UNIVERSIDADE FEDERAL DO PAMPA

Esther Salgado Favero

Um Protótipo de Referência para Ferramentas CASE de Modelagem em Ambiente Web

Alegrete
2019

Esther Salgado Favero

Um Protótipo de Referência para Ferramentas CASE de Modelagem em Ambiente Web

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. João Pablo Silva da Silva

Alegrete
2019

Esther Salgado Favero

Um Protótipo de Referência para Ferramentas CASE de Modelagem em Ambiente Web

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 10 de Dezembro de 2019

Banca examinadora:

Prof. Dr. João Pablo Silva da Silva
Orientador
UNIPAMPA

Prof. Dr. Gilleanes Thorwald Araujo Guedes
UNIPAMPA

Prof. Me. Miguel Julio Zinelli da Costa Junior
UNIPAMPA

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me dado força, sabedoria e por ter atendido à todas as minhas orações nos momentos mais difíceis da minha caminhada. Também, sou grata a mim por nunca ter desistido mesmo em meio às adversidades. Agradeço aos meus pais por sempre me apoiarem de todas as formas para que eu chegasse até aqui. Agradeço à minha família, em geral, por sempre, com muito amor, me motivar a continuar. Agradeço aos meus colegas e amigos, que sempre estiveram comigo me ajudando e contribuindo para o meu desenvolvimento intelectual e pessoal. Agradeço ao meu orientador, por dedicar boa parte do seu tempo me dando as diretrizes para construir um bom trabalho. Obrigada por me manter motivada desde o início até o fim da minha pesquisa. Agradeço, também, à todos os professores que passaram pelo meu caminho, colaborando para o meu crescimento. Por fim, agradeço à minha universidade, por ter me proporcionado os mais desafiadores e melhores anos da minha vida.

“O sucesso é a capacidade de ir de um fracasso ao outro sem perder o entusiasmo.”
(Winston Churchill)

RESUMO

As ferramentas *Computer-Aided Software Engineering* (CASE) proveem suporte para os engenheiros de software ao longo de todo o ciclo de desenvolvimento. Essas ferramentas são dotadas de uma variedade de recursos, os quais nem sempre possuem uma boa usabilidade. Foi evidenciado na literatura que a interface das ferramentas CASE são negligenciadas, ocasionando grande complexidade na interação do usuário e dificultando o seu uso. O objetivo do nosso trabalho é propor um protótipo de referência com um alto nível de usabilidade para ferramentas CASE de modelagem em ambiente web. Para isso, fizemos uma revisão de literatura em conjunto com uma análise de dados aprofundada a fim de identificar requisitos de usabilidade que nos permitam atingir o nosso objetivo. Os requisitos encontrados a partir da literatura foram a visibilidade, a satisfação, a baixa taxa de erro, a memorabilidade, a aprendizibilidade e a eficiência. Além disso, utilizamos o método de prototipação para desenvolvê-lo. O processo consiste em determinar as necessidades, desenvolver o protótipo, avaliar e refatorar até se obter um resultado satisfatório. Desenvolvemos o protótipo de referência cobrindo todas as necessidades encontradas, isto é, os requisitos de usabilidade mapeados. Por fim, planejamos e executamos a avaliação do protótipo. Escolhemos 3 professores de engenharia de software, 3 graduandos de engenharia de software, 3 mestrandos de engenharia de software e 3 desenvolvedores de software para avaliar o protótipo. A técnica que utilizamos para fazer a avaliação foi o System Usability Scale (SUS). Através dessa escala, pudemos encontrar os resultados da avaliação. A média resultante da avaliação foi de 88,93 pontos, o que podemos considerar um bom resultado, pois, segundo a literatura, acima de 68 pontos, pode-se considerar que o sistema possui uma boa usabilidade. Com isso, acreditamos ter alcançado nosso objetivo, quer dizer, ter um protótipo de referência com um bom nível de usabilidade.

Palavras-chave: Ferramentas CASE. Usabilidade. Prototipação. Modelagem de Software.

ABSTRACT

Computer-Aided Software Engineering (CASE) tools provide support for software engineers throughout the development cycle. These tools are endowed with a variety of features, which do not always have good usability. It was evidenced in the literature that the interface of CASE tools are neglected, causing great complexity in user interaction and making its use difficult. The aim of our work is to propose a reference prototype with a high level of usability for CASE web modeling tools. To this end, we conducted a literature review in conjunction with an in-depth data analysis to identify usability requirements that enable us to achieve our goal. The requirements found from the literature were visibility, satisfaction, low error rate, memorability, learning, and efficiency. In addition, we use the prototyping method to develop it. The process is to determine the needs, develop the prototype, evaluate, and refactor until a satisfactory result is obtained. We developed the reference prototype covering all needs encountered, i.e. the mapped usability requirements. Finally, we planned and executed the prototype evaluation. We choose 3 software engineering professors, 3 software engineering undergraduates, 3 software engineering master's students and 3 software developers to evaluate the prototype. The technique we used to make the assessment was the System Usability Scale (SUS). Through this scale, we could find the results of the evaluation. The average resulting from the evaluation was 88.93 points, which we can consider a good result, because, according to the literature, above 68 points, it can be considered that the system has a good usability. With this, we believe we have achieved our goal, that is, to have a reference prototype with a good level of usability.

Key-words: CASE Tools. Usability. Prototyping. Software Modeling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CASE *Computer-Aided Software Engineering*

IDE *Integrated Development Environment*

IHC Interação Humano-Computador

RBUIS *Role-Based UI Simplification*

SUS *System Usability Scale*

SUMÁRIO

1 INTRODUÇÃO

A tecnologia permite que o ser humano realize inovações para as mais variadas necessidades. O mercado é rico em artefatos desenvolvidos por pessoas capacitadas que atuam nessa área. No entanto, sabemos que, para construir um determinado produto, seja qual for, é necessário projetá-lo levando em conta a percepção do usuário final, isto é, aquele que fará o uso desse produto. No que diz respeito à área da computação, mais especificamente, julga-se que desenvolvedores de software são profissionais capacitados a construir sistemas a fim de garantir que o usuário final faça o uso do produto construído de forma fácil e intuitiva. O termo utilizado para garantir essa eficiência do usuário em utilizar aquilo que foi desenvolvido é chamado de "usabilidade" (ALBERT; TULLIS, 2013). Ainda, segundo Barbosa e Silva (2010), usabilidade está diretamente relacionada à qualidade de uso.

A usabilidade permite a facilidade de uso de um produto de software que possui Interação Humano-Computador (IHC). Conforme Pereira (2019), IHC é uma área de estudo que relaciona homem e máquina através da interação e/ou comunicação entre eles. Segundo o autor, ela está inserida nas ciências da computação e da informação e nas ciências sociais e comportamentais.

Sendo assim, para que o usuário tenha êxito na interação com o sistema desenvolvido e consiga obter uma maior produtividade do mesmo, é necessário que esse produto seja projetado considerando os conceitos de usabilidade.

Conforme mencionado anteriormente, os desenvolvedores de software devem preocupar-se com a percepção do usuário final para com o seu produto. Porém, quando se trata de desenvolver ferramentas para o seu próprio uso, ou para o uso de outros profissionais dessa área, os princípios de usabilidade frequentemente são deixados de lado (DILLON; THOMPSON, 2016).

Produzir software de fato não é uma tarefa trivial. Para isso, é necessária a utilização de vários recursos para a sua construção. Na engenharia de software, as ferramentas utilizadas para construir software são chamadas de ferramentas *Computer-Aided Software Engineering* (CASE). Essas são usadas nas diversas etapas de desenvolvimento de software e são dotadas de uma vasta quantidade de mecanismos indispensáveis na sua composição, como elementos de modelagem, por exemplo. Uma das tarefas a serem realizadas ao longo do ciclo de desenvolvimento é a modelagem de software. Ela consiste em especificar os requisitos e representar o projeto de software a ser desenvolvido (PRESSMAN; MAXIM, 2016). Dentre as variadas ferramentas existentes, estão as ferramentas CASE de modelagem de software. Geralmente, este tipo de ferramenta possui uma grande quantidade de elementos usados na modelagem de um sistema. Foram criadas com o intuito de auxiliar no desenvolvimento de software, automatizando as tarefas de rotina e diminuindo a carga cognitiva de seus usuários para que possam concentrar-se no processo da engenharia de software em si (HOU; WANG, 2009).

1.1 Motivação

É visto que a interação entre homem e máquina é obtida através da interface. Barbosa e Silva (2010) afirmam que "ela é o único meio de contato entre usuário e sistema". A interface das ferramentas CASE voltadas à modelagem de software são enriquecidas de itens, mecanismos e ferramentas que são indispensáveis na hora de modelar um sistema. No entanto, esse grande número de opções acaba dificultando o uso por parte dos engenheiros de software. Os engenheiros que já têm um certo nível de familiaridade com esse tipo de ferramenta não são tão afetados. Porém, os que estão iniciando neste domínio muitas vezes ficam perdidos e sem saber por onde começar (ZOU et al., 2008).

Há várias ferramentas dessa categoria no mercado. O Eclipse[1], por exemplo, é uma das ferramentas CASE bastante utilizada para o desenvolvimento de software (LUCKOW; MELO, 2010), ainda que sua interface possua uma poluição visual, composta por vários ícones na tela. A má elaboração das interfaces das ferramentas CASE de modelagem impactam diretamente na eficiência da construção da modelagem de software (WEINRICH, 1999), afetando as demais fases de construção de sistema. Modelar um software de forma rápida, intuitiva e simples seria a chave para garantir que o tempo gasto com essa atividade fosse otimizado e seu aproveitamento melhorado. A motivação desta pesquisa é descobrir quais são as características necessárias em uma ferramenta CASE para que ela tenha uma boa usabilidade.

1.2 Objetivo

O objetivo geral deste trabalho é o desenvolvimento de um protótipo de alta fidelidade como referência para ferramentas CASE de modelagem em ambiente web que respeite os princípios de usabilidade. Com isso, esperamos melhorar a utilização desse tipo de ferramenta, tendo como benefício a otimização e eficiência na modelagem de software por meio dos engenheiros de software. De forma complementar, decompomos nosso objetivo geral nos seguintes objetivos específicos:

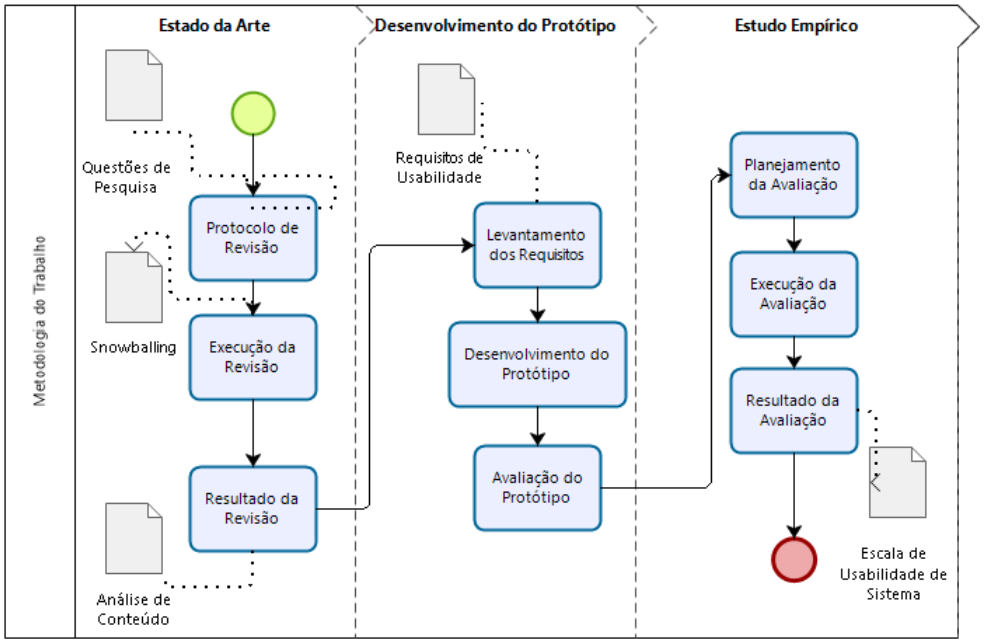
- verificação do estado da arte sobre as melhores técnicas de usabilidade direcionadas a ferramentas CASE para garantir os princípios de usabilidade dos protótipos;
- desenvolvimento do protótipo funcional de uma ferramenta CASE que se adeque aos princípios de usabilidade encontrados na etapa anterior;
- avaliação do protótipo desenvolvido por meio de um estudo empírico.

[1] https://www.eclipse.org

1.3 Metodologia

Apresentamos na Figura 1 a decomposição do trabalho necessário para atingir o objetivo proposto. Primeiramente, verificamos o estado da arte para obter o conhecimento do que há na literatura, até então, sobre usabilidade de ferramentas CASE. Para que o estado da arte fosse examinado, fizemos uma revisão sistemática de literatura, na qual foi definido um protocolo a ser seguido a fim de executar a revisão sistemática. Feito isso, os dados extraídos a partir da revisão foram analisados para que pudéssemos aplicá-los no desenvolvimento do protótipo de referência para ferramentas CASE de modelagem de software.

Figura 1 – Metodologia do trabalho.



Fonte: elaborado pela autora

Para desenvolver o protótipo de referência, nos baseamos em todo o conhecimento adquirido através da revisão sistemática da literatura. Isto é, implementamos todos os requisitos de usabilidade encontrados. Além do mais, tivemos como base outras ferramentas CASE existentes. O desenvolvimento do protótipo foi feito de uma forma iterativa e incremental, na qual, a cada produção feita, o protótipo era avaliado e incrementado, para assim, seguir o ciclo. Nós efetuávamos essa avaliação, semanalmente, juntamente com um grupo de pesquisa. Nesse grupo, nós apresentávamos o que havia sido desenvolvido, até então, e os membros do grupo mencionavam o que poderia ser melhorado ou incrementado para a próxima semana.

Na última fase, na avaliação do protótipo de referência, realizamos um estudo empírico para descobrir qual é o nível de usabilidade que os protótipos construídos possuem. Para isso, definimos um planejamento, contendo um escopo e um protocolo a ser seguido.

Em seguida, executamos o estudo empírico e, por fim, avaliamos os resultados obtidos a partir dele.

1.4 Contribuições

A principal contribuição do nosso trabalho é disponibilizar à sociedade um protótipo de ferramenta CASE de modelagem em ambiente web, que possui um bom nível de usabilidade. Para que os diferentes tipos de pessoas, com seus diferentes cargos na área de software, possam ter a oportunidade de melhorar suas experiências com o uso desse tipo de ferramenta. Além dessa, segue abaixo as contribuições secundárias:

- um estudo bibliográfico da literatura sobre usabilidade de ferramentas CASE;
- a definição de quais são os requisitos de usabilidade cruciais para qualquer ferramenta CASE;
- a utilização de uma técnica de avaliação capaz de medir o nível de usabilidade do protótipo;
- o colhimento da opinião de diferentes grupos de pessoas que fazem o uso de ferramentas CASE.

1.5 Organização do Documento

O restante desta monografia está organizada como segue. No Capítulo 2, apresentamos uma visão geral dos principais conceitos e tecnologias relacionadas ao trabalho. No Capítulo 3, reportamos os resultados obtidos com a realização da revisão sistemática da bibliografia. No Capítulo 4, descrevemos como foi realizado todo o processo de construção dos protótipos. No Capítulo 5, mostramos quais foram os resultados alcançados no estudo empírico. Finalmente, no Capítulo 6, apresentamos nossas considerações finais.

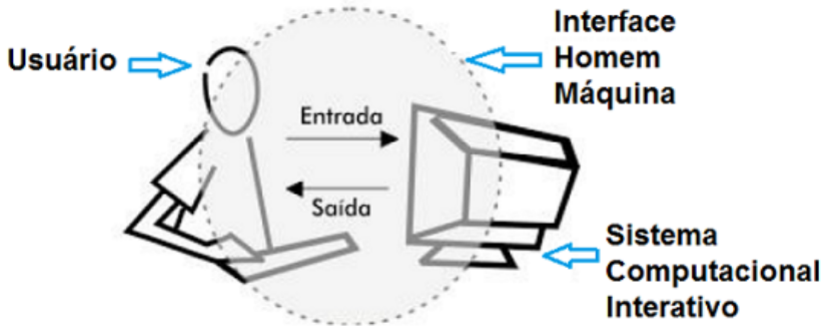
2 FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Neste capítulo, apresentamos uma visão geral dos conceitos e tecnologias relacionados ao trabalho. Na Seção 2.1, nós definimos IHC, e como ela é avaliada. Na Seção 2.2, definimos usabilidade e caracterizamos a engenharia de usabilidade. Na Seção 2.3, descrevemos um processo de prototipação. Na Seção 2.4, explicamos o conceito de Ferramentas CASE. Na Seção 2.5, expomos a ferramenta de criação de protótipos Axure. Para finalizar, na Seção 2.6, apresentamos as lições do capítulo.

2.1 Interação Humano-Computador

Segundo Hix e Hartson (1993) a IHC é dada por qualquer atividade feita em busca de um determinado objetivo que envolva um ser humano e uma máquina. Tal interação é feita através da interface, a qual é responsável por fazer o contato entre o usuário e o sistema (BARBOSA; SILVA, 2010). A interface com o usuário é a relação entre o usuário e o sistema computacional interativo, no qual o usuário emite uma entrada e o computador emite uma saída, conforme mostra a Figura 2.

Figura 2 – Interface de usuário.



Fonte: Augusto (2019)

Preece, Rogers e Sharp (2015) utilizam controles remotos para evidenciar exemplos bons e ruins de IHC. O primeiro deles é o controle remoto que acompanha as *smart* TVs. Esses controles são dotados de botões pequenos, muito coloridos e mal posicionados. Os autores ainda relatam que muitas pessoas, ao sentar na sala para assistir à TV, acabam ficando perdidas ao se depararem com tais botões, inclusive para realizar atividades simples, como pausar ou encontrar o menu principal. Aliás, muitas vezes, torna-se necessário que o indivíduo ponha os óculos para enxergá-los.

Em contraste, os autores citam um bom exemplo de controle remoto, o qual permite uma fácil interação entre o usuário e o dispositivo. Este é nomeado por controle remoto TiVo. Possui características como botões largos, coloridos e com desenhos, sendo clara e logicamente organizados. Além disso, dispõe uma interface de menu que aparece

no monitor. Tem um formato de amendoim do tamanho da palma da mão de uma pessoa, facilitando ao usuário a sua utilização, sem ficar perdido ou ter que colocar óculos.

O exemplo acima ilustra a importância da IHC na engenharia de software pois, através dela, é possível garantir uma maior produtividade, diminuição de erros e uma redução do custo de treinamentos, do suporte técnico ou até mesmo do desenvolvimento, no que diz respeito a um sistema computacional (BARBOSA; SILVA, 2010).

2.1.1 Avaliação de IHC

A avaliação de IHC é uma atividade muito importante no ciclo de vida de um produto de software, visto que ela possibilita encontrar falhas que, caso corrigidas, permitem produzir interfaces que possuam boa qualidade na perspectiva do usuário final. Essa atividade é responsável por permitir que o avaliador faça um julgamento de valor sobre a qualidade de uso do sistema em questão, além de permitir que o mesmo possa identificar possíveis erros na interface e problemas no que diz respeito a interação do usuário com o sistema. (BARBOSA; SILVA, 2010)

Prates e Barbosa (2003) citam alguns conceitos de qualidade de uso, como:

- **usabilidade:** está relacionada à facilidade de uso;
- **comunicabilidade:** refere-se à capacidade que os usuários têm de entender o sistema tal como foi projetado;
- **aplicabilidade:** tem relação com a utilidade do sistema interativo em determinados problemas e situações.

Nielsen (1994) propõe um método de avaliação que se resume a uma lista de problemas de usabilidade de interfaces. Essa lista é chamada de *severity ratings* ou classificação de severidade e é encarregada de priorizar os problemas de usabilidade presentes nas interfaces. Conforme o autor, essa classificação de severidade se aplica da seguinte forma:

- 0 = isto não é um problema de usabilidade.
- 1 = é apenas um problema cosmético - não precisa ser corrigido a menos que haja tempo extra disponível no projeto.
- 2 = menor problema de usabilidade - consertar isso deve receber prioridade baixa.
- 3 = grande problema de usabilidade - importante para corrigir, por isso deve ter prioridade alta.
- 4 = catástrofe de usabilidade - indispensável corrigir isto antes que o produto seja liberado.

2.2 Usabilidade de Software

A NBR ISO 8402 (1994) trata a qualidade de produto, processo ou organização como o conjunto de atributos que essas entidades possuem que são capazes de acatar as necessidades explícitas e implícitas de um usuário. A qualidade de produtos de software ou qualidade de software está ligada à satisfação do usuário. Isso quer dizer que ela está relacionada ao grau de satisfação do utilizador em servir-se de tal produto (DENNING, 1992). A usabilidade é uma das características de qualidade de software que deve ser levada em conta na construção de um produto de software de boa qualidade (ISO/IEC 9126-1, 2001).

Para Nielsen (1994), usabilidade é o termo utilizado para descrever a facilidade que o usuário tem em realizar alguma atividade através de uma interface de forma interativa, juntamente com a sua satisfação em realizá-la. Barbosa e Silva (2010), além de concordarem com isso, dizem que a usabilidade trata os aspectos humanos como a cognição, a capacidade de interação com a interface e a capacidade que o usuário tem de captar o retorno emitido pelo sistema computacional interativo.

Para entender melhor o que é usabilidade e o que a falta dela ocasiona, Norman (2013) cita um exemplo:

> "Eu empurro portas que devem ser puxadas, puxo portas que deveriam ser empurradas, e entro de cara em portas que deveriam correr sobre trilhos. Além disso, vejo outras pessoas terem as mesmas dificuldades — dificuldades desnecessárias." (NORMAN, 2013)

O autor se refere às situações do dia-a-dia em que todos passam e que carecem de usabilidade, isto é, de uma facilidade de uso.

Na engenharia de software, especialmente, a usabilidade é exercida nos produtos de software gerados. A fim de garantir que seja possível desenvolver um produto de software no qual a usabilidade esteja presente, Jakob Nielsen, o conhecido como "pai da usabilidade", propôs 10 heurísticas de usabilidade, que são recomendações a serem seguidas na construção de produtos de software para que ele seja considerado de boa usabilidade. Para Nielsen (1995), um sistema de software deve conter:

1. **visibilidade do estado do sistema** - o sistema deve sempre situar o usuário sobre o que está acontecendo;
2. **correspondência entre o sistema e o mundo real** - o sistema deve falar com usuário na linguagem dele, de forma natural e lógica;
3. **controle e liberdade ao usuário** - o sistema deve fornecer ao usuário as opções de refazer e desfazer;
4. **consistência e padrões** - os usuários não devem se confundir com palavras, situações ou ações acreditando que sejam a mesma coisa;

5. **prevenção de erros** - evitar situações que gerem erros, inclusive pedir confirmação de ações para o usuário é uma boa prática;

6. **reconhecimento em vez de relembrar** - as instruções devem ser visíveis ao usuário, além de serem facilmente recuperáveis;

7. **flexibilidade e eficiência de uso** - o sistema deve permitir que usuários personalizem suas ações frequentes;

8. ***design* estético e minimalista** - o sistema deve conter apenas informações relevantes e necessárias;

9. **suporte para reconhecer, diagnosticar e recuperar-se de erros** - o sistema deve mostrar mensagens de erros em uma linguagem simples, fornecendo uma possível solução;

10. **ajuda e documentação** - a documentação de ajuda deve ser dada de forma fácil, curta, focada na tarefa do usuário e deve listar as etapas concretas a serem executadas.

2.2.1 Engenharia de Usabilidade

Mayhew e Mayhew (1999) definiram um processo chamado de engenharia de usabilidade, a qual provê recursos de IHC que contribuem para a criação de um bom sistema interativo. Esse processo consiste em basicamente três etapas: análise de requisitos, *design*/avaliação/desenvolvimento e instalação.

A primeira fase, a análise de requisitos, é onde são realizadas as seguintes atividades:

- definir metas de usabilidade de acordo com o perfil de cada usuário;
- fazer análise das tarefas, possibilidades e limitações da plataforma em que o sistema será executado;
- definir os princípios gerais de *design* de IHC.

Já na fase de *design*/avaliação/desenvolvimento é onde é criado um sistema interativo de acordo com as metas de usabilidade definidas na fase anterior. Esse sistema interativo precisa ser projetado dividindo-se em três níveis:

1. o *designer* deve fazer a reengenharia do trabalho, projetar o modelo conceitual, criar protótipos do modelo conceitual e avaliar interativamente o modelo conceitual;

2. o *designer* deve definir os padrões de *design* de tela, criar protótipos dos padrões de *design* de tela e avaliar interativamente os protótipos de tela;

3. o *designer* deve criar o *design* detalhado da interface com usuário e avaliá-lo.

A última etapa é a instalação, que é responsável por colher as opiniões do usuário para que eles possam deixar registradas as possíveis melhorias que serão aplicadas nos próximos sistemas interativos (BARBOSA; SILVA, 2010).

2.3 Prototipação

Segundo Ambler (2004), prototipação é um processo responsável por construir protótipos de interface de usuário de sistemas. Esses possuem várias finalidades, como:

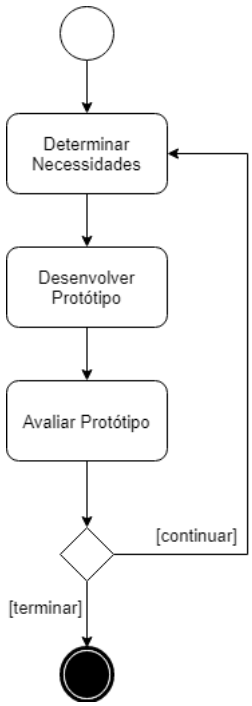
- artefato de análise que permite a exploração de problemas com os *stakeholders* do projeto;
- artefato de requisitos para ter uma visão inicial do sistema;
- artefato de *design* que permite a exploração de soluções do sistema;
- veículo que permite a comunicação de *designs* para a interface de usuário do sistema;
- base inicial para continuar a desenvolver o sistema.

Conforme a Figura 3, o processo de prototipação é constituído por três etapas. A primeira delas é determinar as necessidades da interface de usuário, na qual é investigado e definido o que os usuários deste produto realmente desejam. A segunda, é o desenvolvimento do protótipo, que inicialmente pode ser feito a mão e, posteriormente, melhorado por meio de ferramentas próprias para isso. Já na última, é realizada a avaliação desse protótipo. Após a sua construção, o protótipo deve ser avaliado pelos *stakeholders* a fim de garantir que ele atende, de fato, os requisitos definidos na primeira etapa. Este processo é feito iterativa e incrementalmente, até que não haja mais nada a ser adicionado ou modificado.

2.4 Ferramentas CASE

Na engenharia de software, as ferramentas CASE são de extrema importância, visto que são softwares que podem ser utilizados tanto nas etapas de desenvolvimento quanto manutenção de sistemas (ISO/IEC 14102, 2008). Os benefícios fornecidos pelas ferramentas são diversos. No entanto, os que mais se destacam são: a capacidade de *feedback* ao usuário em termos de mudanças de requisitos, o aumento de produtividade no desenvolvimento de produtos de software, a diminuição do tempo gasto no desenvolvimento de software, a qualidade do sistema, a padronização, a opção de substituir pessoas em projetos e a capacidade de resolver problemas grandes e complicados (CASE, 1985; FREEDMAN, 1986; STAMPS, 1987).

Figura 3 – Processo de prototipação.



Fonte: Ambler (2004)

Segundo Weinrich (1999), nos anos 80, foram feitos estudos sobre problemas de software, os quais mostraram que a maioria dos problemas encontrados estavam presentes nas etapas de análise de requisitos e especificação de projetos. Além disso, na etapa de implementação de software a maioria dos erros eram causados por má codificação, tornando então necessário o uso de ferramentas CASE para auxiliá-los e minimizar esses problemas.

Dentro do ciclo de vida de um projeto de software, mais especificamente nas fases de análise e projeto, essas ferramentas permitem que o modelo de software seja construído, contendo o fluxo de controle dos dados, o conteúdo dos dados, as representações de processos, as especificações diversas de controles e os outros mecanismos pertencentes à modelagem de software. (WEINRICH, 1999)

Há várias ferramentas CASE no mercado, porém, como este trabalho é voltado a ferramentas CASE de modelagem, temos alguns exemplos desse tipo: UML *Designer*[1], Papyrus[2] e Modelio[3] – ferramentas de softwares livres, IBM Rational Rose[4], Enterprise Architect[5] e Astah Professional[6] – ferramentas de softwares proprietários.

