uso, por ser uma biblioteca leve, ter uma simples instalação e pela sua grande comunidade que proporciona um grande suporte para os problemas encontrados, todos os componentes utilizados no sistema são componentes da biblioteca.

Para o controle de *URL* foi implementado o LoginListener que verifica em cada página acessada se já existe um usuário logado caso não exista o usuário será direcionado para a página de login para poder acessar o sistema.

A integração do *Diabetes Control* com o *Diabetes Control Web*, deu-se através do *Web Service*. Este está em um projeto a parte e só foi modificado para poder ser integrado com o sistema desenvolvido. Foi alterado o provedor JPA de *EclipseLink* para *Hibernate*, adicionado as anotações para o controle no banco, alterado os métodos de busca e disponibilização do *entityManager* para o controle de transações.

A figura 2 apresenta a tela onde os médicos podem cadastrar novos pacientes para serem gerenciados pelo sistema, Nela é informado os campos: nome do paciente, data de nascimento, email, sexo, código do paciente e senha. Esta tela é acessada pelo menu Cadastrar paciente.

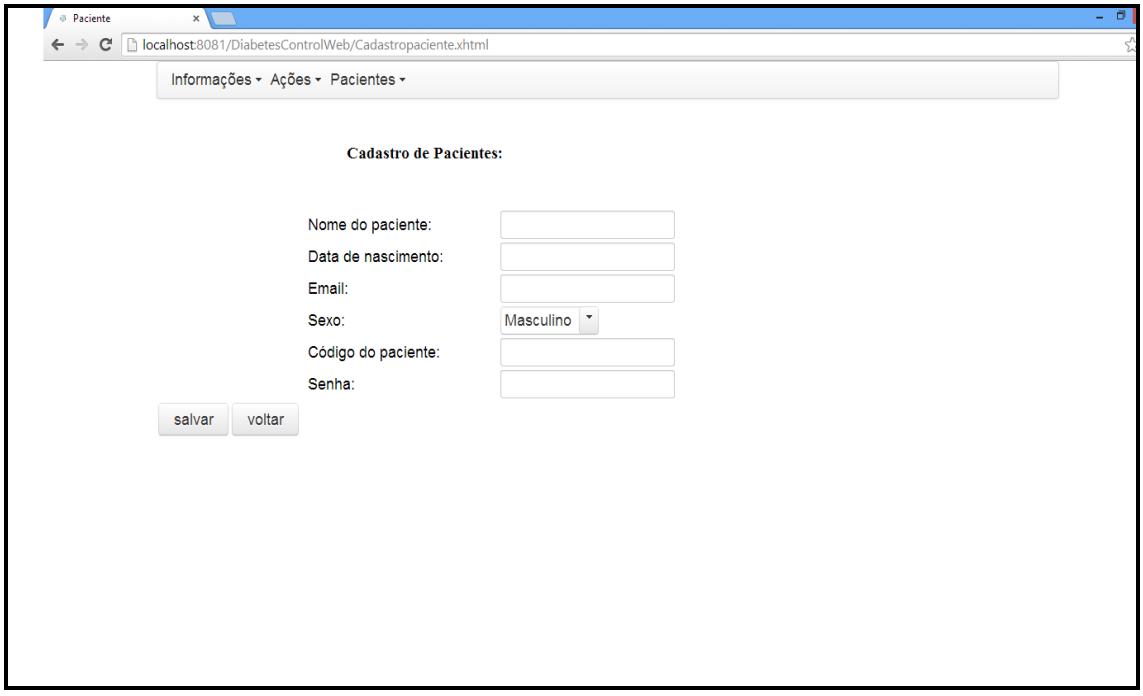


Figura 2 – Cadastro de pacientes

A tela de adicionar paciente serve para o médico informar os dados do paciente que ele deseja gerenciar. Somente a partir deste momento o médico consegue visualizar no sistema os dados do paciente. A figura 3 mostra a tela onde o médico adiciona os pacientes.

