UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FELIPE ALBERTO DA SILVA NOGUEIRA

Monitoramento domiciliar de hábitos e ações de pessoas de terceira idade através de conceitos da Internet das Coisas

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Alexandre Carissimi

Porto Alegre
2018

“We shouldn’t be looking for heroes,
we should be looking for good ideas.”
— NOAM CHOMSKY

AGRADECIMENTOS

Agradeço primeiramente ao estado e ao povo brasileiro, que me possibilitaram de receber uma formação de alto nível e gratuíta. Pretendo me empenhar ao máximo para poder retribuir ao país esse investimento.

Aos meus pais, Eliane e Joaquim, que são a principal razão para eu estar onde estou hoje, que me deram todas as condições para eu ir atrás dos meus sonhos e que eu sei que sempre vão estar do meu lado.

Aos meus irmãos, Monique, Michele e Ângelo, que sempre foram amorosos e atenciosos e, apesar do meu jeito por vezes complicado, sempre me apoiaram. Especialmente ao meu irmão, que, apesar de não estar mais nesse mundo, eu tenho certeza que estaria orgulhoso de mim, assim como eu tenho orgulho dele.

Ao meu orientador, Alexandre Carissimi, que teve toda paciência, disponibilidade e conhecimento que me permitiram concluir este trabalho.

A minha namorada, Queren, que me apoiou e me incetivou, estando do meu lado nos momentos difíceis e sempre querendo meu melhor.

Aos meus amigos, que são uma parte importantíssima da minha vida e que eu tive a sorte de ter encontrado.

E por fim a esta universidade, que foi responsável por uma parte essencial da minha formação acadêmica, profissional e pessoal, e onde eu passei momentos incríveis da minha vida que jamais irei esquecer.

RESUMO

Uma das principais preocupações que as pessoas têm quando alcançam a terceira idade é a capacidade de se manter independente em suas rotinas diárias, já que a idade avançada pode trazer problemas de saúde que podem transformar tarefas cotidianas em situações de risco, se não supervisionadas. Nesse contexto, os conceitos da Internet das Coisas podem ser usados para monitorar e medir atividades domésticas de pessoas idosas através de sensores estrategicamente instalados ao longo da casa. Este trabalho tem como objetivo discutir e propor uma solução para auxiliar idosos a serem supervisionados de forma segura, mas não demasiadamente intrusiva.

Palavras-chave: Internet das coisas, monitoramento domiciliar, inteligência ambiental.

ABSTRACT

One of the biggest concerns that people have when reach the old age is the capability to remain independent in their daily life routine, as the advanced age can bring health problems that may turn ordinary tasks in risk situations, if not supervised. In this context, the concepts of Internet of Things (IoT) can be used to monitor and measure home activities of elderly people through sensors strategically placed along the house. This paper intends to discuss and propose a solution to assist senior citizens to be supervised in a safe, but not overly intrusive, way.

Keywords: Internet of things, home monitoring, ambient intelligence.

LISTA DE ABREVIATURAS E SIGLAS

AmI	*Ambient Intelligence*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
BSON	*Binary JSON*
CDC	*Centers for Disease Control and Prevention*
CoAP	*Constrained Application Protocol*
DTLS	*Datagram Transport Layer Security*
EC2	*Elastic Compute Cloud*
FCM	*Firebase Cloud Messaging*
GND	*Ground*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
I^2C	*Inter-Integrated Circuit*
IETF	*Internet Engineering Task Force*
IoT	*Internet of Things*
IP	*Internet Protocol*
ITU	*International Telecommunication Union*
JSON	*JavaScript Object Notation*
MIT	*Massachusetts Institute of Technology*
MQTT	*Message Queuing Telemetry Transport*
NFC	*Near Field Communication*
ONU	Organização das Nações Unidas
PIR	*Passive Infrared Sensor*

QoS *Quality of Service*

REST *Representational State Transfer*

RFID *Radio-Frequency IDentification*

SCL *Serial Clock Line*

SDA *Serial Data Line*

SOA *Service Oriented Architecture*

SOM *Service-Oriented Middleware*

SPI *Serial Peripheral Interface*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

UML *Unified Modeling Language*

URI *Uniform Resource Identifier*

VCC *Voltage Common Collector*

WSN *Wireless Sensor Networks*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é contemplado o contexto no qual este trabalho foi idealizado. Primeiramente, é abordada a motivação inicial, mostrando dados e pesquisas da sociedade atual que corroboram com a relevância do tema proposto. Após isso, baseado nessa motivação, é explicado o objetivo do trabalho, definido mais concretamente o que será desenvolvido. Por fim, é dado um panorama geral da organização do texto.

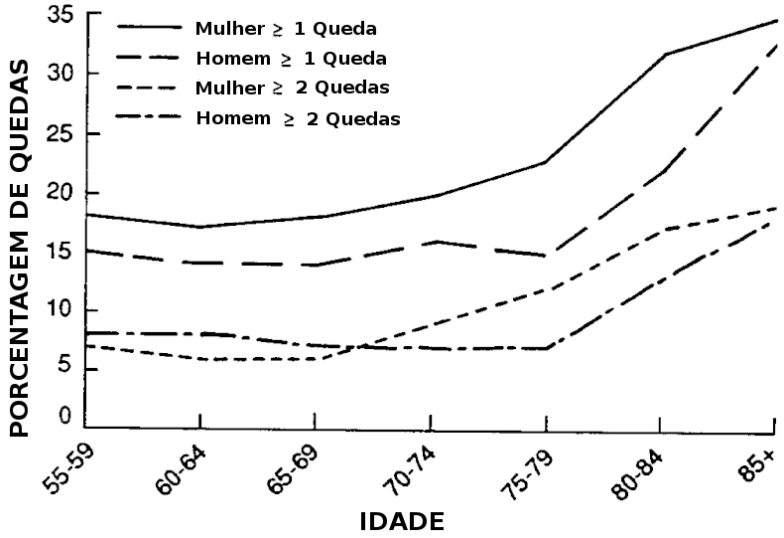
1.1 Motivação

Segundo a ONU (Organização das Nações Unidas), a raça humana presencia um envelhecimento populacional sem precedentes em sua história, e que tende a se acentuar em um ritmo ainda mais acelerado durante o século XXI. É estimado que, até 2050, cerca de 21% da população terá mais do que 60 anos, passando o número de 2 bilhões de pessoas. No Brasil, essa porcentagem é prevista em 29% (United Nations, 2015). Entre os principais fatores para essa mudança no perfil demográfico mundial estão o aumento da expectativa de vida, a redução da taxa de natalidade e as notórias melhorias nas condições sanitárias e de saúde.

Como consequência dessas projeções, a qualidade de vida de pessoas de terceira idade é cada vez mais uma discussão presente na sociedade. Além de já terem que conviver com as complicações naturais provenientes da idade avançada, essas pessoas acabam, muitas vezes, tendo sua liberdade e independência afetadas pela necessidade de monitoramento, e pelos riscos de morarem sozinhas. Essa condição é ainda acentuada quando se leva em conta que normalmente são pessoas que estavam há muito tempo acostumadas a não depender de ninguém.

Uma queda pode ser definida como involuntariamente ir ao chão, ou a alguma outra superfície de nível mais baixo, sem que seja o resultado de uma síncope ou de uma intensa força externa (AGOSTINI; BAKER; BOGARDUS, 2001). De acordo com dados publicados pela CDC (*Centers for Disease Control and Prevention*), quedas representam a principal causa de morte associada a acidentes não intencionais, e a nona causa de morte dentre todas as causas, em pessoas com mais de 65 anos (Centers for Disease Control and Prevention, 2015). Na Figura 1.1, é possível verificar dados de uma pesquisa norte-americana, onde cerca de um quarto das pessoas com mais de 75 anos reportaram algum tipo de queda no ano anterior, e que esse número tende a crescer com o avanço da idade.

Figura 1.1: Porcentagem de indivíduos idosos que reportaram quedas nos últimos 12 meses, agrupados por idade e sexo.



Fonte: (National Center for Health Statistics, 1987)

Em uma pesquisa feita com 6.329 participantes entre 2003 e 2004, foi constatado que pessoas na faixa etária entre 70-85 anos passam cerca de 67% de suas horas ativas em atividades sedentárias (MATTHEWS et al., 2008), correspondendo ao grupo de idade onde comportamentos sedentários são mais comuns. Esses resultados são compreensíveis quando se leva em conta as limitações da idade, porém eles também mostram que essas pessoas necessitam, mais do que qualquer outro grupo etário, de incentivos para superar essa barreira e ter uma rotina mais saudável.

O envelhecimento populacional reflete também os avanços na medicina. Juntamente com o avanço etário, surgem inúmeras patologias e, cada vez mais, as pessoas idosas têm a possibilidade de recorrer a medicamentos modernos visando o prolongamento de suas vidas. Sendo assim, um outro aspecto que merece atenção é a taxa de adesão dessas pessoas a esses medicamentos, visto que é essencial que os mesmos sejam tomados corretamente e na frequência prescrita pelo médico. Para se ter um parâmetro, um estudo acerca desse tema, com indivíduos com uma média de idade de 75 anos, foi realizado por (INSEL et al., 2006), onde 38% dos participantes tiveram uma adesão aos medicamentos menor que 85%. Existem diversas motivações para essa falta de adesão, como, por exemplo, esquecimento e não concordância com o tratamento, mas é de suma importância que ao menos o conhecimento dessa condição exista por parte de outras pessoas próximas.

Como esses dados reforçam, a necessidade de cuidado domiciliar é uma realidade na vida da maioria dos idosos. Esse cuidado pode ser dado por familiares ou pessoas próximas, porém os mesmos nem sempre possuem a disponibilidade necessária para esse

tipo de tarefa. Outra alternativa seria contratar um serviço terceirizado, o que acaba sendo custoso e, até mesmo, desconfortável, por envolver profissionais da saúde até então desconhecidos. É nesse contexto que surge uma oportunidade para o avanço tecnológico, juntamente com a interconectividade que a Internet nos traz, poder prover uma solução que melhore a qualidade de vida dessas pessoas e dê mais autonomia às suas rotinas diárias.

1.2 Objetivos

A instrumentalização do ambiente domiciliar pode propiciar um monitoramento controlado, dando independência ao idoso e tranquilidade aos seus parentes. Além disso, o uso dessas tecnologias também possibilita a coleta de dados sobre a rotina e hábitos diários, que podem servir para alertar sobre comportamentos não saudáveis ou identificar situações anômalas. Por exemplo, através de sensores poderiam ser detectadas quedas e outros eventos que necessitem de algum tipo de amparo imediato, notificando as pessoas responsáveis. Também seria possível analisar os dados coletados, com o intuito de reconhecer comportamentos de risco, como a não adesão de remédios ou padrões sedentários.

O objetivo deste trabalho é propor uma solução de monitoramento para pessoas de terceira idade, utilizando conceitos de Internet das Coisas (*IoT - Internet of Things*) e de Inteligência Ambiental (*AmI - Ambient Intelligence*). Essa solução utilizará sensores de movimento e aceleração, interconectados a um controle central, estrategicamente instalados ao longo do ambiente domiciliar e com o intuito de permitir uma supervisão remota, coletando dados sensíveis e detectando situações de risco. Esses dados serão enviados a um serviço na nuvem, e poderão ser acessados a distância por pessoas responsáveis pelo idoso. O principal foco é possibilitar esse monitoramento necessário sem afetar demasiadamente a independência e intimidade desses idosos.

1.3 Organização do texto

Este trabalho é organizado em seis capítulos, considerando esta introdução. Inicialmente, o Capítulo 2 contém uma fundamentação teórica dos conhecimentos necessários para a elaboração do projeto. Após isso, a proposta do trabalho e suas especificações são detalhadas no Capítulo 3. Na sequência, é descrita a fase de implementação e detalhamen-

tos técnicos no Capítulo 4. No Capítulo 5 são apresentados os resultados das avaliações e testes realizados. Finalmente, no Capítulo 6 é feita uma conclusão do trabalho.

2 FUNDAMENTAÇÃO

Toda a teorização e desenvolvimento deste trabalho são baseados principalmente no conceito de Internet das Coisas, pois este foi considerado o mais adequado para a realização do objetivo proposto no capítulo anterior. Apesar de uma solução de Internet das Coisas possuir diversas formas de ser implementada, já que não é uma tecnologia bem definida, mas sim um paradigma tecnológico, existem boas práticas e tentativas de padronização que podem ser utilizadas para nortear o desenvolvimento de uma solução do tipo. Esses aspectos técnicos e outros conceitos importantes para o contexto deste trabalho são aprofundados neste capítulo.

2.1 Internet da Coisas

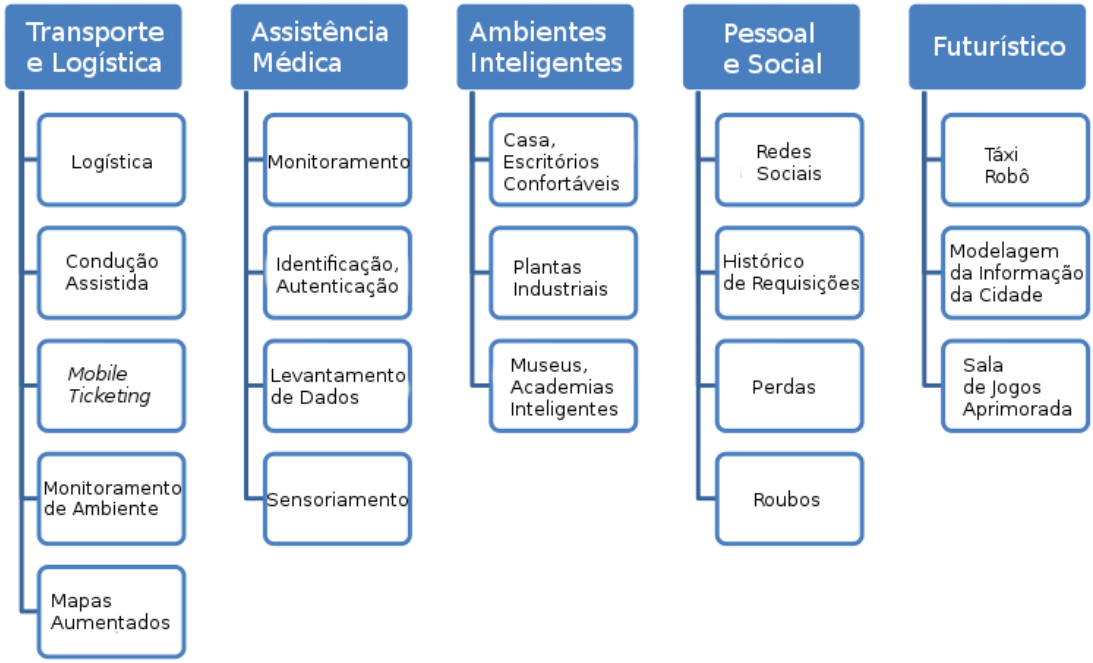
A Internet das Coisas é um conceito relativamente novo, porém vem se mostrando como uma das principais tendências tecnológicas do início deste século. Ainda não existe uma definição unânime para esse conceito, mas a ideia básica por trás é a de que a diversidade de objetos, ou coisas, em volta de nós – como RFID (*Radio-Frequency IDentification*), sensores, atuadores, celulares, veículos, etc. – seja capaz de interagir entre si e cooperar com seus vizinhos para alcançar um objetivo em comum (ATZORI; IERA; MORABITO, 2010). Por exemplo, a ITU (*International Telecommunication Union*) define a Internet das Coisas como "uma infraestrutura global para a sociedade da informação, que possibilita serviços avançados através da interconexão (física e virtual) das coisas, baseados em tecnologias interoperáveis, existentes e em evolução, da informação e da comunicação"(ITU, 2012).

O termo *Internet of Things* pode ter suas origens rastreadas até o grupo de pesquisa Auto-ID Center, do MIT (*Massachusetts Institute of Technology*). Fundado em 1999, esse grupo trabalhava na área de redes interligadas por identificadores de radiofrequência (RFID). O Auto-ID Center consistia de sete instituições de pesquisa localizadas em quatro continentes, que foram selecionadas para projetar a primeira arquitetura do que viria a ser conhecido como Internet das Coisas (EVANS, 2011).

Desde então, a abrangência da Internet das Coisas vai muito além do escopo de tecnologias RFID. Atualmente, suas aplicações são inúmeras e extremamente diversificadas, se estendendo a virtualmente todas as áreas do nosso dia a dia. Pode-se citar, por exemplo, aplicações na área da domótica, que se refere a automação do ambiente domici-

liar; de fábricas inteligentes, que também é conhecida como Indústria 4.0; da assistência médica, ajudando no monitoramento de pacientes; de cidades inteligentes, coletando dados a respeito do tráfego, poluição, transporte, etc., entre outras aplicações e conceitos que derivam de tecnologias IoT (WORTMANN; FLüCHTER, 2015). A Figura 2.1 ilustra esses diferentes domínios de aplicações existentes.

Figura 2.1: Domínios de aplicações e cenários mais relevantes.



Fonte: (ATZORI; IERA; MORABITO, 2010)

2.1.1 Visão Geral

A palavra "Internet", no termo Internet das Coisas, pode ser interpretada de duas formas. No sentido metafórico, expressando a ideia de que, da mesma forma que passamos a maior parte do tempo conectados a *Web*, em um futuro próximo as coisas também estarão interconectadas entre si, utilizando serviços e fornecendo dados. Já no sentido técnico, existe a visão de que o protocolo IP será usado por objetos físicos, ou seus respectivos *proxies*, para se comunicarem entre si, sendo literalmente incorporados a Internet (VINEELA; RANI, 2015).

No âmbito de uma visão mais técnica, esses objetos físicos devem possuir três principais habilidades: serem identificáveis univocamente, serem capazes de se comunicar e serem capazes de interagir, seja entre si ou com outras entidades da rede (MIORANDI et al., 2012). É baseado nessa visão que aplicações e soluções, antes não ima-

ginadas, estão sendo desenvolvidas, impactando diretamente a vida de diversas pessoas e também criando desafios no âmbito tecnológico.

No nível de componente, a Internet das Coisas é constituída, como o próprio nome diz, por coisas. Essas coisas podem ser vistas como objetos inteligentes. Segundo (MIORANDI et al., 2012), um objeto inteligente pode ser definido como uma entidade que possua:

- Um corpo físico, com tamanho, formato, etc.
- Um conjunto mínimo de funcionalidades para comunicação, como a capacidade de receber e responder mensagens.
- Um identificador único.
- Associações a, pelo menos, um nome e um endereço, de forma que o objeto possa ser encontrado na rede.
- Capacidades computacionais básicas, podendo ser desde a habilidade de receber mensagens a, até mesmo, realizar operações mais complexas.
- Pode possuir formas de sentir fenômenos físicos através de sensores, ou de executar ações que tenham efeitos na realidade física através de atuadores.

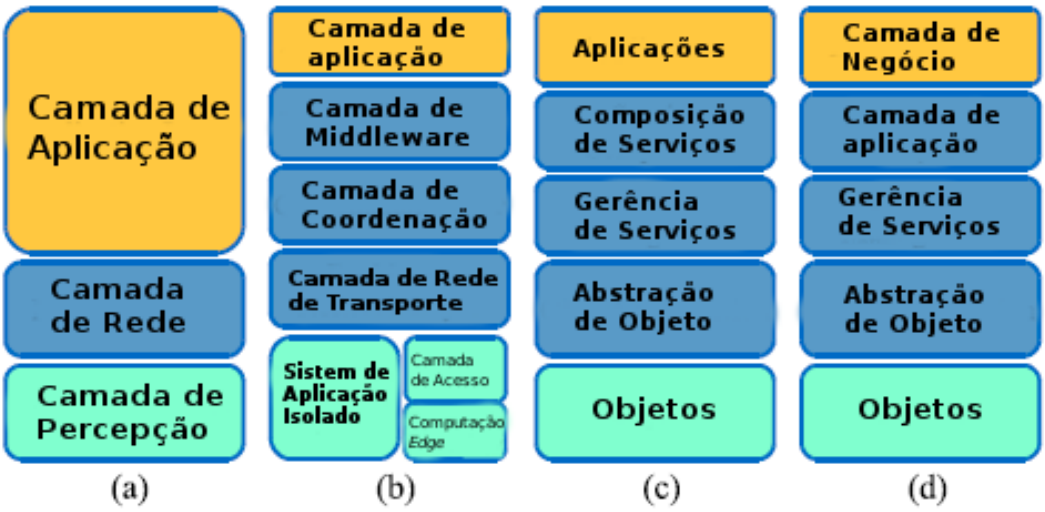
A inserção desses objetos inteligentes na Internet cria diversos desafios no aspecto de manter a interoperabilidade da rede. Vários desses componentes possuem capacidades computacionais restritas, e, muitas vezes, não podem operar com todas as camadas de protocolos que sistemas finais operam. Sendo assim, a Internet das Coisas revela uma rede com foco em informações do mundo físico, altamente dinâmica e distribuída, com entidades agindo como provedores e/ou consumidores, e não em comunicações fim-a-fim (MIORANDI et al., 2012).

2.1.2 Arquitetura

Não há um consenso a respeito da arquitetura para IoT, até mesmo por ser uma tecnologia ainda em nível de amadurecimento e consolidação. Existem alguns esforços no aspecto de criar definições e modelos de referência para esse tipo de arquitetura, partindo de um perspectiva de WSN (*Wireless Sensor Networks*), e se baseando em análises sobre as necessidades de pesquisadores e da indústria, como por exemplo o projeto IoT-A (IOT-A, 2012) e o SENSEI (SENSEI, 2008), este último conduzido pela União Europeia (GUBBI et al., 2009).

A Figura 2.2 ilustra alguns modelos baseados em camadas já propostos. Atualmente, o mais aceito é o de três camadas (WU et al., 2010), que oferece uma boa visão de alto nível da arquitetura IoT. As três camadas são: camada de percepção, camada de rede e camada de aplicação. O nível de complexidade de cada camada vai depender do contexto da aplicação e de suas necessidades.

Figura 2.2: Modelos de arquitetura IoT: (a) Três camadas (b) Baseado em *middleware* (c) Baseado em SOA (*Service-Oriented Architecture*) (d) Cinco camadas.



Fonte: (AL-FUQAHA et al., 2015)

A camada de percepção é a camada física, responsável principalmente por detectar outros objetos e coletar dados referentes ao meio em que o dispositivo está inserido (SETHI; SARANGI, 2017). Esse primeiro estágio de detecção pode se basear em tecnologias como RFID, WSN, GPS (*Global Positioning System*), NFC (*Near Field Communication*), etc., enquanto que para a coleta de dados podem ser utilizados toda uma diversidade de sensores. Além disso, a camada de percepção também é responsável pela conversão desses dados em sinais digitais, que são mais convenientes para a subsequente transmissão dos mesmos (ROMDHANI; ABDMEZIEM; TANDJAOUI, 2015).

A principal função da camada de rede é traduzir e transmitir a informação obtida pela camada de percepção. Assim que um sensor capta algum tipo de dado, ele, em seguida, o envia para um segundo plano, onde é feito o devido tratamento e transmissão desse dado, e isso se dá através da camada de rede. Nessa camada, se encontram mecanismos de gerência da rede, possibilitando a conexão e a comunicação com outros objetos e/ou sistemas finais. As principais mídias para essa comunicação incluem FTTx, 3G/4G, Wi-Fi, Bluetooth, ZigBee, infravermelho, entre outras.

Por último, a camada de aplicação, onde é oferecido aos usuários serviços específicos da aplicação em questão. É nessa camada que, ao interpretar e entregar funcionalidades, se cria valor para os dados gerados pelas outras camadas, o que pode ser visto como o objetivo final do desenvolvimento de softwares no contexto da Internet das Coisas (WANG; WANG; WANG, 2015). Esses serviços podem ser oferecidos em diversas plataformas, como *web* ou *mobile*, que podem estar hospedadas em uma estrutura na nuvem, por exemplo.

2.2 Protocolos

A comunicação na Internet é, em grande parte, possibilitada pela utilização de diversos protocolos padrões, que criam convenções para como seus *hosts* devem interagir entre si. As aplicações da Internet das Coisas estão sendo inseridas nesse meio e, sendo assim, nada mais intuitivo que usar parte dessa infraestrutura e convenções já existentes.

Segundo a *Internet Protocol for Smart Objects Alliance* (IPSO), o protocolo IP tem se mostrado uma tecnologia duradoura, estável e altamente escalável, sendo capaz de suportar uma grande escala de aplicações e dispositivos (DUNKELS; VASSEUR, 2008). Nesse contexto, o grupo de trabalho 6LoWPAN da IETF (*Internet Engineering Task Force*) criou uma série de especificações e mecanismos que permitem que pacotes IPv6 sejam carregados sobre redes baseadas no padrão IEEE 802.15.4, que define a operação de redes sem fio pessoais, de baixas taxas de transmissão, ideais para aplicações IoT. O 6LoWPAN foi originado no conceito de que o protocolo IPv6 deveria e poderia ser aplicado até mesmo nos menores dispositivos (MULLIGAN, 2006).

