Universidade Federal do Pampa

Gean Trindade Pereira

Uma Abordagem Distribuída para o Auxílio ao Tratamento da Pneumonia Adquirida na Comunidade

Alegrete

2015

Gean Trindade Pereira

Uma Abordagem Distribuída para o Auxílio ao Tratamento da Pneumonia Adquirida na Comunidade

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Dr. Juliano F. Kazienko

Alegrete

2015

Gean Trindade Pereira

Uma Abordagem Distribuída para o Auxílio ao Tratamento da Pneumonia Adquirida na Comunidade

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 03 de Dezembro de 2015.

Banca examinadora:

Dr. Juliano F. Kazienko
Orientador

Dr. Ewerson L. S. Carvalho
UNIPAMPA

Dr. Igor A. C. Melnik
UNIPAMPA

Este trabalho é dedicado a minha família que sempre me apoiou em minhas decisões me passando confiança e segurança, e aos meus amigos que não apenas estiveram comigo nos momentos de felicidade e euforia mas principalmente estavam ao meu lado nos momentos difíceis e mais desafiadores que passei nestes 4 anos de bacharelado.

Agradecimentos

Sem dúvida alguma não conseguiria superar todas as barreiras que ultrapassei até o dia de hoje em que escrevo esses agradecimentos, sem a presença e o apoio das pessoas queridas que me ajudaram chegar até aqui. São muitas as pessoas importantes que gostaria de agradecer, uma delas é a pessoa mais importante e que mais admiro desde que tive meu primeiro vislumbre de consciência, minha mãe Tânia. Saibas que és a pessoa que me inspira a melhorar a cada dia e que me espelho em ti desde muito pequeno, e só espero poder retribuir um dia todo o bem que já me fez. Gostaria de agradecer ao meu pai Glênio que sempre fez de tudo para me oferecer as melhores condições, sendo um exemplo de pessoa pra mim. Agradeço a minhas irmãs Géssica e Gabriele por serem compreensivas comigo nos momentos de estresse e por se fazerem presentes sempre que precisei de apoio.

Agradeço aos meus grandes amigos e colegas Alex, Douglas e Miguel por todos os momentos em que rimos e satirizamos as dificuldades enfrentadas durante esse período tão conturbado de TCC. Agradeço ao grande amigo Gabriel Moro, pessoa pela qual tenho um grande respeito e admiração, e que em muitos momentos de angústia e tristeza frente as dificuldades me aconselhou e motivou, dando-me forças para continuar. Agradeço ao saudoso amigo e colega Jean Rangel, um amigo companheiro, confiável e uma grande pessoa com a qual convivi e aprendi muito durante esses 4 anos de perrengues e histórias vivenciadas. Agradeço também com carinho meus amigos de longa data Diego, Eduardo, Gustavo, Luciano e Vitor pelas festas, jogatinas e as conversas que mesmo a distância nunca perderam o significado e a importância para mim.

Não poderia deixar de agradecer também ao meu Orientador Juliano Kazienko por todo o suporte que me proporcionou durante o período de TCC, e que sem toda a sua dedicação e paciência a conclusão desse trabalho não seria possível. Por fim mas não menos importante, agradeço imensamente ao professor Daniel Welfer que viabilizou o tema desse trabalho e que sem o seu apoio tudo isso não teria se realizado.

"O mundo não é um grande arco-íris. É um lugar sujo, é um lugar cruel. Que não quer saber o quanto você é durão. Vai botar você de joelhos e você vai ficar de joelhos para sempre se você deixar. Você, eu, ninguém vai bater tão duro como a vida. Mas não se trata de bater duro. Se trata de quanto você aguenta apanhar e seguir em frente. O quanto você é capaz de aguentar e continuar tentando. É assim que se consegue vencer!
(Rocky V, 1990)

Resumo

A Pneumonia Adquirida na Comunidade (PAC) é uma infecção pulmonar grave com um grande número de ocorrências ao redor do mundo. No Brasil é a quarta causa de morte geral mais recorrente, e nos EUA é a líder causadora de mortes por doenças infecciosas com aproximadamente 600.000 hospitalizações por ano. Tratar a PAC não é uma tarefa trivial pois envolve uma diversidade de fatores, e por esse motivo são criados protocolos médicos contendo uma série de regras, algoritmos e estratificações que visam orientar os médicos no tratamento da doença. Apesar de úteis esses protocolos são complexos e extensos, o que torna sua consulta difícil e demorada. Em resposta a isso *softwares* que automatizam esses protocolos tornando sua consulta mais rápida e simples têm sido propostos. Entretanto a grande maioria não explora recursos como o compartilhamento de dados, algo amplamente benéfico se aplicado a área médica. Como objetivo principal deste trabalho é apresentado um sistema distribuído composto por um aplicativo móvel Android denominado *SysPAC*, que se comunica via Wi-Fi de forma segura através da troca de mensagens cifradas com um *Rest Web Service* Java. Esse sistema automatiza um protocolo médico de tratamento da PAC que auxilia médicos no gerenciamento de pacientes portadores da doença. A metodologia utilizada nesse trabalho incluí a utilização de um protocolo de revisão sistemática que auxiliou a busca por referencial teórico, além da adoção do processo iterativo e incremental *OpenUP* que guiou o desenvolvimento do sistema proposto. Para validar a ferramenta fora realizado comparações de desempenho/armazenamento/consumo de dados na rede com outros *softwares* de cunho médico semelhantes, além da realização de um experimento em ambiente controlado que contou com a presença de 6 voluntários que executaram tarefas pré-definidas na ferramenta a fim de validar aspectos de usabilidade do aplicativo. Como resultados obtidos foi possível concluir que a ferramenta utiliza poucos recursos do Android em relação a outras ferramentas semelhantes, além de apresentar uma progressão de aprendizado rápida visto a evolução do desempenho dos usuários conforme executavam tarefas. Além disso, com a utilizado de um questionário submetido aos usuários a fim de obter a sua percepção sobre a ferramenta, foi possível concluir que o aplicativo atendeu a seu propósito.

Palavras-chave: Pneumonia Adquirida na Comunidade. Sistema de Suporte a Decisão. Sistema Distribuído. Android. Segurança.

Abstract

The Community Acquired Pneumonia (CAP) is a serious lung infection with a great number of events around the world. In Brazil is the fourth cause of overall death most frequent, and in the USA is the leading cause of deaths from infectious diseases with approximately 600,000 hospitalizations per year. Treat CAP is not a trivial task because it involves a variety of factors, so therefore are created medical protocols containing a set of rules, algorithms and stratifications that aim to guide doctors in the treatment of disease. Although useful these protocols, they are complex and extensive, which makes difficult and time-consuming their query. In order to resolve that, softwares that automate these protocols making faster and simpler their query have been proposed. However the vast majority does not exploit features such as data sharing, something widely beneficial if applied to the medical field. As the main objective of this work is presented a distributed system composed of a Android mobile app called SysPAC, which communicates via Wi-Fi securely through the exchange of encrypted messages with a Rest Web Service Java. This system automates a medical treatment protocol of CAP that assists physicians in manage patients with the disease. The methodology used in this study include the use of a systematic review protocol that helped the search for theoretical framework, besides the adoption of iterative and incremental process OpenUP that guided the development of the proposed system. To validate the tool was conducted performance comparison / Storage / data consumption on the network with similar medical softwares, besides being conducted an experiment in a controlled environment that included the presence of 6 volunteers who carried out pre-defined tasks in the tool in order to validate the application usability aspects. As results it was concluded that the tool uses a few resources from Android compared to other similar tools, and has a quick learning progression seen the evolution of the users performance as performed tasks. Moreover, with the use of a questionnaire submitted to users in order to get their perception about the tool, it was concluded that the application met its purpose.

Key-words: Community Acquired Pneumonia. Decision Support System. Distributed System. Android. Security.

Lista de ilustrações

Lista de tabelas

Lista de siglas

API *Application Programming Interface*

EUA Estados Unidos da América

HNSC Hospital Nossa Senhora da Conceição

HTTP *Hypertext Transfer Protocol*

IP *Internet Protocol*

JSON *JavaScript Object Notation*

JVM *Java Virtual Machine*

NRTO Normas e Rotinas Técnico-Operacionais

PAC Pneumonia Adquirida na Comunidade

PAH Pneumonia Adquirida no Hospital

REST *Representational State Transfer*

RTT *Round Trip Time*

SGBD *Sistema de Gerenciamento de Banco de Dados*

SOAP *Simple Object Access Protocol*

UML *Unified Modeling Language*

URL *Uniform Resource Locator*

UTI Unidade de Terapia Intensiva

XML *eXtensible Markup Language*

Sumário

1 Introdução

Apesar dos contínuos avanços alcançados no campo da medicina, infecções pulmonares que apresentam altos índices de mortalidade como a pneumonia ainda são comuns em grande parte do mundo (AULER et al., 2010). Os números envolvendo a doença são preocupantes. A pneumonia é a causa mais frequente de internação entre as doenças respiratórias, sendo a causa de 47.661 hospitalizações no ano de 2007, considerando somente o estado do Rio Grande do Sul (AULER et al., 2010). Esse cenário se torna ainda mais preocupante quando analisado em escala nacional. No Brasil ela é a quarta causa de morte geral mais recorrente, atingindo os números de 24.756 casos de morte em 2005 (AULER et al., 2010). Já a incidência mundial estimada de casos de pneumonia que resultam em morte é de 12/1.000 pacientes/ano, o que representa 1.920.000 casos/ano no Brasil (AULER et al., 2010).

A pneumonia é uma inflamação ou infecção dos pulmões mais comumente causada por uma bactéria ou vírus (ER; YUMUSAK; TEMURTAS, 2010). Por sua vez, a Pneumonia Adquirida na Comunidade (PAC) é dita como um tipo de pneumonia que atinge pessoas que se encontram fora do ambiente hospitalar. A PAC é a doença infecciosa mais comum nos países ocidentais segundo Lim et al. (2009), sendo a líder causadora de mortes por doenças infecciosas nos EUA (WELFER; SILVA; KAZIENKO, 2014). Estima-se que mais de 900.000 casos ocorrem todo ano em pessoas com idade maior que 65 anos nos EUA, com cerca de 600.000 hospitalizações (WELFER; SILVA; KAZIENKO, 2014) (BUTT; SWIATLO, 2011) (MAJUMDAR et al., 2006) (BURMAN; WRIGHT, 2007) (JACKSON et al., 2004).

Outro fator preocupante em relação a PAC é o fator econômico. Estima-se que os custos com o seu tratamento estejam por volta dos 8 bilhões de dólares por ano nos EUA, sendo que mais de 90 % desse custo está relacionado a decisão do médico pela admissão do paciente ao tratamento ambulatorial, transformando essa decisão em algo extremamente delicado (RAUT et al., 2009).

Para que profissionais da saúde possam diagnosticar e tratar a PAC é comum que sejam definidas diretrizes em um protocolo interno ao órgão de saúde – seja esse uma clínica, hospital ou outra unidade de cuidados – que irão guiar os profissionais no tratamento da doença. Esses protocolos são documentos complexos que visam especificar regras, algoritmos e estratificações, a fim de orientar os profissionais da saúde em como proceder para diagnosticar e tratar pacientes (WELFER; SILVA; KAZIENKO, 2014). Tendo em vista que esses protocolos possuem uma documentação detalhada e complexa, a consulta de informações de forma rápida é dificultada. Fatores como a agilidade na

execução de procedimentos médicos são extremamente relevantes no contexto hospitalar, visto que o tempo é um fator de grande importância quando médicos lidam com a condição da saúde de pacientes. Além disso, gerenciar uma quantidade tão detalhada e grande de informações de maneira a fazer isso em tempo hábil e com acurácia é um grande desafio, principalmente para profissionais menos experientes e habituados com o uso desses protocolos.

A utilização de sistemas de informação aplicados a esses cenários pode trazer muitos benefícios acerca de tornar o processo menos demorado, cansativo e sujeito a falhas. O uso da computação móvel, por exemplo, possui grande aplicação na área da saúde conforme apontam pesquisas (SOARES et al., 2013). O uso de um sistema móvel além de auxiliar consideravelmente na diminuição das taxas de erros durante a execução de procedimentos médicos rotineiros, facilita o acesso a informação visto que é um meio mais prático – diferente de sistemas *Desktop* – e proporciona maior dinamismo aos ambientes hospitalares (SOARES et al., 2013).

Aliado a esses fatores, utilizar recursos como o compartilhamento de dados potencializa os benefícios que sistemas móveis podem prover. Sistemas de apoio médico que utilizam de uma abordagem distribuída ao invés de serem *standalone*[1], além de contar com uma base de dados mais rica não limitando-se apenas as informações armazenadas localmente, podem viabilizar: (1) A criação de históricos de diagnósticos de pacientes, uma vez que vários sistemas trabalham em conjunto compartilhando dados a fim de popular uma base de dados comum; (2) A não duplicação de registros médicos, uma vez que o recurso da comunicação entre os sistemas se faz presente, os mesmos podem se coordenar a fim de evitar essas falha; E (3) Um melhor tratamento de falhas no que diz respeito à perda de informações já que em uma abordagem distribuída os dados gerenciados por uma sistema podem estar armazenados em vários dispositivos.

Considerando o que fora descrito anteriormente e tendo em vista que não foram encontradas soluções que abordem o tratamento da PAC, de forma a não limitar-se apenas a soluções *standalone*, e preocupando-se com a segurança das informações de pacientes, nosso trabalho surge com o intuito de propor uma arquitetura distribuída que considera o compartilhamento de dados acerca de diagnósticos e tratamentos da PAC com segurança.

1.1 Declaração do Problema de Pesquisa

Essa pesquisa fora originada do trabalho de Welfer, Silva e Kazienko (2014) que apresentam uma solução móvel e *standalone* para o tratamento da PAC. Este estudo não visa resolver um problema já abordado pelos autores ao automatizar o protocolo de Auler

[1] **Sistema *Standalone*** - Aquele que consegue funcionar *offline* sem utilização de uma conexão de rede e sem compartilhar dados.

et al. (2010). O presente trabalho se estende ao que foi apresentado pelos autores, onde é proposto uma solução também móvel – porém para a plataforma Android e não para iOS como o proposto pelos autores – onde o foco da pesquisa é o compartilhamento seguro de dados de diagnósticos e tratamentos médicos da PAC. O problema de pesquisa que guia o estudo apresentado nesse trabalho é: "como resolver o problema do compartilhamento de dados de diagnósticos e tratamentos médicos da PAC de maneira segura?".

1.2 Objetivo Geral

O presente estudo consiste em propor um sistema distribuído que automatiza os procedimentos de tratamento da PAC apresentados no protocolo médico proposto por Auler et al. (2010). Esse sistema conta com um aplicativo desenvolvida na plataforma Android denominado *SysPAC*, que se comunica com um *Rest Web Service* desenvolvido na plataforma Java.

1.3 Objetivos Específicos

A fim de satisfazer o objetivo geral apresentado, os objetivos específicos dessa proposta são:

- Estudar a PAC e seu protocolo médico de tratamento proposto por Auler et al. (2010);
- Projetar a arquitetura do sistema visando contemplar os requisitos necessários a abordagem distribuída visada nesse trabalho;
- Implementar a primeira versão totalmente funcional do sistema para ser validada em ambiente de laboratório controlado;
- Refatorar e finalizar a implementação do sistema, levando em consideração as observações oriundas do experimento realizado em ambiente controlado.

1.4 Organização do Documento

O presente trabalho está estruturado da seguinte maneira: No Capítulo 2 são apresentados os principais conceitos que norteiam essa pesquisa. Em seguida no Capítulo 3, é apresentada uma síntese do estudo dos trabalhos relacionados encontrados, seguido do Capítulo 4 onde é apresentado o sistema distribuído proposto nesse trabalho. No Capítulo 5 é dado enfoque na avaliação da ferramenta, e por fim no Capítulo 6 é realizado um fechamento do trabalho onde são descritos os resultados obtidos, limitações, desafios e os trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são abordados os principais conceitos que fundamentam essa pesquisa. Algumas informações sobre a Pneumonia Adquirida na Comunidade (PAC) são apresentados detalhadamente na seção 2.1, juntamente com o protocolo de tratamento da doença na subseção 2.1.1. Por seguinte, são introduzidos os conceitos relacionados a Sistemas de Suporte a Decisão na seção 2.2, Computação Móvel na seção 2.3, onde também é descrito a arquitetura e demais características do sistema operacional Android na subseção 2.3.1, Sistemas Distribuídos na seção 2.4 e por fim, são abordados conceitos sobre Segurança da Informação na seção 2.5.

2.1 Embasamento da Pneumonia

Segundo Wardlaw, Johansson e Hodge (2006), a pneumonia é uma forma grave de infecção respiratória aguda que afeta os pulmões. Os pulmões são constituídos por milhares de tubos (brônquios) que se subdividem em pequenas vias aéreas (bronquíolos), e terminam em pequenos cavidades (alvéolos) (WARDLAW; JOHANSSON; HODGE, 2006). Os alvéolos contêm células capilares onde o oxigênio é adicionado ao sangue e o dióxido de carbono é removido. Quando uma pessoa tem pneumonia, pus e fluídos enchem os alvéolos de um ou ambos os pulmões, o que dificulta a absorção de oxigênio tornando a respiração difícil (WARDLAW; JOHANSSON; HODGE, 2006).

As informações sobre as causas específicas da pneumonia são limitadas e muitas vezes difíceis de interpretar. No entanto, sabe-se que na maioria dos casos a pneumonia é causada pelo agente patogênico bacteriano *Streptococcus pneumoniae* (WARDLAW; JOHANSSON; HODGE, 2006). Além disso, a pneumonia pode ser causada por outras bactérias, vírus e até fungos.

A pneumonia bacteriana geralmente gera febre alta e respiração demasiadamente rápida em pacientes. Já as infecções virais muitas vezes tendem a piorar gradualmente o quadro do paciente ao longo do tempo. Alguns sintomas comuns de pneumonia em crianças e recém-nascidos, por exemplo, incluem respiração acelerada, tosse, febre, calafrios, dores de cabeça, perda de apetite e chiado no peito (WARDLAW; JOHANSSON; HODGE, 2006).

A pneumonia também pode ser classificada de acordo com o local de origem, o que acaba diferindo o seu tratamento e diagnóstico. A pneumonia nosocomial ou Pneumonia Adquirida no Hospital (PAH), corresponde a um tipo de pneumonia que se desenvolve entre 48-72 horas após a admissão hospitalar (MARAGOUDAKIS et al., 2004). Já a

Pneumonia Adquirida na Comunidade (PAC) é contraída na comunidade, mais especificamente fora do ambiente hospitalar ou em alguma instituição de cuidados como afirma Maragoudakis et al. (2004), que é o caso de instituições que fornecem serviços de *Home-Care* (Serviço de saúde a distância que visa à continuidade do tratamento hospitalar no domicílio).

Para que médicos e demais profissionais da saúde possam diagnosticar e tratar doenças infecciosas (como a PAC), é comum que sejam definidas diretrizes internas aos hospitais para auxiliá-los (LODE, 2007). A seguir na subseção 2.1.1 é descrito justamente o que são essas diretrizes/protocolos de tratamento e como se relacionam com a presente pesquisa.

2.1.1 Protocolo de Tratamento da Pneumonia Adquirida na Comunidade

Conforme já comentado anteriormente, essas diretrizes, ou protocolos como são comumente chamados, são documentos detalhados que contêm uma série de informações a respeito de como estratificar o risco das doenças, percentagem de letalidade, diagnósticos, algoritmos de decisão para tratamentos, e mais uma série de informações relevantes para o diagnóstico e tratamento de pacientes, que variam para cada doença e instituição de saúde.

Nesse trabalho, o protocolo proposto por Auler et al. (2010) é utilizado como referencial para essa pesquisa, sendo esse um conjunto de Normas e Rotinas Técnico-Operacionais (NRTO) do Programa de Controle de Infecção Hospitalar do Hospital Nossa Senhora da Conceição (HNSC) de Porto Alegre/RS para o tratamento da PAC. Essas NRTO abordam o tratamento de pacientes adultos com PAC, e não se aplicam a todos os pacientes portadores da doença, mas sim a um grupo de pacientes nos quais o diagnóstico etiológico não é óbvio. O algoritmo de decisão presente no protocolo é basicamente dividido em 3 partes: (1) Estratificação da gravidade da PAC, risco de morte e decisão da necessidade de tratamento hospitalar; (2) Avaliação da etiologia mais provável; e (3) Escolha do tratamento inicial. Essas três partes não são necessariamente executadas nessa ordem, pois o protocolo deve ser flexível as necessidades dos médicos.

O primeiro procedimento médico apresentado no protocolo é a estratificação de risco da PAC que é alcançada através da soma das variáveis presentes na Tabela 1. Na Tabela 1 estão presente 20 características a serem avaliadas de acordo com as especificidades de cada paciente, como por exemplo, fatores demográficos, comorbidades, dados resultantes de exames físicos, laboratoriais e radiológicos. Quando uma condição específica do paciente coincide com uma das características analisadas na Tabela 1, a pontuação especificada para essa característica (coluna "Pontuação"da Tabela 1) é adicionada a soma para a realização da estratificação.

Tabela 1 – Estratificação de Risco da PAC.

Característica	**Pontuação**
Fator Demográfico	
Idade em anos (homens)	Idade
Idade em anos (mulheres)	Idade - 10
Residência em asilo	+10
Comorbidades	
Neoplasia	+30
Hepatopatia	+20
Insuficiência cardíaca congestiva	+10
Doença cerebrovascular	+10
Doença renal	+10
Achado no Exame Físico	
Alteração mental	+20
Frequência respiratória $\geq$ 30 mpm	+20
Pressão arterial sistólica $<$ 90 mmHg	+20
Temperatura $<$ 35 °C ou $>$ 40 °C	+15
Frequência cardíaca $\geq$ 125 bpm	+10
Achados Laboratoriais e Radiológicos	
pH arterial $<$ 7,35	+30
Uréia $>$ 10,7 mmol/L	+20
Sódio $<$ 130 mmol/L	+20
Glicose $\geq$ 250 mg/dl	+10
Hematócrito $<$ 30%	+10
PaO2 $<$ 60 mmHg	+10
Derrame pleural	+10

Fonte: Auler et al. (2010).

Após a soma das características do paciente ser concluída é realizada a verificação da classe de risco em que esse paciente se enquadra, onde de acordo com sua pontuação o paciente é classificado em uma das cinco classe de risco existentes conforme mostra a Tabela 2. Pacientes com pontuação até 69 pontos pertencem a classe de risco **I**, possuem índice de letalidade de 0,4%, e a sugestão de tratamento é Ambulatorial. Pacientes com pontuação igual à 70 pontos pertencem a classe de risco **II**, possuem índice de letalidade de 0,7%, e a sugestão de tratamento assim como a da classe **I** é Ambulatorial. Para pacientes com uma pontuação de 71 à 90 pontos a letalidade sobe para 2,8%, a sugestão de tratamento passa a ser Hospitalar breve e os pacientes são colocados na classe de

risco **III**. Já para pacientes com pontuação de 91 à 130 pontos a classe de risco é **IV**, o índice de letalidade é 8,5%, e o tratamento sugerido é o Hospitalar. Por fim, os pacientes colocados na classe de risco **V** são aqueles que possuem uma pontuação maior que 130 pontos, possuem letalidade de 31,1%, e a sugestão de tratamento é a Hospitalar.

