UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

WALISON CAVALCANTI MOREIRA

Uma Arquitetura de Software para Sistemas de Pesquisa das Pneumonias na Infância

Goiânia
2012

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Uma Arquitetura de Software para Sistemas de Pesquisa das Pneumonias na Infância

Autor(a): Walison Cavalcanti Moreira

Goiânia, 02 de Outubro de 2012.

Walison Cavalcanti Moreira – Autor

Dr. Leandro Luís Gaudino de Oliveira – Orientador

WALISON CAVALCANTI MOREIRA

Uma Arquitetura de Software para Sistemas de Pesquisa das Pneumonias na Infância

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Sistemas de Informação.

Orientador: Prof. Dr. Leandro Luís Gaudino de Oliveira

Goiânia
2012

WALISON CAVALCANTI MOREIRA

Uma Arquitetura de Software para Sistemas de Pesquisa das Pneumonias na Infância

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Computação, aprovada em 02 de Outubro de 2012, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Leandro Luís Gaudino de Oliveira
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Edmundo Sérgio Spoto
INF – UFG

Profa. Dra. Lisandra Manzoni Fontoura
DELC – UFSM

Walison Cavalcanti Moreira

Graduou-se em Ciências da Computação na PUC Goiás - Pontifícia Universidade Católica de Goiás. Especializou-se em MBA em Gerenciamento de Projetos na Fundação Getúlio Vargas. Foi professor na graduação da PUC Goiás no curso de Ciências da Computação, professor no curso de especialização em Desenvolvimento de Aplicações Web com Interfaces Ricas na UFG e professor em cursos técnicos no SENAI, SENAC e outras instituições de ensino. Sempre atuando com disciplinas de arquitetura, projeto e construção de software. Trabalha com sistemas web e Java a mais de 12 anos e atualmente é gerente de desenvolvimento na PC Sistemas.

À minha esposa e filhas e aos meus pais.

Agradecimentos

Agradeço ao Leandro, meu orientador e amigo, por ter acreditado e me apoiado dentro da universidade e fora dela. Sem dúvida, uma das pessoas que mais contribuiu com a minha carreira.

Sou grato pela oportunidade que o Prof. Vagner Sacramento (INF UFG) me deu para trabalhar com pesquisa na UFG e por ele ter me mostrado a importância de aproximar a ciência aos negócios.

À Profa. Ana Lúcia (IPTSP UFG) agradeço o investimento feito numa idéia que, com bastante esforço, se realizou num software de pesquisa que contribui para a vigilância de pneumonia infantil avaliando o impacto de vacinação.

A perfeição não é alcançada quando já não há mais nada para adicionar, mas quando já não há mais nada que se possa retirar.

Antoine de Saint-Exupéry,
Em refexão sobre o que é perfeito.

Resumo

Moreira, Walison Cavalcanti. **Uma Arquitetura de Software para Sistemas de Pesquisa das Pneumonias na Infância**. Goiânia, 2012. 92p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

As pneumonias estão entre as principais causas de morte das crianças com menos de 5 anos de idade. Várias entidades de saúde no mundo todo, públicas e privadas, estão empenhadas em investigar a doença e avaliar a eficiência dos mecanismos de prevenção e combate existentes.

As infecções que causam pneumonia podem ser evitadas. No entanto, principalmente em países pobres, os recursos para promover a prevenção são escassos. Assim as ações de combate precisam ser muito eficientes e eficazes. Para garantir a efetividade dessas ações, como as vacinas, são necessárias informações estatísticas como faixa etária, região, época, condição social e histórico obtido através de pesquisa em campo.

Este trabalho propõe e implementa uma arquitetura de software para construção, uso e manutenção de sistemas de pesquisa das pneumonias na infância. As técnicas, tecnologias, ferramentas e serviços utilizados na definição da arquitetura foram escolhidos com foco no baixo custo. Dessa forma fica muito mais viável a utilização de softwares para sistemas de pesquisa automatizados por entidades de saúde que possuem poucos recursos financeiros.

Palavras–chave

pneumonia infantil, sistema de pesquisa, arquitetura de software, baixo custo

Abstract

Moreira, Walison Cavalcanti. **A Software Architecture for Survey Systems of Childhood Pneumonia**. Goiânia, 2012. 92p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Pneumonia is of the main causes of death of children under 5 years of age. Several health organizations worldwide, public and private, are engaged in investigating the disease and evaluate the effectiveness of mechanisms to prevent and combat existing.

Infections causing pneumonia can be avoided. However, especially in poor countries, the resources to promote prevention are scarce. Thus the combat actions need to be very efficient and effective. To ensure the effectiveness of these actions, such as vaccines, are necessary statistical information like age range, region, period, social status and history obtained through field research.

This paper proposes and implements a software architecture for the construction, use and maintenance of research of childhood pneumonias. The techniques, technologies, tools and services used in defining the architecture were chosen with a focus on low cost. This way is much more feasible to use software for automated search systems by healthcare entities that have few financial resources.

Keywords

child pneumonia, survey systems, software architecture, low cost

Sumário

Lista de Figuras

Lista de Códigos de Programas

CAPÍTULO 1

Introdução

1.1 Contexto

Segundo relatório do Fundo das Nações Unidas para a Infância (Unicef), cerca de dois milhões de mortes infantis, causadas anualmente pela diarreia e pneumonia, podem ser evitadas. As doenças são responsáveis por um terço das mortes de crianças com menos de cinco anos no mundo (figura 1.1). Em alguns países da África e Ásia, quase 90% de todas as mortes infantis são causadas por essas duas enfermidades [41].

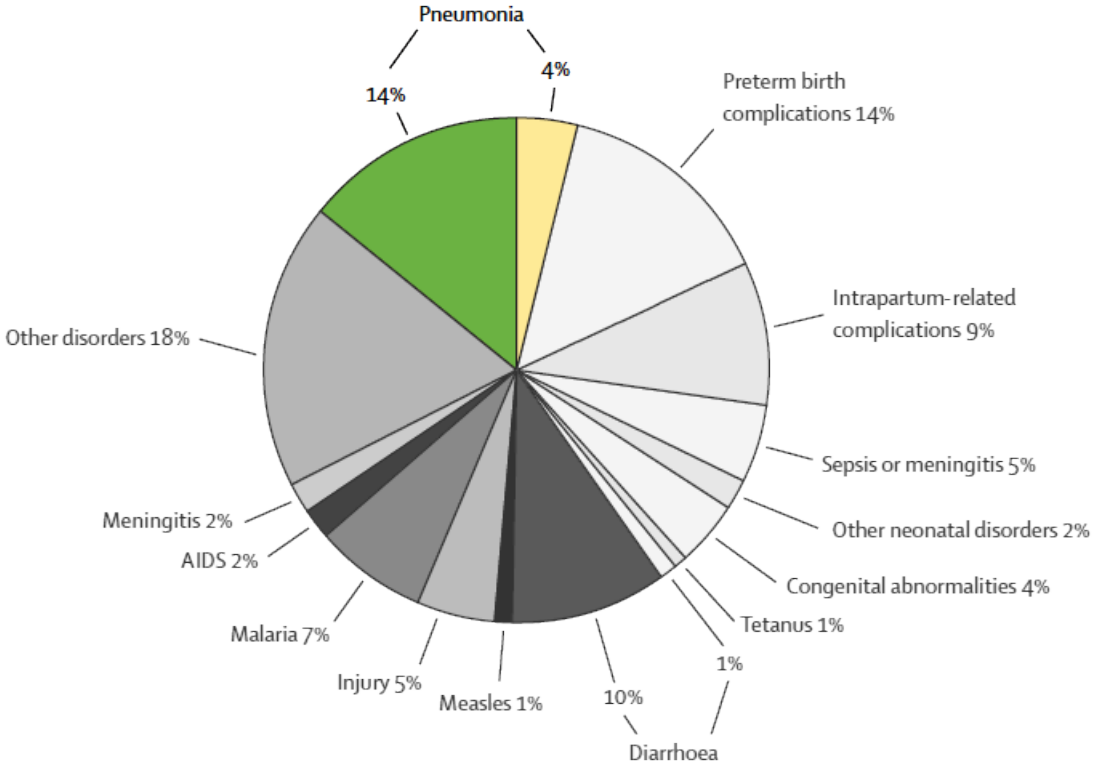

Figura 1.1: *Causas de mortalidade infantil no mundo em 2010.*

Já se sabe quais são os tratamentos eficientes contra a diarreia e a pneumonia, as duas doenças que mais atingem as classes pobres. Para a pneumonia uma das ações para tratamento da doença é a utilização de antibióticos. Esses medicamentos precisam

ser aplicados e sua eficiência medida para que sejam planejadas as próximas campanhas. Os responsáveis por essas campanhas geralmente são instituições de saúde que possuem poucos recursos financeiros para combater a pneumonia cobrindo grandes populações.

O uso de tecnologias móveis tem sido um ótimo recurso para profissionais de saúde no combate à pneumonia infantil muitas vezes em áreas remotas. Com esses recursos tecnológicos é possível alcançar comunidades isoladas que, de outra forma, seria impossível ou inviável alcançar [63].

As tecnologias para auxiliar o combate às pneumonias na infância através de verificação de efetividade de vacinas e outras ações de prevenção com medições em campo estão disponíveis. No entanto o desafio está em encontrar uma combinação de tecnologias que ao mesmo tempo atenda os requisitos (segurança, desempenho, usabilidade, entre outros) e custos impostos pelas condições das instituições de saúde pública principalmente.

1.2 Motivação

Esse trabalho foi criado a partir da necessidade de desenvolvimento de um sistema para apoio à um processo de pesquisa para verificação de eficiência de uma vacina contra pneumonia infantil. O Instituto de Patologia Tropical e Saúde Pública (IPTSP) da Universidade Federal de Goiás (UFG) precisava de apoio para realizar vigilância prospectiva para estimar o impacto da vacina pneumocócica conjugada com proteina D do Haemophilusinfluenzae (PHiD-CV) contra a pneumonia em crianças hospitalizadas em Goiânia - GO.

Dada a necessidade foi criado um projeto para uso de tecnologia móvel e banco de dados em nuvem para vigilância de pneumonia para avaliar o impacto da vacinação contra o pneumococo.

O uso de tecnologia móvel em pesquisas na área da saúde em países em desenvolvimento tem-se mostrado eficaz, principalmente para a coleta e relato de casos de doenças de notificação compulsória. No entanto, segundo especialistas do IPTSP-UFG, tecnologias móveis de baixo custo ainda não foram utilizada para avaliação do impacto de vacinas. Assim, a utilização desta tecnologia por meio de celulares em estudos populacionais poderia significar economia de tempo, de custo e melhoria da qualidade dos dados para políticas de saúde.

1.3 Objetivos

O objetivo então é a criação de uma arquitetura para o desenvolvimento de um sistema de pesquisa para coleta de dados individuais de crianças com suspeita de pneumonia em hospitais e transmissão automatizada. Este sistema será desenvolvido

com tecnologias abertas, desde sistemas operacionais (de celulares a servidores) até APIs (Application Programming Interface) para apoio ao desenvolvimento. O sistema, inicialmente, será utilizado por enfermeiras e pesquisadores do IPTSP-UFG para coleta e análise de dados individuais das crianças menores que 3 anos hospitalizadas com diagnostico de pneumonia no municipio de Goiânia - GO. É imprescindível o atendimento de requisitos como facilidade de uso, aprendizado, acurácia, rapidez na obtenção dos dados e custo.

Realizamos um estudo para encontrar trabalhos semelhantes de automação de formulários de pesquisa [9] [56]. Não encontramos alternativas que atendessem aos requisitos de custo e qualidade. As soluções mais comuns utilizam soluções tecnológicas de alto custo de aquisição (smartphones caros e frágeis) e manutenção (plano de dados da operadora de telefonia e contrato de manutenção do software) ou ainda utilizam processos manuais caros (coleta manual em formulários de papel) e de resposta lenta (dias para se ter dados estatísticos).

1.4 Sumário da Dissertação

No Capítulo 2 as funcionalidades do sistema de pesquisa das pneumonias na infância é apresentado. Essa parte é muito importante pois mostra o que motivou as decisões arquiteturais.

No Capítulo 3 apresentamos a estrutura detalhada da arquitetura construída, as ferramentas utilizadas e as técnicas que possibilitaram a construção do sistema de pesquisa.

No Capítulo 4 abordaremos o suporte que a arquitetura dá para trabalhar utilizando boas práticas de desenvolvimento ágil de software e as técnicas e ferramantas utilizadas para a construção do sistema que originou a arquitetura.

No Capítulo 5 apresentamos oportunidades de melhoria para fazer com que a arquitetura seja capaz de produzir softwares mais completos, abrangentes e com mais qualidade.

No Capítulo 6 mostraremos os resultados do trabalho, as novas perspectivas da arquitetura bem como a contribuição para a saúde pública através do apoio à luta contra a mortalidade infantil devido a pneumonia.

CAPÍTULO 2

Sistema

Apresentaremos aqui o sistema que deu origem à arquitetura. Serão apresentados, através da UML [40] [27] [12] [18], diagramas de alto nível que indicarão a base da construção da arquitetura. As questões tecnológicas serão abordadas individualmente nos capítulos seguintes.

2.1 Atores

O sistema que motivou a construção dessa arquitetura possui quatro atores. O Administrador, o Analista de Saúde, O Agente de Saúde e o Radiologista.

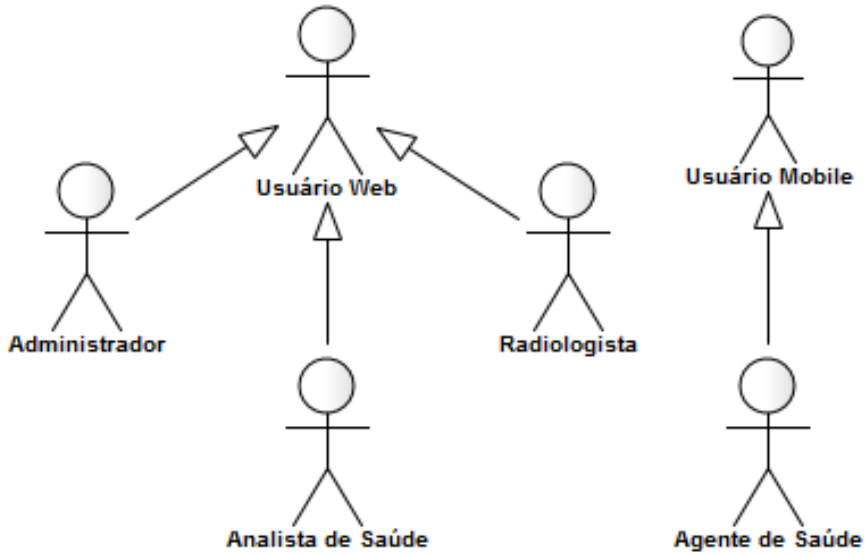

Figura 2.1: *Atores do sistema web, atores do sistema mobile e suas hierarquias.*

Cada usuário do sistema pode assumir um ou mais desses papéis. Uma pessoa que é Analista de Saúde pode perfeitamente ser também um Administrador.

2.1.1 Administrador

O Administrador é responsável pelo acesso ao sistema. Lembrando que o sistema pode ser acessado pelo celular e pela web através da internet. Os usuários podem eventualmente ser desligados e o acesso deve ser cortado. O sistema possui dados de pacientes e a confidencialidade é um dos requisitos mais fortes do sistema. Geralmente o Administrador é um auxiliar de confiança que é funcionário da instituição de saúde.

2.1.2 Analista de Saúde

O Analista de Saúde é um profissional que conhece muito do processo de coleta de informações, de pesquisa em campo e da rotina dos Agentes de Saúde. Ele é responsável pela criação dos formulários/questionários que serão usados em campo e que gerarão as informações para análise. O Analista de Saúde possui o conhecimento do negócio. É função dele verificar a qualidade das informações que estão sendo inseridas no sistema. Se necessário correções serão feitas por ele. Para a coleta de informações são necessárias definições de regras que garantam a integridade dos dados e que a pesquisa não será afetada.

O Analista de Saúde analisará estatisticamente os dados coletados através de outro software. No IPTSP-UFG o Analista de Saúde exporta o banco de dados do sistema no formao CSV (compatível com o Microsoft Excel) e o importa num software de análise e visualização de dados. A exportação do banco de dados formatado é uma das funcionalidades do sistema.

2.1.3 Agente de Saúde

O Agente de Saúde é a pessoa que, junto com o paciente coleta as informações através de formulários contendo questionários da pesquisa. Ele entrevista o responsável pelo paciente. Já que o paciente é uma criança com menos de 5 anos de idade. É o ator que mais usa o sistema. O sucesso da pesquisa depende do desempenho dele e da qualidade dos dados que ele coleta. Geralmente é uma pessoa da área de saúde. Um enfermeiro. Que fica num hospital aguardando possíveis casos de pneumonia infantil. O paciente chega, ele coleta os dados através de um celular e em seguida os envia para o banco de dados central pela internet do celular.

2.1.4 Radiologista

O Radiologista é um ator que pouco participa do sistema mas fornece a informação mais preciosa. O paciente está o não com pneumonia. Ele realiza esse diagnóstico através da web acessando um caso que já foi enviado e está disponível no banco de dados.

O Agente de Saúde junto com as informações envia para o banco de dados central a imagem da radiografia do paciente que ele tirou com o próprio celular. É essa imagem que será avaliada pelo Radiologista. Ele deve também atestar a qualidade da imagem quando necessário. Para realizar o laudo, ele terá acesso à poucas informações do paciente para não influenciar no resultado.

2.2 Processo

2.2.1 Fluxo

O fluxo do processo de negócio, em alto nível, pode ser descrito da seguinte forma:

1. O Administrador define os usuário do sistema.
2. O Analista de Saúde criar os formulários que serão usados na pesquisa. Os formulários são questionários com seções, perguntas e alternativas de resposta. Essa atividade é feita no começo do estudo e então sofre poucas modificações durante o perío ro de pesquisa.
3. O Agente de Saúde, de posse do celular com o aplicativo já instalado, entra no sistema com seu usuário e senha.
4. O Agente de Saúde, por questões de segurança, altera a senha para uma particular que só ele saiba.
5. O Agente de Saúde, assim que aparece um paciente com suspeita de pneumonia, abre um caso no celular. Cada paciente suspeito vira um caso.
6. A medida que as informações forem adquiridas o Agente de Saúde as registra num caso no celular. Esse registro é importante porque ele grava fisicamente o caso no celular. Se a bateria do celular acabar, por exemplo, ele ainda estará disponível.
7. Ao término da coleta de informações do paciente o Agente de Saúde envia o caso para o banco de dados central através da internet do celular. Todos os casos precisam ser enviados.
8. Se necessário, o Analista de Saúde corrige ou complementa as informações de um caso. Açõa feita com frequência.
9. Após um caso ter sido considerado como pronto, o Radiologista realiza o laudo. Informando se o paciente está ou não com pneumonia. Essa atividade é feita frequentemente. Todos os casos devem ser laudados.
10. Por fim, o Analista de de saúde obtém o banco de dados através da web num formato de planilha CSV para realizar análises estatísticas em outros softwares.

2.2.2 Diagrama

Esses passos estão também representados através de um diagrama de sequência da UML. Veja figura 2.2.

