Pontifícia Universidade Católica
do Rio de Janeiro

Chrystinne Oliveira Fernandes

IoT4Health: Um Framework no Domínio de E-Health para Acompanhamento de Pacientes utilizando Agentes de Software

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Carlos José Pereira de Lucena

Rio de Janeiro

Agosto de 2015

Chrystinne Oliveira Fernandes

IoT4Health: Um Framework no Domínio de E-Health para Acompanhamento de Pacientes utilizando Agentes de Software

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Carlos José Pereira de Lucena
Orientador
Departamento de Informática – PUC-Rio

Prof. Ruy Luiz Milidiú
Departamento de Informática – PUC-Rio

Prof. Andrew Diniz da Costa
Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 27 de agosto de 2015

Chrystinne Oliveira Fernandes

Bacharel em Ciência da Computação pela Universidade Federal da Paraíba (UFPB). Mestranda pela PUC-Rio e Pesquisadora do Laboratório de Engenharia de Software (LES) da PUC-Rio.

Ficha Catalográfica

Fernandes, Chrystinne Oliveira

IoT4Health: Um Framework no Domínio de E-Health para Acompanhamento de Pacientes utilizando Agentes de Software / Chrystinne Oliveira Fernandes; orientador: Carlos José Pereira de Lucena. - Rio de Janeiro, PUC, Departamento de Informática, 2015.

v., 114 f,; il. ; 29,7 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas

1. Informática – Teses. 2. Internet of Things. 3. Multi-Agent Systems. 4. E-Health. 5. Framework. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para o meu pai, Gentil Fernandes de Andrade, que nos deixou há 31 primaveras. De quem não tenho lembranças, mas sei que herdei a paixão pela leitura, pelos números e pela lógica.

Agradecimentos

Ao meu orientador, Prof. Carlos Lucena, por toda dedicação e suporte dados a este trabalho. Sinto-me lisonjeada por ter um orientador que é referência na Ciência da Computação e também um ser humano incrível.

Aos membros da banca, Prof. Ruy Milidiú e Prof. Andrew Diniz, muito obrigada pelas contribuições. Aos professores do DI da PUC-Rio, por terem despertado em mim o desejo de continuar na vida acadêmica.

Aos colegas do LES e da SESIA, pelo apoio ao longo dessa caminhada. A Ariel, pela troca de conhecimentos.

À minha família, minha mãe, Fátima, meus irmãos, Chrystiane, Chrystofanes e Chrystofferson, e meu esposo, Rafael, que torceram para que tudo desse certo.

A Deus, pelo dom da vida.

Finalmente, agradeço à CAPES, por financiar esta pesquisa.

Resumo

Fernandes, Chrystinne Oliveira; Lucena, Carlos José Pereira de. **IoT4Health: Um Framework no Domínio de E-Health para Acompanhamento de Pacientes utilizando Agentes de Software.** Rio de Janeiro, 2015. 114p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A busca por soluções inovadoras no domínio de E-Health tem motivado amplamente a realização de pesquisa científica nesta área, cuja exploração pode trazer inúmeros benefícios à sociedade. Apesar dos recursos tecnológicos disponíveis atualmente, ainda existem muitos problemas no ambiente hospitalar. Visando contribuir para o desenvolvimento de soluções tecnológicas aplicadas à área da Saúde, destacamos a concepção do IoT4Health, um framework no domínio de e-Health para acompanhamento remoto de pacientes que utiliza agentes de software. Três instâncias do framework foram criadas para servir como provas de conceito: 1-Agents4Health; 2-Remote Patient Monitoring (RPM); 3-EHealth System. O Agents4Health é um sistema multi-agentes no domínio de E-Health desenvolvido como uma solução IoT para automatizar técnicas comumente utilizadas no acompanhamento de pacientes e no processo de coleta de sinais vitais. Esta solução compreende agentes de software e protótipos de hardware compostos por microcontroladores e sensores médicos, que trabalham em conjunto para tornar ambientes hospitalares mais proativos. Além disso, a solução provê acesso remoto aos dados dos pacientes armazenados em plataformas baseadas em nuvem, permitindo profissionais externos trabalharem de forma colaborativa com a equipe médica local. Um sistema web permite a visualização em tempo real dos dados dos pacientes capturados pelos sensores, como temperatura e frequência cardíaca, os quais são exibidos em forma de *line charts*. Agentes de software monitoram constantemente os dados coletados para detectar anomalias no estado de saúde dos pacientes e enviar alertas aos profissionais. O RPM, segunda instância do Iot4Health, também suporta o monitoramento de pacientes, através de aplicativos móveis, tendo foco na evolução do estado de

saúde do paciente. Por último, o EHealth System compõe o conjunto de aplicações criadas como forma de validar nossa ferramenta.

Palavras-chave

Internet of Things; Multi-Agent Systems; E-Health; Framework.

Abstract

Fernandes, Chrystinne Oliveira; Lucena, Carlos José Pereira de (Advisor). **IoT4Health: A Framework in E-Health Domain for Patient Monitoring by using Software Agents.** Rio de Janeiro, 2015. 114p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The search for innovative solutions in E-Health domain has largely motivated the conduct of scientific research in this area, whose exploration can bring numerous benefits to society. Despite the technological resources available nowadays, there are still many problems in the hospital environment. Aiming contribute to the development of technological solutions applied to this area, we highlighted the development of the IoT4Health framework. Two instances of this framework were built to serve as concept proof: 1-Agents4Health; 2-Remote Patient Monitoring (RPM); 3-EHealth System. The Agents4Health is a multi-agent system in E-health domain supported by an Internet of Things (IoT) solution to automate techniques commonly used in patients' treatment and data collection processes. This solution comprises software agents and hardware prototypes including sensors, micro-controllers that work together to make hospital environments more proactive. In addition, the solution provides remote storage of patient data in cloud-based platforms, allowing external professionals to work collaboratively with the local team. A web system enables real-time visualization of patient's record captured through sensors, such as temperature and heart rate values displayed as graphical charts through an intuitive interface. Software agents constantly monitor collected data to detect anomalies in patients' health status and send alerts to health professionals. The RPM also supports patient monitoring activities by using mobile applications, with focus on patient evolution. Finally, the Ehealth system comprises the set of applications created in order to validate our tool.

Keywords

Internet of Things; Multi-Agent Systems; E-Health; Framework.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

IoT	Internet of Things / Internet das Coisas
UbiComp	Computação Ubíqua
IA	Inteligência Artifical
SMA	Sistemas Multi-Agentes
JADE	Java Agent DEvelopment Framework

"O que está em jogo não é a transmissão daquilo que se inventa, mas antes a transmissão do poder de inventar."

Juan David Nasio

1
Introdução

Estamos vivendo um período em que a tecnologia está sendo cada vez mais usada para melhorar nossa qualidade de vida, seja aumentando nossa segurança, conforto e bem-estar, seja otimizando recursos humanos limitados como o tempo ou a capacidade de armazenar e processar dados, para extrair deles informação útil e gerar novos conhecimentos.

A Internet of Things (IoT)[1] é uma área de pesquisa relativamente recente que vem garantindo seu espaço nesse processo de mudanças tecnológicas que se reflete em nossos hábitos cotidianos e em nossa cultura de modo geral. A ideia básica por trás do conceito de IoT é ter um cenário onde coisas, pessoas e animais são identificadas de maneira única e podem comunicar-se pela Internet, com pouca ou nenhuma intervenção humana. Vale salientar que as Coisas, no âmbito da IoT, podem ser interpretadas como entidades do mundo real que vão fazer parte da rede IoT, compreendendo desde simples objetos como uma cadeira, um eletrodoméstico ou um carro, até seres vivos como plantas, animais e pessoas (Fernandes & Lucena, 2015a).

Desse modo, interações na rede, que até então estavam restritas aos tipos *human-to-human* ou *human-to-computer*, estão sendo ampliadas para suportar outros conceitos de interação envolvendo coisas, como os tipos *human-to-thing* e *thing-to-thing*. A possibilidade de comunicação das coisas é um dos fatores inovadores que a IoT tem trazido para expandir a aplicação da Internet. Como é o caso, por exemplo, da geladeira que passa a ser capaz de informar quando um determinado alimento aproxima-se ou já passou do prazo de validade.

Materializando este conceito de Internet das Coisas, atualmente já temos exemplos de sua aplicação em diversas áreas das nossas vidas, como transporte e logística e cuidados médicos. A tendência é que haja uma expansão do alcance da IoT, de modo a possibilitar o surgimento de ambientes cada vez mais inteligentes,

[1] Os termos Internet of Things e Internet das Coisas estão sendo abreviados como IoT.

os quais vislumbramos que possam evoluir para a dimensão de cidades inteligentes, que é um dos temas de pesquisa amplamente explorados neste campo.

Podemos relacionar diversos problemas comuns em nossas cidades que retardam a ocorrência de avanços sociais, como por exemplo: a gestão ineficiente de recursos públicos, que leva a desperdícios de verbas e à consequente insatisfação da população; O alto congestionamento de veículos que impacta negativamente a vida das pessoas, consumindo seu tempo e muitas vezes promovendo stress;

A busca de soluções para problemas desta natureza, bem como a perspectiva de que o conceito de cidades inteligentes torne-se cada vez mais próximo da nossa realidade, servem de motivação para a realização de pesquisas no contexto da IoT.

1.1. Declaração do Problema

No decorrer do trabalho, realizamos atividades de análise de domínio, onde identificamos diversos problemas no âmbito de e-Health, como por exemplo: (i) uso de processos manuais de coleta de dados de pacientes, propensos a erros; (ii) utilização de técnicas obsoletas de identificação de pacientes; (iii) armazenamento de informações sobre o estado de saúde dos pacientes em papel, o que traz uma série de desvantagens, como: dificuldade de distribuir informações entre os profissionais envolvidos no tratamento; necessidade de se ter espaço físico para manter os dados armazenados em papel; mais tempo alocado para indexar, buscar, gerenciar e atualizar estas informações; maior possibilidade de se ter duplicação nos dados; necessidade de refazer exames por perda de resultados;

1.2. Motivação

Como motivação da nossa pesquisa, buscamos desenvolver uma plataforma para lidar com alguns dos problemas levantados na etapa de análise de domínio. Para isto, focamos principalmente na concepção de um framework que suporta o desenvolvimento de aplicações no domínio de e-Health. Como prova de conceito desta ferramenta, apresentamos três aplicações no contexto de healthcare que foram desenvolvidas a partir da instanciação do nosso framework.

De modo geral, nossa pesquisa visa contribuir para que a medicina seja praticada de modo mais inteligente, diminuindo seu ônus e aumentando sua eficácia, através da análise, projeto, implementação, implantação e validação de soluções e-Health inovadoras.

1.3. Contribuições

O principal resultado deste trabalho é uma "teoria" sobre e-health, expressa sob um conjunto de módulos e organizada como frozen spots do nosso framework.

1.4. Metodologia utilizada

A estratégia utilizada para desenvolver o IoT4Health consistiu, inicialmente, em encontrar cenários com grande potencial para investigação dos conceitos de IoT e que também fossem alvos para a ampla aplicação de agentes de software. Nesta etapa, priorizamos e-Health como domínio de aplicação, o qual foi explorado através da nossa plataforma para distribuição de informações de healthcare, que compreende dispositivos de hardware, aplicações web desenvolvidas como sistemas multi-agentes e aplicativos móveis. Esta plataforma permite realizar interações que podem auxiliar o monitoramento de pacientes e aumentar a inteligência do ambiente onde estes estão sendo tratados.

Foram realizados dois estudos de caso, onde ambos foram planejados para investigar questões, como, por exemplo: Quais benefícios a coleta automática de dados vitais, realizada através de sensores, pode trazer? Quais anomalias podem ser detectadas através da nossa plataforma?

Além do levantamento dessas e de outras questões de pesquisa, na fase de planejamento do experimento também foram definidas algumas hipóteses com base no conhecimento prévio sobre o tema. Tanto as questões quanto as hipóteses serão apresentadas mais adiante, na seção 6.

A realização dos estudos de caso possibilitou o levantamento do conjunto de *features* necessárias às etapas de análise de domínio, modelagem, projeto, implementação, documentação e validação do framework IoT4Health.

1.5. Organização da Dissertação

As seções seguintes estão organizadas da seguinte forma: a Seção 2 apresenta o contexto em que está inserida nossa pesquisa, discutindo aspectos da Computação Ubíqua, SMA e IA. Na Seção 3, citamos trabalhos relacionados que encontramos na literatura e na indústria que estão alinhados à nossa pesquisa. A Seção 4 contém uma visão geral da Internet das Coisas, mostrando em mais detalhes aplicações comuns neste campo e as tecnologias mais utilizadas. Na Seção 5, abordamos conceitos gerais sobre Frameworks. Na Seção 6, apresentamos a metodologia utilizada, relacionando nossas questões de pesquisa, objetivos e a descrição do planejamento dos estudos de caso realizados. A Seção 7 traz os resultados obtidos, mostrando detalhes sobre o framework IoT4Health, com relação à análise de domínio, caracterização do domínio da aplicação, projeto, arquitetura, frozen spots, hot spots, implementação, documentação e provas de conceito. Na Seção 8, concluímos este trabalho, apresentando nossas contribuições e trabalhos futuros em que vislumbramos possíveis contribuições que podem ser trazidas pela continuação desta pesquisa.

2
Contextualização da Pesquisa

2.1.
Áreas de Pesquisa Envolvidas

A IoT corresponde ao campo de pesquisa no qual prevalecem os maiores esforços de investigação empreendidos neste trabalho. No entanto, a nossa proposta de trabalho assume um caráter multi-disciplinar, visto que nossas pesquisas se estendem a outras áreas como Sistemas Multi-Agentes (SMA), Computação Ubíqua e Inteligência Artificial (IA). Para explorar estas áreas em nossos estudos de caso, focamos em aplicações nos domínios de *Monitoring* e *Smart Environment*. (Figura 1).

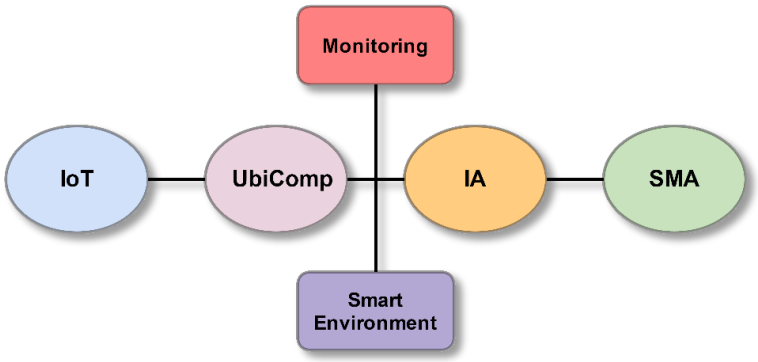


Figura 1 - Áreas de Pesquisa envolvidas e domínios de aplicação explorados.

A seguir, apresentamos brevemente alguns dos conceitos que consideramos fundamentais no contexto da nossa pesquisa e que são úteis para a compreensão do trabalho realizado.

2.1.1.
Ubiquitous Computing

> "*Why shouldn't we obtain digital information from our environment, from things that are familiar to us? I truly believe that soon our clocks, furniture and chairs will all talk with one another over the Internet. The big problem is that today everything still runs through the PC, through this narrow channel between monitor, keyboard and mouse. That's unnatural*".
>
> **Mark Weiser**, Cientista Chefe da Xerox Palo Alto Research Center (PARC).

Quando Mark Weiser cunhou o termo Ubiquitous Computing (UbiComp) no início dos anos noventa (Weiser, 1993), ele trouxe uma mudança de visão para a Ciência da Computação, onde o foco saía das máquinas e voltava-se para os seres humanos: computadores deveriam se tornar invisíveis para os usuários e serem capazes de resolver necessidades reais e cotidianas das pessoas. Ele imaginou que o poder computacional, ao invés de estar centrado em supercomputadores de alto custo, tamanho e potência, poderia ser distribuído entre dispositivos de computação menores e mais baratos espalhados pelo ambiente. Tais dispositivos poderiam estar dispersos em uma sala, por exemplo, associados a objetos que utilizamos em nosso dia-a-dia, como cadeiras, persianas e eletrodomésticos, tornando-os capazes de comunicarem-se pela Internet (Figura 2).

Figura 2 – Poder computacional distribuído pelo ambiente[2].

Segundo Weiser, a noção de ubiquidade surgiu a partir da observação das coisas do mundo real, como simples equipamentos de escritório. Weiser concebeu uma abordagem diferente para dispositivos computacionais com base nesta observação. Ao invés de projetar as diversas funcionalidades destes dispositivos em

[2]Fonte: http://www.siemens.com/innovation/en/publikationen/publications_pof/pof_fall_2002/internet_articles/ubiquitous_computing.htm

um único computador, ele propôs o caminho inverso: transferir o poder computacional de supercomputadores para dispositivos simples. Dessa forma, cada um desses dispositivos seria capaz de processar informações.

Weiser definiu a Computação Ubíqua como o método de aumentar o uso de computadores, disponibilizando-os através do ambiente físico, mas tornando-os efetivamente invisíveis para o usuário (Weiser, 1993).

2.1.2. Sistemas Multi-Agentes

Um agente (Wooldridge & Jennings, 1995) é um elemento de um sistema computacional que está situado em algum ambiente no qual é capaz de realizar ações autônomas, a fim de cumprir os objetivos que lhe foram delegados. Apesar de não existir uma definição universal do termo agente, existe um consenso na literatura de que a autonomia é uma característica chave para a atuação do agente (Wooldridge, 2009). Neste contexto, autonomia significa a possibilidade de atuação sem a intervenção humana ou de outros sistemas. Desse modo, nenhuma intervenção por pessoas ou outros sistemas é necessária para que os agentes decidam como realizar suas atividades, ainda que o conjunto de ações possíveis deva ser definido previamente.

Embora os agentes controlem seu estado interno e comportamento, eles não detêm inteiramente o controle do seu ambiente. Um agente possui um conjunto de ações que pode realizar e cuja execução pode resultar na modificação do ambiente. Por essa razão, considera-se que o agente tem controle parcial sobre seu ambiente, podendo influenciá-lo, de acordo com a ação que decidir realizar. Desse modo, um agente utiliza sua autonomia para decidir como agir, de modo a satisfazer seus objetivos. Outras propriedades dos agentes, por sua vez, são desejáveis apenas em domínios específicos, como é o caso do aprendizado, ou seja, a capacidade que possui de aprender a partir da sua experiência.

Agentes inteligentes tem capacidades como **reatividade**, **proatividade** e **habilidade social**, as quais diferem pelo modo como os ajudam a satisfazerem seus objetivos. A reatividade diz respeito à capacidade de perceber e responder tempestivamente às mudanças no ambiente. Já a proatividade é a capacidade de ter um comportamento orientado ao seu objetivo, através da tomada de iniciativa. E a

habilidade social é a capacidade de interagir com outros agentes e, possivelmente, com humanos. Ela não envolve apenas troca de informações, mas também a possibilidade de realizarem relações de negociação e cooperação (Russell & Norvig, 2013).

Com relação à arquitetura para agentes inteligentes, podemos dizer que pertencem a quatro classes, dependendo de como os agentes realizam o processo de tomada de decisão. Os agentes que tomam decisões através de dedução lógica são classificados como **agentes baseados em lógica**; existem os **agentes reativos**, em que a tomada de decisão é implementada em alguma forma de mapeamento direto da situação para a ação; em alguns casos, a tomada de decisão depende da manipulação de estruturas de dados representando as crenças, desejos e intenções dos agentes. Esta é uma propriedade dos **agentes BDI** (Belief-Desire-Intention) (Rao, Georgeff, & others, 1995); e, por último, os que seguem uma **arquitetura em camadas**, em que a tomada de decisão é realizada através de várias camadas de software, onde em cada uma delas o raciocínio sobre o ambiente é mais ou menos explícito (Wooldridge, 1999).

Um agente pode ocupar diferentes tipos de ambientes, que se classificam de acordo com suas propriedades, em (Russell & Norvig, 2013): **Acessível ou inacessível**, dependendo da informação que o agente possui sobre ele. Diz-se que o ambiente é acessível quando o agente pode obter informações completas, precisas e atualizadas sobre o estado do ambiente; **determinístico ou não-determinístico**, de acordo com a garantia de efeito que uma determinada ação terá neste ambiente. Um ambiente determinístico é aquele em que qualquer ação tem a garantia de um efeito único – não existe incerteza sobre o estado que irá resultar da execução de uma ação; **estático ou dinâmico**, dependendo do controle que o agente tem sobre as mudanças no ambiente. Um ambiente pode ser considerado estático quando pode-se assumir que ele irá permanecer inalterado, exceto pela execução de uma ação por parte do agente. Em contrapartida, um ambiente dinâmico é aquele no qual as mudanças ocorridas estão fora do controle do agente e que tem outros processos operando sobre ele; e, por último, um ambiente pode ser **discreto ou contínuo**, considerando as possíveis ações e percepções que o agente pode realizar. Em um ambiente discreto, existe um número finito e fixo de ações e percepções.

2.1.3.
Inteligência Artificial

Inteligência Artificial é o campo que tenta não apenas compreender, mas construir entidades inteligentes (Norvig & Russell, 2014), que podem ser interpretadas como agentes de software. O conceito de agente é amplamente abordado em IA, bem como o uso de técnicas que podem ser utilizadas pelos agentes para execução de tarefas e tomada de decisões.

Outro aspecto bastante explorado em IA é o processo de aprendizagem. O aprendizado é uma das possíveis características dos agentes. Pode-se dizer que um agente irá aprender se conseguir melhorar o seu desempenho nas tarefas futuras, a partir da observação do mundo. Dessa forma, o conhecimento sobre o mundo faz-se necessário para que os agentes possam tomar boas decisões. Este conhecimento é armazenado em uma base de conhecimento, através de uma linguagem de representação do conhecimento (Norvig & Russell, 2014).

3
Trabalhos Relacionados

Realizamos uma revisão sistemática para buscar trabalhos com enfoque nas áreas envolvidas nesta pesquisa - UbiComp, IoT, SMA e IA -, que estavam relacionados ao domínio de e-Health. Encontramos alguns trabalhos com abordagens similares à nossa, como é o caso da proposta dos autores Su e Wu- "*JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring*" (Su & Wu, 2011). O trabalho apresenta a implementação de uma infraestrutura de informação distribuída que utiliza o paradigma de agentes inteligentes para notificar automaticamente os profissionais responsáveis pelos cuidados dos pacientes a respeito de anormalidades em seu estado de saúde, além de oferecer aconselhamento médico à distância e a possibilidade de realização de monitoramento contínuo do estado de saúde dos pacientes.

Nos processos de monitoramento de saúde tradicionais, os pacientes precisam se deslocar com frequência até hospitais ou outras unidades de atendimento médico, o que consiste em um procedimento oneroso para os pacientes. Com o intuito de melhorar tais processos, Su & Wu defendem a adoção de sistemas de monitoramento móveis e ubíquos, que permitam análise imediata dos dados fisiológicos de cada paciente, bem como um feedback personalizado de sua condição em tempo real, através do uso de alarmes e lembretes. Nesta solução, pacientes podem ser avaliados, diagnosticados e tratados de modo remoto e ubíquo. Caso os pacientes apresentem sinais de rápido agravamento em suas condições de saúde, o sistema automaticamente notifica a equipe médica através de chamadas telefônicas ou mensagens SMS, dando suporte a um apoio médico de primeiro nível.

Um dos pontos interessantes deste trabalho e que apresentamos como um dos trabalhos futuros para a nossa pesquisa é o uso de um plug-in da ferramenta

JADE (Java Agent DEvelopment Framework)[3], chamado JADE-S[4], que permite implementar aspectos de segurança como controle de acesso e comunicação segura. Considerando que os dados dos pacientes são bastante sensíveis, a segurança é uma questão crítica que precisa ser levada em consideração quando se tem a intenção de avaliar o funcionamento desse tipo de sistema em ambientes reais.

A abordagem de (Su & Chu, 2014), por sua vez, foca no design e desenvolvimento de um sistema de informação distribuída baseada em agentes móveis, para permitir monitoramento fetal automático em tempo real, a partir de qualquer localização, sendo feito através de PDA, smartphone, laptop ou computador desktop. De acordo com Su e Chu, os benefícios dos Agentes Móveis consistem em: superar as limitações dos dispositivos clientes; customização; alta serventia; computação assíncrona e autônoma; interoperabilidade e acesso a dados locais; descentralização de processamento; otimização de recursos computacionais e de comunicação; e suporte ao projeto de aplicações que interagem com usuários humanos.

De acordo com (Mohammadzadeh & Safdari, 2014), inovações no campo da tecnologia da informação em aplicações de healthcare como sistemas de saúde móveis são muito importantes para se alcançar todo o benefício que a área de e-Health tem potencial para fornecer. Aplicações de diagnóstico médico remoto e sistemas de monitoramento proveem aos profissionais da saúde acesso a informações de pacientes e bases de dados centrais. *Mobile health* pode ser a solução para transpor as barreiras do serviço de saúde personalizado, acesso oportuno a informações do paciente relacionadas à saúde, especialmente em emergências, bem como evitar duplicação de testes, atrasos e erros no tratamento do paciente.

Como exemplo de solução e-Health com abordagem semelhante à que estamos usando, encontramos um dispositivo chamado HealthPatch MD - Vital Connect health-monitoring sensor[5] (Figura 3). Trata-se de um pequeno adesivo com um módulo que controla atividade cardíaca (ECG), frequência cardíaca (e variabilidade), frequência respiratória, temperatura da pele, postura e detecção de

[3] Fonte: http://jade.tilab.com/
[4] Fonte: http://jade.tilab.com/doc/tutorials/JADE_Security.pdf
[5] Fonte: http://www.engadget.com/2015/01/06/healthpatch-md-vital-connect/

quedas. Além disso, o dispositivo possui conexão para evitar que o indivíduo precise se deslocar a um centro médico para ser monitorado.

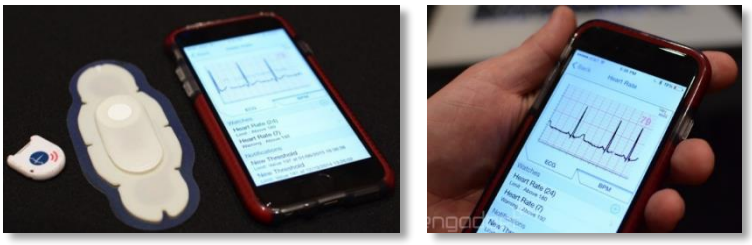


Figura 3 - HealthPatch MD – Dispositivo de monitoramento[6].

Outras soluções vêm se destacando na indústria, a exemplo do sistema IBM Watson para auxílio no tratamento de pacientes com câncer. IBM Watson for Oncology[7] é um sistema de computação cognitiva projetado para dar suporte aos médicos. Sistemas de computação cognitiva aprendem e interagem naturalmente com pessoas para estender aquilo que ambos, homens ou máquinas, poderiam fazer por conta própria. Eles ajudam especialistas humanos a tomarem melhores decisões ao lidarem com um volume de dados do alcance do Big Data.

Nesse projeto, o sistema Watson foi treinado pelo Memorial Sloan Kettering (MSK) Cancer Center, com o objetivo de disseminar a expertise do MSK para auxiliar médicos na avaliação de opções de tratamento para seus pacientes. Ele atua sugerindo opções de tratamento aos médicos, a partir da extração de informações chave nos registros de dados de saúde de cada paciente, fazendo referências cruzadas com a expertise do MSK e avaliando *case histories*.

O IBM Watson está evoluindo para ser capaz de auxiliar médicos a encontrar padrões clínicos com mais rapidez e precisão (Figura 4). Em uma sala de emergência, radiologistas podem visualizar até 200 casos e até 100.000 imagens por dia. Um projeto de pesquisa da IBM está ensinando IBM Watson a executar esta tarefa.

[6] Fonte: http://www.engadget.com/2015/01/06/healthpatch-md-vital-connect/
[7] http://www.ibm.com/smarterplanet/us/en/ibmwatson/watson-oncology.html

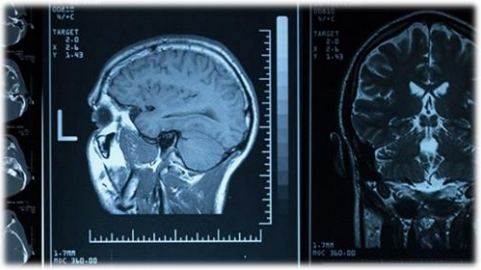


Figura 4 - Reconhecimento de padrões em imagens para apoio aos Radiologistas[8].

[8] Fonte: http://www.ibm.com/smarterplanet/us/en/think/healthcare/

4
Internet of Things

Embora os primeiros trabalhos em Computação Ubíqua tenham promovido a noção de distribuição do poder computacional entre os dispositivos dispersos pelo ambiente, estes ainda dependiam das pessoas para capturar dados e publicá-los na Internet.

Quando Kevin Ashton- cofundador e diretor executivo do laboratório Auto-ID Center do MIT-, cunhou o termo "Internet of Things", em 1999 (Ashton, 2009), ele considerou o cenário da época, onde um grande poder computacional estava acessível para processamento de dados, mas os computadores eram quase totalmente dependentes de pessoas para obter e publicar informações. Quase todo o volume de dados disponíveis na Internet até então havia sido capturado e criado por um ser humano – digitando, pressionando um botão de gravar, tirando uma foto ou lendo um código de barras. Essa dependência de intervenção humana gerava um gargalo devido a suas limitações de tempo, atenção e precisão, evidenciando que seres humanos não são tão eficientes em capturar dados sobre coisas do mundo real.

Além disso, se a dependência de seres humanos fosse limitada, de modo que os computadores fossem capazes de capturar os dados que deveriam processar, os custos de medir, rastrear e controlar coisas seriam reduzidos. Desse modo, através de sensores, atuadores e da tecnologia RFID, as coisas do ambiente poderiam ser identificadas, percebidas e controladas de forma mais autônoma. Nesse caso, as próprias coisas poderiam informar quando precisariam ser substituídas, consertadas ou até mesmo se estão adequadas para o consumo (Ashton, 2009).

De modo geral, a intenção, quando do surgimento da IoT, era que coisas pudessem ser identificadas e conectadas à Internet, com a capacidade de comunicarem-se de modo independente ou com intervenção mínima dos seres humanos. Esta mudança visava automatizar atividades humanas dispendiosas como a de coletar dados e inseri-los na rede. Desta forma, tarefas como estas passariam a ser delegadas às próprias coisas, que poderiam interagir entre si e também com pessoas e sistemas.