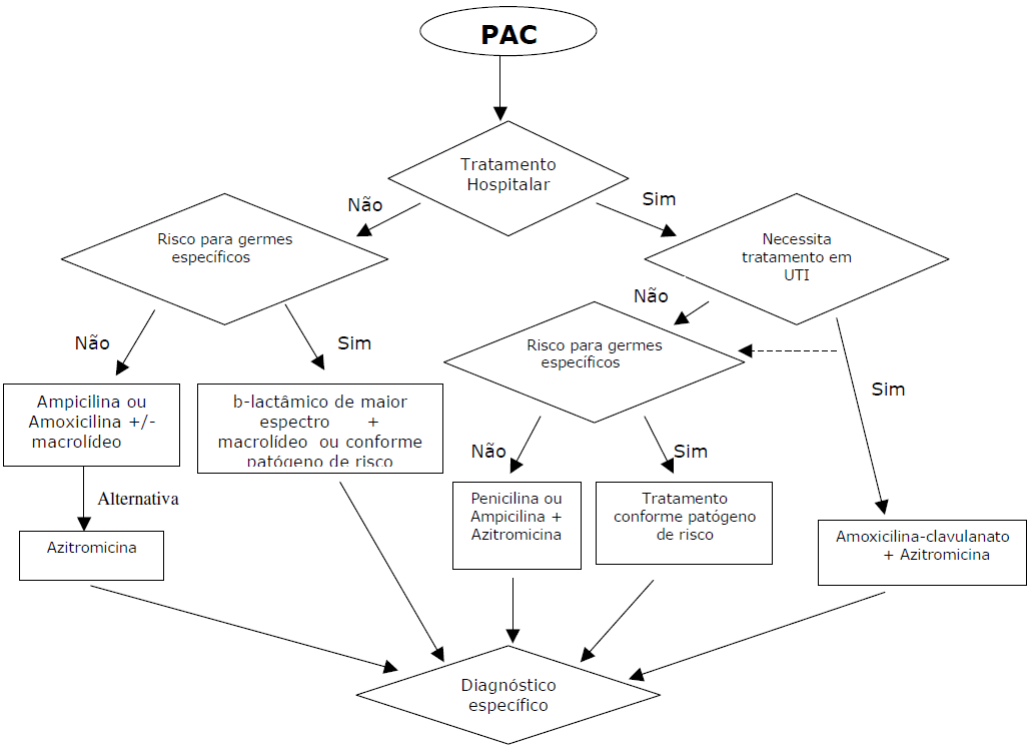
Tabela 2 – Classes de risco para pacientes com PAC.

Classe de Risco	**Pontuação**	**Letalidade**	**Tratamento**
I	-	0,4%	Ambulatorial
II	= 70	0,7%	Ambulatorial
III	71-90	2,8%	Hospitalar breve
IV	91-130	8,5%	Hospitalar
V	> 130	31,1%	Hospitalar

Fonte: Auler et al. (2010).

Independentemente se as sugestões de tratamento foram de internação hospitalar ou não, o procedimento presente no protocolo de Auler et al. (2010) para a escolha do tratamento inicial deve ser seguido. Ele é representado pelo algoritmo da Figura 1.

Figura 1 – Algoritmo de Decisão para o Tratamento da PAC.



Fonte: Auler et al. (2010).

Esse algoritmo foi desenvolvido por uma equipe interna do HNSC, e contém cinco níveis onde são determinados os tratamentos adequados ao paciente, de acordo com suas informações previamente obtidas.

O algoritmo em si é bastante simples e de fácil entendimento. A primeira decisão a ser tomada é determinar se o paciente necessita ou não de tratamento hospitalar. Caso seja necessário (o especialista pode consultar as informações do paciente obtidas através do uso das Tabela 1 e Tabela 2), é verificado se o paciente precisa de tratamento na Unidade de Terapia Intensiva (UTI). Em caso de internação, o tratamento recomendado é Amoxicilina-Clavulanato + Azitromicina. Contudo, algumas decisões do algoritmo dependem do conhecimento sobre os patógenos de risco, e para esses casos é necessário uma consulta de informações a respeito dos fatores de risco do patógeno e os tratamento recomendados.

Nesse contexto, a Tabela 3 presente no protocolo de Auler et al. (2010) lista os possíveis patógenos que causam a pneumonia, os fatores de risco, e o seu tratamento.

Tabela 3 – Tratamento da PAC conforme a Etiologia.

Patógeno	**Fatores de Risco**	**Tratamento**
Pneumococo	Patógeno número 1, sem necessidade de fatores de risco específicos.	Penicilina G cristalina 4 a 8 milhões de UI ao dia ou amoxicilina VO.
Pneumococo resistente à penicilina	Uso de antibióticos β-lactâmicos (penicilinas, cefalosporinas, aztreonam e cabapenêmicos) nos últimos 3 meses, doença imunossupressora, uso de corticóide e idade superior a 65 anos em alguns estudos (estes achados, entretanto não foram suficientes para que penicilina deixasse de ser o antibiótico de escolha para pneumonia pneumocócica).	Penicilina G cristalina 2 milhões de UI de 4/4 horas ou amoxicilina 1 g 8/8 horas (VO).
Hemophilus influenzae	Extremos de idade, doença causadora de alteração da estrutura do parênquima pulmonar (DPOC).	Ampicilina, amoxicilina ou Azitromicina.

Hemophilus influenzae produtor de β-lactamase (resistente à ampicilina)	Uso prévio de antibióticos β-lactâmicos nos últimos 3 meses.	Amoxicilina + Clavulanato.
Enterobactérias	Indivíduos residentes em instituições, doença cardio pulmonar, comorbidades médicas múltiplas, alcoolismo, uso recente de antibióticos.	Amoxicilina + Clavulanato.
Pseudomonas aeruginosa	Doença com comprometimento da estrutura do pulmão (ex: bronquiectasias), corticoterapia prolongada (>10 mg/dia de prednisona por mais de 2 semanas), desnutrição, uso de antibiótico de amplo espectro.	Piperacilina + Tazobactam.
Anaeróbios	Dentes sépticos, doença neurológica, alteração de consciência, alcoolismo, distúrbio da deglutição, neoplasia causando obstrução da via aérea.	Penicilina G cristalina 18 a 24 milhões UI ao dia.
Staphylococcus aureus	Drogadição, doença comprometimento da estrutura do pulmão (para MRSA: hospitalização recente).	Oxacilina 2g 4/4 horas, (Vancomicina se MRSA).
Legionella pneumophila	Água contaminada.	Azitromicina.
Chlamydia psittacci	Contato com pássaros.	Doxiciclina ou Macrolídeo.
Mycoplasma pneumoniae	Associado a microepidemias, principalmente familiares, acometendo mais crianças, adolescentes e adultos jovens.	Doxiciclina ou Macrolídeo.
Vírus	História de contato.	Herpesviroses (VZV, HSV): Aciclovir. Influenza A: Amantadina ou Oseltamivir. Influenza B: Oseltamivir.

Fonte: Auler et al. (2010).

2.2 Sistemas de Suporte a Decisão

Como a ferramenta proposta nesse trabalho visa oferecer um suporte a médicos no que diz respeito a decisões de tratamento de pacientes, cabe definir o que são *softwares de suporte a decisão*.

Segundo Schurink et al. (2005), sistemas de suporte a decisão são *softwares* com a capacidade de auxiliar usuários na tomada de decisão para problemas específicos, como por exemplo, determinar um diagnóstico a um paciente com uma doença infecciosa.

Esses sistemas podem oferecer inúmeros benefícios quando aplicados na área médica conforme aponta Schurink et al. (2005), pois vários profissionais da saúde acreditam que seu uso aplicado a medicina irá prover melhorias na qualidade dos cuidados com pacientes, oferecendo melhores decisões de tratamento e diminuindo custos.

Além disso, sistemas de suporte a decisão podem simplificar o acesso a dados necessários para a tomada de decisões clínicas, prover recursos que facilitem o trabalho dos especialistas, como ferramentas de gerenciamento de pacientes, agendas de atendimento, monitoramento de dados dos pacientes, além de poupar o tempo dos profissionais na realização dessas tarefas manualmente (PAYNE, 2000).

Nas seções a seguir são definidos alguns conceitos tecnológicos relacionados a abordagem adotada para a construção da ferramenta proposta. Como parte da solução proposta consiste na implementação de um aplicativo móvel, na seção 2.3 são definidos alguns conceitos e particularidades sobre a Computação Móvel, posteriormente na subseção 2.3.1 é discutido acerca do Sistema Operacional Android – plataforma utilizada para a implementação da Aplicação Cliente da abordagem móvel adotada pela ferramenta –, e em seguida, já que a ferramenta proposta segue uma abordagem distribuída, são abordados os conceitos sobre Sistemas Distribuídos na seção 2.4 e Segurança da Informação na seção 2.5.

2.3 Computação Móvel

Conforme já citado anteriormente, a ferramenta apresentada nessa pesquisa adota uma abordagem distribuída, onde parte de sua solução consiste em um aplicativo móvel.

Tal plataforma fora adotada justamente por apresentar grande praticidade de utilização e mobilidade, uma vez que suas aplicações se caracterizam por serem simples e intuitivas, podendo ser acessadas facilmente em "qualquer lugar", algo que não se aplica a computação convencional.

Sendo assim, nessa seção são discutidos os conceitos envolvendo a computação móvel juntamente com suas particularidades.

A computação móvel surgiu da miniaturização dos dispositivos convencionais como os computado *Desktop*, sendo motivada por seu dinamismo no meio cotidiano e pela possibilidade de se obter conectividade constante (COULOURIS; DOLLIMORE; KINDBERG, 2007). Segundo Coulouris, Dollimore e Kindberg (2007) ela surgiu com o paradigma de que seus usuários poderiam carregar seus computadores pessoais consigo e manter certa conectividade com outras máquinas.

Algo que é relacionado quando o assunto é computação móvel de forma recorrente é a comunicação com rede sem fio. Nos dias atuais os dispositivos móveis como *smartphones* se conectam a todo momento através de uma rede Wi-Fi à Internet. Sem o recurso de comunicação esses dispositivos perdem muito na questão de recursos, pois a todo momento são disponibilizados novas atualizações de *software* das fabricantes, aplicativos para variadas funções entre uma gama de outros recursos que apenas são obtidos através da utilização de uma comunicação em rede.

Aproveitando-se da miniaturização dos dispositivos e da conectividade fortemente presente nos dispositivos, profissionais da área médica passaram a utilizar cada vez mais os dispositivos móveis para auxiliá-los nos procedimentos médicos rotineiros, assim facilitando muito suas atividades no que diz respeito a agilidade, gerenciamento e acesso a informações (SOARES et al., 2013).

Alguns pontos importantes a considerar relacionados a computação móvel conforme aponta Mateus e Loureiro (1998) são:

- **Interface Limitada** - As dimensões dos dispositivos são consideravelmente menores em comparação com computadores *Desktop* convencionais, o que torna a área de interação com o usuário menor;
- **Processamento Limitado** - Devido suas dimensões menores os componentes de um dispositivo móvel são mais simplificados a fim de compor o espaço do aparelho, o que acaba impactando no poder de processamento dos componentes, capacidade de armazenamento, consumo energético entre outros fatores;
- **Consumo Energético** - Também devido a interface limitada, os dispositivos móveis possuem baterias menores e com menos capacidade de carga. Além disso, esses sistemas são construídos de acordo com estratégias para poupar energia em toda sua arquitetura, tanto em *hardware* quanto em *software*, acrescentando ainda mais complexidade a plataforma;
- **Dificuldade de Desenvolvimento** - No que diz respeito ao desenvolvimento de *software* para a plataforma deve ser levado em consideração alguns fatores como: (1) O poder de processamento necessário para execução do *software* (visto os recursos limitados da plataforma); (2) O consumo energético gerado pelo seu processamento

e utilização de recursos de comunicação (os sinais emitidos consomem energia); e o (3) Desempenho da aplicação (dificultado devido os recursos limitados);

- **Comunicação Limitada** - As redes sem fio utilizadas pelos dispositivos móveis possuem largura de banda limitada e elevadas taxas de erros em suas transmissões devido a interferências e quedas nas conexões;
- **Segurança nas Comunicações** - Em razão do uso frequente de redes sem fio os dispositivos móveis são mais sujeitos a ataques maliciosos, pois os dados trafegam por redes sem fio e podem ser interceptados facilmente se a comunicação não utilizar recursos de segurança como a criptografia.

2.3.1 Google Android

Como já citado anteriormente, o Sistema Operacional adotado como plataforma de desenvolvimento do aplicativo que automatiza o protocolo médico de tratamento da PAC também já citado, fora o Android. O Android fora adotado devido a dois principais fatores: (1) Popularidade de utilização, sendo um dos Sistemas Operacionais Móveis mais utilizados no mundo atualmente (LECHETA, 2013); e (2) Pela facilidade de desenvolvimento que sua detentora, a Google, oferece, pois o mesmo não necessita de uma licença paga para ser utilizado, e o mesmo vale para suas ferramentas de desenvolvimento. Dessa forma, nessa subseção são introduzidos alguns conceitos sobre esse sistema.

Segundo (LECHETA, 2013) o Android é um sistema operacional embarcado baseado em Linux para *smartphones* que se tornou uma das plataformas de desenvolvimento para aplicativos móveis mais populares atualmente devido sua variedade de recursos nativos, sua arquitetura poderosa, flexível, e por ser um sistema operacional de código aberto. Essa última característica é um dos principais atrativos do Android, pois a comunidade de desenvolvedores da plataforma pode contribuir ativamente para a evolução da mesma.

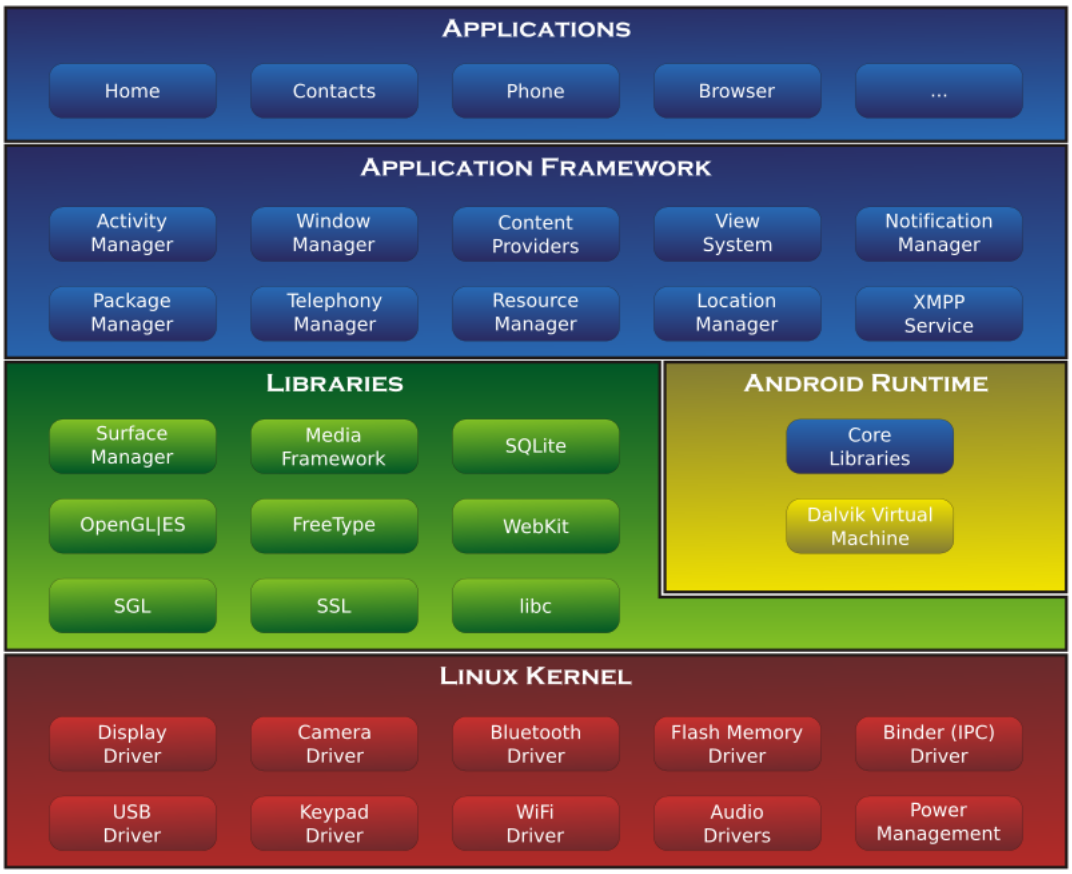
Por ser um sistema operacional baseado em Linux o Android evita que o desenvolvedor tenha que lidar com questões de gerenciamento de memória e processos, já que seu *Kernel* se encarrega destas questões permitindo que várias aplicações sejam executadas em paralelo ou até em segundo plano sem que o usuário perceba (LECHETA, 2013).

A linguagem padrão adotada pelo Android para o desenvolvimento das aplicações é o Java. Para que aplicações desenvolvidas em Java sejam executadas é necessário uma *Java Virtual Machine* (JVM). Porém, no Android não existe uma JVM e sim a Dalvik, uma máquina virtual otimizada para execução em dispositivos móveis (LECHETA, 2013). Dessa maneira segundo Lecheta (2013), o desenvolvedor se preocupa em apenas utilizar do Java e todos os seus recursos normalmente, e assim que o bytecode (.class) é gerado e compilado ele é então convertido pela máquina virtual Dalvik para o formato **.dex** (*Dalvik Executable*), que representa o aplicativo Android compilado. Logo após isso esse

arquivo **.dex** e demais arquivos da aplicação (imagens, vídeos etc), são compactados em um único arquivo **.apk** (*Android Package File*), que representa a aplicação final que pode ser instalada ou distribuída em *smartphones* suportados (LECHETA, 2013).

Segundo descreve Lecheta (2013), a arquitetura do Android é dividida em camadas onde cada uma delas é responsável por gerenciar seus respectivos processos. Uma representação da arquitetura do Android é exibida na Figura 2.

Figura 2 – Arquitetura em Camadas do Android.



Fonte: Google (2013).

Cada uma das camadas da arquitetura apresentada na Figura 2 possui uma série de componentes e responsabilidades. Baseando-se em Lecheta (2013), Google (2013), Tosin (2011) e Gadelha (2012), as características dessas camadas são listadas a seguir:

- **Applications** - Esta é a camada mais alto nível da arquitetura, trata-se da camada de interação com o usuário. Nela estão contidas as aplicações que executam diretamente sobre a plataforma Android. Algumas são aplicações já nativas como o navegador *browser*, gerenciador de contatos, agenda entre outros. Outras são

aplicações desenvolvidas por terceiros, como as obtidas através da plataforma de distribuição de aplicativos oficial do Android, a Google Play;

- **Application Framework** - Nesta camada se encontram as APIs do Android que representam os serviços essenciais da plataforma. Elas gerenciam e são utilizadas pelas aplicações que executam sobre a plataforma. Os desenvolvedores utilizam essas APIs para construir aplicações mais complexas. Entre alguns exemplos estão serviços de localização, notificação e telefonia;

- **Libraries** - Nesta camada estão as bibliotecas nativas do Android que fornecem serviços a vários dos componentes do sistema. Entre essas bibliotecas estão algumas escritas em C/C++ como Media Libraries (fornecem suporte a vários formatos de áudio e vídeo), LibWebCore (Motor *Web Browser* utilizado pelo *browser* nativo do Android) e 3D Libraries (Implementação baseada no OpenGL que oferece suporte a renderização 3D);

- **Android Runtime** - Nesta camada estão localizadas as bibliotecas *core* do Java além da própria máquina virtual Dalvik. Esta camada é responsável por possibilitar que as aplicações da plataforma sejam executadas. Presentes nessa camada estão bibliotecas de estrutura de dados, acesso a arquivos, acesso a rede, bibliotecas gráficas entre outras;

- **Linux Kernel** - Nesta camada se encontra o *Kernel* Linux que é a base do Android. Esse *Kernel* é baseado na versão padrão 2.6.24 do Linux. Nessa camada estão os programas que gerenciam memória, processos, configurações de rede e segurança, *display*, câmera, áudio entre outros vários *drivers* de *hardware*.

2.4 Sistemas Distribuídos

Conforme já citado anteriormente e descrito em detalhes no Capítulo 4, a ferramenta apresentada nesse trabalho adota um abordagem distribuída, e sendo assim, é caracterizada como um Sistema Distribuído. Portanto, essa seção visa conceituar o que são Sistemas Distribuídos e suas particularidades.

Segundo Coulouris, Dollimore e Kindberg (2007) um Sistema Distribuído consiste em componentes de *hardware* e *software* localizados em computadores independentes interligados em uma rede que se comunicam e coordenam suas ações através da troca de mensagens. Uma outra definição apresentada por Tanenbaum (2003), apresenta como sendo um Sistema Distribuído uma coleção de computadores independentes que aparecem para o usuário como sendo um único computador. Apesar dessas duas definições parecem diferir, as duas em sua essência fundamentam o que é um Sistema Distribuído,

apenas mudando a perspectiva de como ele é visto. Entre alguns exemplos de sistemas distribuídos estão a própria Internet, as Intranets, a Computação Móvel e Ubíqua (telefones, *smartphones*, computadores portáteis) e o Correio eletrônico (COULOURIS; DOLLIMORE; KINDBERG, 2007) (TANENBAUM, 2003).

Segundo Coulouris, Dollimore e Kindberg (2007) e Tanenbaum (2003), entre as características de um Sistema Distribuído estão:

- **Comunicação através de mensagens** - Os componentes de um Sistema Distribuído se comunicam através de mensagens pois não existem variáveis globais partilhadas;
- **Compartilhamento de recursos** - Tanto *hardware* (impressoras e discos rígidos), quanto *software* (banco de dados e ferramentas de trabalho cooperativo) são compartilhados entre componentes de um Sistema Distribuído, o que por sua vez levanta questões sobre segurança;
- **Concorrência no acesso a recursos** - Vários componentes se comunicam simultaneamente, logo é necessário coordenar o acesso aos recursos partilhados;
- **Comunicação assíncrona** - Não existe um relógio global, diferentes componentes possuem velocidades de processamento distintas e não existe um limite para o tempo de comunicação;
- **Tolerância a falhas** - Em Sistemas Distribuídos falhas são tratadas de forma independente, pois a falha de um componente não pode comprometer os demais;
- **Heterogeneidade** - Um Sistema Distribuído pode possuir diferentes tipos de rede, *hardware*, representações de dados, protocolos de comunicação entre outros. A fim de mascarar tais características distintas e possibilitar que sistemas diferentes se comuniquem é utilizado uma camada de *software* intermediária entre esses sistemas conhecida como *middleware*.

Uma tecnologia bastante conhecida na âmbito de desenvolvimento de *software* distribuído são os chamados *Web Services*. Essa tecnologia facilita o trabalho do desenvolvedor de *software* no que diz respeito a facilidade que ela trabalha com o envio/recebimento de mensagens numa abordagem distribuída, e como as gerencia. Essa tecnologia fora utilizada para implementar a parte da Aplicação Servidora da ferramenta aqui apresentada, conforme visto em detalhes no Capítulo 4. Sendo assim, a seguir na subseção 2.4.1 ela é conceituada.

2.4.1 Web Service

Um *Web Service* é um tipo de *software* conhecido como *middleware*, ou em outras palavras, é uma "camada de software"que tem como função criar uma interface de transparência para que sistemas distintos possam se comunicar. Com a utilização de um *Web Service* é possível usar da linguagem XML para realizar o envio e recebimento de dados, dessa maneira desacoplando os demais componentes de um sistema distribuído e possibilitando que o mesmo possa utilizar uma variedade de tecnologias e linguagens de programação diferentes, e mesmo assim graças ao uso de um *Web Service* a comunicação entre os sistemas é garantida e gerenciada por ele.

O modelo *Representational State Transfer* (REST) representa uma possibilidade para a criação de *Web Services* que se difere do modelo SOAP na utilização semântica dos métodos HTTP (*GET*, *POST*, *PUT* e *DELETE*) (COUTINHO, 2014). Esse modelo REST trabalha com o envio de pacotes de dados mais leves e simples, fazendo desnecessária a criação de camadas intermediárias como os envelopes utilizados no SOAP para encapsular os dados. No modelo REST uma requisição HTTP é como uma chamada de método (ou seja, uma operação) em um objeto (seria um recurso) contido no servidor (COUTINHO, 2014).

