UNIVERSIDADE FEDERAL DE SANTA CATARINA

SISTEMA DO PROCESSO DE ENFERMAGEM INFORMATIZADO NO AUXÍLIO À PACIENTES POLITRAUMATIZADOS INTERNADOS EM UTI - UMA ABORDAGEM UTILIZANDO ARQUITETURA WEB SERVICE

Aline de Matos Fonseca de Souza

Marcelo Cardoso de Souza

Orientador: Prof. Dr. Fernando Augusto da Silva Cruz

Florianópolis - SC

2013/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

SISTEMA DO PROCESSO DE ENFERMAGEM INFORMATIZADO NO AUXÍLIO À PACIENTES POLITRAUMATIZADOS INTERNADOS EM UTI - UMA ABORDAGEM UTILIZANDO ARQUITETURA WEB SERVICE

Aline de Matos Fonseca de Souza

Marcelo Cardoso de Souza

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis – SC

2013/2

Aline de Matos Fonseca de Souza

Marcelo Cardoso de Souza

SISTEMA DO PROCESSO DE ENFERMAGEM INFORMATIZADO NO AUXÍLIO À PACIENTES POLITRAUMATIZADOS INTERNADOS EM UTI - UMA ABORDAGEM UTILIZANDO ARQUITETURA WEB SERVICE

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Fernando Augusto da Silva Cruz

Banca examinadora

Prof. Dr. João Bosco Mangueira Sobral

Profª. Dr ª. Grace Teresinha Marcon Dal Sasso

RESUMO

Com estudos relacionados à informatização dos processos de enfermagem, surgiu o SIPE – Sistema do Processo de Enfermagem ao paciente politraumatizado. Este trabalho propõe reestruturar o SIPE, utilizando a arquitetura de Web Service, e adotando padrões de projeto no seu desenvolvimento. Assim o SIPE conseguirá se integrar com outras aplicações de diferentes plataformas. Este trabalho contribuirá para a disponibilização do SIPE para dispositivos móveis, utilizando a tecnologia Android. O servidor foi desenvolvido na linguagem Java, o cliente Web em linguagem PHP, o cliente móvel em Java para Android e o SGBD – Sistema de Gerenciamento de Banco de Dados em PostgreSQL. Este trabalho alcançou seus objetivos, ao implementar a reestruturação proposta, sendo possível comprovar os benefícios trazidos por ela ao SIPE. Através dos padrões de projeto, que facilitam a manutenção do sistema, bem como a arquitetura de Web Service, que permite a integração fácil e segura com outras aplicações.

Palavras-Chave: SIPE – Sistema do Processo de Enfermagem, padrões de projeto, Web Service, Android.

ABSTRACT

Along with studies related to the informatization of nursing processes, came the SIPE – Sistema do Processo de Enfermagem ao paciente politraumatizado (Nursing Process System to polytraumatized pacients). This paper proposes the restructure of the SIPE utilizing the Web Service architecture while adopting design patterns in its development. Therefore, the SIPE will be able to integrate itself to other applications of different platforms. This paper will contribute to SIPE's availability to mobile devices, utilizing Android's technology. The server was developed in Java language, the Web client in PHP language, the mobile client in Java for Android and the SGBD – Sistema de Gerenciamento de Banco de Dados (Data Base Management System) in PostgreSQL. This paper has achieved its objectives as to implementing the proposed restructuration, being possible to verify its benefits on the SIPE. Through design patterns, that facilitate the system's maintenance, altogether with the Web Service architecture, that allows an easy and safe integration to other applications.

Keywords: SIPE – Sistema do Processo de Enfermagem, Nursing Process System, design patterns, Web Service, Android.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

API - Application Programming Interface (Interface de Programação de Aplicativos)

ADT - Android Developer Tools

AES - Advanced Encryption Standard

AOP - Aspect Oriented Programming

B2B – Business to Business

C - Linguagem de Programação Compilada chamada C

CGI - Common Gateway Interface

CIPE - Classificação Internacional para a Prática em Enfermagem

CORBA - Common Object Request Broker Architecture

CSS - Cascading Style Sheets

CXF - Celtix e XFire.

DAO - Data Access Object

DES - Data Encryption Standard

DTO - Data Transfer Object

HTML - HyperText Markup Language (Linguagem de Marcação de Hipertexto)

HTTP - HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto)

HTTPS - HyperText Transfer Protocol Secure

IDE - Integrated Development Environment

IDEA - International Data Encryption Algorithm

IIS - Internet Information Service

J2EE - Java 2 Platform, Enterprise Edition

JAX-RS - Java API for XML RESTful

JAX-WS - Java API for XML Web Services (API Java para Serviços Web XML)

JMS - Java Message Service

JSP – JavaServer Pages

JVM - Java Virtual Machine (Máquina Virtual Java)

MVC - Model, View e Controller (modelo, visão e controle)

PDA - Personal Digital Assitant

PHP - Personal Home Pages; **P**HP: **H**ypertext **P**reprocessor

PHP/FI - Personal Home Pages/Forms Interpreter

POJO - Plain Old Java Objects

REST - Representational State Transfer (Transferência do Estado Representativo)

RESTful – Assim são chamados os sistemas que seguem os princípios REST

SAML - Security Assertion Markup Language

SDK – Software Developement Kit

SGBD - Sistema Gerenciador de Banco de Dados

SIPE - Sistema do Processo de Enfermagem Informatizado ao Paciente Politraumatizado

SOAP - Simple Object Access Protocol (Protocolo Simples de Acesso a Objetos)

SQL - Structured Query Language (Linguagem de Consulta Estruturada)

SSL - Secure Socket Layer

UDDI – Universal Description, Discovery and Integration

UFSC - Universidade Federal de Santa Catarina

URL - Uniform Resource Locators (Localizador-Padrão de Recursos)

UTI - Unidades de Terapia Intensiva

W3C – World Wide Web Consortium

WSDL - Web Services Description Language

WSS4J – Web Services Security for Java

XML - eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A informática vem se incorporando em nosso cotidiano e com suas ferramentas vem se tornando cada vez mais presente nas atividades profissionais e educacionais. São inúmeros os benefícios gerados pelas tecnologias: agilidade e eficiência na execução de tarefas são dois exemplos que servem até mesmo como justificativa para a utilização desta ciência.

Na área de saúde também não é diferente. A informática se faz presente em vários segmentos, auxiliando desde a administração até o operacional, onde estão os profissionais da medicina, enfermagem. Diversos sistemas informatizados têm sido desenvolvidos para esta área, principalmente com a função de melhorar o atendimento aos usuários ou pacientes.

Especificamente na área da enfermagem são grandes os esforços para se chegar a um sistema que esteja de acordo com as necessidades dos profissionais e, de preferência, que siga uma padronização necessária para que haja maior clareza na interpretação dos diagnósticos, diminuindo assim os riscos de uma intervenção errada.

As Unidades de Terapia Intensiva – UTI caracterizam-se por ser uma zona crítica, onde erros não podem ser admitidos, pois os riscos são iminentes. Neste contexto os sistemas devem contribuir para maior precisão nos diagnósticos e consequentemente nas intervenções e decisões a serem tomadas.

Por iniciativa de pesquisadores do curso de Enfermagem da UFSC, surgiram estudos sobre a sistematização e informatização da assistência Enfermagem na UTI (DAL SASSO 1999). Estes deram início ao desenvolvimento e avaliação do SIPE –

Sistema do Processo de Enfermagem Informatizado ao Paciente Politraumatizado (ANTUNES 2006). Desde os primeiros estudos foi levada em consideração a CIPE® – Classificação Internacional para a Prática em Enfermagem na versão Beta 2. Em 2006 os estudos concluíram uma proposta da estrutura do sistema na forma WEB contemplando o Cuidado de Enfermagem em Terapia Intensiva utilizando a CIPE® versão 1.0 (ANTUNES 2006).

1.1 Contextualização

O SIPE – Sistema do Processo de Enfermagem Informatizado ao Paciente Politraumatizado teve origem nos estudos de Grace Dal Sasso (DAL SASSO 1999) (DAL SASSO 2001) e nos estudos de Zabotti e Souza (ZABOTTI e SOUZA 2002), que desenvolveram a primeira estrutura do processo de enfermagem informatizado ao paciente com alterações respiratórias e em seguida com alterações cardiovasculares em terapia intensiva. Para esta versão foi utilizando a CIPE® – Classificação Internacional para a Prática em Enfermagem, versão Beta 1 e desenvolvida na linguagem Delphi (DAL SASSO 1999).

Em 2006 surgiu o Sistema Informatizado ao Paciente Politraumatizado de Terapia Intensiva via Web (ANTUNES 2006). Desenvolvido na linguagem PHP e banco de dados relacional, baseado na CIPE® versão 1.0, sendo tema de dissertação de mestrado em Enfermagem, do Programa de Pós-Graduação da Universidade Federal de Santa Catarina – UFSC (ANTUNES 2006).

Em outro estudo também como dissertação de mestrado em Enfermagem, do Programa de Pós-Graduação da UFSC (BARRA 2008), foi realizada a

disponibilização do SIPE através de dispositivos móveis como PDA – Personal Digital Assitant, que possuem um navegador Web, com o intuito de prover a entrada dos dados junto ao leito do paciente (BARRA 2008).

Atualmente o SIPE apresenta cinco funcionalidades: prontuário, avaliações, diagnósticos, intervenções e administração. Para os nove sistemas do corpo humano: respiratório, cardiovascular, neurológico, musculoesquelético, gastrointestinal, renal, tegumentar, reprodutivo e biopsicossocial. Além de armazenar informações de exames laboratoriais, sinais vitais, balanço hidroeletrolítico, fluidoterapia e intervenções específicas (ANTUNES 2006).

1.2 Justificativas

Outros sistemas foram surgindo ao longo do tempo através de estudos realizados pelo grupo de pesquisadores do curso de Enfermagem da UFSC, muitos são destinados a dar suporte ao profissional da área da saúde, alguns possuem uma relação entre si, porém muitas vezes não são capazes de se integrar para formar um sistema mais completo.

Sendo assim a proposta deste trabalho é reestruturar o SIPE, para que ele se adapte e permita a sua integração com outras aplicações de forma fácil e segura. Para isso será aplicada a arquitetura de Web Service, que permite essa integração com diferentes aplicações em diferentes ambientes.

Sabendo que o SIPE ainda não foi utilizado no dia-a-dia de uma UTI, salvo nos testes realizados pelos pesquisadores, e acreditando no potencial do sistema é imprescindível que se verifique a sua qualidade, para que possa ser utilizado em

definitivo nas UTI's. Sendo efetivamente usado, com todo o seu potencial, para dar suporte aos profissionais de Enfermagem.

Neste trabalho não serão avaliadas as regras de negócio, nem a gestão do sistema, pois ele ainda está em estudo e sendo amplamente discutido pelos pesquisadores. Sendo assim, ainda podem surgir mudanças e alterações em tempo.

1.3 Objetivos gerais

Este trabalho tem como objetivos reestruturar o SIPE, organizando a estrutura do seu código fonte e permitir que o sistema se integre a outras aplicações de modo simples e seguro.

Verificar a viabilidade para as eventuais manutenções do sistema, principalmente as evolutivas, visando à complexidade dos arquivos de código fonte.

1.4 Objetivos específicos

Para alcançar tais metas, serão adotados os seguintes objetivos específicos:

- Aplicar padrões de projeto ao SIPE, a fim de obter uma melhor estrutura do sistema e possibilitar sua manutenção;
- Possibilitar a integração do SIPE com outros sistemas, de forma simples, Disponibilizando suas funcionalidades através de Web Service, garantindo a segurança dos dados trafegados pelo sistema;
- Viabilizar a utilização do sistema em diversos ambientes computacionais.

2 SISTEMA SIPE

No presente capítulo estão descritas as principais funcionalidades do sistema, sendo que estas estão divididas em cinco grupos: prontuários, avaliações, diagnósticos, intervenções e administração.

De forma geral o sistema propicia o cadastro dos dados coletados pelo usuário em relação ao paciente internado em unidade de terapia intensiva agrupando a coleta de acordo com os sistemas do corpo humano. Estes dados consistem em avaliações realizadas, diagnósticos definidos e intervenções realizadas (ANTUNES 2006).

A fim de facilitar e guiar os procedimentos por parte do usuário o sistema direciona os possíveis diagnósticos, dadas às avaliações realizadas e uma vez definidos os diagnósticos, o sistema informa possíveis intervenções a serem realizadas (DAL SASSO et al 2013).

2.1 Prontuários

Esta funcionalidade é responsável pelos cadastros, visualizações e alterações dos prontuários de pacientes. Caso o paciente não esteja cadastrado no sistema, o seu cadastro ocorre juntamente com o cadastro do prontuário (ANTUNES 2006). Além disso, o usuário terá acesso aos dados do prontuário, como os últimos dados da coleta dos sinais vitais, escala de dor, oximetria (exame não invasivo, que mede a quantidade de oxigênio no sangue) e capnografia (registro da pressão do dióxido de carbono – CO_2 no ar expirado), bem como selecionar entre os nove sistemas do corpo humano para iniciar uma avaliação (ANTUNES 2006).

2.2 Avaliações

Ao se escolher um dos sistemas do corpo humano na tela de prontuário o usuário visualizará as últimas avaliações e diagnósticos já realizados, referentes ao sistema do corpo humano atualmente selecionado, além dos últimos dados dos sinais vitais, oximetria e capnografia caso eles já tenham sido coletados (ANTUNES 2006).

Esta funcionalidade permite que o usuário cadastre uma nova avaliação, assim, preenchendo o formulário para inserir os dados referentes à avaliação no momento que for requisitado (ANTUNES 2006).

2.3 Diagnósticos

Esta funcionalidade permite associar novos diagnósticos ao paciente de acordo com o sistema do corpo humano selecionado, caso haja necessidade. Para isso, basta selecionar a ação de preencher e o SIPE irá disponibilizar um novo formulário para informar os dados (ANTUNES 2006). Esse formulário será determinado de acordo com as avaliações realizadas, ou seja, o formulário será montado com os possíveis diagnósticos conforme as avaliações anteriormente registradas (DAL SASSO et al 2013).

Aqui o usuário possui visualização dos últimos diagnósticos, avaliações realizadas para o sistema do corpo humano selecionado no prontuário, além dos sinais vitais, oximetria e capnografia (ANTUNES 2006).

Os itens de diagnósticos sugeridos automaticamente podem ser insuficientes, caso o usuário necessite de outras opções de diagnósticos, basta solicitar a visualização dos itens que foram omitidos (DAL SASSO et al 2013).

2.4 Intervenções

Para informar novas intervenções, basta acionar a ação de preencher para poder associar novas intervenções (ANTUNES 2006). As opções de intervenções serão apresentadas de acordo com os diagnósticos selecionados anteriormente (DAL SASSO et al 2013). Disponibilizando os dados dos sinais vitais, oximetria e capnografia, além das últimas intervenções e diagnósticos para o sistema do corpo humano escolhido (ANTUNES 2006).

Os itens de intervenções automaticamente sugeridos podem ser insuficientes, caso o usuário necessite de outras opções de intervenções, basta pedir a visualização dos itens omitidos (DAL SASSO et al 2013).

2.5 Administração

Esta funcionalidade agrupa as ações administrativas do SIPE, tais como cadastro, alteração ou exclusão dos usuários; cadastro, alteração ou exclusão de itens de formulário para diagnósticos e intervenções; em qualquer um dos nove sistemas do corpo humano disponível (ANTUNES 2006).

Além destes também são realizados os cadastros de alertas, que serão informados automaticamente pelo sistema, conforme os dados das avaliações de cada paciente (ANTUNES 2006).

Permite também estabelecer associações entre as avaliações, diagnósticos e intervenções para que o sistema possa filtrar os campos necessários para serem disponibilizados ao usuário, conforme os campos que já foram preenchidos. É permitido ao usuário visualizar e selecionar os campos omitidos pelo sistema caso julgue necessário (DAL SASSO et al 2013).

3 ARQUITETURA

Todas as funcionalidades da aplicação que tem seu comportamento descrito através de modelos conceituais podem ser descritas em termos de funções, classes, estruturas de dados, entre outros, chamados de componentes. Estes componentes que implementam cada função interagem entre si. Esta estrutura de componentes que se integram formando o software, recebe o nome de arquitetura de componentes de software, ou simplesmente arquitetura de software (LEITE 2000).

Com o avanço da informática surgiram padronizações destinadas a favorecer o desenvolvimento de sistemas e suas alterações. Existem padrões para diversas rotinas de um sistema que vão da estrutura de diretórios no caso de um sistema Web até a persistência dos dados.

Devido a grande variedade de padrões fica difícil destacar qual seja o melhor padrão para determinada tarefa, sendo assim este trabalho dará ênfase aos padrões de arquitetura, que buscam apoiar o desenvolvimento de sistemas através da organização estrutural do sistema.

3.1 Padrão MVC

O MVC - Model-view-controller é um padrão de desenvolvimento de software, este modelo de arquitetura separa a informação da interação do usuário com software (WIKIPEDIA_3 2013).

Cada vez mais as aplicações desenvolvidas tornam-se complexas, e por isso é de extrema importância que haja a separação da aplicação em camadas conforme

a sua finalidade. As camadas no padrão MVC são separadas em três, sendo elas Modelo (Model), Visualização (View) e Controle (Controller) (WIKIPEDIA_3 2013).

Modelo: Como o próprio nome diz é um modelo que representa os dados da aplicação, as regras do negócio que controlam o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo (UNIFIL 2012).

Visualização: Um componente de visualização renderiza o conteúdo de uma parte específica do modelo e envia para o controlador as ações que devem ser tomadas. Acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados (UNIFIL 2012).

Controle: O controlador define o comportamento da aplicação, é ele que interpreta as ações requisitadas e as mapeia para chamadas do modelo (UNIFIL 2012).

Em um cliente de aplicações Web essas ações poderiam ser cliques de botões ou seleções de menus. As ações realizadas pelo modelo incluem ativar processos de negócio ou alterar o estado do modelo. Com base na ação e no resultado do processamento do modelo, o controlador seleciona uma visualização a ser exibida como parte das respostas a solicitação do usuário (UNIFIL 2012).

Arquitetura MVC é aplicada para melhorar a organização e a modelagem de um sistema, caso necessite alguma manutenção, seja corretiva ou evolutiva, ela ocorre em menor tempo e apenas nas camadas específicas.

Logo, é preciso tomar cuidado para não confundir MVC com separação de camadas. Camadas dizem como agrupar os componentes. O MVC diz como os

componentes da aplicação interagem (OFICINA DA NET 2013). Na Figura 1 é possível visualizar o diagrama que representa a arquitetura MVC.

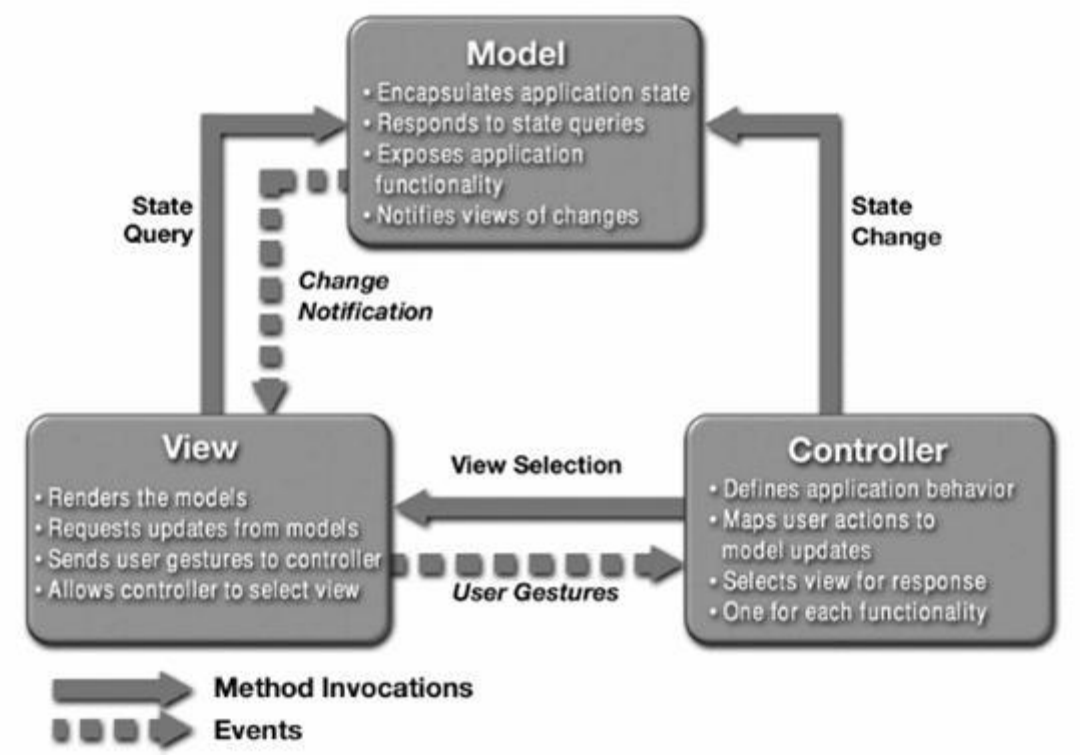


Figura 1 - Arquitetura MVC

O MVC também traz consigo outros padrões de projeto inerentes conforme mostra a Figura 2.

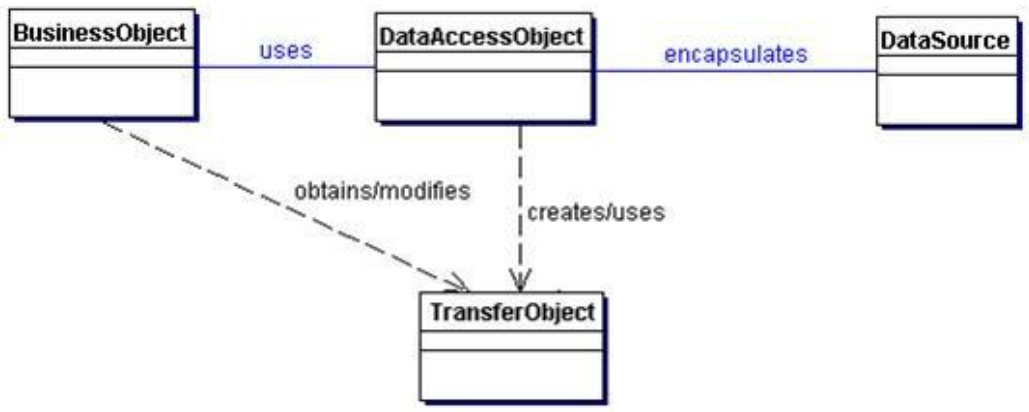


Figura 2 - Padrões Utilizados Para Arquitetura MVC

3.2 Data Transfer Object

O DTO - Data Transfer Object, tem a função de encapsular dados para que sejam transferidos a outras camadas (FOWLER 2003).

Quando se está trabalhando com aplicações em camadas, cada chamada para outras camadas pode ser custosa do ponto de vista computacional (tempo, processamento, armazenamento entre outros). Sendo assim, é preciso reduzir o número de chamadas, o que significa transferir mais dados com cada chamada. Uma maneira de fazer isto é a utilização de vários valores no retorno da chamada. No entanto, isso é muitas vezes difícil de programar e dependendo da linguagem utilizada pode ser impossível. Um exemplo é o Java que retorna um único valor (FOWLER 2003).

Como solução para esta restrição que algumas linguagens apresentam, seria criar um objeto de transferência de dados, que poderá conter todos os dados que a camada que realizou a chamada necessite. Normalmente, um tradutor é usado no lado do servidor para transferir dados entre o DTO e quaisquer objetos de domínio (FOWLER 2003). É possível encontrar em algumas referências o termo VO – Value Object para este padrão.

3.3 Data Access Object

O DAO (Data Access Object) é um padrão para persistência de dados que permite separar as regras de negócio das regras de acesso ao banco de dados. Numa aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados devem ser feitas por classes de DAO, tais como obter as conexões,

mapear objetos para tipos de dados SQL ou executar comandos SQL (BESEN 2007).

