FILIPE GENTIL FARIA ARENA
PEDRO NAPOLITANO SANT ANA
RAFAEL COREGLIANO RING

# EIR

## Sistema para Monitoramento Remoto de Pacientes em Assistência Domiciliar

São Paulo
2016

**FILIPE GENTIL FARIA ARENA**
**PEDRO NAPOLITANO SANT ANA**
**RAFAEL COREGLIANO RING**

# EIR

## Sistema para Monitoramento Remoto de Pacientes em Assistência Domiciliar

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

São Paulo
2016

FILIPE GENTIL FARIA ARENA
PEDRO NAPOLITANO SANT ANA
RAFAEL COREGLIANO RING

# EIR

## Sistema para Monitoramento Remoto de Pacientes em Assistência Domiciliar

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

Area de concentração:
Engenharia de Computação

Orientadora:
Profa. Dra. Selma Shin Shimizu Melnikoff

São Paulo
2016

# DEDICATÓRIA

Aos meus pais, Luiz e Cristiana, e minha irmã Catarina, que estiveram sempre ao meu lado, aconselhando-me nas escolhas de minha vida.
À minha namorada Key, que sempre me apoiou e fez companhia, mesmo nos momentos difíceis.
Aos meus colegas do Coop 16, que compartilharam comigo estes maravilhosos últimos anos de faculdade e estarão para sempre em minha memória.
- Filipe Gentil Faria Arena

Aos meus pais, Terezinha e Guaracy, e meus irmãos, Laura, Luana e Fábio, que me apoiaram e auxiliaram durante toda a vida para que eu chegasse até onde estou.
Aos amigos do Coop 16, com quem enfrentei junto todos os desafios desses últimos anos e compartilhei momentos inesquecíveis.
- Pedro Napolitano Sant'Ana

Aos meus pais, Regina e Antônio, por sempre me ajudarem e apoiarem durante a minha vida, incentivarem meu estudo e por todo o amor. Ao meu irmão, Lucas, por toda a parceria e diversão. Aos meus avós, Antal, Esther, Vicente e Aparecida, que sempre fizeram de tudo por mim. E à minha namorada, Isabel, por estar do meu lado e ser minha companheira e melhor amiga para sempre.
- Rafael Coregliano Ring

# AGRADECIMENTOS

À professora Selma Shin Shimizu Melnikoff, nossa orientadora e colega, que com toda sua experiência e conhecimento nos ajudou a produzir este trabalho, direcionando nosso esforço sempre para o caminho correto e oferecendo críticas construtivas para garantir um resultado maduro e consistente.

A todo o Departamento de Engenharia de Computação e Sistemas Digitais (PCS) e à Escola Politécnica da Universidade de São Paulo, que nos proporcionaram um ótimo aprendizado ao longo destes anos de faculdade, aprendizado este que levaremos conosco para o resto de nossas vidas.

Aos nossos colegas do Coop-16, que participaram de grande parte desse processo de aprendizado e que foram companheiros muito valiosos nas mais diversas ocasiões. A presença de vocês propiciou um ambiente muito agradável e divertido para cultivar o conhecimento.

# RESUMO

A assistência domiciliar provê para o paciente a oportunidade de continuar seu tratamento dentro do conforto de sua residência. Apesar de estar longe do hospital, o paciente ainda necessita de um monitoramento contínuo, seja para exames diários ou assistência em geral. Este trabalho visa o desenvolvimento de um sistema que permita ao médico e aos familiares acompanhar o paciente através de um aplicativo móvel, visualizando as últimas medidas relativas às condições do enfermo, como batimentos cardíacos, pressão e temperatura, e possibilitando a eles o recebimento de notificações e alertas no celular caso o paciente apresente um quadro de potencial risco. Pode-se dizer que os resultados obtidos neste trabalho foram satisfatórios, pois permitiram a visualização de dados em tempo real e dados de histórico, envio e recebimento de alertas dinâmicos e gestão de pacientes.

**Palavras-chave:** assistência domiciliar, monitoramento, aplicativo móvel, medidas, notificações.

# ABSTRACT

Home Care provides an opportunity for the patient to continue his treatment inside the comfort of his house. Although far from the hospital, the patient still needs continuous monitoring, whether for daily exams or assistance. This work aims to develop a system that allows the doctor and family members to follow the patient through a mobile application, visualizing the latest data regarding the patient's conditions, such as heart rate, pressure and temperature, and enabling them to receive push notifications and alerts on the cell phone in the eventuality of the patient presenting a potential risk scenario. It can be said that the results obtained in this work were satisfactory, since they allowed the visualization of data in real time and historical data, sending and receiving dynamic alerts and patient management.



**Keywords**: Home Care, monitoring, mobile application, data, push notifications.

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| **API** | *Application Programming Interface* |
| **APP** | Aplicativo |
| **CRUD** | *Create, Read, Update and Delete* |
| **HTTP** | *Hypertext transfer protocol* |
| **IDE** | *Integrated Development Environment* |
| **JSON** | *JavaScript Object Notation* |
| **NOSQL** | *Not Only Structured Query Language* |
| **SDK** | *Software Development Kit* |
| **SO** | Sistema Operacional |
| **USB** | *Universal Serial Bus* |
| **XML** | *eXtensible Markup Language* |

# SUMÁRIO

# 1. Introdução

## 1.1 Motivação

Com o passar do tempo, a humanidade tem cada vez mais investido tempo e dinheiro nas mais diversas soluções para ajudar a elevar a expectativa de vida das pessoas. Tais soluções envolvem a elaboração de novos métodos cirúrgicos, desenvolvimento de novos remédios e diagnósticos eficazes, estudos mais precisos sobre enfermidades, ou ainda, métodos de monitoramento de pacientes em estado crítico. Tais métodos de monitoramento são uma das ferramentas chave para garantir a segurança e acompanhamento contínuo de um paciente. Eles são geralmente implementados em hospitais, através de sensores, que transmitem os dados do paciente para uma central localizada no próprio hospital. Assim, quando ocorre uma variação significativa sobre algum dos dados capturados, é acionado um alerta para o enfermeiro ou médico responsável.

Apesar de ser eficaz, este sistema necessita que o paciente fique no hospital em tempo integral e isso pode ser prejudicial ao próprio paciente, pois o força a ficar constantemente num ambiente hospitalar que facilita o contato com outras doenças, mantêm o paciente em uma situação desconfortável, distante de seus familiares, e agrega um custo elevado ao tratamento. Deve ser levado em conta também o fator psicológico do paciente que, ao estar num local desconhecido, pode ter seu tempo de recuperação prolongado.

Com estes fatores em mente, o trabalho visa melhorar a segurança e a comodidade, facilita o tratamento e barateia os custos agregados ao monitoramento do paciente, permitindo que ele esteja em assistência domiciliar com uma qualidade de monitoramento similar àquela utilizada nos hospitais.

## 1.2 Objetivo

O objetivo deste trabalho é desenvolver o sistema EIR, que recebeu o nome de uma deusa nórdica associada à medicina. O EIR é um sistema de monitoramento remoto de pacientes em assistência domiciliar, que permite o médico

ou o familiar acessar dados em tempo real, através de um aplicativo móvel, ter acesso a um histórico detalhado e receber notificações referentes à saúde do paciente.

Os dados são obtidos constantemente através de sensores e enviados a um servidor na nuvem, onde serão armazenados para consultas de históricos. No aplicativo móvel, o médico pode monitorar e receber notificações de qualquer um de seus pacientes, enquanto os familiares podem fazer o mesmo apenas para o paciente do qual são responsáveis. Como prova de conceito, foi utilizado o sensor de batimentos cardíacos para o desenvolvimento do sistema.

## 1.3 Justificativa

A medicina atual, que dispõe de uma série de instrumentos eletrônicos analógicos e digitais, é capaz de gerar uma quantidade enorme de dados referentes às mais diversas medidas obtidas a partir de um indivíduo. No entanto, tais dados nem sempre são aproveitados da forma mais útil possível ou dispostos de forma clara e intuitiva ao médico e ao paciente.

Dessa forma, o trabalho em questão, além de tornar o acompanhamento do paciente mais agradável, simples e barato, é capaz de utilizar os dados obtidos de forma mais eficaz, agregando valor às informações captadas pelos sensores do sistema, possibilitando também, o desenvolvimento das mais diversas funcionalidades como mecanismo de alertas ou geração e comparação de históricos de dados, por exemplo.

Agregar valor aos dados obtidos significa não só um tratamento diferenciado sobre os dados, mas também a apresentação delas. Para isto, o sistema dispõe de um aplicativo móvel responsivo que, com uma simples conexão com a Internet, é capaz de apresentar tanto os dados do paciente, como a integração com os recursos adicionais do sistema, de forma clara, objetiva, agradável e intuitiva.

### 1.3.1 Decisão pelo Acompanhamento Através do Celular

Os celulares são ferramentas poderosas. Atualmente eles estão presentes praticamente no cotidiano da grande maioria das pessoas, porque são fáceis de usar, intuitivos, apresentam uma gama impressionante de funcionalidades, mesmo nos modelos mais simples, e tudo isso a um custo acessível. Esses equipamentos são, portanto, ideais para o desenvolvimento de aplicativos móveis de monitoramento, uma vez que estão sempre ao alcance do usuário, além de possuírem uma conexão constante com a Internet que, no caso deste trabalho, foi essencial.

#### 1.3.1.1 Facilidade de Uso

O celular está se tornando cada vez mais intuitivo. Desde o lançamento do primeiro iPhone, as empresas do ramo de aparelhos móveis estão se surpreendendo com a quantidade de pessoas no mundo que passou a aderir fervorosamente ao uso do pequeno dispositivo. Sejam pessoas idosas ou crianças, a grande maioria das pessoas estão sendo capazes de usufruir dos mais diversos recursos. Uma das principais razões por trás disso é a análise de usabilidade que a Apple realizou no desenvolvimento do Iphone. Descobriu-se, com isso, fatos interessantes que quanto mais simples a interface gráfica e menos poluição visual na pequena tela do aparelho, mais intuitiva seria a navegação do usuário pelo aplicativo móvel [22].

#### 1.3.1.2 Mobilidade

Mobilidade é uma palavra-chave no contexto atual. As pessoas querem estar em todos os lugares, a qualquer hora do dia, e querem poder levar suas notícias no bolso, sua agenda de compromissos e sua câmera para onde forem. Ficar dependente de um *desktop* ou de um telefone fixo é impensável nos dias de hoje.

Portanto, a característica principal do celular mais importante para este trabalho é a capacidade de reunir diversas funcionalidades em um só lugar, em qualquer lugar e hora do dia.

#### 1.3.1.3 Bibliotecas Gráficas

Com a comunidade de desenvolvimento de aplicativos móveis crescendo ano após ano, é de se esperar que haja uma enorme quantidade de bibliotecas gráficas, capazes de facilitar a criação dos mais diversos tipos de gráficos e em tempo real. De fato, isso é verdade. Existe um imenso suporte de bibliotecas para o Android, pronto para ser usado [23].

#### 1.3.1.4 Conectividade e Tempo Real

Boa parte das funcionalidades do celular depende de uma comunicação constante com as operadoras de telefonia móvel ou uma conexão *Wi-Fi*. Felizmente, nos dias atuais, isso não é um problema porque a cobertura de sinal das operadoras está cada vez melhor, assim como os de planos de Internet providos por elas, o que permite um tráfego de dados em alta velocidade e, consequentemente, o uso de aplicativos móveis cada vez mais sofisticados, com recursos em tempo real.

### 1.3.2 Aplicações Baseadas em Monitoramento

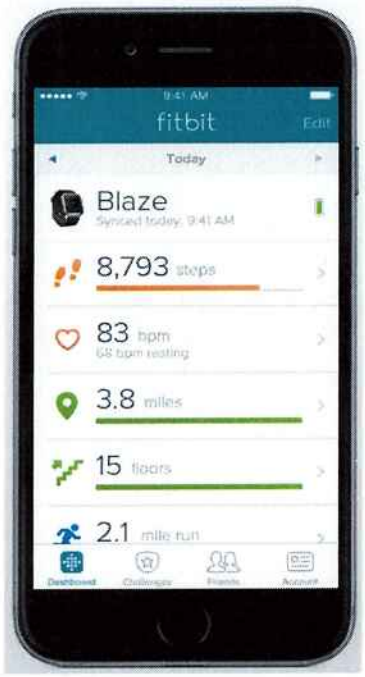
Com relação às aplicações, que oferecem recursos de monitoramento de atividades de pessoas, existem alguns exemplos já implementados. Tais atividades variam desde distância percorrida a batimentos cardíacos do usuário, ou ainda, qualidade do sono. A seguir, seguem alguns exemplos desses tipos de aplicação.

### 1.3.2.1 FitBit

Fitbit é um aplicativo móvel e *web*, desenvolvido pela FitBit, uma empresa responsável por criar soluções de monitoramento de atividades do usuário através de sensores localizados em *wearables* como pulseiras e relógios, por exemplo [14].

As atividades monitoradas variam desde quantos passos o usuário deu no dia, distância percorrida em km, quantos batimentos cardíacos e qualidade de sono. Como pode ser visto na Figura 1, todas essas informações obtidas são exibidas de forma simples e agradável ao usuário, que por sua vez, pode criar metas a serem alcançadas por si próprio ou pelos amigos cadastrados na rede da FitBit.

Figura 1 - Aplicativo Fitbit




Fonte: https://www.fitbit.com/app (2016)

### 1.3.2.2 Wahoo Tickr X Heart Rate Monitor

Tickr X Heart Rate Monitor é um aparelho desenvolvido pela Wahoo Fitness, cujo objetivo é monitorar o exercício realizado pelo usuário, através da análise de movimentos e intensidade, avaliando a eficiência do exercício. A Figura 2 apresenta

a imagem do aparelho, capaz de coletar dados sobre batimentos cardíacos, queima de calorias, repetições de exercício e dados de corrida. O dispositivo também apresenta uma memória interna que o torna independente de um celular. No entanto, sua conectividade via *Bluetooth* permite que seja conectado com a maioria dos dispositivos móveis, para que seus dados possam sem compartilhados [9].

*Figura 2 - Wahoo Tickr X Heart Rate Monitor*


Fonte: http://www.wahoofitness.com/devices/wahoo-tickr-x-heart-rate-strap (2016)

### 1.3.3 Aplicações Baseadas em Alertas

Com relação às aplicações que oferecem recursos de alertas, existem algumas opções no mercado. Tais aplicações envolvem funcionalidades como o acionamento de um botão pelo usuário para enviar um alerta silencioso para os responsáveis ou, ainda, a própria aplicação detectar uma situação de emergência e prosseguir com envio de alertas. A seguir, é apresentado um exemplo deste tipo de aplicação.

ManDown App é um aplicativo móvel desenvolvido por bombeiros, com intuito de salvar vidas através do envio de alertas para contatos de emergência.

O aplicativo funciona através do monitoramento do movimento do celular pelo uso do giroscópio. Caso o celular do usuário fique 30 segundos sem se mover, é disparado um pequeno alerta sonoro. Em seguida, caso o celular fique mais 30 segundos sem movimentação, é disparado o alarme sonoro principal. Além disso, o aplicativo envia, ao usuário, uma mensagem de texto e e-mail, e realiza uma ligação para os contatos de emergência, usando o GPS do celular para enviar seu local. O aplicativo também conta com recursos de alarmes silenciosos, de forma que o

usuário é capaz de, através de um botão, alertar seus contatos de uma situação de risco [7]. A Figura 3 apresenta algumas telas do aplicativo.

*Figura 3 - ManDown App*


Fonte: http://www.mandownapp.com/ (2016)

### 1.3.4 Aplicações Baseadas no Setor de Saúde

Com relação às aplicações desenvolvidas para o setor de saúde, existem poucas que são realmente utilizadas atualmente no mercado. Provavelmente isso ocorre pela dificuldade de conciliar tecnologia evolutiva, a complexidade e a burocracia que o setor de saúde apresenta. A seguir, é apresentado o dr.consulta, que é um dos poucos modelos de sucesso, que vem ganhando espaço no setor [3].

Através do dr.consulta, o usuário é capaz de agendar consultas e exames em centros médicos que fazem parte da rede do aplicativo, ver resultados de exames e gerenciar seus agendamentos. A Figura 4 apresenta um exemplo de tela do aplicativo.

*Figura 4 - Aplicativo dr.consulta*


Fonte: https://www.drconsulta.com (2016)

### 1.3.5 Sistemas Baseados no Monitoramento Remoto de Pacientes

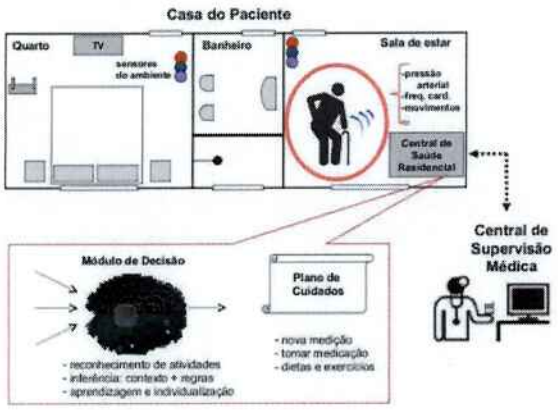
Nesta seção, há uma breve descrição de três sistemas similares ao EIR, baseados no monitoramento de pacientes em assistência domiciliar. Esses sistemas serviram como referência para o desenvolvimento do EIR, possibilitando avaliar como um mesmo problema foi abordado por outros profissionais e quais funcionalidades poderiam ser melhoradas ou adicionadas para prover uma melhor solução.

#### 1.3.5.1 Monitoramento Remoto de Pacientes em Ambiente Domiciliar (UFF, UFG, UERJ)

Esse trabalho foi desenvolvido por integrantes da Universidade Federal Fluminense (UFF), Universidade Federal de Goiás (UFG) e Universidade do Estado do Rio de Janeiro (UERJ), e trata da instalação de uma central de monitoramento na casa do paciente, que analisa os dados do ambiente e do paciente e se comunica

com uma central de supervisão médica para envio de alertas e dados coletados [1]. A Figura 5 apresenta o escopo do sistema.

*Figura 5 - Monitoramento Remoto de Pacientes em Ambiente Domiciliar*




Fonte: http://www.lbd.dcc.ufmg.br/colecoes/sbrc/2010/0071.pdf (2016)

Pode-se observar na figura que os sensores de ambiente foram instalados no quarto e na sala de estar do paciente e, juntamente com os sensores sem fio localizados no paciente, que se comunicam com a Central de Saúde Residencial para transmitirem as devidas informações sobre o paciente.

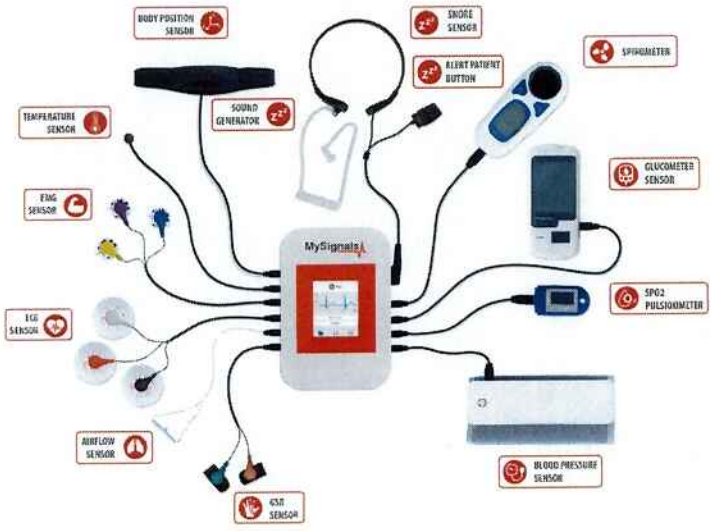
#### 1.3.5.2 SCIADS - Sistema Computacional Inteligente de Assistência Domiciliar à Saúde

É o protótipo de um sistema desenvolvido a partir do sistema de Monitoramento Remoto de Pacientes em Ambiente Domiciliar (UFF, UFG, UERJ), descrito na seção 1.3.5.1. O site do sistema disponibiliza o *download* de uma versão de testes e avaliação do protótipo para as centrais de saúde residencial e médica, permitindo ainda que os desenvolvedores simulem os sensores necessários por meio de um simulador fornecido juntamente com o *software* do protótipo [21].

### 1.3.5.3 MySignals

Durante o desenvolvimento do trabalho de formatura, a empresa da qual o grupo adquiriu o kit de sensores, a Libelium, desenvolveu o MySignals, uma segunda versão do kit utilizado, que contempla alguns aspectos planejados pelo grupo do trabalho de formatura, como a integração com um servidor na nuvem e a exibição de dados obtidos pelos sensores em forma de gráficos [24]. A Figura 6 apresenta a configuração do MySignals.

*Figura 6 - MySignals*




Fonte: http://www.my-signals.com/ (2016)

Apesar disso, o EIR pode ser justificado como trabalho de formatura, pois:

- O kit adquirido possui apenas os sensores de medições, sem auxílio visual algum. Foi necessária a criação de programas para obter os dados e transmiti-los.

- O trabalho de formatura em questão implementa o acompanhamento, via aplicativo móvel, a geração de alertas emergenciais e o uso específico para *Home Care*.

## 1.4 Metodologia

Inicialmente os integrantes estudaram a documentação dos sensores e o *hardware,* necessários para o desenvolvimento do trabalho, como por exemplo, o Arduíno e linguagens de programação essenciais para a implementação do sistema.

A ideia inicial e que foi mantida até o final do trabalho, foi separar o desenvolvimento do sistema em três grandes blocos:

1. **HARDWARE**: Configurar o hardware, desenvolver a coleta de dados através dos sensores e enviar para um servidor, utilizando um programa na linguagem C++ que realize o *upload* dos dados.

2. **SERVIDOR**: Configurar um servidor para receber os dados, armazená-los e gerar uma API que o aplicativo móvel seja capaz de acionar para obter os dados.

3. **APLICATIVO MÓVEL**: Fazer o aplicativo móvel que, através da interação com o servidor, apresente os dados de forma fácil e intuitiva, permitindo algumas configurações personalizáveis para a organização de dados do servidor.

Em seguida, foi necessário focar na realização de testes de fluxo do sistema, tais procedimentos estão detalhados na seção 3.4. Com os resultados dos testes em mãos, foram corrigidos os possíveis erros, para depois dar prosseguimento ao desenvolvimento das funcionalidades do sistema.

Neste ponto, a cada série de funcionalidades novas desenvolvidas, foi realizada, em seguida, uma fase de testes sobre o bloco de funcionalidades, para garantir o seu funcionamento.

O desenvolvimento de EIR seguiu os seguintes passos:

A. Entrar em contato com especialista da área que possa sugerir direcionamentos, funcionalidades, melhorias e etc.

B. Estudar funcionamento do kit de sensores adquiridos.
C. Desenvolver o programa para *upload* de dados obtidos pelos sensores para o servidor.
D. Implementar o servidor para receber e armazenar dados, fazer o envio de notificações de emergência e criar API para que o aplicativo móvel possa receber e enviar dados.
E. Desenvolver o aplicativo móvel.
F. Realizar testes básicos de funcionamento
G. Com base nos resultados de testes, corrigir possíveis erros e continuar desenvolvimento.
H. Realizar o teste fim-a-fim do sistema.

## 1.5 Estrutura do Documento

A seguir é descrita a estrutura do restante deste documento:

**Capítulo 2 – Conceitos e tecnologias**
Neste capítulo estão apresentados os principais conceitos necessários e as tecnologias utilizadas ao longo deste trabalho, através de uma breve descrição dos tópicos.

**Capítulo 3 – Desenvolvimento do sistema**
Neste capítulo está descrito o processo de desenvolvimento do sistema, através de suas fases e respectivos produtos gerados, além de decisões de trabalho, direcionamentos tomados e tecnologias implementadas. Também estão apresentadas as avaliações e os testes realizados para verificar o funcionamento do sistema final.

**Capítulo 4 – Considerações finais**
Neste capítulo estão descritas as conclusões obtidas na elaboração do trabalho, as contribuições e as propostas para trabalhos futuros.

**Apêndice A – Casos de uso**

Neste apêndice encontram-se os casos de uso que fazem parte do escopo do trabalho e os diagramas de sequência associados a eles.

**Apêndice B – Protótipos de tela**

Neste apêndice encontram-se os protótipos de tela previamente criados para direcionar o desenvolvimento do layout do aplicativo móvel.

# 2. Conceitos e Tecnologias

## 2.1 Considerações Iniciais do Capítulo

Neste capítulo são expostos os conceitos e as tecnologias utilizados e referenciados no desenvolvimento do sistema EIR. Cada seção a seguir exibe um breve resumo sobre o assunto, relatando também sua importância e sua utilidade para o sistema final.

Os conceitos descritos foram incluídos nesta seção com o intuito de esclarecer seu significado e possíveis dúvidas que o leitor venha a ter, uma vez que são utilizados em outros capítulos deste documento.

As tecnologias utilizadas no sistema foram escolhidas pelo grupo, com intuito de favorecer um ambiente de tecnologia de ponta, com alto desempenho, que fosse escalável, mantendo ainda a facilidade e simplicidade de uso para o usuário final.

