UNIVERSIDADE ESTADUAL DA PARAÍBA

CAMPUS I

CENTRO DE CIÊNCIA E TECNOLOGIA

CURSO DE COMPUTAÇÃO

LUCAS MIRANDA DOURADO NUNES

SISTEMA DE GERENCIAMENTO DE HEURÍSTICAS DE USABILIDADE

CAMPINA GRANDE

2016

LUCAS MIRANDA DOURADO NUNES

SISTEMA DE GERENCIAMENTO DE HEURÍSTICAS DE USABILIDADE

Trabalho de Conclusão de Curso de Graduação em Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação

Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Daniel Scherer.

CAMPINA GRANDE

2016

N972s Nunes, Lucas Miranda Dourado.
Sistema de gerenciamento de heurísticas de usabilidade [manuscrito] / Lucas Miranda Dourado Nunes. - 2016.
64 p. : il. color.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2016.
"Orientação: Prof. Dr. Daniel Scherer, Departamento de Computação".

1. Heurísticas. 2. Usabilidade. 3. Banco de dados. I. Título.
21. ed. CDD 005.74

LUCAS MIRANDA DOURADO NUNES

SISTEMA DE GERENCIAMENTO DE HERÍSTICAS DE USABILIDADE

Trabalho de Conclusão de Curso em Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação.

Área de concentração: Engenharia de Software.

Aprovada em: 03 de novembro de 2016.

BANCA EXAMINADORA

Prof. Dr. Daniel Scherer (Orientador)
Universidade Estadual da Paraíba (UEPB)

Prof. Dra. Luciana de Queiroz Leal Gomes
Universidade Estadual da Paraíba (UEPB)

Prof. Dr. Ademar Virgolino da Silva Netto
Universidade Federal da Paraíba (UFPB)

AGRADECIMENTOS

Quero agradecer a meu Deus, que é digno de toda honra, glória e louvor. Sem ele não conseguiria concluir o curso.

Quero agradecer aos meus pais, Antônio Augusto e Sirlane Miranda, pelo amor dado a mim, ajuda financeiro, moral e pela confiança.

Quero agradecer aos meus irmãos, Isabela, Mateus e André pelo carinho e apoio diante de tantas dificuldades. A toda a família pelo carinho e atenção.

Quero agradecer aos meus amigos, Bruno Clementino, Fabio Dias, Lukas Teles, Luana Janaina, Melquisedec Andrade e Sidney Pimentel, minha segunda família. Pelos momentos que passamos juntos. Agradecer pela paciência, ajuda e por momentos inesquecíveis que com certeza não irei esquece-los.

Quer agradecer ao meu professor que me acompanhou por quase toda a graduação. Me ajudando nos projetos de pesquisa e também na orientação da monografia. Agradeço pelos conselhos e pela amizade que construímos durante a vida acadêmica.

Também não poderia deixar de agradecer a todos que diretamente ou indiretamente me ajudou durante os meus estudos.

“Numa xícara de café, pode-se colocar a beleza do mundo. Numa xícara de café, pode-se sentir o sabor amargo e doce da vida” – Jorge Amado

RESUMO

A avaliação de usabilidade é um dos processos mais importantes para o desenvolvimento de um sistema. Fazer uma avaliação heurística usabilidade de um sistema é uma das técnicas mais fácil de encontrar possíveis erros de interface e funcionamento. A avaliação heurística de usabilidade é uma das técnicas de baixo custo em relação as outras técnicas como teste de usabilidade. Para este trabalho de conclusão de curso foi desenvolvido o sistema de Armazenamento de Heurísticas de Usabilidade (SAHU). Ele servirá para registrar e gerenciar as heurísticas de usabilidade do laboratório em uma base de dados. A base de dados servirá como recurso para o sistema que fará a avaliação heurística no Laboratório de Usabilidade e Fatores Humanos (LUFH) que está localizado no NUTES – Núcleo de Tecnologias Estratégicas em Saúde. Com o sistema SAHU em funcionamento o processo de avaliação será mais rápido, pois a busca pela heurística adequada ficará à disposição dos avaliadores. Com maiores opções de heurísticas o responsável pela avaliação poderá escolher a heurística que mais se adequa aos objetivos da avaliação, possibilitando assim diminuir as chances de subjetividades nas avaliações. Para o desenvolvimento do sistema foi utilizada metodologia ágil, foi feito um estudo e pesquisa das heurísticas e estudo das tecnologias a serem utilizadas.

Palavras-Chave: Heurísticas, usabilidade, Banco de dados.

ABSTRACT

The usability evaluation is one of the most important processes for the development of a system. Making an assessment of a heuristic usability of the system is easier to find possible errors interface and operation techniques. The heuristic evaluation usability is one of the techniques of low cost compared to other techniques such as usability testing. For this course conclusion work was developed Heuristics Storage System Usability (SAHU). It will serve to register and manage the lab usability heuristics in a database. The database will serve as a resource for the system that will heuristic evaluation at the Laboratory of Usability and Human Factors (LUFH) which is located in Strategic Technologies Center for Health (NUTES). With SAHU system running the evaluation process will be fast as the search for appropriate heuristic will be available to evaluators. With greater options heuristics responsible for evaluation may choose heuristic that best suits the objectives of the evaluation, thus enabling reduce the chances of subjectivities in the ratings. For the development of the system was used agile methodology, it was made a study and research of heuristics and study of technologies to be used.

Keywords: Heuristics, usability, Database.

LISTA DE ILUSTRAÇÕES

Figura 1 - Ciclo do Scrum.....19
Figura 2 - Scrum solo.....20
Figura 3 - Limite do PostgreSQL.....23
Figura 4 - Arquitetura genérica em camadas 24
Figura 5 - Façade.....25
Figura 6 - Relacionamento entre as classes Tag e Conjunto 29
Figura 7 - Diagrama de caso de uso 31
Figura 8 - Diagrama de classe 31
Figura 9 - Exemplo de anotações.....32
Figura 10 - Diagrama de entidade e relacionamento.....32
Figura 11 - Views 33
Figura 12 - Diagrama de pacote.....33
Figura 13 - Telas do sistema.....35
Figura 14 - Paginação das telas.....36
Figura 15 - Relatório 36
Figura 16 - Editar Conjunto de heurística.....37

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
BSD	Berkeley Software Distribution
CRUD	Create, Remove, Update e Delete
HTML	HyperText Markup Language
IDE	Integrated Development Environment
LUFH	Laboratório de Usabilidade e Fatores Humanos
MVC	Model View Controller
MIT	Massachusetts Institute of Technology
NUTES	Núcleo de Tecnologia Estratégicas em Saúde
SAHU	Sistema de Armazenamento de Heurística de Usabilidade
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
UEPB	Universidade Estadual da Paraíba

SUMÁRIO

1. INTRODUÇÃO 11

1.1. OBJETIVOS 12

1.2. JUSTIFICATIVA 12

2. REVISÃO DE LITERATURA 14

2.1. USABILIDADE 14

2.1.1. AVALIAÇÃO DE INTERFACE 15

2.1.1.1. AVALIAÇÕES HEURÍSTICAS 16

2.2. SCRUM 18

2.3. OUTRAS FERRAMENTAS DE DESENVOLVIMENTO 21

2.3.1. VISUAL STUDIO COMMUNITY 21

2.3.2. PLATAFORMA .NET 21

2.3.3. RAZOR 22

2.4. POSTGRESQL 23

2.5. ARQUITETURA EM CAMADAS 23

2.6. PADRÃO FAÇADE 25

2.7. BOOTSTRAP 25

3. METODOLOGIA 27

4. SOLUÇÃO PROPOSTA – O SISTEMA SAHU 29

4.1. TAGS 29

4.2. CASO DE USO 30

4.3. DIAGRAMA DE CASO DE USO 30

4.4. DIAGRAMA DE CLASSE 31

4.5. DIAGRAMA DE ENTIDADE E RELACIONAMENTO 32

4.6. DIAGRAMA DE PACOTE 33

5. RESULTADOS 35

6. CONSIDERAÇÕES FINAIS 39
REFERÊNCIAS 40
ANEXO A 42
APÊNDICE A 50
APÊNDICE B 58

1. INTRODUÇÃO

Nos últimos vinte anos foram criadas inúmeras ferramentas, equipamentos, softwares e produtos que mudaram a humanidade em diversas áreas, como saúde, cultura, educação e lazer. Desde a crise do software, com a definição do termo Engenharia de Software, passando pelo amadurecimento dos processos de projeto e desenvolvimento de sistemas e a chegada dos padrões de projeto, tem-se uma evolução significativa da informática em si. Além disto, muito provável em função desta evolução da computação, o grupo de pessoas que se utilizam de sistemas informatizados (muitas vezes conhecidos como usuários) tem crescido de forma impressionante. Assim, por um lado temos um aumento na demanda por softwares e por consequência, a necessidade de criar formas de agilizar o processo de desenvolvimento. Então foram criadas metodologias de desenvolvimento, novos padrões de projeto, arquiteturas de software, definições de boas práticas de programação, dentre outras formas.

Por outro lado, a expansão do universo de usuários também leva a uma gama maior de níveis de conhecimento e fluências em tecnologias. Neste caso, mesmo que em certas áreas da engenharia software haja uma evolução significante, algumas áreas ainda não tiveram a mesma dinâmica, a exemplo, da área da usabilidade, principalmente por ser uma área nova, que tem ganhado espaço desde o início dos anos 90. Segundo Jacob Nielsen “Usabilidade é um atributo de qualidade que avalia a facilidade de uso da interface do usuário” (NIELSEN, 2012).

A usabilidade é extremamente importante para um sistema interativo. Ter um sistema com usabilidade é essencial para determinar se um produto tenha sucesso, ou seja, aceitável aos usuários. Para isso, existem formar para avaliar se um sistema seja aceitável ou não a uma determinada população de usuários.

A avaliação já faz parte do processo de design do sistema, porém sem os rigores pela busca da identificação dos problemas de usabilidade e que poderão gerar erros ao usuário, que podem ser desde má interpretação de um ícone, passando por situação que leva a um incidente ou mesmo até um acidente.

Existem vários métodos de avaliação de usabilidade, escolher qual usar dependerá dos recursos que estarão disponíveis aos avaliadores. Dentre as técnicas de avaliação, pode-se destacar: avaliação analítica, avaliação heurística e inspeção

por lista de verificação. A avaliação heurística é uma das técnicas de avaliação barata e rápida em relação ao teste de usabilidade em que é necessário um laboratório e equipamentos para a realização da avaliação. Ela utiliza especialistas em usabilidade que fazem uma inspeção/verificação no sistema baseada em padrões ou heurísticas de usabilidade.

Este trabalho de conclusão de curso está estruturado da seguinte forma: sessão 2 é feita uma revisão de literatura apresentado todos os conceitos e tecnologias que fez necessário para alcançar os objetivos; sessão 3 é apresentado a metodologia; sessão 4 é feita a discursão sobre o processo de desenvolvimento até o momento que alcançou o objetivo; sessão 5 é apresentado os resultados alcançados e finalmente a sessão 6 tem as considerações finais do trabalho.

1.1. OBJETIVOS

O objetivo deste trabalho de conclusão de curso é desenvolver um sistema web que permitirá gerenciar uma base de dados de heurísticas de usabilidade. A base de dados, que também deverá desenvolvida, será utilizada por outro sistema que será desenvolvido para realizar a avaliação de usabilidade. O site deverá estar disponível no ambiente de testes e também em todo ambiente de pesquisa e desenvolvimento do NUTES (Núcleo de Tecnologia Estratégicas em Saúde).

1.1.1. OBJETIVOS ESPECÍFICOS

- ✓ Implementar um sistema de armazenamento e gerenciamento de heurísticas de usabilidade;
- ✓ Criar uma base de dados para fornecer os recursos a outro sistema que fará a análise de heurísticas e avaliação de usabilidade;
- ✓ Criar um manual de instalação.

1.2. JUSTIFICATIVA

Para fazer uma avaliação de heurísticas de usabilidade é necessário que o responsável pela avaliação procure por diversas fontes (ex.: sites, autores, documentos); o que leva um consumo considerável de tempo para encontrar os dados desejados e para organizá-los de forma adequada, bem como problema de subjetividade na avaliação.

Visando diminuir a subjetividade na avaliação, os usuários poderão inserir uma descrição mais detalhada nas heurísticas, inserir exemplos, referencias ou imagens. Portanto os avaliadores poderão ter menos dúvidas ou diferenças de interpretação nos conceitos apresentados.

Com o Sistema de Armazenamento de Heurísticas de Usabilidade (SAHU) as bases de heurísticas ficarão concentradas em um único local, que será alimentado pelos participantes do laboratório. Com isto, espera-se que com o sistema evite o desperdício de tempo para buscar heurísticas, bem como forneça uma base para se chegar a uma categorização de heurísticas (visando reduzir a subjetividade).

Assim, partir do sistema será mais fácil:

- ✓ diminuir a subjetividade;
- ✓ identificar heurísticas que podem ser utilizadas para as avaliações;
- ✓ ter dados estatísticos das heurísticas, autores e área de aplicação das heurísticas;

2. REVISÃO DE LITERATURA

Para a execução do projeto, foi necessário levar em conta uma serie de conteúdos, a citar: área de usabilidade e avaliação heurística; metodologia de desenvolvimento *Scrum*; as ferramentas e recursos de desenvolvimento (Visual Studio Community, as linguagens C#, Razor) e o sistema de gerenciamento de banco de dados PostgreSQL. Foram apresentados também, os conceitos de arquitetura em camadas e o padrão Façade.

2.1. USABILIDADE

Segundo (NIELSEN, 2012) “usabilidade é um atributo de qualidade que avalia quão fácil uma interface de usuário é de usar. A palavra usabilidade também se refere a métodos para melhorar a facilidade de utilização durante o processo de concepção”.

Já para norma ABNT NBR IEC 62366, usabilidade é: “característica das interfaces operador-equipamento que estabelece eficácia, eficiência, facilidade de aprendizagem do operador e satisfação do operador” (ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2010).

Portanto, pode-se colocar que a usabilidade é a qualidade que se busca para o uso de um sistema interativo; sendo sistema interativo a relação entre usuário, tarefa e o contexto em que o sistema atua.