3.4 Web Service

Um Web Service, segundo definição do W3C – World Wide Web Consortium, é composto por duas estruturas: o serviço e a descrição do serviço. O serviço é um módulo de software instalado com acesso à rede, sendo oferecido pelo provedor de serviços, existindo para ser usado por um consumidor. A descrição do serviço traz detalhes da interface e da implementação de um serviço, pode incluir metadados e informação de categorização facilitando as atividades de descoberta e utilização por consumidores do serviço (OLIVEIRA 2012).

Logo o Web Service é um serviço que está disponível para que qualquer aplicação cliente utilize os seus recursos fornecidos, permitindo que sistemas sejam integrados mesmo que tenham sido desenvolvidos em plataformas diferentes, além de possibilitar que aplicações diferentes se comuniquem entre si e utilizem recursos diferentes (JAVAFREE 2013). Cada aplicação, independentemente da sua linguagem, deve ser traduzida para uma linguagem genérica e padronizada que é o formato XML – Extensible Markup Language.

Web Service se tornou a tecnologia mais indicada para comunicação entre sistemas, frequentemente usada em aplicações B2B – Business to Business. O uso de padrão na comunicação entre os serviços possibilita a independência de plataforma e de linguagem de programação (JAVAFREE 2013).

Uma desvantagem desta tecnologia é a falta de segurança, porém existem mecanismos de segurança (SSL, XML signature, XML encryption, Ws-security, SAML) (WIKIPEDIA_5 2013). As questões de segurança mais relevantes são:

- Autenticidade (certeza de que uma transação do Web Service ocorreu entre servidor e cliente);
- Privacidade (garantir que as mensagens trocadas entre servidor e cliente não serão interceptadas por alguém não autorizado);
- Integridade (mensagens trocadas entre servidor e cliente, e vice-versa devem permanecer inalteradas);

A Figura 3 traz uma representação formal de um Web Service baseado no protocolo SOAP, onde o provedor de serviços executa os seguintes passos (OLIVEIRA 2012):

1. Cria a descrição do Web Service (WSDL);
2. Publica o arquivo WSDL no repositório UDDI (opcional);
3. O cliente faz uma busca no repositório (opcional);
4. Obtém o WSDL do serviço a partir do qual cria um cliente;
5. Finalmente comunica-se com o serviço.

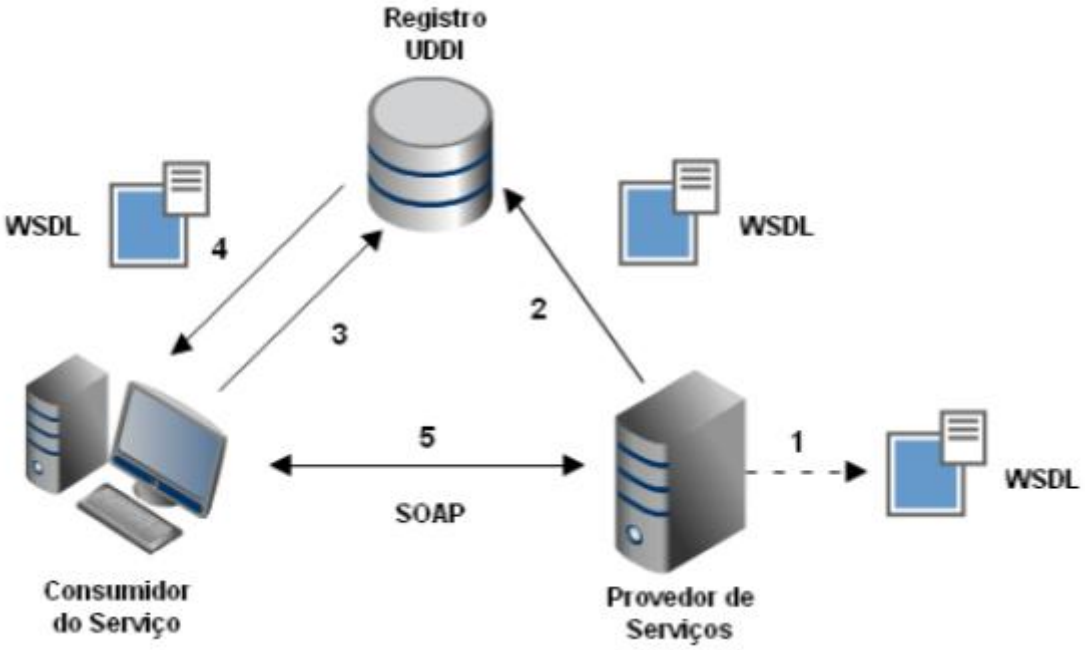


Figura 3 – Representação formal de um Web Service

3.4.1 Padrões e Protocolos

A arquitetura de um Web Service, é basicamente composta por três padrões fundamentais: SOAP, WSDL e UDDI. Inicialmente a interação de outros sistemas com o Web Service ocorre através de troca mensagens SOAP – Simple Object Access Protocol (JAVAFREE 2013), normalmente transmitidas por protocolo de transporte HTTP e padronizadas pelo XML, através da interface definida pelo WSDL – Web Services Description Language onde serão descritos os serviços.

Na arquitetura de Web Service existe ainda o UDDI – Universal Description, Discovery and Integration, que oferece os mecanismos para publicação e localização dos serviços em um repositório (OLIVEIRA 2012).

A Figura 4 mostra um diagrama para exemplificar as mensagens trocadas entre cliente e servidor na comunicação SOAP, onde duas aplicações se comunicam um Client Wrapper e um Server Wrapper que estão disponibilizando a transparência

para as aplicações, entre eles só trafega XML, seguindo o protocolo SOAP sobre HTTP (JAVAFREE 2013).

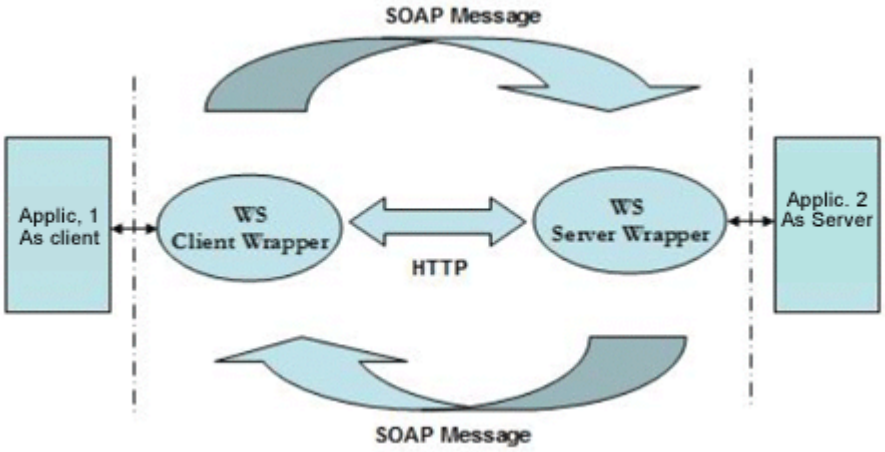


Figura 4 – Diagrama de protocolo SOAP

SOAP – Simple Object Access Protocol

O protocolo SOAP compõe uma das principais tecnologias de um Web Service, é um protocolo de comunicação de dados, ele define a sintaxe, semântica e a ordem das mensagens que serão trocadas entre aplicações e servidores (OLIVEIRA 2012).

Sendo uma tecnologia construída com base em XML e HTTP ou outro mecanismo de transporte. O SOAP permite que os documentos XML enviados ou recebidos suportem um protocolo comum de transferência de dados, portanto é o SOAP que providência o transporte de dados para os Web Services (WIKIPEDIA_5 2013).

WSDL - Web Services Description Language

A WSDL é uma especificação que permite descrever a interface do serviço de modo padronizado e estruturado, em formato XML.

Ela descreve um Web Service dividido em duas partes: abstrata e concreta. A abstrata descreve como o serviço pode ser chamado pelo cliente. A concreta define o protocolo e o endereço onde o serviço estará disponibilizado.

Esta separação é proposital, pois um mesmo serviço pode ser disponibilizado através de endereços e protocolos diferentes (VIEGAS 2013).

RESTful – Representational State Transfer

O RESTful não se trata de um padrão e sim um estilo arquitetural para construir sistemas distribuídos, foi definido por Roy Thomas Fielding no ano de 2000. Surgiu para simplificar o desenvolvimento de serviços Web (OLIVEIRA 2012).

Padrões já existentes, amplamente utilizados na Web, como XML, HTTP, URI entre outros, foram usados para o desenvolvimento dos serviços oferecidos pelo RESTful, evitando o excesso de padronização contidos nos serviços Web SOAP (OLIVEIRA 2012).

4 TECNOLOGIAS

As tecnologias empregadas neste trabalho são todas livres e independentes de plataforma. Como a aplicação será dividida em servidor e cliente a tecnologia aplicada em cada parte é distinta.

No servidor a linguagem de programação utilizada será o Java, por possuir maiores recursos para a construção de um Web Service e tornando assim o sistema independente de plataforma.

O framework CXF – Celtix e XFire também feito em Java, auxilia na disponibilização dos serviços Web. Na camada de negócio será utilizado o framework Spring para instancialização de objetos e controle de transações. Para camada de persistência foi utilizado o framework Hibernate, que faz o mapeamento objeto-relacional.

O banco de dados será o PostgreSQL, trata-se de um banco objeto-relacional de código aberto, bastante robusto, com vantagem de executar nos principais sistemas operacionais.

No cliente Web a opção foi de utilizar a linguagem PHP que é bastante flexível para a criação de aplicações Web, permite utilizar recursos como o Smarty que possibilita a utilização de modelos de layout nas telas, assim como o framework BootStrap que facilita a criação das telas.

No cliente móvel foi utilizada a plataforma Android, devido ao fato de oferecer suporte à linguagem Java, e também oferece recursos que facilitam o desenvolvimento independente do sistema operacional utilizado.

O uso destas tecnologias no lado do servidor e cliente sugere a utilização da arquitetura MVC (Model-View-Controller), que separa a aplicação em camadas:

modelo, visualização e controle. Facilitando a manutenção e customização do sistema.

4.1 Java

Java, até a conclusão deste trabalho, é considerada uma das linguagens mais populares nos últimos anos (TIOBE 2013). Ela não é somente uma linguagem, mas também uma plataforma de desenvolvimento segue os paradigmas de programação orientada a objeto e tem como um de seus princípios a portabilidade, que é a capacidade de se utilizar a mesma aplicação em diversos sistemas operacionais e dispositivos sem a necessidade de reescrever o código (JAVA 2012).

Isso se deve ao fato de que ao invés de ser compilada para código nativo da máquina, ela é traduzida para um código intermediário chamado *bytecode*, que é interpretado por uma máquina virtual conhecida como JVM – Java Virtual Machine, tornando o código criado independente de plataforma, sendo necessário que uma versão da JVM esteja instalada para interpretar o *bytecode* (JAVA 2012).

Desenvolvida pela Sun Microsystems na década de 90 e atualmente pertencente a Oracle, a versão utilizada neste trabalho será a 7.0 (JAVA 2012).

4.2 Android

A plataforma Android é baseada no sistema operacional Linux, customizada para funcionamento em dispositivos móveis foi desenvolvida por um consórcio de empresas chamado de Open Handset Alliance sendo o Android efetivamente o

primeiro projeto e deve seu lançamento em novembro de 2007 sendo distribuído com uma licença Apache 2.0 e GPLv2 (STENZEL 2013).

Este mesmo consórcio disponibilizou para quem deseja desenvolver aplicações para esta plataforma um SDK – Software Developement Kit, que permite a criação de aplicações na linguagem de programação Java, mesmo o Android não possuindo uma máquina virtual Java. Ele possui outra máquina virtual chamada Dalvik, que é um projeto Open Source especialmente projetado para dispositivos móveis. Porém existe uma aplicação chamada "dx" que é responsável por converter o *bytecode* gerado pelo Java no *bytecode* da Dalvik (THE CODE BAKERS 2013). Sua estrutura é dividida em camadas, normalmente utilizada nas plataformas de sistemas embarcados em dispositivos móveis. Esta estrutura pode ser observada na Figura 5 – Arquitetura do ambiente Android

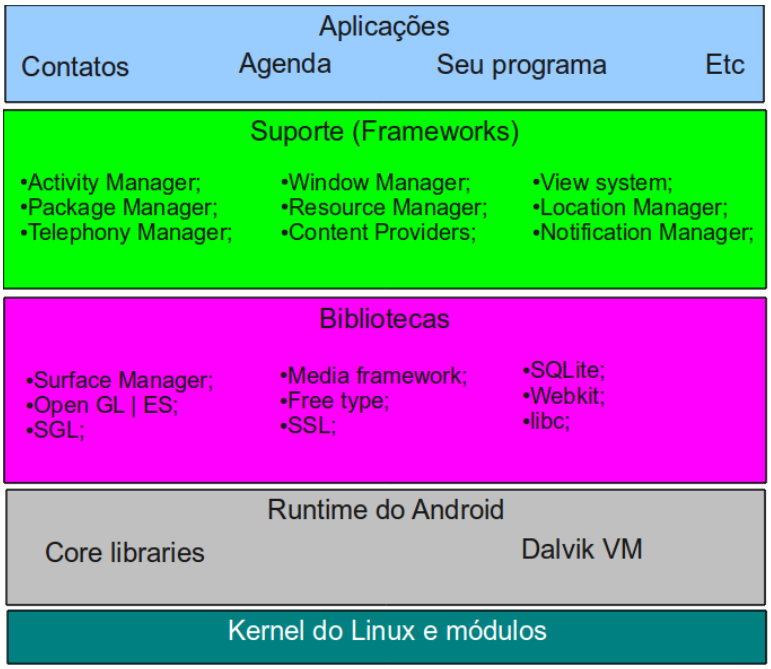


Figura 5 – Arquitetura do ambiente Android

Na camada de aplicação ocorre a interação com o usuário através dos aplicativos disponíveis que são desenvolvidos a partir da API disponível pela camada inferior. A camada framework fornecer suporte com componentes que podem ser reutilizados pelas aplicações da camada superior. Diferentemente das arquiteturas tradicionais esta camada fornece suporte para o compartilhamento de dados entre aplicativos (PIANTINO, 2008).

Abaixo desta, encontra-se a camada de bibliotecas, as quais permitem a manipulação de recursos como áudio, vídeo, banco de dados e Internet. Paralelamente a ela está a camada de *runtime*, a qual estende as funcionalidades da linguagem Java e delega ao sistema operacional o controle sobre as threads e execução das aplicações, nesta camada que se encontra a máquina virtual Dalvik que é responsável por recompilar os *bytecodes* para um formato mais otimizado. E por último, mas não menos importante, está a camada do *kernel* Linux, que fornece as abstrações entre o hardware e as camadas superiores, possibilitando acesso a recursos de áudio, vídeo e protocolos de rede (PIANTINO, 2008).

Uma aplicação Android é orientada a eventos. Este evento consiste em um sinal, que pode ter sido gerado pelo usuário ou não. Toda aplicação possui a sua disposição um conjunto de componentes com funções bem definidas que são fornecidos pela camada framework e estão divididos em Activities, Services, Content Providers e Broadcast Receivers (THE CODE BAKERS 2013).

As Activities são responsáveis pela formação das telas e captura dos eventos gerados pelo usuário e podem invocar outros componentes como: outras Activities, Services entre outros (THE CODE BAKERS 2013).

Os Services são componentes que executam operações de longa duração, sem que seja necessária intervenção do usuário podendo ser executadas em segundo plano mesmo que o usuário saia da aplicação (THE CODE BAKERS 2013).

Já os Content Provider são responsáveis pelo armazenamento e recuperação dos dados, disponibilizando-os para outras aplicações, sendo assim, só é utilizado caso se queira compartilhar dados com outras aplicações (THE CODE BAKERS 2013).

E os Broadcast Receiver são classes que recebem os alertas gerados pelo sistema, como por exemplo, o bloqueio da tela, possibilitando tomar atitudes de acordo com o tipo de alerta recebido (THE CODE BAKERS 2013).

4.3 PHP

PHP, originalmente Personal Home Page, atualmente chamado de *PHP: Hypertext Preprocessor,* trata-se de uma linguagem de programação de computadores interpretada, de software livre, muito utilizada para gerar conteúdo dinâmico na Web (PHP 2013).

A linguagem foi criada 1994 por Rasmus Lerdof, que escreveu uma implementação em C, a qual permitia às pessoas desenvolverem de forma muito simples suas aplicações para Web (PHP 2013).

No início era formada por um conjunto de scripts voltados à criação de páginas e à medida que essa ferramenta foi crescendo em funcionalidades, novas atualizações foram lançadas (PHP 2013).

O código passou por uma reforma completa em 1996, e Rasmus nomeou essa versão de PHP/FI (Personal Home Pages/Forms Interpreter) e disponibilizou seu código na Web, para compartilhar com outras pessoas (PHP 2013).

Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores Web. É muito parecida, em sintaxe, em tipos de dados e até mesmo em funções, com a linguagem C ou com C++, o que a torna ferramenta acessível para várias aplicações (PHP 2013). Neste trabalho será utilizada a versão 5.4.

4.4 Smarty

O Smarty possibilita o gerenciamento de *templates* para PHP, oferecendo uma forma simples de controlar a separação da aplicação lógica e o conteúdo de sua apresentação. O principal objetivo do Smarty é a separação das regras de negócio da lógica de apresentação. Com isto os *templates* contém somente a lógica da apresentação (SMARTY 2013).

O Smarty começou a ser desenvolvido por volta de 1999. Após o término das especificações, se deu início a um sistema que possibilite o gerenciamento de *templates* em linguagem C que se esperava ser aceito para rodar com o PHP. Não só esbarraram em muitas barreiras técnicas, como também houve um enorme debate sobre o que exatamente um sistema de *template* deveria ou não fazer. A partir desta experiência, o sistema de *template* foi escrito para ser uma classe do PHP, para que fosse utilizado da forma mais conveniente (SMARTY 2013).

4.5 BootStrap

Criado por um designer e um programador no Twitter (rede social), tornou-se um dos Frameworks mais populares em projetos de código aberto do mundo para construção de páginas para a Web (BOOTSTRAP 2013). Ele consiste na utilização de classes CSS e JavaScript nas paginas HTML, possibilitando de forma facilitada a criação de layout das telas de sistemas Web.

BootStrap foi desenvolvido em meados de 2010 e antes de ser um framework open-source, era conhecido como *Twitter Blueprint*. Ele serviu como guia de estilo para o desenvolvimento de ferramentas internas na empresa por mais de um ano antes de seu lançamento público (BOOTSTRAP 2013).

Sendo lançado em agosto 2011, já tendo passado por mais de vinte versões, incluindo dois grandes releases com v2 e v3. Na versão 2, foram adicionadas funcionalidades sensíveis a toda a estrutura como um estilo opcional. Na versão 3, a biblioteca foi reescrita novamente para tornar os layouts desenvolvidos compatíveis com dispositivos móveis de forma transparente ao desenvolvedor (BOOTSTRAP 2013). Na realização deste trabalho será utilizada a versão 3.0.1.

4.6 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados (SGBD) objeto-relacional de código aberto, além de contar com avançados recursos. Com uma arquitetura que adquiriu forte reputação de confiabilidade, integridade de dados e

conformidade a padrões, podendo ser executado nos principais sistemas operacionais (POSTGRESQL 2013).

Este SGBD também possui interfaces nativas de programação das principais linguagens, outra característica é a alta escalabilidade (capacidade de gerenciar grande quantidade de dados, e de usuários concorrentes) (POSTGRESQL 2013).

Foi desenvolvido pela Universidade de Berkeley na Califórnia, a qual descontinuou o projeto na sua quarta versão, foi quando a comunidade de software livre assumiu a continuação de seu desenvolvimento (POSTGRESQL 2013). Será utilizada neste trabalho a versão 9.1 deste SGBD.

4.7 Frameworks

Framework em um ambiente de desenvolvimento de software trata-se de uma abstração que une códigos comuns entre vários projetos de software criando uma funcionalidade genérica. Pode chegar a uma funcionalidade específica, por meio de configuração, durante a programação de uma aplicação. O framework ao contrário das bibliotecas é quem determina o fluxo de controle da aplicação (WIKIPÉDIA_1 2013).

Framework trata-se de um conjunto de códigos responsáveis por executar tarefas de determinado domínio em uma aplicação (FAYAD e SCHMIDT apud WIKIPÉDIA_1 2013). Serão empregados alguns frameworks de desenvolvimento, apoiando e garantindo uma correta utilização da arquitetura e dos padrões que serão utilizados.

4.7.1 Apache CXF

Apache CXF é um framework desenvolvido na linguagem Java e de código livre sendo desenvolvido pela Apache a partir da combinação de outros dois projetos, Celtix e XFire (APACHE_CXF 2013).

O que levou os engenheiros da Apache a unirem os dois projetos foi a necessidade de sempre serem utilizados em conjunto. O Celtix é um aglomerado de APIs, que fornece recursos de segurança e transporte e tem por objetivo simplificar e facilitar a construção, integração e o reuso de componentes por meio de uma arquitetura orientada a serviços. O XFire por outro lado oferece suporte a todos os padrões relacionados a Web Services, além de ter perfeita integração com o Spring e com os protocolos HTTP, JMS e outros (APACHE_CXF 2013).

Ele oferece suporte a elaboração e consumo de Web Services. Seguindo as especificações JAX-WS e JAX-RS e oferecendo suporte a uma grande diversidade de protocolos de mensagem: SOAP, XML/HTTP, RESTful HTTP ou CORBA. Ainda permite trabalhar com os protocolos de transporte HTTP e JMS (APACHE_CXF 2013).

A utilização do tipo de protocolo, seja de mensagem ou transporte, pode ser definido em seu arquivo de configuração como será visto mais a frente.

A arquitetura do CXF é baseada nos componentes, como pode-se ver na Figura 6 abaixo.

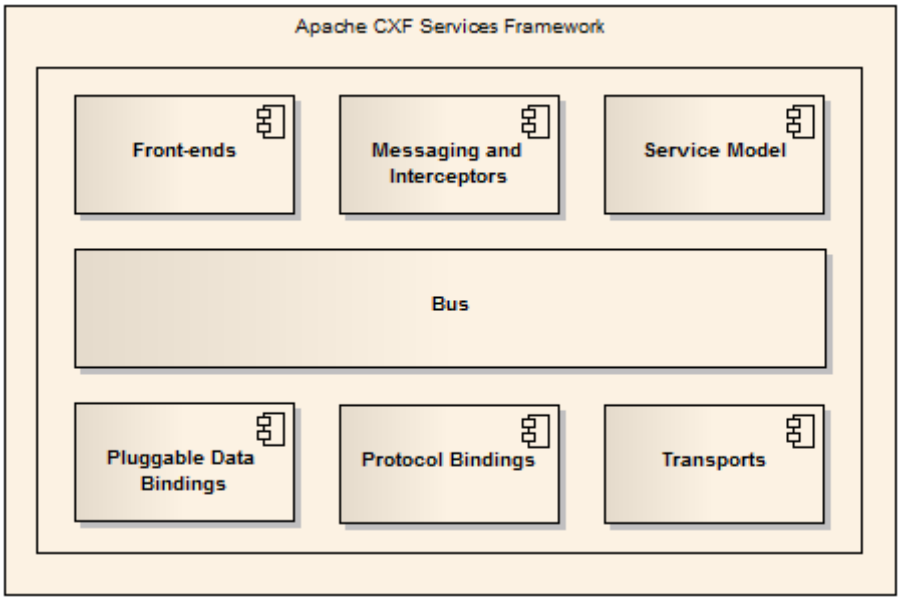


Figura 6 - Arquitetura CXF

Possui uma arquitetura bem flexível em que o componente Bus é um provedor de recursos e os demais componentes possuem responsabilidades específicas e podem ser ajustados de acordo com a necessidade de cada aplicação (APACHE_CXF 2013). Neste trabalho será utilizada a versão 2.7.6.