## 2.2 Descrição dos Conceitos e Tecnologias

As seções a seguir apresentam os conceitos e as tecnologias que foram utilizados ao longo do desenvolvimento do sistema.

### 2.2.1 NoSQL

NoSQL (*Not Only Structured Query Language*) é o termo utilizado para descrever bancos de dados não-relacionais de alto desempenho. Esse tipo de banco de dados se difere dos relacionais, principalmente por não serem organizados em tabelas ou *schemas* [5]. Normalmente uma chave é utilizada para recuperar valores ou documentos como JSON ou XML.

Os bancos de dados NoSQL são conhecidos por serem fáceis de serem desenvolvidos e acessados, terem um desempenho altamente escalável, alta disponibilidade e serem resilientes. A escalabilidade em NoSQL é horizontal, ou

seja, eles são capazes de adicionar mais nós ao sistema, tais como um novo computador com uma aplicação para clusterizar o software.

Neste trabalho, são utilizados dois bancos de dados baseados em NoSQL: RethinkDB e MongoDB.

### 2.2.2 RethinkDb

RethinkDB é um banco de dados NoSQL orientado a documentos e *open source*. Ele foi criado com o intuito de ser escalável e é utilizado em aplicações de tempo real [4].

Com o uso do RethinkDB, as aplicações solicitam, ao banco de dados, o envio de apenas dados atualizados, ao invés de receber os dados e buscar as alterações. Sua arquitetura de envio de dados reduz tempo e esforço necessários para construir aplicações escaláveis, com características de tempo real.

No caso de conectividade de dispositivos que compartilham frequentemente milhares de informações entre si, o RethinkDB é a escolha adequada, pois é capaz lidar com o volume de informações, garantindo a flexibilidade e escalabilidade desejada. Além disso, o RethinkDB oferece uma linguagem de *query* flexível, operações intuitivas e APIs de monitoramento fáceis de configurar e rápidas de se aprender.

Neste trabalho, um dos bancos de dados escolhidos foi o RethinkDB. Ele é responsável pela troca de informações volumosa que existe entre os dispositivos do sistema e que precisam refletir alterações instantâneas no aplicativo móvel.

### 2.2.3 JSON

JSON ou *JavaScript Object Notation* é um modelo de armazenamento e transmissão de dados de fácil leitura e escrita [6]. Sua sintaxe é clara. Para cada valor representado, utiliza-se um nome que descreve seu significado. Esta forma de sintaxe deriva do próprio JavaScript, quando nesta linguagem é representado um objeto.

Neste trabalho, o formato JSON é utilizado como a estrutura de armazenamento de dados predominante no banco de dados.

### 2.2.4 MongoDB

MongoDB é um banco de dados *open source* baseado em documentos, criado com o intuito de ser fácil de desenvolver e escalar. Ele utiliza a estrutura de armazenamento de dados em JSON sendo, portanto, fácil de manipular, indexar e manipular dados internos armazenados. As principais vantagens de se utilizar o MongoDB são seu alto desempenho, escalabilidade e disponibilidade. Como a maioria dos outros bancos de dados NoSQL, no entanto, ele carece de consistência. No entanto, pela facilidade de uso e de grande parte dos outros atributos que apresenta, o trabalho em questão utiliza o MongoDB para armazenar os dados do sistema que não demandam uso em tempo real [15].

### 2.2.5 NodeJS

NodeJS é uma plataforma JavaScript criada utilizando o motor V8 do Chrome [25]. Ele utiliza um modelo orientado a eventos leve e eficiente, apropriado para construir aplicações em tempo real, com grande volume de dados distribuído entre diversos dispositivos.

Seu maior benefício é ser uma plataforma que facilita a construção de programas de rede escaláveis. Isso acontece porque o NodeJS resolve uma série de problemas de escalabilidade de aplicações de rede: custos, velocidade de tráfego, velocidade de processamento e uso de memória.

### 2.2.6 Express

Express é um *framework* minimalista que apresenta uma série de recursos para aplicações *web* e *mobile* [10]. Ele facilita o desenvolvimento de aplicações

baseadas em NodeJS. Com ele é fácil e rápido configurar um *middleware* que atende a chamadas HTTP, elaborar uma API e iniciar um servidor.

### 2.2.7 Android

Android é um Sistema Operacional para dispositivos móveis, baseado no Linux. Seu projeto foi desenvolvido pela Google, com objetivo de oferecer diversas funcionalidades e ser um sistema aberto, ou seja, os desenvolvedores podem utilizar este Sistema Operacional para criar novas melhorias [28].

Atualmente é um dos Sistemas Operacionais mais utilizados no mercado. De acordo com o site Kantar, especializado em fornecer dados de *market-share* sobre *smartphones* no mercado mundial, no Brasil, o ano de 2015 fechou com o Android liderando as vendas de *smartphones* com 91,8%, contra 2,8% do IOS da Apple [11]. Esses dados revelam que, para o atual poder aquisitivo do cidadão brasileiro, o custo benefício dos aparelhos que utilizam o Android, é insuperável. Além disso, o *hardware* presente nos *smartphones* de baixo custo mais vendidos no país, é suficiente para executar a maior parte dos atuais aplicativos móveis mais utilizados.

Com esses dados de volume de uso e acessibilidade em mente, o grupo optou por escolher o Sistema Operacional Android como alvo de desenvolvimento da aplicação *mobile*, além de ser uma tecnologia mais familiar aos membros do grupo.

### 2.2.8 Android Studio

Android Studio é uma IDE (Integrated Development Environment), ou ambiente de desenvolvimento integrado, desenvolvida pela Google I/O na plataforma Android com base no InteliJ. O Android Studio apresenta um ambiente de desenvolvimento, depuração e testes para Android. Sua interface é agradável e intuitiva e a IDE apresenta uma vasta customização de atalhos disponíveis para o desenvolvedor [29].

O Android Studio permite a criação de uma aplicação Android completa, dando suporte tanto ao código base quanto ao projeto das telas e dos componentes, o qual pode ser feito através de uma interface mais amigável para usuários novos, quanto diretamente em XML, permitindo total customização.

A IDE também possui um emulador de dispositivos, o que oferece ao desenvolvedor a possibilidade de testar sua aplicação em dispositivos com diferentes configurações, como tamanho de tela e versão do sistema operacional Android, sem precisar adquirir aparelhos físicos para tal.

### 2.2.9 SDK

SDK (*Software Development Kit*) é um kit de desenvolvimento de software utilizado para auxiliar no desenvolvimento e criação de aplicativos móveis para plataformas específicas. Esses kits são geralmente criados por empresas de forma que ajuda os desenvolvedores externos a participar do crescimento da plataforma em questão [29].

No caso do Android, por exemplo, existem diversos SDKs. Entre os mais recentes, pode-se mencionar o Nougat, versão 7.1, Marshmellow, versão 6.0 e Lollipop, versão 5.1. Cada um desses SDKs dispõe de novas ferramentas para desenvolvimento das versões mais atuais do Android, quando comparadas com as versões anteriores.

### 2.2.10 XML

XML (*eXtensible Markup Language*) é uma linguagem de marcação de fácil leitura e escrita, a qual pode ser utilizada para descrição de objetos, respostas de APIs, até descrição de interfaces gráficas. O objetivo de seu design é ser simples e utilizável através da Internet [30].

Neste trabalho, é utilizado XML para descrição das interfaces gráficas e componentes do aplicativo móvel Android.

### 2.2.11 Git

Git é um sistema de controle de versão através do qual diversas pessoas podem colaborar simultaneamente no desenvolvimento de sistema, criando versões do mesmo, editando-o sem interferir no que os outros estão fazendo ou sobrescrever atualizações criadas por outros [31].

Um dos conceitos mais importantes do Git é o uso de *branches* ou galhos que são ramificações de um mesmo projeto, em momentos e estados distintos entre si. O uso de *branch* serve para garantir a integridade do trabalho sendo desenvolvido, de forma que o andamento do projeto não fique restrito à versão que está sendo utilizada pelos desenvolvedores.

### 2.2.12 GitHub

GitHub, por sua vez, é um serviço de hospedagem ou aplicação *web* baseado no Git, que oferece armazenamento remoto e gestão de projetos. Sua interface é simples e amigável, que facilita o trabalho dos desenvolvedores [32].

### 2.2.13 Java

Java é uma linguagem de programação muito utilizada, em diversos ramos da computação. Ela foi criada pela Sun Microsystems em 1995 [12]. É também uma plataforma que serve de base para diversas aplicações, de forma que, caso o usuário não tenha Java instalado no computador, tais aplicações não poderão ser executadas.

Devido à sua versatilidade, a linguagem Java foi adotada como base para o sistema operacional Android e suas aplicações e, por isso, o trabalho em questão utiliza Java no desenvolvimento do aplicativo móvel.

### 2.2.14 Internet das Coisas

Internet das Coisas, é um conceito que se refere a uma revolução tecnológica que tem como objetivo conectar dispositivos e objetos reais ao mundo virtual, ou seja, à rede Internet. O intuito da Internet das Coisas é fazer com que tais objetos se comuniquem entre si, com a rede Internet, servidores, sensores, atuadores e afins, com objetivo de otimizar processos e melhorar o conforto dos seres humanos [27].

Considerando o fato de o Arduino ser uma plataforma aberta para experimentação no mundo de Internet das Coisas, pode-se dizer que este conceito está inteiramente ligado ao trabalho, uma vez que no sistema existem dispositivos reais, no caso sensores biométricos, que se comunicam com o restante do sistema, através de uma plataforma aberta com conexão à Internet.

### 2.2.15 Arduino

Arduino é uma plataforma eletrônica *open source* de fácil utilização e configuração, muito relevante no âmbito de Internet das Coisas. Ele apresenta conectores disponíveis, que podem ser utilizados para ligar a plataforma aos mais diversos dispositivos, além de apresentar um microcontrolador embutido responsável por controlar a plataforma [13].

Com o passar dos anos, o Arduino foi centro de diversos projetos, desde escolares até projetos científicos avançados. Isso ocorre devido ao fato de o Arduino ter um custo acessível, ser multi-plataforma (sendo possível rodá-lo em Linux, Windows e Mac) e ter um ambiente de programação simples e amigável.

### 2.2.16 e-Health Sensor V2.0

O kit de sensores da e-Health Sensor Plataform V2.0 foi desenvolvido pela Cooking Hacks, uma divisão de *hardware* da Libelium [2]. Ele foi criado com o intuito de ajudar pesquisadores e desenvolvedores a coletar uma série de dados

biométricos: posição do corpo, temperatura, eletromiografia, eletrocardiografia, fluxo de ar, resposta galvânica da pele, pressão sanguínea, pulso e glicosímetro.

O kit permite o desenvolvimento de aplicações médicas que não dispõe de investimento e certificações médicas para os aparelhos utilizados.

### 2.2.17 Backend

*Backend* é um termo que se refere à parte de um sistema com a qual o usuário final não interage ou não enxerga. Geralmente, encontra-se no lado do *Backend* a parte de servidores e bancos de dados, que tendem a ser transparentes para o usuário e que desempenham um papel essencial na estruturação do sistema.

### 2.2.18 Frontend

*Frontend*, por sua vez, é o termo complementar à *Backend*, definido em 2.2.17, o qual se refere à parte de um sistema através do qual o usuário final interage e, portanto, enxerga. Normalmente, encontra-se no lado do *Frontend* a parte do código que ajuda a construir o *layout* do sistema e a forma como o usuário interage com ele.

### 2.2.19 Aplicativo Móvel

O Aplicativo Móvel, mais conhecido como *app*, é o nome dado a um software desenvolvido para dispositivos móveis, como celulares, *tablets*, etc. Os aplicativos móveis normalmente são disponibilizados em lojas *on-line* como Apple Store, Google Play e Windows Phone Store, de forma que o usuário escolhe o aplicativo móvel desejado e faz o *download* dele em seu dispositivo.

O objetivo dos aplicativos móveis normalmente é facilitar o dia-a-dia dos usuários finais, fornecendo uma série de recursos e contemplando uma enorme gama de possibilidades.

Neste trabalho, a parte de interação dos usuários finais com o sistema é feita inteiramente através de um aplicativo móvel responsivo.

### 2.2.20 Push Notification

*Push notification* é definido como o envio de informação de um servidor para uma aplicação, sem uma solicitação prévia desse conteúdo. No ambiente de aplicações móveis, *push notifications* são utilizados para informar o usuário do acontecimento de determinados eventos ou enviar novas informações ao mesmo [33].

Cabe ao desenvolvedor de cada aplicativo móvel, configurar o envio automático de *push notifications*, de acordo com acontecimentos predeterminados ou mesmo realizar envios por demanda. É também papel do desenvolvedor configurar como cada aplicativo móvel se comporta ao receber um *push notification*. Seu comportamento padrão é exibir uma notificação no dispositivo do usuário, com a qual ele pode interagir para consumir seu conteúdo.

### 2.2.21 Microservices

*Microservices* é um padrão de desenvolvimento de software que foca em dividir uma grande aplicação em pequenas aplicações com funções mais especializadas (serviços), de modo que seja possível fazer o escalonamento de forma independente para cada um dos serviços [34].

O propósito principal é isolar componentes separados da aplicação, permitindo que, além de escalonados, sejam também modificados sem que haja falhas na operação da aplicação como um todo.

Neste trabalho, devido ao requisito de desempenho da aplicação, bem como sua alta disponibilidade, foi escolhida a arquitetura de *microservices* para o servidor.

### 2.2.22 Home Care

*Home Care*, se traduzido diretamente do inglês, apresenta a palavra "casa" (*home*) juntamente com a palavra "cuidados" (*care*). É um termo usado para designar a prestação de serviços na área da saúde, normalmente na residência do indivíduo. Essas atividades são centradas no atendimento de pacientes e os familiares dele, em um ambiente externo ao hospital.

O propósito do *Home Care* é prover um complemento a serviços já prestados em um hospital, de forma que o paciente possa, no conforto de sua casa, ser atendido até que seu estado natural de saúde seja restabelecido ou que ele seja monitorado, de forma a reduzir os eventuais riscos à vida.

Neste trabalho, o cenário central, que permeia o uso do EIR, é aquele no qual se encontra a prestação de serviços no formato do *Home Care*.

### 2.2.23 Responsividade

Responsividade ou *design* responsivo é uma característica de aplicações ou software, a qual permite a sua adaptação aos diferentes tamanhos de tela de dispositivos, de forma a manter o conteúdo bem estruturado e definido, sem gerar quebras de conteúdo ou imagens [35].

Esta característica é muito importante para produtos que são usados nos diversos dispositivos, para garantir que a entrega de conteúdo seja feita com a mesma qualidade.

Neste trabalho, o aplicativo móvel foi construído com características responsivas, de forma que, independentemente do aparelho celular Android que o usuário possuir, ele possa visualizar o mesmo conteúdo de qualquer outro usuário do sistema.

### 2.2.24 C++

C++ é uma linguagem de programação orientada a objetos criada por Bjarne Stroustrup, sendo ela uma extensão da conhecida linguagem C [17].

Neste trabalho, foi utilizada esta linguagem para escrever o programa do Arduino que realiza a obtenção de dados provenientes dos sensores para obter as medidas biométricas.

### 2.2.25 CRUD

CRUD é a abreviação de *Create, Read, Update* e *Delete*. Ela se refere às operações básicas realizadas sobre os dados em uma estrutura de banco de dados.

### 2.2.26 Mongoose

Mongoose é uma biblioteca do NodeJS que facilita a criação de modelos e estrutura de dados, permitindo que o desenvolvedor tenha acesso aos comandos de CRUD dentro do MongoDB de maneira rápida e simples [18].

Neste trabalho, utilizou-se o Mongoose justamente para ter acesso às operações de CRUD dentro do MongoDB.

### 2.2.27 Client-Server

*Client-Server* é um estilo de arquitetura de software, em que existem dois componentes principais: o *Server* é o fornecedor de serviços ou dados e *Client,* o consumidor ou solicitante dos serviços ou dados [36].

### 2.2.28 OAuth

OAuth é um protocolo *open source*, capaz de estruturar uma autenticação segura dentro de aplicações [19]. É um dos padrões de autenticação mais utilizados, permitindo que os desenvolvedores configurem permissões de usuário de maneira simples.

### 2.2.29 Middleware

*Middleware* é o termo usado para se referir à parte de um sistema que atua como intermediador de serviços e requisições entre outras partes, normalmente entre o cliente e o servidor [37].

### 2.2.30 Python

Python é uma linguagem de programação orientada a objetos e interpretada de alto nível. Foi criada por Guido van Rossum em meados de 1980, sendo atualmente a terceira linguagem mais popular [20].

Neste trabalho, utilizou-se Python, para criar o código que recebe os dados da placa Arduino via USB, formata-os e envia-os para o servidor.

### 2.2.31 Postman

Postman é uma aplicação que apresenta uma série de recursos para desenvolvimento de uma API. Com ele é possível fazer requisições HTTP para uma API escolhida de forma a testar, documentar, avaliar e monitorar requisições [38].

## 2.3 Considerações Finais do Capítulo

No capítulo 2, foram apresentados os conceitos, necessários e as tecnologias selecionadas para a implementação do EIR. Sendo assim, a base de recursos, utilizada no desenvolvimento contempla: conceitos gerais envolvidos, ambientes de desenvolvimento, ferramentas de teste e gestão de código, linguagens de programação, bancos de dados, entre outros.

É importante ressaltar que inicialmente o grupo iniciou o desenvolvimento do *Backend* apenas com o MongoDB mas, no decorrer da implementação, observou-se que ele carecia de recursos para transferência de dados em tempo real e, portanto, foi necessário buscar um banco de dados complementar. Foi nesse momento que se optou pelo uso do RethinkDB que, como foi dito na seção 2.2.2, é uma das melhores soluções do mercado para banco de dados para sistemas de tempo real, na opinião dos membros do grupo.

# 3. Desenvolvimento do Sistema

Neste capítulo estão dispostas as seções que abordam o desenvolvimento do sistema, incluindo requisitos, modelagem, projeto, implementação e testes.

## 3.1 Especificação de Requisitos

Nesta seção, são apresentados os requisitos funcionais e não funcionais do sistema.

### 3.1.1 Requisitos Funcionais

A seguir são descritos os requisitos funcionais do sistema desenvolvido.

**Monitoramento**: O sistema deve permitir que os familiares e o médico acompanhem o paciente, através de um monitoramento contínuo.

**Alertas e notificações**: O sistema deve enviar alertas e/ou notificações para familiares e médico, na eventualidade de o paciente encontrar-se em um estado de emergência.

**Histórico**: O sistema deve armazenar um histórico consistente das medições obtidas no banco de dados.

**Gestão de pacientes**: O sistema deve permitir que o médico gerencie os dados dos pacientes que está atendendo.

A tabela 1 mostra um mapeamento entre os requisitos funcionais levantados e os casos de uso do sistema (S) e concentrador de dados (C), que se encontram na seção 3.2.2.

Tabela 1 – Mapeamento de requisitos funcionais com casos de Uso

| Requisito Funcional | Casos de Uso |
|---|---|
| Monitoramento | S5, S21, C1, C2 |
| Alertas e notificações | S11, S16, S22, S23 |
| Histórico | S5, S18, S19, S20, S24 |
| Gestão de Pacientes | S1, S2, S3, S4, S6, S7, S8, S9, S10, S12, S13, S14, S15, S17, S18, S19, S20, S24, S25 |

### 3.1.2 Requisitos Não-Funcionais

Foram levantados também requisitos não-funcionais para o sistema. Devido ao escopo do projeto, esses requisitos foram utilizados para orientar o desenvolvimento, não sendo avaliados formalmente ao longo do desenvolvimento.

**Disponibilidade**: Tratando-se de um sistema voltado para saúde, o sistema deve estar idealmente disponível cerca de 99,9% do tempo. No restante do tempo, considera-se que o sistema estará indisponível em casos de eventuais falhas elétricas ou no servidor de hospedagem utilizado. Como o projeto não chegou a ser implantado na nuvem, sendo apenas hospedado localmente durante intervalos intermitentes, não foi possível avaliar sua disponibilidade.

**Responsividade**: O aplicativo móvel do sistema deve ser capaz de se adaptar a diferentes tamanhos de tela de dispositivo, de forma que o usuário final seja capaz de observar o conteúdo da mesma forma em qualquer dispositivo com sistema operacional Android. Esse requisito pôde ser validado através de diversos testes em dispositivos reais e simuladores que permitiram avaliar o comportamento do aplicativo em celulares com diferentes tamanhos de tela.

**Desempenho**: O sistema deve ter o desempenho suficiente para realizar a transferência de um grande volume de dados em tempo real e detectar situações de risco rapidamente. Além disso, o sistema deve apresentar um atraso mínimo aceitável entre os dados obtidos pelos sensores e aqueles apresentados para o usuário final. Para avaliar o tempo de resposta do sistema, seu funcionamento foi testado em diferentes condições, variando a conexão do aplicativo à rede (Wi-Fi ou 3G) e o intervalo de tempo entre as coletas de dados.

**Escalabilidade**: O sistema deve possuir uma alta escalabilidade, através de recursos configurados para suportar um aumento repentino na demanda de dados. Mesmo sem serem realizados testes concretos de escalabilidade, a arquitetura do sistema em microserviços prevê a possibilidade de escalar separadamente diferentes funcionalidades do mesmo, priorizando as mais críticas no caso de eventuais sobrecargas.

**Segurança**: O sistema deve apresentar mecanismos de autenticação e níveis de acesso, de forma que o usuário final deve estar sempre autenticado no sistema para acessar os recursos a que terá permissão.

## 3.2 Modelagem do Sistema

Nesta seção, estão apresentados os dois diagramas de caso de uso do sistema, além do diagrama de classes. Maiores detalhes a respeito da modelagem do sistema encontram-se no Apêndice A.

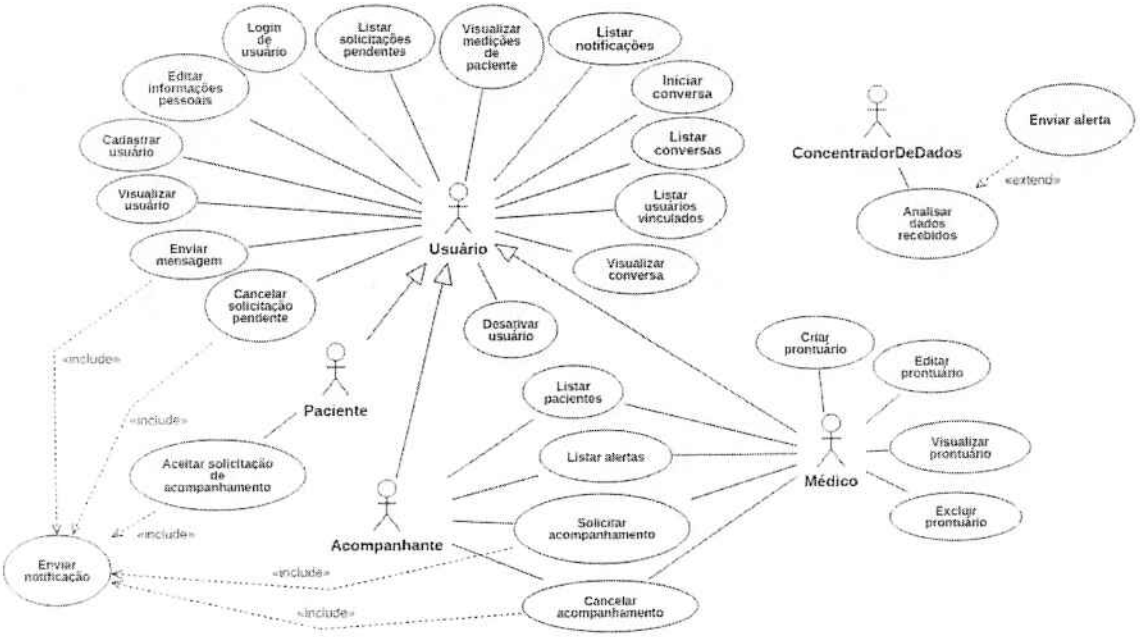
### 3.2.1 Modelo de Casos de Uso

O EIR é composto pelo sistema central e pelo concentrador de dados e foi elaborado um diagrama de casos de uso para cada parte. O sistema central apresenta cinco atores: Usuário, Paciente, Médico, Acompanhante e Concentrador

de Dados. O concentrador de dados, por sua vez, apresenta quatro atores: Sistema, Paciente Monitorado, Relógio de Coleta e Relógio de Envio.

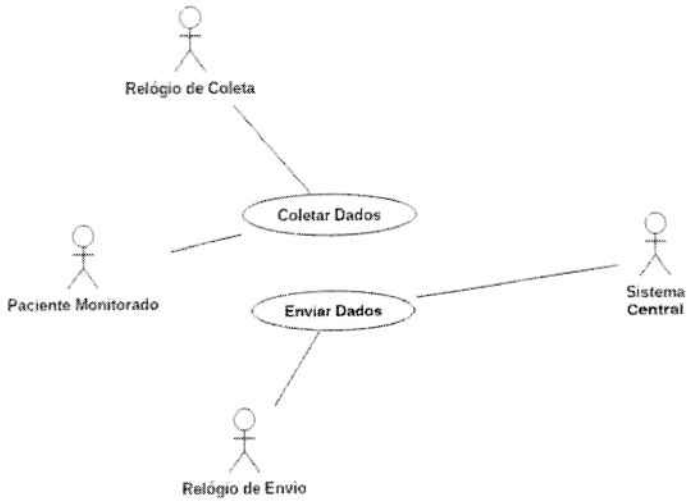
A Figura 7 representa o digrama de casos de uso do sistema central, considerando o concentrador de dados como ator.