Um sistema que não possui uma boa usabilidade pode possuir problema(s) de interação, ocasionando ao usuário dificuldades na realização de alguma tarefa. Segundo (CYBIS, FAUST e BETIOL, 2010), os problemas identificados têm origem na ergonomia da interface, ou seja:

> “A ergonomia é a qualidade da adaptação de um dispositivo a seu operador e à tarefa que este realiza. A usabilidade se revela quando os usuários empregam o sistema para alcançar seus objetivos em um determinado contexto de operação, sendo caracterizado pelo nível de eficácia, eficiência e satisfação alcançado pelo usuário durante seu uso”.

Tem-se, portanto, que a usabilidade leva em conta a **eficácia,** que é a capacidade que o sistema tem em alcançar seus objetivos; a **eficiência**, que é a quantidade de recursos que o sistema solicita para alcançar os objetivos; e a

satisfação, que é a emoção que o sistema proporciona ao usuário quando se alcança os objetivos.

Usabilidade pode ser abordada tanto do ponto de vista do desenvolvedor quanto de aquisição. Pensando-se em desenvolvimento, a usabilidade pode se fazer presente em várias etapas do ciclo de vida de um sistema, desde o momento de projeto, passando por prototipação até para avaliações de protótipos finais (ex.: alfa teste). Já do ponto de vista de aquisição, a usabilidade pode ser utilizada como requisito de avaliação de adequação do sistema a um propósito (usuário, contexto e tarefa).

2.1.1. AVALIAÇÃO DE INTERFACE

A avaliação pode fazer parte do processo de desenvolvimento de um design do sistema ou do processo de aquisição de um sistema. Para tanto, os avaliadores coletam informações dos protótipos, telas, artefatos ou sistema. O foco da avaliação está tanto na usabilidade do sistema (ou seja, o quanto é fácil de usar) quanto na experiência do usuário ao interagir com o sistema (por exemplo, quão agradável foi a interação com o sistema) (ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2010).

Desta forma, é importante ressaltar que, mesmo que designers/analistas/programadores consigam utilizar o produto por eles projetado, não significa que qualquer usuário poderá utilizar (ROGERS, SHARP e PREECE, 2013). Pois, como afirma (NIELSEN, 2012):

> “A experiência do usuário abrange todos os aspectos da interação do usuário final. O primeiro requisito para uma experiência de usuário exemplar é atender às necessidades específicas do cliente, sem protestos ou incômodos. Em seguida vêm a simplicidade e a elegância, que produzem produtos que são uma alegria ter, uma alegria usar. ”

Assim, a partir de uma avaliação é possível verificar se um determinado produto é aceitável a uma população que utiliza para uma tarefa em um determinado contexto.

Segundo (ROGERS, SHARP e PREECE, 2013) as avaliações podem ser classificadas em três grandes categorias:

- ✓ **Ambiente controlados envolvendo usuários:** principal método de avaliação é o teste de usabilidade;
- ✓ **Ambiente natural envolvendo usuários:** sendo que o principal método são os estudos de campo, por exemplo, entrevista e observação;
- ✓ **Qualquer ambiente não envolvendo usuários:** tendo como principais métodos de avaliações: inspeção, heurística, percurso e método baseados em modelos e dados analíticos.

O presente trabalho envolve a última classificação (‘Qualquer ambiente não envolvendo usuários’) com o método de avaliação heurística.

2.1.1.1. AVALIAÇÕES HEURÍSTICAS

Segundo (CYBIS, FAUST e BETIOL, 2010):

> “As técnicas de verificação conhecidas como avaliações heurísticas enfocam principalmente a interface do sistema e se baseiam nos conhecimentos ergonômicos e nas experiências dos avaliadores que percorrem a interface ou seu projeto para identificar aspectos da interface que atrapalhem os usuários durante as interações”.

Ou seja, uma avaliação heurística representa o julgamento de valor sobre as qualidades das interfaces humano-computador. A avaliação é feita por especialistas em usabilidade, que examinam o sistema e identificam potenciais problemas de usabilidade.

Para identificarem os problemas, os especialistas baseiam-se em heurísticas ou padrões de usabilidade gerais, definidos por especialistas na área, tais como: Jacob Nielsen, Ben Shneiderman, Dominique Scapin e Christian Bastien.

Vantagens da avaliação:

- ✓ Baixo custo em relação a outras técnicas de avaliação como o teste de usabilidade em que é necessário um laboratório com computadores e câmeras;
- ✓ Avaliação rápida em relação a teste de usabilidade;
- ✓ Pode obter ótimos resultados, levando em consideração ao seu custo e os seus avaliadores;

- ✓ Não depende de usuários para avaliação;
- ✓ Não precisa de estrutura para avaliação (ex.: laboratório de usabilidade).

Dentre as desvantagens destacam-se:

- ✓ Necessária (e fortemente recomendável) que a avaliação seja realizada por especialistas em usabilidade;
- ✓ Os resultados dependem dos especialistas (e seu grau de expertise no assunto avaliado);
- ✓ subjetividade nas avaliações;
- ✓ Por consequência, dificuldade em se conseguir a reprodução de avaliações, com resultados equivalentes/similares.

Atualmente, principal estratégia para reduzir a influência da subjetividade dos resultados encontrados, implica na execução da avaliação por no mínimo três especialistas (ou cinco) (NIELSEN, 1992). Além disto, deve-se realizar uma reunião com todos os especialistas envolvidos na avaliação, na busca da consolidação dos resultados.

Desta forma, (CYBIS, FAUST e BETIOL, 2010) sugerem um plano de avaliação, que será apresentado a seguir.

2.1.1.1.1. PLANO DE AVALIAÇÃO HEURÍSTICA

Em seu livro, (CYBIS, FAUST e BETIOL, 2010), cita oito etapas para realizar uma avaliação heurística. Essas etapas são uma sugestão de plano de trabalho que pode diminuir a subjetividade, aumentar a abrangência de problemas identificados e evitar resultados equivocados. As etapas do plano de trabalho são:

a. Análise do contexto da avaliação.
b. Montagem da equipe de avaliadores;
c. Análise do contexto de operação do sistema;
d. Análise do conhecimento disponível;
e. Reunião de preparativos para a avaliação;
f. Execução da avaliação;
g. Redação do relatório, e;
h. Reunião de apresentação do relatório.

Na etapa ‘a’ são identificados os recursos disponíveis e os objetivos da avaliação. A partir da primeira etapa é que o responsável pela avaliação poderá definir a quantidade e expertise dos avaliadores especialistas (etapa ‘b’). A escolha dos avaliadores deverá ser analisada em função da experiência e competência na avaliação de sistemas iguais ou similares.

Na etapa ‘c’ os especialistas deverão analisar o contexto de operação do sistema por meio da existência de um documento de especificação. Caso não tenha o documento deverá ser realizada técnica de análise (questionários ou entrevistas).

Na etapa ‘d’ o gerente de avaliação, que é o responsável pela avaliação, deverá procurar o conhecimento sobre as qualidades esperadas para a interface e a usabilidade do sistema.

Na etapa ‘e’ é realizada uma reunião em que todo o conhecimento é compartilhado e uniformizado, sobre o contexto da avaliação e da operação do sistema. Serão definidos os critérios de avaliações e os cenários do uso do sistema.

Finalmente após os preparos, é realizada a avaliação (etapa ‘f’). E na etapa ‘g’ o chefe da equipe redige o relatório de avaliação. O relatório deverá conter os problemas identificados e as propostas de soluções sugeridas pelos avaliadores.

Para a conclusão da avaliação, na última etapa (‘h’), é realizada uma reunião com os responsáveis pela avaliação e os envolvidos com o projeto do sistema. O sistema SAHU servirá de apoio nas etapas ‘d’ á ‘g’.

2.2. SCRUM

Scrum é um *framework* utilizado para o desenvolvimento rápido e gestão do desenvolvimento de sistemas. O *Scrum* utiliza uma abordagem iterativa e incremental para entregar os resultados com uma frequência determinada, assim, reduzindo os possíveis riscos (SABBAGH, 2013).

O *Scrum* destaca quatro prerrogativas básicas por que os modelos tradicionais de desenvolvimento, por exemplo cascata, não funcionam. Destaca-se:

- ✓ Nem sempre os requisitos são bem compreendidos no início do projeto;
- ✓ Os usuários só sabem exatamente o que querem após ver as primeiras versões do sistema;
- ✓ Os requisitos mudam frequentemente durante o desenvolvimento, e

- ✓ Novas ferramentas e tecnologias tornam as estratégias de desenvolvimento imprevisíveis.

Pelos pontos apresentados anteriormente foi realizada uma reunião nos anos de 1990 com dezesseis líderes em uma estação de esqui nos Estados Unidos, em que discutiram novas ideias, metodologias e processos que permitissem ter um desenvolvimento mais ágil. Após está reunião foi criado o manifesto para o desenvolvimento Ágil de software. O manifesto valorizava:

- ✓ Indivíduos e interações mais que processos e ferramentas;
- ✓ Software em funcionamento mais que negociação abrangente;
- ✓ Colaboração com o cliente mais que negociação de contratos, e;
- ✓ Responder a mudança mais que seguir um plano.

Apesar de tornar o processo de desenvolvimento mais rápido o *Scrum* possui a desvantagem que a documentação é mínima. Segundo (SABBAGH, 2013) uma das vantagens do uso do Scrum são:

- ✓ Redução dos riscos do projeto;
- ✓ Maior qualidade no produto;
- ✓ Visibilidade progressiva do projeto;
- ✓ Aumento produtividade;
- ✓ Entregas frequentes.

O *Scrum* é um processo interativo, veja a Figura 1. Para o funcionamento do *Scrum* é necessário um *Product Owner* que é o responsável por definir, comunicar e manter a visão do projeto. Também existe o *time de Scrum* que é uma pequena equipe que possui membros de desenvolvedores mais o *Product Owner* que irá definir tudo sobre o sistema.

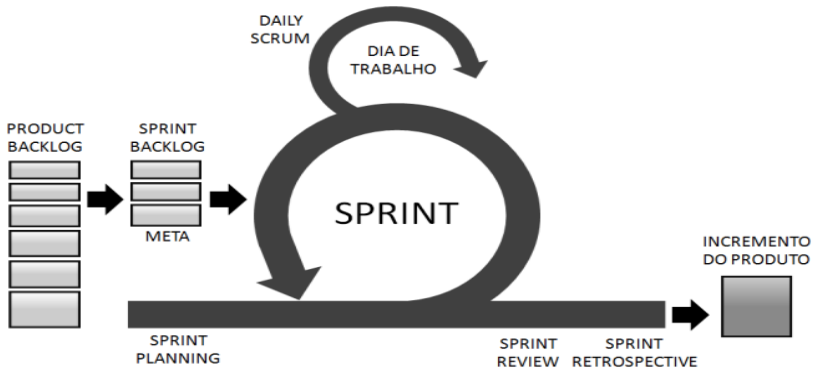

FIGURA 1 - CICLO DO SCRUM
Fonte: (SABBAGH, 2013)

No início do desenvolvimento o *Product Owner* inicia a partir da visão do sistema, cria uma lista de produtos que é incompleta e poderá aumentar a partir do desenvolvimento. Esta lista é chamada de *Product Backlog*. Os produtos que estão mais no topo da lista são os mais importantes, veja a Figura 1.

Após a criação do *Product Backlog*, o *time* pode começar o primeiro *Sprint*. Os *Sprints* são os ciclos de desenvolvimento, após a conclusão do primeiro ciclo de desenvolvimento o *time* define novos *Sprint backlog* que serão desenvolvidos durante o próximo *Sprint* e assim é feito até que o sistema é concluído.

Cada ciclo de *Sprint* pode durar de 2 a 4 semanas, no início de cada *Sprint* tem uma reunião onde são feitos os ajustes no desenvolvimento e são definidos novos *Sprint Backlog*. Existe outra reunião que é muito importante, *Day Scrum*. O *Day Scrum meeting* é realizado nos primeiros momentos do dia, nela são apresentadas as dificuldades, o que será feito e o que já foi feita. A duração do encontro é no máximo 15 minutos. O *Scrum Master* tem o papel de verificar se o processo está acontecendo como o esperado e resolver qualquer impedimento para a realização da atividade.

Como não existiu uma equipe de desenvolvedores, fez necessária o uso do *Scrum solo*. O *Scrum solo* é uma customização do *Scrum* voltada para o desenvolvimento individual de software.

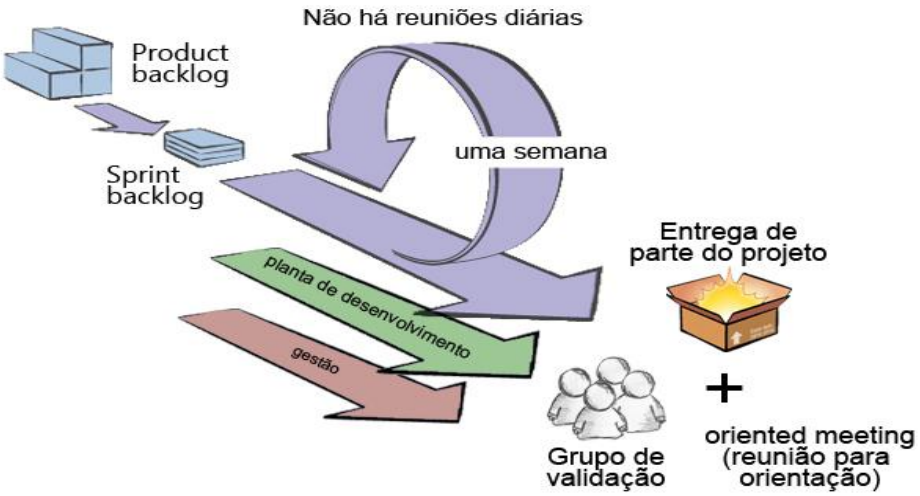

FIGURA 2 - SCRUM SOLO

Fonte: (PAGOTTO, FABRI, *et al.*)

Na Figura 2 é possível perceber alguma semelhança com o *Scrum*. O *Product backlog* e o *Sprint backlog* funcionam da mesma maneira. No *Scrum Solo* sugere

que os *Sprints* tenham durações de até uma semana e não aconteça as reuniões diárias (*Day Scrum meeting*) (PAGOTTO, FABRI, *et al.*). As reuniões dos *Sprints* no projeto foram realizadas em dois momentos: nas terças-feiras e sextas-feiras.