As aplicações na Internet são, geralmente, implementadas baseando-se no protocolo HTTP (*HyperText Transfer Protocol*). Porém, diferentemente do caso do protocolo IP, o protocolo HTTP não é considerado adequado para o contexto de ambientes com recursos limitados, como é o caso da Internet das Coisas, pois é demasiadamente verboso por natureza, o que acaba gerando um grande sobrecusto computacional (SETHI; SARANGI, 2017). Por conseguinte, diversos protocolos alternativos foram desenvolvidos visando resolver esse problema, como por exemplo, o CoAP, o MQTT, entre outros.

2.2.1 *Constrained Application Protocol* (CoAP)

O *Constrained Application Protocol* (RFC 7252) se assemelha ao próprio protocolo HTTP, porém possui otimizações e mecanismos que visam obter um bom desempenho em ambientes de recursos limitados. Ele opera em cima do UDP (*User Datagram Protocol*), e oferece funcionalidades como *multicast*, controle de congestionamento e troca assíncrona de mensagens.

O CoAP se baseia na arquitetura REST (*Representational State Transfer*), o que facilita o mapeamento com o protocolo HTTP (BORMANN; CASTELLANI; SHELBY, 2012). Os recursos no CoAP são identificados por URIs (*Uniform Resource Identifier*) e são acessados por métodos similares aos do HTTP: GET, POST, PUT e DELETE. Existem três tipos de códigos de resposta: 2xx (*success*), 4xx (*client error*) e 5xx (*server error*). Além disso, como o CoAP utiliza o protocolo UDP, ele tem a necessidade de definir seu próprio mecanismo de confiabilidade, disponibilizando mensagens de confirmação. O protocolo DTLS (*Datagram Transport Layer Security*) (RFC 6347) também é utilizado para atender requisitos de segurança, provendo privacidade, autenticidade e integridade.

Além do modelo *request/reply*, implementado pelos métodos HTTP, o CoAP também suporta o modelo *publish/subscribe*, que possibilita *multicasting*. Nesse modelo, um cliente se inscreve para ser atualizado quando um recurso tem alguma mudança, evitando assim o *overhead* de ter que ficar constantemente pedindo atualizações do estado desse recurso (VILLAVERDE; PESCH; ALBEROLA, 2012).

2.2.2 *Message Queuing Telemetry Transport* (MQTT)

O *Message Queuing Telemetry Transport* (OASIS, 2014) é um protocolo de mensagens leve, desenvolvido pela IBM ainda na década de 90. Foi originalmente projetado para operar em lugares remotos, em que seja necessário enviar dados através de redes com banda limitada. Por causa disso, o protocolo reúne diversos conceitos também pertinentes a redes em ambientes com recursos limitados, se tornando assim um protocolo adequado para aplicações IoT.

O MQTT executa em cima de TCP (*Transmission Control Protocol*) e utiliza um modelo *publish/subscribe* para troca de mensagens. Nesse modelo, um cliente pode publicar, ou se inscrever, para um determinado tópico, que é análogo ao conceito de URI.

A gerência do recebimento, enfileiramento e envio das mensagens é responsabilidade do *broker*, que basicamente é um *middleware* que opera como intermediário das mensagens enviadas. Fica a cargo dos *subscribers* se conectarem com o *broker* para receberem as mensagens das quais possuem interesse. O MQTT disponibiliza três níveis de QoS (*Quality of Service*) em suas conexões:

- QoS 0 (*at most once*): é um serviço de melhor esforço (*best effort*), onde as mensagens são enviadas no máximo uma vez, e nenhuma mensagem de confirmação é definida.
- QoS 1 (*at least once*): As mensagens são retransmitidas até que uma mensagem de confirmação seja recebida, o que pode causar que mensagens duplicadas cheguem ao destino.
- QoS 2 (*exactly once*): Nesse nível, além da garantia do recebimento da mensagem, também é garantido que apenas uma mensagem seja recebida pelo destino.

É importante salientar que é responsabilidade do cliente informar ao *broker* qual o nível de QoS desejado, o que possibilita que existam níveis de QoS diferentes para um mesmo tópico (HUNKELER; TRUONG; STANFORD-CLARK, 2008).

2.3 Middlewares

Um *middleware* tem a função de abstrair as complexidades de um sistema, ou de um *hardware*, para que o desenvolvedor possa focar inteiramente na solução final, sem ter que se preocupar com problemas não diretamente pertinentes ao escopo da aplicação (RAZZAQUE et al., 2016). Geralmente, essas complexidades são relacionadas a comunicação, ou a questões mais específicas do sistema. Em um meio tecnológico tão heterogêneo como o da Internet das Coisas, o uso de *middlewares* tem se tornado cada vez mais comum.

A arquitetura e as funcionalidades de cada *middleware* vão depender dos problemas e dificuldades que cada um se propõe a resolver. Entre os principais desafios encontrados estão: interoperabilidade entre dispositivos, descoberta de dispositivos e serviços que se encontrem na vizinhança, escalabilidade, *big data*, segurança e privacidade, compatibilidade com serviços na nuvem, interpretação de contexto para os dados coletados, etc. (SETHI; SARANGI, 2017).

Como dito acima, existem diversas questões a serem abordadas, dando espaço ao aparecimento de diversos tipos de soluções para *middlewares*, cada um com um foco diferente. Essas soluções podem ser classificadas em cinco grupos, de acordo com seu projeto (RAZZAQUE et al., 2016):

- Baseado em eventos: Nesse modelo, os componentes interagem entre si através de eventos. Cada evento possui um tipo e um conjunto de parâmetros específicos, cujos valores descrevem as mudanças no estado do produtor do evento. Os eventos são gerados pelos produtores e processados pelos consumidores, o que pode ser visto como um modelo *publish/subscribe*, onde entidades se inscrevem para receber notificações de um evento em particular.
- Orientado a serviço: Um SOM (*Service-Oriented Middleware*) é baseado na arquitetura SOA (*Service Oriented Architecture*), onde existem módulos independentes que oferecem serviços através de interfaces accessíveis. Os recursos são vistos como provedores de serviços e são oferecidos em um repositório de serviços, onde consumidores podem descobri-los e utilizá-los conforme a necessidade.
- Semântico: Esse tipo de *middleware* tem como principal objetivo a interoperabilidade entre os diferentes tipos de dispositivos. A ideia é criar uma semântica em comum na comunicação dos dispositivos, utilizando adaptadores para mapear os diferentes formatos de dados e, assim, possibilitar que recursos distintos se comuniquem entre si, independentemente do protocolo que utilizem.
- Orientado a banco de dados: Aqui, os componentes de rede dos dispositivos IoT são vistos como um sistema de banco de dados relacional virtual. Dessa forma, para acessar os recursos, as aplicações devem fazer requisições usando uma linguagem de consulta (*query language*).
- Específico para aplicação: Esse tipo de *middleware* é desenvolvido especificamente para algum tipo de aplicação. Nesse caso, existe um grande nível de acoplamento entre o *middleware* e a aplicação.

A escolha do tipo de *middleware* vai depender do teor e das necessidades da aplicação. Fazer uma escolha adequada pode significar uma grande economia de tempo durante a fase de desenvolvimento.

2.4 Inteligência Ambiental

O termo Inteligência Ambiental (*AmI - Ambient Intelligence*) foi originalmente proposto em 1998 em uma apresentação feita por Eli Zelkha e Brian Epstein, durante uma série de *workshops* encomendados pelo conselho de administração da empresa Philips (RUYTER; AARTS, 2004). Os *workshops* tinham como objetivo discutir possíveis futuros cenários, que pudessem vir a utilizar em larga escala a industria de eletrônicos da época.

AmI se refere a ambientes eletrônicos que sejam sensíveis e responsivos a presença de pessoas, e a eventos e condições externas. Isso geralmente é alcançado através de sensores e atuadores estrategicamente instalados ao longo do ambiente, e que sejam capazes de se comunicar entre si, ou a uma central de gerenciamento. Esses sensores podem ser implementados utilizando conceitos da Internet das Coisas, consolidando uma evidente possibilidade de integração entre os dois conceitos. Está também fortemente ligada ao conceito de computação ubíqua, que tem como objetivo tornar a interação homem-computador transparente, ao integrar a computação a rotina e aos afazeres das pessoas.

Essas definições da Inteligência Ambiental podem ser sumarizadas em seis principais características: sensibilidade ao meio, responsividade, adaptabilidade, transparência, comportamento ubíquo e inteligência (COOK; AUGUSTO; JAKKULA, 2009). As soluções em AmI não precisam necessariamente atender a todas essas características, mas são os principais aspectos que definem um ambiente inteligente.

Existem diversas oportunidades de aplicações para a Inteligência Ambiental, e a tendência é que, em um futuro próximo, esse tipo de tecnologia esteja presente em vários aspectos do cotidiano humano. Umas das oportunidades que recebem mais pesquisa e atenção é a assistência médica, dado o eminente envelhecimento da população. Porém, não é só baseado no conforto e bem estar de pessoas de terceira idade, o motivo para que esse tipo de aplicação receba tanta atenção. De acordo com (Commission of the European Communities, An i2010 Initiative, 2007), cerca de 1,5 bilhões de Euros poderiam ser economizados todo ano com a introdução de monitoramento remoto. Outro dado interessante é que, no Reino Unido, cuidados institucionais custam cerca de 21.840,00 libras por domicílio, enquanto que a utilização do *telecare* (assistência à distância a idosos) custaria apenas 7.121,00 libras.

Esse aprimoramento na qualidade de vida de pessoas idosas através da AmI pode ser alcançado através de diferentes abordagens. Uma delas poderia ser prover uma forma

de garantia de segurança, monitorando qualquer evento suspeito, como quedas, frequência de exercícios, dietas, etc., de forma a tranquilizar familiares próximos. Outra abordagem seria providenciar suporte a pessoas com problemas cognitivos, fazendo lembretes de tarefas e atividades importantes de suas rotinas. Também podem ser feitas avaliações a respeito do progresso dessas doenças cognitivas, como por exemplo avaliar a eficiência dessas pessoas na hora de realizar tarefas cotidianas. Por fim, tendo em mente que a idade avançada traz consigo muitas vezes o isolamento social, seria possível analisar as atividades sociais do dia a dia e, com esses dados, fornecer conselhos de como esse aspecto poderia ser aprimorado (COOK; AUGUSTO; JAKKULA, 2009).

2.5 Considerações Finais

Neste capítulo foram vistos os principais aspectos e considerações técnicas dos conceitos de Internet das Coisas e Inteligência Ambiental. Primeiro foi abordado uma visão mais geral do conceito, contextualizando seu surgimento e motivação. Após isso, o tema foi aprofundado para questões técnicas, como modelos de arquitetura, protocolos e *middlewares*. O próximo capítulo tem como objetivo propor uma aplicação para monitoramento remoto de pessoas de terceira idade, utilizando para isso esses conceitos vistos.

3 PROPOSTA

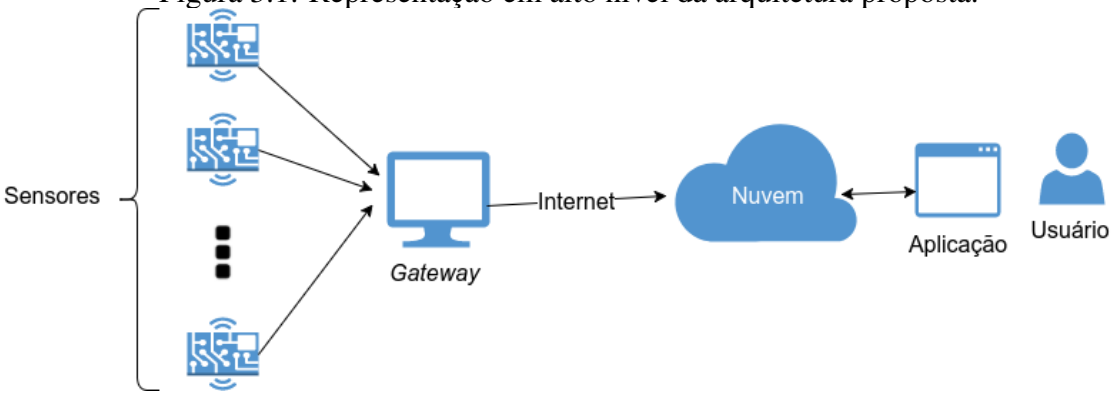
Antes de iniciar a etapa de implementação, é necessário a elaboração de uma proposta bem definida, levando em conta questões técnicas e funcionais, que possam auxiliar o desenvolvimento e evitar possíveis retrabalhos mais adiante. Neste capítulo é apresentada uma proposta para uma aplicação de monitoramento remoto, começando por uma visão geral da solução, e após isso a estruturando de forma mais detalhada através da coleta de requisitos funcionais e não funcionais, de um diagrama de casos de uso e, por fim, da prototipagem do aplicativo.

3.1 Visão Geral

A proposta deste trabalho é desenvolver uma solução que permita prover monitoramento domiciliar remota a pessoas de terceira idade, utilizando conceitos estudados de IoT e AmI. Basicamente, essa solução consistirá de sensores, estrategicamente instalados ao longo do domicílio do idoso, que irão coletar dados e fornecê-los através de um serviço na nuvem, que, por sua vez, será consumido por uma aplicação acessada por uma outra pessoa, possibilitando assim o monitoramento remoto.

Uma representação em alto nível da arquitetura proposta para solução está ilustrada na Figura 3.1 Essa arquitetura foi baseada no modelo de três camadas apresentado anteriormente na seção 2.1.2.

Figura 3.1: Representação em alto nível da arquitetura proposta.



Fonte: [Autor]

A camada de percepção é constituída pelos sensores, que têm a função de coletar dados sensíveis e de se comunicar com o *gateway*. Um acelerômetro será utilizado para a detecção de quedas, e deverá, portanto, estar instalado em alguma região do corpo da

pessoa monitorada, como o pulso, por exemplo. Sensores magnéticos instalados nas portas terão a serventia de detectar quando essas portas forem abertas. Essas portas podem ser tanto a principal da casa, possibilitando detectar quando o idoso saiu de casa, como a do armário de remédios, indicando assim quando, e se, os remédios estão sendo tomados corretamente. Também será utilizado um sensor de presença, com a finalidade de detectar momentos atípicos de inatividade, e de auxiliar a identificar quando o monitorado de fato saiu de casa. Para a comunicação desses sensores com o *gateway*, serão analisadas tecnologias, como Wi-Fi, Bluetooth e Zigbee, para se determinar qual é a mais adequada para a situação.

O *gateway* representa a camada de rede, e tem como principal objetivo transmitir os dados coletados pelos sensores para o serviço em nuvem. Esse *gateway* pode ser implementado por um computador pessoal existente no domicílio monitorado, ou até mesmo por uma placa de desenvolvimento, como uma Raspberry Pi, por exemplo. A comunicação com a nuvem será feita através da própria Internet, utilizando-se do protocolo IP.

A camada de aplicação é onde o monitoramento será efetivamente feito. Ela consistirá de uma aplicação *mobile*, que poderá ser acessada pela pessoa que deseja fazer o monitoramento e prestar assistência ao idoso, e de um servidor hospedado na nuvem, que será acessado tanto pelo *gateway* como pela aplicação *mobile*, devendo ser assim capaz de receber e fornecer dados. O modelo de comunicação que a aplicação utilizará para consumir os dados do servidor na nuvem poderá ser tanto o *request/reply* como o *publish/subscribe*, dependendo das necessidades e características da aplicação, que serão futuramente analisadas.

A aplicação *mobile* será responsável por mostrar os dados sensíveis captados pelos sensores em uma tela inicial, para que a pessoa que esteja realizando o monitoramento possa ter uma visão geral da situação atual de quem está monitorando. Além disso, essa aplicação também será responsável por disponibilizar uma interface de configuração para os hábitos da pessoa a ser monitorada, como por exemplo o horário de sono e o horário para se tomar os medicamentos, dados esses que deverão ser levados em conta pelo servidor na hora de realizar o tratamento dos dados recebidos pelos sensores.

O servidor hospedado na nuvem será responsável por toda a lógica de negócio. Será para ele que os sensores, através do *gateway*, enviarão os dados coletados na camada de percepção, assim como também será nele que esses dados vão ser tratados e armazenados. Além dessa comunicação com a camada de rede, o servidor também deverá fornecer uma API (*Application Programming Interface*) para a aplicação *mobile*, onde serão dis-

ponibilizados serviços para leitura dos dados relevantes da tela inicial da aplicação, e para leitura e escrita das configurações de preferência da pessoa a ser monitorada. E, por fim, o servidor também será responsável por detectar quando algum evento relevante tenha acontecido e enviar uma notificação à aplicação. Para que essa implementação seja possível, esse servidor deverá ser assíncrono, sendo capaz de receber dados dos sensores, responder as requisição da aplicação *mobile* e fazer constantes verificações nos dados até então recebidos para, se necessário, enviar uma notificação.

3.2 Engenharia de Requisitos

Buscando facilitar a fase seguinte de implementação, uma boa prática é o recolhimento dos requisitos da aplicação. De acordo com o *standard* publicado por (IEEE, 1990), um requisito pode ser definido como:

1. Uma condição ou capacidade necessária por um usuário para resolver um problema ou alcançar um objetivo.
2. Uma condição ou capacidade que precisa ser atendida, ou possuída, por um sistema ou um componente de um sistema para satisfazer um contrato, norma, especificação, ou outra forma de documento formalmente imposto.
3. Uma representação documentada de uma condição ou capacidade, como especificado em 1 ou 2.

Engenharia de requisitos é o ramo da engenharia de software preocupado com objetivos do mundo real para funções e restrições de sistemas de software, assim como também com a relação desses fatores para definir especificações de comportamentos dele (ZAVE, 1997). Ou seja, definindo as funções e as restrições de um sistema, também é possível definir o comportamente dele.

Segundo (PAETSCH; EBERLEIN; MAURER, 2003), o objetivo dessa etapa é ajudar a saber o que se vai construir, antes que o desenvolvimento do sistema comece, com o intuito de evitar possíveis custos de retrabalho. Esse conceito é baseado em duas premissas:

- Quanto mais tarde erros são encontrados, mais caro será para corrigi-los.
- É possível determinar um conjunto estável de requerimentos antes que a implementação e o *design* do sistema comecem a ser realizados.

Nesse contexto, os requisitos recolhidos são geralmente divididos em dois tipos: funcionais e não funcionais, detalhados nas próximas seções.

3.2.1 Requisitos funcionais

Requisitos funcionais são, de acordo com (IEEE, 1990), um requisito que especifique uma função que o sistema, ou um componente de sistema, deve ser capaz de realizar. Os requisitos funcionais definidos para a aplicação de monitoramento domiciliar proposto por este trabalho são os seguintes:

- A aplicação deve ser capaz de monitorar eventos importantes gerados pelos sensores, recebendo seus dados e os armazenando para posterior consulta.
- A aplicação deve ser capaz de mostrar os dados recentes mais revelantes ao usuário que estará realizando o monitoramento.
- A aplicação deve oferecer a possibilidade de configurar hábitos da rotina do monitorado, como horário de tomar os medicamentos ou horário de sono.
- A aplicação deve ser capaz de notificar o usuário realizando o monitoramento de qualquer evento relevante que tenha sido detectado através dos sensores, como quedas, inatividade, uso de medicamentos, ou entradas e saídas do domícilio monitorado.
- Os sensores devem ser configurados para apenas enviarem dados que sejam relevantes para o contexto da aplicação.
- O aplicativo *mobile* deve ser capaz de receber notificações mesmo que esteja sendo executando em *background*.

Os requisitos funcionais definidos fornecem uma visão geral de qual deve ser o comportamento do sistema e seus componentes. Eles servem também como base na elaboração do diagrama de casos de uso, que será apresentado na seção 3.3.

3.2.2 Requisitos não funcionais

Na área de requisitos de software, o termo requisito não funcional é usado para se referir a considerações que não são relacionadas à funcionalidade do sistema (CHUNG; LEITE, 2009). Esses requisitos podem se referir a uma variedade de especificações,

como: desempenho, compatibilidade de dispositivos, restrições do projeto, qualidade, segurança, entre outras. É importante definir esses requisitos antes da fase de desenvolvimento, pois eles podem afetar escolhas técnicas como banco de dados, linguagem de programação e sistema operacional (PAETSCH; EBERLEIN; MAURER, 2003), e essas escolhas seriam muito custosas de serem mudadas após o início do desenvolvimento.

Os requisitos não funcionais definidos para esse projeto são:

- A aplicação deve ser compatível com o sistema operacional Android.
- Devem ser utilizados apenas soluções em software livre e hardwares de baixo custo.
- A aplicação deve ser simples e intuitiva, se propondo apenas a mostrar as informações necessárias ao usuário.
- A aplicação deve ser tolerante a possíveis falhas de comunicação entre o servidor e os sensores, ou a aplicação *mobile*.
- O sistema deve funcionar 24 horas por dia e ser sempre capaz e emitir uma notificação assim que detectado o evento.
- A solucação deve ser confortável, para evitar que questões de usabilidade desencorajem seu uso.
- A solução deve ser de fácil instalação e configuração.

Esses requisitos não funcionais coletados têm um importante papel durante o desenvolvimento do sistema, pois servem como critério de seleção para a escolha entre as diversas alternativas de tecnologias e soluções disponíveis. Erros de omissão em levar esses requisitos devidamente em conta estão entre os mais caros e difíceis de serem corrigidos, uma vez que a fase de implementação do sistema tenha sido finalizada (CHUNG et al., 2012).

3.3 Diagrama de casos de uso

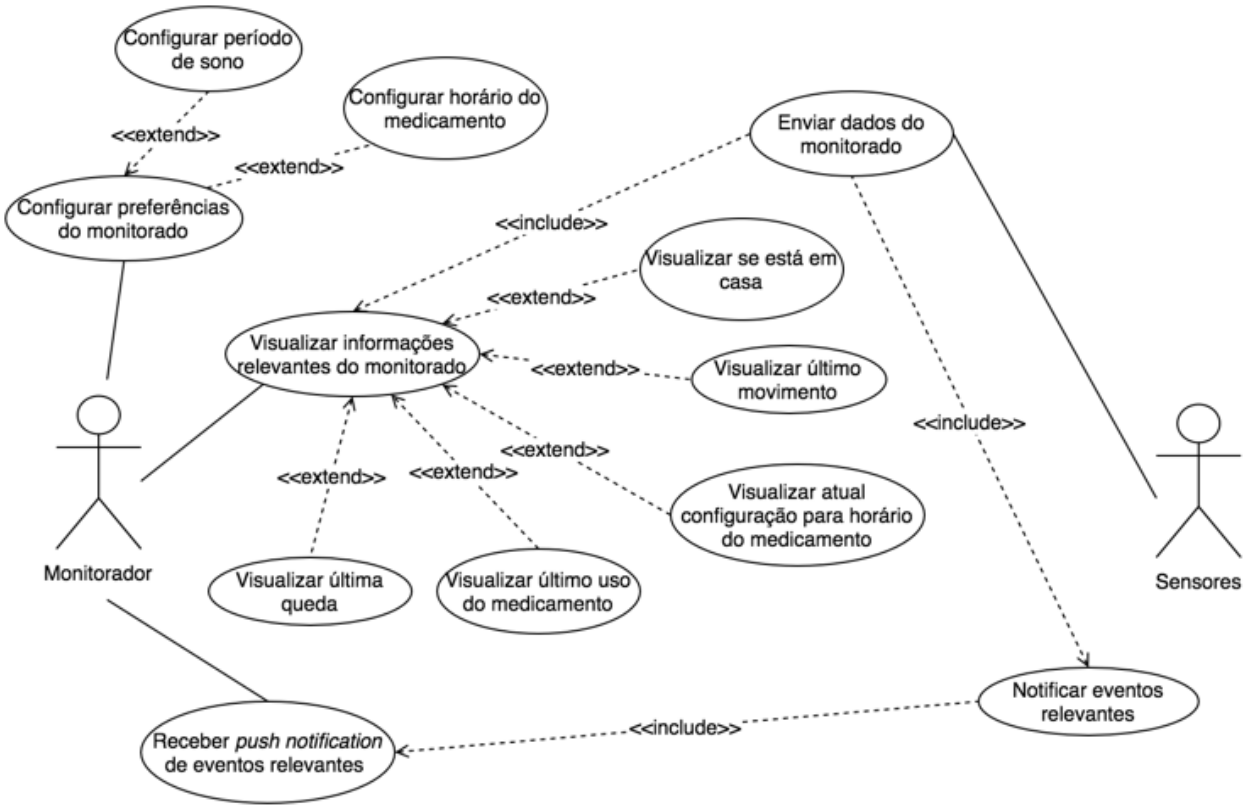
De acordo com (SOMÉ, 2006), um caso de uso descreve um pedaço de comportamento de um sistema sem revelar as estruturas internas desse sistema. Eles são úteis para capturar e documentar requisitos externos, assim como para validação de requisitos através de prototipagem. Nesse contexto, um ator pode ser definido como uma pessoa, ou um sistema, que tem um objetivo no sistema em discussão. Casos de uso podem ser dispara-

dos por esses atores, com a intenção de se atingir um determinado objetivo (BERTOLINO et al., 2002).