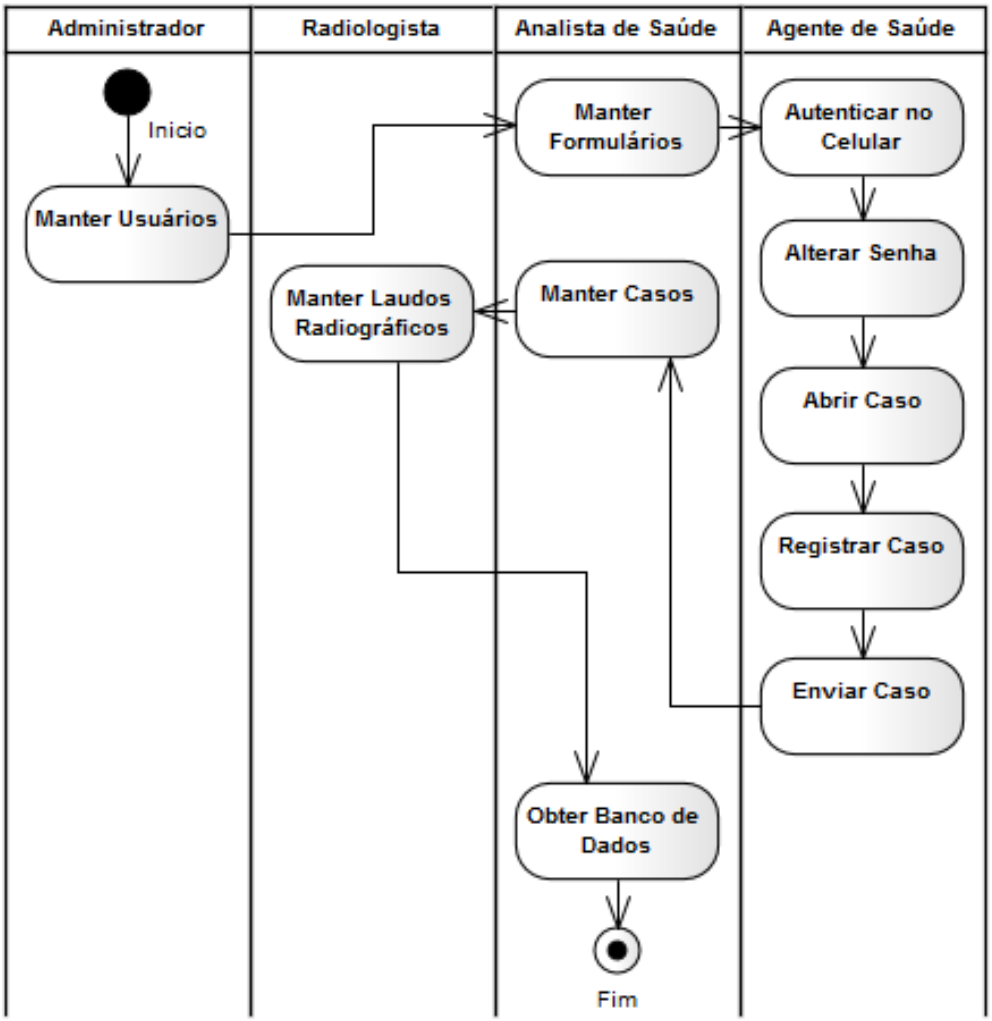

Figura 2.2: *Processo do Sistema de Pesquisa das Pneumonias na Infância com atividades de alto nível e atores responsáveis.*

2.3 Domínio

O domínio do negócio pode ser descrito através de 8 conceitos: Caso, Laudo, Questionário, Seção, Pergunta, Alternativa, Resposta e Usuário.

Esses conceitos estão representados num diagrama de classe da UML. Veja figura 2.3.

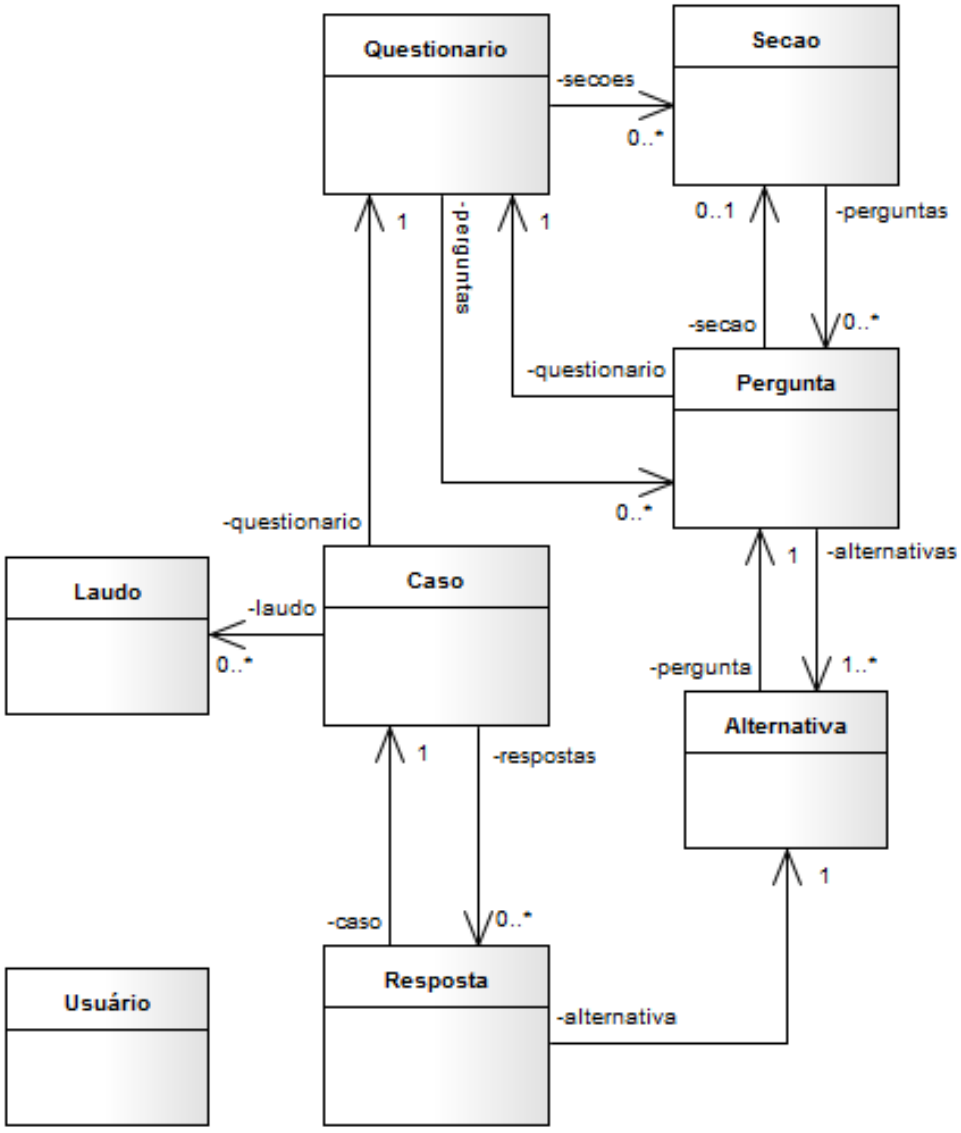

Figura 2.3: *Domínio do negócio destacando apenas os principais conceitos.*

2.3.1 Caso

Todas as informações de um paciente com suspeita de pneumonia coletadas por um Agente de Saúde são definidas como “Caso”.

Um caso está associado a um e somente um questionário. Não existe caso sem questionário. As respostas das perguntas do questionário ficam associados ao caso. Além do questionário, o caso possui algumas imagens. Nesse sistema a imagem do raio x e duas fotos do cartão de vacina. Frente e verso. Para ajudar nas evidências se o caso, por algum motivo, for auditado. O laudo realizado pelo Radiologista também fica associado ao caso.

2.3.2 Laudo

Laudo é o resultado da análise radiográfica feita pelo Radiologista. Pode ser negativo, ou seja, o raio x está normal. Ou positivo, o paciente encontra-se com alguma enfermidade ou o raio x possui algum problema. O Radiologista deve registrar no laudo as conclusões que ele conseguiu tirar. Todas essas conclusões ajudarão nas análises estatísticas.

2.3.3 Questionário

É o conjunto de perguntas realizadas numa pesquisa. Geralmente, cada pesquisa possui apenas um questionário. O questionário possui seções que, no caso do IPTSP-UFG os analistas chamam de "formulários". Essas divisões são importantes para auxiliar o Agente de Saúde durante o preenchimentos dos formulários. Antes de o questionário ser confeccionado os analistas de saúde estudam quais variáveis deverão entrar no estudo. Que informações serão coletadas para atingir o objetivo da pesquisa. Qual a ordem mais importante das perguntas e o formato dos dados de entrada. Esse trabalho é fundamental para o sucesso da pesquisa e só pode ser feito por um time de especialistas em epidemias.

2.3.4 Seção

Seção, também chamado de formulário, é uma divisão lógica das perguntas de um questionário. Essa divisão é mais que simplesmente um organizador. Ela possui regras de transição. Por exemplo, o formulário 1 é sobre "Elegibilidade" e o formulário 2 é sobre "Dados Demográficos", no celular só aparecerá o formulário 2 se a resposta para a pergunta "Criança é elegível?" for igual a "Sim". Essa estratégia de transição facilita muito o trabalho do Agente de Saúde e diminui muito os erros e informações desnecessárias além de acelerar muito o processo.

2.3.5 Pergunta

É um questão do questionário que precisa ser respondida durante a entrevista feita pelo Agente de Saúde. A pergunta pode ser um item para ser escolhido como "Sexo": "Masculino" ou "Feminino". Ou um campo livre. Como nome da criança. Pode ter regra para aparecer ou não numa seção do questionário conforme respostas de perguntas de seções anteriores.

2.3.6 Alternativa

Uma das opções de uma pergunta. A alternativa pode ter uma regra para validação. Algumas dessas regras são verificadas no próprio celular outras no servidor durante o processo de envio de caso. As regras de validação podem envolver mais de uma alternativas de outras questões. Por exemplo, não se pode informar o o bairro se a cidade não tiver sido informada anteriormente. Um campo livre numa pergunta, na verdade, é uma única alternativa com entrada de dados livre.

2.3.7 Resposta

É o dado informado para uma pergunta. As respostas que serão efetivamente enviadas para o banco de dados. Cada resposta está associada a uma alternativa e assim está ligada a um questionário. Ela também está ligada ao caso.

2.3.8 Usuário

São pessoas que podem acessar o sistema. Existem usuários que podem acessar o aplicativo no celular e usuários que podem acessar o sistema na web. O que determina o acesso é o perfil associado ao usuário. As senhas dos usuários não são armazenadas. Por questões de segurança apenas um hash composto de outras informações é armazenado. Assim, é possível verificar se uma senha está correta sem ter que comparar seu conteúdo literalmente.

2.4 Casos de Uso

Aqui estão os casos de uso do sistema utilizados pelo Administrador, Analista de Saúde, Agente de Saúde e Radiologista.

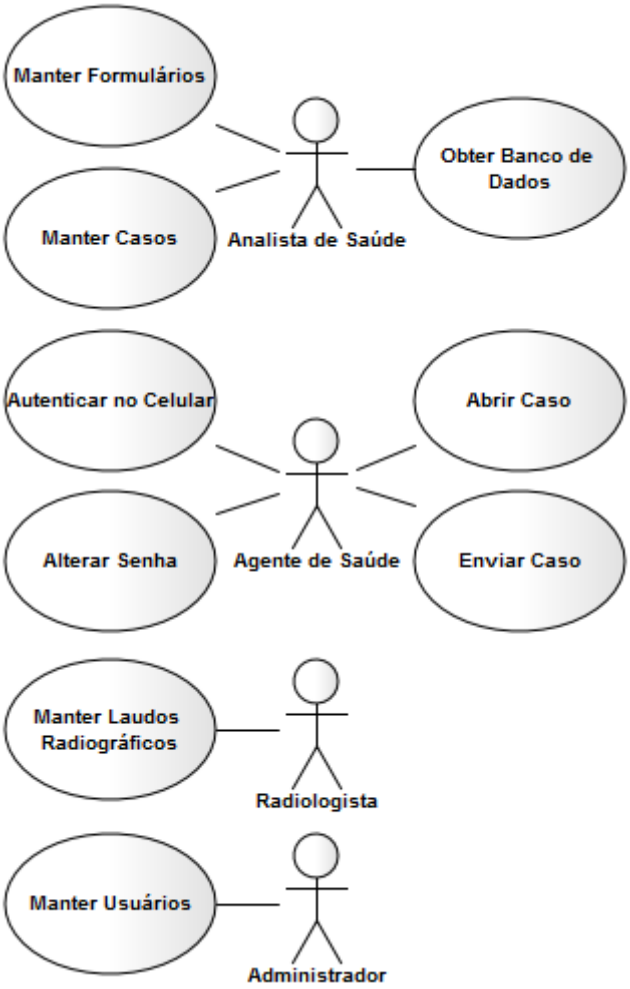

Figura 2.4: *Casos de uso do sistema.*

2.4.1 Manter Usuários

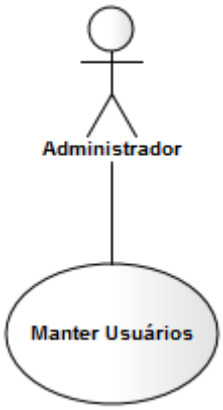

Figura 2.5: *Diagrama de casos de uso indicando a relação entre o ator Administrador e o caso de uso Manter Usuários.*

O Administrador inclui, altera, excluir, ativa, inativa e lista os usuários do sistema. Cada usuário possui um ou mais perfis. Os perfis são equivalentes aos atores.

Veja em 2.1. O usuário possui uma senha. Ela não é armazenada no banco de dados mas sim um valor (hash) derivado dela. Com isso nem mesmo os que administram o sistema conseguem ter acesso às senhas dos usuários olhando no banco de dados.

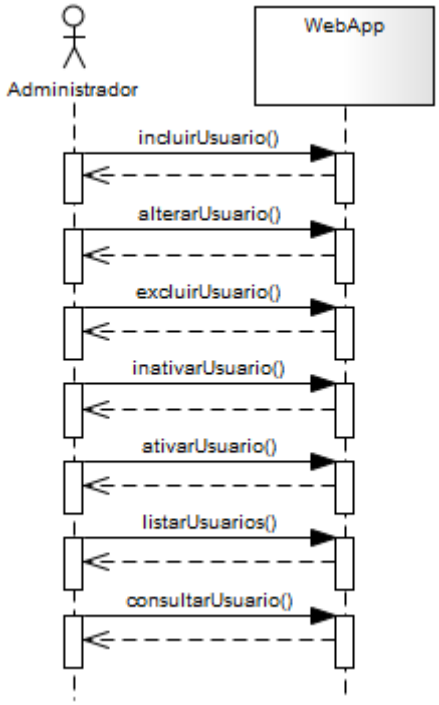

Figura 2.6: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Administrador e o sistema no caso de uso Manter Usuários.*

Os usuários do sistema precisam ser mantidos com frequência, dada a rotatividade de agentes de saúde no projeto. E como o banco de dados possui informações de pacientes, sigilo e confidencialidade são requisitos muito fortes. Se um Agente de Saúde sair do projeto, ele deve ser inativado o quanto antes.

Algumas operações gravam o usuário que a realizou. Portanto tanto no celular quanto na web, o tempo todo existe um objeto representando o usuário corrente. Dependendo da operação, o usuário corrente é passado como parâmetro.

2.4.2 Manter Formulários

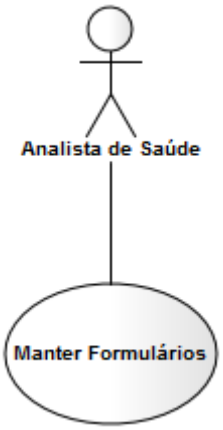

Figura 2.7: *Diagrama de casos de uso indicando a relação entre o ator Analista de Saúde e o caso de uso Manter Formulários.*

Os formulários são, na verdade, um conjunto de perguntas e suas alternativas divididas por seções. Ou seja são questionários. O questionário é construído para atender uma necessidade de pesquisa. No caso do IPTSP-UFG, para avaliar a efetividade de uma vacina. É de responsabilidade do Analista de Saúde definir as seções, as perguntas, as alternativas, a ordem de apresentação e todas as regras que regem o funcionamento do questionário.

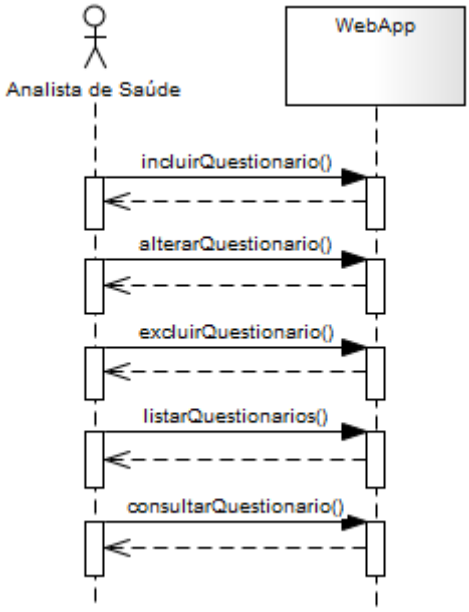

Figura 2.8: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Manter Formulários para o conceito Questionário.*

Cada questionário está associado a uma pesquisa. Portanto ele tem um nome, objetivo e um intervalo de execução. Os casos coletados só podem ser enviados se estiverem no período de validade do questionário. O Analista de Saúde pode incluir, alterar, excluir e listar questionários. No entanto qualquer alteração feita no questionário afetará os dados existentes. Remover um questionário significa remover todos os casos associados a ele. Então é de extrema importância o planejamento cuidadoso do questinário já que uma alteração pode ser desastrosa. Um questionário entrará em vigor e estará disponível para os Agentes de Saúde se ele estiver marcado como ativo. Questionários inativos não aparecerão para os agentes nos celulares e não receberão casos.

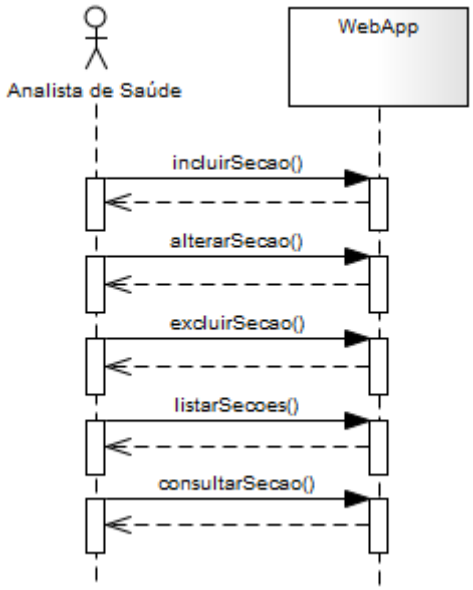

Figura 2.9: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Manter Formulários para o conceito Seção.*

As seções, quando não estão associadas a nenhuma regra pouco impactam na execução de um questionário. Elas são até opcionais. No entanto são ótimas ferramentas de organização e de preenchimento condicional. Por exemplo, uma seção pode ter uma regra associada a ela definindo que todas as suas perguntas só aparecerão se a pergunta “Data de nascimento” for preenchida. Esse tipo de recurso pode fazer com que o preenchimento dos questionários fique bastante otimizado fazendo com que os Agentes de Saúde façam seu trabalho mais rápido e de forma mais precisa. As seções de um questionário podem ser incluídas, excluídas, alteradas e listadas. Uma seção não pode ser reutilizada por outro questionário. Como as seções podem interferir no funcionamento dos formulários nos celulares sua manutenção deve ser realizada com muito cuidado.

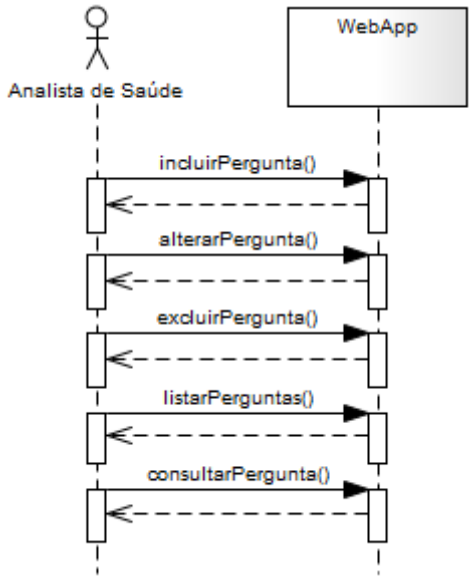

Figura 2.10: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Manter Formulários para o conceito Pergunta.*

O Analista de Saúde, durante a construção do questionário, pode incluir, excluir, alterar e listar perguntas. As respostas das perguntas é que formarão os dados que serão analisados ao final do processo. A alteração de uma pergunta pode ter muito impacto. As perguntas possuem um status “ativo” ou “inativo”. Se estiver “ativo” a pergunta aparece no aplicativo do celular, se estiver “inativo” não aparece e as regras associadas a ela são ignoradas. Campo obrigatório, por exemplo. Talvez seja melhor inativar uma pergunta do que removê-la. Remover uma pergunta significa remover todas as respostas de todos os casos. Se a alteração de uma pergunta mudar seu sentido, é melhor inativá-la e criar outra.

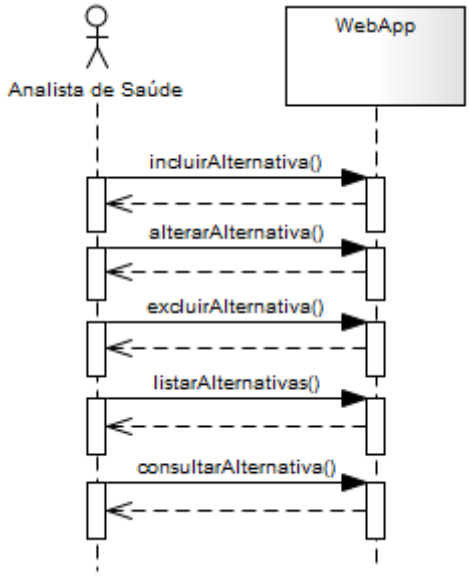

Figura 2.11: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Manter Formulários para o conceito Alternativa.*

Uma alternativa é um dos possíveis valores que podem ser escolhidos como opções de uma pergunta. Uma pergunta com apenas uma opção livre, nome da criança por exemplo, também possui alternativa. Não existem perguntas sem alternativas. Uma alternativa não pode ser reaproveitada em outras perguntas. Ela possui um valor que será gravado no banco de dados e uma descrição que aparecerá para o usuário.

