



Universidade Federal de Uberlândia – Campus Patos de Minas
Graduação em Engenharia Eletrônica e de Telecomunicações
Faculdade de Engenharia Elétrica

RAFAEL CAIXETA DA SILVA

***SOFTWARE* GERENCIADOR DE DADOS DE UM AMBIENTE
HOSPITALAR**

Patos de Minas
2017

RAFAEL CAIXETA DA SILVA

***SOFTWARE* GERENCIADOR DE DADOS DE UM AMBIENTE HOSPITALAR**

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia, campus Patos de Minas, como requisito parcial para obtenção do bacharelado em Engenharia Eletrônica e de Telecomunicações.

Orientadora: Prof^a. Dr^a. Eliana Pantaleão

Patos de Minas - MG
Julho de 2017

SOFTWARE GERENCIADOR DE DADOS DE UM AMBIENTE HOSPITALAR

Trabalho de conclusão de curso
apresentado à Faculdade de Engenharia
Elétrica da Universidade Federal de
Uberlândia, campus Patos de Minas, como
requisito parcial para obtenção do
bacharelado em Engenharia Eletrônica e
de Telecomunicações.

Orientadora: Prof^a. Dr^a. Eliana Pantaleão

Patos de Minas, 20 de Julho de 2017.

Prof^a. Dr^a. Eliana Pantaleão

Prof^a. Dr^a. Karine Barbosa Carbonaro

Prof. Me. Alexander Bento Melo

Agradecimentos

A toda a minha família pelo grande suporte oferecido para que eu pudesse cursar um ensino superior com tranquilidade. Principalmente ao meu irmão Renato Roberto e minha mãe Maria Mercês, por acompanhar de perto a minha graduação, sempre me motivando e cobrando nas horas certas.

A todo o corpo docente, técnicos de laboratório e administrativos da Universidade Federal de Uberlândia, pelo excelente trabalho prestado a todos os alunos. Em especial, a minha orientadora Prof. DR^a Eliana Pantaleão, pela paciência e dedicação como orientadora.

A todas as amigas adquiridas durante a graduação, foram muito importantes para tornar este ciclo menos tenso, e ter boas histórias para contar na segunda feira. Levarei para a vida toda.

Resumo

Este trabalho tem objetivo a apresentação do Software de interface para um sistema que monitora a posição de pacientes acamados dentro de uma UTI/CTI para que não desenvolvam nem um tipo de lesão por pressão. O *Software* caracteriza-se pelo cadastro de pacientes e profissionais da saúde. Com a funcionalidade de gerenciar dados, estes são enviados de um sensor de posição que vai colado junto ao peito paciente. O envio se dá pela rede *wireless* local, pois o *software* tem características de servidor, já que recebe todos os dados de posição de cada sensor. Ao receber os dados de posição, é preciso gerenciá-los e atribuí-los para cada paciente no qual foram cadastrados. Tendo todos os dados em mãos, é preciso informar aos profissionais da saúde a hora certa da movimentação de cada paciente. Para que a movimentação seja feita na hora e no paciente certo, a tela principal do *Software* mostra o tempo exato que falta para que a mudança de decúbito seja feita. Assim que a mudança de posição é executada, o sensor envia os novos dados para o Software. Esta por sua vez, já começa uma contagem regressiva de quando será a próxima mudança de decúbito. Assim o ciclo se inicia novamente até que o paciente saia da UTI/CTI, evitando que o mesmo adquira uma lesão por pressão. Apesar de ser um *software* voltado para o ambiente hospitalar, ele pode futuramente ser expandido para *Home Care*.

Palavras-chave: lesão por pressão, *Software*, servidor, *ethernet*, hospital e pacientes.

Abstract

This paper aims to present the Interface Software to a system that monitors the position of patients bedridden within a ICU / ICU so that they do not develop a type of pressure injury. The Software is characterized by the registration of patients and health professionals. With the functionality of managing data, these are sent from a position sensor that is glued together to the patient chest. The sending is via the local wireless network, since the software has server characteristics, since it receives all the position data of each sensor. When receiving the position data, it is necessary to manage them and assign them to each patient in which they were registered. Having all the data in hand, you must inform the health professionals at the right time of the movement of each patient. In order for the movement to be made in the right time and in the right patient, the main screen of the Software shows the exact time that is needed for the change of decubitus. As soon as the change of position is performed, the sensor sends the new data to the Software. This in turn, already begins a countdown of when will be the next change of decubitus. Thus, the cycle starts again until the patient leaves the ICU / ICU, avoiding the patient to acquire a pressure injury. Although it is software aimed at the hospital environment, it can be expanded in the future to Home Care.

Key words: pressure injury, Software, server, ethernet, hospital and patients.

Lista de Ilustrações

Figura 1 - Modelo de classe UML.....	15
Figura 2 - Símbolo da linguagem Java.....	17
Figura 3 - Modelo TCP/IP.....	18
Figura 4 - Rede <i>Mesh</i> utilizada na Guerra do Iraque.	20
Figura 5 - Estágios da UPP.....	22
Figura 6 – Escala de Braden para cadastro de pacientes.	25
Figura 7 - Tela principal do <i>Software</i>	26
Figura 8 - Tela para consultar um paciente através de seu CPF.	27
Figura 9 – Diagramas de Classe do <i>Software</i> Turnit.....	28
Figura 10 - Divisão das responsabilidades e necessidades dos stakeholders.	29
Figura 11 - Esquema do ciclo do sistema Turnit.	344

Lista de Siglas

UPP	Úlcera por pressão
USB	Barramento Serial Universal
BLE	Bluetooth de baixa energia
UTI	Unidade de Terapia Intensiva
RN	Regra de Negócio
RF	Requisito Funcional
POR-RNF	Portabilidade – Requisito não Funcional
DES-RNF	Desempenho – Requisito não Funcional
COM-RNF	Confiabilidade – Requisito não Funcional
USA-RNF	Usabilidade - Requisito não Funcional
RDS-RNF	Restrição de Design - Requisito não Funcional
IUS-RNF	Interface do Usuário – Requisito não Funcional
ICO-RNF	Interface de Comunicação - Requisito não Funcional
SDK	Ferramenta de desenvolvimento
IDE	Ambiente de Desenvolvimento Integrado
CPF	Cadastro de Pessoa Física
MAC	Controle de Acesso de Mídia
LLC	Controle lógico de link
CAT5	Cabo par trançado 100Mhz
CAT6	Cabo par trançado 250Mhz
IP	Protocolo de Internet

SMTP	Protocolo de transferência de correio simples
FTP	Protocolo de Transferência de Arquivos
HTTP	Protocolo de Transferência de Hipertexto
TCP	Protocolo de controle de transmissão
SGB	Sistema que gerencia banco de dados
POO	Programação Orientada a Objetos
UML	Linguagem de Modelagem Unificada