Um diagrama de casos de uso é uma forma de representar graficamente os atores, casos de usos e relações entre essas entidades. Esse diagrama é definido na linguagem UML (*Unified Modeling Language*) e têm o intuito de fornecer uma visão abstrata de alto nível do sistema. Nele, casos de uso e atores se interagem entre si através de relações, que podem ser tanto do tipo *include* como do tipo *exclude*. Relações do tipo *include* denotam a inclusão de um caso de uso como subprocesso de outro caso de uso (caso de uso base). Já uma relação *exclude*, denota a extensão de um caso de uso como uma adição de pedaços de comportamentos definidos em outros casos de uso (SOMÉ, 2006).

Na Figura 3.2 é apresentado um diagrama de casos de uso concebido para a aplicação proposta neste trabalho, onde se usou como base seus requisitos funcionais na elaboração dos casos de uso.

Figura 3.2: Diagrama de casos de uso.



Fonte: Autor

3.4 Prototipagem

Segundo (HSIA; DAVIS; KUNG, 1993), no contexto de engenharia de requisitos, prototipagem é a construção de um modelo de sistema com o intuito de aperfeiçoar o conhecimento sobre o problema. Dessa forma, prototipagem é uma ferramenta eficiente para se reduzir riscos em projetos de software, focando na viabilidade e recolhimento de requisitos, deixando assim mais claro para o desenvolvedor o que será desenvolvido e as dificuldades envolvidas nisso.

Se o desenvolvimento de um software é separado na construção de diversas camadas diferentes entre si, porém, constituintes de um sistema como um todo, se tem a possibilidade de definir o nível de prototipagem desejado. Seja, por exemplo, uma funcionalidade em particular e todas as camadas necessárias para o funcionamento da mesma, ou apenas uma camada isoladamente. É nesse contexto em que é definido por (BUDDE et al., 1992) a diferença entre prototipagem horizontal e vertical:

- Prototipagem horizontal: apenas camadas específicas de um sistema são construídas. Por exemplo, a interface de usuário, dando um foco maior na usabilidade e experiência de usuário, ou até mesmo funcionalidades *core* como as transações do banco de dados, nesse caso visando atender a aspectos técnicos e de desempenho.
- Prototipagem vertical: uma parte inteira do sistema é implementada na forma de um protótipo, incluindo todas as camadas necessárias para que essa parte seja funcional.

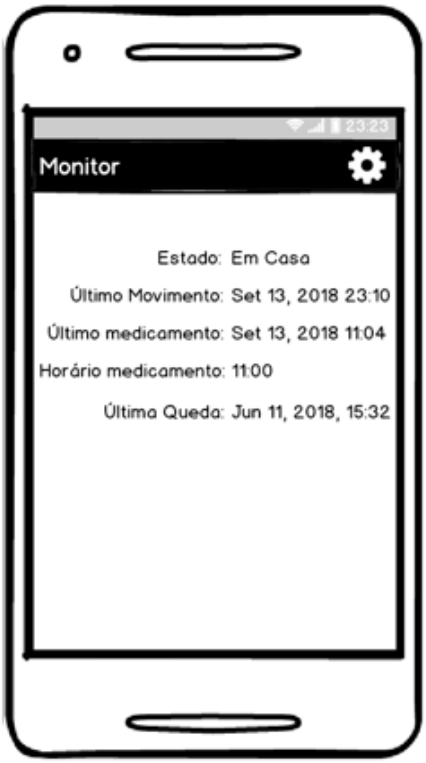
Para este trabalho, foi escolhida uma prototipagem horizontal da interface de usuário, pois o intuito é fazer uma modelagem em alto nível de uma aplicação capaz de atender os requisitos funcionais recolhidos, deixando a discussão técnica para a fase de implementação. As telas desenhadas também são chamadas de *mockups* e o fluxo de interação entre elas recebe o nome de *storyboard*. A seguir se encontra os protótipos criados, assim como uma explicação das funcionalidades atendidas por eles.

O aplicativo se divide basicamente em dois principais conjuntos: a tela de monitoramento e a tela de configuração das preferências. Na primeira, são mostradas as principais informações referentes a pessoa a ser monitorada, para uma rápida visualização do estado atual dela. Já na segunda, é possível que o usuário visualize e edite as configurações de preferências para a pessoa monitorada.

A Figura 3.3 ilustra a tela inicial de monitoramento, onde pode se encontrar as seguintes informações:

- Estado: se o monitorado no momento está em casa, com visita ou fora de casa.
- Último movimento: data e horário da última informação de movimento coletada dos sensores, que inclui desde um movimento brando, como apenas mexer os braços, até um mais substancial, como uma caminhada.
- Último medicamento: data e horário do último momento em que o monitorado tomou o medicamento.
- Horário medicamento: atual configuração para o horário de se tomar o medicamento.
- Última queda: data e horário do último evento de queda detectado.

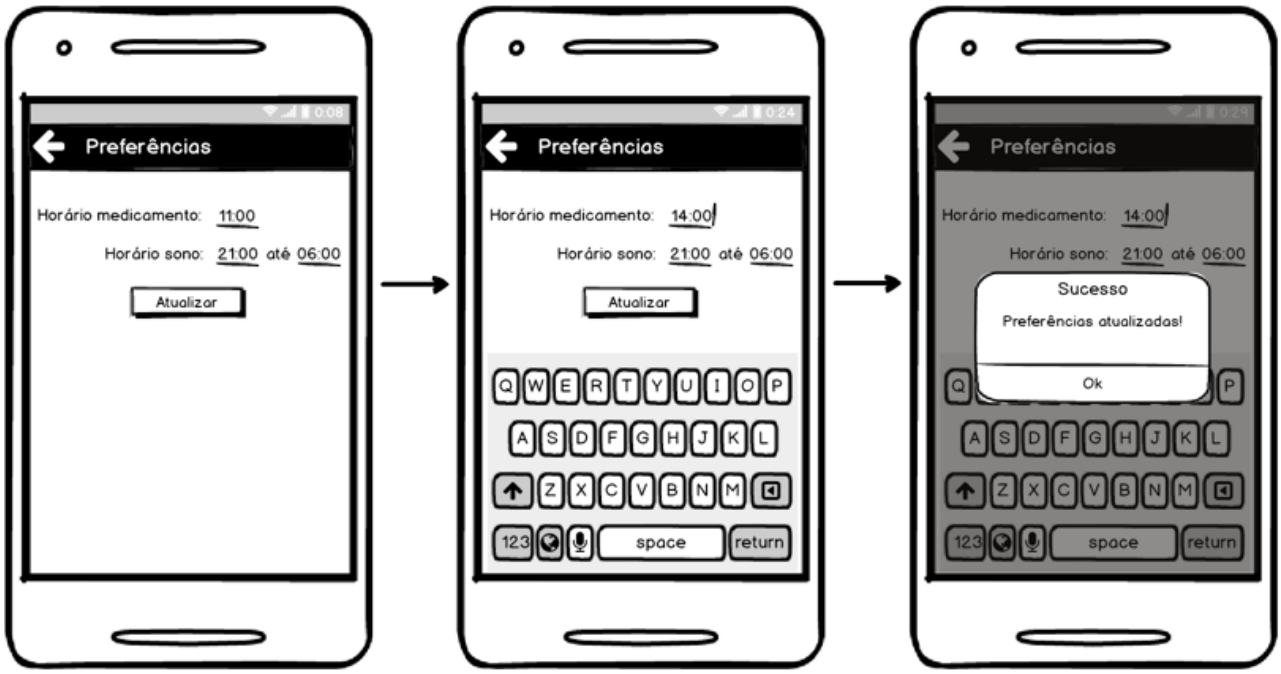
Figura 3.3: Protótipo da tela inicial de monitoramento.



Fonte: Autor

Além da tela inicial, é possível configurar os hábitos da pessoa monitorada, como o horário de sono e o horário do medicamento na tela de preferências. O horário de sono corresponde ao período no qual o monitorado está dormindo, informando a aplicação que durante esse intervalo a detecção de inatividade deve ser ignorada. Já o horário de medicamento se refere ao horário no qual o monitorado deve tomar seus medicamentos, no caso sendo possível apenas configurar um horário, que se aplica a todos os dias. Esse fluxo de configuração é mostrado na Figura 3.4.

Figura 3.4: Protótipos representando o fluxo para atualizar os hábitos da tela de preferências.



Fonte: Autor

Existem também as notificações, ou *push notifications*, que a aplicação poderá receber quando um determinado evento relevante aconteça. Essas notificações devem ser mostradas mesmo que a aplicação esteja executando em *background*. Como a interface do sistema de notificações é nativo do sistema operacional Android, não há muita liberdade de prototipagem, sendo usado o *design* já existente.

3.5 Considerações Finais

Neste capítulo foi apresentada uma proposta de solução para monitoramento remoto de pessoas idosas, assim como seus requisitos, funcionais e não funcionais, diagrama de casos de uso e os protótipos do aplicativo *mobile*. No capítulo seguinte será mostrada a implementação dessa solução, abordando questões técnicas, como escolha de técnologias e detalhes do desenvolvimento.

4 IMPLEMENTAÇÃO

É durante a etapa de implementação que as tecnologias a serem utilizadas são escolhidas, assim como a estratégia de implementação e a solução propriamente dita. Neste capítulo são discutidos esses aspectos do trabalho, apresentando detalhes da implementação e os motivos para determinadas preferências de tecnologia.

4.1 Arquitetura da Implementação

Como já abordado no Capítulo 3, a arquitetura da solução se baseia no modelo de três camadas, sendo representadas por um conjunto de sensores instalados estrategicamente pela casa monitorada, o modem local, que atua como *gateway*, e um servidor hospedado na nuvem, que, em conjunto com a aplicação *mobile*, representa a camada de aplicação.

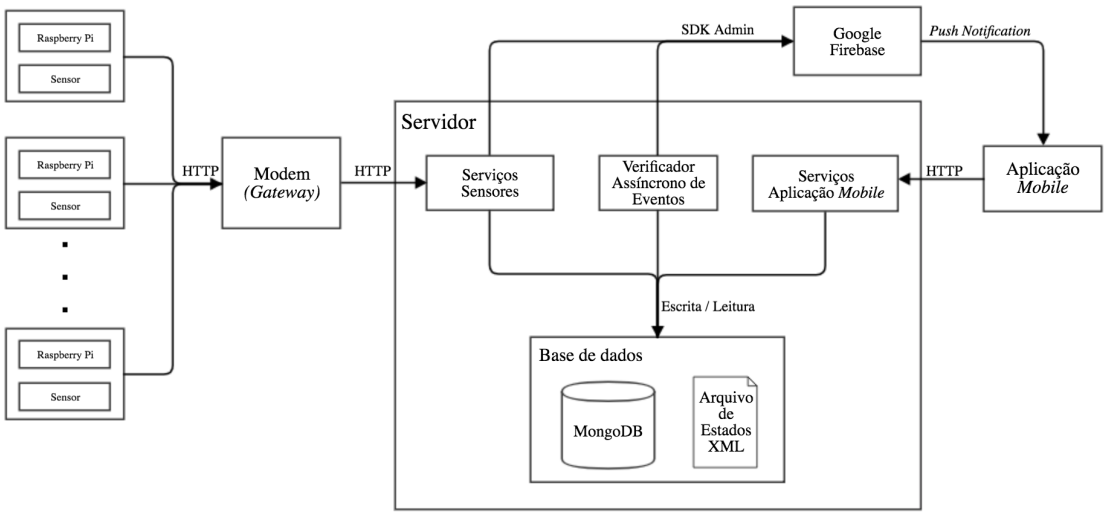
Cada um dos sensores é acoplado a uma Raspberry Pi (Raspberry Pi Foundation, 2018) executando um *script* em Python, que pode ser considerada como a inteligência do sensor, sendo capaz de processar os dados coletados e enviá-los para a nuvem, através da rede Wi-Fi fornecida pelo *gateway*. A alimentação da Raspberry Pi pode ser de duas formas, dependendo da particularidade da instalação. Através de uma fonte ligada diretamente na tomada, para os casos onde a instalação é estática e dentro do domícilio. Ou, utilizando uma bateria, adequado para o caso de quando o sensor tem que ser instalado no corpo da pessoa a ser monitorada.

O servidor hospedado na nuvem é desenvolvido em Node.js (Node.js Foundation, 2018), e tem a capacidade de receber os dados enviados pelos sensores, processá-los, e tomar as ações adequadas, como atualização de estados ou envio de notificações. Para isso, serviços de POST e GET são expostos para a comunicação com os sensores e com a aplicação *mobile*. O servidor também é responsável pela persistência dos eventos detectados, que é necessária para que a aplicação possa mostrar as informações relevantes mais recentes na tela inicial de monitoramento, criando a possibilidade uma análise do histórico de eventos. Além disso, um processo para verificação de eventos, como inatividade ou falta do consumo da medicação no horário correto, é executado em laço assincronamente dos outros serviços, enviando notificações através da plataforma Firebase da Google (GOOGLE, 2018), caso algum desses eventos seja detectado.

Por fim, a aplicação *mobile* é desenvolvida para Android, suportando as versões 7 e 8. Ela é a interface do usuário com os dados coletados, exibindo informações relevantes e permitindo a configuração de alguns parâmetros. A aplicação se comunica com o servidor através dos serviços expostos por ele, realizando tanto requisições de GET, para receber os dados a serem exibidos na tela, como de POST, para atualiar os valores das preferências.

A Figura 4.1 fornece uma visão geral mais detalhada da arquitetura da aplicação. Nela é possível ver os diversos componentes citados anteriormente e as interações que fazem entre si.

Figura 4.1: Arquitetura da implementação.



Fonte: Autor

4.2 Protocolo de comunicação

Uma importante decisão a ser feita no começo da etapa de implementação é a escolha dos protocolos a serem utilizados, tanto para a comunicação do sensor com o servidor, como para a comunicação da aplicação *mobile* com o servidor. Como já abordado no Capítulo 3, existem diversos protocolos desenvolvidos para as dificuldades normalmente encontradas nos ambientes limitados das aplicações IoT, que prezam pela economia de recursos como banda e bateria. Entretanto, o contexto da aplicação proposta neste trabalho é outro, já que o ambiente no qual os dados são coletados pelos sensores é a própria casa do monitorado, o que possibilita a garantia de uma boa conexão com a Internet através de um ponto de acesso como um modem, que também corresponde ao nosso *gateway* como explicado no capítulo anterior.

Levando em conta esse contexto de maior abundância de recursos, o protocolo de comunicação escolhido foi o HTTP. Essa escolha se deu por alguns fatores, como: maior familiaridade do autor com o protocolo, maior suporte de bibliotecas *open sources* devido ao fato de ser um protocolo amplamente utilizado nas mais diversas soluções e confiabilidade, pois opera em cima do TCP/IP. Como o ambiente de atuação dos sensores conta facilmente com Internet através de um ponto de acesso, o custo computacional causado pelo HTTP está sendo relevado, porém isso não seria necessariamente verdade para outras aplicações IoT.

A aplicação utiliza dois métodos HTTP na sua comunicação: o GET e o POST. O método GET é utilizado nas requisições onde se deseja ler algum tipo de dado, como quando a aplicação *mobile* exibe as informações da tela inicial de monitoramento, ou quando exibe os valores atuais das configurações de preferências. Já o método POST é utilizado nas requisições onde o intuito é o de escrever algum dado, como quando os sensores enviam dados recém coletados, ou quando o usuário atualiza os valores das preferências através do aplicativo.

4.3 Servidor

No servidor é onde se encontra a maior parte das regras de negócio da aplicação. É nele que os dados gerados pelos sensores são recebidos, processados e, caso necessário, notificações são disparadas. Além disso, o servidor também é responsável por receber requisições da aplicação *mobile*, tanto de leitura, para mostrar os dados mais recentes na tela inicial, como de escrita, para a atualização das preferências. Para isso, o servidor deve ser hospedado na nuvem e fornecer uma API para *hosts* externos, que correspondem a aplicação e aos sensores.

4.3.1 Escolha de tecnologias

Com o intuito de poder usufruir de todas as vantagens oferecidas pela tecnologia Node.js, a linguagem escolhida para o desenvolvimento do servidor é a linguagem JavaScript. Node.js é um ambiente de tempo de execução (*run-time environment*) para JavaScript que se utiliza da *engine* V8, desenvolvida originalmente pelo The Chromium Project para o famoso navegador Google Chrome. Tanto V8, como Node, foram de-

senvolvidos com foco em desempenho e baixo consumo de memória, mas enquanto o V8 suporta principalmente o uso da linguagem JavaScript no navegador, o Node tem o intuito de suportar processos em servidores de longo tempo de execução (TILKOV; VINOSKI, 2010). Existe uma grande variedade de bibliotecas *open source* disponíveis para Node, também chamadas de módulos, e que podem ser facilmente utilizadas através do gerenciador de pacotes chamado *npm*. Entre esses módulos, um que merece citação é o Express, que basicamente é um *framework* para aplicações *web* e que facilita o desenvolvimento de APIs, especialmente compatíveis com o protocolo HTTP. Além do Express, um dos principais *frameworks* para Node, outro grande motivador para a escolha dessa tecnologia, no contexto deste trabalho, foi sua natureza assíncrona, que facilita bastante a implementação de suporte a chamadas concorrentes e execuções paralelas. Outro fator também importante na escolha do Node.js foi o fato de sua execução ser *single-thread*, ou seja, apenas uma *thread* é responsável por toda a execução da aplicação, o que propicia uma grande economia de recursos computacionais, como memória RAM e processador.

Como o servidor necessita ser hospedado na nuvem, é preciso escolher de um serviço de hospedagem. A plataforma escolhida foi o AWS (*Amazon Web Services*), mais especificamente o serviço EC2 (*Elastic Compute Cloud*), que possibilita a locação de máquinas virtuais na nuvem onde é possível executar aplicações próprias. Apesar de existirem diversos provedores de qualidade, o da Amazon foi escolhido por oferecer um plano inicial grátis, que é suficiente para o contexto de prova de conceito deste trabalho, e ter um processo de configuração simples e bem documentado.

Uma importante funcionalidade da aplicação é o envio de notificação quando determinados eventos acontecem. Para isso, a forma mais fácil de se enviar essas notificações a dispositivos Android é através da plataforma Firebase, que pertence a Google. O Firebase oferece diversas funcionalidades, entre elas o FCM (*Firebase Cloud Messaging*), que permite o envio de *push notifications* de uma forma onde a maior parte da lógica é abstraída, deixando a cargo do desenvolvedor apenas as configurações referentes as notificações em si, como remetente e conteúdo, por exemplo. Foi escolhido um plano inicial gratuito da plataforma Firebase que atende as necessidades deste trabalho.

Por fim, para a persistência de dados foi escolhido o banco NoSQL MongoDB (MongoDB Inc., 2018). Bancos de dados NoSQL são bancos que não se encaixam na definição tradicional de bancos relacionais, e podem se referir a uma grande variedade de tipos. O MongoDB especificamente é um banco de dados *open source* orientado a documentos, que guarda seus dados na forma representações binárias chamadas BSON

(*JSON Binary*) em estruturas chamadas de coleções, que são o equivalente as tabelas do modelo relacional. Entre os principais motivos para essa escolha estão a maior flexibilidade em relação a modelagem do banco, visto que o conceito de colunas não existe mais, e a escalabilidade, já que o MongoDB foi projetado para ser escalável, possuindo seu próprio sistema de balanceamento de dados (CHODOROW, 2013). Além disso, a estrutura de documentos utilizada é expressa no formato JSON (*JavaScript Object Notation*), que já é nativo para o JavaScript, contribuindo assim para uma fácil integração entre as tecnologias. Outro aspecto interessante do MongoDB, que foi utilizado neste projeto, é a possibilidade de criar *capped collections*, que basicamente são coleções onde pode ser definido um tamanho máximo de memória utilizada. Foi definido um limite de 8GB, pois esse é o total de armazenamento disponível no plano gratuíto do AWS. Caso o banco chegue a esse limite, os registros mais antigos começam a ser deletados, para dar lugar aos novos inseridos.

4.3.2 Serviços para a aplicação *mobile*

Para que a aplicação *mobile* possa fornecer as funcionalidades requeridas, como mostrar informações relevantes e editar as configurações de preferências, o servidor deve disponibilizar uma série de serviços que possibilite essa interação da aplicação com os dados gerados pelos sensores. Esses serviços são tanto de leitura como de escrita e são descritos na Tabela 4.1.

Esses serviços são chamados pela aplicação de acordo com as interações do usuário com o aplicativo. O corpo das respostas do servidor é sempre em JSON e os valores retornados podem ser do tipo *Date* ou do tipo *String*. Serviços que retornam a data da última ocorrência de um determinado evento possuem o tipo *Date* que, além do dia, também contém o horário do evento. Já o serviço que retorna o estado atual da pessoa monitorada, e os serviços que retornam os valores dos preferências, são do tipo *String*.

Como os valores das configurações de preferências podem ser considerados como estados, não sendo necessário o armazenamento de um histórico de modificações, a estratégia de persistência é diferente da utilizada para os eventos recebidos pelos sensores. Em vez de armazenar esses valores no MongoDB, a persistência deles é feita em um arquivo XML (*Extensible Markup Language*) através de uma biblioteca *open source* para Node.js chamada xml2js, que basicamente é um *parser* utilizado para ler e escrever em arquivos XML. Além dos horários de sono e do medicamento, também é guardada a quantidade

Tabela 4.1: Caminhos disponíveis para os serviços usados pela aplicação *mobile*.

Caminho	Método HTTP	Descrição
/ws/mobile/home-status	GET	Retorna o estado atual do monitorado. Valores possíveis: • Em casa • Em casa com *n* visita(s) • Fora de casa
/ws/mobile/last-movement	GET	Retorna a data do último evento de movimento detectado. Pode ter sido tanto detectado pelo acelerômetro, como pelo sensor de presença.
/ws/mobile/last-medicine	GET	Retorna a data do último evento de abertura de porta do armário de medicamento.
/ws/mobile/last-fall	GET	Retorna a data do último evento de queda.
/ws/mobile/preferences	GET	Retorna todos os valores atuais de preferência.
/ws/mobile/preferences	POST	Atualiza os valores de preferência de acordo com os valores do *payload* recebido.
/ws/mobile/preferences/medicine-time	GET	Retorna o horário configurado do monitorado para tomar os medicamentos. O formato de hora retornado é *hh:mm*.
/ws/mobile/preferences/sleep-time	GET	Retorna o horário de começo e fim do tempo de sono do monitorado. O formato de hora retornado é *hh:mm*.

Fonte: Autor

de pessoas que se encontram na casa, que é atualizada pelo servidor de acordo com os eventos recebidos. Esse arquivo é estruturado para que cada *tag* contenha apenas um valor numérico, de forma que através de hierarquia do arquivo se possa chegar a informação completa. Por exemplo, para extrair o valor do horário de medicação, o servidor pega os valores contidos nas *tags* de *hour* e *minute*, que são filhas da *tag medicineTime*, que por sua vez é filha da *tag* raiz *states*.

4.3.3 Serviços para os sensores

O servidor deve fornecer uma forma pela qual os sensores possam enviar as notificações de eventos detectados. Para isso, o serviço HTTP POST para os sensores é exposto pelo servidor, que no recebimento das requisições processa o *payload* e toma as ações necessárias, seja ela notificar a aplicação *mobile* ou apenas persistir o evento no banco de dados. Esse serviço, exposto especificamente para o recebimento de dados dos sensores, é acessado pelo caminho */ws/sensors*. Dado que existem diferentes tipos de sensores, a

única forma do servidor identificar qual evento se trata a requisição é através da análise do *payload*, visto que todas as requisições são enviadas para o mesmo *endpoint*. O formato de dados escolhido para o *payload* é o JSON, principalmente por já possuir uma integração nativa com a linguagem JavaScript, e a sua estrutura ser relativamente simplesmente, com apenas três propriedades:

- *Type*: se refere ao tipo do sensor e pode ter os seguintes valores:
  - *Door*: Sensor magnético de porta.
  - *Passage*: Sensor de distância ultrassônico ou sensor de presença infravermelho.
  - *Movement*: Acelerômetro.
- *Info*: informação necessária para, combinado com o *type*, distinguir o tipo do evento, visto que o mesmo tipo de sensor pode se referir a diferentes eventos. Os valores possíveis são os seguintes:
  - *Main*: Porta principal da casa.
  - *Medicine*: Porta do armário de remédios.
  - *Hall*: Corredor principal da casa.
  - *Moving*: Ação de atividade detectada.
  - *Fall*: Ação de queda detectada.
- *CreatedDate*: a data de criação do evento, ou seja, a data correspondente ao momento em que o evento ocorreu.

A Tabela 4.2 mostra todas as possíveis combinações dos valores dessas duas propriedades e o seus significados. Caso a combinação enviada ao servidor não seja nenhum dessas (simbolizadas por um 'X' na tabela), ou o valor de alguma das propriedades não seja um desses listados, o servidor deve responder com um estado HTTP de erro, utilizando o código 400 para informar que a requisição não é válida, e descartar o evento recebido.

Cada evento recebido sempre é persistido no banco de dados, porém a maioria dos eventos específicos requerem algum tipo de processamento a mais, como enviar uma notificação à aplicação *mobile* ou atualizar o arquivo XML de estados. Apenas os eventos de detecção de atividade não exigem nenhum processamento imediato a mais, que são representados pelos pares *Passage*/*Hall* e *Movement*/*Moving*.