2.4.3 Autenticar no Celular

Os Agentes de Saúde utilizam o celular para coletar e enviar casos de pacientes hospitalizados com suspeita de pneumonia infantil. Cada Agente de Saúde possui um celular com o aplicativo de coleta e envio de casos previamente instalado pela equipe de suporte da pesquisa.

No entanto, o mesmo celular pode ser repassado para outro Agente de Saúde. Ele não é de acesso exclusivo. Com isso, para usar o sistema, não basta ter o celular e o programa instalado. É necessário uma identificação e uma senha para efetuar o login.

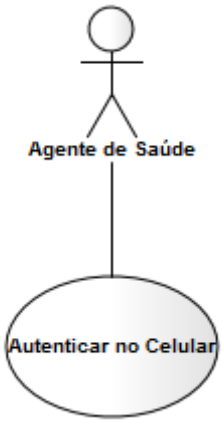

Figura 2.12: *Diagrama de casos de uso indicando a relação entre o ator Agente de Saúde e o caso de uso Autenticar no Celular.*

Além do motivo de um celular ser utilizado por mais de Agente de Saúde, ao enviar um caso o sistema web identifica qual Agente de Saúde está realizando a operação. Essa informação, o autor, é enviada junto com o caso e fica gravada com ele no banco de dados central. Assim é possível, na análise estatística, verificar desempenho e falhas por Agente de Saúde, por exemplo.

O login precisa acontecer com o celular online ou offline. É muito comum nos hospitais, em alguns ambientes, não pegar sinal da operadora e um caso não pode deixar de ser coletado por não haver sinal com a operadora de celular.

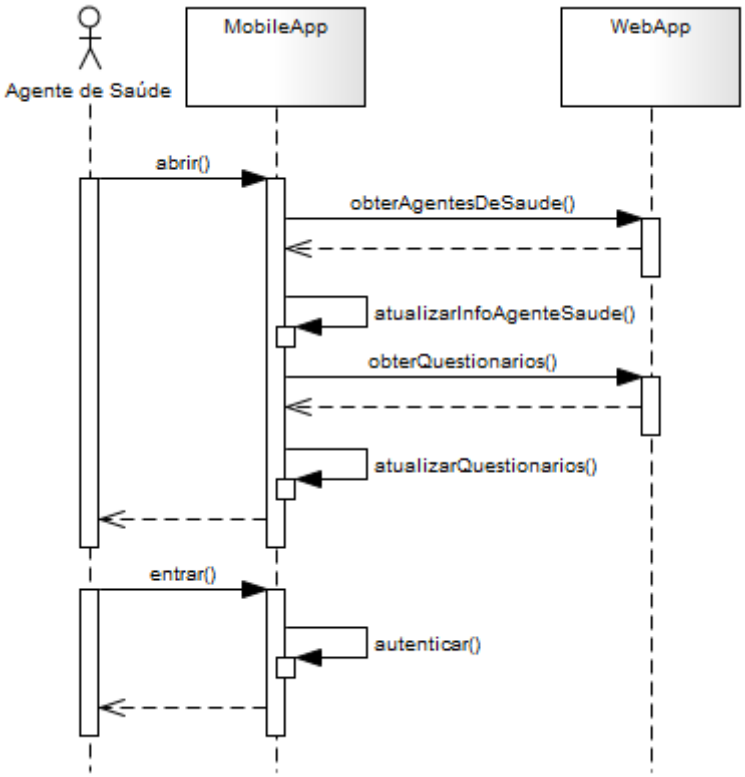

Figura 2.13: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Agente de Saúde e o sistema no caso de uso Autenticar no Celular.*

Dessa forma ao abrir o aplicativo no celular ele tenta estabelecer uma conexão com o servidor web. Caso a conexão seja bem sucedida, o aplicativo no celular solicita a lista de todos os agentes de saúde cadastrados no sistema. Essa lista não é grande e, portanto, a operação demora por volta de dois segundos. Essa lista possui informações como a identificação do agente (“Usuário”) e o hash de sua senha (não a senha literal é apenas uma informação derivada que possibilitará realizar verificação). Então a lista é armazenada no banco de dados local do celular para realização de autenticação offline. Em seguida o aplicativo solicita a lista dos questionários disponíveis no sistema web. Para que o agente possa saber, após autenticar, quais pesquisas ele pode realizar. Essa lista também é armazenada no banco de dados local do celular. Se a conexão com o servidor web não for bem sucedida (por falta de sinal do celular, por exemplo) não acontecerá nenhum erro. O fluxo segue normalmente e então aparecerá a tela de login. Veja figura 2.14.

Figura 2.14: *Tela do sistema do celular utilizada pelo Agente de Saúde para autenticação.*

Na tela de login aparecerão dois campos: “Usuário"e “Senha". O Agente de Saúde preencherá os campos e então solicitará entrar no sistema. Veja botão “Entrar"na figura 2.14. Nesse momento a autenticação acontecerá offline e o agente terá acesso ao menu do sistema. Veja figura 2.15.

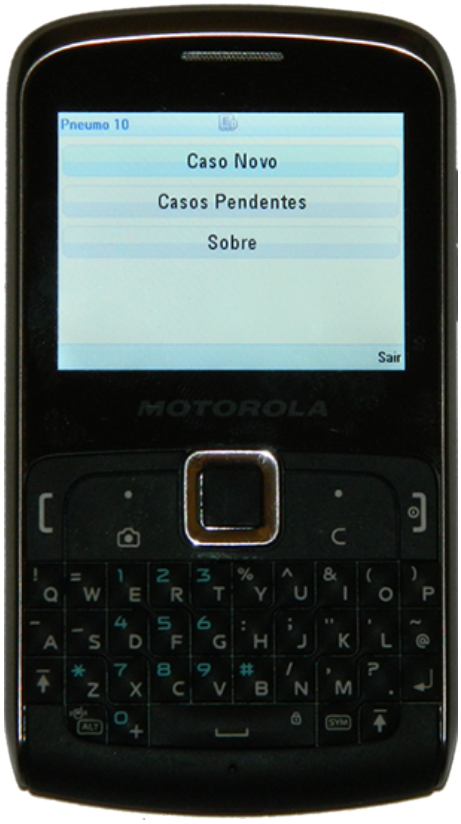

Figura 2.15: *Tela do sistema do celular com as opções do menu da aplicação utilizada pelo Agente de Saúde.*

2.4.4 Alterar Senha

Uma das maiores preocupações do projeto é com a segurança. Os dados de pacientes devem, por lei, estar sob sigilo e confidencialidade. Sendo assim, é necessário que a autenticação do agente de saúde seja a mais segura possível, pois ele é quem coleta e envia para o banco de dados central os dados dos pacientes.

Os usuários inicialmente são cadastrados no sistema web com uma senha padrão. A senha literal não é armazenada no banco de dados e sim um valor gerado a partir dela para verificação durante autenticações. Em seguida, no primeiro acesso do agente de saúde, o sistemas do celular pede para o usuário substituir a senha por uma nova.

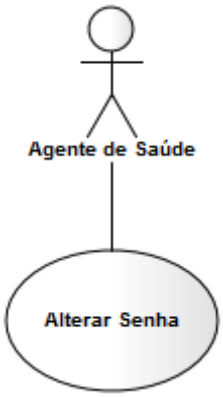

Figura 2.16: *Diagrama de casos de uso indicando a relação entre o ator Agente de Saúde e o caso de uso Alterar Senha.*

Nesse instante, o usuário informa a senha antiga, informada a ele pelo administrador do sistema, informa a nova senha e em seguida solicita alteração de senha para o sistema. O sistema no celular gera um valor a partir da nova senha digitada e envia uma solicitação de alteração de senha para o sistema no servidor passando a ele o valor gerado e não a nova senha digitada. Durante o processo, o celular deve estar conectado à internet para possibilitar comunicação com o servidor. Então no primeiro acesso o agente de saúde precisa ter acesso à internet do celular. As próximas autenticações podem ser realizadas offline.

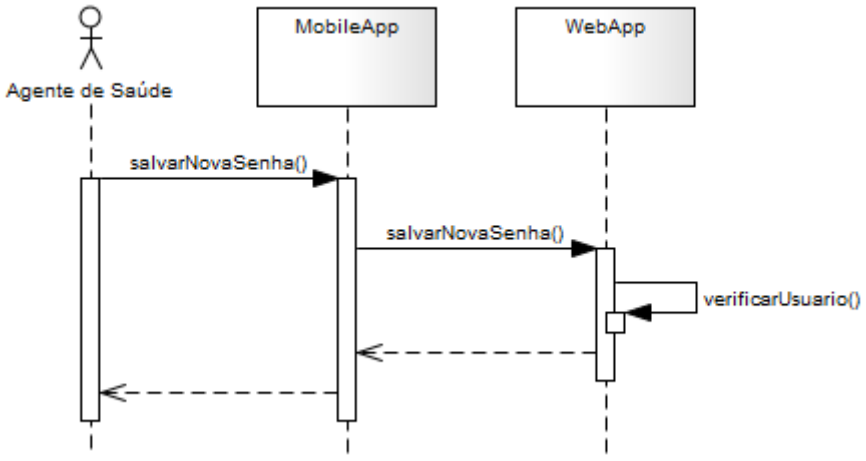

Figura 2.17: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Agente de Saúde e o sistema no caso de uso Alterar Senha.*

2.4.5 Abrir Caso

Esse é o início da funcionalidade mais utilizada em todo o sistema. Diariamente os Agentes de Saúde coletam informações de dezenas de pacientes. É pela coleta que as

informações mais importantes da pesquisa são obtidas. As coletas acontecem offline. O Agente de Saúde não precisa ter acesso à internet do celular para realizar as entrevistas. A entrevista acontece em passos e em momentos distintos. Podendo até começar num dia e terminar dias depois. Então o ato de abrir um caso inicia esse processo de coleta de dados de um paciente.

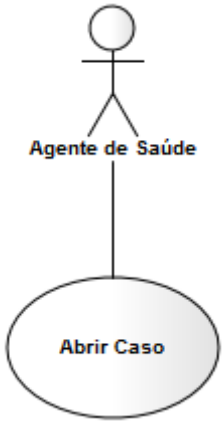

Figura 2.18: *Diagrama de casos de uso indicando a relação entre o ator Agente de Saúde e o caso de uso Abrir Caso.*

A partir do menu principal do aplicativo para celular (veja figura 2.15) o agente de saúde escolhe a opção para começar um novo caso. Em seguida o aplicativo realizar algumas validações como verificação de espaço físico para criar um novo caso e permissão para manter mais um caso aberto. Por questões de segurança e organização, um agente de saúde não pode manter muitos casos abertos sem enviar para o servidor. Isso educa o agente de saúde para que ele siga o processo corretamente. Se ele mantiver vários casos no celular e ele o perder, por exemplo, todos os casos não enviados se perderão e isso é inadimissível.

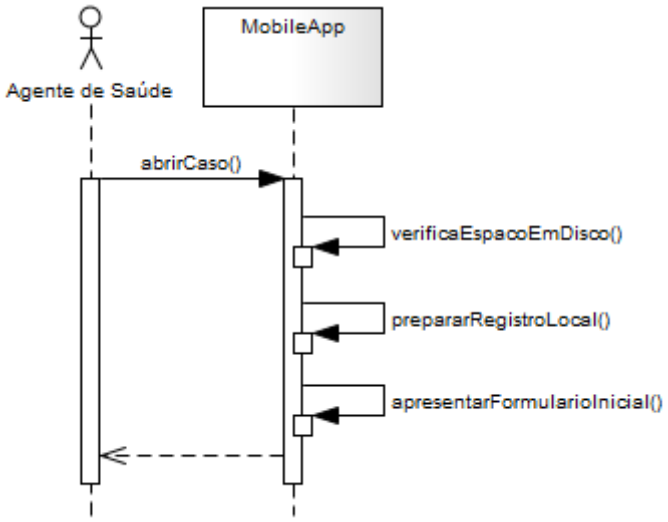

Figura 2.19: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Agente de Saúde e o sistema no caso de uso Abrir Caso.*

Por fim, o aplicativo abre a tela com o primeiro formulário para o agente realizar a entrada de dados. Veja figura 2.25.

2.4.6 Registrar Caso

O registro do caso acontece durante a entrevista com o paciente e/ou o responsável por ele. A cada nova etapa da entrevista os resultados parciais são gravados no celular e enviados para o banco de dados central através da internet do celular.

O registro parcial do caso é importante para que os analistas de saúde possam acompanhar a evolução das coletas e para proativamente tomarem ações afim de manterem a qualidade da pesquisa. Com esses resultados parciais, por exemplo, os gestores da pesquisa conseguem saber o tempo médio entre o inicio de uma coleta e o seu término.

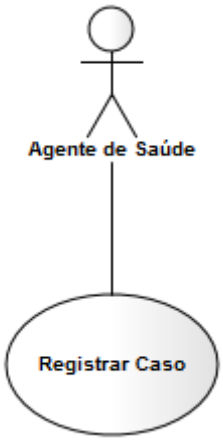

Figura 2.20: *Diagrama de casos de uso indicando a relação entre o ator Agente de Saúde e o caso de uso Registrar Caso.*

A cada entrada de dados num campo do formulário os informações são gravadas temporariamente no celular para que não se percam em caso de pane no aparelho, por exemplo. Mas a cada formulário (veja figura 2.25) concluído, acontece o registro do caso. Esse registro compreende a gravação de todos os dados do formulário corrente no banco de dados do celular, envio dos dados do formulário para o servidor e a gravação dos dados do formulário no banco de dados do servidor.

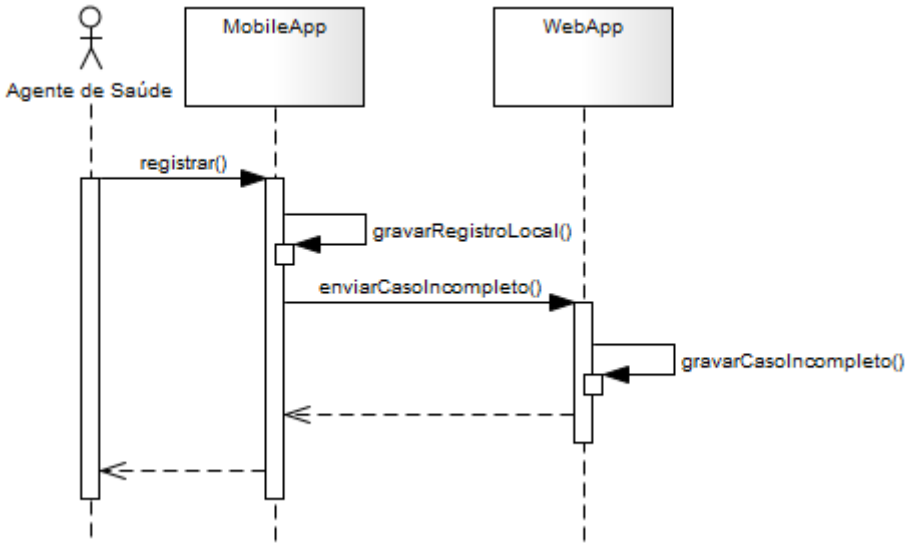

Figura 2.21: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Agente de Saúde e o sistema no caso de uso Registrar Caso.*

Esse caso registrado ainda está incompleto e seu estado é definido como tal. Dessa forma os analistas de saúde o excluem da análise de dados.

Figura 2.22: *Tela de entrada de dados para o primeiro formulário de um caso.*

2.4.7 Enviar Caso

Ao concluir a entrada de dados no aplicativo do celular, acontece o envio do caso. Uma vez concluído o processo de envio, o caso fica disponível para análise dos dados e manutenções eventuais. Nesse ponto o trabalho do agente de saúde termina para este caso.

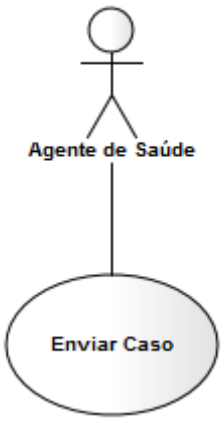

Figura 2.23: *Diagrama de casos de uso indicando a relação entre o ator Agente de Saúde e o caso de uso Enviar Caso.*

Ao enviar, veja figura 2.25, acontece uma validação dos dados no próprio

celular. Essa validação é preliminar, uma vez que a validação completa só pode ser feita no servidor. Estamos falando de campos obrigatórios, formato de datas, entre outras pequenas validações.

Se tudo estiver validado, o caso é transmitido para o servidor pela internet do celular. No servidor, acontece outra validação. Todas as que são necessárias. E então o caso é gravado no banco de dados do servidor.

Após o envio, o aplicativo do celular envia uma nova requisição para o servidor para obter informações sobre os agentes de saúde do sistema e sobre evoluções nos questionários. Essas informações serão úteis para o funcionamento do aplicativo no celular.

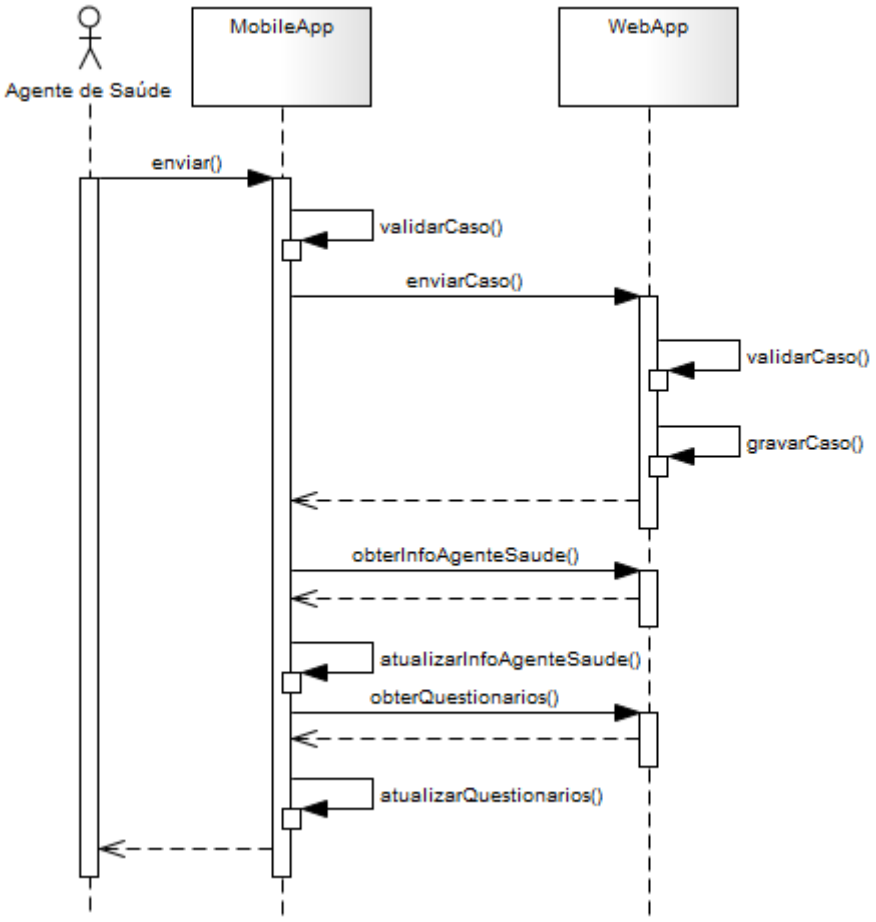

Figura 2.24: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Agente de Saúde e o sistema no caso de uso Enviar Caso.*

Essas outras requisições acontecem nesse ponto por conveniência. Nesse instante o agente de saúde reserva um momento e um local com bom sinal de celular para poder enviar o caso. Essas condições são também ideais para realizar atualizações de

informações no celular, pois ele pode ficar bastante tempo offline. Essas requisições secundárias não interferem no envio do caso em situações de falha.

Figura 2.25: *Tela de envio de caso.*

2.4.8 Manter Casos

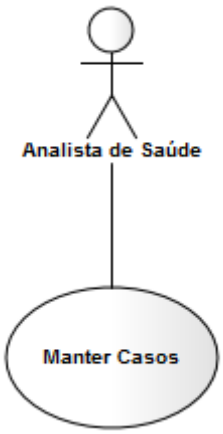

Figura 2.26: *Diagrama de casos de uso indicando a relação entre o ator Analista de Saúde e o caso de uso Manter Casos.*

Num fluxo normal, um caso será criado no celular, enviado e depois os dados daquele caso serão avaliados estatisticamente. No entanto, é comum o Analista de Saúde

precisar incluir, alterar e excluir casos. Existem casos que são enviados por celular mas as fotos não. Então o caso precisa ser alterado para receber as fotos que são enviadas por e-mail, por exemplo. Outra situação é quando um Agente de Saúde, por algum motivo, preenche um caso, ou parte dele, utilizando uma ficha impressa de papel. Então essa ficha será inserida no sistema através do caso de uso Manter Caso.