Segundo Coutinho (2014), suas principais características são: (1) Utiliza o HTTP para determinar a operação realizada sobre determinado recurso, como por exemplo, para recuperar um recurso utiliza-se *GET*, *POST* para criar, *PUT* para alterar e *DELETE* para deletar; (2) O recurso alvo da requisição é indicado direto na URL; (3) Parâmetros podem ser indicados na própria URL da requisição; (4) Tanto a parte cliente quanto o servidor devem se comunicar com o mesmo tipo de dado nas requisições, que geralmente são JSON ou XML.

2.5 Segurança da Informação

Para todo o sistema que de alguma forma troca informações através de uma rede com outro(s) sistema(s), como é o caso de um Sistema Distribuído, esse acaba sujeito a ataques de invasores mal intencionados.

Como a ferramenta aqui apresentada segue uma abordagem distribuída, e levando em considerações o viés médico da ferramenta, onde informações pessoais de pacientes são comunicadas, a preocupação com a segurança dessas informações é algo crítico no contexto dessa aplicação.

Sendo assim, nessa seção é falado um pouco sobre a segurança da informação em Sistemas Distribuídos, e também é apresentado algumas propriedades de segurança importantes a se garantir para que um sistema possa ser taxado como "seguro".

Quando existem sistemas que trocam informações através de um meio, como na rede mundial de computadores por exemplo, a preocupação com a segurança desses dados é comumente posta em pauta. Sistemas Distribuídos possuem um forte apelo por segurança, uma vez que as informações trocadas por eles podem ser de grande importância para certos fins. Sem que haja a devida preocupação de como essa informação é protegida de terceiros que não veriam ter acesso a ela, toda a arquitetura de um sistema acaba ficando desprotegida e sujeita a ataques de invasores.

Esses invasores podem ler informações que não estão autorizados, modificá-las ou até mesmo ter acesso direto a elas podendo transferi-las para outros locais. Dependendo do sistema ele pode lidar com dados altamente sigilosos como dados bancários ou médicos, o que torna o cenário ainda mais preocupante e sujeito a ataques. Os ataques contra a segurança de um Sistema Distribuído podem ser variados, como formas de intromissão de mensagens trocadas (lendo ou alterando informações), mascaramento, falsificação ou negando serviços (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Um ponto importante a se considerar quando se está pensando em segurança da informação, são quais propriedades de segurança se objetiva garantir. Nesse trabalho, visando garantir que a ferramenta de suporte médico tivesse o mínimo de segurança, garantiu-se que fosse atendida no momento a propriedade de Confidencialidade dos dados. Outras exemplos de propriedades são vistas a seguir (KUROSE, 2005):

- **Confidencialidade** - Somente o remetente e o destinatário alvo devem ter a capacidade de compreender as informações trocadas;
- **Autenticação** - Tanto remetente quanto destinatário devem confirmar a identidade da outra parte envolvida na comunicação;
- **Integridade** - Garantir que o conteúdo das mensagens trocadas não seja alterado durante a transmissão;
- **Disponibilidade** - Garantir a operacionalidade do sistema, certificando-se que esteja sempre disponível.

3 Trabalhos Relacionados

A fim de estudar o que já existe de mais semelhante em relação a nossa pesquisa, esse capítulo apresenta a síntese do estudo dos trabalhos relacionados encontrados tendo em vista o tema proposto. Para guiar a pesquisa por referencial teórico foi utilizado um protocolo de revisão sistemática (seção 3.1) que define uma série de parâmetros a serem atendidos e dessa forma auxiliando na busca por trabalhos relacionados. Os trabalhos selecionados foram divididos em dois grupos para melhor entendimento de seus propósitos, sendo classificados como Ferramentas de Suporte a Decisão de Diagnóstico da Pneumonia (seção 3.2) e Ferramentas de Suporte a Decisão de Tratamento da Pneumonia (seção 3.3).

3.1 Metodologia de Pesquisa

Com o intuito de guiar o desenvolvimento da pesquisa por trabalhos relacionados, utilizamos o Protocolo para Mapeamento Sistemático da Literatura de (PETERSEN et al., 2008). Nele são definidos o *Objetivo de Pesquisa* e as *Perguntas de Pesquisa* que devem ser respondidas pelos trabalhos encontrados, o *Mecanismo de Busca* utilizado, *Palavras-chaves*, *Critérios de Inclusão* e *Exclusão*. Fora realizado 6 mapeamentos (representados na imagem como M1, M2, M3 etc) a partir de um conjunto de palavras chaves para que fosse possível cobrir uma área maior de trabalhos relacionados. Esse protocolo com os parâmetros estabelecidos pode ser visualizado na Tabela 4.

Tabela 4 – Protocolo de Mapeamento Sistemático da Literatura.

Objetivo:	Estabelecer o estado da arte de sistemas informatizados que auxiliam o diagnóstico e/ou tratamento da Pneumonia Adquirida na Comunidade (PAC).
Questões de Pesquisa:	Q1) Como tem sido aplicado o uso de sistemas automatizados no auxilio ao diagnóstico e/ou tratamento da PAC? Q2) Que tecnologias, métodos, e abordagens esses sistemas utilizam para diagnosticar e/ou tratar da PAC? Q3) Esses sistemas de diagnóstico e/ou tratamento da PAC utilizam de arquiteturas que possibilitam o compartilhamento de dados? Q4) Esses sistemas se preocupam com segurança? De que maneira esses sistemas tratam questões de segurança com os dados dos pacientes armazenados?

Mecanismo de Busca:	Scholar Google (contabilização por base de publicação científicaa).
Palavras-chaves:	notítulo:"Community-Acquired Pneumonia"+ "Support system"(M1) "Community-Acquired Pneumonia"+ "Artificial Intelligence"(M2) "Community-Acquired Pneumonia"+ "decision support system"+ "exchange data"(M3) "Community-Acquired Pneumonia"+ "decision support system"+ "client-server"(M4) notítulo:"Mobile"+ "pneumonia"+ "decision support system"(M5) notítulo:"Mobile"+ notítulo:"pneumonia"(M6)
Critérios de Inclusão:	Artigos completos com no mínimo 6 (seis) páginas. Abordar o uso de algum sistema informatizado que auxilie no diagnóstico ou tratamento da PAC. Artigos que abordem conceitos e dados quantitativos sobre a PAC.
Critérios de Exclusão:	Artigos que sejam outros mapeamentos ou revisões sistemáticas. Artigos repetidos ou versões resumidas de trabalhos completos.

3.2 Suporte a Decisão de Diagnóstico da Pneumonia

Nessa seção são apresentados os trabalhos que tem como objetivo ou de alguma maneira fornecem suporte a decisão no diagnóstico da pneumonia a médicos e demais profissionais da saúde.

3.2.1 STRadio

Soares et al. (2013) apresentam a ferramenta denominada STRadio, um aplicativo móvel para a plataforma Android que realiza análises de variações de interpretação em imagens radiográficas de tórax para casos de pneumonia infantil. Além disso, o sistema serve de auxílio ao treinamento de profissionais inexperientes na leitura desse tipo de radiografia. O sistema utiliza um banco de imagens com cerca de 10.000 imagens radiográficas de tórax de crianças menores de 5 anos que possuem diagnóstico clínico sugestivo de pneumonia. Assim que o usuário realiza a autenticação no sistema, são carregadas e

exibidas a ele 20 imagens de raios-X de tórax oriundas do banco de dados de imagens. Para cada imagem o usuário realiza seu diagnostico através de *checkboxes*, onde ele expressará suas interpretações sobre a imagem. Ao final da análise de todas as imagens o sistema informa os erros, acertos, tempo de conclusão, tempo de interpretação de cada imagem analisada, e os diagnósticos corretos.

A ferramenta apresenta uma proposta interessante para que profissionais que trabalham com a leitura de radiografias de tórax e diagnóstico de pneumonia infantil possam aprimorar suas habilidades analíticas e discutir sobre as divergências de opiniões que possam ocorrer. Um ponto a criticar sobre o trabalho de Soares et al. (2013) é a falta de evidências e detalhes sobre a ferramenta STRadio. Não fica claro como o usuário realiza o diagnóstico para cada imagem de radiografia apresentada, apenas é citado o uso de *checkboxes* para tal, imagens da interface ou outros detalhes acerca do processo não são apresentados. O trabalho também apresenta limitações por abordar apenas casos de pneumonia infantil, não estendendo para outros tipos de pneumonia como a pneumonia contraída por adultos que é muito comum e possui altos índices de hospitalizações e mortalidade (AULER et al., 2010). Além disso, os autores não mostram relatos de resultados obtidos, validações ou qualquer outro indício de testes da ferramenta com usuários, o que acaba por prejudicar o trabalho visto que não fora feito uma validação da aplicabilidade da ferramenta.

3.2.2 Sistema Imunológico Artificial

Er, Yumusak e Temurtas (2012) apresentam um *Sistema Imunológico Artificial* de apoio a decisão no diagnóstico de várias doenças pulmonares, dentre elas a pneumonia. Conforme explicam os autores, um *Sistema Imunológico Artificial* é fundamentado sobre os princípios de funcionamento do sistema imunológico do corpo humano. Para diagnosticar as doenças pulmonares o sistema utilizou como base um algoritmo com 5 etapas bem definidas: (1) Criar a população de anticorpos e determinar o limite de supressão; (2) Gerar clones (novo anticorpo/antígeno) para cara anticorpo; (3) Calcular a semelhança entre células anticorpos, eliminar os anticorpos em que a semelhança for menor que o limite de supressão definido e então determinar o número de anticorpos após a supressão; (4) Caso não seja assegurado que a população de memória é constante, retornar ao passo (2); (5) Classificar os valores resultantes.

A fim de testar esse sistema foram utilizados relatórios de sínteses analíticas de pacientes oriundos do banco de dados do *Hospital Diyarbakir de Doenças Pulmonares.* O estudo incluiu dados a respeito de 357 pacientes sofrendo de uma variedade de doenças respiratórias, e 38 pacientes considerados saudáveis. Esses pacientes foram qualificados como portadores de Tuberculose, Doença Pulmonar Obstrutiva Crônica, Pneumonia, Asma, Câncer de Pulmão ou como Não Doentes. Os resultados obtidos pelo sistema fo-

ram comparados com os resultados de outros métodos utilizados em estudos anteriores dos autores e que foram tomados como referencial teórico da pesquisa. Para cada doença foi realizada uma comparação dos resultados obtidos levando em consideração a precisão na classificação das doenças. Em relação aos demais métodos comparados o sistema proposto se saiu bem, pois no comparativo individual de cada doença na maioria das vezes obteve o maior índice (inclusive na média final), ou pelo menos ficou em igualdade. A única exceção onde acabou não obtendo o melhor resultado foi na classificação *Normal* (classificação de pacientes saudáveis).

O estudo apresentado por Er, Yumusak e Temurtas (2012) conta com um bom número de resultados e comparativos que auxiliam o leitor no entendimento da pesquisa e exaltam a eficácia do método proposto. O sistema desenvolvido alcançou um desempenho considerável, obtendo índices de mais de 90% de acurácia no diagnóstico de uma série de doenças respiratória o que levou os autores a considerar o uso do sistema no cenário real de classificação médica. Apesar de ser possivelmente viável e útil a aplicação de tal sistema em ambientes ambulatoriais, de forma a contribuir na tomada de decisão de diagnóstico para uma série de doenças respiratórias, o mesmo possui limitações quanto a quantidade de funcionalidades que oferece aos especialistas. Um sistema de apoio a decisão mais completo que oferece além do auxilio ao diagnóstico, possibilidades de gerenciamento de pacientes ou até sugestões de tratamento, auxiliaria consideravelmente os especialistas em termos de praticidade e agilidade nos processos. Além disso, um sistema com essas características conseguiria apoiar o trabalho de médicos novatos que ainda não estão adaptados com os processos de diagnóstico e tratamento das doenças.

3.2.3 Aplicativo de Classificação através de Sensores Móveis

Li (2015) apresenta um método para detectar a frequência respiratória de pacientes através do uso de sensores de um *smartphone* Android, e dessa forma diagnosticar casos de pneumonia. O sistema proposto é constituído de três partes: (1) O aplicativo Android que lê os dados oriundos dos sensores; (2) A parte responsável pelo processamento dos dados obtidos; e (3) A parte visual de exibição dos dados ao usuário. O primeiro procedimento realizado pelo sistema é a coleta dos sinais vitais dos pacientes. Os dados são coletados através do uso de sensores de acelerômetro e giroscópio de um *smartphone* colocado em frente ao peito ou abdômen do paciente. Esses dados são então encaminhados a parte lógica do sistema, responsável por remover ruídos dos sinais e calcular a frequência respiratória. Por fim, os dados processados são enviados a parte visual do sistema e exibidos ao usuário. Caso os dados de frequência respiratória de um paciente sejam considerados elevados de acordo com a definição da Organização Mundial de Saúde, o paciente é diagnosticado como portador de pneumonia.

O sistema proposto por Li (2015) foi testado com 6 pessoas de idade variando

entre 22 e 52 anos. Fora utilizado o *smartphone HTC one*, posicionado sobre o peito ou abdômen dos voluntários em posições pré-definidas. A autora comparou os resultados obtidos quando posicionado o *smartphone* no peito e no abdômen de pacientes, e também em quais eixos (x, y ou z) são obtidos melhores resultados. Para os dados mensurados a partir do acelerômetro, os canais mais confiáveis foram y e z quando o *smartphone* estava posicionado sobre o peito. Já para os dados do giroscópio não foi possível chegar a conclusão semelhante, pois os sinais foram afetados por vários fatores como a respiração e os batimentos cardíacos. Como conclusão do trabalho a autora expressa que os resultados obtidos foram satisfatórios onde descreve que o método proposto consegue determinar a frequência respiratória dentro de um minuto e com uma taxa de erros de apenas ± 2 respirações por minuto.

Apesar dos resultados satisfatórios a solução proposta apresenta limitações. Para que os sensores funcionem corretamente é necessário que sejam posicionados totalmente sobre a superfície do corpo humano, visto que o menor deslocamento pode afetar a aquisição dos dados e assim o diagnóstico. Esse fator pode ser considerado um problema se aplicado a um cenário real. Para diagnósticos de crianças, por exemplo, que podem não se comportar da maneira adequada quanto a movimentação do corpo, o método proposto não conseguiria realizar os diagnósticos adequadamente. Além disso, é citado durante o trabalho que os dados mensurados das atividades respiratórias são enviados à nuvem para que seja realizado certo processamento. No entanto isso é brevemente informado e não é detalhado quais mecanismos o sistema utiliza para realizar essas comunicações.

Outro ponto importante que o trabalho não considera é a preocupação com a segurança no envio desses dados a nuvem, característica importante a se considerar em sistemas com caráter médico. Desconsiderando a preocupação com a segurança o sistema fica sujeito a ataques de pessoas mal intencionadas que podem vir a interceptar o envio desses dados, lendo e até lhes alterando. Dessa maneira os diagnósticos do sistema seriam alterados o que colocaria a saúde dos pacientes em risco devido aos diagnósticos equivocados. Outro ponto de crítica do trabalho não elencada pela autora como uma limitação é a dependência de uma estrutura em rede, que como visto é necessária para que o processamento que ocorre em nuvem seja realizado.

3.2.4 PNEUMON-IA

Em Verdaguer et al. (1992) fora realizado a validação de um sistema especialista baseado em conhecimento denominado PNEUMON-IA. Esse sistema tem como objetivo determinar a etiologia da Pneumonia Adquirida na Comunidade (PAC) a partir de dados clínicos, laboratoriais e radiológicos obtidos nos estágios iniciais da pneumonia. Segundo os autores, o diagnóstico etiológico da pneumonia implica em grandes incertezas o que torna difícil estabelecer padrões. Partindo desse pressuposto o sistema utiliza um motor

de inferências chamado *Milord* que usa *lógica fuzzy* para expressar incertezas. Além disso, o PNEUMON-IA opera através de um conjunto de regras agrupadas em módulos e estratégias que por sua vez são controladas por meta-regras em uma arquitetura multinível. Isso tudo é utilizado em conjunto com o *Milord* para que o PNEUMON-IA faça inferências e realize os diagnósticos.

Os autores descrevem o PNEUMON-IA como sendo um sistema *Desktop* que possui versões para os sistemas operacionais *VMS* e *UNIX*. O PNEUMON-IA considera 22 possíveis agentes etiológicos para PAC. Para cada caso o sistema exibe como saída a etiologia mais provável juntamente com o seu grau de possibilidade. O status do PNEUMON-IA apresentado em Verdaguer et al. (1992) possui uma base de conhecimento que compreende 487 proposições e variáveis, 659 regras, 92 meta-regras e 25 módulos. A validação do sistema fora feita com registros médicos de 76 pacientes com diagnóstico de pneumonia comprovada. Os diagnósticos etiológicos fornecidos pelo PNEUMON-IA foram então comparados com os de 5 especialistas a fim de verificar o desempenho do sistema. Os autores relatam que nas comparações de resultados obtidos pelo PNEUMON-IA ele se manteve na média com os resultados obtidos pelos outros 5 especialistas, se aproximando mais do especialista que obteve o melhor desempenho entre os demais.

Por se tratar de um sistema complexo que considera incertezas para diagnosticar a PAC, o PNEUMON-IA se destaca no comparativo com outros sistemas de propósito semelhante que não consideram questões de incerteza, como os apresentados por (WELFER; SILVA; KAZIENKO, 2014) e (FRIEDMAN et al., 1999). No entanto ele é um sistema antigo e já não acompanha as mudanças constantes na área da tecnologia e na saúde. Desde a publicação do trabalho de Verdaguer et al. (1992) tecnologias mais práticas surgiram, a computação móvel é uma delas. Com ela é possível proporcionar mais dinamismo e praticidade a ambientes hospitalares, além de oferecer mais agilidade ao contrário de sistemas *Desktop* como o PNEUMON-IA.

3.3 Suporte a Decisão de Tratamento da Pneumonia

Nessa seção serão apresentados os trabalhos que tem como objetivo ou de alguma maneira fornecem suporte a decisão no tratamento da pneumonia.

3.3.1 Aplicativo que automatiza um Protocolo Médico da PAC

Welfer, Silva e Kazienko (2014) apresentam um aplicativo móvel desenvolvido para o sistema operacional móvel iOS que tem como objetivo prover auxílio ao diagnóstico e tratamento da Pneumonia Adquirida na Comunidade (PAC). Nesse trabalho os autores automatizaram o protocolo de tratamento da PAC proposto por Auler et al. (2010), utilizado no auxílio a profissionais da saúde para diagnosticar e tratar a doença no Hospi-

tal Nossa Senhora da Conceição (HNSC). Nesse protocolo médico constam basicamente: (1) As informações referentes a classificação de severidade da pneumonia adquirida pelo paciente; (2) A classificação do grupo de risco em que o paciente se encaixa; (3) O tratamento da pneumonia de acordo com a etiologia da doença; e por fim (4) O algoritmo de tratamento da pneumonia. A fim de validar o uso do aplicativo, os autores utilizaram questionários que foram submetidos a médicos do HNSC. A partir das respostas obtidas, os autores descrevem que segundo os médicos o aplicativo trouxe benefícios em relação a velocidade e praticidade no processo de gerenciamento dos pacientes com PAC.

Apesar do aplicativo apresentado por Welfer, Silva e Kazienko (2014) se mostrar útil no cenário médico real do HNSC, o mesmo possui limitações técnicas por tratar-se de um sistema puramente *standlone*. Isso significa que o aplicativo não utiliza uma estrutura em rede com troca de dados, limitando-se apenas as informações locais presentes nele. A exploração de arquiteturas distribuídas aplicadas a área médica possui um grande potencial visto as possibilidades que o compartilhamento de dados entre instancias de uma aplicação pode prover. Como por exemplo, fornecer uma quantidade maior de informações relevantes ao diagnóstico de pacientes – visto que vários sistemas trabalhariam em conjunto, gerando mais informações –, o que também contribuiria para a criação de históricos de doenças em pacientes.

3.3.2 Sistema de Classificação através de Processamento de Texto

Friedman et al. (1999) apresentam uma aplicação que automatiza diretrizes de classificação de severidade da Pneumonia Adquirida na Comunidade (PAC). Essas diretrizes classificam pacientes em 5 classes de risco a partir de um conjunto de variáveis extraídos de dados médicos. A aplicação proposta realiza consultas baseadas nas saídas geradas por um processador de linguagem natural denominado *MedLEE*. Esse processador codifica informações clínicas presentes em texto, assim fornecendo dados a respeito das variáveis necessárias para determinar o risco da PAC em pacientes. Dentre esses dados estão comorbidades, sinais vitais, sintomas presentes em resumos de alta, e informações de raio-x de peito.

Segundo os autores, os resultados obtidos foram positivos, baseando-se no comparativo com um padrão de referência obtido manualmente por um especialista. A aplicação demonstrou uma acurácia de 93%, sensibilidade de 92%, e especificidade de 93% para o processamento de resumos de alta. A respeito das informações de raio-x de peito, a aplicação demonstrou uma acurácia, sensibilidade e especificidade de 96%, 87%, e 98% respectivamente. A acurácia dos valores de sinais vitais foi de 85%, e para a determinação das classes de risco fora de 80%.

Apesar dos altos índices de acurácia, sensibilidade e especificidade para vários dos procedimentos realizados pela aplicação, no que diz respeito a classificação da classe de

risco de pacientes a aplicação obteve uma taxa de 80% de acertos. Apesar do índice remanescente de falha ser pequeno sua consequência pode ser grave caso a classificação erroneamente da classe de risco impactar na diferença de tratamento do paciente, o que colocaria a saúde do paciente em risco. Além disso o trabalho de Friedman et al. (1999) apresenta limitações quanto as suas funcionalidades, que deixam a desejar se aplicadas a um cenário real de utilização. A solução proposta apesar de interessante por resolver o problema da classificação de risco de pacientes – que é muito recorrente em hospitais – é de certa forma incompleta, pois não fornece outras ferramentas de auxílio aos especialistas como na questão de gerenciamento dos pacientes ou no auxílio ao seu tratamento, diferente do que é visto em outros trabalhos relatados no levantamento realizado nesse trabalho.

3.4 Fechamento do Capítulo

De forma geral os trabalhos relatados focaram-se de alguma maneira no diagnóstico ou tratamento da PAC/Pneumonia. Em relação as arquiteturas utilizadas por esses trabalhos, 3 utilizam arquiteturas móveis Welfer, Silva e Kazienko (2014), Soares et al. (2013) e Li (2015), já os demais trabalhos não mencionam. Quanto a questões referentes a compartilhamento de dados apenas o trabalho de Li (2015) utiliza-os de alguma forma, pois envia dados clínicos à nuvem a fim de realizar processamento. Apesar de apenas Li (2015) utilizar recursos de rede seu trabalho não leva em consideração questões sobre a segurança das informações.

Nesse capitulo foi relatado a busca por trabalhos relacionados ao tema dessa pesquisa. Entretanto, não fora encontrado trabalhos tão semelhantes ao que estamos propondo, ou seja, estudos que relatem o uso de arquiteturas móveis para fornecer suporte a decisão no tratamento da PAC, utilizando compartilhamento de dados e preocupando-se com a segurança. O mais próximo encontrado fora o trabalho de Welfer, Silva e Kazienko (2014), pois justamente utiliza de uma arquitetura móvel que auxilia profissionais da saúde no tratamento de pacientes com PAC. Porém, mesmo o trabalho de Welfer, Silva e Kazienko (2014) não utiliza abordagens distribuídas sendo um sistema totalmente *standlone* sem compartilhamento de recursos.