4.1.
Aplicações IoT

Atualmente, aplicações IoT já são uma realidade e vem gerando impacto em diversas áreas da nossa vida: na área pessoal, por exemplo, surgiram os conceitos de *domotics* e *smart house*; No campo profissional, surgiram os *smart offices*; Na Indústria, aplicações no ramo de transporte e logística despontam como grandes promessas para otimização de recursos; Na Administração pública, estamos experimentando aplicações que tratam da gestão mais eficiente de recursos públicos. Temos ainda uma grande aplicação da IoT na área de saúde, que vem explorando conceitos como e-Health e *Assisted living* (Figura 5).

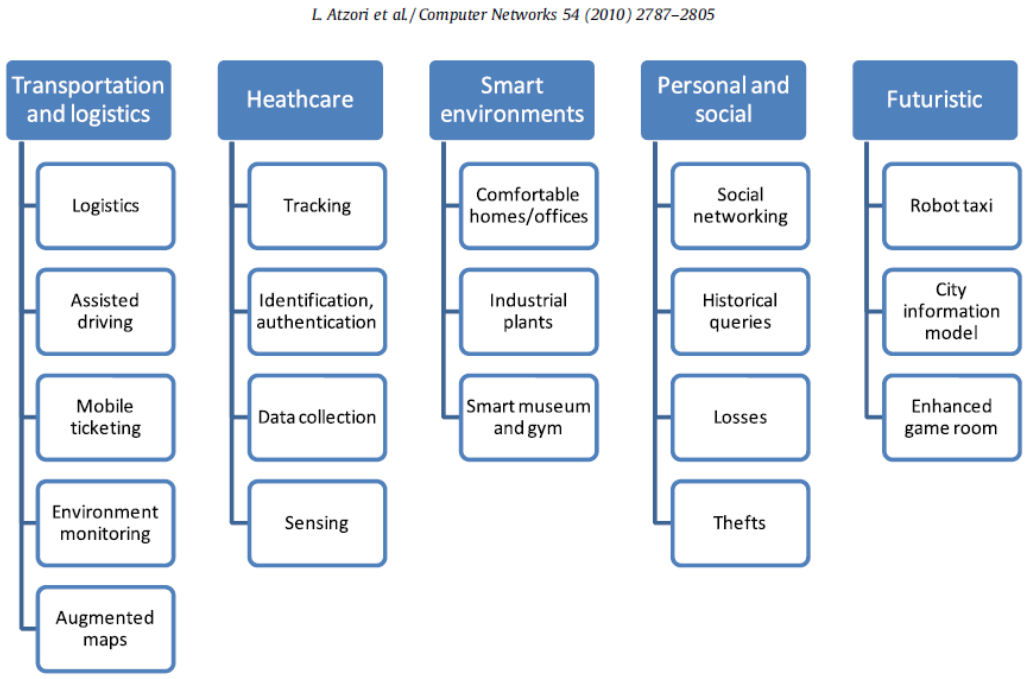


Figura 5 – Exemplos de domínios de aplicação da IoT.

4.2.
Tecnologias IoT

Existem diversas tecnologias que podem ser utilizadas no desenvolvimento de aplicações IoT. A concepção deste trabalho foi baseada no conceito de três tecnologias fundamentais para o embasamento das nossas investigações: RFID (Radio-Frequency IDentification) (D. K. Finkenzeller, 2010), microcontroladores e Sensores.

4.2.1.
Sistemas RFID e Auto-ID

RFID ou identificação por rádio frequência é um método de identificação automática que utiliza sinais de rádio, recuperando e armazenando dados remotamente através de dispositivos denominados etiquetas ou tags RFID (Figura 6). Estes dispositivos são usados para fins de identificação, sensoriamento e comunicação (Atzori, Iera, & Morabito, 2010).

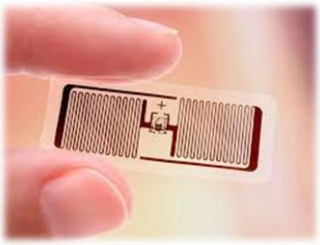


Figura 6 - Etiqueta RFID.

Sistemas de identificação automática (Auto-ID) estão tornando-se cada vez mais populares na indústria de serviços, compras e logística de distribuição, assim como em empresas de manufatura e sistemas de fluxo de material (Figura 7). Além de sistemas RFID, diversos procedimentos de identificação automática estão sendo adotados para fornecer informações sobre pessoas, animais, bens e produtos em trânsito. Podemos citar, por exemplo (Finkenzeller, 2010): 1- Sistemas de Códigos de Barras; 2- *Optical Character Recognition* (OCR); 3- Procedimentos Biométricos (Identificação por Voz e Identificação por Impressão Digital ou Datiloscopia); 4- Cartões Inteligentes (Cartões de memória e Microprocessadores);

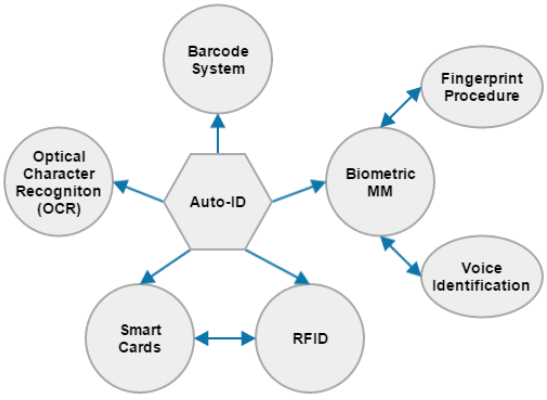


Figura 7 - Principais Sistemas de Identificação Automática (Auto-ID Systems)[9].

[9] Fonte: Adaptado de Finkenzeller (2010).

Apesar de extremamente baratas, as etiquetas de códigos de barras possuem a desvantagem de ter baixa capacidade de armazenamento e não poderem ser reprogramadas (Finkenzeller, 2010). Enquanto uma etiqueta de código de barras serve apenas para identificar o tipo de um produto, tags RFID identificam produtos individualmente. Dessa forma, a tecnologia RFID tem a capacidade de conceber a Internet das Coisas, uma vez que cada objeto que contiver uma tag RFID possuirá um código de identificação único que poderá funcionar como um endereço IP. Isto permite que eles sejam conectados à Internet para que possam ser monitorados e rastreados, desde a produção até a reciclagem. A tabela a seguir apresenta um comparativo entre os diferentes tipos de sistemas Auto-ID existentes:

Tabela 1 - Comparação entre diferentes sistemas de identificação automática[10].

System parameters	***Barcode***	***OCR***	***Voice recognition***	***Biometry***	***Smart card***	***RFID systems***
Typical data quantity (bytes)	1–100	1–100	–	–	16–64 k	16–64 k
Data density	Low	Low	High	High	Very high	Very high
Machine readability	Good	Good	Expensive	Expensive	Good	Good
Readability by people	Limited	Simple	Simple	Difficult	Impossible	Impossible
Influence of dirt/damp	Very high	Very high	–	–	Possible (contacts)	No influence
Influence of (optical) covering	Total failure	Total failure	–	Possible	–	No influence
Influence of direction and position	Low	Low	–	–	Unidirectional	No influence
Degradation/wear	Limited	Limited	–	–	Contacts	No influence
Purchase cost/reading electronics	Very low	Medium	Very high	Very high	Low	Medium
Operating costs (e.g. printer)	Low	Low	None	None	Medium (contacts)	None
Unauthorised copying/modification	Slight	Slight	Possible* (audio tape)	Impossible	Impossible	Impossible
Reading speed (including handling of data carrier)	Low ~4 s	Low ~3 s	Very low >5 s	Very low >5–10 s	Low ~4 s	Very fast ~0.5 s
Maximum distance between data carrier and reader	0–50 cm	<1 cm Scanner	0–50 cm	Direct contact**	Direct contact	0–5m, microwave

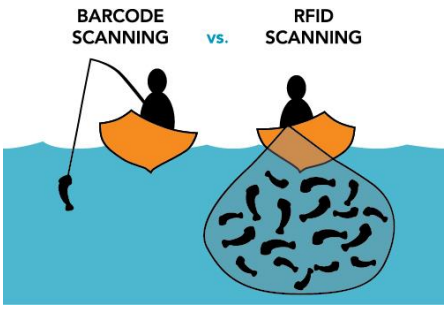
* O perigo de ocorrer repetição pode ser reduzido pela utilização de um gerador aleatório, de modo que o texto a ser falado não seja conhecido com antecedência.
**Isto apenas se aplica para impressão digital. No caso da avaliação da retina ou íris, o contato direto não é necessário ou mesmo possível.

[10] Fonte: Adaptado de Finkenzeller (2010).

Podemos citar inúmeros benefícios trazidos pela tecnologia RFID, como por exemplo, a possibilidade de um cliente passar por um caixa de um supermercado sem precisar retirar as compras do carrinho. Para isto, bastaria que cada produto contivesse uma tag RFID, cuja leitura seria feita por um leitor ao lado do caixa, por meio de uma conexão sem fios (Figura 8). Além disso, supermercados poderiam ter leitores RFID distribuídos por suas prateleiras, informando sempre que houvesse a necessidade de repor determinado produto.

Figura 8 - Supermercado com identificação automática de produtos[11].

Outra vantagem importante da tecnologia RFID diz respeito à sua transmissão por ondas de rádio, de forma que nenhum contato visível é necessário para que seu código de identificação seja lido. Isto possibilita a leitura simultânea de muitos códigos, inclusive à distância (Figura 9). Se os produtos etiquetados estiverem dentro de caixas, não há sequer a necessidade de abri-las. Esta tecnologia permite ainda que se tenha diversas facilidades para o controle de produtos, sendo possível, por exemplo, contar todos os itens de uma loja apenas pressionando um botão. Além disso, a contagem de produtos rotulados com RFID é cerca de cem vezes mais rápida[12] do que produtos identificados com códigos de barras e mais precisa[13].



[11] Fonte: http://blueskydesigngroup.com.au/project/coles-future-market-design-research-project/?doing_wp_cron=1434425014.5596399307250976562500

[12] http://rfidarena.com/2013/12/2/6-most-common-faqs-of-barcode-vs-rfid.aspx

[13] http://atlasrfid.com/jovix-education/auto-id-basics/rfid-vs-barcode/

Figura 9 - Leitura de código de barras versus leitura de tags RFID[14].

Conforme observa-se na Tabela 1, sistemas RFID apresentam inúmeras vantagens quando comparados com os outros sistemas de identificação. Eles estão conquistando os mercados de massa, sendo utilizados em cartões inteligentes sem contato, comumente utilizados como bilhetes em transportes públicos de curta distância (Figura 10).

Figura 10 - Leitor RFID e cartão inteligente sem contato[15].

Um Sistema RFID é sempre constituído de dois componentes (Finkenzeller, 2010):

1. O **Transponder**, que está localizado no objeto a ser identificado;
2. E o **Leitor** ou **Interrogador**, que, dependendo do design ou tecnologia utilizada, pode ser um dispositivo de leitura ou de leitura/escrita.

O transponder é um dispositivo de transporte de dados eletrônico. O fornecimento de energia e a troca de dados entre ele e o leitor são feitos através de campos magnéticos e eletromagnéticos. Normalmente, consiste de um elemento de acoplamento (*coupling element*) e um microchip eletrônico. Além disso, o transponder não possui seu próprio fornecimento de tensão (bateria), o que o torna totalmente passivo quando não está na zona de interrogação de um leitor. A energia requerida para ativá-lo é fornecida por meio da unidade de acoplamento (sem contato), assim como o pulso de temporização e os dados (Figura 11).

Um leitor tipicamente contém um módulo de rádio frequência (transmissor e receptor), uma unidade de controle e um elemento de acoplamento para o transponder. Além disto, muitos leitores são equipados com uma interface adicional

[14] http://rfidarena.com/2013/12/2/6-most-common-faqs-of-barcode-vs-rfid.aspx

[15] Fonte: http://www.rfidjournal.com/articles/view?1209

(RS 232, RS 485 etc.) para habilitá-los a repassar os dados recebidos para outros sistemas (PC, sistema de controle de robôs etc).

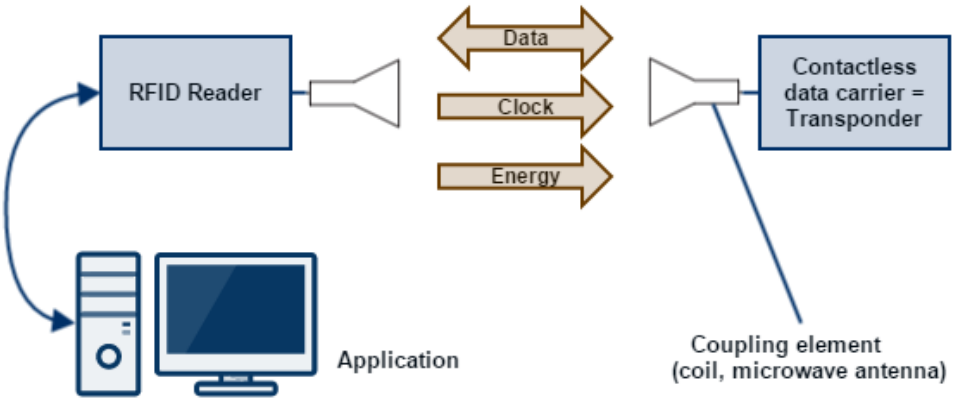


Figura 11 - Leitor RFID e Transponder - principais componentes de todo Sistema RFID[16].

4.2.2. Microcontroladores

Em nosso trabalho, utilizamos microcontroladores Arduino[17] - plataforma de prototipagem eletrônica (Figura 12), opensource, flexível, disponível na versão stand-alone ou para comunicação com software executando em computador pessoal. O Arduino consiste em um microcontrolador que pode ser programado para processar entradas e saídas de componentes externos conectados a ele. Utiliza-se do conceito de computação física ou embarcada para permitir a construção de sistemas que interagem com o ambiente por meio de hardware e software. Por serem baseados em hardware e software fáceis de usar, os arduinos são destinados a qualquer pessoa que queira desenvolver projetos interativos (Doukas, 2012).

Figura 12 - (À esquerda) Arduino Uno R3[18] e (à direita) Arduino Yún[19].

[16] Fonte: Adaptado de Finkenzeller (2010)
[17] https://www.arduino.cc/
[18] Fonte: http://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg
[19] Fonte: http://www.arduino.cc/en/uploads/Main/ArduinoYunFront_2.jpg

4.2.3.
Sensores

A terceira tecnologia utilizada foram os sensores (Figura 13). No contexto da IoT, diversos tipos de sensores podem ser utilizados para coletar dados úteis em aplicações IoT, como a frequência de batimentos cardíacos (Figuras 14 e 15), nível de açúcar no sangue (para controlar pacientes diabéticos), taxa respiratória, temperatura, umidade, nível de líquido, nível de oxigênio, nível de gás carbônico, acelerômetro, entre outros.

Figura 13 – Exemplos de sensores para a área da Saúde[20].

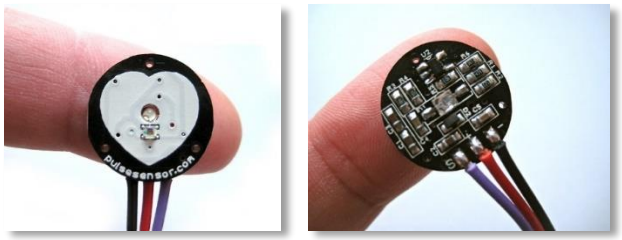


Figura 14 – Pulse Sensor - Sensor de frequência cardíaca[21].

Figura 15 – O Pulse Sensor pode ser utilizado na ponta do dedo (à esquerda) ou na orelha (à direita).

[20]Fonte: http://www.libelium.com/e-health-low-cost-sensors-for-early-detection-of-childhood-disease-inspire-project-hope/

[21] Fonte: http://pulsesensor.com/

5
Frameworks

5.1.
Considerações Gerais

Frameworks são ferramentas usadas para gerar aplicações relacionadas a um domínio específico, isto é, para atacar uma família de problemas relacionados (Markiewicz & de Lucena, 2001). A utilização dos frameworks existentes, bem como o desenvolvimento de novos geradores de aplicações, justifica-se pela habilidade que este tipo de ferramenta possui em prover reuso de design e de código. Este fato permite que se tenha aumento de produtividade e um menor *time-to-market* na atividade de desenvolvimento de software, quando comparado a abordagens tradicionais.

Quando um desenvolvedor deseja construir uma solução de software para resolver um determinado problema de um domínio, ele pode fazer isto desenvolvendo uma única aplicação. No entanto, se acredita que este problema pode evoluir para uma família de problemas relacionados, dentro de um único domínio, onde cada problema pode exigir o desenvolvimento de uma aplicação customizada, ele pode optar por desenvolver um framework. Neste caso, o framework pode fornecer pontos de extensão que podem ser customizados para se adequarem à cada aplicação, de modo que ele seja capaz de gerar aplicações para o domínio inteiro (Markiewicz & de Lucena, 2001).

5.2.
Pontos Flexíveis e Pontos Fixos

Frameworks possuem pontos flexíveis e fixos que são conhecidos como hot spots e frozen spots, respectivamente. Hot spots são pontos de extensão que permitem ao desenvolvedor criar aplicações a partir da instanciação do framework. Para isto, o desenvolvedor irá criar código específico da aplicação para cada hot spot, através da implementação de classes e métodos abstratos definidos no framework. Na implementação dos hot spots, o código da aplicação instanciada irá

informar quais eventos serão responsáveis por sua ativação, via mecanismo de call-back realizado pelo código do framework (Markiewicz & de Lucena, 2001).

Frozen spots, por sua vez, constituem o *kernel* do framework, que são suas partes fixas, previamente implementadas, difíceis de se alterar, que irão chamar um ou mais hot spots da aplicação e estarão presentes em cada instância do framework. Para que um framework possa funcionar, os métodos e classes dos seus hot spots precisam ser implementados por uma aplicação, de modo que estes códigos serão usados pelo *kernel* do framework (Markiewicz & de Lucena, 2001).

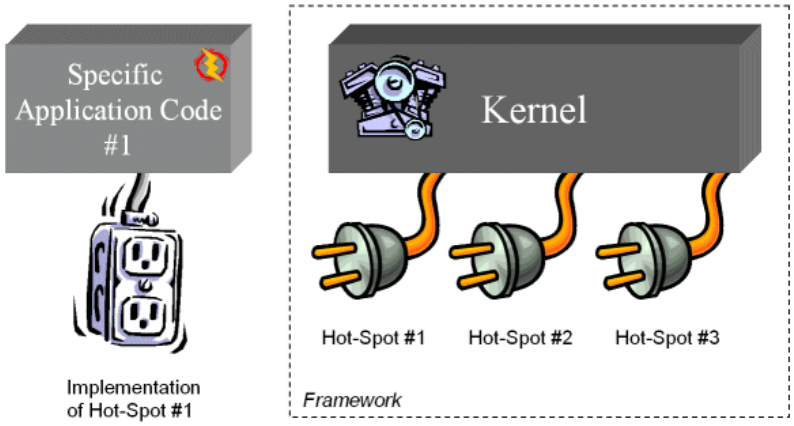


Figura 16 – À esquerda, exemplo de código específico da aplicação, que implementa o Hot-Spot #1. À direita, exemplos de pontos de extensibilidade do *kernel* (Hot-Spot #1, Hot-Spot #2 e Hot-Spot #3).

5.3. Etapas do Desenvolvimento

O desenvolvimento de um framework compreende três etapas principais: 1- Análise do domínio; 2- Projeto do framework; 3- Instanciação do framework.

A etapa de análise consiste em fazer um levantamento dos atuais e possíveis futuros requisitos do domínio. A estratégia para descoberta destes requisitos pode considerar fatores como, por exemplo, experiências pessoais do desenvolvedor ou disseminadas por outros desenvolvedores, análise de padrões e outros sistemas existentes. Além dos requisitos, hot spots e frozen spots são descobertos durante a análise de domínio.

A etapa de projeto do framework encarrega-se de esboçar os hot spots e frozen spots através de uma linguagem de modelagem, que pode ser a UML, por exemplo. Nesta etapa, além de diagramas UML, podem ser usados padrões de projeto para

modelar as abstrações do framework definidas na análise de domínio, mostrando os pontos de extensibilidade e flexibilidade do framework.

A fase de instanciação do framework corresponde à geração de aplicações, através da implementação dos hot spots. Vale lembrar que cada aplicação terá partes em comum que são os frozen spots. Em comparação ao desenvolvimento tradicional de software, onde uma aplicação executável é construída para resolver um problema específico, a fase de instanciação do framework compreende as fases de construção e transição da abordagem de desenvolvimento tradicional.

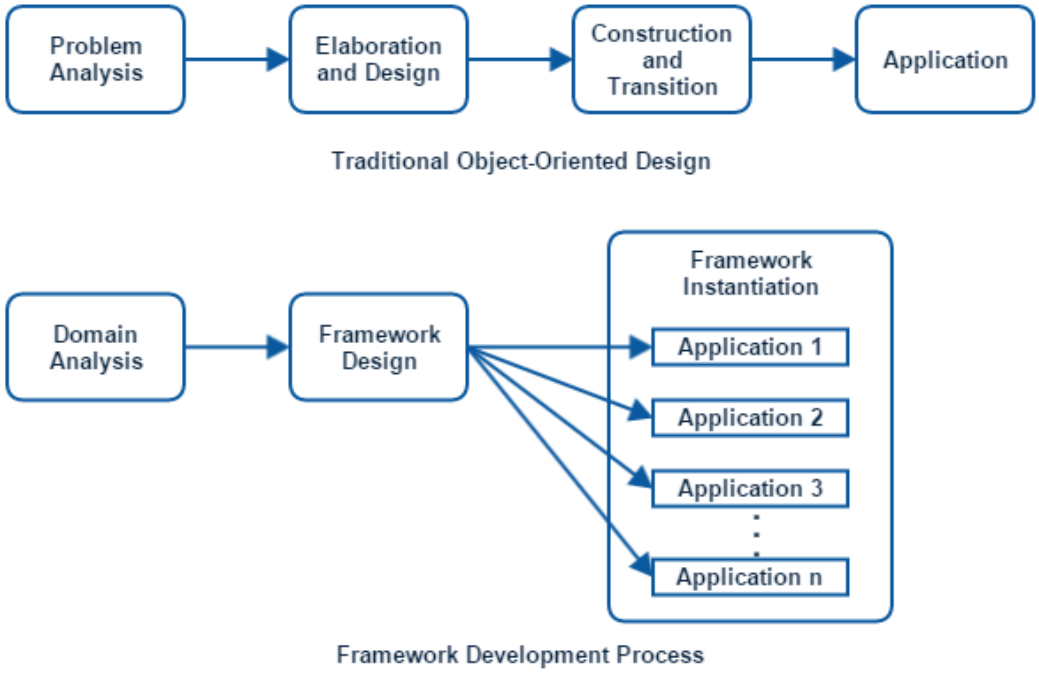


Figura 17 - Comparativo entre a abordagem de Desenvolvimento de Software Tradicional e o Processo de Desenvolvimento de um Framework.

5.4. Custo-benefício do Desenvolvimento

Apesar de promissora, a escolha de desenvolver um framework deve ser ponderada. Acredita-se que frameworks podem ser adequados para muitos domínios, mas não para todos. Ao considerar o desenvolvimento de um framework, devem ser levantadas algumas questões como as que seguem abaixo.

Um framework não é uma aplicação, mas um instanciador de aplicações por customização. Seu custo de desenvolvimento é no mínimo o custo de uma

aplicação, mas normalmente é bem mais caro e complexo. A análise dos requisitos do domínio deve possibilitar a avaliação do custo-benefício de se desenvolver um framework, para decidir se o domínio em questão requer um nível de flexibilidade que justifique seu custo e complexidade de construção.

Caso verifique-se que mais de uma aplicação precisará ser criada neste domínio, deve-se considerar ainda se o custo de desenvolver um gerador de aplicações será menor que o de desenvolver as aplicações customizadas.

Além disso, para que um framework seja utilizado, o desenvolvedor deve considerar, em tempo de desenvolvimento, como construí-lo de modo que não seja tão difícil de ser integrado a outros frameworks. Isto porque a integração entre frameworks é uma realidade inevitável. Caso contrário, estes frameworks tendem a ser abandonados. Entretanto, vale salientar que a integração entre frameworks não é uma tarefa fácil. De acordo com Michael Mattson, os desenvolvedores deparam-se com vários problemas quando realizam integração entre um ou mais frameworks. Ainda segundo Mattson, estes problemas são provenientes de cinco causas principais: comportamento coeso, cobertura de domínio, *design intention*, código fonte inacessível e inexistência de padrões para o framework.

Uma das formas de considerar a composição em tempo de desenvolvimento é manter um conjunto de APIs que encapsula os serviços oferecidos pelo framework. Neste caso, quando uma aplicação precisar fazer uma composição com o framework, ela apenas precisará conhecer sua interface, ou seja, quais funções e parâmetros precisará chamar, sem precisar conhecer detalhes de como estes foram desenvolvidos internamente no framework. Outra opção é criar uma camada de mediação para converter os requisitos dos frameworks envolvidos na composição.

5.5. Classificação

Um framework pode ser classificado de acordo com sua extensibilidade. Ele pode ser usado como uma caixa branca, uma caixa preta ou ainda como uma caixa cinza. As três abordagens diferem entre si com relação à forma como o framework será instanciado e possuem uma alta curva de aprendizado.

Um framework caixa branca também é conhecido como framework orientado à arquitetura. Seu processo de instanciação requer um bom entendimento

de seu funcionamento e só é possível através da criação de novas classes, por herança ou composição.

Frameworks caixa preta também são conhecidos como frameworks dirigidos a dados e são fáceis de usar. Eles produzem instâncias usando scripts de configuração, onde uma ferramenta gráfica pode ser utilizada para automatizar o processo de instanciação. Para estendê-lo, os usuários não precisam conhecer seus detalhes internos de implementação.

Por fim, frameworks caixa cinza são aqueles que possuem características de ambos os frameworks, caixa branca e caixa preta.

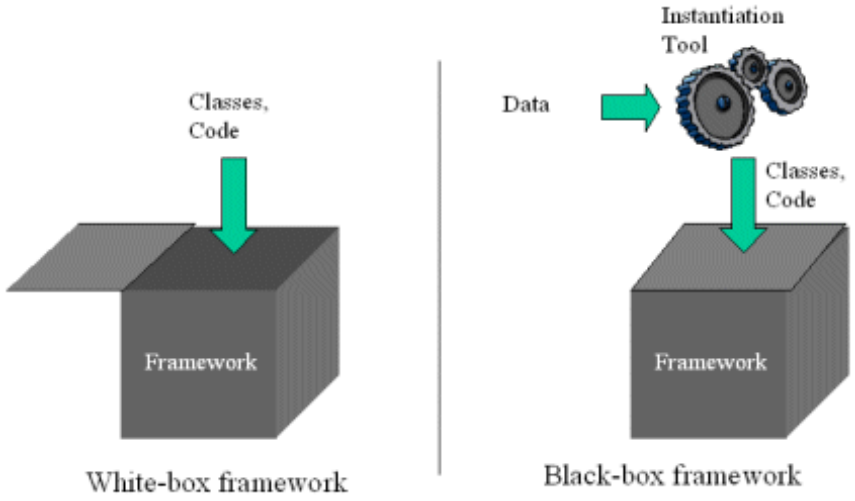


Figura 18 – Framework Caixa Branca e Framework Caixa Preta.

5.6. Questões sobre Frameworks

A habilidade de criação de sistemas executáveis a partir de frameworks irá depender da usabilidade de seu mecanismo de instanciação e da sua documentação disponível. Para que sejam utilizados, frameworks precisam oferecer guias que expliquem como estendê-lo e/ou ferramentas de instanciação.

Muitas orientações para documentação de frameworks foram propostas, como por exemplo: *hot spot cards*, que focam nos pontos flexíveis do framework; criação de *cookbooks* que discutem como o framework deve ser implementado e os passos requeridos, além de receitas contendo um passo-a-passo de como resolver problemas específicos. Outra abordagem consiste no mapeamento de soluções arquiteturais usadas durante o design do framework.

Apesar de não existirem padrões para documentação de frameworks, recomenda-se que sejam utilizados padrões de projeto em suas etapas de modelagem e projeto, de modo que, em sua documentação, o uso destes padrões seja ilustrado em diagramas UML, para facilitar a compreensão do seu processo de instanciação.

Na documentação do IoT4Health, utilizamos o padrão de projeto *Strategy* (Gamma, Helm, Johnson, & Vlissides, 1995) para definir pontos de extensibilidade do framework. A subseção 7.3 apresenta a relação dos hot spots previstos durante a etapa de projeto do IoT4Health.

Para exemplificar o uso de padrão de projeto que poderia ser adequado para nossa aplicação, a Figura 19 apresenta a utilização do padrão Strategy para modelar como podemos alternar entre diversos algoritmos usados para definir o comportamento de agentes cognitivos. Neste caso, um ponto de extensibilidade (hot spot) poderia permitir que diferentes algoritmos pudessem ser utilizados (plugados) no framework sem alterar o código escrito previamente.

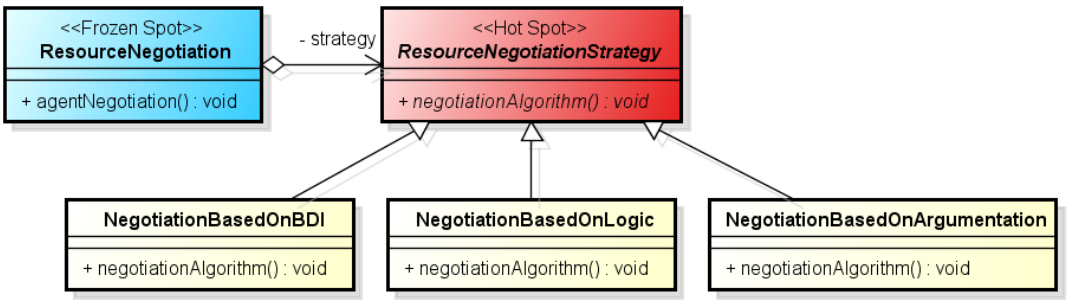


Figura 19 - Exemplo de uso do padrão de projeto Strategy para modelar o comportamento de um agente cognitivo.

Como a análise de domínio deve cobrir uma classe inteira de problemas, um dos desafios envolvidos nesta etapa é encontrar indivíduos cuja experiência cubra todo o domínio. Na fase de análise, também podem ser utilizados protótipos e sistemas de software similares.

Durante a construção de um framework, o domínio deve ser cuidadosamente escolhido e a delimitação do escopo deve considerar as restrições de desenvolvimento com relação ao tempo, custo e complexidade, bem como fatores como a experiência dos *stakeholders* envolvidos. Se o domínio é muito amplo, leva-se muito tempo para reunir e avaliar informações e recursos relevantes relacionados

a ele. Por outro lado, se o domínio é muito restrito, a aplicabilidade do framework será reduzida e as aplicações geradas a partir dele serão muito semelhantes para justificar o desenvolvimento de um framework.