*Figura 7 - Diagrama de Casos de Uso do EIR*




A Figura 8, por sua vez, representa os casos de uso do concentrador, considerando o sistema central como ator.

Figura 8 - Diagrama de Casos de Uso do Concentrador



### 3.2.2 Descrição Sucinta de Casos de Uso

Nesta seção estão apresentados os casos de uso do sistema, com uma breve descrição. A descrição completa dos casos de uso se encontram no Apêndice A.

#### 3.2.2.1 Casos de Uso do Sistema

**CASO DE USO 01 - Editar dados pessoais:**
Editar dados pessoais do perfil, de acordo com usuário (ex. Usuário edita sua data de nascimento, médico edita seu CRM).

**CASO DE USO 02 - *Login* de usuário:**
Autenticar o usuário no sistema através de dados de acesso.

**CASO DE USO 03 - Listar solicitações pendentes:**
Listar as solicitações de acompanhamento que ainda não foram aceitas ou recusadas.

**CASO DE USO 04 - Cancelar solicitação pendente:**
Cancelar uma solicitação de acompanhamento pendente.

**CASO DE USO 05 - Visualizar medidas do paciente:**
Exibir os dados coletados do paciente, através de valores médios e/ou gráficos.

**CASO DE USO 06 - Iniciar conversa:**
Iniciar uma conversa com um usuário vinculado.

**CASO DE USO 07 - Enviar mensagem**
Envia uma mensagem a um usuário vinculado.

**CASO DE USO 08 - Listar conversas:**

Listar conversas iniciadas.

**CASO DE USO 09 - Visualizar conversa:**

Visualizar histórico da conversa com outro usuário vinculado.

**CASO DE USO 10 - Listar usuários vinculados:**

Listar usuários vinculados ao usuário selecionado.

**CASO DE USO 11 - Listar notificações:**

Listar notificações recebidas.

**CASO DE USO 12 - Cadastrar usuário:**

Cadastrar o usuário, de acordo com seu papel.

**CASO DE USO 13 - Aceitar solicitação de acompanhamento:**

Permitir acesso às informações do paciente.

**CASO DE USO 14 - Solicitar acompanhamento:**

Solicitar acesso às informações de um paciente pelo médico ou acompanhante.

**CASO DE USO 15 - Cancelar acompanhamento:**

Cancelar o acompanhamento de um paciente.

**CASO DE USO 16 - Listar alertas:**

Listar alertas de saúde recebidos.

**CASO DE USO 17 - Listar paciente:**

Listar pacientes registrados no sistema através do nome/CPF.

**CASO DE USO 18 - Criar prontuário**

Criar prontuário médico de um paciente.

**CASO DE USO 19 - Editar prontuário:**
Editar prontuário médico de um paciente.

**CASO DE USO 20 - Visualizar prontuário:**
Visualizar o prontuário médico de um paciente.

**CASO DE USO 21 - Analisar dados recebidos**
Classificar, tratar e analisar medidas recebidas de um paciente, para identificar possíveis situações que requeiram atenção do médico e/ou acompanhante.

**CASO DE USO 22 - Enviar notificação:**
Enviar notificações relacionadas a atividades de outros usuários no sistema.

**CASO DE USO 23 - Enviar alerta:**
Enviar alertas de saúde para médicos e acompanhantes.

**CASO DE USO 24 - Visualizar usuário:**
Exibir dados pessoais de um usuário.

**CASO DE USO 25 - Desativar usuário:**
Desativar o acesso do usuário ao sistema e o acesso aos seus dados por outros usuários.

#### 3.2.2.2 Casos de Uso do Concentrador

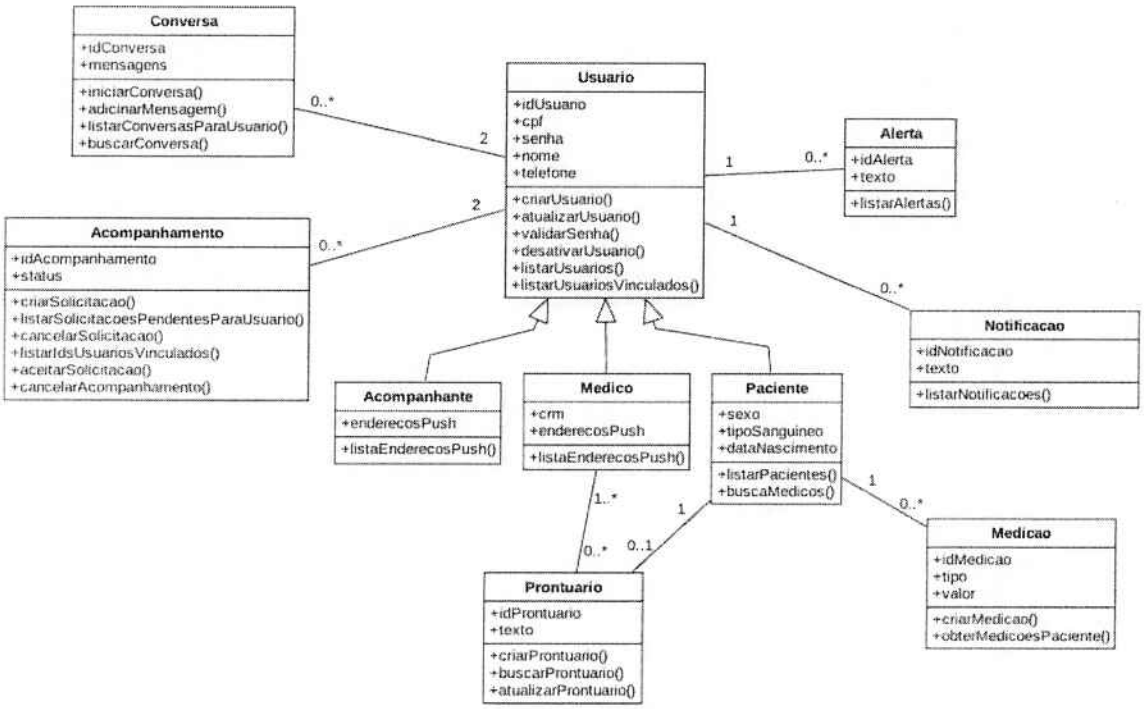
**CASO DE USO 1 - Coletar Dados:**
Coletar de dados do paciente monitorado a cada acionamento do relógio de coleta.

**CASO DE USO 2 - Enviar Dados:**
Enviar dados do paciente monitorado a cada acionamento do relógio de envio.

### 3.2.3 Diagrama de Classes

A Figura 9, apresenta o diagrama de classes do sistema central, representando as classes envolvidas assim como os atributos individuais de cada uma, além de ações específicas.

*Figura 9 - Diagrama de Classes*




## 3.3 Projeto e Implementação

Essa seção apresenta os resultados da fase de projeto e implementação, realizadas após a conclusão da modelagem do sistema.

### 3.3.1 Arquitetura Geral

Como apresentado nas seções anteriores, o EIR permite a coleta de dados biométricos através de um paciente, o processamento dos mesmos para identificar

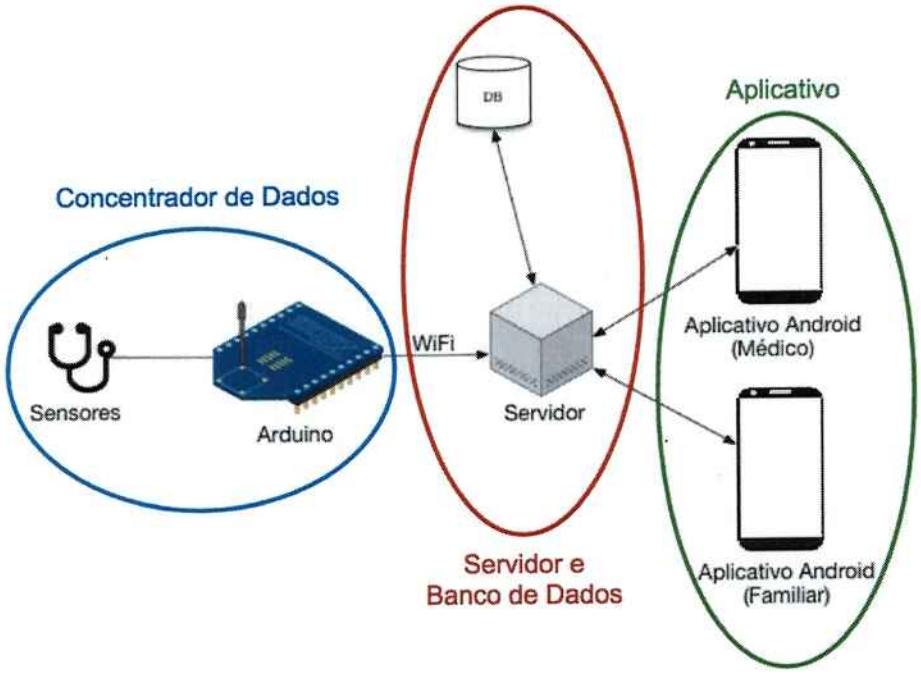
possíveis situações de risco e o acompanhamento dos dados coletados e das alertas, a partir de um aplicativo móvel. Tem-se, portanto, a presença de sensores em um dos extremos do sistema e um aplicativo móvel no outro, com um servidor possibilitando essa interação.

Foram, então, identificados três diferentes segmentos do sistema, que operam de forma semi-independente, mas dependem da interação com os demais para garantir o funcionamento do sistema.

O primeiro segmento é o concentrador de dados, responsável por coordenar a coleta de dados através de seus sensores. Através de uma conexão à Internet, o concentrador se comunica com o segundo segmento, composto pelo servidor e bancos de dados, o qual é responsável por receber os dados do concentrador, armazená-los e processá-los.

O servidor também se comunica com o terceiro segmento do sistema, o aplicativo móvel. Este recebe, do servidor, os dados coletados pelos sensores, as alertas sobre o paciente e demais informações relacionadas aos usuários do sistema, como perfis, mensagens e prontuários. O aplicativo móvel também envia dados dos usuários ao servidor, como informações de cadastro e solicitações de acompanhamento. A Figura 10 apresenta os três segmentos do sistema.

*Figura 7 – Segmentação do Sistema*



### 3.3.2 Concentrador de Dados

As principais funções do concentrador de dados são a constante coleta de dados através de um conjunto de sensores e o envio dos mesmos para o servidor. Para tal, podem ser identificados três principais componentes: o componente de sensores, a placa Arduino e o módulo de *Wi-Fi.*

O primeiro componente é constituído pelos sensores utilizados para a coleta de dados. Foi adquirido o kit *e-Health Sensor Platform*, desenvolvido pela *Cooking Hacks*, a divisão de *hardware* aberto da Libelium, uma empresa espanhola voltada para soluções de Internet das Coisas [2]. O kit foi concebido para auxiliar pesquisadores e desenvolvedores a coletar dados biométricos para experimentação e testes, possibilitando uma alternativa mais barata em relação a equipamentos médicos. Mas como consequência, não possui as devidas certificações para ser usado em pacientes em condições críticas ou que necessitem de medições precisas. Para o propósito deste trabalho, no entanto, ele atende às principais necessidades do grupo.

O kit inclui um conjunto de sensores, capazes de coletar informações como o batimento cardíaco do paciente, sua pressão, oxigenação do sangue e posição, além de uma placa que possibilita a integração desses sensores com uma placa Arduino ou Raspberry Pi. A Figura 11 apresenta o kit de sensores da e-Health conectados à uma placa Arduino.

*Figura 8 – Kit e-Health conectado a uma placa Arduino*

Fonte: https://www.cooking-hacks.com (2016)

O segundo componente, de Concentrador de Dados, é uma placa Arduino, que coordena a coleta de dados dos sensores e o envio dos dados coletados ao servidor. A placa foi programada para acionar a medição dos sensores desejados em intervalos constantes de tempo, assim como conectar-se ao servidor para enviar os dados coletados periodicamente.

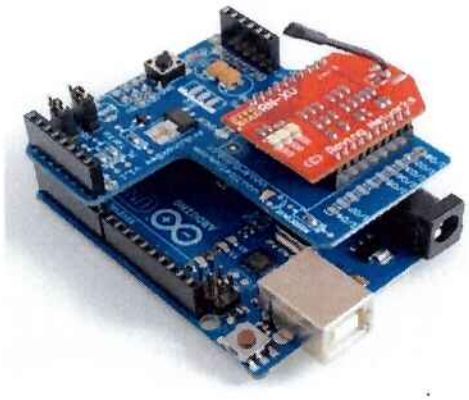
O kit e-Health também possui uma API para facilitar a coleta de dados dos sensores. Para realizar uma medição de temperatura, por exemplo, basta realizar a chamada:

```
float temperature = eHealth.getTemperature();
```

Um aspecto importante, quanto à coleta e ao envio dos dados, é a diferença entre os intervalos de tempo que coordenam cada uma das ações. Os dados precisam ser coletados constantemente, mas podem ser agrupados quando forem enviados ao servidor, evitando comunicação em excesso. No entanto, a memória interna da placa Arduino é limitada, e a diferença entre os tempos de coleta e envio é regida de acordo com o número de sensores utilizados e a frequência do acionamento de medição dos mesmos.

O terceiro componente a ser tratado é o módulo de *Wi-Fi* que foi conectado à placa Arduino para possibilitar a conexão à Internet através de redes *wireless* e, posteriormente, a comunicação do concentrador de dados com o servidor. A Figura 12 apresenta o módulo de Wi-Fi conectado à uma placa Arduino.

*Figura 9 – Módulo Wi-Fi conectado a uma placa Arduino*



Fonte: http://cooking-hacks.com (2016)

#### 3.3.2.1 Dificuldades Iniciais

Inicialmente, o grupo planejava enviar os dados para o servidor por meio de *Wi-Fi*, utilizando um módulo adicional do Arduíno. Após adquirir este módulo percebeu-se uma grande dificuldade em permitir a configuração da leitura dos sensores e a conexão a uma rede *Wi-Fi*, possivelmente devido ao conflito entre pinos da placa do Arduino. De forma a acelerar o desenvolvimento, sem prejudicar o restante do sistema, foi decidido que envio dos dados ao servidor seria feito por meio de um computador, de forma que o Arduino envia serialmente os dados ao USB do computador e um programa converte os dados para o formato JSON e os envia para o servidor.

#### 3.3.2.2 Programa do Arduino

O programa do Arduino, escrito em C++ utiliza a biblioteca e-health fornecida pelo fabricante dos sensores. Em um intervalo de tempo de cinco segundos, o programa acessa os sensores coletando os devidos valores de cada medida, cria uma *string* separando os valores por vírgula e os envia para o computador.

#### 3.3.2.3 Programa de Envio

Para realizar a leitura dos dados enviados através de USB pelo Arduino, foi criado um programa utilizando a linguagem Python. Este programa utiliza a biblioteca pySerial, que realiza leituras dos dados seriais recebidos via porta USB constantemente.

Ao receber um dado, o programa separa os valores contidos entre vírgulas presentes na *string* e monta um objeto no formato JSON incluindo também o *timestamp*, intervalo de medidas e o ID do paciente.

Para que não sejam feitas requisições em excesso, o programa espera e constrói um pacote com seis valores seguidos, antes de enviá-los ao servidor. O sistema foi desenvolvido de modo que o tempo de coleta e envio dos é configurável e independente para cada sensor ou dado coletado.

Durante o desenvolvimento foi adotado um intervalo de coleta de 5 segundos para a medição de batimentos cardíacos (bpm) e um envio a cada 30 segundos e, com isso, a recepção de novo pacote de dados do sistema é, portanto, a cada 30 segundos.

### 3.3.3 Aplicativo Móvel

A interface com o usuário final, seja ele médico, acompanhante ou familiar do paciente, é através de um aplicativo móvel.

A primeira decisão tomada envolveu a escolha da plataforma para desenvolvimento do aplicativo móvel. Diante de três diferentes sistemas operacionais (Android, iOS e Windows Phone), foi preciso escolher um deles e se o aplicativo móvel seria nativo ou utilizaria um *framework* para desenvolvimento multiplataforma, como React Native, Ionic ou Xamarin. Caso fosse optado pelo desenvolvimento nativo, o grupo deveria se limitar a apenas um sistema operacional, devido ao tempo adicional que seria requerido para desenvolver para as demais plataformas.

Caso fosse optado pelo desenvolvimento com auxílio de um dos *frameworks* mencionados, o tempo de desenvolvimento possivelmente seria mais curto e o resultado seria capaz de rodar em qualquer dispositivo móvel, sem restrições de sistema operacional ou tamanho de tela. No entanto, o desempenho final seria muito diferente se comparado ao de um nativo. Aplicativos *web* ou híbridos, gerados por *frameworks*, normalmente apresentam mais travamentos e picos de lentidão, além da dificuldade ou falta de acesso a recursos nativos do celular, que é um fator muito relevante para o desenvolvimento de aplicativos móveis completos.

Portanto, considerando os resultados das devidas análises, optou-se pelo desenvolvimento de um aplicativo móvel nativo, de forma que o resultado final fosse

mais robusto, tivesse um desempenho aceitável e, principalmente, que fosse capaz de usufruir de da maioria dos recursos nativos disponíveis no celular do usuário.

Feita a decisão sobre o desenvolvimento nativo, restava apenas escolher o sistema operacional a ser utilizado. Visando compatibilidade com um maior número de dispositivos móveis e a experiências prévias dos integrantes do grupo, optou-se por desenvolver o aplicativo móvel para o sistema Android.

#### 3.3.3.1 Dificuldades Iniciais

Feita a opção pelo desenvolvimento do aplicativo móvel nativo para o sistema Android, o principal desafio encontrado foi garantir que os diferentes usuários do sistema tivessem acesso às funcionalidades necessárias do melhor modo possível, garantindo a experiência de usuário inicialmente proposta pelos objetivos iniciais do sistema. Foram feitos alguns protótipos iniciais de tela, que se encontram no Apêndice B, para que fosse possível determinar os componentes necessários para garantir a correta navegação e o acesso a cada funcionalidade do sistema.

Outros pontos importantes a serem considerados foram: a necessidade de garantir o sistema de recebimento e tratamento de alertas, que seriam enviados pelo servidor, a visualização dos dados históricos e dados em tempo real.

#### 3.3.3.2 Navegação e Experiência de Usuário

A mudança de telas e controle do conteúdo exibido em aplicativos Android é realizada com o auxílio de duas classes nativas: *Activities* e *Fragments*.

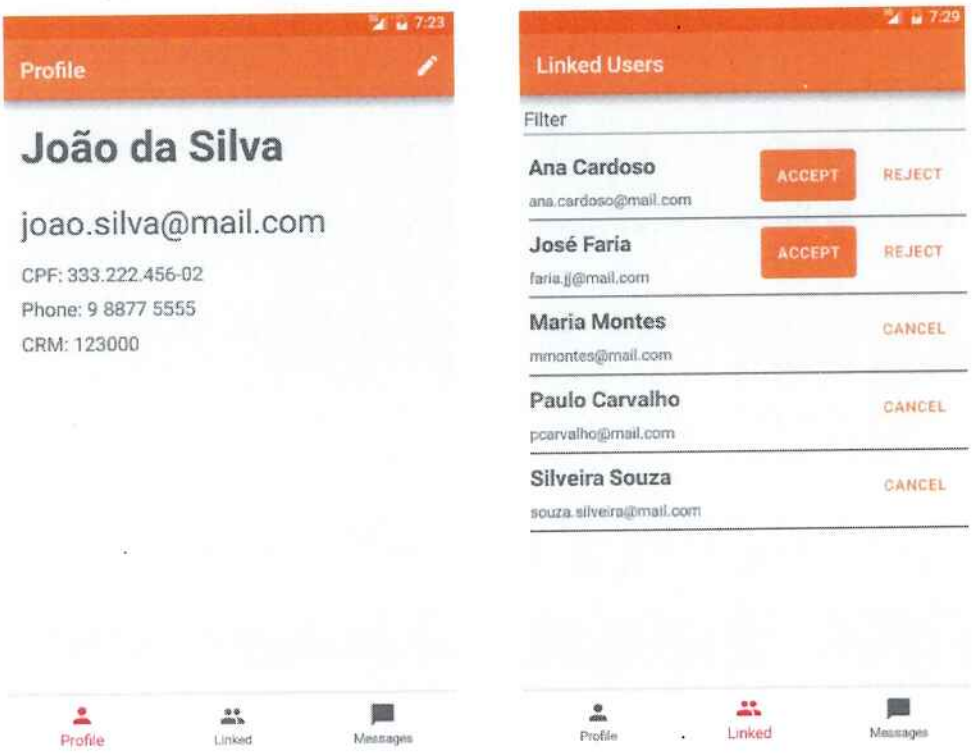
*Activities* em geral são utilizadas quando há uma mudança de contexto mais significativa durante a navegação e pode ser interpretada como uma nova tela adicionada no topo da tela anterior.

*Fragments* são utilizados para alterar ou introduzir conteúdo dentro de uma *Activity*, possibilitando que o usuário tenha acesso a novas informações ou funcionalidades sem interrupção. *Fragments* também permitem que um mesmo conteúdo seja utilizado em diferentes *Activities*, o que facilita a manutenção do

padrão de telas adotado no aplicativo móvel e a duplicação desnecessária de código.

Para o aplicativo móvel do sistema, optou-se por concentrar algumas funcionalidades em sua página inicial, como perfil do usuário e listagem de usuários vinculados, que podem ser acessados através de um menu na parte inferior da tela. Tem-se no caso uma *HomeActivity*, cujo conteúdo é controlado com a troca do *Fragment* a ser exibido. Para o perfil e a listagem de usuários vinculados, por exemplo, há um *ProfileFragment* e um *LinkedUsersFragment*. Demais funcionalidades do aplicativo móvel podem ser acessadas a partir dessa página inicial. A Figura 13 apresenta exemplos de tela do aplicativo móvel na *HomeActivity*.

*Figura 13 – ProfileFragment e LinkedUsersFragment vistos na HomeActivity*



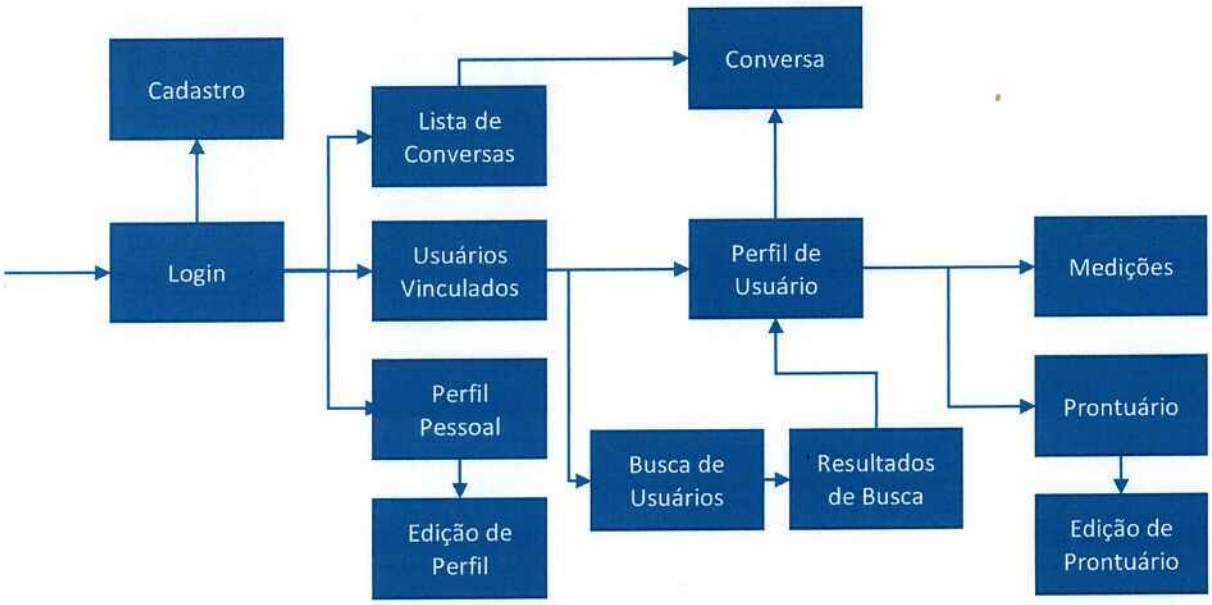

A utilização de *Fragments* também permitiu evitar a reescrita de código, uma vez que a apresentação do perfil do usuário autenticado do sistema e a visualização do perfil de outro usuário eram muito similares.