2.2.1. Motivo do uso

Com o Scrum solo o desenvolvimento do sistema proporcionou um desenvolvimento mais rápido, como menor impacto nas mudanças de requisitos do sistema, possibilitando assim uma maior eficiência e agilidade no desenvolvimento e entregas mais rápidas ao cliente.

2.3. OUTRAS FERRAMENTAS DE DESENVOLVIMENTO

A seguir serão expostos os conceitos sobre as tecnologias. Apresentando as ferramentas de desenvolvimento como ambiente de desenvolvimento (IDE), a plataforma de desenvolvimento da Microsoft e um framework criada pela Microsoft para exibir conteúdo em HTML nas páginas do cliente.

2.3.1. VISUAL STUDIO COMMUNITY

Segundo o site (MSDN MICROSOFT, 2016) o *Visual Studio Community* é uma IDE (*Integrated Development Environment*) com um conjunto de ferramentas de criação de software distribuído gratuitamente. O Visual Studio pode ser usado para vários tipos de soluções, de simples aplicativo para o *Windows store* e jogos a sistemas de grande porte e complexos.

Com a IDE Visual Studio poderá ser criada aplicativos para o Windows Phone, Android e IOS, sites e serviços para a web, aplicativos para plataformas e jogos e aplicativos para jogos (Xbox). Por padrão o Visual Studio oferece recursos para o desenvolvimento nas linguagens C#, C, C++, JavaScript, F# e Visual Basic.

2.3.1.1. Motivo do uso

Optou-se pelo Visual Studio Community 2015 (versão 14.0.25425), uma vez que a linguagem de desenvolvimento definida foi o C#.

2.3.2. PLATAFORMA .NET

O *framework* .NET é uma plataforma de desenvolvimento da Microsoft que tem o objetivo principal o desenvolvimento em Web Services. A linguagem C# faz parte do conjunto de ferramenta oferecida na plataforma .NET. Uma das vantagens de se utilizar o .NET são (LIMA e REIS, 2002):

- ✓ Independência de linguagem de programação;
- ✓ Reutilização de código legado;
- ✓ Tempo de execução compartilhado "*runtime*";
- ✓ Sistema de autoexplicativos e controle de versões;
- ✓ Simplicidade na resolução dos problemas

2.3.2.1. Motivo do uso

Optou-se pelo uso da plataforma .NET versão 4.5 por ser a versão estável e a linguagem utilizada é a C#. Também por ser comum nas pesquisas desenvolvidas no laboratório do NUTES utilizar a linguagem de programação C#.

2.3.3. RAZOR

Razor é o nome do mecanismo de exibição que a Microsoft introduziu no ASP.NET MVC 3. O Razor permite acrescentar códigos de servidor juntamente ao HTML de forma fácil. No Razor tem o Helpers que são classes que facilitam a criação das páginas dinâmicas e na geração dos elementos HTML (*HyperText Markup Language*), pode-se destacar os Helpers: Chart, WebGrid, Crypto, WebImage e WebMail.

O Razor permite inserir tanto códigos no 'lado' do servidor quanto na página de visualização do cliente (HTML). Pode se destacar que o Razor possui uma sintaxe limpa e concisa, fácil de aprender e permite o teste unitário. As páginas que utilizam a sintaxe Razor tem uma extensão especial (.cshtml ou .vbhtml). O servidor reconhece a extensão, executa o código no servidor e envia para o navegador.

2.3.3.1. Motivo do uso

Para o desenvolvimento do sistema optou-se na utilização da Microsoft ASP.NET Razor versão 3. O motivo foi do prazo de entrega do sistema e por que o desenvolvedor não tem conhecimentos de desenvolvimento em JavaScript.

2.4. POSTGRESQL

Segundo o site oficial (POSTGRESQL BRASIL, 2016) o PostgreSQL é um sistema de gerenciador de banco de dados (SGBD) objeto-relacional de código aberto. Pode ser usado em vários sistemas operacionais. Possui uma licença Open Source com as licenças MIT (POSTGRESQL, 2016). O site oficial da Comunidade Brasileira de PostgreSQL fornece uma tabela com os limites do banco de dados (Figura 3).

Limite	Valor
Tamanho Máximo do Banco de Dados	Ilimitado
Tamanho máximo de uma Tabela	32 TB
Tamanho Máximo de uma Linha	1.6 TB
Tamanho Máximo de um Campo	1 GB
Máximo de Linhas por Tabela	Ilimitado
Máximo de Colunas por Tabela	250–1600 dependendo do tipo de coluna
Máximo de Índices por Tabela	Ilimitado

FIGURA 3 - LIMITE DO POSTGRESQL

Fonte: (POSTGRESQL BRASIL, 2016)

De acordo com o site Open Source Initiative (OPEN SOURCE INITIATIVE, 2016) a licença MIT permite usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar e/ou vender cópias do Software, e permitir que as pessoas a quem o Software é fornecido façam isso, sujeitas às condições da não garantia de comercialização, os autores não serão responsáveis por qualquer reclamação, danos ou outras responsabilidades.

2.4.1.1. Motivo do uso

Para a escolha do banco de dados, optou-se na escolha do PostgreSQL na versão 9.6, pois dentre os SGBD's disponíveis (Mysql, SQL Server, Oracle) o PostgreSQL é o único que permite o seu uso de forma gratuita pois tem a licença MIT.

2.5. ARQUITETURA EM CAMADAS

Segundo (LARMAN, 2007), a ideia essencial de usar camadas é de organizar as estruturas lógicas com distintas responsabilidades. O uso da arquitetura em camadas estimula a organização da arquitetura do sistema em um conjunto de camadas coesas com fraco acoplamento entre eles.

Cada camada deverá possuir uma responsabilidade definida. A camada superior conhece apenas a camada imediatamente abaixo dela. Segundo (SILVEIRA, SILVEIRA, *et al.*) uso de muitas camadas pode gerar código de difícil manutenção e tornar a aplicação mais lenta.

Pode se destacar que o aumento do número de classes é uma das desvantagens. A alta coesão é uma vantagem que deve-se destacar, pois segundo o Larman: sistemas, subsistemas, classes ou métodos que tem baixa coesão sofrem com alguns problemas: difícil de se compreender; reutilizar; manter; e constantemente sofrem modificações.

Na Figura 4 tem-se um exemplo de uma arquitetura em camadas. A camada de baixo inclui os softwares de apoio ao sistema e o banco de dados. A próxima camada contém os componentes relacionados ao funcionamento da aplicação. A terceira camada está responsável pelo gerenciamento das interfaces de usuário e fornecimento de autenticação e autorização de usuário. A camada superior fornece os recursos da interface.

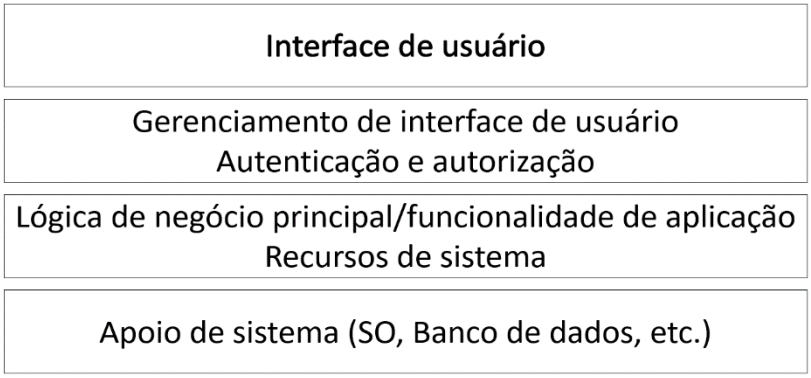

FIGURA 4 - ARQUITETURA GENÉRICA EM CAMADAS

Fonte: (SOMMERVILLE, 2011)

O uso de camadas do sistema ajuda a resolver alguns problemas, dos quais, pode-se destacar (LARMAN, 2007):

- ✓ Reduz o acoplamento e as dependências, melhorando a coesão, aumentando o potencial de uso e aumentando a clareza;
- ✓ A complexidade relacionada é encapsulada;
- ✓ As camadas inferiores contêm funções reusáveis;
- ✓ Algumas camadas podem ser distribuídas, por exemplo a camada de domínio;

- ✓ O desenvolvimento em equipe é mais fácil por causa da divisão lógica.

2.5.1.1. Motivo do uso

Utilizar um padrão de projeto foi uma forma de organizar os arquivos do sistema e deixar o código para os futuros desenvolvedores com maior clareza de como está estruturado o sistema.

2.6. PADRÃO FAÇADE

O padrão façade ou fachada é uma maneira de minimizar a comunicação e as dependências entre subsistemas. Uma maneira de atingir o objetivo é criar uma fachada, o qual fornece uma interface única entre os recursos e os subsistemas. Na Figura 5 tem-se o funcionamento de um sistema com e sem o façade.

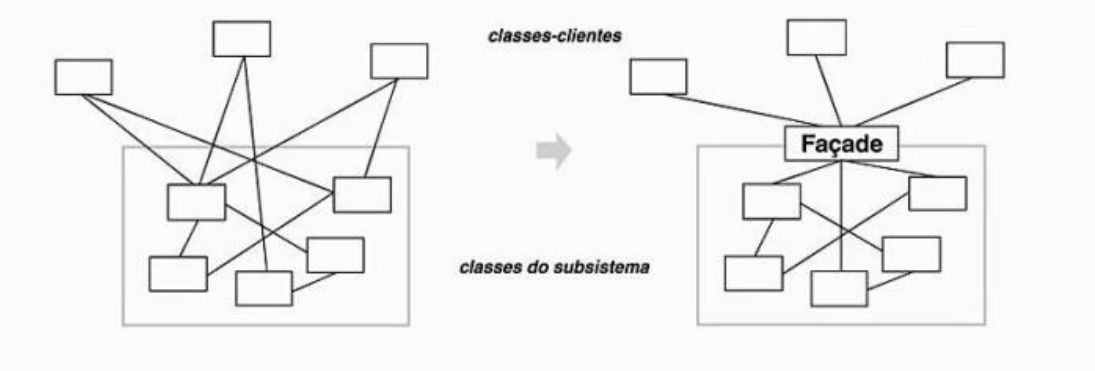

FIGURA 5 - FAÇADE
Fonte: (GAMMA, HELM, *et al.*, 2000)

Segundo (GAMMA, HELM, *et al.*, 2000) Façade deve ser usado quando:

- ✓ deseja fornecer uma interface simples para um subsistema complexo;
- ✓ deseja promover independência e portabilidade dos subsistemas;
- ✓ deseja estruturar os subsistemas em camadas.

2.6.1.1. Motivo do uso

Foi decidido utilizar o padrão Façade como um método de tornar o sistema com maior portabilidade e escalável.

2.7. BOOTSTRAP

O Bootstrap é um *framework* distribuído gratuitamente (*Open Soure*) mantido e distribuído pelo GitHub (BOOTSTRAP, 2016). Com o Bootstrap o desenvolvimento *Front-End* torna-se mais rápido e fácil. Ele possui um conjunto de ferramentas criadas para facilitar o desenvolvimento de sites.

É compatível com o HTML5 e CSS3. O *framework* possibilita a criação de sites responsivos, ou seja, as telas poderão se adaptar a diferentes resoluções de telas ou dispositivos como: *notebook*, celular ou *tablet*.

Dentre as vantagens pode destacar:

- ✓ documentação detalhada e de fácil entendimento;
- ✓ é otimizado para desenvolvimento de *layouts* responsivos;
- ✓ funciona na maioria dos navegadores atuais (ex.: Chrome, Safari, Firefox, IE, Opera);

As desvantagens destacam-se:

- ✓ Os códigos deverão seguir o padrão de desenvolvimento do Bootstrap;
- ✓ As telas ficam com o padrão de cores estabelecidas pelo framework.

2.7.1.1. Motivo do uso

Por ser o framework popular, gratuito e o desenvolvedor possuir experiência no uso, o Bootstrap foi uma solução viável para o projeto.

3. METODOLOGIA

A pesquisa realizada para a construção do presente trabalho pode ser classificada como bibliográfica, visto que houve a necessidade de realizar um estudo de análise do sistema e estudo de soluções para os problemas identificados.

Considerando que o objetivo do trabalho é: desenvolver um sistema que armazene e gerencie as bases de heurísticas de usabilidade; foram realizadas pesquisas para fornecerem o embasamento necessário para se alcançar a meta almejada.

Para tanto seguiu-se os seguintes passos:

- ✓ Pesquisa sobre heurísticas de usabilidade;
- ✓ Levantamento de requisitos iniciais;
- ✓ Definir o processo de desenvolvimento;
- ✓ Pesquisar sobre as tecnologias a serem utilizadas;
- ✓ Desenvolvimento;
- ✓ Criação do manual de instalação;

3.1. PESQUISA SOBRE HEURÍSTICAS DE USABILIDADE

Para o entendimento do sistema foi necessária a pesquisa de heurísticas de usabilidade. Foram utilizadas três bases de heurísticas, afim de identificar as informações que seriam necessárias para a criação do sistema. As heurísticas utilizadas foram as 10 heurísticas de Nielsen (NIELSEN, 2012), heurística de Shneiderman conhecidas como as Oito regras de ouro (SHNEIDERMAN, 2010) e heurísticas de Zhang (ZHANG, TODD, *et al.*, 2003). As heurísticas citadas estão no anexo A.

3.2. LEVANTAMENTO DE REQUISITOS INICIAIS

Para o levantamento dos requisitos do sistema, foram necessárias reuniões com o cliente (prof. Daniel Scherer) para o entendimento do que seria o sistema. Para a modelagem do sistema é importante conhecer o local em que será utilizado o sistema e os usuários.

3.3. DEFINIR O PROCESSO DE DESENVOLVIMENTO

Considerando o objetivo de desenvolver um sistema para gerenciar heurísticas, era necessária a definição do processo de desenvolvimento que seria adotado.

Os processos de desenvolvimento rápido de software são concebidos para produzir software rapidamente e que sejam úteis. O software foi desenvolvido em uma metodologia Scrum. Também para o desenvolvimento fez necessário a criação de uma documentação mínima do sistema (disponível no SharePoint, no site Usabilidade na pasta SAHU).