[1] http://www.umldesigner.org/
[2] https://www.eclipse.org/papyrus/
[3] https://www.modelio.org/
[4] https://www.ibm.com/br-pt/products
[5] https://www.sparxsystems.com.au/products/ea/index.html
[6] http://astah.net/editions/professional

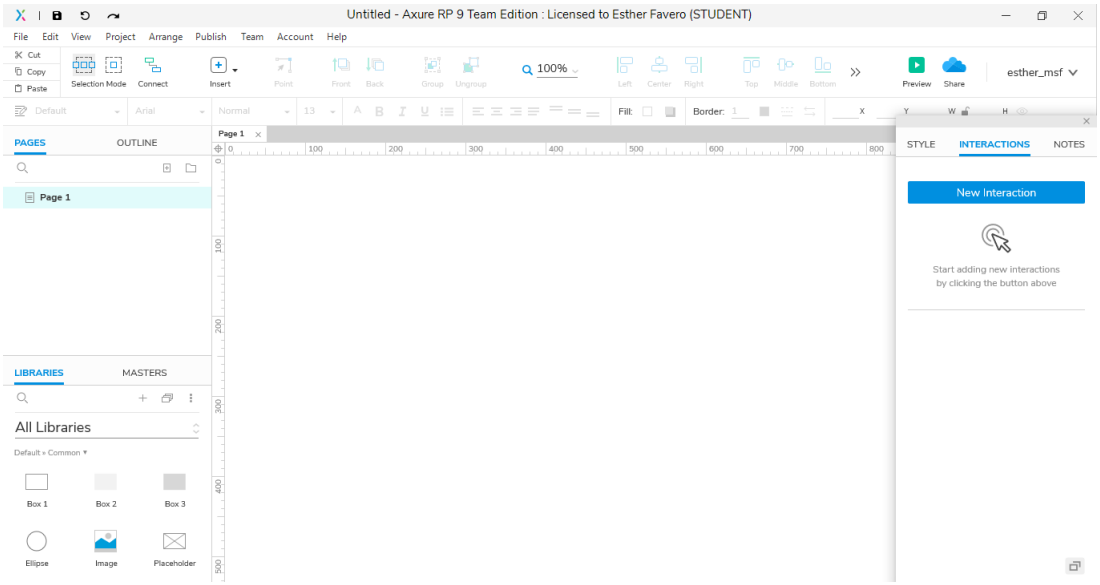
2.5 Axure RP 9

Como pode ser visto na Figura 4, Axure é uma ferramenta de criação e documentação de protótipos que não precisa do uso de código. Ela é utilizada por grandes empresas, como a Amazon, a BCC, a Microsoft, entre outras. Além de tudo, ela permite documentar problemas e enquadrá-los todos em uma mesma página.

Nós sabemos que uma boa forma para testar projetos de software é através de protótipos. Sabendo disso, a Axure criou mecanismos que deixam os protótipos funcionais e interativos, com o intuito de ajudar na hora de tomar as decisões de projeto.

Os protótipos desenvolvidos na ferramenta são documentados e podem ser utilizados por desenvolvedores, visto que a ferramenta permite a exportação de código (AXURE, 2019).

Figura 4 – Ferramenta Axure RP 9.



Fonte: Elaborado pela própria autora.

2.6 Lições do Capítulo

Nesse capítulo, fundamentamos os conceitos que envolvem o tema deste trabalho. Dentre eles, o conceito de IHC foi explicado, o qual consiste na interação que ocorre entre homem e máquina. Vimos também a concepção de Avaliação de IHC, que elucida como avaliar tal interação entre usuário e sistema, verificando o nível de qualidade de uso do produto desenvolvido. Uma das formas de avaliar a IHC foi apresentada no texto, o *"severity ranting"*, desenvolvido por Nielsen (1994).

Além desses, foi feita uma relação entre os conceitos de Usabilidade de Software e satisfação do usuário e qualidade do produto. Citamos as 10 heurísticas de Nielsen para determinar os atributos que um produto de software deve ter em termos de qualidade de

uso. Ainda falando sobre usabilidade, apresentamos o termo Engenharia de Usabilidade, que consta em três etapas: análise de requisitos, *design*/avaliação/desenvolvimento e instalação a fim de garantir um sistema interativo.

Apresentamos também o processo de prototipação, que consiste em descobrir as necessidades que os usuários desejam na interface do sistema, em desenvolver o protótipo e avaliá-lo. Todo o processo é feito iterativo e incrementalmente.

Ademais, esclarecemos que ferramentas CASE são instrumentos responsáveis por auxiliar no desenvolvimento e manutenção de software.

Por último, descrevemos as características da ferramenta de prototipação, Axure, responsável por criar protótipos interativos de alta fidelidade.

3 REVISÃO DO ESTADO DA ARTE

Neste capítulo reportamos a execução e os resultados de uma revisão sistemática sobre usabilidade em ferramentas CASE. Na Seção 3.1, mostramos outras revisões da literatura que são similares ao tema do nosso trabalho. Já na Seção 3.2, expomos o protocolo da revisão, o qual foi constituído baseado no modelo elaborado por Kitchenham e Charters (2007). Na Seção 5.3, apresentamos os resultados da pesquisa. Finalmente, na Seção 3.4, apresentamos as lições do capítulo.

3.1 Outras Revisões da Literatura

Efetuamos uma pesquisa nas bases de dados Scopus[1], a IEEE Digital Library[2] e a ACM Digital Library[3], com o intuito de descobrir outras revisões sistemáticas da literatura que possuíssem relação com o tema "usabilidade de ferramentas CASE". No entanto, o único trabalho que encontramos foi o do Lima (2017) – "Um Projeto de Interface para Ferramentas CASE de Modelagem UML Baseadas em Eclipse".

A revisão de literatura executada por Lima (2017) tinha como intuito descobrir as seguintes questões de pesquisa: "quais são as estratégias que uma ferramenta CASE necessita seguir para melhorar a sua usabilidade para desenvolvedores?" e "como essas estratégias foram avaliadas?". No total, foram selecionados 7 artigos, dos quais responderam às questões de pesquisa. Em resposta para a primeira questão, foi mencionado que uma das estratégias existentes na literatura é a adaptação de interfaces de usuário, ou seja, construir sistemas interativos que promovam uma adaptação de acordo com o perfil de cada usuário. Em resposta à segunda questão, foi feito uma avaliação com usuários a fim de monitorar seu desempenho ao utilizar interfaces adaptativas.

No entanto, pelo fato de não haver nenhum trabalho que revelasse, de fato, quais são as características, no aspecto de usabilidade, que as ferramentas CASE devem ter, decidimos fazer uma nova revisão sistemática.

3.2 Protocolo da Revisão

O propósito da nossa pesquisa é investigar o estado da arte relacionado à usabilidade em ferramentas CASE.

3.2.1 Questões de Pesquisa

Nesta revisão, buscamos entender os problemas relacionados à usabilidade em ferramentas CASE, as características requeridas para uma boa usabilidade e como isso

[1] https://www.scopus.com/home.uri
[2] https://ieeexplore.ieee.org/Xplore/home.jsp
[3] https://dl.acm.org/

tem sido desenvolvido nas ferramentas CASE. Com isso, definimos três questões de pesquisa a serem buscadas na literatura:

- **RQ-1. Qual é a motivação para melhorar a usabilidade de ferramentas CASE?**
- **RQ-2. Quais são os requisitos de usabilidade que têm sido definidos para ferramentas CASE?**
- **RQ-3. Como a engenharia de usabilidade tem sido usada no desenvolvimento de ferramentas CASE?**

3.2.2 Processo de Busca

O processo de busca definido para nossa pesquisa foi o *Snowballing*. *Snowballing* é um processo que utiliza a lista de referências de um artigo e suas citações a fim de encontrar novos artigos. A atividade executada para encontrar artigos a partir da lista de referências se denomina *backward snowballing*, já a atividade para encontrar artigos a partir das citações do mesmo, se denomina *forward snowballing* (WOHLIN, 2014).

O processo inicia por uma semente, isto é, um conjunto de artigos relacionados ao tema da pesquisa. A semente utilizada para o nosso trabalho foi o conjunto de trabalhos resultantes da revisão feita por Lima (2017), o qual possuía 7 artigos.

3.2.3 Processo de Seleção

O processo de seleção é composto por três tipos de critérios: os critérios de inclusão, os critérios de exclusão e os critérios de qualidade. Definimos somente um critério de inclusão com a finalidade de englobar artigos que realmente são relacionados ao tema do nosso trabalho. O critério definido é:

- são selecionados os estudos primários completos que tenham menção explícita a uma solução orientada à usabilidade em ferramentas CASE no título, palavras chaves ou resumo.

Nós definimos quatro critérios de exclusão, os quais possibilitaram a filtragem de artigos válidos para o nosso estudo. Os critérios de exclusão são os seguintes:

1. não são selecionados estudos secundários ou terciários;
2. não são aceitos trabalhos diferentes da língua inglesa;
3. não são aceitos trabalhos duplicados;
4. não são aceitos trabalhos com menos de 5 páginas.

Enfim, determinamos o critério de qualidade para selecionar apenas artigos que contivessem algum método científico que comprovasse seu estudo, sendo ele:

- o estudo avaliou sistematicamente aquilo que foi proposto.

3.2.4 Extração dos Dados

Com o propósito de extrair os dados dos artigos selecionados, criamos uma ficha de leitura contendo informações de identificação dos artigos e informações que respondiam às questões de pesquisa. A ficha de leitura é composta pelas seguintes atributos:

- ID;
- título;
- autor(es);
- data da publicação;
- trecho(s) do texto que apresenta(m) qual a motivação para melhorar a usabilidade de ferramentas CASE;
- trecho(s) do texto que apresenta(m) os requisitos de usabilidade definidos para ferramentas CASE;
- trecho(s) do texto que apresenta(m) como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

3.2.5 Estratégia de Análise

A estratégia escolhida para analisar os dados foi a Análise de Conteúdo. Trata-se de um método de pesquisa que permite analisar dados sistematicamente de quaisquer tipos de documentos. Além disso, ela é capaz de proporcionar uma interpretação de dados que em uma leitura comum não seria possível (MORAES, 1999). A análise de conteúdo é composta por 5 etapas:

1. **Preparação:** fazer uma leitura dos documentos a serem analisados, selecionar os conteúdos que são úteis para a análise e codificá-los;
2. **Unitarização:** reler os trechos selecionados na etapa anterior e unitarizar os conteúdos, isto é, separar os trechos em unidades menores e codificá-los;
3. **Categorização:** agrupar as unidades de análise definidas na etapa anterior de acordo com a semelhança entre elas;
4. **Descrição:** sintetizar as categorias expressando seus respectivos significados;
5. **Interpretação:** fundamentar as categorias sintetizadas.

3.3 Resultado da Revisão

Conforme antes citado, para iniciar o processo de busca foi utilizado o trabalho do Lima (2017) como a semente do *snowballing*. O conjunto de artigos continha o total de 7 trabalhos. Aplicamos o processo de seleção nesse conjunto de artigos. Executamos o critério de inclusão e 2 artigos foram excluídos. Depois, aplicamos os critérios de exclusão e 1 artigo foi excluído. Após, aplicamos o critério de qualidade, no entanto, nenhum artigo foi excluído. A semente, então, conteve 4 artigos.

O *snowballing* é constituído por interações, chamados de *rounds*. Estes *rounds* são compostos pela lista de artigos que foram selecionados com base no *backwards* e *forwards*, tendo em consideração o critério de inclusão. Nós rodamos 5 *rounds*, conforme exibido na Tabela 1, no período de 01 de Maio de 2019 a 08 de Maio de 2019. O primeiro deles de 2 artigos selecionados restou 1. No segundo *round*, de 8 artigos selecionados restou 1. No terceiro *round* de 1 artigo restou 1. No quarto *round*, de 1 artigo selecionado restou 1. E, por fim, no quinto *round*, de 2 artigos selecionados restou 1. Estas 5 interações resultaram em 5 artigos selecionados. Após chegar neste número o processo de seleção foi aplicado.

Tabela 1 – Snowballing.

Rounds	**Artigos Revisados**	**Artigos Selecionados**
Round 0	7	4
Round 1	2	1
Round 2	8	1
Round 3	1	1
Round 4	1	1
Round 5	2	1
Total	21	9

Fonte: Elaborado pela autora.

Em cada *round*, aplicamos o critério de inclusão para selecionar os estudos relevantes. Após o término das interações, estabelecemos uma lista dos artigos resultantes do *snowballing*. Então, aplicamos os critérios de exclusão nesta lista e nenhum artigo foi excluído, mantendo os 5. Logo após, aplicamos o critério de qualidade nesta lista e novamente nenhum artigo foi excluído.

A lista final dos artigos contém 9 artigos. Dos quais os dados foram extraídos e analisados. A lista final dos trabalhos selecionados está na Tabela 2.

Com a lista final dos artigos delineada, nós aplicamos a análise de conteúdo para que pudéssemos analisar, com profundidade, a opinião de cada autor sobre a usabilidade das ferramentas CASE.

Conforme mencionado anteriormente, na etapa de extração dos dados foi elaborada uma ficha de leitura contendo campos que referenciassem cada artigo junto com partes

Tabela 2 – Artigos selecionados.

Título do Artigo	Autores	Ano
An empirical analysis of the evolution of user-visible features in an integrated development environment	Hou, D. e Wang, Y.	2009
Cedar studio: an IDE supporting adaptive model-driven user interfaces for enterprise applications	Akiki, P.; Bandara, A. e Yu, Y.	2013
Developing an adaptive user interface in eclipse	Leung, A.; Morisson, S.; Wringe, M. e Zou, Y.	2006
Engineering adaptive model-driven user interfaces	Akiki, P.; Bandara, A. e Yu, Y.	2016
EyeDE: gaze-enhanced software development environments	Glücker, H.; Raab, F.; Echtler, F. e Wolff, C.	2014
Improving the usability of Eclipse for novice programmers	Storey, M.; Damian, D.; Michaud, J.; Myers, D.; Mindel, M.; German, D.; Sanseverino, M. e Hargreaves, E.	2004
Integrated Development Environments (IDEs) for Novice Users	Zou, Y.; Lerner, M.; Leung, A.; Morisson, S. e Wringe, M.	2008
U can touch this: touchifying an ide	Biegel, B.; Hoffmann, J.; Lipinski, A. e Diehl, S.	2014
Usability of a Domain-Specific Language for a Gesture-Driven IDE	Bačíková, M.; Maričák, M. e Vančík, M.	2015

Fonte: Elaborado pela autora.

do texto que respondessem as questões de pesquisa. Tal ficha de leitura foi feita em uma planilha. Esta planilha serviu como alicerce para a primeira etapa da análise de conteúdo, a denominada Preparação, visto que todos os conteúdos relevantes dos artigos já estavam ali organizados.

Na etapa de Unitarização, nós utilizamos a mesma planilha, porém fizemos uma análise dos dados e codificamos aqueles que expressaram melhor as respostas das questões de pesquisa.

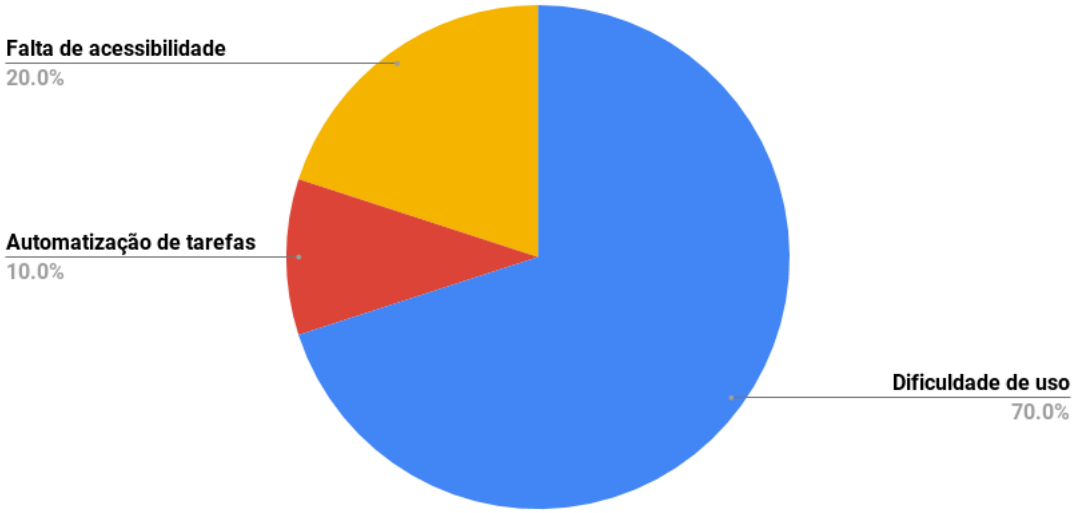
Já nas etapas 3 e 4, que são Categorização e Descrição, nós agrupamos todos os conteúdos que tiveram semelhança de acordo com cada questão de pesquisa. Após o agrupamento dos dados, nós fizemos a descrição de cada agrupamento. Para isto, nós utilizamos outra planilha contendo uma cédula para cada questão de pesquisa.

Na última etapa, a Interpretação, nós interpretamos os dados a fim de responder as três questões de pesquisa.

3.3.1 Qual é a motivação para melhorar a usabilidade de ferramentas CASE?

Como pode ser visto na Figura 7, a maior motivação para melhorar a usabilidade de ferramentas CASE é porque elas possuem uma alta complexidade e um nível elevado de dificuldade de uso. Leung et al. (2006), Glücker et al. (2014) e Akiki, Bandara e Yu (2013) afirmam que o grande número de funcionalidades existentes nas interfaces de usuário destas ferramentas causa confusão ao utilizá-las. Akiki, Bandara e Yu (2016) ainda afirmam que aplicativos de software em geral sofrem de grandes problemas de usabilidade. Biegel et al. (2014) dizem que *Integrated Development Environment* (IDE) modernas

Figura 5 – Motivação para melhorar a usabilidade.



Fonte: Elaborado pela autora.

possuem atalhos complexos e funcionalidades ocultas, tornando difícil a navegação no projeto. Ainda, Storey et al. (2003) julgam a interface do Eclipse como sendo muito complexa para a aprendizagem. Zou et al. (2008) relatam que, na maioria das IDEs, todas as funcionalidades aparecem em todos os momentos para os usuários, deixando desorganizada a interface de usuário.

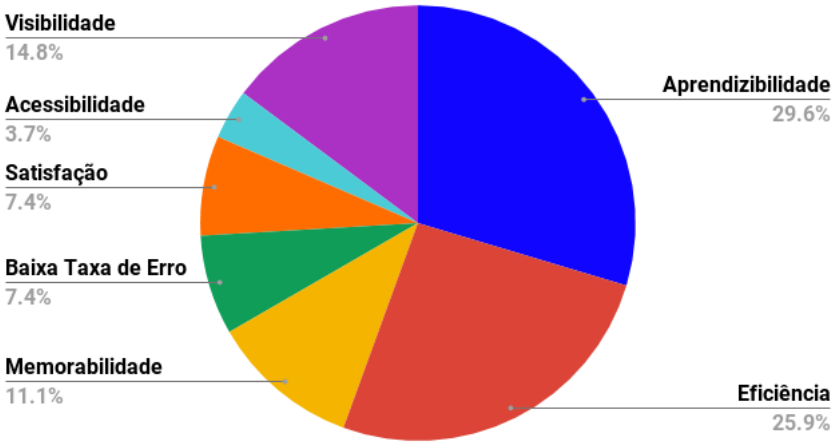
Alguns artigos alegam que a falta de acessibilidade das ferramentas CASE de modelagem também é um problema. Baćíková, Marićák e Vanćík (2015) justificam que as IDEs atuais não estão preparadas para atender às necessidades como toques na tela, pois estas se limitam ao uso de mouses e teclados. Eles ainda alegam que só existem duas ferramentas atualmente que possuem esta opção, que é o Eclipse e o IntelliJ IDEA, porém ambos disponibilizam somente funcionalidades básicas.

Hou e Wang (2009) explicam que as ferramentas CASE deveriam ter um bom nível de usabilidade visto que elas são ferramentas que permitem a automatização de tarefas de rotina e uma redução da carga congnitiva para que a engenharia de software de fato seja feita de uma forma mais produtiva.

3.3.2 Quais são os requisitos de usabilidade que têm sido definidos para ferramentas CASE?

A análise de conteúdo nos permitiu verificar quais eram os requisitos de usabilidade que mais eram mencionados nos trabalhos. A Figura 6 apresenta os 7 requisitos definidos nos artigos. O requisito mais discutido nos trabalhos é a aprendizibilidade, isto é, o quanto as ferramentas CASE são capazes de fornecer a facilidade de aprender alguma funcionalidade (LEUNG et al., 2006). Além de Leung et al. (2006), outros autores men-

Figura 6 – Requisitos de usabilidade.



Fonte: Elaborado pela autora.

cionam a importância da aprendizibilidade, como Hou e Wang (2009), Baćíková, Marićák e Vancík (2015) Akiki, Bandara e Yu (2013), Biegel et al. (2014) e Storey et al. (2003).

Em segundo lugar, está a eficiência. Os artigos argumentam que uma ferramenta CASE deve ter eficiência – responsável por medir a facilidade de uso e o nível de produtividade que o usuário deve atingir (LEUNG et al., 2006). Os autores que mencionam este requisito são: Leung et al. (2006), Hou e Wang (2009), Akiki, Bandara e Yu (2016) e Zou et al. (2008).

Os autores Hou e Wang (2009) e Storey et al. (2003) falam que a visibilidade é um fator bastante relevante, já que o usuário deve sempre estar informado sobre o que está acontecendo, através do *feedback* do sistema (NIELSEN, 1995).

A memorabilidade também é outro fator importante que deve ser levado em conta no desenvolvimento de ferramentas CASE, visto que o usuário deve lembrar facilmente das funcionalidades existentes. Os trabalhos que defendem este requisito com relevância são os de Leung et al. (2006), Hou e Wang (2009) e Baćíková, Marićák e Vancík (2015).

A baixa taxa de erro é um atributo que garante que o usuário erre menos ao fazer o uso de uma ferramenta CASE. Leung et al. (2006) e Hou e Wang (2009) são os autores que argumentam isto.

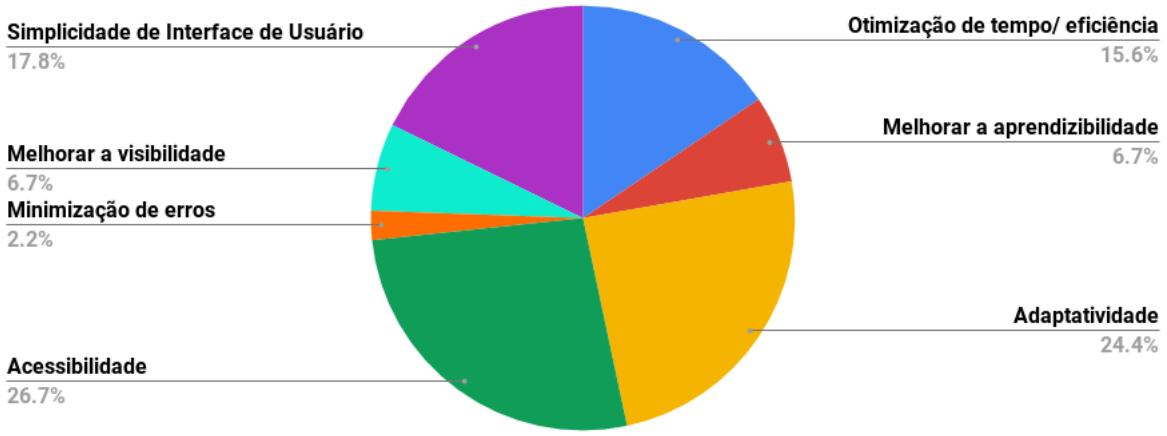
Já a satisfação do usuário ao lidar com a ferramenta também deve ser levada em conta. Os autores Leung et al. (2006) e Biegel et al. (2014) mencionam o requisito satisfação em seus trabalhos.

Por fim, o requisito menos citado nos artigos analisados é a acessibilidade. Glücker et al. (2014) relatam que pessoas que sofrem com alguma lesão devem ter a capacidade de utilizar estas ferramentas.

3.3.3 Como a engenharia de usabilidade tem sido usada no desenvolvimento de ferramentas CASE?

Na questão 2 (ver Subseção 3.3.2), a acessibilidade foi o requisito menos abordado como uma necessidade crucial para o desenvolvimento de ferramentas CASE. Em contra partida, ao analisar como a engenharia de usabilidade tem sido considerada no desenvolvimento de ferramentas CASE, nossa questão 3, a forma de como melhorar a acessibilidade tomou uma grande proporção. O trabalho de Glücker et al. (2014) propõe tal melhoria através de um sistema que permite que pessoas com ou sem deficiência possam ler e navegar no código-fonte somente com os olhos. Para construir esse ambiente os autores dizem que devem evitar distrações, para que os usuários não tomem ações inesperadas. Além disso, o *design* da interface é composto apenas por tons de cinza claro, contendo uma única cor de destaque, o azul claro, para mostrar o status do sistema.

Figura 7 – Utilização da Engenharia de Usabilidade.



Fonte: Elaborado pela autora.

Outra proposta de acessibilidade oferecida é uma ferramenta que permita gesto multitoque. Biegel et al. (2014) relata que este tipo de acessibilidade permite uma interação direta com o usuário. E, para que esta interação ocorra, os autores dizem que a interface de usuário original deve ser modificada o mínimo possível, deixando o usuário habilitado a usar o mouse e teclado caso ele deseje. Os gestos devem ser intuitivos e comuns, como os utilizados em *tablets* e *smartphones*. Além do mais, todas as refatorações devem ser feitas com um único gesto.

O artigo de Baćíková, Marićák e Vancík (2015) também apresenta uma maneira de melhorar a acessibilidade. Os autores apontam que sistemas com interfaces de toque são fáceis e naturais. Eles propõem Gestos de Propósito Geral (GPGs), isto é, opções de toque simples, toque duplo, deslize e aperto. Além destes, os gestos desenhados, os quais permitem que o usuário desenhe um círculo que selecione vários itens em uma lista, por exemplo.

Depois da acessibilidade, a adaptatividade tem sido bastante considerada no desenvolvimento de ferramentas CASE. Visto que ferramentas que se adaptam com o perfil de cada usuário proporcionam uma maior produtividade. Leung et al. (2006) e Zou et al. (2008) propõem ferramentas que, após a coleta de dados suficientes sobre quais são as funcionalidades que os usuários mais utilizam ou não, um *plug-in* oculta as funcionalidades menos usadas. Hou e Wang (2009) desenvolveram uma ferramenta que disponibiliza a opção do programador personalizar o IDE de acordo com as suas preferências, colocando opções de refazer/desfazer, marcadores, histórico de navegação do editor e lista de arquivos usada recentemente, por exemplo. Akiki, Bandara e Yu (2016) propuseram um método chamado de *Role-Based UI Simplification* (RBUIS), o qual fornece ao usuário um conjunto mínimo de recursos de acordo com o contexto de uso. Storey et al. (2003) criaram a ferramenta chamada *Gild*, voltada à aprendizagem, que permite que alunos e instrutores possam adicionar ou excluir recursos. Akiki, Bandara e Yu (2013) ainda desenvolveram uma ferramenta capaz de projetar modelos de tarefas e atribuir funções a eles através de uma caixa de diálogo.

A simplicidade de interface de usuário também tem sido empregada nas ferramentas CASE. Akiki, Bandara e Yu (2016) apresentam um mecanismo que transmite ao usuário uma interface simples com recursos mínimos e um *layout* ideal para o perfil de cada usuário. Storey et al. (2003) dizem que refatoraram a visualização de tarefas para que apenas os erros do projeto em andamento apareçam. Ademais, para executar o projeto é necessário apenas um clique, o código é depurado passo a passo, as opções de importar e exportar são simplificadas e os números de linha são padrão. Já Akiki, Bandara e Yu (2013), propõem *scripts* dinâmicos que otimizam o *layout* das interfaces de usuário e uma funcionalidade que diminui o conjunto de recursos para um determinado contexto.