4.7.2 Spring

O Spring é um framework também de código livre para a plataforma Java, oferece suporte de infraestrutura para aplicações Java, projetado para não ser invasivo, sendo baseado nos padrões de projeto para inversão de controle e injeção de dependência (SPRING 2013). Inversão de controle é um padrão de desenvolvimento, onde a o controle (sequência) de chamadas dos métodos não é diretamente determinado pelo programador, e sim delegado a um terceiro como um container ou outro componente, característica comum de alguns frameworks (WIKIPÉDIA_4 2013). Injeção de dependência é um padrão de desenvolvimento,

utilizado para manter baixo o nível de acoplamento entre diferentes módulos de um sistema, onde as dependências entre os módulos não são definidas por programação, mas pela configuração de um container, responsável por "injetar" em cada componente suas dependências declaradas (WIKIPÉDIA_4 2013).

Este framework é formado por recursos organizados em aproximadamente 20 módulos, estes são agrupados em Core Container, Data Access / Integração, Web, AOP – Aspect Oriented Programming, instrumentação e teste (SPRING 2013).

O núcleo container é o responsável em instanciar classes (criar fisicamente uma representação da classe) da aplicação Java bem como definir relacionamentos entre elas através do arquivo de configuração em formato XML, o chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Com isso o framework garante um baixo acoplamento entre as classes em aplicações orientadas a objetos (SPRING 2013).

4.7.3 Hibernate

O Hibernate é um framework utilizado para realizar o mapeamento objeto-relacional escrito na linguagem Java. Existe também uma versão disponível em .Net com o nome NHibernate (HIBERNATE 2013).

Ele simplifica o mapeamento dos atributos entre um SGBD e o modelo objeto da aplicação, através de arquivos (XML) ou anotações. Tornando o desenvolvimento do sistema mais fácil e ágil, tendo em vista que uma parte considerável do tempo de desenvolvimento de um projeto é gasto em consultas SQL e tratamento de transações (HIBERNATE 2013).

4.8 Servidores

Cada linguagem de programação necessita de um servidor específico para que possam funcionar. Aqui será descrito os servidores que serão utilizados tanto para Java, como para o PHP.

4.8.1 Servidor Apache TomCat

O Apache Tomcat é um servidor de aplicações Java para Web que implementa as tecnologias JavaServlets e JavaServer Pages, é de código livre como as outras tecnologias utilizadas neste trabalho. Ele pode se comportar como um servidor Web (HTTP) ou funcionar integrado a um servidor Web dedicado (como o Apache ou o IIS) (APACHE_TOMCAT 2013). Originou no Projeto Apache Jakarta e é oficialmente autorizado pela Sun como a implementação de referência para as tecnologias Servlet e JavaServer Pages (JSP).

Atende as especificação J2EE (Java 2 Platform, Enterprise Edition), sendo possível fazer a conexão com o datasource através dele e até mesmo tratar requisitos de segurança. Como é inteiramente escrito em Java para ser executado necessita de uma JVM (Java Virtual Machine - Máquina Virtual Java) instalada (APACHE_TOMCAT 2013). Neste trabalho a versão utilizada será a 7.0.42.

4.8.2 Servidor PHP

Para poder executar aplicações em PHP é necessário instalar os módulos relativos à linguagem num servidor Web dedicado, seja o IIS da Microsoft ou

Apache. Este trabalho utilizará o Apache na versão 2 e foram instalados os módulos do PHP na versão 5.4 (PHP 2013).

Ao utilizar a versão 5.4 do PHP o uso do Apache em ambiente de desenvolvimento já não se faz necessária uma vez que esta versão traz consigo um servidor Web interno, porém para deixar o ambiente mais próximo de um cenário real optou-se por utilizar o Apache (PHP 2013).

4.9 Eclipse

Para auxiliar no desenvolvimento foi utilizada a IDE (Integrated Development Environment) Eclipse, que possui suporte a diversas linguagens, dentre elas Java e PHP que foram utilizadas neste trabalho. Foi desenvolvida em Java, é de código aberto, e bastante flexível. Possibilita a instalação de vários módulos, permitindo agregar recursos capazes de automatizar algumas tarefas e consequentemente otimiza o desenvolvimento. O projeto Eclipse foi originalmente criado pela IBM em Novembro de 2001 e apoiado por um consórcio de empresas de software (ECLIPSE 2013).

4.9.1 Android Developer Tools plugin (ADT)

O Android Developer Tools (ADT) é um plugin que permite adequar a IDE Eclipse para o desenvolvimento de aplicativos Android. Ele fornece recursos que ajudam a construir, depurar, testar e criar instaladores. Fornece um assistente para criação de projetos padrão Android e componentes, o que facilita para quem está iniciando no desenvolvimento com esta plataforma. Permite a criação das telas de

forma fácil através de recurso gráficos como arrastar e soltar componentes (STENZEL 2013).

Além destes recursos ele permite simular a aplicação em diversos tipos de dispositivos ou até mesmo executá-la diretamente no dispositivo sem que seja necessária sua instalação no mesmo, permitindo a análise de consumo da memória, desempenho da CPU entre outros recursos do dispositivo (STENZEL 2013).

5 SEGURANÇA

Em aplicações que utilizam a Internet como meio para troca de informações, uma das principais preocupações é a segurança, sendo seus principais desafios à integridade, confidencialidade, autenticação e autorização. A integridade e confidencialidade garantem que os dados transmitidos não sejam alterados ou visualizados indevidamente. A autenticação e autorização garantem a identidade da fonte das informações e que somente quem possuir permissão poderá acessar os serviços disponibilizados. (RECKZIEGEL 2006).

Para se alcançar os requisitos mínimos de integridade e confidencialidade se faz necessário o uso de criptografia, que consiste em técnicas pelas quais a informação pode ser transformada da sua forma original para outra ilegível, de modo que possa ser revertida facilmente para sua forma original, por quem possuir a "chave secreta", que dependendo da técnica utilizada somente o destinatário a detém (SAMPAIO 2008).

Entre as diversas técnicas existentes pode-se destacar a criptografia simétrica e a criptografia assimétrica.

A chave simétrica deve ser conhecida por quem está trocando as mensagens, pois ela é utilizada tanto para criptografar, quanto para descriptografar, na Figura 7 podemos observar este modelo.

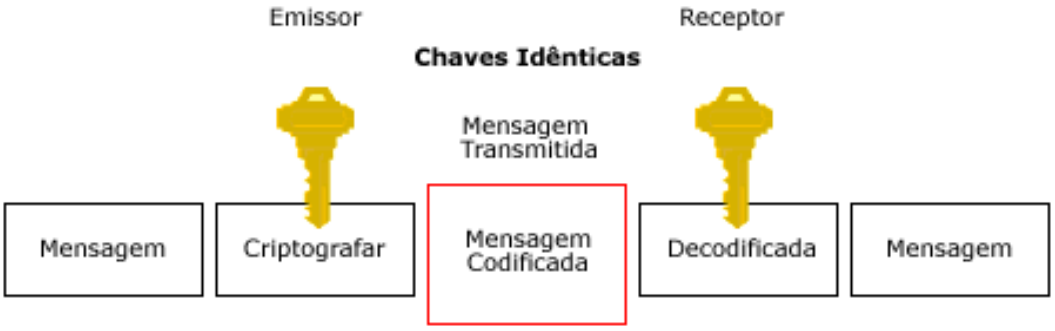


Figura 7 - Exemplo de Criptografia Simétrica

Esta técnica tem como vantagem, sobre a criptografia assimétrica, a velocidade em cifrar ou codificar as mensagens, pois o tamanho das chaves utilizadas é menor, porém o fato de utilizar chave única pode comprometer a segurança, pois a chave deve ser compartilhada, ocorrendo este compartilhamento pela Internet a chave pode ser capturada, e quem a possuir poderá decodificar qualquer mensagem que venha a ser cifrada com ela (SAMPAIO 2008).

Em contra partida a criptografia assimétrica faz uso de um par de chaves distintas, chamadas de pública e privada, que se relacionam matematicamente. Assim a chave pública poderá ser distribuída sem que ocorra quebra de segurança, pois a chave privada fica em posse somente do dono do par de chaves e quando uma chave é utilizada para criptografar a outra é necessária para que seja possível descriptografar para a mensagem original (RECKZIEGEL 2006).

Com o uso desta técnica é possível se garantir a confidencialidade ou a autenticidade da mensagem, esta garantia vai depender de qual chave seja utilizada para criptografar. Na Figura 8 podemos observar o uso das chaves quando o objetivo é ter confidencialidade, pois somente o portador da chave privada é que poderá descriptografar (JUNIOR 2012).

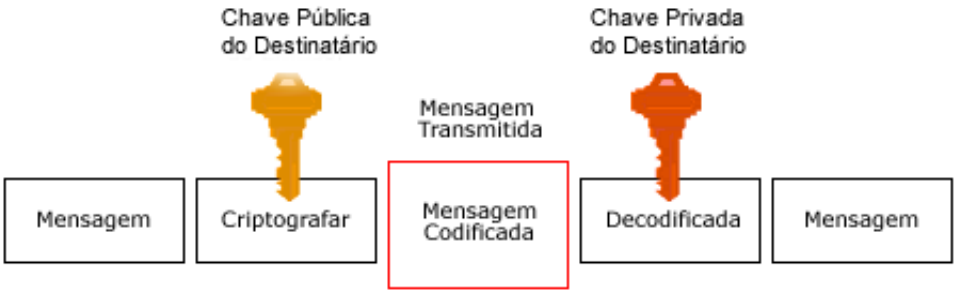


Figura 8 - Exemplo de Criptografia Assimétrica com Confidencialidade

E na Figura 9 temos o exemplo quando se quer conseguir autenticidade, pois somente a chave pública de quem criptografou poderá descriptografar.

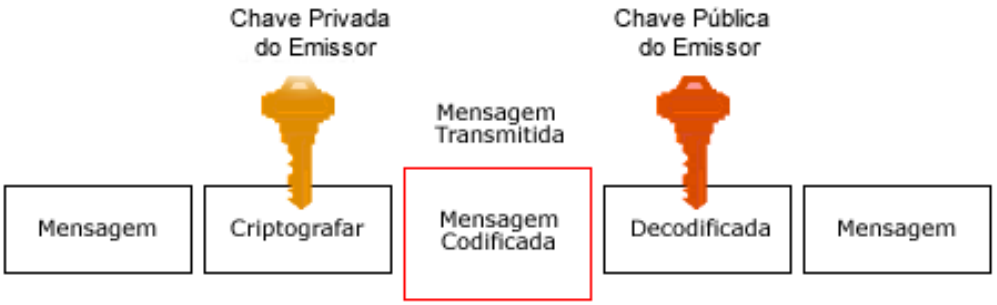


Figura 9 - Exemplo de Criptografia Assimétrica com Autenticidade

Tanto a técnica de criptografia simétrica quando a assimétrica possuem diversos algoritmos de possibilitam suas implementações, dentre eles destacamos os algoritmos DES, IDEA, AES para criptografia simétrica é os algoritmos RSA, Diffie-Hellman para criptografia assimétrica. Uma relação mais detalhada dos algoritmos simétricos pode ser visualizada no Anexo I e no Anexo II os algoritmos para assimétricos.

5.1 Segurança em Web Service

A arquitetura de Web Service somente, não garante as questões de segurança pois seus protocolos não preveem a proteção de mensagens. O

problema está na natureza de um aplicativo de serviços da Web onde cada mensagem precisa trafegar por um ou mais nós, qualquer um deles pode ler e/ou alterar uma mensagem (CHASE 2013).

A especificação do protocolo SOAP não prevê a proteção de mensagens, mas fornece especificação para incluir informações de segurança, que serão incluídas no cabeçalho da mensagem, porém não especifica quais deveriam ser essas informações. Para tratar isso, existe a especificação de WS-Security a qual determina a forma de repassar estas informações (CHASE 2013).

Com a utilização do WS-Security, é possível obter de forma seletiva cada um dos requisitos da tríade de segurança (confidencialidade, integridade e autenticidade), possibilitando que um ou todos eles sejam adotados como solução, porém seu uso pode ser muito custoso do ponto de vista computacional e dependendo do serviço sua adoção pode gerar problemas de performance.

Na Figura 10 é apresentado um comparativo em relação ao tempo de resposta do serviço utilizando diferentes formas de segurança especificada pelo WS-Security. Os testes foram executados usando cada uma das seguintes configurações (IBM_WS 2013):

- *plain*: Sem segurança;
- ssl: HTTPS usado para conexão com o servidor;
- *username*: UsernameToken de texto simples do WS-Security nos pedidos;
- *sign*: Assinatura do WS-Security de corpo e cabeçalhos, com registro de data e hora;
- *encr*: Criptografia do WS-Security do corpo;
- *signencr*: Assinatura do WS-Security de corpo e cabeçalhos, com registro de data e hora e criptografia do corpo.

Os tempos de resposta variam mais de dez vez entre a configuração *plain* e a configuração *signencr* (IBM_WS 2013).

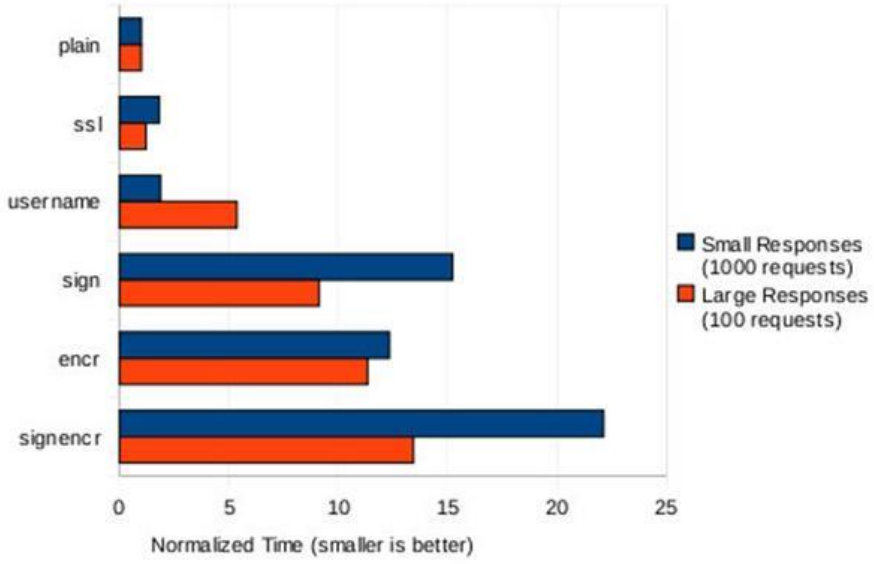


Figura 10 - Comparação de Tempo entre diversas formas de segurança (IBM_WS 2013)

A Figura 11 mostra os tamanhos das mensagens com as diferentes formas de seguranças:

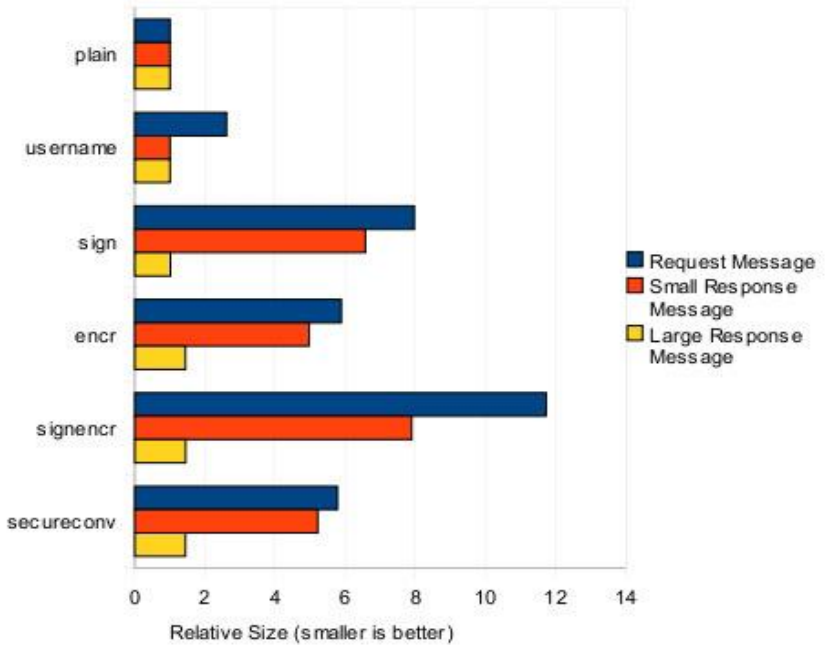


Figura 11 - Comparação do Tamanho da Mensagem (IBM_WS 2013)

6 ANÁLISE DO SIPE

Os estudos que envolvem o SIPE estão em constante aprimoramento, principalmente nos processos de enfermagem, portanto o sistema necessita de adequação aos novos requisitos que vão sendo levantados nestes estudos. Desde o desenvolvimento de novas funcionalidades, mudanças de layout para melhorar a eficiência e a ergonomia, até mesmo mudanças nas regras de negócio.

Para que o sistema possa acompanhar a evolução dos estudos ele deve seguir as boas práticas de programação facilitando eventuais manutenções evolutivas ou corretivas.

Uma análise mesmo que simplificada deste sistema se faz necessária, seja para conhecimento do mesmo, como para levantar pontos fracos ou vulnerabilidades que este possa ter. Sendo extremamente importante para atingir os objetivos propostos neste trabalho.

Para essa análise alguns pontos deverão ser avaliados, seguindo alguns requisitos da boa prática de programação.

6.1 Organização dos arquivos

Para facilitar as manutenções que venham a ocorrer neste sistema, seria bom se os arquivos estivessem bem organizados. Para isso busca-se agrupar os arquivos por suas finalidades para facilitar sua localização e entendimento de como estão estruturados. Além disso, evitam a duplicidade de arquivos que podem comprometer a integridade dos dados e ainda podem dificultar as manutenções.

Outra boa prática é separar as bibliotecas de terceiros que são utilizadas no sistema.

O SIPE encontra-se com uma estrutura de diretórios que sugerem duplicidade de arquivos e diretórios distintos que exercem a mesma função, como mostra a Figura 12. Ao realizar uma pesquisa dentre os arquivos do sistema, foram encontrados 133 arquivos que possuem mais de uma versão.

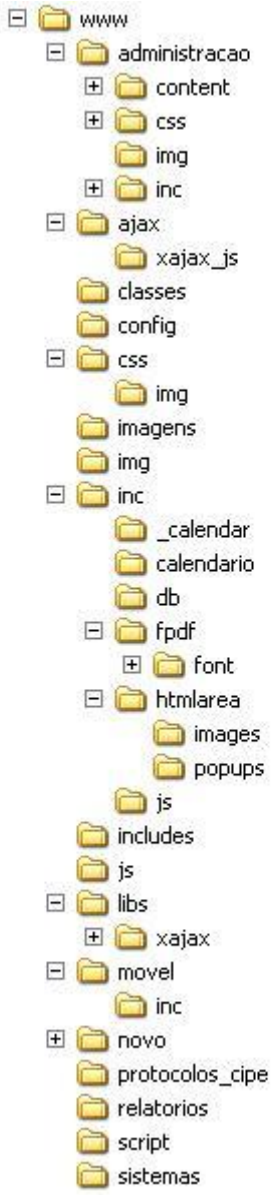


Figura 12 - Estrutura de diretórios

Além destes pontos também foram encontradas bibliotecas de terceiros em diversos pontos na estrutura de diretórios do sistema.

6.2 Estruturas do código fonte

Ao Analisar alguns trechos do código fonte a fim de verificar como estavam estruturados, foi constatado que no mesmo código existiam instruções SQL, regras de negócio e código HTML. Um exemplo deste caso pode ser visualizado no Anexo III deste trabalho.

Também foram buscados indícios de parâmetros de configurações como, por exemplo, usuário de acesso à base de dados, encontrados em mais de um arquivo com estas configurações, como mostra a Figura 13.

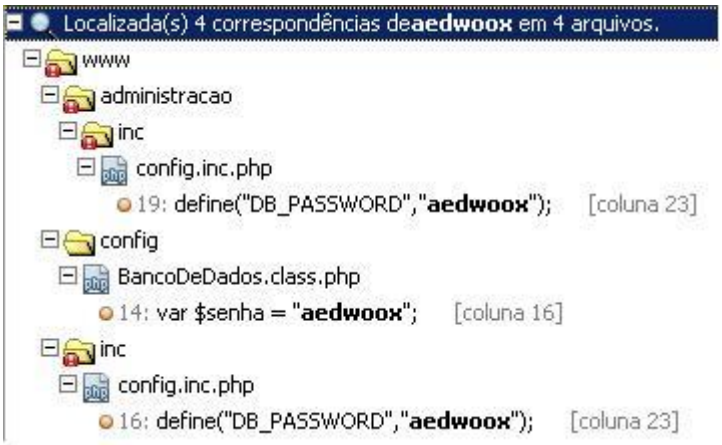


Figura 13 – Padrões de configuração duplicados

Ainda dentro dessa questão de acesso à base de dados foram encontradas mais de uma classe ou função com mesma finalidade, como mostra a Figura 14.

Figura 14 – Scripts duplicados

Assim gerando duplicação de código, caso alguma questão que envolva a base de dados tenha que ser alterada isso dificultará a manutenção.

Além destas questões foram encontrados parâmetros de configuração no mesmo arquivo, que pelo nome, fazia alusão somente ao banco de dados, como mostra o Anexo IV.

Após breve análise avaliando alguns dos principais quesitos, conforme as boas práticas de programação, que deveriam ser atualmente adotadas pelos desenvolvedores de software, não foram observadas no SIPE. Ao invés disso o sistema se apresentava desorganizado e desestruturado, deixando assim o sistema vulnerável sob o ponto de vista estrutural, prejudicando a manutenção do sistema.

O resultado desta análise serve para justificar um dos principais objetivos deste trabalho que é a reestruturação do SIPE, que passará a ter uma arquitetura de Web Service, e aplicação de padrão de projeto conforme o modelo MVC.

7 IMPLEMENTAÇÃO

Como proposto nos objetivos a reestruturação do sistema SIPE ficou dividida em três partes. A primeira chamada de servidor, onde estarão implementados os serviços Web que serão disponibilizados para que outras aplicações possam utilizá-lo, a segunda parte chamada de cliente Web, onde será implementada a utilização dos serviços Web disponibilizados pelo servidor e a terceira chamada de cliente móvel. Cada parte será descrita como foram estruturadas e implementadas.

7.1 Segurança

Durante a implementação uma preocupação foi em relação à segurança, tanto o sigilo das informações durante a transmissão dos dados entre a aplicação cliente e o servidor, como o controle de acesso para que aplicações e/ou usuários não possam realizar ações além daquelas a eles atribuídos.

Para garantir a segurança é necessária a utilização de técnicas, para evitar que alguém, além do destinatário desejado, leia as informações. Também é preciso se evitar que aplicações, além das que possuem autorização, enviem mensagens ao servidor. Técnicas que quando aplicadas em arquitetura de Web Service são definidas na aplicação servidor.

Ao se utilizar o framework CXF no servidor, a aplicação de padrões do WS-Security ocorre de forma facilitada, pois o CXF implementa as especificações de segurança através da biblioteca WSS4J - Web Services Security for Java, que é de código aberto e pode ser configurada através dos arquivos de configuração do próprio framework. Abaixo seguem dois trechos de códigos que devem ser inseridos

entre a tag “jaxws:endpoint”, o primeiro exemplifica o uso somente de usuário e senha, a fim de se garantir que somente possam acessar o serviço quem estiver devidamente autorizado, e o segundo que mostra como configurar o WS-Security para utilizar criptografia nas mensagens.

```
<jaxws:properties>
      <entry key="ws-security.callback-handler"
               value=" br.ufsc.tcc.sin.seguranca.ServerCallback" />
</jaxws:properties>
```

Controle de usuário e senha

```
<jaxws:properties>
     <entry key="ws-security.signature.properties"
               value="server-crypto.properties" />
     <entry key="ws-security.signature.username" value="serverkey" />
     <entry key="ws-security.encryption.username" value="useReqSigCert" />
     <entry key="ws-security.callback-handler"
               value=" br.ufsc.tcc.sin.seguranca.ServerCallback" />
</jaxws:properties>
```

Criptografia

Esta mesma facilidade não está presente na linguagem de programação PHP, pois ela não possui suporte ao WS-Security. Pensando neste fato, e que a mesma situação possa ocorrer com outras linguagens, este padrão não foi adotado neste trabalho. Ficando sob responsabilidade da própria aplicação implementar sua segurança.