Sumário

Agradecimentos	iv
Resumo	v
<i>Abstract</i>	vi
Lista de Ilustrações	vii
Lista de Siglas	viii
Capítulo 1	12
1.1 Introdução	12
1.2 Objetivo	13
1.3 Organização do Texto	13
Capítulo 2	14
Revisão Bibliográfica	14
2.1 Orientação a Objeto	14
2.1.1 Classe	14
2.1.2 Objetos	15
2.1.3 Herança	15
2.1.4 Polimorfismo	16
2.2 Linguagem Java	16
2.3 Banco de Dados	17
2.4 Protocolo TCP/IP	18
2.5 Rede Mesh	19
2.6 Lesão por pressão, Escara, Úlcera por pressão	21
Capítulo 3	23
Metodologia	23
3.1 Requisitos do <i>Software</i>	23
3.2 Requisitos do <i>Hardware</i>	24
3.3 Solução Proposta	24
3.4 Necessidades e Responsabilidades dos <i>Stakeholders</i>	28
3.5 Regras de Negócio	29
3.6 Requisitos	30
3.6.1 Requisitos Funcionais	30
3.6.2 Requisitos Não-Funcionais	30

3.6.3 Requisitos de Interfaces	31
3.7 Programação	32
Capítulo 4	35
Resultados e Discussões	35
Capítulo 5	36
Conclusão	36
5.1 Trabalhos Futuros	36
Referências	37

Capítulo 1

1.1 Introdução

Com o progresso constante da tecnologia da informação (TI), telecomunicações, eletrônica, computação e outras, uma área muito beneficiada é a da saúde. Entretanto, por mais que o mundo está informatizado, com dispositivos de monitoramento inteligente de parâmetros fisiológicos do corpo humano, ainda é necessário o acompanhamento dos profissionais da saúde. Esses profissionais são indispensáveis até o momento, surge a possibilidade de criar tecnologias inovadoras para auxiliá-los em seu trabalho diário. Como o trabalho humano é crucial, e a integração da tecnologia neste ambiente ainda é pouco explorada, os sistemas médico-hospitalares acabam possuindo características arcaicas.

Na primeira concepção, o público-alvo seriam as clínicas, hospitais, centros de saúde e até *home care*. O sistema, que se chama *Turnit*, pode ser personalizado de acordo com as necessidades e demandas do cliente, já que seu intuito é auxiliar o responsável pela mudança de posição dos pacientes. No caso do ambiente hospitalar, a cada expansão, são ofertados novos sensores para cada paciente ou leito acrescentado, ou então removido do sistema, caso o leito seja desocupado.

O Sistema *Turnit* pretende atuar no ambiente hospitalar com inovação e confiabilidade no monitoramento do paciente e dos processos envolvidos na arte do cuidar, pretende utilizar tecnologia para melhorar a qualidade de vida. A ideia é oferecer uma garantia ao paciente de que o mesmo será bem atendido e bem cuidado, pois se o cliente possuir o sistema instalado, ele terá todas as garantias que o sistema oferece para que seja disponibilizado o melhor tratamento.

As oportunidades de empreender surgiram em torno da pouca utilização de tecnologia sem fio para aquisição de sinais biológicos no ambiente hospitalar. Outra oportunidade se gera na ocorrência de simples erros humanos que podem causar graves consequências, que podem ser evitadas com um sistema de monitoramento em tempo real. O serviço de saúde que optar por essa tecnologia poderá utilizar *marketing* para destacar seu diferencial em relação a concorrentes.

1.2 Objetivo

O objetivo deste trabalho é apresentar o *Software* do Sistema Turnit, um sistema que faz parte da Startup Turnit. Para isso, descreve como o software foi construído, e como as necessidades do usuário final foram consideradas na hora da criação do mesmo. Este trabalho também mostra uma forma de monitorar e acompanhar remotamente os processos envolvidos na prevenção de úlceras por pressão no ambiente hospitalar, além de mostrar que este sistema pode ser aplicado para outros ambientes.

1.3 Organização do Texto

O trabalho está dividido em 5 capítulos. Neste primeiro capítulo foi contextualizado o Sistema Turnit e seu *Software*. O capítulo 2 aborda uma revisão bibliográfica sobre Orientação a Objeto, Protocolo TCP/IP, Banco de Dados, Linguagem Java, Escara e Rede *Mesh*, que são fundamentais para total entendimento do problema que o *Software* pretende resolver. No capítulo 3 estão descritos os meios e métodos usados para a engenharia do *software*. O capítulo 4 apresenta os resultados obtidos com o que foi programado até o presente momento. No capítulo 5 são feitas conclusões a partir dos resultados obtidos e são levantados futuros passos do *Software*.

Capítulo 2

Revisão Bibliográfica

2.1 Orientação a Objeto

Alan Kay (1960), um dos pais do paradigma da orientação a objetos [1], fez uma analogia da programação a um sistema biológico, “Como seria um sistema de *Software* que funcionasse como um ser vivo?” [2]. Com isso, uma das grandes características da orientação a objetos é se aproximar do modo como os seres vivos se comunicam e, com isso, as linguagens que a utilizam como Java, C++, Python [1], dentre outras são consideradas linguagens de alto nível. Já as linguagens que se aproximam da comunicação de máquinas como C, Pascal, Assembly, são tratadas como linguagens de baixo nível [1].

Um sistema de *Software* orientado a objetos consiste de objetos em colaboração com o objetivo de realizar as funcionalidades deste sistema, onde cada objeto é responsável por tarefas específicas, e é através da cooperação entre eles que o sistema se desenvolve [3].

2.1.1 Classe

A definição das classes e suas funcionalidades é uma das principais etapas na construção de um *Software* [4]. O resultado do planejamento da classe é expresso em linguagens de modelagem como Unified Modeling Language (UML) [1]. É na classe que definimos os atributos que o objeto terá e as funcionalidades que podem ser atribuídas a ele; funcionalidades estas que são descritas através de métodos [4], que se assemelham com funções da linguagem de baixo nível. Um modelo de classe pode ser visto na Figura 1, que mostra uma classe projetada em UML, na qual temos o Nome, atributos e métodos da classe respectivamente.

Figura 1 - Modelo de classe UML

Fonte: o autor.

2.1.2 Objetos

Objetos são instâncias das classes que servem para o desenvolvimento de sistemas orientados a objeto complexos [4] e de alta eficiência. Para que se defina um objeto é necessário abstrair algumas características do mesmo, de forma a solucionar o problema proposto. Caso objetos se apresentem similares [3], os mesmo devem ser agrupados em classes.

Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como um ser vivo se relaciona com o meio, só que utilizando mensagens para outros objetos [5].

2.1.3 Herança

Herança é um termo que se usa para uma classe que herda características de uma classe base [3], ou seja, herda os atributos e métodos da classe base e acrescenta a ela características particulares a ela. A subclasse é uma extensão da superclasse [6], onde é acrescentado nela atributos e/ou métodos.

Cada nível de uma hierarquia pode ser visto como um nível de abstração [5] do objeto. Cada classe em um nível da hierarquia herda as características das classes nos níveis acima e adiciona a ela novos atributos/métodos. Quanto menor o nível da abstração da classe, mais especializada ela se torna[4-6].