Apesar de ser um único aplicativo móvel, o mesmo pode ser utilizado por diferentes tipos de usuários (pacientes, médicos e acompanhantes) e, portanto, seu comportamento deveria ser diferente para cada um deles. Como o acesso ao

aplicativo móvel só é possível após a autenticação do usuário, é possível utilizar a informação do tipo de perfil que está utilizando a aplicação e, portanto, controlar os componentes e as funcionalidades que estarão disponíveis para aquele tipo de usuário. Médicos e acompanhantes, por exemplo, têm acesso ao perfil de pacientes e podem realizar uma solicitação de acompanhamento, enquanto pacientes só conseguem acessar o perfil de usuários que o acompanham e aceitar as solicitações recebidas.

Um diagrama contendo o fluxo de telas do aplicativo móvel é apresentado na Figura 14.

*Figura 14 – Fluxo de telas do aplicativo*




### 3.3.3.3 Visualização das Medições

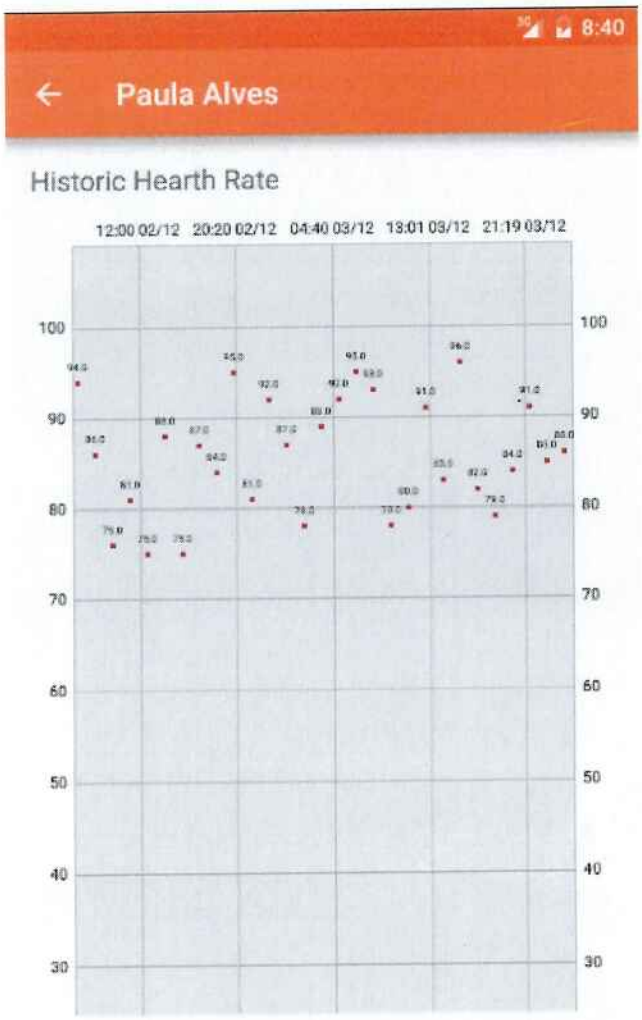
Utilizando o aplicativo móvel, o usuário deve acompanhar as medições do paciente, através de histórico recente ou, ainda, em tempo real. Para tal, foi necessário configurá-lo para suportar as duas fontes de dados e permitir a sua visualização de forma clara.

Utilizando uma biblioteca gráfica *open source* para Android, chamada MPAndroidChart, foi possível criar gráficos interativos que possibilitam ao usuário

um maior controle sobre a visualização dos dados coletados [26]. A Figura 15 apresenta um exemplo de visualização dos dados no aplicativo móvel.

Com os dados históricos sendo processados no servidor, a sua obtenção pelo aplicativo móvel é realizada através de uma simples requisição, bastando configurar a criação dos gráficos desejados com a biblioteca gráfica.

*Figura 15 – Exemplo de visualização de dados*




A obtenção dos dados em tempo real é realizada de maneira diferente, uma vez que realizar requisições constantes para o servidor solicitando os dados atualizados não é a prática ideal, considerando a possibilidade de milhares de usuários utilizando o sistema e gerando dezenas de solicitações por minuto cada.

Utilizando o banco de dados RethinkDB, é possível configurar uma conexão de modo a ser informado toda vez que ocorrer uma alteração em uma tabela ou em

um conjunto de dados. Para não expor o banco de dados ao aplicativo móvel, é necessário que essa conexão seja criada entre o servidor e o banco de dados, enquanto uma conexão similar deve ser estabelecida entre o aplicativo móvel e o servidor. Essas conexões ocorrem através de *sockets*, que permitem que a troca de informações só ocorra na presença de determinados eventos, evitando requisições desnecessárias ao servidor.

Quando o usuário acessa a tela de medições em tempo real, o aplicativo móvel estabelece a conexão com o servidor, solicitando ser informado sempre que novos dados estiverem disponíveis para determinado usuário. O servidor então cria uma outra conexão com o banco de dados, e fica à espera de um evento que indique que a tabela com as medições do usuário solicitado foi alterada. Caso novos dados sejam incluídos no banco de dados, durante o período em que essa conexão estiver estabelecida, o servidor os recebe, para então encaminhá-los ao aplicativo móvel, que pode atualizar os gráficos exibidos na tela com as informações mais recentes. Quando a tela de medições é fechada, ambas as conexões são encerradas.

#### 3.3.3.4 Alertas e Notificações

A plataforma Android possui um sistema de notificações nativo, que permite que aplicativos móveis notifiquem o usuário até mesmo quando o celular estiver com a tela desligada. Notificações podem disparar um toque sonoro e vibração do dispositivo, configurados pelo usuário, e podem ser visualizadas na parte superior da tela do celular, onde o usuário pode interagir com as mesmas.

O aplicativo móvel do sistema utiliza as notificações para dois casos distintos. O primeiro envolve o aviso ao usuário sobre interações com outros usuários, como o recebimento de uma nova mensagem ou solicitação de acompanhamento. O segundo trata do aviso de potenciais situações de risco, observadas durante a coleta de dados dos pacientes monitorados. Em ambos os casos, a notificação é disparada pelo aplicativo móvel, após o recebimento de uma *push notification* enviada pelo servidor, após o acontecimento de um dos eventos mencionados anteriormente.

Desde o envio de uma *push notification* pelo servidor até a sua exibição pelo aplicativo móvel, alguns passos devem ser realizados.

O primeiro envolve configurar o recebimento do conteúdo enviado, utilizando *Receivers*, que são classes nativas do Android. Elas são responsáveis por receber o conteúdo enviado e disparar novos eventos para demais classes do aplicativo móvel.

O passo seguinte é configurar as classes que recebem esse evento e o respectivo tratamento. É possível configurar o aplicativo móvel de tal modo que, caso determinada tela estiver aberta, pode-se suprimir o disparo da notificação e optar por atualizar o conteúdo do mesmo. Ao receber um *push* indicando a presença de uma nova solicitação de requisição, por exemplo, pode-se optar por apenas atualizar a listagem de usuários, caso o usuário já esteja visualizando a mesma. Se a notificação não for suprimida, um diferente tipo de *Receivers* recebe o conteúdo do *push* e configura a notificação a ser exibida para o usuário.

Por fim, foi necessário configurar o comportamento desejado após a interação do usuário com a notificação. Após clicar em uma notificação indicando que há uma nova solicitação de acompanhamento, o aplicativo móvel é aberto e a listagem de usuários vinculados é exibida. No caso de uma notificação, indicando uma possível situação de emergência, o usuário deve ser redirecionado para a tela de medições do paciente, por exemplo.

### 3.3.4 Servidor e Banco de Dados

Devido à grande quantidade de requisições e volume de dados gerados pelos sensores dos pacientes, foi desenvolvida uma arquitetura utilizando microserviços e dois bancos de dados distintos.

A tecnologia escolhida para o software do servidor foi o NodeJS utilizando Express como o *framework* web, devido à familiaridade dos integrantes do grupo com essa plataforma, bem como sua capacidade de operar com alto desempenho.

Para os bancos de dados, são necessários, tanto o armazenamento dos dados de interação com o usuário, como também o armazenamento e a

transmissão de dados em tempo real. Para que ambos os requisitos fossem atendidos adequadamente, foram adotadas duas tecnologias simultaneamente:

- **MongoDB**: que facilita o desenvolvimento do software por ser orientado à interação com o usuário e permite a clusterização, de modo a aumentar a vazão de dados.

- **RethinkDB**: que permite que *sockets* sejam abertos diretamente com o banco de dados, a fim de serem notificados assim que novas entradas esteja disponível.

Os dados da interação com o usuário ficam então armazenados no banco de dados MongoDB, enquanto os dados referentes às últimas medições, que devem ser obtidas em tempo real, são armazenados no RethinkDB. Um exemplo dessa separação pode ser observado na figura 16, que apresenta a estrutura dos dados referentes a usuários e o histórico de medições em um banco e as medições mais recentes em outro.

A aplicação principal do servidor funciona da seguinte forma:

- Assim que um novo conjunto de medidas é recebido, é armazenado diretamente no banco de dados RethinkDB, de modo que os dados relevantes sejam propagados para os dispositivos responsáveis.

- Em seguida, a medida é processada por um microserviço separado, responsável por validar cada um dos parâmetros recebidos e gerar um alerta conforme a necessidade.

*Figura 16 – Estrutura de dados em cada banco de dados*

| MongoDB | RethinkDB |
|---|---|
| <Usuário><br>{<br>id: String,<br>email: String,<br>senha: String,<br>tipo: Inteiro,<br>cpf: String,<br>nome: String,<br>telefone: String,<br>crm: String,<br>enderecosPush: [String],<br>prontuario: {<br>parametrosMonitoramento: {<br>ligado: Booleano,<br>maximo: Inteiro,<br>minimo: Inteiro,<br>variancia: Inteiro<br>]<br>},<br>idsUsuariosMonitorados: [String],<br>idsUsuariosPendentes: [String]<br>] | <Medição><br>{<br>idPaciente: String,<br>bpm: [Inteiro],<br>intervaloEntreMedidas: Inteiro,<br>timestampInicial: Inteiro,<br>sincronizado: Booleano<br>} |
| <Histórico><br>{<br>idPaciente: String,<br>historicoDiario: {<br>bpm: [Inteiro],<br>intervaloEntreMedidas: Inteiro,<br>timestampInicial: Inteiro<br>]<br>} | |

Os alertas são enviados na forma de *push notifications*, utilizando o serviço GCM (*Google Cloud Messaging*), fornecido pelo Google para a plataforma Android.

Como descrito na sessão 3.3.2.3, o intervalo de envio de dados é relativamente pequeno, mas a quantidade de dados enviados ao servidor, por paciente, é considerável, o que leva ao surgimento de diversas questões de desempenho e escalabilidade do servidor para que fosse possível suportar milhares de pacientes simultâneos. Considerando a configuração adotada de 1 pacote enviado a cada 30 segundos por paciente com 5 medidas, tem-se:

- 3600 segundos/hora / 30 segundos/pacote = 2 requisições por minuto por paciente.

- 24 horas * 3600 segundos / 5 segundos por medida = 17280 medidas por sensor por dia.

Como mencionado anteriormente, foi adotada uma arquitetura de microserviços visando atender a esse grande volume de dados e requisições, permitindo a escalabilidade individual de diferentes módulos do sistema, de acordo com a demanda, para manter o tempo de processamento, resposta e intervalo de detecções baixo.

Medindo o desempenho do servidor, foram obtidos tempos de resposta abaixo de 150 milissegundos, o que garante uma boa precisão quando comparados ao intervalo de 30 segundos adotado entre cada envio de dados.

#### 3.3.4.1 Dificuldades Iniciais

Inicialmente, o trabalho foi feito considerando o MongoDB para o banco de dados, devida à familiaridade dos integrantes do grupo com essa tecnologia. Porém, durante a implementação das medições em tempo real, percebeu-se que o MongoDB não atenderia aos requisitos, uma vez que o aplicativo móvel precisaria ficar constantemente acessando o banco de dados, o que causaria problemas de desempenho. Para contornar este problema adotou-se o RethinkDB como banco de dados secundário, focado apenas nas interações em tempo real. Esta tecnologia foi escolhida, pois permite que seja aberto um *socket* direto entre o aplicativo móvel e o servidor, de modo que o aplicativo móvel torna-se passivo e é avisado ao surgirem novos dados.

#### 3.3.4.2 Express

Foi utilizado o Express para criação dos *webservices* do servidor, de modo que a aplicação do servidor foi dividida em três camadas:

- ***Models***: Nesta camada foram criados os objetos para representar os dados com os quais o servidor trabalha. Essas classes foram criadas utilizando a biblioteca Mongoose, de modo que são definidos *schemas* para os dados, definindo a estrutura a ser salva no banco de dados. Com

isso são automaticamente geradas diversas funcionalidades que facilitam o CRUD dos elementos.

- ***Controllers***: Esta camada é responsável pela a lógica de negócio da aplicação. Através dela, os dados são manipulados no banco de dados, processando as informações recebidas do *49liente* e retornando os dados prontos para serem consumidos. Os *controllers* foram separados de acordo com suas responsabilidades, de forma que o principal *controller* do servidor chama-se *MeasurementController* no qual as medições são recebidas e os dados são encaminhados para o MongoDB e o RethinkDB, tornando-se prontos para serem consumidos pela aplicação e outros serviços.

- **Routes**: É a camada que contém as definições dos pontos de acesso do servidor, fazendo a interface com o mundo externo.

#### 3.3.4.3 Autenticação

No sistema desenvolvido, as APIs utilizam um controle de acesso e autenticação baseada em *tokens,* seguindo o padrão Oauth [19]. A partir do *login* do usuário com e-mail e senha, é gerado um *token* único e encriptado, contendo um ID e tipo de usuário.

Todas as requisições, exceto o *login,* requerem que um *token* válido seja passado no campo *Authorization* do *header*. Foi criado um *middleware* do Express que pré-processa as requisições, decodificando o *token* e recupera o ID do usuário. A partir do ID é feita uma validação em nível de acesso para verificar se o usuário solicitante da requisição possui as permissões para acessá-la. Caso não seja fornecido um *token* ou o usuário não possua as permissões necessárias, é retornado o código HTTP 401, indicando "Não Autorizado".

#### 3.3.4.4 Microserviços

Seguindo a arquitetura de microserviços, o servidor foi implementado com funcionalidades isoladas e independentes que se intercomunicam. Para atender ao escopo da aplicação foram criados três microserviços:

***App***: Responsável pela parte de CRUD de dados, gerenciamento de configurações do usuário e recebimento de dados. Este microserviço é o principal a ser acessado diretamente pelo aplicativo móvel.

***History***: Responsável por processar as medições armazenadas pelo servidor e gerar em seguida dados estatísticos de histórico do usuário. Com isso, o microserviço *APP* apenas armazena dados brutos de medidas, acelerando sua vazão de dados.

***Monitoring***: Responsável por analisar as medições recebidas e detectar situações de risco para o paciente, gerando notificações conforme o necessário. Por ser um microserviço, permite que esta funcionalidade crítica do sistema seja escalável, mantendo o tempo de resposta dentro dos limites aceitáveis.

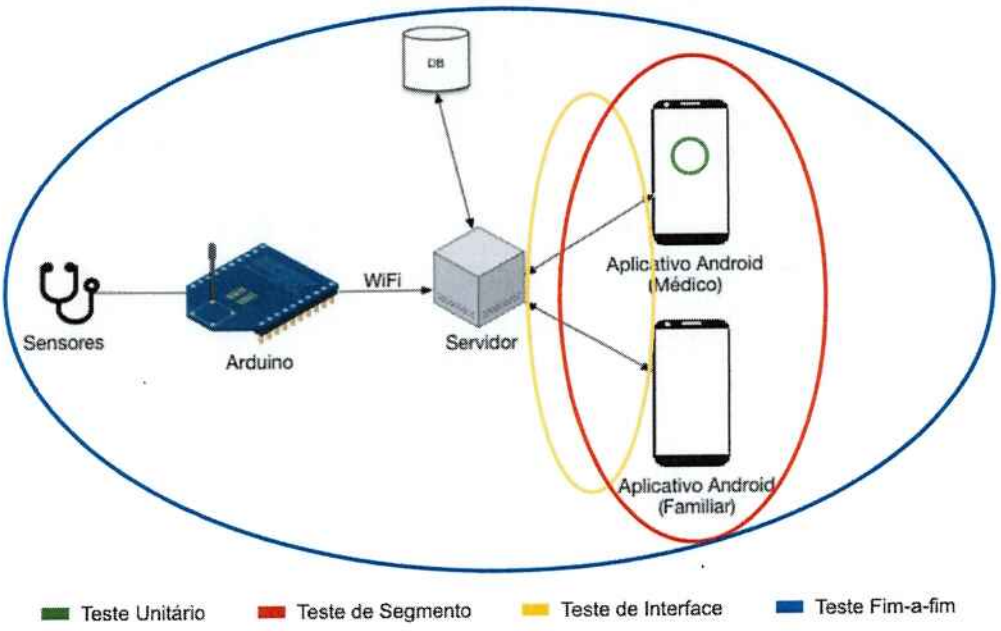
#### 3.3.4.5 Monitoramento de Emergências

A detecção de situações de emergência é feita baseada em parâmetros pré-configurados para cada paciente em questão. Esta configuração é feita em termos de limites mínimo, máximo e variação máxima dos sinais vitais, e enviada ao servidor por meio do aplicativo móvel. O microserviço *Monitoring*, por sua vez, a cada nova medição recebida, valida cada um dos valores em relação ao seu valor mínimo, máximo e sua variação. Caso algum desses valores fuja dos valores pré-configurados, é detectada uma situação de emergência e *push notifications* são enviados aos usuários associados ao paciente.

## 3.4 Avaliação e Testes

O desenvolvimento do sistema foi feito de forma que os componentes e segmentos envolvidos fossem independentes entre si. Isso significa que o sistema final deveria ser capaz de futuramente ter componentes modificados ou substituídos sem prejudicar o funcionamento do conjunto. Para que essa independência fosse obtida, foram executadas quatro categorias de testes: unitário, de segmento, de interface e fim-a-fim. A figura 17 apresenta as categorias em relação aos componentes do sistema.

*Figura 17 – Categorias de teste executados*




### 3.4.1 Testes Unitários

Os testes unitários foram testes executados sobre estruturas atômicas de código, como funções do aplicativo, do servidor e do banco de dados.

São também desta categoria, os testes do sensor batimento cardíaco a partir da conexão dele com a porta serial da placa Arduino, verificando os valores obtidos

em duas condições diferentes: desconectado e conectado a uma pessoa, de forma a determinar se os resultados obtidos eram realistas.

### 3.4.2 Testes de Segmento

Como já visto na Figura 10, o sistema foi desenvolvido em três segmentos isolados: aplicativo móvel, servidor e banco de dados e concentrador de dados. Portanto inicialmente, cada segmento foi testado individualmente, com intuito de garantir o funcionamento individual. Para isso, foram utilizadas ferramentas que simulam requisições de um segmento para o outro. As interfaces do servidor, por exemplo, foram testadas utilizando a ferramenta Postman, que permite realizar chamadas à API, e simulando dados reais no mesmo formato esperado pelos envios do concentrador de dados e aplicativo móvel.

Para garantir o funcionamento do segmento do concentrador de dados, foi testada a conexão do sensor de batimento cardíaco com o Arduino e validada a captura dos dados pelo programa de varredura.

### 3.4.3 Testes de Interface

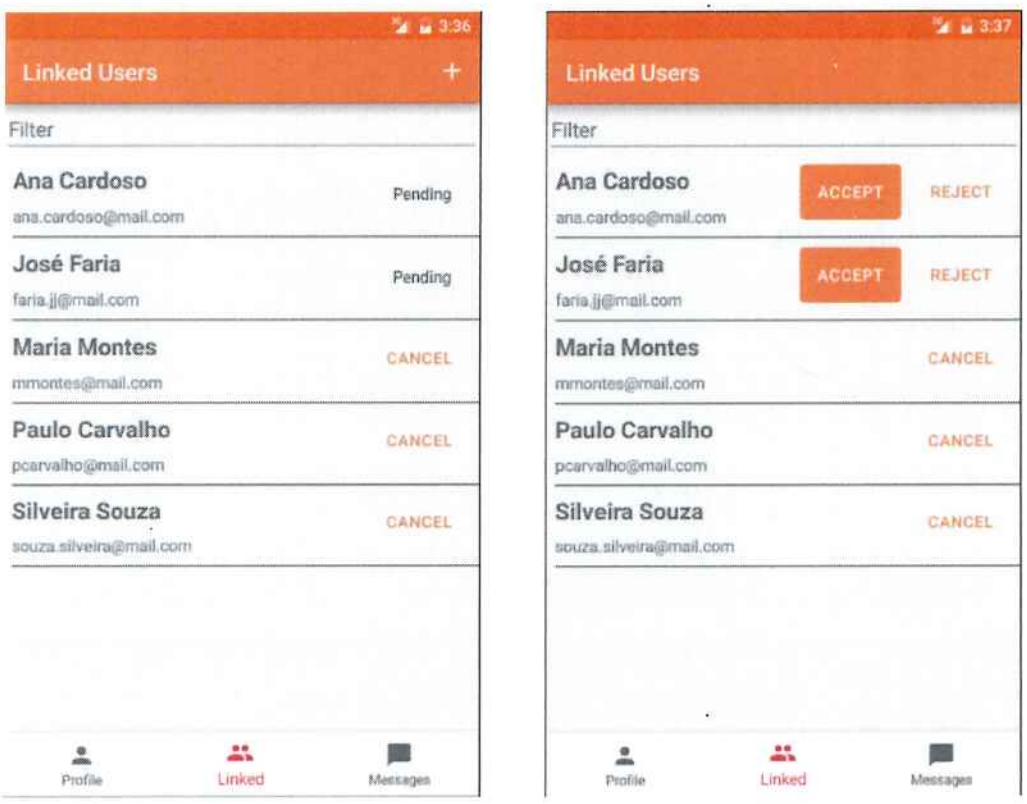
Para verificar o funcionamento do aplicativo móvel e sua interface com o servidor, foram realizados dois tipos de teste distintos.

O primeiro tipo permitu observar o comportamento do aplicativo móvel de acordo com o usuário autenticado e suas interações, visando verificar de que forma o aplicativo móvel se comportava para os diferentes tipos de usuário e se a navegação entres as diferentes telas ocorreria de acordo com o planejado. Para esse teste, foram utilizados dados fictícios, gerados pelo próprio aplicativo de forma a popular os elementos da tela. Um exemplo de verificação de funcionamento das permissões entre os diferentes tipos de usuário foi a comparação de tela de usuários vinculados de um médico e de um paciente. Na tela do médico, esperava-se que as solicitações pendentes fossem listadas indicando seu estado, além da

presença de um botão para realizar novas solicitações no canto superior direito. Na visão de um paciente, esse mesmo botão não deveria estar presente, e as solicitações pendentes deveriam apresentar as opções de aceitação e recusa. Esse comportamento pode ser verificado na Figura 18.

Para o segundo tipo de teste, para a interface com o servidor, optou-se por observar o comportamento do aplicativo móvel durante a visualização de dados de histórico e em tempo real. Para tal, foi inserido no banco de dados um grande volume de medições para determinado usuário, a fim de compor os dados de histórico. A tela de medições no aplicativo móvel foi então aberta, para que fosse verificado que os dados foram obtidos através do servidor e exibidos corretamente. Foram então enviadas novas medições ao servidor, novamente utilizando o Postman, de forma que se pudesse confirmar a atualização automática dos gráficos de tempo real a cada novo envio de dados. Na Figura 19, pode ser observada a tela de medições, contendo os gráficos de tempo real e dados de histórico, instantes antes e após o envio de um novo conjunto de medições ao servidor, verificando-se que os novos dados foram incluídos no gráfico de tempo real.

*Figura 18 – Tela de usuários vinculados na visão de um médico (esq.) e paciente*



Para realização do teste de interface do concentrador de dados com o servidor, foi feita a conexão do Arduino com a API do servidor para que os pacotes de medidas coletadas fossem enviados. Utilizando o sensor de batimento cardíaco, foram observados os valores coletados e enviados ao servidor (Figura 20), assim como os valores posteriormente observados no banco de dados (Figura 21).