Tabela 4.2: Possíveis combinações de valores para o *payload* das requisições.

Info / *Type*	*Main*	*Medicine*	*Hall*	*Moving*	*Fall*
Door	Porta principal foi aberta	Porta do armário de medicamentos foi aberta	X	X	X
Passage	Alguém passou pela porta principal	X	Alguém passou pelo corredor	X	X
Movement	X	X		Movimento detectado	Queda detectada

Fonte: Autor

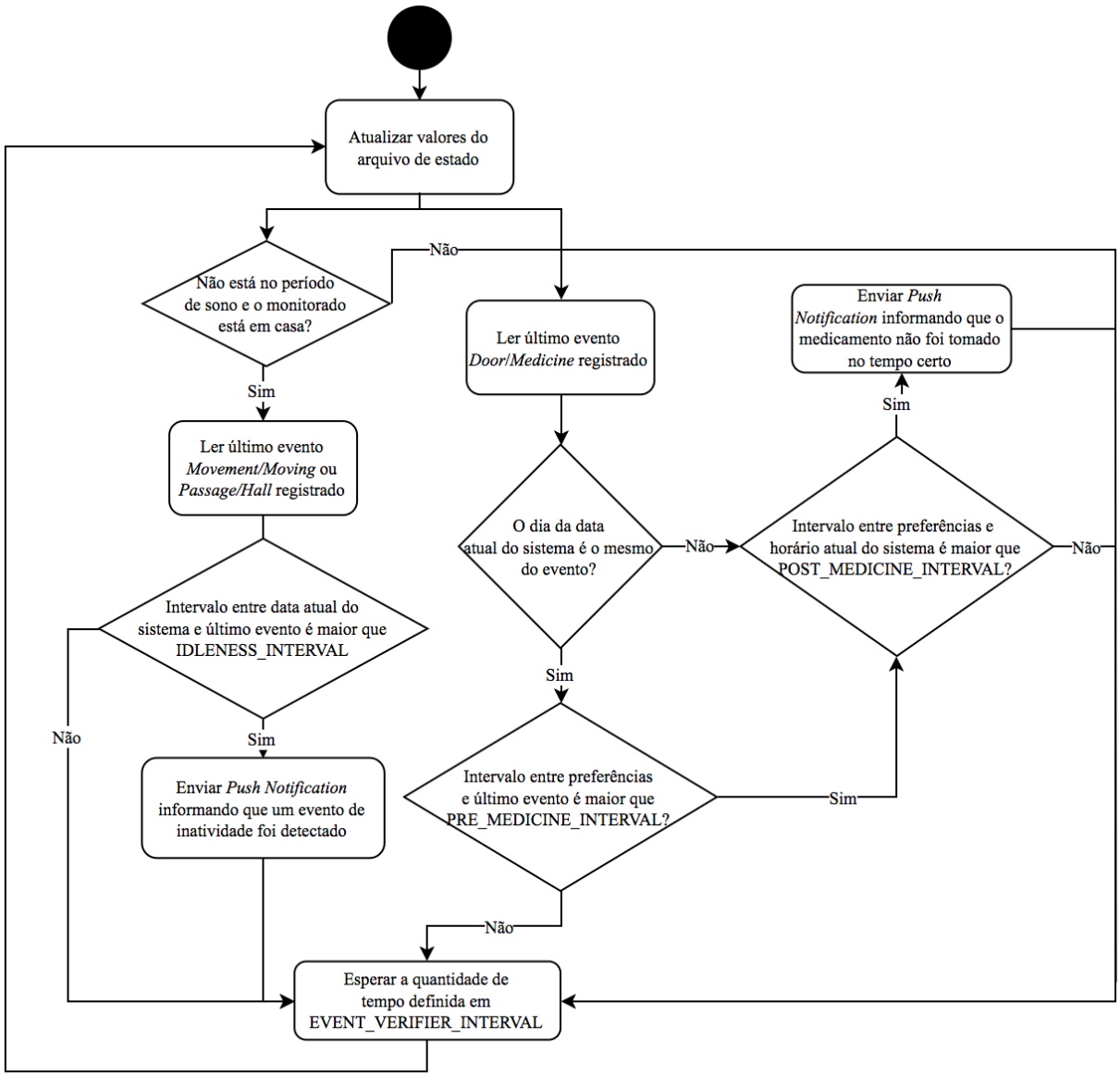
Para a detecção de entrada ou saída de pessoas da casa, uma análise adicional é feita no recebimento de dois eventos, o de abertura da porta principal (*Door*/*Main*) e o de passagem pela porta principal (*Passage*/*Main*). No momento do recebimento de um evento *Door*/*Main*, é verificado se logo antes ocorreu um evento *Passage*/*Main*, caso positivo significa que alguém acabou de entrar na casa, e então o servidor incrementa o número de pessoas na casa no arquivo XML de estados e, caso esse número seja 1 após a atualização, envia uma *push notification* através do FCM para a aplicação *mobile* informando que o monitorado acabou de chegar de casa. Analogamente, ao receber um evento *Passage*/*Main*, o servidor verifica se logo antes ocorreu um evento *Door*/*Main*, e se for verdadeiro decrementa o número de pessoas na casa e, caso esse número seja 0, envia uma *push notification* informando que o monitorado acabou de sair de casa. Para evitar que eventos falso positivos, como um carteiro que vai apenas deixar a correspondência na porta e é detectado pelo sensor, ou quando o monitorado, por algum motivo, abre a porta principal apenas para ver a rua, sem a intenção de sair, é definida uma constante de tempo que é utilizada na verificação do evento anterior. Ou seja, quando o servidor faz a verificação se determinado evento ocorreu anteriormente, esse evento deve ter acontecido há menos tempo que o valor da constante. Dessa forma, eventos que não são verdadeiramente subsequentes, como acontece em uma real situação de uma pessoa entrar na casa, são ignorados por essa análise.

Existem também alguns eventos que disparam instantaneamente uma *push notification* para a aplicação *mobile*, sem a necessiade de nenhum tipo de análise adicional. Esses eventos são o *Door*/*Medicine* e o *Moving*/*Fall*. O recebimento do primeiro faz o servidor enviar uma notificação informando que o monitorado acabou de tomar sua medicação. Já no recebimento do segundo, a notificação enviada é informando que o monitorado acabou de sofrer uma queda.

4.3.4 Verificador de eventos assíncrono

Além das verificações e processamentos que ocorrem durante o recebimento de requisições por parte do servidor, existe também a necessidade de se fazer algumas verificações assíncronas aos serviços, para situações que só podem ser corretamente detectadas após algum tempo, como as verificações de inatividade e de não consumo do medicamento no horário configurado. Para isso, o servidor executa assincronamente um verificador de eventos, que a cada determinado tempo verifica se algum desses eventos ocorreu e, caso positivo, envia uma *push notification* para a aplicação *mobile*. Para melhor entendimento, a Figura 4.2 mostra o diagrama de atividade desse verificador.

Figura 4.2: Diagrama de atividades do verificador de eventos assíncrono.



Fonte: Autor

Como pode ser visto no diagrama, o verificador depende de algumas constantes para o seu funcionamento, que são configuradas no próprio código do servidor. É uma tarefa difícil determinar os valores ideais para essas constantes, pois passa pela definição de questões subjetivas como a partir de quanto tempo sem movimentos pode ser considerado que um evento de inatividade ocorreu. Os valores definidos pelo autor para as constantes, que podem ser modificados dependendo das particularidades de cada pessoa a ser monitorada, são mostrados na Tabela 4.3. Além disso, também é válido observar que antes de cada rodada de verificações, o servidor primeiramente atualiza os valores das preferências lendo o arquivo XML de estados, pois o mesmo pode ter sido alterado no meio tempo

Tabela 4.3: Valores das constantes usadas no verificador de eventos assíncrono.

Constante	Valor (segundos)
EVENT_VERIFIER_INTERVAL	1800
IDLENESS_INTERVAL	7200
POST_MEDICINE_INTERVAL	3600
PRE_MEDICINE_INTERVAL	21600

Fonte: Autor

Para a detecção de inatividade, o servidor primeiramente verifica se o horário atual está dentro do período de sono configurado nas preferências ou se o monitorado está fora de casa, pois, nesses casos, nenhuma detecção de inatividade deve ser feita, já que é esperado que o monitorado esteja sem atividades enquanto está dormindo, e que não é possível detectar eventos de atividade se ele estiver fora de casa. Se não for nenhum desses casos, o último registro de um evento *Movement/Moving* ou *Passage/Hall* registrado é lido, e então é verificado se o intervalo entre a data atual do sistema e a data do último evento de atividade é maior que a constante IDLENESS_INTERVAL. Caso positivo, uma *push notification* é enviada informando que um evento de inatividade foi detectado.

Já para a verificação se a pessoa monitorada tomou seu medicamento no tempo correto, a lógica é um pouco mais complicada. Primeiro, o servidor lê o último evento *Door/Medicine* que foi registrado, e então verifica se o dia atual é o mesmo que o desse evento. Caso não seja, é verificado se o intervalo entre o horário configurado nas preferências e a hora do sistema é maior que a constante POST_MEDICINE_INTERVAL, e, se verdadeiro, uma *push notification* é enviada informando que o monitorado não tomou seus remédios no tempo certo. Caso seja o mesmo dia, é feita uma segunda verificação para saber se o intervalo entre o horário configurado nas preferências e do último evento *Door/Medicine* é maior que a constante PRE_MEDICINA_INTERVAL, caso positivo é feita a mesma verificação anterior com a constante POST_MEDICINE_INTERVAL e, da

mesma forma, se verdadeiro, uma *push notification* é enviada. Essa lógica adicional para quando o último evento ocorreu no mesmo dia da data do sistema é necessária, pois não basta apenas saber se o remédio não foi tomado no dia atual, já que, em determinadas configurações, como o remédio ter sido tomado muito cedo e o horário configurado nas preferências ser muito tarde, mesmo que a pessoa já tenha tomado o remédio no dia em questão, não é seguro considerar que o medicamento não precisa mais ser tomado pelo resto do dia. Por exemplo, supondo a situação onde a medicação é tomada em um horário cedo, como duas horas da manhã, e o horário configurado é um horário tarde no mesmo dia, como dez horas da noite, nesse caso o fato de já existir um evento de consumo do medicamento no mesmo dia, não significa que uma notificação não deva ser enviada quando o horário do sistema passar das dez horas da noite.

4.4 Sensores

Para a coleta de dados referentes a pessoa a ser monitorada, foram escolhidos uma variedade de sensores que, instalados de maneiras específicas, têm a capacidade de detectar e medir os eventos relevantes para permitir o monitoramento. Os sensores escolhidos são: um acelerômetro, dois sensores magnéticos de porta e dois sensores de passagem. É importante salientar que um mesmo tipo de sensor pode ser instalado para detectar diferentes tipos de informação. Por exemplo, o sensor magnético de porta serve tanto para ajudar a detectar quando alguma pessoa sai, ou entra, na casa monitorada, como para dectar quando o remédio foi tomado, após a abertura da porta do armário de remédios.

Esses sensores são o que podemos chamar de "coisas"no contexto deste trabalho e, além da medição e detecção, têm como principal característica a capacidade de se conectar e enviar dados pela Internet para o servidor. Essa capacidade de conexão com a Internet só é permitida graças a integração de cada sensor com uma placa Raspberry Pi 3 Model B que suporta conexão via Wi-Fi. A escolha da Raspberry Pi se deu pela sua extensa documentação existente na Internet, por sua facilidade de obtenção no mercado e pelo fato de existir um modelo com módulo Wi-Fi integrado, dispensando assim a necessidade de obtenção e instalação de um *shield* que fornecesse essa funcionalidade.

4.4.1 Raspberry Pi

A Raspberry Pi é um microcomputador de placa única de baixo custo desenvolvido pela Raspberry Pi Foundation. Ela possui diversos pinos que podem ser configurados como entrada ou saída, assim como também os pinos de alimentação (VCC) e aterramento (GND), onde é possível conectar sensores, atuadores, controladores, etc. Devido ao seu baixo custo e pequeno tamanho, a Raspberry Pi é ideal para integração com sensores, com o objetivo de dar inteligência e facilitar a comunicação dos mesmos.

As principais funcionalidades da Raspberry Pi no contexto deste trabalho são as de leitura e pré-processamento dos dados fornecidos pelos sensores e o posterior envio para o servidor. Para isso, é necessária a implementação de um *script* diferente para cada sensor, que seja capaz de ler os dados e interpretá-los, pois os sensores enviam informações apenas através de sinais nos pinos de entrada que, sem o devido tratamento, não possuem um significado claro, e, se for o caso, enviá-los para o servidor na nuvem em um formato que o mesmo consiga entender. Foi escolhida a linguagem Python para a implementação desses *scripts*, devido a grande quantidade de bibliotecas *open source* disponíveis para a interação com as portas da Raspberry Pi. Além disso, para que esse *script* seja capaz ser executado sem a necessidade de um disparo manual, ele é adicionado ao *boot* da Raspberry Pi, dessa forma, sempre que a mesma for inicializada, o *script* será executado e estará pronto para escutar os sensores nas portas adequadas.

4.4.2 Acelerômetro

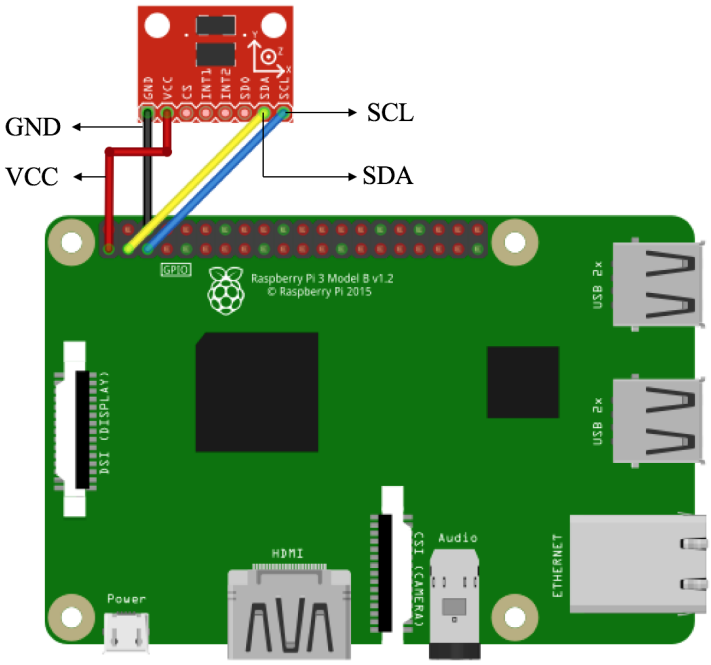
O sensor de acelerômetro tem duas principais funcionalidades: detecção de quedas e detecção de inatividade. No caso da detecção de queda o sensor deve sempre enviar uma notificação ao servidor, enquanto que para detecção de atividade o sensor deve ter sua frequência de envio controlada, visto que se o monitorado estiver realizando movimentos constantes o sensor acabaria enviando informações redundantes ao servidor.

O modelo de acelerômetro escolhido é o ADXL345. Esse modelo foi escolhido principalmente por ser um sensor digital, o que contribui para sua compatibilidade com a Raspberry Pi, pois a mesma não possui portas analógicas disponíveis, e também por possuir funções de detecção de eventos já integradas em seu circuito.

Para a comunição serial da Raspberry Pi com o sensor são suportados dois tipos diferentes de protocolos de comunição: o SPI (Texas Instruments, 2012) e o I^2C (NXP

Semiconductors, 2014). Foi escolhido o protocolo I^2C devido a sua simplicidade e menor quantidade de pinos necessários. O primeiro passo foi habilitar a interface I^2C na Raspberry Pi, através do comando *raspi-config*. Após isso, a comunicação pode ser realizada pelos pinos 3 e 5, conectando-os as entradas SDA (*Serial Data Line*) e SCL (*Serial Clock Line*) do sensor, respectivamente. Para a implementação do *script* para a leitura do sensor, foi utilizada a biblioteca *open source* Adafruit Python GPIO (ADAFRUIT, 2015), principalmente o módulo para comunicação I^2C. A Figura 4.3 mostra o esquema do circuito da conexão da Raspberry Pi com o sensor, onde é possível ver, além das conexões das entradas SDA e SCL, as conexões de alimentação dos pinos VCC e GND.

Figura 4.3: Esquema do circuito para o acelerômetro.



Fonte: Autor

Como pode ser visto em seu *datasheet* (ANALOG DEVICES, INC., 2009), o acelerômetro ADXL345 possui suporte para interrupções geradas por determinados eventos predefinidos. Essas interrupções podem ser habilitadas e configuradas ativando os bits referentes a cada evento no registrador INT_ENABLE, e definindo o pino de saída da interrupção, INT1 ou INT2, no registrador INT_MAP. As interrupções também podem ser monitoradas pelo registrador INT_SOURCE, onde cada bit se refere a uma interrupção específica, e essa foi a forma de implementação escolhida para a detecção dos eventos no *script* em Python. Dentre as interrupções disponíveis, são utilizadas a de detecção de atividade e a de queda livre, que também necessitam da configuração de alguns registradores para seu funcionamento. A interrupção de atividade é acionada quando uma aceleração maior que o valor contido no registrador THRESH_ACT é percebida em qualquer um dos

eixos configurados no registrador ACT_INACT_CTL. Já a interrupção de queda livre é acionada quando uma aceleração maior que o valor contido no registrador THRESH_FF é percebida por um tempo maior que o do valor contido no registrador TIME_FF em todos os eixos (AND lógico). Os valores configurados para cada um desses registradores pode ser visto na Tabela 4.4. Os valores de alguns registradores, como THRESH_FF e TIME_FF, foram determinados de forma empírica, após sucessivos testes para se encontrar valores apropriados.

Tabela 4.4: Tabela dos valores de configuração dos registradores.

Registrador	Binário	Observações
POWER_CTL	00001000	Habilita o sensor para o estado de medição.
BW_RATE	00001010	Configura o sensor para operar no modo normal e com uma taxa de saída de 100 Hz.
DATA_FORMAT	00001000	Configura o sensor para usar o intervalo de medição da força-g para ±2 g. Também define que as interrupções são ativadas com sinal alto.
INT_ENABLE	00010100	Habilitar apenas as interrupções de atividade de queda livre.
INT_MAP	00000000	Como o monitoramento de interrupções é feito pelo registrador INT_SOURCE, todas as interrupções são configuradas para a saída INT1 por motivos de simplificação.
THRESH_ACT	00100000	Corresponde ao valor 2 g (força-g), utilizando a escala 62.5 mg/LSB especificada no *datasheet*. Sempre que a magnitude da aceleração medida for superior a esse valor, uma interrupção de atividade será acionada.
ACT_INACT_CTL	01110000	Configura o modo *dc-coupled*, para que a aceleração corrente seja comparada diretamente contra o valor de *threshold*. Também habilita a verificação de todos os 3 eixos na detecção de atividade.
THRESH_FF	00001000	Corresponde ao valor 0,5 g (força-g), utilizando a escala 62.5 mg/LSB especificada no *datasheet*. A soma das raízes quadradas dos valores de todos os eixos é calculada e comparada com esse valor para determinar se um evento de queda livre ocorreu.
TIME_FF	00010100	Corresponde ao valor 0,1 segundos, utilizando a escala 5 ms/LSB especificada no *datasheet*. Define por quanto tempo a condição especificada no cálculo do registrador THRESH_FF deve ser satisfeita para que a interrupção de queda livre ocorra.

Fonte: Autor

Sempre que a interrupção de queda livre é detectada o servidor deve ser informado para que possa notificar a pessoa que está realizando o monitoramento, pois se trata de um evento de risco. Nesse caso, uma requisição HTTP é enviada com a propriedade *type* igual a *movement*, e a propriedade *info* igual a *fall*, comunicando assim que se trata de um evento de queda.

No caso da interrupção de atividade, a requisição é enviada para o servidor apenas se fazem mais do que 10 minutos desde o último envio. Essa validação é feita para evitar o envio constante de informação redundante, visto que, como já explicado, o intervalo de tempo necessário para que um evento de inatividade seja detectado é relativamente alto, sendo portanto desnecessária uma alta frequência de detecção desse tipo de evento. Esse intervalo pode representar uma diminuição na capacidade de reação por parte pessoa que está realizando o monitoramento, por isso é um valor que pode ser avaliado e, se necessário, modificado. Quando enviada, essa requisição possui a propriedade *type* como *movement* e a propriedade *info* como *moving*, o que significa que o evento detectado foi de movimentação através do acelerômetro.

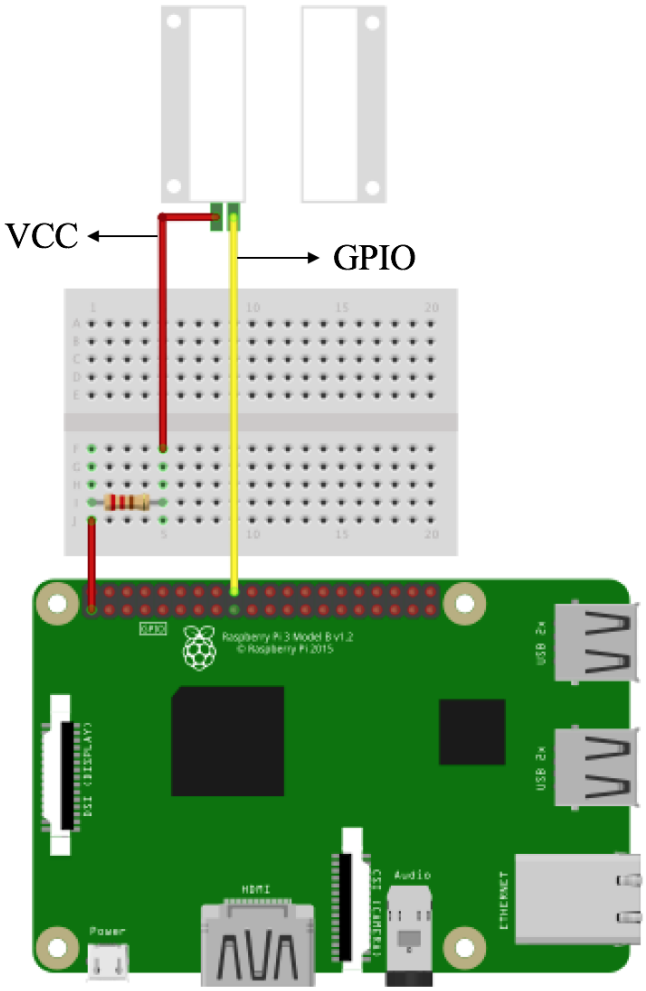
4.4.3 Sensor magnético de porta

Os sensores magnéticos de porta servem, essencialmente, para detectar quando uma porta é aberta ou fechada. No contexto deste trabalho são utilizados dois sensores, um na porta principal da casa, que serve para auxiliar na detecção de quando alguma pessoa saiu ou entrou no domicílio, e outro na porta do armário de remédios, que serve para detectar quando o monitorado tomou seus medicamentos. Ambos tem o funcionamento semelhante, só mudando os valores das propriedades da requisição HTTP.

Esse tipo de sensor funciona da mesma forma que uma chave *switch*, utilizada em circuitos eletrônicos. Quando a porta está fechada, as duas partes magnéticas estão encostadas, e o sensor permite a passagem de corrente. Quando a porta se abre, as partes magnéticas se separam, e a passagem de corrente é interrompida. A detecção desse evento só é feita na descida do sinal, pois apenas a abertura em si é de interesse da aplicação, não sendo relevante quanto tempo a porta ficou aberta ou quando foi fechada. A Figura 4.4 ilustra a configuração do circuito utilizado. O resistor colocado no pino de alimentação de 5 V da Raspberry tem o intuito de limitar a corrente e evitar possíveis danos a placa. O valor do resistor utilizado é de 1k ohms, que, dividindo pela voltagem fornecida ao sensor de 5 V, limita o valor máximo da corrente no pino de entrada da Raspberry Pi a 5 mA, que é um valor considerado seguro de acordo com as especificações da placa.

No momento da detecção da abertura de porta, uma requisição é enviada para o servidor informando desse evento. Para ambos os sensores a propriedade *type* tem o valor *door*, enquanto que a propriedade *info* tem o valor *medicine* para a detecção da porta do armário de medicamentos, e *main* para a porta principal do domicílio.

Figura 4.4: Esquema do circuito para o sensor magnético de porta.