3.4. PESQUISAR SOBRE AS TECNOLOGIAS A SEREM UTILIZADAS

Para a implementação do sistema foi decidido o uso da plataforma de desenvolvimento da Microsoft .NET com o foco principal no desenvolvimento em serviços *Web Service*. Uma das vantagens do uso da plataforma .NET é “permitir que as aplicações, sejam elas desktop, web ou mobile se comuniquem e troquem dados de forma simples e transparente, independente do sistema operacional ou da linguagem de programação” (LIMA e REIS, 2002). Fez necessário a escolha do sistema de gerenciamento de banco de dados (PostgreSQL).

3.5. DESENVOLVIMENTO

Para o desenvolvimento foi feita a análise do sistema, sendo criado um documento do sistema em que tem escrito tudo o que o cliente deseja. Foi feita uma pesquisa de três bases de heurísticas de usabilidade para identificar quais as informações básicas das heurísticas. Após a identificação das informações foi feita a modelagem do sistema e apresentada ao cliente. Em seguida foi iniciada a implementação do sistema.

3.6. CRIAÇÃO DO MANUAL DE INSTALAÇÃO

Foi feito um manual de instalação e configuração do site que ficará disponibilizado no SharePoint do NUTES.

4. SOLUÇÃO PROPOSTA – O SISTEMA SAHU

A seguir serão apresentados: Tags, lista dos casos de uso, diagrama de caso de uso. Para mais detalhes veja o APÊNDICE B.

4.1. TAGS

Para facilitar na busca das heurísticas, foi necessário criar palavras-chaves ou "*Tags*" para rotular/classificar a heurística na área de aplicação (ex.: web, mobile, saúde e hardware). O usuário deverá cadastrar as Tags com o nome da área de aplicação da heurística, por exemplo, as heurísticas de Zhang foram criadas para aplicação em dispositivos médicos, então ao cadastrar uma Tag, o usuário poderá cadastrar 'Dispositivos médicos'. As Tags são importantes, pois através delas que os usuários ou responsável pela avaliação irão pesquisar as heurísticas.

O sistema permite adicionar quantas Tags forem necessárias. Desta forma, quando o usuário for pesquisar no site irá apresentar as heurísticas de acordo com a ordem de inserção das Tags.

Para facilitar o entendimento suponha que o site possui duas heurísticas cadastradas. A heurísticas A tem as Tags: Prontuário eletrônico e Web. A heurística B possui: Web e Saúde. Caso o usuário queira fazer uma busca nas heurísticas com a Tag Web, a ordem de apresentação das heurísticas é determinada pelas Tags que estão com prioridade maior para a menor. Em seguida as que aparecerem com a segunda prioridade. Na situação apresentada seria: Heurísticas B depois A.

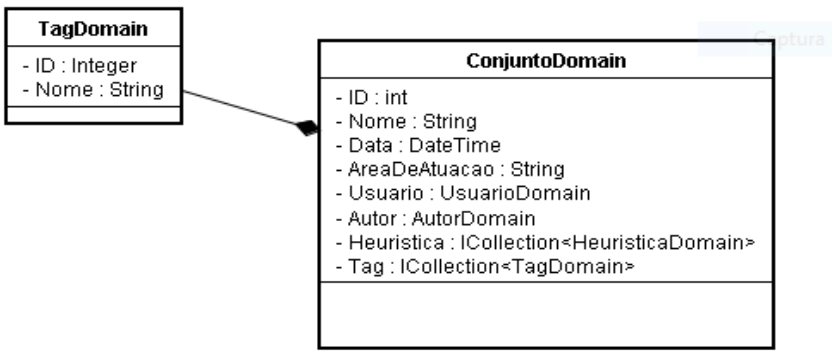

FIGURA 6 - RELACIONAMENTO ENTRE AS CLASSES TAG E CONJUNTO

Fonte: Autor

Na Figura 6 é apresentado o relacionamento de composição entre as classes: ConjuntoDomain e TagDomain. A classe ConjuntoDomain tem um identificador (ID),

nome, data, área de atuação (descrição), autor, uma lista de heurística e uma lista de Tags. A classe TagDomain tem um identificador (ID) e o nome da Tag.

4.2. CASO DE USO

Na Tabela 1 é apresentada os principais casos de uso do sistema SAHU. Para mais detalhes dos casos de uso veja o APÊNDICE B.

Código	Nome	Atores
UC01	Cadastrar usuário	Administrador
UC02	Remover usuário	Administrador
UC03	Atualizar usuário ou mudar de perfil	Administrador
UC04	Inserir heurística	Administrador; especialista; capacitação
UC05	Remover heurística	Administrador
UC06	Atualizar heurística	Administrador; especialista; capacitação
UC07	Cadastrar autor	Administrador; especialista; capacitação
UC08	Remover autor	Administrador; especialista; capacitação
UC09	Atualizar autor	Administrador; especialista; capacitação

TABELA 1 - CASOS DE USO

4.3. DIAGRAMA DE CASO DE USO

Na Figura 7 é apresentado um diagrama de caso de uso. Todas as funcionalidades apresentadas no diagrama deverão ser registradas os logs registrando a atividade realizada, usuário e a data da modificação do sistema.

No diagrama é apresentado todos os perfis de usuário do sistema (atores) (administrador, especialista, técnico e capacitação). As funcionalidades apresentadas no diagrama deverão se comportar de acordo com o tipo de perfil de cada usuário. Por exemplo, todas as atividades feitas pelo usuário do perfil capacitação não deverá estar disponível para os outros perfis como o administrador; especialista ou técnico.

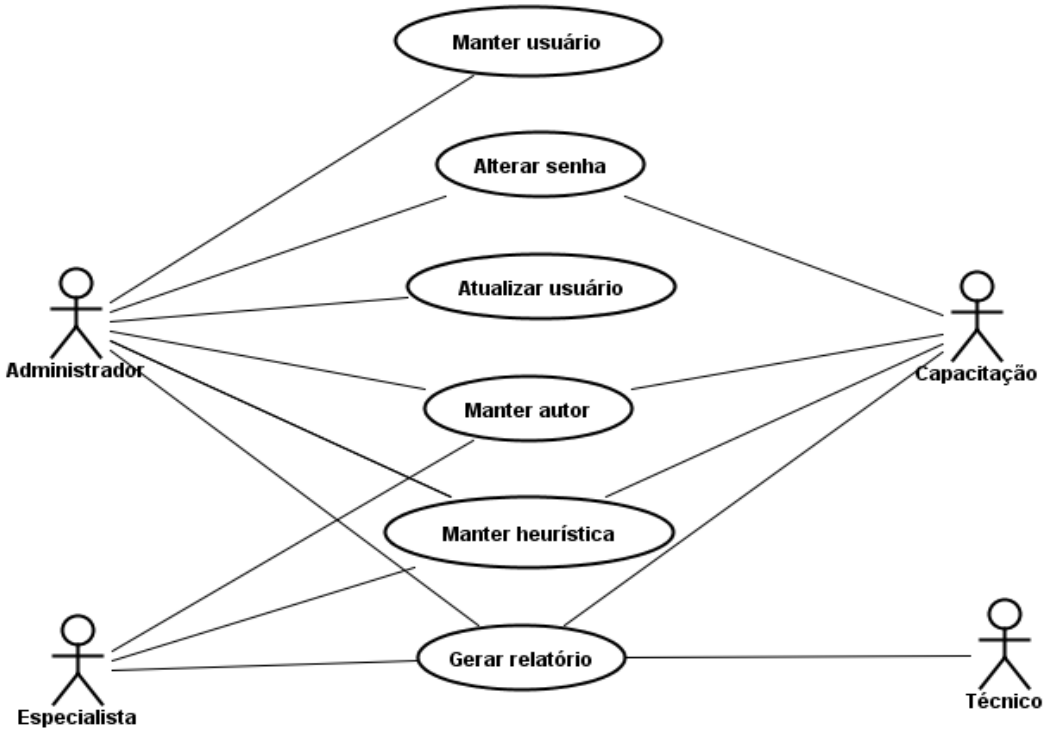

FIGURA 7 - DIAGRAMA DE CASO DE USO

Fonte: autor

4.4. DIAGRAMA DE CLASSE

A seguir tem-se o diagrama de classe da camada *Domain* do sistema com todas as classes de lógica do sistema.

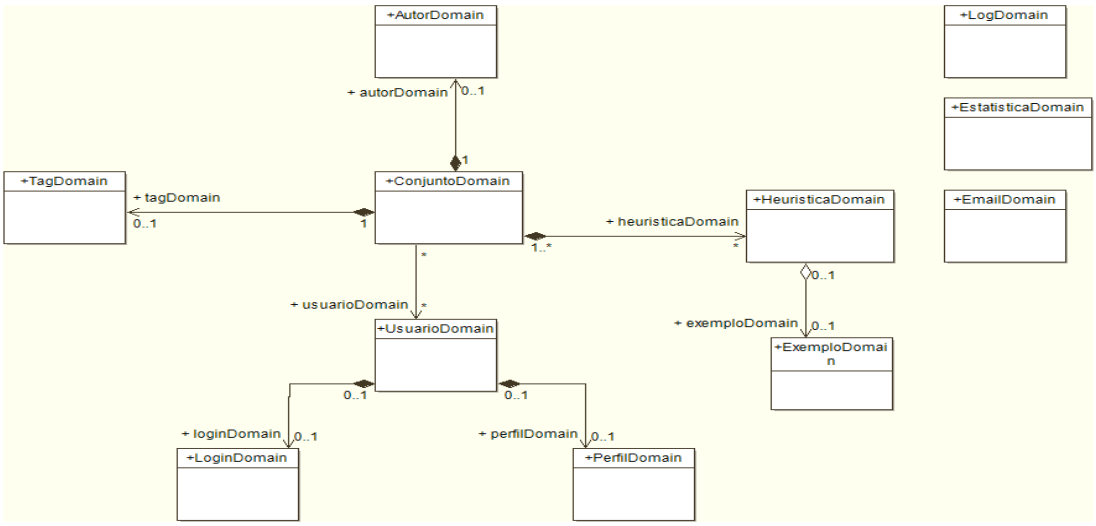

FIGURA 8 - DIAGRAMA DE CLASSE

Fonte: Autor

Na Figura 9 tem uma classe de UsuárioDomain. Nela apresenta anotações para a validações dos campos dos formulários, por exemplo o nome do usuário não poderá exceder a 100 caracteres. Com os annotations fica mais fácil de dar o feedback ao usuário.

```
1
2   using System.ComponentModel;
3   using System.ComponentModel.DataAnnotations;
4
5   namespace Heuristica.Domain.Modelos
6   {
7       public class UsuarioDomain
8       {
9           public int ID { get; set; }
10          [DisplayName("Nome")]
11          [StringLength(100, ErrorMessage = "O campo Nome permite no máximo 100 caracteres!")]
12          public string Nome { get; set; }
13          [DisplayName("Nome")]
14
15          public string Telefone { get; set; }
16          public virtual PerfilDomain Perfil { get; set; }
17          public virtual LoginDomain Login { get; set; }
18
19      }
20  }
21
```

FIGURA 9 - EXEMPLO DE ANOTAÇÕES

4.5. DIAGRAMA DE ENTIDADE E RELACIONAMENTO

Na Figura 10 é apresentado o diagrama de entidade e relacionamento.

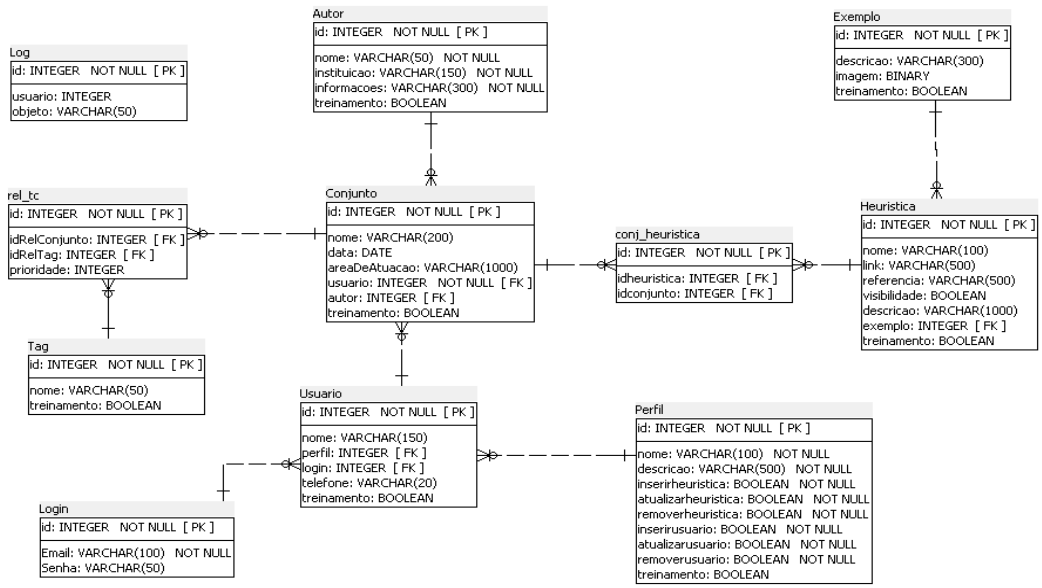

FIGURA 10 - DIAGRAMA DE ENTIDADE E RELACIONAMENTO

Fonte: Autor

Para o acesso aos dados fez necessários a criação de Views, veja Figura 11. Uma view é uma tabela única derivada de outras tabelas ou de outra view anteriormente definidas. A view não existe fisicamente, mas é considerada uma tabela virtual segundo (ELMASRI e NAVATHE, 2011).

Um dos motivos da criação das views foram: O código fica mais limpo, não precisa criar todas as vezes o mesmo código SQL para acessar os dados, é selecionado apenas informações necessárias e é mais seguro utilizar views.