Frameworks podem precisar evoluir em termos de projeto ou implementação, compreendendo desde evoluções simples, como pequenas atualizações, a alterações de maior impacto, envolvendo seu completo *redesign*. Isto significa que podem ser necessárias mudanças em sua arquitetura ou em seus requisitos. Aplicações geradas pelo framework também devem evoluir de modo a manter sua compatibilidade com o framework e continuar em operação.

Antes de utilizar um framework, é importante observar algumas de suas características, como projeto, documentação oferecida, requisitos suportados e existência de uma interface pública com os serviços que ele oferece. Esta observação pode ajudar na escolha de qual framework usar, considerando a sua probabilidade de manter-se em funcionamento e sua capacidade de evolução.

Como frameworks são geradores de aplicações que cobrem um domínio inteiro, ao invés de problemas específicos, eles são mais flexíveis que sistemas de software tradicionais. Sua flexibilidade deve-se à sua capacidade de ser estendido, através de seus hot spots. A extensibilidade do framework é conseguida através de conceitos da orientação a objeto, tais como herança e ligação dinâmica.

A análise do quão flexível um framework precisa ser é um ponto extremamente importante no projeto do framework, visto que sua flexibilidade tem impacto em sua performance e complexidade de desenvolvimento e extensão. Muitas vezes, os desenvolvedores adicionam muitos pontos de extensão ao framework, acreditando, equivocadamente, que ferramentas mais flexíveis são necessariamente mais poderosas. No entanto, esta abordagem irá gerar complexidade desnecessária e problemas de performance, sem representar ganhos em termos de funcionalidades.

Deste modo, durante a etapa de projeto do framework, devem ser analisados cuidadosamente quais hot spots precisam estar presentes, para que seu uso abusivo não resulte na criação de um framework demasiadamente genérico, complexo e com performance sofrível.

Frameworks podem gerar aplicações em que o código dos frozen spots esteja misturado ao código dos frozen spots. Este fato faz com que a atividade de debug ultrapasse o código da aplicação e alcance também o código do framework. Este é

um sinal de que o framework não está bem projetado, uma vez que as chamadas do código dos hot spots devem estar separadas das chamadas de código dos frozen spots. Além disso, a mistura de código entre hot spots e frozen spots acaba introduzindo complexidade ao framework. Um emaranhado de código dificulta sua instanciação, uma vez que desenvolvedores devem conhecer cuidadosamente o código para evitar modificar o código dos frozen spots, que não deve ser alterado.

Frameworks são recomendados para construção de aplicações cujos requisitos mudam rapidamente. Eles podem ser construídos a partir de uma metodologia de desenvolvimento incremental, introduzindo hot spots mais simples em um primeiro momento e incrementando-o em versões posteriores.

Considerando o desenvolvimento de frameworks, algumas questões ainda estão em aberto, tais como o fato de não existir uma padronização para a documentação de frameworks, bem como ferramentas para analisar o seu custo de desenvolvimento, em relação ao custo de desenvolvimento de uma aplicação.

Ao utilizar um framework, é importante que o desenvolvedor leia sua documentação e procure um guia de como instanciá-lo, além de observar a quantidade de esforço envolvida na geração de aplicações a partir dele (Markiewicz & de Lucena, 2001).

6
Metodologia

Para explorar melhor o domínio de E-Health e responder as questões de pesquisa apresentadas na próxima subseção, realizamos dois estudos de caso que abordam os subdomínios de *Monitoring* e *Smart Environment*. A realização desta etapa resultou no desenvolvimento de duas aplicações de E-Health - **Agents4Health** e **RPM** -, que foram essenciais para a definição do *kernel* do nosso framework IoT4Health, tornando sua construção possível através de uma estratégia de desenvolvimento *bottom up*. Desta forma, a partir da identificação das funcionalidades comuns às duas aplicações, juntamente com a incorporação de outras funcionalidades presentes em cada uma das aplicações que julgamos essenciais para a cobertura do escopo do domínio que pretendíamos atingir, projetamos o kernel do IoT4Health, juntamente com seus pontos de flexibilização. A conclusão desta etapa resultou na análise de requisitos, modelagem, projeto, implementação e modelagem do framework. Uma vez concluída a implementação do IoT4Health, criamos uma terceira instância para servir também como prova de conceito – o **E-Health System**.

6.1.
Questões de Pesquisa

Para nortear nossas investigações, elencamos uma questão essencial e três subquestões a serem discutidas neste trabalho.

Questão Central: É viável a concepção de um framework para instanciação de aplicações IoT no domínio de E-Health para acompanhamento remoto de pacientes, utilizando-se agentes de software?

Subquestão 1: Quais os benefícios da coleta automática de dados vitais de pacientes, realizada através de sensores?

Subquestão 2: Quais tipos de anomalias podem ser detectadas automaticamente a partir de aplicações criadas como instâncias de nosso framework?

Subquestão 3: Quais benefícios o monitoramento remoto de dados dos pacientes pode trazer?

6.2. Objetivos

Nosso principal objetivo é o desenvolvimento de um framework para construção de sistemas no domínio de E-Health, baseados em agentes de software. À priori, no processo de instanciação do framework, temos dado enfoque a aplicações que destacam a atividade de monitoramento de pacientes em tempo real.

Optamos pelo monitoramento remoto, de modo que não exista dependência entre a localização física dos pacientes monitorados e os interessados em seu monitoramento, tendo em vista os benefícios que esta abordagem pode trazer.

6.3. Hipóteses

Antes de realizar nossos estudos de caso, formulamos três hipóteses com base em nosso conhecimento prévio sobre o assunto, a serem confrontadas com os dados obtidos neste estudo, para discutir suas aplicabilidades.

H1. Pode-se coletar dados de pacientes automaticamente, utilizando aplicações IoT instanciadas a partir de nosso framework.

H2. É possível detectar comportamentos indesejáveis (anomalias) nos pacientes em tempo real, através de aplicações IoT desenvolvidas por meio do IoT4Health.

H3. O monitoramento de pacientes pode ser feito de modo remoto, através do uso de aplicações criadas como instâncias do IoT4Health.

6.4. Busca por Soluções Reutilizáveis

Em termos de recursos tecnológicos, atualmente dispomos de uma grande oferta de hardware e software para construção de aplicações IoT, a preços acessíveis e com complexidade de uso razoável. A depender do tempo e orçamento disponíveis para implementar uma solução, é possível definir estratégias como, por exemplo, reutilizar componentes disponíveis no mercado, consumir serviços mais

genéricos ou até mesmo optar por desenvolver inteiramente a solução, endereçando necessidades reais e próprias, com uma quantidade moderada de esforço.

As etapas de elicitação de requisitos e modelagem das aplicações IoT que desejávamos implementar, foram realizadas em paralelo à atividade de investigação de trabalhos relacionados na literatura, para nos dar uma visão geral de como o problema estava sendo abordado pela academia. Nosso foco principal foi descobrir quais trabalhos representavam o estado da arte em pesquisas neste campo.

Após a definição dos requisitos e conclusão da modelagem, partimos para o projeto de arquitetura, onde decidimos como deveria estar estruturado o nosso framework. O resultado desta etapa foi o projeto de uma arquitetura dividida em três camadas [L1-L3]: L1- Camada de Distribuição de Dados; L2-Camada de Comunicação de Dados; L3-Camada de Gerenciamento de Dados;

A partir da visão macro do projeto da arquitetura, iniciamos a atividade de análise das ferramentas disponíveis no mercado para investigar quais seriam capazes de atender nossos requisitos.

A conclusão desta análise resultou na seguinte situação: para as camadas L1 e L2, encontramos no mercado plataformas que proviam serviços capazes de atender nossas necessidades iniciais; ao passo que a camada de gerenciamento de dados, L3, precisaria ser inteiramente desenvolvida. Apresentaremos mais detalhes sobre a arquitetura das aplicações e ferramentas utilizadas na subseção 7.3.

6.5. Planejamento do Estudo de Caso 1: Agents4Health

6.5.1. Definição do Problema

Muitos são os desafios evidenciados pela área da Saúde atualmente, visto que se trata de um setor onde os profissionais lidam com questões críticas como o estado de saúde dos pacientes e com os riscos envolvidos em seus tratamentos médicos, desde exames de rotina a procedimentos cirúrgicos de mais alta gravidade.

Além disso, questões éticas muitas vezes precisam ser resolvidas antes mesmo de dar início a qualquer método de pesquisa científica que tenha participação direta dos pacientes. Questões como esta devem ser consideradas na

etapa de planejamento do método de pesquisa, para que seja viabilizada a aplicação do experimento no ambiente real.

De modo mais específico, a motivação para realizar este estudo de caso veio de problemas como a predominância da reatividade, em detrimento da proatividade, no ambiente em que o paciente está sendo tratado. O que significa, por exemplo, que, em muitos casos, medidas são tomadas pela equipe médica somente depois que o paciente já teve o seu quadro piorado de alguma forma.

Fatos como este são bastante problemáticos, principalmente porque, muitas vezes, o intervalo de tempo decorrido entre o momento em que o paciente teve complicações em seu estado de saúde e o momento em que este fato foi detectado pelos profissionais de saúde, que aqui estamos chamando de **Detection Anomaly Interval** (DAI), é alto. (Fernandes & Lucena, 2015b)

Considerando este contexto, uma anomalia deve ser interpretada como qualquer anormalidade perceptível no estado de saúde do paciente, da qual decorra a necessidade de intervenção pelo profissional da saúde responsável pelos seus cuidados, no instante em que ela ocorre. (Fernandes & Lucena, 2015a)

Algumas vezes, o grande lapso de tempo que se verifica no processo de detecção de anomalias pode ter consequências desastrosas, podendo resultar em sequelas ou, no pior dos casos, até mesmo no óbito do paciente.

Um dos fatores que contribuem para que o DAI tenha uma forte tendência em apresentar um alto valor é o fato de que a inspeção no estado de saúde dos pacientes ainda é comumente realizada de modo presencial. Desta forma, o profissional da saúde precisa se deslocar até o local em que se encontra o paciente para inspecioná-lo, coletando seus dados, através da medição de seu pulso, temperatura, dentre outros dados que podem ser verificados.

6.5.2. Objetivos

De modo mais geral, desejava-se tornar mais proativa a atividade de monitoramento de pacientes. Também pretendia-se aumentar a capacidade de interação entre as coisas do ambiente, as quais serão apresentadas na subseção 7.5.1.2. Com relação aos objetivos específicos, tivemos, por exemplo, a pretensão de tentar diminuir o DAI, em nosso ambiente simulado de monitoramento.

6.5.3.
Solução Proposta

A solução planejada neste estudo de caso consistiu no desenvolvimento de um sistema multi-agentes para monitoramento remoto de pacientes em tempo real, que simulou a ocorrência de problemas da ordem dos que foram mencionados anteriormente, juntamente com ações que poderiam ser tomadas nestes casos.

6.6.
Planejamento do Estudo de Caso 2: Remote Patient Monitoring (RPM)

6.6.1.Definição do Problema

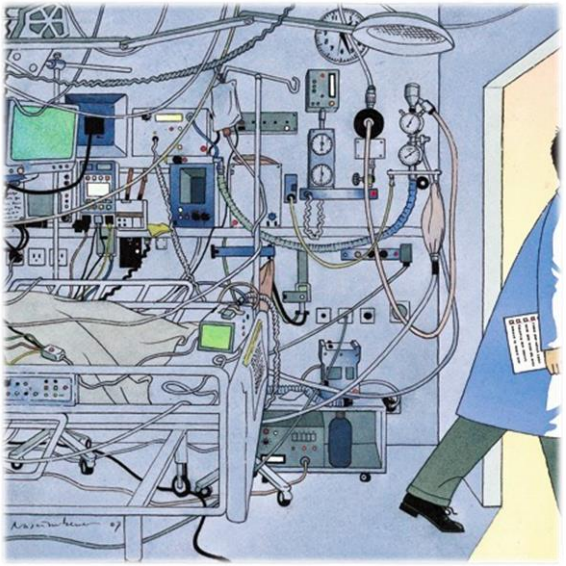


Figura 20 – Processo de coleta de dados do paciente em uma UTI[22].

A realização de pesquisa científica no contexto de healthcare é uma tarefa complexa. Ela envolve lidar com questões delicadas como os riscos associados ao tratamento de paciente e incluem desafios de diversas naturezas. Barreiras para conduzir pesquisas nesta área não estão limitadas à aspectos estritamente técnicos. O envolvimento direto de pacientes em qualquer pesquisa científica demanda a definição de protocolos de ética. Além disso, equipe médica e pacientes podem estar envolvidos nas etapas de análise de requisitos e projeto durante o processo de desenvolvimento de software, uma vez que sua experiência e perspectiva podem

[22] Fonte: http://www.newyorker.com/magazine/2007/12/10/the-checklist

contribuir para o planejamento do experimento. Entretanto, estas partes interessadas no projeto podem ter pouco tempo disponível para colaborar com a pesquisa.

Observa-se também que, em muitos casos, o monitoramento de pacientes é feito de modo reativo, o que significa que a equipe médica apenas toma ações após detectar um declínio no estado de saúde do paciente. Outros fatores limitadores dizem respeito aos processos manuais de identificação de pacientes, bem como de coleta e monitoramento de dados vitais dos pacientes. Podemos citar também o uso limitado de soluções tecnológicas que facilitem a rotina de trabalho da equipe médica, como aplicações para distribuição de informações entre os profissionais envolvidos no tratamento de pacientes.

6.6.2. Objetivos

Definimos os seguintes objetivos para guiar o desenvolvimento da nossa aplicação, visando:

1- Minimizar o DAI;
2- Automatizar o processo de coleta de dados vitais dos pacientes;
3- Permitir o monitoramento remoto de dados dos pacientes;
4- Compartilhar informações relacionadas à saúde e ao tratamento dos pacientes com a equipe médica e colaboradores externos;
5- Permitir a visualização de dados dos pacientes a partir de aplicativos móveis e de forma mais amigável, através de *line charts*;
6- Equipar o corpo médico com uma ferramenta de suporte à análise de dados e ao diagnóstico colaborativo;

Mais detalhes sobre as aplicações Agents4Health e RPM System serão apresentados no próximo capítulo (Framework IoT4Health).

6.6.3. Solução Proposta

Para lidar com os problemas citados acima, desenvolvemos o RPM System, que suporta detecção automática de anomalias e distribuição de dados dos pacientes em plataforma baseadas em nuvem. Ele compreende ainda um aplicativo *mobile* que provê acesso aos dados dos pacientes armazenados remotamente e permite sua visualização em tempo real (Fernandes, Lucena, Lucena, & Azevedo, 2015).

7
Framework IoT4Health

Neste capítulo, apresentamos em detalhes o framework IoT4Health, no que diz respeito à análise de domínio, caracterização do domínio da aplicação, projeto, arquitetura, definição dos frozen spots e hot spots, implementação, documentação e provas de conceito.

7.1.
Análise de domínio

Nas etapas iniciais da nossa pesquisa, realizamos uma análise do domínio de e-Health. Primordialmente, fizemos um levantamento de alguns dos problemas enfrentados na rotina dos profissionais de saúde atualmente, conforme descrevemos anteriormente. Esta fase prosseguiu com o estudo e implementação de ferramentas que pudessem ajudar a resolver tais problemas e apoiar o trabalho da equipe médica em atividades de acompanhamento remoto de pacientes em tempo real.

Durante a etapa de análise, optamos por desenvolver um framework como ferramenta para a criação de soluções para os problemas encontrados, por acreditar que o custo-benefício de desenvolver uma ferramenta mais genérica e complexa que um sistema de software seria recompensado pelo fato de que não estávamos lidando com um problema específico, mas com uma família de problemas relacionados, conforme evidenciado ao final da etapa de análise do domínio.

7.2.
Caracterização do domínio da aplicação: Acompanhamento, Monitoramento e Visualização

Considerando o contexto de healthcare, o principal foco do IoT4Health é dar suporte a construção de aplicações que se proponham a realizar acompanhamento de pacientes, permitindo o contínuo monitoramento de seu estado de saúde e a visualização de seus sinais vitais em tempo real.

Aplicações instanciadas a partir do IoT4Health serão capazes de realizar acompanhamento do paciente de modo remoto. Para isto, o paciente deverá utilizar um dispositivo composto por micro controladores e sensores que irá coletar seus dados vitais e enviar para uma base de dados remota, através de conexão wireless ou Ethernet.

Uma vez armazenados na nuvem, os dados dos pacientes poderão ser visualizados pelos profissionais da saúde responsáveis por seus cuidados, desde que estes tenham sido previamente autorizados a acessá-los. Neste caso, os interessados poderão visualizar os sinais vitais dos pacientes em tempo real a partir de dispositivos como laptops, tablets ou smartphones.

Embora a equipe médica tenha a possibilidade de visualizar os dados dos pacientes de qualquer lugar e a qualquer momento, o monitoramento do estado de saúde dos pacientes é realizado de modo autônomo, por meio de agentes de software. Para suportar esta funcionalidade, o sistema possui uma etapa inicial de configuração, que consiste basicamente do preenchimento de um formulário por parte de um especialista no domínio, onde este irá especificar valores de referência para cada tipo de sinal vital coletado do paciente. Deste modo, a tarefa dos agentes de software será monitorar continuamente se os sinais vitais dos pacientes estão dentro dos valores esperados. Caso os agentes percebam que os sinais vitais estão fora dos limites normais, estes são encarregados de enviar alertas para os profissionais da saúde responsáveis por agir em estados críticos, através de envio de mensagens SMS ou de e-mail. Tais profissionais devem ter sido previamente cadastrados no sistema, na etapa de configuração.

É importante ressaltar as vantagens destas abordagens de acompanhamento, visualização e monitoramento suportadas pelo IoT4Health. Em primeiro lugar, o fato de o médico poder acompanhar remotamente o estado de saúde do paciente traz uma série de benefícios tanto para o médico quanto para o paciente, tais como: o médico não fica limitado a acompanhar seu paciente apenas quando estiverem fisicamente juntos; é possível que o médico trabalhe com outros profissionais de forma colaborativa na discussão de um diagnóstico; para o paciente, o ônus de ter que se deslocar até a unidade de atendimento médico é reduzido.

A visualização dos sinais vitais dos pacientes, que estão sendo coletados automaticamente por sensores, em tempo real, também tem benefícios, como: anormalidades em seu estado de saúde podem ser detectadas instantaneamente; a

coleta feita por sensores diminui o custo que se teria em alocar um profissional para fazer a coleta dos sinais vitais manualmente, além do custo de se alocar um leito na unidade de atendimento médico para realizar este procedimento de forma presencial.

Por último, a autonomia no monitoramento também tem como principal vantagem a economia de recursos, pois libera o médico e sua equipe de apoio de realizar a tarefa monótona e exaustiva de monitorar constantemente o status de seus pacientes. Além disto, o monitoramento realizado de modo automatizado é menos propenso a erros que o resultante da avaliação humana. Nesse sentido, caso alguma anomalia passasse despercebida pelo olho humano, este fato poderia influenciar negativamente o tratamento do paciente, visto que o médico poderia dar um diagnóstico equivocado em virtude de ter pedido alguma informação relevante.

7.3. Arquitetura do IoT4Health – Divisão em três camadas

A arquitetura do IoT4Health está estruturada em três camadas que possuem funcionalidades bem definidas (Figura 21).

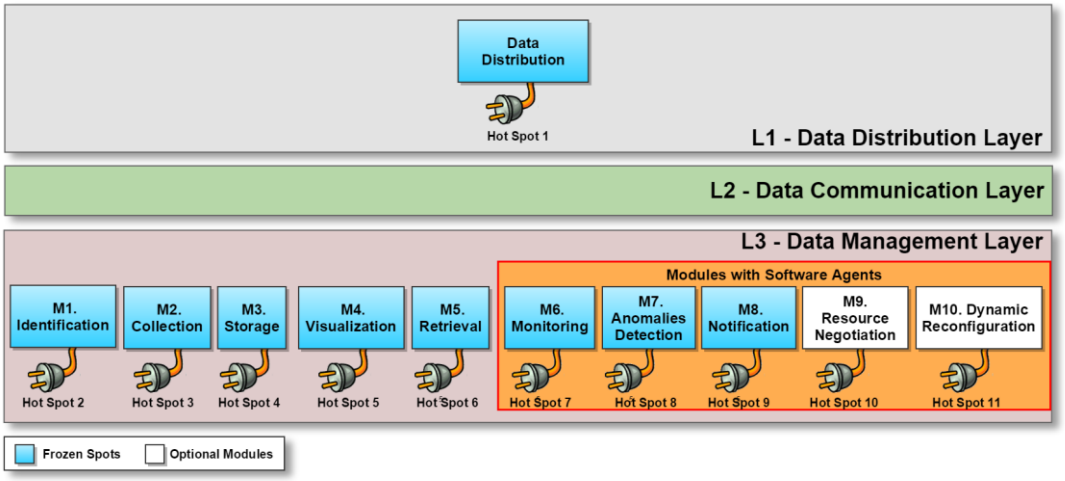


Figura 21 - A arquitetura do Agentes4Health com suas três camadas (L1-L3). A Camada de Gerenciamento de Dados (L3) interage com a Camada de Distribuição de Dados (L1) através da API REST (L2).

7.3.1.
L1- Camada de Distribuição de Dados

Ao ser implementada pelas instâncias do framework, a camada de distribuição de dados irá funcionar como um repositório remoto dos dados vitais dos pacientes. Assim, nela poderão ser armazenados diferentes tipos de dados, como temperatura do paciente, batimento cardíaco, dentre outros.

7.3.2.
L2- Camada de Comunicação de Dados

Permite a interação entre as camadas L1 e L2, através de API REST. Por meio desta camada, uma aplicação IoT gerada a partir do IoT4Health poderá se comunicar com a plataforma de armazenamento remoto de dados.

7.3.3.
L3- Camada de Gerenciamento de Dados

A camada L3 é responsável pelo gerenciamento dos dados envolvidos em uma aplicação IoT. Ela irá realizar todo o processamento de informações das aplicações instanciadas, compreendendo desde os processos de identificação de pacientes, coleta, armazenamento, recuperação, visualização e monitoramento dos seus dados, até tarefas de detecção de anomalias, envio de notificações, negociação por recursos e reconfiguração dinâmica das aplicações. Atualmente, esta camada é composta por 10 módulos com responsabilidades bem definidas.

A seguir, apresentamos uma breve explicação sobre os módulos que compõem as aplicações IoT criadas como instâncias do IoT4Heatlh, juntamente com exemplos de como a arquitetura do framework pode ser estendida, considerando cada módulo individualmente.

7.3.3.1.
Módulo 1 (M1) - Identificação

Aplicações IoT geradas a partir do framework IoT4Health devem implementar o módulo M1 para realizar o processo de identificação de pacientes. A arquitetura do IoT4Health prevê a possibilidade de customização deste processo de identificação, onde o desenvolvedor pode utilizar diferentes estratégias, como,

por exemplo: 1- Uso de código de identificação único, como Cpf do paciente, por exemplo; 2- Utilização de tags RFID, que podem estar presentes em objetos como braceletes, cartões ou outros elementos de identificação; 3- Uso de biometria.

7.3.3.2. Módulo 2 (M2) - Coleta

O módulo de coleta prevê a possibilidade de coletar dados dos pacientes e de seu ambiente. Podem ser realizadas coletas de dados vitais dos pacientes, tais como frequência cardíaca, pressão arterial, temperatura, dentre outros. Além disso, informações relativas ao ambiente no qual o paciente está inserido também podem ser relevantes, como luz, nível de ruído, umidade etc.

O processo de coleta pode ser realizado de duas formas: 1- Manual; 2- Automática. Na coleta manual, a aplicação pode ser implementada de modo a prever que os dados sobre o paciente ou seu ambiente sejam informados por um usuário da aplicação, através de texto ou comandos de voz.

Na coleta automática, a aplicação pode ser implementada de modo a suportar o uso de protótipos de hardware, compostos por microcontroladores e sensores para coletar dados dos pacientes e de seu ambiente automaticamente.

7.3.3.3. Módulo 3 (M3) - Armazenamento

Através da implementação deste módulo, a aplicação pode armazenar os dados vitais coletados. Exemplos de estratégias de armazenamento que podem ser desenvolvidas como extensão da arquitetura do IoT4Health são: 1- Armazenamento local; 2-Armazenamento cacheado; 3- Armazenamento remoto.

O armazenamento local deverá ser escolhido quando não se tem acesso à Internet no ambiente em que a aplicação será executada ou quando o desenvolvedor não tem interesse em distribuir os dados da sua aplicação.

O armazenamento cacheado, por sua vez, é utilizado quando o desenvolvedor tem a intenção de distribuir os dados da aplicação, mas enfrenta problemas de conectividade. Neste caso, os dados serão armazenados localmente, sendo enviados à plataforma em nuvem quando a conectividade for reestabelecida.

Por fim, o armazenamento remoto é usado quando se deseja manter um repositório dos dados na nuvem. O envio destes dados pode ser realizado através de conexão via Ethernet ou Wireless.

7.3.3.4. Módulo 4 (M4) - Visualização

Tanto no armazenamento local quanto no remoto, os dados dos pacientes serão salvos em tabelas. Para proporcionar ao usuário uma forma mais amigável de visualizar estes dados, o projeto do IoT4Health previu que podem ser implementadas algumas estratégias de visualização, utilizando: 1- Sistema web; 2- Sistema web com interface acessível; 3- Aplicativo mobile.

As três estratégias que podem ser implementadas para visualizar os dados operam fazendo requisições ao repositório remoto e exibindo os dados recuperados por meio de uma interface gráfica amigável. Todas as abordagens podem plotar os dados em forma de gráficos do tipo *line charts*, por exemplo, que são atualizados em tempo real pelos agentes de software.

Caso o desenvolvedor opte por implementar este módulo desenvolvendo um sistema web com interface acessível, ele poderá tornar a aplicação acessível a usuários com deficiências visuais, por exemplo.

7.3.3.5. Módulo 5 (M5) Recuperação

Este módulo é responsável por recuperar os dados dos pacientes armazenados na plataforma baseada em nuvem.

7.3.3.6. Módulo 6 (M6) - Monitoramento

Este módulo monitora continuamente os dados coletados do paciente através de agentes que devem reagir em caso de detecção de anomalias.

7.3.3.7.
Módulo 7 (M7) – Detecção de Anomalias

O sistema possui uma etapa obrigatória de configuração para cada paciente, que funciona da seguinte forma: um usuário administrador define o **Desired Value Range** (DVR) para cada sensor.

Estamos chamando os valores que estão fora do DVR de **Anomalous Values** (AV). Os valores correspondentes aos limites mínimo e máximo do DVR são associados a anomalias. Cada uma destas anomalias recebe um rótulo representativo dentro do contexto de healthcare, de modo que faça sentido para um especialista do domínio. O objetivo é habilitar o profissional a identificar rapidamente qual problema está ocorrendo quando a anomalia for detectada pelo sistema. Além disso, o administrador do sistema deve indicar um profissional para tratar cada anomalia, o qual deverá ser alertado no momento da sua ocorrência, através de envio de notificações.

Agentes de software reativos suportam estes mecanismos de monitoramento e detecção de anomalias. Neste caso, os agentes vão avaliar se os dados relativos a cada sensor estão dentro do padrão de normalidade definido, monitorando os dados em busca de anomalias e enviando alertas aos profissionais caso sejam detectadas. Pode-se observar que esta tarefa ajuda a tornar o ambiente mais proativo, uma vez que o sistema alerta a anomalia em tempo real e de forma autônoma, de modo que os profissionais responsáveis por responder aos eventos sejam capazes de tomar providências de forma imediata.

7.3.3.8.
Módulo 8 (M8) – Notificação

O módulo de notificação do IoT4Health oferece a possibilidade de se utilizar diferentes estratégias para envio de alertas à equipe médica, como: 1- Envio de alertas através de mensagens SMS; Notificação via mensagem de e-mail; Alertas através de mensagens de aúdio;

7.3.3.9.
Módulo 9 (M9) - Negociação por Recursos

O módulo de negociação por recursos utiliza o conceito de agentes cognitivos que, neste contexto, seriam responsáveis pelo uso de técnicas de argumentação para atuarem de forma colaborativa no compartilhamento de recursos, tornando sua gestão mais eficaz. Dessa forma, uma aplicação poderia ampliar a arquitetura do IoT4Health, através da implementação de comportamentos para agentes cognitivos. Neste caso, o agente poderia ser responsável por adotar estratégias de negociação para conseguir recursos hospitalares para um dado paciente. Esta funcionalidade poderia ser modelada de modo que, à cada paciente fosse associado um agente que seria responsável por representá-lo no processo de negociação. Este agente teria acesso às informações referentes aos sinais vitais que estão sendo coletados e, com isso, poderia usar estes dados para gerar conhecimento sobre o estado de saúde do paciente. De posse desse conhecimento, o agente seria capaz de atuar em benefício do paciente que ele representa e requerer acesso a um dado recurso, como uma medicação ou a visita de um profissional da saúde, à medida que este se fizesse necessário.

7.3.3.10.
Módulo 10 (M10) - Reconfiguração Dinâmica

O módulo de coleta de dados foi projetado de modo que uma aplicação que o estenda possa fazê-lo tornando-a sensível ao contexto, ou seja, o sistema pode ser reconfigurado dinamicamente, em resposta a mudanças no ambiente. A mudança de um paciente para outro ambiente pode fazer com que os valores definidos como anômalos tornem-se inadequados para seu novo contexto. Nesse caso, a aplicação irá solicitar a reconfiguração destes valores. Esta reconfiguração pode ser feita pelo usuário, através da interface da aplicação ou de forma autônoma pelos agentes cognitivos. Neste último caso, os agentes irão trocar informações entre si, de modo que novos agentes possam reconfigurar seu ambiente fazendo consultas aos outros agentes que já conhecem os valores de DVR e AV para o novo ambiente.

7.4. Projeto do Framework

Subsequentemente à análise, partimos para o projeto do IoT4Health, com a definição da arquitetura que iríamos utilizar, juntamente com a modelagem de seus requisitos e confecção da sua documentação.

Além da arquitetura geral do framework, definimos seus pontos fixos e flexíveis, que serão apresentados nas subseções a seguir. São relacionados alguns dos serviços projetados e que foram modelados a partir de diagramas UML, com a utilização de padrões de projeto.

7.4.1. Frozen Spots e Hot Spots

O projeto inicial do IoT4Health apresenta um total de onze hot spots. Tais pontos de flexibilidade oferecem ao desenvolvedor que irá instanciar o framework a oportunidade de criar suas aplicações de forma customizada.

Cada um dos módulos possui pontos de extensão que ampliam a arquitetura do framework, conforme apresentados nas subseções seguintes.