2.1.4 Polimorfismo

Polimorfismo é o princípio pelo qual duas classes ou mais que foram derivadas de uma superclasse podem implementar métodos com a mesma assinatura com comportamentos diferentes [1]. Desse modo, é possível criar diferentes comportamentos para o método para cada subclasse.

Quando o polimorfismo é utilizado, os métodos devem ter a mesma assinatura dos que estão contidos na classe principal [2]; porém devem ser acompanhados do mecanismo de redefinição de métodos [1-3], o *@overriding*. Para que o compilador saiba qual o método a ser executado, há um mecanismo chamado de *late binding*, *dynamic binding* [4] ou ainda *run-time binding* [3], que é uma decisão tomada em tempo de execução.

2.2 Linguagem Java

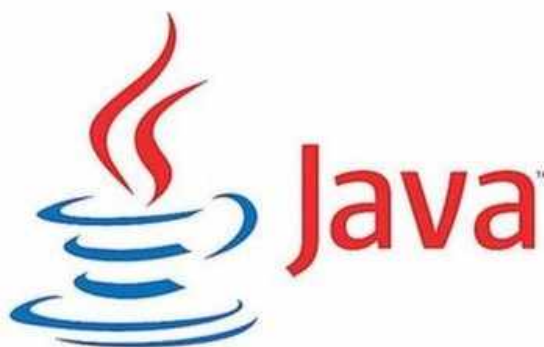
Em meados de 1991, um projeto foi desenvolvido por um grupo coordenado por James Gosling e Patrick Naughton [2] da desenvolvedora Sun Microsystems, chamado “Green” [1].

Em maio de 1995 foi feito o lançamento de um kit de desenvolvimento de aplicativos corporativos de grande porte, que era gratuito e distribuído pela Sun, em que era possível fazer o download dele pelo site da empresa. [3]. A princípio essa linguagem foi projetada para a Internet, pois possuía segurança e portabilidade [1], além de outras vantagens como os recursos de redes e o grande conjunto de bibliotecas. Sua sintaxe era muito semelhante à linguagem C/C++.

O símbolo da linguagem Java é uma xícara de café, como mostrado na Figura 2. Isso foi originado por uma ilha da Indonésia que se chama Java [4], onde a principal atividade econômica do país é a exportação de café. O nome original da linguagem seria Oak [3], porém já existia uma linguagem de programação com este nome. Então seus criadores que consumiam muito café enquanto desenvolviam, e enquanto buscavam um novo nome para a linguagem [4], descobriram que o café que consumiam era produzido na ilha de Java, fato que segundo a lenda, levou a escolha do nome.

Várias IDEs surgiram ao longo do tempo para trabalhar com esta linguagem que utiliza o compilador javac. Dentre estas IDEs, algumas que podem ser citadas como o *Netbeans*, *JavaMaker*, *Eclipse*, e o *Java WebIDE* que é um compilador online da linguagem Java.

Figura 2 - Símbolo da linguagem Java.



Fonte: Java Logo Vector, <<https://seeklogo.com/vector-logo/158094/java>> Acesso em 11/05/2017.

2.3 Banco de Dados

Banco de dados, ou base de dados é organizado em forma de tabela estruturada contendo registro sobre pessoas, animais, lugares ou coisas [7]. Sempre que possível agrupar informações que se relacionam por um assunto em uma tabela, teremos então um banco de dados, pois uma função do banco de dados é justamente organizar os arquivos de maneira que um dado possa ser facilmente encontrado dentre tantos outros disponíveis [8].

Um sistema que gerencia um banco de dados (SGBD) é um Software que possui recursos que são capazes de manipular as informações contidas no banco de dados e interagir realizando alguma busca, cadastro ou exclusão de itens [7]. Alguns exemplos de SGBDs mais conhecidos são: *Oracle*, *SQL Server*, *DB2*, *MySQL*, ou até mesmo o *Access* que é instalado juntamente com o pacote *Office*.

Existem algumas soluções que integram aplicações Java Orientadas a Objeto com banco de dados. Uma dessas soluções é utilizar a linguagem SQL [7], uma linguagem padronizada para manipulação de banco de dados. Para se utilizar esta linguagem, é preciso criar o banco externamente antes de trazê-lo para dentro das

aplicações Java. Existem alguns *Softwares* que permitem essa funcionalidade, como por exemplo o MySQL, que é o SGBD [8] mais popular dentre os desenvolvedores, devido a seu código fonte ser aberto.

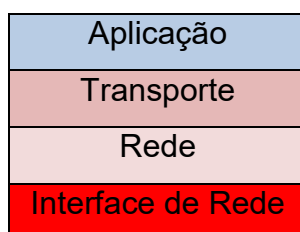
2.4 Protocolo TCP/IP

Nos tempos atuais, o protocolo TCP/IP é o mais utilizado no mundo todo [9]. Para que dois dispositivos troquem mensagens entre si, uma certa “sequencia” deve ser estabelecida, esta “sequencia” é chamada de protocolo. O TCP/IP vai além de um protocolo, pois se torna uma pilha deles, o TCP (*Transmission Control Protocol*), o IP (*Internet Protocol*) [10] e outros que serão abordados.

Na Figura 3, podemos ver o modelo TCP/IP que possui 4 camadas. A primeira camada, a de aplicação, utilizará um protocolo definido pelo programa [11] que esta sendo utilizado no momento, caso seja um e-mail, ele utilizará o SMTP, se estiver transferindo arquivos, o FTP, ou HTTP caso esteja navegando na internet [9].

Logo que a requisição é processada, a camada de aplicação conversa com a camada de transporte [11]. Na camada de transporte, os dados são fragmentados em pacotes e enviados para o destinatário, que ao recebê-los remonta-os na sequencia correta [9], já que os pacotes podem chegar embaralhados. Este processo de fragmentação e recomposição dos dados é feito pelo protocolo TCP. O IP incrementa o TCP na camada de rede, onde o endereço (IP) do dispositivo de origem e do dispositivo de destino são inseridos aos pacotes, que nesta camada são chamados de datagramas [9].

Figura 3 - Modelo TCP/IP.



Fonte: o autor.

Na camada de Interface, os pacotes são transportados até o destinatário final [10]. Como na camada de Aplicação dependia do programa a ser utilizado, a camada Interface de Rede depende do tipo de rede que o computador destinatário

está inserido; podendo ser Ethernet (par trançado CAT5 ou CAT6) [11] ou Wi-Fi (sem fio). Nesta etapa, os pacotes transmitidos são denominados de quadros, e é aí que camadas de Ethernet, como o MAC (*Media Access Control*) e LLC (*Logic Link Control*) são utilizados para finalizar este ciclo de envio [9].

2.5 Rede Mesh

Uma rede que possui equipamentos com função de roteador/ repetidor, é denominada *Wireless Mesh* [11]. Equipamentos conectados a rede, como computadores, *smartphones* e PDA's podem ter a função de rotear e repetir sinal *wireless*, requisito fundamental para se ter a rede *Mesh*, também chamada de *multi-hop*, ou seja, tecnologia de múltiplos saltos [12].