Para facilitar esta implementação os serviços disponibilizados recebem e retornam um único parâmetro. Este parâmetro consiste em uma variável do tipo *string* que possui como conteúdo as informações que se quer transmitir no formato XML. Sobre este *string* será aplicada a regra de segurança, que garantirá a confidencialidade das informações trocadas entre cliente e servidor. Para

implementar a regra de confidencialidade duas técnicas de criptografia foram adotadas, a criptografia assimétrica com o uso do algoritmo RSA e a criptografia simétrica com o algoritmo AES.

A criptografia assimétrica é aplicada para possibilitar o envio da chave simétrica de forma segura, e uma vez a chave enviada, as próximas trocas de informação ocorrem com a utilização da criptografia simétrica. Na Figura 15 é possível visualizar a representação gráfica deste processo.

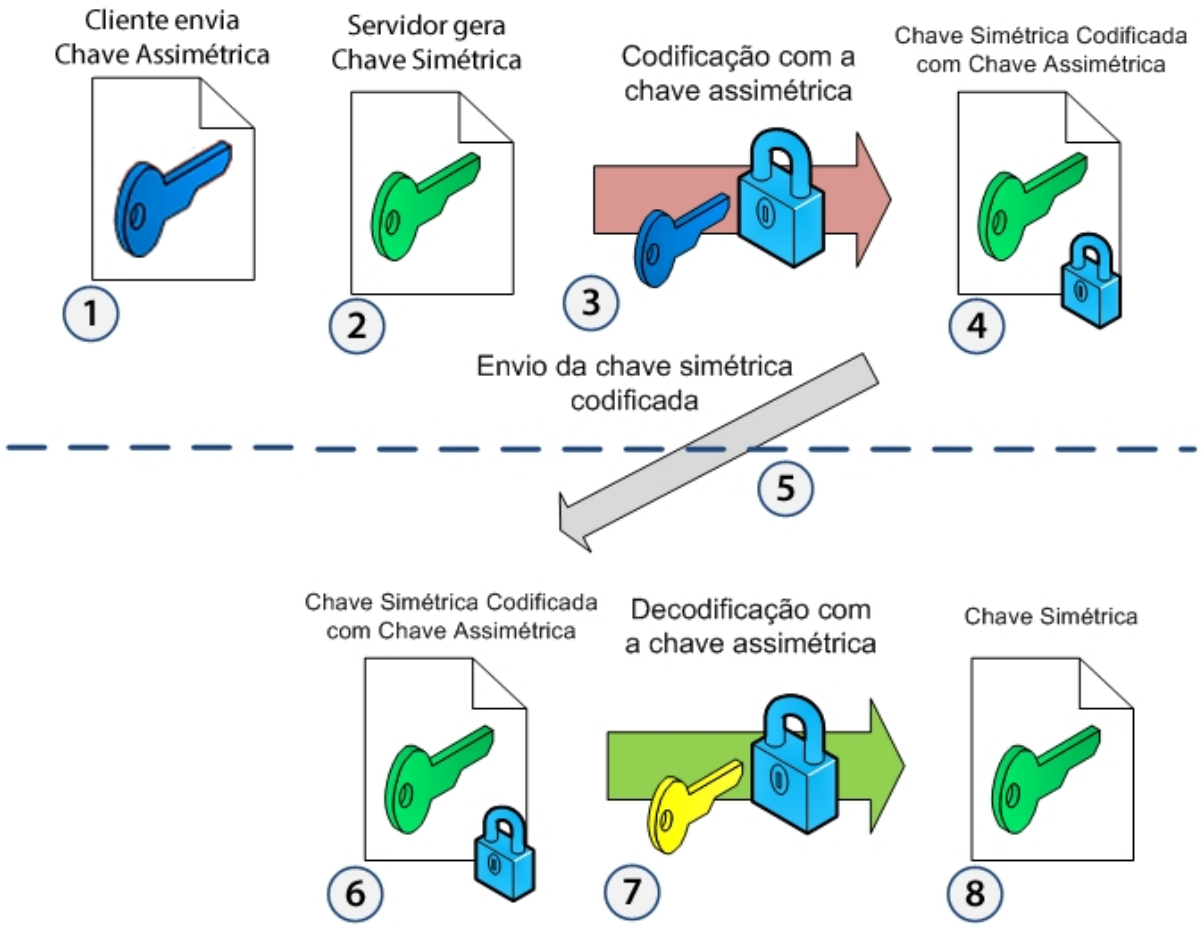


Figura 15 - Processo de troca de chaves

As trocas de chave ocorrem da seguinte forma. A aplicação cliente realiza chamada ao servidor passando sua chave pública, o servidor registra o início da

troca de mensagens, registrando o início da sessão e retorna ao cliente a chave que será utilizada, nas próximas trocas de mensagens, para criptografar e decodificar as mensagens, juntamente com o código de acesso que servirá para identificar a sessão ativa, e qual chave está se utiilizando na criptografia. O envio destas informações será realizado de forma criptografada, com a chave pública recebida, o que garante que somente o destinatário poderá visualizar as informações.

Abaixo é possível visualizar a troca de mensagem quando a aplicação cliente inicia a comunicação enviando sua chave pública e recebe a resposta criptografada com sua chave pública.

```
INFO: Inbound Message
----------------------------
Address: http://localhost:8080/TCC/acesso
Encoding: UTF-8
Http-Method: POST
Content-Type: text/xml; charset=utf-8
Headers: {Authorization=[Basic d2ViOndlYg==], connection=[Keep-Alive], Content-
Length=[852], content-type=[text/xml; charset=utf-8], host=[localhost:8080],
SOAPAction=[""], user-agent=[PHP-SOAP/5.4.12]}
Payload:
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ns1="http://servico.sin.tcc.ufsc.br/">
      <SOAP-ENV:Body>
            <ns1:iniciarComunicacao>
                  <chavePublica>
rO0ABXNyABRqYXZhLnNlY3VyaXR5LktleVJlcL35T7OImqVDAgAETAAJYWxnb3JpdGhtdAASTGphdmEvbG
FuZy9TdHJpbmc7WwAHZW5jb2RlZHQAAltCTAAGZm9ybWF0cQB+AAFMAAR0eXBldAAbTGphdmEvc2VjdXJp
dHkvS2V5UmVwJFR5cGU7eHB0AANSU0F1cgACW0Ks8xf4BghU4AIAAHhwAAAAojCBnzANBgkqhkiG9w0BAQ
EFAAOBjQAwgYkCgYEAtQsp9YgZ7AwZPakHJWKH1KpFI71Em1P4RgltE76gaCer39hJIIridFZ+aqRR8Rtv
fLvntPA/u05yx17VFjzdSE4WviuOqIExUsy7oTUOHI/E2ICV0ogDQJmDQaEQ/EKq1Y55VuD21lnG2UEvLd
TrgjzX9+IYCARyyf7FD8UG5JUCAwEAAXQABVguNTA5fnIAGWphdmEuc2VjdXJpdHkuS2V5UmVwJFR5cGUA
AAAAAAAAABIAAHhyAA5qYXZhLmxhbmcuRW51bQAAAAAAAAAAEgAAeHB0AAZQVUJMSUM=
                  </chavePublica>
            </ns1:iniciarComunicacao>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--------------------------------------
INFO: Outbound Message
--------------------------------------
Encoding: UTF-8
Content-Type: text/xml
Headers: {}
Payload:
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
      <soap:Body>
            <ns1:iniciarComunicacaoResponse>
                  <return>
PHJldG9ybm8+CjxrZXk+Q2ZqQ3V3RGVnL0xEbStjSE1DbEZ5d3RwTXJlQ2pCcndORVo1b3FML0JxWnAyL0
NRRnNLOTQvM0o0WExXV1cwL0Z4ZXdZNzN6VWZrcU02YXFITXA5UnkvS1dwbzZ5dFUyYy8vdnh5b3hTWlBs
Ui9VL2xKTFVkOWJDV0NKTndYMHM2R0JCR1NzZ0tuWGE0MEphTGtkOWFPRmhzU3BFdUIwaFIwSHFBMVlmQ0
hvPTwva2V5Pgo8Y29udGV1ZG8+TkFCU3RJQXRzZExGSzFYZjQxK2Z6d0EwdXludEczdWJuRFp2MHM5bVhi
TDZOWlZHaG1hSkVOanp1Mml2ODY5U1RKMURubFhsSGVSVGFVSkQ1TFN0RE1Pc2x1ajFDTDd5VGNWYWZJS0
dnY3l0cC8xaEEwUTFoYWwvdW5GQXRJamtza1lKVGo1cE8xOXJlM0pFc0hmWDJVSkVDNkNYZ252OXoxUzZv
a0dhOGZ1RXRhaklIWklUejI1NGpFUWp6TjErYzB2alVsVVBrSm9oamFqWFpqRUZLbEptczZrUzMvYytORy
tuQSt2WHluQ3FDQ2VSRjZJSU1icVZ1YzIwZHE1MlV3Uk5EejcrdlA4ZkcxeGNyOWJmdkdsUE9YTlVzUnR2
R1ZVT1F0Ri9HNDJoZ0NnMmZwWDc4eWhVNzMvcVlzSnNLTmRCRkVrZm5RdWtRZlo0aE5HcG9QcFcyMkxaVE
dQaHhiOWJuZTNKeEI3OHdKSm1CTWVYNEZnMkwyOUtaZ3dYN0ZEclNZeTZCNTRxNWk3R0VSQXU0MWZoNExy
ekFSb0JqbkJvSmx5UEwxb3RsdzZqd3NPYWJSWFMxeGNzcUN2am5vWEQvSW9ucE9NMWd4eVZGa1IxUTNrSU
JwanpieGtpcUE5OHVZVVZyUHViY1Ztc0FIVWtFeWNnQnliZWk4WmhNbW5tSUNKQVRTNEdVRkJIa0VSV2ps
STkvNkVucDR2citKYS8vMWplRzFnREdpOTZGT3RpQkxtQlUxQVROTzltekdkM2hkeGdrWGY1eGcwczNIck
hBc3lDQm5OZ2dCaXpQSjNFZSsxa0Fjc0xYY1lMS2NSekFXQk5IK0U0RTB0YXVUNjRPbEEwQlROamRDc1I3
N0E5Q2dWTWZxN3lBS2hNcGxpdi9nWU5HYnVZT251bDByR2RSNVpBSnRGZHhTWTVvOEY1Z1BlbUx6Q2lGMU
VZSFhmNTk2bHBFWU1WRmpzZWh1YWg4WTkxZWsyM3g5YndpaWJyQkppd0ROczVDTFhLWlg1VFJKS2g2a3hE
MHZaUjBxekZrNTdKRjdMb1BrWitna3lWa3ljdlVBKzNaMGFyZnNtcTVZR0lrSjVRZkJ1VXlSYjlOd0J1TE
tvVEVPMFdENmFBOU12bGk3bjA0SEhSUkR0SHA4d0U4ZEtjUEdDNkduV0pCZ0VCL0pSanVzTE1IU2tvczA3
K2Q5b3RkV1B2N2tSNVEyeHFmQUtNUEVZcmt0YTgyYkk4OFk2dlVFV1lqYkphT1NuRlNDRXpxQit6VnlkZn
FvTTBzQSsvSmlTQzc5ZTB2TVBsbUd2anUyK3BMTUdmSDdobC9aVjdWaS83TlJVZWI0TjJFMVF1OUIxeW56
NkJKeCtrVEtYNkxCNVMwTEFVTWJPa2JucGVqYmxxcVQ2UGRsY21xMUE9PC9jb250ZXVkbz4KPC9yZXRvcm
5vPgo=
                  </return>
            </ns1:iniciarComunicacaoResponse>
      </soap:Body>
</soap:Envelope>
```

A seguir é possível visualizar as mensagens trocadas referentes ao serviço “autenticarUsuario” utilizando a criptografia com a chave simétrica.

```
---------------------------
INFO: Inbound Message
---------------------------
ID: 11
Address: http://localhost:8080/TCC/acesso
Encoding: UTF-8
Http-Method: POST
Content-Type: text/xml; charset=utf-8
Headers: {Authorization=[Basic d2ViOndlYg==], connection=[Keep-Alive], Content-
Length=[859], content-type=[text/xml; charset=utf-8], host=[localhost:8080],
SOAPAction=[""], user-agent=[PHP-SOAP/5.4.12]}
Payload: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ns1="http://servico.sin.tcc.ufsc.br/">
      <SOAP-ENV:Body>
            <ns1:autenticarUsuario>
                  <parametros>
PHBhcmFtZXRyb3M+PHRpY2tldD5oQU14YW9JMWR6b0c3ZXQ0WnFaUWRNY0l3Z2ZhYVhEMXljbjNDSk0zOV
VVTXNSRm9MeDwvdGlja2V0PjxrZXk+Q0VWVUlXNXlwZzNUMzFHTnd3OW8xS2JDN281WXJNRjlBUC94V0JI
RWUwbys1ZDhuVytRcGxuTGNBMEpyV1I5elZJN3VPc1FNdHUzdmJDS2djU0JHMWJvUXU4SVM5SUZuZXNhNz
```

```
d1bmxTQXlVYXBJODBudUhkOEZHVVBVa0s2RWZrcEJLTVZpOUJqUkFVSVZsd0U1QkJOSXkrb2NzTFV6K3BU
YlFwenBDNjVFPTwva2V5Pjxjb250ZXVkbz5pL1MvZFlLME9nNjVvQ2FaRzhFVkpzaVB0VW5RVzlGWEdTOG
dzME1SbUpvUWVjSjV2dUJTYTlMeDZ1NlN1STdQcXFIcmI2cXlMRjdCUHJHNERFbER2eWcvU0MwSjAzQ29Y
ZVFlOWF1YnVuS3VIRFpkS2M5Q3pDUWpEbjJsbXBCUjwvY29udGV1ZG8+PC9wYXJhbWV0cm9zPg==
                  </parametros>
            </ns1:autenticarUsuario>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--------------------------------------
INFO: Outbound Message
--------------------------------------
ID: 11
Encoding: UTF-8
Content-Type: text/xml
Headers: {}
Payload:
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
            <ns1:autenticarUsuarioResponse>
                  <return>
tYS1p3aUhlMk9vUUtVWE1NbFlOdTlEdGRFeHY2Z1oyTEhUR21Cdmt4UUQzRk9IdldiWXR2V2s4WUlUTE9j
UFh2MmhlNUc4TWl0NkFzUlR2QXNsUXkvR24vUnVYTGRnbW5yUmtVKzVxWEhtMDc4Z3VoOXh2YXBSTXV3RU
xNPTwva2V5Pgo8Y29udGV1ZG8+djNnaFF0b05WOVYzTnpNb0xabFRUTXYwUTRNN05aakxtejR0aEdYMmk1
M1B2L3dCL1Bwam5JUjltMWR3NGYvYXdqYmVqQlVnTHlJVzM2eUg1WkZPTjFHcC9JK1M0dUo2ZXdLWW9QS1
RRZlFDRlRlVE5ucDU1VzBtK1orVmQ1aENFMFpHT1NMeVFxMHgzL25nNk12SVJQVVMzeXFqNVNidGNtSUZn
endycWkrSE82SlpSUXNMWVV4NGhHdXB3YWV5UHIrNHB6OVVlSmpMeHFwNkpJU3FwQy81NjRqK3U4RmNDYn
BpUWtwKzdQQnJZUEtLNHpjbjJYazltOFNiTUdVMTlKRUc4QmNpT1JqV2Z5K0FVdURGVUJ3OEMwZDZpRzhH
S1k5d2lJdXBaNVBKb2NmNHFCSTdGSElEaUx6cWkrbURGRGlMNld4QWxoUkgxNWNsbTRHU2dvUUFBZlp5Nj
k2d1ZSbm1rYkk2ZmI5S284d29WdTNPbGFuWmtRL3I4K1doMkdPaEE2enZ4Myt1SUFJNTZtYnI1ZmJmNFE1
NnpFVVhrZ053NWc5VjlWSU12d2NoYXNJUFMwL2g3dWhRZDdkNlpQWjBPMTEzKzcybjZBcEJPY1kyT1hMTV
lxRXdEZVFybGtPSHlxc0NIWHVNM3FtME1JdmFTTjN6elB6THVuK0tLU2FOQ3lQSENxclJuUEF1S00vQUFS
VVNtc1R6L2IwZ05odklZdFlVQ1dpdE1WZ2YwQVRpV0JZZlg2QWN5dWgzdU1qWUlPWEtDT29xUU5xcTNBek
83WDNWdXVHdGR0VjBKZGowK29UeHlHZDhHZDlxQXRxcFA5c2tJTUVVWW9aaUVGRzNwcFJhb1pxY1MwVHVN
NkZJSE1OejV1bzIvQVJlR0tEUzBqbTBtM1ZuSzNxQzBKZGgwL1A5WXFNQnkvemJuMjZYcTVoemtxTlhxWT
JUMWwzNk56L3lBc2E3eVVmVEh2NzZlSG55V2lGMVVnejNuYTFFait0cmg0bFcxbnFxTHI2Sk1ncGZmNWxZ
eTVnZWd2S3NIVFpPZ3gyRDFkVll5STBHZUY4by91TThSdXFrMm9XQnROTFBaRmIvOUs2NVBaZEhGaHNPYk
VONWZtalBoSWxLbmlidWhEL3pLZ1NNZFN3OUhwb09qT09mTWc2WHN3TUlsY0FmNGE4MStLNHd0aTJCUGgx
TEp6b0ZXRmRWLzZ0L1BLQTc1cHFaZHVLaU1OZzVqYUZRbXBQV3B6WXMzVFdDM2RQdE1lQk52TFg3RFRTTX
MyTFJBbDhIQ3paeGR3elBOeVI4VU83QVBycnhxdG85UXBCcXBrOWR3eWVhajE2RWs4RURiMWpURzE1QXE5
R0VWc21hUFdWVHZpT2MzSjJjY29Qa3VKTG9hNW5Ec25PSmpVV1FkOXZrZVAvd3A2UFA1MmN0YU81MGJEdV
FPUnFWc051QXVHZ01nNjMwTXFuRmgvQ1ZBTjZHc3BaV3hoaTlJRlNGN0lEb0JLb0hHbVZ0UVBEL2JwTXQw
MHVTMTc0T090WlpLcE5abndGLzhIcXJ2cFVlcGhna0JVa0Y5MGxxMXgwREVpVEgxMHVrN1NsQ1FJRjFPUW
42d0xwbGNaa3JuQzNBbjU4SUd2Y2Mrc0RGM2JQT3Z5ZHZEOEUvZnFSVkdTb08zNlkvV2VlSVk3OTF3TERl
dVRpQzk2amFRNmZBc2RxMXFGSVZXdEZuQ3dTam9MVVV1WFFVMWF2QjN5SGFPQkZ3LzFUVzc4VEovY3MzTl
puMEVXODd3YTRuNFZZYTYvTWZ6OUNEY012RXZRa0syd1V5LzZOVWpPUC9SOUNubVNhRmIxOEdyZFNXd3VL
bENtRHg1WEdZYVhhbFZGdkJJNFcxQXJ5Y3pmaEJWV2JnbGNsQ214U0NnVDVzZFZTQWpDVVZMcTZMRWpKY0
o3aThNZWdWRys0cStSRUtGamphYUQ3dnc2U0JiVlFqRHN0YnppUGFHZTg3a2o5bjZ0T3ZTWXdBSko4eGZS
WW9aQSs5UnhQRnZMT2F3V3VpemNKQVZHYzdDT0MzRHhqUzhyeDZseVBRNHRzUFh3aU83ZE83ejhhTEJwTV
B2SUVVdjQrV002SjU2b0dmYlpiRC8zd0FCZHFBNUpzNUpCYVVxK2NZcnJtdkxqcDlMcTFYSDdLNEk5MUY5
lTQmhCbmJORHhjNzI1Zk9GQVgycjB0R1ExTUI3TEVPdFZhanorMjJQdkh5NmU4dz09PC9jb250ZXVkbz4K
PC9yZXRvcm5vPgo=
                  </return>
            </ns1:autenticarUsuarioResponse>
      </soap:Body>
</soap:Envelope>
```

E nos trechos abaixo é possível visualizar as mensagens trocadas durante o acesso ao serviço “autenticarUsuario”, porém sem nenhuma criptografia, embora o servidor tenha sido implementado utilizando criptografia, foi realizado testes sem criptografia com o objetivo de comparação.

```
INFO: Inbound Message
----------------------------
ID: 12
Address:
http://localhost:8080/TCC/acesso
Encoding: UTF-8
Http-Method: POST
Content-Type: text/xml; charset=utf-8
Headers: {Authorization=[Basicd2ViOndlYg==], connection=[Keep-Alive],
Content-Length=[561],content-type=[text/xml; charset=utf-8],
host=[localhost:8080],SOAPAction=[""], user-agent=[PHP-SOAP/5.4.12]}
Payload: <?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP-ENV:Body>
            <ns1:autenticarUsuario>
                  <parametros>

      <ticket>EYF7sUAQlj2rygfr5c7NwgzDrnujCHCR4p3p1bZM0Yor6KtFGm</ticket>
                        <conteudo>
                              <autenticarUsuario>
                                    <email>admin@tcc.ufsc.br</email>
                                    <senha>12345</senha>
                              </autenticarUsuario>
                        </conteudo>
                  </parametros>
            </ns1:autenticarUsuario>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--------------------------------------
INFO: Outbound Message
--------------------------------------
ID: 12
Encoding: UTF-8
Content-Type: text/xml
Headers: {}
Payload:
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
            <ns1:autenticarUsuarioResponse>
                  <return>
                        <ns2:erros />
                        <ns2:chaveSimetrica>
rO0ABXNyABRqYXZhLnNlY3VyaXR5LktleVJlcL35T7OImqVDAgAETAAJYWxnb3JpdGhtdAASTGphdmEvbG
FuZy9TdHJpbmc7WwAHZW5jb2RlZHQAAltCTAAGZm9ybWF0cQB+AAFMAAR0eXBldAAbTGphdmEvc2VjdXJp
dHkvS2V5UmVwJFR5cGU7eHB0AANSU0F1cgACW0Ks8xf4BghU4AIAAHhwAAAAojCBnzANBgkqhkiG9w0BAQ
EFAAOBjQAwgYkCgYEAr0Pkh7oQqG/BgMWpmw8FrIfeXazSVhPjV0GmjjmHBt8dl4dVrUoNWbMOlXirSI/X
+3H2Vm8KtwvQRTE57/Um1MzvWAlXA5cvfFlhjY6/QRLwUInWPisxz9sdBMxwhortukQdkChvNBTnURrMl1
```

```
SBmwxgd3ITHNK0IKVocrpEAn8CAwEAAXQABVguNTA5fnIAGWphdmEuc2VjdXJpdHkuS2V5UmVwJFR5cGUA
AAAAAAAAABIAAHhyAA5qYXZhLmxhbmcuRW51bQAAAAAAAAAAEgAAeHB0AAZQVUJMSUM=
                        </ns2:chaveSimetrica>
                        <ns2:permissoes>
                              <ns3:PermissaoDto>
                                    <ns3:chave>MASTER</ns3:chave>
                                    <ns3:descricao>Acesso </ns3:descricao>
                              </ns3:PermissaoDto>
                              <ns3:PermissaoDto>
                                    <ns3:chave>TESTE</ns3:chave>
                                    <ns3:descricao> descricao</ns3:descricao>
                              </ns3:PermissaoDto>
                        </ns2:permissoes>
                        <ns2:servicos />
                        <ns2:ticket>
                              hAMxaoI1dzoG7et4ZqZQdMcIwgfaaXD1ycn3CJM39UUMsRFoLx
                        </ns2:ticket>
                        <ns2:usuario>
                              <ns3:cpf>99999999999</ns3:cpf>
                              <ns3:email>admin@tcc.ufsc.br</ns3:email>
                              <ns3:nome>Administrador Sistema</ns3:nome>
                        </ns2:usuario>
                  </return>
            </ns1:autenticarUsuarioResponse>
      </soap:Body>
</soap:Envelope>
-------------------------------------
```

A sessão iniciada no primeiro acesso será encerrada caso ocorra inatividade em tempo superior ao pré definido nas configurações do servidor, e caso isso ocorra o processo de troca de chaves deve ser iniciado novamente, pois a cada nova sessão uma nova chave simétrica será gerada. Como complementação a regra de confidencialidade implementada com a criptografia pela aplicação, pode-se utilizar o protocolo HTTPS que garante criptografia em todo o pacote trafegado, pois ela ocorre na camada de transporte.