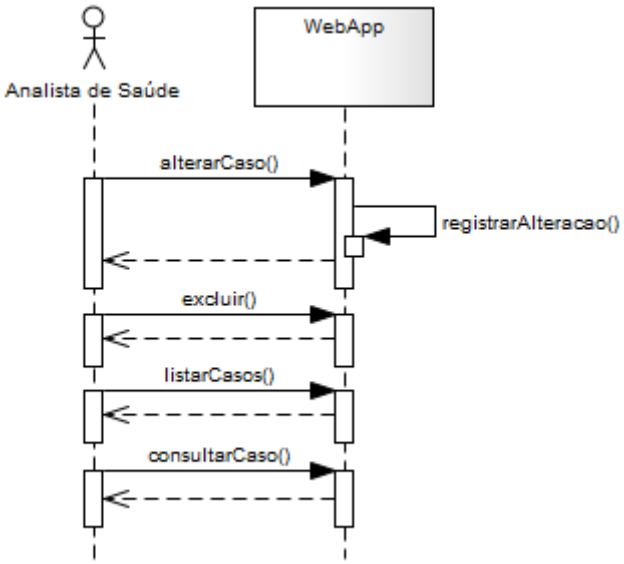

Figura 2.27: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Manter Casos.*

Deve-se usar essas funcionalidades de manutenção de casos apenas em condições especias. Ou então a qualidade da pesquisa pode ser afetada. O "correto" é seguir o fluxo normal, partindo do Agente de Saúde e transmitido pelo celular para o sistema web.

A exclusão de caso acontece em situações raras como em testes e casos enviados acidentalmente. Todas as informações são importantes para a pesquisa. Inclusive casos de pacientes que podem não ser elegíveis a avaliação. Essas informações no mínimo servirão para melhorar o processo de coleta no futuro.

2.4.9 Manter Laudos Radiográficos

Os laudos radiográficos são uma particularidade desse sistema. Outras funcionalidades como manutenção de questionários e manutenção de usuários aplicam-se a várias situações. Laudar radiografias por um radiologista fez com que a arquitetura tivesse preocupações que outras não teriam. Como baixa largura de banda para manipulação de imagens de alta definição.

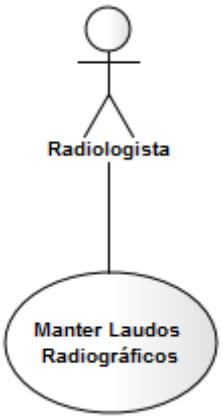

Figura 2.28: *Diagrama de casos de uso indicando a relação entre o ator Radiologista e o caso de uso Manter Laudos Radiográficos.*

Apenas os radiologistas têm acesso à essa funcionalidade. Na prática, tendo como referência o IPTSP-UFG, existem mais de um laudando os casos. Eles fazem remotamente de onde estiverem. No computador, tablet ou qualquer dispositivo que tenha um navegador web compatível com HTML5.

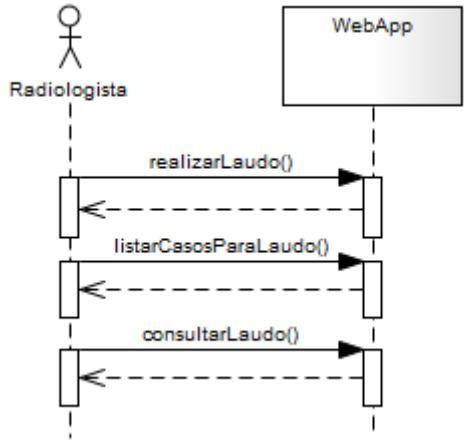

Figura 2.29: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Radiologista e o sistema no caso de uso Manter Laudos Radiográficos.*

Para que os laudos pudessem ser realizados, fotos de alta resolução precisaram ser tiradas das radiografias em condições ideais. Portanto um procedimento foi passado para os Agentes de Saúde para que eles tirassem as fotos adequadamente considerando o celular ou máquina fotográficas, luminosidade e condições do raio x.

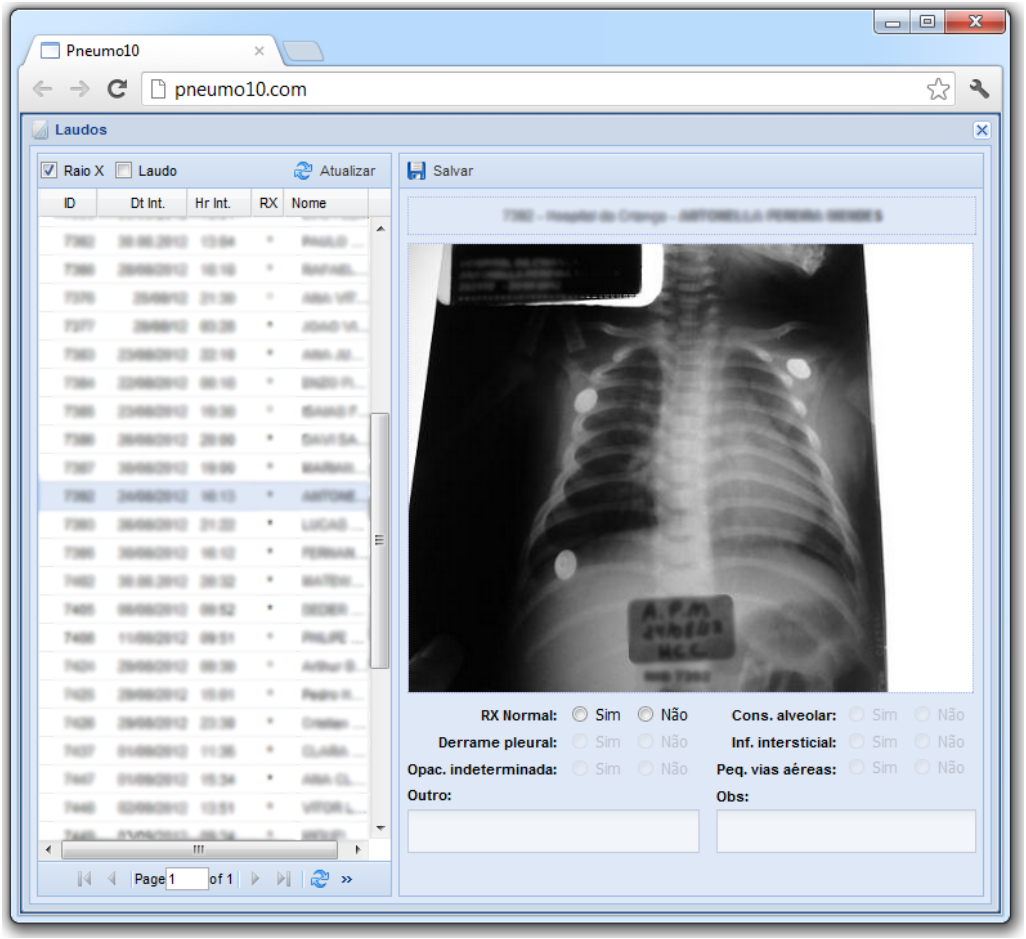

Figura 2.30: *Tela do sistema utilizada pelo Radiologista para realizar laudos em casos que possuem raio x.*

Ao clicar na imagem pequena à direita (veja figura 2.30) ela maximizará com toda sua resolução para melhor atender as exigências do Radiologista (veja figura 2.31).

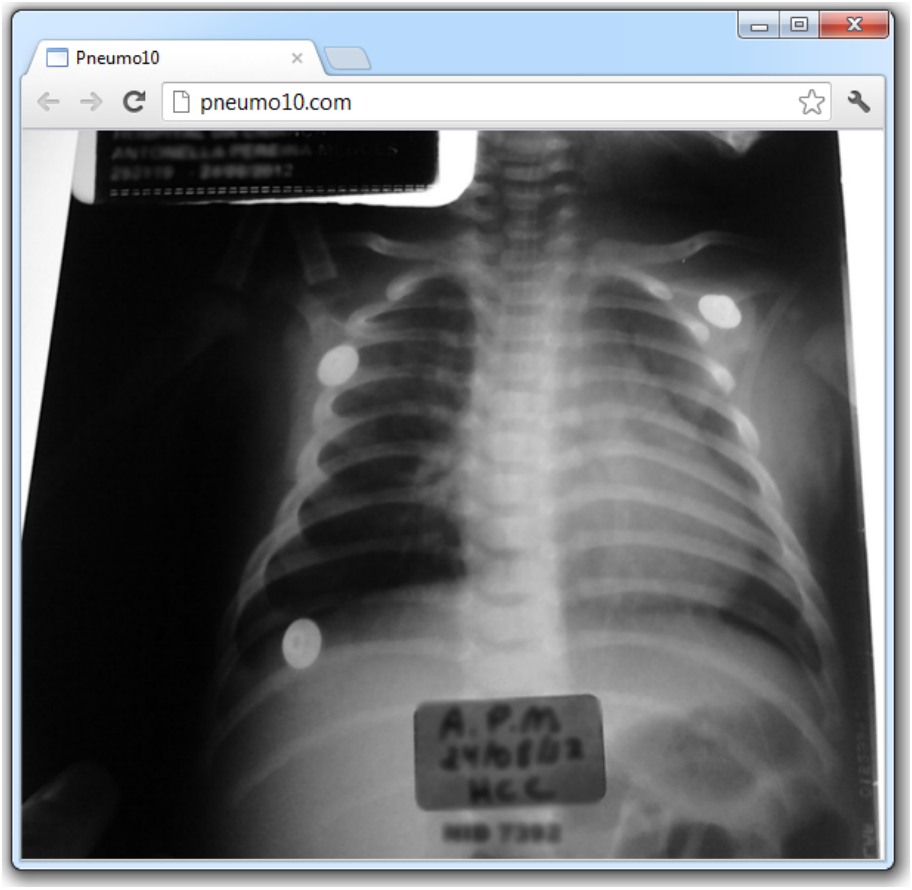

Figura 2.31: *Tela do sistema utilizada pelo Radiologista para visualização de raio x com toda a resolução disponível.*

No IPTSP-UFG as enfermeiras tiram fotos com 3 megapixels de resolução (2.048 pixels de largura por 1.536 pixels de altura). Essa resolução é o suficiente para realizar laudos. Acontece de a radiografia aparecer invertida ou com baixa qualidade devido a falta de cuidado do Agente de Saúde. Nesses casos o Radiologista volta para a tela de laudo e coloca a observação sobre as condições em que a imagem se encontra. Mesmo depois de o caso ter sido enviado para o banco de dados é possível alterar a imagem através do caso de uso Manter Caso.

2.4.10 Obter Banco de Dados

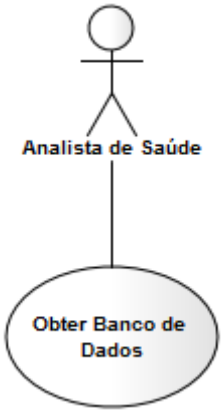

Figura 2.32: *Diagrama de casos de uso indicando a relação entre o ator Analista de Saúde e o caso de uso Obter Banco de Dados.*

A análise estatística dos dados não é feita pelo sistema. Ela é realizada por outros softwares. Então foi necessário criarmos uma funcionalidade que exportasse o banco de dados num formato que promovesse a interoperabilidade entre softwares. O formato adotado foi o Comma Separated Value (CSV) [11]. Ele resolvia a necessidade de exportação de dados em formato texto mas não é um formato adequado para transferir imagens. Mas, para a análise estatística, as imagens não eram importantes.

O banco de dados nesse formato mostrou-se bastante pesado na transmissão. E como a transmissão é feita via HTTPS, e portanto consome recursos caros da nuvem, resolvemos compactar o banco a medida que ele era baixado. Como em streams. Apesar de o usuário acessar um endereço “http://. . . /banco-de-dados.zip"não existe um arquivo com esse nome no servidor mas sim um serviço que cria os bytes que serão trafegados em tempo de processamento. Se essa medida não tivesse sido tomada, precisaríamos mudar nosso plano de hosting na nuvem e assim o projeto ficaria menos viável e demoraria bem mais tempo.

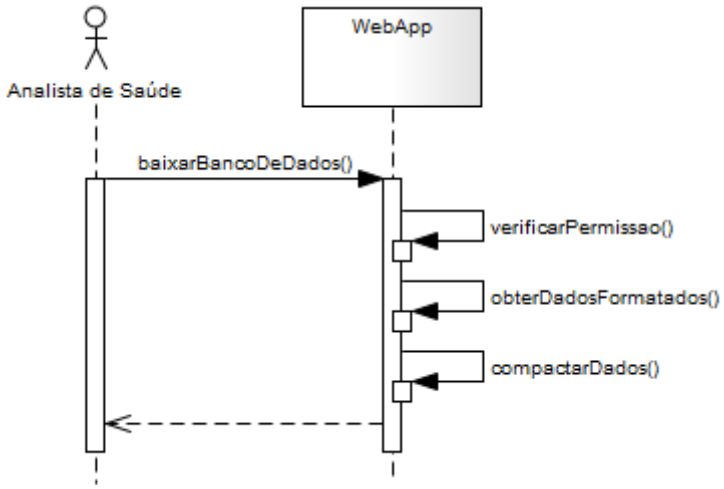

Figura 2.33: *Diagrama de sequência de alto nível indicando a troca de mensagens entre o ator Analista de Saúde e o sistema no caso de uso Obter Banco de Dados.*

O download do banco de dados é feito sempre que os dados estatísticos precisam ser analisados. Isso pode acontecer várias vezes ao dia. Então a otimização dos recursos envolvidos nessa operação foi muito importante.

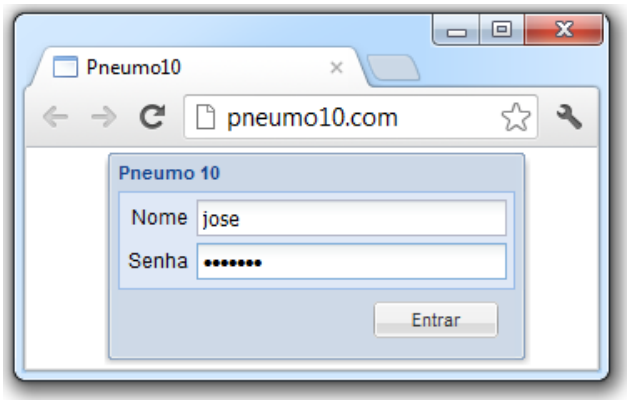

Figura 2.34: *Tela do sistema utilizada para, através de autenticação, entrar no sistema web.*

Essa funcionalidade é realizada da seguinte forma:

1. Um usuário do sistema web entra no sistema. Veja figura 2.34. Não é possível baixar o banco de dados sem antes entrar no sistema com um usuário e senha que tenha permissão para isso.
2. O usuário informa na barra de endereço do navegador a url para baixar o banco de dados. Veja figura 2.35.
3. Em seguida, começa o download do banco de dados como se fosse um arquivo comum. Veja figura 2.35. Durante o download o navegador não mostra a porcentagem para término do download. Isso acontece porque a medida que as informações são

lidas do banco de dados, linha a linha, elas são enviadas via HTTPS funcionando como stream. Rádios online funcionam dessa forma. Assim pouca memória é alocada no servidor e o tempo de processamento é o mínimo necessário.

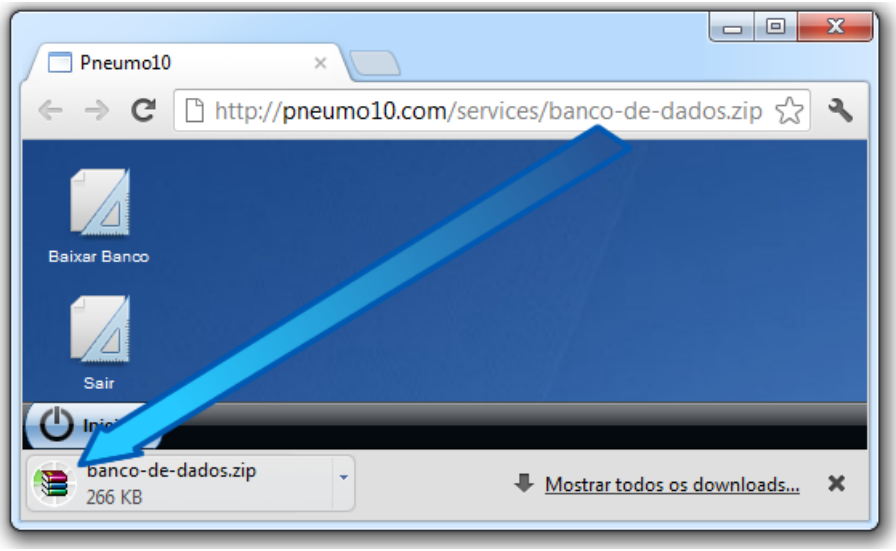

Figura 2.35: *Tela do sistema utilizada para baixar o banco de dados num formato específico para análise de informações estatísticas.*

CAPÍTULO 3

Arquitetura

A solução arquitetural consiste de um sistema dividido em dois softwares. Um aplicativo para celulares de baixo custo e uma aplicação web hospedada em nuvens também de baixo custo que receberá formulários digitais preenchidos por agentes de saúde em campo. A aplicação web foi desenvolvida para executar em nuvens baratas. Essas nuvens são infraestruturas de hardware remotas que podem estar localizadas e espalhadas em qualquer localidade geográfica. Dessa forma a solução completa terá alta flexibilidade, compatibilidade e um custo mínimo tanto de aquisição quanto de manutenção.

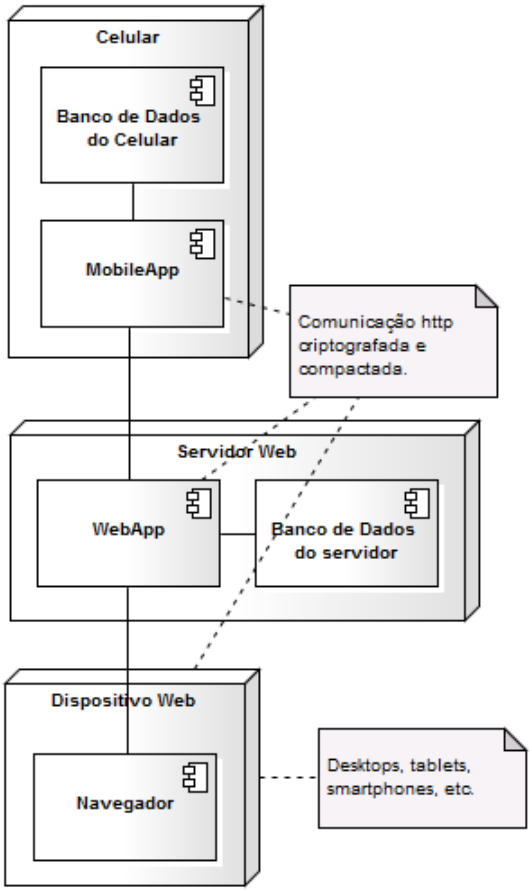

Figura 3.1: *Os componente que fazem parte da arquitetura e a relação entre eles.*

Realizamos estudos para adoção de tecnologias. Adotamos a plataforma Java para a maioria dos itens de software da arquitetura por acreditarmos que Java nos atendia bem na maioria das soluções. Desde o celular até o servidor de aplicações.

O aplicativo para celulares deveria ao mesmo tempo ser ágio, amigável e compatível com muitos celulares de baixo custo. Das tecnologias pesquisadas a que mais atendia foi a Java Micro Edition [32]. JME roda atualmente em mais de 3 bilhões de aparelhos no mundo [48]. No entanto para uma aplicação amigável precisaríamos da ajuda de um framework que fornecesse infraestrutura de interface com o usuário. A interface com o usuário foi uma das preocupações predominantes do projeto. Dos frameworks pesquisados o adotado foi o J2ME Polish [31]. Ele possui elementos gráficos e entradas de dados avançadas além de permitir alta customização da aparência da aplicação sem prejudicar a usabilidade. Esse framework oferece também ferramentas para auxílio na comunicação remota entre aplicações e na manipulação de dados em celulares que não possuem bancos de dados. Apesar de muito poderoso ele consome poucos recursos e possui alta compatibilidade com dispositivos móveis [65].

A aplicação web deveria ser robusta para suportar várias conexões simultâneas,

ter uma interface com o usuário muito boa e fácil de usar. Pesquisamos opções para o lado "client" e para o lado "server" do sistema.

Para a parte "client", resolvemos utilizar padrões web (HTML5) para garantir vida longa ao software, compatibilidade e pouco consumo de recursos. Se a aplicação web precisasse de de um plugin Flash, por exemplo, ela não iria executar na maioria dos tablets mais populares. Para auxiliar na construção das telas, adotamos para a interface com o usuário da parte Web, o Google Web Toolkit [25]. Utiliza-se a linguagem Java para desenvolver nessa tecnologia, no entanto, o resultado final será HTML, CSS e JavaScript. Grandes aplicações web foram desenvolvidas com essa tecnologia. O Orkut (rede social do Google muito popular no Brasil), por exemplo, foi desenvolvido com GWT.