Fonte: Autor

4.4.4 Sensor de presença

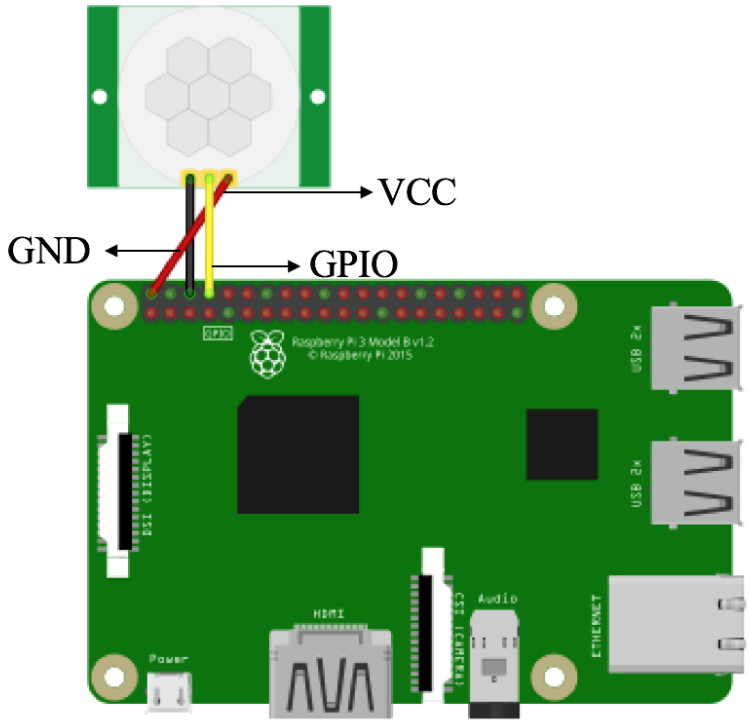
Para a detecção de atividade dentro da casa, foi escolhido um sensor de presença infravermelho do tipo PIR (*Passive Infrared Sensor*). Esse sensor deve ser instalado em um local principal de passagem no interior da casa monitorada, como um corredor que ligue a sala aos quartos, por exemplo, de forma que exista uma alta probabilidade do monitorado passar pelo sensor quando estiver se movimentando pela casa. Esse sensor serve para auxiliar na detecção de inatividade da pessoa a ser monitorada, ou seja, para informar o servidor sempre que alguma atividade for detectada, o que significa que na ausência desses eventos um eventual detecção de inatividade pode acontecer.

O modelo de sensor escolhido é o DYP-ME003. Essa escolha se deu pelo fato de ser um sensor de baixo custo, fácil utilização e grande alcance, esse último sendo especialmente importante, pois para esse tipo de evento a precisão não é importante, mas sim a capacidade de detectar o máximo de atividade possível. Ele funciona através da

detecção de luz infravermelha pelo seu sensor piroelétrico, que gera energia ao ser exposto ao calor. Dessa forma, um sinal é emitido quando uma pessoa, ou até mesmo um animal, passa pelo alcance de detecção do sensor. O sensor também é coberto por uma lente de Fresnel, que serve para focar a luz infravermelho diretamente no sensor piroelétrico, o que aumenta o alcance e define um padrão de detecção.

Esse sensor possui apenas três pinos, um para alimentação (VCC), o terra (GND) e o pino de saída, que recebe um sinal alto quando alguma atividade é detectada pelo sensor piroelétrico. A sua instalação é bastante simples e é mostrada na Figura 4.5. Além disso, também foi ajustado o alcance de detecção, através de um potenciômetro que se encontra no próprio sensor, para o máximo possível que corresponde ao valor de 7 metros.

Figura 4.5: Esquema do circuito para o sensor de presença infravermelho.



Fonte: Autor

Uma requisição deve ser enviada ao servidor sempre que alguma presença for detectada. Para que essa requisição seja corretamente interpretada pelo servidor, a propriedade *type* deve ter o valor *passage* e a propriedade *info* o valor *hall*.

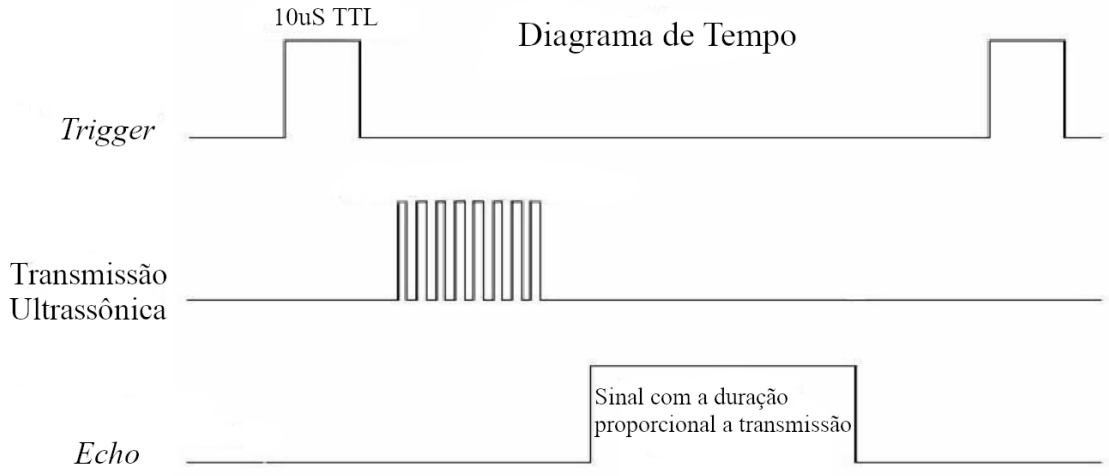
4.4.5 Sensor de passagem

Como já dito, uma das funcionalidades previstas para a aplicação é a capacidade de identificar quando alguém saí ou entra na casa da pessoa monitorada. Para isso, é

necessária a detecção de passagem do lado de fora da porta principal da residência, através de um sensor instalado no batente da porta, de forma que sempre que uma pessoa sair ou entrar na casa, ela tenha que passar esse sensor. Essa detecção, alinhada com a detecção de abertura de porta, é capaz de fornecer os dados necessários para que o servidor possa concluir a entrada e saída de indivíduos do domicílio a ser monitorado.

Para essa função, foi escolhido o sensor ultrassônico HC-SR04, pois, diferentemente do sensor de presença, é importante que a detecção de passagem seja precisa, somente notificando o servidor quando realmente alguém estiver passando pela frente da porta, evitando falsos positivos de pessoas que estejam apenas passando perto, como um vizinho de porta, por exemplo. O sensor funciona emitindo ondas ultrassônicas assim que um sinal é emitido ao pino de *trigger*, recebendo a reflexão dessas ondas e ecoando essa reflexão em um pino de saída chamado de *echo*, com a duração igual ao tempo levado pela onda para atingir o objeto e voltar. Para melhor ilustrar esse funcionamento, a Figura 4.6 mostra em um diagrama de tempo a interação entre esses sinais e os pinos do sensor. Na figura, é possível ver que primeiro vem a emissão do sinal para o pino de *trigger*, após isso, acontece a transmissão ultrassônica, que é recebida de volta pelo sensor ao ser refletida por algum objeto, e por fim um sinal de eco proporcional ao tempo decorrido entre a transmissão e recepção do sinal ultrassônico.

Figura 4.6: Diagrama de tempo para o sensor ultrassônico.



Fonte: (ELECFREAKS, 2010)

Com a duração do sinal no pino *echo*, e conhecendo o valor da velocidade do som, é possível calcular a distância entre o sensor e o objeto a sua frente com a fórmula expressa na equação 4.1.

$$L = \frac{TC}{2} \tag{4.1}$$

Onde:

L = Distância do objeto até o sensor

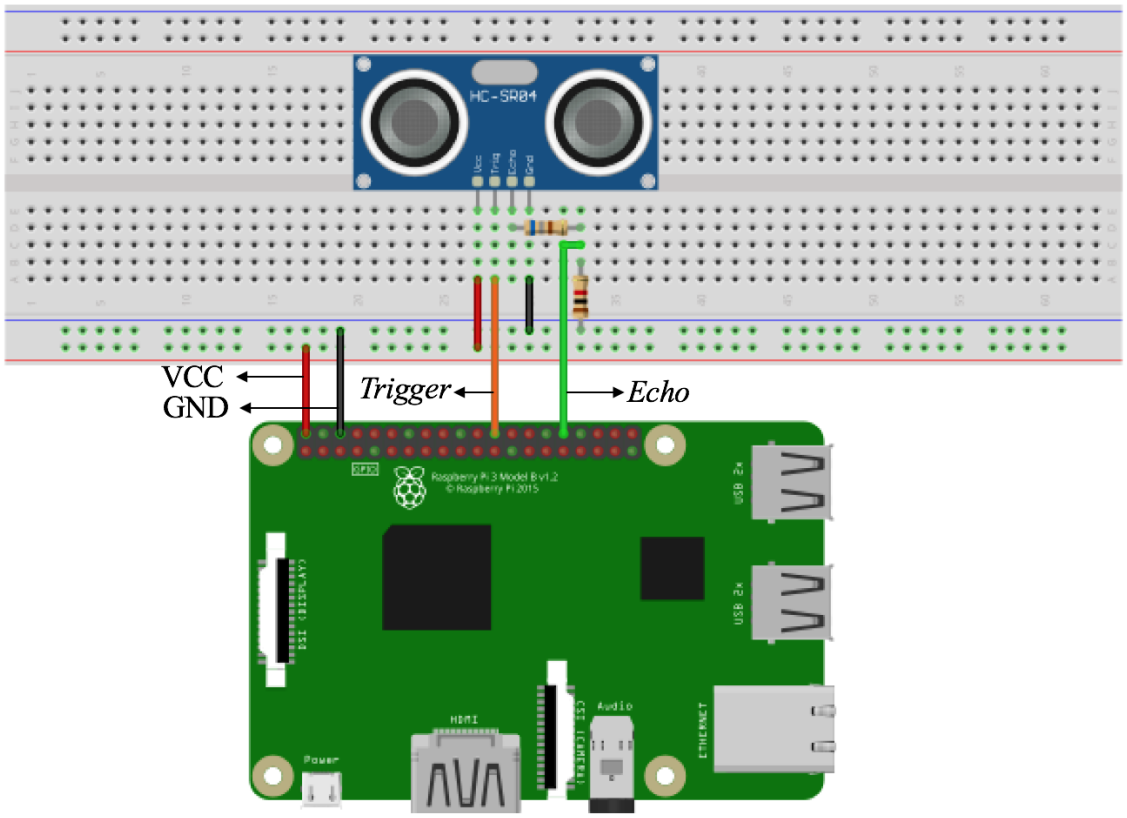
T = Duração da onda de eco, que corresponde ao tempo entre transmissão e recepção

C = Velocidade do som

A divisão por 2 é devido ao fato que o tempo de transmissão e recepção corresponde a ida e a volta do sinal, e se deseja apenas a distância entre o objeto e o sensor, e não a distância total percorrida pelo sinal. A velocidade do som utilizada foi de 343,37 m/s (BOHN, 1988).

A Figura 4.7 mostra a instalação do circuito para o sensor ultrassônico. Novamente, um resistor de 1k ohms foi colocado no pino de saída *echo* e a Raspberry Pi para limitar a corrente e prevenir possíveis danos a Raspberry. Além disso, um resistor *pull-down* de 10k ohms foi colocado entre o *echo* e o GND, para aterrar o sinal quando nada estiver sendo emitido, e assim evitar flutuações na leitura do mesmo.

Figura 4.7: Esquema do circuito para o sensor ultrassônico de passagem.



Fonte: Autor

No contexto da aplicação, não é importante a distância de um objeto, mas sim se algum objeto, no caso uma pessoa, está passando pelo sensor. Para isso, no momento da instalação do sensor é necessária uma medição, através de um teste com o próprio sensor,

da distância entre o sensor instalado e o objeto mais próximo, como uma parede, por exemplo. Esse valor é então fornecido ao *script* do sensor através de uma constante no código, e será usado como base para o cálculo. Caso a distância medida pelo sensor seja igual ou maior a distância base, nenhuma detecção é feita, mas caso a distância seja menor que a distância base, então temos uma detecção de passagem e o sensor pode enviar uma notificação para o servidor. Essa notificação é enviada com o valor de *type* igual a *passage*, e o valor de *info* igual a *main*. Além disso, após uma notificação ter sido enviada, o valor da distância da última detecção é guardado, e então outra notificação só é enviada se o novo valor da distância medido for diferente do antigo, ou se uma determinda quantidade de tempo, também configurada em uma constante, tenha passado desde a última detecção. Isso é feito para evitar que sucessivas requisições sejam enviadas para o servidor quando, por exemplo, uma pessoa fique parada na frente do sensor.

4.5 Aplicação *mobile*

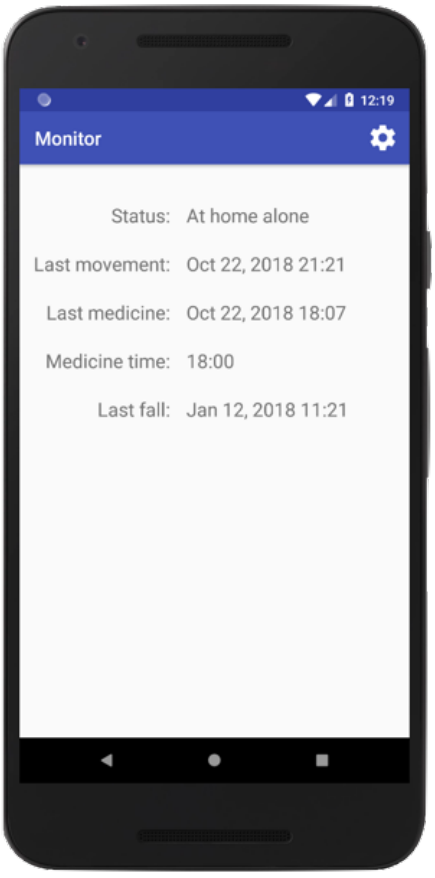
A aplicação *mobile* permite visualizar as informações coletadas do monitoramento, assim como também onde é possível pode configurar as preferências do monitorado, como horário de sono e horário dos medicamentos. Além dessas funcionalidades, é através dessa aplicação que as notificações, também conhecidas como *push notifications*, são recebidas, mesmo a aplicação sendo executada em *background*. Assim, como definido durante a definição dos requisitos não funcionais, a aplicação deve ser simples e intuitiva, mostrando apenas o mínimo necessário para o suporte de todas as funcionalidades previstas.

Essa aplicação é desenvolvida para o sistema operacional Android, mais especificamente para as API *levels* entre 24 e 27, que correspondem às versões 7 e 8. Nessa escolha, foi levado em conta principalmente o fato do Android ser *open source*, ser um sistema operacional utilizado em cerca de 90% dos *smartphones* do mundo (DARWIN, 2017), e consequentemente, possuir um grande suporte de sua comunidade, e que sua principal alternativa, o iOS, requer dispositivos da Apple para o desenvolvimento de aplicações. A linguagem de programação empregada no desenvolvimento Android é a linguagem Java, que foi utilizada junto com a IDE (*Integrated Development Environment*) Android Studio, que é fornecida gratuitamente pela Google.

Na tela inicial do aplicativo se encontram as informações mais relevantes e recentes coletadas pelos sensores, dando assim uma visão geral do monitorado e seu estado

atual. Essas informações são obtidas através de serviços específicos fornecidos pelo servidor, que foram descritos na seção sobre servidor. Para o acesso desses serviços foi utilizada a biblioteca Volley, que fornece suporte para requisições HTTP e também toda a gerência de cache e concorrência de chamadas. A Figura 4.8 mostra a tela inicial da aplicação já desenvolvida, sendo executada em um simulador disponibilizado pelo Android Studio. O conceito dessa tela é bastante simples: mostrar de uma forma geral a situação atual da pessoa que está sendo monitorada. Os dados da tela são atualizados a cada vez que a tela é carregada, e caso aconteça algum erro na obtenção desses dados uma mensagem de erro em vermelho aparece no lugar do dado em questão.

Figura 4.8: Tela inicial.

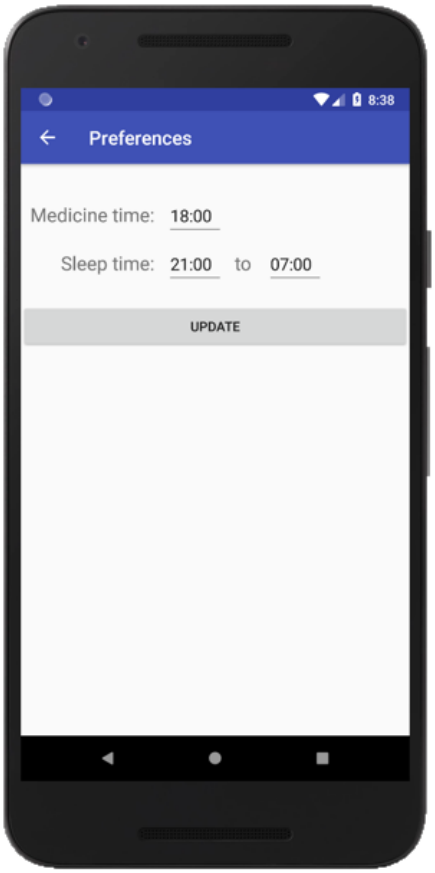


Fonte: Autor

A outra tela desenvolvida para a aplicação é a tela de configurações de preferência. Essa tela é acessada ao se apertar no botão com o ícone de engrenagem, presente na tela inicial. A Figura 4.9 mostra essa tela, onde pode se ver que o requisito não funcional de simplicidade e intuição também foi seguido, disponibilizando ao usuário apenas o essencial que permita o acesso total as funcionalidades requeridas. Nessa tela, como já explicado, o usuário pode configurar o horário do medicamento e do sono do monitorado.

Ao pressionar o botão de *update*, uma requisição de escrita é enviada para o servidor, que por sua vez grava esses dados, mudando assim o estado das preferências. Caso aconteça algum erro durante essa operação, uma janela de erro é mostrada para o usuário, informando que a aplicação não conseguiu salvar os dados com sucesso. Caso a operação seja realizada com sucesso, uma mensagem de confirmação também é mostrada.

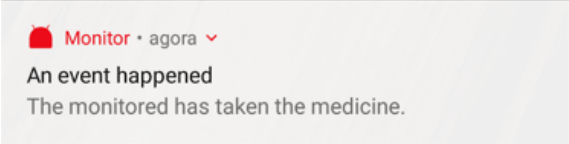
Figura 4.9: Tela de configuração de preferências.



Fonte: Autor

Por fim, outra funcionalidade da aplicação também é o disparo de notificações em determinados eventos detectados pelo servidor. Para isso, foi utilizado o serviço de *Push notification* oferecido nativamente pelo Android, que é disparado através do Google Firebase, como já explicado na seção passada sobre o servidor. A notificação pode ser vista na Figura 4.10, e contém apenas as informações basicas do evento que aconteceu. Ao clicar na notificação a aplicação é executada em primeiro plano no dispositivo, abrindo a tela inicial já com as informações atualizadas.

Figura 4.10: Exemplo de notificação.



Fonte: Autor

4.6 Considerações Finais

No capítulo atual foi detalhada a fase de implementação da solução proposta neste trabalho. Inicialmente, foram especificadas as tecnologias escolhidas e a arquitetura utilizada, para depois entrar em mais detalhes a respeito do desenvolvimento propriamente dito. Esse detalhamento foi apresentado em três partes: servidor sensores e aplicação *mobile*. O próximo capítulo é onde será avaliada essa implementação e seus resultados.

5 AVALIAÇÃO

Após a fase de implementação, é necessária a validação do protótipo desenvolvido através de testes experimentais, a fim de constatar o devido funcionamento da aplicação proposta. Neste capítulo, são abordados esses experimentos e seus resultados, assim como a metodologia empregada na sua realização.

5.1 Plataforma experimental

Como a maioria dos sensores podem ser testados individualmente, pois detectam eventos independentes entre si, e também como forma de contenção de gastos, foi usada a mesma Raspberry Pi para os testes de diferentes sensores. A única exceção foi o experimento para testar a detecção de entrada e saída de pessoas na casa, que exige a atuação conjunta de dois sensores isolados e, consequentemente, duas placas distintas. Os sensores utilizados são os abordados no capítulo anterior, e suas conexões com a Raspberry foram feitas através de uma *protoboard* e de *jumpers*, conforme também ilustrado na seção 4.4. Além disso, o domicílio onde foram realizados os experimentos foi a própria casa do autor, que conta com conexão Wi-Fi via um Modem da marca Sagemcom, que realiza a função de *gateway*. O próprio autor que realizou a função de pessoa sendo monitorada, simulando as atividades físicas necessárias para o estímulo dos sensores. Por fim, a aplicação *mobile* foi instalada em um celular Motorola G4 Plus, com sistema operacional Android.

5.2 Metodologia

Para avaliar que todo o sistema está funcionando como previsto, desde os sensores, passando pelo servidor, até a aplicação *mobile*, é necessária uma série de cenários diferentes, onde eventos especifícos relevantes ao contexto da aplicação são simulados, e então pode-se avaliar se todos os componentes se comportam como esperado. Os cenários foram divididos em três tipos: os que são iniciados a partir de uma ação fisíca do autor, com o intuito de estimular os sensores; os que são iniciados a partir de uma requisição da aplicação *mobile*, sendo isolados de qualquer dependência do funcionamento dos sensores; e os que testam a verificação assíncrona de eventos. O servidor foi avaliado nos três tipos

de cenários, visto que todos dependem do correto processamento dos dados e/ou resposta das requisições por parte dele.

A avaliação referente aos sensores conta com cinco cenários distintos, que foram baseados em todas as possíveis detecções por parte do servidor acionadas pelo recebimento dos dados coletados. Esses cenários são listados abaixo:

- Detecção da abertura da porta de medicamentos e envio de notificação à aplicação *mobile*.
- Detecção de queda e envio de notificação à aplicação *mobile*.
- Detecção de evento de atividade.
- Detecção de saída do monitorado da casa e envio de notificação à aplicação *mobile*.
- Detecção da volta do monitorado à casa e envio de notificação à aplicação *mobile*.

Para a validação da aplicação *mobile*, dois cenários foram considerados, que correspondem as duas principais funcionalidades da aplicação:

- Visualização na tela inicial das informações recentes mais relevantes.
- Visualização e edição das preferências do monitorado.

Por último, a verificação de eventos assíncrona também foi testada através de três cenários:

- Detecção de inatividade e envio de notificação à aplicação *mobile*.
- Não detecção de inatividade durante o horário de sono configurado.
- Detecção do não consumo do medicamento no horário configurado e envio de notificação à aplicação *mobile*.

Cada um dos cenários envolve o funcionamento de diferentes componentes do sistema que, apesar de não serem o elemento principal do experimento, também necessitam estar funcionando plenamente para que o cenário seja validado, como a conexão Wi-Fi, a hospedagem do servidor na nuvem e sua capacidade de receber requisições, a persistência dos dados no banco e a configuração da plataforma Firebase para utilização do FCM.

5.3 Experimentos

Nesta seção são descritos os experimentos realizados e seus resultados, também fornecendo mais detalhes acerca do ambiente de teste e de como foi feita a validação.

Eles estão divididos nas seções seguintes da mesma forma que foram listados acima, sendo três principais avaliações: sensores, aplicação *mobile* e verificador assíncrono de eventos.

5.3.1 Avaliação dos sensores

Como já abordado, para os experimentos referentes aos sensores foram realizados 5 cenários diferentes, cada um com objetivo de detectar um evento em específico. Os cenários foram preparados instalando os sensores, cada um já devidamente conectado com a Raspberry Pi e com seu *script* em Python executando, nos locais adequados na residência do autor. Para a validação dos cenários foi utilizado os *logs* gerados pelo servidor, requisições ao banco de dados e, nos casos onde era previsto uma *push notification* por parte do servidor, o recebimento dela no dispositivo utilizado no experimento.

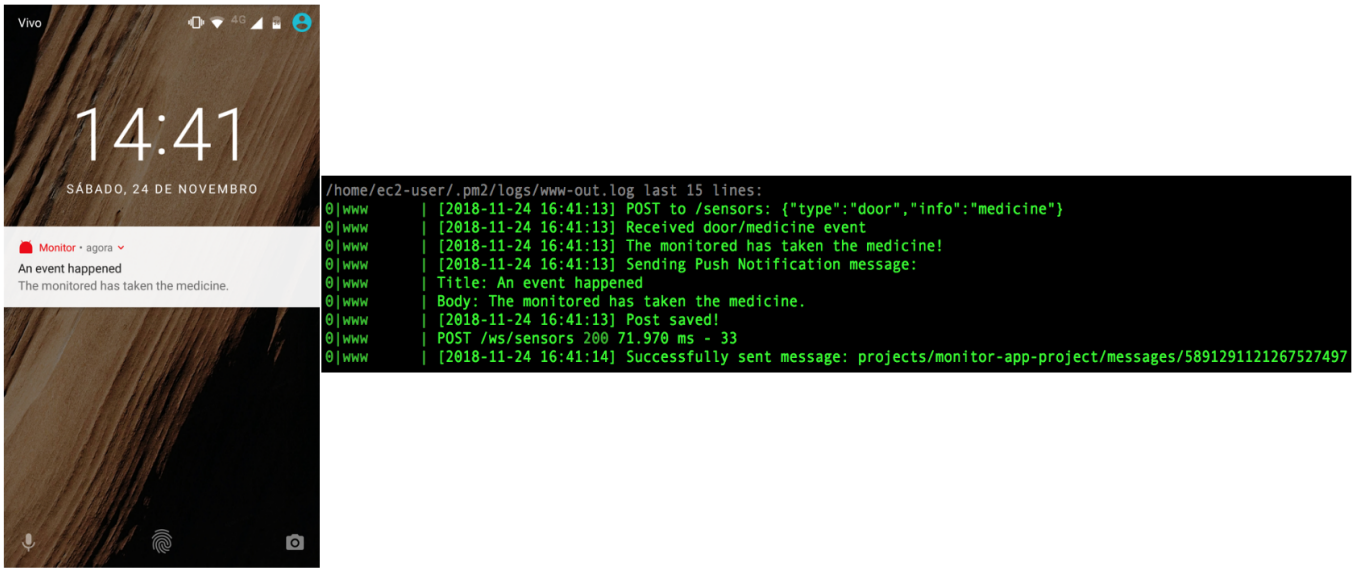
O primeiro cenário avaliado tem como objetivo testar a detecção da abertura da porta do armário de remédios e o envio da notificação de que o monitorado acabou de tomar o medicamento. A ideia do experimento é simples: a porta é aberta e então se verifica se foi recebida a notificação no dispositivo móvel utilizado no teste. No caso desse experimento, ao invés de uma porta foi utilizada uma gaveta, mas o conceito permanece o mesmo. A Figura 5.1 mostra uma foto tirada da instalação.