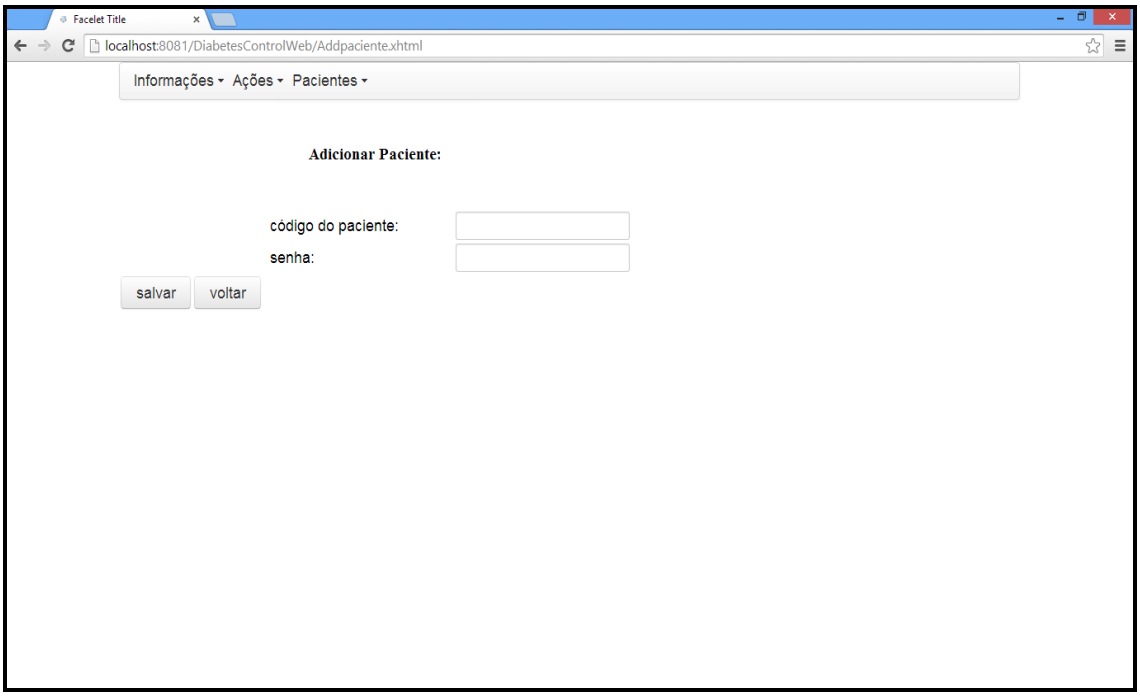


Figura 3 - Tela de adicionar paciente.

Com o paciente adicionado ele então começa a ser gerenciado, agora o paciente está sendo listado na tela Pacientes, onde se encontram duas funcionalidades a de exportar, que exporta o conteúdo da tabela em arquivos do tipo Portabel Document Format (PDF), Excel(XLS), Comma-separeted Values (CSV) e XML, para assim os dados possam ser integrados com outros aplicativos, e a função de visualizar, que possibilita o médico visualizar ao paciente selecionado. A figura 4 representa a tela de listagem de pacientes.

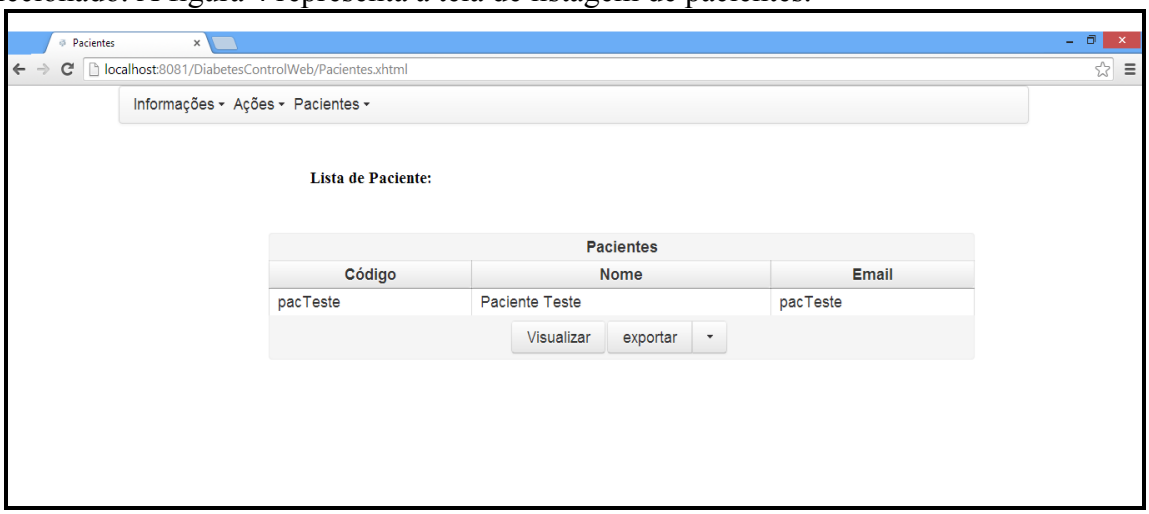


Figura 4 - Lista de pacientes

Visualizando o paciente selecionado o médico abre a tela de visualização de pacientes, nesta tela é demonstrado todos os dados cadastrais do paciente: nome, data, email, sexo e código. Logo abaixo tem-se o botão de registros que redireciona para página de listagem de índices de diabetes do paciente selecionado, esta página pode ser acessada tanto por médico quanto por pacientes, por médicos a partir do fluxo descrito acima, e por pacientes pelo menu Lista de