Somente o uso da criptografia não garante a tríade da segurança, ela não garante a disponibilidade que é a propriedade que garante que a informação esteja sempre disponível para o uso legítimo, ou seja, por aqueles usuários autorizados.

Para solucionar esta questão de que somente aplicações e usuários autorizados possam acessar os serviços, a aplicação servidor realiza quatro validações, são elas:

1. A aplicação cliente é correta;
2. A aplicação cliente consome o serviço correto;
3. O usuário que está acessando o sistema é correto;
4. O usuário está acessando pela aplicação correta.

Seguindo este modelo, cada aplicação cliente deve ser cadastrada na base de dados, relacionadas com os serviços que terão acesso e receberão um usuário e senha de acesso que deve ser enviado a cada requisição dentro do cabeçalho da mensagem.

Já o usuário a ser cadastrado será associado a um ou mais perfis, e estes estarão associados a uma ou mais aplicações. De igual modo os usuários devem fornecer suas credenciais para poder acessar o serviço desejado, porém para que não fique trafegando pela rede as credenciais dos usuários, foi implementado o conceito de ticket, tornando necessário o envio das credencias somente uma única vez.

Ao enviar as credenciais o servidor retorna um código de acesso (ticket) que deve ser utilizado nas próximas requisições. Esses tickets são invalidados caso fiquem sem utilização em um intervalo de tempo que pode ser definido no arquivo de configuração.

Sendo as principais regras de segurança aplicadas no servidor, a aplicação cliente não necessita de regras rígidas. A principal é o controle de acesso onde cada

usuário deverá informar seu usuário e senha para acessar o sistema e sua autenticação ocorre através de serviço Web.

Outro controle realizado é fornecer ao usuário somente ações que ele possa realizar. Poderia deixar todas as ações liberadas, pois o servidor irá bloquear qualquer acesso indevido, mas isso poderia gerar um sentimento negativo no usuário, o que levaria a uma não aceitação na utilização do sistema.

Como a aplicação cliente será disponibilizada na Web foram tomadas as seguintes providências para que somente o diretório “public” possa ser acessado pelos navegadores. Estas providências consistem em configurar o servidor Apache para que ele aponte como diretório da aplicação o “public”, e que toda requisição seja redirecionada ao arquivo “index.php” do “public”.

Além disso, por questão de segurança foi acrescentado em todos os diretórios dois arquivos, exceto ao “public”. Um deles o “index.php” que realiza o redirecionamento caso alguém consiga acessar outro diretório, e o arquivo “.htaccess” que assegura que nenhum arquivo do diretório onde ele se encontra seja acessado via Web.

Outro ponto importante a se destacar é o sigilo das informações, como o cliente é Web o usuário acessa o sistema através da Internet, isso faz com que as informações trafeguem pela rede, e de nada adianta o servidor garantir sigilo se a aplicação cliente não o fizer. Para se resolver esta questão no cliente é utilizado o protocolo HTTPS.

7.2 Aplicação Servidor

O servidor foi desenvolvido utilizando a linguagem Java na versão 1.7 juntamente com o framework CXF na versão 2.7.6 que traz consigo o framework Spring na versão 3.0 e utilizando também o framework de persistência Hibernate na versão 3.5.6. Além destes foi utilizado o banco de dados PostgreSQL na versão 9.1.

Como protocolo de mensagem optou-se pelo SOAP, mas como visto anteriormente isso pode ser facilmente mudado através das configurações do CXF e o protocolo será HTTP.

7.2.1 Arquitetura do Servidor

Aqui segue a descrição de como será estruturado o servidor e como as camadas estarão distribuídas e como elas se relacionam.

7.2.1.1 Camada de serviço

Esta camada é responsável por disponibilizar e gerenciar os serviços Web que serão utilizados por outras aplicações.

As classes dessa camada que são mapeadas no arquivo de configuração do CXF, que será visto mais a frente, e que recebem as devidas anotações (anotações são fragmentos de código criados para descrever classes, campos e métodos) serão adicionadas no WSDL que é gerado automaticamente pelo framework CXF. Abaixo pode se observar um trecho de código com estas anotações iniciadas por “@”.

```
@Service(value = "acessoServico")
@WebService(serviceName = "AcessoService", targetNamespace = "http://servico.sin.tcc.ufsc.br/")
public interface AcessoServico extends GenericServico {
    AutenticacaoResponse autenticarUsuario(
        @WebParam(name = "email") @XmlElement(required = true) String email,
        @WebParam(name = "senha") @XmlElement(required = true) String senha
    );
}
```

A camada de serviço recebe os dados enviados por outra aplicação e os envia para a camada de negócio, que por sua vez os trata e retorna os dados que devem ser enviados a aplicação que requisitou o serviço.

As classes que compõem esta camada estão no pacote "servico" e são instanciadas pelo Spring através de seu arquivo de configuração.

7.2.1.2 Camada modelo

A camada modelo foi subdividida em outras duas: a primeira camada é a de negócio e a segunda de persistência.

A camada negócio trata as regras de controle da aplicação. Ela recebe as requisições vindas da camada de serviço, as trata realizando as devidas verificações. Ela é formada pelas classes contidas no pacote “negocio”, cada grupo de serviço possui uma dessas classes onde cada ação pertinente à classe é disponibilizada através de métodos. São instanciadas pelo Spring, para isso são

mapeadas no arquivo de configuração “spring-beans.xml”, onde podem receber parâmetros e ainda se pode configurar o controle das transações e até mesmo o *cache* da aplicação caso seja necessário. Também é responsável por fazer as requisições à camada de persistência, tratar seu retorno e repassá-los a camada de serviço.

A camada de persistência está dividida nos pacotes “dao” e “entidade”, sendo as classes do “dao” responsáveis pela comunicação com a base de dados e as classes do pacote “entidade” representam, por assim dizer, as tabelas da base de dados. Estes dois pacotes se comunicam entre si, e a comunicação, com a base de dados, nesta aplicação, se dá através do framework Hibernate. Este por sua vez mapeia as tabelas do banco de dados e as transforma em objetos do pacote “entidade” de acordo com as anotações utilizadas em cada classe, de modo que as consultas SQL básicas deixem de ser necessárias. Estas classes também são instanciadas pelo Spring.

7.2.2 Configuração do Servidor

A aplicação servidor possui cinco arquivos de configuração, serão descritos cada um deles, começando pelos arquivos “properties”, os quais possuem propriedades que serão utilizadas pelo sistema. As propriedades contidas nestes arquivos são aquelas que precisam ser alteradas de forma rápida e que podem variar de acordo com o ambiente em que a aplicação será instalada, como endereço do banco de dados, formato dos logs, entre outros. Estes arquivos podem ser alterados sem que necessite recompilar o código.

A estrutura destes arquivos pode ser visualizada abaixo:

```
# Endereço DB
dataSource.url=jdbc:postgresql://localhost:5432/tcc
# Usuario DB
dataSource.user=usuario
# Senha DB
dataSource.password=senha
```

Ao todo são três arquivos de propriedade, o primeiro é o “application.properties”, o qual possui parâmetros como tempo de sessão, e-mail do administrador do sistema, e algumas informações para o funcionamento do sistema.

O “hibernate.properties” contém os parâmetros necessários para a conexão com o bando de dados como *driver*, url, usuário e senha.

E por último temos o “log4j.properties” no qual encontram-se as configurações pertinentes aos *logs* do sistema.

As demais configurações estão em arquivos XML, estas configurações poderiam ser concentradas em um único arquivo, porém para melhorar a organização é preferível que as configurações fiquem separadas e agrupadas conforme seus objetivos.

Começando pelo “Web.xml”, este arquivo é padrão em aplicações Java para Web. Nele estão as configurações para que o sistema seja inicializado, e se fosse para concentrar todas as configurações em um único arquivo seria neste. Os arquivos que possuem outras configurações devem ser referenciados aqui.

O próximo arquivo é o “spring-beans.xml” nele estão as configurações para o framework Spring. Neste arquivo estão mapeadas todas as classes às quais se

deseja que fiquem sob o controle do Spring. As classes de serviço, negócio, DAO, e demais classes necessárias para o funcionamento da aplicação.

O Spring não obriga que todas as classes que ele vai gerenciar sejam declaradas nas suas configurações. Ele permite que sejam usadas anotações dentro do código Java para que o próprio Spring possa identificar durante a inicialização da aplicação quais as classes deve gerenciar.

Abaixo segue o trecho do código que diz para o framework, quais pacotes ele deve verificar a existência das anotações e também trecho do código Java com estas anotações.

```
<context:component-scan
        base-package="br.ufsc.tcc.sin.servico.impl,
            br.ufsc.tcc.sin.negocio,
           br.ufsc.tcc.sin.modelo.dao"
/>
```

Configuração do Spring

```
package br.ufsc.tcc.sin.negocio;

import org.springFramework.beans.factory.annotation.Autowired;
import org.springFramework.stereotype.Component;

@Component
public class AcessoNegocio extends GenericNegocio {

    @Autowired
    private UsuarioDao usuarioDao;
```

Código Java

Através da configuração o Spring obtém os pacotes que devem ser verificados, que no exemplo acima é o pacote "negocio", e na classe podemos observar o uso das anotações @Componet e @Autowire. A primeira diz ao Spring que a classe deve ser registrada como um *bean* para que possa ser utilizada por outras classes e a segunda diz ao framework que deve injetar o *bean*, no caso “usuarioDao”.

E por último, mas não menos importante, que é a configuração do CXF no arquivo “cxf-servicos.xml”. Este arquivo registrará os serviços que serão disponibilizados, qual protocolo a ser utilizado, se terá ou não alguma regra de segurança entre outras configurações pertinentes a visão do Web Service pelas aplicações que o iram consumir.

Lembrando que os *beans* referenciados na configuração do CXF devem estar registrados na configuração do Spring. Abaixo pode se observar um trecho desta configuração.

```
<!-- Endpoint WSDL para o Apache CXF -->
<!-- Dizemos o endereço, o ID do serviço, em qual bean ele depende -->

    <jaxws:endpoint id="acesso"
      address="/acesso"
          implementorClass="br.ufsc.tcc.sin.servico.impl.AcessoServicoImpl"
          implementor="#acessoServico"  >
          <jaxws:serviceFactory>
                <ref bean="jaxws-and-aegis-service-factory" />
          </jaxws:serviceFactory>
```

```
                    <jaxws:features>
                <bean class="org.apache.cxf.feature.LoggingFeature" />
            </jaxws:features>
                </jaxws:endpoint>
```

Estes arquivos “spring-beans.xml” e “cxf-servicos.xml” podem ser visualizados integralmente nos Anexo V e Anexo VI respectivamente.

7.2.3 Desenvolvimento

Nesta seção será descrito o desenvolvimento do sistema no quesito servidor, essa descrição será dividida em camadas a fim de facilitar o entendimento.

7.2.3.1 Camada de serviços

Nas classes desta camada estão contidos os métodos que se tornarão serviços através das anotações neles descritas e das configurações do CXF, abaixo pode se observar trecho de código que contém as anotações necessárias para disponibilizar o serviço.

```
    @Path("/") // caminho onde o serviço REST fica disponibilizado, seguido pelo @Path de cada parametro
    @Produces({ "application/xml", "application/json" }) // tipos de retorno que o nosso REST pode produzir
    @WebService(targetNamespace = "http://servico.sin.tcc.ufsc.br/") // definimos aqui que essa interface é um WebService WSDL
```

```
public interface AcessoServico extends GenericServico {


        @GET // tipo do metodo REST
        @Path("/autenticar/{email}/{senha}") // caminho do método
        @WebMethod (operationName = "autenticarUsuario")// nome do metodo no WSDL
        AutenticacaoResponse autenticarUsuario(
          @PathParam("email") //Nome do parametro no Rest
          @WebParam(name = "email") //Nome do parametro no WSDL
          @XmlElement(required = true) //Indica que o parametro é obrigatorio
                String email,
          @PathParam("senha") @WebParam(name = "senha")
      @XmlElement(required = true) String senha);


}
```

É também nesta camada que se realiza parte do controle de acesso aos serviços. Aqui se verifica se a aplicação cliente possui permissão de acesso para serviço requisitado. Todos os serviços chamam o método *permissaoDeAcesso()* da super classe para verificar essa permissão. Este método retorna o valor booleano *true* caso possua permissão ou *false* para não, e o serviço que o chamou trata este valor para poder retornar a quem o requisitou para que este possa tomar as devidas providências.

Todos os serviços retornam objetos do tipo DTO que possuem somente os atributos necessários para suprir a demanda do serviço. Esse atributos são de tipo primitivo como *string, inteiro, double, booleano* e *array*. O próprio CXF se responsabiliza em especificar para as aplicações clientes cada tipo do objeto

retornado através do WSDL que é gerado em tempo de execução. Estas classes DTO não precisam de nenhuma configuração, em específico, salvo quando se deseja que o serviço também seja disponibilizado através do protocolo REST. Para isto basta colocar a anotação @XmlRootElement antes de se declarar a classe como mostra o código a baixo.

```
@XmlRootElement
public class AutenticacaoResponse extends WSResponse implements Serializable {
```

Para melhorar a organização foram divididos os serviços em quatro grupos:

- Acesso
- Administração
- Formulário
- Prontuário

Cada grupo possui um WSDL próprio e serviços específicos, salvo o serviço "testeServico" que consiste em um serviço de verificação para que as aplicações clientes possam testar sua comunicação com o servidor, este serviço está implementado na classe que todas as classes deste pacote herdam. No Anexo VII pode se observar a tabela com todos os serviços disponibilizados por cada grupo.

A seguir será detalhado cada grupo e seus serviços principais.

Grupo Acesso

Os serviços deste grupo são referentes às regras de acesso da aplicação com destaque para o serviço "*autenticarUsuario"* que é o responsável pela validação do usuário. Este serviço retorna as informações do usuário como nome, CPF, suas

permissões e serviços que podem ser acessados conforme a aplicação que solicitou sua autenticação. Abaixo, na Figura 16, observa-se a representação gráfica do WSDL deste grupo.

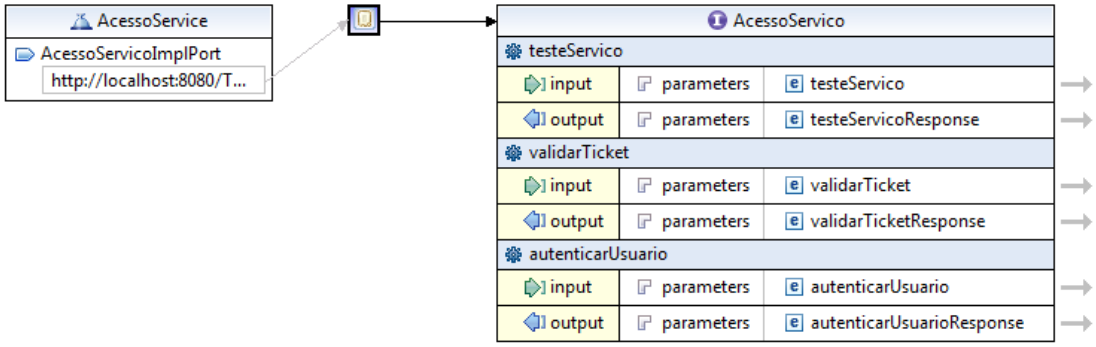


Figura 16 - Diagrama WSDL Acesso

Na Figura 17 pode se ver a assinatura deste serviço.

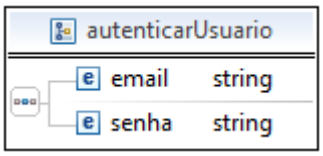


Figura 17 - Serviço autenticarUsuario

Grupo Administração

Neste grupo estão todas as tarefas de gestão da aplicação, como cadastro de usuário, cadastros das informações de alguns formulários, como avaliações, diagnósticos e intervenções.

Grupo Formulário

Este grupo contém os serviços pertinentes à construção de formulário no cliente, é ele que vai fornecer as descrições e valores que os formulários devem

assumir. Em sua maioria eles retornam uma lista onde cada elemento tem um identificador e uma descrição.

Neste grupo destaca-se o serviço “*ultimasAlteracoes”* onde é retornada uma lista com o nome do serviço de formulário e a data da última alteração de seus dados. Este serviço foi criado pensando nos formulários que sofreram poucas alterações, e sendo assim a aplicação cliente pode fazer seu controle, caso opte poderá armazenar em *cache* estas informação e somente acessar o serviço específico do formulário caso ocorra alguma alteração.

Grupo Prontuário

Neste grupo estão todos os serviços responsáveis pela manutenção dos dados do paciente junto ao sistema.

7.2.3.2 Camada de negócio

Nesta camada estão implementadas as regras de negócio da aplicação, a camada de serviço recebe as requisições e as repassa a esta camada que por sua vez faz as devidas verificações e provê interfaces de acesso à camada de persistência, permitindo o reuso do código e a implementação de recursividade assim propiciando um baixo acoplamento da aplicação.

As classes desta camada recebem a anotação @Component que informa ao Spring no momento da inicialização da aplicação que esta deve vir sob seu controle e para instanciar objetos que estão sob controle do Spring seja por anotação ou parâmetro de configuração, utiliza-se a anotação @Autowired caso não a usasse

teria que declarar no arquivo de configuração ”spring-beans.xml" e declarar os métodos de *set* e *get* para cada atributo, o que com a anotação não é obrigatória.

A maioria das classes de negócio possuem objetos que são responsáveis por acessar a base de dados, pois as classes de negócio não os acessam diretamente.

Abaixo temos uma classe de negócio:

```
package br.ufsc.tcc.sin.negocio;

import java.util.List;

import org.springFramework.beans.factory.annotation.Autowired;
import org.springFramework.stereotype.Component;

import br.ufsc.tcc.sin.exception.DaoException;
import br.ufsc.tcc.sin.exception.NegocioException;
import br.ufsc.tcc.sin.modelo.dao.AplicacaoDao;
import br.ufsc.tcc.sin.modelo.dao.PermissaoDao;
import br.ufsc.tcc.sin.modelo.dao.UsuarioDao;
import br.ufsc.tcc.sin.modelo.entidade.acesso.Aplicacao;
import br.ufsc.tcc.sin.modelo.entidade.acesso.Permissao;
import br.ufsc.tcc.sin.modelo.entidade.acesso.Usuario;

@Component
public class AcessoNegocio extends GenericNegocio {

	@Autowired
	private UsuarioDao usuarioDao;

	@Autowired
	private PermissaoDao permissaoDao;

	@Autowired
	private AplicacaoDao aplicacaoDao;

	public Usuario autenticarUsuario(String email, String senha)
			throws NegocioException {
		if((email != null && !email.isEmpty()) && (senha != null && !senha.isEmpty())){
			try {
				return usuarioDao.findByEmailSenha(email, senha);
			} catch (DaoException e) {
				e.printStackTrace();
				throw new NegocioException(e.getMessage());
			}
		}else{
			throw new NegocioException("E-mail e senha não foram informados");
		}

	}
```

```
		public List<Permissao> buscarPermissoesUsuario(String email)
				throws NegocioException {
			try {
				return permissaoDao.findByEmail(email);
			} catch (Exception e) {
				e.printStackTrace();
				throw new NegocioException(e.getMessage());
			}
		}

		public boolean aplicacaoPodeAcessarServico(String identificador, String
chaveAplicacao, String chaveServico) {
			try {
				Aplicacao aplicacao =
aplicacaoDao.findByChaveServico(identificador, chaveAplicacao, chaveServico);
				if(aplicacao == null){
					return false;
				}else{
					return true;
				}
			} catch (Exception e) {
				e.printStackTrace();
				return false;
			}
		}

	}
```

No caso da classe exemplificada acima ela possui três objetos que acessam a base de dados. Estes objetos são do pacote *dao* da camada de persistência que vem a seguir.

7.2.3.3 Camada de persistência

Esta camada possui dois pacotes de classes, “dao” e “entidade”.

As classes do pacote “dao”, chamadas de DAOs, recebem a anotação *@Repository* que desempenha a mesma função da anotação utilizada nas classes de negócio. Estas implementam os métodos de acesso ao banco de dados, cada uma destas classes faz referência a uma classe do pacote entidade, apenas estas devem acessar a base de dados. Sendo assim para se executar qualquer alteração

seja um *insert* ou *update* ou somente consultar dados através de um *select*, obrigatoriamente a requisição tem que passar por esta classe.

Todas as classes DAO independente da entidade que elas correspondem estendem a classe "*GenericDaoHibernate"* que é genérica, esta classe implementa as funções básicas de acesso à base de dados. Conforme se vê na Figura 18 os métodos e atributos desta classe.

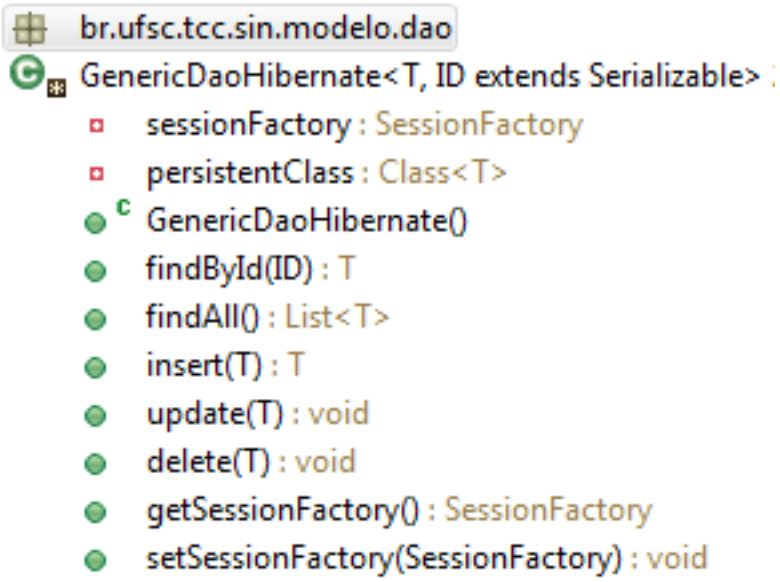


Figura 18 - Métodos classe genérica

Como todas as DAOs estendem a classe genérica algumas delas não possuem nenhum método a não ser os da superclasse, como é o caso da classe "*SistemaDao"* como pode ser observado no código abaixo. Já outras podem possuir vários métodos.

```
package br.ufsc.tcc.sin.modelo.dao;

import org.spring*Framework*.stereotype.Repository;

import br.ufsc.tcc.sin.modelo.entidade.Sistema;

@Repository
public class SistemaDao extends GenericDaoHibernate<Sistema, Integer> {

}
```

As classes do pacote entidade possuem as informações para que o framework Hibernate possa mapeá-las, estas informações são passadas através de anotações. Nestas classes também é permitido declarar através de anotações as consultas que serão utilizadas na classe DAO. Estas consultas podem seguir o padrão do Hibernate ou os padrões do SQL. Abaixo se pode observar uma destas classes, os métodos get e set foram suprimidos assim como o construtor, pois seguem os padrões da linguagem.

```
package br.ufsc.tcc.sin.modelo.entidade;

import java.io.Serializable;
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Sistema implements Serializable {
        private static final long serialVersionUID = 1L;
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String descricao;

    @Column(name = "ordem", nullable = true)
    private Short ordem;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "sistema", fetch =
FetchType.LAZY)
    private List<FormDiagnostico> formDiagnosticoList;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "sistema", fetch =
FetchType.LAZY)
    private List<CasoClinico> casoClinicoList;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "sistema", fetch =
FetchType.LAZY)
    private List<FormAvaliacao> formAvaliacaoList;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "sistema", fetch =
FetchType.LAZY)
    private List<FormIntervencao> formIntervencaoList;
```

Lembrando que as entidades devem ser configuradas no arquivo “spring-beans.xml”, para que o Hibernate possa saber quais classes devem ser mapeadas. Abaixo tem um trecho desta configuração.

```
<property name="annotatedClasses">
    <list>
        <value>br.ufsc.tcc.sin.modelo.entidade.Alerta</value>
        <value>br.ufsc.tcc.sin.modelo.entidade.Avaliacao</value>
        <value>br.ufsc.tcc.sin.modelo.entidade.CasoClinico</value>
```

7.3 Aplicação Cliente Web

Para o cliente Web será utilizada a linguagem PHP na versão 5.4. Contando com o suporte do Smarty, na versão 3.1.14, que permite separar o layout do código PHP. Assim como o BootStrap na versão 3.0.0 que fornece ferramentas para a criação dos layouts. O BootStrap funciona juntamente como jQuery que é um framework em JavaScript (é uma linguagem de programação interpretada pelos navegadores Web cliente).