O lado server da aplicação web tinha a missão de consumir poucos recursos sem comprometer o desempenho e estabilidade da aplicação. Ela deveria ser robusta para suportar várias conexões simultâneas. As tecnologias Java adotadas foram baseadas no Java Enterprise Edition (JEE) [35].

Utilizamos um framework para nos auxiliar no desenvolvimento do lado server. O Spring Framework. Essa combinação, Spring com JEE, garante a robustez e compatibilidade com várias infraestruturas de nuvem. Do JEE a tecnologia que mais utilizamos foi a "Servlet". Que é uma tecnologia leve e padrão para comunicação HTTP de alta concorrência como Java. Outras tecnologias JEE como Enterprise Java Beans (EJB) não foram utilizadas porque não foi preciso utilizar objetos distribuídos remotos e porque esse tipo de objeto consome muito recurso.

Para a manipulação e objetos persistentes e bancos de dados usamos a técnica de mapeamento objeto relacional (ORM) para possibilitar utilizar qualquer banco de dados das nuvens disponíveis no mercado. Algumas delas não suportam SQL mas a maioria suporta ORM com JPA, por exemplo.

Para construir os softwares adotamos o Eclipse como plataforma e ambiente de desenvolvimento padrão para produção dos códigos fonte do sistema. A arquitetura por si só não garante a produtividade dos desenvolvedores, é necessário ajuda de ferramentas adequadas para o projeto. Pode acontecer de uma ferramenta ajudar muito num projeto e ajudar pouco, ou até atrapalhar, em outro com características diferentes.

A partir desse ponto mostraremos as tecnologias, técnicas e a forma como foram utilizadas.

3.1 Celular

Um ponto estratégico para o sucesso do sistema foi a escolha do celular. Esse foi o maior investimento financeiro inicial no projeto para os gestores da pesquisa. Depois de escolhido, o instituto de pesquisa adquiriu 14 celulares para iniciar a operação. Esse

celular deveria ser barato, robusto, fácil de usar, ágil, oferecer os recursos necessários e possuir suporte acessível. Esses foram os principais requisitos.

Após intensa pesquisa no mercado local por um aparelho que obedecesse os requisitos estabelecidos, o Motorola EX115 [46] foi escolhido. Veja imagem 3.2.

Figura 3.2: *Motorola EX115. O celular escolhido para utilização no sistema de pesquisa das pneumonias na infância.*

Abaixo estão os principais motivos que levaram à escolha do aparelho e alguns resultados de testes realizados.

3.1.1 Baixo Custo

A escolha do aparelho foi realizada no segundo semestre de 2011. Nessa época, o celular escolhido podia ser encontrado por R$ 250,00 sem qualquer tipo de fidelidade. Com fidelidade à algum plano de alguma operadora o celular podia sair até sem custo. Existiam smartphones mais poderosos e com muito mais recursos no mercado. No entanto com preços até 10 vezes mais caros.

O custo do plano de dados também foi considerado. Sem dúvida é um ponto de preocupação, mas como a arquitetura do sistema foi projetada para ser muito otimizada em comunicações, o plano de dados mais barato de todas as operadoras nos atendia muito bem. Por R$ 0,50 por dia o sistema funcionaria com bom desempenho. E como o sistema no celular pode funcionar offline, nem todos os dias o agente de saúde acessará a internet do aparelho.

O fato de ter que ser de baixo custo está associado às condições e recursos financeiros dos institutos de pesquisa em saúde pública nos países em desenvolvimento. É comum terem pouco ou nenhum orçamento para investimento em tecnologias de apoio a pesquisa das pneumonias na infância.

Os celulares de baixo custo com o mínimo de recursos que atendem o projeto são os que suportam tecnologia Java Micro Edition (JME) [32]. Essa tecnologia já é bastante madura e consolidada. Acreditamos que enquanto existir demanda por celulares simples, robustos e baratos, o JME estará presente no mercado.

Cerca de 70% de todos os celulares do mundo são compatíveis com a tecnologia JME. Essa plataforma, portanto, possui a maior base instalada em celulares. Apenas 27% dos dispositivos móveis são smartphones. Esse número é menor ainda se considerarmos apenas os países emergentes. Isso conclui que, principalmente em economias em desenvolvimento JME é a principal opção para distribuição de aplicativos para dispositivos móveis [57].

Na época da escolha, os smartphones Android estavam começando a se destacar com a progressiva baixa de preços, mas, ainda longe do orçamento do instituto de pesquisa.

3.1.2 Robustez

Os celulares podem ser utilizados por até 24 horas diárias, pois são usados por profissionais de saúde que eventualmente realizam plantões. Celulares frágeis não atendem. O aparelho precisa ter um teclado confortável e ágil e suportar pequenas quedas e choques.

Em hospitais, públicos principalmente, de países em desenvolvimento, as condições de trabalhos fazem com que seja necessário um aparelho robusto que suporte intensa atividade. Alguns modelos touch-screen foram descartados pela fragilidade apresentada.

Um ponto bastante importante que descartou vários modelos é a capacidade da bateria. Quanto menos vezes o agente de saúde tiver que abastecer o celular, melhor para o processo de coleta de dados dos pacientes. Como o equipamento escolhido possui poucos recursos que consomem bateria, multimídia por exemplo, ele apresentou uma boa autonomia durante os testes e posteriormente durante a operação em campo.

A robustez do equipamento escolhido, é muito satisfatória. Há mais de um ano os celulares adquiridos pelo instituto de pesquisa estão sendo utilizados nas pesquisas das pneumonias na infância sendo manipulados por mais de 12 horas diárias e ainda estão em perfeito estado de funcionamento.

3.1.3 Fácil de Usar

Não são todas as pessoas que possuem intimidade com tecnologia. O aplicativo no celular deve ser muito simples e o equipamento deve possibilitar a entrada de dados fácil e rápida. Depois de avaliado os modelos existentes no mercado, decidiu-se adotar um com teclado físico completo. Os chamados teclados “QWERTY”. Veja figura 3.3.

Figura 3.3: *Teclado normal, o mais comum entre os celulares, e o teclado QWERTY.*

Esse tipo de teclado QWERTY físico apresentou uma entrada de dados muito confortável. Os teclados comuns necessitam de até 5 ou 6 toques numa tecla para entrar com o caracter desejado. Os do tipo QWERTY são mais diretos.

Teclados virtuais em display touch-screen, além de serem frágeis, poderiam apresentar dificuldades na entrada de dados devido a manipulação de resíduos no hospital. A utilização de luvas, por exemplo, poderia até impossibilitar a entrada de dados com esse tipo de teclado.

Teclados QWERTY físicos aumentam a produtividade dos agentes de saúde, já que a entrada de dados é parecida com um teclado normal de computador, além de ele cansar menos. Digitar num teclado de celular durante horas pode dar fadiga no digitador. O aplicativo e o celular precisam fazer com que o usuário não tenha que se esforçar para realizar entrada de dados.

Outra característica do modelo escolhido é a tela confortável para interagir. Telas muito pequenas dificultam a entrada e visualização de informações e portanto, são mais difíceis de usar. As cores e resolução do display devem ter qualidade tal que a visualização das fotos de raio x e cartões de vacinação no equipamento seja agradável.

3.1.4 Ágil

O celular escolhido apesar de não ser um smartphone possui ótima capacidade de processamento. Talvez pelo fato de o aplicativo consumir poucos recursos do celular. De qualquer forma, o desempenho de processamento do aparelho é muito bom e atende muito bem as necessidades do sistema.

Para IO (entrada e saída de dados) o celular é lento com grandes quantidades de dados. Lembrando que grandes quantidades de dados num celular pode ser muito pequena para um computador. 1 MB de dados pode ser considerado uma grande quantidade

de dados. Um caso inteiro, todo preenchido, por exemplo, possui mais ou menos esse tamanho. Mas como IO de muitos dados acontece poucas vezes por dia, o processo de coleta não é substancialmente afetado.

Um problema de desempenho que existe e atrapalha é o observado na transmissão de dados. A transmissão de um caso para o servidor pode demorar de cinco a trinta segundos devido à qualidade da rede de dados da operadora. Mas esse tipo de problema de desempenho não seria resolvido por outro celular já que o problema é da operadora. Então o sistema terá que conviver e contornar, quando possível, esse tipo de deficiência da solução técnica. De cinco a trinta segundos é um tempo aceitável para a transmissão de um caso. Poucos casos são enviados por dia para cada agente de saúde.

3.1.5 Recursos

Além de atender aos requisitos obrigatórios o aparelho ainda possui dois chips. Então o Agente de Saúde pode usar um chip para dados, fornecido pelo instituto de pesquisa, e outro, se necessário, para comunicação pessoal.

O recurso mais importante que os celulares candidatos obrigatoriamente deveriam ter é uma câmera com boa qualidade de imagem. A qualidade da imagem é essencial para o sucesso do projeto. É a qualidade dela que possibilita a realização dos laudos radiográficos pelo Radiologista.

Com os testes internos que foram feitos o mínimo aceitável de qualidade seria de uma câmera com 3 megapixels em condições de luminosidade ideais. Os modelos com câmera com essa qualidade, ou superior, eram muito caros. O EX 115 foi um dos poucos que tinham uma boa câmera e um bom preço.

Se a qualidade da imagem for muito grande, teremos outro problema. A transmissão da imagem. Imagens tiradas com 5 ou mais megapixels produzem imagens muito grandes para uma transmissão via internet 2 G do celular. Ficaria muito lento transmitir os casos com imagens para o servidor sem falar nas tentativas e erros nas falhas de conexão.

Outro recurso imprescindível era rodar Java. O aplicativo construído é compatível com celulares que rodam Java com JME (Java Micro Edition). A maioria dos celulares candidatos no mercado tinham esse recursos. Não foi problema selecionarmos os modelos por esse critério.

3.1.6 Suporte Local

O fato de o celular escolhido ser da Motorola ajudou nesse critério. Existe assistência técnica para esse tipo de celular nas principais capitais. As operadoras de celular também conhecem esse modelo e quando for necessário suporte online, devido a popularidade do modelo, este será realizado sem maiores problemas. É comum a

operadora ajudar os agentes de saúde e a equipe do instituto de pesquisa a configurar a internet nos celulares.

Por o modelo do celular escolhido ser bastante vendido na região, é bastante fácil encontrar acessórios como capas protetoras, fones, peças sobressalentes, etc. Isso sem dúvida é um diferencial do aparelho.

3.2 Nuvem

Computação em nuvem é um movimento que tem ganhado muita força e apoio da indústria de software e hardware. Significa, basicamente, a disponibilização de infraestrutura de hardware e software como serviço [7]. Devido a economia e a simplificação de seus negócios, as empresas estão confiando cada vez mais nesse tipo de serviço e terceirizando suas infraestruturas de TI (tecnologia da informação). Grandes empresas como Google, IBM, Microsoft, Red Hat, entre outras e mais um universo de empresas menores estão fornecendo serviços na "nuvem".

3.2.1 Tipos

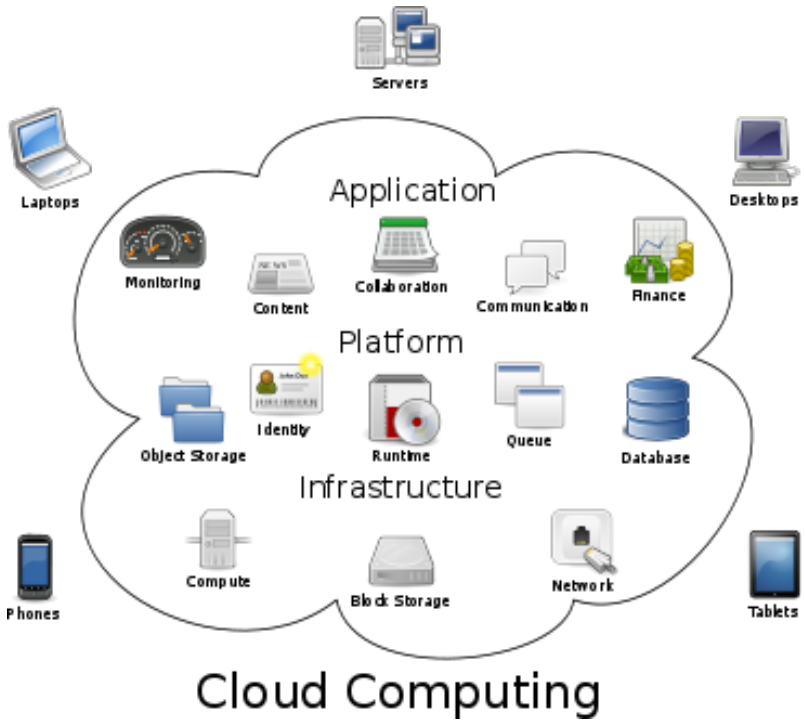

Figura 3.4: *Estrutura da computação em nuvem.*

Existem vários tipos de nuvens. Até o momento existem por volta de 11 categorias [68]. Mas basicamente podem se dividir em aplicações, plataforma e infraestrutura. Veja figura 3.4.

Os dois tipos que se aplicariam à arquitetura é a plataforma como serviço e a infraestrutua como serviço. Aplicação como serviço, na verdade é o que será oferecido após o sistema estar em produção. Um sistema nas nuvens para pesquisa das pneumonias na infância.

3.2.2 Plataforma como Serviço

Plataforma como serviço, na prática, é um servidor, ou cluster de servidores, com uma série de softwares já instalados como sistema operacional, servidores de aplicação, bancos de dados, serviços de email, entre outros. Esse tipo de nuvem é bem fácil de utilizar, no entanto paga-se um preço por essa facilidade. O valor é por utilização (pague apenas pelo que usar) mas ainda assim é alto. Com a plataforma como serviço nos preocupamos apenas em publicar a aplicação na nuvem. O resto acontece de forma automatizada. Pelo alto custo, esse tipo de serviço apresenta bastante limitações e pouco desempenho mas compensam com alta disponibilidade e escalabilidade.

No mercado, atualmente, existem várias opções de plataforma como serviço. As mais populares que suportam Java [49] estão listadas abaixo:

- Google App Engine [22]. Da Google.
- Beanstalk [1]. Da Amazon.
- Cloud Foundry [66]. Da VMware.
- OpenShift [52]. Da Red Hat.
- Heroku [29].
- CloudBees [6].
- Jelastic [36].

Essas plataformas como serviços ou nos limitam quanto à arquitetura e tecnologia ou são caras, mesmo para projetos pequenos. Num futuro próximo acredita-se que elas serão as melhores opções. Mas por hora existem opções melhores nas Infraestruturas como Serviços.

3.2.3 Infraestrutura como Serviço

As infraestruturas como serviço oferecem apenas a máquina (dedicada ou compartilhada) e, em alguns casos, poucos softwares de gerenciamento e de monitoramento. No entanto são mais baratas, oferecem mais recursos, possuem melhor desempenho e geralmente não oferecem restrições arquiteturais.

Os preços costumam ser cobrados por mês. Independente do quanto você está utilizando da máquina. O trafego mensal de rede na internet possui uma cota mensal máxima. A elasticidade da infraestrutura como serviço é menor, quando possível. Entenda

como elasticidade a capacidade de mudarmos as configurações da máquina de forma transparente sem impacto para os sistemas que a estão utilizando. Mais memória, mais processador, mais máquinas, mais discos, etc. Então é importante planejar a capacidade antes de adquirir uma máquina.

Algumas das opções mais populares:

- Amazon EC2 [2].
- Rackspace [51].
- Linode [42].
- GoGrid [21].

Essas nuvens basicamente oferecem uma máquina virtual. A instalação, configuração e administração dos serviços fica por sua conta. O sistema operacional, o servidor de aplicação e o banco de dados precisarão ser instalados manualmente. A nuvem não oferece suporte pelos softwares que você instala.

3.2.4 Aquisição

Foram investigadas várias nuvens. Grandes, pequenas, locais, em outros países, novas, consolidadas e finalmente encontramos uma que poderia nos atender. WebKeepers. $ 5.79 por mês se contratar um plano de 12 meses. Com 512 M de RAM e 100 G de HD. Uma máquina equivalente em outras IaaS (Infraestrutura as a Service) [30] pode chegar a ter um preço até 50 vezes maior.

Essas configurações atendem muito bem as necessidades da arquitetura. Mas para funcionar com tão pouca quantidade de memória, para um servidor, tivemos que prepará-la para isso. O Spring Framework e o Google Web Toolkit foram as tecnologias que mais contribuíram para fazer com que o sistema funcionasse numa nuvem com recursos tão restritos.

O sistema operacional utilizado no servidor na nuvem é o CentOS [5]. Ele é um Linux gratuito derivado e compatível com o Red Hat Enterprise Linux. Muito utilizado em servidores nas nuvens.

O servidor de aplicação, para executar o sistema, utilizado foi o Jetty [38]. Ele é muito pequeno e ágil. O Google o adotou na criação de sua própria PaaS [67].

O banco de dados utilizado foi o Apache Derby [10]. Um banco de dados que era da IBM e foi doado para a Apache. Ele é todo feito em Java e possui mais de 15 anos de maturidade. Consome pouca memória e processamento e tem todas as funcionalidades de um banco de dados relacional. O Derby possui muitas ferramentas administrativas integradas diminuindo ou eliminando a necessidade de uma pessoa para administrar o banco de dados (DBA). Ideal para o projeto. Tarefas como, compactação de tabelas e

backup são feitas de dentro da própria aplicação que construímos. Ele realmente foi feito para ser embutido em aplicações.

Com isso o servidor nas nuvens estava completo e pronto para receber a aplicação web.

3.3 Ambiente de Desenvolvimento

Explicaremos aqui quais ferramentas utilizamos para a construção do projeto e como elas foram adquiridas e montadas. Entendemos que o ambiente também faz parte da arquitetura do sistema. Sem ele não conseguiríamos transformar requisitos em software com a qualidade e velocidade necessárias. O ambiente de desenvolvimento foi escolhido pensando em produtividade, simplicidade, velocidade, custo e trabalho em equipe.

3.3.1 Eclipse

As ferramentas utilizadas na construção dos softwares são baseadas no Eclipse [17]. Eclipse é uma plataforma de desenvolvimento. É a mais popular das IDEs (ambiente de desenvolvimento integrado) e a mais largamente utilizada pelos desenvolvedores Java no mundo [24] [20] [28]. Essa ferramenta é free e open source, sem custo algum para quem a utiliza [17].

O A plataforma Eclipse por si só não oferece nada. Ela só fornece a base para construção e utilização de plugins. Da mesma forma que o Linux por si só não faz muita coisa sem as distribuições e seus pacotes de utilitários. Portanto para que ela seja útil precisamos instalar alguns plugins específicos para o projeto.

3.3.2 Java Development Tools

Usamos o plugin JDT. Java Development Tools [34]. É o que faz o Java funcionar no Eclipse. Edição de arquivos Java, refatoração, depuração, testes, entre outros recursos. Ao contrário do que a maioria acredita, o Eclipse não vem com suporte a Java. As distribuições padrão que são oferecidas no site principal do Eclipse já possuem suporte a Java e então tem-se essa falsa impressão.

3.3.3 Google Plugin for Eclipse

Para desenvolvimento das interfaces web com o usuário utilizamos o GPE. GPE é o Google Plugin for Eclipse [23]. Esse plugin, construído pelo Google, também é free e open source. Ele ajuda muito na construção e validação de telas feitas com o GWT (Google Web Toolkit [25]). Essa tecnologia foi adotada para construção da interface web

com o usuário e também será abordada nesse trabalho. O GPE possui um servidor de aplicações embutido, o Jetty, que nos permite testar a aplicação web no ambiente de desenvolvimento da máquina do desenvolvedor. Tudo de forma integrada inclusive com depuração de código.

3.3.4 GWT Designer

Em conjunto com Google Plugin for Eclipse foi instalado também o plugin GWT Designer [26]. Ele é um editor visual para construção de telas feitas com o Google Web Toolkit. Com ele o desenvolvedor constrói uma tela web como se estivesse fazendo uma tela desktop (como no Visual Studio da Microsoft, uma das melhores ferramentas do mercado para desenvolvimento de sistemas desktop). Veja na figura 3.5.

Na imagem podemos ver a edição de uma tela de “Login”. Na parte superior estão a paleta de componentes, o inspetor de propriedades dos componentes selecionados e a área de design. Na parte inferior está o código fonte. A alteração na parte superior refletirá na inferior e vice versa. Não parece, mas tudo isso se transformará em HTML, CSS e JavaScript depois de compilado. Ganhamos muita produtividade com esse plugin pois boa maior parte de esforço de desenvolvimento está na construção das telas, das interfaces com o usuário.