registros, nela contém as informações cadastradas na tela de cadastro de registros, estes registros são: tipo, valor, data e categoria. A listagem possui três funcionalidades a de exportar que exporta o conteúdo da tabela em arquivos do tipo PDF, XLS, CSV e XML, a de ver nota que possibilita ser visualizado as notas que o médico cadastrou para aquele registro do paciente selecionado em especifico, estas notas são visualizadas através de um *Dialog box* que é aberto quando o usuário seleciona a funcionalidade de ver nota. E por último a edição que leva o paciente para a tela gerenciamento dos indicadores de diabetes. Caso o usuário seja um médico é adicionado a funcionalidade de adicionar nota no lugar da de edição de registros, sendo redirecionado para a tela de adicionar notas. A figura 5 representa a tela de listagem de registros.

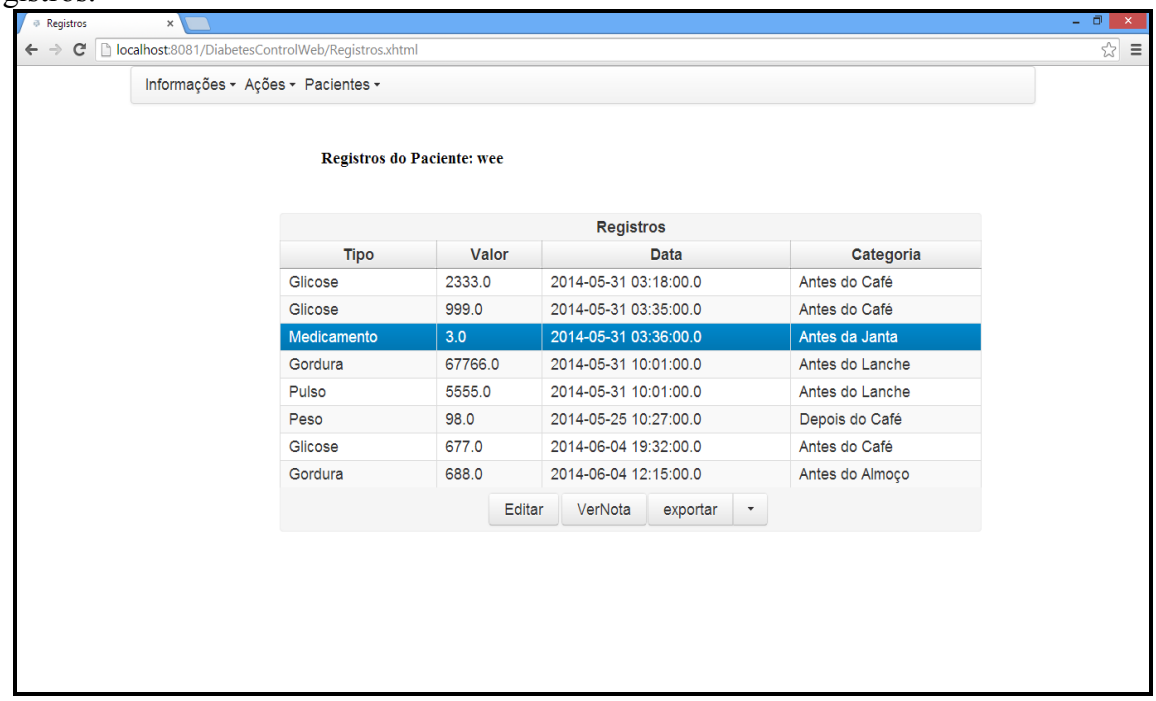


Figura 5 - Tela de listagem de registros

Selecionado a opção de editar, o paciente é direcionado para a tela de adicionarRegistro. Esta tela pode ser acessada de duas maneiras a primeira pela tela de listagem de registros sendo assim o paciente seleciona o registro e clica em editar tornando o registro em modo de edição. Pelo menu Adicionar Registro onde toda a informação referente aos índices de diabetes do paciente seria um novo registro. Esta tela é responsável por cadastrar todos os dados de indicadores de diabetes, selecionando o tipo será possível escolher entre os indicadores: glicose, medicamento, peso, pressão, pulso, gordura e Hba1c, informando a data do registro, categoria e valor. Esta tela está entre as mais utilizadas do sistema, como pode ser observado na figura 6.