Figura 5.1: Instalação do sensor magnético de porta no armário de medicamentos.

Fonte: Autor

Como esperado, assim que a porta foi aberta o *script* detectou o evento e enviou, através da rede Wi-Fi da casa, uma notificação à aplicação *mobile*. A Figura 5.2 mostra a forma de validação desse cenário, na direita é possível ver o *log* extraído do servidor, onde consta o evento recebido, e na esquerda é mostrada uma captura de tela do celular no momento do recebimento da *push notification*.

Figura 5.2: Validação do sensor magnético de porta no armário de medicamentos.



Fonte: Autor

Uma situação de possível detecção falso positiva que foi identificada seria o caso da pessoa a ser monitorada abrir a porta do armário de medicamentos, porém não chegar a de fato consumir nenhum dos medicamentos, seja por engano ou por um ato intencional. Para diminuir a possibilidade dessa situação acontecer, o ideal é que no armário, ou gaveta, em questão, tenham apenas os medicamentos, eliminando a necessidade de se abrir a porta a não ser para o acesso dos medicamentos.

O segundo cenário valida a detecção de queda do monitorado e o envio da notificação ao dispositivo móvel. O experimento foi realizado acoplando o acelerômetro ao pulso do autor, e então o autor simulou uma movimento de queda em uma superfície acochoada, e após isso foi verificado se o evento de queda foi de fato detectado. A Figura 5.3 mostra uma foto da instalação do sensor no pulso do autor.

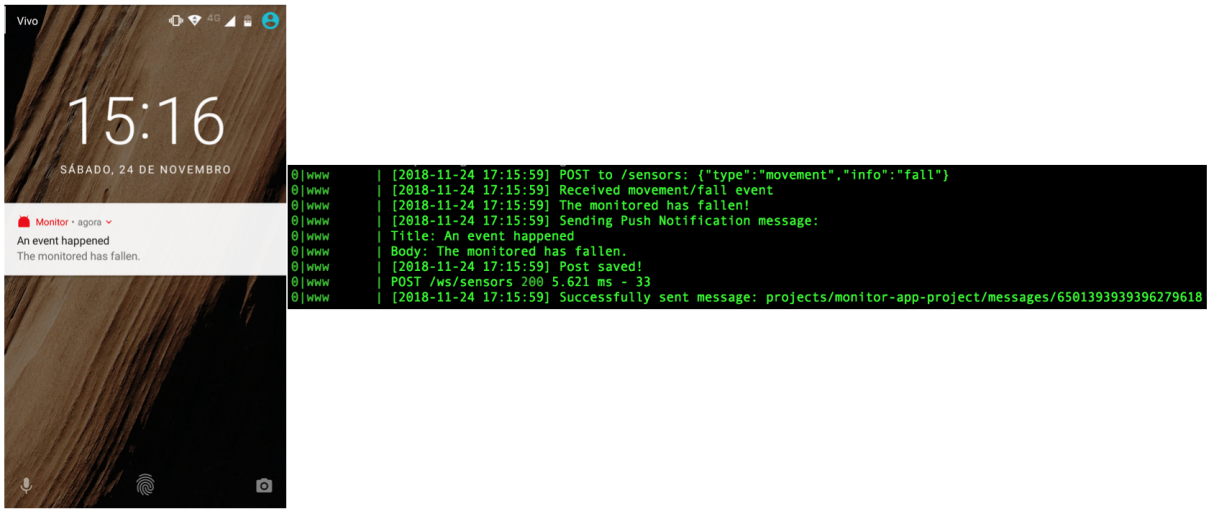
O cenário foi válidado com sucesso, como pode ser visto na Figura 5.4. Assim como no cenário anterior, a validação se deu através do *log* do servidor (à direita da figura), e do recebimento da notificação no celular (à esquerda da figura).

O terceiro cenário é referente a detecção de atividade, que pode ser feita através de dois tipos de sensores diferentes: o acelerômetro, ou o sensor de presença. Os dois

Figura 5.3: Instalação do acelerometro.

Fonte: Autor

Figura 5.4: Validação do acelerômetro para o evento de queda.



Fonte: Autor

sensores foram testados, sendo a validação dos experimentos feitas da mesma forma para ambos. No teste para o acelerômetro, a instalação do sensor é a mesma mostrada na Figura 5.3 do cenário anterior, apenas sendo diferente a ação do autor, que nesse caso apenas realizou um movimento brando, levantando o braço. Para o sensor de presença infravermelho, a instalação foi feita no meio da sala da residência, que é o lugar de maior circulação da casa, como pode ser visto na Figura 5.5. A ação do autor nesse caso foi a de simplesmente caminhar dentro da área de alcance do sensor.

Como a detecção de atividade não ocasiona um envio de notificação à aplicação *mobile*, a validação foi feita apenas através do *log* do servidor e de uma leitura no banco de dados, para conferir que de fato um evento de atividade foi recebimento e devidamente persistido. A Figura 5.6 mostra o resultado dessa verificação, onde na parte de cima é possível ver o *log* do servidor, e na parte de baixo a leitura ao banco de dados. Também foi verificado que, no teste do acelerômetro, apenas um evento de atividade é gerado a cada 10 minutos, mesmo que o autor continue se movimentando após a primeira detecção.

Figura 5.5: Instalação do sensor infravermelho de presença.

Fonte: Autor

Figura 5.6: Validação dos sensores de detecção de atividade.

```
0|www      | [2018-11-24 17:29:48] POST to /sensors: {"type":"passage","info":"hall"}
0|www      | [2018-11-24 17:29:48] Received passage/hall event
0|www      | [2018-11-24 17:29:48] Post saved!
0|www      | POST /ws/sensors 200 2.763 ms - 32
0|www      | [2018-11-24 17:32:38] POST to /sensors: {"type":"movement","info":"moving"}
0|www      | [2018-11-24 17:32:38] Received movement/moving event
0|www      | [2018-11-24 17:32:38] Post saved!
0|www      | POST /ws/sensors 200 2.485 ms - 35
```

```
> db.sensors.find({ $or: [ { type: "movement", info: "moving" }, { type: "passage", info: "hall" }] }).pretty()
{
        "_id" : ObjectId("5bf98a8ceeaf0a135bb2a919"),
        "type" : "passage",
        "info" : "hall",
        "createdDate" : ISODate("2018-11-24T17:29:48.642Z"),
        "__v" : 0
}
{
        "_id" : ObjectId("5bf98b36eeaf0a135bb2a91a"),
        "type" : "movement",
        "info" : "moving",
        "createdDate" : ISODate("2018-11-24T17:32:38.288Z"),
        "__v" : 0
}
```

Fonte: Autor

Como o sensor de infravermelho não é capaz de distinguir a origem da radiação detectada, caso a pessoa monitorada tenha algum animal de estimação, podem ser detectados eventos de movimentação falso positivos. Uma solução para essa situação seria a de utilizar um sensor mais sofisticado, que permitisse uma análise detalhada da radiação detectada, e dessa forma diferenciar se o movimento detectado veio de um animal ou de uma pessoa.

Os últimos dois cenários têm o objetivo de testar a detecção de entrada e saída da pessoa monitorada de casa. Para isso, o sensor magnético de abertura de porta foi instalado na porta principal da casa, enquanto que o sensor de passagem ultrassônico foi instalado do lado de fora, de forma que sempre que alguma pessoa entre ou saia da casa,

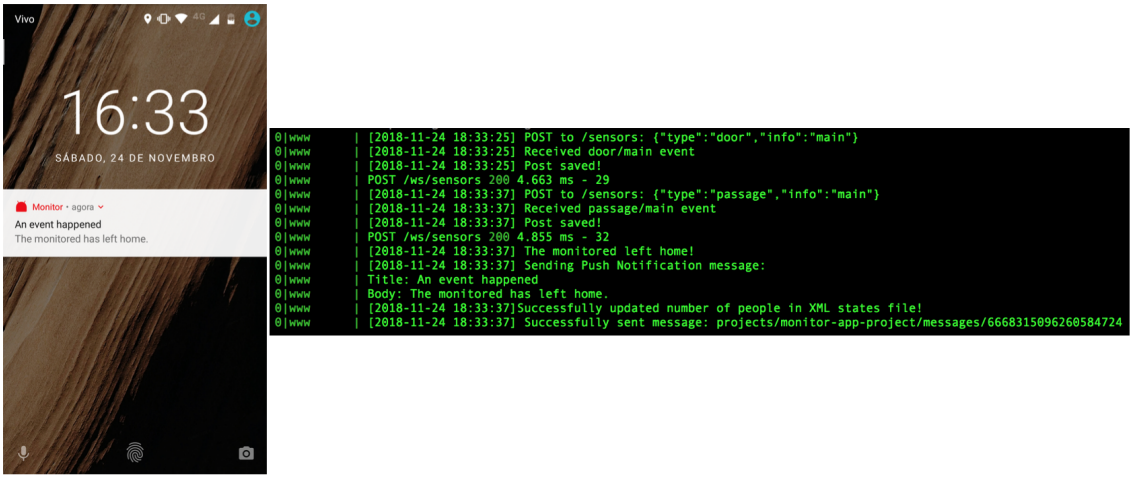
passe por ele. A Figura 5.7 mostra a instalação de ambos os sensores, sendo na esquerda do sensor ultrassônico, e na direita o magnético. Além da instalação dos sensores, foi também necessário inicializar manualmente no servidor o número de pessoas da casa no arquivo XML de estados em 1, para que na detecção de saída e entrada o servidor identifique que a pessoa em questão é o monitorado. Essa instalação foi feita considerando que a residência só tenha uma porta de entrada, mas no caso de possuir mais, a mesma instalação deve ser replicada para as outras portas.

Figura 5.7: Instalações do sensor ultrassônico (esquerda) e do sensor magnético na porta principal (direita).

Fonte: Autor

Primeiramente, o autor simulou o evento da saída, abrindo a porta e saindo da casa passando pelo sensor de passagem. Como esperado, ambos os eventos foram recebidos e o servidor corretamente detectou que o monitorado saiu da casa. A Figura 5.8 mostra a validação, com o *log* do servidor e o recebimento da *push notification* no celular.

Figura 5.8: Validação da detecção de saída da casa.

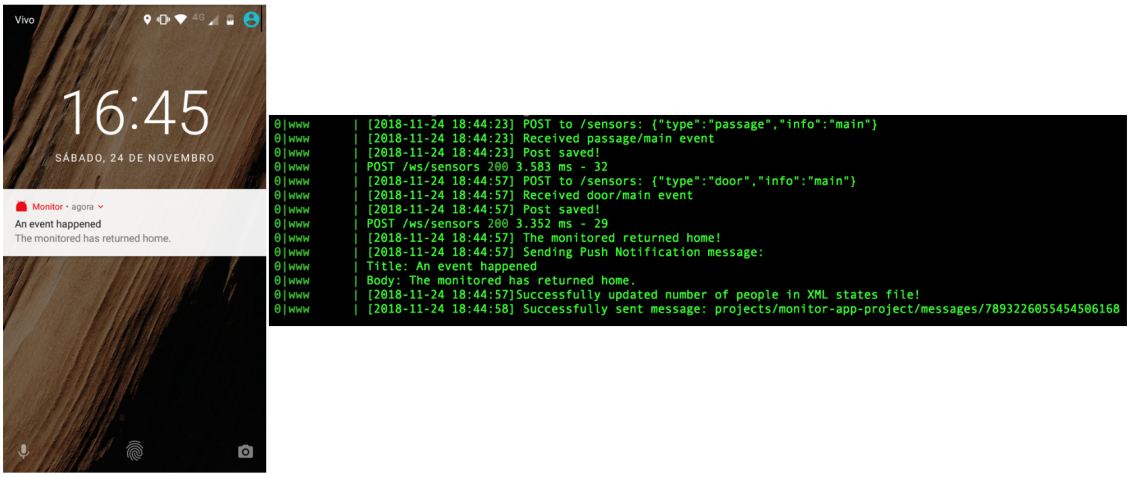


Fonte: Autor

Após a saída da casa, o autor simulou o evento de entrada, ao passar pelo sensor ultrassônico, abrir a porta, e entrar na casa. Nesse último cenário, o envio dos eventos

também ocorreu da forma esperada, e o servidor foi capaz de identitificar que o autor entrou na casa. Na Figura 5.9 é possível ver o *log* do servidor e a notificação recebida no dispositivo móvel.

Figura 5.9: Validação da detecção de entrada na casa.



Fonte: Autor

Existem algumas formas de provocar interpretações falso positivas por parte do servidor, que foram identificadas durante a realização do último cenário. Por exemplo, caso alguém passe pelo sensor de passagem e, por algum motivo, como a campainha ter sido tocada, a pessoa monitorada abra a porta em sequência, o servidor detectaria que alguma pessoa entrou na casa, apesar de que isso não seria necessariamente verdade. Uma outra situação é a de mais de uma pessoa entrarem juntas na casa, ou em um intervalo menor que o tempo configurado na constante de tempo utilizada no sensor ultrassônico para eventos subsequentes.

5.3.2 Avaliação da aplicação *mobile*

Para os testes da aplicação *mobile* foram preparados dois cenários, cada um cobrindo uma das duas principais funcionalidades previstas no aplicativo. Por motivos de simplificação, o histórico de eventos para os cenários foi preparado utilizando a ferramenta Postman, que é basicamente um cliente HTTP que permite o envio de requisições, de forma que é possível simular o comportamento dos sensores. Definido o histórico de eventos, o aplicativo foi testado e os dados retornados foram comparados com os esperados.

Primeiramente, através do Postman foram enviadas uma série de requisições para o servidor, para simular os eventos mais recentes e definir que informações o aplicativo

deve mostrar no experimento. Além dos campos de identificação do evento, também foi utilizado o campo *CreatedDate* no *payload* da requisição HTTP, para definir a data de criação desse evento. Também foram definidos valores iniciais para as preferências e o número de pessoas na casa. A Tabela 5.1 mostra as datas de cada evento e os valores das preferências definidos.

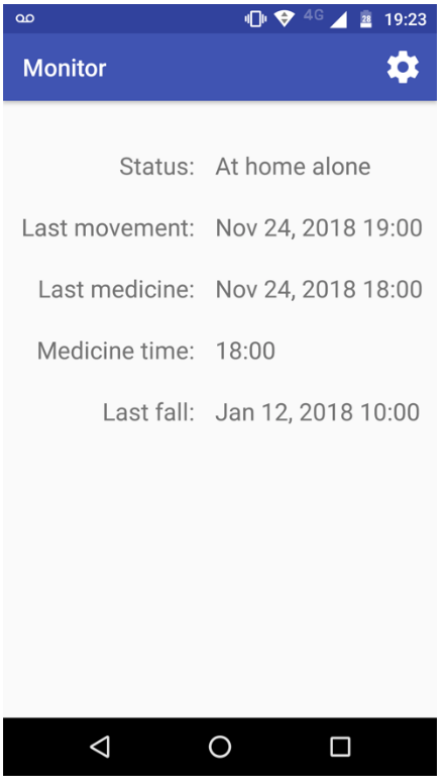
Tabela 5.1: Tabela dos valores inciais utilizados na avaliação da aplicação *mobile*.

Informação	Valor
Último evento de movimento	Nov 24, 2018 19:00
Último evento de medicação	Nov 24, 2018 18:00
Último evento de queda	Jan 12, 2018 10:00
Horário de medicação	18:00
Horário de sono (começo)	22:00
Horário de sono (fim)	08:00
Número de pessoas em casa	1

Fonte: Autor

O primeiro cenário testa a exibição das informações mais relevantes na tela inicial. Esse experimento foi realizado de forma simples, com o autor abrindo o aplicativo e conferindo os valores exibidos. A Figura 5.10 mostra a tela inicial da aplicação para esse cenário, onde pode ser verificado que as informações estão de acordo com os dados iniciais definidos.

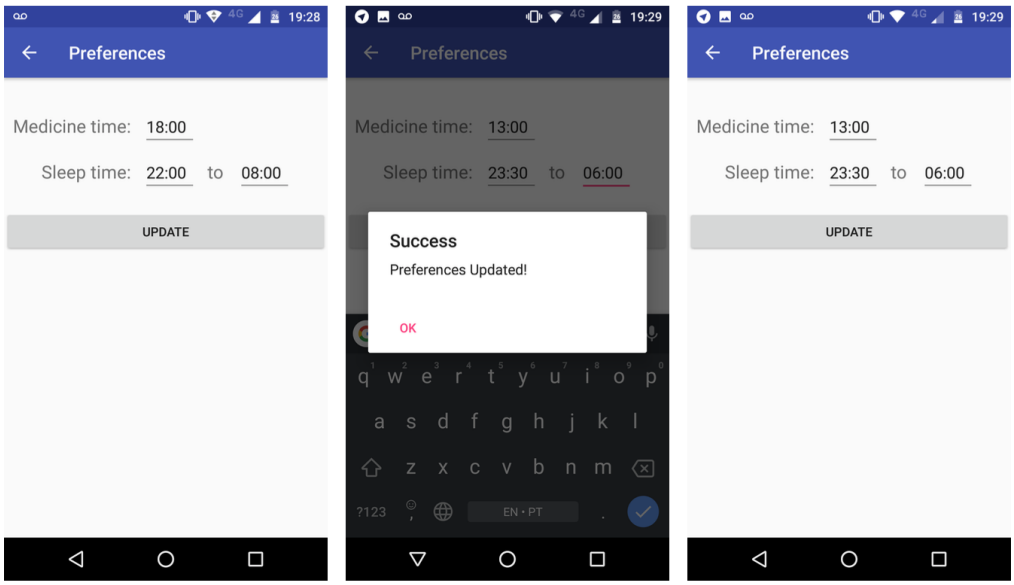
Figura 5.10: Validação da tela inicial da aplicação *mobile*.



Fonte: Autor

O segundo cenário envolve três validações, a primeira verifica se os dados exibidos ao entrar na tela de configuração das preferências são os mesmos que os valores iniciais definidos, a segunda testa a edição e atualização desses dados, e a terceira verifica novamente os dados exibidos, para conferir se são os mesmos fornecidos no momento da edição. A Figura 5.11 mostra três telas em sequência, que correspondem as três validações. Na primeira tela à esquerda, é possível ver que os valores das preferências são os mesmos definidos inicialmente no experimento. Na segunda tela ao centro, pode se ver a mensagem de sucesso ao editar esses valores. Por fim, a terceira à direita mostra os novos valores de prefrências recém atualizados.

Figura 5.11: Validação da tela de edição de preferências da aplicação *mobile*.



Fonte: Autor

5.3.3 Avaliação do verificador assíncrono de eventos

Para os experimentos referentes ao verificador assíncrono de eventos, três cenários foram realizados: dois relativos ao evento de inatividade e um ao evento de não consumo do medicamento no horário configurado. Assim como na avaliação anterior, também foi utilizado a ferramenta Postman para a simulação dos sensores e preparação dos cenários.

Para a facilitar a realização do experimento, algumas constantes definidas tiveram seu valor diminuindo, sem que a lógica desenvolvida tenha sido afetada. A Tabela 5.2 mostra os novos valores dessas constantes.

O primeiro cenário visa testar se o servidor está corretamente detectando um evento de inatividade. Para o experimento, primeiramente foi enviado através do Postman

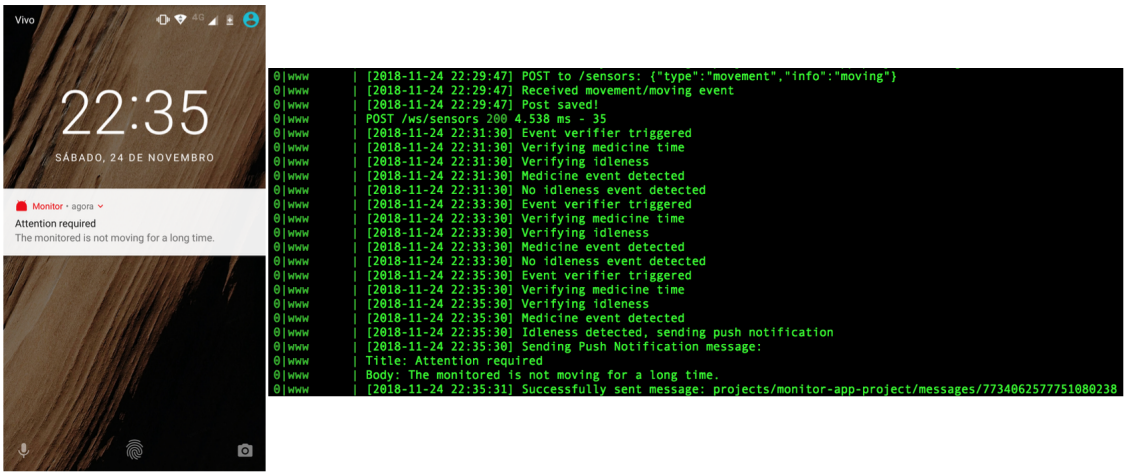
Tabela 5.2: Tabela dos novos valores utilizados para algumas constantes.

Constante	Valor (segundos)
EVENT_VERIFIER_INTERVAL	120
IDLENESS_INTERVAL	300

Fonte: Autor

um primeiro evento de atividade, para simular o último momento no qual foi detectado um movimento por parte da pessoa monitorada. Após isso, se esperou 5 minutos, que é o novo valor definido na constante IDLENESS_INTERVAL, e então foi verificado se o evento de inatividade foi detectado. É válido também notar que, como o valor da constante EVENT_VERIFIER_INTERVAL é menor que IDLENESS_INTERVAL, acontecem duas verificações adicionais que não detectam nada entre o tempo do recebimento do evento de atividade e a verificação que de fato detecta a inatividade, pois os 5 minutos ainda não tinham passado. A validação foi feita conferindo os *logs* do servidor e se a notificação foi recebida pela aplicação *mobile*, conforme é mostrado na Figura 5.12.

Figura 5.12: Validação da detecção de inatividade.



Fonte: Autor

O segundo cenário também é relacionado com o evento de inatividade, mas no caso é testada a não detecção dele durante o horário de sono configurado nas preferências. Para esse cenário, o horário de sono foi configurado para que incluísse a hora local no momento da realização do experimento, e então foi aguardado um período de 4 minutos, que corresponde a duas verificações levando em conta o novo valor da constante EVENT_VERIFIER_INTERVAL, e verificado se nenhum evento de inatividade era gerado. A Figura 5.13 mostra os *logs*, onde é possível ver que o servidor deixou de fazer qualquer detecção relacionada a inatividade exatamente devido ao fato do monitorado se encontrar no horário de sono.

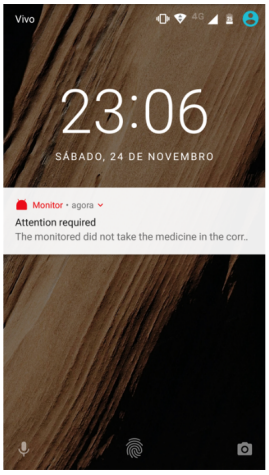
Figura 5.13: Validação da não detecção de inatividade durante o horário configurado de sono.

```
0|www   | [2018-11-24 22:49:30] Event verifier triggered
0|www   | [2018-11-24 22:49:30] Verifying medicine time
0|www   | [2018-11-24 22:49:30] Verifying idleness
0|www   | [2018-11-24 22:49:30] Monitored is in sleep time
0|www   | [2018-11-24 22:49:30] Medicine event detected
0|www   | [2018-11-24 22:51:30] Event verifier triggered
0|www   | [2018-11-24 22:51:30] Verifying medicine time
0|www   | [2018-11-24 22:51:30] Verifying idleness
0|www   | [2018-11-24 22:51:30] Monitored is in sleep time
0|www   | [2018-11-24 22:51:30] Medicine event detected
```

Fonte: Autor

Por fim, o terceiro cenário testa a detecção de quando o monitorado deixa de tomar os seus medicamentos no tempo correto. Nesse cenário, foi necessária a configuração do horário do medicamento para um horário anterior ao atual no momento do teste, com uma diferença entre os dois horários (configurado e atual) superior ao valor definido na constante POST_MEDICINE_INTERVAL. Além disso, nenhum evento de abertura da porta de medicamentos estava registrado para o dia do experimento. Depois de preparar o cenário, foi aguardado a próxima iteração do verificador, que então detectou corretamente que o monitorado deixou de tomar o remédio no horário configurado. A validação pode ser vista na Figura 5.14, onde é mostrado o *log* do servidor e a notificação recebida.