Para otimizar o tempo e aumentar a eficiência no uso das ferramentas, Leung et al. (2006) utilizam uma métrica que calcula o tempo médio que um usuário leva para encontrar um elemento qualquer no menu. Diminuindo este tempo, a eficiência do menu é aumentada. Hou e Wang (2009) criaram integrações de ações que fornecem ao usuário mais eficiência, como por exemplo, usar menus de contexto. Outra forma de elevar a eficiência é por meio da automação de várias ações de uma só de vez, por exemplo, importar vários projetos em uma única etapa. Do mesmo modo, salvar processos e dados que demoram a serem criados, a fim de serem utilizados posteriormente, gera um aumento na eficiência. Já no caso de ferramentas que possuem a opção de toque na tela, trazer a possibilidade de configurar toques que sejam similares aos de *smartphones* e *tablets* otimizam o tempo (BIEGEL et al., 2014). E no contexto da acessibilidade, as ferramenta que utilizam apenas os olhos para fazer algumas tarefas, dispensam a necessidade do programador estar movendo repetidamente suas mãos entre o teclado e o mouse (GLÜCKER et al., 2014).

Com o intuito de melhorar a visibilidade, Hou e Wang (2009) colocam informações locais que não sejam estruturais, como por exemplo mostrar uma assistência rápida para uma linha de código específica do editor. Outra maneira proposta pelos autores é usar recursos com informações estruturais. Ao passar o mouse sobre o nome de uma determinada classe, por exemplo, o *Javadoc* seria mostrado. Storey et al. (2003) abordam uma distinção entre recursos através de decoradores, na qual um arquivo *.class* teria um decorador diferente de um *.java*.

Objetivando aperfeiçoar a aprendizibilidade, Leung et al. (2006) dizem que mover os elementos mais clicados para o topo do menu, aumenta a capacidade de aprender e a eficiência de uso. Já Hou e Wang (2009), apresentam uma ideia de que fornecer ao usuário documentação de ajuda é uma boa prática. Ainda, expõem a inserção de um navegador HTML que vincule e inclua dinamicamente o código-fonte.

Finalmente, para diminuir a taxa de erro cometida pelos usuários, Hou e Wang (2009) explicam que inserir o mesmo recurso a novos dados ou criar um recurso semelhante em um novo contexto pode diminuir esta taxa.

3.4 Lições do Capítulo

Neste capítulo foi apresentado o protocolo de revisão com seus respectivos processos, juntamente com os resultados da pesquisa. Dentre os processos utilizados, a análise de conteúdo foi o método que aplicamos para analisar os dados extraídos de uma forma mais profunda com a finalidade de responder as questões de pesquisa.

Descobrimos que a principal motivação que leva à necessidade de melhorar a usabilidade de ferramentas CASE é justamente a complexidade que as mesmas possuem pelo fato de haver uma variedade de recursos e todos serem disponibilizados, ao mesmo tempo, para todos os tipos de usuários.

Os requisitos de usabilidade mais mencionados nos artigos analisados são a aprendizibilidade, a eficiência, a visibilidade, a memorabilidade, a baixa taxa de erro, a satisfação e a acessibilidade, nesta ordem. Portanto, a maneira como estes requisitos têm sido implementados na construção das ferramentas CASE é por meio de sistemas de gestos, personalização de ferramentas com o mínimo de recursos possíveis, diminuição do tempo que um usuário leva para encontrar um elemento na tela e reutilização de processos a fim de aumentar a eficiência, documentação dinâmica para partes específicas da ferramenta e distinção por meio de decoradores dos recursos existentes, documentação de ajuda e inserção dos mesmos recursos à novos dados.

4 PROTÓTIPO DE REFERÊNCIA PARA CASES WEB

Neste capítulo, apresentamos como foi desenvolvido o nosso protótipo de referência. Um protótipo criado como referência para ser usado por qualquer tipo de ferramenta CASE, visto que ele possui requisitos de usabilidade necessários para que os usuários dessas ferramentas possam aproveitá-las de um forma melhor. Na Seção 4.1, mostramos quais são as necessidades levantas para a construção do protótipo. Na Seção 4.2, explicamos o desenvolvimento em sí do protótipo de referência. Por fim, na Seção 4.3, temos as lições do capítulo.

4.1 Definição das Necessidades

Conforme abordado anteriormente, nós sabemos que a grande maioria das ferramentas CASE existentes no mercado carecem de usabilidade. Esse problema pode ocasionar menor eficiência do processo de modelagem de software, desinteresse dos usuários pela ferramenta e, inclusive, potenciais impactos na qualidade do projeto do software a ser desenvolvido (ZOU et al., 2008).

O objetivo do nosso trabalho é desenvolver um protótipo de referência de ferramenta CASE genérico, que pode ser utilizado por qualquer ferramenta web de modelagem. Para isso, fomos em busca do conhecimento. Realizamos a revisão sistemática a fim de encontrar quais seriam os requisitos de usabilidade mais abordados na literatura. A partir da análise dos trabalhos relacionados, identificamos um conjunto de requisitos de usabilidade que servem para nortear as atividades de projeto. Os requisitos seguem abaixo:

- **RQ1**: visibilidade - corresponde a deixar as funcionalidades da interface visíveis ao usuário, para que ele possa encontrá-las facilmente (REBELO, 2019);
- **RQ2**: satisfação - está relacionado em como o usuário se sente ao utilizar um sistema (KNOR; AVELAR, 2015);
- **RQ3**: baixa taxa de erro - quer dizer que quando usuários fazem o uso de sistemas com baixa taxa de erro, conseguem realizar suas atividades sem dificuldades (KNOR; AVELAR, 2015);
- **RQ4**: memorabilidade - significa permitir ao usuário retornar a um sistema, que não utiliza a algum tempo, e lembrar como executar as atividades (KNOR; AVELAR, 2015);
- **RQ5**: aprendizibilidade - refere-se à facilidade de aprendizado que é transmitida ao usuário para a execução de suas tarefas (KNOR; AVELAR, 2015);
- **RQ6**: eficiência - está associado a quão produtivo o usuário pode ser ao realizar suas atividades (KNOR; AVELAR, 2015).

A acessibilidade foi um requisito também abordado nos trabalhos relacionados. Sabemos que esse requisito é crucial em sistemas interativos. No entanto, por uma questão de complexidade, optamos por explorar a usabilidade em um primeiro momento, deixando então a acessibilidade para ser implementada em trabalhos futuros.

Além desses requisitos, devem ser levados em conta as seguintes restrições:

- a plataforma de desenvolvimento é web;
- a ferramenta permite o trabalho colaborativo.

A partir da definição desses requisitos de usabilidade, acreditamos que qualquer ferramenta CASE de modelagem pode usá-los como referência.

4.2 Desenvolvimento do Protótipo

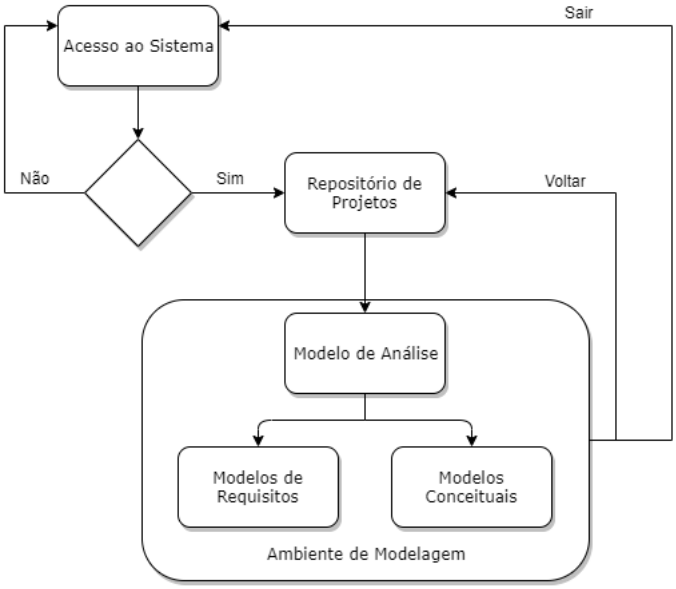
O protótipo de referência que nós elaboramos é uma prova de conceito dos requisitos de usabilidade levantados. Para representar essa prova de conceito, nós escolhemos a Linguagem de Modelagem Unificada (UML). A UML, por ser bastante difundida é, inclusive, um padrão na indústria de desenvolvimento de software, utilizada por grande parte dos engenheiros em projetos orientados a objetos (SIAU; CAO, 2001).

O primeiro passo definido para o desenvolvimento da nossa solução foi um mapa de navegação. Nele é possível estabelecer o comportamento navegacional básico de qualquer aplicação. A Figura 8, mostra como essa navegação é mapeada. Para ter acesso à ferramenta, o usuário precisa fazer o *login* na plataforma. Feito isso, ele é encaminhado para o local onde há o Repositório do Projeto, ou seja, onde estão todos os projetos já criados por ele. Por fim, ele é conduzido para o Ambiente de Modelagem, em que ele pode escolher qual tipo de modelagem ele deseja acessar ou produzir.

O protótipo funcional de alta fidelidade foi elaborado por meio da ferramenta AXURE (confira na seção 2.5). O desenvolvimento dos protótipos foi feito tendo como base os requisitos de usabilidade citados anteriormente. Para que todos os requisitos fossem atendidos, realizamos vários ciclos de desenvolvimento. Em que, a cada avaliação da solução, o protótipo era refatorado e incrementado. Nós desenvolvemos a prototipação, em geral, de acordo com um padrão de cores para os ícones (ver Figura 9), letras, fundos (ver Figura 10) e etc. As cores dos ícones estão expostas na Figura 9.

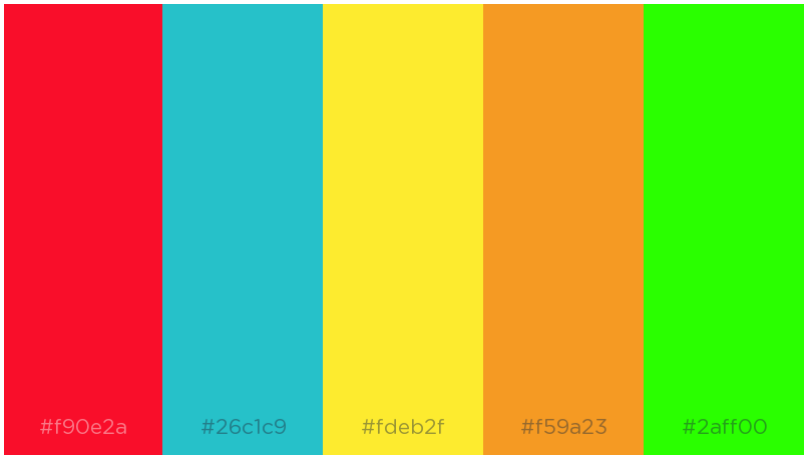
Além disso, definimos um padrão de organização das partes do ambiente de modelagem das ferramentas CASE. Assim, esse padrão deve ser mantido por todos os tipos de modelos, independente de suas propriedades. A Figura 11 exibe como os itens ficaram dispostos no navegador. A seção laranja indica o padrão do cabeçalho web. Já na seção rosa, é o local do cabeçalho da ferramenta. Na seção em vermelho ficam todos os diagramas existentes na ferramenta. Na seção em lilás, são dispostos os elementos referentes ao diagrama selecionado. A seção em verde é pertinente à área de modelagem. A seção

Figura 8 – Mapa de navegação.



Fonte: Elaborado pela autora.

Figura 9 – Paleta de cores dos ícones.



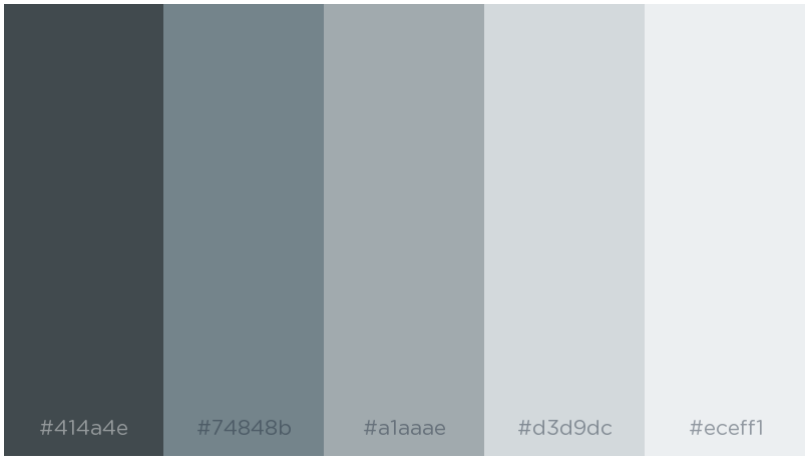
Fonte: Elaborado pela autora.

em azul representa o local da estrutura do navegador do projeto. Por fim, na seção em amarelo, estão estabelecidas as propriedades do modelo selecionado.

A primeira tela que foi prototipada é a de acesso ao sistema. A Figura 12 exibe o protótipo de *login*. Nós mantivemos o padrão de cores, com um *design* minimalista e seguimos o padrão de *login* da maioria das ferramentas web.

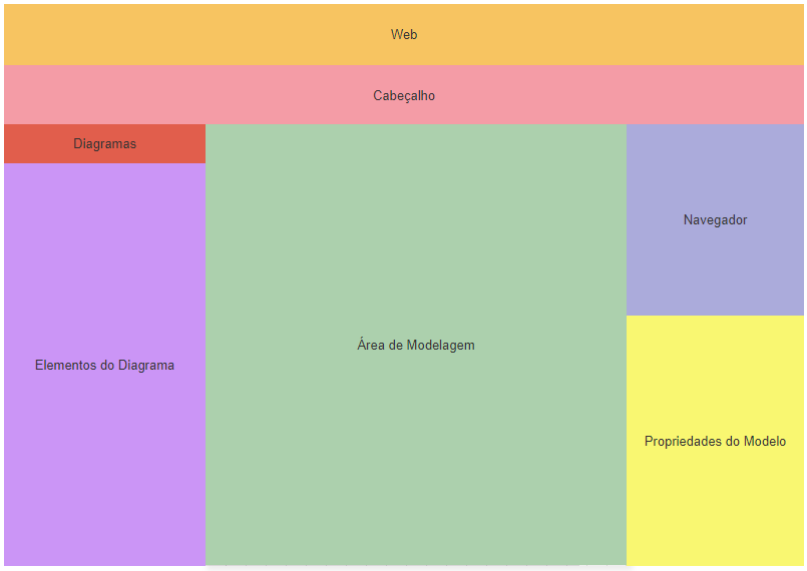
A Figura 13 apresenta a primeira parte do repositório de projetos, que foi desenvolvida na sequência do protótipo de *login*. Nele estão todos os projetos criados pelo usuário, tanto pessoais, quanto compartilhados. O cabeçalho do protótipo contém um campo de busca, dando a possibilidade do usuário aumentar a eficiência (RQ6) pesquisando o projeto que ele deseja manipular. Cada projeto possui uma cor diferente, facilitando a visibilidade

Figura 10 – Paleta de cores dos fundos e letras.



Fonte: Elaborado pela autora.

Figura 11 – Organização do ambiente de modelagem.

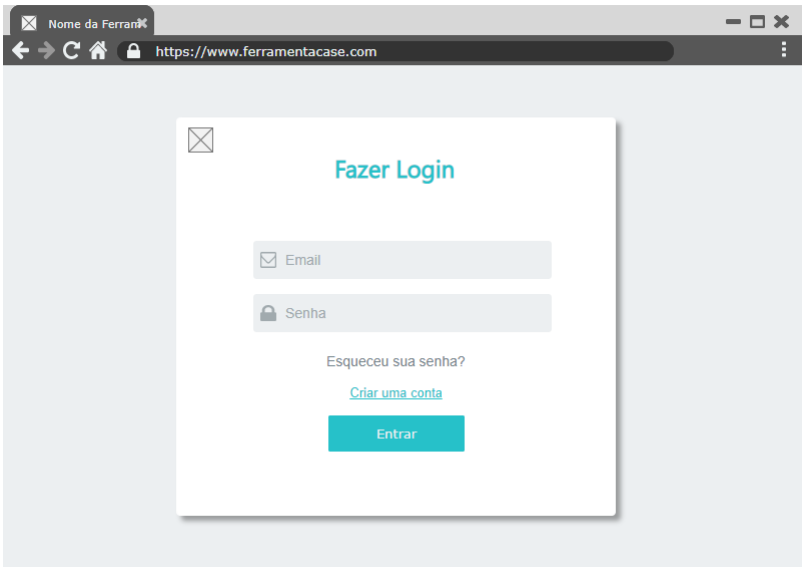


Fonte: Elaborado pela autora.

(RQ1) e a eficiência (RQ6). Aliás, para cada projeto há a data da última modificação e uma opção de edição e exclusão.

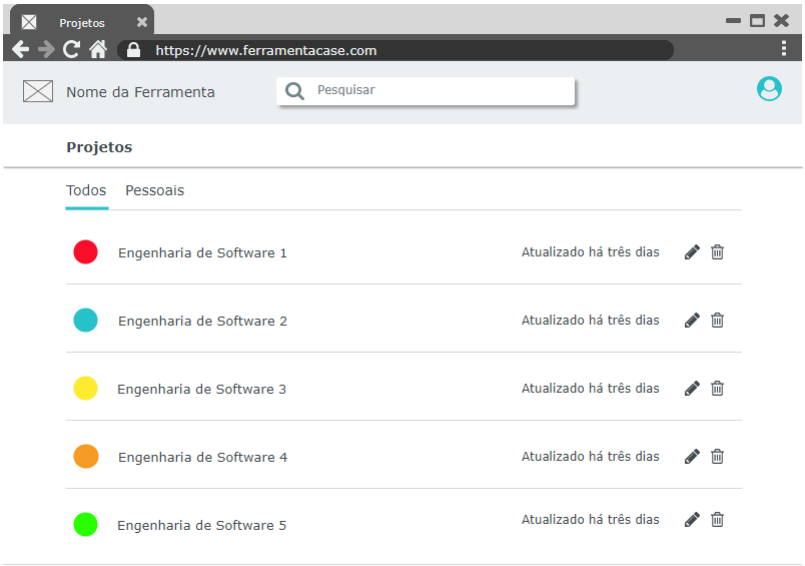
O próximo protótipo criado foi a segunda parte do repositório de projetos. A Figura 14 mostra o protótipo da navegação interna à um projeto já existente. Nele definimos um *design* minimalista, melhorando a visibilidade (RQ1) da ferramenta. Determinamos um local onde o logotipo da ferramenta aparece, juntamente com o nome da ferramenta e uma opção de pesquisa dos modelos. Logo abaixo, indicamos de qual projeto os modelos construídos se tratam, melhorando a memorabilidade (RQ4) do usuário. Definimos uma área onde ficam bem visíveis os modelos criados mais recentemente pelo usuário, aumentando sua eficiência (RQ6). E, abaixo, situamos todos os modelos, divididos de acordo

Figura 12 – Tela de *login*.



Fonte: Elaborado pela autora.

Figura 13 – Tela de repositório de projetos.

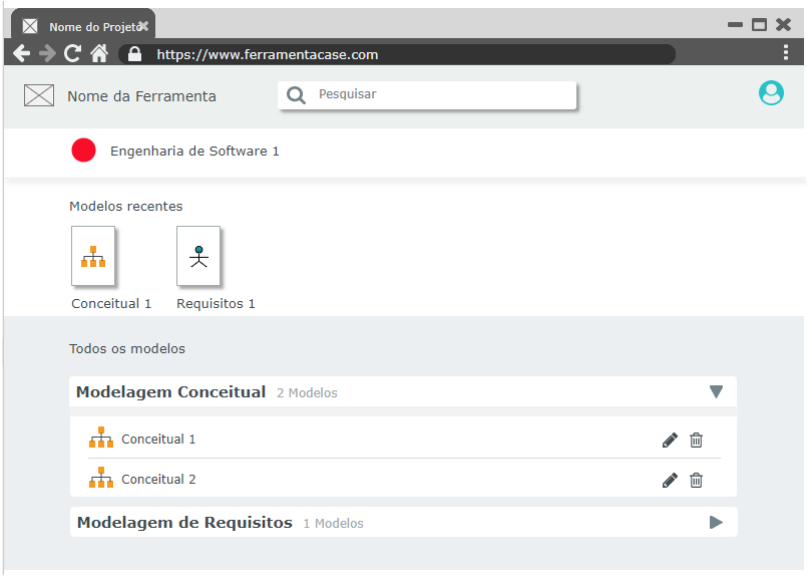


Fonte: Elaborado pela autora.

com seu tipo de modelagem, que melhora a aprendizibilidade (RQ5).

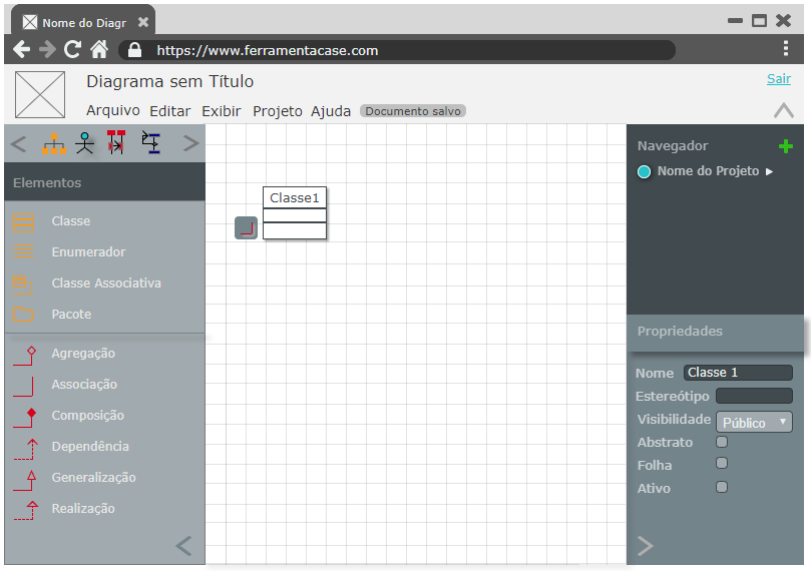
A Figura 15 apresenta como ficou o ambiente de modelagem conceitual. Nós tentamos deixar as funcionalidades ainda mais visíveis, e fugindo um pouco do padrão das ferramentas CASE existentes. Do lado esquerdo nós deixamos apenas os Elementos. Do lado direito nós deixamos o Navegador e as Propriedades. Acreditamos que dessa forma o usuário terá uma interação mais eficiente com as funcionalidades, tornando-as mais facilmente identificáveis e visíveis na interface de uso.

Figura 14 – Tela de modelos de análise.



Fonte: Elaborado pela autora.

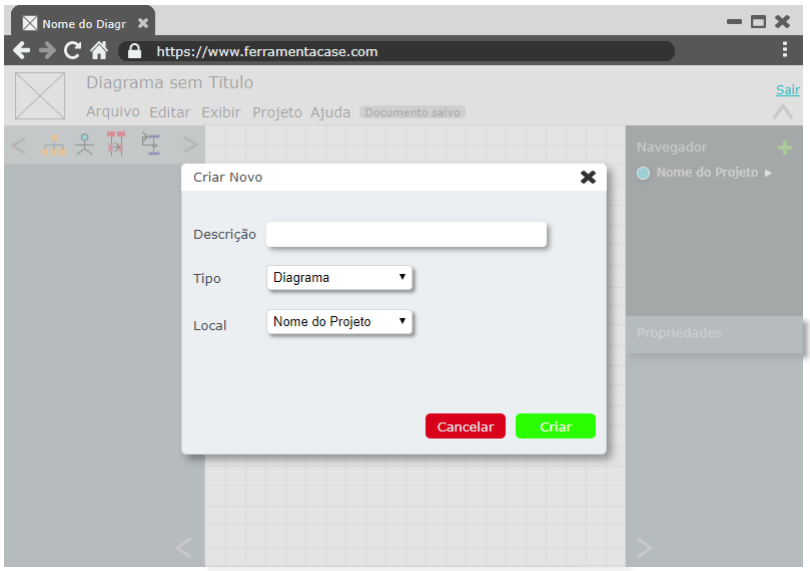
Figura 15 – Tela de modelagem conceitual.



Fonte: Elaborado pela autora.

Além disso, a fim de aumentar a aprendizibilidade (RQ5), colocamos mais de uma opção para criar diagramas, elementos ou pacotes. Assim, o usuário pode escolher qual forma ele mais se agrada para criar seus modelos. Para isso, criamos um botão representado com "+", que transmite a ideia de adição. A Figura 16 exemplifica esse cenário. Criamos uma janela modal, onde todas as informações estão claras e objetivas, diminuindo a taxa de erro do usuário e aumentando a memorabilidade (RQ4), visto que essas informações são fáceis de serem lembradas.

Figura 16 – Protótipo de criação de diagramas, elementos ou pacotes.



Fonte: Elaborado pela autora.

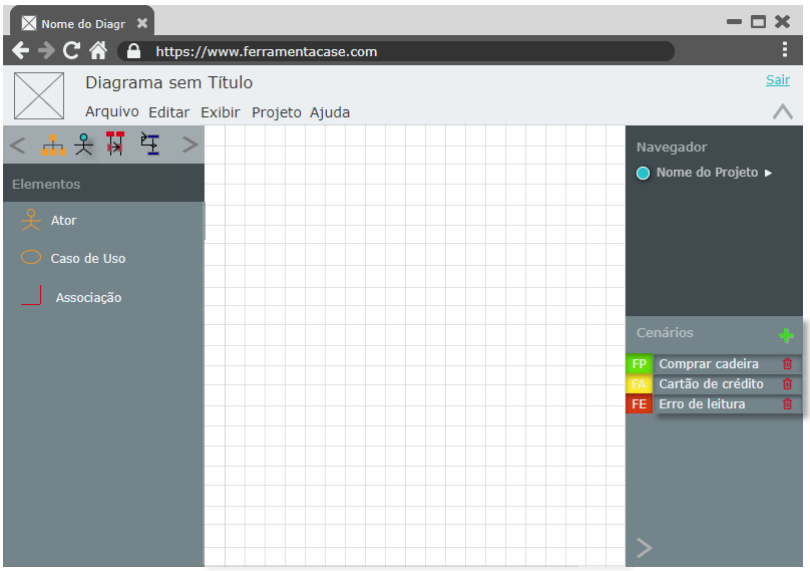
O ambiente de modelagem de requisitos obedeceu o mesmo padrão da modelagem conceitual. Conforme pode ser visto na Figura 17, nós optamos por deixar os elementos de modelagem do lado esquerdo e o navegador do lado direito, com o propósito de tornar a ferramenta consistente, aumentando a aprendizibilidade (RQ5). No entanto, pelo fato da modelagem de requisitos ter características diferentes, criamos uma seção para que o usuário possa adicionar cenários de casos de uso, repetindo o botão "+". Todos os cenários criados localizam-se na seção inferior direita, dando ao usuário a possibilidade de editar ou excluir os cenários já criados. Também, mantendo a consistência da ferramenta e seguindo o padrão, criamos uma modal, que pode ser vista na Figura 18, na qual o usuário pode criar os cenários do seu caso de uso, com sua descrição, o tipo, se ele é fluxo principal, alternativo ou de exceção e seus respectivos passos.

A fim de diminuir a taxa de erro, colocamos mensagens de confirmação nas principais ações da ferramenta. A Figura 19 apresenta um exemplo de mensagem de confirmação. Consideramos que, com a implantação de todos esses requisitos, a satisfação (RQ2) do usuário ao utilizar a ferramenta também será aumentada.

4.3 Lições do Capítulo

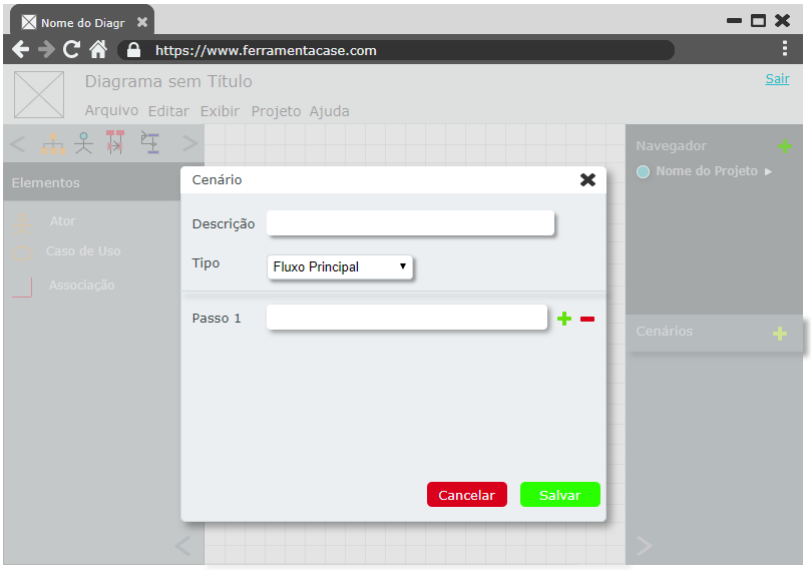
Nesse capítulo, explicamos como foi elaborado todo o projeto de desenvolvimento do protótipo de referência. O qual poderá ser utilizado como referência por várias ferramentas CASE. Mencionamos quais foram os requisitos de usabilidade encontrados na literatura e que foram implementados nas telas. Os requisitos implementados foram a visibilidade (RQ1), a satisfação (RQ2), a baixa taxa de erro (RQ3), a memorabilidade

Figura 17 – Tela de modelagem de requisitos.