Similar à aplicação servidor, o cliente Web foi separado em camadas a fim melhor estruturar o código fonte e facilitar futuras manutenções, sejam corretivas ou evolutivas.

7.3.1 Arquitetura Cliente Web

Será descrita a arquitetura implantada no cliente Web. Esta arquitetura é similar à implantada no servidor.

7.3.1.1 Camada de visualização

A camada de visualização é composta por arquivos PHP que fazem as chamadas a camada de negócio de acordo com os parâmetros que recebe via requisições Web, os arquivos do Smarty que possuem extensão .tpl, que por sua vez utilizam arquivos javascript e CSS para formar as páginas Web que serão mostradas ao usuário através do navegador.

O Smarty possui *tags* próprias que facilitam a escrita das páginas Web nos arquivos .tpl, mas caso o desenvolvedor prefira poderá utilizar somente *tags* HTML.

Os arquivos .tpl são lidos pelo PHP, que por sua vez retorna código HTML para que possa ser interpretado pelos navegadores. Abaixo segue exemplo de um trecho de código HTML puro.

```
<select name=paciente_id>
        <option value="1">Paciente Um</option>
        <option value="2" selected="selected">Paciente Dois</option>
        <option value="3">Paciente Três</option>
        <option value="4">Paciente Quatro</option>
</select>
```

A seguir o mesmo resultado, mas utilizando a *tag* do Smarty.

```
<select name=paciente_id>
        {html_options options=$pacientes selected=$paciente_id}
</select>
```

Além das *tags* que o Smarty possui em sua distribuição ele permite criar novas caso seja necessário.

7.3.1.2 Camada modelo

A camada modelo está dividida em duas: a primeira camada é a de negócio que desempenha a mesma função da camada de mesmo nome na aplicação servidor e a segunda que aqui será chamada de serviço.

A camada negócio trata as regras de controle da aplicação cliente, ela recebe as requisições vindas da camada de visualização e ela as trata quando necessário.

Ela é formada pelas classes contidas no diretório *negocio*, cada arquivo PHP da camada de visualização corresponde a uma dessas classes onde cada ação pertinente a classe é disponibilizada através de métodos. Além de ser responsável por fazer as requisições à camada de serviço, tratar seu retorno e repassá-los a camada de visualização.

A camada de serviço é responsável pela comunicação com o servidor de serviços. Ao receber os dados do servidor ela os mapeia em objetos conhecidos pela aplicação cliente e os repassa a camada de negócio.

7.3.2 Configuração Cliente Web

A aplicação cliente Web possui quatro arquivos de configuração que podem ser divididos em dois grupos. Primeiro os arquivos de propriedades onde estão os atributos que possuem valores fixos e de pouca atualização, como endereço do servidor, título da aplicação, entre outros. Ele possui dois arquivos, o “config.ini” e “site_Smarty.conf”.

O arquivo do tipo .ini é o padrão do PHP para armazenar configurações, e a linguagem possui métodos específicos para leitura destes arquivos. Em um mesmo arquivo podemos agrupar as configurações para melhor organização. Abaixo podemos observar trecho deste arquivo.

```
[path]
page_dir = ../page/
templates_dir = ../templates/
```

```
[servico]
wsdl.acesso = http://localhost:8080/TCC/acesso?wsdl
wsdl.formulario = http://localhost:8080/TCC/formulario?wsdl
wsdl.prontuario = http://localhost:8080/TCC/prontuario?wsdl
wsdl.administracao = http://localhost:8080/TCC/administracao?wsdl
```

O outro arquivo deste grupo é exclusivo do Smarty, suas informações serão utilizadas dentro dos arquivos .tpl. A seguir tem um exemplo de sua estrutura.

```
title = PROCESSO DE ENFERMAGEM INFORMATIZADO
autor = Marcelo C. de Souza e Aline M. F. de Souza
```

Outro grupo de arquivos contém trechos de código PHP, que são responsáveis por indicar a aplicação quais diretórios estão as classes necessárias para o perfeito funcionamento do sistema e por instanciar objetos que serão utilizados em todo o contexto da aplicação. Os arquivos que compõe este grupo são “autoload.php” e o “BootStrap.php”, eles poderiam estar em um único arquivo mas mantê-los separados melhora a organização. O primeiro arquivo é quem informa ao sistema os diretórios. Abaixo um Trecho pode ser visto.

```
<?php
set_include_path(get_include_path() . PATH_SEPARATOR . APPLICATION_PATH . 'libs'.
DIRECTORY_SEPARATOR);
set_include_path(get_include_path() . PATH_SEPARATOR . APPLICATION_PATH . 'src' .
DIRECTORY_SEPARATOR . 'negocio' . DIRECTORY_SEPARATOR);
set_include_path(get_include_path() . PATH_SEPARATOR . APPLICATION_PATH . 'src' .
DIRECTORY_SEPARATOR . 'criteria' . DIRECTORY_SEPARATOR);
```

O segundo é responsável por instanciar os objetos que serão utilizados pela aplicação. Este pode ser comparado ao arquivo “spring-beans.xml” do servidor. A seguir tem o conteúdo deste arquivo.

```
<?php

require_once 'Smarty.class.php';

require_once 'Zend/Loader/Autoloader.php';

$autoloader = Zend_Loader_Autoloader::getInstance();
$autoloader->setFallbackAutoloader(true);

$Smarty = new Smarty();
$Smarty->debugging = false;
$Smarty->caching = false;
$Smarty->cache_lifetime = 120;

$Smarty->setConfigDir(APPLICATION_PATH . 'config' . DS);
$Smarty->setTemplateDir(APPLICATION_PATH . 'templates' . DS);
$Smarty->setCompileDir(APPLICATION_PATH . 'templates_c' . DS);
$Smarty->assign('subMenus', null);

$Smarty->addPluginsDir(APPLICATION_PATH . 'src' . DS . 'pluginsSmarty');
```

7.3.3 Desenvolvimento do Cliente Web

Esta seção descreverá o desenvolvimento do cliente Web, de igual modo à seção desenvolvimento referente à aplicação servidor essa descrição será dividida em camadas a fim de facilitar o entendimento.

7.3.3.1 Camada de visualização

As requisições que são realizadas a aplicação devem ocorrer somente pelo arquivo “index.php” contido no diretório “public”. Este diretório e todo seu conteúdo

ficarão visíveis na Web, sendo assim ele possui somente arquivos que são interpretados pelos navegadores como código css, javascript e imagens. Na Figura 19 se observa a estrutura de diretórios.

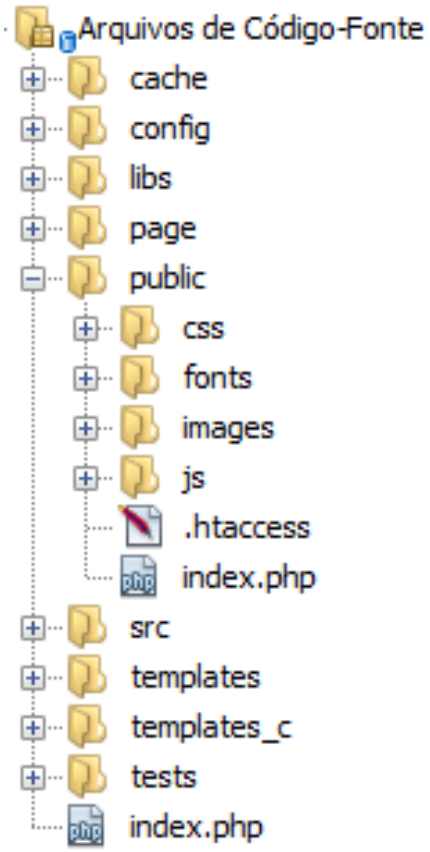


Figura 19 - Estrutura de diretórios do cliente Web

Também fazem parte desta camada os arquivos contidos no diretório “pages”. O arquivo “index.php” do diretório “public” recebe as requisições Web e as URL's que estão sendo acessadas. Estas URL's seguem o seguinte padrão: domínio da aplicação / página / ação / identificador. Qualquer outro padrão informado não é aceito pela aplicação e a mesma retorna uma página de erro. Somente o domínio da aplicação é obrigatório, e caso não seja informado nenhum outro parâmetro, o sistema exibirá a página padrão definida no arquivo de configuração.

Cada página tem um arquivo correspondente, este arquivo é responsável por fazer a chamada à camada de negócio, e cada arquivo referenciasse somente a uma classe de negócio. Abaixo podemos verificar o conteúdo de um destes arquivos.

```php
<?php

$prontuarioNegocio = new ProntuarioNegocio();

if(isset($_GET['acao']) && method_exists($prontuarioNegocio, $_GET['acao'])){
    if(isset($_GET['id'])){
        $prontuarioNegocio->$_GET['acao']($_GET['id']);
    }else{
        $prontuarioNegocio->$_GET['acao']();
    }
}else{
    $prontuarioNegocio->padrao();
}
```

É nesta camada que se utiliza o Framework BootStrap e o Smarty, sendo o primeiro utilizado na construção dos modelos das telas do sistema e o segundo e responsável pela junção dos dados vindos da camada de negocio com o modelo para se gerar a tela que será exibida. Os arquivos de *templates* ficam separados do código PHP em um diretório do mesmo nome, sendo acessados pela instância do Smarty que foi inicializada no arquivo “BootStrap.php”.

A utilização do framework BootStrap nos fornece muitos recursos para a criação das telas, destacando-se a capacidade de reorganizar os elementos da tela de acordo com as dimensões do navegador que está acessando a aplicação. Fica visível este recurso através das Figura 20 e Figura 21, onde a primeira mostra o acesso através de um computador pessoal e a segunda o acesso através de um *smartphone*.

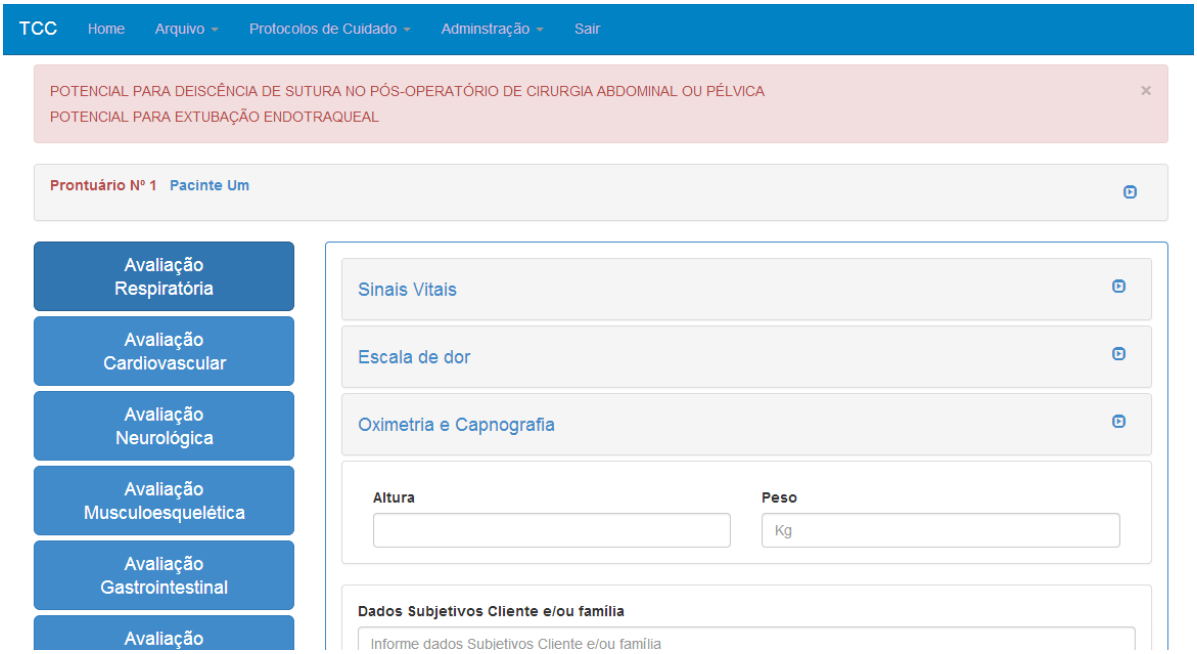


Figura 20 - Acesso via desktop

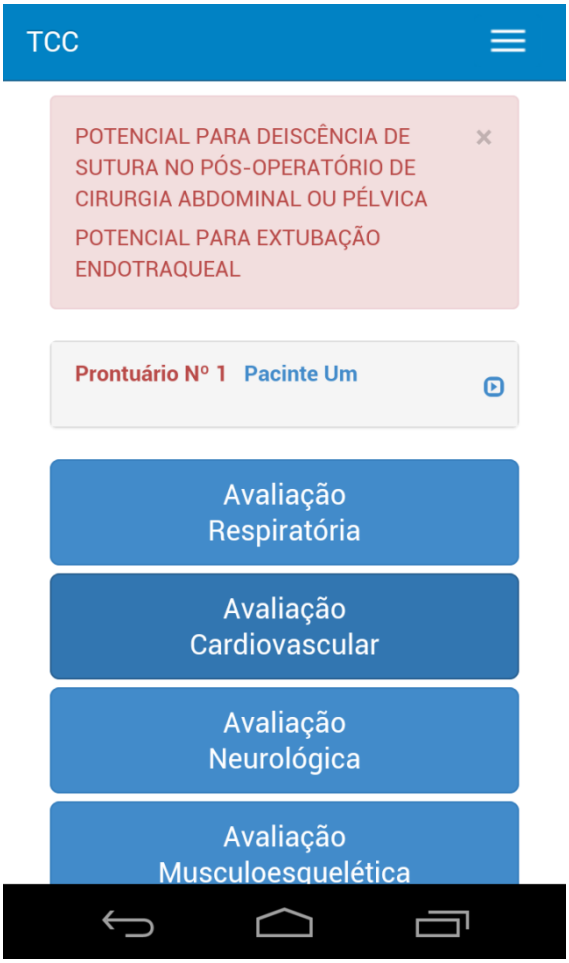


Figura 21 - Acesso via Smartphone

7.3.3.2 Camada de negócio

Nesta camada serão tratadas as requisições vindas da camada de visualização. Fazem parte desta camada as classes contidas no diretório "negocio". Esta camada faz requisições à camada de serviço, e com o retorno que recebem repassam para o Smarty. A seguir um trecho do código quando cliente acessa a URL "/prontuario/acessar/1" onde "prontuario" é a página que será chamada, "acessar" corresponde ao método acessar e "1" é o código do prontuário que se quer acessar.

```
<?php

class ProntuarioNegocio {

    public function acessar($codigo) {
        $this->acessarProntuario($codigo);
        Utils::getSmarty()->assign('erros', $this->errors);
        Utils::getSmarty()->assign('template', $this->template);
    }

    public function acessarProntuario($codigo) {

        $retorno = $this->servico->acessarProntuario($codigo);
        if ($retorno->getErros() != null && count($retorno->getErros()) > 0) {
            foreach ($retorno->getErros() as $erro) {
                $this->errors[] = new Error($erro->getCod(), $erro->getMsg());
            }
```

```
            $this->buscarInternados();
        } else {
            Utils::getSmarty()->assign('cliente', $retorno->getProntuario());
            $this->template = Utils::getTemplate('prontuario/index');
        }
    }
```

Através do código "$retorno = $this->servico->acessarProntuario($codigo);" a camada negócio faz a chamada à camada de serviço e pelo código "Utils::*getSmarty*()->assign('clientes', $retorno->getProntuario());" repassa ao Smarty os dados do prontuário desejado.

7.3.3.3 Camada de serviço

As classes desta camada estão no diretório “ws” onde cada grupo dos serviços disponibilizados pelo servidor possuem uma classe correspondente e todas *extendem* a classe “*GenericServico”* que é responsável pela implementação da comunicação com o servidor e pelo mapeamento do retorno enviado pelo servidor em objetos conhecidos pela aplicação.

Nestas classes somente serão implementadas as chamadas específicas de cada serviço. Com isso se for preciso alterar a forma como os serviços são acessados ou a forma de acesso ao serviço, isso é realizado somente na superclasse. Para que se possa acessar o Web Service utilizando o protocolo SOAP o suporte ao mesmo deve estar habilitado nas configurações do PHP.

7.4 Aplicação Cliente Móvel

A aplicação móvel será desenvolvida para a plataforma Android. Esta plataforma foi escolhida devido à familiaridade com a linguagem Java a que ela tem suporte, e por possuir ferramentas de desenvolvimento *open source* e independente do sistema operacional diferentemente de outras plataformas móveis.

Tendo a aplicação móvel como principal objetivo comprovar o conceito do Web Service, que é a interoperabilidade entre diferentes plataformas, ela é mais simples em relação ao cliente Web no que diz respeito ao consumo dos serviços e das funcionalidades do sistema original. Tem como principal funcionalidade visualizar os dados dos sinais vitais coletados e de possíveis alertas que o sistema venha a emitir.

Ao iniciar a aplicação é verificado se existe armazenado o *login* e senha do usuário e caso positivo o usuário poderá visualizar a relação de pacientes internados e se possui algum alerta, como pode ser observado na Figura 22.

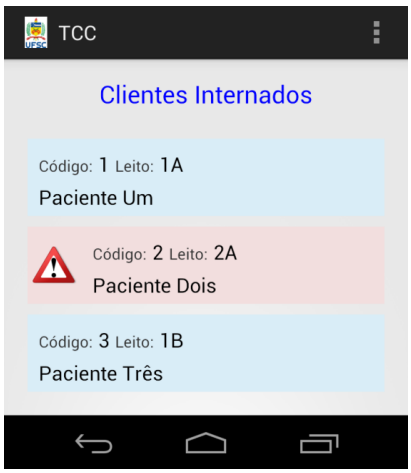


Figura 22 - Tela de listagem de pacientes

Caso não seja encontrado o sistema exibirá a tela de *login* solicitando e-mail e senha do usuário. Abaixo, na Figura 23 se verifica esta tela, inclusive com uma validação caso o e-mail seja inválido.

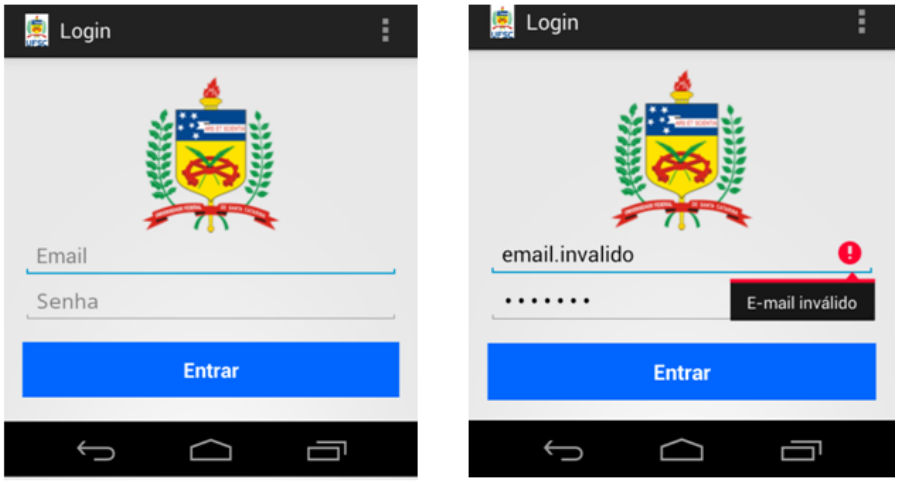


Figura 23 - Tela de login

Na tela de listagem dos pacientes é possível acessar as informações do mesmo no que diz respeito aos sinais vitais, e caso o paciente possua algum alerta este será exibido, conforme mostra a Figura 24.

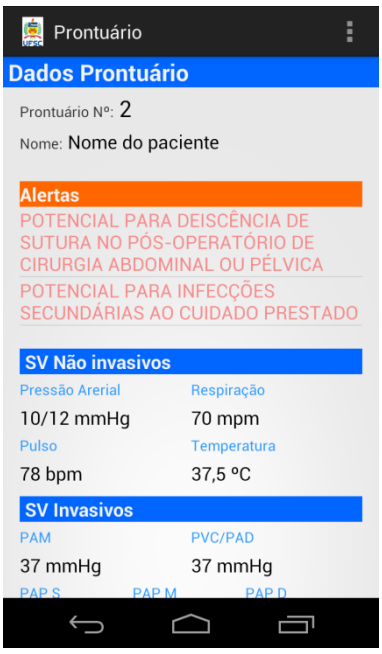


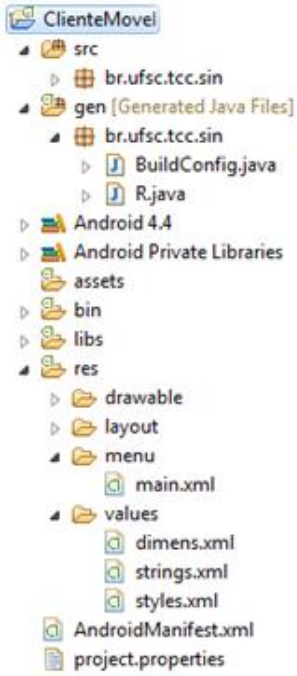
Figura 24 - Tela dos dados do paciente

O aplicativo também emite notificações ao usuário quando algum paciente entra em alerta. Esta notificação é emitida mesmo que a aplicação não esteja em execução. Na Figura 25 é possível observar uma destas notificações recebidas, basta o usuário clicar sobre ela, para visualizar a tela com os detalhes do paciente.

Figura 25 - Tela de notificação

7.4.1 Arquitetura Cliente Móvel

A plataforma Android traz consigo a obrigatoriedade de se utilizar uma estrutura básica de diretórios, onde cada diretório é responsável por uma camada do sistema, conforme representado na Figura 26.



Pasta	Descrição
src	Pasta que contém as classes Java do projeto.
gen	Contém a classe R.java que é gerada automaticamente e permite que a aplicação acesse qualquer recurso como arquivos e imagens utilizando as constantes dessa classe. Esta classe nunca deve ser alterada manualmente. Possui também a classe BuildConfig.java que define uma constante DEBUG, que é gerada automaticamente, conforme especificada sua aplicação.
assets	Contém arquivos opcionais ao projeto, como por exemplo, uma fonte customizada.
res	Pasta que contém os recursos da aplicação como imagens, layouts de telas e arquivos de internacionalização. Tem três subpastas: *drawable, layout* e *values*
drawable	Pasta com as imagens da aplicação. Nessa pasta é possível customizar as imagens para a resolução do dispositivo apresentado e por isso existem 4 (quatro) pastas para imagens: drawable-ldpi, drawable-mdpi, drawable-hdpi e drawable-xhdpi
layout	Contém os arquivos XML de layouts para construir as telas das aplicações.
values	Contém os arquivos XML utilizados para a internacionalização da aplicação e configurações.