*Figura 19 – Exibição de dados históricos e em tempo real*


*Figura 20 – Dados coletados pelo concentrador de dados observados em um terminal*

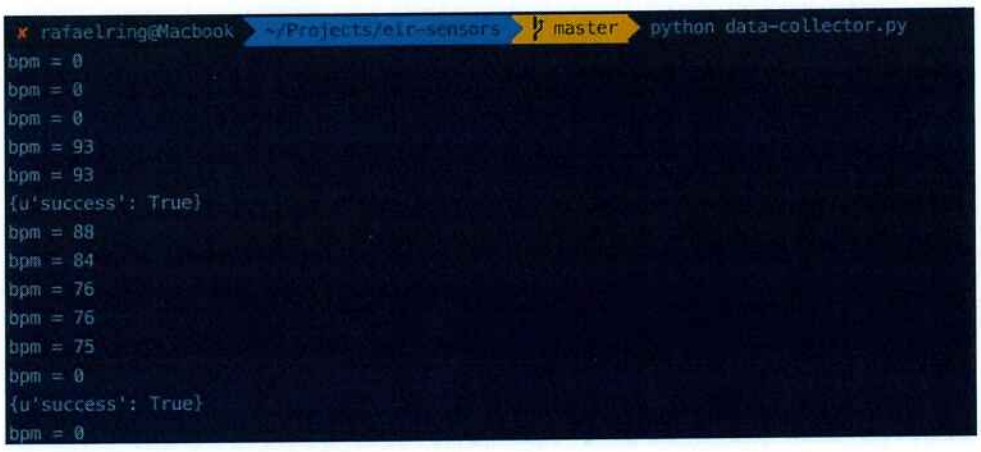


*Figura 21 – Dados coletados observados no banco de dados*


```
{
  "new_val": {
    "bpm": [
      84 ,
      76 ,
      76 ,
      75 ,
      0 ,
      0
    ] ,
    "initialTime": 1480468077 ,
    "pacientId": "57d7fc276f99f92f5fd3dc4a" ,
    "synced": false ,
    "timeInterval": 5
  } ,
  "old_val": {
    "bpm": [
      0 ,
      0 ,
      0 ,
      93 ,
      93 ,
      88
    ] ,
    "initialTime": 1480468077 ,
    "pacientId": "57d7fc276f99f92f5fd3dc4a" ,
    "synced": false ,
    "timeInterval": 5
  }
}
```


### 3.4.4 Testes Fim-a-fim

Finalmente, para realização do teste fim-a-fim, envolvendo a integração de todos os segmentos do sistema desenvolvido e suas interfaces, os componentes foram conectados de forma que as medições obtidas pelo sensor fossem transmitidas do Arduino para o Programa de Envio, do Programa de Envio para o servidor e banco de dados e finalmente, destes para o consumo final e exibição no aplicativo móvel. Considerando que todos os testes anteriores foram realizados e

concluídos com sucesso, a execução do teste fim-a-fim, por sua vez, também foi realizada sem dificuldades.

## 3.5 Considerações Finais do Capítulo

Neste capítulo foram apresentados os resultados das fases do desenvolvimento do EIR, destacando-se as tomadas de decisão em diversos passos. Pode-se dizer que os requisitos foram elicitados com base em sistemas similares existentes e opiniões de especialistas do assunto. Ainda, dada a arquitetura escolhida, as decisões de projeto tomadas e a sequência de testes realizados, a integração dos componentes do sistema e seu funcionamento como um todo foram concluídos com sucesso.

# 4. Considerações Finais

## 4.1 Conclusões

Considerando que o trabalho em questão apresentava como ideia central o desenvolvimento de um sistema capaz de realizar medições de sinais vitais de um paciente e expor essas informações em um aplicativo móvel, gerando alertas e permitindo o acompanhamento do paciente por outros usuários, os integrantes do grupo acreditam que o trabalho, desde o preparo inicial até a implementação, foi uma valiosa fonte de conhecimento e experiência como um todo.

Dado que o tamanho e a complexidade do sistema desenvolvido foram consideráveis, desde a fase inicial até os momentos finais do trabalho, pode-se dizer que o grupo foi capaz de colocar em prática diversos aprendizados adquiridos ao longo dos anos de faculdade.

Para verificar a viabilidade e o valor de um sistema como este, os integrantes do grupo entraram em contato com pessoas do setor de saúde que mostraram grande interesse na ideia, mas que apontavam para as dificuldades de se implementar um sistema na área da saúde, principalmente por envolver questões jurídicas quando se lida com a saúde das pessoas. De fato, as questões jurídicas e os termos de responsabilidade envolvidos são sensíveis e requerem uma atenção adicional. Desta forma, o grupo concluiu que o sistema não é um substituto aos prestadores de serviço de *Home Care*, mas sim, um complemento, um instrumento de facilitação que pode, ou não, ser utilizado por empresas de *Home Care*, por exemplo.

Como o sistema concebido, inicialmente, teria diversas funcionalidades, cenários e tipos de usuário, foi importante fazer sua modelagem de forma a entender, detalhar e documentar os aspectos desejados. Foram construídos casos de uso, diagramas de classe e diagramas de sequência que trouxeram um entendimento e clareza às funcionalidades especificadas.

A seguir, foi dado início o projeto, separando o sistema em três segmentos principais, como detalhado no capítulo 3: concentrador de dados, servidor e banco de dados e aplicativo móvel, para que a implementação desses segmentos pudesse ser realizada de forma mais independente. Pode-se dizer que, além dos

conhecimentos adquiridos ou evoluídos durante o desenvolvimento destas três segmentos da arquitetura do sistema, foram também de alta valia, as decisões de trabalho tomadas, a respeito de direcionamentos do sistema, ou ainda, as escolhas de tecnologias que ocorreram durante todo o processo.

## 4.2 Contribuições

Atualmente, ainda existem poucos produtos desenvolvidos de aplicações móveis na área da saúde, com foco em Internet das Coisas. Apesar da existência de um grande número de *weareables* integrados com aplicativos móveis, a maioria deles foca simplesmente a prática de exercícios físicos, o que deixa uma grande margem para desenvolvimento de sistemas na área da saúde, com suporte de tecnologias avançadas.

Dentro desse contexto, pode-se dizer que o desenvolvimento de EIR é uma contribuição para esse nicho de aplicação, considerando que o fornecedor do kit de sensores somente lançou o seu produto de monitoração de pacientes quando o grupo estava trabalhando no seu trabalho de conclusão do curso.

Do ponto de vista da condução do trabalho, procurou-se utilizar o conhecimento adquirido ao longo curso, na tentativa de trazer benefícios reais para a sociedade, mostrando o quanto a tecnologia pode mudar a vida das pessoas, quando bem utilizada.

Em comparação com os sistemas similares analisados durante a fase de concepção, no capítulo 2, o trabalho realizado pelo grupo explorou diversos aspectos importantes. Entre eles pode-se mencionar:

- Utilização de dois bancos de dados com responsabilidades diferentes, sendo um para dados em tempo real e o outro para histórico e dados estáticos como cadastro de usuário.

- Criação de alertas dinâmicos e configuráveis para situações de emergência.

- Gestão de pacientes através dos perfis: Acompanhante e Médico.

- Utilização de uma arquitetura do sistema que trouxe contribuições na implementação, testes e integração de componentes. Essa organização garantiu o funcionamento esperado do sistema e a independência de cada componente.

## 4.3 Trabalhos Futuros

De forma a complementar o sistema desenvolvido e melhorar a experiência do usuário como um todo, existem certas melhorias e funcionalidades adicionais que poderiam ser incorporadas ao sistema atual:

- ***Streaming* de vídeo do paciente**. Este recurso utilizaria câmeras de vídeo em gravação constante do local, em que o paciente se situa, e transmitiria esses dados para o sistema. Com este recurso, os familiares e o médico seriam capazes de solicitar através do aplicativo móvel, a imagem em tempo real do paciente, o que descartaria erros de sensores que inicialmente poderiam sinalizar situações de emergência, mas se tratariam de sensores desconectados, por exemplo.

- **Mecanismo automático de ligação direta para a emergência**. Uma funcionalidade adicional sugerida seria, na eventualidade da detecção de uma situação de risco, fazer uma ligação automática para o hospital e um robô previamente configurado leria uma mensagem de voz com a localização do paciente, para que fosse feito o envio de uma ambulância ao local.

- **Uso de equipamentos médicos reais**. Uma melhoria futura seria também a troca dos sensores de baixo custo por equipamentos médicos reais já certificados e homologados pelos devidos órgãos de saúde responsáveis.

- **Expansão e melhorias no aplicativo móvel**. Existe uma série de melhorias que poderiam ser feitas no aplicativo móvel. Uma delas seria uma maior customização de funcionalidades para cada usuário, de forma que ele

gerencie as informações desejadas e a forma como elas são exibidas, dando maior controle de gestão de informações ao usuário.

- **Aumento do número de tipos de medições realizadas**. Outra melhoria importante a ser incorporada seria aumentar a quantidade de medições obtidas sobre o paciente. Para isso, seria necessário adquirir mais sensores ou aparelhos médicos especializados em obter dados biométricos diferentes daqueles já utilizados.

- **Concentração da captura de dados em menos sensores**. Talvez um dos maiores incômodos a respeito do uso do sistema seja a necessidade de o usuário estar conectado a uma série de sensores independentes com fios e conectores diferentes. Uma das melhorias futuras, que poderia ser realizada, é a junção de sensores em um mesmo dispositivo, por exemplo em uma pulseira que seja capaz de medir temperatura, pressão e batimentos, melhorando a comodidade do paciente com relação ao uso do sistema.

- **Analise de Dados**: Um ponto interessante sobre o trabalho desenvolvido é que o sistema trabalha com a captura de uma quantidade imensa de dados referente à saúde dos pacientes e, portanto, seria possível utilizar tamanho volume de dados para criar relações e análises entre determinados valores biométricos e potenciais riscos à saúde humana. A ideia seria aproveitar os dados obtidos com o sistema e usá-los, de forma anônima, para auxiliar no desenvolvimento de pesquisas e diagnósticos no setor da saúde.

# REFERÊNCIAS

[1] CARVALHO, T. S. **Monitoramento Remoto de Pacientes em Ambiente Domiciliar**. Disponível em <http://www.lbd.dcc.ufmg.br/colecoes/sbrc/2010/0071.pdf>. Acesso em: 25 ago. 2016

[2] LIBELIUM. **e-Health Sensor Platform V2.0 for Arduino and Raspberry Pi [Biometric / Medical Applications]**. Disponível em <https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical>. Acesso em: 10 set. 2016

[3] DR. CONSULTA. **Dr.consulta**. Disponível em <https://www.drconsulta.com/>. Acesso em: 20 nov. 2016

[4] RETHINKDB. **Thirty-second quickstart with RethinkDB**. Disponível em <https://www.rethinkdb.com/docs/quickstart/>. Acesso em: 28 set. 2016

[5] AMAZON WEB SERVICES, INC. **O que é NoSQL?**. Disponível em <https://aws.amazon.com/pt/nosql/>. Acesso em: 21 set. 2016

[6] JSON. **Introducing JSON**. Disponível em <http://www.json.org/>. Acesso em: 21 set. 2016

[7] MANDOWN APP. **ManDown App**. Disponível em <http://www.mandownapp.com/>. Acesso em: 23 out. 2016

[8] MONGODB, INC. **The MongoDB 3.2 Manual**. Disponível em <http://docs.mongodb.org/manual/>. Acesso em: 21 jul. 2014

[9] WAHOO FITNESS. **Tickr x heart rate monitor**. Disponível em <http://www.wahoofitness.com/devices/wahoo-tickr-x-heart-rate-strap>. Acesso em: 23 out. 2016

[10] STRONGLOOP, INC. **Express: Framework web rápido, flexível e minimalista para Node.js**. Disponível em <http://expressjs.com/pt-br/>. Acesso em: 15 ago. 2016

[11] KANTAR WORLDPANEL COMTECH. **Smartphone OS sales market share**. Disponível em <http://www.kantarworldpanel.com/smartphone-os-market-share/>. Acesso em: 10 out. 2016

[12] ORACLE. **What is Java technology and why do I need it?**. Disponível em <https://java.com/en/download/faq/whatis_java.xml>. Acesso em: 23 out. 2016

[13] ARDUINO. **What is Arduino?**. Disponível em <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 23 out. 2016

[14] FITBIT. **Fitbit App**. Disponível em <https://www.fitbit.com/app>. Acesso em: 23 out. 2016

[15] SOARES, J. **O que é MongoDB e porque usá-lo?**. Disponível em <http://codigosimples.net/2016/03/01/o-que-e-mongodb-e-porque-usa-lo/>. Acesso em: 23 out. 2016

[16] RETHINKDB. **RethinkDB Documentation**. Disponível em <https://rethinkdb.com/docs/>. Acesso em: 30 jul 2016.

[17] TECHNOPEDIA INC. **C++ Programming Language**. Disponível em <https://www.techopedia.com/definition/26184/c-programming-language>. Acesso em: 20 nov. 2016

[18] MIT. **Mongoose Docs**. Disponível em <http://mongoosejs.com/docs/guide.html>. Acesso em: 05 nov. 2016

[19] OAUTH. **Oauth**. Disponível em <https://oauth.net/>. Acesso em: 06 nov. 2016

[20] PYTHON SOFTWARE FOUNDATION. **Python Docs**. Disponível em <https://www.python.org/doc/>. Acesso em: 01 de nov. 216

[21] SCIADS. **SCIADS - Sistema Computacional Inteligente de Assistência Domiciliar à Saúde**. Disponível em <https://sites.google.com/site/sistemadetelesaude/>. Acesso em: 25 ago. 2016

[22] TIME INC. **6 Reasons Apple Is So Successful**. Disponível em <http://techland.time.com/2012/05/07/six-reasons-why-apple-is-successful/>. Acesso em: 10 nov. 2016

[23] BAUER V. **Android developer portal with tools, libraries, and apps**. Disponível em <https://android-arsenal.com/tag/40>. Acesso em 10 set. 2016

[24] LIBELIUM. **MySignals**. Disponível em <http://www.my-signals.com/>. Acesso em: 8 de out. 2016

[25] NODEJS FOUNDATION. **Nodejs**. Disponível em <https://nodejs.org/en/>. Acesso em: 25 out. 2016

[26] JAHODA P. **MPAndroid Chart**. Disponível em <https://github.com/PhilJay/MPAndroidChart>. Acesso em: 15 de nov. 2016

[27] DUNCAN, G. **You can't avoid the 'Internet of things' hype, so you might as well understand it**. Disponível em: <http://www.digitaltrends.com/home/heck-internet-things-dont-yet/>. Acesso em: 29 nov. 2016.

[28] MOBILE NATIONS. **Android History.** Disponível em: <http://www.androidcentral.com/android-history>. Acesso em: 20 nov. 2016.

[29] ANDROID STUDIO. **Update the IDE and SDK Tools.** Disponível em: <https://developer.android.com/studio/intro/update.html>. Acesso em: 20 nov. 2016.

[30] ROUSE M. **XML File Format.** Disponível em: <http://whatis.techtarget.com/fileformat/XML-eXtensible-markup-language>. Acesso em: 20 nov. 2016.

[31] GIT. **Getting Started - Git Basics.** Disponível em: <https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>. Acesso em: 20 nov. 2016.

[32] HOW-TO GEEK. **What is Github, and what is it used for?** Disponível em: <http://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/>. Acesso em: 20 nov. 2016.

[33] GOOGLE DEVELOPERS. **Google Cloud Messaging.** Disponível em: <https://developers.google.com/cloud-messaging/>. Acesso em: 20 nov. 2016.

[34] RICHARDSON C. **Pattern: Microservices Architecture.** Disponível em: <http://microservices.io/patterns/microservices.html>. Acesso em: 20 nov. 2016.

[35] ROCHELEAU J. **Responsive Web Layouts for Mobile Screens: Intro, Tips and Examples.** Disponível em: <http://www.hongkiat.com/blog/responsive-for-mobile-screens/>. Acesso em: 20 nov. 2016.

[36] ORACLE. **Client/Server Architecture.** Disponível em: <https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch20.htm>. Acesso em: 20 nov. 2016.

[37] ORACLE. **What is Middleware.** Disponível em: <https://docs.oracle.com/cd/E21764_01/core.1111/e10103/intro.htm>. Acesso em: 20 nov. 2016.

[38] POSTMAN. **Postman - Modern software is built on APIs**. Disponível em: <https://www.getpostman.com/>. Acesso em: 27 nov. 2016.

# Apêndice A

## 1. Casos de Uso do Sistema

Neste apêndice encontra-se a descrição detalhada dos casos de uso do sistema desenvolvido, além dos diagramas de sequência associados a cada caso de uso.

**CASO DE USO 01 -** Editar dados pessoais

**Descrição:** Editar dados pessoais do perfil, de acordo com usuário (ex. Usuário edita sua data de nascimento, médico edita seu CRM).

**Ator:** Usuário

**Evento Iniciador:** Solicitação para editar dados pessoais

**Pré-condição:** Usuário autenticado no sistema, informações pessoais do usuário exibidas.

**Sequência de Eventos:**

1. Sistema exibe todos os campos de informação pessoal para alteração.
2. Usuário realiza as alterações desejadas e clica no botão para salvar alterações.
3. Sistema solicita confirmação das alterações.
4. Usuário confirma alterações.
5. Sistema atualiza informações e exibe uma mensagem de sucesso da operação.

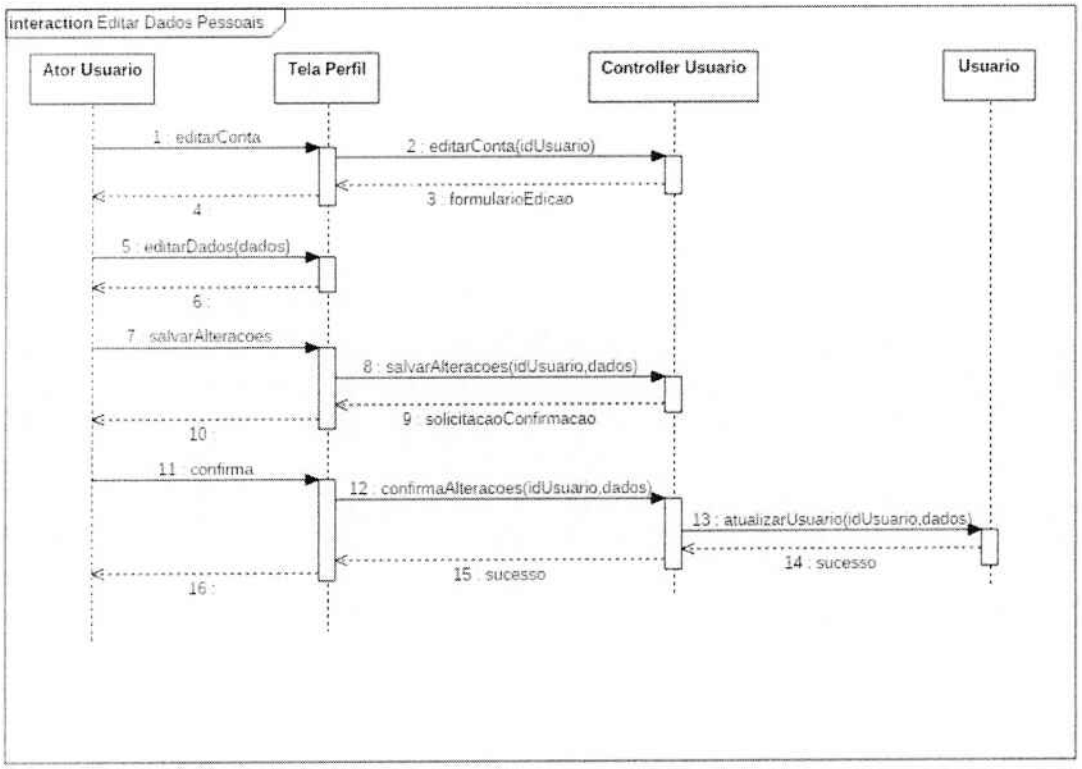
**Pós-Condição:** Dados do usuário alterados no sistema e dados pessoais do usuário exibidos na tela.

**Extensões:**

1. Dados inválidos (passo 3) - Sistema exibe mensagem de erro e informa campos com dados inválidos.
2. Usuário cancela alterações (passo 4) - Usuário cancela alterações, sistema retorna à tela de alterações (passo 1).

**Inclusões:** -

*Figura 22 - Caso de uso: Editar Dados Pessoais*



**CASO DE USO 02 -** *Login* de usuário

**Descrição:** Autenticar o usuário no sistema através de dados de acesso.

**Ator:** Usuário

**Evento Iniciador:** Solicitação de acesso ao sistema.

**Pré-condição:** -

**Sequência de Eventos:**

1. Sistema exibe tela de *login* solicitando dados de acesso.
2. Usuário preenche campos com dados de acesso e clica no botão de *login*.
3. Sistema identifica que o ator em questão já está registrado no sistema e o redireciona para a tela inicial do usuário no sistema.

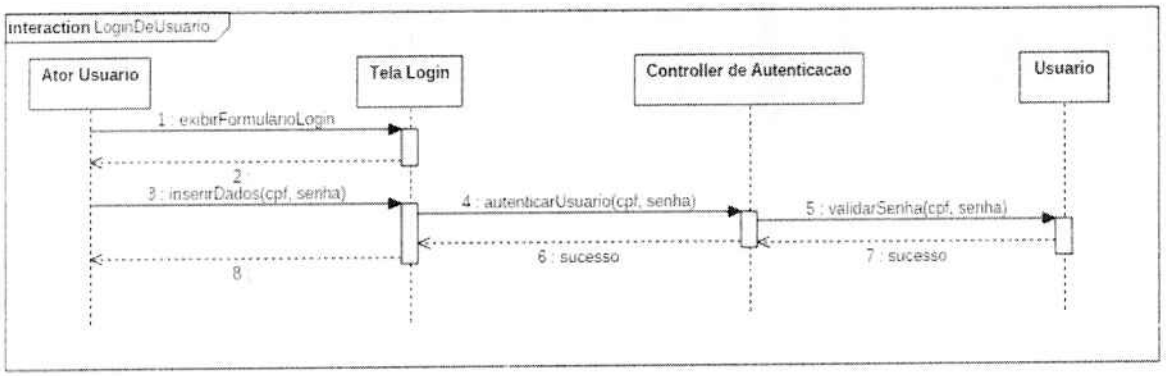
**Pós-Condição:** Usuário autenticado no sistema e tela inicial do ator exibida.

**Extensões:**

1. Sistema identifica que os dados preenchidos pelo Usuário não constam no sistema (passo 3) - Apresenta mensagem de erro e retorna ao passo 1.

**Inclusões:** -

*Figura 23 - Caso de uso: Login de Usuario*



**CASO DE USO 03 -** Listar solicitações pendentes

**Descrição:** Listar as solicitações enviadas (Acompanhante, Médico) ou recebidas (Paciente).

**Ator:** Usuário

**Evento Iniciador:** Solicitação da lista de solicitações pendentes.

**Pré-condição:** Usuário autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe a lista de solicitações pendentes relacionadas ao Usuário.

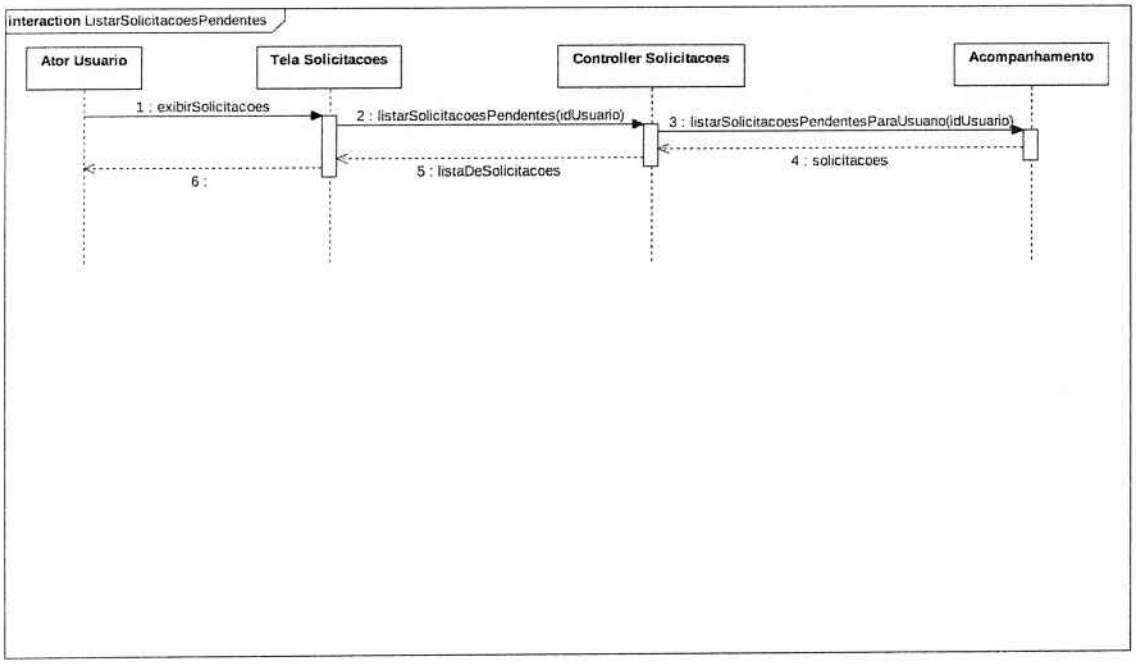
**Pós-Condição:** Lista de solicitações pendentes exibida.

**Extensões:**

1. Lista vazia (passo 1) - Sistema exibe mensagem de lista vazia e encerra caso de uso.

**Inclusões:** -

*Figura 24 - Caso de uso: Lista Solicitações Pendentes*



**CASO DE USO 04 -** Cancelar solicitação pendente

**Descrição:** Cancelar uma solicitação enviada (Médico ou Acompanhante) ou recebida (Paciente).

**Ator:** Usuário.

**Evento Iniciador:** Solicitação de cancelamento através do botão ao lado da solicitação desejada.

**Pré-condição:** Usuário autenticado no sistema e lista de solicitações pendentes exibida.

**Sequência de Eventos:**

1. Sistema solicita confirmação do cancelamento.
2. Usuário confirma cancelamento.
3. Sistema exibe mensagem de sucesso, exclui a solicitação e envia notificação de cancelamento da solicitação ao usuário que a enviou (Médico ou Acompanhante) ou recebeu (Paciente).