Fonte: Elaborado pela autora.

Figura 18 – Tela de criação de cenários.



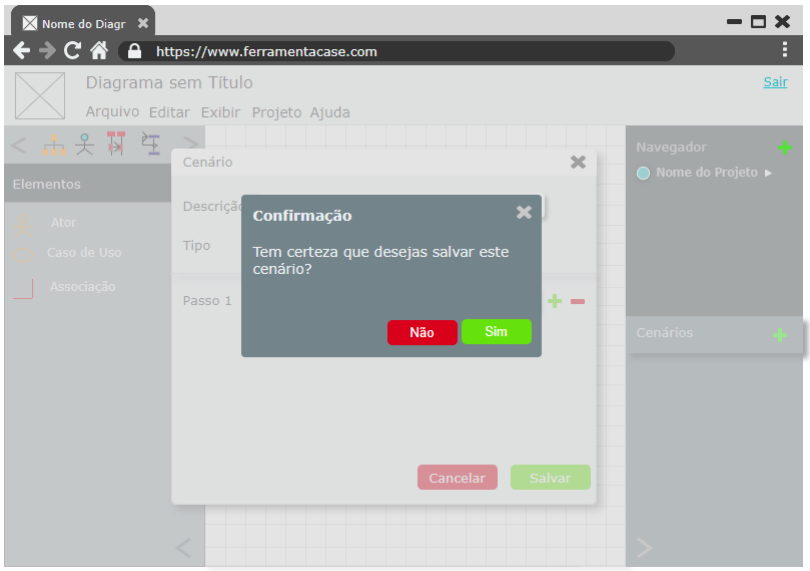
Fonte: Elaborado pela autora.

(RQ4), a aprendizibilidade (RQ5) e a eficiência (RQ6). Todos eles contribuem para uma ferramenta CASE com usabilidade.

Além do mais, exibimos como foi estabelecido o mapa de navegação. Ele foi elaborado baseado na navegação básica de qualquer aplicação do tipo, na qual o usuário pode realizar todas as suas atividades na ferramenta iniciando pelo acesso ao sistema e finalizando com sua saída da ferramenta.

Ainda, expomos as paletas de cores utilizadas para manter todos os protótipos

Figura 19 – Tela de mensagem de confirmação.



Fonte: Elaborado pela autora.

consistentes. Também, apresentamos cada tela com suas respectivas explicações das funcionalidades e de que forma os requisitos de usabilidade foram implementados para cada uma delas. Em algumas telas, conseguimos implantar todos os requisitos, em outras conseguimos aplicar apenas alguns. Porém, o importante é que, de uma maneira geral, todos os requisitos de usabilidade foram atendidos.

5 AVALIAÇÃO DO PROTÓTIPO PROPOSTO

Neste capítulo nós expomos como foi realizado todo o processo de avaliação do protótipo de referência produzido. Nele explicamos todas as etapas necessárias para a validação da solução desenvolvida e os resultados obtidos a partir dessa avaliação. Na Seção 5.1, apresentamos como foi estruturado o planejamento da avaliação do protótipo. Logo, na Seção 5.2, mostramos como essa avaliação foi executada. Na Seção 5.3, exibimos os resultados adquiridos. No final, na Seção 5.4, apresentamos as lições do capítulo.

5.1 Planejamento da Avaliação

Utilizamos a metodologia chamada Goal-Question-Metric (GQM) para definir o escopo da avaliação. Segundo Solingen e Berghout (1999), o GQM é utilizado para definir um plano de medição. No nosso caso, o GQM foi usado para estabelecer o escopo, onde a partir de uma meta de investigação, delineamos as questões de pesquisa e enunciamos como responder tais questões através das medidas. A seguir, dispomos a meta, questão de pesquisa e medida definida para a avaliação.

A **Meta** da nossa avaliação é "Analisar a usabilidade no intuito de avaliar o protótipo no que tange a usabilidade na perspectiva do usuário no contexto de ferramentas CASE".

A **Questão de Pesquisa** que definimos para o nosso trabalho é "Qual é o nível de usabilidade que os protótipos possuem?". O principal intuito da nossa pesquisa é descobrir o quão bem a usabilidade foi implementada no protótipo e se os requisitos de usabilidade realmente contribuíram para o nosso trabalho.

A **Medida** que escolhemos para medir a usabilidade é a *System Usability Scale* (SUS). Aplicamos como medida uma técnica encontrada na literatura e bastante utilizada, que define o grau de usabilidade de sistemas.

O SUS é uma escala *likert*, criada por Brooke et al. (1996), responsável por dar uma visão geral da usabilidade de algum sistema que esteja sendo avaliado. Ele é composto por 10 questões, das quais cada uma possui um peso e deve ser calculado ao final da avaliação. As posições vão de 1 à 5, sendo 1 - descordo fortemente e 5 - concordo fortemente. Segue abaixo as questões do SUS:

1. Eu acho que eu gostaria de utilizar este sistema frequentemente.
2. Eu achei o sistema desnecessariamente complexo.
3. Eu achei o sistema fácil de usar.
4. Eu acho que eu precisaria da ajuda de um técnico para eu conseguir utilizar o sistema.
5. Eu acho que as várias funcionalidades do sistemas estão bem integradas.

6. Eu achei que tinha muito inconsistência no sistema.
7. Eu acho que a maioria das pessoas iriam aprender facilmente a utilizar esse sistema.
8. Eu achei o sistema muito incômodo de usar.
9. Eu me senti muito confiante em utilizar o sistema.
10. Eu precisei aprender muito antes de utilizar esse sistema.

Para obter o resultado das questões é preciso fazer um cálculo. As questões de número ímpar (1,3,5,7 e 9) são calculadas pela posição da escala subtraindo 1. Já as questões pares (2,4,6,8 e 10) são calculadas por 5 menos a posição da escala. De acordo com Sauro (2011), para ter um bom nível de usabilidade, o resultado do SUS deve estar acima de 68 pontos.

Dado que, o SUS foi instrumento de avaliação definido, a próxima etapa foi decidir quais seriam os participantes da avaliação. Decidimos contatar o total de 12 pessoas, das quais 3 eram professores de universidade da engenharia de software, 3 eram desenvolvedores de software, 3 eram alunos de graduação de engenharia de software e 3 eram alunos de mestrado de engenharia de software. Dessa forma, conseguimos atingir um público variado que faz ou já fez o uso de ferramentas CASE.

A fim de atingir todas as funcionalidades desenvolvidas nos protótipos funcionais, delineamos 3 cenários. Os cenários foram distribuídos entre os grupos de participantes. Cada cenário possuía uma sequência de passos com seu respectivo caminho esperado.

5.1.1 Cenários de Avaliação

O cenário 1 tem como objetivo explorar o ambiente de modelagem conceitual. Ele permite que o usuário navegue na ferramenta e consiga inserir elementos, alterar o estereótipo das classes, a visibilidade e defina se a classe é abstrata ou não. Assim como qualquer aplicação, o cenário contém um início, meio e fim. O usuário inicia fazendo *login* na ferramenta, executa as atividades demandadas pelo avaliador e sai da ferramenta. Você pode conferir o cenário 1 no Apêndice B.

Da mesma maneira que o cenário 1, o cenário 2 também explora a modelagem conceitual. Porém, nesse, decidimos fazer com que o usuário experimentasse como é feita a criação dos diagramas, elementos e pacotes de uma forma secundária. Dando a ele mais de uma opção para realizar essas atividades. Além disso, o usuário deveria tomar conhecimento sobre como navegar no projeto. Seguindo o mesmo padrão do cenário anterior, os participantes da avaliação deveriam logar-se na ferramenta, realizar suas atividades e sair da ferramenta. Confira o cenário 2 no Apêndice B.

O cenário 3 diferencia-se dos demais, visto que ele explora a modelagem de requisitos. Nele, estabelecemos atividades relacionadas aos casos de uso. Permitimos que o

usuário criasse cenários, onde fosse possível definir o tipo de cenário, sendo ele fluxo principal, fluxo alternativo e fluxo de exceção com seus respectivos passos. Ainda, instruímos os participantes desse cenário a navegar no projeto e excluir um diagrama. Esse cenário também seguiu a mesma linha de navegação dos demais. Você pode conferir o cenário 3 no Apêndice B.

5.2 Execução da Avaliação

Como pode ser visto na Tabela 3, nós delineamos um cenário para cada perfil de participante, sendo eles professores, alunos de graduação, alunos de mestrado e desenvolvedores. Nós distribuímos os cenários em ordem crescente e de acordo com a ordem alfabética de cada nome dos participantes.

Tabela 3 – Categorização dos cenários.

Perfil	**Participantes**	**Cenários**
Professor	P1	Cenário 1
Professor	P2	Cenário 2
Professor	P3	Cenário 3
Aluno de Graduação	P4	Cenário 1
Aluno de Graduação	P5	Cenário 2
Aluno de Graduação	P6	Cenário 3
Aluno de Mestrado	P7	Cenário 1
Aluno de Mestrado	P8	Cenário 2
Aluno de Mestrado	P9	Cenário 3
Desenvolvedor	P10	Cenário 1
Desenvolvedor	P11	Cenário 2
Desenvolvedor	P12	Cenário 3

Fonte: Elaborado pela autora.

Elaboramos um termo de aceite, relatando sobre o que se tratava a avaliação e qual era o tempo máximo de duração da avaliação. Também, estabelecemos um formulário referente ao perfil de cada tipo de participante. A avaliação foi realizada individualmente, com o uso do computador do avaliador.

O tempo médio levado para responder o termo de aceitação, o perfil do usuário a execução das atividades e o preenchimento do SUS, foi em média de 6 minutos.

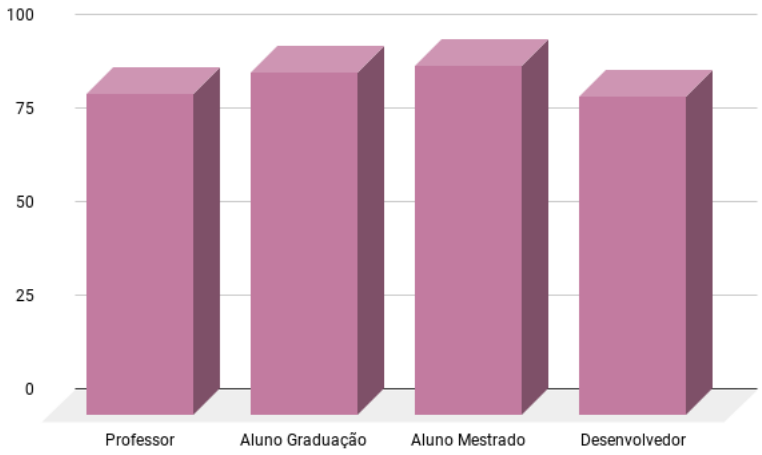
O período de execução da avaliação foi de 09/10/2019 a 17/10/2019.

5.3 Resultados da Avaliação

Conforme mencionado antes, para obter-se um bom nível de usabilidade, é considerado que o resultado da avaliação tenha um valor superior à 68 pontos. Com isso, como pode ser visto na Figura 20, tivemos um bom resultado. A média do grupo de professores de engenharia de software foi de 85,75 pontos. Já o resultado do grupo de alunos de graduação de engenharia de software, foi de 91,5 pontos. O grupo de alunos de

mestrado de engenharia de software, foi de 93,25 pontos. Por fim, o grupo de desenvolvedores classificou o SUS com 85 pontos. Dessa forma, segundo Brooke et al. (1996) e Sauro (2011), podemos afirmar que os protótipos construídos obtiveram um ótimo resultado, já que apresentam um bom nível de usabilidade. A Tabela 4, exibe o resultado detalhado da avaliação.

Figura 20 – Resultado da Avaliação.



Fonte: Elaborado pela autora.

Tabela 4 – Resultado detalhado.

Perfil	**Participante**	**Pontuação**	**Média por Perfil**	**Resultado Final**
	P1	33		
Professor	P2	35	34,3	85,75
	P3	33		
	P4	40		
Aluno Graduação	P5	36	36,6	91,5
	P6	34		
	P7	39		
Aluno Mestrado	P8	37	37,3	93,25
	P9	36		
	P10	37		
Desenvolvedor	P11	31	34	85
	P12	34		

Fonte: Elaborado pela autora.

Aplicamos um questionário para cada tipo de participantes, a fim de capturar os dados referente aos perfis. No caso dos professores, gostaríamos de saber há quanto tempo eles trabalhavam como professores de engenharia de software, se haviam trabalhado na indústria e por quanto tempo. Com isso, obtivemos o resultado que 2 dos participantes trabalharam na indústria por tempo bem variado (confira a Tabela 5).

No caso dos alunos de graduação, queríamos saber se eles já haviam participados de projetos, no curso, que fizessem o uso de ferramentas CASE de modelagem. Também,

Tabela 5 – Perfil dos professores.

Participante	**Tempo de Professor**	**Trabalhou na Indústria**	**Tempo de Indústria**
P1	3 anos	Sim	8 anos
P2	10 anos	Não	
P3	6 meses	Sim	6 meses

Fonte: Elaborado pela autora.

queríamos saber se eles já haviam trabalhado na indústria. Dois alunos de graduação participaram de projetos que envolviam o uso de ferramentas CASE, porém nenhum deles trabalhou na indústria (confira a Tabela 6).

Tabela 6 – Perfil dos alunos de graduação.

Participante	**Participação em Projeto**	**Trabalhou na Indústria**	**Tempo de Indústria**
P4	Sim	Não	
P5	Sim	Não	
P6	Não	Não	

Fonte: Elaborado pela autora.

Já para os alunos de mestrado, elaboramos um questionário perguntando quando eles haviam finalizado a graduação, quando haviam ingressado no mestrado, se haviam trabalhado na indústria e por quanto tempo. Todos ingressaram no mestrado em 2019 e apenas 1 aluno de mestrado trabalhou na indústria (confira a Tabela 7).

Tabela 7 – Perfil dos alunos de mestrado.

Participante	**Graduação**	**Mestrado**	**Trabalhou na Indústria**	**Tempo de Indústria**
P7	2016	2019	Não	
P8	2019	2019	Sim	2 anos
P9	2018	2019	Não	

Fonte: Elaborado pela autora.

Por fim, sobre os desenvolvedores nós queríamos saber se eles já haviam se graduado, em qual curso e por quanto tempo trabalhavam na indústria. Apenas 1 dos desenvolvedores era formado e todos trabalhavam, em média, há 2 anos na indústria (confira a Tabela 8).

Notamos que alguns participantes tiveram dificuldade ao encontrar alguns itens da tela. Um dos professores sugeriu mudar a cor dos botões, deixando-os com cores mais saturadas, melhorando a encontrabilidade dos mesmos. Outro ponto que analisamos foi a forma costumeira dos usuários de ferramentas CASE de arrastar os elementos de modelagem na tela, sendo que nas telas isso não foi possível. Porém, podemos considerar isso para trabalhos futuros.

Tabela 8 – Perfil dos desenvolvedores.

Participante	É Graduado	Curso	Tempo de Indústria
P10	Não		2 anos
P11	Não		2 anos
P12	Sim	Engenharia Ambiental	1,5 ano

Fonte: Elaborado pela autora.

5.4 Lições do Capítulo

Nesse capítulo nós apresentamos todo o processo de avaliação dos protótipos desenvolvidos ao longo do trabalho. Explicamos como o planejamento da avaliação foi constituído. Quer dizer, através da abordagem GQM, na qual foram definidas a meta, a questão de pesquisa e a medida necessária para a avaliação.

Ainda, nós explanamos o porquê de utilizarmos a técnica SUS, responsável por medir o nível de usabilidade de uma ferramenta. Também, explicamos como foi feita a preparação da avaliação.

Da mesma forma, nós demonstramos a execução da avaliação, apresentando os tipos de participantes, com seus respectivos cenários. Também, expusemos o tempo médio levado para executar a avaliação e o período.

Especificamos o resultado obtido através de um gráfico. O qual exibe a pontuação da escala SUS para os professores de engenharia de software, os alunos de graduação e de mestrado da engenharia de software e os desenvolvedores de software. Tivemos um bom resultado de acordo com Sauro (2011).

6 CONSIDERAÇÕES FINAIS

O objetivo do nosso trabalho é desenvolver um protótipo de referência para ferramentas CASE de modelagem em ambiente web. O qual servirá como referência para qualquer tipo de ferramenta CASE desse domínio. Para cumprir com o objetivo do nosso trabalho, realizamos uma revisão sistemática a fim de descobrir quais são as características necessárias que uma ferramenta CASE com usabilidade deveria comportar em termos de usabilidade. Após ter descoberto quais são os requisitos de usabilidade mencionados na literatura, iniciamos o desenvolvimento do nosso protótipo de referência. Por fim, realizamos uma avaliação do protótipo abrangendo um público variado que faz o uso desse tipo de ferramenta. Convocamos professores de engenharia de software, alunos de graduação de engenharia de software, alunos de mestrado de engenharia de software e desenvolvedores de software para participar. Usamos uma técnica, denominada SUS, que mede o nível de usabilidade dos sistemas. Os estudos dizem que o resultado superior a 68 pontos garante um bom nível de usabilidade. Tivemos um resultado, contando com os 4 tipos de participantes, na média de 88,37 pontos. Com isso, consideramos ter um protótipo de referência com um bom nível de usabilidade.

A principal contribuição do nosso trabalho é a criação de um protótipo de referência que pode ser utilizado por qualquer ferramenta CASE de modelagem. Portanto tivemos alguns desafios para chegar ao objetivo final. Na literatura, por exemplo, foi difícil encontrar trabalhos que tivessem relação com o nosso tema. Percebemos que essa área não é muito abordada, apesar de sua importância. Tendo isso em vista, realizamos uma revisão sistemática a fim de identificar quais eram os requisitos de usabilidade necessários para uma ferramenta CASE. Outro desafio foi criar o primeiro protótipo. Visto que, era preciso saber por onde começar e como começar. Os artigos resultantes da revisão sistemática relatavam sobre usabilidade, porém não comentavam como iniciar um protótipo com usabilidade. Para superar isso, tomamos, como base o trabalho elaborado por (LIMA, 2017), que tem relação com o tema, a fim de investigar as oportunidades de melhoria para construir uma versão melhorada e, a partir disso, continuar o processo de construção do protótipo.

6.0.1 Trabalhos Futuros

Como trabalhos futuros podemos apontar a exploração de outros tipos de diagramas, que sejam diferentes da linguagem UML, a fim de validar se os requisitos levantados nesse trabalho também se aplicariam à outras notações de modelagem. Construindo, então, mais provas de conceito.

Ademais, podemos considerar a experiência de outras populações. Convocar mais desenvolvedores de software e outros tipos de usuários de ferramentas CASE para participar da avaliação do protótipo de referência.

Enfim, uma outra possibilidade de trabalho futuro será o desenvolvimento de uma

ferramenta CASE real, utilizando o protótipo de referência produzido.

REFERÊNCIAS

AKIKI, P. A.; BANDARA, A. K.; YU, Y. Cedar studio: an ide supporting adaptive model-driven user interfaces for enterprise applications. In: ACM. **Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems**. London, 2013. p. 139–144. Citado 3 vezes nas páginas 37, 39 e 41.

AKIKI, P. A.; BANDARA, A. K.; YU, Y. Engineering adaptive model-driven user interfaces. **IEEE Transactions on Software Engineering**, IEEE, v. 42, n. 12, p. 1118–1147, 2016. Citado 3 vezes nas páginas 37, 39 e 41.

ALBERT, W.; TULLIS, T. **Measuring the user experience: collecting, analyzing, and presenting usability metrics**. Newnes: Science Direct, 2013. Citado na página 21.

AMBLER, S. W. **The object primer: Agile model-driven development with UML 2.0**. New York: Cambridge University Press, 2004. Citado 2 vezes nas páginas 29 e 30.

AUGUSTO, H. **Quando utilizar uma IHM de baixo custo em uma automação?** 2019. Disponível em: <https://www.hitecnologia.com.br/blog/quando-utilizar-uma-ihm-de-baixo-custo-em-uma-automacao/>. Acesso em: 18 nov 2019. Citado na página 25.

AXURE. **Welcome to the New Axure**. 2019. Disponível em: <https://www.axure.com/blog/welcome-to-the-new-axure>. Acesso em: 18 nov 2019. Citado na página 31.

BAĆÍKOVÁ, M.; MARIĆÁK, M.; VANĆÍK, M. Usability of a domain-specific language for a gesture-driven ide. In: **2015 Federated Conference on Computer Science and Information Systems (FedCSIS)**. Lodz: IEEE, 2015. p. 909–914. Citado 3 vezes nas páginas 38, 39 e 40.

BARBOSA, S.; SILVA, B. **Interação humano-computador**. Brasil: Elsevier Brasil, 2010. Citado 6 vezes nas páginas 21, 22, 25, 26, 27 e 29.

BIEGEL, B. et al. U can touch this: touchifying an ide. In: **Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering**. Hyderabad: ACM, 2014. p. 8–15. Citado 4 vezes nas páginas 37, 39, 40 e 41.

BROOKE, J. et al. Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996. Citado 2 vezes nas páginas 53 e 56.

CASE, A. F. Computer-aided software engineering (case): technology for improving software development productivity. **ACM SIGMIS Database: the DATABASE for Advances in Information Systems**, ACM, v. 17, n. 1, p. 35–43, 1985. Citado na página 29.

DENNING, P. J. What is software quality. **Communications of the ACM**, ACM, v. 35, n. 1, p. 13–15, 1992. Citado na página 27.

DILLON, B.; THOMPSON, R. Software development and tool usability. In: IEEE. **2016 IEEE 24th International Conference on Program Comprehension (ICPC)**. Austin, 2016. p. 1–4. Citado na página 21.

FREEDMAN, D. H. Programming without tears. **High technology**, v. 6, n. 4, p. 38–43, 45, 1986. Cited By 7. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0022699141&partnerID=40&md5=34965cd4a7fb920812ae881e56a89928>. Acesso em: 18 nov 2019. Citado na página 29.

GLÜCKER, H. et al. Eyede: gaze-enhanced software development environments. In: ACM. **Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems**. Toronto, 2014. p. 1555–1560. Citado 4 vezes nas páginas 37, 39, 40 e 41.

HIX, D.; HARTSON, H. R. **Developing user interfaces: ensuring usability through product & process**. New York: John Wiley & Sons, Inc., 1993. Citado na página 25.

HOU, D.; WANG, Y. An empirical analysis of the evolution of user-visible features in an integrated development environment. In: IBM CORP. **Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research**. Riverton, 2009. p. 122–135. Citado 5 vezes nas páginas 21, 38, 39, 41 e 42.

ISO/IEC 14102. **Information technology – Guideline for the evaluation and selection of CASE tools**. 2008. Disponível em: <https://www.iso.org/standard/43189.html>. Acesso em: 18 nov 2019. Citado na página 29.

ISO/IEC 9126-1. **Gestão da qualidade e garantia da qualidade - Terminologia**. 2001. Disponível em: <Softwareengineering--Productquality--Part1:Qualitymodel>. Acesso em: 18 nov 2019. Citado na página 27.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Citado na página 33.

KNOR, F. C.; AVELAR, N. D. **Proposta de melhoria da usabilidade do sistema de gestão de estágios da Universidade Tecnológica Federal do Paraná**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2015. Citado na página 43.

LEUNG, A. et al. Developing an adaptive user interface in eclipse. In: **Proc. the Eclipse Technology Exchange Workshop at European Conference on Object Oriented Programming**. [S.l.: s.n.], 2006. Citado 5 vezes nas páginas 37, 38, 39, 41 e 42.

LIMA, M. A. d. O. Um projeto de interface para ferramentas de modelagem uml baseadas em eclipse. Universidade Federal do Pampa, 2017. Citado 4 vezes nas páginas 33, 34, 36 e 59.

LUCKOW, D. H.; MELO, A. A. de. **Programação Java para a WEB**. Brasil: Novatec Editora, 2010. Citado na página 22.

MAYHEW, D. J.; MAYHEW, D. **The usability engineering lifecycle: a practitioner's handbook for user interface design**. San Francisco: Morgan Kaufmann, 1999. Citado na página 28.

MORAES, R. Análise de conteúdo. **Revista Educação, Porto Alegre**, v. 22, n. 37, p. 7–32, 1999. Citado na página 35.

NBR ISO 8402. **Gestão da qualidade e garantia da qualidade - Terminologia**. 1994. Disponível em: <https://www.abntcatalogo.com.br/norma.aspx?ID=58542>. Acesso em: 18 nov 2019. Citado na página 27.

NIELSEN, J. **Usability engineering**. UK: Elsevier, 1994. Citado 3 vezes nas páginas 26, 27 e 31.

NIELSEN, J. 10 usability heuristics for user interface design. **Nielsen Norman Group**, v. 1, n. 1, 1995. Citado 2 vezes nas páginas 27 e 39.

NORMAN, D. **The design of everyday things: Revised and expanded edition**. New York: Basic books, 2013. Citado na página 27.

PEREIRA, R. **Interação Humano-Computador**. 2019. Http://www.sbc.org.br/14-comissoes/390-interacao-humano-computador. March 07, 2019. Disponível em: <http://www.sbc.org.br/14-comissoes/390-interacao-humano-computador>. Acesso em: 18 nov 2019. Citado na página 21.

PRATES, R. O.; BARBOSA, S. D. J. Avaliação de interfaces de usuário–conceitos e métodos. In: **Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo**. Brasil: SBC, 2003. v. 6, p. 28. Citado na página 26.

PREECE, J.; ROGERS, Y.; SHARP, H. **Interaction design: beyond human-computer interaction**. Nova Jersey: John Wiley & Sons, 2015. Citado na página 25.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. Brasil: McGraw Hill Brasil, 2016. Citado na página 21.

REBELO, I. **7 Princípios de projeto de interação**. 2019. Jun 05, 2019. Disponível em: <https://irlabr.wordpress.com/apostila-de-ihc/parte-1-ihc-na-pratica/7-principios-de-projeto-de-interacao/>. Acesso em: 18 nov 2019. Citado na página 43.

SAURO. **MEASURING USABILITY WITH THE SYSTEM USABILITY SCALE (SUS)**. 2011. Disponível em: <https://measuringu.com/sus/>. Acesso em: 18 nov 2019. Citado 3 vezes nas páginas 54, 56 e 58.

SIAU, K.; CAO, Q. Unified modeling language: A complexity analysis. **Journal of Database Management (JDM)**, IGI Global, v. 12, n. 1, p. 26–34, 2001. Citado na página 44.

SOLINGEN, D. R. van; BERGHOUT, E. W. **The Goal/Question/Metric Method: a practical guide for quality improvement of software development**. [S.l.]: McGraw-Hill, 1999. Citado na página 53.

STAMPS, D. Case: Cranking out productivity. **Datamation**, v. 33, n. 13, p. 55–60, 1987. Citado na página 29.

STOREY, M.-A. et al. Improving the usability of eclipse for novice programmers. In: ACM. **Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange**. Anaheim, 2003. p. 35–39. Citado 4 vezes nas páginas 38, 39, 41 e 42.

WEINRICH, J. Software de apoio à avaliação e seleção de ferramentas case baseado na norma iso/iec 14102. **Trabalho de Conclusão de Curso, Universidade Regional de Blumenau. Blumenau**, 1999. Citado 2 vezes nas páginas 22 e 30.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the 18th international conference on evaluation and assessment in software engineering**. New York: Citeseer, 2014. p. 38. Citado na página 34.

ZOU, Y. et al. Adapting the user interface of integrated development environments (ides) for novice users. **Journal of Object Technology**, v. 7, n. 7, p. 55–74, 2008. Citado 5 vezes nas páginas 22, 38, 39, 41 e 43.