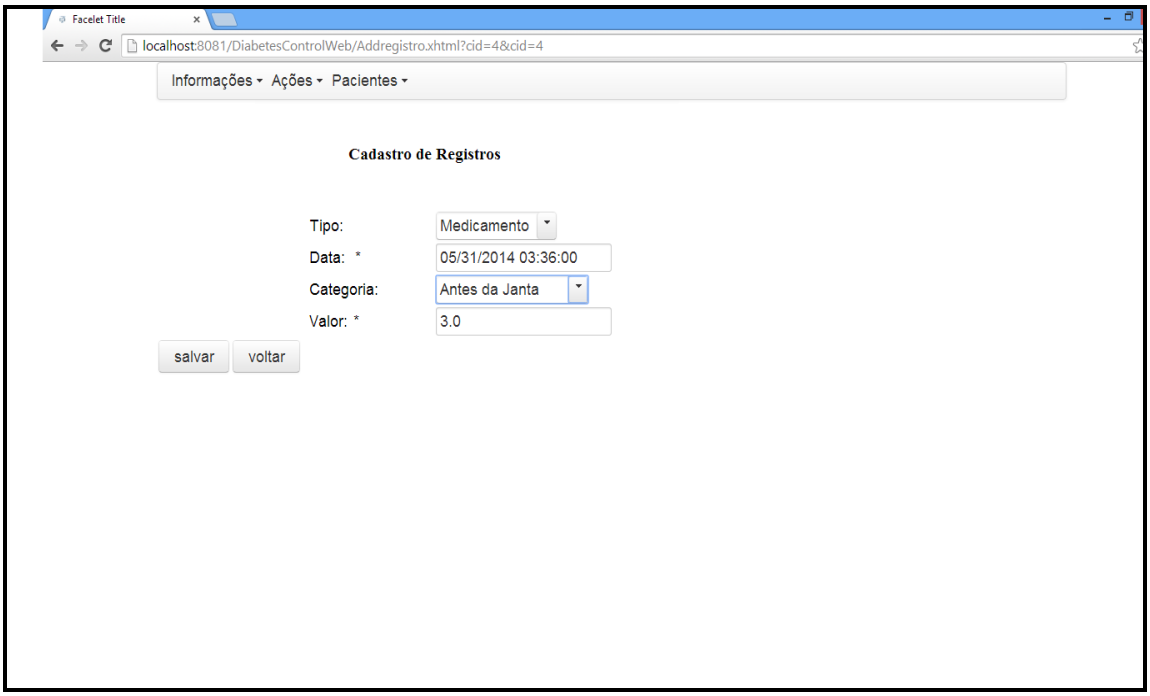


Figura 6 - Tela de cadastro de registros

Com o registro cadastrado o médico pode adicionar a nota para ele. Na tela de listagem de registros o médico seleciona a opção de adicionar nota e então é redirecionada para página de adicionar nota onde contém somente o campo da descrição da nota, quando cadastrada o paciente já pode observar a nota que foi dada para o registro em questão, pela tela de listagem de registros na funcionalidade de ver notas.

Por tratar-se de duas aplicações distintas o *Web Service* foi disponibilizado em um projeto rodando GlassFish na porta 8080 e o *Diabetes Control Web* rodando na porta 8081.

4. Conclusão

Com base no atendimento dos objetivos apresentados neste trabalho, que, resumidamente consiste em desenvolver um sistema Web associado a uma tecnologia de integração de dados com o aplicativo móvel, com o *Diabetes Control.* Acredita-se que pode facilitar e aprimorar o gerenciamento de informações médico paciente, proporcionando uma melhor qualidade de vida a população que sofre desta doença, bem como a contribuição com o Sistema único de Saúde, no que tange a diminuição de tratamentos complexos, advindos de maus cuidados com pacientes diabéticos.

Neste contexto, pode-se abordar que a troca de informações entre o médico e o paciente que é alimentada pelo responsável, podendo ser o próprio paciente, é feito pelo aplicativo *Diabetes Control*, na qual, o médico pode acompanhar remotamente os dados e tomar uma decisão com relação aso níveis de glicemia apresentados, podendo evitar a complicação dos quadros clínicos em diversas situações. O sistema desenvolvido, apresenta integração via *Web Service*, o qual foi adaptado utilizando o padrão JAX-WS. A aplicação web foi desenvolvida utilizando tecnologias como Java, JSF, JPA e *Web Services*.

Com a Utilização da solução desenvolvida é possível que os médicos acompanhem em tempo real a evolução dos seus pacientes referentes a diabetes e gerenciem o tratamento de acordo com os indicadores apresentados.

5. References

Peres, L.A.B. et al. Aumento na Prevalência de Diabete Mellito Como Causa de Insuficiência Renal Crônica Dialítica – Análise de 20 Anos na Região Oeste do Paraná. **Arquivos Brasileiros de Endocrinologia e Metabologia**, 51(1): 111-115, 2007.

DUCKETT, Jon. Beginning HTML, XHTML, CSS, and JavaScript. Indianapolis: Wiley Publishing, 2010. 866 p.

BURKE, Bill. RESTful Java with JAX-RS. Sebastopol: O'Reilly Media, 2010. 312 p.