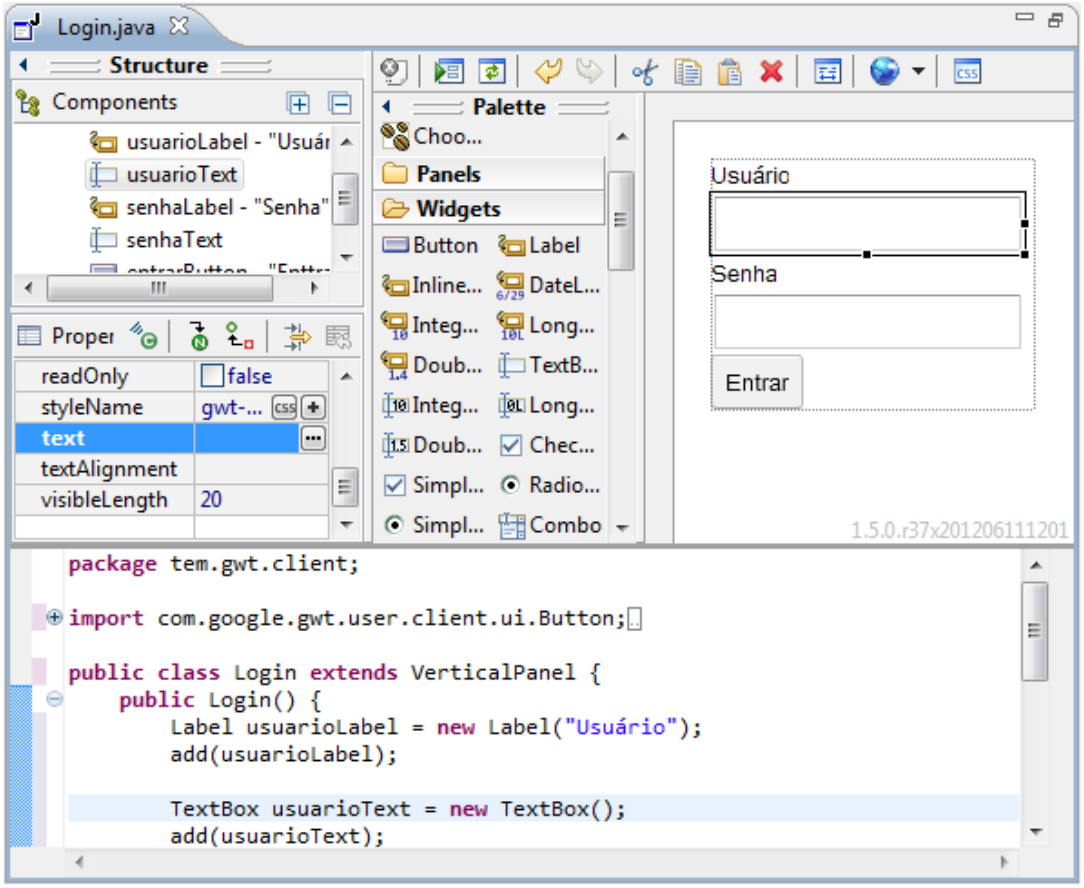

Figura 3.5: *GWT Designer no Eclipse.*

3.3.5 Maven to Eclipse

Utilizamos também o M2E (Mavem para Eclipse) [43] para ajudar no gerenciamento de dependências de bibliotecas. O M2E, por sua vez utiliza o Maven [45] que é a ferramenta que realmente realiza o gerenciamento de dependências entre bibliotecas.

Essencial para projetos grandes que utilizam muitas bibliotecas Java. Se precisarmos da biblioteca “hibernate-core” versão 4.1.6, por exemplo, o Maven através do M2E automaticamente baixa e instala a biblioteca “dom4j” versão 1.6.1. Veja imagem 3.6. Isso economiza horas de pesquisa e testes de compatibilidades entre bibliotecas. Além disso, projetos que utilizam Maven já estão automaticamente preparados para serem utilizados em ferramentas como o Sonar [58] (gerenciamento de qualidade de código) e o Jenkins [37] (ferramenta de integração contínua).

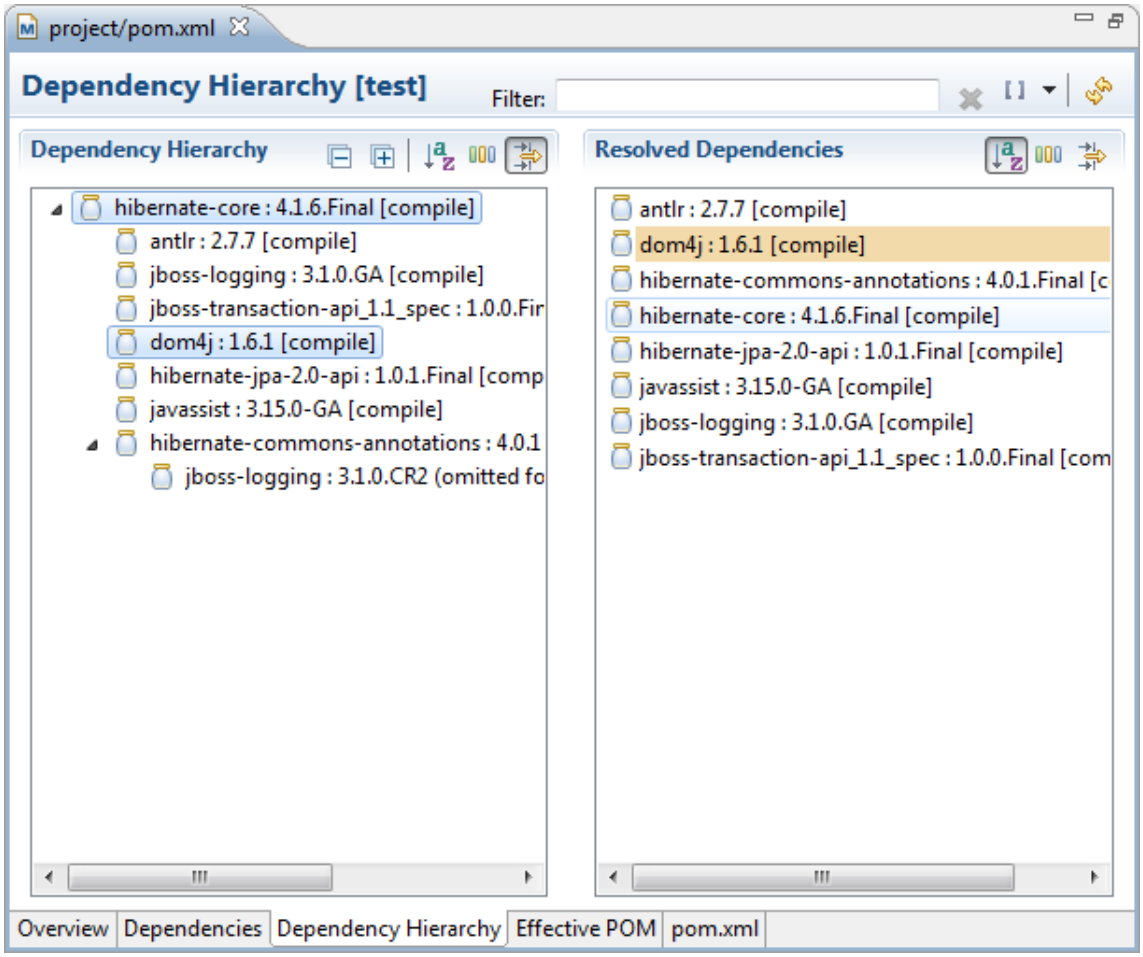

Figura 3.6: *Exemplo de resolução de dependência de bibliotecas Java com o M2E.*

3.3.6 Eclipse Web Developer Tools

Para termos suporte a XML, HTML, CSS e JavaScript precisamos instalar o Eclipse Web Developer Tools [69]. Esse plugin ajuda na edição e validação dos arquivos utilizados na construção do conteúdo web. Ele corrige erros e sugere correções caso o desenvolvedor não esteja obedecendo os padrões. Esse plugins ajuda bastante pois apesar de o desenvolvimento utilizar plataforma Java, na web, precisamos usar muito arquivos XML, HTML, CSS e JavaScript. Não tem como fugir completamente dessas tecnologias.

3.3.7 JPA Support

Como utilizamos mapeamento objeto-relacional no projeto com Java Persistence API, era importante ter uma ferramenta que nos ajudasse. Existe um plugin do Eclipse que nos ajuda com construção, validação e testes de entidades. O JPA Support [15]. Esse plugin também faz a ligação da entidade Java com a entidade no banco de dados relacional

em tempo de desenvolvimento. Existe nele também um editor que permite em tempo de desenvolvimento realizar queries no banco de dado com JPQL ou SQL.

3.3.8 Subversive

Utilizamos intensamente controle de versão com o Subversion [60] e o Eclipse possui suporte para trabalhar de forma integrada com essa ferramenta de controle de versão de arquivos e pastas. Esse suporte é dado através do plugin Subversive [61]. Com ele, de dentro do Eclipse podemos realizar commit, update, synchronize, merge, branch e outras operações envolvendo o controle de versão da gerência de configuração. Veja figura 3.7.

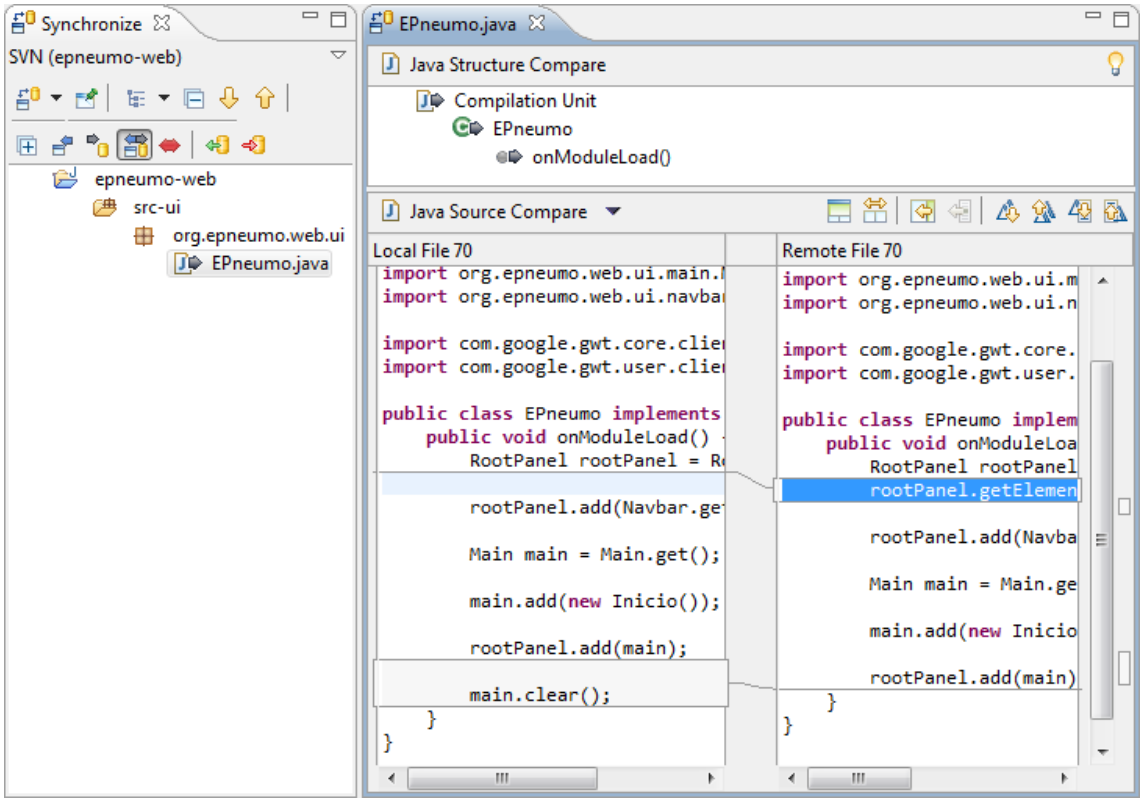

Figura 3.7: *Exemplo de sincronização com o repositório do servidor do Subversion e comparação de versão de um arquivo.*

Seria muito difícil trabalhar em time (mais de uma pessoa ou até mesmo sozinho) sem uma ferramenta como essa. O Subversive possui conectores para comunicação com o servidor do Subversion. Dos que são fornecidos, escolhemos o SVNKit [62] por ser 100% Java, rodar em qualquer plataforma, ser bastante estável e apresentar um bom desempenho nas tarefas mas comuns (commit, update, compare e synchonize).

3.3.9 Antes e Depois

Abaixo a comparação do Eclipse antes e depois da instalação dos plugins. Veja figura 3.8. Esse é um Eclipse personalizado para atender as necessidades do projeto. Os plugins instalados mas que não foram abordados nessa seção foram instalados por dependência.

Quando utilizamos apenas os plugins necessários para o projeto o ambiente fica muito mais leve e o código fonte não sofre modificações desnecessárias devido a ações de plugins indesejados. E isso se reflete em produtividade.

O ambiente é um artefato do projeto e é gerenciado como tal. Se alguém no futuro precisar manter o sistema, usará esse ambiente com esses plugins.

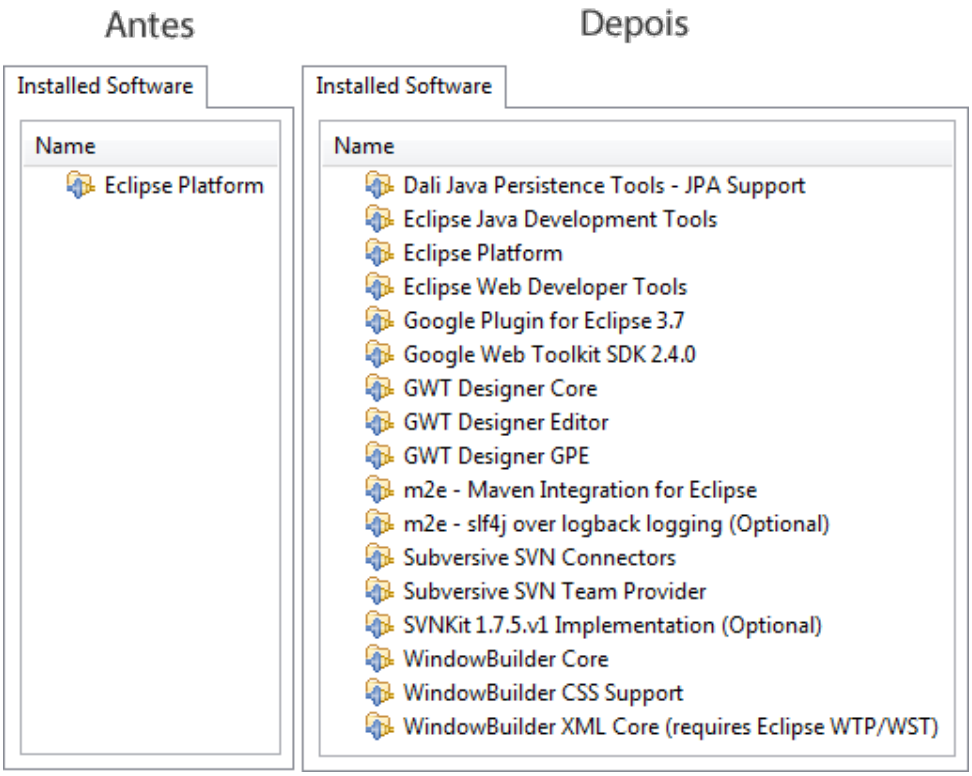

Figura 3.8: *Plugins do Eclipse antes, com apenas o Eclipse Platform, e depois da construção do ambiente completo, com os outros plugins.*

3.4 J2ME Polish

J2ME Polish [31] foi a tecnologia escolhida para a construção da interface com o usuário no celular. A evolução dos dispositivos móveis é muito rápida. Um modelo de celular que é popular hoje, em alguns meses pode estar ultrapassado. É muito caro acompanhar as evoluções tecnológicas dos dispositivos. Se fosse preciso alterar o sistema para cada tipo de celular utilizado, gerando várias versões, a manutenção ficaria muito

cara e complexa. A incompatibilidade entre esses dispositivos é bastante alta e é nesse ponto que o J2ME Polish mais agrega valor. Outras opções de frameworks não ofereciam solução para esse problema de fragmentação de dispositivos [19] ou eram muito caras.

3.4.1 Framework

Esse framework possui tanto uma API para codificação da aplicação quanto um mecanismo de construção que gera o binário do sistema para um conjunto de dispositivos semelhantes ou até mesmo para cada celuar individualmente de forma automatizada. Veja figura 3.9.

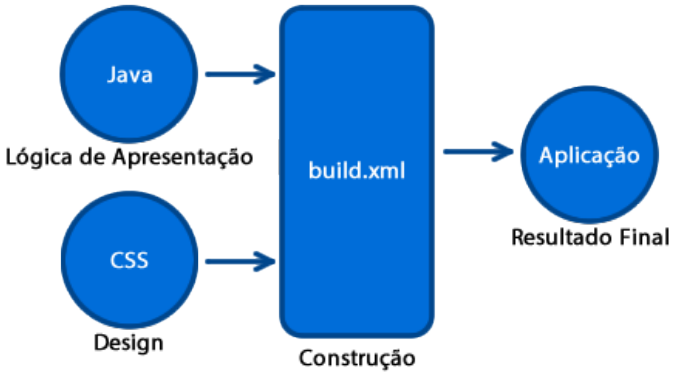

Figura 3.9: *Processo de construção de uma aplicação com J2ME Polish.*

O binário gerado aproveitará ao máximo o dispositivo (como o seu display) por ter sido compilado para ele. Isso é possível graças a um banco de dados de celulares e suas características. Com esse processo de construção é possível gerar aplicação inclusive para outras plataforma além da Java Micro Edition. Android, iOS, BlackBerry, Symbian e outras. Veja figura 3.10.

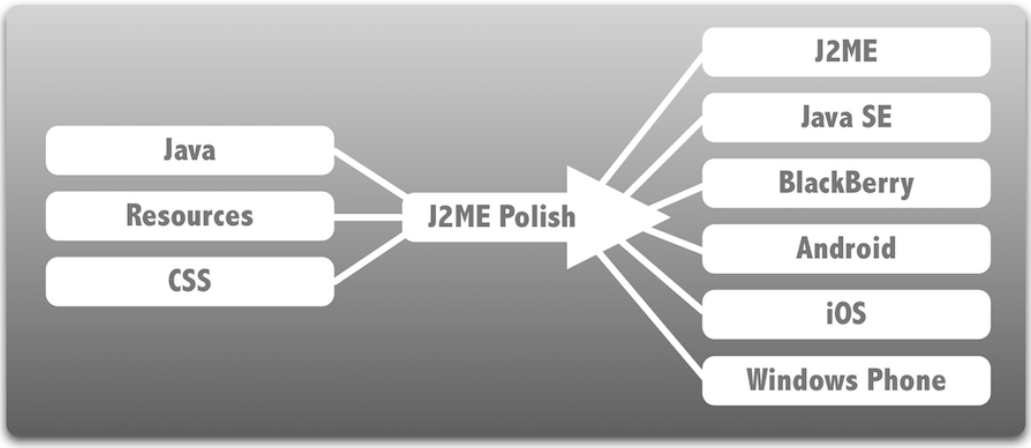

Figura 3.10: *J2ME Polish e as plataformas suportadas.*

3.4.2 Interface Gráfica

Uma das principais vantagens desse framework é a utilização de estilos CSS para construção de interfaces ricas para dispositivos móveis de pouco poder de processamento. É quase tão fácil quanto colocar estilo em páginas web.

Abaixo, na figura 3.11, podemos ver uma aplicação feita com os estilos CSS do J2ME Polish e outra sem estilo com Java Micro Edition "puro". Com Java Micro Edition o estilo da aplicação será o mesmo das aplicações padrão do celular. E muitas vezes, a aparência final é imprevisível prejudicando muito a experiência do usuário. Com J2ME Polish você pode definir o seu próprio estilo e ele será o mesmo entre vários celulares.

Figura 3.11: *A mesma tela no celular com e sem estilo CSS.*

O CSS do J2ME Polish não é exatamente como o CSS da web. Existem recursos específicos da tecnologia. Veja um exemplo em 3.1.