Apêndices

APÊNDICE A – DADOS DA ANÁLISE DE CONTEÚDO

Análise de Conteúdo

A análise de conteúdo constitui uma metodologia de pesquisa usada para descrever e interpretar o conteúdo de toda classe de documentos e textos. Essa análise, conduzindo a descrições sistemáticas, qualitativas ou quantitativas, ajuda a reinterpretar as mensagens e a atingir uma compreensão de seus significados num nível que vai além de uma leitura comum.

É constituída de cinco etapas:

1 - Preparação das informações;

2 - Unitarização ou transformação do conteúdo em unidades;

3 - Categorização ou classificação das unidades em categorias;

4 - Descrição;

5 - Interpretação.

Análise de Conteúdo - Unitarização

Artigo 1

ID: P1

Título: Developing an adaptive user interface in eclipse

Autor: Leung, Alex
Morisson, Scott
Wringe, Matt
Zou, Ying

Data da Publicação: 2006

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P1.1 (1) learnability, which measures the ease of learning the application functionality;
P1.2 (2) efficiency, which measures the ease of use and the level of productivity attainable by the user;
P1.3 (3) memorability, which measures the ease of remembering the application's functionality;
P1.4 (4) low error rate, which measures how the application support users in making less errors; and
P1.5 (5) user's satisfaction, which measures how the users enjoy the application. In our research, we aim to improve all five usability qualities through the use of an adaptive user interface.

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

To better understand how to increase usability of the menu system, we establish quantitative usability measurements, and construct a simplified user model that simulates how a user interacts with the menu system and maintains the knowledge of the user's usage patterns. Therefore, optimal modifications to the menu system can be made to improve usability.

P1.6 One of the metrics that can be determined from the user model is the **average time** to element that is the time takes a user to find any random element within the menu.

P1.6.1 By minimizing the average time to element, we can maximize the efficiency of the menu system.

P1.7 Moving elements that have the highest probability of being clicked next to the top of the menu improves learnability and efficiency.

P1.8 After sufficient user usage pattern data has been collected, the plug-in will hide menu elements that the user infrequently menus.

P1.9 Arquitetura:
• The AMS is the main form of interaction the user has with the menu system.
• The AUI Wizard provides guidance and notification to the user when the user interface is started for the very first time after changes are made to the existing user interface.
• The AUI Preferences are preferences set by the user about how the AUI should act. These preferences will be accessed the same way as all other preferences are accessed in the Eclipse IDE.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P1.10 the user interface of Integrated Development Environments (IDEs) is often neglected, leaving the UI complex and difficult to use. IDEs often have a wide variety of functions for different development projects and they are designed to cater to a wide range of uses. Therefore, all functionality is made available at all times to a user, increasing UI clutter.

Análise de Conteúdo - Unitarização

Artigo 2

ID: P2

Título: EyeDE: Gaze-enhanced Software Development Environments

Autor:
Glücker, Hartmut
Raab, Felix
Echtler, Florian
Wolff, Christian

Data da Publicação: 2014

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P2.1 programmers suffering from repetitive strain injury (RSI) may benefit from multimodal development environment setups and possibly even retain their ability to produce code – which is another reason why research should also consider **accessibility** improvements in the context of software development.

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P2.2 We want to simplify reading and navigating source code by appropriately reacting to gaze. We propose a design that enables quick and natural navigation through gaze-controlled menus in IDEs. The interaction is designed to support hands-free navigation, not only enabling handicapped developers to browse the source code, but also preventing able- bodied developers from repeatedly moving their hands between the keyboard and the mouse for certain tasks

P2.3 In a visual environment controlled by gazes, it is important to avoid distractions as they may cause users to focus on them either voluntarily or accidentally.

P2.4 Besides the display of source code, the EyeDE UI therefore only utilizes light gray tints with low contrast (see Figure 7), with the only highlighting color being a light blue for displaying the status of certain actions.

P2.5 We regard the cautious use of colors as important so that our animated contextual menus do not draw too much attention to themselves while still being noticeable.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P2.6 Integrated development environments (IDEs) offer a vast number of tools for developer assistance. However, IDEs often lack ease of use with regard to how these tools can be found or activated – which may be part of the reason why the Eclipse developer team has recently focused on usability-related enhancements for their development environment [3]

Análise de Conteúdo - Unitarização

Artigo 3

ID: P3

Título: An empirical analysis of the evolution of user-visible features in an integrated development environme

Autor:
Hou, Daqing
Wang, Yuejiao

Data da Publicação: 2009

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P3.1 broader applicability
P3.2 visibility
P3.3 info integration
P3.4 user control
P3.5 action integration
P3.6 efficiency
P3.7 persistent abstraction
P3.8 learnability

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P3.9 Broader applicability refers to work that applies the same feature to new data or create a similar feature in a new context.

P3.10 visibility concerns mainly local, non-structural information (e.g., showing the availability of a quick assist for a particular line of code in the editor)

P3.11 info integration is used to refer to features that present structural information or automate tasks that involve structural information (e.g., showing the Javadoc when hovering the mouse over a class name).

P3.12 Features in the **user control** category give programmers more control over the ways they interact with the IDE. Examples include allowing programmers to change view layouts; organizing information by means of sorting, filtering, and grouping; and customizing and configuring the IDE by setting preferences. This category also includes features that encourage exploration and reduce a user's memory load, for example, redo/undo, bookmarks, the "editor navigation history", and the "recently used file list."

P3.13 Action integration refers to a special subset of efficiency improving features that integrate actions into the active context, making them available where the programmer is working. For example, this can be achieved by introducing context menus, using popup windows to display information in the editor rather than in separate views, or allowing for conducting certain tasks at the best possible moment and place, like adding a project into working sets directly in the project wizard rather than forcing the programmers to wait until the project is created.

P3.14 Efficiency can be achieved by using both general UI methods, like the use of shortcuts, and IDE specific methods, like faster stepping (debug) and block commenting. A particularly interesting family of features is automating multiple actions in one single step such as fixing multiple problems, formatting multiple files, or importing multiple projects, all at once.

P3.15 Persistent abstractions are features that allow programmers to save processes or data that may take time to create so that they can be reused later or in other contexts. For example, over time, Eclipse added support for persisting launch configurations, execution environments, working sets, and

P3.16 There are also 10 entries that are aimed to improve **learnability** of Eclipse. For example, the welcome page added in 3.0 provides links to tutorials, samplers, and summaries of new features. Other examples include Cheat Sheets, online user and development manuals, and contextual help.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P3.17 As much as possible, a well-engineered IDE should help programmers automate routine tasks and reduce cognitive load so that they can concen- trate on the creative aspects of the software engineering process. As a result, like any useful software, Java IDEs are increasingly becoming more powerful and usable as new features are added and usability concerns are addressed.

Análise de Conteúdo - Unitarização

Artigo 4

ID: P4

Título: Engineering adaptive model-driven user interfaces

Autor:
Akiki, Pierre A
Bandara, Arosha K
Yu, Yijun

Data da Publicação: 2016

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P4.1 A study that we conducted showed that UIs adapted with RBUIS improve usability by increasing the perceived usability of end-users and helping them in fulfilling their daily tasks more **efficiently**

P4.2 and effectively.

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P4.3 Adaptive UIs are aware of their context-of-use and are capable of providing a response to changes in this context, by adapting one or more of their characteristics using rules that are either predefined or deduced by learning over time.

P4.4 Role-Based UI Simplification (RBUIS), a mechanism for improving usability by providing end- users with a minimal feature-set and an optimal layout based on the context-of-use.

three novel technical contributions: the Cedar Architecture, the Role-Based UI Simplification (RBUIS) mechanism, and their supporting IDE, Cedar Studio

P4.5 The **Cedar Architecture** has three server-side technology-independent layers including: decision components, adaptation components,and adaptive behavior and UI models. It also has a technology-specific client components layer. This layer's components are part of an API, which integrates in a software application's code and allows it to connect to the server-side layers in order to adapt its user interfaces. The Cedar Architecture adopts interpreted run- time models (Section 2.1) as a modeling approach,in order to support more advanced runtime adaptations.

P4.6 RBUIS supports feature-set minimization by assigning roles to task models for providing end-users with a minimal feature-set based on the context- of-use.

P4.6.1 Layout optimization is supported by assigning roles to workflows that represent adaptive UI behavior visually and through code before being applied to CUI models.

P4.7 A brief overview of **Cedar Studio** was presented in Section
5. This IDE supports the development of adaptive model- driven UIs based on the Cedar Architecture and using RBUIS. Cedar Studio provides stakeholders such as developers and IT personnel, with visual-design and code-editing tools for defining and managing artifacts such as UI models and adaptive behavior.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P4.8 Enterprise applications, e.g., enterprise resource planning (ERP) systems, are a common example of such types of software applications. Hence, it is not surprising that these applications suffer from many usability problems [1], some of which are caused by their complex off-the-shelf UIs.

Análise de Conteúdo - Unitarização

Artigo 5

ID: P5

Título: U can touch this: touchifying an ide

Autor:
Biegel, Benjamin
Hoffmann, Julien
Lipinski, Artur
Diehl, Stephan

Data da Publicação: 2014

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P5.1 Furthermore, most navigation tasks were performed by direct manipulations because they felt more **intuitive**

P5.2 and **comfortable.**

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P5.3 (multi-)touch gestures offer a direct user interaction without losing the focus on the screen

P5.4 Keep the bento box design: The original user interface should be modified as little as possible. In most IDEs the user interface is split into several views. The challenge is to keep this so called bento box design [3] unaffected while adding new direct manipulation interactions.

P5.5 Maintain existing user interactions: The developer should still have access to all the functions of the IDE, including the familiar keyboard and mouse interactions. That is why the touch gestures have to be applied as an additional layer without disturbing the existing interac- tion opportunities.

P5.6 Meet developer's expectations: By using specific touch gestures, it is very likely that developers will have a clear expectation of the behavior of the user interface. Hence, the challenge is not only to find new intuitive gestures but also to adapt the user interface as expected.

P5.7 Map multiple refactorings to a single gesture: The developer should be able to create a natural association from the desired refactorings to the required gestures. Thus, similar refactorings have to be mapped to a single gesture, and further, the selection of the desired refactor- ing depends on the particular context.

P5.8 Use common touch gestures: To provide a quick and easy start, only common touch gestures should be used that are mainly known from smartphones or tablets. We assume that most developers are familiar with these ges- tures.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P5.9 Modern IDEs are feature-rich, making it more and more difficult for the developer to navigate through the software project, to reorder views, or to use functionality that is hidden in complex menu structures or shortcuts.

Análise de Conteúdo - Unitarização

Artigo 6

ID: P6

Título: Usability of a Domain-Specific Language for a Gesture-Driven IDE

Autor:
Bačíková, Michaela
Maričák, Martin
Vančík, Matej

Data da Publicação: 2015

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

The focus is mainly on usability [6], [7], [8], domain usability and **specificity** (específico, particular)[9]

P6.1 simplicity of use

P6.2 learnability

P6.3 compliance- conformidade with standard design patterns and guidelines.

P6.4 When compared to a physical keyboard, we hypothesize that touch gestures are more **learnable** and **memorable** than key shortcuts.

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

***** Touch interfaces are natural and easy to get used to and they also affect the way people try to control other devices such as common PCs or notebooks. Therefore, hybrid devices1 have emerged and operation systems try to adapt their UIs and functionality to touch.

P6.5 Our goal is not to force the work exclusively via touch gestures but to create a symbiosis of keyboard, touch and, alternatively, mouse.

P6.6 1) General-purpose gestures (GPGs) - used for standard interaction. They can be used in different context, however their meaning is always the same or at least very similar. The most common GPGs are touch, double-touch, swipe or pinch. This category can be considered platform- independent although according to Wroblewski [13], not all gestures are applied on all platforms consistently to the same functionalities.

P6.7 2) Drawn gestures - a specific type of gestures that have no standard or specification indicating their form, purpose or semantics. For example, drawing a circle can be used for restoration, rotation or selecting multiple items in a list. The design of semantics of drawn gestures is often intuitive and closely linked to the context of their use.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P6.8 Not only the current IDEs are not ready for advanced work on touch displays, they are not even prepared for everyday touch use stereotypes and are largely limited to interaction via traditional hardware devices (e.g., the ubiquitous keyboard and mouse) [3].

P6.9 From the IDEs we have analyzed, the only IDE ready touch is Eclipse, statistically the most used IDE for Java language. However, it still offers just fundamental features. As for the second in line, IntelliJ IDEA, despite marked innovative and usable, does not even support the most fundamental ones

Análise de Conteúdo - Unitarização

Artigo 7

ID: P7

Título: Improving the usability of Eclipse for novice programmers

Autor:
Storey, Margaret-Anne
Damian, Daniela
Michaud, Jeff
Myers, Del
Mindel, Marcellus
German, Daniel
Sanseverino, Mary
Hargreaves, Elizabeth

Data da Publicação: 2004

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P7.1 Don Norman lists four key user interface principles of **feedback,**

P7.2 visibility

P7.3 mappings

P7.4 affordance

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P7.5 - incluir quadro ao lado we have designed a set of plug-ins (collectively called Gild) that provides two perspectives – one for instructors and one for students. The in- structor perspective provides support for managing a programming course, and creating course content that is closely integrated with code examples. The student perspective presents a simplified develop- ment environment that has additional features to help them easily navigate between course material and program examples.

P7.6 Task view: changed defalut behaviour to show only errors for the current project

P7.7 Resource View: decorators on resources to indicate file state and distinction between .class and .java files

P7.8 New features to support learning needs - HTML browser with new features to link to and dynamically include source code

P7.9 Launch: one-click

P7.10 Debugger: single thread, step over

P7.11 Project import/export simplified

P7.12 Line numbers: on by default

P7.13 Support different teaching and leaning styles: Configurable settinns for Gild: Use of debugger, debugginf filters, CVS, automatic build

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P7.14 Admittedly, Eclipse has many powerful features that can assist both expert and novice pro-grammers and Java teachers. However, we believe that Eclipse can be improved to offer enhanced support to novice programmers as well as instruc- tors. Firstly, the Eclipse user interface may be overly complex and that this complexity could pose a barrier to learning.
Secondly, improving Java instruction through Eclipse requires offering support for features that are commonly available in web-based learning tools.

Análise de Conteúdo - Unitarização

Artigo 8

ID: P8

Título: Integrated Development Environments (IDEs) for Novice Users

Autor:
Zou, Ying
Lerner, Michael
Leung, Alex
Morisson, Scott
Wringe, Matt

Data da Publicação: 2008

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P8.1 Our adaptive algorithms determine the optimal changes to the menu system in order to improve the user's **efficiency** when searching for the next element to click in a menu.

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

P8.2 In this paper, we propose an Adaptive User Interface (AUI) architecture. The AUI dynamically adapts the UI to the daily routines, such as coding, compiling, and debugging, of individual novice users.

The Architecture for Our Adaptive User Interface Plug-in

P8.3 The AMS is the main form of interaction which the user has with the menu system.

P8.4 The AUI wizard provides guidance and notification to the user when the UI is started for the very first time after changes are made to the existing UI.

P8.5 The AUI preferences are preferences set by the user about how the AUI should act. These preferences will be accessed in the same fashion as other Eclipse preferences. In our work, we set the AUI preference with respect to the novice users.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P8.6 which make it easy for users to access the most frequently used menu elements. However with these advancements, the UI of Integrated Development Environments (IDEs) is often neglected; leaving the UI complex and difficult to learn. In particular, IDEs are designed to cater to a wide range of users by offering a variety of functions. Such functionality is made available at all times to all users, increasing UI clutter.

Análise de Conteúdo - Unitarização

Artigo 9

ID: P9

Título: Cedar studio: an IDE supporting adaptive model-driven user interfaces for enterprise applications

Autor:
Akiki, Pierre A
Bandara, Arosha K
Yu, Yijun

Data da Publicação: 2013

Trecho do texto que apresenta os requisitos de usabilidade definidos para ferramentas CASE:

P9.1 simplification

Trecho do texto que apresenta como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE.

* Cedar Studio supports model-driven UI development, based on the CEDAR architecture, through a set of visual design and code editing tools that can be used by both developers and I.T. personnel. Additionally, Cedar Studio supports integrated testing of the devised adaptive behavior by running the developed UI from within the IDE itself.

P9.2 (1) Task Models - (1) Visual Adaptive Behavior Workflows

P9.3 (2) Domain Models - (2) Dynamic Scripts for optimizing a UI's layout

P9.4 (3) Abstract UI (AUI) Models - (3) Visual Role Assignments

P9.5 (4) Concrete UI (CUI) Models - (4) Code-Based Rules for minimizing a UI's feature-set to a particular context

P9.6 (5) Goal Models - (5) SQL-based Model Constraints for verifying manually created models.

Trecho do texto que apresenta qual a motivação para melhorar a usabilidade de ferramentas CASE.

P9.7 Furthermore, an IDE style UI could provide the necessary ease-of-use for managing the complex user interface and adaptive behavior artifacts of large-scale enterprise applications.

Análise de Conteúdo - Categorização e Descrição						
Requisitos de usabilidade definidos para ferramentas CASE						
Learnability	Efficiency	Memorability	Low error rate	Satisfaction	Acessibility	Visibility
P1.1	P1.2	P1.3	P1.4	P1.5	P2.1	P3.2
P3.8	P3.6	P3.4	P3.1	P5.2		P7.2
P6.3	P4.1	P6.4				P3.3
P6.2	P8.1					P7.1
P9.1	P3.5					
P5.1	P4.2					
P7.4	P3.7					
P7.3						

Análise de Conteúdo - Categorização e Descrição						
Como a engenharia de usabilidade é usada no desenvolvimento de ferramentas CASE						
Otimização de tempo / eficiência	Melhorar aprendizabilidade	Adaptavidade	Acessibilidade	Minimização de erros	Melhorar a Visibilidade	Simplicidade de Interface de Usuáro
P1.6	P1.7	P1.8	(P2.2)	P3.9	P3.10	P4.4
P1.6.1	P3.16	P1.9	P2.3		P3.11	P7.6
P3.13	P7.8	P3.12	P2.4		P7.7	P7.9
P3.14		P4.5	P2.5			P7.10
P3.15		P4.6	P5.3			P7.11
P5.8		P4.7	P5.4			P7.12
P2.2		P7.13	P5.5			P9.5
		P8.3	P5.6			P9.3
		P8.4	P5.7			
		P8.5	(P5.8)			
		P9.2	P6.6			
			P6.7			

Análise de Conteúdo - Categorização e Descrição		
Qual a motivação para melhorar a usabilidade de ferramentas CASE		
Complexidade e Dificuldade de uso	Automatização tarefas de rotina e redução a carga cognitiva	Falta de acessibilidade
P1.10	P3.17	P6.8
P2.6		P6.9
P4.8		

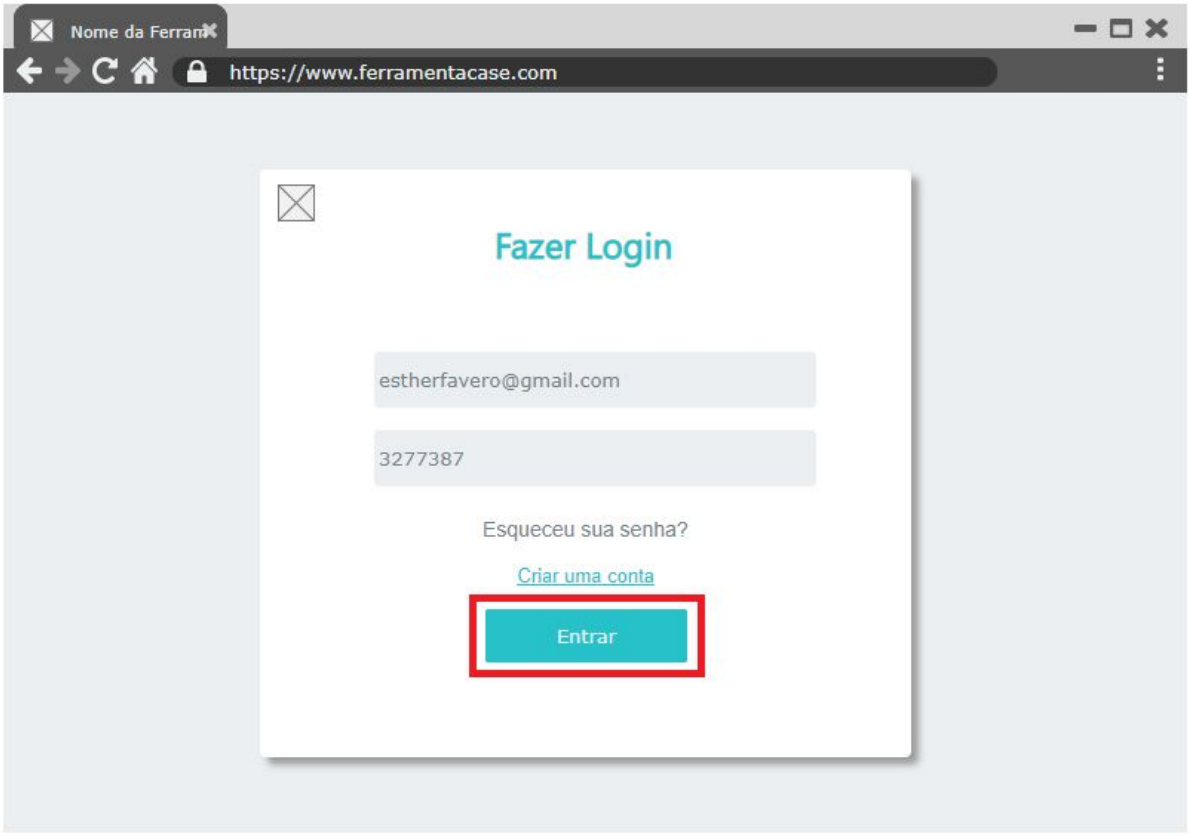
P5.9
P7.14
P8.6
P9.7

APÊNDICE B – CENÁRIOS DA AVALIAÇÃO DO PROTÓTIPO

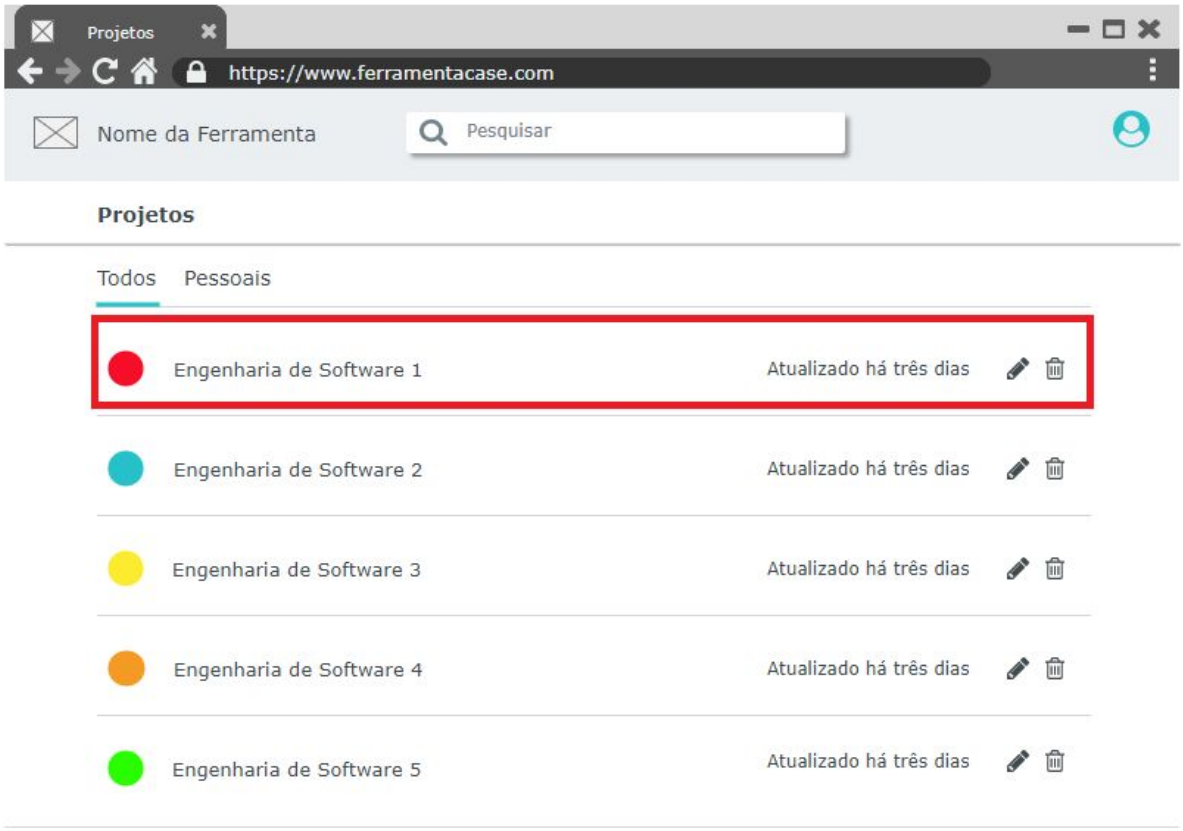
Cenário 1

Para o cenário 1 o participante deveria seguir a sequência de passos:

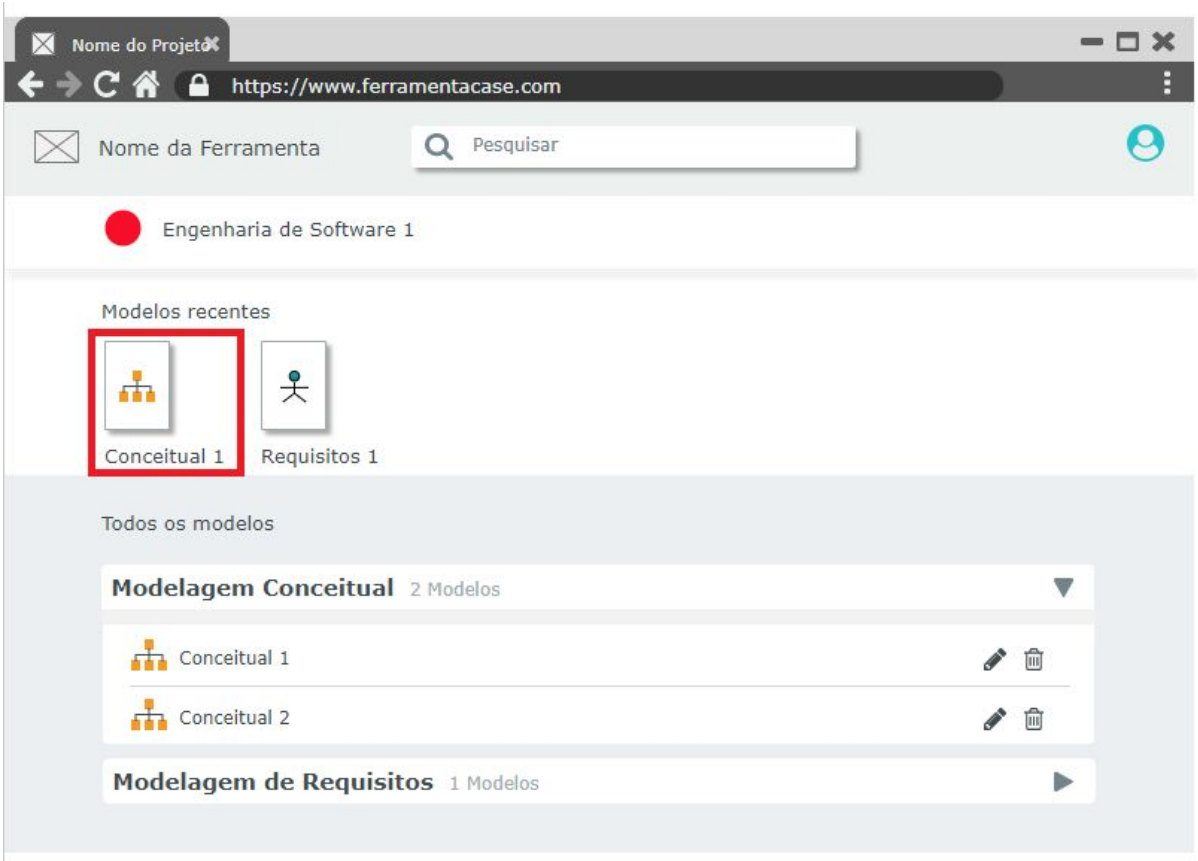
1. Fazer Login
    a) Informar Email
    b) Informar Senha
    c) Pressionar botão “Entrar”