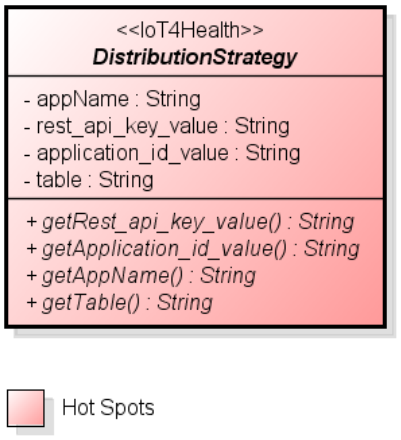


Figura 22 – Diagrama de Classes do Módulo de Distribuição (M0) do IoT4Health.

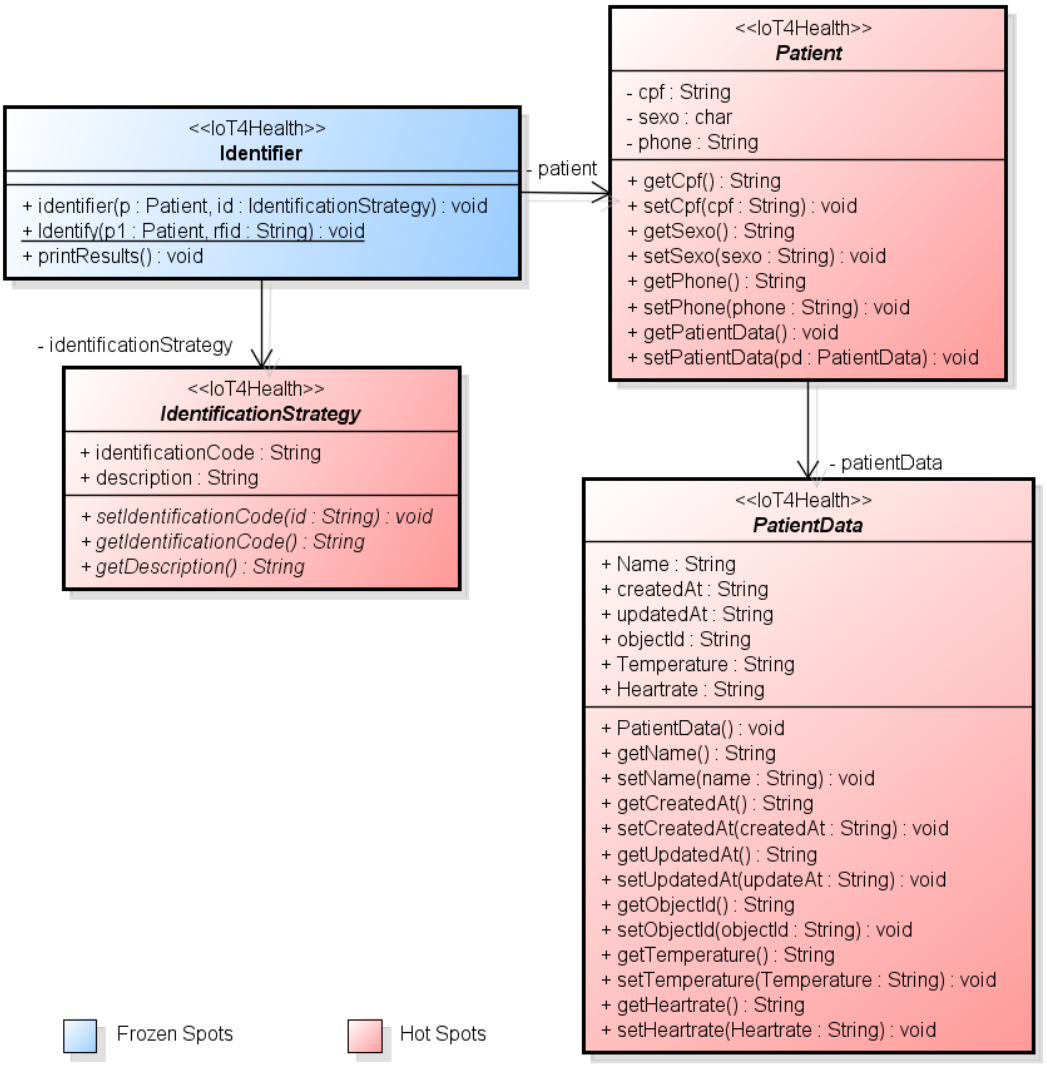


Figura 23 – Diagrama de Classes do Módulo de Identificação (M1) do IoT4Health.

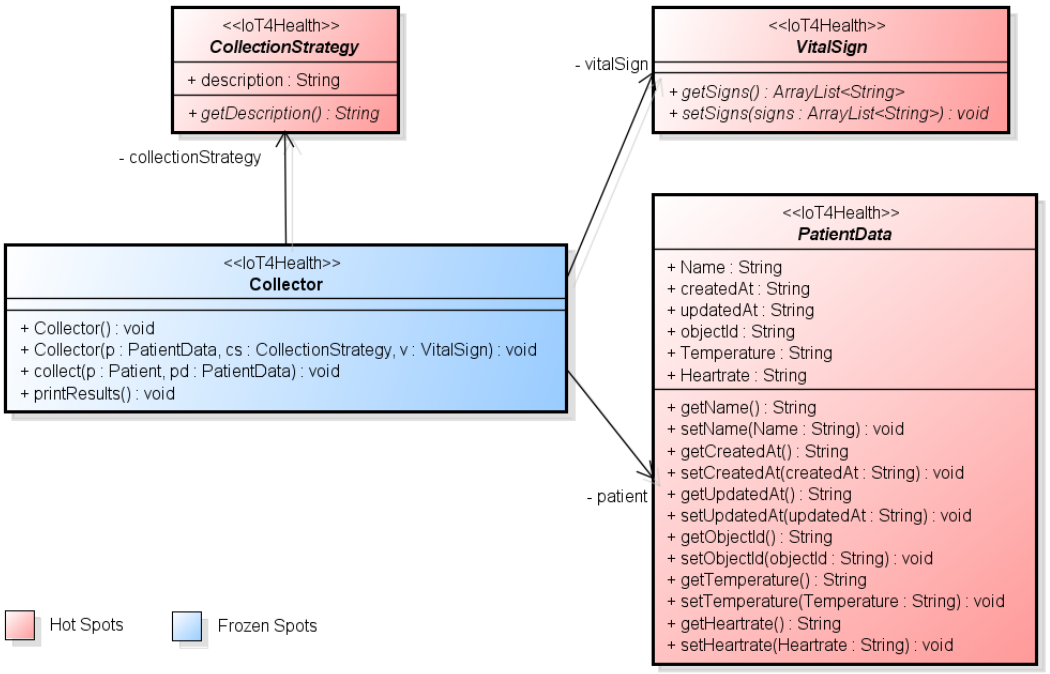


Figura 24 – Diagrama de Classes do Módulo de Coleta (M2) do IoT4Health.

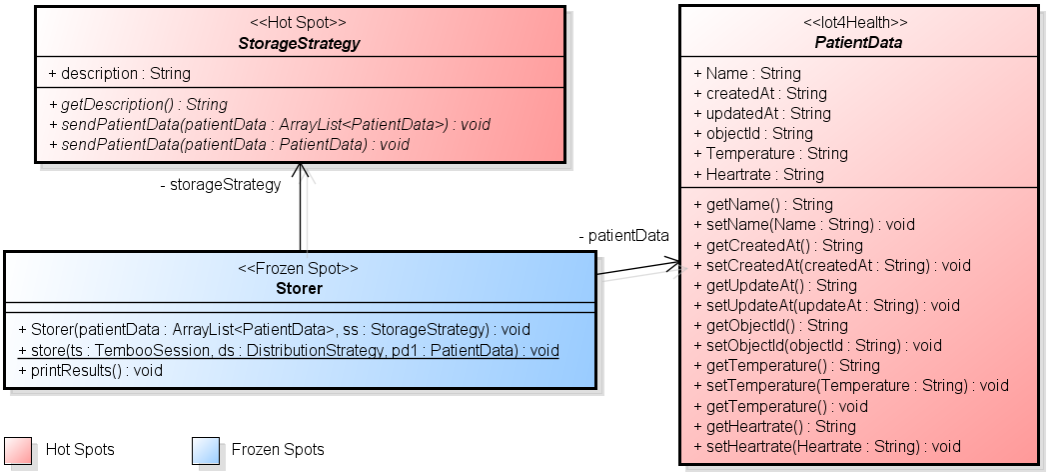


Figura 25 – Diagrama de Classes do Módulo de Armazenamento (M3) do IoT4Health.

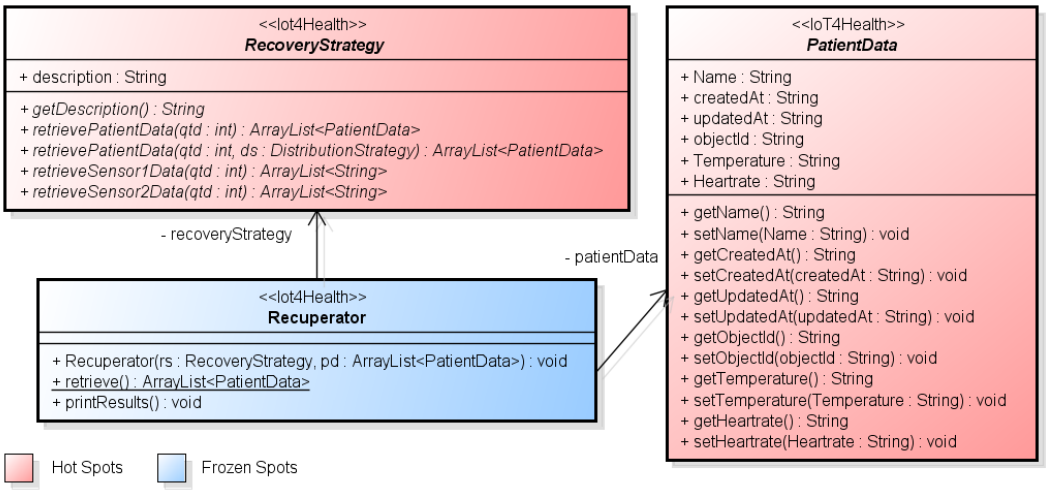


Figura 26 – Diagrama de Classes do Módulo de Recuperação (M4) do IoT4Health.

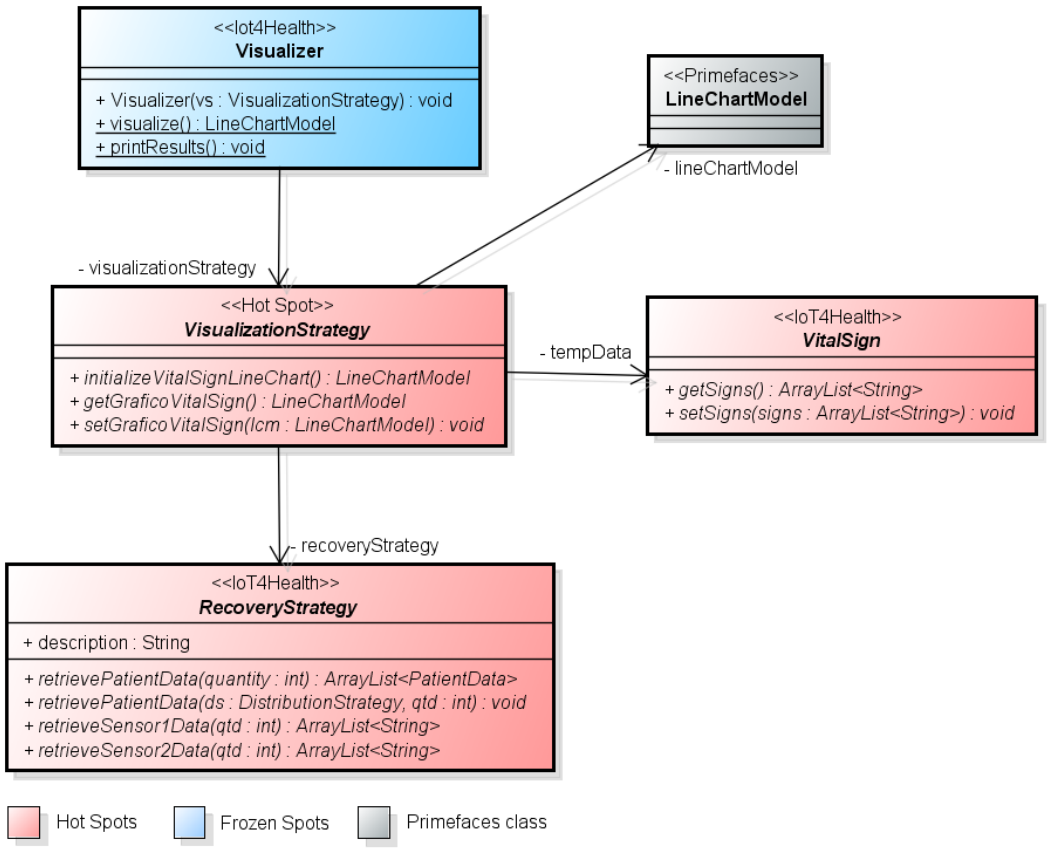


Figura 27 – Diagrama de Classes do Módulo de Visualização (M5) do IoT4Health.

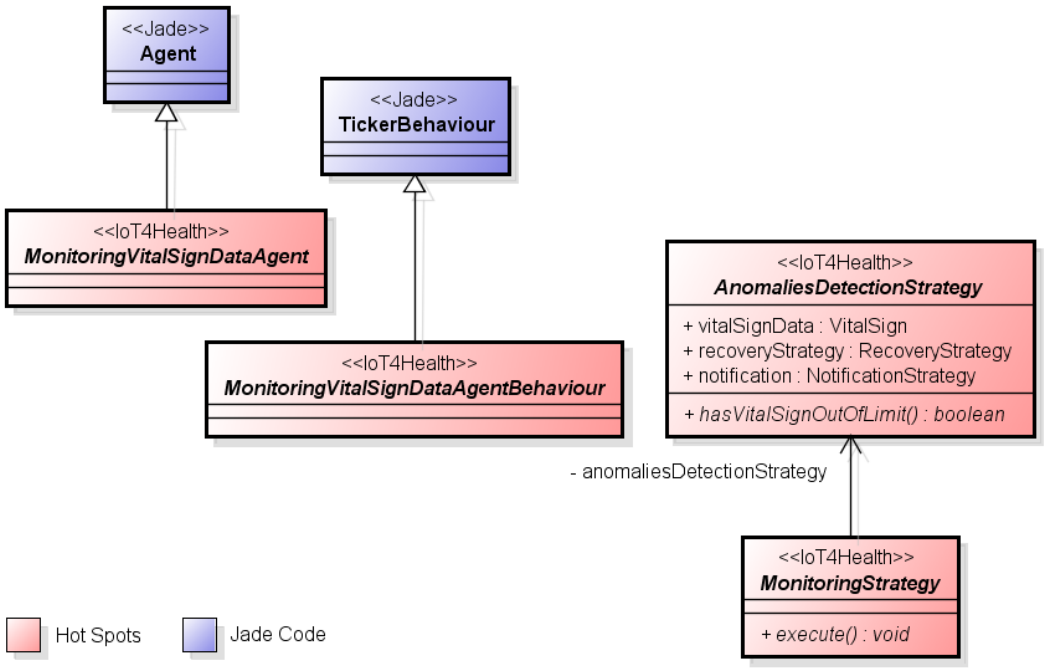


Figura 28 – Diagrama de Classes do Módulo de Monitoramento (M6) do IoT4Health.

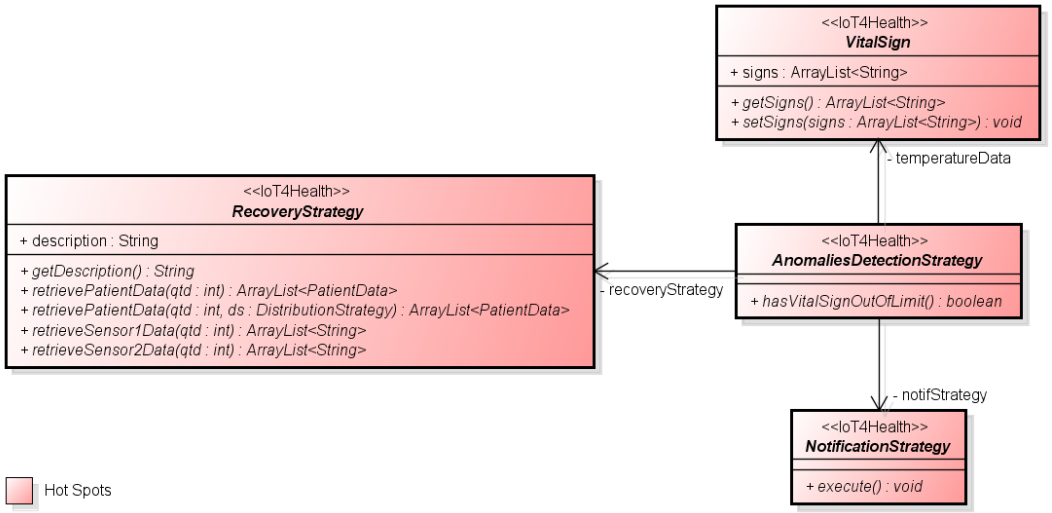


Figura 29 – Diagrama de Classes do Módulo de Detecção de Anomalias (M7) do IoT4Health.

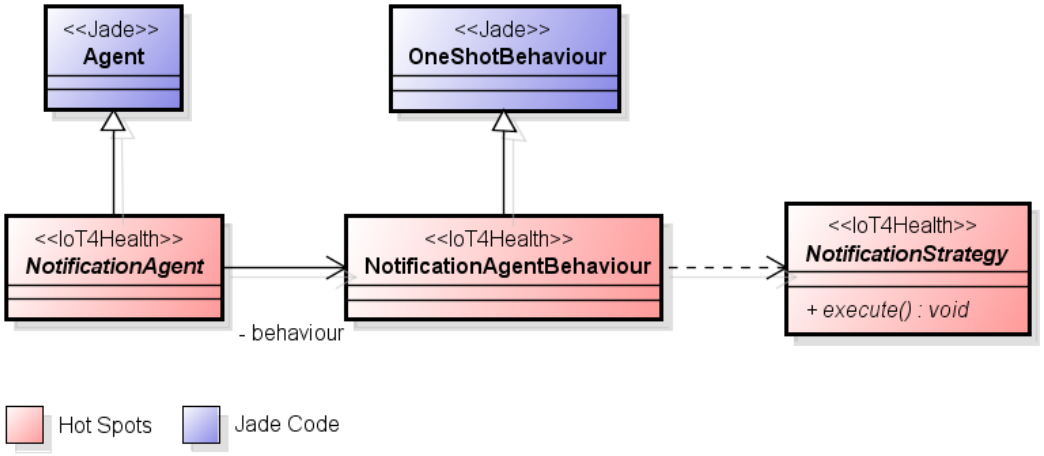


Figura 30 – Diagrama de Classes do Módulo de Notificação (M8) do IoT4Health.

7.5. Instanciação do Framework IoT4Health – Provas de Conceito

Três instâncias do framework foram criadas para servir como provas de conceito da ferramenta que contruímos:

1- Agents4Health;
2- Remote Patient Monitoring (RPM);
3- E-Health System;

As subseções a seguir apresentam os detalhes referentes à modelagem, projeto de arquitetura e aspectos de implementação relacionados às tecnologias utilizadas na etapa de construção destes sistemas, juntamente com a motivação por trás destas escolhas.

Serão apresentados também os frozen spots das aplicações desenvolvidas, ou seja, o código que representa o *kernel* das aplicações, juntamente com os hot spots que foram implementados em cada uma delas como forma de estender a arquitetura do IoT4Health.

7.5.1.
1ª Instância do IoT4Health - Agents4Health

7.5.1.1.
As Coisas modeladas no Agents4Health

A tabela a seguir relaciona as coisas envolvidas na unidade de tratamento hospitalar onde o paciente irá receber o atendimento médico. Elas foram modeladas no sistema conforme mostra a tabela a seguir.

Tabela 2 - Coisas modeladas no Agents4Health.

Coisa	Descrição
C1	Pacientes
C2	Braceletes com tags RFID
C3	Profissionais de saúde
C4	Ambiente do paciente (UTI, CTI, Sala de cirurgia, apartamento)

7.5.1.2.
Os Agentes modelados no Agents4Health

Foram modelados os seguintes agentes de software no sistema:

Tabela 3 - Agentes modelados no Agents4Health, com seu tipo de comportamento baseado no modelo de comportamentos do JADE.

Identificador do Agente	Descrição	Behaviour
MonitoringSensorTemperatureDataAgent	Procura anomalias nos dados sensoreados	*TickerBehaviour*
NotificationBySMSAgent	Notifica anomalias, através do envio de mensagens	*OneShotBehaviour*

Para construção do Agents4Health, foi utilizada a linguagem de programação Java. A arquitetura da aplicação foi projetada de modo que esta fosse composta por módulos com responsabilidades bem definidas. Os agentes foram modelados no sistema a fim de compor um módulo específico, que conteve agentes reativos e foi implementado utilizando-se a versão 4.3.0 da ferramenta JADE. Jade é um

framework para desenvolvimento de aplicações de agentes em Java, que simplifica a implementação de sistemas multi-agentes através de um middleware que está em conformidade com o padrão FIPA (Foundation For Intelligent, Physical Agents) e utiliza um conjunto de ferramentas gráficas que suportam as fases de depuração e implantação.

A API JADE disponibiliza dois tipos de classes de comportamentos que podem ser estendidos para os agentes: Primitivos e Compostos. O comportamento dos agentes do Agents4Health foi implementando a partir deste modelo de comportamentos, mais especificamente da classe de comportamentos primitivos (*Primitive Behaviours*). Cada agente da nossa aplicação é uma extensão da classe **Agent** e possui um comportamento que corresponde à extensão da classe **Behaviour** do JADE. Todo agente do sistema tem seu comportamento definido através do seu método **setup**, onde seu comportamento é configurado, através do método **addBehaviour**.

O Agents4Health possui agentes reativos com dois tipos de comportamentos: **TickerBehaviour** e **OneShotBehaviour**. Comportamentos do tipo **TickerBehaviour** executam suas ações de forma cíclica. Por esse motivo, os agentes do nosso cenário que precisam realizar atividades de monitoramento contínuo implementam este comportamento, como é o caso dos agentes *MonitoringSensorTemperatureDataAgent*, que precisam monitorar continuamente dados vitais dos pacientes a fim de detectar anomalias;

Outros agentes, por sua vez, são encarregados de executar tarefas que não são realizadas em um intervalo de tempo predefinido, ocorrendo apenas sob demanda, em resposta a um determinado evento. É o caso do agente *NotificationBySMSAgent*, que envia mensagens à equipe médica, alertando-os sobre anomalias.

7.5.1.3. Considerações sobre a implementação

A construção deste sistema foi feita a partir das linguagens Java e C++. Utilizamos o *Integrated Development Environment* (IDE) do Arduino para implementar os módulos M2 e M3 do Agents4Health, na linguagem de programação C++. Alguns sensores requiseram a instalação de bibliotecas externas, fornecidas pelos seus fabricantes. No caso do Agents4Health, instalamos a

biblioteca do sensor de pulso, PulseSensor[23], bem como a biblioteca da API Temboo[24].

Os módulos restantes da aplicação (M0, M1, M4, M5 e M6, M7 e M8) foram desenvolvidos na linguagem Java. Para implementar os agentes de software contidos no módulo de Monitoramento (M6), usamos o framework JADE.

7.5.1.4. Design do Agents4Health

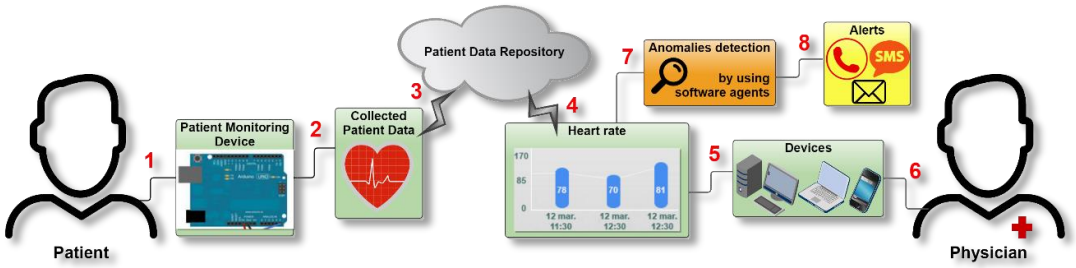


Figura 31 - Fluxo do Agents4Health (Design em alto nível de abstração).

A tabela a seguir ilustra o fluxo da aplicação Agents4Health, seguindo 8 passos:

Tabela 4 - Fluxo do Agents4Health em 8 passos.

Passo	Descrição
1	Paciente usa um dispositivo IoT para coleta de dados vitais, composto por micro-controladores e sensores (Figura 36).
2	Dados de temperatura e batimento cardíaco do paciente são coletados por este dispositivo.
3	Os dados coletados são enviados para o repositório remoto de dados (Figura 37), através do dispositivo IoT, via conexão wireless ou Ethernet.
4	Os dados recuperados da base de dados são exibidos de forma amigável aos usuários, sob a forma de *line charts* (Figura 39).
5	Através de um sistema web, são recuperados os dados do repositório.
6	Médicos e demais usuários autorizados podem visualizar os dados dos pacientes.
7	Agentes de software monitoram continuamente os dados do paciente a procura de valores anômalos definidos no sistema por um especialista no domínio.
8	Quando anomalias são detectadas, agentes de software enviam alertas à equipe médica, por SMS.

[23] http://pulsesensor.com/pages/code-and-guide
[24] https://www.temboo.com/

7.5.1.5.
Arquitetura do Agents4Health

Como instância do nosso framework, o sistema Agents4Health segue a arquitetura do IoT4Health, sendo estruturado em três camadas, conforme ilustrado abaixo:

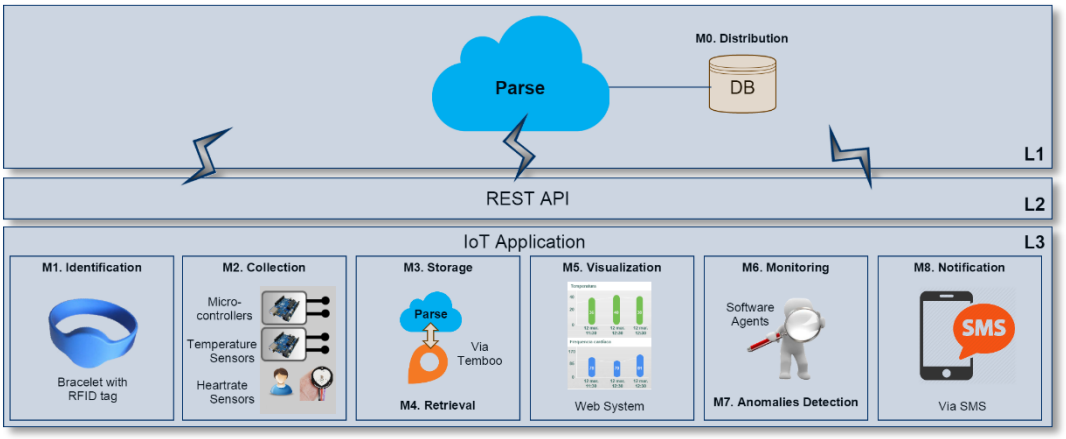


Figura 32 - A arquitetura do Agents4Health com suas três camadas (L1-L3). A aplicação IoT (L3) interage com a plataforma baseada em nuvem (L1) através da API REST (L2).

7.5.1.6.
L1. Camada de Distribuição de Dados

Para prover o serviço de distribuição de dados da aplicação, nós usamos um serviço de armazenamento de dados remoto gratuito chamado Parse[25]. Desenvolvedores podem utilizar o Parse para configurar facilmente o *backend* de um aplicativo. Esta plataforma em nuvem provê funcionalidades como salvar objetos de dados, armazenar arquivos, suportar *push notifications* e gerenciar usuários. Além disso, ela permite fazer portabilidade dos dados, oferecendo a opção de exportar todos os dados da base, no formato da API REST, além da possibilidade de importação de dados neste formato. A plataforma também possui um importador CSV, que aceita arquivos em formatos XLS, TXT e CSV.

[25] https://parse.com/

7.5.1.7.
L2. Camada de Comunicação de Dados

O Agents4Health realiza a comunicação da aplicação IoT (L3) com a plataforma de armazenamento na nuvem (L1), através da API REST. A aplicação envia e recupera dados da nuvem, através de requisições HTTP.

Para implementar o serviço da camada L2, nós usamos uma tecnologia chamada Temboo. O Temboo é uma plataforma que pode ser usada para facilitar a codificação de aplicações e a interação com diversas APIs. Ela possui uma biblioteca específica para o Parse, chamada biblioteca Parse, que possui sete pacotes para facilitar a criação de código para consumo dos servidos oferecidos por esta plataforma em nuvem. Nesta aplicação, estamos usando o seu pacote **Queries** para facilitar a criação de consultas para inserir e recuperar dados do Parse. Desse modo, através da biblioteca Parse da API Temboo, a aplicação Agents4Health faz upload dos dados do paciente para o repositório remoto dos dados.

7.5.1.8.
L3. Camada de Gerenciamento de Dados

É responsável por todo o processamento de informações do Agents4Health. Compreende a aplicação IoT, com seus oito módulos (M1-M8) apresentados a seguir.

7.5.1.8.1.
M1. Identificação de pacientes

Para a identificação do paciente, usamos um sistema RFID que inclui um dispositivo correspondente a um bracelete com uma tag RFID (Figura 33) e um leitor RFID (Figuras 34 e 35). Deste modo, a identificação é realizada através de uma interface de rádio frequência, onde cada paciente recebe uma pulseira contendo uma tag RFID que será utilizada como código de identificação único no sistema. Também pode ser usada como input do sistema, disparando a inicialização do módulo de visualização da aplicação (M5). Neste caso, irá permitir a rápida visualização dos dados vitais do paciente que está sendo monitorado.

Figura 33 – Acessórios com tags RFID utilizados na identificação de pacientes.

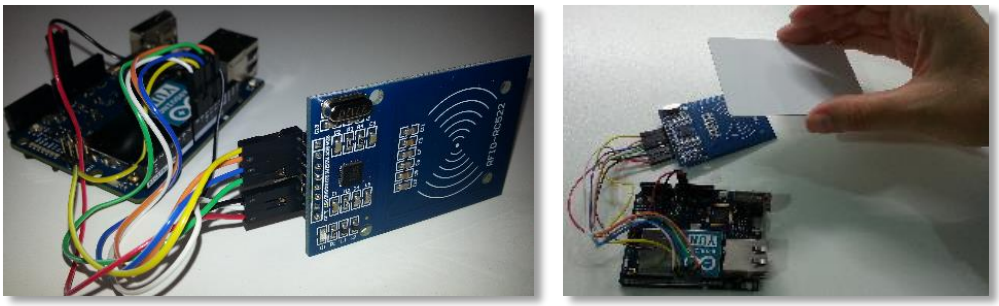


Figura 34 - Protótipo para identificação de pacientes através de leitor RFID.