Código 3.1 Estilo CSS do J2ME Polish utilizado nas caixas de texto.

```
.textField {
  layout: expand;
  padding: 3;
  margin-right: 2px;
  margin-left: 2px;
  border {
    type: simple;
    color: #B5B8C8;
    width: 1;
  }
  background {
    type: vertical-gradient;
    top-color: #dee3e6;
    bottom-color: #ffffff;
    start: 0%;
    end: 100%;
  }
  focused-style: textFieldFocus;
  label-style: textFieldLabel;
}
```

Código 3.2 Estilo da caixa de texto quando está selecionada.

```
1  .textFieldFocus {
2    layout: expand;
3    padding: 1;
4    margin-right: 2px;
5    margin-left: 2px;
6    border {
7      type: simple;
8      color: #7eadd9;
9      width: 3;
10   }
11   background {
12     type: vertical-gradient;
13     top-color: #dee3e6;
14     bottom-color: #ffffff;
15     start: 0%;
16     end: 100%;
17   }
18 }
```

Código 3.3 Estilo do rótulo da caixa de texto.

```
1  .textFieldLabel {
2    layout: newline-after;
3    padding-right: 50px;
4    margin-left: 2px;
5  }
```

Esses estilos são definidos e depois utlizados no código Java em forma de comentários. Um pré-processador de código avalia o comentário especial e gera o código final com o estilo aplicado durante o processo de construção (build). Veja exemplo em 3.4.

Código 3.4 Trecho de código Java utilizando o estilo .textField.

```
1  //...
2
3    //#style title
4    form = new Form("Pneumo 10");
5
6    //#style subTitle
7    StringItem titulo = new StringItem("Login");
8    form.append(titulo);
9
10   //#style textField
11   TextField usuario = new TextField("Usuario");
12   form.append(usuario);
13
14   //#style textField
15   PasswordField senha = new PasswordField("Senha");
16   form.append(senha);
17
18   //#style button
19   sair = new Command("Sair");
20   form.addCommand(sair);
21
22   //#style button
23   entrar = new Command("Entrar");
24   form.addCommand(entrar);
25
26 //...
```

O estilo aplicado foi 100% personalisado para que a aparência da aplicação no celular ficasse com a aparência da aplicação na web. A identidade visual entre as duas aplicações é muito importante para o usuário. Ele intuitivamente associa as duas aplicações apenas pela aparência.

O estilo da aplicação no celular, uma vez definido, permanece o mesmo, mesmo entre dispositivos. Mas se o celular tiver teclado diferente e/ou tiver as dimensões da tela menores, a aparência não se altera. Veja figura 3.12

Figura 3.12: *Compatibilidade de estilos entre celulares diferentes. O design se adapta ao dispositivo.*

3.4.3 Persistência

O J2ME Polish também fornece um mecanismo de persistência de objetos 3.13. A plataforma Java Micro Edition, para celulares com poucos recursos como o que foi escolhido para o projeto, não fornece mecanismos de persistência de alto nível. Com JME temos apenas os RMSs (Record Management Store) para armazenar dados no celular. Esses dados, com RMS, são manipulados em forma de arrays de bytes. É bastante difícil para o programador realizar o mapeamento dos objetos, e sua estrura de dados, com os arrays de bytes.

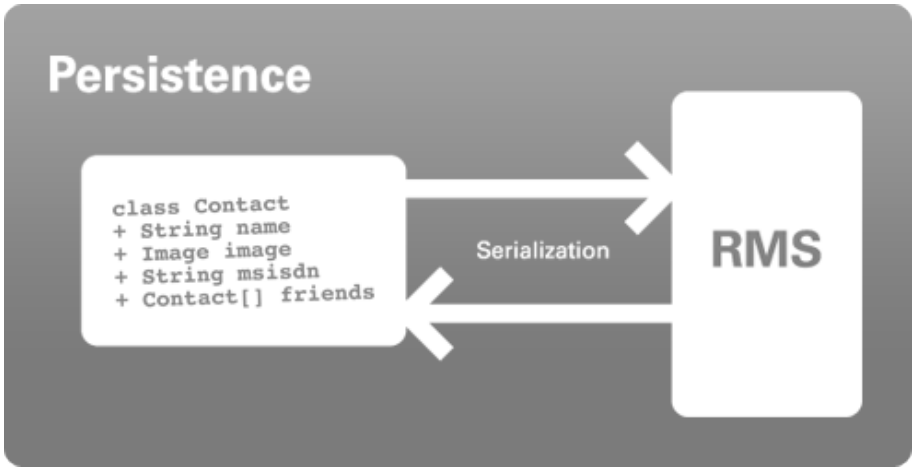

Figura 3.13: *J2ME Polish Persistence.*

Veja abaixo no código 3.5 o que se deve fazer para gravar um registro de "usuário" no banco de dados do celular utilizando JME RMS.

Código 3.5 Trecho de código Java fazendo persistência com JME RMS.

```
1  //...
2
3    usuario.setNome(nomeTextField.getValue);
4    usuario.setSenha(senhaPasswordField.getValue);
5
6    RecordStore rs =
7      RecordStore.openRecordStore("Usuario" +
8      usuario.hashCode(),
9      true);
10
11   byte[] nomeBytes = usuario.getNome().getBytes();
12   rs.addRecord(nomeBytes, 0, nomeBytes.length);
13
14   byte[] senhaBytes = usuario.getSenha().getBytes();
15   rs.addRecord(senhaBytes, 0, senhaBytes.length);
16
17   rs.closeRecordStore();
18
19 //...
```

Fazendo o mesmo com o J2ME Polish Persistence 3.6.

Código 3.6 Trecho de código Java fazendo persistência com J2ME Polish Persistence.

```
1 //...
2
3    usuario.setNome(nomeTextField.getValue);
4    usuario.setSenha(senhaPasswordField.getValue);
5    PersistenceManager.save(usuario);
6
7 //...
```

Além de diminuir a possibilidade de erros aumenta a legibilidade do código e a produtividade do desenvolvedor.

3.4.4 Comunicação

A transmissão de dados entre o celular e o servidor web é a operação mais complexa e demorada que ocorre no celular. As redes de dados móveis são instáveis e lentas. Transferir grandes quantidades de dados (1 M já é grande quantidade de dados) passa a ser crítico em condições como essas. No entanto o J2ME Polish nos ajuda com uma funcionalidade que facilita a operação de transmissão de dados. O J2ME Polish RMI (Remote Method Invocation). Veja figura 3.14.

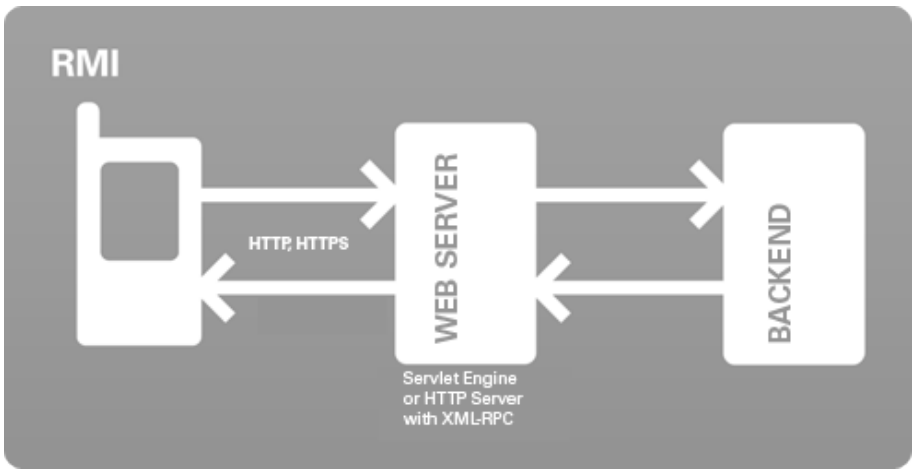

Figura 3.14: *J2ME Polish RMI (Remote Method Invocation).*

Abaixo o código do lado client (celular) e server (web).

Código 3.7 Trecho de código Java para realizar a chamada remota com RMI. Lado client (celular).

```
1  Arquivo Pneumo10Rmi.java
2
3  //...
4
5  public interface Pneumo10Rmi extends Remote {
6    public Resposta enviarCaso(Caso caso) throws RemoteException;
7  }
8
9  //...
10
11 Arquivo EnviarCasoView.java
12
13 //...
14
15 Pneumo10Rmi rmi = (Pneumo10Rmi)
16   RemoteClient.open("pneumo10.mob.rmi.Pneumo10Rmi",
17   "http://pneumo10.com/Pneumo10Rmi");
18
19   rmi.enviarColeta(caso);
20
21 //...
```

O código acima serializa e envia o caso pela internet.

Código 3.8 Trecho de código Java para receber a chamada remota com RMI. Lado server (web).

```
Arquivo web.xml

<!-- ... -->

<servlet>
  <servlet-name>Pneumo10Rmi</servlet-name>
  <servlet-class>
    pneumo10.mob.rmi.Pneumo10RmiImpl
  </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>Pneumo10Rmi</servlet-name>
  <url-pattern>/Pneumo10Rmi</url-pattern>
</servlet-mapping>

<!-- ... -->

Arquivo Pneumo10MobRmiImpl.java

//...
public class Pneumo10MobRmiImpl
  extends RemoteHttpServlet
  implements Pneumo10MobRmi {

  public Resposta enviarCaso(Caso caso)
    throws RemoteException {

    return salvarCaso(caso);
  }

}

//...
```

O código acima desserializa o caso e o processa.

O RMI do J2ME Polish alésm de facilitar a transmissão de objetos ainda a faz com velocidade e de forma compactada. Apenas o mínimo possível de informações é

trafegado.

3.5 Google Web Toolkit

O desenvolvimento de aplicações Web que utilizam padrões, basicamente, se resumem em HTML, JavaScript e CSS. O que vemos no navegador são essas tecnologias não importanto o que esteja do lado server (Java, .Net, PHP, etc).

O Google Web Toolkit (GWT) [25] é uma tecnologia open source desenvolvida pelo Google para auxilio na construção de interface com usuário em sistemas Web. Com GWT você escreve o código fonte em Java e o kit de desenvolvimento trata de transformar em HTML, CSS e JavaScript. Veja figura 3.15.

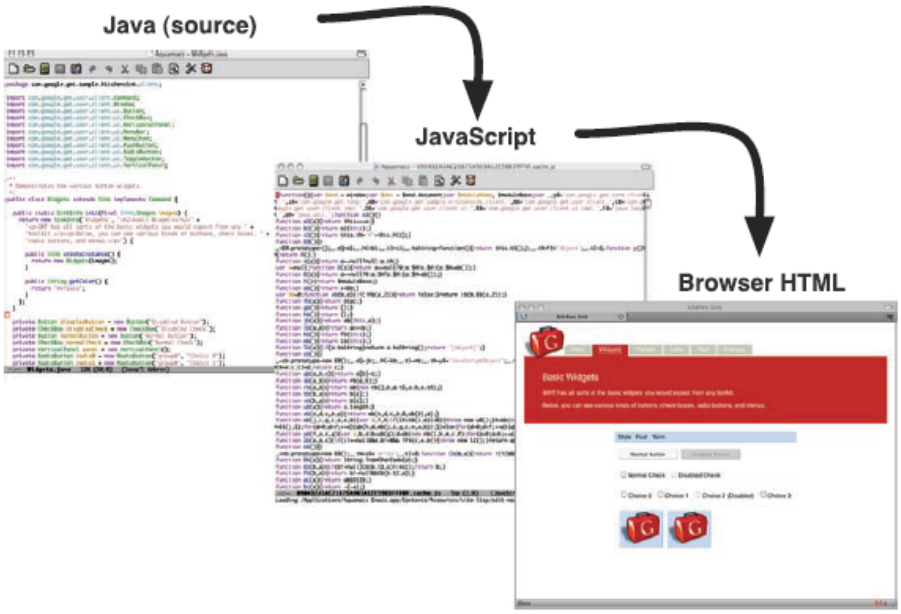

Figura 3.15: *Compilação de Java para JavaScript e HTML com GWT.*

3.5.1 Motivação

É possível desenvolver grandes softwares web apenas com JavaScript, mas o desenvolvimento de aplicações é mais difícil do que precisava ser. Existe também a dificuldade de efetivamente gerenciar um projeto usando JavaScript. Facilidades como ferramentas de teste e ambientes de desenvolvimento poderosos com capacidade de depuração são comuns para linguagens tipadas, como Java. JavaScript, por outro lado, talvez por ser uma linguagem de script não possui as mesmas facilidades [54].

Claro, pode-se gerenciar um projeto JavaScript com sucesso, mas a necessidade de desenvolver e manter várias versões diferentes do código para navegadores diferentes é uma dor de cabeça. Além disso, não é fácil encontrar no mercado desenvolvedores

JavaScript que estão cientes dos problemas e nuances do navegador e que também estão em um nível de maturidade suficiente com os processos de produção de qualidade de desenvolvimento para entregar um projeto grande (em comparação com o número de programadores Java) [53].

A arquitetura proposta se preocupou também com aspectos não técnicos como mão de obra disponível no mercado. É mais fácil e barato encontrar bons desenvolvedores Java do que bons desenvolvedores JavaScript.

3.5.2 Funcionamento

O GWT possui um compilador. A responsabilidade do compilador do GWT é converter o código Java em código JavaScript, de forma parecida como o compilador Java compila código Java em bytecode.

O compilador do GWT é inteligente o bastante para despresar o código que foi escrito mas não está sendo utilizado. Isso é muito útil para desenvolvimento de aplicações de larga escala. O código JavaScript gerado com a compilação é um código ofuscado e compactado. Sendo quase impossível de decifrar. Além disso o compilador gera automaticamente uma versão da aplicação para cada navegador disponível no mercado e trata das particularidades de cada um deles sem que presisemos nos preocupar com isso [54].

Existem dois modos de execução de uma aplicação GWT. Hosted Mode e Web Mode. No Hosted Mode o compilador trabalha sob demanda a medida que o usuário navega pelas páginas. Esse modo é útil apenas durante o desenvolvimento. No Web Mode não existe compilação sob demanda. Todo o HTML, CSS e JavaScript necessários para executar a aplicação já estão compilados. Esse é o modo como será executado em produção. Veja itens 1 e 2 na figura 3.16.

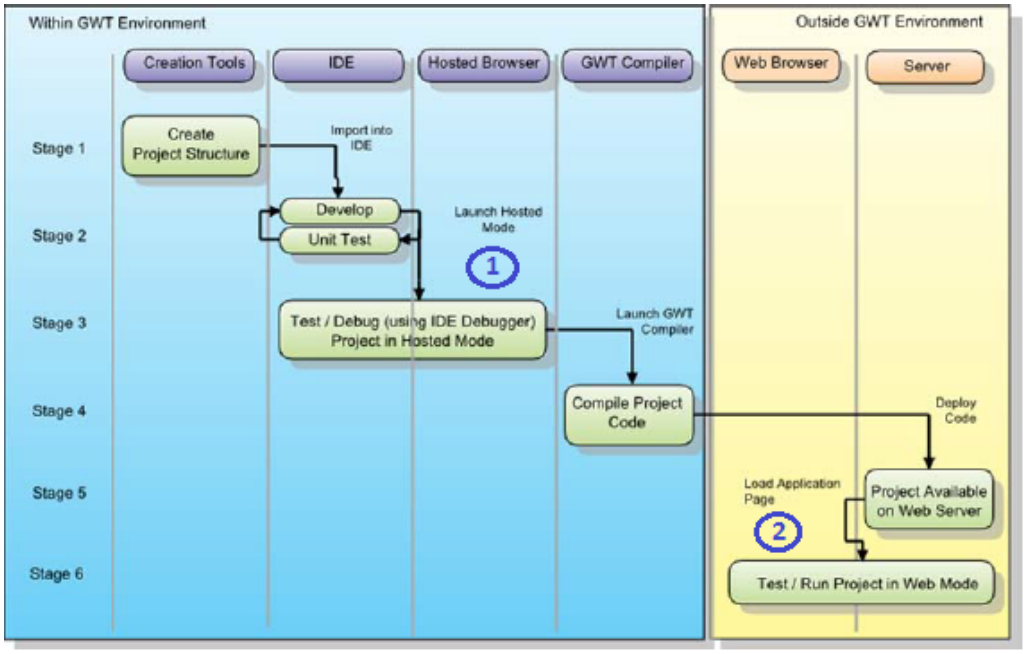

Figura 3.16: *Funcionamento do GWT em Hosted Mode e Web Mode.*

3.5.3 Utilização

O GWT foi utilizado para a construção da interface gráfica do sistema web. O ambiente de desenvolvimento possui um editor e design gráfico para construção das telas. Isso facilita muito o trabalho do desenvolvedor. Ao contrário de outras ferramentas o código fonte gerado é muito limpo contendo apenas o necessário para a tela funcionar. Se o desenvolvedor durante a construção da interface visual trabalhar no código diretamente sem o editor gráfico, o GWT Design automaticamente executa o código fonte e mostra o resultado da execução em tempo de desenvolvimento. Ou seja, é possível trabalhar direto no código fonte, apenas no design ou em ambos. Ganha-se muita produtividade com essa combinação de ferramentas. Veja figura 3.17.

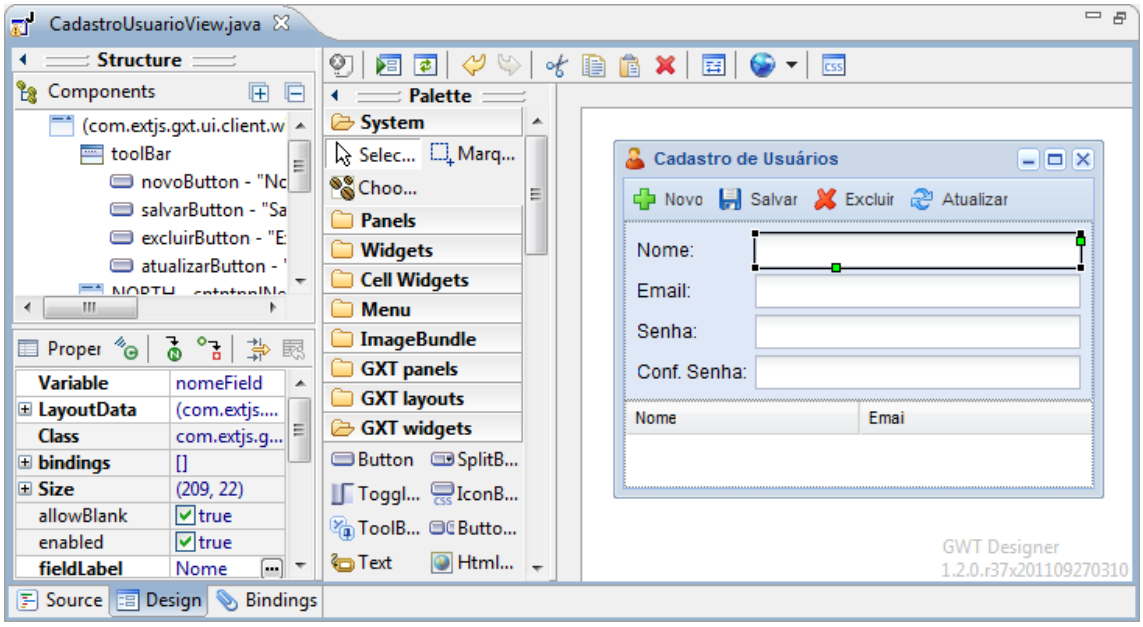

Figura 3.17: *Interface com o usuário web para o caso de uso Manter Usuários em modo de desenvolvimento.*

O código fonte da interface com o usuário por ser Java nos permite utilizar todo o poder da programação orientada a objetos. Foi identificado padrões de telas e para cada padrão uma hierarquia de classes foi criada. Assim, para as telas mais comuns fica muito fácil criar uma interface completa. No caso de um cadastro simples, como o da figura 3.17, basta criarmos uma classe que herde de "CadView", definir algumas variáveis e pronto, a funcionalidade está completa. E ainda assim, o editor visual (GWT Designer) continua operacional. Veja o código 3.9.

Código 3.9 Código fonte da funcionalidade Manter Usuário utilizando uma hierarquia para cadastro.

```
1  package pneumo10.web.ui.cadastrousuario;
2  import pneumo10.web.ui.framework.*;
3  public class CadastroUsuarioView extends CadView {
4    public CadastroUsuarioView() {
5      setTitle("Cadastro de Usuários");
6      setIcon("usuario");
7      setService("Usuario");
8      addFormField(new TextField("Nome"));
9      addFormField(new TextField("Email"));
10     addFormField(new PasswordField("Senha"));
11     addFormField(new PasswordField("Conf. Senha"));
12     addColumn(new Column("nome"));
13     addColumn(new Column("email"));
14   }
15 }
```

3.5.4 Comunicação

A comunicação de uma tela com o servidor web é realizada via protocolo HTTP/REST (Representational State Transfer) com JSON (Java Script Object Notation). Esse padrão tem se tornado um dos mais utilizado em aplicações web com interfaces ricas.

Uma requisição é montada no client e em seguida enviada para un serviço no lado server. Do lado server uma árvore de objetos é montada e devolvida para o client no formato JSON que por sua vez realiza o bind para seus componentes visuais para mostrar o resultado. Esse processo todo é bastante rápido pois o que trafega na rede são apenas dados. Nenhum html, css, javascript, imagens e outros recursos são transferidos nas chamadas REST com JSON.