4 Desenvolvimento da Ferramenta

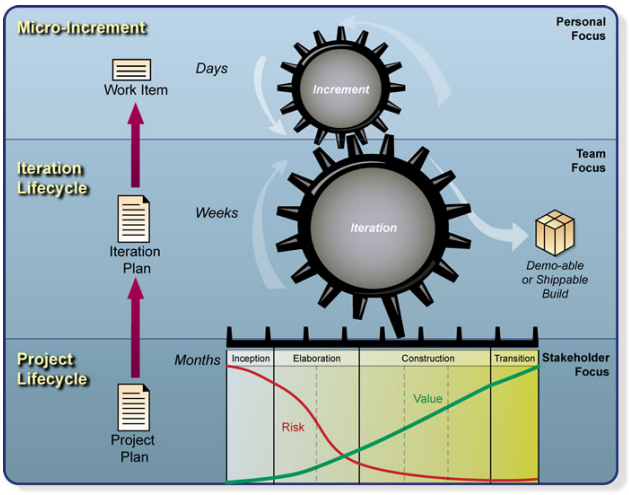
Neste capítulo são abordadas as etapas de desenvolvimento da ferramenta apresentada nesse trabalho, sendo elas: Análise (seção 4.2), Projeto (seção 4.3), Desenvolvimento(seção 4.4) e Testes (seção 4.5).

4.1 Processo de Software e Ambiente de Desenvolvimento

O Processo Unificado Aberto ou OpenUP é um processo que teve sua origem a partir do Processo Unificado Rational ou como é mais conhecido, o RUP. Apesar de manter muita das suas características e boas práticas, ele se difere do RUP por ter menos formalidade, ser mais ágil por adotar práticas de auto-organização do time de projeto, apresentar uma quantidade menor de produtos de trabalho, papéis e tarefas (Eclipse Foundation, 2015) (MEIRA, 2015) (SANTOS, 2015).

O OpenUP é um processo de desenvolvimento leve que aplica uma abordagem iterativa e incremental dentro de um ciclo de vida estruturado. Por se tratar de um processo de baixa cerimônia ele não está associado a nenhuma ferramenta específica e pode ser estendido para atender a uma ampla variedade de tipos de projetos (Eclipse Foundation, 2015). Ele é estruturado em 3 camadas distintas sendo elas o *Ciclo de Vida do Projeto*, *Ciclo de Vida de Iteração* e *Micro Incremento*, como é possível visualizar na Figura 3. Esse processo serviu como um guia no desenvolvimento da solução apresente nessa pesquisa, e com ele fora possível construir a solução de maneira iterativa e incremental.

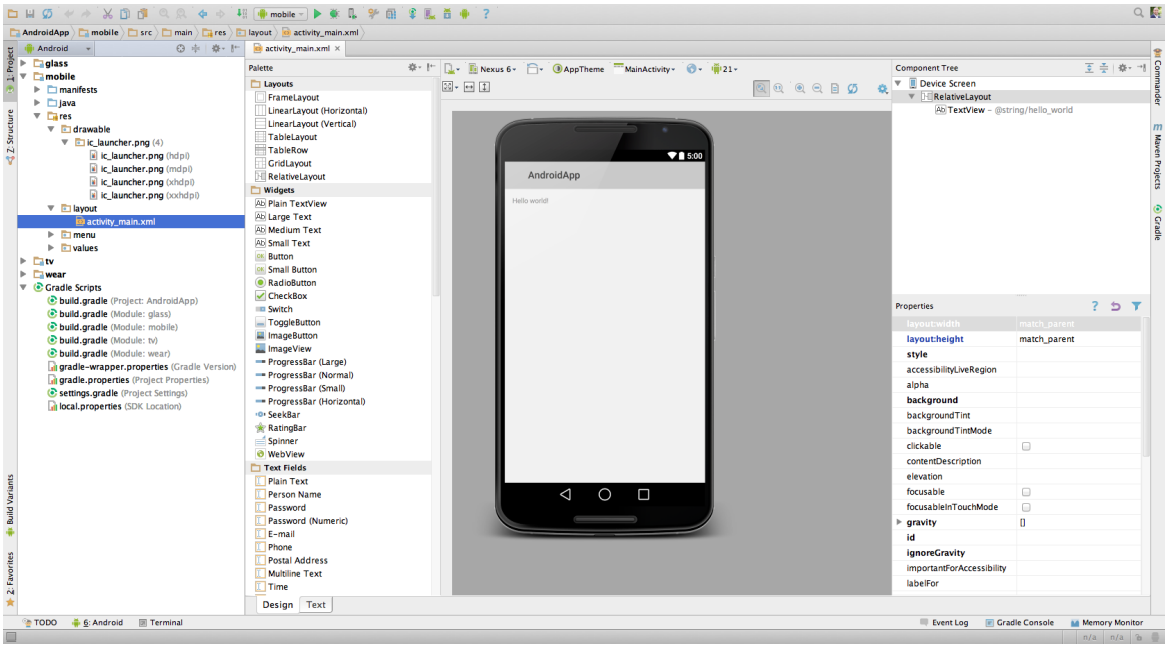
Figura 3 – Camadas do OpenUP.



Fonte: Eclipse Foundation (2015).

Além do processo escolhido, outro ponto importante a destacar que influenciou muito o processo de desenvolvimento da ferramenta descrito nessa seção, fora a utilização da IDE Android Studio, ambiente de desenvolvimento padrão de aplicativos para a plataforma Android adotada pela Google. O Android Studio é uma ferramenta poderosa e otimizada no desenvolvimento de aplicativos pro Android, pois já da suporte a versionamento de código, injeção de dependências, *framework* de testes JUnit, emulador entre outras ferramentas de forma nativa, o que facilita muito o desenvolvimento das aplicações. Na Figura 4 é exibido a título de demonstração a interface do Android Studio.

Figura 4 – Interface gráfica do Android Studio.



Fonte: Google (2015).

4.2 Análise

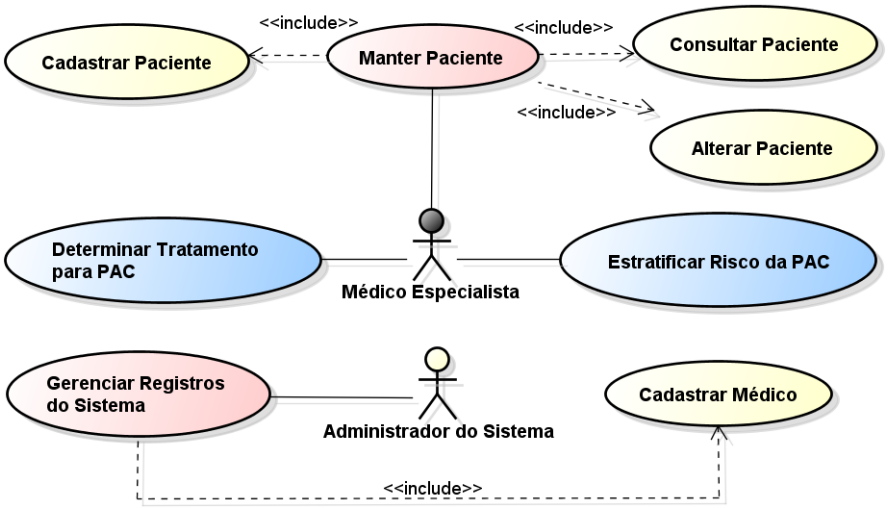
Nesta etapa inicial de construção da ferramenta é enfatizado as atividades de análise dos requisitos, onde visa-se compreender o problema a ser resolvido sem que haja preocupação neste ponto do desenvolvimento com a arquitetura ou implementação do sistema. Para essa etapa foram utilizados Casos de Uso (subseção 4.2.1) e Prototipação de telas do sistema (subseção 4.2.2).

4.2.1 Casos de Uso

O *Caso de Uso* é uma ferramenta importante na análise de requisitos de um projeto pois viabiliza uma visão das funcionalidades na perspectiva dos atores (usuários), defi-

nindo quais deles são responsáveis ou vinculados a determinadas funcionalidades (SOMMERVILLE, 2011). A Figura 5 ilustra um *Caso de Uso de Alto Nível* através de um *Diagrama de Casos de Uso* da *Unified Modeling Language* (UML). Casos de uso de alto nível são muito úteis para o entendimento da solução pois com eles é possível ter uma visão geral das funcionalidades existentes e dos atores vinculados a elas (WAZLAWICK, 2004).

Figura 5 – Diagrama de Casos de Uso de Alto Nível.



Nesse diagrama estão modelados basicamente dois tipos de casos de uso, os *CRUDs* e os *PENs*. As elipses com tom de cor rosa (*Manter Paciente* e *Gerenciar Registros do Sistema*) são os casos de uso considerados CRUD (acrônimo de Create, Read, Update e Delete, que representam as quatro operações básicas mais realizadas por uma aplicação sobre alguma informação). Para melhor entendimento de cada funcionalidade agregada por esses casos de uso CRUD foi utilizado as elipses de cor amarela ligadas a eles através de *includes* (utilizado na UML para especificar que um caso de uso inclui o comportamento de outro caso de uso).

Os casos de uso das elipses de cor azul (*Determinar Tratamento para PAC* e *Estratificar Risco da PAC*) são os chamados PEN (Processo Elementar de Negócio, é um termo utilizado na engenharia de processos de negócio). Um PEN é definido como uma atividade que é significativa para o(s) usuário(s) e que adiciona valor mensurável ou observável ao negócio, o deixando sempre em um estado consistente. Em outras palavras os PENs são as funcionalidades mais complexas que trazem valor agregado à aplicação, diferentemente dos CRUDs que são mais simples e sistemáticos.

Apesar do Diagrama de Casos de Uso da UML ser muito interessante por possibilitar uma visualização dos casos de uso existentes em um panorama geral de forma simples e rápida, apenas essa ferramenta por si só não é suficiente para expressar de forma

satisfatória os requisitos de um sistema. Justamente por essa necessidade de descrever melhor os requisitos que existem as *Narrativas de Caso de Uso*. Nelas os requisitos do sistema são especificados em detalhes através de fluxos de ações onde são feitas referências a cenários do sistema, atores e suas ações.

A seguir, nas Tabelas 5, 6, 7, 8 e 9 são apresentadas as Narrativas de Casos de Uso ilustrados na Figura 5. As Tabelas 5, 6 e 9 apresentam os três casos de uso CRUD presentes no sistema. Apesar de serem apresentadas em elipses separadas na Figura 5, cada uma das quatro operações de um CRUD compõem um único caso de uso e por essa razão são descritos em uma só narrativa onde essas operações se fazem presente. O motivo pela divisão das operações no diagrama fora apenas a fim de facilitar o entendimento das funcionalidades. Por seguinte, nas Tabelas 7 e 8 são presentados os casos de uso PEN do sistema.

Tabela 5 – Narrativa do Caso de Uso *Manter Paciente*.

Caso de Uso:	Manter Paciente.
Ator Principal:	Médico Especialista e Administrador do Sistema.
Cadastrar Paciente	
Pré-Condições:	
1. Médico já ter acessado a tela referente ao cadastro de Paciente(s).	
Fluxo Principal:	
1. Médico informa os dados obrigatórios (Nome, Idade, Sexo e RG) e interage com o item de confirmação de cadastro.	
2. Aplicação Cliente verifica se os campos obrigatórios foram preenchidos e se esses foram preenchidos corretamente, e então notifica o Médico.	
3. Aplicação Cliente salva o cadastro do novo Paciente inicialmente apenas de forma local no dispositivo móvel.	
Fluxo Alternativo:	
2a. Aplicação Cliente verifica que os campos obrigatórios não foram preenchidos, ou que esses não foram preenchidos de forma adequada.	
2a.1) Aplicação Cliente notifica o erro ao Médico e solicita sua correção.	
3a. Aplicação Cliente após realizar a persistência dos dados do novo Paciente localmente verifica se existe conexão com o Servidor.	
3a.1) Aplicação Cliente exibe janela de diálogo ao Médico a fim de indagá-lo se o mesmo deseja enviar os dados do novo Paciente ao Servidor.	
3a.2) Médico confirma o envio dos dados no novo Paciente ao Servidor.	

3a.3) Aplicação Cliente envia uma requisição a Aplicação Servidora a fim de salvar os dados do novo Paciente no Banco de Dados remoto.
3a.4) Aplicação Servidora verifica se o registro de Paciente recebido já não existe no Banco de Dados, e caso verifique que se trata de um novo registro então o persiste no Banco de Dados e envia resposta com o status da operação a Aplicação Cliente.
3a.4a) Aplicação Servidora verifica que o registro enviado pela Aplicação Cliente trata-se de um registro já existente ou que inflige regras de negócio da base de dados.
3a.4a.1) Aplicação Servidora envia resposta da requisição a Aplicação Cliente notificando-a que o registro em questão trata-se de um registro inválido e não é possível integrá-lo à base de dados.
3a.5) Aplicação Cliente recebe a confirmação da persistência no Banco de Dados remoto e notifica o Médico.
Consultar Paciente
Pré-Condições:
1. Médico já ter acessado a tela referente a consulta de Paciente(s).
Fluxo Principal:
1. Médico informa algum dado a respeito do Paciente que deseja localizar entre os registros existentes (Nome, Idade, Sexo ou RG).
2. Aplicação Cliente pesquisa inicialmente apenas nos registros locais presentes no dispositivo móvel e exibe resultados ao Médico.
3. Médico localiza o Paciente desejado.
Fluxo Alternativo:
3a. Médico não consegue localizar o Paciente desejado e então interage com o item de busca remota no Servidor.
3a.1) Aplicação Cliente exibe janela de diálogo ao Médico a fim de indagá-lo se o mesmo deseja pesquisar por Paciente (1) informando dados como parâmetro de busca ou (2) se deseja consultar os registros de todos os Pacientes presentes no Banco de Dados remoto.
3a.2) Médico opta por uma das opções exibidas na janela de diálogo e assim a Aplicação Cliente envia a requisição a Aplicação Servidora.
3a.3) Aplicação Servidora realiza consulta no Banco de Dados por Pacientes a partir do parâmetro de busca informado pelo Médico e retorna os registros encontrados a Aplicação Cliente.
3a.4) Aplicação Cliente recebe a resposta do Servidor com os registros encontrados e então os armazena localmente no dispositivo móvel.
3a.5) Fluxo de ações retorna ao *passo 1* do *Fluxo Principal* de *Consultar Paciente*.
Alterar Paciente
Pré-Condições:

1. Médico já ter acessado a tela referente a consulta de Paciente(s) e ter selecionado um registro em específico que deseja alterar.
Fluxo Principal:
1. São executados os passos descritos em *Cadastar Paciente*, com a diferença que ao invés de ser inserido um novo registro o que será realizado é a alteração dos dados de um registro existente.

Tabela 6 – Narrativa do Caso de Uso *Cadastrar Médico*.

Caso de Uso:	Cadastrar Médico.
Ator Principal:	Administrador do Sistema.
Pré-Condições:	
1. Administrador estar devidamente autenticado no Servidor Web e com acesso ao Banco de Dados remoto.	
Fluxo Principal:	
1. Administrador insere diretamente no SGBD um novo registro de Médico informando os dados obrigatórios (Nome, Idade, Sexo, RG, CPF e Chave Criptográfica) e/ou opcionais (Área de Atuação e Especialidade).	
2. SGBD analisa os dados informados e caso estejam em conformidade salva o novo registro.	
Fluxo Alternativo:	
2a. SGBD verifica que algum campo informado pelo Administrador não fora especificado de forma correta.	
2a.1) SGBD informa o erro ao Administrador solicitando sua correção.	

Tabela 7 – Narrativa do Caso de Uso *Estratificar Risco da PAC*.

Caso de Uso:	Estratificar Risco da PAC.
Ator Principal:	Médico Especialista.
Pré-Condições:	
1. Existir algum registro de Paciente cadastrado no Banco de Dados ou armazenado localmente no dispositivo móvel.	
Fluxo Principal:	

1. Médico acessa a tela da Aplicação Cliente referente a Estratificação de Risco e interage com item de busca por registro de Paciente.
2. Médico da entrada na pesquisa por registros de Pacientes informando algum dado como parâmetro de busca (Nome, Idade, Sexo, ou RG).
3. Aplicação Cliente realiza busca a partir do parametro de pesquisa informado pelo Médico nos registros armazenados localmente no dispositivo móvel e exibe os resultados encontrados.
4. Médico seleciona o registro de Paciente desejado.
5. Aplicação Cliente exibe ao Médico a interface para a realização da Estratificação de Risco da doença.
6. Médico realiza a Estratificação marcando nos itens de checagem apresentados na interface e ao final do processo o finaliza interagindo com um botão de "próxima tela".
7. Aplicação Cliente salva localmente o registro da nova Estratificação gerada e exibe o resultado da Estratificação de Risco do Paciente selecionado assim como o restante de seus dados cadastrados.
Fluxo Alternativo:
4a. Médico não encontra o registro desejado.
4a.1) Médico opta por realizar o cadastro de um novo registro de Paciente (fluxo de ações executadas é descrito no caso de uso *Manter Paciente*).
4a.1a) Médico opta por buscar no Banco de Dados remoto pelo registro desejado.
4a.1a.1) Médico informa parâmetros de entrada da pesquisa por Paciente desejado (Nome, Idade, Sexo ou RG).
4a.1a.2) Aplicação Cliente envia requisição a Aplicação Servidora por registro de Paciente.
4a.1a.3) Aplicação Servidora consulta no Banco de Dados remoto pelo parametro de pesquisa recebido e retorna a resposta a Aplicação Cliente.
4a.1a.4) Aplicação Cliente exibe o resultado recebido e o Fluxo de ações executadas volta ao *passo 4* do *Fluxo Principal*.
7a. Médico depois de visualizar os dados da Estratificação gerada interage com o botão de *upload* do novo registro para a base de dados remota.
7a.1) Aplicação Cliente envia requisição de armazenamento do novo registro no Banco de Dados remoto a Aplicação Servidora.
7a.2) Aplicação Servidora verifica se o novo registro de Estratificação já não existe no Banco de Dados.
7a.3) Aplicação Servidora verifica que trata-se de um registro ainda não existente e então persiste as informações no Banco de Dados, enviando ao fim dessa operação a resposta com o status do procedimento a Aplicação Cliente.

7a.4) Aplicação Cliente recebe a resposta do Servidor e notifica o Médico.

Tabela 8 – Narrativa do Caso de Uso *Determinar Tratamento para a PAC*.

Caso de Uso:	Determinar Tratamento para a PAC.
Ator Principal:	Médico Especialista.
Pré-Condições:	
1. Existir algum registro de Estratificação de Risco cadastrado no Banco de Dados ou armazenado localmente no dispositivo móvel.	
Fluxo Principal:	
1. Médico após realizar uma Estratificação de Risco e visualizar seus dados, interage com um botão de "próxima tela".	
2. Aplicação Cliente então exibe uma série de telas com perguntas ao Médico (com opções "Sim ou Não") a fim de determinar qual o tratamento recomendado ao Paciente previamente selecionado quando o registro de Estratificação de Risco fora gerado.	
3. Médico responde as perguntas exibidas pela interface e ao final das perguntas a Aplicação Cliente exibe o resultado de Tratamento recomendado para a PAC, salvando o novo registro de Diagnóstico localmente.	
Fluxo Alternativo:	
3a. Médico após visualizar os dados do Tratamento recomendado interage com o botão de *upload* do novo registro de Diagnóstico gerado a base de dados remota.	
3a.1) Aplicação Cliente envia requisição de armazenamento do novo registro no Banco de Dados remoto a Aplicação Servidora.	
3a.2) Aplicação Servidora verifica se o novo registro de Diagnóstico já não existe no Banco de Dados.	
3a.3) Aplicação Servidora verifica que trata-se de um registro ainda não existente e então persiste as informações no Banco de Dados, enviando ao fim dessa operação a resposta com o status do procedimento a Aplicação Cliente.	
3a.4) Aplicação Cliente recebe a resposta do Servidor e notifica o Médico.	

Tabela 9 – Narrativa do Caso de Uso *Gerenciar Registros do Sistema*.

Caso de Uso:	Gerenciar Registros do Sistema.

Ator Principal:	Administrador do Sistema.
Cadastrar Registro de Paciente/Médico/Estratificação/Diagnóstico	
Pré-Condições:	
1. Administrador estar devidamente autenticado no Servidor Web e com acesso ao Banco de Dados remoto.	
Fluxo Principal:	
1. Administrador insere diretamente no SGBD um novo registro de Paciente/Médico/Estratificação/Diagnóstico informando seus dados obrigatórios e/ou opcionais.	
2. SGBD analisa os dados informados e caso estejam em conformidade salva o novo registro.	
Fluxo Alternativo:	
2a. SGBD verifica que algum campo informado pelo Administrador não fora especificado de forma correta.	
2a.1) SGBD informa o erro ao Administrador solicitando sua correção.	
Consultar Registro de Paciente/Médico/Estratificação/Diagnóstico	
Pré-Condições:	
1. Administrador estar devidamente autenticado no Servidor Web e com acesso ao Banco de Dados remoto.	
Fluxo Principal:	
1. Administrador informa diretamente no SGBD algum dado a respeito do registro de Paciente/Médico/Estratificação/Diagnóstico que deseja localizar entre os registros existentes.	
2. SGBD realiza a busca a partir dos parâmetros de busca especificados e retorna ao Adminsitrador os resultados encontrados.	
3. Administrador localiza o registro de Paciente/Médico/Estratificação/Diagnóstico desejado.	
Fluxo Alternativo:	
3a. Administrador não localiza o registro desejado.	
3a.1) Sequência de passos volta ao *passo 1* do *Fluxo Principal*.	
Alterar Registro de Paciente/Médico/Estratificação/Diagnóstico	
Pré-Condições:	
1. Administrador estar devidamente autenticado no Servidor Web e com acesso ao Banco de Dados remoto.	
2. Administrador já ter realizado uma busca por registros de Pacientes/Médicos/Estratificações/Diagnósticos e ter selecionado um registro em específico que deseja alterar.	

Fluxo Principal:
1. São executados os passos descritos em *Cadastar Registro de Paciente/Médico/Estratificação/Diagnóstico* com a diferença que ao invés de ser inserido um novo registro, o que será realizado é a alteração dos dados de um registro existente.
Excluir Registro de Paciente/Médico/Estratificação/Diagnóstico
Pré-Condições:
1. Administrador estar devidamente autenticado no Servidor Web e com acesso ao Banco de Dados remoto.
2. Administrador já ter realizado uma busca por registros de Pacientes/Médicos/Estratificações/Diagnósticos e ter selecionado um registro em específico que deseja excluir.
Fluxo Principal:
1. Administrador confirma a exclusão do registro selecionado.
2. SGBD verifica se existem dependências entre registros e caso estejam em conformidade realiza a exclusão do(s) registro(s).
Fluxo Alternativo:
1a. SGBD não consegue excluir o registro selecionado pois o mesmo possui dependência com outros registros.
1a.1) SGBD notifica o Administrador do erro.

4.2.2 Protótipos de Telas do Sistema

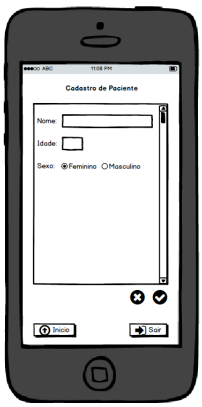
A utilização de protótipos de telas podem ajudar o Engenheiro de Software a compreender de forma mais concreta os requisitos de um sistema. A prototipagem ajuda a visualizar as funcionalidades, como estão distribuídas, como interagem entre si e principalmente que usuários estão relacionados a elas. Quando se desenvolve um protótipo é possível envolver os *stakeholders* (partes interessadas) no processo de desenvolvimento de maneira mais ativa. Com a uso dessa técnica é possível obter validações prévias com os usuários, o que ajuda a prevenir futuras deficiências no andamento do projeto.