Figura 5.14: Validação da detecção do não consumo da medicação no horário correto.




```
0|www   | [2018-11-24 23:05:51] Event verifier triggered
0|www   | [2018-11-24 23:05:51] Verifying medicine time
0|www   | [2018-11-24 23:05:51] Verifying idleness
0|www   | [2018-11-24 23:05:51] Monitored is in sleep time
0|www   | [2018-11-24 23:05:51] No medicine event detected, sending push notification
0|www   | [2018-11-24 23:05:51] Sending Push Notification message:
0|www   | Title: Attention required
0|www   | Body: The monitored did not take the medicine in the correct time.
0|www   | [2018-11-24 23:05:51] Successfully sent message: projects/monitor-app-project/messages/4640352378110372563
```

Fonte: Autor

5.4 Considerações Finais

Através dos experimentos demonstrados neste capítulo, foi possível testar a aplicação de monitoramento domiciliar desenvolvida. Apesar dos cenários previstos terem sido validados de forma satisfatória, foram identificados alguns cenários de detecções

falso positivas que podem levar o servidor a fazer interpretações equivocadas. O próximo capítulo consiste na conclusão deste trabalho, onde será recapitulado o trabalho feito e também levantadas possíveis futuras melhorias para a solução.

6 CONCLUSÃO

A questão do envelhecimento populacional é uma realidade no mundo atual, e suscita questões até então não amplamente discutidas, como o conforto e a independência no dia-a-dia de uma pessoa de terceira idade. Ao mesmo tempo que essas questões são levantadas, a sociedade passa por uma fase de intenso desenvolvimento tecnológico, que possibilita que novas soluções para os mais diversos problemas sejam propostas. É esse contexto que este trabalho teve a intenção abordar, analisando dados referentes a população idosa e propondo uma solução que possa efetivamente melhorar sua qualidade de vida, usando para isso conceitos estudados de Internet das Coisas e Ambientes Inteligentes.

A solução de monitoramento remoto proposta foi inicialmente abordada em alto nível através da coleta de requisitos funcionais e não funcionais, da elaboração do diagrama de casos de uso e da prototipagem da interface da aplicação *mobile*. Na sequência, foi apresentada uma visão mais baixo nível da arquitetura da solução, assim como também foram detalhadas as principais tecnologias utilizadas e a motivação por trás de suas escolhas. Ainda na fase de implementação, foram especificados aspectos técnicos do desenvolvimento, começando pelo servidor, indo para os sensores, e finalmente chegando na aplicação *mobile*. Posteriormente, foram realizados experimentos em diversos cenários diferentes com o intuito de validar o sistema desenvolvido e verificar se ele se comporta como esperado.

Os resultados da fase de avaliação foram bastante satisfatórios, sendo todos os cenários elaborados válidados, cumprindo assim o papel proposto por este trabalho de ser uma prova de conceito para a utilização da Internet das Coisas em soluções de monitoramento domiciliar. Entretanto, foram identificadas algumas situações capazes de gerar detecções falso positivas por parte do servidor, como quando a pessoa sendo monitorada abre a porta do armário de medicamentos sem de fato consumir nenhuma medicação, ou quando ela abre a porta principal para alguém, porém esse alguém não entra em seguida no domicílio. Essas situações poderiam ser evitadas com a instalação de sensores adicionais e mais precisos, que conseguissem determinar com mais clareza se alguma porta em questão foi de fato cruzada por algo, ou alguém, ou até mesmo com o uso de um sistema de reconhecimento de imagem.

Além dos falsos positivos, no decorrer do projeto também se vislumbrou a possibilidade de outras melhorias no projeto, visando desde aprimoramentos na capacidade

de detecção de eventos, até funcionalidades adicionais para usuário da aplicação *mobile*. Essas melhorias estão listadas abaixo:

- Uso do histórico de eventos armazenado no banco de dados para análises e detecções de padrões comportamentais. Essas informações poderiam ser mostradas na aplicação no formato de gráficos e relatórios.
- Melhorias no *design* e na ergonomia dos sensores, principalmente o acelerômetro instalado no pulso da pessoa, que necessita diminuir bastante de tamanho em relação ao protótipo para se tornar confortável.
- Uso de microprocessadores mais modestos, projetados para atender apenas as necessidades específicas da aplicação, com o intuito de tornar a instalação dos sensores menos intrusiva e de diminuir custos.
- Possibilidade da pessoa monitorada tomar mais de um medicamento por dia, sendo para isso necessário o suporte para múltiplos horários de medicamento configurados.
- Novos tipos sensores, como de batimamento cardíaco, por exemplo.
- Instalação de identificadores RFID na pessoa monitorada para que, através de sensores de RFID, seja possível determinar com precisão se a pessoa está em casa ou não.
- Detecções mais elaboradas, como o de invasão de residência.
- Melhorias na interface da aplicação *mobile*, como telas mais elaboradas e disponibilizando informações para o usuário.
- Elaboração de uma interface administrativa para a configuração das constantes que o servidor usa, dando maior poder de personalização a solução.

Como dito, este trabalho teve como finalidade a elaboração e implementação de uma prova de conceito, sem pretensões de representar um produto propriamente dito. Embora as validações tenham sido feitas utilizando cenários reais, foram considerados situações ideais, sendo que em um ambiente de produção novas falhas poderiam ser detectadas. As melhorias descritas acima foram consideradas fora do escopo prático do projeto, porém poderiam ser implementadas para que a solução se tornasse mais robusta e confiável.

REFERÊNCIAS

ADAFRUIT. **Adafruit Python GPIO Library**. 2015. <https://github.com/adafruit/Adafruit_Python_GPIO> (Visitado em 04/11/2018).

AGOSTINI, J. V.; BAKER, D. I.; BOGARDUS, S. T. Prevention of falls in hospitalized and institutionalized older people. In: . [S.l.]: The Agency for Healthcare Research and Quality, 2001.

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. **Communications Surveys Tutorials**, v. 17, n. 4, p. 2347–2376, 2015.

ANALOG DEVICES, INC. **Data Sheet Digital Accelerometer ADXL345**. One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A, 2009. Rev. E.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, 2010.

BERTOLINO, A. et al. Use Case Description of Requirements for Product Lines. In: **International Workshop on Requirements Engineering for Product Lines**. [S.l.: s.n.], 2002. p. 12–18.

BOHN, D. A. Environmental Effects on the Speed of Sound*. **J. Audio Eng. Soc.**, Audio Engineering Society, v. 36, n. 4, 1988.

BORMANN, C.; CASTELLANI, A. P.; SHELBY, Z. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. **IEEE Internet Computing**, v. 16, n. 2, p. 62–67, 2012.

BUDDE, R. et al. Prototyping. In: ____. [S.l.]: Springer, Berlin, Heidelberg, 1992. chp. Prototyping.

Centers for Disease Control and Prevention. Web-based Injury Statistics Query and Reporting System (WISQARS). In: . [S.l.]: National Center for Injury Prevention and Control, Centers for Disease Control and Prevention (producer), 2015.

CHODOROW, K. Mongodb: The definitive guide: Powerful and scalable data storage. In: ____. [S.l.]: O'Reilly Media, Inc, 2013. chp. Introduction MongoDB.

CHUNG, L.; LEITE, J. C. S. do P. On Non-Functional Requirements in Software Engineering. **Conceptual Modeling: Foundations and Applications. Lecture Notes in Computer Science**, v. 5600, 2009.

CHUNG, L. et al. Non-functional requirements in software engineering. In: ____. [S.l.]: Springer Science Business Media, 2012. chp. Introduction.

Commission of the European Communities, An i2010 Initiative. **Ageing well in the Information Society**. 2007. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52007DC0332&from=EN> (Visitado em 26/11/2018).

COOK, D. J.; AUGUSTO, J. C.; JAKKULA, V. R. Ambient intelligence: Technologies, applications, and opportunities. **Pervasive and Mobile Computing**, v. 5, n. 4, p. 277–298, 2009.

DARWIN, I. F. Android cookbook: Problems and solutions for android developers. In: ____. [S.l.]: O'Reilly Media, Inc, 2017. chp. Preface.

DUNKELS, A.; VASSEUR, J. P. **IP for smart objects (IPSO Alliance White Paper)**. 2008. <http://dunkels.com/adam/dunkels08ipso.pdf> (Visitado em 26/1/2017).

ELECFREAKS. **Data Sheet Ultrasonic Ranging Module HC - SR04**. 2010. <https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf> (Visitado em 26/11/2018).

EVANS, D. **The Internet of Things: How the Next Evolution of the Internet Is Changing Everything (Cisco White Paper)**. 2011. <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf> (Visitado em 26/11/2017).

GOOGLE. **Firebase Documentation**. 2018. <https://firebase.google.com/docs/> (Visitado em 27/11/2018).

GUBBI, J. et al. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645–1660, 2009.

HSIA, P.; DAVIS, A.; KUNG, D. Status report: requirements engineering. **IEEE Software**, IEEE, v. 10, n. 6, p. 75–79, 1993.

HUNKELER, U.; TRUONG, H. L.; STANFORD-CLARK, A. MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In: **3rd International Conference on Communication Systems Software and Middleware and Workshops**. [S.l.: s.n.], 2008.

IEEE. **IEEE Std 610.12-1990 - Standard Glossary of Software Engineering Terminology**. 1990.

INSEL, K. et al. Executive Function, Working Memory, and Medication Adherence Among Older Adults. **J Gerontol B Psychol Sci Soc Sci**, v. 61, n. 2, p. 102–107, 2006.

IOT-A. **The Internet-of-Things Architecture**. 2012. <http://www.meet-iot.eu/deliverables-IOTA/D1_3.pdf> (Visitado em 29/06/2017).

ITU. **New ITU standards define the internet of things and provide the blueprints for its development**. 2012. <http://www.itu.int/ITU-T/newslog/New+ITU+Standards+Define+The+Internet+Of+Things+And+Provide+The+Blueprints+For+Its+Development.aspx> (Visitado em 14/06/2017).

MATTHEWS, C. E. et al. Amount of Time Spent in Sedentary Behaviors in the United States. **American Journal of Epidemiology**, v. 167, n. 7, p. 875–881, 2008.

MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, v. 10, n. 7, p. 1497–1516, 2012.

MongoDB Inc. **MongoDB Documentation**. 2018. <https://docs.mongodb.com/> (Visitado em 27/11/2018).

MULLIGAN, G. The 6LoWPAN architecture. In: **4th workshop on Embedded networked sensors**. [S.l.: s.n.], 2006. p. 78–82.

National Center for Health Statistics. National Health Interview Survey's 1984 Supplement on Aging. In: . [S.l.]: Public Health Service. Washington. U.S. Government, 1987.

Node.js Foundation. **Node.js Documentation**. 2018. <https://nodejs.org/en/docs/> (Visitado em 27/11/2018).

NXP Semiconductors. **I²C-bus specification and user manual**. 2014. <https://www.nxp.com/docs/en/user-guide/UM10204.pdf> (Visitado em 27/11/2018).

OASIS. **MQTT Version 3.1.1**. 2014. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html> (Visitado em 29/06/2017).

PAETSCH, F.; EBERLEIN, A.; MAURER, F. Requirements engineering and agile software development. In: **WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises**. [S.l.: s.n.], 2003. p. 308–313.

Raspberry Pi Foundation. **Raspberry Pi Documentation**. 2018. <https://www.raspberrypi.org/documentation/> (Visitado em 27/11/2018).

RAZZAQUE, M. A. et al. Middleware for Internet of Things: A Survey. **ceE Internet of Things Journal**, v. 3, n. 1, p. 70–95, 2016.

ROMDHANI, I.; ABDMEZIEM, R.; TANDJAOUI, D. Robots and sensor clouds. In: ____. Special. [S.l.]: Springer, 2015. (Studies in Systems, Decision and Control), chp. Architecting the Internet of Things: State of the Art.

RUYTER, B. de; AARTS, E. Ambient Intelligence: visualizing the future. In: **Proceedings of the working conference on Advanced visual interfaces**. [S.l.: s.n.], 2004.

SENSEI. **SENSEI - Integrating the Physical with the Digital World of the Network of the Future**. 2008. <http://cordis.europa.eu/pub/fp7/ict/docs/future-networks/projects-sensei-ec-summary_en.pdf> (Visitado em 29/06/2017).

SETHI, P.; SARANGI, S. R. Internet of Things: Architectures, Protocols, and Applications. **Journal of Electrical and Computer Engineering**, 2017.

SOMÉ, S. S. Supporting use case based requirements engineering. **Information and Software Technology**, Elsevier, v. 48, n. 1, p. 43–58, 2006.

Texas Instruments. **KeyStone Architecture Serial Peripheral Interface (SPI)**. 2012. <http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf> (Visitado em 27/11/2018).

TILKOV, S.; VINOSKI, S. Node.js: Using JavaScript to Build High-Performance Network Programs. **IEEE Internet Computing**, IEEE, v. 14, n. 6, p. 80–83, 2010.

United Nations. **World Population Prospects: The 2015 Revision**. [S.l.], 2015.

VILLAVERDE, B. C.; PESCH, D.; ALBEROLA, R. D. P. Constrained Application Protocol for Low Power Embedded Networks: A Survey. In: **4th workshop on Embedded networked sensors**. [S.l.: s.n.], 2012.

VINEELA, A.; RANI, L. S. Internet of Things -Overview. **International Journal of Research in Science Technology**, 2015.

WANG, R.; WANG, J.; WANG, N. Analysis of key technologies in the Internet of things. In: **3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME 2015)**. [S.l.: s.n.], 2015.

WORTMANN, F.; FLüCHTER, K. Internet of Things: Technology and Value Added. **Springer Fachmedien Wiesbaden**, 2015.

WU, M. et al. Research on the architecture of internet of things. In: **3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)**. [S.l.: s.n.], 2010. v. 5, p. 484–487.

ZAVE, P. Classification of Research Efforts in Requirements Engineering. **ACM Computing Surveys**, v. 29, n. 4, p. 315–321, 1997.

ANEXO A - TRABALHO DE GRADUAÇÃO I
Assistência domiciliar para pessoas de terceira idade através de conceitos da Internet das Coisas

Felipe A. S. Nogueira[1], **Alexandre Carissimi**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{fasnogueira,asc}@inf.ufrgs.br

Abstract. *One of the biggest concerns that people have when reach the old age is the capability to remain independent in their daily life routine, as the advanced age can bring health problems that may turn ordinary tasks in risk situations, if not supervised. In this context, the concepts of Internet of Things (IoT) can be used to monitor and measure home activities of elderly people through sensors strategically placed along the house. This paper intends to discuss and propose a solution to assist senior citizens to be supervised in a safe, but not overly intrusive, way.*

Resumo. *Uma das principais preocupações que as pessoas têm quando alcançam a terceira idade é a capacidade de se manter independente em suas rotinas diárias, já que a idade avançada pode trazer problemas de saúde que podem transformar tarefas cotidianas em situações de risco, se não supervisionadas. Nesse contexto, os conceitos da Internet das Coisas podem ser usados para monitorar e medir atividades domésticas de pessoas idosas através de sensores estrategicamente instalados ao longo da casa. Este trabalho tem como objetivo discutir e propor uma solução para auxiliar idosos a serem supervisionados de forma segura, mas não demasiadamente intrusiva.*

1. Introdução

Segundo a ONU (Organização das Nações Unidas), a raça humana presencia um envelhecimento populacional sem precedentes em sua história, e que tende a se acentuar em um ritmo ainda mais acelerado durante o século XXI. É estimado que, até 2050, cerca de 21% da população terá mais do que 60 anos, passando o número de 2 bilhões de pessoas. No Brasil, essa porcentagem é prevista em 29% [United Nations 2015]. Entre os principais fatores para essa mudança no perfil demográfico mundial estão o aumento da expectativa de vida, a redução da taxa de natalidade e as notórias melhorias nas condições sanitárias e de saúde.

Como consequência dessas projeções, a qualidade de vida de pessoas de terceira idade é cada vez mais uma discussão presente na sociedade. Além de já terem que conviver com as complicações naturais provenientes da idade avançada, essas pessoas acabam, muitas vezes, tendo sua liberdade e independência afetadas pela necessidade de monitoramento, e pelos riscos de morarem sozinhas. Essa condição é ainda acentuada quando se leva em conta que normalmente são pessoas que estavam há muito tempo acostumadas a não depender de ninguém.

Uma queda pode ser definida como involuntariamente ir ao chão, ou a alguma outra superfície de nível mais baixo, sem que seja o resultado de uma síncope ou de uma intensa força externa [Agostini et al. 2001]. De acordo com dados publicados pela CDC (*Centers for Disease Control and Prevention*), quedas representam a principal causa de morte associada a acidentes não intencionais, e a nona causa de morte dentre todas as causas, em pessoas com mais de 65 anos [Centers for Disease Control and Prevention 2015]. Na Figura 1, é possível verificar dados de uma pesquisa norte-americana, onde cerca de um quarto das pessoas com mais de 75 anos reportaram algum tipo de queda no ano anterior, e que esse número tende a crescer com o avanço da idade.

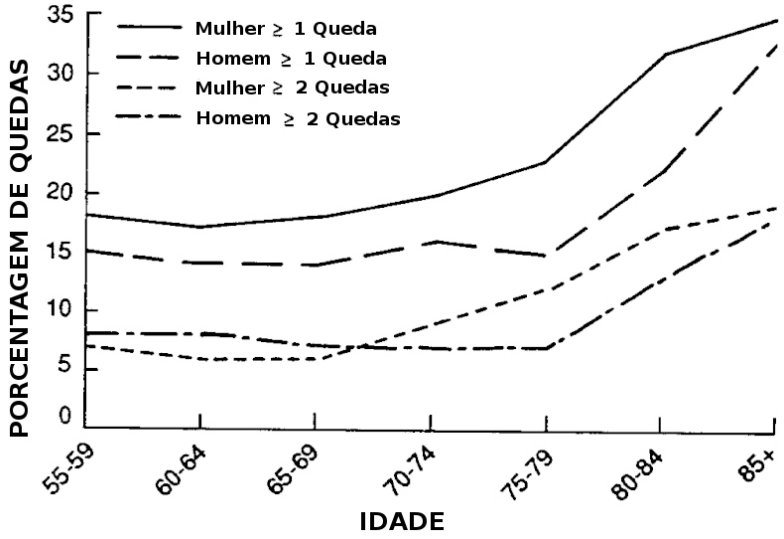


Figura 1. Porcentagem de indivíduos idosos que reportaram quedas nos últimos 12 meses, agrupados por idade e sexo.
Fonte: [National Center for Health Statistics 1987]

Em uma pesquisa feita com 6.329 participantes entre 2003 e 2004, foi constatado que a população de terceira idade corresponde ao grupo de idade onde comportamentos sedentários são mais comuns. Adultos na faixa de 70-85 anos passam cerca de 67% de suas horas ativas em atividades sedentárias [Matthews et al. 2008]. Esses resultados são compreensíveis quando se leva em conta as limitações da idade, porém eles também mostram que essas pessoas necessitam, mais do que qualquer outro grupo etário, de incentivos para superar essa barreira e ter uma rotina mais saudável.

O envelhecimento populacional reflete também os avanços na medicina. Juntamente com o avanço etário, surgem inúmeras patologias e, cada vez mais, as pessoas idosas têm a possibilidade de recorrer a medicamentos modernos visando o prolongamento de suas vidas. Sendo assim, um outro aspecto que merece atenção é a taxa de adesão dessas pessoas a esses medicamentos, visto que é essencial que os mesmos sejam tomados corretamente e na frequência prescrita pelo médico. Para se ter um parâmetro, um estudo acerca desse tema, com indivíduos com uma média de idade de 75 anos, foi realizado por [Insel et al. 2006], onde 38% dos participantes tiveram uma adesão aos medicamentos menor que 85%. Existem diversas motivações para essa falta de adesão, como, por exemplo, esquecimento e não concordância com o tratamento, mas é de suma importância que ao menos o conhecimento dessa condição exista por parte de outras pessoas próximas.

Como esses dados reforçam, a necessidade de cuidado domiciliar é uma realidade na vida da maioria dos idosos. Esse cuidado pode ser dado por familiares ou pessoas próximas, porém os mesmos nem sempre possuem a disponibilidade necessária para esse tipo de tarefa. Outra alternativa seria contratar um serviço terceirizado, o que acaba sendo custoso e, até mesmo, desconfortável, por envolver profissionais da saúde até então desconhecidos. É nesse contexto que surge uma oportunidade para o avanço tecnológico, juntamente com a interconectividade que a Internet nos traz, poder prover uma solução que melhore a qualidade de vida dessas pessoas e dê mais autonomia às suas rotinas diárias.

A instrumentalização do ambiente domiciliar pode propiciar um monitoramento controlado, dando independência ao idoso e tranquilidade aos seus parentes. Além disso, o uso dessas tecnologias também possibilita a coleta de dados sobre a rotina e hábitos diários, que podem servir para alertar sobre comportamentos não saudáveis ou identificar situações anômalas. Por exemplo, através de sensores poderiam ser detectadas quedas e outros eventos que necessitem de algum tipo de amparo imediato, notificando as pessoas responsáveis. Também seria possível analisar os dados coletados, com o intuito de reconhecer comportamentos de risco, como a não adesão de remédios ou padrões sedentários.

O objetivo deste trabalho é propor uma solução de monitoramento e assistência domiciliar para pessoas de terceira idade, utilizando conceitos como da Internet das Coisas (*IoT - Internet of Things*) e de Inteligência Ambiental (*AmI - Ambient Intelligence*). Essa solução utilizará sensores de movimento e aceleração, interconectados entre si através de um controle central, estrategicamente instalados ao longo do ambiente domiciliar e com o intuito de permitir uma supervisão remota, coletando dados sensíveis e detectando situações de risco. Esses dados serão enviados a um serviço na nuvem, e poderão ser acessados a distância por outras pessoas responsáveis. O principal foco é possibilitar esse monitoramento necessário sem afetar demasiadamente a independência e intimidade desses idosos.

2. Internet das Coisas

A Internet das Coisas é um conceito relativamente novo, porém vem se mostrando como uma das principais tendências tecnológicas do início deste século. Ainda não existe uma definição unânime para esse conceito, mas a ideia básica por trás é a de que a diversidade de objetos, ou coisas, em volta da gente – como RFID (*Radio-Frequency IDentification*), sensores, atuadores, celulares, veículos, etc. – seja capaz de interagir entre si e cooperar com seus vizinhos para alcançar um objetivo em comum [Atzori et al. 2010]. Por exemplo, a ITU (*International Telecommunication Union*) define a Internet das Coisas como "uma infraestrutura global para a sociedade da informação, que possibilita serviços avançados através da interconexão (física e virtual) das coisas, baseados em informação interoperável e tecnologias de comunicação, existentes e em evolução"[ITU 2012].

O termo *Internet of Things* pode ter suas origens rastreadas até o grupo de pesquisa Auto-ID Center, do MIT (*Massachusetts Institute of Technology*). Fundado em 1999, esse grupo trabalhava na área de redes interligadas por identificadores de radiofrequência (RFID). O Auto-ID Center consistia de sete instituições de pesquisa localizadas em quatro continentes, que foram selecionadas para projetar a primeira arquitetura do que viria a ser conhecido como Internet das Coisas [Evans 2011].

Desde então, a abrangência da Internet das Coisas vai muito além do escopo de tecnologias RFID. Atualmente, suas aplicações são inúmeras e extremamente diversificadas, se estendendo a virtualmente todas as áreas do nosso dia a dia. Pode se citar, por exemplo, aplicações na área da domótica, que se refere a automação do ambiente domiciliar, de fábricas inteligentes, que também pode ser conhecia como Indústria 4.0, da assistência médica, ajudando no monitoramento de pacientes, de cidades inteligentes, coletando dados a respeito do tráfego, poluição, transporte, etc., entre outras aplicações e conceitos que derivam de tecnologias IoT [Wortmann and Flüchter 2015]. A Figura 2 ilustra esses diferentes domínios de aplicações existentes.

Figura 2. Domínios de aplicações e cenários mais relevantes.
Fonte: [Atzori et al. 2010]

2.1. Visão Geral

A palavra "Internet", no termo Internet das Coisas, pode ser interpretada de duas formas. No sentido metafórico, expressando a ideia de que, da mesma forma que passamos a maior parte do tempo conectados a *Web*, em um futuro próximo as coisas também estarão interconectadas entre si, utilizando serviços e fornecendo dados. Já no sentido técnico, existe a visão de que o protocolo IP será usado por objetos físicos, ou seus respectivos *proxies*, para se comunicarem entre si, sendo literalmente incorporados a Internet [Vineela and Rani 2015].

No âmbito de uma visão mais técnica, esses objetos físicos devem possuir três principais habilidades: serem identificáveis, serem capazes de se comunicar e serem capazes de interagir, seja entre si ou com outras entidades da rede [Miorandi et al. 2012]. É baseado nessa visão que aplicações e soluções, antes não imaginadas, estão sendo desenvolvidas, impactando diretamente a vida de diversas pessoas e também criando desafios no âmbito tecnológico.