A tecnologia teve origem no *Defense Advanced Research Projects Agency* (DARPA), que é o centro de desenvolvimento militar dos Estados Unidos [11]. O objetivo desta tecnologia foi criar uma rede que permitisse a comunicação fim a fim sem a necessidade de um nó central, dentre outras características como:

- Banda Larga;
- Suporte IP;
- Transmissão de voz e vídeo;
- Posicionamento geográfico;
- Suportar comunicação móvel em velocidades de até 400km/h.

Apesar dessa técnica ter sido desenvolvida a muito tempo, ela vem sido utilizada na guerra dos Estados Unidos contra o Iraque, assim como mostrado na Figura 4.

O diferencial atual desta tecnologia é a sua utilização com as novas tecnologias sem fio que as redes oferecem, e, ao invés de depender de cabos conectando pontos, todos os pontos que estão em um alcance mútuo podem se comunicar [11], eliminando a necessidade de cabos, e expandindo o alcance da rede *wireless*. Comparado a uma rede *wireless* comum, podemos citar algumas vantagens da rede *wireless Mesh* como:

- Distâncias maiores de transmissão mantendo uma taxa de transmissão alta. Como a rede funciona com “saltos”, os dados são repassados para o dispositivo mais próximo, que mantém uma velocidade maior do que o

dispositivo final situado longe do emissor; assim os dados são repassados até chegar ao seu destino com uma boa taxa de transmissão de dados e uma grande distância entre o emissor e o receptor, já que no modelo comum, a taxa de transferência *wireless* [12] cai de acordo com a distância entre os dois pontos;

- Perda da necessidade da linha de visada. Um obstáculo pode ser contornado com as redes adjacentes, eliminando assim a necessidade de se ter uma linha de visada fixa, já que sempre existirá outra rota por onde os dados poderão ser transmitidos;
- Redução da quantidade de Access Points conectados à internet, já que todos os nós se comunicam mesmo sem a internet;
- Uma rede mais independente, pois não necessita de um nó principal, e na falta de um nó qualquer, as transmissões são feitas por nós ali próximos, contornando de maneira fácil o que seria um problema sério se o módulo principal apresentasse defeito.

Figura 4 - Rede *Mesh* utilizada na Guerra do Iraque.



Fonte: *MeshNetworks, Inc.*

No cenário atual, a tecnologia *wireless Mesh* está sendo desenvolvida por diversas empresas, a maioria delas startups [12]. Estes desenvolvimentos, apesar de respeitarem, em muitos aspectos, algumas especificações atualmente utilizadas, como a 802.11 e 802.16 [11], apresentam diversos aspectos particulares para cada

aplicação nas quais são desenvolvidos, e por isso são devidamente patenteados pelas empresas que os desenvolvem.

2.6 Lesão por pressão, Escara, Úlcera por pressão

Lesões por pressão, escaras ou úlceras por pressão, são os nomes dados às feridas causadas pela má movimentação de uma pessoa acamada.

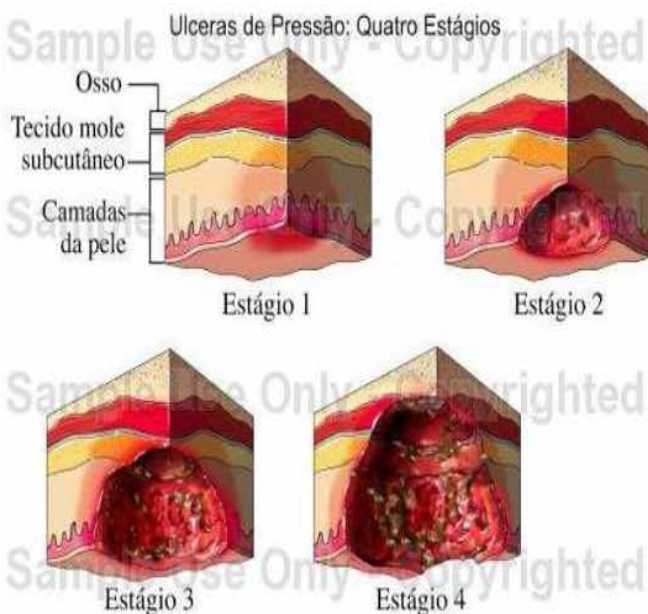
As feridas crônicas são um tipo de ferida que não cicatrizam de forma ordenada e oportuna, mais especificamente, as feridas que não cicatrizam no prazo de três meses são consideradas crônicas [13]. As úlceras por pressão (UPP) são um caso específico de ferida crônica podem ser causadas por fatores biológicos intrínsecos como idade avançada, estado nutricional ruim, perfusão tecidual, doenças cardiovasculares e crônicas como o diabetes. Já dentre os fatores extrínsecos estão a umidade da pele e as forças mecânicas de cisalhamento, fricção e pressão que o próprio corpo do paciente faz sobre sua pele. A pressão no tecido é o principal fator extrínseco causador da UPP e sua patologia está relacionada à intensidade, duração e tolerância tecidual. [14].

A úlcera por pressão ocorre quando a pressão intersticial excede a pressão intracapilar, originando uma deficiência de perfusão capilar, o que impede o transporte de nutrientes ao tecido. Esta situação é mais comum em áreas de proeminências ósseas, onde o osso e a superfície de contato onde o paciente se apoia (cama ou cadeira) exercem uma pressão sobre a pele e partes moles sobre o osso maior do que a pressão capilar. Quando esta isquemia tecidual gerada pela pressão é mantida por um tempo maior do que o necessário à recuperação do tecido frente à isquemia, ocorre a liberação de fatores inflamatórios, que alteram a permeabilidade vascular, gerando edema e piorando a isquemia, caso a pressão intersticial permaneça aumentada. A isquemia a nível celular leva à morte celular, gerando a liberação de mais fatores inflamatórios e fatores de necrose tecidual.[15]

Muitas vezes existe certa dificuldade em se classificar as úlceras, devido à característica crônica das lesões, mas a *National Pressure Ulcer Advisory Panel*, parte de um sistema internacional de classificação das UPP, estabelece que a pele intacta com rubor numa área não branqueável, normalmente sobre uma proeminência óssea, já é considerada UPP de estágio I, como podemos ver na Figura 5.

No estágio II, há uma perda parcial da espessura da pele. A pele apresenta-se com uma coloração vermelho pálida. Pode apresentar-se ainda como uma bolha, intacta ou aberta/ rompida.

Figura 5 - Estágios da UPP.



Fonte: Úlcera por pressão, disponível em < <http://ulcerap.blogspot.com.br/>> Acesso em 17/05/2017.

No estágio III, há perda do tecido em sua espessura total. A gordura subcutânea pode estar visível, sem exposição de osso, tendão ou músculo. A profundidade da úlcera por pressão em estágio III varia conforme a localização da mesma. Áreas com uma camada grossa de gordura podem desenvolver UPPs em estágio III bastante profundas.