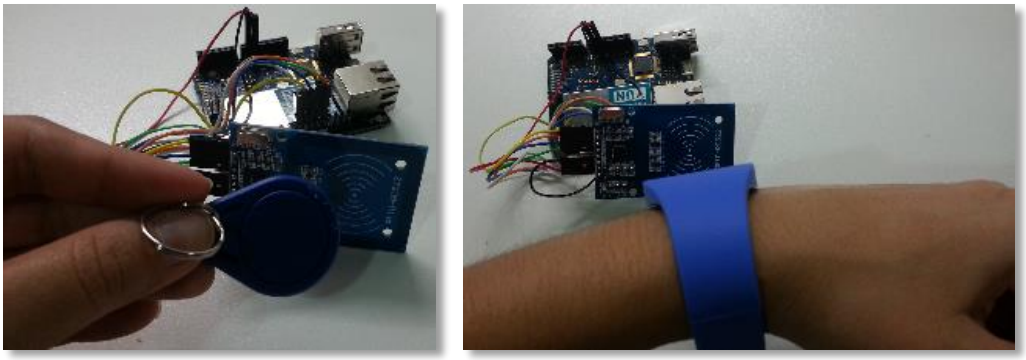


Figura 35 - Leitura de acessórios com tags RFID através do nosso protótipo.

7.5.1.8.2. M2. Coleta

Permite a coleta de dados vitais do paciente, como pulso e temperatura corpórea. Este módulo pode ser estendido para coletar outros dados do paciente como quantidade de passos que ele deu em determinado intervalo de tempo, além de dados do ambiente em que ele está inserido, como luz, humidade, nível de ruído, dentre outros.

Este processo, que também chamamos de sensoriamento, é realizado de forma automática. Para que esta solução funcione, é preciso que se utilize os

microcontroladores e que, a cada microcontrolador, sejam adicionados sensores de diferentes tipos para realizar a coleta de dados. Em nossa aplicação, utilizamos microcontroladores Arduino, juntamente com sensores de batimento cardíaco e temperatura, que formaram um dispositivo IoT capaz de coletar os dados paciente sem intervenção humana (Figura 36). 1

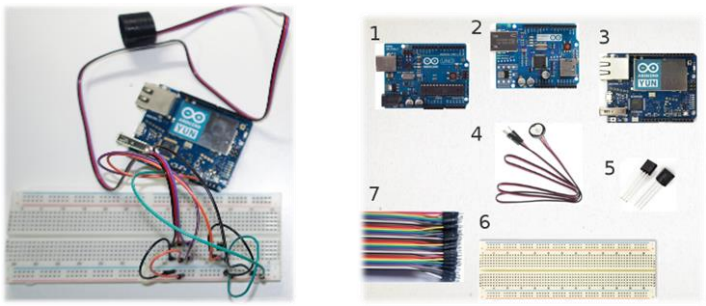


Figura 36 - Protótipo IoT para o monitoramento de pacientes. Componentes do protótipo: 1-Arduino Uno R2; 2- Shield Ethernet; 3-Arduino Yún; 4-Sensor de pulso; 5-Sensores de temperatura; 6-Protoboard; 7-Fios.

7.5.1.8.3. M3. Armazenamento

Uma vez coletados, a aplicação transmite os dados dos pacientes para a plataforma baseada nuvem, através da Internet. O armazenamento remoto permite que os dados possam ser acessados, por meio de uma interface *user-friendly*, a partir de qualquer dispositivo (computador, smartphone ou tablet) e por qualquer pessoa autorizada previamente.

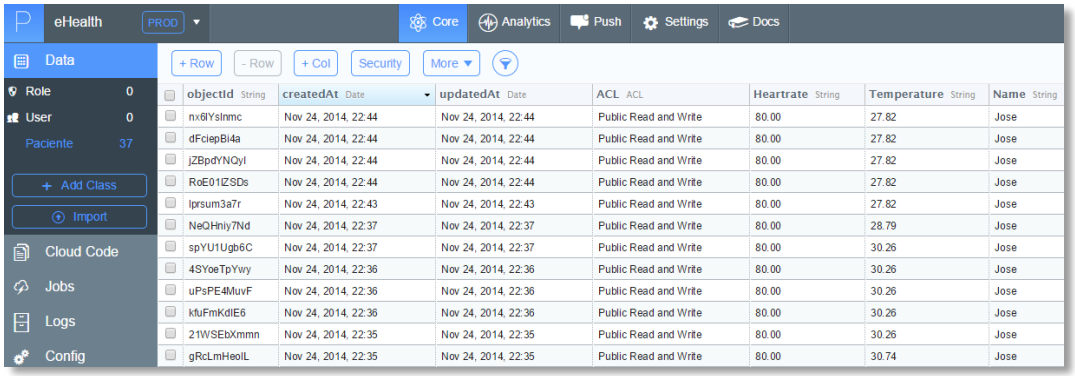


Figura 37 - Dados do paciente no Parse, plataforma baseada em nuvem que estamos utilizando para implementar a camada L1.

7.5.1.8.4.
M4. Visualização

Com o intuído de facilitar a visualização dos dados armazenados em tabelas, foi desenvolvida uma aplicação web que interage com a base de dados na nuvem.

Figura 38 - Tela inicial do sistema web desenvolvido para interagir com a base de dados remota e apresentar os dados dos pacientes aos usuários.

O sistema web opera realizando consultas aos dados do paciente e retornando para a aplicação web os resultados para serem exibidos em forma de gráficos, sendo plotados em *line charts*. Nesta implementação, temos um gráfico para os valores coletado por cada um dos sensores (Figura 39).

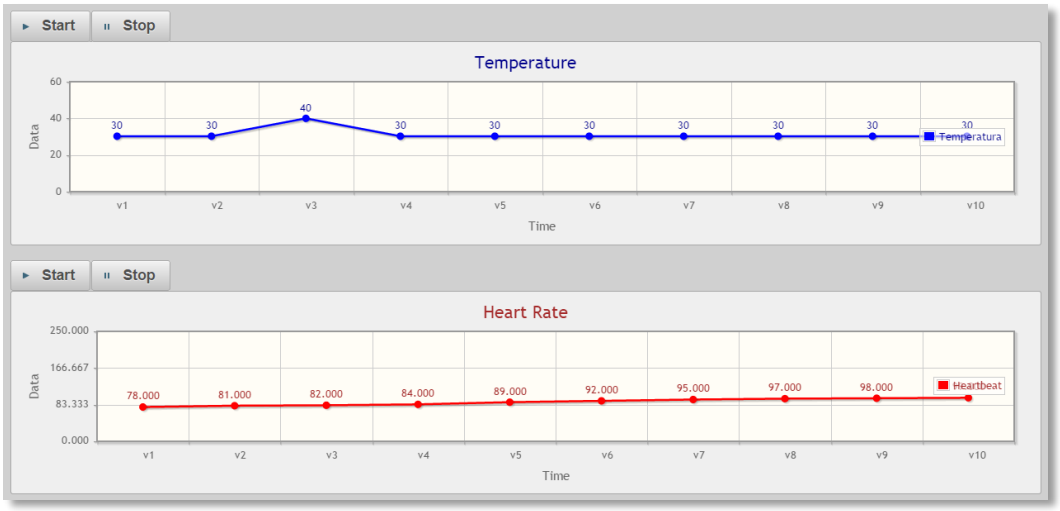


Figura 39 - Gráficos mostrando dados de temperatura e batimento cardíaco de um paciente, obtidos através de sensores.

7.5.1.8.5. M5. Monitoramento

Este módulo monitora continuamente os dados coletados do paciente através de agentes de software que devem reagir em caso de detecção de anomalias. Ele é composto pelos agentes reativos: MonitoringSensorTemperatureDataAgent e NotificationBySMSAgent. O agente MonitoringSensorTemperatureDataAgent, conforme seu identificador sugere, possui a tarefa de monitorar os dados vitais que estão sendo sensoreados. O agente NotificationBySMSAgent, por sua vez, envia mensagens de alerta aos profissionais envolvidos no tratamento de um determinado paciente, quando este apresenta alguma anormalidade.

Seguindo o que prevê o framework IoT4Health, no Agentes4Health foi realizada a etapa obrigatória de configuração de informações relativas ao mecanismo de detecção de anomalias, para cada paciente. Nesta etapa, um usuário administrador definiu: o **Desired Value Range (DVR)** para cada sensor (no contexto dessa aplicação, os valores esperados de temperatura e pulso); os **Anomalous Values (AV)**; um rótulo representativo para cada anomalia; um profissional da equipe para tratar cada anomalia, o qual deverá ser alertado no momento da sua ocorrência, via SMS (Fernandes & Lucena, 2015).

Verificamos com Dr. Bruno Azevedo[26] quais valores poderiam ser considerados preocupantes pelos profissionais da saúde, de modo que um médico poderia estar interessado em saber quando seu paciente os apresentassem, bem como quais tipos de anomalias podem estar associadas a estes valores. As tabelas abaixo contém os valores anômalos e tipos de anomalias informados, considerando cada um dos sinais vitais utilizados no Agents4Health.

[26] Dr. Bruno de Azevedo é cardiologista e pesquisador. Participa de pesquisas no LES voltadas para área da Saúde e é coautor de um dos artigos publicados como resultado desde trabalho, no qual o RPM System é apresentado.

Tabela 5 - Exemplo de configuração para o módulo de detecção de anomalias, considerando valores anômalos para frequência cardíaca.

Valores anômalos para Frequência Cardíaca	**Tipos de Anomalias aos quais podem estar associados**
50	Bradicardia como bradicardia sinusal ou bloqueio atrioventricular.
110	Taquicardia como a fibrilação atrial, taquicardias supraventriculares e taquicardia ventricular.

Tabela 6 - Exemplo de configuração para o módulo de detecção de anomalias, considerando valores anômalos para temperatura.

Valores anômalos para Temperatura	**Tipos de Anomalias aos quais podem estar associadas**
35	Hipotermia.
38	Hipertermia, febre: estados inflamatórios, infecções etc.

Uma vez coletados e armazenados, os dados dos pacientes podem ser acessados pelo agente *MonitoringSensorTemperatureDataAgent*, de modo que possa monitorá-los para detectar possíveis anomalias. A Figura 48 exemplifica a atuação deste agente, onde ele detecta um valor anormal na temperatura de um dado paciente.

Quando um agente *MonitoringSensorTemperatureDataAgent* detecta uma anomalia, ele se comunica com o agente *NotificationBySMSAgent*, enviando uma mensagem que informa sobre a necessidade de alertar o profissional da saúde responsável por agir no tratamento da anomalia detectada. Deste modo, o agente *NotificationBySMSAgent* envia uma mensagem SMS para o profissional responsável. A Figura 49 mostra um exemplo de mensagem SMS enviada pelo agente *NotificationBySMSAgent* ao detectar uma anormalidade na temperatura de um dado paciente.

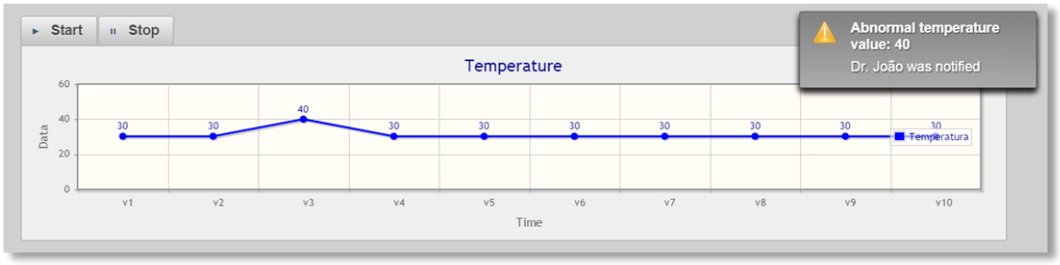


Figura 40 – Visualização em tempo real de temperatura de paciente. Notificação de detecção de anomalia pelo agente MonitoringSensorTemperatureDataAgent.

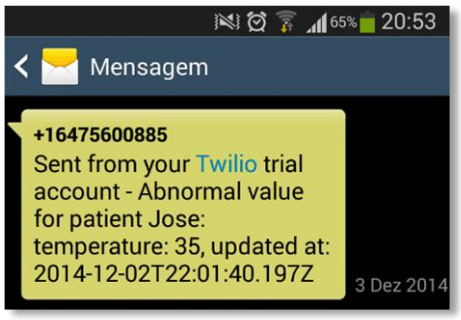


Figura 41 - SMS enviado ao profissional de saúde pelo agente NotificationBySMSAgent

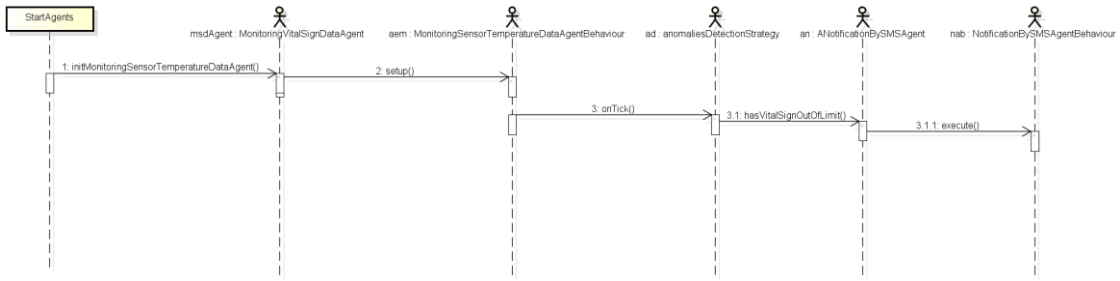


Figura 42 - Diagrama de Sequências mostrando a interação entre os agentes no sistema.

Considerando o cenário dos agentes de software que atuam no Agents4Health, podemos classificar seu ambiente como: inacessível, não-determinístico, dinâmico e contínuo. Neste contexto, é **inacessível** por não ser possível ter conhecimento completo sobre os dados do ambiente, uma vez que eles refletem o estado de saúde do paciente e são coletados em tempo real. É **não-determinístico** porque não é possível garantir que cada ação estará associada à exatamente um único efeito possível. **Dinâmico**, pois o ambiente está em constante mudança, de acordo com eventos que ocorrem e que fogem ao controle dos agentes. E, por fim, **contínuo**, já que existem infinitas e variáveis ações que podem acontecer neste cenário.

No Agents4Health, os critérios utilizados pelos agentes reativos para fazer o diagnóstico de um paciente especifico são prescritos e mantidos pelos médicos responsáveis e codificados em formato XML. (Figura 43).

```
<anomaly>
    <type>Bradyarrhythmia</type>
    <criteria type="comparison">
        <function symbol="<"/>
        <data>Heartrate</data>
        <value>50</value>
    </criteria>
</anomaly>
```

```
<anomaly>
    <type>Tachyarrhythmias</type>
    <criteria type="comparison">
        <function symbol=">"/>
        <data>Heartrate</data>
        <value>110</value>
    </criteria>
</anomaly>
```

```
<anomaly>
    <type>Hypothermia</type>
    <criteria type="comparison">
        <function symbol="<"/>
        <data>Temperature</data>
        <value>35</value>
    </criteria>
</anomaly>
```

```
<anomaly>
    <type>Hyperthermia</type>
    <criteria type="comparison">
        <function symbol=">"/>
        <data>Temperature</data>
        <value>38</value>
    </criteria>
</anomaly>
```

Figura 43 – Documento XML que codifica os tipos de anomalias e compõe a base de conhecimento dos agentes.

Com relação às tecnologias escolhidas, utilizamos uma biblioteca do Temboo, chamada biblioteca Twilio[27], para enviar alertas aos profissionais da saúde através de SMS. Twilio é uma plataforma com uma API de comunicações que disponibiliza APIs web-services, permitindo que os usuários construam suas próprias aplicações para comunicação por voz e SMS. Diferentes soluções usam o Twilio, inclusive na indústria de healthcare. Desenvolvedores de software podem usar sua API de serviços web para fazer e receber chamadas e enviar e receber mensagens de texto. Serviços de Twilio são acessados através do protocolo HTTP. Em nossa aplicação, fizemos uso do serviço do Twilio através da biblioteca **Twilio** do Temboo. Esta biblioteca possui o pacote SMSMessages que permite enviar e receber mensagens SMS.

[27] https://www.twilio.com/

7.5.1.9. Documentação do Agents4Health

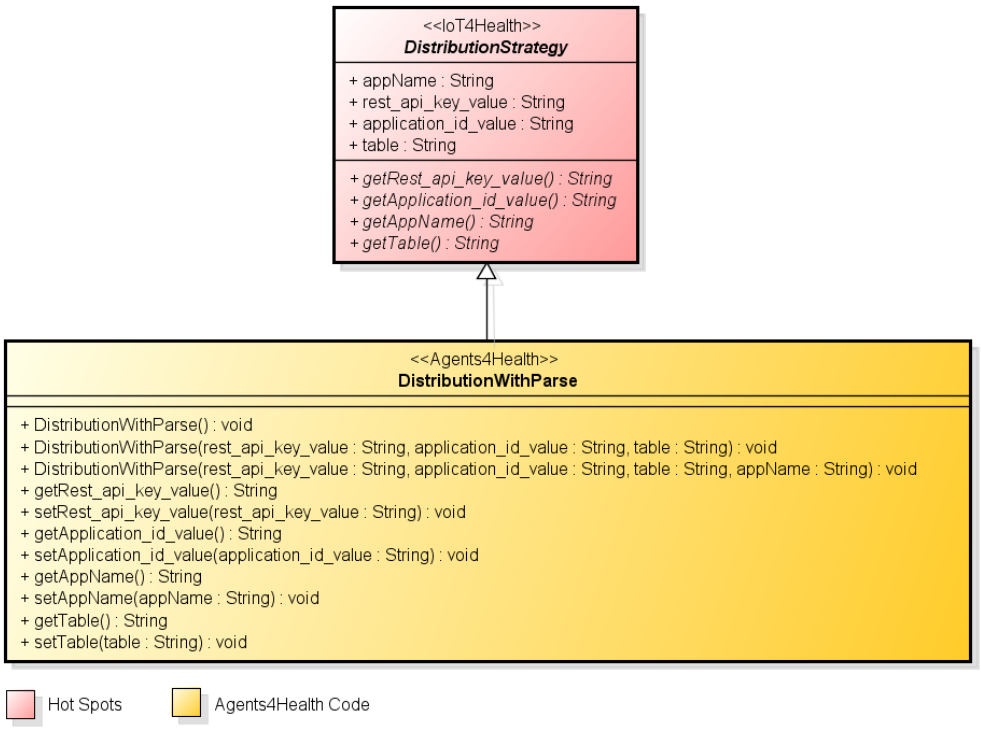


Figura 44 - Diagrama de Classes do Módulo de Distribuição (M0) do Agents4Health.

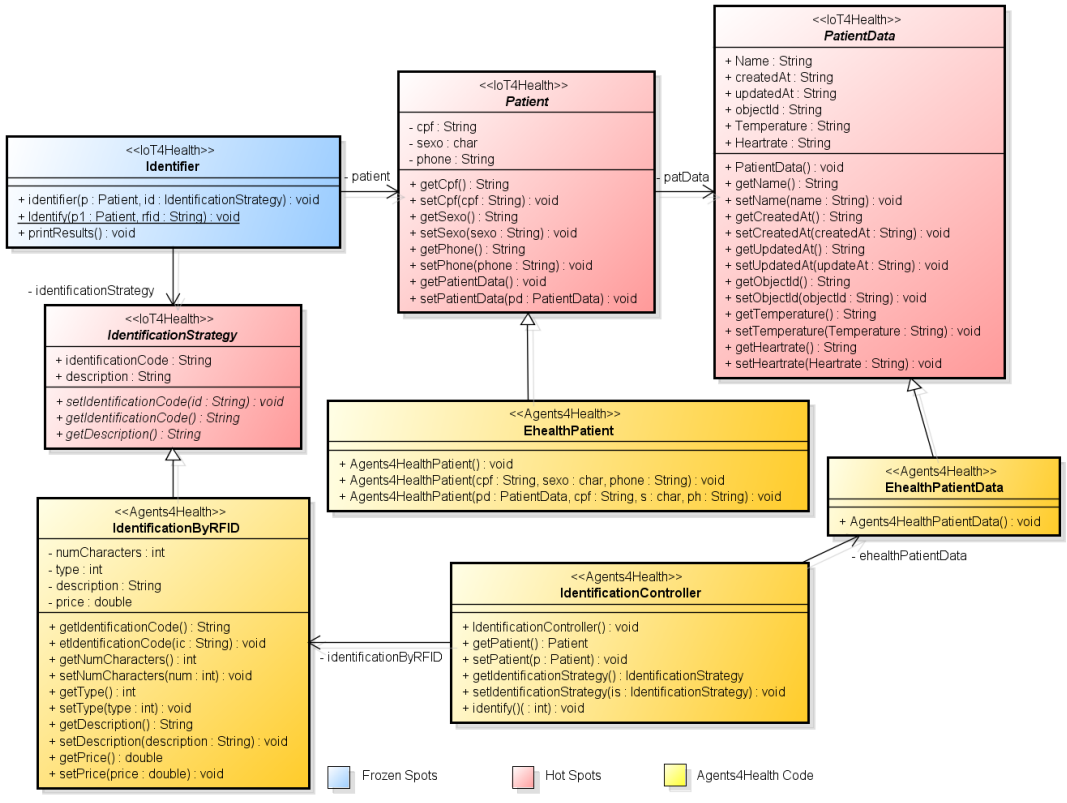


Figura 45 - Diagrama de Classes do Módulo de Identificação (M1) do Agents4Health.

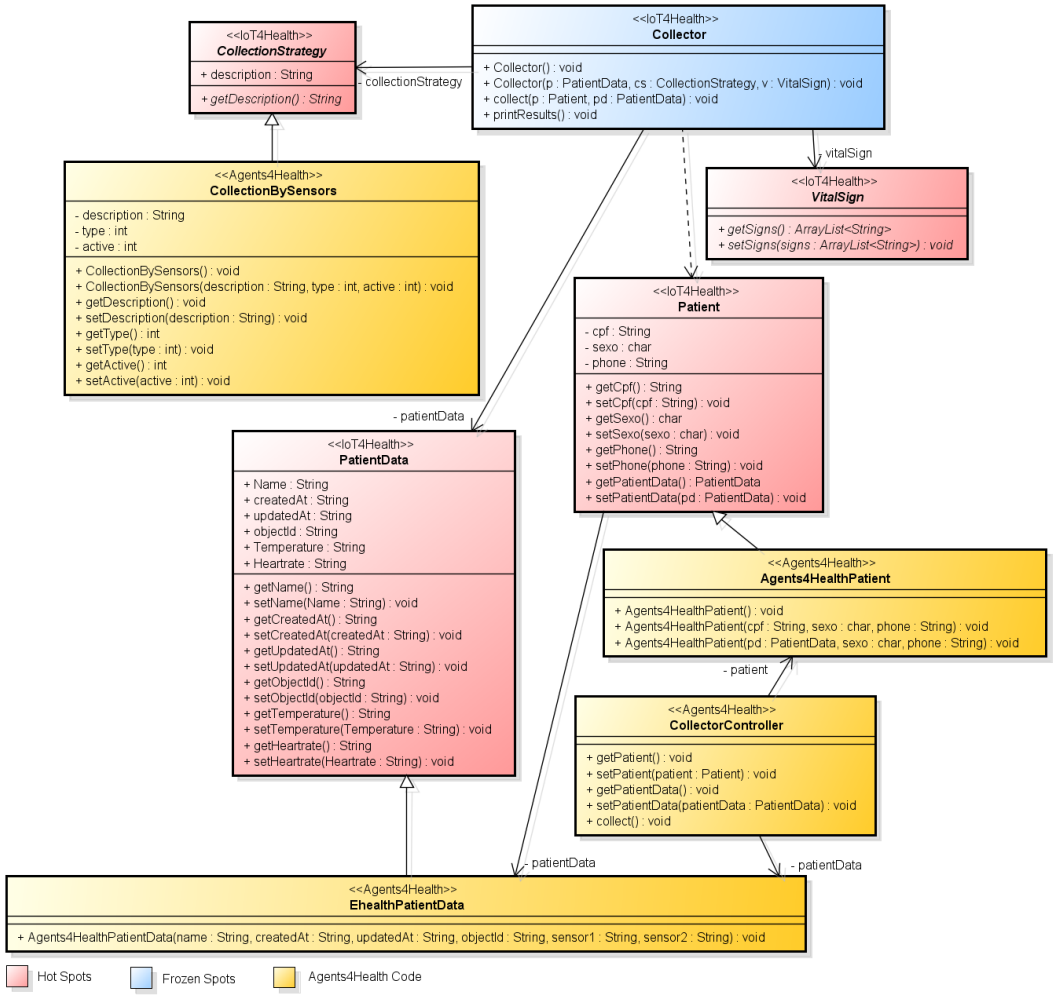


Figura 46 - Diagrama de Classes do Módulo de Coleta (M2) do Agents4Health.

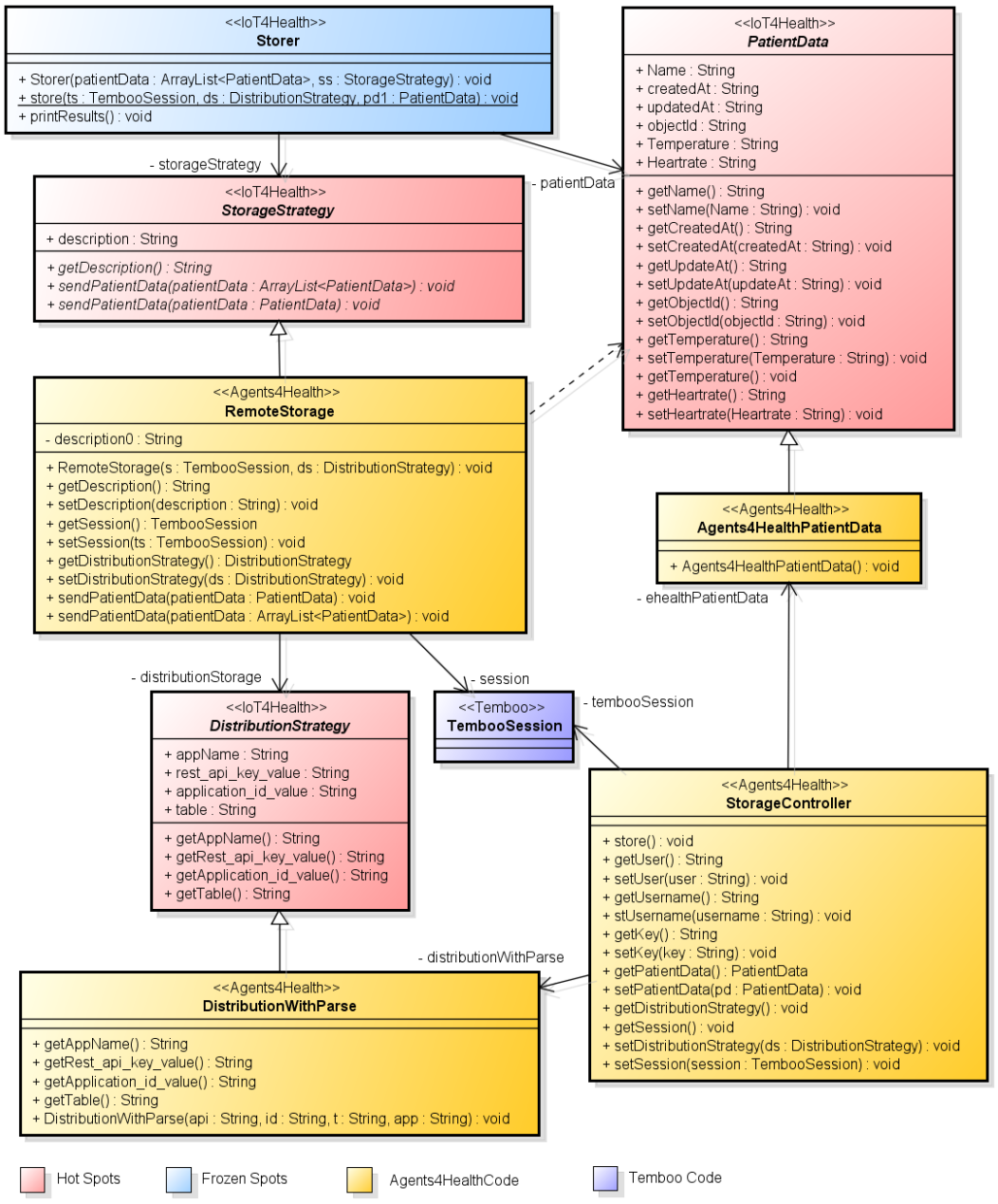


Figura 47 - Diagrama de Classes do Módulo de Armazenamento (M3) do Agents4Health.