Figura 26 - Estrutura diretórios Android (PAVEI, 2012)

7.4.1.1 Camada de visualização

Compõe a camada de visualização as classes que extende as *Activities, View* e os arquivos XML utilizados para elaborar as tela. As *Activities* são responsáveis por capturar as ações do usuário. As *Views* são todos os componentes utilizados na formação da tela. Os arquivos XML definem a disposição destes componentes.

As informações que serão fornecidas nas telas podem ser definidas diretamente nos arquivos XML ou através das classes Java como mostrado na Figura 27.

Código Java

```
TextView text = new TextView(this);
text.setId(123456);
text.setWidth(LayoutParams.FILL_PARENT);
text.setHeight(LayoutParams.FILL_PARENT);
text.setText(R.string.hello);
```

Arquivo XML

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android=
"http://schemas.android.com/apk/res/android"
android:id="@+id/texto"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello" />
```

Figura 27 - Comparativo entre classe Java é arquivo XML

7.4.1.2 Camada modelo

Similar ao cliente Web esta camada está dividida em negócio e serviço, e são organizadas em pacotes de mesmo nome. Aqui serão destacadas as classes da camada de serviço, pois devido a exigência da plataforma todas devem extender a classe *"AsyncTask"* que implementa *thread*. Essa obrigatoriedade se deve ao fato desta camada ser responsável pela comunicação com o servidor, e para que a aplicação não fique parada esperando uma resposta esta deve ser executada em paralelo ao resto da aplicação.

7.4.2 Configuração Cliente Móvel

Similar ao servidor as aplicações desenvolvidas para o Android utilizam arquivo "properties" para armazenar informações pertinentes às configurações da aplicação. Além deste arquivo a aplicação exige a existência do arquivo "*AndroidManifest.xml".* Nele serão definidos o nome da aplicação, a versão mínima do Android, quais recursos a aplicação vai utilizar além de descrever todos os componentes que compõe a aplicação.

7.4.3 Desenvolvimento do Cliente Móvel

No desenvolvimento do cliente móvel destaca-se a utilização da biblioteca kSOAP para prover comunicação com o servidor, poderia se utilizar outras para essa finalidade, como por exemplo, o Axis. Mas em se tratando de dispositivos móveis devemos sempre pensar em otimização, e utilizando o kSOAP foi alcançad este objetivo, por ele ser uma biblioteca que consome poucos recursos. Em contra partida exige um pouco mais de esforço por parte do desenvolvedor, pois como retorno do consumo dos serviços a biblioteca fornece somente dois tipo de objetos, o “SoapPrimitive” e o “SoapObject”.

O objeto do tipo “SoapPrimitive” pode ser convertido para os tipos primitivos de dados como *string, inteiro, double* ou *booleano*, já os objetos “SoapObject” possuem coleções de outros “SoapObject” ou de “SoapPrimitive”. Sendo assim conversão do retorno dos serviços devem ser implementados pelo desenvolvedor.

CONCLUSÃO

Este trabalho realizou a reestruturação do SIPE – Sistema do Processo de Enfermagem Informatizado ao Paciente Politraumatizado, modificando sua arquitetura para a de Web Service, além de aplicar padrões de projeto segundo o modelo MVC.

Com uma análise simples, se verificou o estado crítico, que se encontrava o sistema, desestruturado e vulnerável, prejudicando sua manutenção. A partir de desses achados, que justificam a proposta deste trabalho, foram aplicadas as técnicas necessárias para implementar o Web Service e aplicar padrões de projeto no modelo MVC. Sem deixar de fora os cuidados com a segurança das informações e dados operados pelo sistema.

Essa reestruturação trouxe benefícios ao SIPE como: facilidade de manutenção, integração com aplicações de diferentes plataformas, segurança das informações trafegadas, disponibilização para dispositivos móveis com Android, entre outras. Com isso os objetivos propostos foram alcançados, através da padronização e da reestruturação em forma de Web Service.

Atualmente, estudos nesta área e relacionados ao SIPE continuam sendo realizados, pelo grupo de pesquisadores do curso de Enfermagem da UFSC, buscando melhorias, modernização ou complementação. Com o resultado positivo deste trabalho, os esforços serão minimizados, sob o ponto de vista computacional, deixando a equipe direcionada para as regras e gestão de seu conteúdo.

Mas o principal legado deixado com este trabalho foi ter desenvolvido uma aplicação segura e confiável, que possa ser efetivamente utilizada em uma unidade

de terapia intensiva, como ferramenta de suporte ao profissional de enfermagem, e com isso auxiliando na manutenção de vidas.

Sugestões de trabalhos futuros

Sendo o SIPE ainda objeto de estudos que visam seu aperfeiçoamento e/ou sua atualização dentro das técnicas de enfermagem, existem alguns pontos que ainda podem ser explorados em trabalhos de pesquisa futuros:

- Implantar no sistema o uso de inteligência artificial, uma vez que dada certa entrada de dados o sistema forneça ao usuário os dados de saída específicos, permitindo que o usuário os altere criando um novo padrão.
- Integrar o sistema com os equipamentos de monitoramento que estão ligados ao paciente, pois é cada vez maior a capacidade destes equipamentos fornecerem informações por meio digital que possam ser lidos por outros sistemas.
- Possibilitar, através dos dados coletados, a geração e disponibilização de um boletim médico em horário previamente determinado, disponível tanto para o sistema hospitalar, quanto para o responsável pelo paciente.
- Estudos sobre ergonomia e usabilidade deste sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTUNES, Camila R. **Processo de enfermagem informatizado ao paciente politraumatizado de terapia intensiva via Web**. 164 f. Dissertação - Pós-Graduação em Enfermagem, UFSC, Florianópolis, 2006.

APACHE_CXF. **An Open-Source Services *Framework***. Disponível em: < http://cxf.apache.org/ >. Acesso em: 01 de Novembro de 2013.

APACHE_TOMCAT. **Apache Tomcat**. Disponível em: <http://tomcat.apache.org/>. Acesso em: 01 de Novembro de 2013.

BARRA, Daniela C. C. **Processo de Enfermagem Informatizado em Terapia Intensiva em Ambiente PDA (Personal Digital Assistant) a Partir da CIPE® Versão 1.0**. 159 f. Dissertação de Mestrado - Pós-Graduação em Enfermagem, UFSC, Florianópolis, 2008.

BESEN, Rafael. **Informatização e Modernização de Processos de Prefeituras Utilizando Software Livre: sistema de controle de almoxarifado de farmácias públicas**. Trabalho de Conclusão de Curso - Departamento de Informática e Estatística, UFSC. Florianópolis, 2007.

BOOTSTRAP. **About**. Disponível em: <http://getBootStrap.com/about/>. Acesso em: 01 de Novembro de 2013.

CHASE, Nicholas. **Entendendo Especificações de Serviços da Web, Parte 4: WS-Security**. IBM: Developer Works. Disponível em: < http://www.ibm.com/developerworks/br/Webservices/tutorials/ws-understand-Web-services4/>. Acesso em 05 de novembro de 2013.

DALL'OGLIO, Pablo. **Programando com Orientação a Objetos: Inclui Design Patterns**. 1ª edição. São Paulo: Novatec, 2007.

DAL SASSO, Grace T. M. **Uma proposta do Processo de Enfermagem Informatizado em Terapia Intensiva a partir da CIPE versão B1**. Instituto de Cardiologia - Secretaria de Estado da Saúde. 45f. Mimeografado. São José (SC)1999.

DAL SASSO, Grace T. M. **A Concepção do Enfermeiro na produção tecnológica informatizada para ensino/aprendizagem em reanimação cárdio-respiratória**. 203 f. Tese Doutorado em Enfermagem – Curso de Pós-Graduação em Enfermagem, UFSC. Florianópolis, 2001.

DAL SASSO, Grace T. M. **Sistematização e informatização da assistência Enfermagem na UTI. In: XI Jornada Sul Brasileira de Terapia Intensiva**. Florianópolis, 2005.

DAL SASSO, Grace T. M. et al. **Processo de enfermagem informatizado: metodologia para associação da avaliação clínica, diagnósticos, intervenções e resultados**. Artigo Científico. Revista da Escola de Enfermagem da USP. São Paulo. 2013.

DARLAN, Diego. **Vantagens e Desvantagens no Uso de *Framework*s**. Disponível em:<http://www.oficinadanet.com.br/artigo/682/vantagens_e_desvantagens_no_uso_de_*Framework*>. Acesso em 01 de junho de 2009.

ECLIPSE. **About the Eclipse Foundation**. Disponível em: < http://www.eclipse.org/org/>. Acesso em: 20 de novembro de 2013.

FAYAD, Mohamed, e Douglas SCHMIDT. **Object-Oriented Application *Framework*s Communications of the ACM**. New York, 1997, 32-38.

FOWLER, Martin. **Patterns of Enterprise Application Architecture**. 1ª Edição. Addison-Wesley Professional, 2003.

HIBERNATE. **Informações sobre o *Framework***. Disponível em: < http://www.hibernate.org/>. Acesso em: 01 de Novembro de 2013.

IBM_WS. **Serviços da Web Java: O Alto Custo de (WS-)Security: IBM: Developer Works**. Disponível em: < http://www.ibm.com/developerworks/br/java/library/j-jws6/index.html>. Acesso em: 05 de novembro de 2013.

JAVA. **Obtenha Informações sobre a Tecnologia Java**. Disponível em: <. http://www.java.com/pt_BR/about/>. Acesso em: 01 de Novembro de 2013.

JAVAFREE. **Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME**. Disponível em: <http://javafree.uol.com.br/artigo/871485/ >. Acesso em: 10 de novembro de 2013.

LEITE, Jair C. **Design da Arquitetura de Componentes de Software**. Notas de aula de Engenharia de Software. 2000. Disponível em: <http://www.dimap.ufrn.br/~jair/ES/c7.html>. Acesso em: 08 de dezembro de 2013.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP**. 2ª Edição. São Paulo: Novatec Editora, 2011.

OFICINA DA NET. **MVC - O padrão de arquitetura de software**. disponível em: <http://www.oficinadanet.com.br/artigo/1687/mvc_-_o_padrao_de_arquitetura_de_software>. Acesso em: 05 de novembro de 2013.

OLIVEIRA, Ricardo Ramos. **Avaliação de Manutenibilidade entre as Abordagens de Web Services RESTful e SOAP-WSDL**. 99 f. Dissertação de Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computação, Universidade São Paulo, São Carlos, 2012.

OLIVEIRA, Ronielton Rezende. **Criptografia simétrica e assimétrica: os principais algoritmos de cifragem**. Artigo Científico. Revista Segurança Digital. 2012.

ORACLE. **Oracle Technology Network - JAVA**. Disponível em: < http://www.oracle.com/technetwork/java/index.html>. Acesso em: 14 de novembro de 2013.

PAVEI, Vinicius Z. ***Framework* para Criação de Formulários para Sistema Operacional Android**. 62 f. Trabalho de Conclusão de Curso, Sistemas de Informação, UFSC, Florianópolis, 2012.

PHP. **Manual do PHP**. Disponível em: <http://www.php.net/manual/pt_BR/index.php >. Acesso em: 01 de Novembro de 2013.

PIANTINO, André P. L. **Análise do Suporte à Automação de Testes na Plataforma Aberta**. 62 f. Trabalho de Conclusão de Curso, Sistemas de Informação, UFSC, Florianópolis, 2008.

POSTGRESQL. **Sobre o PostgreSQL**. Disponível em: < http://www.postgresql.org.br/sobre>. Acesso em: 01 de Novembro de 2013.

RECKZIEGEL, M. **Criptografia em PHP e WebServices**. Disponível em: < http://imasters.com.br/artigo/4802/>. Acesso em: 01 de Dezembro de 2013.

SAMPAIO, E. **Introdução a criptografia e seu uso no PHP**. Disponível em: < http://www.devmedia.com.br/introducao-a-criptografia-e-seu-uso-no-php/10045 >. Acesso em: 01 de Dezembro de 2013.

SMARTY. SMARTY **Template Engine**. Disponível em: < http://www.Smarty.net/manual/pt_BR/what.is.Smarty.php>. Acesso em: 14 de novembro de 2013.

SPRING. ***Framework*** **Reference Documentation**. Disponível em: < http://spring.io/docs>. Acesso em: 14 de Novembro de 2013.

STENZEL, C. **BLOG TI - O que é Android**. Disponível em: http://www.carlosstenzel.com/TI/o-que-e-android/>. Acesso em: 20 de novembro de 2013.

THE CODE BAKERS. **The Code Bakers - Desenvolvimento para Android**. Disponível em: <http://www.thecodebakers.org/p/licao-1-desenvolvimento-para-android.html >. Acesso em: 20 de novembro de 2013.

TIOBE. **Programming Community Index**. Disponível em: < http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html >. Acesso em: 01 de Novembro de 2013.

UNIFIL. **Programação MVC e ZEND *Framework***. Disponível em: <http://www.unifil.br/portal/arquivos/publicacoes/paginas/2012/11/516_875_publipg.pdf>. Acesso em 05 de novembro de 2013.

VIEGAS, Charles. **Aquele blog de SOA**. Disponível em: < http://www.aqueleblogdesoa.com.br/2008/08/anatomia-do-wsdl/>. Acesso em 20 de novembro de 2013.

WIKIPÉDIA_1. ***Framework***. Disponível em: < https://pt.wikipedia.org/wiki/*Framework* >. Acesso em: 05 de novembro de 2013.

WIKIPÉDIA_2. **Data Access Object - Objeto de acesso a dados**. Disponível em: < http://pt.wikipedia.org/wiki/Data_Access_Object>. Acesso em 05 de novembro de 2013.

WIKIPEDIA_3. **MVC**. Disponível em: < http://pt.wikipedia.org/wiki/MVC >. Acesso em: 05 de novembro de 2013.

WIKIPÉDIA_4. **Spring Framework**. Disponível em: < http://pt.wikipedia.org/wiki/Spring_Framework >. Acesso em 05 de novembro de 2013.

WIKIPEDIA_5. **Web Service**. Disponível em: < http://pt.wikipedia.org/wiki/WebServices >. Acesso em: 14 de novembro de 2013.

ZABOTTI, C.; SOUZA, J. **Metodologia Eletrônica de Cuidados de Enfermagem aos Pacientes de terapia intensiva com alterações respiratórias utilizando CIPE**. 109 f. Trabalho de Conclusão de Curso – Faculdade de Enfermagem, Universidade do Sul de Santa Catarina. Palhoça, 2002.

ANEXOS

Anexo I - Principais algoritmos de chave privada ou criptografia simétrica (Oliveira, 2012)

Algoritmo	Bits	Descrição
AES	128	O Advanced Encryption Standard (AES) é uma cifra de bloco, anunciado pelo National Institute of Standards and Technology (NIST) em 2003, fruto de concurso para escolha de um novo algoritmo de chave simétrica para proteger informações do governo federal, sendo adotado como padrão pelo governo dos Estados Unidos, é um dos algoritmos mais populares, desde 2006, usado para criptografia de chave simétrica, sendo considerado como o padrão substituto do DES. O AES tem um tamanho de bloco fixo em 128 bits e uma chave com tamanho de 128, 192 ou 256 bits, ele é rápido tanto em software quanto em hardware, é relativamente fácil de executar e requer pouca memória.
DES	56	O Data Encryption Standard (DES) foi o algoritmo simétrico mais disseminado no mundo, até a padronização do AES. Foi criado pela IBM em 1977 e, apesar de permitir cerca de 72 quadrilhões de combinações, seu tamanho de chave (56 bits) é considerado pequeno, tendo sido quebrado por "força bruta" em 1997 em um desafio lançado na internet. O NIST que lançou o desafio mencionado, recertificou o DES pela última vez em 1993, passando então a recomendar o 3DES.
3DES	112 ou 168	O 3DES é uma simples variação do DES, utilizando o em três ciframentos suscessivos, podendo empregar uma versão com duas ou com três chaves diferentes. É seguro, porém muito lento para ser um algoritmo padrão.
IDEA	128	O International Data Encryption Algorithm (IDEA) foi criado em 1991 por James Massey e Xuejia Lai e possui patente da suíça ASCOM Systec. O algoritmo é estruturado seguindo as mesmas linhas gerais do DES. Mas na maioria dos microprocessadores, uma implementação por software do IDEA é mais rápida do que uma implementação por software do DES. O IDEA é utilizado principalmente no mercado financeiro e no PGP, o programa para criptografia de e-mail pessoal mais disseminado no mundo.
Blowfish	32 a 448	Algoritmo desenvolvido por Bruce Schneier, que oferece a escolha, entre maior segurança ou desempenho através de chaves de tamanho variável. O autor aperfeiçoou o no Twofish.
Twofish	128	É uma das poucas cifras incluídas no OpenPGP. O Twofish é uma chave simétrica que emprega a cifra de bloco de 128 bits, utilizando chaves de tamanhos variáveis, podendo ser de 128, 192 ou 256 bits. Ele realiza 16 interações durante a criptografia, sendo um algoritmo bastante veloz. A cifra Twofish não foi patenteada estando acessível no domínio público, como resultado, o algoritmo Twofish é de uso livre para qualquer um utilizar sem restrição.
RC2	8 a 1024	Projetado por Ron Rivest (o R da empresa RSA Data Security Inc.) e utilizado no protocolo S/MIME, voltado para criptografia de e-mail corporativo. Também possui chave de tamanho variável. Rivest também é o autor dos algoritmos RC4, RC5 e RC6.
CAST	128	É um algoritmo de cifra de bloco, sendo criado em 1996 por Carlisle Adams e Stafford Tavares. O CAST-128 é um algoritmo de Feistel, com 12 a 16 iterações da etapa principal, tamanho de bloco de 64 bits e chave de tamanho variável (40 a 128 bits, com acréscimos de 8 bits). Os 16 rounds de iteração são usados quando a chave tem comprimento maior que 80 bits.

Anexo II - Principais algoritmos de chave pública ou criptografia assimétrica (Oliveira, 2012)

Algoritmo	Descrição
RSA	O RSA é um algoritmo assimétrico que possui este nome devido a seus inventores: Ron Rivest, Adi Shamir e Len Adleman, que o criaram em 1977 no MIT. Atualmente, é o algoritmo de chave pública mais amplamente utilizado, além de ser uma das mais poderosas formas de criptografia de chave pública conhecidas até o momento. O RSA utiliza números primos. A premissa por trás do RSA consiste na facilidade de multiplicar dois números primos para obter um terceiro número, mas muito difícil de recuperar os dois primos a partir daquele terceiro número. Isto é conhecido como fatoração. Por exemplo, os fatores primos de 3.337 são 47 e 71. Gerar a chave pública envolve multiplicar dois primos grandes; qualquer um pode fazer isto. Derivar a chave privada a partir da chave pública envolve fatorar um grande número. Se o número for grande o suficiente e bem escolhido, então ninguém pode fazer isto em uma quantidade de tempo razoável. Assim, a segurança do RSA baseia se na dificuldade de fatoração de números grandes. Deste modo, a fatoração representa um limite superior do tempo necessário para quebrar o algoritmo. Uma chave RSA de 512 bits foi quebrada em 1999 pelo Instituto Nacional de Pesquisa da Holanda, com o apoio de cientistas de mais 6 países. Levou cerca de 7 meses e foram utilizadas 300 estações de trabalho para a quebra. No Brasil, o RSA é utilizado pela ICP-Brasil, no seu sistema de emissão de certificados digitais, e a partir do dia 1º de janeiro de 2012, as chaves utilizadas pelas autoridades certificadoras do país, passam a serem emitidas com o comprimento de 4.096bits, em vez dos 2.048bits atuais.
ElGamal	O ElGamal é outro algoritmo de chave pública utilizado para gerenciamento de chaves. Sua matemática difere da utilizada no RSA, mas também é um sistema comutativo. O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado problema do logaritmo discreto. Assim, o ElGamal obtém sua segurança da dificuldade de calcular logaritmos discretos em um corpo finito, o que lembra bastante o problema da fatoração.
Diffie-Hellman	Também baseado no problema do logaritmo discreto, e o criptosistema de chave pública mais antigo ainda em uso. O conceito de chave pública, aliás foi introduzido pelos autores deste criptosistema em 1976. Contudo, ele não permite nem ciframento nem assinatura digital. O sistema foi projetado para permitir a dois indivíduos entrarem em um acordo ao compartilharem um segredo tal como uma chave, muito embora eles somente troquem mensagens em público.
Curvas Elípticas	Em 1985, Neal Koblitz e V. S. Miller propuseram de forma independente a utilização de curvas elípticas para sistemas criptográficos de chave pública. Eles não chegaram a inventar um novo algoritmo criptográfico com curvas elípticas sobre corpos finitos, mas implementaram algoritmos de chave pública já existentes, como o algoritmo de Diffie-Hellman, usando curvas elípticas. Assim, os sistemas criptográficos de curvas elípticas consistem em modificações de outros sistemas (o ElGamal, por exemplo), que passam a trabalhar no domínio das curvas elípticas, em vez de trabalharem no domínio dos corpos finitos. Eles possuem o potencial de proverem sistemas criptográficos de chave pública mais seguros, com chaves de menor tamanho. Muitos algoritmos de chave pública, como o Diffie-Hellman, o ElGamal e o Schnorr podem ser implementados em curvas elípticas sobre corpos finitos. Assim, fica resolvido um dos maiores problemas dos algoritmos de chave pública, o grande tamanho de suas chaves. Porém, os algoritmos de curvas elípticas atuais, embora possuam o potencial de serem rápidos, são em geral mais demorados do que o RSA.

Anexo III - ./paciente.php

```
<?
/* Inicia a sessão */
session_start();

if(isset($_SESSION['dados_usuario'])){

$codP = $_SESSION['prontuario'];
$botao = $_POST['botao'];
$banco = new BancoDeDados();
        $conectado = $banco->conecta();
        if($conectado){
                $sql = "SELECT nome FROM funcionarios WHERE cpf = '$dados[0]'";
                $dados = $banco->executaQuery($sql);
                if($dados){
                        while($valor = mysql_fetch_array($dados)){

                                $funcionario = $valor["nome"];
                        }
                }
                $sql = "SELECT * FROM pacientes WHERE codProntuario = '$codP'";
                $dados = $banco->executaQuery($sql)
                if($dados){
                        $valor = mysql_fetch_array($dados);
                        $nomeP = $valor['nome'];
                }
                $sql = "SELECT * FROM prontuarios WHERE codProntuario = '$codP'";
                $dados = $banco->executaQuery($sql);

                if($dados){
                        $valor = mysql_fetch_array($dados);
                        $prontuario = $valor['codProntuario'];
                        $dataAbertura = $valor['abertura'];
                }
                $sql = "SELECT idInternacoes FROM internacoes WHERE codProntuario = $codP";
                $dados = $banco->executaQuery($sql);
                 while($valor = mysql_fetch_array($dados)){
                                $idInternacao = $valor["idInternacoes"];
                }
        }

        if($botao == "Salvar"){
```

```
                    if($conectado){
                        $sql = "SELECT idAvaliacaoNeurologica FROM
avaliacaoNeurologica WHERE concluido = '0' AND idInternacoes = $idInternacao";
                        $dados = $banco->executaQuery($sql);
                         while($valor = mysql_fetch_array($dados)){
                                $idAvaliacaoNeurologica =
$valor["idAvaliacaoNeurologica"];
                        }
                        if($idAvaliacaoNeurologica <> ""){
                            $sql = "INSERT INTO avaliacoes
(idAvaliacaoNeurologica, idinternacoes) VALUES ('$idAvaliacaoNeurologica',
'$idInternacao')";
                            $dados = $banco->executaQuery($sql);
                            $sql = "UPDATE avaliacaoNeurologica  SET
concluido = '1' WHERE "
                                ." idInternacoes = $idInternacao AND
concluido = '0'";
                            $dados = $banco->executaQuery($sql);
                        }
                    }
                }
        ?>
        <script language="JavaScript">