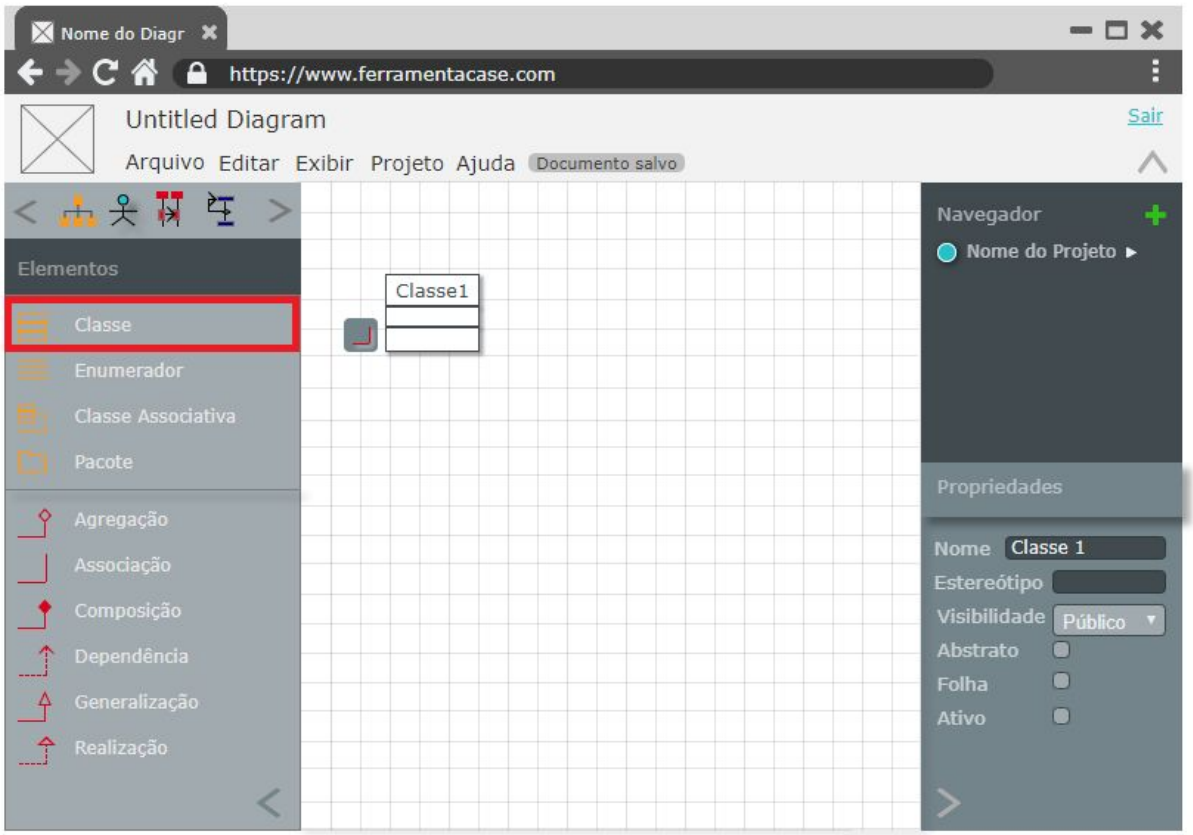
2. Acessar o primeiro projeto
    a) Clicar em “Engenharia de Software 1”



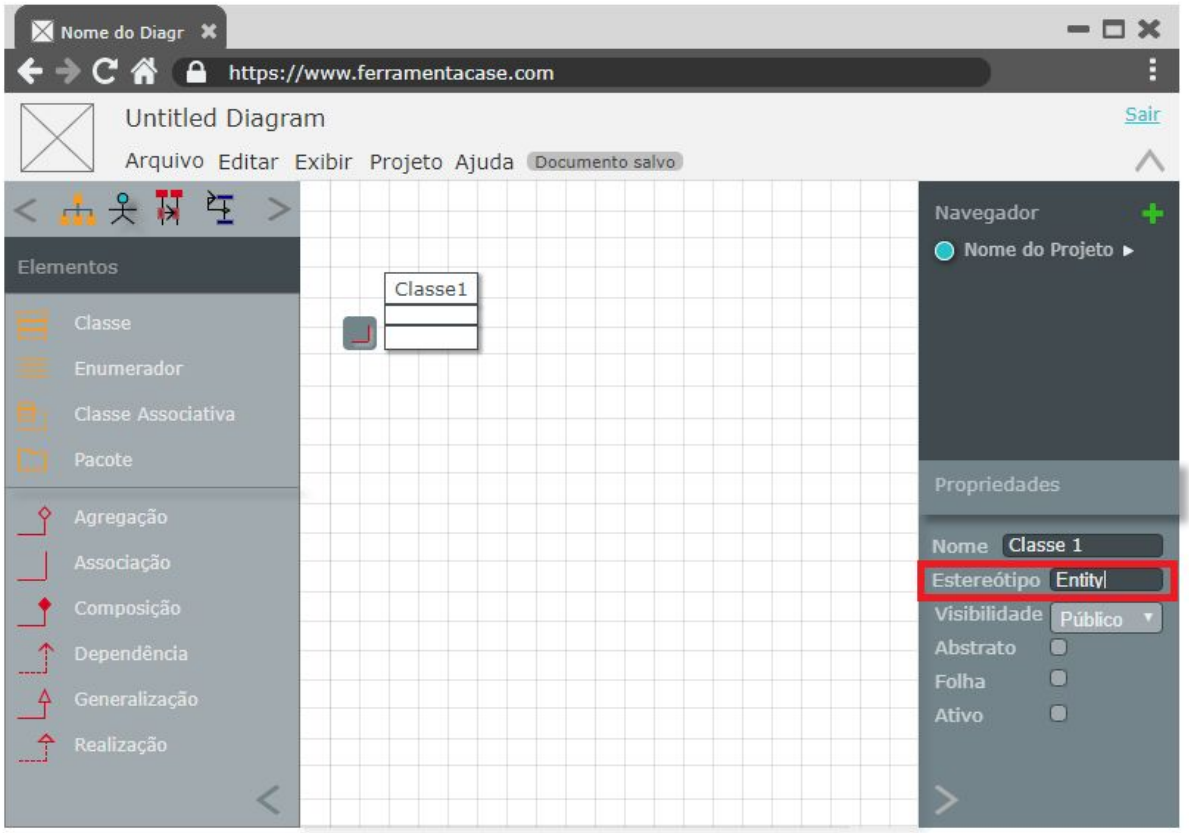
3. Acessar o modelo conceitual mais recente
    a) Clicar em “Conceitual 1”



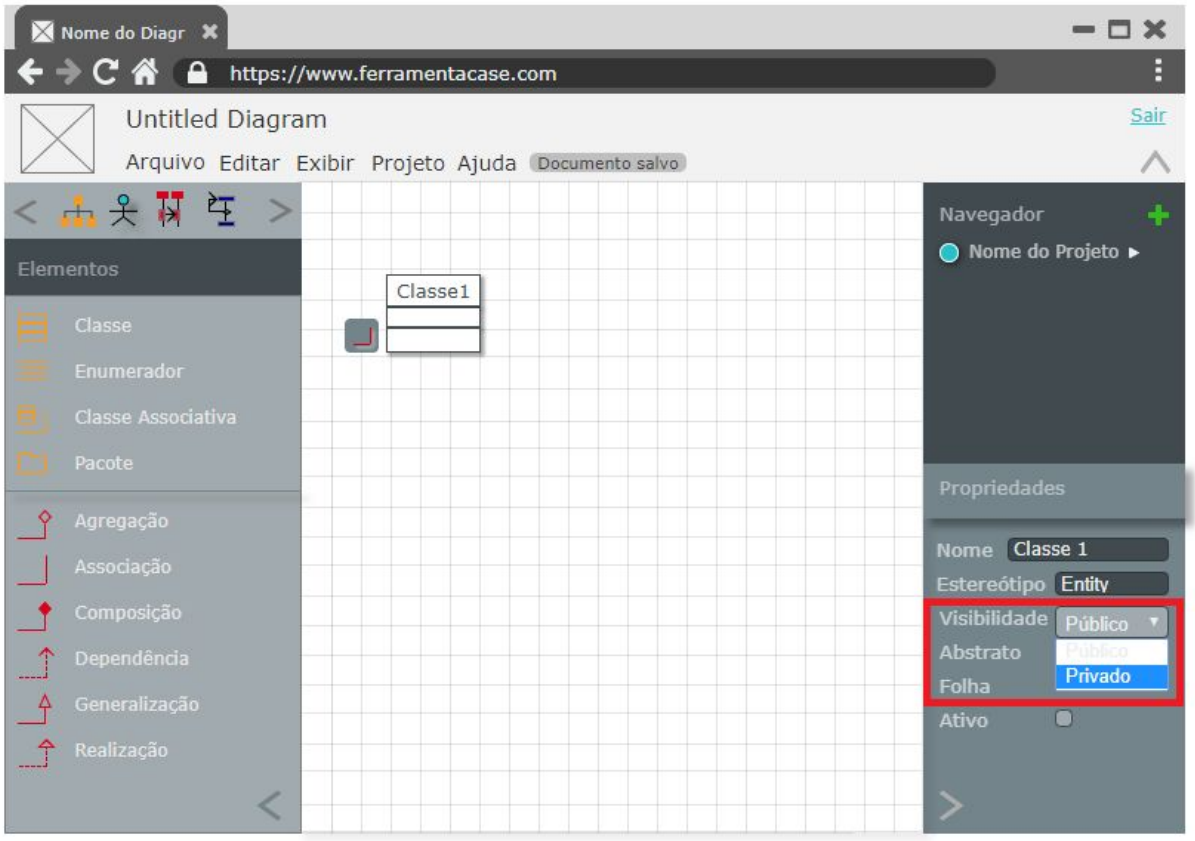
4. Inserir uma classe
    a) Clicar em “Classe”



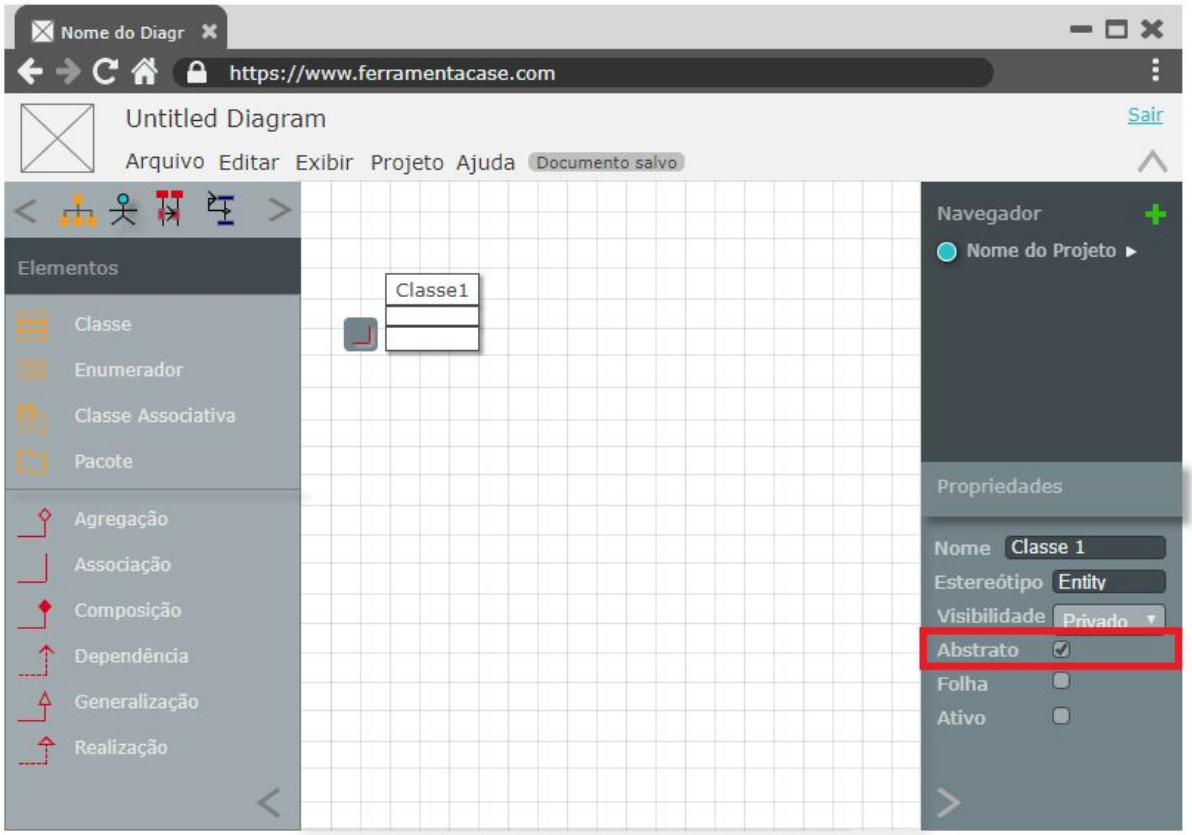
5. Inserir estereótipo
    a) Preencher o campo “Estereótipo”



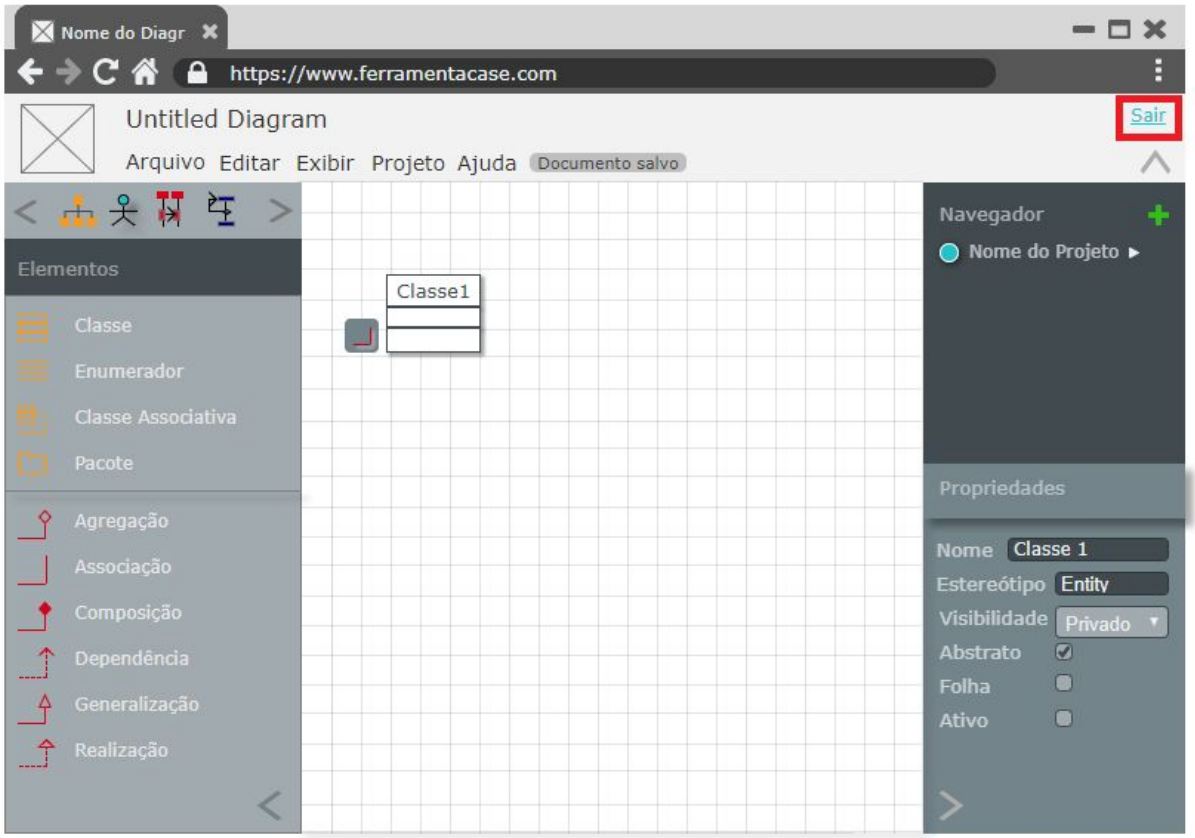
6. Mudar visibilidade para privado
    a) Clicar em “Selecionar dropdown”
    b) Clicar em “Privado”



7. Definir como classe abstrata
    a) Clicar no checkbox de Abstrato



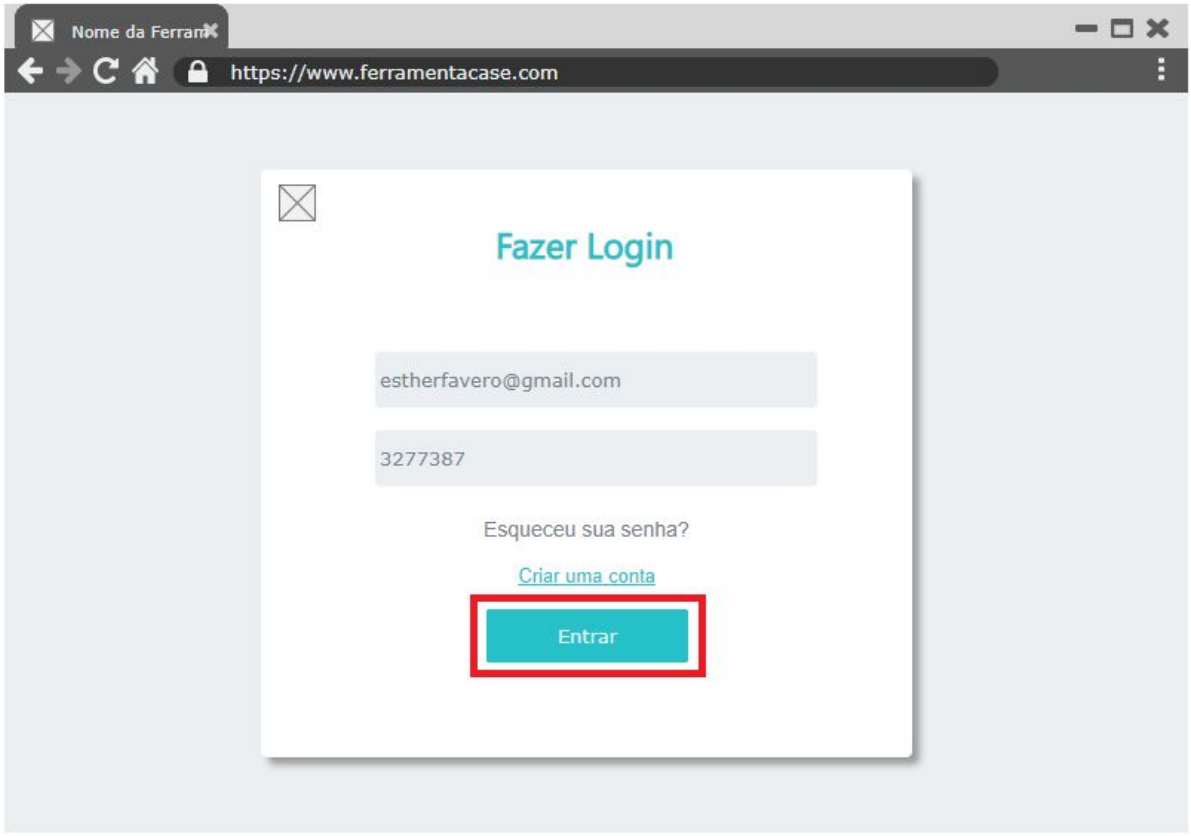
8. Sair do sistema
    a) Clicar em “Sair”


Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Documento salvo
Elementos
Classe
Enumerador
Classe Associativa
Pacote
Agregação
Associação
Composição
Dependência
Generalização
Realização
Classe1
Navegador
Nome do Projeto
Propriedades
Nome
Classe 1
Estereótipo
Entity
Visibilidade
Privado
Abstrato
Folha
Ativo

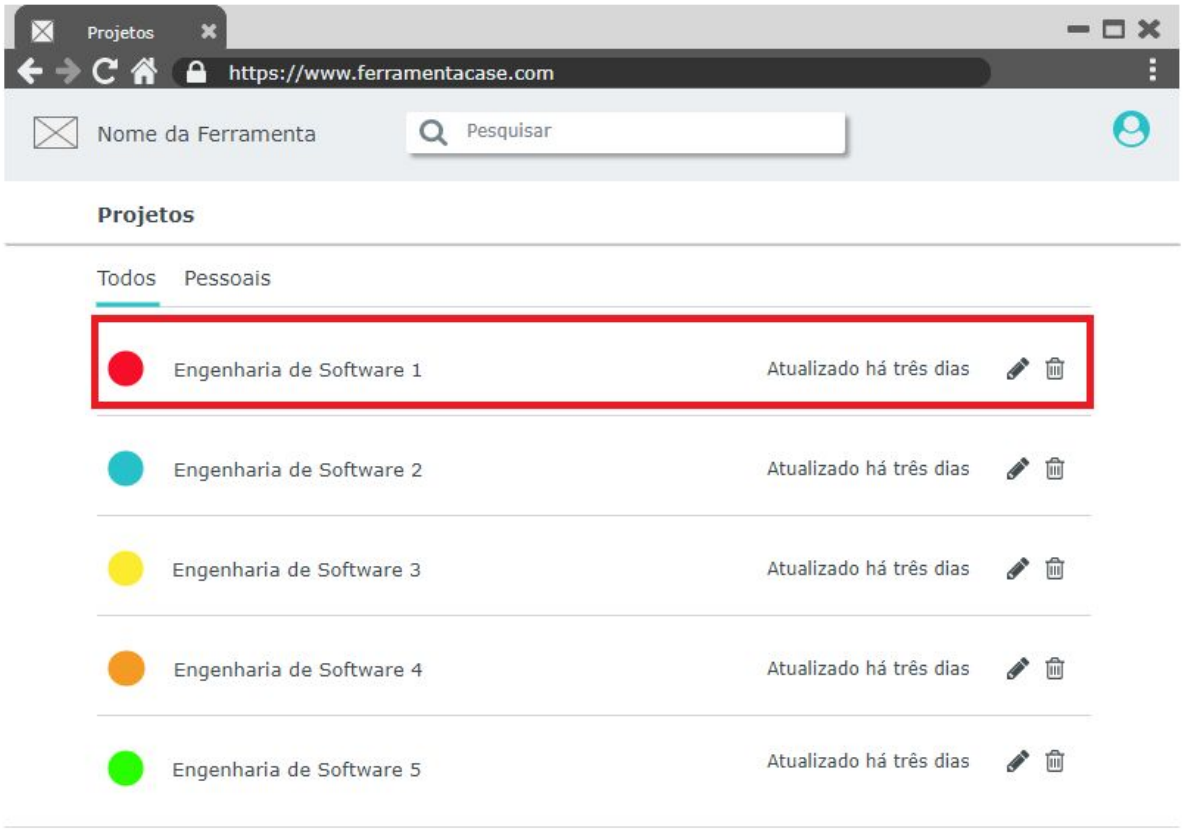
Cenário 2

Para o cenário 2 o participante deveria seguir a sequência de passos:

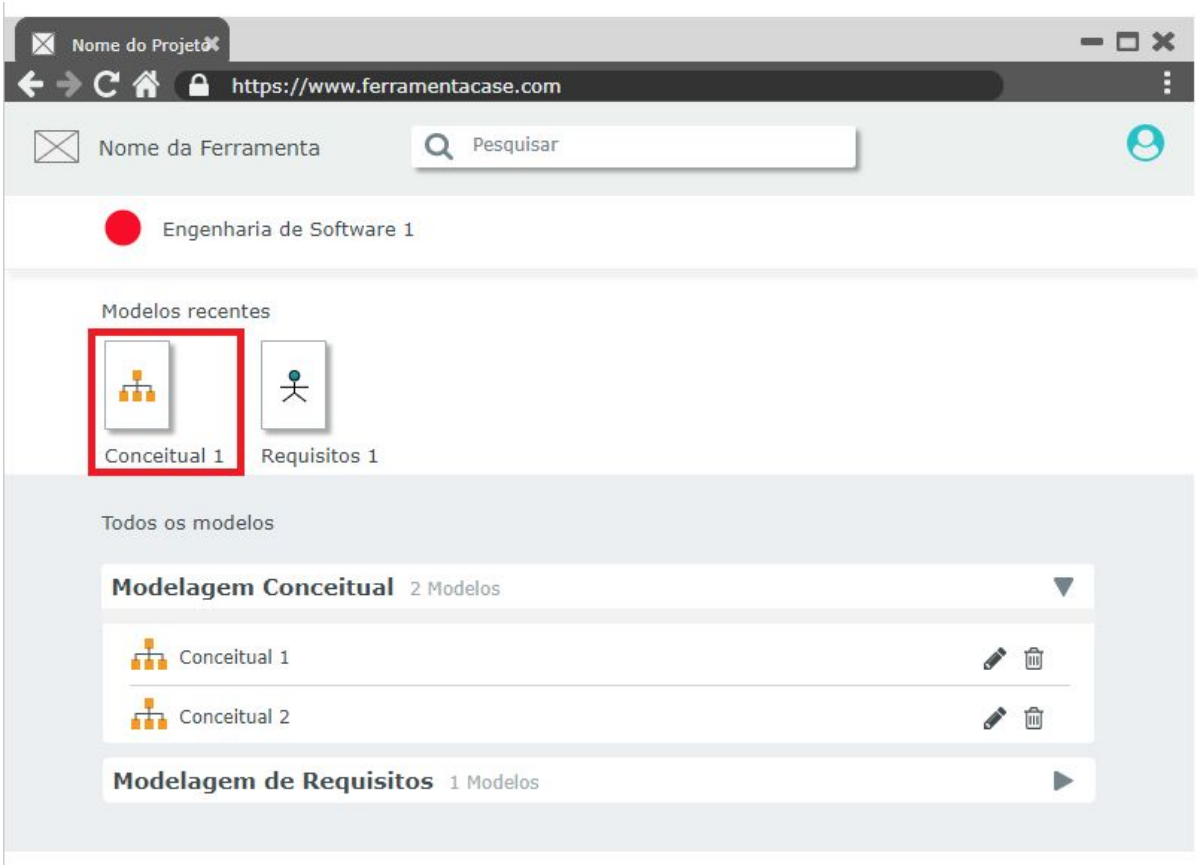
1. Fazer Login
    a) Informar Email
    b) Informar Senha
    c) Pressionar botão “Entrar”



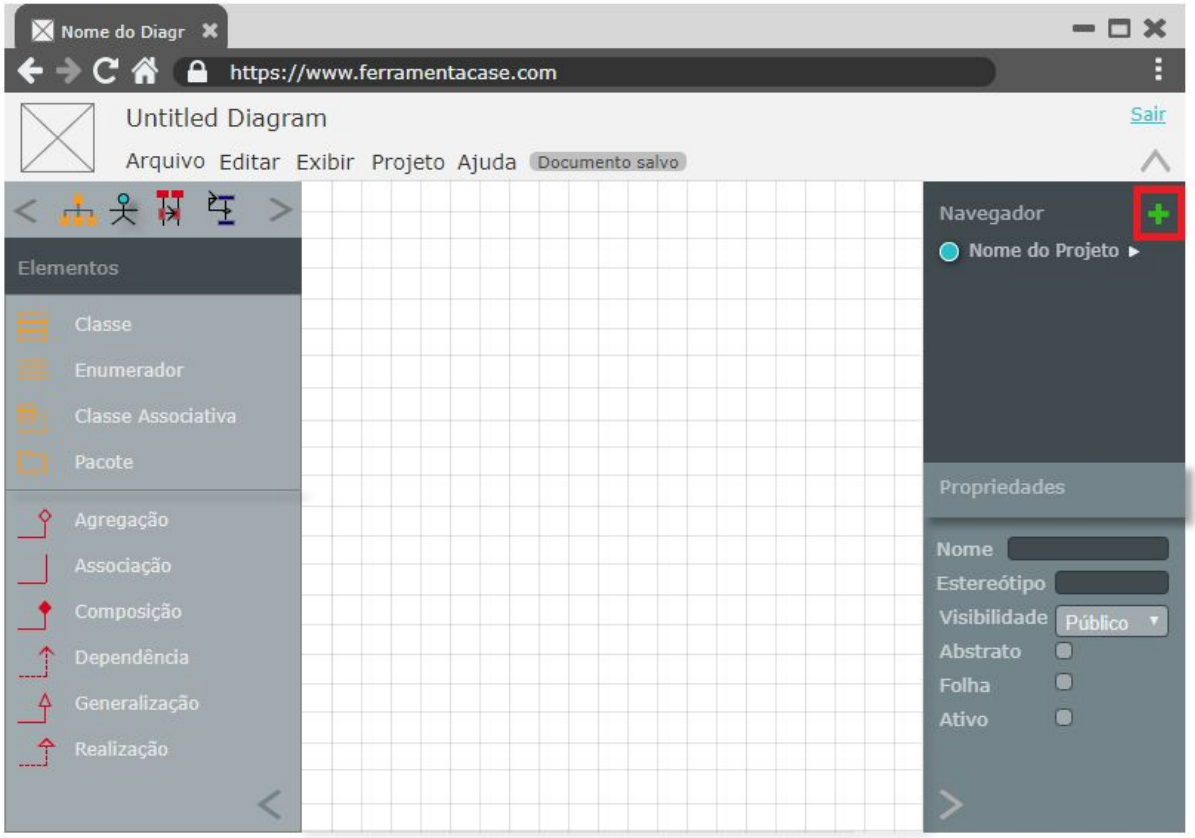
2. Acessar o primeiro projeto
    a) Clicar em “Engenharia de Software 1”