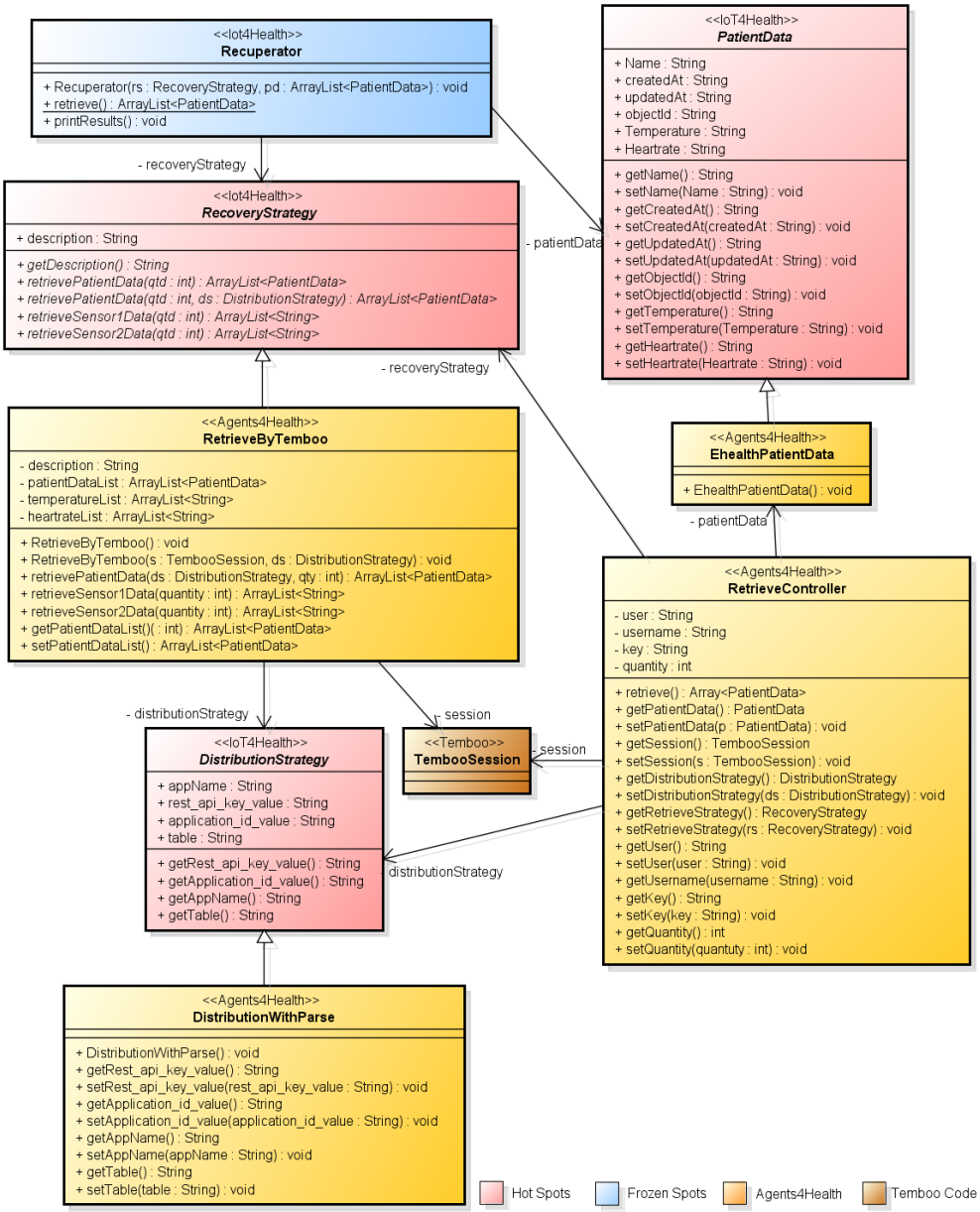


Figura 48 - Diagrama de Classes do Módulo de Recuperação (M4) do Agents4Health.

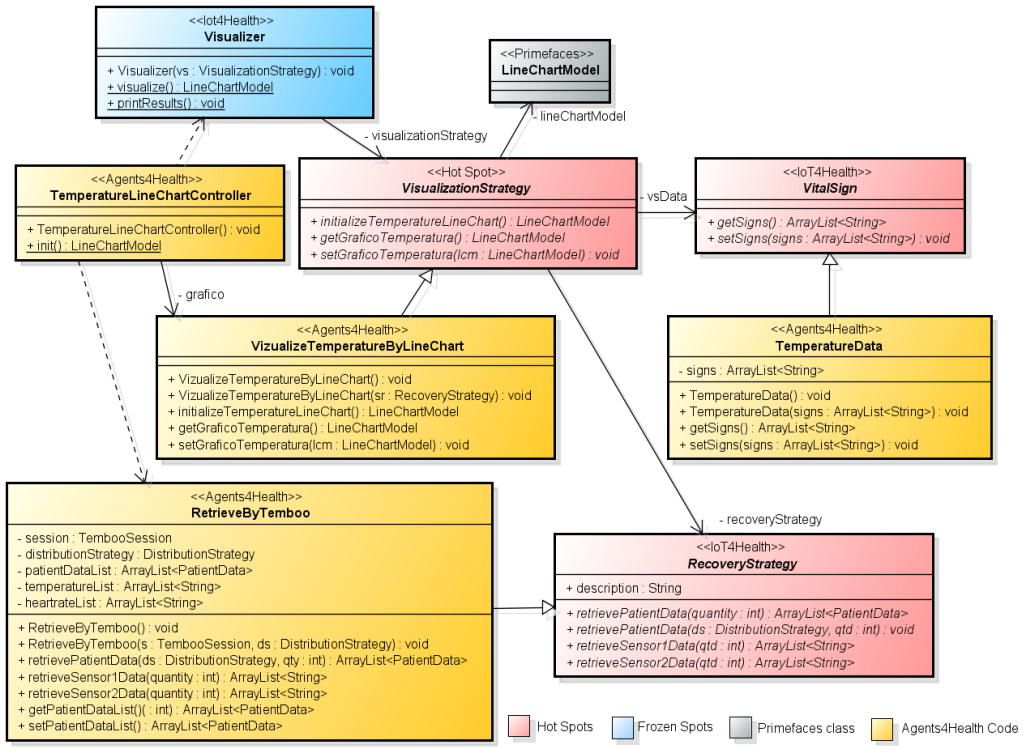


Figura 49 - Diagrama de Classes do Módulo de Visualização (M5) do Agents4Health.

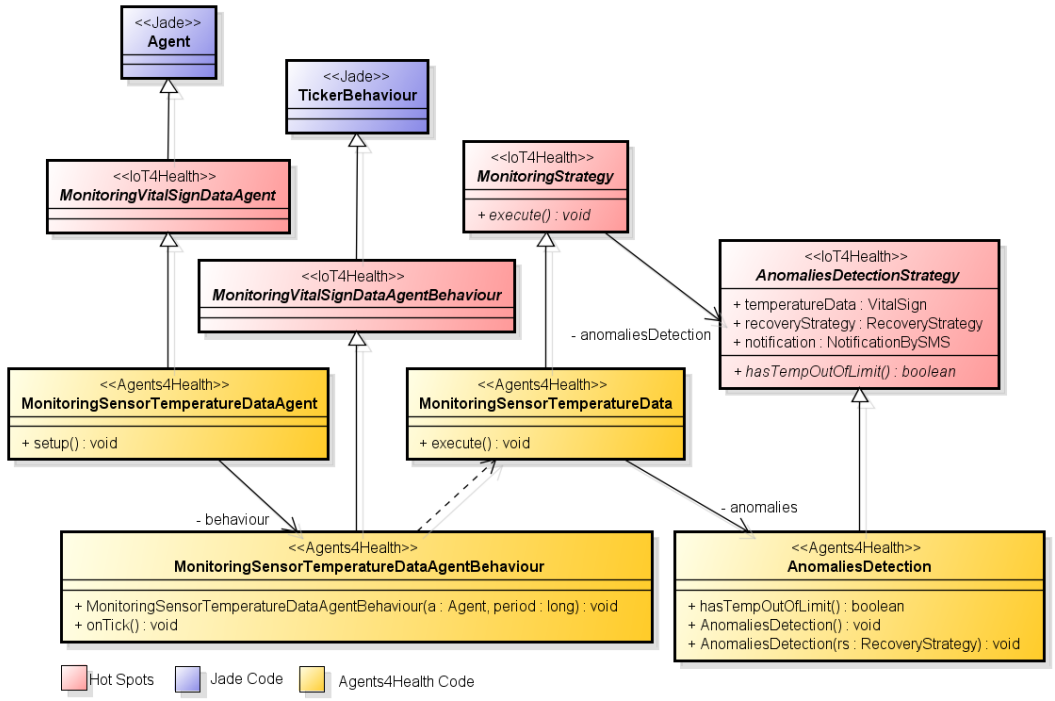


Figura 50 - Diagrama de Classes do Módulo de Monitoramento (M6) do Agents4Health.

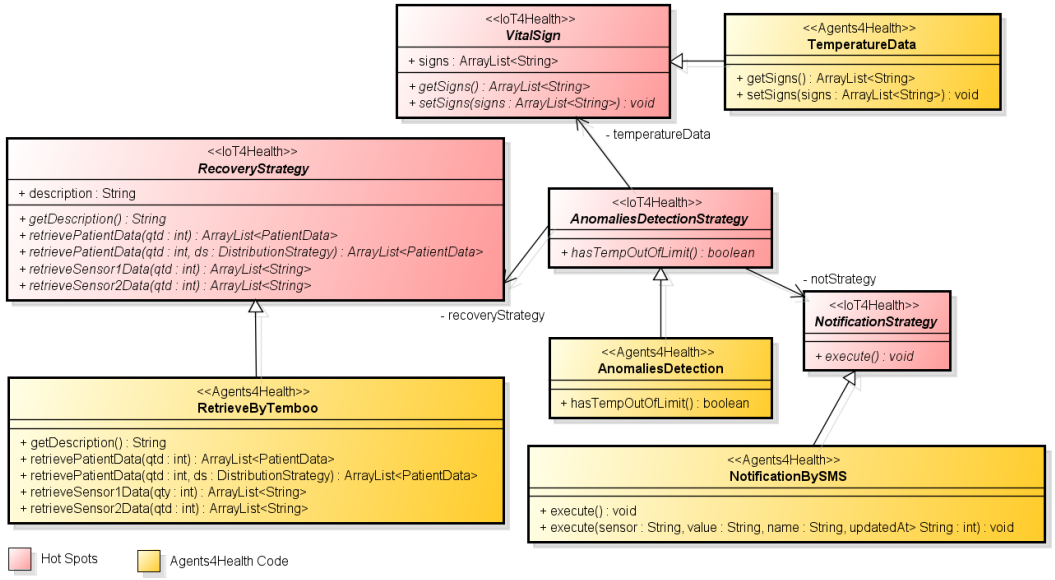


Figura 51 - Diagrama de Classes do Módulo de Detecção de Anomalias (M7) do Agents4Health.

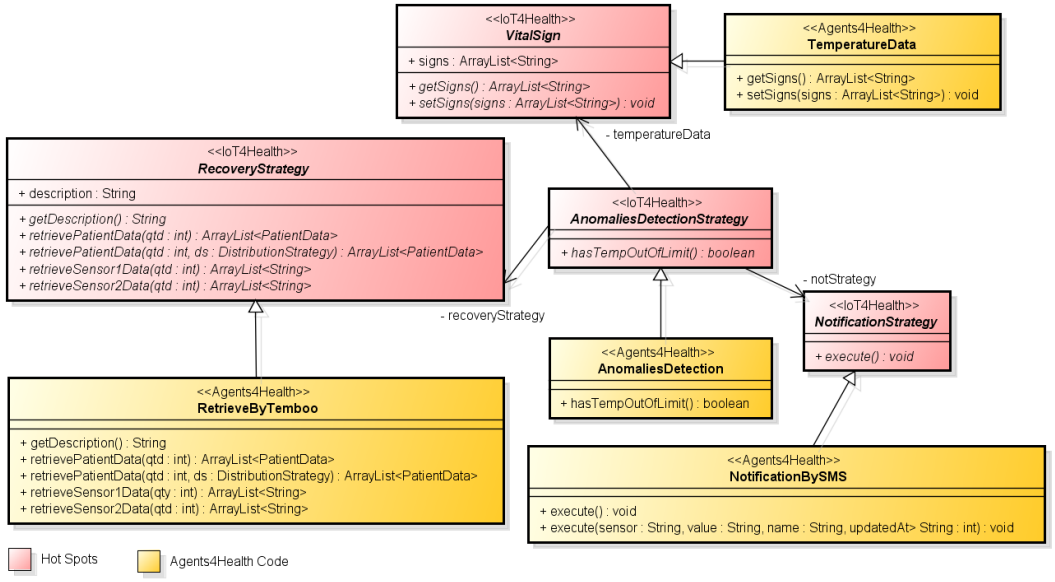


Figura 52 - Diagrama de Classes do Módulo de Notificação (M8) do Agents4Health.

7.5.2.
2ª Instância do IoT4Health – Remote Patient Monitoring (RPM)

O RPM System foi desenvolvido através da integração da nossa pesquisa com um projeto desenvolvido no LES - o MedData. O MedData envolve o desenvolvimento de uma ferramenta de suporte ao trabalho médico, com objetivo de facilitar o gerenciamento de dados envolvidos no tratamento de pacientes internados em UTIs (Unidades de Tratamento Intensivo).

7.5.2.1.
Arquitetura do RPM

Seguindo a arquitetura do IoT4Health, o RPM também está estruturado em três camadas (Figura 53).

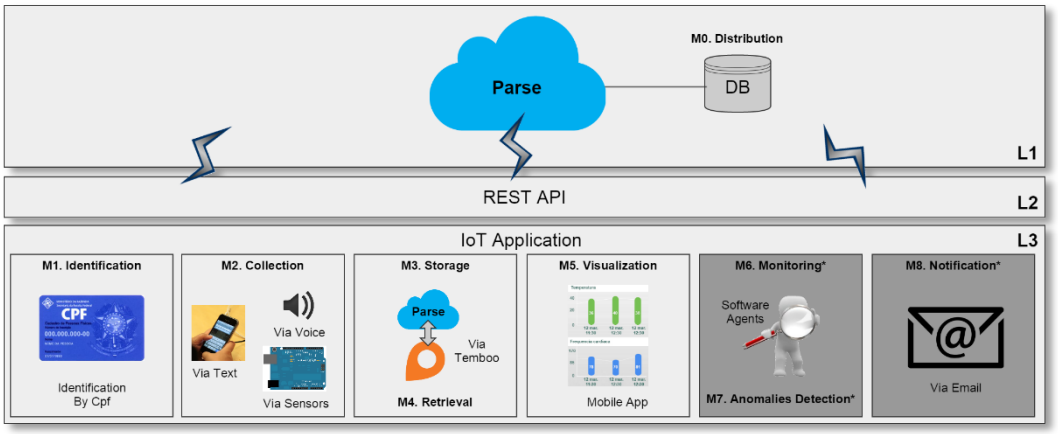


Figura 53 - A arquitetura do RPM em três camadas. A aplicação IoT comunica-se com a camada de distribuição de dados através da API REST.

7.5.2.1.1.
L1- Camada de Distribuição de Dados

Assim como Agents4Health, a camada de distribuição de dados também foi implementada utilizando-se a plataforma Parse. Em ambas as aplicações, o serviço de armazenamento de dados dos pacientes na nuvem foi implementado sem geração de custo, visto que utilizamos a versão gratuita da ferramenta.

7.5.2.2.
L2- Camada de Comunicação de Dados

A comunicação entre a plataforma baseada em nuvem e a nossa aplicação IoT também foi realizada através de API REST. Assim como na primeira instância do framework, também utilizamos a ferramenta Temboo para facilitar esta interação, já que o Temboo disponibiliza uma biblioteca específica para comunicação com o Parse usando a API REST.

7.5.2.3.
L3- Camada de Gerenciamento de Dados

A seguir, relacionamos os módulos do RPM que utilizamos para criar esta segunda aplicação.

7.5.2.3.1.
Módulo 1 (M1) - Identificação

Os pacientes foram identificados através de rádio frequência. Para isto, poderiam ser utilizados braceletes e outros acessórios como cartão ou chaveiro contendo tags RFID.

7.5.2.3.2.
Módulo 2 (M2) - Coleta

Nesta aplicação, a coleta de dados dos pacientes pode ser realizada de diversas formas. O sistema suporta a inserção de dados relacionados ao estado do paciente de forma manual, via texto, bem como a possibilidade de realizar coleta de dados automática, utilizando Arduinos e sensores. A inserção de dados via comandos de voz está em desenvolvimento. O input de dados por voz também será utilizado para permitir que um integrante da equipe médica possa atualizar o quadro de um dado paciente, no qual ele relata a evolução do paciente (Figura 56). Esta funcionalidade já está disponível com input de dados via texto.

7.5.2.3.3.
Módulo 3 (M3) - Armazenamento

Todos os dados dos pacientes são armazenados na nuvem. Conforme dito anteriormente, estamos usando o Parse como repositório dos dados.

7.5.2.3.4.
Módulo 4 (M4) - Visualização

Diferentemente da aplicação Agents4Health que permitia a visualização dos dados através de um sistema web, o RPM utiliza o aplicativo mobile MedData para visualização dos dados dos pacientes. Este aplicativo possui um módulo chamado “Tempo Real”, através do qual os profissionais da saúde podem acompanhar os

dados dos pacientes internados em UTIs, em tempo real (Figura 57). O MedData será explicado em mais detalhes na subseção 7.5.2.4.

7.5.2.3.5. Módulo 5 (M5) - Monitoramento

O MedData é um projeto em desenvolvimento e, em sua versão atual, o monitoramento de dados dos pacientes é realizado de forma manual, pelos profissionais da saúde. As estratégias de monitoramento e notificação de anomalias através de agentes de software foram planejadas como evoluções do aplicativo a serem implementadas. Uma vez que o módulo de visualização dos dados utiliza um aplicativo mobile para exibir as informações dos pacientes, os agentes de software que irão executar no smartphone serão implementados utilizando-se uma extensão do Jade (JADE-LEAP[28]).

7.5.2.4. Aplicativo MedData

O MedData é um projeto que vem sendo desenvolvido desde 2014 na PUC-Rio, envolvendo os departamentos de Informática e Design desta Instituição. Trata-se de uma ferramenta de software com aplicação na área médica, mais precisamente um aplicativo mobile a ser utilizado pelos profissionais da saúde para acompanhamento de pacientes internados em leitos.

O aplicativo vem evoluindo ao longo dos últimos meses, a fim de prover mais funcionalidades e fornecer melhor usabilidade aos seus usuários. Atualmente, o MedData está em sua terceira versão.

A primeira versão da aplicação foi testada no Instituto Nacional de Cardiologia[29] (INC), localizado no Rio de Janeiro. O INC é considerado um Centro de excelência nacional referenciado pelo Ministério da Saúde para atendimento em cardiologia e cirurgia cardíaca, sendo a única unidade pública do Rio de Janeiro que realiza cirurgias cardíacas neonatais e transplantes cardíacos.

[28] http://jade.tilab.com/doc/tutorials/LEAPUserGuide.pdf
[29] http://www.inc.saude.gov.br/ensino-pesquisa.asp

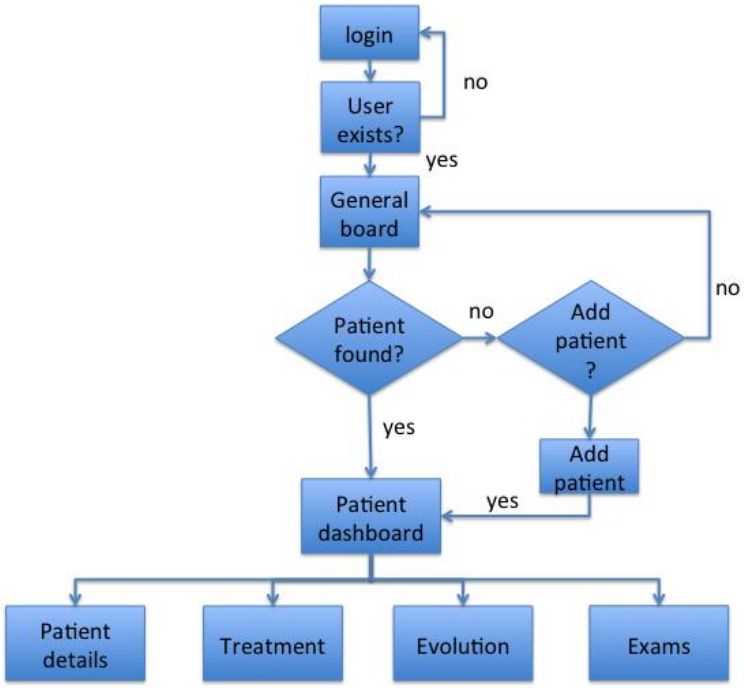


Figura 54 - Fluxograma de informação no MedData.

7.5.2.4.1. Projeto Piloto do MedData – Aplicação em ambiente hospitalar real

O projeto piloto do MedData foi conduzido por um período de seis meses durante o ano de 2014, na enfermaria do Instituto Nacional de Cardiologia e teve a supervisão do Dr. Bruno de Azevedo, que pertence ao corpo clínico do hospital. A ala em que o MedData foi aplicado é composta por trinta leitos, cujos cuidados estão distribuídos entre cinco médicos e dez estagiários da equipe.

Após a configuração inicial de cadastro de leitos do hospital no MedData e treinamento da equipe, os envolvidos no experimento foram informados sobre a necessidade de fazer atualizações diárias no MedData, com informações referentes a resultados de testes de laboratório, altas dos pacientes, transferências de leitos etc.

Nesta etapa, a frequência de uso do MedData foi constantemente observada. Após os primeiros dois meses, observou-se um decréscimo em torno de 20% na atividade de atualização dos dados. Esta queda ocorreu mesmo com os incentivos realizados para que o MedData fosse utilizado, através de alertas de que a integração de dados clínicos entre a equipe médica resultaria na segurança do paciente.

Algumas hipóteses foram sugeridas para justificar o declínio progressivo do uso do MedData:

1- Aproximadamente 50% da equipe utilizava smartphone com sistema operacional diferente do suportado no aplicativo;
2- A equipe relatou ser impraticável preencher grande volume de dados através de smartphones, com input via texto;
3- A enfermaria, onde foi realizado o projeto piloto, já possuía um sistema de registro médico eletrônico, atualizado através de um computador desktop e que não estava integrado ao MedData. Desta forma, o uso do MedData representava uma atividade extra para a equipe;

Baseado nas dificuldades encontradas no projeto piloto e nas questões reportadas pelos usuários, decidiu-se investir no redesign da aplicação, partindo para o projeto de uma segunda versão do aplicativo. Na segunda versão, o MedData poderia executar em diferentes sistemas operacionais como iOS e Android. A segunda mudança planejada e que tinha maior impacto dizia respeito ao método de entrada de dados usado pela equipe médica. Para facilitar o processo de entrada de dados, nas próximas versões do aplicativo, os usuários deveriam ser capazes de inserir os dados através de comandos de voz.

Com relação ao design da interface com o usuário, levou-se em consideração o fato de que o sucesso na utilização da aplicação depende da sua facilidade de uso. De acordo com o supervisor do projeto piloto, Dr. Bruno de Azevedo, em um ambiente médico, qualquer ferramenta que requeira trabalho extra acaba sendo descartada.

Segundo (Viitanen et al., 2011), a rotina médica, seja em um ambulatório ou hospital - departamento de emergência, enfermarias e unidades de terapia intensiva, tem a característica de ser dinâmica, rápida e com muitas tarefas cruciais a serem realizadas. Qualquer processo opcional que não deixar clara a sua utilidade imediata terá dificuldade em ser inserido nessa rotina diária.

Atualmente em sua terceira versão, o MedData contém uma evolução no processo de entrada de dados do paciente, para permitir a coleta automática. Neste momento, temos duas formas de coletar os dados dos pacientes: 1- Manual, via texto ou comandos de voz; 2- Automática, através de sensores.

Conforme descrito anteriormente, a coleta é feita através do protótipo de hardware que foi descrito e ilustrado na Figura 36. Para cada protótipo, nós utilizamos microcontroladores Arduino e sensores de temperatura e batimento

cardíaco. Pretendemos evoluir o protótipo para suportar o uso de outros sensores médicos, bem como sensores capazes de coletar dados do ambiente em que o paciente está sendo monitorado e seja relevante para a análise do estado de saúde do paciente.

As figuras a seguir mostram exemplos de funcionalidades do MedData, considerando o design atual do aplicativo.

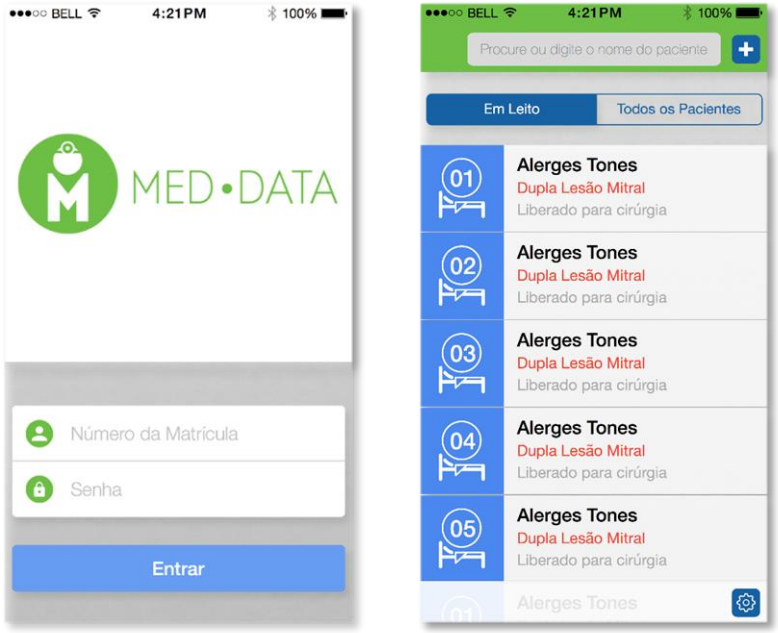


Figura 55 - Telas de Login e Quadro Geral.

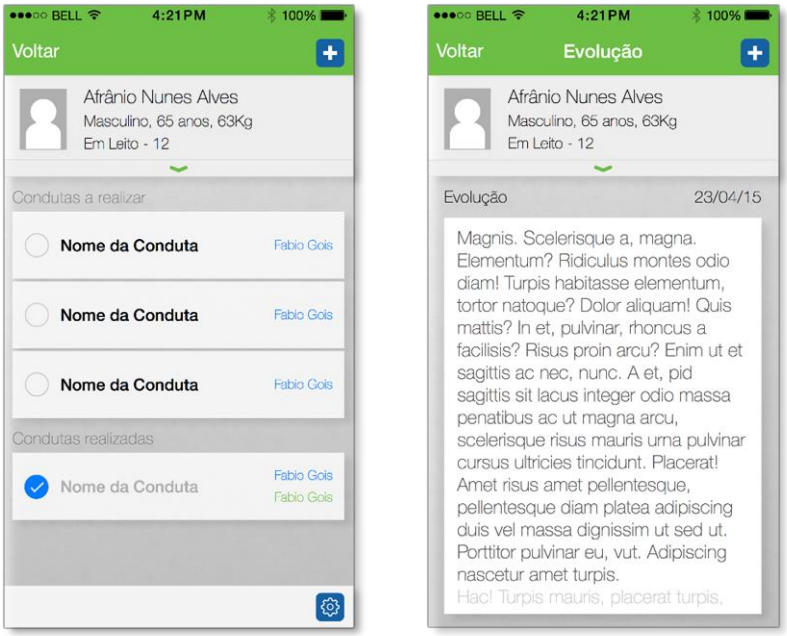


Figura 56 – Telas de Evolução do Paciente.

Figura 57 – Telas de Painel do Paciente e Tempo Real.

7.5.2.5. Documentação do RPM System

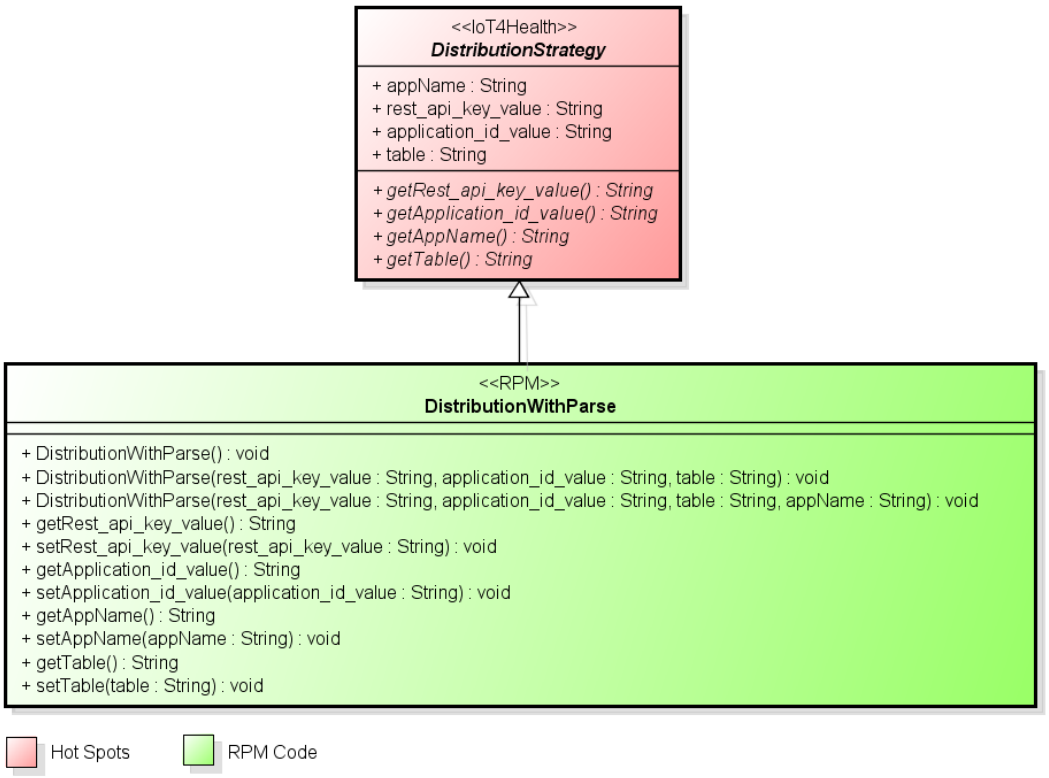


Figura 58 - Diagrama de Classes do Módulo de Distribuição (M0) do RPM System.

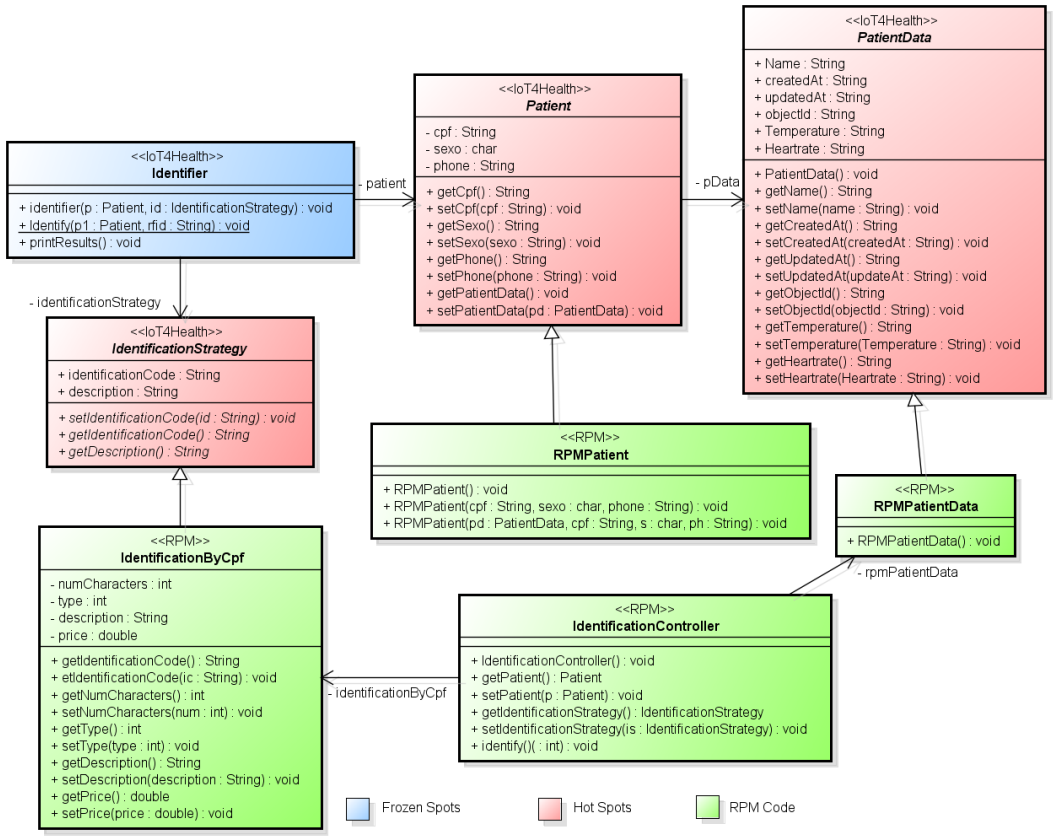


Figura 59 - Diagrama de Classes do Módulo de Identificação (M1) do RPM System.