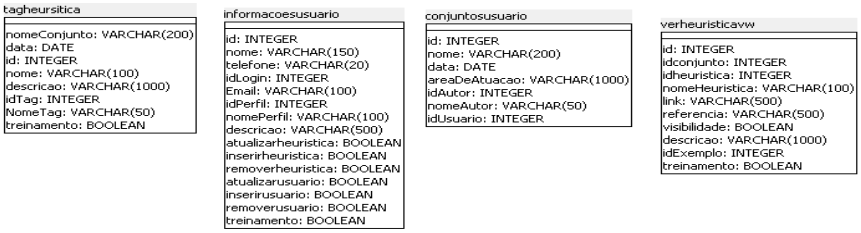

FIGURA 11 - VIEWS

Fonte: Autor

4.6. DIAGRAMA DE PACOTE

A Figura 12Figura 12 - Diagrama de pacote é apresentado o diagrama de pacotes, nela pode ser observado a estrutura em quatro camadas. As camadas superiores têm uma dependência das camadas inferiores e as camadas inferiores não tem acesso as camadas superiores a elas.

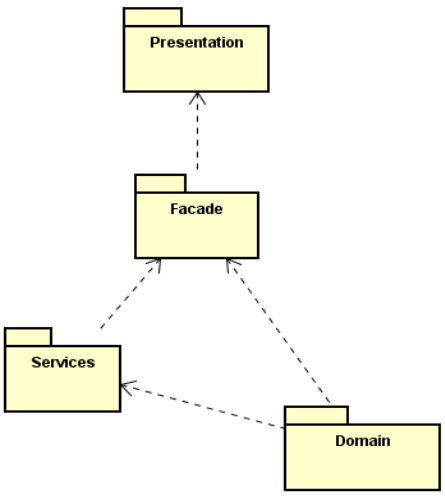

FIGURA 12 - DIAGRAMA DE PACOTE

Fonte: Autor

A camada Domain é definida as classes de modelo. O pacote Services são criadas as conexões de banco de dados e as operações CRUD (Create, Read, Update e Delete). A camada Façade é onde é aplicado o padrão Façade e Presentation onde são criadas as telas dos usuários.

4.7. DESENVOLVIMENTO DO SISTEMA

Após identificar os requisitos do sistema, casos de uso, criar os diagramas de classe e diagrama de caso de uso foi escolhido a linguagem de programação e o sistema de banco de dados.

O cliente, no caso o prof. Daniel Scherer, sugeriu que utilize o C# pois a plataforma .NET possibilita utilizar outras linguagens de programação como C, C++, Visual Basic, JScript, Cobol e Pascal. Outro motivo em utilizar o C# é a facilidade, flexibilidade e por ser a linguagem nativa da plataforma .NET. Também pelo motivo que o NUTES tem uma grande quantidade de colaboradores utilizando o C#.

O *framework* .NET é uma plataforma de desenvolvimento da Microsoft que tem como foco o desenvolvimento de serviços Web (Web Services). A linguagem C# é linguagem principal oferecida pela .NET. Assim a plataforma .NET oferece um conjunto de ferramentas/recursos para o desenvolvimento e além de surgir como uma linguagem simples, robusta, orientada a objeto, fortemente tipada e altamente escalável.

Além do C# foi utilizada o Razor que é um gerador de códigos HTML. Ele foi escolhido por ser fácil de aprendizado e por não necessitar utilizar a linguagem JavaScript para geração de gráficos.

Para a formatação de cores, fonte e possibilitar um design responsivo, foi utilizada o *Framework* Bootstrap.

Para a escolha do SGDB foi feita uma pesquisa nas licenças. O único dos quatro grandes e populares banco de dados (Oracle, SQL Server, MySQL e PostgreSQL) que permitia a utilização dos serviços gratuitamente foi o PostgreSQL.

5. RESULTADOS

Nesta sessão serão apresentados os resultados alcançados.

5.1. Sistema SAHU

Na Figura 13 são apresentadas três telas em dispositivos diferentes (Celular, Tablet e notebook). O sistema foi desenvolvido para ser acessado em qualquer tipo de dispositivo (ex.: notebook, celular ou tablet), ou seja, que seja responsivo.

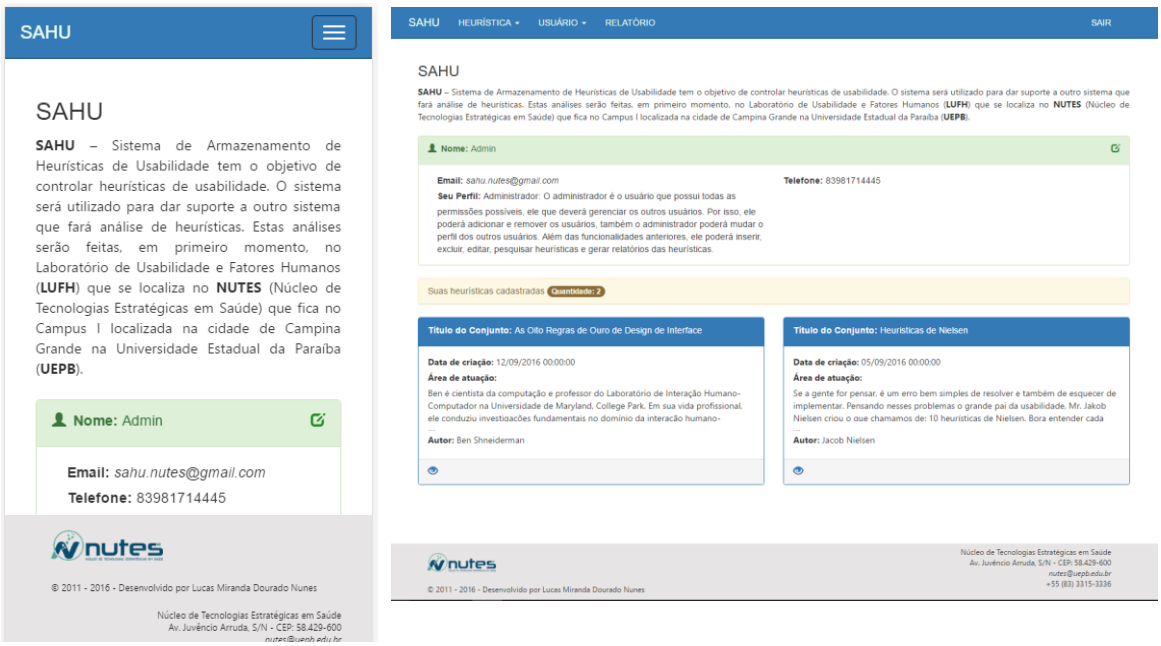

FIGURA 13 - TELAS DO SISTEMA

Fonte: Autor

Para proporcionar uma melhor experiência ao usuário fez necessário pôr as tabelas do sistema paginação como mostra na Figura 14. Para poder gerar a paginação fez necessário o uso da biblioteca “PagedList.dll” fornecida gratuitamente pela Microsoft.

FIGURA 14 - PAGINAÇÃO DAS TELAS

Fonte: Autor

Na Figura 15 é apresentado o relatório do usuário. Os gráficos apresentados foram gerados pelo Helpers Chart. Os gráficos apresentados na Figura 15 são imagens gerados no servidor e enviado a tela do usuário.

FIGURA 15 - RELATÓRIO

Fonte: Autor

Além das figuras apresentadas anteriormente, existe uma opção de gerar arquivos em PDF, como mostrado na Figura 15 na parte superior direito da tela.

Todos os perfis dos usuários foram inseridos, evitando que o administrador faça posteriormente a disponibilização do sistema. Os perfis adicionados foram definidos na sessão “4.3. Tipos de usuários”. Mas caso o administrador, assim desejar poderá modificar as permissões dos perfis dos usuários.

O sistema foi instalado em um computador que ficará a disponível do laboratório LUFH. Foi adicionada três bases de heurísticas (Nielsein, Zhang e Shneiderman).

Na Figura 16 é apresentada uma mensagem de feedback ao usuário. Estas mensagens tem o objetivo de informar aos usuários os passos que deverão fazer para concluir a tarefa ou informar se alguma tarefa foi realizada com sucesso ou não.

Na Figura 16 é apresentada uma página da edição de um conjunto de heurísticas. Na página é apresentada uma mensagem (uma barra de cor vermelho) informando que não foi selecionada nenhuma heurística para adicionar ao conjunto de heurísticas.

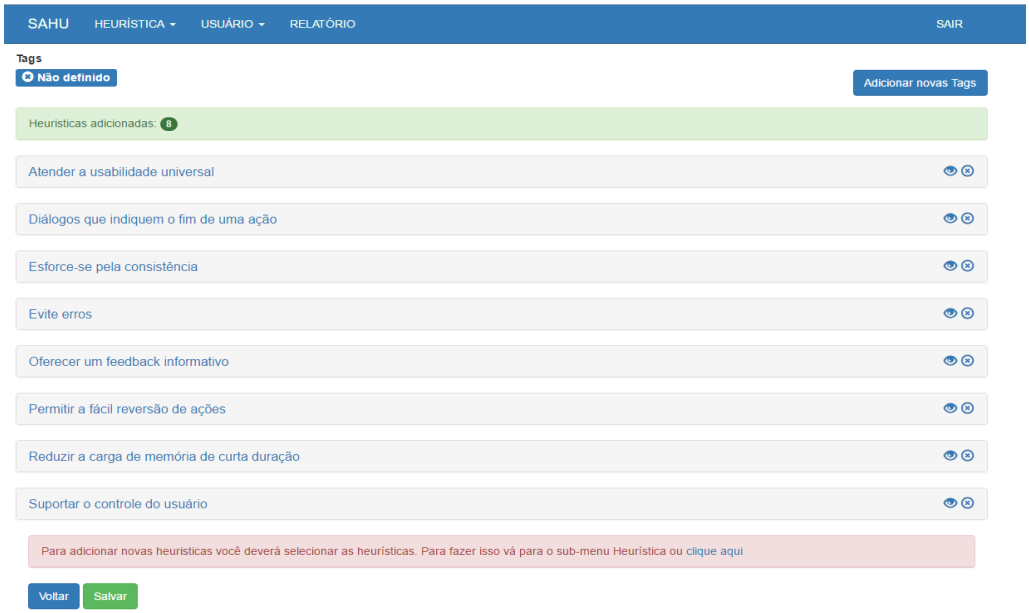

FIGURA 16 - EDITAR CONJUNTO DE HEURÍSTICA

Fonte: Autor

5.2. Manual de instalação

Foi criado um manual de instalação e configuração do site. Apresentando os softwares necessários para o funcionamento do sistema e as configurações de cada software, segue no apêndice A.

6. CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso teve como objetivo implementar uma solução para registrar e gerenciar heurísticas de usabilidade. Além do desenvolvimento do sistema foi criada uma base de dados que será utilizada por outro sistema e, finalmente, foi criado um manual de instalação e configuração.

Para alcançar os objetivos do trabalho, foi utilizado uma metodologia de desenvolvimento que pudesse agilizar o desenvolvimento do sistema e que seja flexível a mudanças de requisitos. Para a implementação do sistema foram coletadas informações do cliente. Criando uma documentação mínima do sistema. Podendo assim inicializar o processo de desenvolvimento.

Mas mesmo que o sistema esteja fazendo o que o cliente pediu. Existem alguns recursos ou processos de desenvolvimento do sistema que poderiam ser aperfeiçoados, destaca-se:

- utilizar ferramentas de criação de relatórios;
- o desempenho do sistema poderia melhorar, a partir de tecnologias, tais como: json, ajax ou JavaScript;
- algumas páginas do site deverá ser feita um estudo para aperfeiçoar (Ex.: pagina de edição dos conjuntos de heurísticas);
- poderia ser aperfeiçoada as mensagens de feedback aos usuários;

Como trabalhos futuros, poderia ser feita:

- avaliação heurística no site SAHU para poder encontrar possíveis problemas de usabilidade;
- instalação do site no servidor do NUTES;
- aperfeiçoamento na segurança das senhas inserindo a criptografia;
- Criar procedures no SGBD para a inserção, remoção e edição. Assim evitaria que os comandos inseridos na base de dados;
- procurar softwares para geração de arquivos em PDF;
- utilizar novas tecnologias como o JavaScript e Ajax;
- uso de sistemas inteligentes para a inserção das heurísticas.

REFERÊNCIAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Produtos para a saúde - Aplicação da engenharia de usabilidade a produtos para a saúde**. ABNT. [S.l.]. 2010.

BOOTSTRAP. Bootstrap. **Bootstrap**, 2016. Disponivel em: <http://getbootstrap.com/>. Acesso em: 27 Setembro 2016.

CYBIS, W.; FAUST, R.; BETIOL, A. H. **Ergonomia e Usabilidade:** Conhecimentos, Métodos e Aplicações. 2ª. ed. São Paulo: Novatec, v. I, 2010. 202-252 p. Acesso em: 23 Agosto 2016.

ELMASRI, R.; NAVATHE, S. B. **Sistema de banco de dados**. 6. ed. São Paulo: Pearson, 2011.

GAMMA, E. et al. **Padrões de projeto:** soluções reutilizáveis de software orientado a objeto. 1ª. ed. Porto Alegre: Bookman, 2000.

LARMAN, C. **Utilizando UML e padrões:** Uma introdução à análise e ao projeto orientado a objeto e ao desenvolvimento iterativo. 3ª. ed. Porto Alegre: Bookman, 2007.

LIMA, E.; REIS, E. **C# e.Net para desenvolvedores**. Rio de Janeiro: Campus, 2002.

MSDN MICROSOFT. Visual Studio IDE. **Developer Network**, 2016. Disponivel em: <https://msdn.microsoft.com/pt-br/library/dn762121.aspx>. Acesso em: 23 Agosto 2016.

NIELSEN, J. Finding usability problems through heuristic evaluation. **CHI '92 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**, New York, 3 Junho 1992. 373-380. Disponivel em: <http://dl.acm.org/citation.cfm?id=142834>.

NIELSEN, J. Usability 101: Introduction to Usability. **NN/G Nielsen Norman Group**, 4 Janeiro 2012. Disponivel em: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>. Acesso em: 15 Março 2016. Tradução feita pelo autor.

OPEN SOURCE INITIATIVE. The MIT License (MIT). **Open Source Initiative**, 2016. Disponivel em: <https://opensource.org/licenses/mit-license.php>. Acesso em: 24 Agosto 2016.

PAGOTTO, T. et al. Scrum Solo - Processo de software para desenvolvimento individual. **11ª Conferencia Ibérica de Sistemas y Tecnologías de Información**.

POSTGRESQL. License. **PostgreSQL**, 2016. Disponivel em: <https://www.postgresql.org/about/licence/>. Acesso em: 24 Agosto 2016.