No último estágio, é apresentada uma perda total de tecido com exposição da camada óssea, de músculo ou tendão. Pode haver presença de necrose em algumas partes do leito da ferida.

A profundidade da úlcera por pressão em estágio IV varia conforme a localização anatômica. As úlceras neste estágio podem estender-se aos músculos e/ou tendões. Com isso há exposição de osso/tendão, que se torna visível ou diretamente palpável, levando a recuperação da ferida a um processo bastante demorado.

Capítulo 3

Metodologia

O desenvolvimento do sistema iniciou-se com uma revisão bibliográfica dos principais temas envolvidos, escolha dos parâmetros a serem monitorados, entradas e saídas necessárias. Em seguida, foram definidos os requisitos de *Hardware*, *Software* e sua arquitetura do sistema. Com isso foram levantados as Regras de Negócio, Requisitos Funcionais, Não Funcionais, Restrições de *Design*, Usabilidade, Desempenho, e Interfaces. Todo este levantamento seguiu regras básicas da Engenharia de *Software* para uma boa construção do programa Turnit, o qual foi desenvolvido na linguagem Java. Para se modelar o *Software* antes de programar, foi utilizado o site <https://www.gliffy.com>, que é uma ferramenta para criar diagramas de classe do sistema.

3.1 Requisitos do *Software*

Barra de tarefas: Cadastro, Consulta, Sobre

- **Usuários**

- ✓ Usuário (Enfermeiro);
- ✓ Usuário (Técnico de enfermagem).

- **Interface principal**

- ✓ Cadastro de pessoas;
- ✓ Cadastro de leitos;
- ✓ Visualização de leitos;
- ✓ Avisos visuais.

- **Sobre**

- ✓ Desenvolvedores;
- ✓ Suporte;
- ✓ Versão do *Software*;

3.2 Requisitos do *Hardware*

- **Sensor**
 - ✓ Acelerômetro;
 - ✓ Microcontrolador;
 - ✓ BLE.

- **Dispositivo gerenciador**
 - ✓ Microcontrolador;
 - ✓ BLE;
 - ✓ Cabo USB;
 - ✓ Acesso a internet.

- **Funcionário**
 - ✓ BLE ativo;
 - ✓ Microcontrolador.

3.3 Solução Proposta

O *Software Turnit* contém um sistema automatizado que computa a mudança de decúbito de um paciente hospitalizado. Outros dados importantes para nosso sistema é a informação dos profissionais que foram responsáveis pela movimentação do paciente, além da data e hora do ocorrido. Para isto, tanto o enfermeiro quanto o técnico em enfermagem deverão ter um crachá com código BLE (*bluetooth low energy*). Cada leito da UTI deve conter um leitor deste cartão BLE integrado a um microcontrolador que ao detectar a presença do(s) funcionário(s) e computar mudanças a um sensor de posição fixado no paciente, que o mesmo foi movido, envia estes dados a um microcontrolador local que repassará as informações para um computador pela rede local. O computador que recebe os dados onde o *Software Turnit* está instalado.

O *Turnit* controla o tempo que o paciente deve ser movido, e assim que ele necessitar de uma mudança de posição, um alerta visual será acionado até o trabalho ser feito pela equipe de enfermagem. Com isso o *Software* gerencia o paciente que foi movido, o horário e os funcionários envolvidos no processo;

facilitando assim para a equipe de enfermagem que será notificada pelo aviso visual, também para os gestores que saberão como a sua equipe está trabalhando, e principalmente para o paciente, que terá um atendimento diferenciado.

O *Software* é composto pelos seguintes módulos:

Módulo de Cadastro: Este módulo permite o cadastro de enfermeiros, técnicos em enfermagem e pacientes acamados. Os cadastros devem ser feitos por um administrador. No momento do cadastro de pacientes é feita a *Escala de Braden*. Esta escala pode ser conferida na Figura 6, e é uma escala de 6 eixos para definir o risco do paciente desenvolver a ferida.

Figura 6 – Escala de Braden para cadastro de pacientes.

Fonte: o autor.

Módulo de Visualização: Permite a visualização dos leitos ocupados do hospital. Outra função desenvolvida para este módulo é a informação vinculada a cada leito, basta um clique em um leito para saber as informações contidas nele. Este módulo é o principal responsável pelo aviso visual que informa os enfermeiros que os mesmos devem agir sobre um determinado paciente. De preferência, este módulo deve ficar exposto em uma TV ou monitor que toda a equipe enxergue com facilidade. A Figura 7 ilustra a interface do sistema, onde temos à esquerda

informações do leito 6, e também podemos notar que o paciente do leito 1 deve ser movimentado.

Figura 7 - Tela principal do Software.

The screenshot shows the main interface of the Turnit software. On the left, there is a patient information panel with the following details:

- Nome: Willian
- CPF: 121.212.121-21
- Idade: 34
- Leito: 6
- Sexo: Masculino
- Escala Braden: 16 - Risco Médio
- Data de internação: 30/09/2016
- Hora da internação: 12:55:50
- Observações:

To the right of the panel is a 4x5 grid of beds, labeled from Leito 1 to Leito 20. The grid is as follows:

Leito 1 0	Leito 2	Leito 3	Leito 4	Leito 5
Leito 6	Leito 7	Leito 52	Leito 9	Leito 10
Leito 11	Leito 12	Leito 13	Leito 14	Leito 15
Leito 16	Leito 17	Leito 18	Leito 19	Leito 20

The 'Leito 1 0' cell is highlighted in red, and the 'Leito 6' cell has a small blue box around it.

Fonte: o autor.

Módulo de Registro: Permite a presença dos profissionais a ser registrada no sistema assim que comparecem junto ao leito. Só é contabilizada a presença do(s) profissional(is) quando os mesmos movimentam o paciente.