Apesar desta pesquisa não dispor de usuários especialistas para realizar as validações dos protótipos, o uso da prototipagem se justifica pois auxilia o desenvolvimento das interfaces do sistema, possibilitando que essas sejam trabalhadas de forma continua com o projeto da arquitetura, por exemplo, e não apenas futuramente nas etapas de implementação do *software*. Protótipos de telas podem ser considerados artefatos de projeto de *software*, porém neste trabalho eles foram utilizados como artefatos de análise para que fosse possível compreender melhor os requisitos e como se comportariam, e por essa razão estão presentes nessa seção. Na Figura 6 podem ser visualizados os protótipos de

telas do aplicativo Android *SysPAC*.

Figura 6 – Protótipos das Telas do Aplicativo *SysPAC*.

(a) Tela Principal do Sistema.



(b) Tela de Cadastro de Pacientes.

(c) Tela de Realização de Diagnóstico.

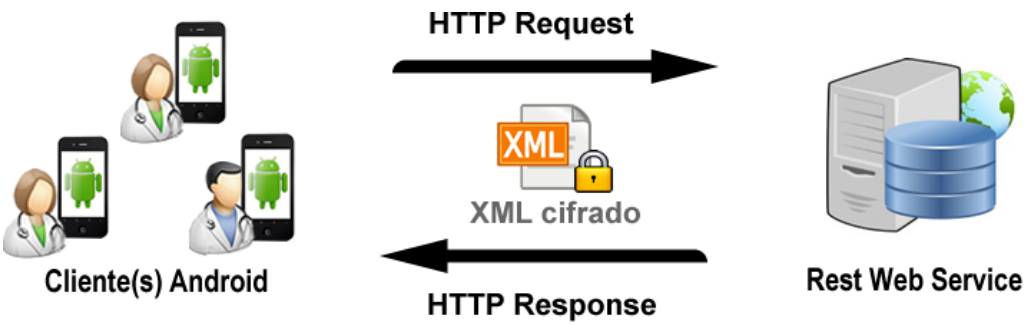
4.3 Projeto

Nesta etapa de desenvolvimento se enfatiza o processo de análise e planejamento da arquitetura da solução proposta. Sendo assim, dividimos os artefatos da arquitetura da solução proposta em diferentes níveis de detalhamento. Na Figura 7 é apresentado a organização dos componentes de *hardware* e *software* da abordagem distribuída , já na Figura 8 é apresentado uma exemplificação do relacionamento entre os componentes da arquitetura de *software* utilizada, por seguinte na Figura 9 é apresentado o modelo de domínio da aplicação, nas figuras 10, 11 e 12 são apresentados e discutidos os diagramas de classes da Aplicação Servidora separadas segundo os componentes da arquitetura adotada, e por fim na Figura 13 é apresentada a modelagem entidade-relacionamento do Banco de Dados do sistema.

4.3.1 Organização dos Componentes da Abordagem Distribuída

Como é possível observar na Figura 7 a abordagem distribuída apresentada nesse trabalho é dividida entre a parte da Aplicação Cliente – que corresponde as aplicações executadas em *smartphones* – e a parte da Aplicação Servidora – que consiste em um *Rest Web Service* que atende requisições dos aplicativos e gerencia um Banco de Dados – que irão se comunicar através de um meio sem fio utilizando o padrão IEEE 802.11, também conhecido como Wi-Fi (*Wireless Fidelity*) (IEEE, 2015).

Figura 7 – Organização dos Componentes da Abordagem Distribuída.



A fim de garantir a confidencialidade dos dados comunicados entre as Aplicações Cliente e a Servidora (*Rest Web Service*), optou-se por utilizar o recurso da criptografia, mais especificamente a *criptografia simétrica* para cifrar os dados trafegados pela rede. Deste modo, todas as informações de pacientes/estratificações/diagnósticos armazenados no formato XML nos *smartphones* são cifradas e enviadas via protocolo HTTP pela Wi-Fi e depois decifrados quando recebidos no servidor. Cabe ressaltar que ambas as operações de cifragem/decifragem das mensagem só são possíveis com o uso da chave criptográfica conhecida apenas pelo médico especialista que opera o Aplicativo, e o receptor (Servidor) que possui a chave do médico armazenada no Banco de Dados.

Conforme é visto na Figura 7, na parte cliente é executado o Aplicativo Móvel que automatiza os protocolos médicos da PAC descritos na subseção 2.1.1. O Aplicativo é responsável por realizar *Cadastros de Pacientes*, *Realizar Estratificações de Risco*, *Determinar Tratamentos para Pacientes*, entre outras funcionalidades abordadas na seção 4.2. Além disso, como o Aplicativo é parte de um sistema distribuído conforme é apresentado nesse trabalho, existe a possibilidade de se utilizar dos recursos de comunicação em rede para enviar dados gerados localmente a base de dados gerenciada pelo *Rest Web Service*, assim como requisitar dados a serem armazenados localmente no dispositivo móvel.

Já na parte referente ao servidor foram atribuídas as execuções dos principais processamentos da arquitetura distribuída com o intuito de delegar o mínimo possível de processamento ao Aplicativo, visto as limitações da plataforma. Essa decisão fora to-

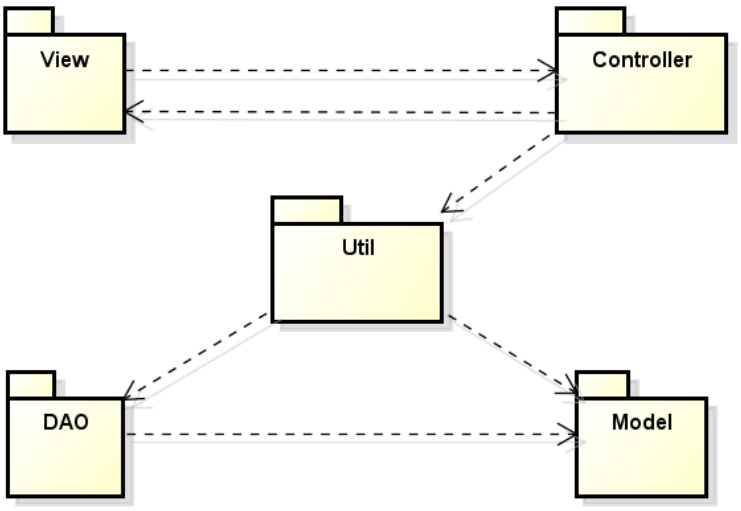
mada a fim de evitar o acréscimo de processamento nos dispositivo móveis que executam o Aplicativo, evitando assim o aumento do consumo energético e quantidade de armazenamento requerido pela ferramenta, pontos importantes a se considerar no desenvolvimento de *software* para arquiteturas embarcadas, conforme visto na seção 2.3.

Como mostra a Figura 7 o *Rest Web Service* presente no esquema tem como propósito atender aos serviços solicitados pelo Aplicativo. Um cenário em que o *Rest Web Service* fornece um recurso importante ao Aplicativo, acontece se caso o mesmo deseje consultar os dados de um paciente no banco de dados remoto. Para isso o médico então irá interagir com os menus apropriados no Aplicativo que por sua vez irá enviar uma requisição REST cifrada ao *Rest Web Service* solicitando um serviço. Na parte servidor esse *Rest Web Service* irá receber a requisição cifrada e utilizará da chave criptográfica do médico (armazenada junto ao seu registro no banco de dados) para decifrar a mensagem e assim atender ao serviço requisitado: buscar pelos dados do paciente requerido na base de dados e ao final do processo enviar uma resposta a aplicação com o que encontrou.

4.3.2 Arquitetura da Aplicação

A seguir são apresentados alguns conceitos de modelagem e projeto utilizados na construção da ferramenta aqui apresentada. No presente trabalho fora utilizado o padrão arquitetural *Model-View-Controller* (MVC) com a adição de dois componentes extras, como é possível observar na Figura 8, que exemplifica o relacionamento entre os componentes da arquitetura. Esse modelo arquitetural fora utilizado tanto para a criação da solução *mobile* quanto para a aplicação executada no servidor.

Figura 8 – Relação entre os Componentes do Modelo Arquitetural.



Em conjunto com o MVC fora utilizado nessa arquitetura mais dois componentes:

Util e a *DAO* (Refere-se a *Data Access Object*). Essa decisão fora tomada a fim de dividir ainda mais as responsabilidades dos componentes deixando a arquitetura menos dependente e mais flexível a refatorações.

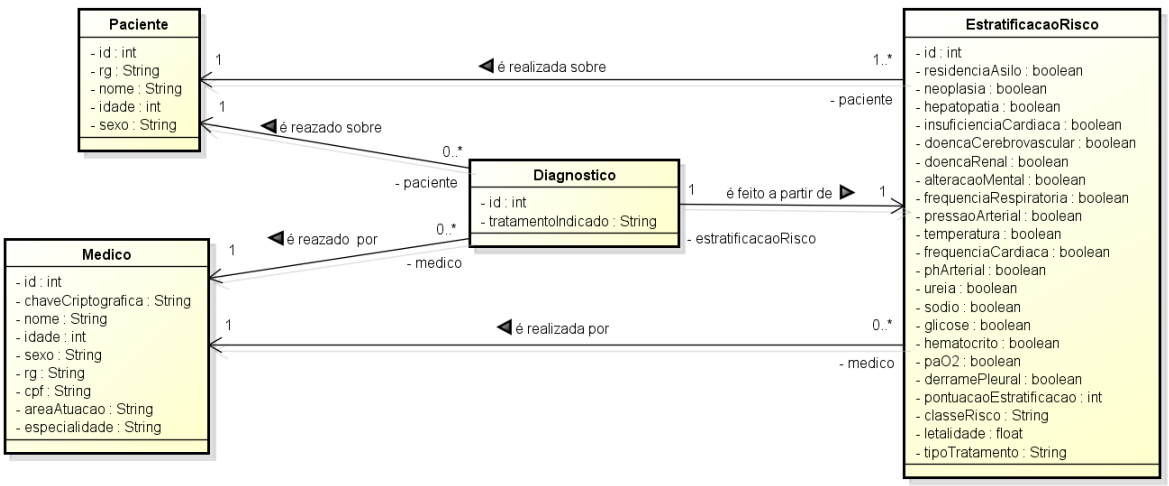
O componente *Util* tem como responsabilidade atender a serviços requisitados pela *Controller*. Optou-se por utilizá-lo pois com esse componente é possível separar os serviços da *Model* deixando-a apenas com as regras de negócio, diferente do que é feito no MVC clássico onde ela também se encarrega de conter as funcionalidades dos objetos.

Já o componente DAO se encarrega das operações envolvendo a persistência de dados e demais interações com o banco de dados. A mesma justificativa adotada para o uso da *Util* também é aplicada para o uso da DAO. Com a sua utilização se visou separar as operações que envolvem a interação com o banco de dados da *Model*, que tradicionalmente no MVC é o componente que lida com essas responsabilidades.

4.3.2.1 Modelo de Domínio da Aplicação

Um modelo de domínio tem como intuito representar de forma visual as informações presentes em um determinado domínio que serão gerenciadas por um sistema. Nesse trabalho o domínio é o conjunto de informações presentes no formulário médico de tratamento da PAC visto na subseção 2.1.1. Na Figura 9 é possível observar as classes que compõem esse domínio de conhecimento que o sistema gerencia e seus relacionamentos.

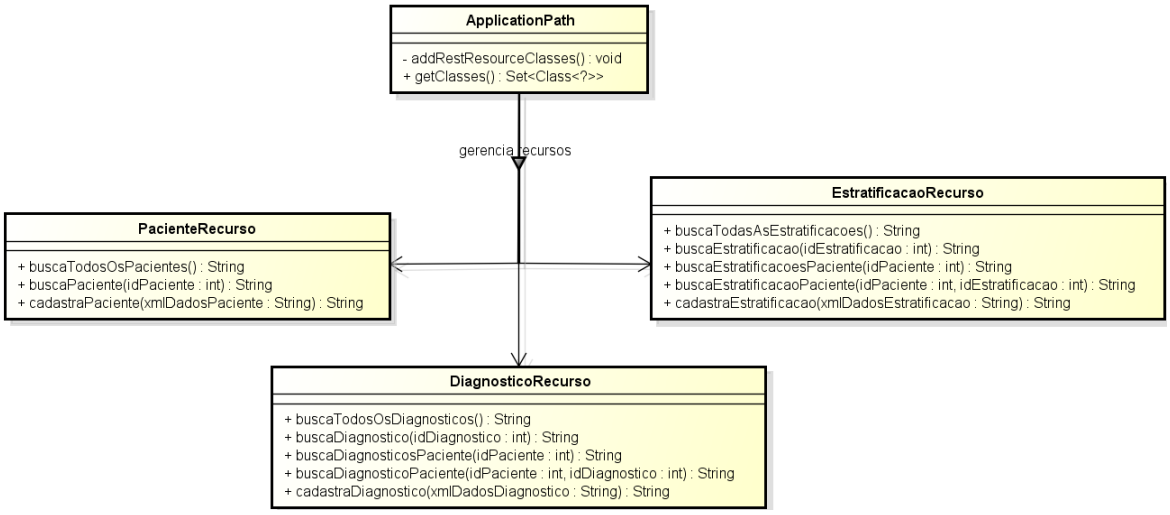
Figura 9 – Diagrama de Classes do Domínio da Aplicação.



4.3.2.2 Classes da Aplicação Servidora

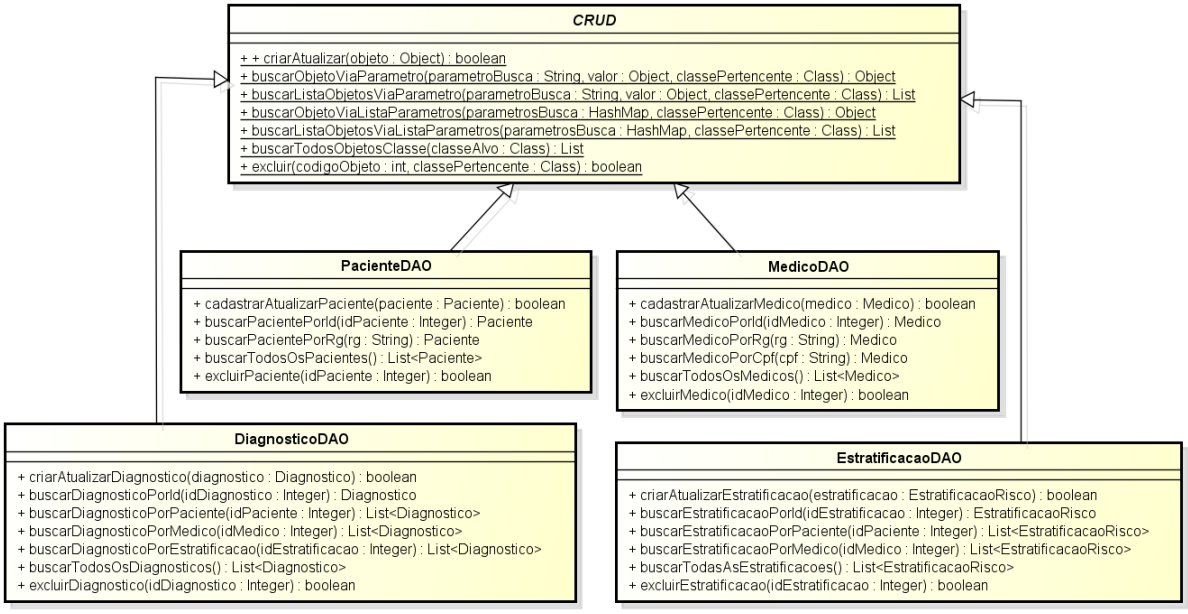
Nas Figuras 10, 11 e 12 a seguir são apresentadas as classes da aplicação servidora divididas pelos componentes da arquitetura MVC das quais fazem parte.

Figura 10 – Classes do Componente *Controller*.



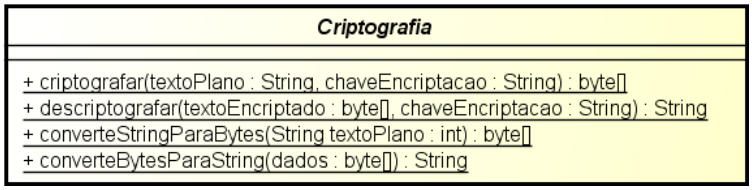
Na Figura 10 são apresentadas as classes do componente *Controller* da aplicação servidora. Como é possível observar existe uma classe chamada *ApplicationPath* que se relaciona com as demais três classes presentes no diagrama, sendo essas *PacienteRecurso*, *DiagnosticoRecurso* e *EstratificacaoRecurso*. Como já citado anteriormente, a tecnologia utilizada para a programação da aplicação servidora fora a implementação de um *Web Service Rest*, que por sua vez trabalha com a ideia de "oferecer recursos a serem consumidos"por outros sistemas. Sendo assim, cada uma das três classes que contêm a palavra *Recurso* no nome, oferecem uma série de funcionalidades ou recursos a aplicação Android que consulta esse *Web Service Rest*, sendo que a classe responsável por instanciá-las e "registrá-las"como recurso do *Web Service* é justamente a classe ApplicationPath.

Figura 11 – Classes do Componente *DAO*.



Na Figura 11 é apresentada a classe *CRUD* que faz parte do componente DAO da arquitetura da aplicação. Nessa classe basicamente estão presentes métodos genéricos criados com o propósito de dar suporte a todas as operações envolvendo a interação com o Banco de Dados, simplificando assim o uso de tais operações. Essa classe e seus métodos foram projetos da forma mais genérica possível para que não houvesse duplicação de procedimentos em outras classes do projeto e para que qualquer uma das classes de domínio pudesse utilizá-la.

Figura 12 – Classes do Componente *Util*.

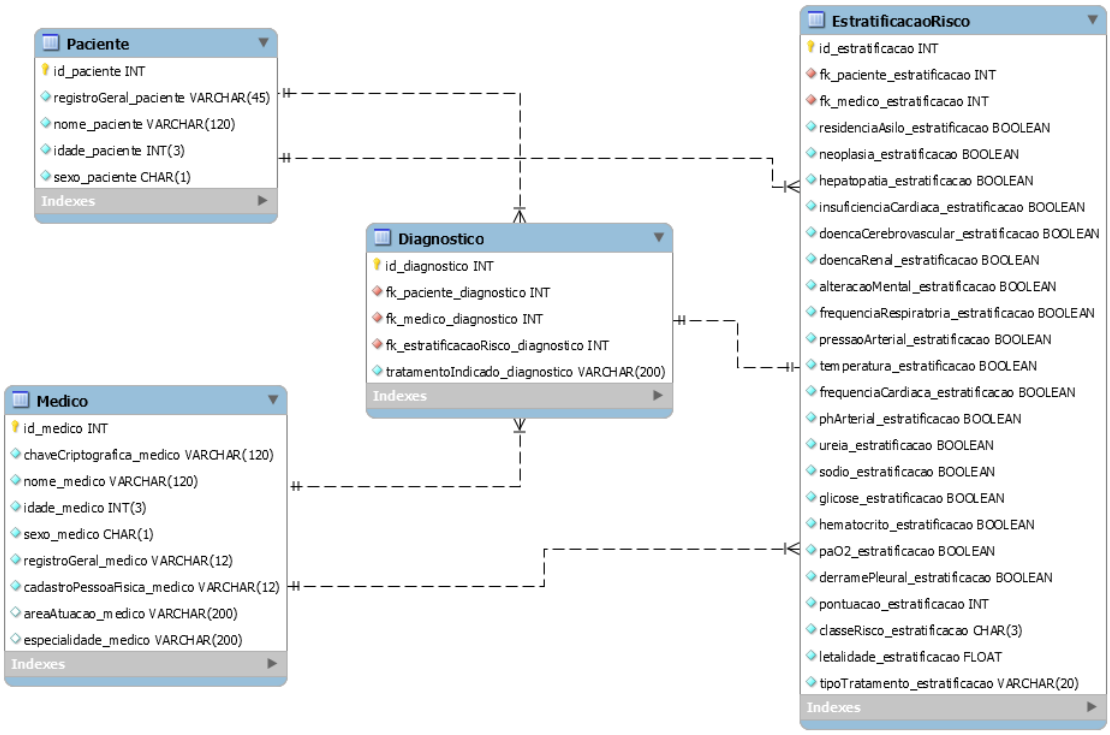


Por fim, na Figura 12 é exibida a classe *Criptografia* pertencente ao componente Util da arquitetura. Trata-se basicamente de uma classe utilitária para a cifragem e decifragem dos arquivos em forma XML que são enviados e recebidos do servidor.

4.3.3 Modelagem do Banco de Dados

Na Figura 13 é possível observar o Diagrama Entidade Relacionamento do banco de dados do sistema armazenado e manipulado pela aplicação servidora.

Figura 13 – Diagrama de Entidade-Relacionamento do Banco de Dados.



O banco de dados em questão é simples como é possível ver na figura, sendo composto por 4 tabelas onde cada uma irá armazenar tanto informações cadastradas pelo administrador do sistema como o gerenciamento dos médicos especialistas que utilizam a aplicação, quanto as informações geradas pela aplicação *mobile* utilizadas pelos médicos, como cadastros de pacientes, estratificações de risco e diagnósticos.

A utilização de um banco de dados ao invés de outra alternativa de armazenamento de dados é justificada nesse trabalho pelo fato de que no cenário pressuposto, a aplicação servidora seria hospedada em um hospital e os registros médicos gerados possivelmente são importantes aos médicos e ao hospital a fim de manter históricos de doenças, consultas já realizadas entre outras informações relevantes ao cenário médico. Além disso, com a utilização de um banco de dados tanto o gerenciamento, quanto a escalabilidade do número de informações, a rapidez nas consultas entre outros fatores é facilitada se comparado a outras alternativas de armazenamentos de informações como o armazenamento em arquivo, como por exemplo, em formato XML como fora feito na aplicação Android desenvolvida.

4.4 Desenvolvimento

Nesta fase é enfatizado o processo de implementação da solução proposta. Nas seções a seguir a implementação do sistema é apresentada de acordo com cada parte fundamental que compõem a ferramenta desenvolvida nesse trabalho.

4.4.1 Interface Gráfica

Como o sistema proposto nesse trabalho utiliza de uma abordagem distribuída onde existem como *Clientes* os Aplicativos Android que executam a aplicação médica, e como *servidor* um *Rest Web Service* que atende as requisições feitas pelos aplicativos, existe apenas interface gráfica nesses aplicativos visto que a parte referente ao servidor não necessita de tal meio de interação.

O Android utiliza a linguagem XML para configurar suas interfaces gráficas. Um exemplo de configuração de uma interface do Android é mostrado na Figura 14.

Figura 14 – XML de configuração de interface do Android.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context="unipampa.edu.br.syspac.controladora.TelaAlgoritmoDecisaoMain">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Tratamento Hospitalar ?"
        android:id="@+id/textView9"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="151dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Não"
        android:id="@+id/buttonMainNao"
        android:layout_centerVertical="true"
        android:layout_alignLeft="@+id/textView9"
        android:layout_alignStart="@+id/textView9" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sim"
        android:id="@+id/buttonMainSim"
        android:layout_alignTop="@+id/buttonMainNao"
        android:layout_alignRight="@+id/textView9"
        android:layout_alignEnd="@+id/textView9" />

</RelativeLayout>
```

Outra característica da plataforma de desenvolvimento do Android é que cada interface possui uma barra superior chamada de *ActionBar* onde podem ser adicionados botões de ação como pesquisa, configurações entre outros. Esse menu é também definido via XML conforme é visto na Figura 15, porém esse é configurado em um arquivo isolado da interface principal que irá acoplá-lo quando renderizada.

Figura 15 – XML de configuração da *ActionBar* da interface do Android.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/search"
        android:title="campo de pesquisa"
        android:showAsAction="collapseActionView|ifRoom"
        android:actionViewClass="android.support.v7.widget.SearchView" />
</menu>
```

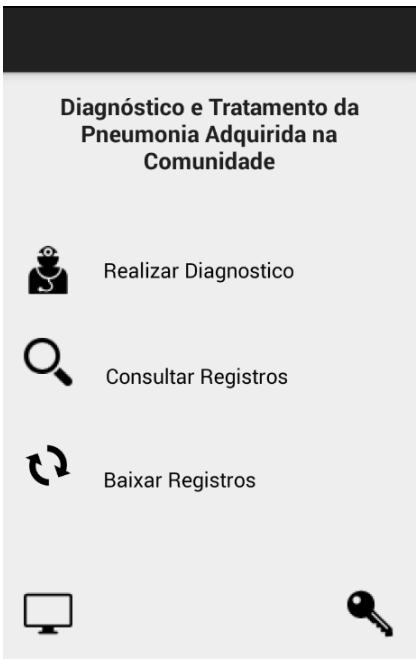
Como toda a interface do Android é configurada por um arquivo no formato XML, cada um desses arquivos é instanciado por uma classe *Activity*. As classes *Activity* funcionam como um componente *Controller*, onde possuem a funcionalidade de realizar o gerenciamento dos componentes de *View* do Android e demais componentes da arquitetura como as classes da *Model*. Na Figura 16 é mostrado um exemplo de implementação de uma *Activity*.