**Pós-Condição:** Solicitação excluída do sistema e da lista de solicitações pendentes exibida.

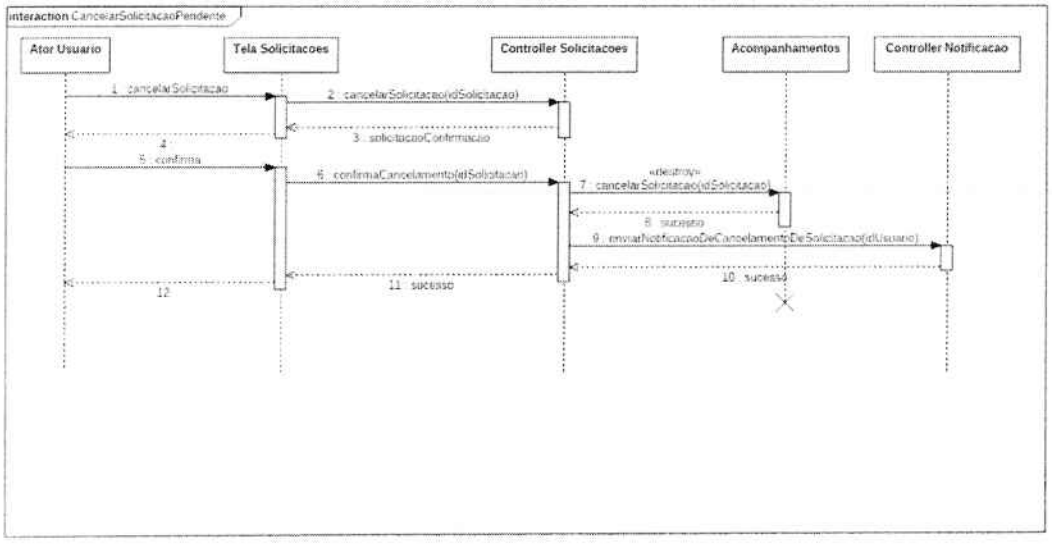
**Extensões:**

1. Usuário não confirma cancelamento (passo 2) - Sistema encerra caso de uso.

**Inclusões:**

1. Caso de uso 22 - Enviar notificação (passo 3)

Figura 25 - Caso de Uso: Cancelar Solicitação Pendente



**CASO DE USO 05 -** Visualizar medidas do paciente

**Descrição:** Exibir os dados coletados do paciente, através de valores médios e/ou gráficos.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para exibir medidas de um paciente.

**Pré-condição:** Usuário autenticado no sistema e tela de detalhes de um paciente exibida.

**Sequência de Eventos:**

1. Sistema agrupa dados coletados e calculados e exibe os dados agrupados.

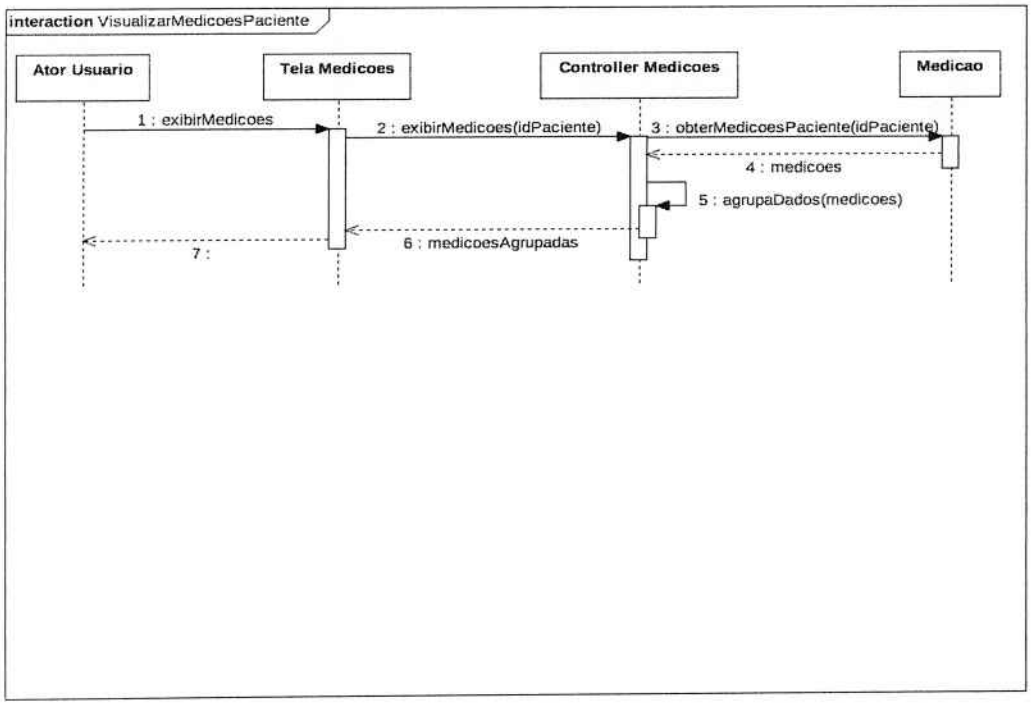
**Pós-Condição:** Dados agrupados de um paciente exibidos.

**Extensões:**

1. Paciente sem dados (passo 1) - Sistema apresenta mensagem de paciente sem dados e encerra o caso de uso.

**Inclusões:** -

*Figura 26 - Caso de uso: Visualizar Medições Paciente*



**CASO DE USO 06 -** Iniciar conversa

**Descrição:** Iniciar uma conversa com um usuário vinculado.

**Ator:** Usuário.

**Evento Iniciador:** Solicitação para iniciar conversa

**Pré-condição:** Usuário autenticado no sistema.

**Sequência de Eventos:**

1. Sistema lista usuários vinculados ao Usuário.
2. Usuário seleciona um usuário vinculado para iniciar conversa.
3. Sistema cria uma nova conversa e abre uma janela de conversa para que o Usuário escreva a mensagem que desejar para o usuário vinculado escolhido.
4. Usuário escreve a mensagern.

**Pós-Condição:** Janela de conversa entre o Usuário e o usuário vinculado aberta e uma mensagem escrita na janela.

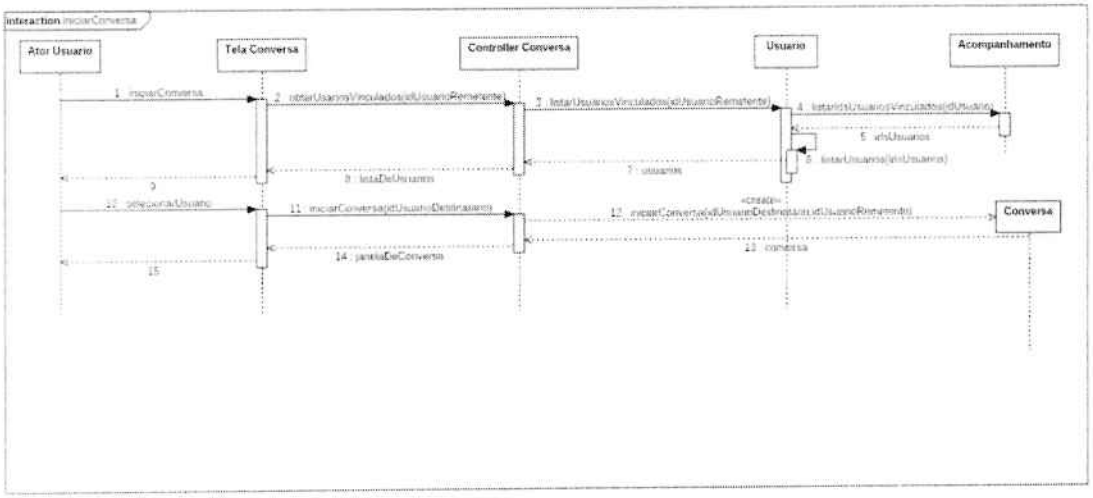
**Extensões:**

1. Já existe uma conversa entre os dois usuários (passo 3) - Sistema exibe a janela de conversa entre os dois.

**Inclusões:**

1. Caso de uso 09 - Visualizar conversa (passo 3).
2. Caso de uso 10 - Listar usuários vinculados (passo 1)

*Figura 27 - Caso de uso: Iniciar Conversa*



**CASO DE USO 07 -** Enviar mensagem

**Descrição:** Envia uma mensagem a um usuário vinculado.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para enviar mensagem.

**Pré-condição:** Usuário autenticado no sistema e janela de conversa entre o Usuário e usuário vinculado aberta e com mensagem escrita.

**Sequência de Eventos:**

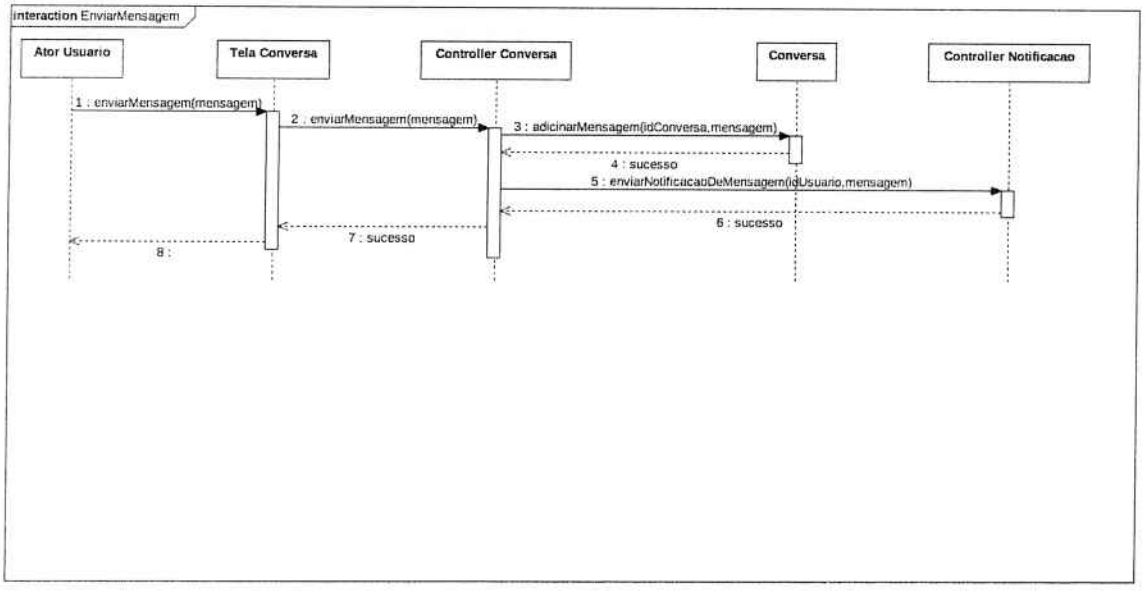
1. Sistema adiciona mensagem à conversa e envia notificação ao outro usuário.

**Pós-Condição:** Sistema exibindo janela de conversa entre o ator e outro usuário, atualizada com a mensagem enviada.

**Extensões:** -

**Inclusões:**
Caso de uso 22 - Enviar notificação

*Figura 28 - Caso de uso: Enviar Mensagem*



**CASO DE USO 08 -** Listar conversas

**Descrição:** Listar conversas iniciadas.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para listar conversas.

**Pré-condição:** Usuário autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe uma lista com o resumo das conversas em que o Usuário participa.

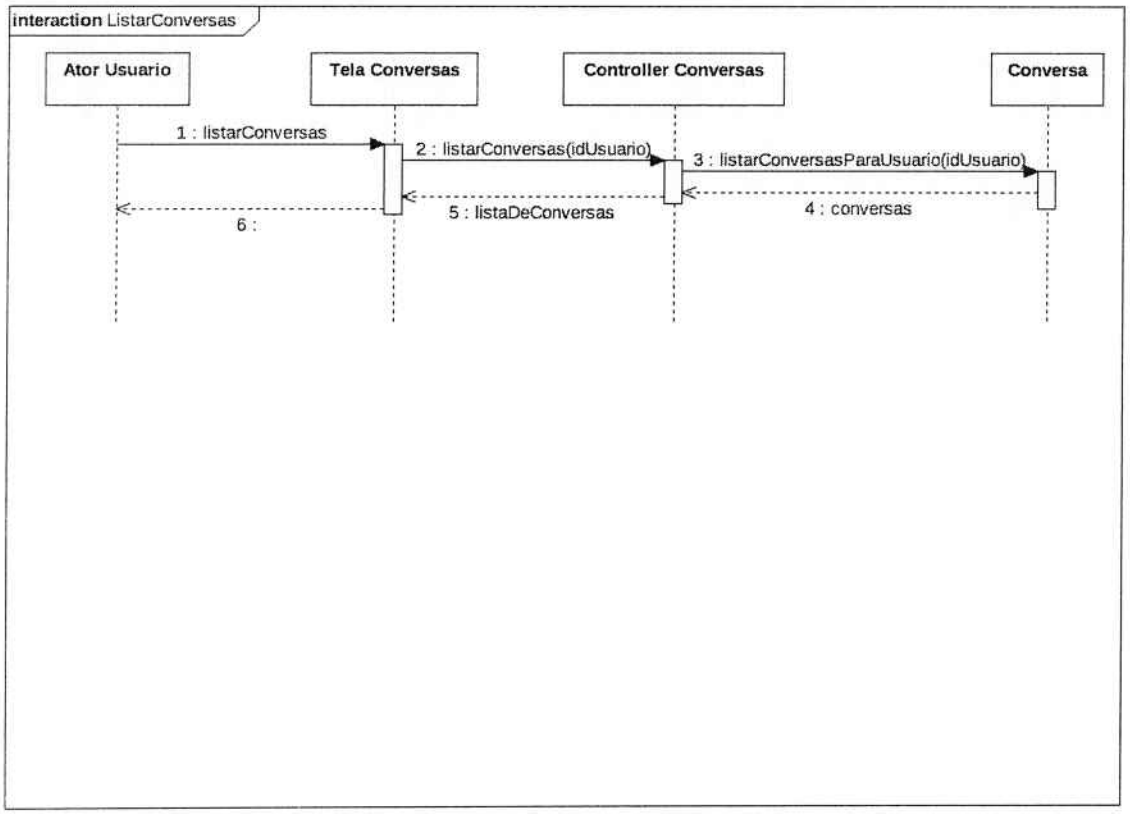
**Pós-Condição:** Resumo das conversas em andamento apresentado.

**Extensões:**

1. Não existe conversas em andamento (passo 1) - Sistema apresenta mensagem correspondente e encerra o caso de uso

**Inclusões:** -

*Figura 29 - Caso de uso: Listar Conversas*



**CASO DE USO 09 -** Visualizar conversa

**Descrição:** Visualizar histórico da conversa com outro usuário vinculado.

**Ator:** Usuário

**Evento Iniciador:** Acionamento de uma conversa da sua lista de conversas.

**Pré-condição:** Usuário autenticado no sistema e lista de conversas do Usuário exibida.

**Sequência de Eventos:**

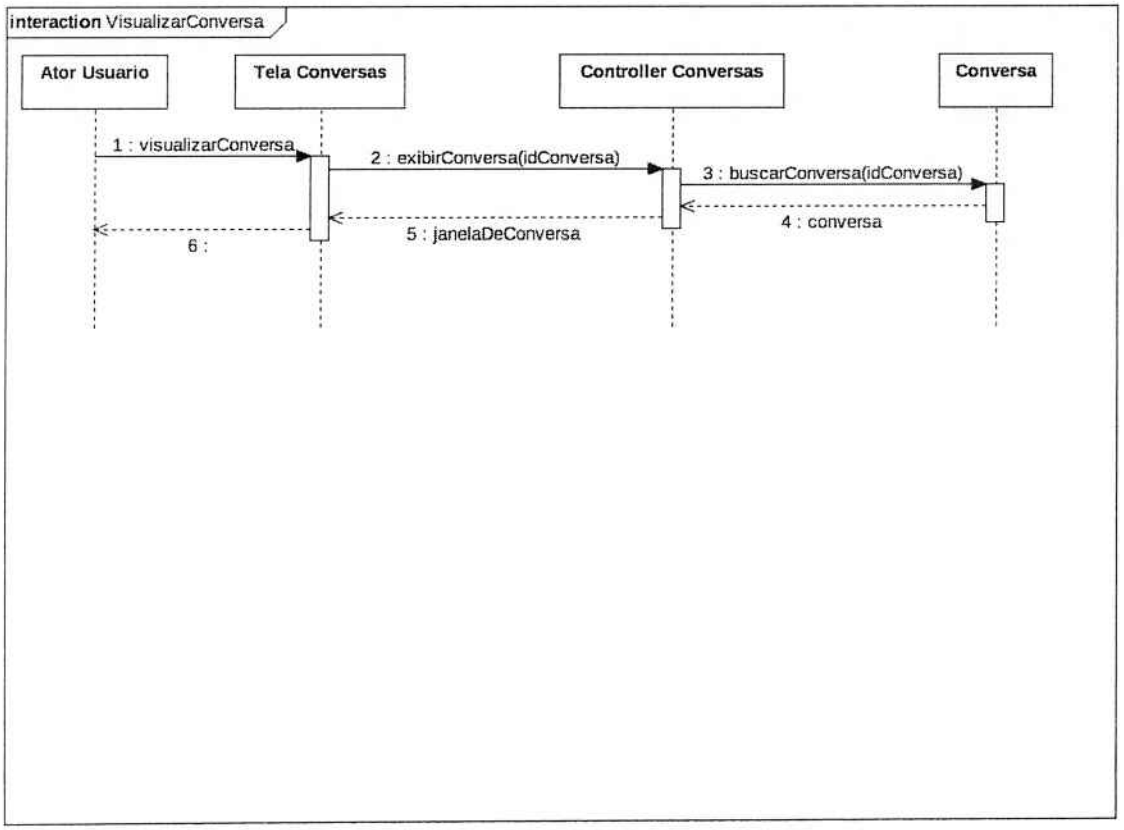
1. Sistema abre a janela da conversa, exibindo o histórico da conversa selecionada.

**Pós-Condição:** Janela de conversa com o histórico da conversa selecionada exibida.

**Extensões:** -

**Inclusões:** -

*Figura 30 - Caso de uso: Visualizar Conversa*



**CASO DE USO 10 -** Listar usuários vinculados

**Descrição:** Listar usuários vinculados ao usuário selecionado.

**Ator:** Usuário

**Evento Iniciador:** Solicitação da lista de usuários vinculados.

**Pré-condição:** Usuário autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe lista de usuários com acompanhamento confirmado com o Usuário.

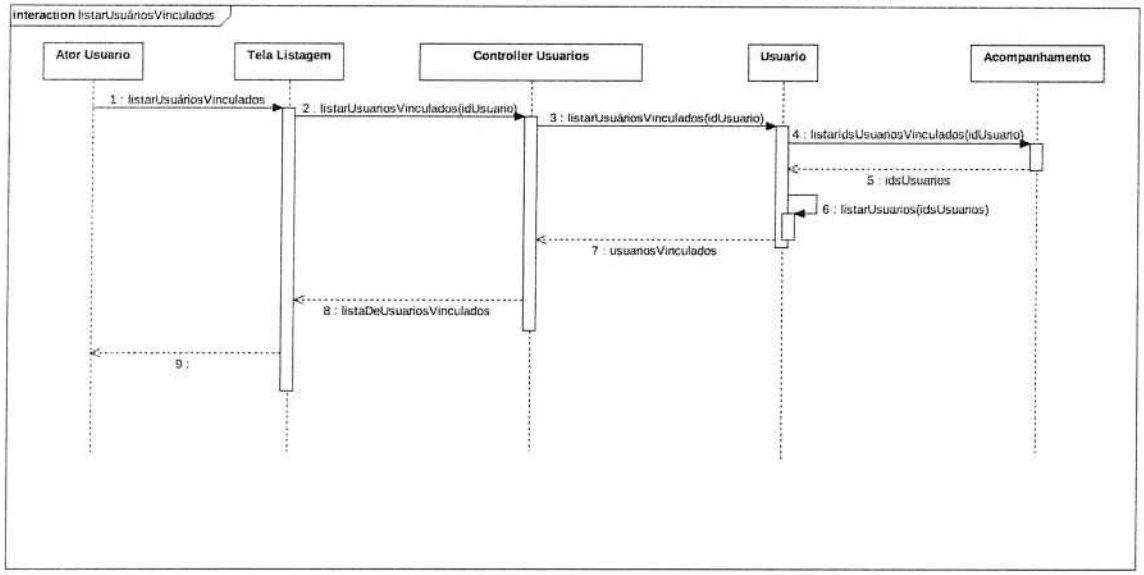
**Pós-Condição:** Lista de usuários vinculados ao Usuário exibida.

**Extensões:**

1. Não existe usuários vinculados ao Usuário (passo 1) - Sistema apresenta mensagem correspondente e encerra o caso de uso.

**Inclusões:** -

*Figura 31 - Caso de uso: Listar Usuários Vinculados*



**CASO DE USO 11 -** Listar notificações

**Descrição:** Listar notificações recebidas.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para listar notificações.

**Pré-condição:** Usuário autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe lista com as notificações recebidas pelo Usuário.

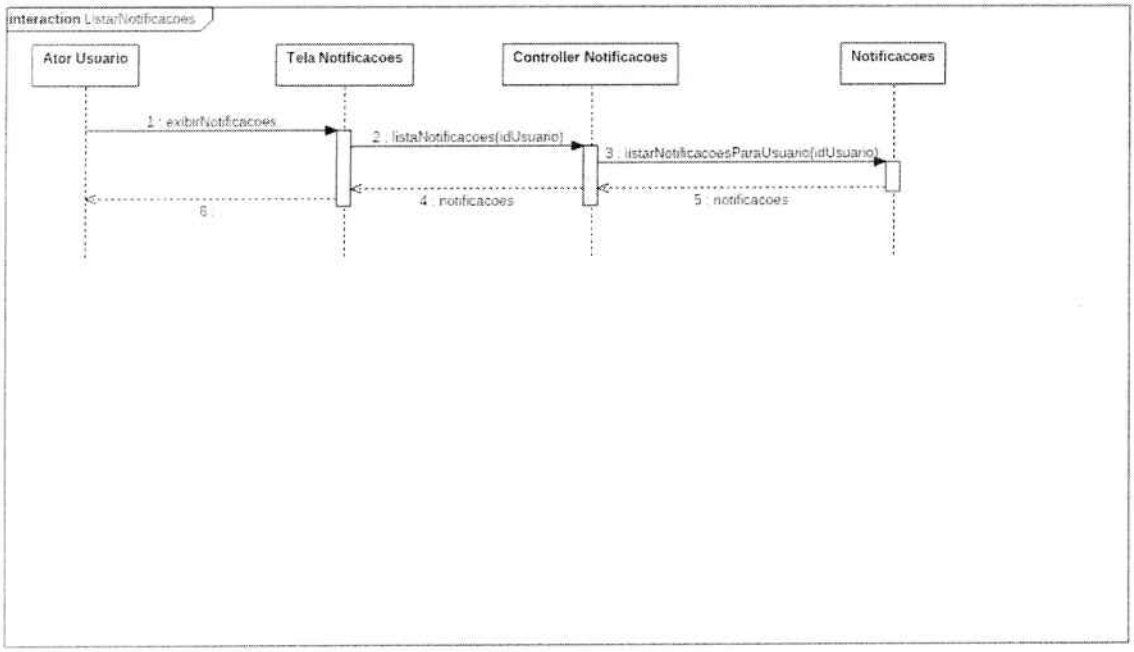
**Pós-Condição:** Lista com as notificações recebidas pelo Usuário exibida.

**Extensões:**

1. Não existe notificação para o Usuário (passo 1) - Sistema apresenta mensagem correspondente e encerra caso de uso.

**Inclusões:** -

*Figura 32 - Caso de uso: Listar Notificações*



**CASO DE USO 12 -** Cadastrar usuário

**Descrição:** Cadastro de usuário, de acordo com seu papel.

**Ator:** Usuário

**Evento Iniciador:** Solicitação de cadastro de usuário.

**Pré-condição:** -

**Sequência de Eventos:**

1. Sistema exibe tela para seleção de tipo usuário a ser criado.
2. Ator seleciona tipo de usuário desejado.
3. Sistema exibe formulário para preenchimento de dados pessoais e de acesso, de acordo com o tipo de usuário selecionado.
4. Ator preenche com os dados e clica no botão para criar usuário.
5. Sistema solicita confirmação.
6. Ator confirma criação.
7. Sistema cria usuário e realiza *login* do novo usuário.