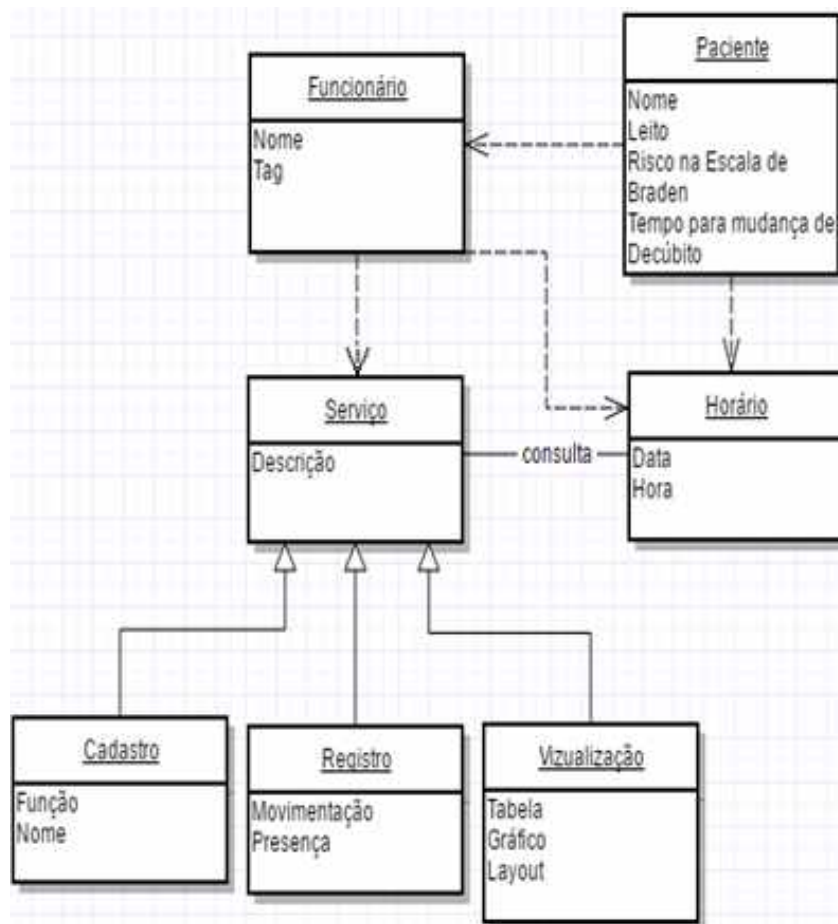
Módulo de Consulta: Permite ao usuário visualizar informações de pacientes e funcionários, além de permitir que se edite dados, ou mesmo a exclusão no sistema de um paciente ou funcionário. Na Figura 8 é possível ver como é fácil a edição ou exclusão de um paciente.

Figura 8 - Tela para consultar um paciente através de seu CPF.

The screenshot shows a software window titled "Consulta de Pacientes". At the top left is a small icon of a notepad and pencil. The window has standard Windows-style controls (minimize, maximize, close) at the top right. Below the title bar, there is a label "CPF" followed by a text input field containing "121.212.121-21" and a blue button labeled "Consultar". Below this, a large rectangular box contains the following patient information: "Nome: Willian", "CPF: 121.212.121-21", "Idade: 34", "Leito: 6", "Sexo: Masculino", "Escala Braden: 16 - Risco Médio", "Data de internacao: 30/09/2016", "Hora da internação: 12:55:50", and "Observações:". At the bottom of the window, there are two blue buttons: "Editar" on the left and "Excluir" on the right.

Fonte: o autor.

Com estas informações em mãos, foi possível dar um norte de como o *Software* deve se apresentar, e para que fique mais fácil de entender o *Software* na hora de programar, foi feito um diagrama pela plataforma *Gliffiy*, um site gratuito para modelagem de *Software*. A modelagem do *Software* pode ser vista na Figura 9, que mostra o comportamento do *Software*, e como ele se comunica.

Figura 9 – Diagramas de Classe do *Software Turnit*.

Fonte: o autor.

3.4 Necessidades e Responsabilidades dos *Stakeholders*

O conjunto dos interessados (*stakeholders*) de um projeto engloba todas as pessoas que de alguma forma podem influir no sucesso do projeto [16]. Assim considera-se interessado desde o patrocinador, até os usuários que são afetados pelo projeto. Cada projeto tem seu grupo de *stakeholders* próprio. No nosso caso, os *stakeholders* encontrados foram os da Figura 10.

Figura 10 - Divisão das responsabilidades e necessidades dos stakeholders.

Envolvido	Necessidade	Responsabilidade
Paciente	Ter um bom atendimento para que não desenvolva feridas	
Enfermeiro (Administrador)	Possuir um sistema que alerte seus funcionários de seu trabalho, e que monitore os mesmos, facilitando assim seu trabalho de fiscalização.	Gerar relatórios e dar um feedback para seus funcionários.
Téc. Enfermagem/ Enfermeiro	Ter acesso a um sistema que o oriente em seu trabalho.	Mudar o paciente de posição assim que o sistema anunciar necessário.

Fonte: o autor.

3.5 Regras de Negócio

Uma RN (Regra de Negócio), no contexto da Engenharia de *Software*, é tratada como um Requisito de *Software* [17], por se tratar de algo que se não funcionar ou não existir, prejudica o andamento do *Software*.

RN01 – Cada funcionário do hospital deverá ter um, e somente um crachá.

RN02 – Os funcionários deverão fazer a mudança de decúbito assim que o sistema acusar que o paciente precisa ser virado.

RN03 – Cada paciente deve ter um sensor acelerômetro para identificar sua mudança de posição.

RN04 – Para que seja registrado no sistema, o funcionário deve comparecer ao leito e movimentar o paciente.

RN05 – O administrador deve cadastrar funcionários e pacientes no sistema.

RN06 – O Técnico em enfermagem pode cadastrar pacientes no sistema.

RN07 – Ao cadastrar um paciente no sistema, deve ser feita a Escala de Braden do mesmo.

RN08 – O computador em que o *Software* estiver instalado deverá possuir um ip fixo.

3.6 Requisitos

Segundo a norma IEEE Std 1220-1994, um requisito é uma sentença identificando uma capacidade, uma característica física ou um fator de qualidade que limita um produto ou um processo [19].

3.6.1 Requisitos Funcionais

Requisitos Funcionais representam algo que o sistema deve ser capaz de fazer, ou seja, uma função esperada do sistema que agregue algum valor a seus usuários.

RF01 – O sistema deve ser capaz de propiciar o reconhecimento de posição de pacientes através de acelerômetro e a presença de funcionários através de seu crachá.

RF02 – O sistema deve ser capaz de cadastrar pacientes e funcionários.

RF03 – O sistema irá gerar tabelas e gráficos com base em dados que foram salvos no servidor.

RF04 – O sistema deve ser capaz de calcular o tempo que falta para cada paciente sofrer a mudança de decúbito.

RF07 – O sistema notificará o funcionário por meio de alerta para avisá-lo de suas obrigações.

RF08 – O sistema deve armazenar todos os dados em um servidor.

RF09 – Cada leito deverá possuir um leitor de crachá.

3.6.2 Requisitos Não Funcionais

Requisitos não funcionais tratam da forma como os requisitos funcionais devem ser alcançados. Eles definem propriedades e restrições do sistema.

Portabilidade

POR-RNF01 – O sistema deverá ser compatível com o sistema operacional Microsoft Windows e outros SO que suportarem a máquina virtual Java.

Desempenho

DES-RNF01 – O sistema deverá responder o mais próximo de tempo real, com tolerância de 2 segundos de *delay*.

DES-RNF02 – Deve-se trocar a bateria do sensor giroscópio todo mês, prevenindo assim que o sistema pare de registrar a mudança de posição.

DES-RNF03 – O sistema deverá ficar disponível vinte e quatro horas por dia.

DES-RNF04 – A transmissão da banda passante deverá ser no padrão 802.11/B/G/N.

Confiabilidade

COM-RNF01 – O *Software* deve ser o mais confiável possível, não podendo haver falhas, e minimizar as chances de erro humano para quem utiliza o *Software*.