POSTGRESQL BRASIL. Sobre o PostgreSQL. **Comunidade Brasileira de PostgreSQL**, 2016. Disponivel em: <https://www.postgresql.org.br/sobre>. Acesso em: 24 Agosto 2016.

ROGERS, Y.; SHARP, H.; PREECE, J. **Design de interação:** além da interação humano-computador. Tradução de Isabela Gasparini. 3ª. ed. Porto Alegre: Bookman, v. I, 2013. Acesso em: 26 Agosto 2016.

SABBAGH, R. **Scrum:** Gestão Ágil para projetos de sucesso. 1ª. ed. São Paulo: Casa do código, v. I, 2013.

SHNEIDERMAN, B. The Eight Golden Rules of Interface Design. **Ben Shneiderman**, 2010. Disponivel em: <https://www.cs.umd.edu/users/ben/goldenrules.html>. Acesso em: 26 Setembro 2016.

SILVEIRA, P. et al. **Introdução à arquitetura e Design de software - Uma visão sobre a plataforma java**. São Paulo: Campus.

SOMMERVILLE, I. **Engenharia de Software**. 9ª. ed. São Paulo: Pearson, 2011. Acesso em: 17 Agosto 2016.

ZHANG, J. et al. Using usability heuristics to evaluate patient safety of medical devices. **Journal of Biomedical Informatics**, 4 Setembro 2003. 23-30. Disponivel em: <http://www.sciencedirect.com/science/article/pii/S1532046403000601>.

ANEXO A

Neste anexo, são apresentadas as heurísticas de Nielsen, Shneiderman e Zhang.

A. HEURÍSTICA DE NIELSEN

Título: 10 Usability Heuristics for User Interface Design

1. Visibility of system status:

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world:

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedon:

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards:

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention:

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

(Read full article on preventing user errors.)

6. Recognition rather than recall:

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

(Read full article on recognition vs. recall in UX.)

7. Flexibility and efficiency of use:

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design:

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors:

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation:

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

B. HEURÍSTICA DE SHNEIDERMAN

Título: The Eight Golden Rules of Interface Design

1. Strive for consistency

Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent color, layout, capitalization, fonts, and so on should be employed throughout. Exceptions, such as required confirmation of the delete command or no echoing of passwords, should be comprehensible and limited in number.

2. Cater to universal usability

Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Novice to expert differences, age ranges, disabilities, and technological diversity each enrich the spectrum of requirements that guides design. Adding features for novices, such as explanations, and features for experts, such as

shortcuts and faster pacing, can enrich the interface design and improve perceived system quality.

3. Offer informative feedback

For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. Visual presentation of the objects of interest provides a convenient environment for showing changes explicitly.

4. Design dialogs to yield closure

Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives operators the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions. For example, e-commerce web sites move users from selecting products to the checkout, ending with a clear confirmation page that completes the transaction.

5. Prevent errors

As much as possible, design the system such that users cannot make serious errors; for example, gray out menu items that are not appropriate and do not allow alphabetic characters in numeric entry fields. If a user makes an error, the interface should detect the error and offer simple, constructive, and specific instructions for recovery. For example, users should not have to retype an entire name-address form if they enter an invalid zip code, but rather should be guided to repair only the faulty part. Erroneous actions should leave the system state unchanged, or the interface should give instructions about restoring the state.

6. Permit easy reversal of actions

As much as possible, actions should be reversible. This feature relieves anxiety, since the user knows that errors can be undone, and encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data-entry task, or a complete group of actions, such as entry of a name-address block.

7. Support internal locus of control

Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don't want surprises or changes in familiar behavior, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result.

8. Reduce short-term memory load

Humans' limited capacity for information processing in short-term memory (the rule of thumb is that we can remember "seven plus or minus two chunks" of information) requires that designers avoid interfaces in which users must remember information from one screen and then use that information on another screen. It means that cell phones should not require re-entry of phone numbers, web-site locations should remain visible, multiple-page displays should be consolidated, and sufficient training time should be allotted for complex sequences of actions.

C. HEURÍSTICA DE ZHANG

Título: Fourteen usability heuristics

1. [**Consistency**] Consistency and standards.

Users should not have to wonder whether different words, situations, or actions mean the same thing. Standards and conventions in product design should be followed.

a. Sequences of actions (skill acquisition).
b. Color (categorization).
c. Layout and position (spatial consistency).
d. Font, capitalization (levels of organization).
e. Terminology (delete, del, remove, rm) and language (words, phrases).
f. Standards (e.g., blue underlined text for unvisited hyperlinks).

2. [**Visibility**] Visibility of system state

Users should be informed about what is going on with the system through appropriate feedback and display of information.

a. What is the current state of the system?
b. What can be done at current state?
c. Where can users go?

d. What change is made after an action?

3. [**Match**] Match between system and world

The image of the system perceived by users should match the model the users have about the system.

a. User model matches system image.
b. Actions provided by the system should match actions performed by users.
c. Objects on the system should match objects of the task.

4. [**Minimalist**] Minimalist.

Any extraneous information is a distraction and a slow-down.

a. Less is more.
b. Simple is not equivalent to abstract and general.
c. Simple is efficient.
d. Progressive levels of detail.

5. [**Memory**] Minimize memory load.

Users should not be required to memorize a lot of information to carry out tasks. Memory load reduces user's capacity to carry out the main tasks.

a. Recognition vs. recall (e.g., menu vs. commands).
b. Externalize information through visualization.
c. Perceptual procedures.
d. Hierarchical structure.
e. Default values.
f. Concrete examples (DD/MM/YY, e.g., 10/20/99).
g. Generic rules and actions (e.g., drag objects).

6. [**Feedback**] Informative feedback.

Users should be given prompt and informative feedback about their actions.

a. Information that can be directly perceived, interpreted, and evaluated.
b. Levels of feedback (novice and expert).
c. Concrete and specific, not abstract and general.
d. Response time.
    i. 0.1 s for instantaneously reacting;
    ii. 1.0 s for uninterrupted flow of thought;

iii. 10 s for the limit of attention.

7. [**Flexibility**] Flexibility and efficiency.

Users always learn and users are always different. Give users the flexibility of creating customization and shortcuts to accelerate their performance.

a. Shortcuts for experienced users.
b. Shortcuts or macros for frequently used operations.
c. Skill acquisition through chunking.
d. Examples:
    i. Abbreviations, function keys, hot keys, command keys, macros, aliases, templates, type-ahead, bookmarks, hot links, history, default values, etc.

8. [**Message**] Good error messages.

The messages should be informative enough such that users can understand the nature of errors, learn from errors, and recover from errors.

a. Phrased in clear language, avoid obscure codes. Example of obscure code: ‘‘system crashed, error code 147.’’
b. Precise, not vague or general. Example of general comment: ‘‘Cannot open document.’’
c. Constructive.
d. Polite. Examples of impolite message: ‘‘illegal user action,’’ ‘‘job aborted,’’ ‘‘system was crashed,’’ ‘‘fatal error,’’ etc.

9. [**Error**] Prevent errors.

It is always better to design interfaces that prevent errors from happening in the first place.

a. Interfaces that make errors impossible.
b. Avoid modes (e.g., vi, text wrap). Or use informative feedback, e.g., different sounds.
c. Execution error vs. evaluation error.
d. Various types of slips and mistakes.

10. [**Closure**] Clear closure.

Every task has a beginning and an end. Users should be clearly notified about the completion of a task.

a. Clear beginning, middle, and end.
b. Complete 7-stages of actions.
c. Clear feedback to indicate goals are achieved and current stacks of goals can be released. Examples of good closures include many dialogues.

11. [**Undo**] Reversible actions.

Users should be allowed to recover from errors. Reversible actions also encourage exploratory learning.

a. At different levels: a single action, a subtask, or a complete task.
b. Multiple steps.
c. Encourage exploratory learning.
d. Prevent serious errors.

12. [**Language**] Use users' language.

The language should be always presented in a form understandable by the intended users.

a. Use standard meanings of words.
b. Specialized language for specialized group.
c. User defined aliases.
d. Users' perspective. Example: ''we have bought four tickets for you'' (bad) vs. ''you bought four tickets'' (good).

13. [**Control**] Users in control.

Do not give users that impression that they are controlled by the systems.

a. Users are initiators of actors, not responders to actions.
b. Avoid surprising actions, unexpected outcomes, tedious sequences of actions, etc.

14. [**Document**] Help and documentation.

Always provide help when needed.

a. Context-sensitive help.

b. Four types of help.
- task-oriented;
- alphabetically ordered;
- semantically organized;
- search.

c. Help embedded in contents.

APÊNDICE A

NUTES – Núcleo de Tecnologia Estratégicas em Saúde

Manual de instalação SAHU

Campina Grande – PB
2016

O QUE É SAHU?

É um sistema que gerencia bases de heurísticas de usabilidade (SAHU – Sistema de Armazenamento de Heurísticas de Usabilidade). O sistema será utiliza no Laboratório de Usabilidade e Fatores Humanos (LUFH) que está localizado no NUTES (Núcleo de Tecnologias Estratégicas em Saúde) que fica no Campus I da Universidade Estadual da Paraíba (UEPB). SAHU deverá ser capaz de armazenar e controlar qualquer heurística de usabilidade.

REQUISITOS DE INSTALAÇÃO

Para a instalação e configuração do site no servidor com sistema operacional Windows 10 é necessário os seguintes programas e framework:

- ✓ Framework .NET 4.5
- ✓ IIS
- ✓ PostgreSQL 9.6

INSTALANDO IIS

Para instalar o IIS (Gerenciador de Serviços de Informações de Internet) abra na página mostrada na figura 1. Caso não saiba como fazer clique “Windows + R” e digite “appwiz.cpl”.

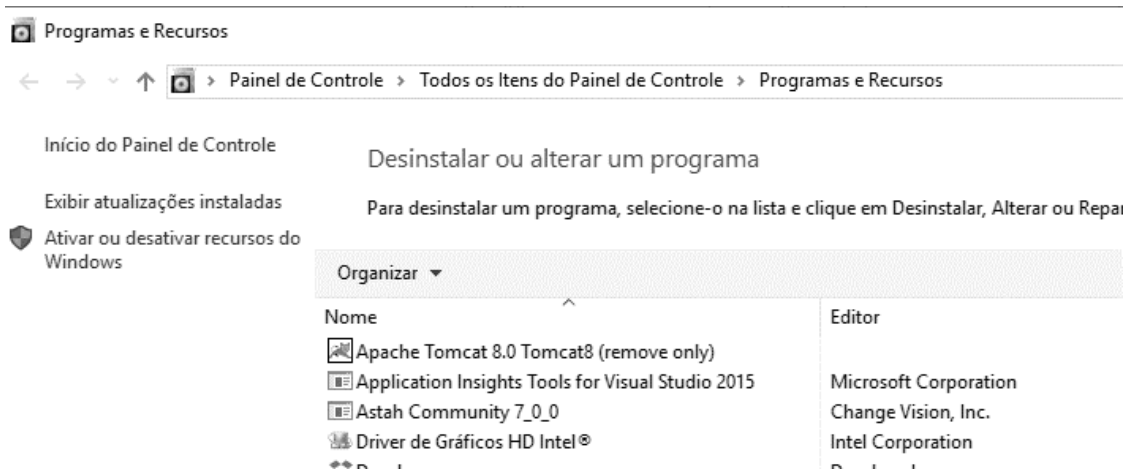

Figura 1 – Ativar recursos do windows

Após abrir na página mostrado na figura 1, clique em “Ativar ou desativar recursos do Windows”. Irá aparecer uma página como mostrada na figura 2.

Clique na sessão “Serviço de informática da internet” e selecione os tópicos como mostrado na imagem abaixo. Após selecionar clique em OK, após clicar será instalado os recursos.

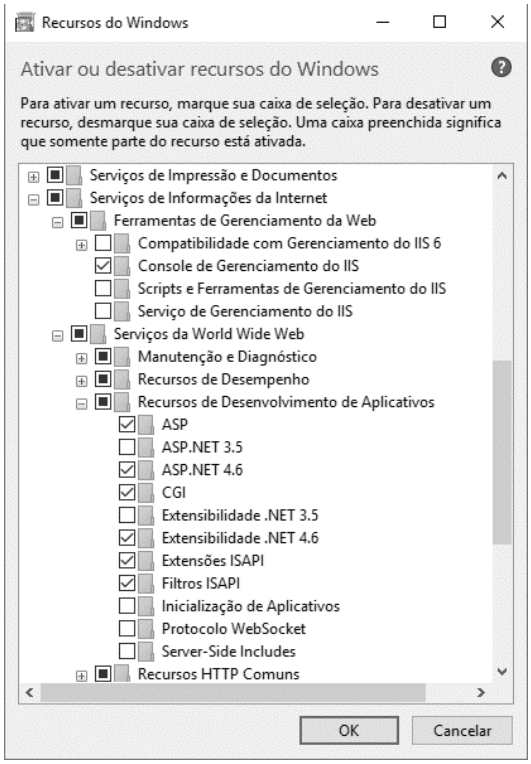

Figura 2 – Selecionar recursos

Para verificar se foi instalado corretamente, vá para o seu navegador e digite “localhost”. Deverá aparecer a página como mostrada na figura 3.

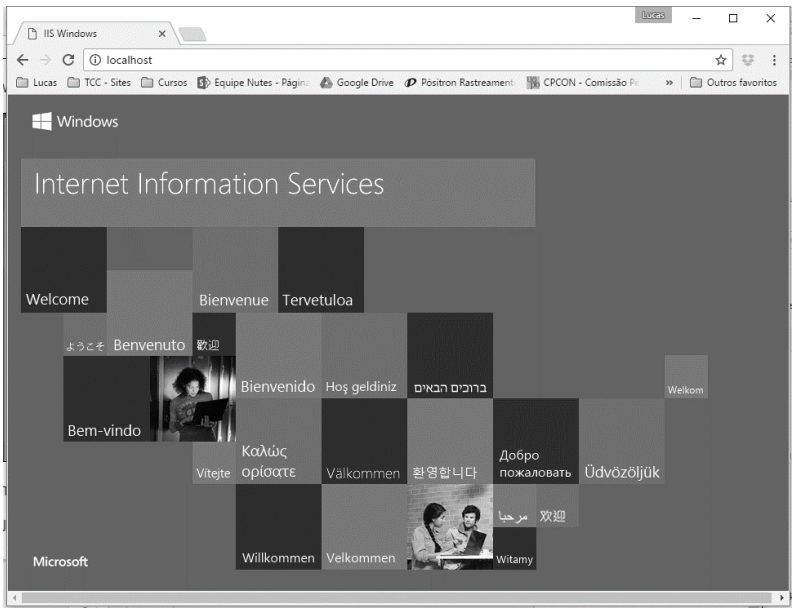

Figura 3 - Localhost

CONFIGURANDO O SITE

Para configurar o site, primeiro crie uma pasta com o nome do site no caminho a seguir “C:\inetpub\wwwroot\sahu”. Veja a figura 4.