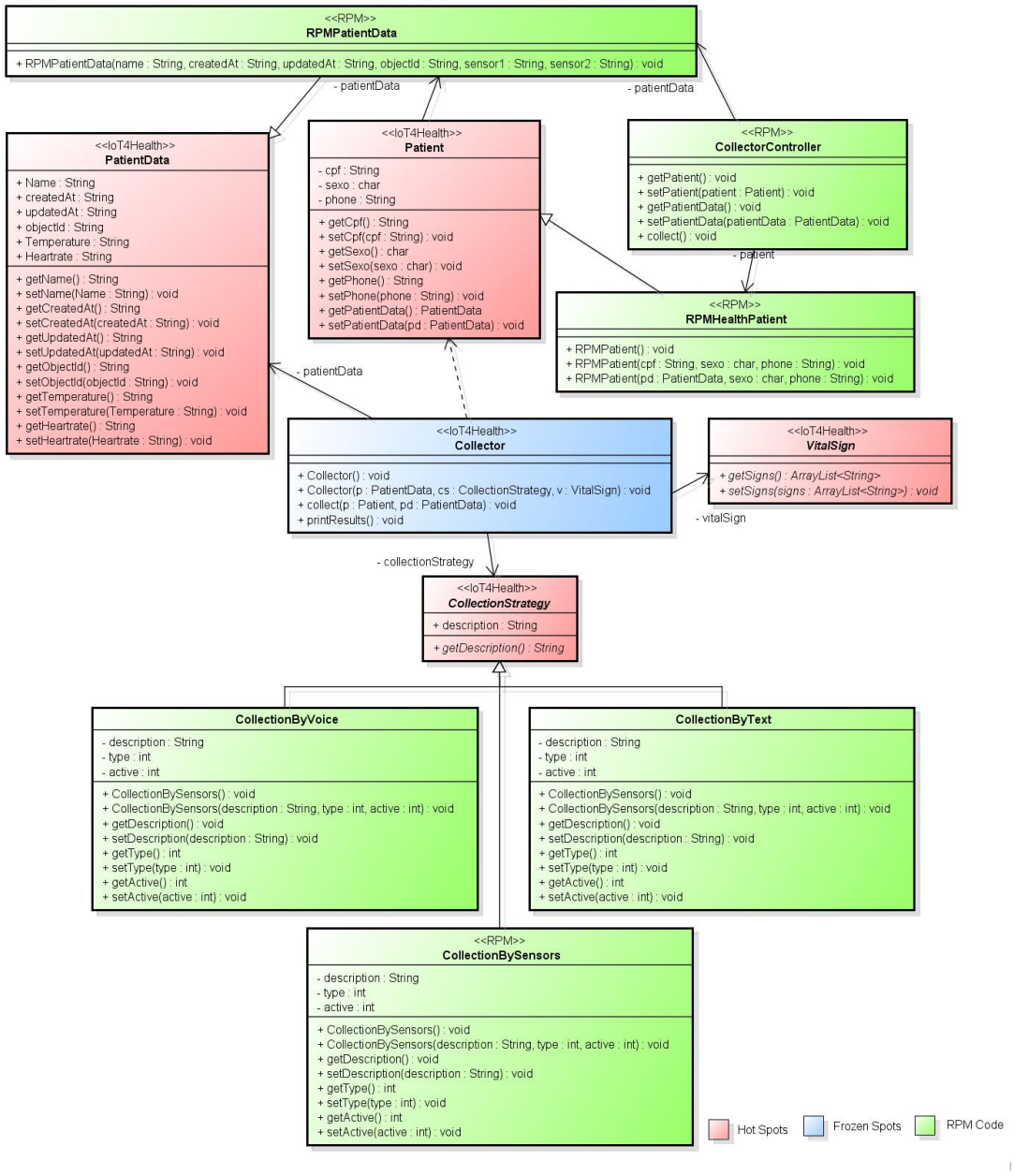


Figura 60 - Diagrama de Classes do Módulo de Coleta (M2) do RPM System.

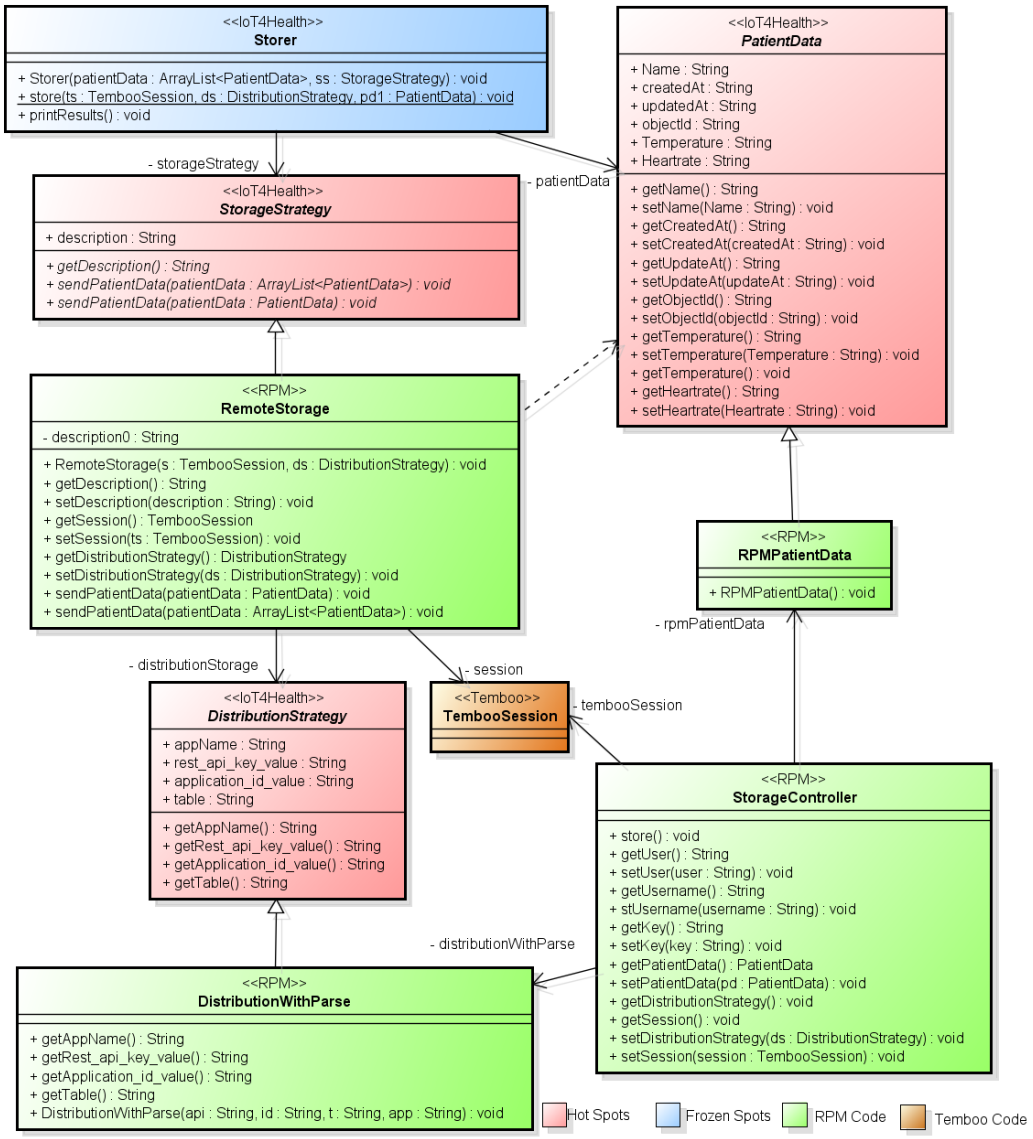


Figura 61 - Diagrama de Classes do Módulo de Armazenamento (M3) do RPM System.

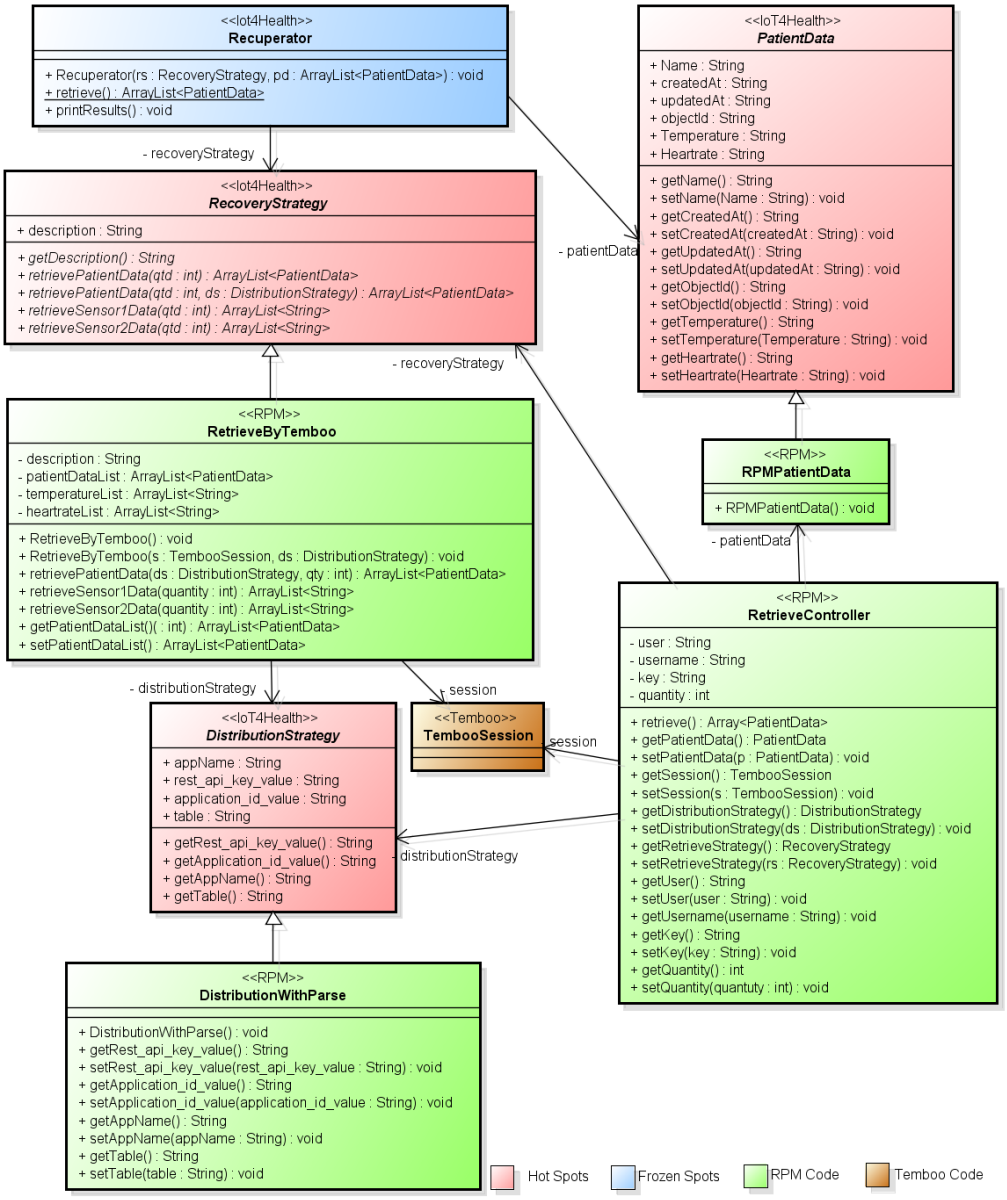


Figura 62 - Diagrama de Classes do Módulo de Recuperação (M4) do RPM System.

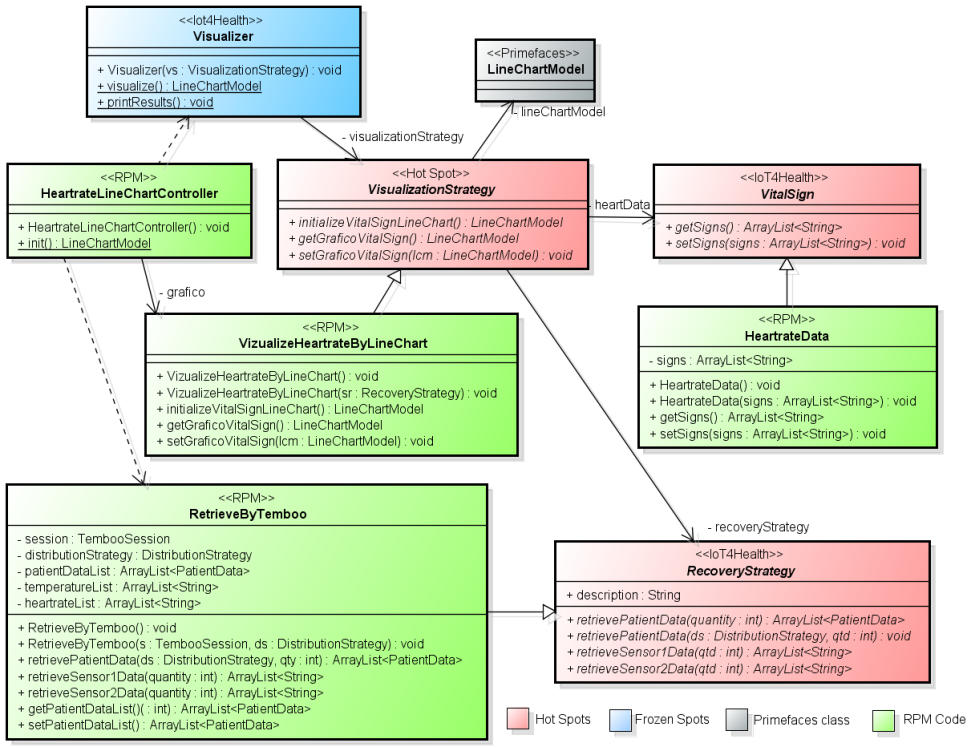


Figura 63 - Diagrama de Classes do Módulo de Visualização (M5) do RPM System.

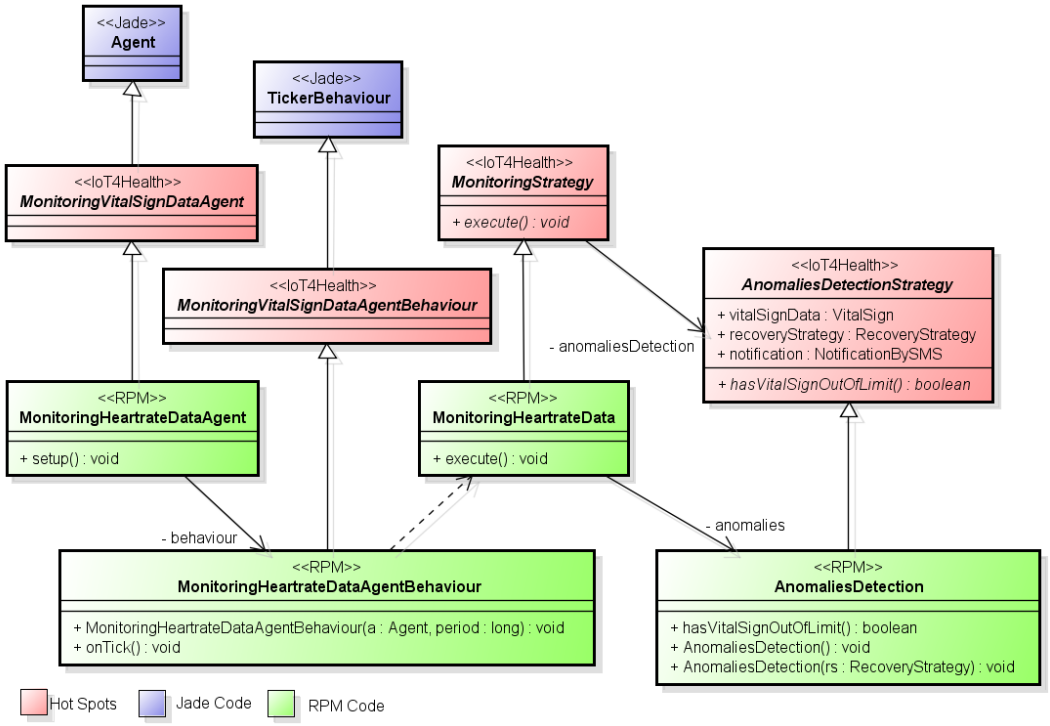


Figura 64 - Diagrama de Classes do Módulo de Monitoramento (M6) do RPM System.

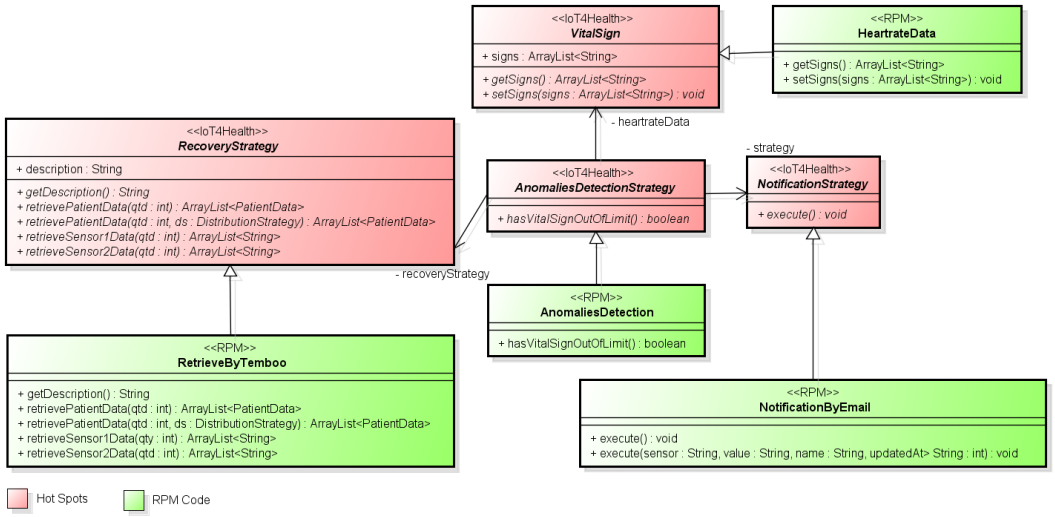


Figura 65 - Diagrama de Classes do Módulo de Detecção de Anomalias (M7) do RPM System.

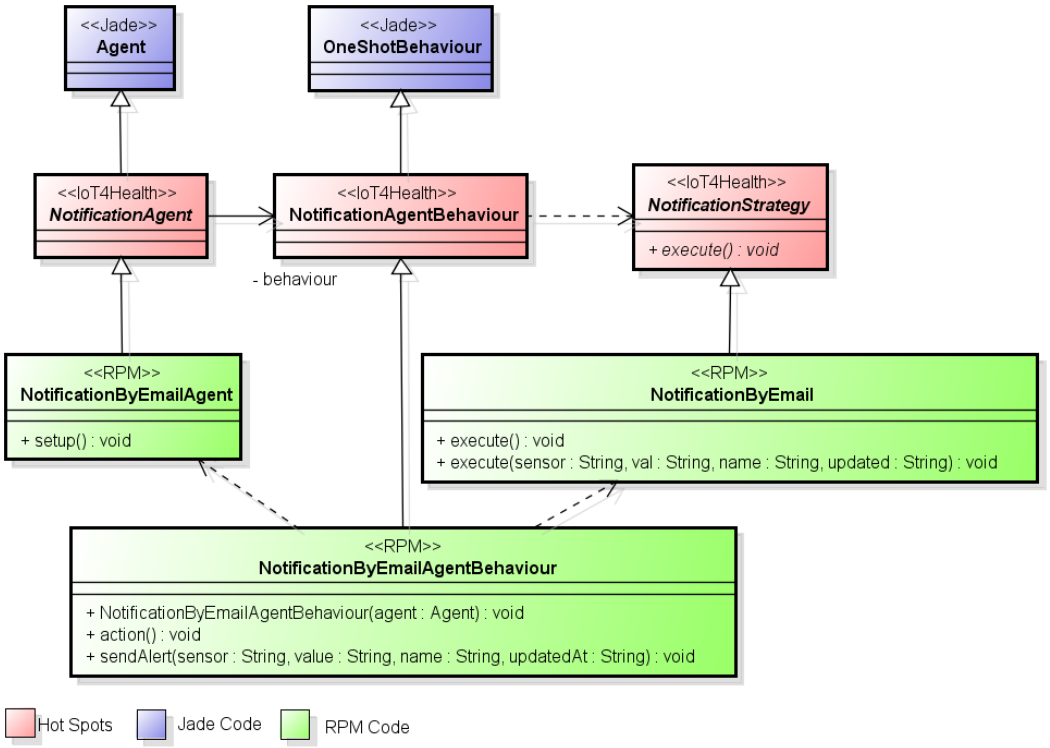


Figura 66 - Diagrama de Classes do Módulo de Notificação (M8) do RPM System.

7.5.3.
3ª Instância do IoT4Healh – EHealth System

7.5.3.1.
Arquitetura do EHealth System

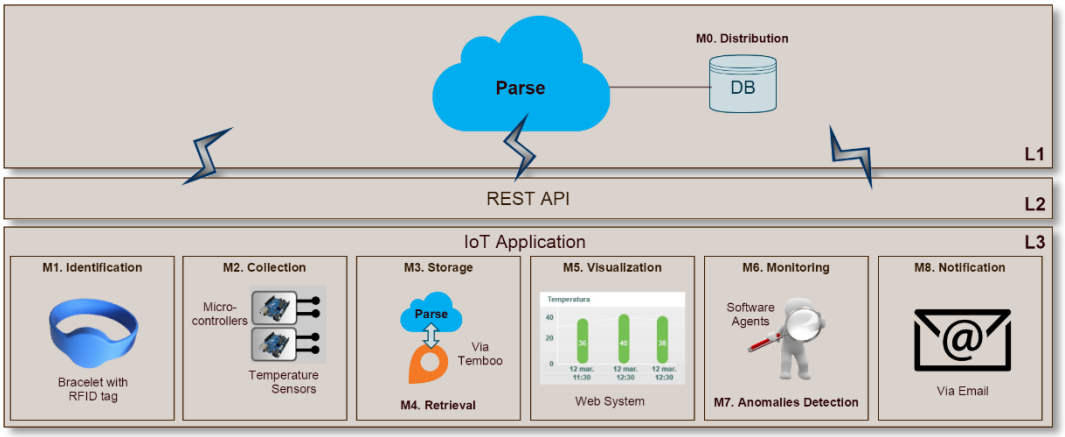


Figura 67 - Arquitetura do EHealth System com suas três camadas (L1-L3). A aplicação IoT (L3) interage com a plataforma baseada em nuvem (L1) através da API REST (L2).

7.5.3.2.
Camada de Distribuição de Dados

A distribuição dos dados dos pacientes no EHealth System também foi feita utilizando-se a plataforma Parse.

7.5.3.3.
Camada de Comunicação de Dados

Para comunicação de dados entre o Parse e o EHealth System utilizamos a API do Temboo para fazer as requisições HTTP para armazenamento e recuperação de dados dos pacientes.

7.5.3.4.
Camada de Gerenciamento de Dados

A tabela a seguir mostra, de modo resumido, a estratégia escolhida para implementação dos hot spots do framework, no que diz respeito aos módulos do EHealth System.

Tabela 7 - Estratégias escolhidas na implementação do EHealth System.

Módulo	Estratégia de implementação utilizada
Identificação	Feita através de tag RFID.
Coleta	Input de dados via Texto.
Armazenamento	Através do Temboo.
Recuperação	Através do Temboo.
Visualização	Dados de Temperatura visualizados a partir de sistemas web, sob a forma de line charts.
Monitoramento	Via agentes de software.
Detecção de Anomalias	Por meio de agentes de software.
Notificação	Via E-mail.

7.5.3.5. Documentação do EHealth System

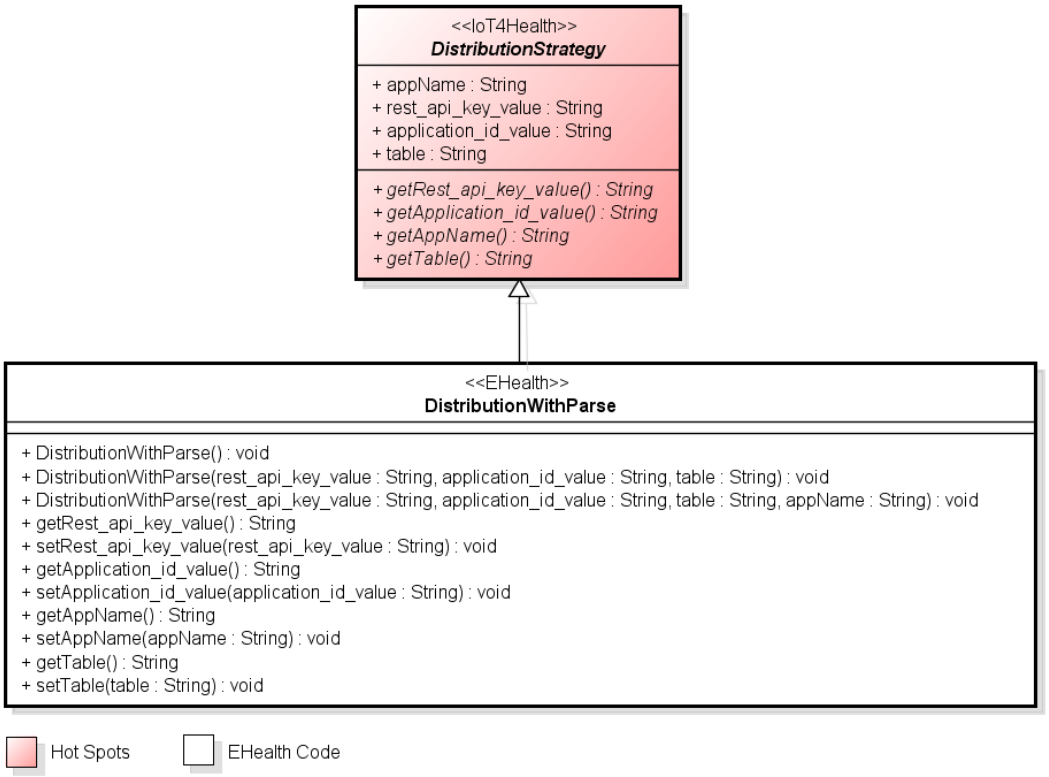


Figura 68 - Diagrama de Classes do Módulo de Distribuição (M0) do EHealth.

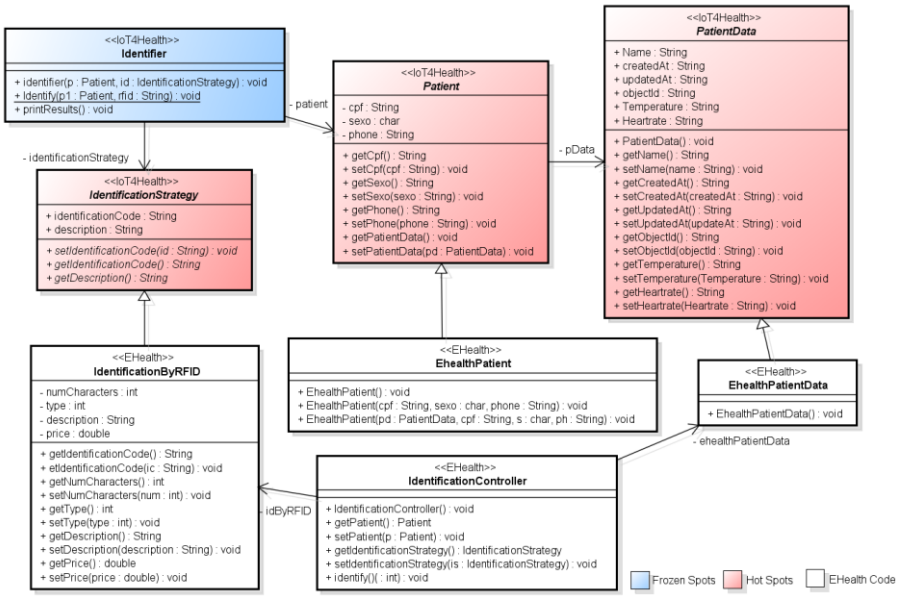


Figura 69 - Diagrama de Classes do Módulo de Identificação (M1) do EHealth.