Usabilidade

USA-RNF01 – Para o treinamento de todos os ingressantes, sejam técnicos ou enfermeiros, a cada venda do produto, haverá um treinamento, de duas horas, explicando o funcionamento de todo o sistema. O treinamento deverá explicar como funciona todo o sistema eletrônico de presença, instruir administradores a utilizarem recursos que o *Software* proporciona e mostrar a importância dos funcionários sempre utilizar o crachá, além de como se orientar a partir de avisos do *Software*.

USA-RNF02 – O *Software* deve ser simples e de rápido aprendizado, sendo possível sua utilização plena após o treinamento.

Restrições de Design

RDS-RNF01 – O sistema deverá ser desenvolvido em linguagem de programação Java.

RDS-RNF02 – O sistema utilizará um leitor de crachá para identificação dos funcionários.

3.6.3 Requisitos de Interfaces

Interfaces do Usuário

IUS-RNF06 – A notificação da necessidade da mudança de decúbito deverá ser feita por uma TV que mostrará o paciente a ser virado, além de emitir um discreto sinal sonoro.

IUS-RNF07 – A cada funcionário que realiza uma mudança de posição, o leitor envia através de uma rede *wireless* uma informação com a tag do funcionário, paciente, hora e data para o *Software* armazenar em um banco de dados.

Interfaces de Comunicação

ICO-RNF01 – O sistema deverá se comunicar via rede local do hospital para envio de dados para o *Software*.

3.7 Programação

O ambiente de desenvolvimento de *Software* Java, Java SDK utilizado foi a versão 8u131 e é formado essencialmente pelas classes fundamentais da linguagem Java. A ferramenta para desenvolvimento de programas na linguagem Java, JavaSDK, utilizada foi o NetBeans IDE, um *Software* bastante utilizado. Esta IDE se encontra na versão 8.2 e está disponível para 32 ou 64 bits, tanto para *Windows*, quanto para Linux.

Um dos primeiros passos da programação foi montar uma estrutura de *Software* consistente. Com os requisitos definidos e modelagem definida, o primeiro passo foi criar um cadastro de funcionários e pacientes, que a princípio guardava as informações em tempo de execução. Com o avanço do *Software*, foi visto uma necessidade de se guardar estas informações para uso futuro.

Como a informação deste *Software* não pode ser perdida, pois atrapalharia todo o ciclo que a proposta do projeto propõe, os dados foram tratados de modo a armazená-los em arquivos txt. Estes arquivos podem receber informações, sofrer alterações e em futuras execuções as informações estarão disponíveis para dar prosseguimento a todo o processo.

Porém esta não é uma função nativa dos compiladores Java, com isso, duas classes foram adicionadas ao projeto para que o armazenamento fosse possível. Uma classe é a *EscritorArquivoBinario.java*, que pega a informação e armazena ou retira do txt no qual está salvando. A outra classe é a *LeitorArquivoBinario.java*, que abre o txt e processa toda a informação contida ali dentro. Com essas classes importadas para o projeto foi possível salvar qualquer informação gerada pelo *Software Turnit*.

O *Software Turnit* não é independente, então teve que se comunicar com outros dispositivos pela rede *ethernet* b/g/n. Um dispositivo ao qual ele se conecta é a placa de desenvolvimento NUCLEO-F401RE, placa esta que possui o microcontrolador STM32F401RET6, que é um microcontrolador de 32 bits, ARM Cortex M4 de 84 MHz, possui 512 KBytes de memória flash e 96 KBytes de SRAM, possui também conversor ADC de 12 bit, PWM. Esta placa, NUCLEO-F401RE é responsável por comunicar-se com o roteador *wireless* na qual está inserida na rede. Mas outra placa também está acoplada a NUCLEO-F401RE, a placa de desenvolvimento X-NUCLEO-IKS01A1, que implementa diversos sensores, acelerômetro, magnetômetro, sensor de pressão, sensor de humidade, temperatura e giroscópio. Essa placa comunica com o microcontrolador usando I2C, e o modelo do acelerômetro que essa placa possui é o LSM6DS0.

Com isso, a placa NUCLEO-F401RE recebe informações dos sensores da placa X-NUCLEO-IKS01A1 pelo protocolo I2C, e depois transmite todos os dados via *wireless*. O *Software Turnit* fica responsável por criar um *web-server*, e abre a porta 32000, que é a mesma que foi configurada na NUCLEO-F401RE. Desse modo, cria-se uma conexão cliente/servidor e e integra-se a placa de desenvolvimento, que futuramente será o sensor do paciente, com o *Software* do projeto.

No código, podemos verificar como foi criado o Socket servidor na porta 32000 para que um cliente possa se conectar a este servidor. A informação que chega da placa NUCLEO-F401RE é apenas a posição do paciente. Ele transmite C, se o paciente estiver de barriga pra cima, E, se estiver deitado para o lado esquerdo, e D se o paciente estiver deitado para o lado direito. À medida que a informação da posição vai chegando, uma contabilização de tempo começa; caso a posição recebida mude, o tempo começa a contabilizar de novo. Caso contrário o tempo vai sendo acrescido e outras etapas do *Software* tratam se está na hora de alarmar a equipe de enfermagem ou não.

```
1  Gerenciador g = new Gerenciador();
2  long startTime = System.currentTimeMillis();
3  int k = 0, aux1 = 0, cont = 0;
4  boolean ajuda = true;
5  ServerSocket servidor = new ServerSocket(32000);
```

```

6  System.out.println("Porta 32000 aberta!");
7  Reminder r1 = new Reminder();
8  Socket cliente = servidor.accept();
9  System.out.println("Nova conexão com o cliente " +
10 cliente.getInetAddress().getHostAddress());
11 Scanner s = new Scanner(cliente.getInputStream());
12 s.close();
13 servidor.close();
14 cliente.close();

```

Futuramente, para não sobrecarregar a rede hospitalar, e principalmente roteadores, a *Startup Turnit* implementará o conceito de rede *mesh* com a placa de desenvolvimento X-NUCLEO-IDS01A5, que dá suporte a este tipo de conexão, onde um sensor comunica com o outro até enviar a informação para o destino final, que no caso é o *Software*.

O sistema completo se encontra hoje como mostra a Figura 11, onde o funcionário possui um crachá, que recebe informações do sensor de posição. Assim que o funcionário se desloca, as informações são descarregadas para a central de dados, que faz parte do *Software* que gerencia todos os pacientes e passa as informações para uma tela onde todos enfermeiros podem ver.

Figura 11 - Esquema do ciclo do sistema Turnit.



Fonte: o autor.

Capítulo 4

Resultados e Discussões

Com as fases anteriores concluídas, o próximo passo foi a realização dos testes, os quais verificaram a efetividade e a corretude dos requerimentos até aqui implementados. Em seguida, uma versão beta do *Software* foi gerada e mostrada para pessoas da área da saúde.