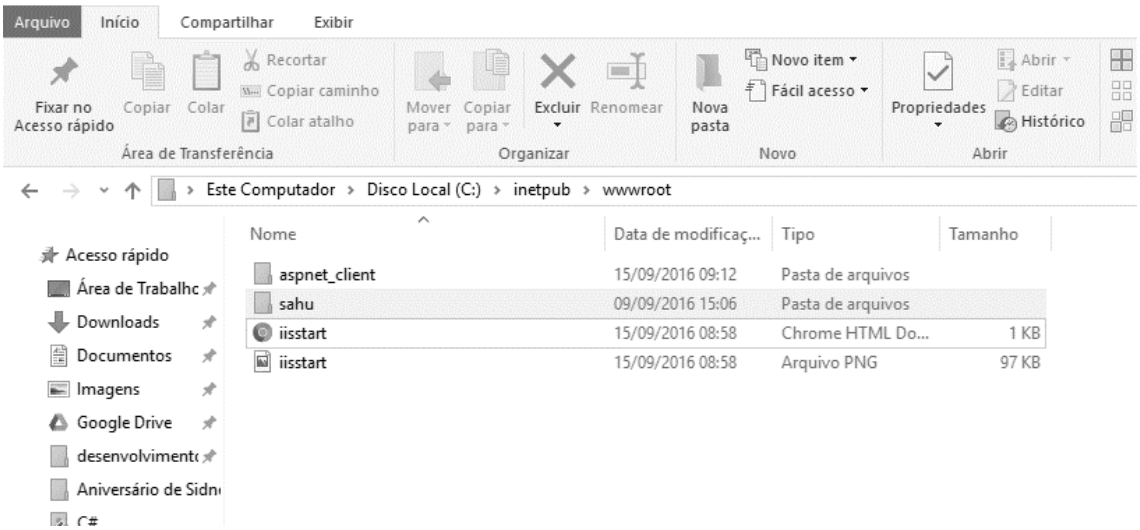

Figura 4 – Repositório do site

Na pasta criada coloque os arquivos disponibilizado no site do SharePoint com o nome “site”.

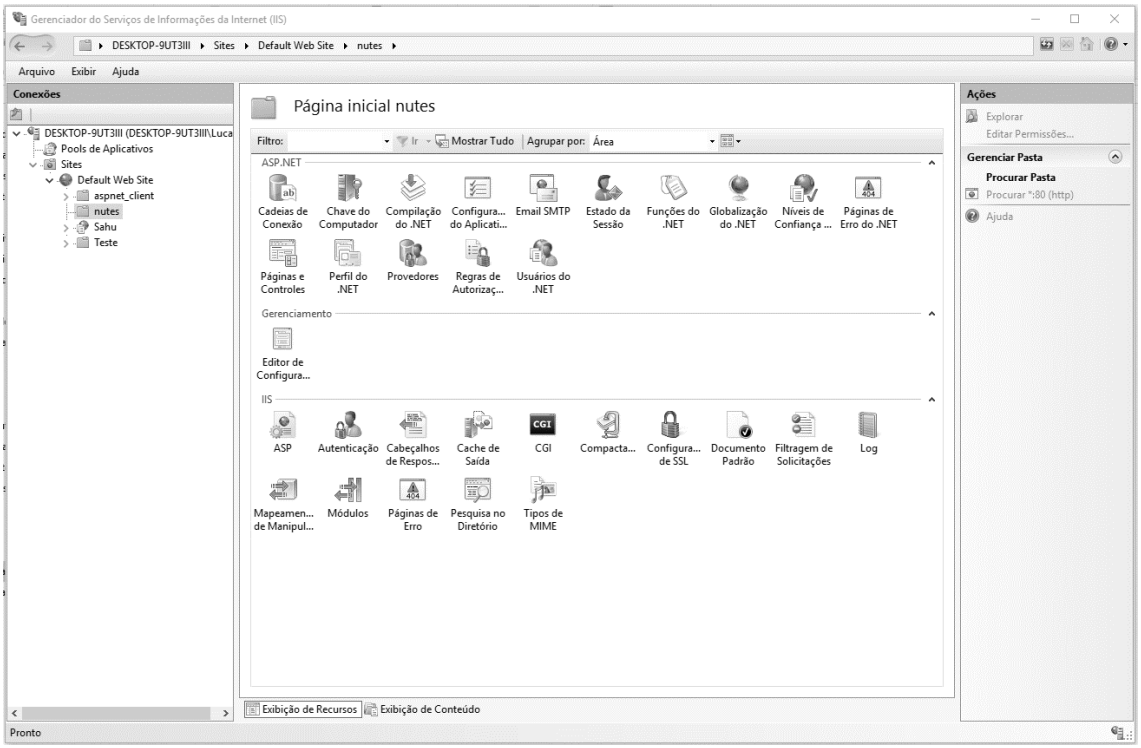

Figura 5 – Configurar site

Para configurar abra no Gerenciador de Serviços de Informações da Internet, veja a figura 5. Clique no botão direito no nome da pasta que criou anteriormente e clique em adicionar aplicativo. Será apresentado a tela como na figura 6.

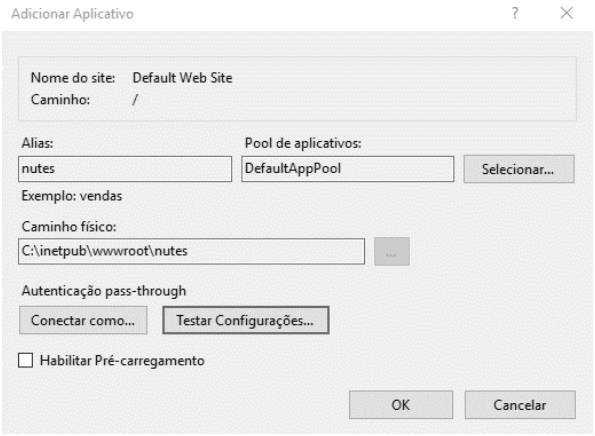

Figura 6 – Adicionar aplicativo

Na tela selecione o Pool de aplicativos DefaultAppPool. Faça o teste de configuração, caso tenha algum problema na autenticação clique em “Conectar Como ...” preencha as informações e teste novamente. Em seguida clique em Ok.

Abra o seu navegador e digite http://localhost/sahu/Index.

Instalando PostgreSQL

A instalação do banco de dados PostgreSQL 9.6 é feita sem nenhum problema. A senha do PostgreSQL utilizada no sistema é ‘root’. Caso queira modificar deverá ser alterado no código fonte do sistema. Ele pode ser adquirido pelo link a seguir () ou pode ser baixado pelo SharePoint do Nutes. Após a instalação do PostgreSQL crie a base de dados. Crie a base de dados com o nome SAHU.

Nome da base de dados "SAHU"

Após criar a base de dados deverá ser criada as tabelas, os códigos SQL seguem abaixo.

```
CREATE TABLE public."Autor" (

  id SERIAL NOT NULL,
  nome character varying(50) NOT
NULL,
  instituicao character varying(150)
NOT NULL,
  informacoes character varying(300)
NOT NULL,
  treinamento boolean,
  CONSTRAINT pk_idautor PRIMARY
KEY (id)
)
```

```
CREATE TABLE public."Exemplo"  (
  id SERIAL NOT NULL,
  descricao character varying(300),
  imagem bytea,
  treinamento boolean,
  CONSTRAINT "PK_IdExemplo"
PRIMARY KEY (id)
)

CREATE TABLE public."Log" (
  id SERIAL NOT NULL,
  data date,
  tarefa character varying(50),
  usuario character varying(100),
```

```
  treinamento boolean,
  CONSTRAINT "PK_id" PRIMARY
KEY (id)
)

CREATE TABLE public."Login" (
  id SERIAL NOT NULL,
  "Email" character varying(100) NOT
NULL DEFAULT 150,
  "Senha" character varying(50),
  CONSTRAINT pk_idlogin PRIMARY
KEY (id)
)

CREATE TABLE public."Perfil" (
  id SERIAL NOT NULL,
  nome character varying(100) NOT
NULL,
  descricao character varying(500)
NOT NULL,
  inserirheuristica boolean NOT NULL,
  atualizarheuristica boolean NOT
NULL,
  removerheuristica boolean NOT
NULL,
  inserirusuario boolean NOT NULL,
  atualizarusuario boolean NOT NULL,
  removerusuario boolean NOT NULL,
  treinamento boolean,
  CONSTRAINT pk_idperfil PRIMARY
KEY (id)
)

CREATE TABLE public."Tag" (
  id SERIAL NOT NULL,
  nome character varying(50),
  treinamento boolean,
  CONSTRAINT pk_idtag PRIMARY
KEY (id)
)

CREATE TABLE public."Usuario" (
  id SERIAL NOT NULL,
  nome character varying(150),
  perfil integer,
  login integer,
  telefone character varying(20),
  treinamento boolean,
  CONSTRAINT pk_idusuario
PRIMARY KEY (id),
  CONSTRAINT "FK_IdPerfil"
FOREIGN KEY (perfil)
      REFERENCES public."Perfil" (id)
MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION,
  CONSTRAINT "FK_idLogin"
FOREIGN KEY (login)
      REFERENCES public."Login" (id)
MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION
)

CREATE TABLE public."Heuristica" (
  id SERIAL NOT NULL,
  nome character varying(100),
  link character varying(500),
  referencia character varying(500),
  visibilidade boolean,
  descricao character varying(1000),
  exemplo integer,
  treinamento boolean,
  CONSTRAINT pk_idheuristica
PRIMARY KEY (id),
  CONSTRAINT "FK_exemplo"
FOREIGN KEY (exemplo)
      REFERENCES public."Exemplo"
(id) MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION
)

CREATE TABLE public."Conjunto" (
  id SERIAL NOT NULL,
  nome character varying(200),
  data date,
  "areaDeAtuacao" character
varying(1000),
  usuario integer,
  autor integer,
  treinamento boolean,
  CONSTRAINT pk_idconjunto
PRIMARY KEY (id),
  CONSTRAINT "FK_idAutor"
FOREIGN KEY (autor)
      REFERENCES public."Autor" (id)
MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION,
```

```
  CONSTRAINT "FK_idUsuario"
FOREIGN KEY (usuario)
      REFERENCES public."Usuario"
(id) MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION
)

CREATE TABLE public.conj_heuristica
(
  id SERIAL NOT NULL,
  idheuristica integer,
  idconjunto integer,
  CONSTRAINT pk_idconjheuristica
PRIMARY KEY (id),
  CONSTRAINT "FK_idRelConjunto"
FOREIGN KEY (idconjunto)
      REFERENCES public."Conjunto"
(id) MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION,
  CONSTRAINT "FK_idRelHeuristica"
FOREIGN KEY (idheuristica)
      REFERENCES public."Heuristica"
(id) MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION
)

CREATE TABLE public.rel_tc (
  id SERIAL NOT NULL,
  "idRelConjunto" integer,
  "idRelTag" integer,
  prioridade integer,
  CONSTRAINT "idTC" PRIMARY KEY
(id),
  CONSTRAINT "FK_idRelConjunto"
FOREIGN KEY ("idRelConjunto")
      REFERENCES public."Conjunto"
(id) MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION,
  CONSTRAINT "FK_relTag"
FOREIGN KEY ("idRelTag")
      REFERENCES public."Tag" (id)
MATCH SIMPLE
      ON UPDATE NO ACTION ON
DELETE NO ACTION
)
```

Também serão necessárias a criação das views. Para a criação, basta executar os códigos abaixo.

```
CREATE OR REPLACE VIEW
public.conjuntosusuario AS
     SELECT c.id,
        c.nome,
        c.data,
        c."areaDeAtuacao",
        a.id AS "idAutor",
        a.nome AS "nomeAutor",
        u.id AS "idUsuario"
       FROM "Conjunto" c
         JOIN "Autor" a ON a.id = c.autor
         JOIN "Usuario" u ON u.id =
c.usuario;

    ALTER TABLE
public.conjuntosusuario
      OWNER TO postgres;

  CREATE OR REPLACE VIEW
public.informacoesusuario AS
     SELECT u.id,
        u.nome,
        u.telefone,
        l.id AS "idLogin",
        l."Email",
        p.id AS "idPerfil",
        p.nome AS "nomePerfil",
        p.descricao,
        p.atualizarheuristica,
          p.inserirheuristica,
          p.removerheuristica,
          p.atualizarusuario,
          p.inserirusuario,
          p.removerusuario,
          p.treinamento
         FROM "Usuario" u
           JOIN "Login" l ON l.id = u.login
           JOIN "Perfil" p ON p.id =
u.perfil;

      ALTER TABLE
public.informacoesusuario
       OWNER TO postgres;
```

```
CREATE OR REPLACE VIEW
public.tagheursitica AS
    SELECT c.nome AS
"nomeConjunto",
       c.data,
       h.id,
       h.nome,
       h.descricao,
       t.id AS "idTag",
       t.nome AS "NomeTag",
       t.treinamento
      FROM rel_tc r
       RIGHT JOIN "Conjunto" c ON
c.id = r."idRelConjunto"
       RIGHT JOIN "Tag" t ON t.id =
r."idRelTag"
       RIGHT JOIN conj_heuristica ch
ON ch.idconjunto = c.id
       RIGHT JOIN "Heuristica" h ON
h.id = ch.idheuristica;

   ALTER TABLE public.tagheursitica
    OWNER TO postgres;
```

```
 CREATE OR REPLACE VIEW
public.verheuristicavw AS
         SELECT ch.id,
            ch.idconjunto,
            ch.idheuristica,
            h.nome AS "nomeHeuristica",
            h.link,
            h.referencia,
            h.visibilidade,
            h.descricao,
            h.exemplo AS "idExemplo",
            h.treinamento
           FROM conj_heuristica ch
            JOIN "Conjunto" c ON c.id =
ch.idconjunto
             JOIN "Heuristica" h ON h.id
= ch.idheuristica;

        ALTER TABLE
public.verheuristicavw
         OWNER TO postgres;
```

Além de criar as tabelas, os códigos criam alguns perfis de usuários e o usuário padrão. Segue abaixo o login e senha do usuário padrão. Após inserir altere a senha do usuário padrão.