Figura 16 – Classe *Activity* responsável por inicializar a interface gráfica.

```java
public class TelaAlgoritmoDecisaoMain extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(layout.activity_tela_algoritmo_decisao_main);

        Intent recebeIntent= getIntent();
        final Bundle extras = recebeIntent.getExtras();

        Button botaoDecisaoNao = (Button) findViewById(R.id.buttonMainNao);
        botaoDecisaoNao.setOnClickListener((v) -> {
                Intent novaIntent = new Intent(TelaAlgoritmoDecisaoMain.this, TelaAlgoritmoDecisaoMainEsquerda.class);
                novaIntent.putExtra("estratificacao", (Serializable) extras.get("estratificacao"));

                startActivity(novaIntent);
        });

        Button botaoDecisaoSim = (Button) findViewById(R.id.buttonMainSim);
        botaoDecisaoSim.setOnClickListener((v) -> {
                Intent novaIntent = new Intent(TelaAlgoritmoDecisaoMain.this, TelaAlgoritmoDecisaoMainDireita.class);
                novaIntent.putExtra("estratificacao", (Serializable) extras.get("estratificacao"));

                startActivity(novaIntent);
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_tela_algoritmo_decisao_main, menu);
        return true;

    }
}
```

A seguir nas Figuras 17, 18, 19, 23 e 24 são mostradas as telas do aplicativo onde o médico realiza as atividades de estratificar o risco da doença de um paciente, realiza diagnósticos e recomenda medicamentos entre outras funcionalidades pertinentes. Essas imagens foram extraídas a partir do emulador *Genymotion* desenvolvido pela empresa *Genymobile*, que fora utilizado para emular um dispositivo Android e assim não depender de um dispositivo real para desenvolver a ferramenta, tornando todo o processo de executar a aplicação bem mais prático.

Figura 17 – Tela Principal do Sistema.

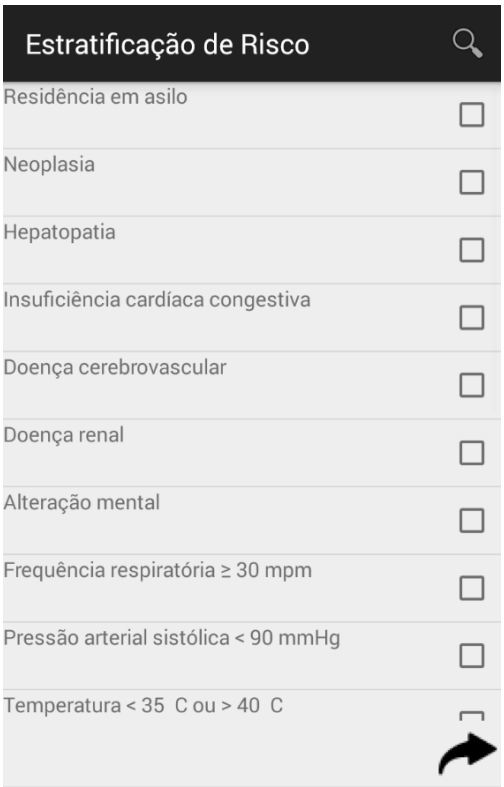


Na Figura 17 é apresentada a principal e primeira tela exibida ao usuário quando o mesmo inicia o aplicativo. A partir dela o usuário tem acesso a todas as funcionalidades do sistema, além de conseguir realizar as configurações necessárias para que o aplicativo se comunique com o servidor que detêm a base de dados. Conforme é possível observar na imagem as funcionalidades do sistema estão localizadas no centro da tela sendo elas: *Realizar Diagnóstico*, onde o médico especialista realiza todo o processo descrito no protocolo médico da PAC conforme já citado na subseção 2.1.1, iniciando-se com a realização da estratificação de risco conforme exibido na Figura 18; *Consultar Registros*, onde o médico consegue consultar registros gerados (paciente, estratificações de risco e diagnósticos) pelo aplicativo ou consultar no banco de dados remoto, conforme mostra a Figura 20; e *Baixar Registros*, onde o médico realiza a atualização dos registros armazenados localmente através do *download* dos registros oriundos do banco de dados remoto.

Já os dois botões restantes localizados na parte inferior da tela são utilizados a fim de configurar o aplicativo para que o mesmo consiga interagir com o servidor remoto. O

botão localizado na parte inferior esquerda (ícone de monitor) é utilizado para a definição do IP e da porta de acesso ao servidor, já o botão localizado na parte inferior direita da tela (ícone de chave) tem como propósito armazenar localmente a chave criptográfica do médico que será utilizada para decifrar as mensagens trocadas entre o aplicativo e o servidor, pois sem o usa dela o aplicativo não conseguirá compreender o conteúdo das mensagens trocadas entre os sistemas.

Figura 18 – Tela de Estratificação de Risco da PAC.

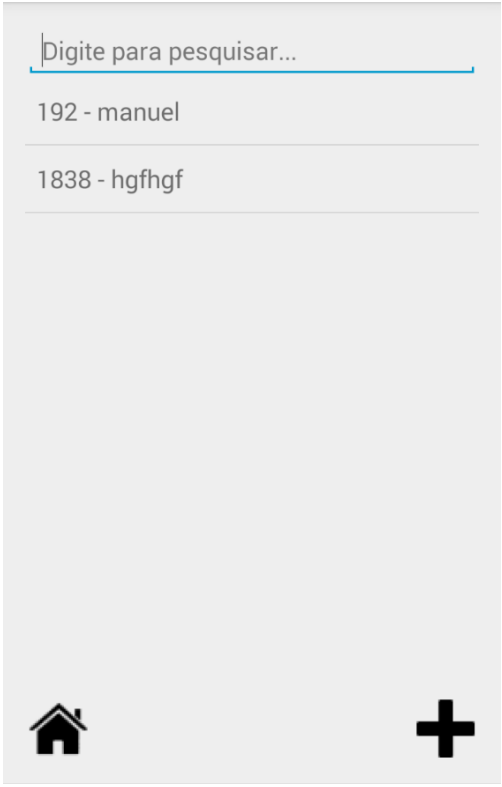


Na Figura 18 é exibida a tela onde o médico especialista realiza a estratificação de risco de um paciente previamente selecionado. Para isso, ele interage com o botão de busca (ícone de lupa) localizado na parte superior direita da tela, e logo é direcionado a tela de busca por registros (Figura 20), onde irá pesquisar e selecionar um dos registros armazenados localmente, irá realizar uma busca no servidor remoto por um registro de paciente caso o desejado não seja exibido, ou ainda cadastrar um novo paciente. Assim que um paciente é selecionado o médico deve utilizar dessa lista de itens apresentada na Figura 18 para marcar os fatores de risco para o paciente selecionado. Quando o médico finaliza o procedimento marcando todos os itens que achou necessário, ele pode interagir com o botão localizado na parte inferior direita (ícone de seta curvada) para finalizar a estratificação de risco, onde será direcionado a Figura 19 que apresenta o resultado da estratificação realizada.

Figura 19 – Tela de Resultado da Estratificação de Risco da PAC.

Na Figura 19 são apresentados os dados do paciente selecionado na parte superior da tela (Nome, Idade, RG e Sexo), e abaixo o resultado da estratificação gerada pelo médico, onde são mostradas as informações do *Tipo de Tratamento* recomendado, *Grau da Letalidade*, *Pontuação* (pois cada item marcado na Figura 18 possui uma pontuação que determinará os demais dados da estratificação) e a *Classe de Risco* em que o paciente se encontra. Além dessas informações, conforme é visto na Figura 19 existem dois botões localizados na parte inferior da imagem. O botão localizado na parte inferior esquerda tem como propósito realizar a persistência do registro, seja localmente, localmente e no servidor ou apenas no servidor, conforme mostrado na Figura 22. Essas opções são padrões para os demais tipos de registros manipulados pelo sistema e não apenas para os dados de estratificação. Já o botão localizado na parte inferior direita (ícone de seta curvada) direciona o médico as telas de execução do algoritmo de decisão de tratamento conforme mostra a Figura 23.

Figura 20 – Tela de Pesquisa de Registros.

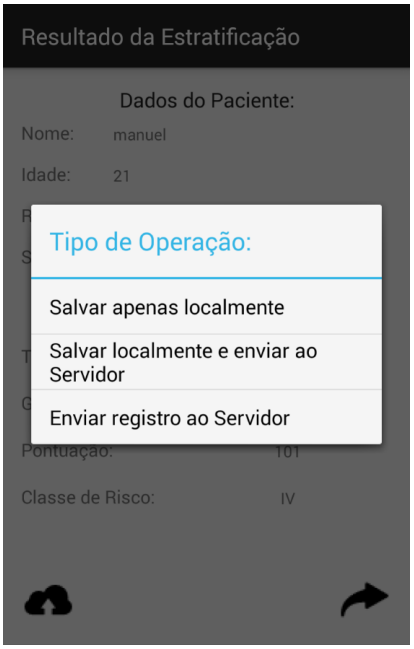


Na Figura 20 é mostrado a tela de pesquisa padrão para todos os tipos de registros gerados e manipulados pelo aplicativo. Nela basicamente constam o campo de pesquisa na parte superior da tela onde o usuário realiza a entrada de dados a fim de localizar determinado registro (no exemplo da imagem tratam-se de registros de pacientes), um botão que leva o usuário de volta a tela inicial do aplicativo (Figura 17) localizado na parte inferior esquerda da tela (ícone de casa), e um botão que leva o usuário a tela de cadastro de um novo registro localizado na parte inferior direita da tela (ícone de sinal positivo). Em relação a ação realizada pelo botão de adicionar novo registro, essa irá depender do propósito da tela de pesquisa em questão. Como na Figura 20 trata-se de uma pesquisa de pacientes, o botão de adicionar caso for acionado irá direcionar o usuário a tela de cadastro de paciente (Figura 21).

Figura 21 – Tela de Cadastro de Paciente.

Na Figura 21 é exibido o formulário de cadastro de paciente. Nele constam os dados obrigatórios para que um novo paciente seja armazenado no banco de dados do sistema. Além disso, como padrão a todos os itens interativos da interface do aplicativo que realizam a operação de persistência de dados, existe na parte inferior direita da tela o botão que irá exibir aos usuários as opções de persistência do novo registro, conforme mostra a Figura 22.

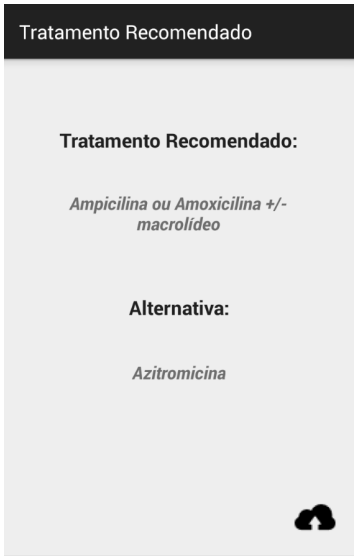
Figura 22 – Tela de Opções de Persistência de Registro.



Na Figura 23 é exibido a tela inicial do algoritmo de decisão do protocolo médico da PAC já citado anteriormente (Figura 1). Na presente tela e nas demais seguintes são exibidas perguntas simples ao médico que deve respondê-las interagindo com os botões "Sim"e "Não". Respondendo a todas as perguntas é exibido ao médico uma das 5 possíveis telas de tratamento recomendado, conforme é visto na Figura 24.

Figura 23 – Tela do Algoritmo de Decisão para Realizar o Diagnóstico da PAC.

Figura 24 – Alternativas de Tratamento Recomendado.

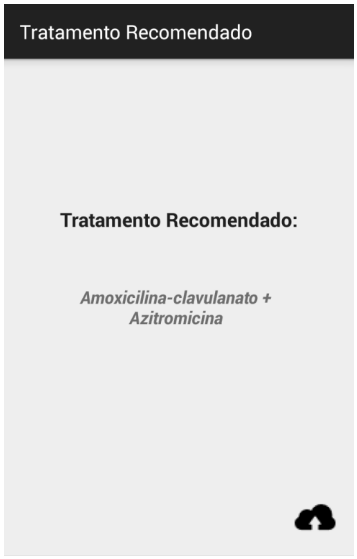


(a) Tela de Tratamento Recomendado #1.

(b) Tela de Tratamento Recomendado #2.

(c) Tela de Tratamento Recomendado #3.

(d) Tela de Tratamento Recomendado #4.



(e) Tela de Tratamento Recomendado #5.

4.4.2 Persistência

A persistência dos dados do sistema distribuído aqui proposto acontece de forma diferente no Cliente (Android) e no Servidor (*Rest Web Service*).

No *Rest Web Service* optou-se pela utilização de um Banco de Dados MySQL para realizar o gerenciamentos dos dados gerados pela Aplicação Android, visto o número considerável de registros que podem ser gerados e levando em consideração que gerenciá-los sem a utilização de um Banco de Dados seria de fato mais trabalhoso e pouco ágil.

Já no que diz respeito a aplicação executada no Android, como esse trata-se de uma arquitetura embarcada conforme já visto na seção 2.3, existem uma série de limitações como poder computacional, espaço de armazenamento, consumo energético entre outros. Justamente pensando nessas limitações optou-se por não utilizar um banco de dados na Aplicação Android para armazenar os registros recebidos do servidor, dessa forma evitando qualquer acrescimento de processamento exigido pela aplicação que consequentemente exigiria outros recursos do dispositivo móvel. Sendo assim, toda a persistência desses registros é feita através de arquivos XML salvos localmente pois esses são fáceis de manipular no Android e ocupam pouco espaço de armazenamento mesmo com um grande número de informações gravadas em arquivo.

Na Figura 25 é mostrado os arquivos XML salvos localmente no diretório *Shared-Preferences* da Aplicação Android. Já nas Figuras 26, 27 e 28 são mostrados os conteúdo desses arquivos. Cabe ressaltar que esses arquivos armazenados localmente são sempre consultados pela aplicação primeiramente antes de qualquer interação com o *Rest Web Service*. Isso é realizado com o intuito de tornar os procedimentos mais ágeis e dar certa flexibilidade quanto a não dependência total de conexão com o servidor para realizar procedimentos na aplicação, sendo essas feitas apenas periodicamente quando necessário.

Figura 25 – Arquivos XML locais gerados no diretório interno da aplicação Android.

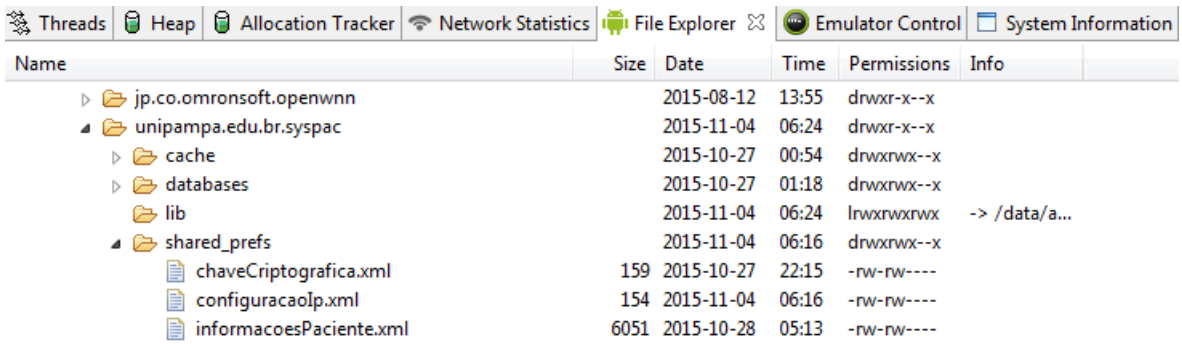


Figura 26 – Conteúdo do arquivo XML contendo a chave criptográfica do Médico.

chaveCriptografica.xml

```
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="idMedico">1</string>
4      <string name="chave">oimeunome1#sAADA</string>
5  </map>
```

Figura 27 – Conteúdo do arquivo XML contendo a configuração do ip de acesso ao Servidor.

configuracaoIp.xml

```
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="porta">8080</string>
4      <string name="ip">192.168.0.103</string>
5  </map>
```

Figura 28 – Conteúdo do arquivo XML contendo as informações dos registros de Pacientes.

informacoesPaciente.xml

```
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <int name="numRegistrosPacientes" value="3" />
4      <int name="idPaciente_1" value="1" />
5      <string name="registroGeralpaciente_1">6534566521</string>
6      <string name="nomePaciente_1">Gean Trindade</string>
7      <int name="idadePaciente_1" value="21" />
8      <string name="sexoPaciente_1">M</string>
9      <int name="idPaciente_2" value="2" />
10     <string name="registroGeralpaciente_2">7034562134</string>
11     <string name="nomePaciente_2">Maria Joaquina</string>
12     <int name="idadePaciente_2" value="44" />
13     <string name="sexoPaciente_2">F</string>
14     <int name="idPaciente_3" value="3" />
15     <string name="registroGeralpaciente_3">5563147852</string>
16     <string name="nomePaciente_3">José Quintana</string>
17     <int name="idadePaciente_3" value="30" />
18     <string name="sexoPaciente_3">M</string>
19 </map>
```

4.4.3 Envio e Recebimento de Mensagens

No que diz respeito ao envio e recebimento de mensagens trocadas entre instancias da aplicação Android e o *Rest Web Service*, essas mensagens são estruturadas no formato XML que é cifrado/decifrado com uma chave criptográfica e só assim enviadas/recebidas via protocolo HTTP pela rede, conforme já descrito na subseção 4.3.1.

Os métodos utilizados para receber e enviar dados foram os métodos *Get* e *Post* respectivamente, visto que esses são padrões do protocolo HTTP. No Android as Figuras 29 e 30 exibem um exemplo das chamadas desses métodos.

Figura 29 – Utilização da operação *Get* no Android.

```
AsyncHttpClient client = new AsyncHttpClient();
client.get("http://" + ip + ":" + porta + "/SysPACServidor/webResources/paciente/buscaTodosOsPacientes",
                                                                    null, new AsyncHttpResponseHandler() {
    @Override
    public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {...}

    @Override
    public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable error) {...}
});
```

Figura 30 – Utilização da operação *Post* no Android.

```
AsyncHttpClient client = new AsyncHttpClient();
RequestParams params = new RequestParams();
params.put("paciente", xmlCriptografadoConvertido);

client.post(url, params, new AsyncHttpResponseHandler() {
    @Override
    public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {...}

    @Override
    public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable error) {...}
});
```

Já no *Rest Web Service* como o mesmo trabalha com a ideia de oferecer recursos a serem consumidos por um ou mais clientes – no cenário desse trabalho os clientes são os dispositivos Android que executam a aplicação médica – cada recurso é representado por uma classe, e esses são adicionados como recursos através do método *addRestResourceClasses* da classe raiz do *Web Service* chamada de *ApplicationConfig*, conforme é visto na Figura 31.

Figura 31 – Classe de Gerenciamento dos Recursos do Web Service: *ApplicationConfig*.

```
@javax.ws.rs.ApplicationPath("webResources")
public class ApplicationConfig extends Application {

    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }

    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(br.edu.unipampa.controlador.DiagnosticoRecurso.class);
        resources.add(br.edu.unipampa.controlador.EstratificacaoRecurso.class);
        resources.add(br.edu.unipampa.controlador.PacienteRecurso.class);
    }
}
```

No *Rest Web Service* existem três classes que oferecem recursos aos clientes Android conforme é visto na Figura 31, sendo elas: *DiagnosticoRecurso*, *EstratificacaoRecurso* e *PacienteRecurso*. Um exemplo da implementação desses recursos pode ser observado na Figura 32.

Figura 32 – Classe de Recursos do Paciente: *PacienteRecurso*.

```
@Path("paciente")
public class PacienteRecurso {

    private final XStream stream;
    private final String CHAVE_CRIPTOGRAFICA = "oimeunome1#sAADA";

    public PacienteRecurso() {...8 linhas }

    @GET
    @Path("/buscaTodosOsPacientes")
    @Produces("text/plan")
    public String buscaTodosOsPacientes() throws Exception {...6 linhas }

    @GET
    @Path("/buscaTodosOsPacientesLimitado")
    @Produces("text/plan")
    public String buscaTodosOsPacientesLimitado() throws Exception {...6 linhas }

    @GET
    @Path("/buscaPaciente/{pacienteId}")
    @Produces("text/plan")
    public String buscaPaciente(@PathParam("pacienteId") Integer id) throws Exception {...6 linhas }

    @POST
    @Path("/cadastraPaciente")
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    public String cadastraPaciente(@FormParam("paciente") String dados) {...23 linhas }
}
```

Na classe de recurso vista na Figura 32 fica em destaque a presença de algumas anotações exigidas pelo *Rest Web Service* para que a classe seja considerada um recurso. Sobre a declaração da classe existe a anotação *@Path* que recebe o valor "paciente". Essa anotação tem como propósito configurar o endereço de acesso ao recurso via URL. Além disso, a anotação *@Path* é também utilizada sobre a declaração de métodos na classe, onde da mesma forma é especificado o caminho de acesso da funcionalidade daquele método. Dessa maneira o acesso aos recursos acontece de forma hierárquica, iniciando-se pelo endereço da classe e posteriormente do método específico pertencente a essa classe.

Na Figura 32 é também possível notar a presença de duas outras anotações de grande importância para o funcionamento do recurso, são elas as anotações *@GET* e *@POST*. Essas anotações caracterizam o tipo de operação suportada pelo método em que estão presentes. Sendo assim, todo o método que contenha a anotação *@GET* irá atender apenas as requisições *Get* oriundas do aplicativo Android, e o mesmo se aplica no caso dos métodos que contêm a anotação *@POST* sobre sua declaração.

4.4.4 Criptografia

A fim de garantir a confidencialidade dos dados trafegados entre os componentes da arquitetura optou-se pela utilização da criptografia, uma vez que não é adequado enviar/receber dados médicos em texto plano já que são informações sigilosas dos pacientes.

Algoritmos criptográficos existem em grande quantidade e para um vasta variedade de casos. Um bastante conhecido e utilizado é o algoritmo de cifra simétrica, que utiliza de uma mesma chave para cifrar e decifrar os dados de uma mensagem. Esse algoritmo fora utilizado para que a confidencialidade dos dados do sistema distribuído aqui proposto fosse garantida, pois além de simples ele é suficientemente seguro e eficaz.

Na Figura 33 são exibidas as implementações de dois métodos que utilizam o algoritmo de criptografia simétrica para cifrar e decifrar dados.