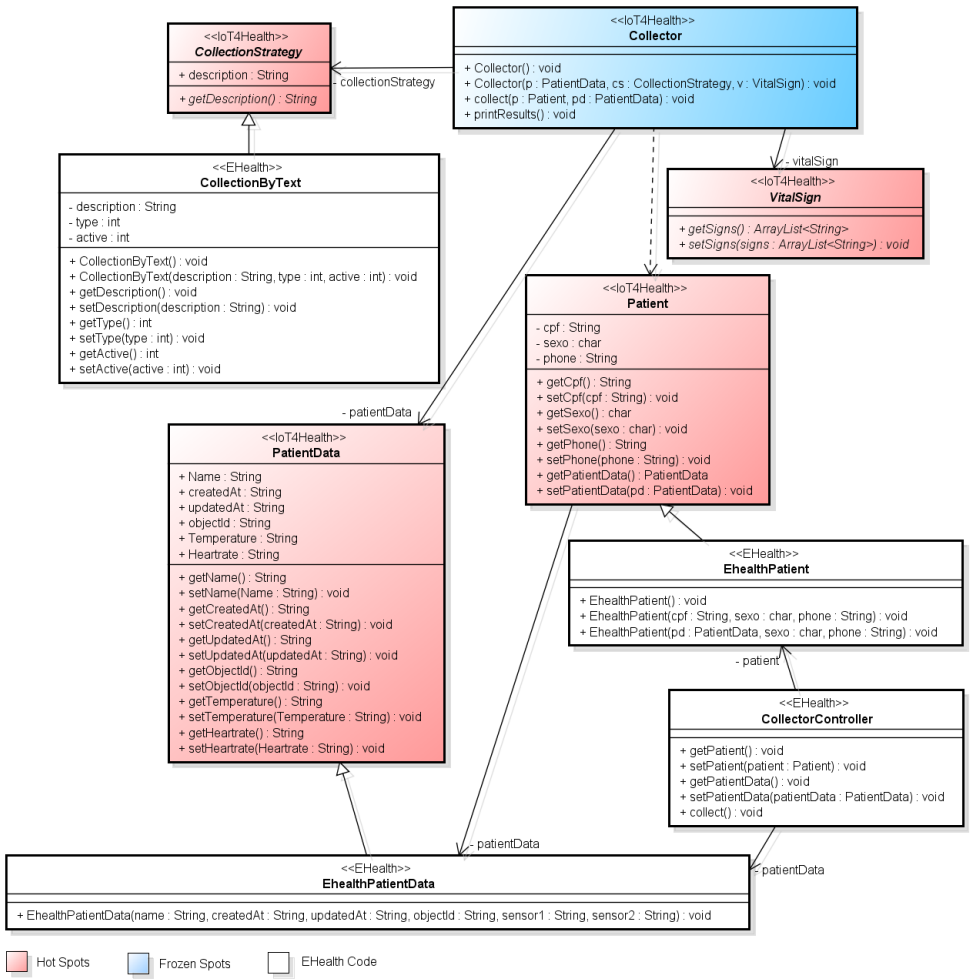


Figura 70 - Diagrama de Classes do Módulo de Coleta (M2) do EHealth.

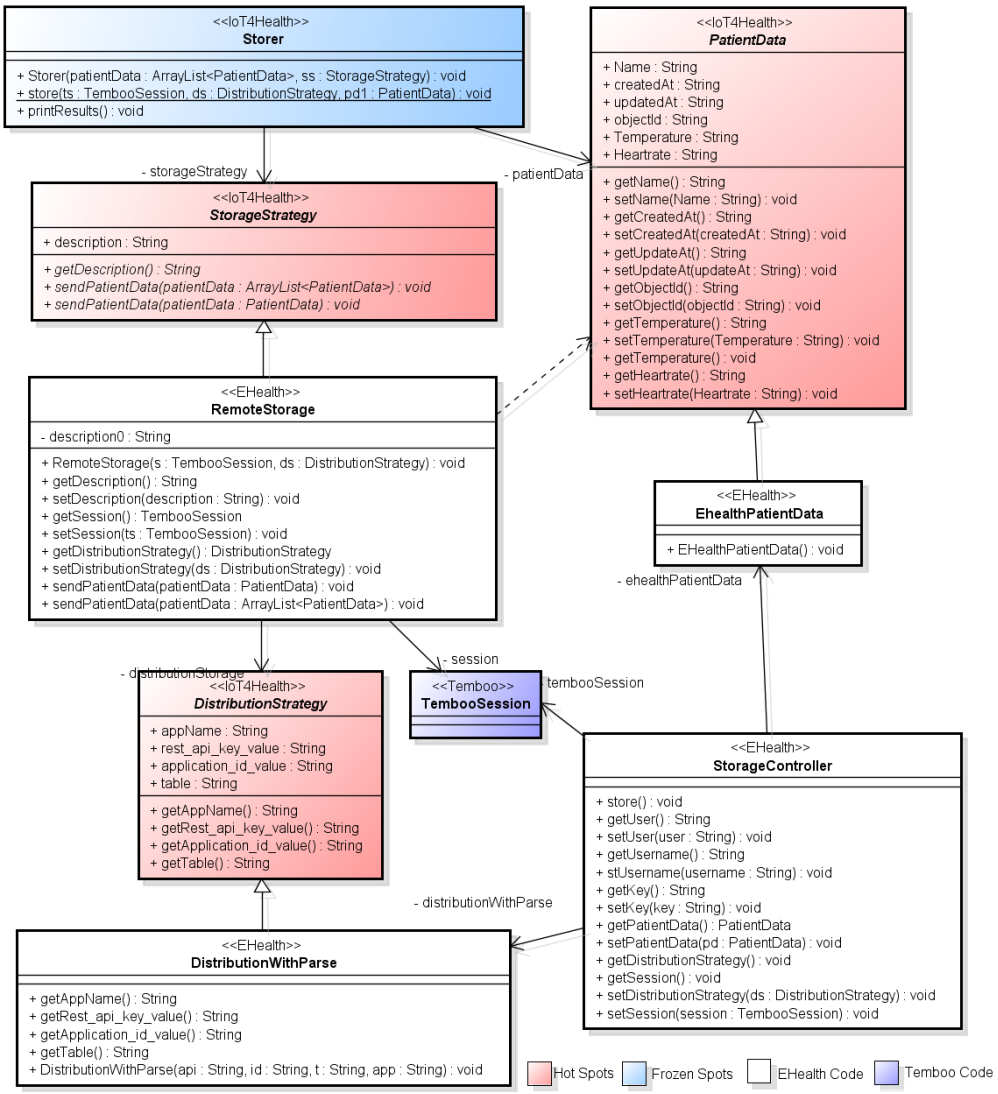


Figura 71 - Diagrama de Classes do Módulo de Armazenamento (M3) do EHealth.

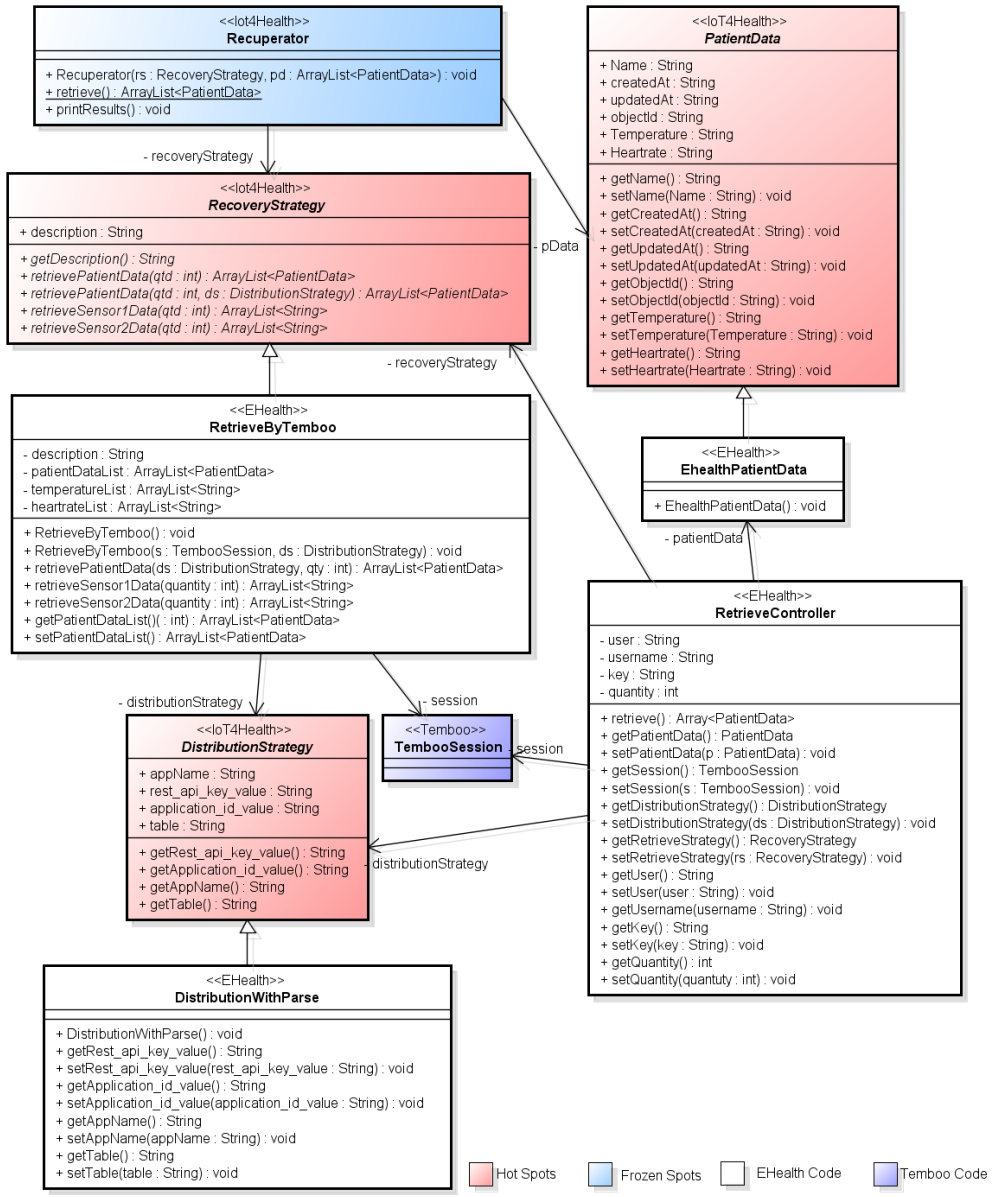


Figura 72 - Diagrama de Classes do Módulo de Recuperação (M4) do EHealth.

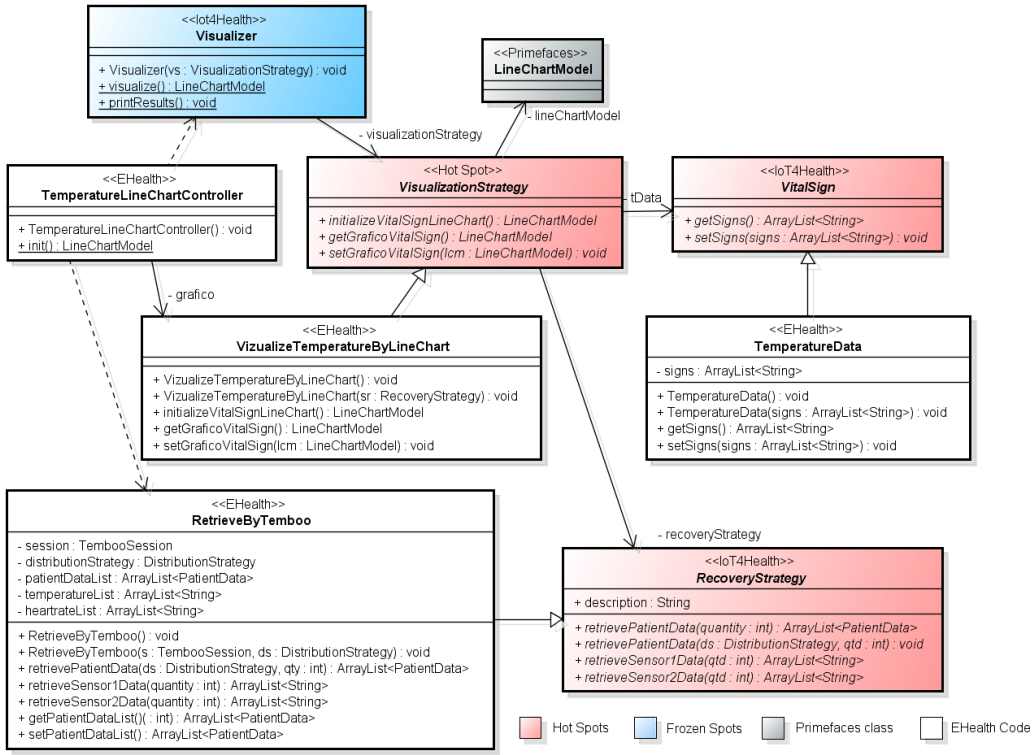


Figura 73 - Diagrama de Classes do Módulo de Visualização (M5) do EHealth.

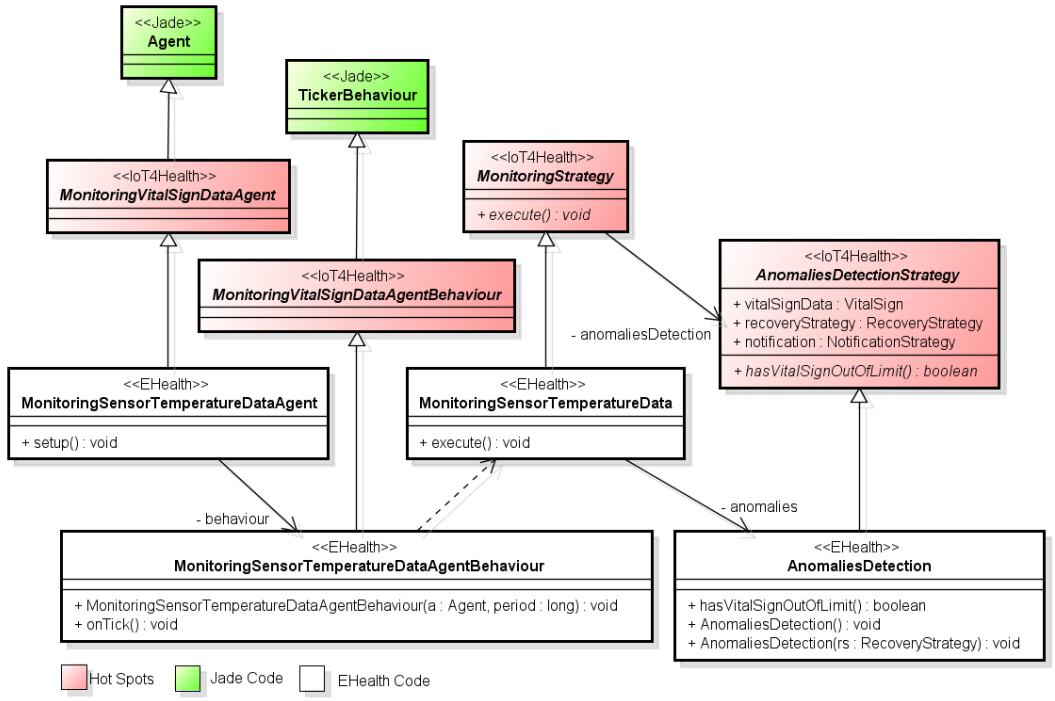


Figura 74 - Diagrama de Classes do Módulo de Monitoramento (M6) do EHealth.

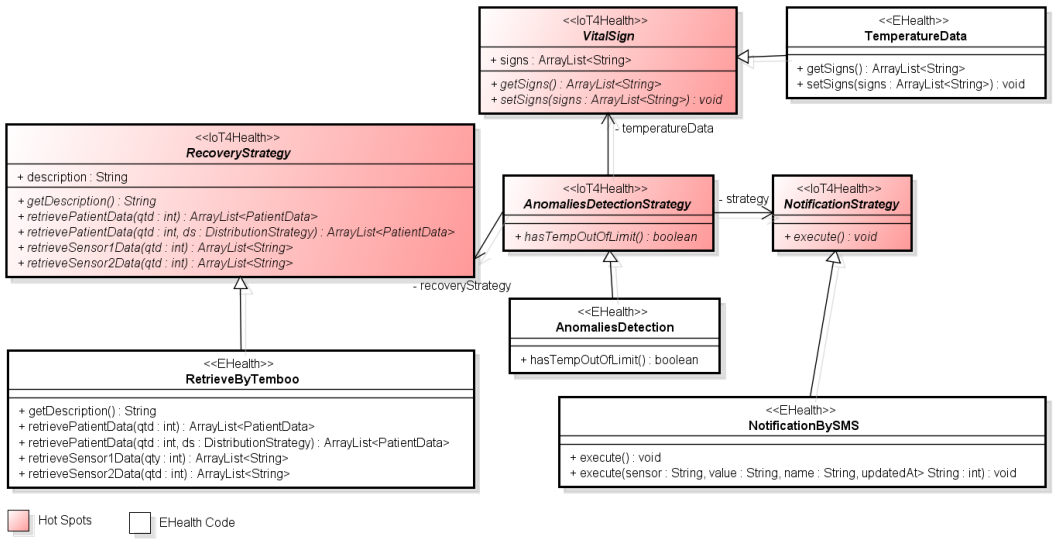


Figura 75 - Diagrama de Classes do Módulo de Detecção de Anomalias (M7) do EHealth.

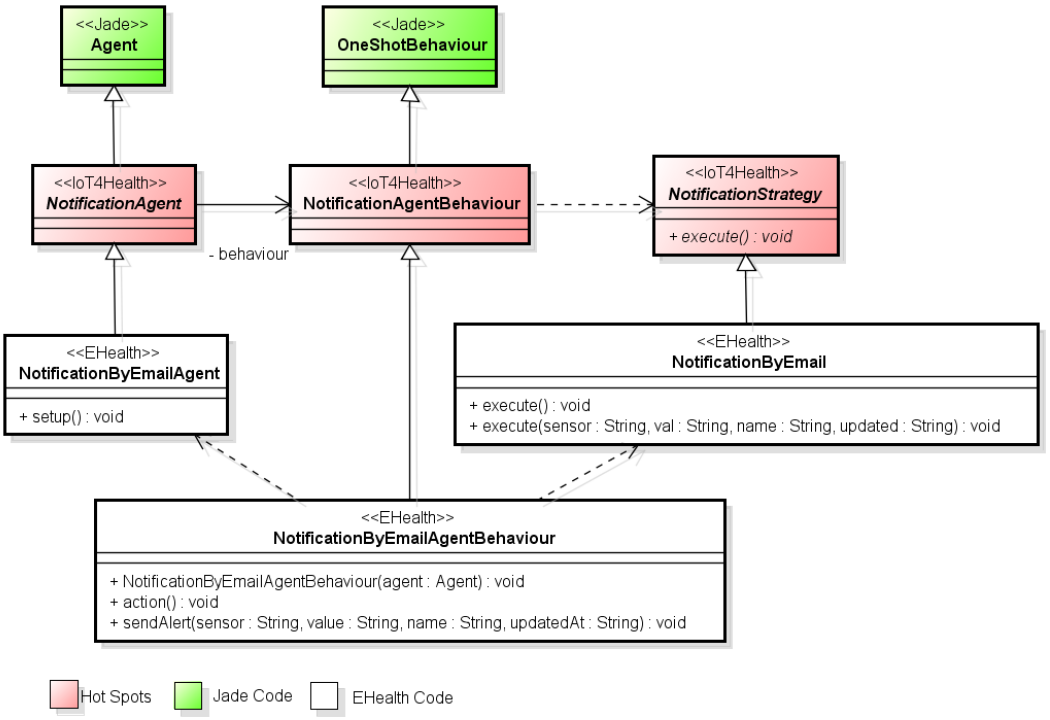


Figura 76 - Diagrama de Classes do Módulo de Notificação (M8) do EHealth.

8
Conclusões e Trabalhos Futuros

8.1.
Conclusões

O principal resultado deste trabalho é uma "teoria" sobre E-health, expressa sob um conjunto de módulos e organizada como frozen spots do nosso framework. O projeto e construção da ferramenta IoT4Health foi idealizado para dar suporte à atividade de projeto e desenvolvimento de aplicações no contexto de healthcare.

Aplicações construídas a partir de nosso framework irão lidar com problemas no domínio de E-health para suportar atividades como acompanhamento de pacientes. Elas podem fazer uso das funcionalidades definidas nos dez módulos da camada L3 da arquitetura do IoT4Health: 1-Identificação; 2-Coleta; 3-Armazenamento; 4-Recuperação; 5-Visualização; 6-Monitoramento; 7-Detecção de Anomalias; 8-Notificação; 9-Negociação por Recursos e Reconfiguração Dinâmica.

O IoT4Health oferece onze pontos de extensibilidade, de modo que uma aplicação instanciada a partir dele possa ampliar a sua arquitetura.

Com relação às três aplicações que desenvolvemos para servir como provas de conceito do nosso framework, podemos citar alguns pontos interessantes que observamos e que podem servir como contribuição para a resolução de problemas comumente encontrados em ambientes hospitalares nos dias de hoje, como a predominância da reatividade, conforme foi introduzido anteriormente.

Na aplicação Agents4Health, podemos observar que ela contribuiu para tornar o ambiente em que o paciente recebe atendimento médico mais proativo, uma vez que o sistema foi capaz de detectar anomalias em tempo real e enviar alertas aos profissionais da saúde em tempo real e de forma autônoma, através de agentes de software. Profissionais responsáveis por agir em caso de anormalidades detectadas no quadro dos pacientes poderiam reagir à ocorrência de anomalias imediatamente em resposta a estes eventos, considerando a utilização de nossa

solução. Uma possível indicação de aumento de proatividade pode ser verificada através da mensuração do DAI.

Também percebemos que atividades comumente realizadas de forma manual pelos profissionais de saúde, como o monitoramento de pacientes, pode ser feito de maneira autônoma pelos agentes de software, o que representa economia de recursos envolvidos no tratamento de pacientes. Os agentes foram capazes de realizar seus objetivos, encontrando anomalias no estado de saúde dos pacientes e alertando os profissionais responsáveis, através de envio de mensagens SMS, de forma gratuita.

Outro benefício trazido pelo Agents4Health diz respeito ao processo de identificação de pacientes. A substituição de técnicas tradicionais de identificação pelo uso de dispositivos como braceletes contendo tags RFID tornam o processo de identificação mais eficiente. Nossa aplicação também melhora o processo de coleta de dados vitais dos pacientes, uma vez que substitui técnicas de coleta manuais por processos automáticos, através do uso de microcontroladores Arduino e sensores médicos. A Coleta automática também traz a vantagem de ser menos propensa a erros.

A distribuição de dados dos pacientes para a plataforma baseada em nuvem, permitindo que qualquer pessoa com acesso previamente autorizado possa visualizar estes dados também traz benefícios, como a possibilidade de profissionais externos trabalharem de forma colaborativa com a equipe médica local.

Considerando o estágio atual da nossa pesquisa, podemos responder as questões de pesquisa definidas da seguinte forma:

Questão Central: É viável a concepção de um framework para instanciação de aplicações IoT no domínio de E-Health para acompanhamento remoto de pacientes, utilizando-se agentes de software?

Conforme apresentado no capítulo anterior (Framework IoT4Health), vimos que foi possível o desenvolvimento do gerador de aplicações no domínio de E-Health. Desse modo, podemos concluir que as aplicações criadas como instâncias do IoT4Health servem para validar a ferramenta, uma vez que tais aplicações apresentaram-se como soluções satisfatórias para os problemas que o framework se propõe a lidar, considerando um ambiente simulado.

Outro benefício que foi possível observar durante a etapa de instanciação do framework diz respeito ao ganho de produtividade na utilização do IoT4Health para criação de aplicações no escopo da ferramenta. A primeira versão da aplicação Agents4Health foi desenvolvida quando o framework ainda não estava implementado e pronto para uso, tendo consumido cerca de 65 horas de desenvolvimento.

Já o desenvolvimento do EHealth System, que por sua vez foi inteiramente implementado através da instanciação do IoT4Health, consumiu aproximadamente 2 horas. Mesmo considerando-se o fato de o IoTHealth ser um framework caixa branca, onde seu processo de instanciação requer conhecimento de seu código, além do nosso inteiro domínio do código, percebe-se que existe um ganho real de produtividade ao se fazer a opção de utilizar o IoT4Health para desenvolver aplicações dentro do seu domínio de aplicação específico. Vale salientar também que a documentação fornecida para o IoT4Health e para suas três provas de conceito também facilita a geração de novas aplicações.

Primeira subquestão: Quais os benefícios da coleta automática de dados vitais de pacientes, realizada através de sensores?

Resposta: Considerando nossa abordagem para coleta automática de dados dos pacientes através de sensores, podemos listar os seguintes benefícios: a coleta de dados utilizando sensores é mais eficiente que o processo manual de coleta de dados, além de ser menos propensa a erros que o processo realizado por seres humanos; Economia de recursos médicos; Curva de tendência em relação à resposta individual a uma intervenção médica, como, por exemplo, resposta terapêutica a um novo medicamento introduzido; Fator psicológico em relação à segurança de estar sendo monitorado.

Segunda subquestão: Que tipo de anomalias podem ser detectadas automaticamente a partir de aplicações criadas como instâncias de nosso framework?

Resposta[30]: Nossa plataforma corresponde a uma ferramenta de suporte ao diagnóstico médico, atuando de forma proativa na detecção de anomalias. Como o sistema alerta o médico sobre a ocorrência de valores anômalos, o especialista poder trabalhar de forma mais proativa.

[30] Resposta elaborada juntamente com o médico cardiologista Dr. Bruno Azevedo.

Analisando os dados de frequência cardíaca do paciente apontados pelos agentes como anomalias, o especialista pode considerar que o paciente poderia apresentar: Bradiarritmias como bradicardia sinusal, bloqueio átrio-ventricular etc. Além de taquiarritmias, tais como fibrilação atrial, taquicardia supra-ventricular e taquicardia ventricular.

Com relação à temperatura, os valores anômalos detectados pelos agentes são analisados pelo especialista para descobrir se poderia tratar-se de quadros de hipotermia, hipertermia, febre, estados inflamatórios, infecções, dentre outros.

Terceira subquestão: Quais benefícios o monitoramento remoto de dados dos pacientes pode trazer?

Resposta[31]: Podemos listar diversos benefícios que o monitoramento remoto de dados dos pacientes pode trazer, tais como: economia de recursos; disponibilidade dos dados; compartilhamento de informações entre especialistas para ter uma segunda opinião; Facilidade em fazer backup dos dados; Possibilidade de se ter um histórico dos dados, o que poderia auxiliar inclusive na prevenção de doenças. Em pacientes que necessitam de monitorização, as curvas de tendência trariam informações valiosas sobre o comportamento fisiológico de cada indivíduo. Nesse sentido, a análise dos dados pode auxiliar o médico, uma vez que é capaz de antecipar a identificação de uma descompensação clínica eminente.

8.2. Trabalhos Futuros

Como continuidade da nossa pesquisa, estamos focando no tema *Smarter Healthcare*, de modo que nossos esforços de pesquisa estarão mais concentrados em investigar como podemos contribuir para que a prática da medicina seja feita de forma mais inteligente, diminuindo seu ônus e aumentando sua eficácia. Nesse sentido, visamos fornecer aos profissionais da saúde recursos computacionais inteligentes e de fácil usabilidade, que os permita realizar um atendimento médico personalizado, rápido e proativo e que sirvam como ferramentas de apoio à tomada de decisão.

Também iremos evoluir os agentes de software que utilizamos atualmente, para que tenhamos agentes mais inteligentes, capazes de tomar decisões de forma

[31] Resposta elaborada juntamente com o médico cardiologista Dr. Bruno Azevedo.

autônoma. Nesse sentido, consideramos utilizar agentes cognitivos para trabalhar colaborativamente no processo de alocação de recursos médicos, promovendo gerenciamento de recursos efetivo através do uso de técnicas de argumentação. Os agentes também irão fazer recomendações de qual paciente deve ser o próximo na fila de atendimento médico, com base na análise da gravidade do seu estado de saúde, em comparação com os demais pacientes. Os agentes também poderão realizar atividades de predições a respeito de condições de saúde dos pacientes que podem resultar em anomalias.

Temos ainda a pretensão de evoluir o trabalho desenvolvido até o momento para tratar de aspectos de segurança, considerando a proteção dos dados contra ataques, bem como garantir sua privacidade e confidencialidade. Também planejamos evoluir os protótipos desenvolvidos para a coleta de dados dos pacientes a partir de sensores, adicionando novos sensores médicos, de modo a permitir o acompanhamento de pacientes diabéticos, epilépticos, em processo de reabilitação motora, gestantes durante o seu pré-natal, dentre outros.

Como trabalho futuro, pretendemos também continuar integrando nossa pesquisa com outros projetos em desenvolvimento no LES[32] voltados à área da Saúde, de modo que novas aplicações possam ser desenvolvidas como instâncias do nosso framework e que, dessa forma, esta ferramenta possa continuar sendo validade e evoluída. Por último, planejamos realizar estudos de caso, aplicando as soluções E-Health em um ambiente hospitalar real.

[32] http://www.les.inf.puc-rio.br/

9
Referências Bibliográficas

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer networks, 54*(15), 2787–2805.

Doukas, C. (2012). *Building Internet of Things with the ARDUINO*. CreateSpace Independent Publishing Platform.

Fernandes, C. O., Lucena, C. J. P., Lucena, C. A. P., & Azevedo, B. A. (2016). Enabling a Smart and Distributed Communication Infrastructure in Healthcare. In Y.-W. Chen, C. Torro, S. Tanaka, J. R. Howlett, & L. C. Jain (Orgs.), *Innovation in Medicine and Healthcare 2015* (p. 435–446). Cham: Springer International Publishing. Recuperado de http://dx.doi.org/10.1007/978-3-319-23024-5_40

Fernandes, C. O., & Lucena, C. J. P. de. (2015a). An Internet of Things Application with an Accessible Interface for Remote Monitoring Patients. In *Design, User Experience, and Usability: Interactive Experience Design - 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part III* (p. 651–661). http://doi.org/10.1007/978-3-319-20889-3_60

Fernandes, C. O., & Lucena, C. J. P. de. (2015b). Construindo Sistemas Multi-agentes no Contexto de Internet das Coisas., 1–12.

Finkenzeller, D. K. (2010). RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, Third Edition, (2), 480.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Markiewicz, M. E., & de Lucena, C. J. P. (2001). Object Oriented Framework Development. *Crossroads*, *7*(4), 3–9. http://doi.org/10.1145/372765.372771

Mohammadzadeh, N., & Safdari, R. (2014). Patient Monitoring in Mobile Health: Opportunities and Challenges. *Medical Archives*, *68*(1), 57–60. http://doi.org/10.5455/medarh.2014.68.57-60

Norvig, P., & Russell, S. (2014). *Inteligência Artificial, 3ª Edição* (Vol. 1). Elsevier Brasil.

Rao, A. S., Georgeff, M. P., & others. (1995). BDI agents: From theory to practice. In *ICMAS* (Vol. 95, p. 312–319).

Su, C.-J., & Chu, T.-W. (2014). A Mobile Multi-Agent Information System for Ubiquitous Fetal Monitoring. *International journal of environmental research and public health*, *11*(1), 600–625.

Su, C.-J., & Wu, C.-Y. (2011). JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. *Applied Soft Computing*, *11*(1), 315–325.

Viitanen, J., Hyppönen, H., Lääveri, T., Vänskä, J., Reponen, J., & Winblad, I. (2011). National questionnaire study on clinical ICT systems proofs: Physicians suffer from poor usability. *International journal of medical informatics*, *80*(10), 708–725.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, *36*(7), 75–84.

Wooldridge, M. (1999). Multiagent Systems. In G. Weiss (Org.), (p. 27–77). Cambridge, MA, USA: MIT Press.