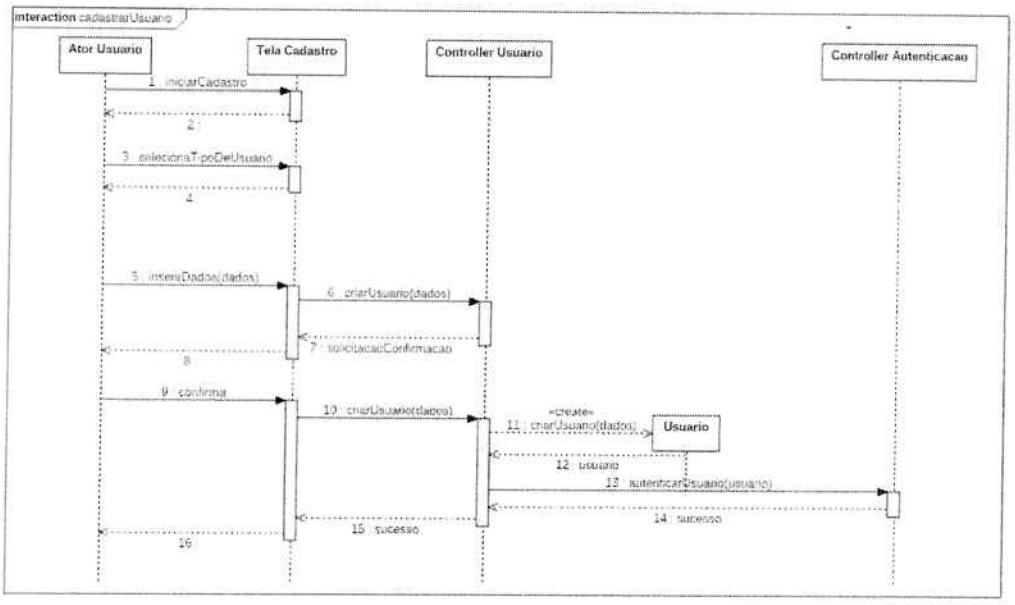
**Pós-Condição:** Usuário cadastrado no sistema e tela inicial do usuário exibida.

**Extensões:**

1. Dados com erro (passo 5) - Sistema exibe mensagem de erro, informa campos com dados inválidos e vai para o passo 4.
2. Ator não confirma criação (passo 6) - Sistema exibe o formulário preenchido e vai para o passo 4.

**Inclusões:** -

*Figura 33 - Caso de uso: Cadastrar Usuário*



**CASO DE USO 13 -** Aceitar solicitação de acompanhamento

**Descrição:** Permitir acesso às informações do paciente.

**Ator:** Paciente

**Evento Iniciador:** Solicitação para registrar solicitação de acompanhamento.

**Pré-condição:** Paciente autenticado no sistema e lista de solicitações pendentes exibida.

**Sequência de Eventos:**

1. Sistema solicita confirmação.
2. Paciente confirma a aceitação da solicitação.
3. Sistema exibe mensagem de sucesso, registra o aceite do acompanhamento e envia notificação de aceitação do acompanhamento ao usuário que solicitou.

**Pós-Condição:** Aceite do acompanhamento registrado, lista de solicitações pendentes exibida.

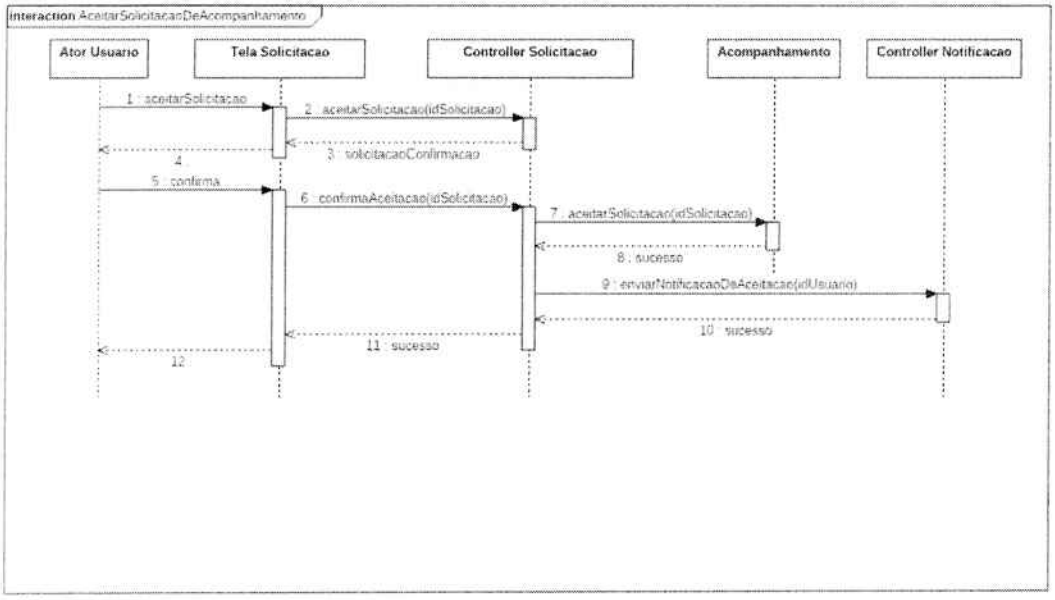
**Extensões:**

1. Usuário não confirma aceitação (passo 1) - Sistema encerra o caso de uso

**Inclusões:**

1. Caso de uso 22 - Enviar notificação (passo 3).

*Figura 34 - Caso de uso: Aceitar Solicitação de Acompanhamento*



**CASO DE USO 14 -** Solicitar acompanhamento

**Descrição:** Solicitar acesso às informações de um paciente pelo médico ou acompanhante.

**Atores:** Médico ou Acompanhante

**Evento Iniciador:** Solicitação de acompanhamento.

**Pré-condição:** Ator autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe campo para busca de pacientes.
2. Ator preenche campo de busca e clica em buscar.
3. Sistema exibe lista de pacientes filtrados pelo campo de busca.
4. Ator seleciona paciente.
5. Sistema solicita confirmação.
6. Ator confirma solicitação.
7. Sistema exibe mensagem de sucesso, registra solicitação e envia notificação de solicitação de acompanhamento ao paciente.

**Pós-Condição:** Solicitação de acompanhamento registrada, mensagem de sucesso apresentada e tela inicial do ator exibida.

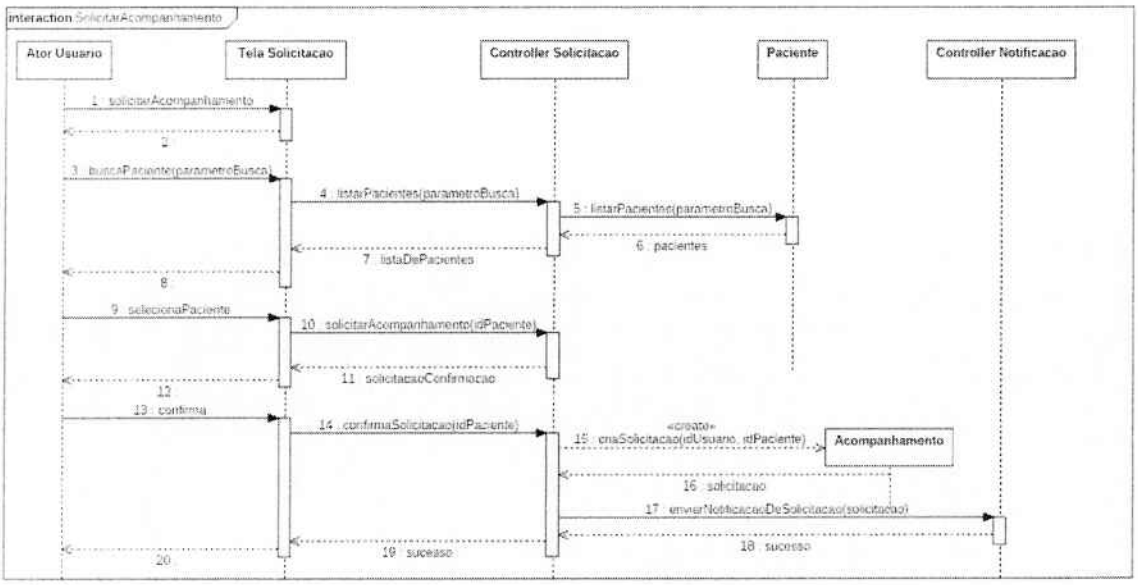
**Extensões:**

1. Campo de busca não preenchido corretamente (passo 3) - Sistema exibe mensagem de erro e retorna ao passo 1.
2. Ator não confirma solicitação (passo 6) - Sistema volta à listagem de pacientes filtrados (passo 3).

**Inclusões:**

1. Caso de uso 17 - Listar paciente (passo 3).

*Figura 35 - Caso de uso: Solicitar Acompanhamento*



**CASO DE USO 15 -** Cancelar acompanhamento

**Descrição:** Cancelar o acompanhamento de um paciente.

**Atores:** Médico ou Acompanhante

**Evento Iniciador:** Solicitação para cancelar acompanhamento.

**Pré-condição:** Ator autenticado no sistema, lista de usuários vinculados exibida.

**Sequência de Eventos:**

1. Sistema solicita confirmação do cancelamento.
2. Ator confirma cancelamento.
3. Sistema cancela acompanhamento e envia notificação de cancelamento ao paciente.

**Pós-Condição:** Acompanhamento cancelado e lista de usuários vinculados exibida.

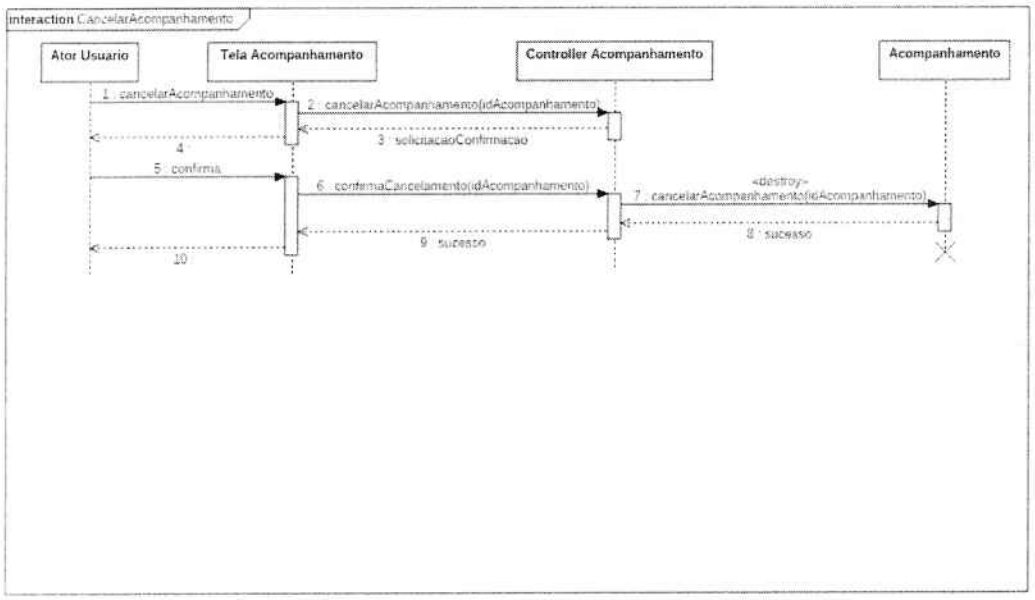
**Extensões:**

1. Ator não confirma cancelamento (passo 2) - Sistema encerra caso de uso.

**Inclusões:**

1. Caso de uso 22 - Enviar notificação (passo 3).

*Figura 36 - Caso de uso: Cancelar Acompanhamento*



**CASO DE USO 16 -** Listar alertas

**Descrição:** Listar alertas de saúde recebidos.

**Atores:** Médico ou Acompanhante

**Evento Iniciador:** Solicitação para listar alertas enviados pelo sistema.

**Pré-condição:** Ator autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe a lista de alertas referentes ao paciente ou pacientes que o ator esteja acompanhando.

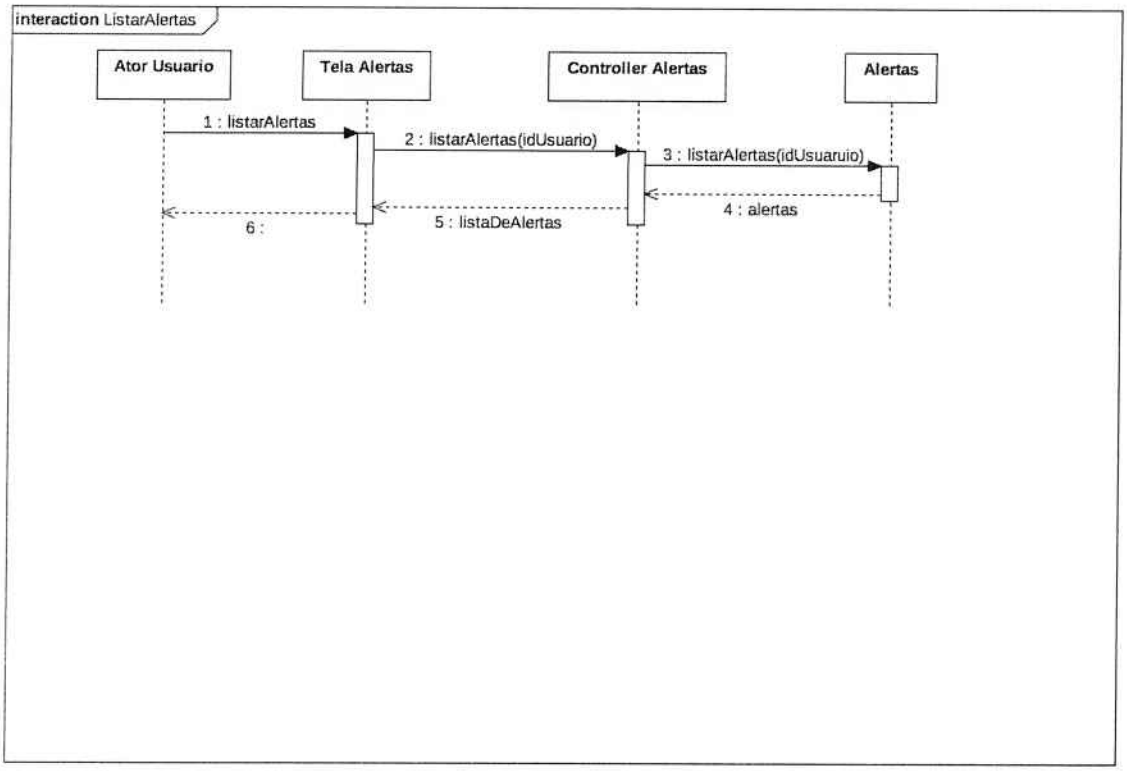
**Pós-Condição:** Lista de alertas exibida pelo sistema.

**Extensões:**

1. Não existem alertas do sistema (passo 1) - Sistema apresenta a mensagem correspondente e encerra o caso de uso.

**Inclusões:** -

*Figura 37 - Caso de uso: Listar Alertas*



**CASO DE USO 17 -** Listar pacientes

**Descrição:** Listar pacientes registrados no sistema através do nome/CPF.

**Atores:** Médico ou Acompanhante

**Evento Iniciador:** Solicitação para listar pacientes.

**Pré-condição:** Ator autenticado no sistema.

**Sequência de Eventos:**

1. Sistema exibe tela de busca com campos de filtro para serem preenchidos
2. Ator preenche o(s) campo(s) com informações do paciente desejado e clica no botão de busca.
3. Sistema apresenta a lista de pacientes encontrados a partir dos dados informados

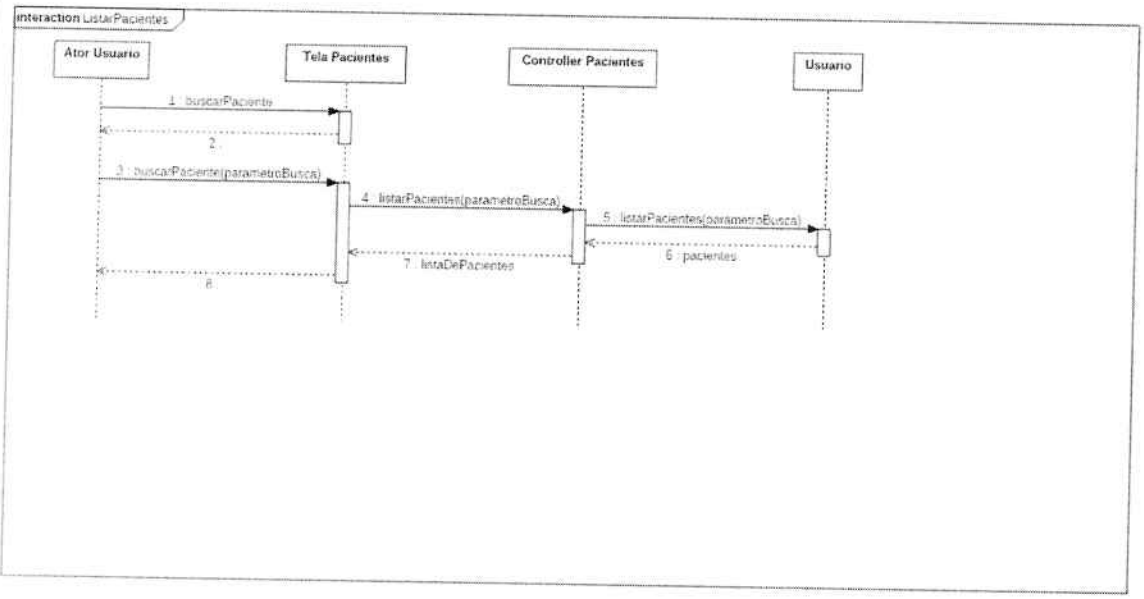
**Pós-Condição:** Lista de pacientes encontrados exibida.

**Extensões:**

1. Nenhum paciente encontrado (passo 3) - Sistema exibe mensagem correspondente e vai para o passo 1.

**Inclusões:** -

*Figura 38 - Caso de uso: Listar Pacientes*



**CASO DE USO 18 -** Criar prontuário

**Descrição:** Criar prontuário médico de um paciente.

**Ator:** Médico

**Evento Iniciador:** Solicitação para criar o prontuário médico do paciente.

**Pré-condição:** Médico autenticado no sistema e tela de detalhes de um paciente exibida.

**Sequência de Eventos:**

1. Sistema exibe tela de criação de prontuário, com campos a serem preenchidos pelo ator.
2. Médico preenche os campos com dados sobre o paciente e clica no botão salvar.
3. Sistema solicita confirmação.
4. Médico confirma criação.
5. Sistema cria prontuário.

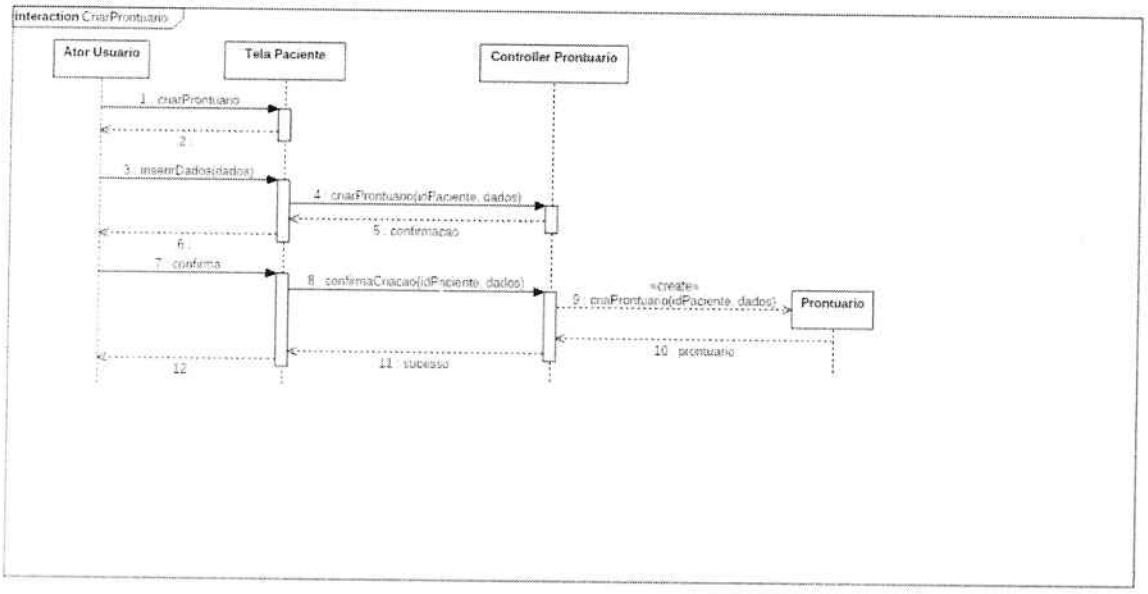
**Pós-Condição:** Tela de detalhe do paciente exibida, com botão para o prontuário médico.

**Extensões:**

1. Médico não confirma criação (passo 4) - Sistema vai para o passo 1.

**Inclusões:** -

*Figura 39 - Caso de Uso: Criar Prontuário*



**CASO DE USO 19 -** Editar prontuário

**Descrição:** Editar prontuário médico de um paciente.

**Ator:** Médico

**Evento Iniciador:** Solicitação para editar o prontuário médico do paciente.

**Pré-condição:** Médico autenticado no sistema e prontuário de um paciente exibido.

**Sequência de Eventos:**

1. Sistema exibe tela de edição de prontuário, com campos que podem ser alterados.
2. Médico realiza alterações desejadas nos campos do prontuário e clica no botão para salvar.
3. Sistema solicita confirmação.
4. Médico confirma alterações.
5. Sistema atualiza prontuário.

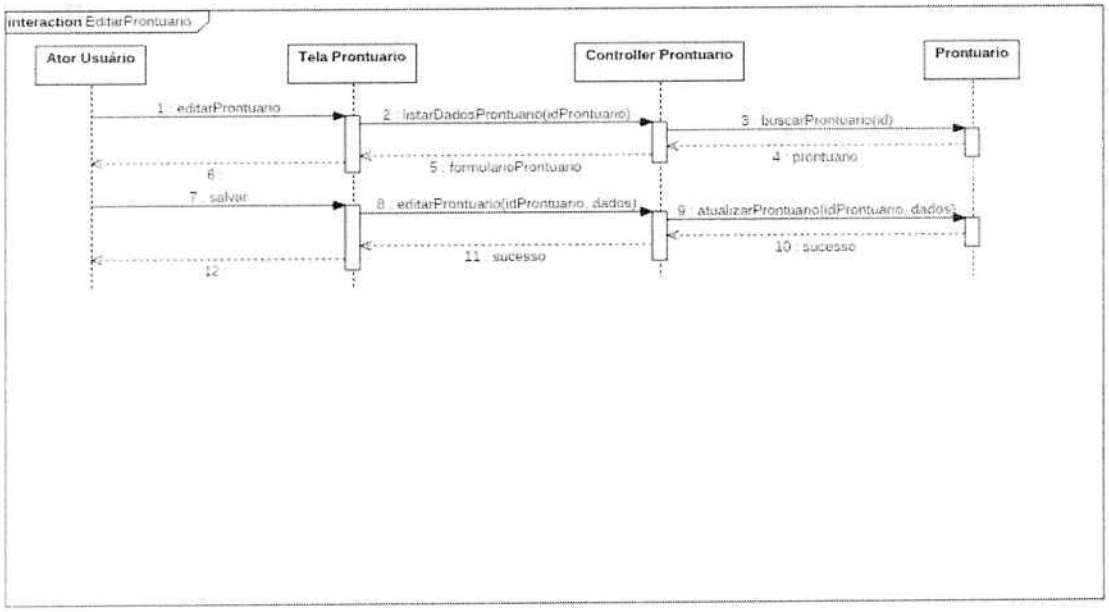
**Pós-Condição:** Tela de detalhe do paciente exibida, com botão para o prontuário médico.

**Extensões:**

1. Ator não confirma alterações (passo 4) - Sistema retorna à tela de edição (passo 1).

**Inclusões:** -

*Figura 40 - Caso de uso: Editar Prontuário*



**CASO DE USO 20 -** Visualizar prontuário

**Descrição:** Visualizar o prontuário médico de um paciente.

**Ator:** Médico

**Evento Iniciador:** Solicitação para visualizar o prontuário médico do paciente.

**Pré-condição:** Médico autenticado no sistema e tela de detalhes de um paciente exibida.

**Sequência de Eventos:**

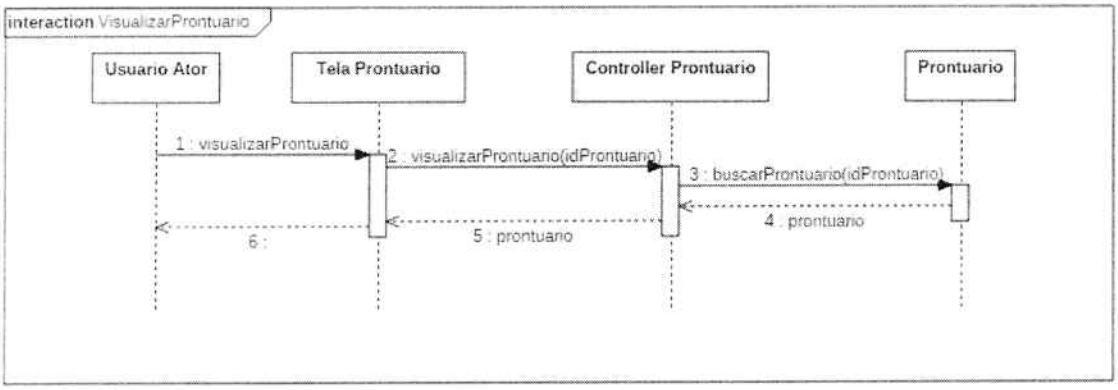
1. Sistema exibe prontuário médico do paciente.