3.6 Spring

O Spring Framework [59] [55] é dividido em grandes módulos e cada módulo possui submódulos. E por ser um bastante extensível possui módulos de terceiros construídos para necessidades específicas. Na arquitetura, foram usados apenas módulos padrões fornecidos pelo próprio Spring Framework. Não foram necessários módulos adicionais.

3.6.1 Módulos

Os submódulos usados na arquitetura foram o “Core”, o “Beans” e o “Context” do módulo “Core Container”, o “JDBC”, o “ORM” e o “Transactions” do módulo “Data Access / Integration"e o “Web” do módulo “Web”. Veja figura 3.18.

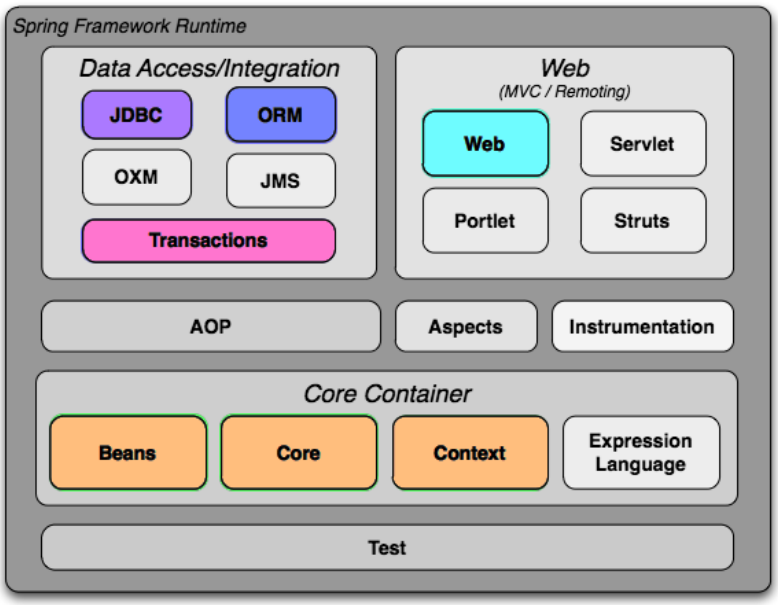

Figura 3.18: *Estrutura do Spring Framework.*

“Core” é a parte do Spring realiza criação e manipulação de singletons (através de fábricas de objetos), inversão de controle / injeção de dependência entre outras funcionalidades do container.

“Beans” define a utilização de objetos java em tempo de execução, a relação entre eles, seus ciclos de vida bem como a precedência e dependência entre objetos.

“Context”, construído sobre o “Core” e o “Beans”, controla como os objetos serão acessados, define internacionalização, realização propagação de eventos, carga de recursos e dá suporte a Java Enterprise Edition para funcionalidades como, EJB, JMX e chamadas remotas.

“JDBC” fornece uma camada de abstração para manipulação direta de banco de dados sem a necessidade de escrever extensos códigos e tratamentos de erros específicos de alguns fornecedores. Oferece grande produtividade e integração com outros módulos como o “Transactions”.

“ORM” possui mecanismos para mapeamento objeto-relacional. Integra transações e JDBC com objetos ligados à registros em tabelas de bancos de dados. Com o ORM do Spring a produtividade é muito elevada e o poder de portabilidade entre bancos de dados é elevado ao máximo.

“Transactions” livra o desenvolvedor de realizar tratamento de transações em bancos de dados relacionais ou não. Na verdade qualquer transação compatível com JTA (Java Transaction API) pode ser controlada com esse submódulo. É uma das funcionalidades mais procuradas pelos desenvolvedores no Spring.

“Web” é um módulo que oferece integração com o modelo web do Java Enterprise Edition fornecendo funcionalidades como upload de arquivos, manipulação de requisições e respostas e implementação do modelo MVC (model-view-controler).

Esses módulos todos reunidos faz com que o desenvolvedor economize muita linha código e tenha um resultado final com muita qualidade e desempenho. O Spring ajuda muito no desenvolvimento do lado server. Além disso o Spring Framework foi construído para faciliar a escrita e execução de testes automatizados. Uma ferramenta ideal para auxiliar no desenvolvimento da camada “server” utilizando a arquitetura em questão.

3.6.2 Serviços

Abaixo temos um exemplo de como fica o código de um serviço escrito com o Spring Framework utilizado no sistema de pesquisa. Trata-se do serviço de manutenção de usuários. A classe “UsuarioService” é o serviço desenvolvido que possui operações de “incluir”, “alterar”, “excluir”, “consultar” e “listar”. Ela herda de uma outra classe. A “CadService” que já possui inteligência para ser um serviço que fornece operações com o protocolo REST/JSON. As operações utilizam uma variável “modelManager” do tipo “ModelManager” que é responsável por persistência de dados e binding (ligação entre objetos) para ler e escrever em objetos e no banco de dados. O resultado final é um código extremamente simples e fácil de manter. Conforme foi planejado para a arquitetura. Veja código 3.10.

Tudo isso realiza uso intenso do Spring Framework. Existe muito código encapsulado no framework que nós não usamos diretamente e teríamos que desenvolver se não fosse ele.

Código 3.10 Implementação de um serviço de manutenção de usuários.

```
1  package pneumo10.web.services;
2
3  import pneumo10.web.services.framework.*;
4  import pneumo10.web.entities.*;
5
6  public class UsuarioService extends CadService {
7
8    public Model incluir(Model model) {
9      modelManager.insert(Usuario.class,  model);
10   }
11
12   public Model alterar(Model model) {
13     modelManager.update(Usuario.class,  model);
14   }
15
16   public Model excluir(Model model) {
17     modelManager.delete(Usuario.class,  model);
18   }
19
20   public Model consultar(Model model) {
21     modelManager.load(Usuario.class,  model);
22   }
23
24   public Model listar(Model model) {
25     modelManager.list(Usuario.class,  model);
26   }
27
28 }
```

Como consequência da implementação da classe “UsuarioService” temos a seguinte estruta de operações disponibilizadas pelo serviço. Veja código 3.11.

Código 3.11 Estrutura do serviço, e suas operações, gerado pela classe UsuarioService.

```
1 http://dominio-do-servidor.com
2     /nome-da-aplicacao-web
3         /services
4             /Usuario
5                 /incluir
6                 /alterar
7                 /consultar
8                 /listar
```

As operações dos serviços estão disponíveis para utilização por qualquer sistema escrito em qualquer plataforma. Isso é devido a interoperabilidade do protocolo de comunicação e das tecnologias utilizadas. Uma aplicação feita em C# poderia usar os serviços feitos em Java, por exemplo.

CAPÍTULO 4

Processo

O sistema de pesquisa implementado para provar a efetividade da arquitetura foi implementado utilizando boas práticas de processos ágeis [8] [44]. Dois analistas e dois desenvolvedores participaram da construção.

Apesar de a equipe estar geograficamente distante [16], utilizamos Scrum para gerenciar o projeto tendo como ferramenta de controle o Google Docs para comunicação e documentação remota. Não foram utilizados métodos formais de especificação de software como especificações de casos de uso. Apenas histórias de usuários e documentação no próprio código fonte. O código fonte e os testes automatizados, depois de concluídos, geraram uma documentação em formato HTML, usando JavaDoc, que foi considerado suficiente para a documentação do sistema. E a interface das classes e métodos com assinaturas bem definidas junto com alguns comentários produziram um resultado final de documentação muito bom.

Todas as unidades de software produzidos, desde a interface com o usuário até a persistência de objetos, são preparadas para realização de testes automatizados [4]. Portanto, para aproveitar essa capacidade foi utilizado o JUnit [39], para alguns tipos de testes unitários e JBehave [33], para testes de aceitação. Apesar de não ter sido explorado no sistema toda a capacidade de realização de testes, pelos estudos, a arquitetura está preparada para diversos cenários.

Foi utilizado, na construção e manutenção do sistema, uma gerência de configuração bem simples com apenas controle de versão. Como ferramenta foi utilizado o Subversion (SVN) [60] e como serviço na nuvem para utilização do SVN foi contratado o serviço Assembla [3] que, sem custo algum, suportou muito bem o projeto. O ambiente de desenvolvimento integra muito bem com esse tipo de serviço e deu muita agilidade e qualidade nas atividades de desenvolvimento envolvidas no controle de versão.

Para controle de versão de bibliotecas e gerenciamento de dependências foi utilizado o Maven [45] com o Nexus [47]. O Maven controla quais bibliotecas e quais versões estão sendo utilizadas pelo sistema e o Nexus é o servidor de biliotecas que serve o projeto com toda a infraestrutura de dependência. É muito difícil controlar tudo isso manualmente até mesmo em softwares de pequeno porte.

Outra prática de métodos ágeis utilizada foi a de integração contínua [13] [64]. Essa prática nos dá ao mesmo tempo automação na geração e publicação de versões da aplicação e métricas de qualidade de código. Para isso foi utilizado o Jenkins [37] e com Sonar [58]. Essas duas ferramentas juntas fornece muita dinâmica no desenvolvimento do software. Os analistas e interessados no projeto acompanhavam em tempo de desenvolvimento a evolução do sistema e os índices de qualidade que estavam sendo produzidos.

A arquitetura foi construída para potencializar as práticas utilizadas nos processos ágeis de desenvolimento de software. O software que utilizar essa arquitetura poderá explorar ao máximo desde testes automatizados com diversos cenários e necessidades até produção e coleta de estatísticas de qualidade de software. Tudo isso utilizando ferramentas open source sem custo de utilização. Sem dúvida uma das combinações mais baratas para se fazer software de qualidade.

CAPÍTULO 5

Trabalhos Futuros

A arquitetura poderia ter dado mais apoio à manutenção de formulários. Quando um formulário é alterado uma nova versão desse formulário poderia ser criada e versões anteriores dele permaneceriam em operação até expirar. Atualmente, após uma alteração de formulários o questionário automaticamente entre em vigor. Campos excluídos podem significar perda de informações. Esse controle de versões de formulários é complexo e a arquitetura poderia ajudar.

As alternativas das questões são muito simples. Alternativas mais completas como as de múltipla escolha aumentaria a aplicação da arquitetura em sistemas de pesquisa.

A análise de dados poderia ser realizada dentro do próprio sistema. Como o sistema já possui os dados coletados, os gráficos e tabelas com as estetísticas seriam extraídos de dentro dele e apresentados em qualquer navegador web. Isso eliminaria a necessidade de extração de dados.

Um dos principais focos da arquitetura foi o baixo custo. No entanto está cada vez mais viável a utilização de smartphones. A arquitetura poderia dar suporte para smartphones Android, iOS e Windows, pelo menos.

A exportação do banco de dados poderia acontecer para vários formatos compatíveis com softwares de análise estatística.

Seria também de grande valor a utilização de algoritimos de reconhecimento de imagens para apoio ao diagnóstico de pneumonia infantil.

As aplicações da arquitetura poderiam ser facilmente extendidas para outras finalidades. Sistemas de pesquisa em geral poderiam se beneficiar dela se não fossem as particularidases.

CAPÍTULO 6

Conclusão

A ausência de sistemas de pesquisa voltados para a saúde pública com baixo custo motivou a criação dessa arquitetura. As decisões arquiteturais foram tomadas a partir do sistema para o IPTSP-UFG para apoio à pesquisa das pneumonias na infância.

Atualmente doze enfermeiras, no papel de agentes de saúde, estão utilizando o sistema de pesquisa das pneumonias na infância em vários hospitais em Goiânia. Em menos de um ano de utilização já são quase 8 mil casos registrados. Essa pesquisa ainda continuará por mais um ano e outras já estão em planejamento.

Novas versões da arquitetura e do sistema surgirão com as novas pesquisas que estão em planejamento. É muito gratificante ver um trabalho acadêmico sendo aplicado em saúde pública e, de certa forma, até salvando vidas. Difilcilmente conseguríamos agregar tanto com uma motivação tão nobre quanto a diminuição da mortalidade infantil.

Tanto a arquitetura quanto o sistema desenvolvido estão contribuindo com a diminuição da mortalidade infantil causada por pneumonia. Outros sistemas utilizando essa arquitetura podem ser usados em outras instituições de saúde pública no Brasil ou até mesmo em outros países.

Referências Bibliográficas

[1] **Amazon beanstalk**. http://aws.amazon.com/elasticbeanstalk, último acesso em agosto de 2012.

[2] **Amazon ec2 iaas**. http://aws.amazon.com/ec2, último acesso em agosto de 2012.

[3] **Assembla**. http://www.assembla.com, último acesso em agosto de 2012.

[4] BECK, K. **Test Driven Development: By Example**. Addison-Wesley Professional, 2002.

[5] **Linux centos**. http://www.centos.org, último acesso em agosto de 2012.

[6] **Cloudbees**. http://www.cloudbees.com, último acesso em agosto de 2012.

[7] **Computação em nuvem**. http://pt.wikipedia.org/wiki/Computação em nuvem, último acesso em agosto de 2012.

[8] DAVID COHEN, M. L.; COSTA, P. **Agile software development**. *DACS State-of-the-Art/Practice Report*, 2003.

[9] DAVID M. AANENSEN1, DEREK M. HUNTLEY, E. J. F. F. A.-O. B. G. S. **Epicollect: Linking smartphones to web applications for epidemiology, ecology and community data collection**. *University of Oxford*, 2009.

[10] **Abache derby sql database**. http://db.apache.org/derby, último acesso em agosto de 2012.

[11] DOMINIC JOHN REPICI, D. B. **The Comma Separated Value (CSV) File Format**. *Creativyst Docs*, 2014.

[12] DOUG ROSENBERG, K. S. **Applying Use Case Driven Object Modeling with UML**. Addison Wesley, 2001.

[13] DUVALL, P. M. **Continuous Integration: Improving Software Quality and Reducing Risk**. Addison-Wesley Professional, 2007.

[14] **Google plugin for eclipse**. http://developers.google.com/eclipse, último acesso em agosto de 2012.

[15] **Eclipse jpa support**. http://www.eclipse.org/webtools/dali, último acesso em agosto de 2012.

[16] ELISA H. M. HUZITA, TANIA F. C. TAIT, T. E. C.-M. A. Q. **Um Ambiente de Desenvolvimento Distribuído de Software - DiSE**. *Universidade Estadual de Maringá*, 2007.

[17] ERICH GAMMA, K. B. **Contributing to Eclipse: Principles, Patterns, and Plug-Ins**. Addison-Wesley Professional, 2003.

[18] FOWLER, M. **UML Distilled: A Brief Guide to the Standard Object Modeling Language**. Addison Wesley, 2003.

[19] **Fragmentation of mobile applications**. http://www.comp.nus.edu.sg/ damithch/df/device-fragmentation.htm, último acesso em agosto de 2012.

[20] GEER, D. **Eclipse becomes the dominant java ide**. *IEEEXplore Digital Library*, 2005.

[21] **Gogrid iaas**. http://www.gogrid.com, último acesso em agosto de 2012.

[22] **Google app engine**. http://developers.google.com/appengine, último acesso em agosto de 2012.

[23] **Google plugin for eclipse**. http://developers.google.com/eclipse, último acesso em agosto de 2012.

[24] **Google trends**. http://www.google.com/trends, colocando o Eclipse e seus concorrentes. Último acesso em agosto de 2012.

[25] **Google web toolkit**. http://developers.google.com/web-toolkit, último acesso em agosto de 2012.

[26] **Gwt designer**. https://developers.google.com/web-toolkit/tools/gwtdesigner, último acesso em agosto de 2012.

[27] HANS-ERIK ERIKSSON, M. P. **Business Modeling with UML: Business Patterns at Work**. John Wiley and Sons, 2000.

[28] HEMRAJANI, A. **Agile Java Development with Spring, Hibernate and Eclipse**. Sams Indianapolis, 2006.

[29] **Heroku**. http://www.heroku.com, último acesso em agosto de 2012.

[30] **Infraestrutura como serviço**. http://en.wikipedia.org/wiki/Cloud_computing, último acesso em agosto de 2012.

[31] **J2me polish**. http://www.enough.de/products/j2me-polish, último acesso em agosto de 2012.

[32] **Jme: Java micro edition**. http://www.oracle.com/technetwork/java/javame, último acesso em agosto de 2012.

[33] **Jbehave**. http://jbehave.org, último acesso em agosto de 2012.

[34] **Jdt: Java development tools**. http://eclipse.org/jdt, último acesso em agosto de 2012.

[35] **Jee: Java enterprise edition**. http://www.oracle.com/technetwork/java/javaee, último acesso em agosto de 2012.

[36] **Jelastic**. http://jelastic.com/, último acesso em agosto de 2012.

[37] **Jenkins**. http://jenkins-ci.org, último acesso em agosto de 2012.

[38] **Jetty: Servidor de aplicação java**. http://www.eclipse.org/jetty, último acesso em agosto de 2012.

[39] **Junit**. http://www.junit.org, último acesso em agosto de 2012.

[40] KHAWAR ZAMAN AHMED, C. E. U. **Developing Enterprise Java Applications with J2EE and UML**. Addison-Wesley, 2002.

[41] LI LIU, HOPE L JOHNSON, S. C. J. P. S. S.-J. E. L. I. R. H. C. R. C. M. L. C. M. R. E. B. **Global, regional, and national causes of child mortality: an updated systematic analysis for 2010 with time trends since 2000**. *The Lancet*, 2012.

[42] **linode**. http://www.linode.com, último acesso em agosto de 2012.

[43] **M2e**. http://www.eclipse.org/m2e, último acesso em agosto de 2012.

[44] MARTIN, R. C. **Agile Software Development, Principles, Patterns, and Practices**. Prentice Hall, 2002.

[45] **Maven**. http://maven.apache.org, último acesso em agosto de 2012.

[46] **Motorola ex115**. http://www.motorola.com/Consumers/BR-PT/Consumer-Product-Services/Mobile-Phones/MOTO-EX115-BR-PT, último acesso em agosto de 2012.

[47] **Nexus**. http://www.sonatype.org/nexus, último acesso em agosto de 2012.

[48] ORACLE. **Learn About Java Technology**. *Oracle*, 2012.

[49] **Plataformas como serviço em java**. http://www.infoq.com/articles/paas_comparison, último acesso em agosto de 2012.

[50] PAUL C. ZIKOPOULOS, GEORGE BAKLARZ, D. S. **Apache Derby: Off to the Races: Includes Details of IBM Cloudscape**. IBM Press, 2006.

[51] **Rackspace iaas**. http://www.rackspace.com, último acesso em agosto de 2012.

[52] **Openshift**. http://openshift.redhat.com, último acesso em agosto de 2012.

[53] ROBERT COOPER, C. C. **GWT In Practice**. Manning Publications, 2008.

[54] ROBERT HANSON, A. T. **GWT In Action**. Manning Publications, 2007.

[55] ROD JOHNSON, JUERGEN HOELLER, A. A. T. R. C. S. **Professional Java Development with the Spring Framework**. Wrox, 2005.

[56] SEBASTIAN NEUBERT, DAGMAR ARNDT, K. T. R. S. **Mobile real-time data acquisition system for application in preventive medicine**. *Institute of Preventive Medicine – University of Rostock*, 2009.

[57] SOFTWARE, E. **Mobile Developer's Guide to The Galaxy**. Enough Software, 2012. http://www.enough.de/fileadmin/uploads/dev_guide_pdfs/Mobile_Developers_Guide_10thEdition.p

[58] **Sonar**. http://www.sonarsource.org, último acesso em agosto de 2012.

[59] **Spring framework**. http://www.springsource.org/spring-framework, último acesso em agosto de 2012.

[60] **Subversion**. http://subversion.tigris.org, último acesso em agosto de 2012.

[61] **Eclipse subversive**. http://www.eclipse.org/subversive, último acesso em agosto de 2012.

[62] **Eclipse subversive**. http://svnkit.com, último acesso em agosto de 2012.

[63] UNICEF. **Pneumonia and diarrhoea – Tackling the deadliest diseases for the world's poorest children**. *UNICEF Report*, 2012.

[64] V. HARDION, A. BUTEAU, N. L. G. A. S. P.-J. Z. S. L. **Assessing software quality at each step of its lifecycle to enhance reliability of control systems**. *Synchrotron Soleil, Gif/Yvette, France*, 2008.

[65] VIRKUS, R. **Pro J2ME Polish – Open Source Wireless Java Tools Suite**. Apress, 2007.

[66] **Cloud foundry**. http://www.cloudfoundry.com, último acesso em agosto de 2012.

[67] WICKESSER, C. **Google chose jetty for app engine: http://www.infoq.com/news/2009/08/google-chose-jetty, último acesso em agosto de 2012**. *InfoQ*, 2009.

[68] WORLD, C. **11 categorias de cloud computing: http://computerworld.uol.com.br/tecnologia/2010/03/03/11-categorias-de-cloud-computing/, último acesso em agosto de 2012**. *Computer World*, 2010.

[69] **Eclipse web tools platform**. http://www.eclipse.org/webtools, último acesso em agosto de 2012.