Assim, concluiu-se que a interface inicial, com a rápida consulta de pacientes e a cor simbolizando o status dos mesmos, está bem acessível para o usuário final do *Software*. Porém, encontrou-se um pouco de resistência por parte dos profissionais com relação ao monitoramento de suas atividades, pois o *Software* amarra uma mudança de posição a um profissional responsável por aquela ação.

Em contrapartida, alguns profissionais apoiaram a ideia do registro de suas atividades. Este registro pode ser usado para que eles provem para seus gestores que a equipe é limitada, está precisando de mais funcionários. Essa pode ser a única solução para que eles mostrem a carência de funcionários do setor, pois não há meios concretos para o levantamento desses dados na atualidade.

No estado de testes em que o produto como um todo se encontra, não é possível afirmar a eficiência da proposta implementada em *Software*. Porém, é cabível afirmar que a parte de notificação e interface com o profissional da saúde se encontra amigável devido a *feedbacks* que ocorreram dentro do programa de aceleração que a *Startup* passou no ano de 2016, o *Academic Work in Capital*, do instituto Tim.

Outra conquista que essa ideia levou à Startup Turnit foi a participação no programa de aceleração Inovativa Brasil, onde ela se encontra entre as finalistas [18]. Além disso, a *Turnit* ganhou o 8º Prêmio de empreendedorismo do Unipam, que contava com mais de 130 *Startups*. Participar destes programas e sempre estar entre os melhores colocados, mostra que a proposta que o *Software* implementa tem sim o seu potencial, e futuro mercado assim que o protótipo de *hardware* se apresentar pequeno o bastante para testar em pessoas acamadas.

Capítulo 5

Conclusão

Este trabalho mostrou como a ideia da criação de um *Software* de ambiente hospitalar se iniciou, suas justificativas e mercado. Não foi apenas a construção de um *Software* que se comunica com sensores de outras plataformas e envolve uma rede *ethernet*, mas também mostrou a validação de uma ideia coesa de uma *Startup* que futuramente quer lançar seu produto no mercado.

Com a elaboração deste *Software*, foi possível colocar em prática conhecimentos de várias áreas da Engenharia Eletrônica e de Telecomunicações, como Eletrônica, Redes de Telecomunicações, Orientação a Objeto, Programação e Engenharia de *Software*. Outro lado bom da elaboração deste projeto, foi exercitar a parte da engenharia, que é projetar um produto que atenda uma demanda do mercado, e nisso realizar todas as etapas até que o mesmo seja comercializável.

O conhecimento adquirido ao término deste trabalho foi proveitoso, pois ajudou a interligar áreas estudadas durante a graduação, como redes de computadores e a própria programação. Assim, um sistema real e comercializável foi possível de ser desenvolvido durante a graduação.

5.1 Trabalhos Futuros

Um dos próximos passos é integrar um banco de dados MySQL ao *Software*, para que o mesmo gere relatórios, gráficos, que serão de muito proveito para a gerência de um hospital. Como a *Startup* se encontra no estágio final do programa Inovativa Brasil, receberá seis semanas de mentorias. Estas mentorias servirão para melhorar a estrutura proposta no *Software*, e assim ajudar na sua futura homologação junto a Anvisa, que será o maior obstáculo para a sua inserção no mercado.

Para o ano de 2018, a Startup pertente participar do programa FIEMG LAB, que fornece mentorias e apoio financeiro para as *Startups* selecionadas. Isso porque *Software* desta complexidade precisa de mais pessoas trabalhando nele para que o mesmo seja homologado na Anvisa.

Referências

- [1] DEITEL, H. M.; DEITEL P. J. Java - Como Programar. Porto Alegre: Prentice Hall, 2010.
- [2] SEBESTA, R. W. Conceitos de Linguagens de Programação, Bookman, Porto Alegre, 2003.
- [3] SOMMERVILLE, Ian. Engenharia de Software. 8ª ed. São Paulo: Pearson Addisison-Wesley, 2007
- [4] LADDAD, Ramnivas. AspectJ in Action – Practical Aspect-Oriented Programming. 2ª ed. Manning, 2003.
- [5] RESENDE, Antônio M. Pereira; SILVA, Claudiney Calixto. Programação Orientada a Aspectos em Java: Desenvolvimento de Software Orientado a Aspectos. Rio de Janeiro – Brasport, 2005.
- [6] BORGES, Felipe da Silva; FREITAS, André Luis Castro. Estudo de Caso Sobre Programação Orientada a Aspectos Utilizando Protocolo RMI. UFRG, 2008. Disponível em <<http://revistaseletronicas.pucrs.br/ojs/index.php/hifen/article/viewFile/4577/3466>> . Acesso em: 15/05/2017.
- [7] Conceitos Fundamentais de Banco de Dados, Disponível em <<http://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>> . Acesso em 16/05/17.
- [8] Oracle MySQL, Disponível em <<https://www.oracle.com/br/mysql/index.html>> Acesso em 16/05/17.
- [9] Entenda como funciona o protocolo TCP/IP. Disponível em <<https://www.citisystems.com.br/protocolo-t/cp-ip/>>, acesso em 16/05/2017.
- [10] J.F. Kurose, Keith W.R.. Redes de computadores e a internet- Uma abordagem top down. 5ª ed. São Paulo: Pearson, 2010.
- [11] TANENBAUM, Andrew S.. Redes de computadores. Rio de Janeiro: Elsevier, 2003.
- [12] Wireless Mesh Networks: Introdução, Disponível em <http://www.teleco.com.br/tutoriais/tutorialwmn/pagina_1.asp> Acesso em 14/05/2017.
- [13] OCHOA, M.; RAHIMI, R.; ZIAIE, B. Flexible Sensors for Chronic Wound Management. IEEE Reviews. In: Biomedical Engineering. v. 7, 2014.

- [14] National Pressure Ulcer Advisory Panel, European Pressure Ulcer Advisory Panel and Pan Pacific Pressure Injury Alliance. Prevention and Treatment of Pressure Ulcers: Quick Reference Guide. Emily Haesler (Ed.). Cambridge Media: Osborne Park, Western Australia, 2014.
- [15] OKAMOTO, R. Fundamentação Teórica: Feridas. In: Universidade Aberta do SUS (UNA-SUS). Especialização em Saúde da Família, 2010.
- [16] Análise dos Stakeholders, Disponível em < <http://www.gestaodeprojeto.info/analise-dos-stakeholders> > Acesso em 17/05/2017.
- [17] O que é Regra de Negócio?, Disponível em < <http://www.ateomomento.com.br/o-que-e-regra-de-negocio/> > Acesso em 20/05/2017.
- [18] Três startups da região são selecionadas entre as finalistas em programa de aceleração, Disponível em < <http://g1.globo.com/minas-gerais/triangulo-mineiro/blog/startup-digital/post/tres-startups-da-regiao-sao-selecionadas-entre-finalistas-em-programa-de-aceleracao.html> > Acesso em 03/07/2017.
- [19] 1220-1994 - IEEE Trial Use Standard for Application and Management of the Systems Engineering Process.