Figura 33 – Métodos de cifragem e decifragem das mensagens.

```
public static byte[] criptografar(String textoPlano, String chaveEncriptacao) throws Exception {
    Cipher encripta = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec key = new SecretKeySpec(chaveEncriptacao.getBytes("UTF-8"), "AES");
    encripta.init(Cipher.ENCRYPT_MODE, key, new IvParameterSpec("AAAAAAAAAAAAAAAA".getBytes("UTF-8")));
    return encripta.doFinal(textoPlano.getBytes("UTF-8"));
}

public static String decriptar(byte[] textoEncriptado, String chaveEncriptacao) throws Exception {
    Cipher decripta = Cipher.getInstance("AES/CBC/PKCS5Padding");
    SecretKeySpec key = new SecretKeySpec(chaveEncriptacao.getBytes("UTF-8"), "AES");
    decripta.init(Cipher.DECRYPT_MODE, key, new IvParameterSpec("AAAAAAAAAAAAAAAA".getBytes("UTF-8")));
    return new String(decripta.doFinal(textoEncriptado), "UTF-8");
}
```

Como é visto em detalhes na Figura 33 tanto o método *criptografar* quanto o *decriptar* sempre recebem como parâmetros o texto a ser cifrado ou decifrado e a chave criptográfica, pois sem ela não é possível realizar qualquer procedimento de cifragem/decifragem dos dados.

Conforme já mencionado, os dados trafegados entre os componentes da arquitetura são colocados no formato XML e só depois cifrados e enviados na rede. Nas Figuras 34 e 35 são mostrados respectivamente os dados estruturados no formato XML e como esses dados ficam após serem cifrados. Inclusive caso um invasor intercepte a informação trafegada o que ele visualizará é justamente o que é visto na Figura 35.

Figura 34 – Exemplo de mensagem estruturada em XML.

```
[ 11-04 06:29:39.032  3546: 3546 D/Msg Recebida Descriptografada:]

<pacientes>
<paciente>
<idPaciente>1</idPaciente>
<registroGeralpaciente>418757896</registroGeralpaciente>
<nomePaciente>Gean </nomePaciente>
<idadePaciente>21</idadePaciente>
<sexoPaciente>M</sexoPaciente>
</paciente>
<paciente>
<idPaciente>2</idPaciente>
<registroGeralpaciente>121225341</registroGeralpaciente>
<nomePaciente>Maria Frantchesca </nomePaciente>
<idadePaciente>23</idadePaciente>
<sexoPaciente>F</sexoPaciente>
</paciente>
<paciente>
<idPaciente>3</idPaciente>
<registroGeralpaciente>7094606543</registroGeralpaciente>
<nomePaciente>Pedro Gaudério</nomePaciente>
<idadePaciente>54</idadePaciente>
<sexoPaciente>M</sexoPaciente>
</paciente>
</pacientes>
```

Figura 35 – Exemplo de mensagem cifrada.

```
[ 11-04 06:29:39.028  3546: 3546 D/Msg Recebida via Get:]

$ ,(   ¨ K HÇûØ/ß ,íË…p ( Ê»òßnüŸ Tùúy¸ ñ D  Î¦¥Yè.·+ì-Â Oß#¹8 +i@£ *õà¬%¯¿q��¤ hÆô “¯ ûRî¹  ma»
ˆûÓÜ*àÈ@ Ú  ÿœ²¼ „ÒÅ\2 sj5-þ .»”gœ2 &§^Ö-  m±”Qm
'êe¤ H@™å ¨sûwƒú„0Ý Ø¿NjËÙ/ô'8²æ „Æ L¬uâÓ9+FyÕ  ä ;Øß°  $uäM-Í›Œ¾± Ë
;ÜÝ%©:  4E ˆÛ¦ âzHÔ PõáØµ þX¥ã.`– ª£Æ‡ ‹¨⅛´–Ã ß  uY· ‹��§Hb-Ù ~:¤¬9fYQNìµM·.3[     > 2 Ø¬Þ Ðzø>iö¾ÜÕ -q
C?9b3í4.  ™Ú°ˆ@Êçí^=/¥ÎÀaA¶  !ÖJEZ{|]æ9-c·Õâá ™ÞKmùÚ2. CÛ"��÷ÃÕl X× ”ä·å  ² ·¹
4��ñ¼%mZ §Ñkĭ¬a CNµ>ã¨ZÍ*¯¯1 Ñ µ Æ·à+^™ alõT£' Ê êŒ ÔÕƒ>_ b ë ¡õ6·ùî%/  üub Ð aj}‡tõùzÇê ˜¾Ö YÉt;œRŠ##
2é/Ã ' ›ÙzÝ°%– ü á e·®û²± Ü½ ?ýàzâ¬bûVY¾ +Å Û t5ë ¥ sà9ø/Å–{¹md*” |éÕ ¿   ÑÃôË¯(é“¯ø6–¦±]^⅛» ·²á8a{
¤Wõ5Û£X–é¯) ·¯æ>ßYÓ²¤“z r.“Å: å,âu;¡  ¬ jN0Òë Qu± ¢  ¤o  û+ªÇ  ñ  8iySÚë Qüçxðæ
Ó 2Ý‡!YM¸l ~b& È¢ ù®O Ø– µgË ·)Q @! „Ô† ?gÉ°Œ£Ìl  z Ò»Þd L  ã¬ ¬°v–ðÞ Ò%B¼Å
Ñ²æ™úXÊ
î+ L ?5 À ó…ª *ÐØ ¢lô~
‹ôÐó ó8 p°hý ~¡h¿å  Ð™ E W  @=ù û  _ã ?uh+°V°>3öÚ+  G  c
7ói8áU 2,¡ârÅÜâXãØ+ \ ô× ò„tl¦I}Èë ¿õ $dQ”  o ��D°ßß©Nq °S%Tt †'Â\VƒMÕ  7G'†øõìý¡Ã%ìhå–óõ6M$ye¾C/Ë
È =°\‡ P2"µä
q,¹›±óï Ô·¿Êb:3Í¯– jÎ!üßý?“ $Ø¹ y{¶ åµm’  6}çg–+ÂZÞ =¤ö"`)Ýh” ›Î ‹*šÏÊ,PX0>± x1$«>M(R ÿ›tÂy  Üw®% ÙR†
›%l  7*Ê ßB × TTR'Å]œA"g }ß B×#4îÞ!'Ñë ª    £ˆSn –¿P Õ¦&ÒY…nhù®Ú³Ën¤* ‘§)ÐìÒ hÂí/‡ßÊU 3 Í+Å Ík2B+Ù› T°
Ñ°Så¼`6 ÌÌ¬}msôÖ lÊó¨rS¥»IÒƒ” Îà* ÊÑ ó'å  Dq¦Iñ¸; à ~ó¬¡ ¤¨ö¼}çy r|‹iJÉ; ”>$p$øÕ. ó8–ù ±ýõfÓõ]  õZW|
```

4.5 Testes

Nesta seção é mostrada a realização dos testes de *software* realizados no decorrer do desenvolvimento da ferramenta. Na subseção 4.5.1 são descritos os testes unitários realizados e na subseção 4.5.2 o compilado de resultados obtidos.

4.5.1 Testes Unitários

Testes unitários tem como propósito testar o menor fragmento de um sistema, sendo esse geralmente um método de uma classe. Testes unitários são independentes de outros tipos de teste pois validam funcionalidades de forma individual. Com eles foi objetivado testar as principais funcionalidade do *software*, pois essas são as de maior risco e devem ser validadas quanto a sua corretude.

Em conjunto com os testes unitários fora utilizado a técnica de teste de caixa preta, onde o objetivo é verificar as saídas esperadas das unidades de acordo com vários tipos de entradas. Essa técnica não considera a estrutura do código da unidade, mas apenas as saídas esperadas conforme as várias possibilidades de entradas de dados que uma unidade pode receber. Essa técnica se justifica uma vez que no desenvolvimento de um *software* é comum que o desenvolvedor crie a solução pensando apenas nos casos de sucesso, onde considera apenas as entradas válidas nas unidades e acaba por esquecer ou negligenciar as possíveis entradas inválidas que as unidades podem receber, e que quando não consideradas ou tratadas podem comprometer o funcionamento do *software*.

4.5.2 Resultados dos Testes

Para a realização dos testes foram consideras classes específicas que realizam papéis fundamentais na ferramenta. A Tabela 10 apresenta o compilado dos resultados dos testes.

Tabela 10 – Compilação do status dos testes realizados.

Classes testadas	**Nº de testes**	**Porcentagem de cobertura**
Componente Controller		
ActivityTelaCadastroPaciente	2	90%
ActivityTelaBuscaRegistro	3	85%
ActivityTelaEstratificacaoRisco	2	90%
ActivityTelaDiagnostico	3	90%
Componente DAO		
CRUD	4	100%
Componente Util		
ManipulaXML	5	95%
Criptografia	2	100%
UtilizadorRest	2	100%

5 Avaliação da Ferramenta

Neste capítulo é descrita a avaliação da ferramenta, onde foram realizadas avaliações com usuários e comparativos de desempenho com demais ferramentas semelhantes. O experimento com usuários realizado em ambiente controlado é descrito em detalhes na seção 5.1, na seção 5.2 são apresentados e discutidos os resultados obtidos desse experimento e na seção 5.3 são apresentados os comparativos com outras ferramentas.

5.1 Descrição do Experimento

O presente experimento fora realizado com a participação de 6 usuários voluntários com idade variando entre 20 e 28 anos, onde todos eles possuíam experiência na utilização de dispositivos móveis como celulares *smartphones* e com a utilização do sistema operacional Android, requisitos necessários para a participação do experimento.

A fim de avaliar de forma satisfatória a ferramenta fora passado uma tarefa simples, porém extremamente relevante no contexto abordado pela ferramenta: Realizar um novo diagnóstico de um paciente do início ao fim, persistindo os dados do novo registro na nuvem. Foram então passadas aos voluntários algumas instruções básicas de como utilizar o aplicativo, porém nada muito detalhado, já que um dos propósitos do experimento era justamente avaliar questões de usabilidade da ferramenta. Essa tarefa delegada aos usuários fora executada no *smartphone Sony Xperia M* 3 vezes, e para cada execução foi capturado o tempo levado por cada um deles para realizar todo o processo de geração de um novo diagnóstico.

Assim que essa etapa prática do experimento foi finalizada fora passado aos voluntários um questionário a ser preenchido a fim de coletar as opiniões a respeito dos seguintes atributos de qualidades avaliados durante o experimento: Acurácia, Apreensibilidade, Capacidade para ser Instalado, Comportamento em Relação ao Tempo, Inteligibilidade, Interoperabilidade e Operacionalidade. A construção do questionário[1] (que pode ser encontrado no Apêndice A) fora embasada na normativa ISO/IEC 9126-1 de 2003.

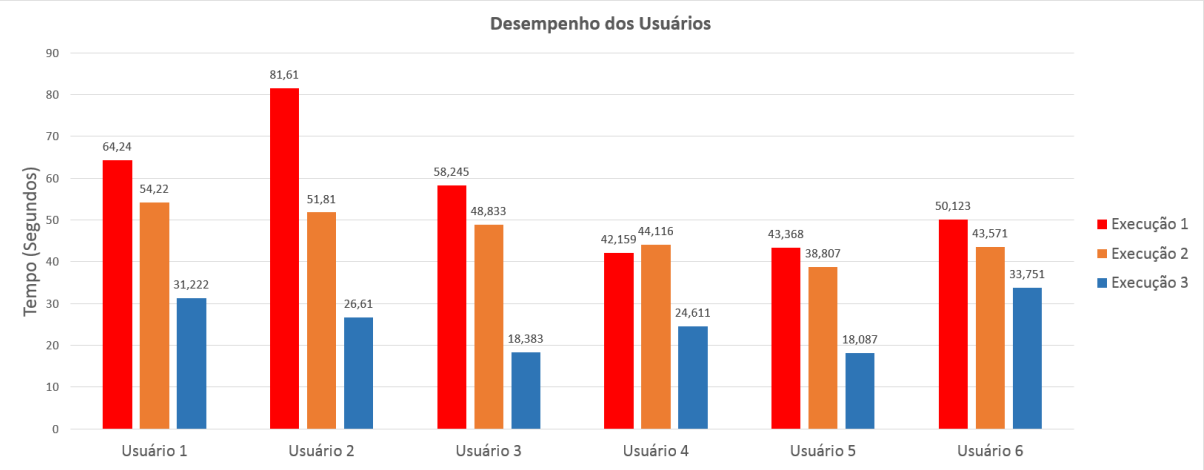
5.2 Resultados Obtidos do Experimento

Conforme já citado anteriormente, durante a execução do experimento foram coletados amostras de tempo para cada uma das 3 execuções do procedimento passado aos usuários. Esses dados foram compilados em dois gráficos, na Figura 36 se deu foco na

[1] **Link online para o questionário** - http://minilink.es/3gex.

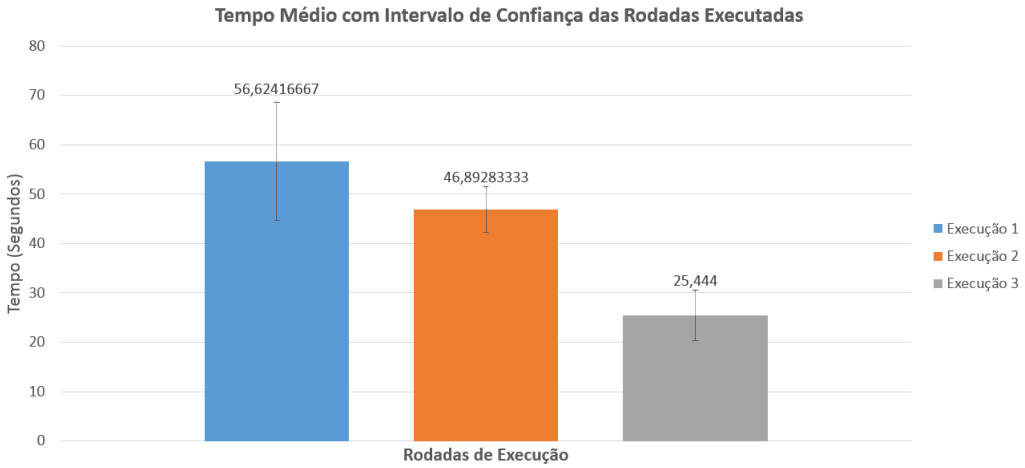
observação dos dados de desempenho individual de cada usuário, e na Figura 37 nos dados de cada uma das 3 execuções realizadas no experimento.

Figura 36 – Gráfico de desempenho dos usuários



Na Figura 36 é visto o tempo levado por cada usuário para realizar as tarefas com amostras em cada uma das três execuções. O interessante desse gráfico é perceber a evolução do tempo decorrido, que conforme mostra o gráfico diminui ao passar das execuções indicando um grau de aprendizagem rápido dos usuários ao utilizar a ferramenta.

Figura 37 – Gráfico de compilação dos dados das rodadas executadas



Já na Figura 37 é mostrado os dados com enfoque na avaliação de cada uma das rodadas de execuções realizadas pelos usuários. Em relação ao tempo médio de cada uma delas esse teve um significante decréscimo ao decorrer das execuções conforme visto no gráfico, reforçando ainda mais a progressão de aprendizagem relativamente rápida na

utilização da ferramenta, e que é fortalecida ainda mais se for considerado o curto período de utilização da mesma por parte dos usuários.

Além de utilizar-se da média de tempo, com o intuito de avaliar os resultados obtidos de forma confiável optou-se por tratá-los estatisticamente utilizando o *Intervalo de Confiança*, muito utilizado em resultados de pesquisas cientificas para indicar o quanto são confiáveis. O intervalo de confiança tem como propósito justamente indicar a confiabilidade de uma estimativa, sendo que quanto menor for mais confiável é a estimativa. Conforme é possível observar no gráfico esse intervalo de confiança é representado pelas linhas na vertical sobre as barras que representam o tempo de cada execução.

Como é possível visualizar na Figura 37 o intervalo de confiança é maior na execução 1. Isso se dá pelo fato de que o tempo levado pelos usuários fora bem diferente uns dos outros, o que acarretou em um desvio padrão maior e consequentemente o intervalo de confiança também fora maior. Já nas demais execuções esse intervalo é menor, porém apenas na execução 3 nota-se uma redução do intervalo de confiança pois esse não intersecta com os intervalos das execuções 1 e 2.

Além da análise do tempo coletado, através do questionário respondido pelos usuários voluntários que participaram do experimento fora possível avaliar aspectos de qualidade do sistema. Na Tabela 11 é possível observar a síntese dos resultados obtidos através do questionário. Conforme exibido na tabela constam os atributos de qualidade analisados e para cada um deles existe um valor derivado da escala likert[2]

Tabela 11 – Resultados obtidos do experimento.

Atributos de Qualidade	**Mínimo**	**Máximo**	**Média**	**Moda**
Acurácia	4	5	4,5	4
Apreensibilidade	4	5	4,66	5
Capacidade para ser Instalado	1	3	2	3
Comportamento em Relação ao Tempo	4	5	4,5	4
Inteligibilidade	4	5	4,5	5
Interoperabilidade	3	5	4,16	4
Operacionalidade	4	5	4,66	5

Na Tabela 11 é possível notar que o valor mínimo obtido fora do atributo *Capacidade para ser Instalado*. Isso ocorreu devido a dificuldade observada pelos usuários na configuração inicial do aplicativo, onde é necessário especificar o IP e a porta de acesso ao servidor, além de ser necessário a definição da chave criptográfica do médico que será utilizada para decifrar as mensagens trocadas entre o aplicativo e o servidor.

Observando a Tabela 11 é possível notar que com exceção do atributo *Capacidade para ser Instalado* mencionado anteriormente, e da *Interoperabilidade* que teve nota mí-

[2] **Escala Likert** - É uma escala comumente utilizada em questionários que mede o nível de concordância do usuário a respeito de uma pergunta em uma escala de 1 a 5, onde 1 corresponde a "Não concordo plenamente"e 5 a "Concordo plenamente".

nima 3, todos os demais atributos obtiveram a nota mínima 4, que na escala analisada que vai de 1 a 5 representa um resultado positivo na avaliação da ferramenta. No que diz respeito as notas máximas, todos os atributos obtiveram nota igual á 5, com exceção da *Capacidade para ser Instalado*, que inclusive teve o menor valor tanto na Média quanto na Moda, justamente pela dificuldade sentida pelos usuários já apontada. As demais notas máximas dos atributos analisados foram igual à 5, e os valores da média se mantiveram a cima de 4. Outro dado interessante observado nos resultados através da métrica Moda, é que a nota máxima 5 fora a que mais apareceu relacionada aos atributos de qualidade. Após todas essas analisas conclui-se que a ferramenta atingiu índices satisfatórios de usabilidade conforme apontaram os usuários, porém deve-se dar atenção a melhoria do processo de configuração inicial do aplicativo como um trabalho futuro.

5.3 Testes Comparativos

Além do teste com usuários descrito na seção anterior, foram realizados testes comparativos da ferramenta desenvolvida com demais aplicativos de cunho médico a fim de verificar aspectos de desempenho como consumo de processamento da CPU, quantidade de armazenamento e quantidade de dados enviados/recebidos em operações realizadas na rede. A seguir são apresentados gráficos que comparam os dados desses 3 aspectos analisados, de maneira que é posto em foco o comparativo do aplicativo *SysPAC* desenvolvido nesse trabalho com 4 ferramentas de cunho médico e com uma que não segue esse perfil. Na Figura 38 é mostrado os dados de porcentagem de processamento da CPU e quantidade de armazenamento utilizados pelos aplicativos, e na Figura 39 as quantidades de dados enviados/recebidos pelas ferramentas.

O principal critério de inclusão dos aplicativos médicos no comparativo fora considerar aqueles mais próximos com o que a ferramenta desenvolvida se propõem a fazer: Auxiliar médicos no tratamento e diagnóstico de um doença. Apesar de não ter sido possível encontrar aplicativos tão semelhantes, foram encontrados aplicativos médicos que de certa forma oferecem suporte a profissionais da saúde provendo informações relevantes seja sobre doenças, medicamentos ou outras informações de relevância aos profissionais do meio. Esses aplicativos foram então ranqueados conforme suas avaliações (fora considerado os aplicativos com avaliação mais positiva) e número de *downloads* na Google Play (loja oficial de aplicativos da Google), e por fim foram selecionados os 4 aplicativos de cunho médico melhor qualificados, que são eles: EM Guidance[3], Epocrates[4], Medscape[5] e Skyscape[6]. Além desses, fora incluído o navegador Chrome[7] para dispositivos móveis,

[3] **EM Guidance** - https://play.google.com/store/apps/details?id=emguidance.tompsa.
[4] **Epocrates** - https://play.google.com/store/apps/details?id=com.epocrates.
[5] **Medscape** - https://play.google.com/store/apps/details?id=com.medscape.android.
[6] **Skyscape** - https://play.google.com/store/apps/details?id=com.medpresso.android.ui.
[7] **Chrome** - https://play.google.com/store/apps/details?id=com.android.chrome.

com intuito de ter presente nos comparativos uma "aplicação de uso popular", e dessa forma ter uma espécie de "padrão de referência"para os dados providos pelas demais aplicações. Cabe salientar que todos esses aplicativos possuem utilização livre, ou seja, não fora necessário adquirir uma licença paga para utilizá-los.

Figura 38 – Comparativo de consumo da CPU e armazenamento de dados.