**Pós-Condição:** Prontuário médico do paciente exibido.

**Extensões:** -

**Inclusões:** -

*Figura 41 - Caso de uso: Visualizar Prontuário*



**CASO DE USO 21 -** Analisar dados recebidos

**Descrição:** Classificar, tratar e analisar medidas recebidas de um paciente, para identificar possíveis situações que requeiram atenção do médico e/ou acompanhante.

**Ator:** Concentrador de dados

**Evento Iniciador:** Recepção de pacote de dados enviado pelo concentrador.

**Pré-condição:** -

**Sequência de Eventos:**

1. Sistema desempacota os dados recebidos.
2. Sistema aplica algoritmo sobre os dados.
3. Sistema armazena os dados.

**Pós-Condição:** Dados do paciente atualizados.

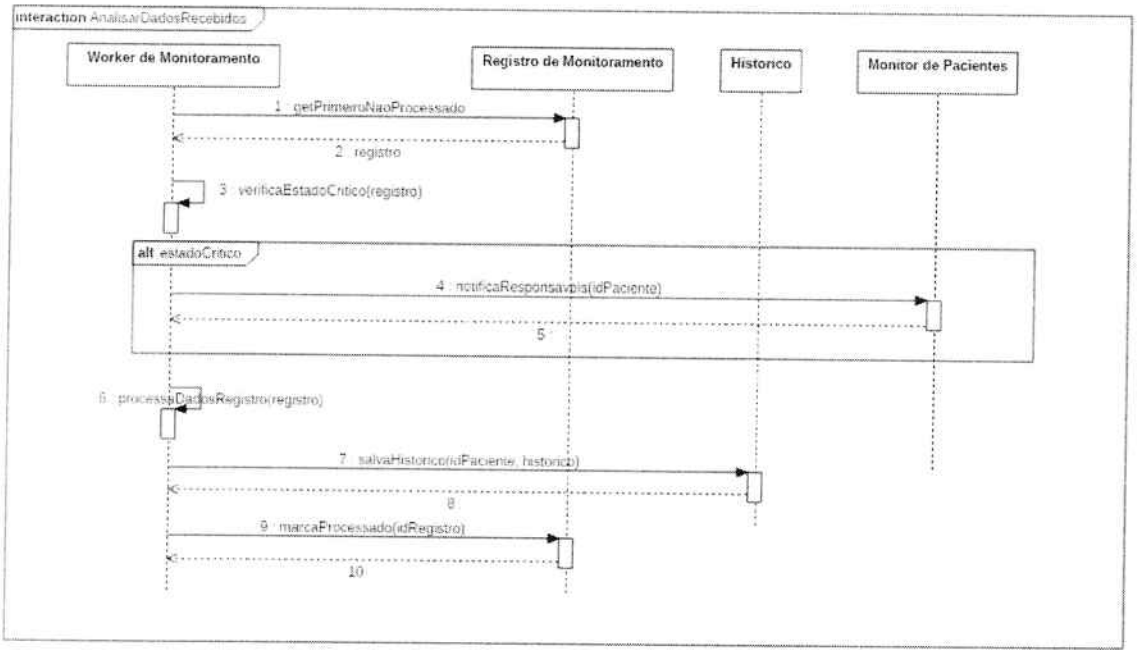
**Extensões:**

1. Algoritmo de análise detecta uma condição que requeira atenção (passo 2) - Sistema envia alerta aos usuários vinculados ao paciente monitorado.

**Inclusões:**

1. Caso de uso 23 - Enviar alerta (extensão 1).

*Figura 42 - Caso de uso: Analisar Dados Recebidos*



**CASO DE USO 22 -** Enviar notificação

**Descrição:** Enviar notificações relacionadas a atividades de outros usuários no sistema.

**Ator:** -

**Evento Iniciador:** Acionamento pelos casos de uso 04, 07, 13, 14 e 15.

**Pré-condição:** -

**Sequência de Eventos:**

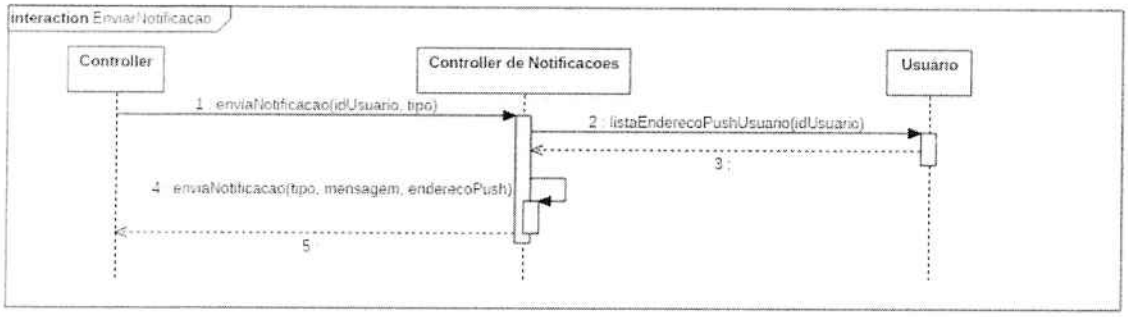
1. Sistema verifica tipo de notificação de acordo com o caso de uso que a solicitou e a envia ao usuário de destino.

**Pós-Condição:** Notificação enviada ao usuário.

**Extensões:** -

**Inclusões:** -

*Figura 43 - Caso de uso: Enviar Notificação*



**CASO DE USO 23 -** Enviar alerta

**Descrição:** Enviar alertas de saúde para médicos e acompanhantes.

**Atores:** -

**Evento Iniciador:** Acionamento pelo caso de uso 21.

**Pré-condição:** -

**Sequência de Eventos:**

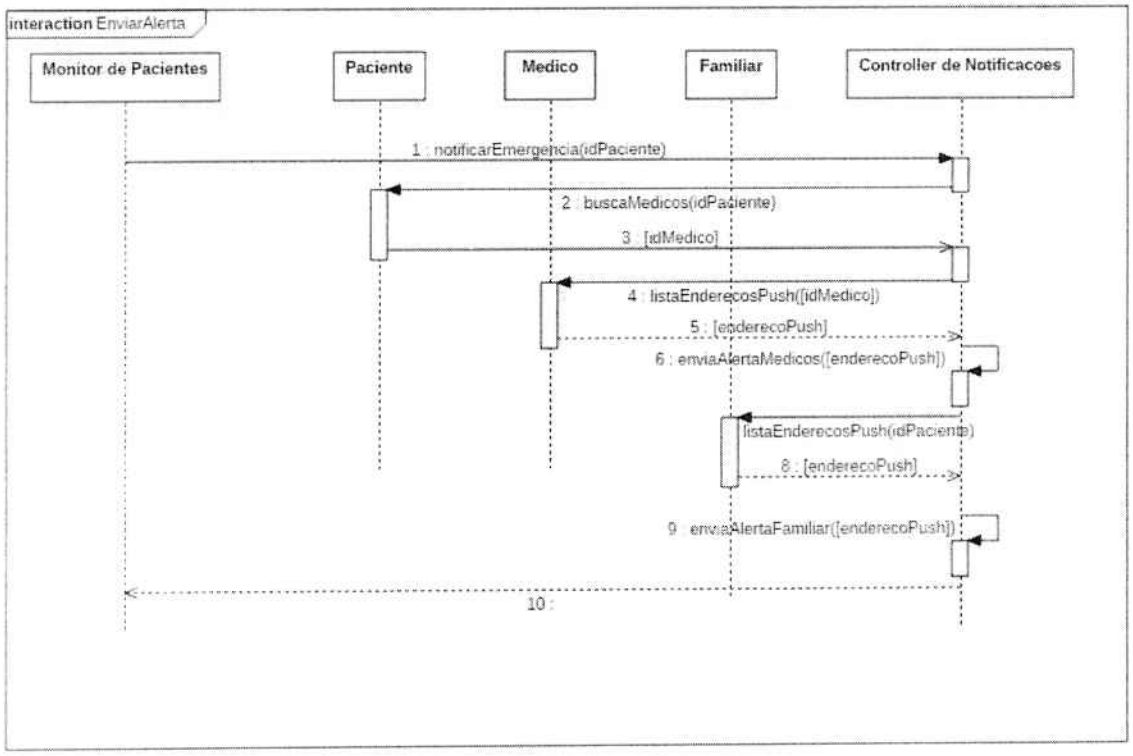
1. Sistema verifica tipo de alerta de acordo com a condição encontrada envia alerta aos usuários vinculados ao paciente monitorado.

**Pós-Condição:** Alerta enviada aos usuários vinculados.

**Extensões:** -

**Inclusões:** -

*Figura 44 - Caso de uso: Enviar Alerta*



**CASO DE USO 24 -** Visualizar usuário

**Descrição:** Exibir dados pessoais de um usuário.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para exibir dados de usuário.

**Pré-condição:** Usuário autenticado no sistema, lista de usuários exibida.

**Sequência de Eventos:**

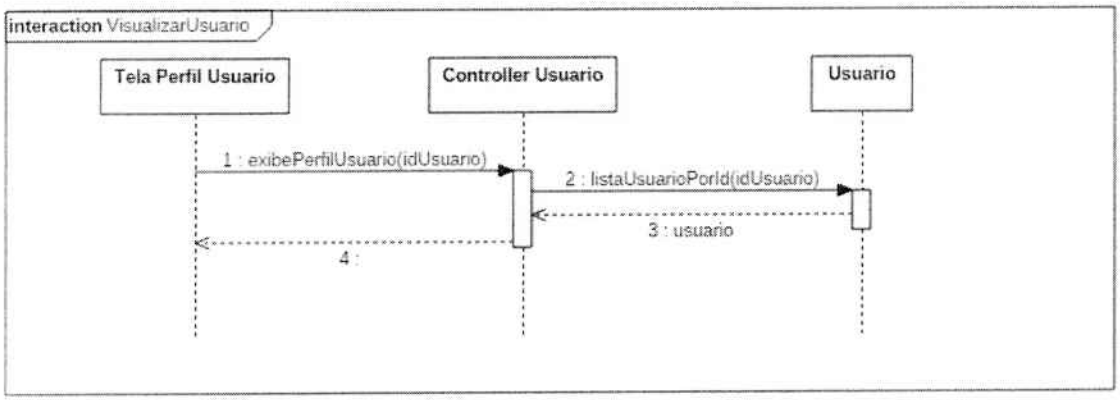
1. Sistema exibe dados pessoais do usuário.

**Pós-Condição:** Tela com dados pessoais do usuário exibida.

**Extensões:** -

**Inclusões:** -

*Figura 45 - Caso de uso: Visualizar Usuário*



**CASO DE USO 25 -** Desativar usuário

**Descrição:** Desativar o acesso do usuário ao sistema e o acesso aos seus dados por outros usuários.

**Ator:** Usuário

**Evento Iniciador:** Solicitação para desativar usuário.

**Pré-condição:** Usuário autenticado e tela de detalhe do Usuário exibida.

**Sequência de Eventos:**

1. Sistema solicita confirmação do desativamento.
2. Usuário confirma desativamento.
3. Sistema atualiza estado do Usuário para desativado.

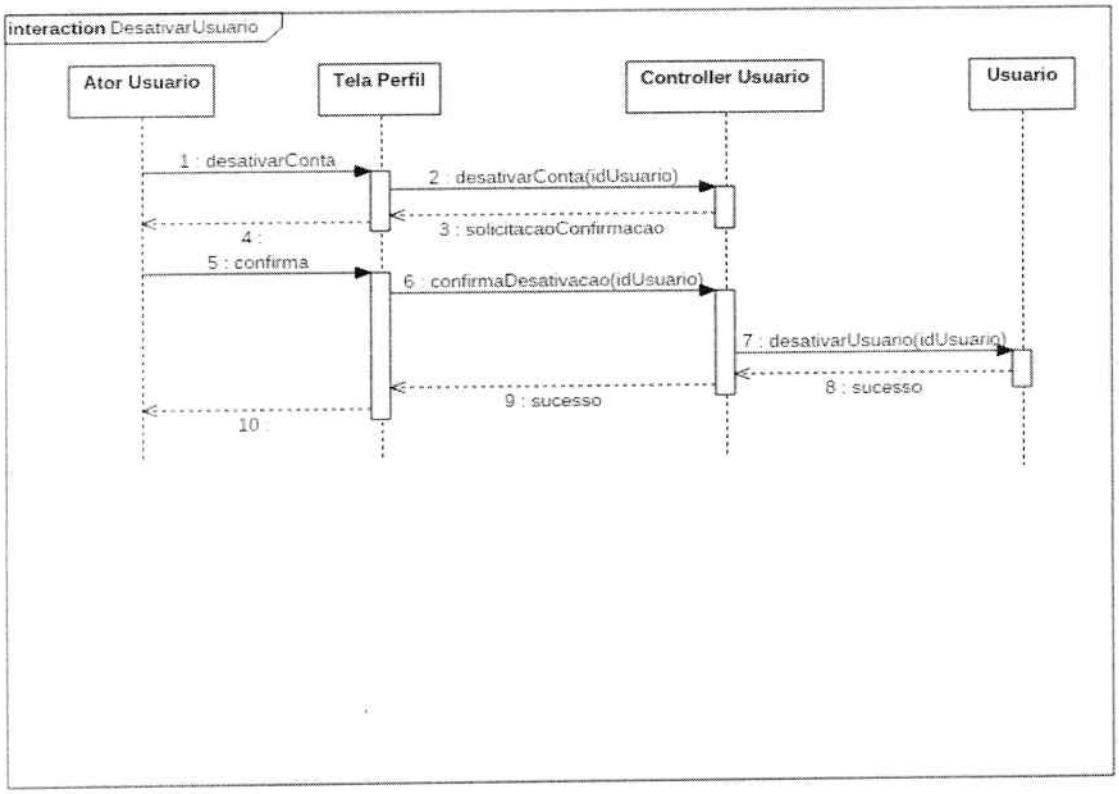
**Pós-Condição:** Usuário em estado desativado, tela de *login* exibida.

**Extensões:**

1. Usuário não confirma desativamento (passo 2) - Sistema encerra caso de uso.

**Inclusões:** -

*Figura 46 - Caso de uso: Desativar Usuário*



## 2. Casos de uso do concentrador

A seguir, são apresentados os dois casos de uso do concentrador de dados, também acompanhados de diagramas de sequência.

**CASO DE USO 1 -** Coletar Dados

**Descrição:** Coletar de dados do paciente monitorado a cada acionamento do relógio de coleta.

**Atores:** Paciente Monitorado ou Relógio de Coleta

**Evento Iniciador:** acionamento do Relógio de Coleta.

**Pré-condição:** Concentrador em *stand by*.

**Sequência de Eventos:**

1. Concentrador faz a leitura de cada sensor conectado ao Paciente Monitorado
2. Concentrador armazena os dados coletados

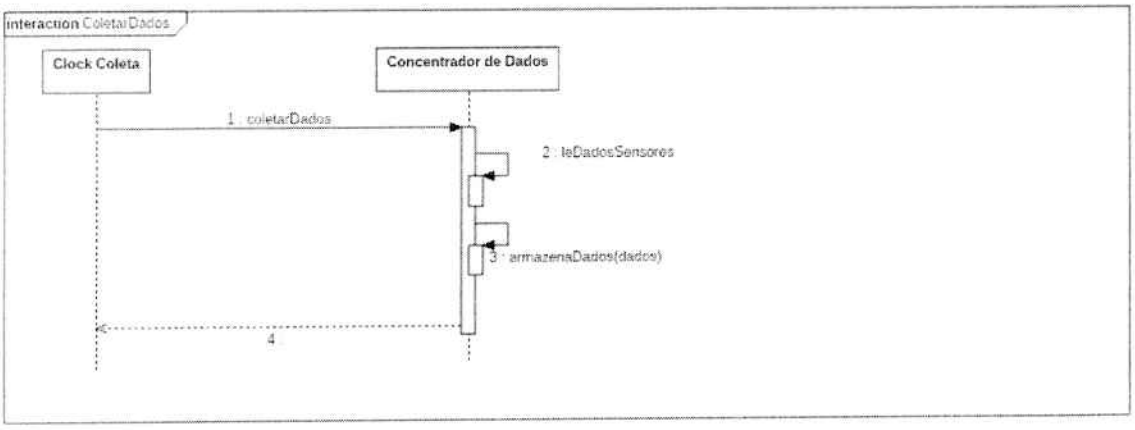
**Pós-Condição:** Dados lidos dos sensores e armazenados

**Extensões:**

1. Concentrador não possui mais espaço para armazenar novos dados (passo 2) - Concentrador descarta as leituras mais antigas.

**Inclusões: -**

*Figura 47 - Caso de uso: Coletar Dados*



**CASO DE USO 2 -** Enviar Dados

**Descrição:** Enviar dados do paciente monitorado a cada acionamento do Relógio de Envio.

**Atores:** Relógio de Envio e Sistema.

**Evento Iniciador:** Acionamento do Relógio de Envio.

**Pré-condição:** Concentrador em stand-by

**Sequência de Eventos:**

1. Concentrador recupera os dados armazenados.
2. Concentrador cria pacote contendo os dados.
3. Concentrador envia mensagem ao Sistema contendo o pacote criado e liga a temporização para detecção de falha de comunicação.

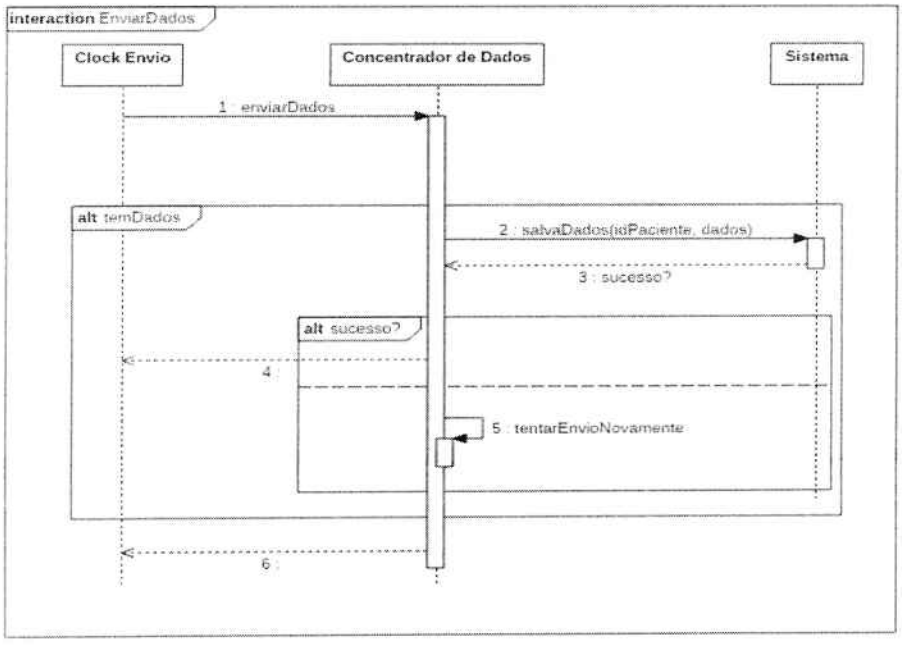
**Pós-Condição:** Pacote de dados enviado e Concentrador em *stand-by*.

**Extensões:**

1. Concentrador não possui dados armazenados (passo 1) - Concentrador encerra o caso de uso.
2. Temporização sinaliza fim de período (passo 3) - Concentrador tenta realizar o envio novamente.

**Inclusões: -**

*Figura 48 - Caso de Uso: Enviar Dados*



# Apêndice B

## 1. Protótipos de Telas

Este apêndice apresenta os protótipos de tela inicialmente desenhados, de forma a direcionar inicialmente o desenvolvimento do aplicativo móvel.

*Figura 49 - Menu Lateral (Paciente)*

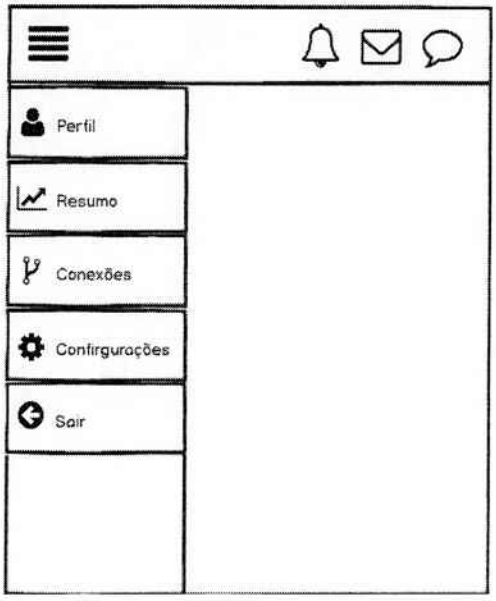



*Figura 50 - Menu Lateral (Médico/Assistente)*

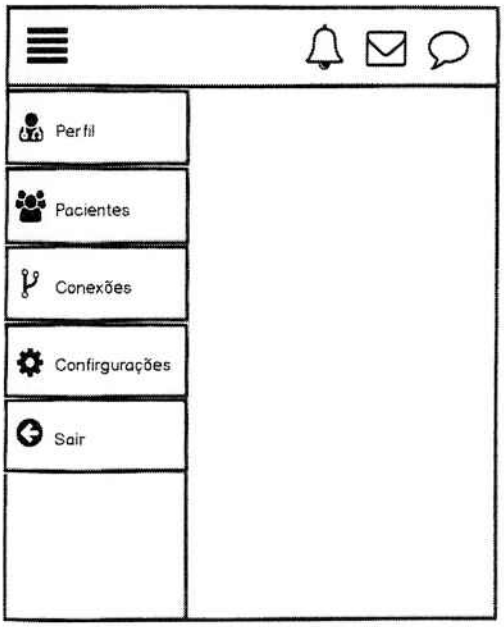


*Figura 51 - Lista de Pacientes*

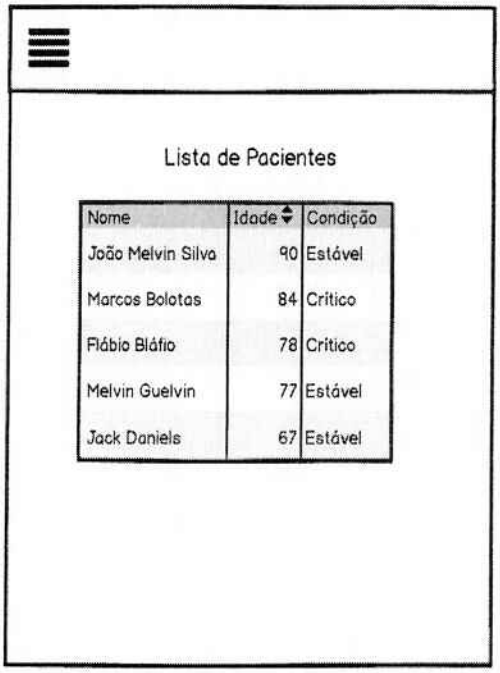



*Figura 52 - Menu Lateral (Médico com paciente selecionado)*

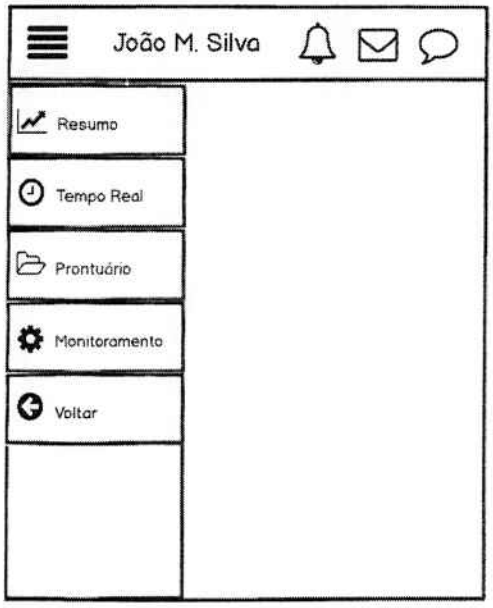


*Figura 53 - Tela de Resumo (Dados em Gráficos)*

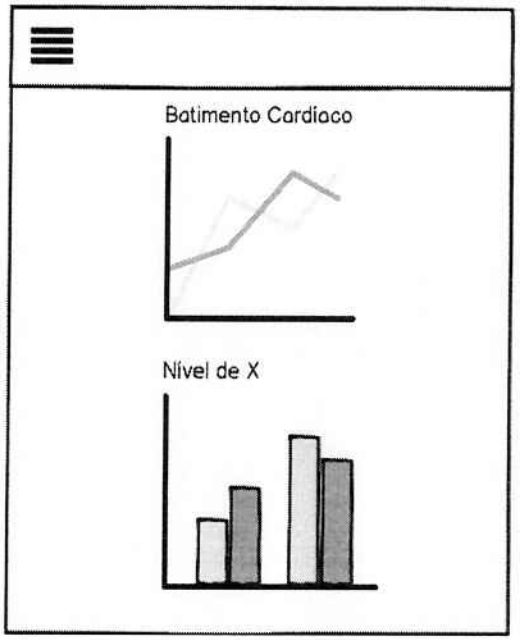



*Figura 54 - Tela de Perfil do Paciente*