Login: sahu.nutes@gmail.com

Senha: 123

APÊNDICE B

1. DESCRIÇÃO DO SISTEMA

O objetivo do sistema é armazenar heurísticas de usabilidade (SAHU – Sistema de Armazenamento de Heurísticas de Usabilidade). O sistema será utilizado para dar suporte a outro sistema que fará análise de usabilidade. Estas análises serão feitas, em primeiro momento, no Laboratório de Usabilidade e Fatores Humanos (LUFH) que está localiza no NUTES (Núcleo de Tecnologias Estratégicas em Saúde) que fica no Campus I da Universidade Estadual da Paraíba (UEPB).

O sistema deverá ser capaz de armazenar qualquer heurística de usabilidade, evitando redundância de informações (heurística). O sistema deverá ter um login de acesso em que os usuários poderão inserir, editar e remover as heurísticas. Terão quatro tipos (perfis) de usuários básicos: administrador, especialista, técnico e capacitação. Também deverão ter métodos de pesquisar as heurísticas para que os especialistas possam estudar antes de fazem as análises dos sistemas.

No Quadro 1 tem um resumo do fluxo de tarefas que os usuários deverão seguir para cadastrar os conjuntos de heurísticas. No primeiro círculo de cor laranja, o usuário deverá cadastrar as Tags. Em seguida, veja o círculo cinza, o usuário deverá cadastrar o autor. No círculo amarelo, o usuário irá cadastrar as heurísticas. Finalmente ele poderá cadastrar o conjunto, veja o círculo azul.

QUADRO 1 - FLUXO DE CADASTRO DE CONJUNTOS DE HEURÍSTICAS

Antes de qualquer cadastro os usuários deverão verificar se já existem as Tags, autores, heurísticas ou conjuntos. Caso não existam eles poderão cadastrar.

2. PÚBLICO ALVO

O sistema ficará disponível para o NUTES, como repositório de heurísticas de usabilidade, este repositório servirá para o sistema que fará a análise. Também será útil para os especialistas que farão as analise de usabilidade, os alunos que participarão dos projetos e todos que poderão participar dos projetos no LUFH.

3. TIPOS DE USUÁRIOS

Como foi informado na descrição do sistema, existem quatro perfis de usuários básicos. A seguir serão descritas as funcionalidades de cada um.

3.1. ADMINISTRADOR

O administrador é o usuário que possui a maioria das permissões possíveis, ele que deverá gerenciar os outros usuários. Por isso, ele poderá adicionar e remover os usuários, também o administrador poderá mudar o perfil dos outros usuários. Além das funcionalidades anteriores, ele poderá inserir, excluir, editar, pesquisar heurísticas e gerar relatórios do sistema.

3.2. ESPECIALISTA

O especialista será o segundo usuário que terá maior importância. Ele poderá inserir, editar, pesquisar heurísticas e os gerar os relatórios.

3.3. TÉCNICO

O técnico será o responsável pelo funcionamento do laboratório em termos técnico, como exemplo, a instalação de câmeras, verificar se tudo está de acordo para ser feito o teste das análises de heurísticas de usabilidade. Por isso que este usuário terá poucas permissões, pois não será necessário a permissão de todas as funcionalidades. Será permitido pesquisar as heurísticas e gerar os relatórios.

3.4. CAPACITAÇÃO

Este usuário tem como objetivo de treinamento. Ele poderá assumir qualquer tipo de usuário mencionado anteriormente (administrador, especialista ou técnico), mas todas as informações não poderão estar disponíveis para o uso das analise de usabilidade.

4. REQUISITOS

Nesta sessão serão listados os requisitos do sistema.

4.1. FUNCIONAIS

Na tabela a baixo temos um modelo que serão apresentadas as informações do sistema.

CÓDIGO	NOME DA FUNCIONALIDADE	PRIORIDADE	PERFIL (USUÁRIO)
DESCRIÇÃO DA FUNCIONALIDADE			

Os níveis de prioridade foram criados com a escala de Alta, Média e Baixa. O critério para a definição foram:

- ✓ Necessidade que a funcionalidade esteja pronta;
- ✓ Nível de dificuldade seja maior;
- ✓ Importância para o sistema.

Os requisitos funcionais são:

RF 001	Login	Alta	Todos
O usuário deverá efetuar o login no sistema. O sistema deverá identificar o perfil do usuário e deverá ser encaminhado para a tela correspondente. Se o usuário esquecer, deverá aparecer uma opção 'Esqueci a senha', quando clicar será enviada uma senha por e-mail e na tela de acesso aparecerá uma mensagem informando a ação.			

RF 002	Cadastro de usuários	Alta	Administrador
O usuário poderá cadastrar novos usuários, terá que escolher qual o tipo de perfil escolher (administrador, especialista, técnico ou capacitação) e preencher as informações básicas e salvar as informações. Quando for salva o cadastro, será enviada um login e senha para o novo usuário.			

RF 003	Excluir usuário	Media	Administrador
O usuário poderá remover o usuário. Mas algumas informações continuaram salvas.			

RF 004	Mudar de perfil	Baixa	Administrador
O usuário poderá mudar o perfil dos outros usuários, ele só poderá mudar de perfil se existir outro semelhante e ele. O sistema deverá evitando que o sistema fique sem o administrador.			

RF 005	Inserir heurística	Alta	Administrador, especialista e Capacitação
O administrador e o especialista poderão inserir novas heurísticas, elas ficarão disponíveis para as análises. Já o perfil Capacitação poderão inserir, no entanto as heurísticas que serão cadastradas não ficarão disponíveis para a análise e nem a			

visualização dos outros usuários de perfil diferente.

RF 006	Remover heurística	Media	Administrador
Somente o administrador poderá remover as heurísticas na base do sistema. Se uma heurística estiver relacionada a um conjunto de heurística, o sistema deverá mandar uma mensagem.			

RF 008	Editar heurística	Media	Administrador, Especialista e Capacitação
O administrador e o especialista poderão editar as heurísticas, se uma heurística estiver relacionada a mais de um autor que faz referência a heurística que se deseja editar, o sistema deverá enviar uma mensagem informando. Os usuários que possuírem o perfil de Capacitação poderão editar apenas as heurísticas que ele criou.			

RF 009	Pesquisar heurística	Alta	Todos
O Especialista e o Técnico quando fizerem a pesquisa serão mostradas todas as heurísticas, menos as que foram cadastradas pelo usuário do perfil Capacitação. O administrador poderá pesquisar qualquer heurística, mas deverá existir um filtro de pesquisa em poderá escolher se deseja que apresente as heurísticas dos usuários de perfil Capacitação.			

RF 010	Gerar relatório	Baixa	Todos
Todos os usuários poderão gerar relatórios. (Ainda serão definidos o que serão esses relatórios)			

RF 011	Log	Baixa	Todos
O sistema deverá armazenar todas as atividades realizadas por todos os usuários. Serão necessários gravar as operações realizadas, data e hora.			

RF 012	Mudar senha	Baixa	Todos
A partir do primeiro acesso o usuário poderá alterar a senha de login.			

5. NÃO FUNCIONAIS

As pesquisas que serão realizadas deverão seguir uma ordem de prioridade. Da ordem de prioridade mais alta para baixa, fica: *Tag* (palavra-chave), conceito e exemplo. As telas da página web deverá ser responsivo, pois poderão ser acessadas pelos computadores Desktop, Notebook, tablete ou smartphone.

6. CASO DE USO

CADASTRAR USUÁRIO

Ator principal: Administrador

Cenário principal:

i. O usuário acessa o site do sistema com um navegador de preferência, se *logar* ao sistema;
ii. Caso a tentativa de acesso for realizada com sucesso ele deverá procurar na sessão de menus o submenu ‘cadastrar usuário’;
iii. Quando clicar na opção, deverá aparecer um formulário que o administrador deverá preencher as informações (nome do novo usuário, e-mail e perfil).
iv. Após preencher o formulário o usuário deverá clicar em salvar cadastro.
v. Caso a operação for realizada com sucesso deverá será direcionada a tela inicial.
vi. O sistema deverá enviar um e-mail com os dados de acesso ao sistema.

REMOVER USUÁRIO:

Ator principal: Administrador

Cenário principal:

i. O usuário acessa o site do sistema com um navegador de preferência, se logar ao sistema;
ii. Faz uma pesquisa pelo nome do usuário;
iii. Clica na opção de remover usuário;
iv. Confirma que deseja remover;
v. A tela do sistema retorna a tela inicial;

ATUALIZAR USUÁRIO OU MUDAR DE PERFIL:

Ator principal: Administrador

Cenário principal:

i. O usuário acessa o site do sistema com um navegador de preferência, se logar ao sistema;
ii. O usuário clica na opção de Mudar de perfil de usuário.
iii. O sistema verifica se o usuário tem permissão para mudar de perfil;
iv. Se o sistema identificar que o usuário não tem a permissão, o sistema irá apresentar uma mensagem informando que o usuário não pode mudar de perfil;
v. Se o sistema verificar que o usuário tem a permissão, o sistema apresenta uma tela em que o deverá ser selecionado o usuário em que será alterado o perfil;

vi. O usuário verifica se é o usuário em que se deseja alterar;
vii. O usuário faz as alterações;
viii. O usuário clica no botão para salvar, se a operação for realizada com sucesso, o sistema apresenta uma mensagem.
ix. Se a operação ocorrer algum problema, é apresentado uma mensagem e permanece na mesma tela. O sistema apresenta uma mensagem dizendo que deverá tentar novamente;
x. O sistema volta a tela inicial.

INSERIR HEURÍSTICA:

Ator principal: Administrador

Cenário principal:

i. O usuário acessa o site do sistema com um navegador de preferência, se logar ao sistema;
ii. O usuário clica na opção de cadastrar nova heurística.
iii. O sistema verifica se o usuário tem permissão para cadastrar nova heurística;
iv. Se o sistema verificar que não é permitido inserir, deverá apresentar uma mensagem informando que ele não tem permissão;
v. Se o sistema verificar que o usuário pode inserir, abrirá uma nova tela com um formulário para cadastro da heurística;
vi. Após o preenchimento das heurísticas, o usuário irá clicar no botão para salvar;
vii. Se a operação for salva com sucesso irá apresentar uma mensagem e o sistema irá encaminhar para a tela inicial.

REMOVER HEURÍSTICA:

Ator principal: Administrador

Cenário principal:

i. O usuário acessa o site do sistema com um navegador de preferência, se logar ao sistema;
ii. O usuário faz uma pesquisa do conjunto de heurística a ser deletado;
iii. Após encontrar o conjunto, o usuário clica em remover heurística;
iv. O sistema verifica se o usuário tem permissão para remover heurística;
v. Se o sistema identificar que não tem permissão, é apresentada uma mensagem informando;

vi. Caso contrário, o sistema apresenta a lista das heurísticas que serão removidas. Cada heurística será apresentada uma observação em que informa se a heurística em que se deseja remover pertence a outro conjunto de heurística;

vii. Se a heurística tiver apenas um relacionamento, então o sistema poderá remover da base de dados;

viii. O usuário poderá desmarcar as heurísticas em que não deseja remover;

ix. Após a análise do usuário, ele deverá clicar em remover. O sistema deverá apresentar uma nova mensagem para que ele confirme se realmente deseja remover.

7. DIAGRAMA DE CASO DE USO

Na Figura 2 é apresentado um diagrama de caso de uso. Todas as funcionalidades apresentadas no diagrama deverão ser registradas os logs registrando a atividade realizada, usuário e a data da modificação do sistema.

No diagrama é apresentado todos os perfis de usuário do sistema (atores) (administrador, especialista, técnico e capacitação). As funcionalidades apresentadas no diagrama deverão se comportar de acordo com o tipo de perfil de cada usuário. Por exemplo, todas as atividades feitas pelo usuário do perfil capacitação não deverá estar disponível para os outros perfis como o administrador; especialista ou técnico.

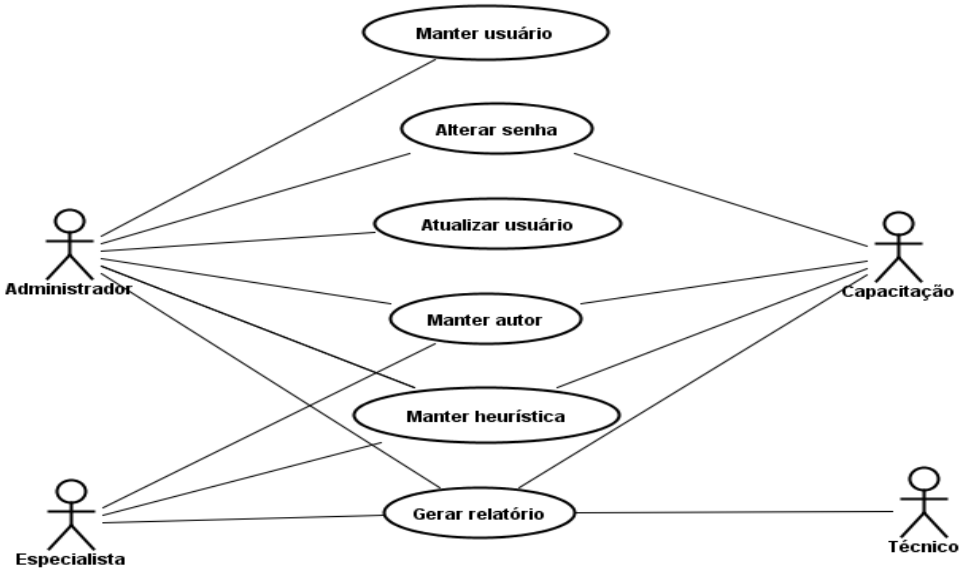

FIGURA 2 - DIAGRAMA DE CASO DE USO