        <!-- BEGIN
            var dateformat = "dd/mm/yyyy";
            current_width  = screen.width-10;
            current_height = screen.height-60;

            function addEvent(elm, event, funct, useCapture) {
                if(elm.attachEvent)//I.E
                  return elm.attachEvent("on"+event, funct);
                else if(elm.addEventListener) { //Mozilla and ?
                  elm.addEventListener(event, funct, useCapture);
                  return true;
                } else
                  return false;
            }

            function validar_atributo(strValue) {
                var objRegExp = new RegExp("[|\'"]");
                return objRegExp.test(strValue);
            }

            String.prototype.trim = function() {
                var str = this.replace(/^\s*/, "");
                str = str.replace(/\s*$/, "");
                return str;
```

```
    }

    function textlength(obj,tam) {
            if(obj.value.length >= tam) {
               obj.value=obj.value.substring(0,tam);
               return false;
            } else
               return true;
    }

    function placeFocus() {
            if (document.forms.length > 0) {
                var field = document.forms[0];
                for (i = 0; i < field.length; i++) {
                    if (
                       (field.elements[i].type == "text") ||
                       (field.elements[i].type == "textarea") ||
                       (field.elements[i].type.toString().charAt(0) == "s")
                       ) {
                       document.forms[0].elements[i].focus();
                       break;
                    }
                }
            }
    }

    //document.oncontextmenu = function(){return false;}
    if(document.all)
      document.onmousedown = function(){return false;}
    else {
      window.captureEvents(Event.MOUSEDOWN);
      window.onmousedown = function(e) {
             if(e.target==document)
                    return false;
      }
    }

    //SCRIPT CRIADO P/ FECHAR O MENU.
    if (parent.CloseAllMenus)
           document.onclick = parent.CloseAllMenus;
    else if (parent.parent.CloseAllMenus)
           document.onclick = parent.parent.CloseAllMenus;
    else if(parent.parent.parent.CloseAllMenus)
           document.onclick = parent.parent.parent.CloseAllMenus;
    else if(parent.parent.parent.parent.CloseAllMenus)
           document.onclick = parent.parent.parent.parent.CloseAllMenus;
//END-->
</script>
```

```
<style type="text/css">

.style1 {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-weight: bold;
	color: #FFFFFF;
}
body {
	background-color: #FFFFFF;
	margin-left: 0px;
	margin-top: 0px;
	margin-right: 0px;
	margin-bottom: 0px;
}
.style3 {font-family: Arial}

</style>
<script language="JavaScript" src="config/telaData.js"></script>



<div id="popupcalendar" class="text"> </div>
	<script>
  function mouseMove ()
	{
		try {
			window.opener.mouseMove();
		} catch (x) { }

		try {
			parent.parent.parent.parent.parent.tmout = 0;
		} catch (x) { }
	}

	document.onmousemove = mouseMove;
  </script>

<form action="" method="post" name="form" id="form">
<table width="98%" border="0" align="center">
  <tr bgcolor="#9966FF">
    <td bgcolor="#0066FF"><span class="style1">Dados do
cliente</span></td>
  </tr>
  <tr>
	<td>
		<table width="100%" border="0">
		  <tr>
			<td width="14%" align="right">Nome Enfermeiro:</td>
```

```
                    <td width="38%" align="left"><input
name="nomeEnfermeiro" type="text" disabled="disabled" id="nomeEnfermeiro"
value="<? echo $funcionario; ?>" size="50"></td>
                    <td width="15%" align="right"> </td>
                    <td width="33%" align="left"> </td>
                  </tr>
                  <tr>
                    <td align="right"><p>Nome cliente:</p>          </td>
                    <td  align="left"><input  name="nomePaciente"  type="text"
id="nomePaciente"  value="<?  echo  $nomeP;  ?>"  size="50"
disabled="disabled"></td>
                    <td align="right">Data / Hora:</td>
                    <td  align="left"><input  name="data"  type="text"  id="data"
value="<? echo date("d/m/Y H:i:s"); ?>" size="20" disabled="disabled">
                      <a                                                  href="#"
onClick="showCalendar(185,165,'form','data','data',",",",",",");return true;"></a></td>
                  </tr>
                  <tr>
                    <td align="right">Prontu&aacute;rio N&ordm;:</td>
                    <td  align="left"><input  name="numProntuario"  type="text"
id="numProntuario" value="<? echo $codP; ?>" size="20" disabled="disabled"></td>
                    <td align="right">Data / Hora Admiss&atilde;o</td>
                    <td  align="left"><input  name="dataAdmissao"  type="text"
id="dataAdmissao"  value="<?  echo  $dataAbertura;  ?>"  size="20"
disabled="disabled">
                      <a                                                  href="#"
onClick="showCalendar(185,165,'form','dataAdmissao','cal',",",",",",");return
true;"></a></td>
                  </tr>
                  <tr>
                    <td align="right">Proced&ecirc;ncia:</td>
                    <td  align="left"><input  name="procedencia"  type="text"
id="procedencia" size="40"></td>
                    <td align="right">Diagnóstico M&eacute;dico :</td>
                    <td  align="left"><input  name="diagMedico"  type="text"
id="diagMedico" size="50"></td>
                  </tr>
                </table>       </td>
          </tr>
          <tr>
            <td> </td>
          </tr>
          <tr>
            <td><table width="100%" border="0">
              <tr>
                <td align="right"><div align="right">SV N&atilde;o Invasivos : </div></td>
                <td  align="left"><input  name="svNaoInvasivo"  type="text"
id="svNaoInvasivo" size="5" /></td>
```

```
            <td align="right"><div align="right">SV Invasivos : </div></td>
            <td align="left"><input name="scInvasivo" type="text" id="scInvasivo"
size="5" /></td>
            <td align="right"><div align="right">PAP : </div></td>
            <td align="left"></td>
            <td align="right"><div align="right">Satura&ccedil;&atilde;o de O2:
</div></td>
            <td align="left"><input name="SaturacaoO2" type="text"
id="SaturacaoO2" size="5" /></td>
          </tr>
          <tr>
            <td align="right"><div align="right">PA : </div></td>
            <td align="left"><input name="pa" type="text" id="pa" size="5" /></td>
            <td align="right"><div align="right">PAM : </div></td>
            <td align="left"><input name="pam" type="text" id="pam" size="5" /></td>
            <td align="right"><div align="right">S : </div></td>
            <td align="left"><input name="papS" type="text" id="papS" size="5"
/></td>
            <td align="right"><div align="right">Capnografia : </div></td>
            <td align="left"><input name="capnografia" type="text" id="capnografia"
size="5" /></td>
          </tr>
          <tr>
            <td align="right"><div align="right">P : </div></td>
            <td align="left"><input name="p" type="text" id="p" size="5" /></td>
            <td align="right"><div align="right">PIC : </div></td>
            <td align="left"><input name="pic" type="text" id="pic" size="5" /></td>
            <td align="right"><div align="right">M : </div></td>
            <td align="left"><input name="papM" type="text" id="papM" size="5"
/></td>
            <td align="right"></td>
            <td align="left"></td>
          </tr>
          <tr>
            <td align="right"><div align="right">R : </div></td>
            <td align="left"><input name="r" type="text" id="r" size="5" /></td>
            <td align="right"><div align="right">PPC : </div></td>
            <td align="left"><input name="ppc" type="text" id="ppc" size="5" /></td>
            <td align="right"><div align="right">D : </div></td>
            <td align="left"><input name="papD" type="text" id="papD" size="5"
/></td>
            <td align="right"><div align="right">(PAM-PIC Fórmula) : </div></td>
            <td align="left"><input name="pam-pic" type="text" id="pam-pic" size="5"
/></td>
          </tr>
          <tr>
            <td align="right"><div align="right">T : </div></td>
            <td align="left"><input name="t" type="text" id="t" size="5" /></td>
```

```
              <td align="right"><div align="right">PVC/PAD : </div></td>
              <td align="left"><input name="pvc" type="text" id="pvc" size="5" /></td>
              <td align="right"></td>
              <td align="left"></td>
              <td align="right"><div align="right">PCP : </div></td>
              <td align="left"><input name="pcp" type="text" id="pcp" size="5" /></td>
              <td align="right"><div align="right">DC : </div></td>
              <td align="left"><input name="dc" type="text" id="dc" size="5" /></td>
              <td align="right"><div align="right">IC : </div></td>
              <td align="left"><input name="ic" type="text" id="ic" size="5" /></td>
            </tr>
          </table></td>
        </tr>
        <tr>
          <td><table width="100%" border="0">
            <tr>
              <td>Dados subjetivos do cliente e/ou fam&iacute;lia : </td>
            </tr>
          </table></td>
        </tr>
        <tr>
          <td><table width="100%" border="0">
            <tr>
              <td><textarea name="dadosSubjetivos" cols="70" rows="3"
id="dadosSubjetivos"></textarea></td>
                    <td><div align="center">
                      <input name="botao" type="submit" id="botao" value="Salvar">
                      </div></td>
                    <td><div align="center">
                      <input type="reset" name="Reset" value="Limpar">
                      </div></td>
                    <td><div align="center">
            <input type="submit" name="Submit2" value="Ajuda" />
            </div></td>
                    <td> </td>
            </tr>
          </table></td>
        </tr>
      </table>
      </form>

      <?
      $banco->desconecta();

      }else {
            header("Location: login.php");
      }
      ?>
```

Anexo IV - ./www/config/BancoDeDados.class.php

```
<?
 class BancoDeDados {
        var $host = "localhost";
        var $usuario = "nfrinfor";
        var $database = "nfrinfor";
        var $senha = "aedwoox";
        var $conexao;

        function BancoDeDados() {

        }

        function conecta() {
        $this->conexao = mysql_connect($this->host,$this->usuario,$this->senha);

                if (!$this->conexao) {
                //  echo "Nï¿½o foi possï¿½vel conectar-se ao Bando de Dados MySQL";
                return false;
          } else {
                        if (!mysql_select_db($this->database,$this->conexao)) {

                                // echo "Banco de dados nï¿½o encontrado";
                                return false;
                        }else {
                                //    echo "Banco de dados conectado!";
                                return true;
                        }
                }
 }
 function desconecta() {
    mysql_close($this->conexao);
 }


 function executaQuery($query) {
    $retorno = mysql_query($query,$this->conexao);
    return $retorno;
 }
 function totalDeResultados($query) {
    $retorno = mysql_numrows($this->executaQuery($query));
    return $retorno;
 }
 function consulta($query) {
    $retorno = mysql_db_query($database,$query,$this->conexao);
    return $retorno;
 }
 function iniciarTransacao() {
      $this->executaQuery('begin');
 }


 function efetuarTransacao() {
    $this->executaQuery('commit');
 }
```

```
  function desfazerTransacao() {
     $this->desfazerTransacao('rollback');
  }
  function id() {
     return mysql_insert_id();
  }
}
//Esta sendo feito aqui pois este arquivo é chamado por todos
define ("PATH_SIS", realpath(dirname(__FILE__).'/..'));
```

Anexo V - spring-beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springFramework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springFramework.org/schema/context"
	xmlns:task="http://www.springFramework.org/schema/task"
xmlns:cxf="http://cxf.apache.org/core"
	xmlns:jaxws="http://cxf.apache.org/jaxws"
xmlns:tx="http://www.springFramework.org/schema/tx"
	xmlns:p="http://www.springFramework.org/schema/p"
	xsi:schemaLocation="http://www.springFramework.org/schema/beans
		http://www.springFramework.org/schema/beans/spring-beans.xsd
	http://www.springFramework.org/schema/context
	http://www.springFramework.org/schema/context/spring-context.xsd

	http://cxf.apache.org/core
	http://cxf.apache.org/schemas/core.xsd
	http://cxf.apache.org/jaxws
	http://cxf.apache.org/schemas/jaxws.xsd

	http://www.springFramework.org/schema/tx
		http://www.springFramework.org/schema/tx/spring-tx.xsd
		http://www.springFramework.org/schema/aop
		http://www.springFramework.org/schema/aop/spring-aop.xsd
		http://www.springFramework.org/schema/task
		http://www.springFramework.org/schema/task/spring-task-3.0.xsd"

	default-autowire="byName">

	<context:annotation-config />

	<context:component-scan
		base-package="br.ufsc.tcc.sin.servico.impl,		br.ufsc.tcc.sin.negocio,
br.ufsc.tcc.sin.modelo.dao" />

	<task:annotation-driven />

	<bean class="br.ufsc.tcc.sin.init.BootStrapData" />

	<bean 		id="propertyPlaceholderConfigurer"
class="org.springFramework.beans.factory.config.PropertyPlaceholderConfigurer"
		p:location="classpath:hibernate.properties" />


	<!-- Precisa ser um singleton -->
	<bean 		id="gerenciadorTickets"
class="br.ufsc.tcc.sin.servico.ticket.GerenciadorTickets"
		scope="singleton">
		<property name="tempoSessaoSegundos" value="3600" />
	</bean>


	<!--
##########################################################################
```

```
                    CONFIGURAÇÃO DO TIMER

        ############################################################################## -->

                <bean                                                   id="scheduledGerenciadorTickets"
class="org.springFramework.scheduling.timer.ScheduledTimerTask">
                        <!-- wait 10 seconds before starting repeated execution -->
                        <property name="delay" value="10000" />
                        <!-- run every 15 minutes -->
                        <property name="period" value="900000" />
                        <property name="timerTask" ref="gerenciadorTickets" />
                </bean>

                <bean                                                                   id="timerFactory"
class="org.springFramework.scheduling.timer.TimerFactoryBean">
                        <property name="scheduledTimerTasks">
                                <list>
                                        <ref bean="scheduledGerenciadorTickets" />
                                </list>
                        </property>
                </bean>


                <!-- ********************************************************************
                                TARGET NTERCEPTORS - sessionFactory will get autowired
                         ******************************************************************** -->

                <bean                                                           id="hibernateInterceptor"
class="org.springFramework.orm.hibernate3.HibernateInterceptor">
                        <property name="sessionFactory">
                                <ref bean="sessionFactory" />
                        </property>
                </bean>

                <!-- ********************************************************************
                        ********************** SESSION FACTORY ****************************
                        ******************************************************************** -->


                <bean id="sessionFactory"

        class="org.springFramework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">

                        <property name="hibernateProperties">
                                <props>
                                        <prop key="hibernate.dialect">
                                                ${hibernate.dialect}
                                        </prop>
                                        <prop
key="hibernate.show_sql">${hibernate.hbm2ddl.show_sql}</prop>
                                        <prop
key="hibernate.hbm2ddl.auto">${hibernate.hbm2ddl.auto}</prop>
                                        <prop
key="hibernate.transaction.flush_before_completion">true</prop>
                                </props>

                        </property>
```

```
                <property name="annotatedClasses">
                  <list>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Alerta</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Avaliacao</value>
                         <value>br.ufsc.tcc.sin.modelo.entidade.CasoClinico</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Diagnostico</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.FormAvaliacao</value
<value>br.ufsc.tcc.sin.modelo.entidade.FormAvaliacaoDetalhe</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.FormDiagnostico</value>
        <value>br.ufsc.tcc.sin.modelo.entidade.FormDiagnosticoIntervencao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.FormIntervencao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Internacao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Intervencao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Oximetria</value>

                        <value>br.ufsc.tcc.sin.modelo.entidade.Prontuario</value>
                <value>br.ufsc.tcc.sin.modelo.entidade.SinalVitalInvasivo</value>
                <value>br.ufsc.tcc.sin.modelo.entidade.SinalVitalNaoInvasivo</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Sistema</value>

                        <value>br.ufsc.tcc.sin.modelo.entidade.acesso.Aplicacao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.acesso.Permissao</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.acesso.Servico</value>
        <value>br.ufsc.tcc.sin.modelo.entidade.acesso.Sessao</value>
                <value>br.ufsc.tcc.sin.modelo.entidade.acesso.SessaoDetalhe</value>
                <value>br.ufsc.tcc.sin.modelo.entidade.acesso.Usuario</value>

                        <value>br.ufsc.tcc.sin.modelo.entidade.Pessoa</value>
                        <value>br.ufsc.tcc.sin.modelo.entidade.Paciente</value>
                <value>br.ufsc.tcc.sin.modelo.entidade.Medico</value>

                  </list>

                </property>

                <property name="dataSource">
                        <ref bean="dataSource" />
                </property>

        </bean>

        <bean id="dataSource"
                class="org.apache.commons.dbcp.BasicDataSource" >

                <property name="driverClassName" value="${dataSource.driver}" />
                <property name="url" value="${dataSource.url}" />
                <property name="username" value="${dataSource.user}" />
                <property name="password" value="${dataSource.password}" />
                <property name="initialSize" value="5" />
                <property name="maxActive" value="20" />
                <property name="maxIdle" value="20" />
                <property name="maxWait" value="30000" />
                <property name="removeAbandoned" value="true" />
                <property name="removeAbandonedTimeout" value="30" />
```

```
		</bean>

		<!-- enable the configuration of transactional behavior based on annotations -->
		<tx:annotation-driven transaction-manager="transactionManager" mode="aspectj" />
		<bean
		class="org.springFramework.dao.annotation.PersistenceExceptionTranslationPostPro
cessor" />

		<bean id="transactionManager"
		class="org.springFramework.orm.hibernate3.HibernateTransactionManager">
			<property name="sessionFactory" ref="sessionFactory" />
		</bean>

		<!-- ********************************************************************
							CUSTOM INTERCEPTORS
			********************************************************************* -->

		<bean name="loggerInterceptor" class="br.ufsc.tcc.sin.interceptor.LoggingInterceptor"
/>
			</beans>
```

Anexo VI - cxf-servicos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springFramework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springFramework.org/schema/context"
  xmlns:cxf="http://cxf.apache.org/core"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xmlns:jaxrs="http://cxf.apache.org/jaxrs"
  xsi:schemaLocation="http://www.springFramework.org/schema/beans

http://www.springFramework.org/schema/beans/spring-beans-2.5.xsd

http://www.springFramework.org/schema/context

http://www.springFramework.org/schema/context/spring-context-2.5.xsd
                                                    http://cxf.apache.org/core
                                                    http://cxf.apache.org/schemas/core.xsd
                                                    http://cxf.apache.org/jaxws
                                                    http://cxf.apache.org/schemas/jaxws.xsd
                                                    http://cxf.apache.org/jaxrs
                                                    http://cxf.apache.org/schemas/jaxrs.xsd" >

        <!-- Carrega as configurações presentes nos jars do Apache CXF -->
    <import resource="classpath:META-INF/cxf/cxf.xml" />
    <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />
    <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

    <!-- Aegis data binding -->
        <bean id="aegisBean" class="org.apache.cxf.aegis.databinding.AegisDatabinding"
            scope="prototype" />
        <bean                                                   id="jaxws-and-aegis-service-factory"
class="org.apache.cxf.jaxws.support.JaxWsServiceFactoryBean"
            scope="prototype">
            <property name="dataBinding" ref="aegisBean" />
            <property name="serviceConfigurations">
                <list>
                    <bean
class="org.apache.cxf.jaxws.support.JaxWsServiceConfiguration" />
                    <bean
class="org.apache.cxf.aegis.databinding.AegisServiceConfiguration" />
                    <bean
class="org.apache.cxf.service.factory.DefaultServiceConfiguration" />
                </list>
            </property>
        </bean>

        <!-- Endpoint WSDL para o Apache CXF -->
    <!-- Dizemos o endereço, o ID do serviço, em qual bean ele depende -->

        <jaxws:endpoint id="acesso"
          address="/acesso"
            implementorClass="br.ufsc.tcc.sin.servico.impl.AcessoServicoImpl"
            implementor="#acessoServico"  >
            <jaxws:serviceFactory>
                <ref bean="jaxws-and-aegis-service-factory" />
```

```
                </jaxws:serviceFactory>
                <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature" />
        </jaxws:features>
            </jaxws:endpoint>

            <jaxws:endpoint id="administracao"
                address="/administracao"
                implementorClass="br.ufsc.tcc.sin.servico.impl.AdministracaoServicoImpl"
                implementor="#administracaoServico"  >
                <jaxws:serviceFactory>
                    <ref bean="jaxws-and-aegis-service-factory" />
                </jaxws:serviceFactory>
                <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature" />
        </jaxws:features>
            </jaxws:endpoint>

            <jaxws:endpoint id="formulario"
                address="/formulario"
                implementorClass="br.ufsc.tcc.sin.servico.impl.FormularioServicoImpl"
                implementor="#formularioServico"  >
                <jaxws:serviceFactory>
                    <ref bean="jaxws-and-aegis-service-factory" />
                </jaxws:serviceFactory>
                <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature" />
        </jaxws:features>
            </jaxws:endpoint>

            <jaxws:endpoint id="prontuario"
                address="/prontuario"
                implementorClass="br.ufsc.tcc.sin.servico.impl.ProntuarioServicoImpl"
                implementor="#prontuarioServico"  >
                <jaxws:serviceFactory>
                    <ref bean="jaxws-and-aegis-service-factory" />
                </jaxws:serviceFactory>
                <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature" />
        </jaxws:features>
            </jaxws:endpoint>


    </beans>
```

Anexo VII - Tabela com relação dos serviços de acordo com o grupo

AcessoServico • testeServico • validarTicket • autenticarUsuario	Endpoint address: http://localhost:8080/TCC/acesso WSDL : {http://servico.sin.tcc.ufsc.br/}AcessoService Target namespace: http://servico.sin.tcc.ufsc.br/
AdministracaoServico • pesquisarMedico • salvarIntervencao • salvarCasoClinicoAvaliacao • salvarAplicacao • buscarAssociacaoDiagnosticoIntervencao • pesquisarAvaliacao • buscarAvaliacao • pesquisarUsuario • salvarMedico • testeServico • pesquisarAplicacao • salvarUsuario • salvarDiagnostico • pesquisarCasoClinico • pesquisarIntervencao • salvarCasoClinico • salvarAvaliacao • buscarIntervencao • buscarAplicacao • buscarUsuarioPorEmail • pesquisarAssociacaoDiagnosticoIntervencao • gerarSenhaUsuario • salvarAssociacaoDiagnosticoIntervencao • buscarDiagnostico • buscarCasoClinicoAvaliacao • pesquisarCasoClinicoAvaliacao • alterarSenhaUsuario • pesquisarDiagnostico • buscarMedico • buscarCasoClinico	Endpoint address: http://localhost:8080/TCC/administracao WSDL : {http://servico.sin.tcc.ufsc.br/}AdministracaoService Target namespace: http://servico.sin.tcc.ufsc.br/
FormularioServico • listaIntervencoesPorPaciente • listaDiagnosticoPorCasoClinico • listaIntervencoesSugestao	Endpoint address: http://localhost:8080/TCC/formulario WSDL : {http://servico.sin.tcc.ufsc.br/}FormularioService

• listaPermissoes • listaCasoClinico • listaIntervencoes • listaOutrosTiposIntervencao • listaEscalaDor • listaDiagnosticosPorPaciente • listaFluidoTerapia • listaSistemas • listaSituacoesInternacao • testeServico • listaDiagnosticosSugestao • listaSubItemExames • listaDiagnosticos • listaExames • listaOximetriaDePulso • listaAvaliacoes • listaItemBalancoHidroeletrolitico • ultimasAlteracoes • listaTipoBalancoHidroeletrolitico • listaServicos • listaBalancoHidroeletrolitico • listaTipoExames • listaItemExames	Target namespace: http://servico.sin.tcc.ufsc.br/
ProntuarioServico • salvarIntervencao • salvarFluidoTerapia • testeServico • intervencoes • alterarSituacao • salvarProntuario • pacientesInternados • diagnosticos • salvarDiagnostico • avaliacoes • localizarPacientes • salvarAvaliacao • examesRealizados • balanco • adircionaExame • novoProntuario • fluidoTerapia • acessarProntuario • adicionarBalanco • alertas	Endpoint address: http://localhost:8080/TCC/prontuario WSDL : {http://servico.sin.tcc.ufsc.br/}ProntuarioService Target namespace: http://servico.sin.tcc.ufsc.br/

Available RESTful services:

Endpoint address: http://localhost:8080/TCC/acessoRest WADL : http://localhost:8080/TCC/acessoRest?_wadl