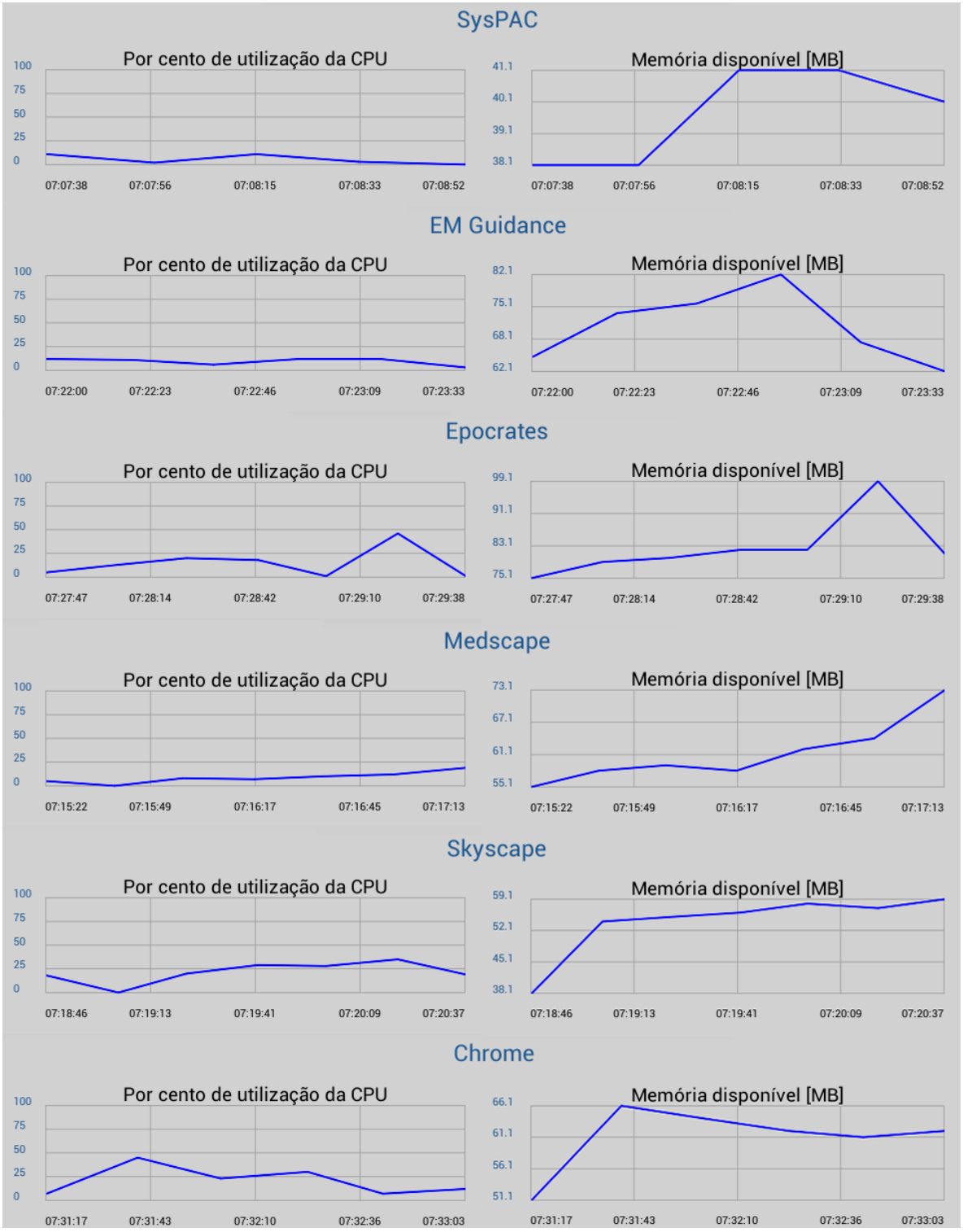
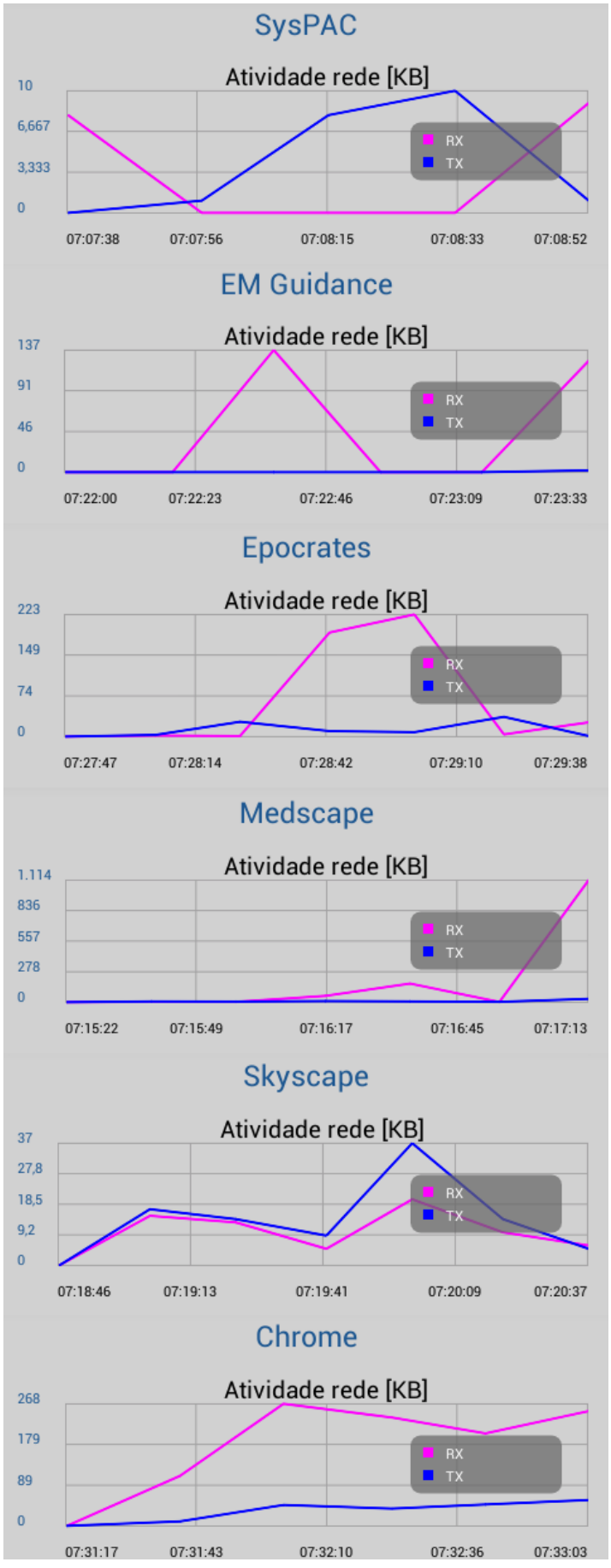


Figura 39 – Comparativo da quantidade de dados enviados/recebidos através da rede.



Nas Figura 38 e Figura 39, é possível observar que os dados foram coletados por um período de tempo pequeno, mais precisamente de apenas 2 minutos devido as limitações das ferramentas disponíveis encontradas para a realização de tal coleta e análise de dados[8]. A fim de minimizar isso foram realizadas várias execuções e amostragens para cada um dos aplicativos a fim de compreender em quais operações esses utilizam recursos consideráveis do *smartphone*, já que não haveria propósito medir dados de aplicações ociosas. Assim que traçados esses padrões, essas operações de cada aplicativo foram reproduzidas e os dados foram coletados. Dessa maneira nos gráficos estão presentes a análise de dados de operações que exigem certo processamento do *smartphone*, onde a intenção fora realizar uma análise mais igualitária possível para que os resultados possuam veracidade.

Tabela 12 – Resumo da análise de desempenho dos aplicativos analisados.

Aplicativo	**Média % utilização da CPU**	**Máxima % utilização da CPU**	**Qtd média de armazenamento (MB)**	**Qtd máxima de armazenamento (MB)**	**Qtd de dados recebida (KB)**	**Qtd de dados enviada (KB)**
SysPAC	5%	11%	39MB	40MB	17KB	20KB
EM Guidance	9%	12%	71MB	82MB	262KB	2KB
Epocrates	14%	46%	82MB	99MB	446KB	85KB
Medscape	8%	19%	61MB	72MB	1MB	64KB
Skyscape	21%	35%	53MB	59MB	69KB	96KB
Chrome	20%	45%	60MB	65MB	1MB	198KB

Como é possível observar na Tabela 12 que apresenta em números os resultados dos gráficos das analises comparativas, o aplicativo *SysPAC* apresenta dados menores em comparação as demais ferramentas, comprovando que utiliza poucos recursos do *smartphone* se comparado a essas aplicações.

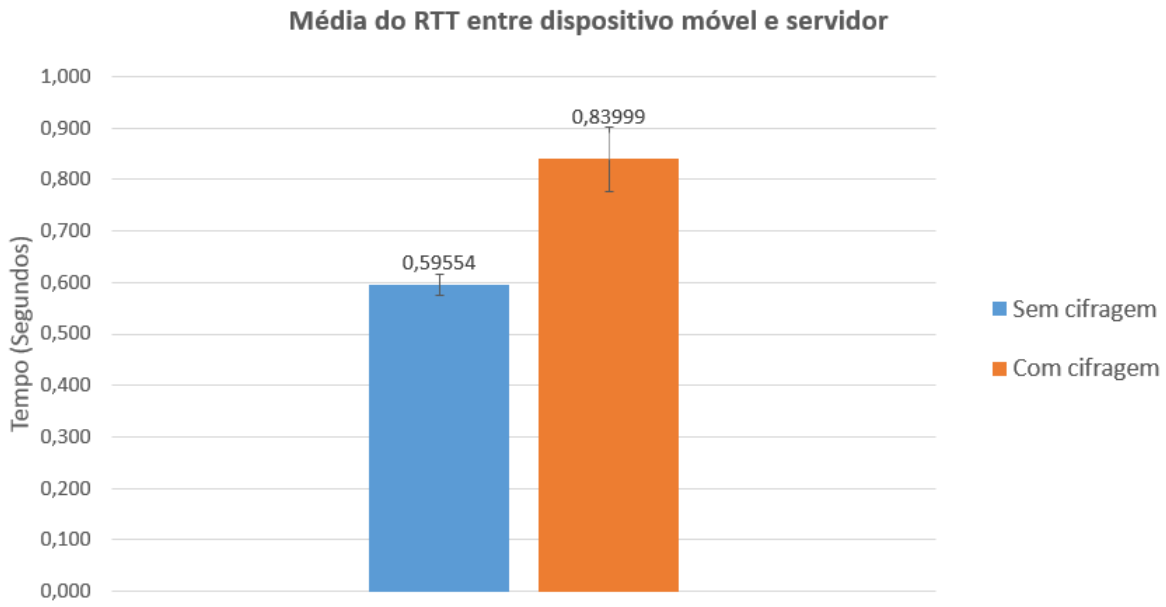
Cabe ressaltar que algumas dessas aplicações possuem recursos mais sofisticados como é o caso do próprio navegador Chrome, e que obviamente utiliza mais recursos do *smartphone*. Sendo assim, querer igualar a aplicação desenvolvida nesse trabalho em termos de consumo de recursos não é algo cabível. Porém, essa comparação é válida até certo ponto para que possa ser traçado certo parâmetro de comparação do quanto consomem de recursos essas aplicações em um *smartphone* executando no sistema Android, e o quanto a ferramenta desenvolvida está consumindo.

Além dos comparativos de uso de recursos das aplicações, a fim de avaliar os atrasos nas comunicações de um *smartphone* com o servidor, fora medido o tempo médio do *Round Trip Time* (RTT). O RTT é o tempo decorrido de uma informação chegar ao destinatário e o remetente receber sua confirmação. No experimento realizado foram enviadas 200 requisições de um *smartphone* executando o aplicativo *SysPAC* a um servidor (*Rest Web Service*) e calculado seu RTT. As requisições por registros enviadas pelo *smartphone* ao

[8] **CPU Monitor** - Ferramenta utilizada para coletar os dados das demais apps, e que pode ser encontrada em: https://play.google.com/store/apps/details?id=com.bigbro.ProcessProfiler.

servidor eram então retornadas em uma *String* no formato XML contendo 30 registros de Pacientes e possuindo um tamanho de 10KB. Como o aplicativo e o servidor trocam mensagens cifradas através da rede, fora testado esses envios de informações com e sem criptografia a fim de avaliar o impacto no tempo que essa representa. Na Figura 40 é possível visualizar o gráfico que compila essa avaliação de atrasados nas comunicações.

Figura 40 – Tempo médio do RTT entre dispositivo móvel e servidor.



Conforme é visto na Figura 40 são exibidos duas barras representando o tempo médio do RTT entre o dispositivo móvel e o servidor para as 200 amostras coletadas, onde foram considerados os tempos utilizando a cifragem na troca das mensagens e outro sem a cifragem (trafegando apenas a informação plana). Como já era esperado, o tempo do RTT das mensagem não cifradas fora menor devido ao processamento menor requerido, já que não são executadas no servidor os algoritmos de criptografia das mensagens.

Por um lado que a utilização da criptografia nas mensagens trocadas acarrete mais atrasado nas comunicações, seu uso é justificável uma vez que através criptografia é possível garantir a confidencialidade dos dados trafegados, sendo dessa forma mais vantajoso que tenha-se um ínfimo atrasado mas que a segurança relacionada à confidencialidade da informação seja garantida.

Na Figura 40 ainda constam as métricas de intervalo de confiança, onde o intervalo menor aparece nos dados coletados das informações que não eram cifradas. Isso ocorreu pelo fato do desvio padrão ser menor nas mensagens não cifradas, comportamento esse que não se manteve presente quando fora utilizado a criptografia no envio das mensagens.

6 Conclusões

Apesar de aparentemente esquecida, a PAC é uma infecção pulmonar grave que ainda gera altos índices de hospitalizações e mortes ao redor do mundo. Propor soluções tecnológicas que auxiliem médicos no tratamento de doenças e a economizar tempo em tarefas manuais cotidianas é algo amplamente bem visto pela comunidade médica mas que é pouco realizado na prática. Visando suprir essa necessidade, a proposta desse trabalho surgiu, onde fora desenvolvido uma ferramenta que utiliza de uma abordagem distribuída e segura para auxiliar médicos no tratamento de pacientes com PAC.

A fim de avaliar a ferramenta proposta foram realizadas diversas validações e análises. Uma delas fora a realização de uma análise comparativa com 4 aplicações de cunho médico semelhantes ao aplicativo *SysPAC* aqui proposto, e uma sem seguir esse perfil. Como resultado foi possível concluir que o aplicativo *SysPAC* utiliza poucos recursos do Android em relação aos demais. Ademais, fora realizado cálculos de 200 amostras do RTT para as mensagens cifradas e não cifradas trocadas entre o dispositivo móvel (executando o aplicativo) e o servidor. Através da média de tempo calculada foi possível concluir que o acréscimo de atraso provido pela utilização da criptografia é ínfimo em comparação com o benéfico da confidencialidade dos dados que ela provê. Além disso, através de um experimento em ambiente controlado realizado com 6 voluntários, onde também fora utilizado um questionário que coletou suas opiniões em relação ao uso da ferramenta, concluiu-se que o aplicativo possui uma progressão de aprendizagem rápida, considerando a evolução do tempo levado pelos usuários na execução das tarefas delegadas. Também a partir das respostas obtidas através do questionário, observou-se que a ferramenta se mostrou intuitiva, fácil de utilizar e adequada ao problema que se propõem a resolver.

Como limitações observadas, além da dificuldade na configuração inicial da ferramenta apontada pelos usuários no questionário, na questão de segurança a ferramenta ainda limita-se a apenas garantir a confidencialidade das informações, deixando de atender outras propriedades de segurança. Além disso, o próprio experimento realizado apresenta limitações visto o pequeno número de usuários participantes, e pelo fato de que nenhum deles é um médico especialista.

Como trabalhos futuros visa-se realizar validações com médicos especialistas e com isso coletar suas percepções e opiniões a fim de evoluir ainda mais a ferramenta. Ademais, fazê-la atender a mais propriedades de segurança como a implementação de rotinas de autenticação seria muito benéfico. Além disso, visa-se implementar versões do aplicativo para outras plataformas como para iPhones que executam o Sistema Operacional iOS e Tablets Android/iOS, e assim estender o público utilizador da ferramenta.

Referências

AULER, A. et al. *Tratamento de Pneumonia Adquirida na Comunidade (PAC)*. Porto Alegre, Rio Grande do Sul, 2010. Citado 9 vezes nas páginas 14, 16, 18, 19, 20, 21, 22, 33 e 36.

BURMAN, M. E.; WRIGHT, W. L. Diagnosis and management of community-acquired pneumonia: Evidence-based practice. *The Journal for Nurse Practitioners*, Elsevier, v. 3, n. 9, p. 633–649, 2007. Citado na página 14.

BUTT, S.; SWIATLO, E. Treatment of community-acquired pneumonia in an ambulatory setting. *The American journal of medicine*, Elsevier, v. 124, n. 4, p. 297–300, 2011. Citado na página 14.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. *Distributed systems: concepts and design*. [S.l.]: pearson education, 2007. Citado 4 vezes nas páginas 24, 27, 28 e 30.

COUTINHO, P. C. *Primeiros passos com os serviços REST*. 2014. Disponível em: <http://www.devmedia.com.br/primeiros-passos-com-os-servicos-rest/28912>. Acesso em: 26/06/2015. Citado na página 29.

Eclipse Foundation. *Introduction to OpenUP*. 2015. Disponível em: <http://epf.eclipse.org/wikis/openup/>. Acesso em: 13/06/2015. Citado na página 39.

ER, O.; YUMUSAK, N.; TEMURTAS, F. Chest diseases diagnosis using artificial neural networks. *Expert Systems with Applications*, Elsevier, v. 37, n. 12, p. 7648–7655, 2010. Citado na página 14.

ER, O.; YUMUSAK, N.; TEMURTAS, F. Diagnosis of chest diseases using artificial immune system. *Expert Systems with Applications*, Elsevier, v. 39, n. 2, p. 1862–1868, 2012. Citado 2 vezes nas páginas 33 e 34.

FRIEDMAN, C. et al. Automating a severity score guideline for community-acquired pneumonia employing medical language processing of discharge summaries. In: AMERICAN MEDICAL INFORMATICS ASSOCIATION. *Proceedings of the AMIA Symposium*. [S.l.], 1999. p. 256. Citado 3 vezes nas páginas 36, 37 e 38.

GADELHA, R. *Desenvolvendo para Android: Arquitetura Android*. 2012. Disponível em: <http://www.tiselvagem.com.br/geral/desenvolvendo-para-android-arquitetura-android/>. Acesso em: 26/06/2015. Citado na página 26.

Google. *Android Anatomy and Physiology*. 2013. Disponível em: <http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf>. Acesso em: 26/06/2015. Citado na página 26.

GOOGLE. *Android Studio Overview*. 2015. Disponível em: <http://developer.android.com/intl/pt-br/tools/studio/index.html>. Acesso em: 20/10/2015. Citado na página 40.

IEEE. *IEEE 802.11^{TM} WIRELESS LOCAL AREA NETWORKS*. 2015. Disponível em: <http://www.ieee802.org/11/>. Acesso em: 20/06/2015. Citado na página 50.

JACKSON, M. L. et al. The burden of community-acquired pneumonia in seniors: results of a population-based study. *Clinical Infectious Diseases*, Oxford University Press, v. 39, n. 11, p. 1642–1650, 2004. Citado na página 14.

KUROSE, J. F. *Computer networking: a top-down approach featuring the Internet*. [S.l.]: Pearson Education India, 2005. Citado na página 30.

LECHETA, R. R. *Google Android-3ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec Editora, 2013. Citado 2 vezes nas páginas 25 e 26.

LI, X. Using mobile phone sensors to detect rapid respiratory rate in the diagnosis of pneumonia. *International Journal of Engineering and Technology*, v. 8, n. 4, 2015. Citado 2 vezes nas páginas 34 e 38.

LIM, W. S. et al. Bts guidelines for the management of community acquired pneumonia in adults: update 2009. *Thorax*, BMJ Publishing Group Ltd and British Thoracic Society, v. 64, n. Suppl 3, p. iii1–iii55, 2009. Citado na página 14.

LODE, H. M. Managing community-acquired pneumonia: a european perspective. *Respiratory medicine*, Elsevier, v. 101, n. 9, p. 1864–1873, 2007. Citado na página 18.

MAJUMDAR, S. R. et al. Statins and outcomes in patients admitted to hospital with community acquired pneumonia: population based prospective cohort study. *Bmj*, BMJ Publishing Group Ltd, v. 333, n. 7576, p. 999, 2006. Citado na página 14.

MARAGOUDAKIS, M. et al. Domain knowledge acquisition and plan recognition by probabilistic reasoning. *International Journal on Artificial Intelligence Tools*, World Scientific, v. 13, n. 02, p. 333–365, 2004. Citado 2 vezes nas páginas 17 e 18.

MATEUS, G. R.; LOUREIRO, A. A. F. *Introdução à computação móvel*. [S.l.]: DCC/IM, COPPE/UFRJ, 1998. Citado na página 24.

MEIRA, F. L. *Introdução ao Processo Unificado Aberto*. 2015. Disponível em: <http://open2up.blogspot.com.br/>. Acesso em: 13/06/2015. Citado na página 39.

PAYNE, T. H. Computer decision support systems. *CHEST Journal*, American College of Chest Physicians, v. 118, n. 2_suppl, p. 47S–52S, 2000. Citado na página 23.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: SN. *12th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.], 2008. v. 17, n. 1. Citado na página 31.

RAUT, M. et al. Estimating the economic impact of a half-day reduction in length of hospital stay among patients with community-acquired pneumonia in the us. *Current medical research and opinion*, Informa UK Ltd UK, v. 25, n. 9, p. 2151–2157, 2009. Citado na página 14.

SANTOS, S. S. *OpenUP: Um processo ágil*. 2015. Disponível em: <http://www.ibm.com/developerworks/br/rational/local/open_up/>. Acesso em: 12/06/2015. Citado na página 39.

SCHURINK, C. et al. Computer-assisted decision support for the diagnosis and treatment of infectious diseases in intensive care units. *The Lancet infectious diseases*, Elsevier, v. 5, n. 5, p. 305–312, 2005. Citado na página 23.

SOARES, M. et al. Development an application for mobile devices to aid in diagnosis and training of radiologists in detection of pneumonia in childhood–stradio. Workshop de Visão Computacional, 2013. Citado 5 vezes nas páginas 15, 24, 32, 33 e 38.

SOMMERVILLE, I. *Engenharia de Software*. PEARSON BRASIL, 2011. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Citado na página 41.

TANENBAUM, A. S. *Computer Networks 4th Edition*. [S.l.: s.n.], 2003. Citado 2 vezes nas páginas 27 e 28.

TOSIN, C. *Conhecendo o Android*. 2011. Disponível em: <http://www.dicas-l.com.br/arquivo/conhecendo_o_android.php>. Acesso em: 26/06/2015. Citado na página 26.

VERDAGUER, A. et al. Validation of the medical expert system pneumon-ia. *Computers and biomedical research*, Elsevier, v. 25, n. 6, p. 511–526, 1992. Citado 2 vezes nas páginas 35 e 36.

WARDLAW, T.; JOHANSSON, E. W.; HODGE, M. Pneumonia: the forgotten killer of children. New York New York UNICEF 2006 Sep., 2006. Citado na página 17.

WAZLAWICK, R. S. *Análise e Projeto de Sistemas da Informação*. [S.l.]: Elsevier Brasil, 2004. v. 2. Citado na página 41.

WELFER, D.; SILVA, R. C. F. da; KAZIENKO, J. F. A mobile application system for diagnosis and management of community-acquired pneumonia. In: IEEE. *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*. [S.l.], 2014. p. 341–346. Citado 5 vezes nas páginas 14, 15, 36, 37 e 38.

Apêndices

APÊNDICE A – Questionário Utilizado no Experimento Controlado

Questionário de Avaliação da Ferramenta SysPAC

Por meio desse questionário visa-se obter a opinião dos usuários que participaram do experimento em laboratório da ferramenta SysPAC.

*Obrigatório

Testador:

Não é obrigatório informar.

Idade: *

Dado necessário para traçar os perfis dos testadores que participaram do experimento.

Nível de experiência utilizando dispositivos móveis? *

Dado necessário para traçar os perfis dos testadores que participaram do experimento.

	1	2	3	4	5	
Pouquíssima experiência	○	○	○	○	○	Muita experiência

Nível de experiência utilizando o sistema operacional Android? *

Dado necessário para traçar os perfis dos testadores que participaram do experimento.

	1	2	3	4	5	
Pouquíssima experiência	○	○	○	○	○	Muita experiência

A instalação da ferramenta e sua configuração são processos simples? *

Leve em consideração o tempo e complexidade da instalação da ferramenta e de sua configuração (definição de: IP, Porta, Chave Criptográfica).

	1	2	3	4	5	
Muito simples	○	○	○	○	○	Díficil

As respostas das ações executadas foram demoradas? *

Leve em consideração o feedback visual da aplicação para cada ação executada.

	1	2	3	4	5	
Muito demoradas	○	○	○	○	○	Rápidas

Você teve dificuldades em utilizar a ferramenta? *

	1	2	3	4	5	
Muita dificuldade	○	○	○	○	○	Pouca dificuldade

As funcionalidades mostraram respostas coerentes com o que se propuseram a fazer? *

	1	2	3	4	5	
Pouco coerentes	○	○	○	○	○	Coerentes

A curva de aprendizado na utilização da ferramenta é lenta? *

O usuário demora para se habituar a utilizar a ferramenta? Demora para entender seu propósito? Suas funcionalidades?

	1	2	3	4	5	
Muito lenta	○	○	○	○	○	Rápida

A ferramenta é intuitiva no que se propõem a fazer? *

	1	2	3	4	5	
Pouco intuitiva	○	○	○	○	○	Intuitiva

As operações que envolviam interação com o Servidor (envio/recebimento de dados), foram demoradas? *

	1	2	3	4	5	
Muito demoradas	○	○	○	○	○	Rápidas

Sugestões e considerações:

Caso queira, expresse o que achou da ferramenta, sugestões do que pode ser melhorado ou qualquer outra informação relatando a experiência do uso da mesma. Sua contribuição é de grande importância para a evolução da ferramenta, obrigado!

Enviar

Nunca envie senhas pelo Formulários Google.