No nível de componente, a Internet das Coisas é constituída, como o próprio nome diz, por coisas. Essas coisas podem ser vistas como objetos inteligentes. Segundo [Miorandi et al. 2012], um objeto inteligente pode ser definido como uma entidade que possua:

- Um corpo físico, com tamanho, formato e etc.
- Um conjunto mínimo de funcionalidades para comunicação, como a capacidade de receber e responder mensagens.
- Um identificador único.
- Associações a pelo menos um nome e um endereço, de forma que o objeto possa ser encontrado na rede.
- Capacidades computacionais básicas, podendo ser desde a habilidade de receber mensagens a, até mesmo, realizar operações mais complexas.
- Sensores capazes de medir e guardar informações referentes ao meio no qual o objeto está inserido.

A inserção desses objetos inteligentes na Internet cria diversos desafios no aspecto de manter a interoperabilidade da rede. Muitos desses componentes possuem capacidades computacionais restritas, e, muitas vezes, não podem operar com todas as camadas de protocolos que sistemas finais operam. Sendo assim, a Internet das Coisas revela uma rede com foco em informações do mundo físico, altamente dinâmica e distribuída, com entidades agindo como provedores e/ou consumidores, e não em comunicações fim-a-fim [Miorandi et al. 2012].

2.2. Arquitetura

Não há um consenso a respeito da arquitetura para IoT, até mesmo por ser uma tecnologia ainda em nível de amadurecimento e consolidação. Existem alguns esforços no aspecto de criar definições e modelos de referência para esse tipo de arquitetura, partindo de um perspectiva de WSN (*Wireless Sensor Networks*), e se baseando em análises sobre as necessidades de pesquisadores e da indústria, como por exemplo o projeto IoT-A [IoT-A 2012] e o SENSEI [SENSEI 2008], este último conduzido pela União Europeia [Gubbi et al. 2009].

A Figura 3 ilustra alguns modelos baseados em camadas já propostos. Atualmente, o mais aceito é o de três camadas [Wu et al. 2010], que oferece uma boa visão alto nível da arquitetura IoT. As três camadas são: camada de percepção, camada de rede e camada de aplicação. O nível de complexidade de cada camada vai depender do contexto da aplicação e de suas necessidades.

A camada de percepção é a camada física, responsável principalmente por detectar outros objetos e coletar dados referentes ao meio em que o dispositivo está inserido [Sethi and Sarangi 2017]. Esse primeiro estágio de detecção pode se basear em tecnologias como RFID, WSN, GPS (*Global Positioning System*), NFC (*Near Field Communication*), etc., enquanto que para a coleta de dados podem ser utilizados toda uma diversidade de sensores e atuadores. Além disso, a camada de percepção também é responsável pela conversão desses dados em sinais digitais, que são mais convenientes para a subsequente transmissão dos mesmos [Romdhani et al. 2015].

A principal função da camada de rede é transmitir e processar a informação obtida pela camada de percepção. Assim que um sensor capta algum tipo de dado, ele, em

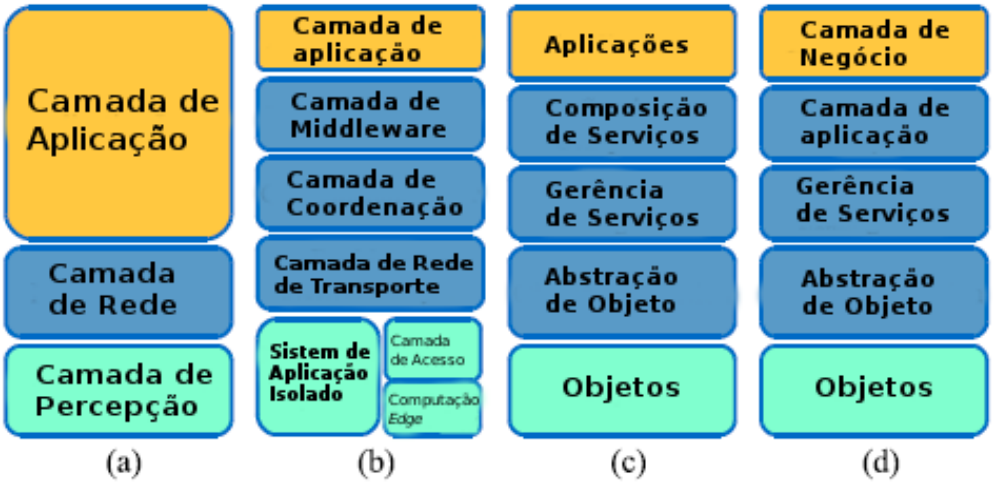


Figura 3. Modelos de arquitetura IoT: (a) Três camadas (b) Baseado em *middleware* (c) Baseado em SOA (*Service-Oriented Architecture*) (d) Cinco camadas.
Fonte: [Al-Fuqaha et al. 2015]

seguida, o envia para um segundo plano, onde é feito o devido tratamento e transmissão desse dado, e isso se dá através da camada de rede. Nessa camada, se encontram mecanismos de gerência da rede, possibilitando a conexão e a comunicação com outros objetos e/ou sistemas finais. As principais mídias para essa comunicação incluem FTTx, 3G/4G, Wifi, Bluetooth, ZigBee, infravermelho, entre outras.

Por último, a camada de aplicação, onde é oferecido aos usuários serviços específicos da aplicação em questão. É nessa camada que, ao interpretar e entregar funcionalidades, se cria valor para os dados gerados pelas outras camadas, o que pode ser visto como o objetivo final do desenvolvimento de softwares no contexto da Internet das Coisas [Wang et al. 2015]. Esses serviços podem ser oferecidos em diversas plataformas, como *web* ou *mobile*, que podem estar hospedadas em uma estrutura na nuvem, por exemplo.

2.3. Protocolos

A comunicação na Internet é em grande parte possibilitada pela utilização de diversos protocolos padrões, que criam convenções para como seus *hosts* devem interagir entre si. As aplicações da Internet das Coisas estão sendo inseridas nesse meio e, sendo assim, nada mais intuitivo que usar parte dessa infraestrutura e convenções já existentes.

Segundo a *Internet Protocol for Smart Objects Alliance* (IPSO), o protocolo IP tem se mostrado uma tecnologia duradoura, estável e altamente escalável, sendo capaz de suportar uma grande escala de aplicações e dispositivos [Dunkels and Vasseur 2008]. Nesse contexto, o grupo de trabalho 6LoWPAN da IETF (*Internet Engineering Task Force*) criou uma série de especificações e mecanismos que permitem que pacotes IPv6 sejam carregados sobre redes baseadas no padrão IEEE 802.15.4, que define a operação de redes sem fio pessoais de baixas taxas de transmissão, ideais para aplicações IoT. O 6LoWPAN foi originado no conceito de que o protocolo IP deveria e poderia ser aplicado até mesmo nos menores dispositivos [Mulligan 2006].

As aplicações na Internet são, geralmente, implementadas baseando-se no proto-

colo HTTP (*Hypertext Transfer Protocol*). Porém, diferentemente do caso do protocolo IP, o protocolo HTTP não é considerado adequado para o contexto de ambientes com recursos limitados, como é o caso da Internet das Coisas, pois é demasiadamente verboso por natureza, o que acaba gerando um grande *overhead* [Sethi and Sarangi 2017]. Por conseguinte, diversos protocolos alternativos foram desenvolvidos visando resolver esse problema, como por exemplo CoAPP, o MQTT, entre outros.

2.3.1. CoAP

O *Constrained Application Protocol* (RFC 7252) se assemelha ao próprio protocolo HTTP, porém possui otimizações e mecanismos que visam obter um bom desempenho em ambientes de recursos limitados. Ele opera em cima do UDP (*User Datagram Protocol*), e oferece funcionalidades como *multicast*, controle de congestionamento e troca assíncrona de mensagens.

O CoAP se baseia na arquitetura REST (*Representational State Transfer*), o que facilita o mapeamento com o protocolo HTTP. Os recursos no CoAP são identificados por URIs (*Uniform Resource Identifier*) e são acessados por métodos similares aos do HTTP: GET, POST, PUT e DELETE. Existem três tipos de códigos de resposta: 2xx (*success*), 4xx (*client error*) e 5xx (*server error*). Além disso, como o CoAP utiliza o protocolo UDP, ele tem a necessidade de definir seu próprio mecanismo de confiabilidade, disponibilizando mensagens de confirmação. O protocolo DTLS (*Datagram Transport Layer Security*) também é utilizado para atender requisitos de segurança, provendo privacidade, autenticidade e integridade.

Além do modelo *request/respond*, implementado pelos métodos HTTP, o CoAP também suporta o modelo *publish/subscribe*, que possibilita *multicasting*. Nesse modelo, um cliente se inscreve para ser atualizado quando um recurso tem alguma mudança, evitando assim o *overhead* de ter que ficar constantemente pedindo atualizações do estado desse recurso [Villaverde et al. 2012].

2.3.2. MQTT

O *Message Queuing Telemetry Transport* [OASIS 2014] é um protocolo de mensagens leve, desenvolvido pela IBM ainda na década de 90. Foi originalmente projetado para operar em lugares remotos, em que seja necessário enviar dados através de redes com banda limitada. Por causa disso, o protocolo reúne diversos conceitos também pertinentes a redes em ambientes com recursos limitados, se tornando assim um protocolo adequado para aplicações IoT.

O MQTT executa em cima de TCP (*Transmission Control Protocol*) e utiliza um modelo *publish/subscribe* para troca de mensagens. Nesse modelo, um cliente pode publicar, ou se inscrever, para um determinado tópico, que é análogo ao conceito de URI. A gerência do recebimento, enfileiramento e envio das mensagens é responsabilidade do *broker*, que basicamente é um *middleware* que opera como intermediário das mensagens enviadas. Fica a cargo dos *subscribers* se conectarem com o *broker* para receberem as mensagens das quais possuem interesse. O MQTT disponibiliza três níveis de QoS (*Qua-*

lity of Service) em suas conexões:

- QoS 0 (*at most once*): é um serviço de melhor esforço (*best effort*), onde as mensagens são enviadas no máximo uma vez, e nenhuma mensagem de *acknowledgment* é definida.
- QoS 1 (*at least once*): As mensagens são retransmitidas até que uma mensagem de *acknowledgment* seja recebida, o que pode causar que mensagens duplicadas cheguem ao destino.
- QoS 2 (*exactly once*): Nesse nível, além da garantia do recebimento da mensagem, também é garantido que apenas uma mensagem seja recebida pelo destino.

É importante salientar que é responsabilidade do cliente informar ao *broker* qual o nível de QoS desejado, o que possibilita que existam níveis de QoS diferentes para um mesmo tópico [Hunkeler et al. 2008].

2.4. Middlewares

Um *middleware* tem a função de abstrair as complexidades de um sistema, ou de um *hardware*, para que o desenvolvedor possa focar inteiramente na solução final, sem ter que se preocupar com problemas não diretamente pertinentes ao escopo da aplicação [Razzaque et al. 2016]. Geralmente, essas complexidades são relacionadas a comunicação, ou a questões mais específicas do sistema. Em um meio tecnológico tão heterogêneo como o da Internet das Coisas, o uso de *middlewares* tem se tornado cada vez mais comum.

A arquitetura e as funcionalidades de cada *middleware* vão depender dos problemas e dificuldades que cada um se propõe a resolver. Entre os principais desafios encontrados estão: interoperabilidade entre dispositivos, descoberta de dispositivos e serviços que se encontrem na vizinhança, escalabilidade, *big data*, segurança e privacidade, compatibilidade com serviços na nuvem, interpretação de contexto para os dados coletados, etc. [Sethi and Sarangi 2017].

Como dito acima, existem diversas questões a serem abordadas, dando espaço ao aparecimento de diversos tipos de soluções para *middlewares*, cada um com um foco diferente. Essas soluções podem ser classificadas em cinco grupos, de acordo com seu projeto [Razzaque et al. 2016]:

- Baseado em eventos: Nesse modelo, os componentes interagem entre si através de eventos. Cada evento possui um tipo e um conjunto de parâmetros específicos, cujos valores descrevem as mudanças no estado do produtor do evento. Os eventos são gerados pelos produtores e processados pelos consumidores, o que pode ser visto como um modelo *publish/subscribe*, onde entidades se inscrevem para receber notificações de um evento em particular.
- Orientado a serviço: Um SOM (*Service-oriented middleware*) é baseado na arquitetura SOA (*Service Oriented Architecture*), onde existem módulos independentes que oferecem serviços através de interfaces accessíveis. Os recursos são vistos como provedores de serviços e são oferecidos em um repositório de serviços, onde consumidores podem descobri-los e utilizá-los conforme a necessidade.
- Semântico: Esse tipo de *middleware* tem como principal objetivo a interoperabilidade entre os diferentes tipos de dispositivos. A ideia é criar uma semântica em comum na comunicação dos dispositivos, utilizando adaptadores para mapear

os diferentes formatos de dados e, assim, possibilitar que recursos distintos se comuniquem entre si, independentemente do protocolo que utilizem.
- Orientado a banco de dados: Aqui, os componentes de rede dos dispositivos IoT são vistos como um sistema de banco de dados relacional virtual. Dessa forma, para acessar os recursos, as aplicações devem fazer requisições usando uma linguagem de consulta (*query language*).
- Específico para aplicação: Esse tipo de *middleware* é desenvolvido especificamente para algum tipo de aplicação. Nesse caso, existe um grande nível de acoplamento entre o *middleware* e a aplicação.

A escolha do tipo de *middleware* vai depender do teor e das necessidades da aplicação. Fazer uma escolha adequada pode significar uma grande economia de tempo durante a fase de desenvolvimento.

3. Inteligência Ambiental

O termo Inteligência Ambiental (*AmI - Ambient Intelligence*) foi originalmente proposto em 1998 em uma apresentação feita por Eli Zelkha e Brian Epstein, durante uma série de *workshops* encomendados pelo conselho de administração da empresa Philips. Os *workshops* tinham como objetivo discutir possíveis futuros cenários, que pudessem vir a utilizar em larga escala a industria de eletrônicos da época [de Ruyter and Aarts 2004].

AmI se refere a ambientes eletrônicos que sejam sensíveis e responsivos a presença de pessoas, e a eventos e condições externas. Isso geralmente é alcançado através de sensores e atuadores estrategicamente instalados ao longo do ambiente, e que sejam capazes de se comunicar entre si, ou a uma central de gerenciamento. Esses sensores podem ser implementados utilizando conceitos da Internet das Coisas, consolidando uma evidente possibilidade de integração entre os dois conceitos. Está também fortemente ligada ao conceito de computação ubíqua, que tem como objetivo tornar a interação homem-computador transparente, ao integrar a computação a rotina e aos afazeres das pessoas.

Essas definições da Inteligência Ambiental podem ser sumarizadas em seis principais características: sensibilidade ao meio, responsividade, adaptabilidade, transparência, comportamento ubíquo e inteligência [Cook et al. 2009]. As soluções em AmI não precisam necessariamente atender a todas essas características, mas são os principais aspectos que definem um ambiente inteligente.

Existem diversas oportunidades de aplicações para a Inteligência Ambiental, e a tendência é que, em um futuro próximo, esse tipo de tecnologia esteja presente em vários aspectos do cotidiano humano. Umas das oportunidades que recebem mais pesquisa e atenção é a assistência médica, dado o eminente envelhecimento da população. Porém, não é só baseado no conforto e bem estar de pessoas de terceira idade, o motivo para que esse tipo de aplicação receba tanta atenção. De acordo com [Commission of the European Communities 2007], cerca de 1,5 bilhões de Euros poderiam ser economizados todo ano com a introdução de monitoramento remoto. Outro dado interessante é que, no Reino Unido, cuidados institucionais custam cerca de 21.840,00 libras por domicílio, enquanto que a utilização do *telecare* (assistência à distância a idosos) custaria apenas 7.121,00 libras.

Esse aprimoramento na qualidade de vida de pessoas idosas através da AmI pode ser alcançado através de diferentes abordagens. Uma delas poderia ser prover uma forma

de garantia de segurança, monitorando qualquer evento suspeito, como quedas, frequência de exercícios, dietas, etc., de forma a tranquilizar familiares próximos. Outra abordagem seria providenciar suporte a pessoas com problemas cognitivos, fazendo lembretes de tarefas e atividades importantes de suas rotinas. Também podem ser feitas avaliações a respeito do progresso dessas doenças cognitivas, como por exemplo avaliar a eficiência dessas pessoas na hora de realizar tarefas cotidianas. Por fim, tendo em mente que a idade avançada traz consigo muitas vezes o isolamento social, seria possível analisar as atividades sociais do dia a dia e, com esses dados, fornecer conselhos de como esse aspecto poderia ser aprimorado [Cook et al. 2009].

4. Proposta

A proposta deste trabalho final é desenvolver uma solução que permita prover assistência domiciliar remota a pessoas de terceira idade, utilizando conceitos estudados de IoT e AmI. Basicamente, essa solução consistirá de sensores, estrategicamente instalados ao longo do domicílio, que irão coletar dados e fornecê-los através de um serviço na nuvem, que por sua vez será consumido por uma aplicação acessada por uma outra pessoa, possibilitando assim o monitoramento remoto.

Uma representação em alto nível da arquitetura proposta para solução está ilustrada na Figura 4. Essa arquitetura foi baseada no modelo de três camadas apresentado anteriormente.

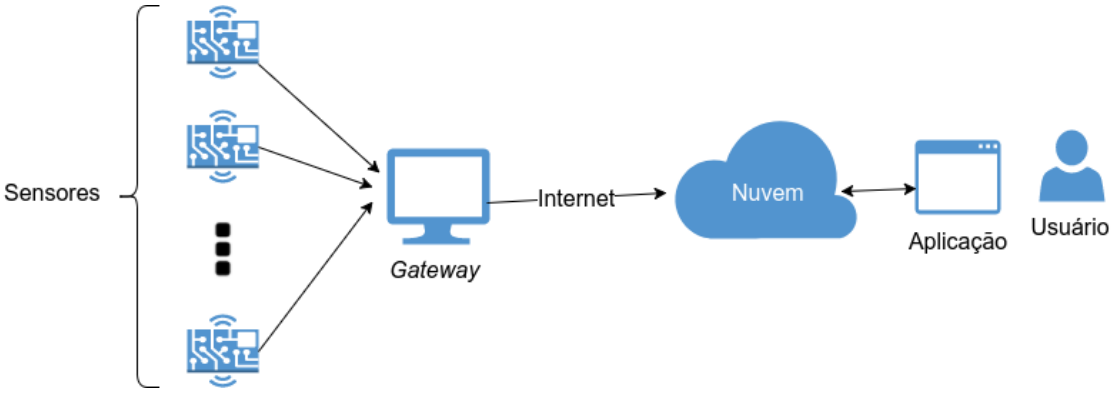


Figura 4. Representação em alto nível da arquitetura proposta.
Fonte: [Autor]

A camada de percepção é constituída pelos sensores, que têm a função de coletar dados sensíveis e de se comunicar com o *gateway*. Um acelerômetro será utilizado para a detecção de quedas, e deverá, portanto, estar instalado em alguma região do corpo da pessoa monitorada, como o pulso, por exemplo. Sensores magnéticos instalados nas portas terão a serventia de detectar quando essas portas forem abertas. Essas portas podem ser tanto a principal da casa, possibilitando detectar quando o idoso saiu de casa, como a do armário de remédios, indicando assim quando e se os remédios estão sendo tomados corretamente, ou até mesmo a da geladeira, fornecendo uma análise de hábitos alimentares, por exemplo. Também será utilizado um sensor de presença, com a finalidade de detectar momentos de ociosidade atípicos, e de auxiliar a identificar quando o monitorado de fato saiu de casa. Para a comunicação desses sensores com o *gateway*, serão analisadas

tecnologias, como Wifi, Bluetooth e Zigbee, para se determinar qual é a mais adequada para a situação.

O *gateway* representa a camada de rede, e tem o principal objetivo de transmitir os dados coletados pelos sensores para o serviço em nuvem. Esse *gateway* pode ser implementado por um computador pessoal existente no domicílio monitorado, ou até mesmo por uma placa (*kit*) de desenvolvimento, como uma Raspberry Pi, por exemplo. A comunicação com a nuvem será feita através da própria Internet, utilizando-se do protocolo IP.

A camada de aplicação é onde o monitoramento será efetivamente feito. Ela consistirá de uma aplicação *web*, que poderá ser acessada pela pessoa que deseja fazer o monitoramento e prestar assistência, e fornecerá os dados captados pelos sensores. O modelo de comunicação que a aplicação utilizará para consumir os dados do serviço na nuvem poderá ser tanto o *request/respond* como o *publish/subscribe*, dependendo das necessidades e características da aplicação, que serão futuramente analisadas.

5. Cronograma

O cronograma de atividades previsto para a execução deste trabalho final pode ser visto na Tabela 1.

Tabela 1. Cronograma de atividades.

	AGO	SET	OUT	NOV	DEZ
Definição e avaliação de tecnologias	X				
Implementação e testes		X	X	X	
Avaliação / Validação				X	
Redação da monografia			X	X	
Apresentação					X

Fonte: [Autor]

Referências

Agostini, J. V., Baker, D. I., and Bogardus, S. T. (2001). Prevention of falls in hospitalized and institutionalized older people. The Agency for Healthcare Research and Quality.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *Communications Surveys Tutorials*, 17(4):2347–2376.

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805.

Centers for Disease Control and Prevention (2015). Web-based Injury Statistics Query and Reporting System (WISQARS). National Center for Injury Prevention and Control, Centers for Disease Control and Prevention (producer).

Commission of the European Communities (2007). Ageing well in the Information Society. *i2010 Initiative*.

Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.

de Ruyter, B. and Aarts, E. (2004). Ambient Intelligence: visualizing the future. In *Proceedings of the working conference on Advanced visual interfaces*.

Dunkels, A. and Vasseur, J. P. (2008). IP for smart objects. *IPSO Alliance White Paper*.

Evans, D. (2011). The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. *Cisco White Paper*.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2009). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Hunkeler, U., Truong, H. L., and Stanford-Clark, A. (2008). MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In *3rd International Conference on Communication Systems Software and Middleware and Workshops*.

Insel, K., Morroz, D., Brewer, B., and Figueredo, A. (2006). Executive Function, Working Memory, and Medication Adherence Among Older Adults. *J Gerontol B Psychol Sci Soc Sci*, 61(2):102–107.

IoT-A (2012). The Internet-of-Things Architecture. `http://www.meet-iot.eu/deliverables-IOTA/D1_3.pdf` (Visitado em 29/06/2017).

ITU (2012). New ITU standards define the internet of things and provide the blueprints for its development. `http://www.itu.int/ITU-T/newslog/New+ITU+Standards+Define+The+Internet+Of+Things+And+Provide+The+Blueprints+For+Its+Development.aspx` (Visitado em 14/06/2017).

Matthews, C. E., Chen, K. Y., Freedson, P. S., Buchowski, M. S., Beech, B. M., Pate, R. R., and Troiano, R. P. (2008). Amount of Time Spent in Sedentary Behaviors in the United States. *American Journal of Epidemiology*, 167(7):875–881.

Miorandi, D., Sicari, S., Pellegrini, F. D., and Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516.

Mulligan, G. (2006). The 6LoWPAN architecture. In *4th workshop on Embedded networked sensors*, pages 78–82.

National Center for Health Statistics (1987). National Health Interview Survey's 1984 Supplement on Aging. Public Health Service. Washington. U.S. Government.

OASIS (2014). MQTT Version 3.1.1. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html` (Visitado em 29/06/2017).

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., and Clarke, S. (2016). Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, 3(1):70–95.

Romdhani, I., Abdmeziem, R., and Tandjaoui, D. (2015). *Robots and Sensor Clouds*, chapter Architecting the Internet of Things: State of the Art. Studies in Systems, Decision and Control. Springer, special edition.

SENSEI (2008). SENSEI - Integrating the Physical with the Digital World of the Network of the Future. `http://cordis.europa.eu/pub/fp7/ict/`

docs/future-networks/projects-sensei-ec-summary_en.pdf (Visitado em 29/06/2017).

Sethi, P. and Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*.

United Nations (2015). World Population Prospects: The 2015 Revision. Report, Department of Economic and Social Affairs, Population Division.

Villaverde, B. C., Pesch, D., and Alberola, R. D. P. (2012). Constrained Application Protocol for Low Power Embedded Networks: A Survey. In *4th workshop on Embedded networked sensors*.

Vineela, A. and Rani, L. S. (2015). Internet of Things -Overview. *International Journal of Research in Science Technology*.

Wang, R., Wang, J., and Wang, N. (2015). Analysis of key technologies in the Internet of things. In *3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME 2015)*.

Wortmann, F. and Flüchter, K. (2015). Internet of Things: Technology and Value Added. *Springer Fachmedien Wiesbaden*.

Wu, M., lie Lu, T., Ling, F.-Y., ling Sun, and Du, H.-Y. (2010). Research on the architecture of internet of things. In *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, volume 5, pages 484–487.