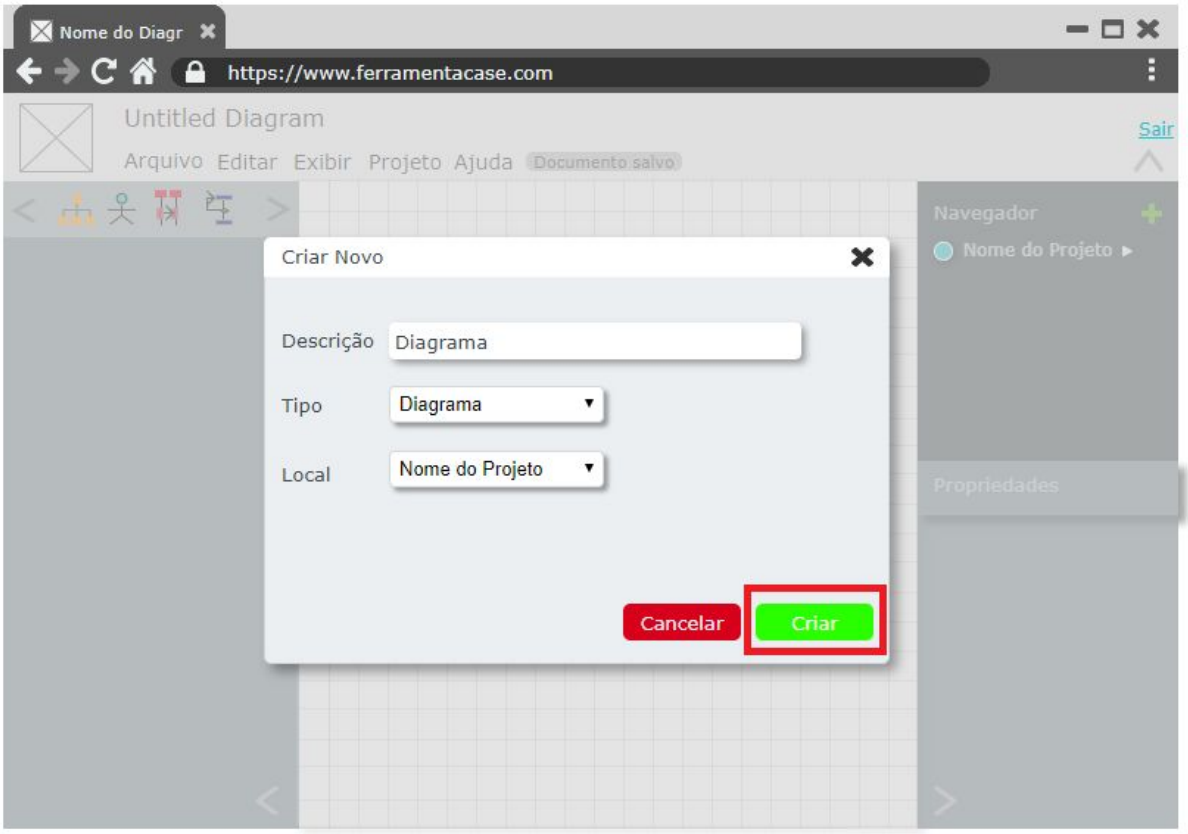


3. Acessar o modelo conceitual mais recente
    a) Clicar em “Conceitual 1”

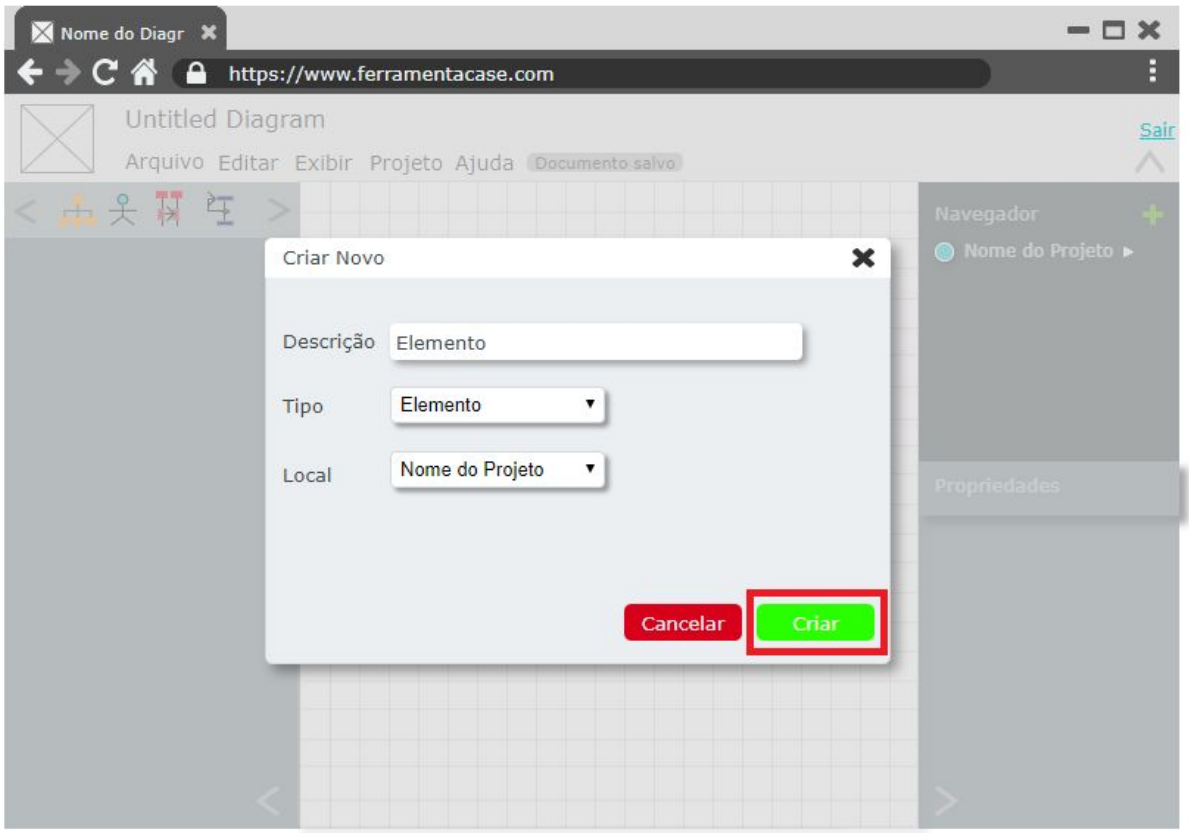


4. Criar um novo Diagrama
    a) Clicar no botão “+”
    b) Inserir a Descrição
    c) Informar o Tipo “Diagrama”
    d) Informar o Local
    e) Clicar no botão “Criar”

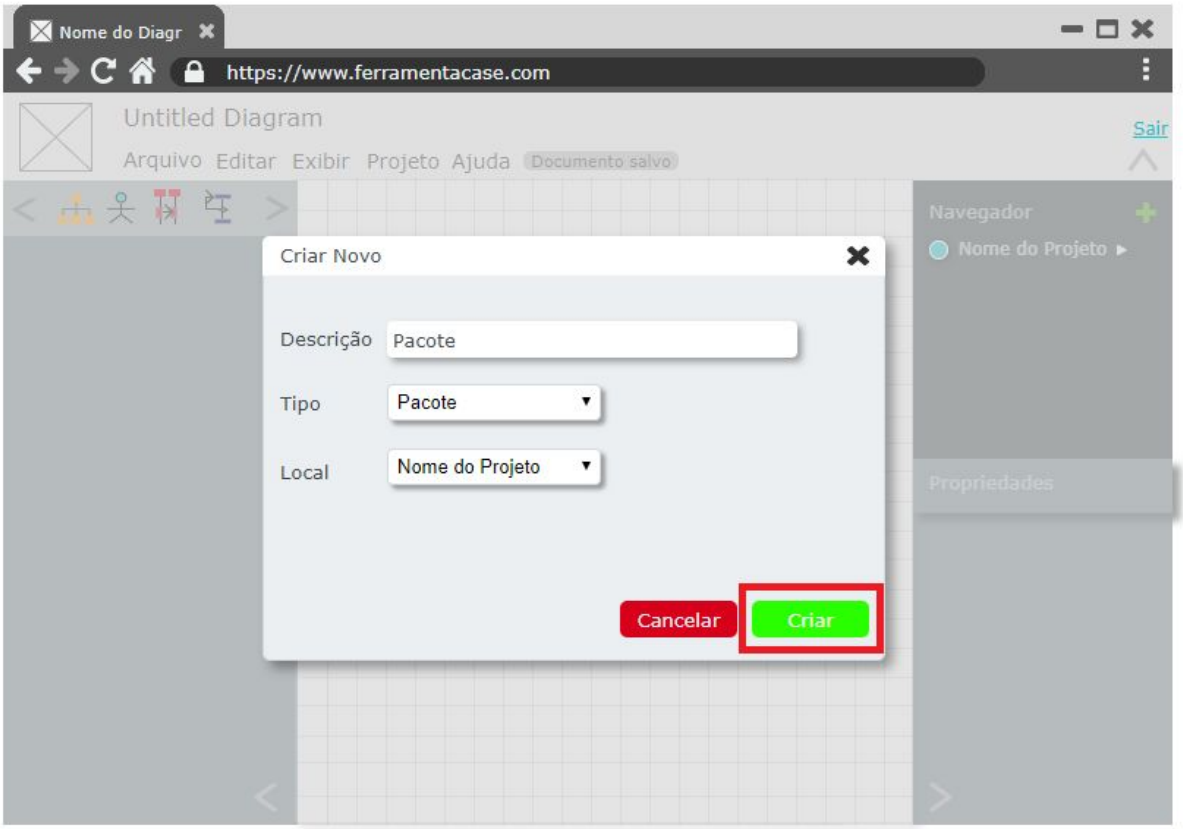

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Documento salvo
Navegador
Nome do Projeto
Elementos
Classe
Enumerador
Classe Associativa
Pacote
Agregação
Associação
Composição
Dependência
Generalização
Realização
Propriedades
Nome
Estereótipo
Visibilidade
Público
Abstrato
Folha
Ativo


Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Documento salvo
Navegador
Nome do Projeto
Propriedades
Criar Novo
Descrição
Diagrama
Tipo
Diagrama
Local
Nome do Projeto
Cancelar
Criar

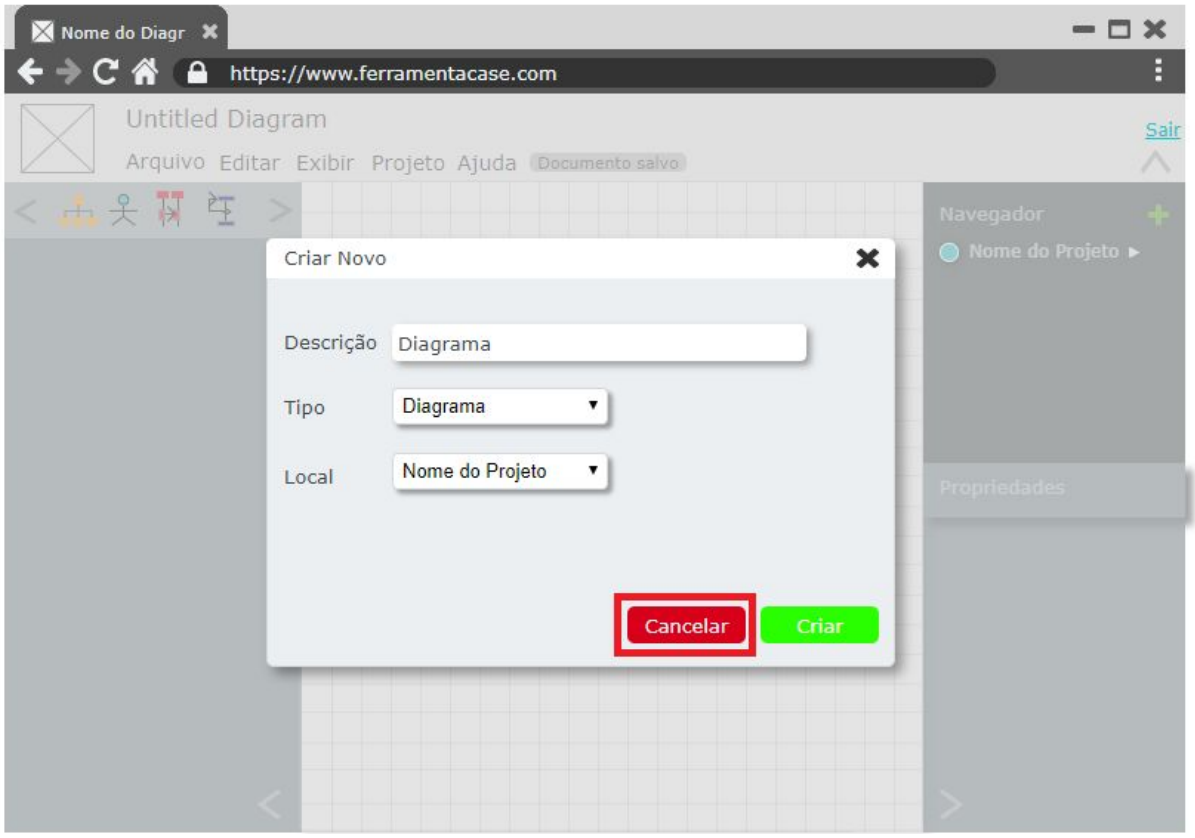
5. Criar um novo Elemento
    a) Clicar no botão “+”
    b) Inserir a Descrição
    c) Informar o Tipo “Elemento”
    d) Informar o Local
    e) Clicar no botão “Criar”



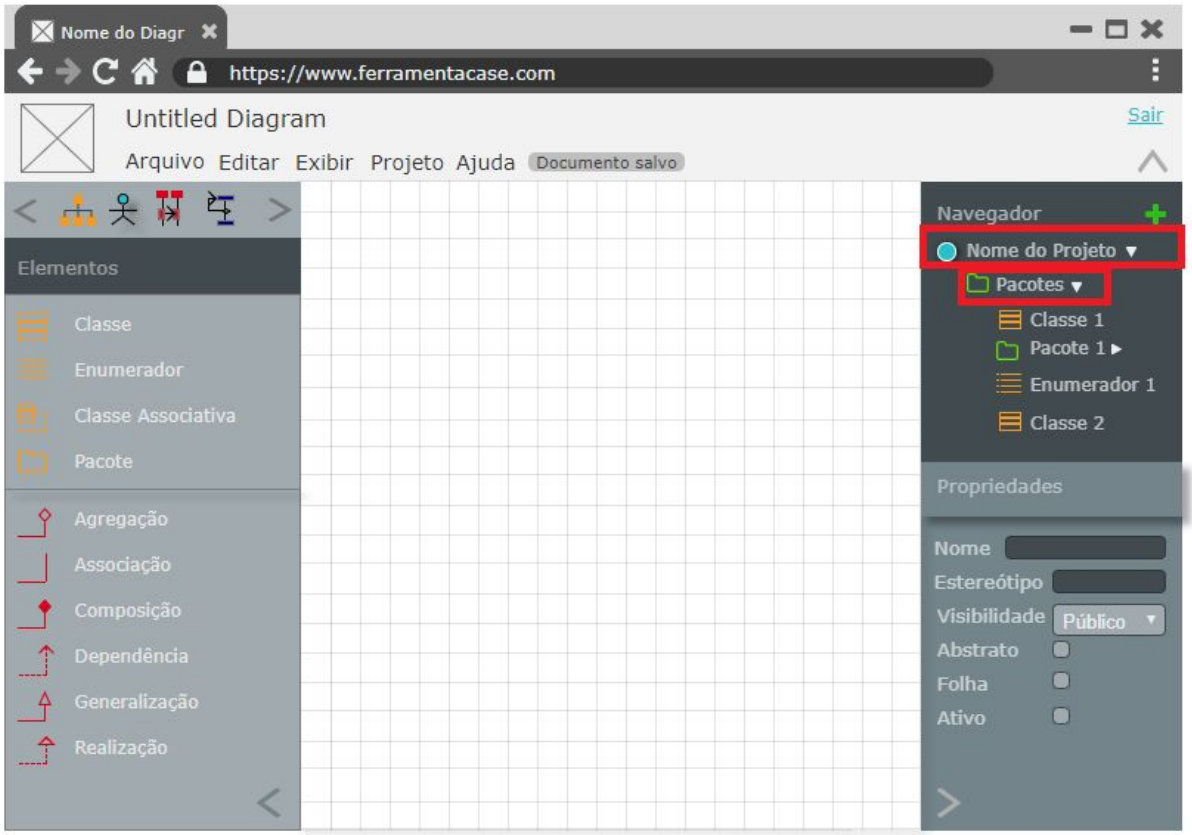
6. Criar um novo Pacote
    a) Clicar no botão “+”
    b) Inserir a Descrição
    c) Informar o Tipo “Pacote”
    d) Clicar no botão “Criar”



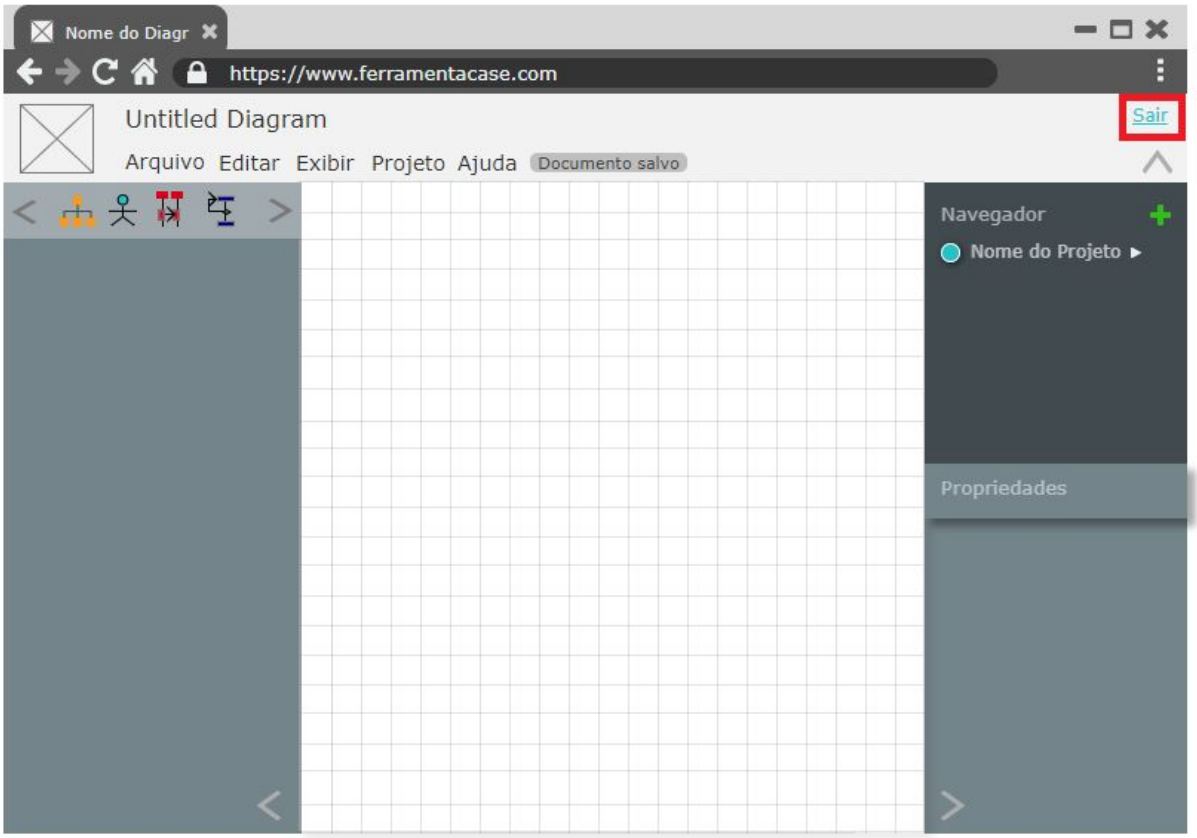
7. Criar um novo Diagrama e cancelar
    a) Clicar no botão “+”
    b) Inserir a Descrição
    c) Informar o Tipo “Diagrama”
    d) Informar o Local
    e) Clicar no botão “Cancelar”



8. Navegar no projeto
    a) Clicar em “Nome do Projeto” ou na flecha
    b) Clicar em “Pacotes”



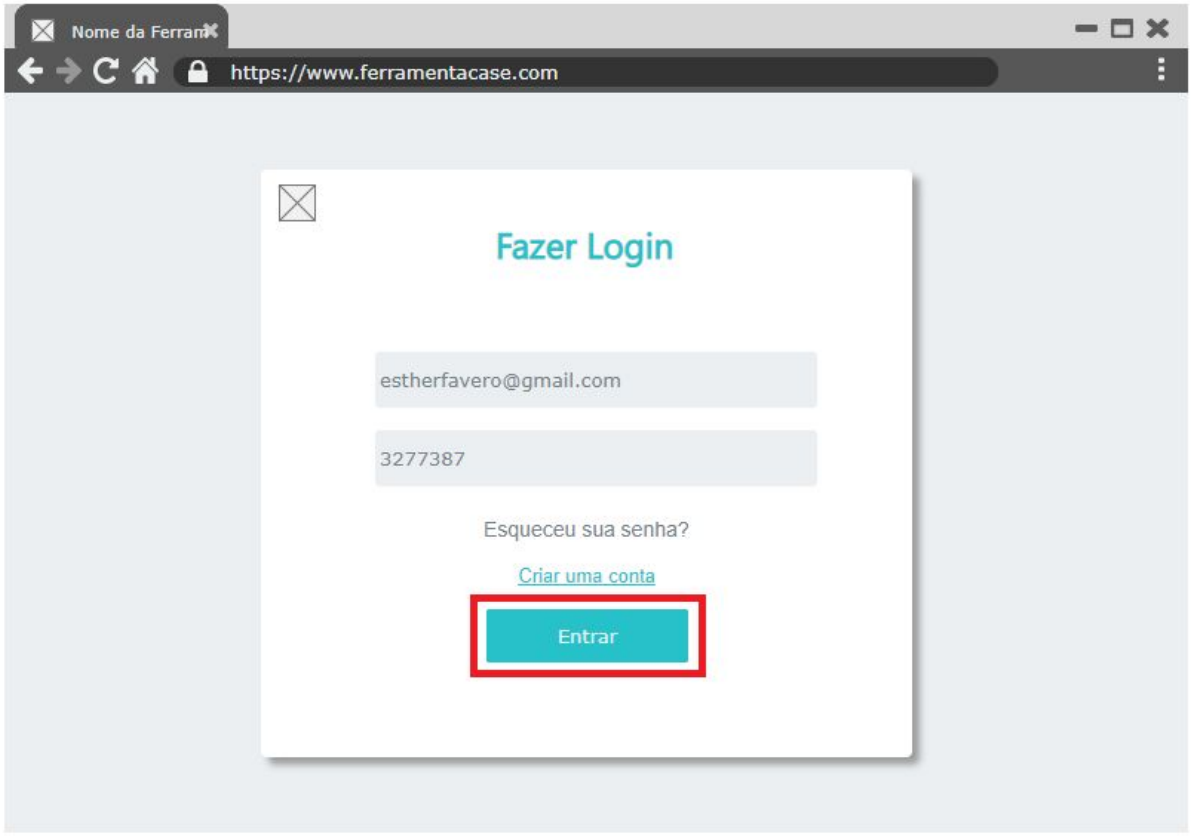
9. Sair do sistema
    a) Clicar em “Sair”


Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Documento salvo
Navegador
Nome do Projeto
Propriedades

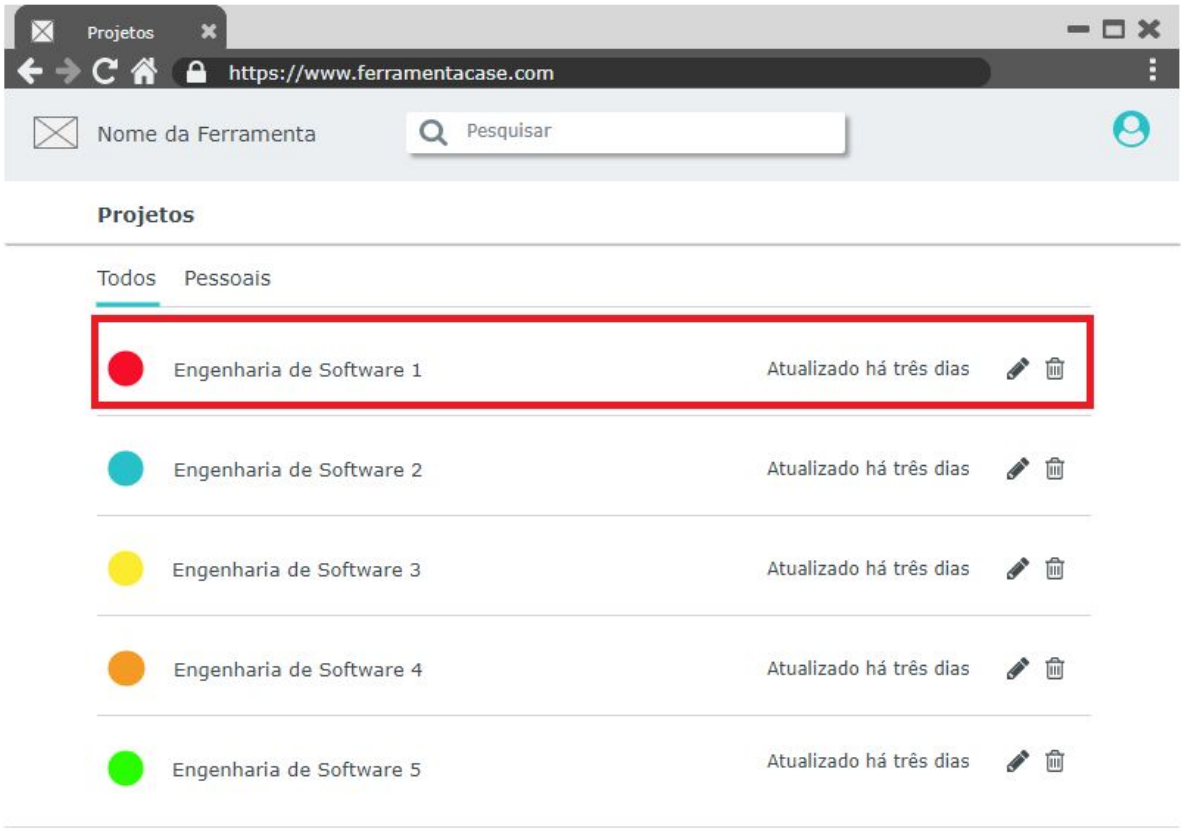
Cenário 3

Para o cenário 3 o participante deveria seguir a sequência de passos:

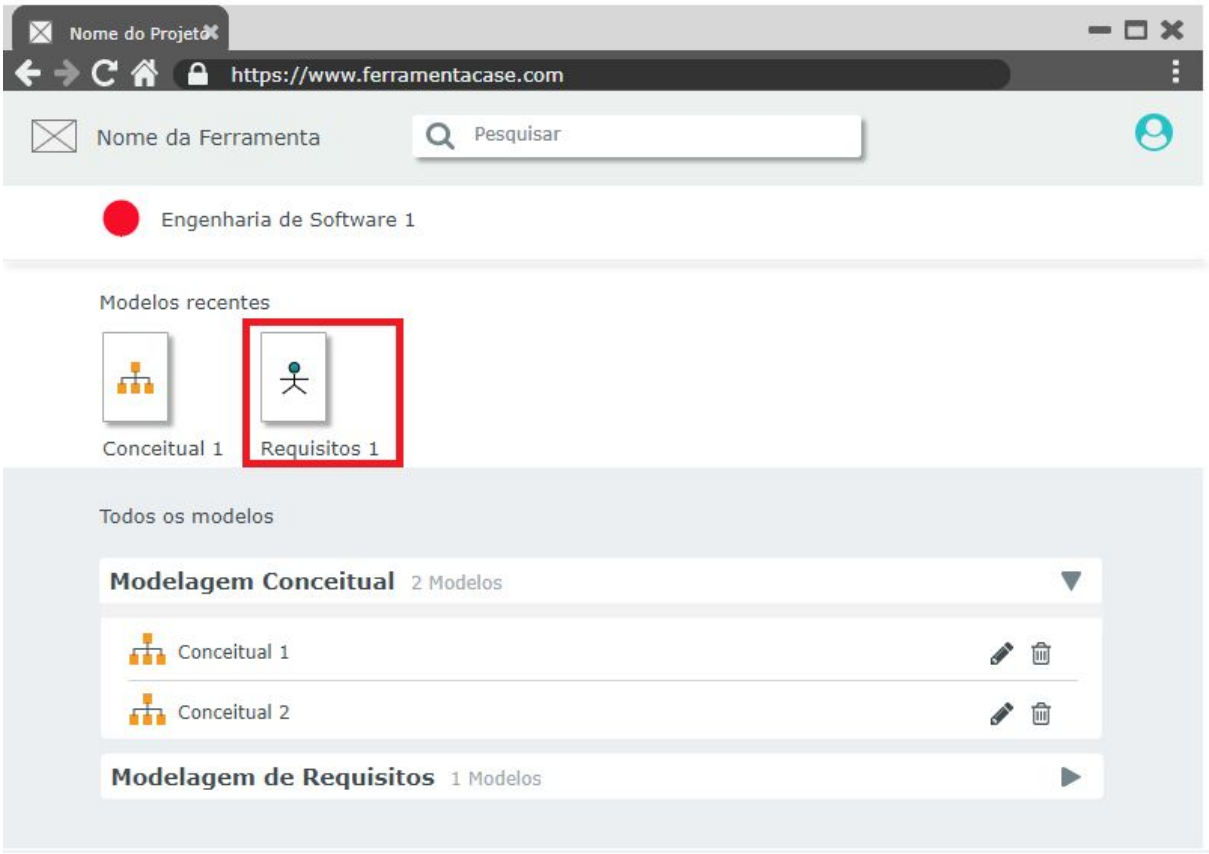
1. Fazer Login
    a) Informar Email
    b) Informar Senha
    c) Pressionar botão “Entrar”



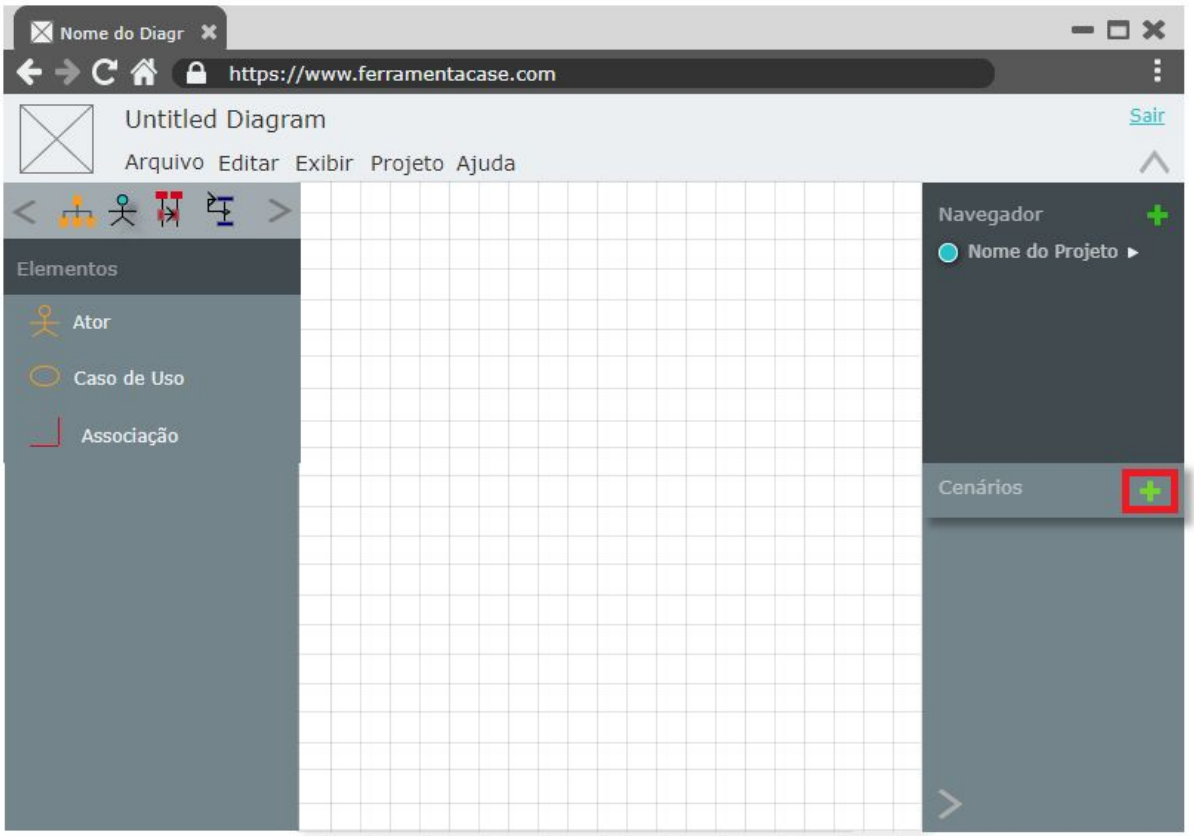
2. Acessar o primeiro projeto
    a) Clicar em “Engenharia de Software 1

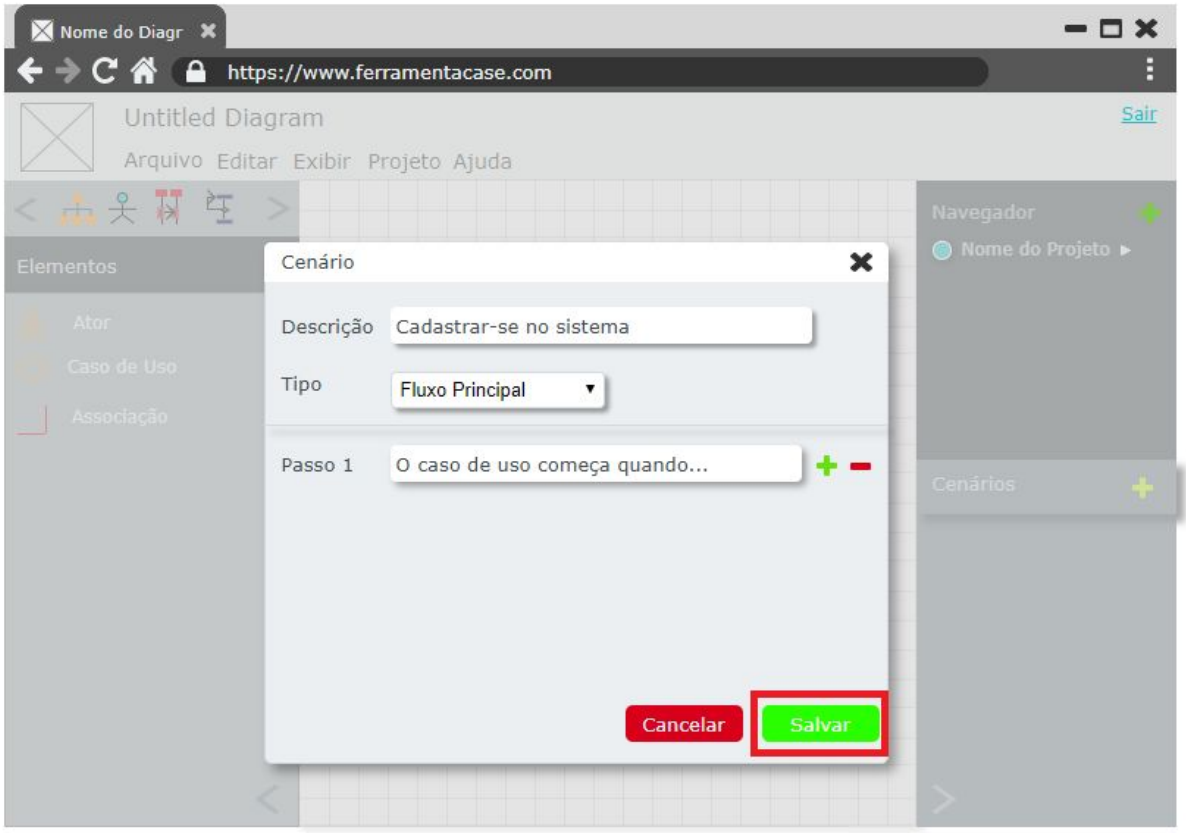


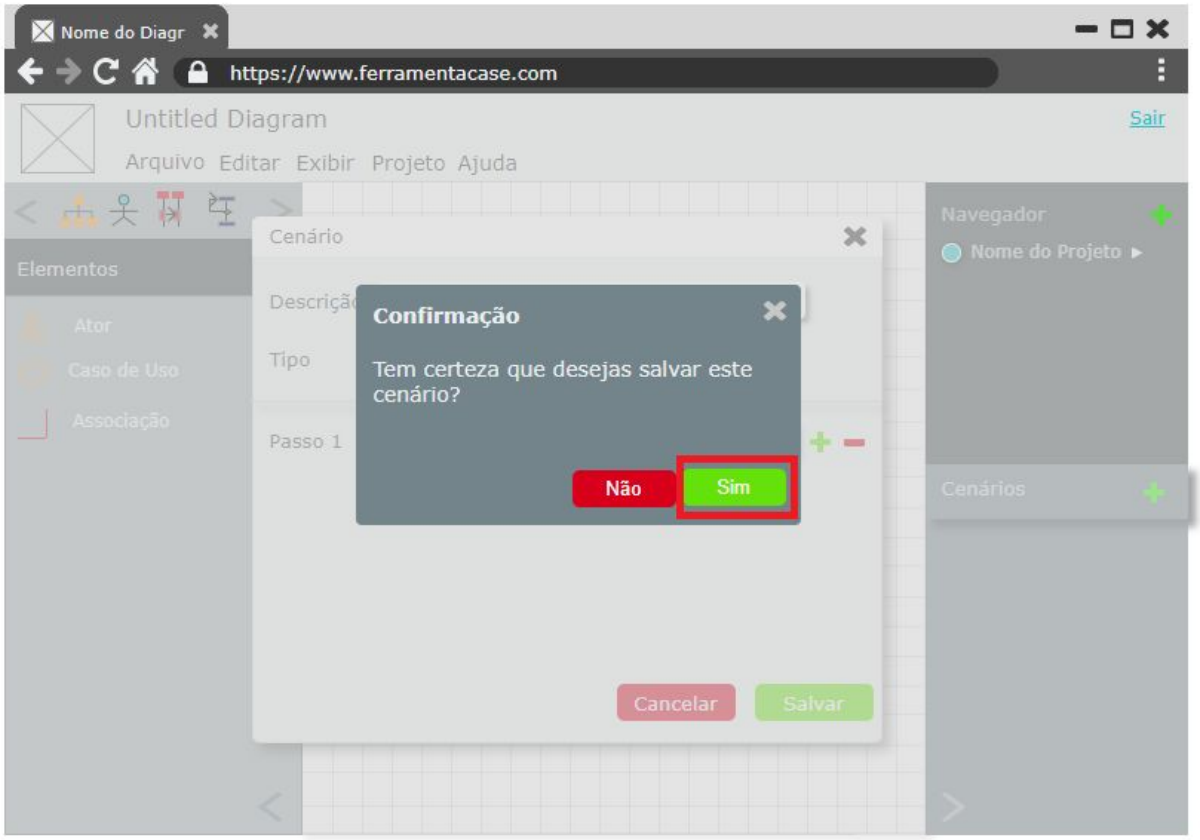
3. Acessar o modelo de requisitos mais recentes
    a) Clicar em “Requisitos 1”



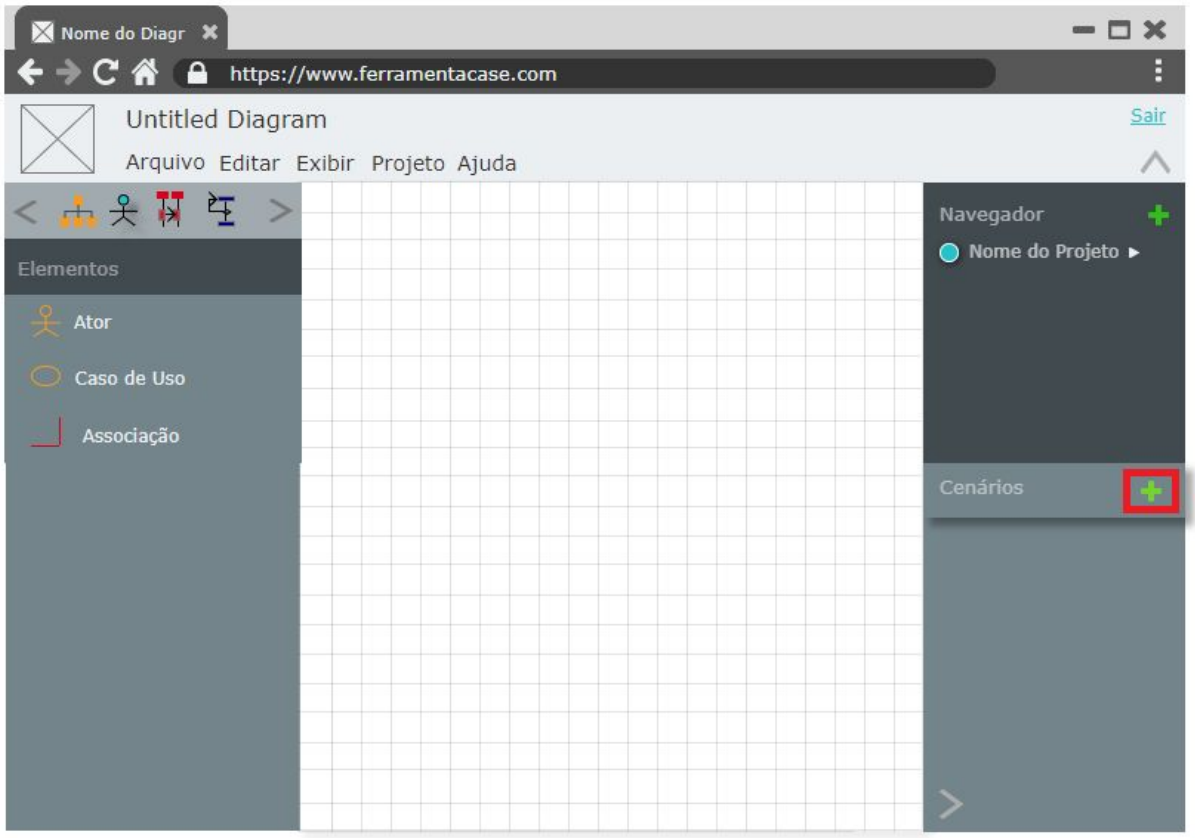
4. Adicionar um novo Cenário com o Fluxo Principal
    a) Informar a Descrição
    b) Selecionar o Tipo “Fluxo Principal”
    c) Adicionar os passos
    d) Clicar no botão “Salvar”
    e) Clicar no botão “Sim” para confirmar

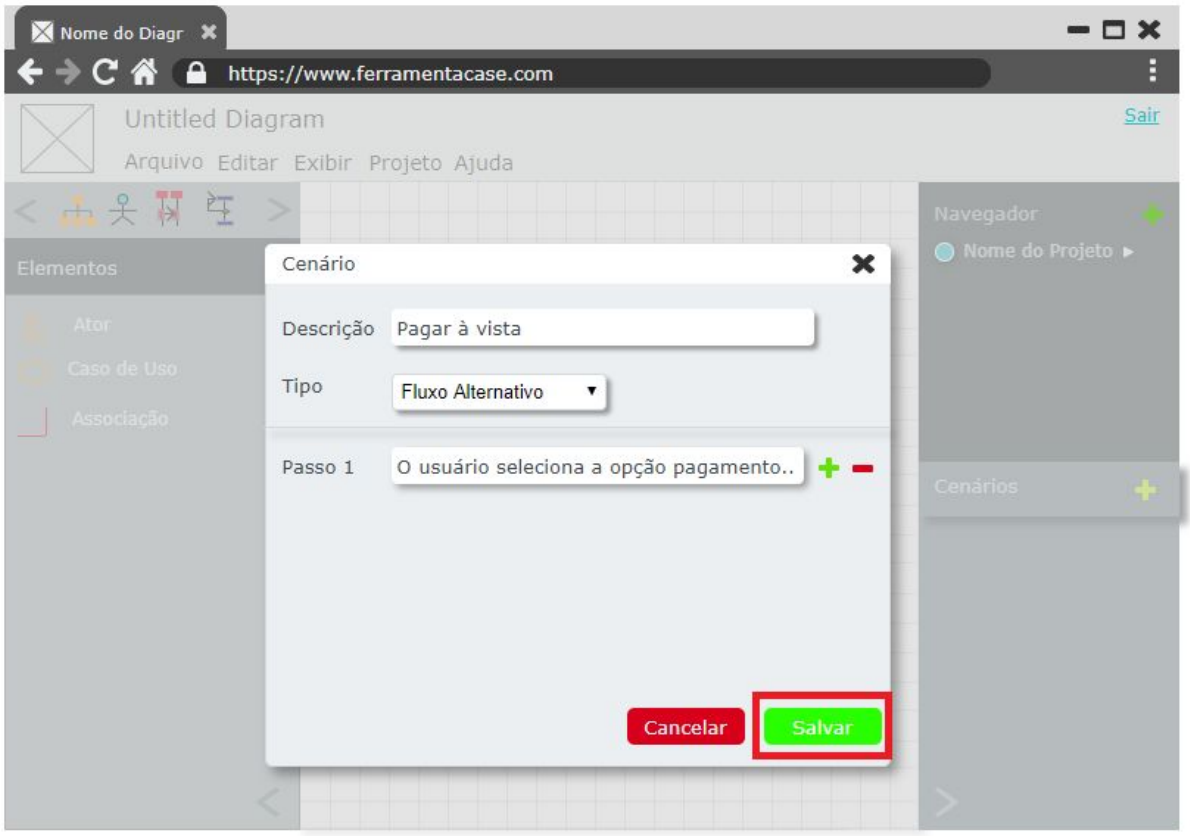

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários

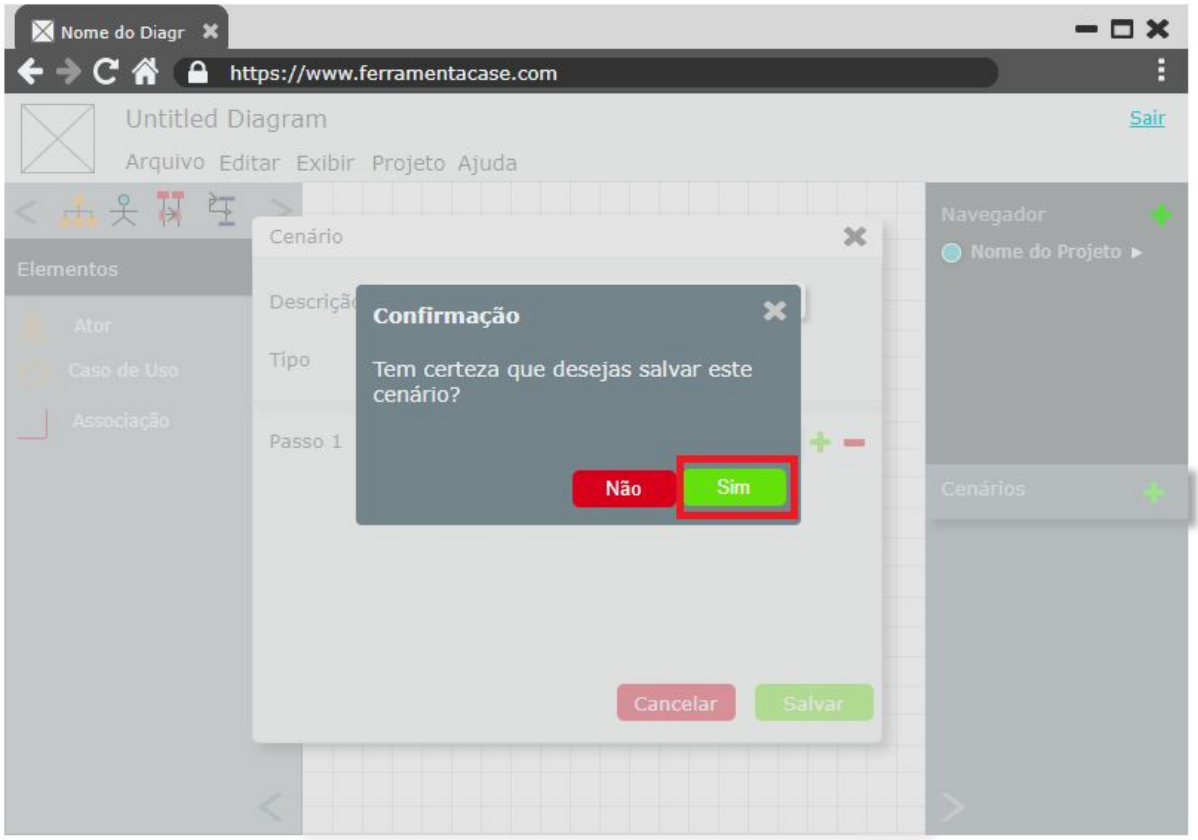

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários
Cenário
Descrição
Cadastrar-se no sistema
Tipo
Fluxo Principal
Passo 1
O caso de uso começa quando...
Cancelar
Salvar



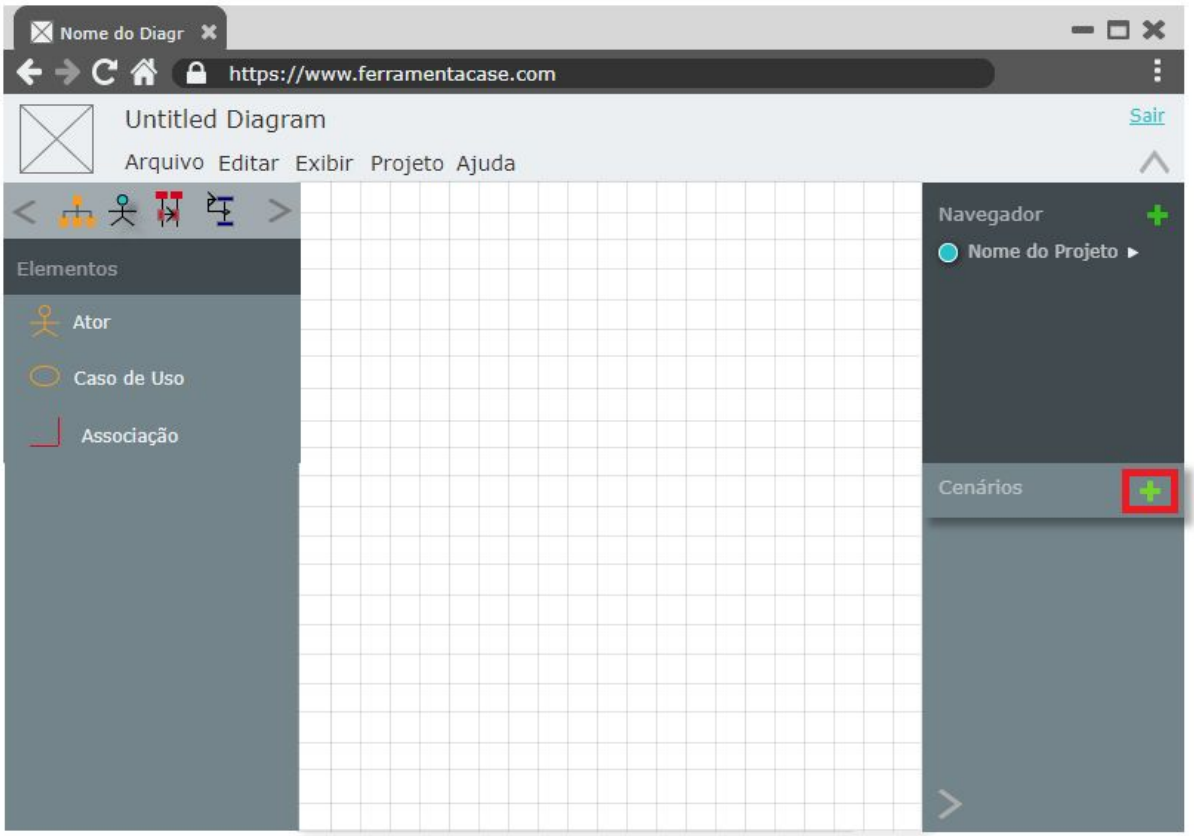
5. Adicionar um novo Cenário com o Fluxo Alternativo
    a) Informar a Descrição
    b) Selecionar o Tipo “Fluxo Alternativo”
    c) Adicionar os passos
    d) Clicar no botão “Salvar”
    e) Clicar no botão “Sim” para confirmar

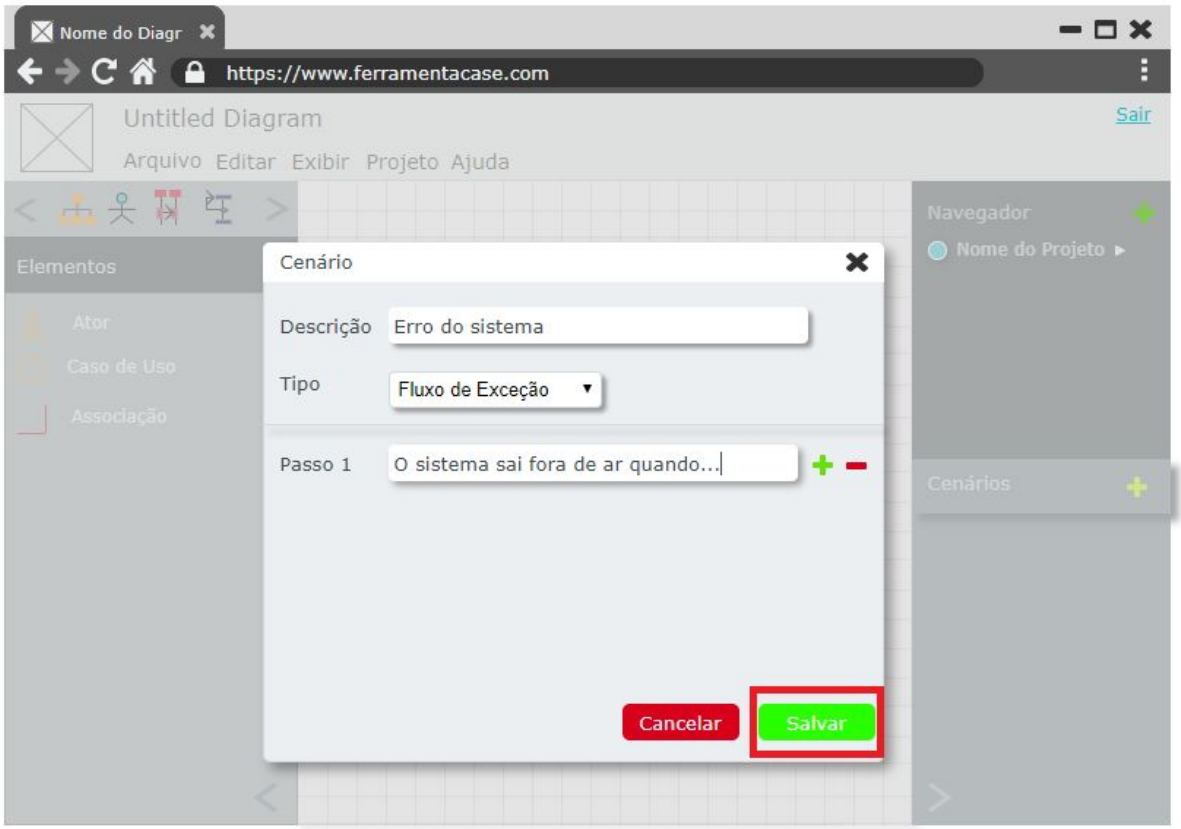

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários

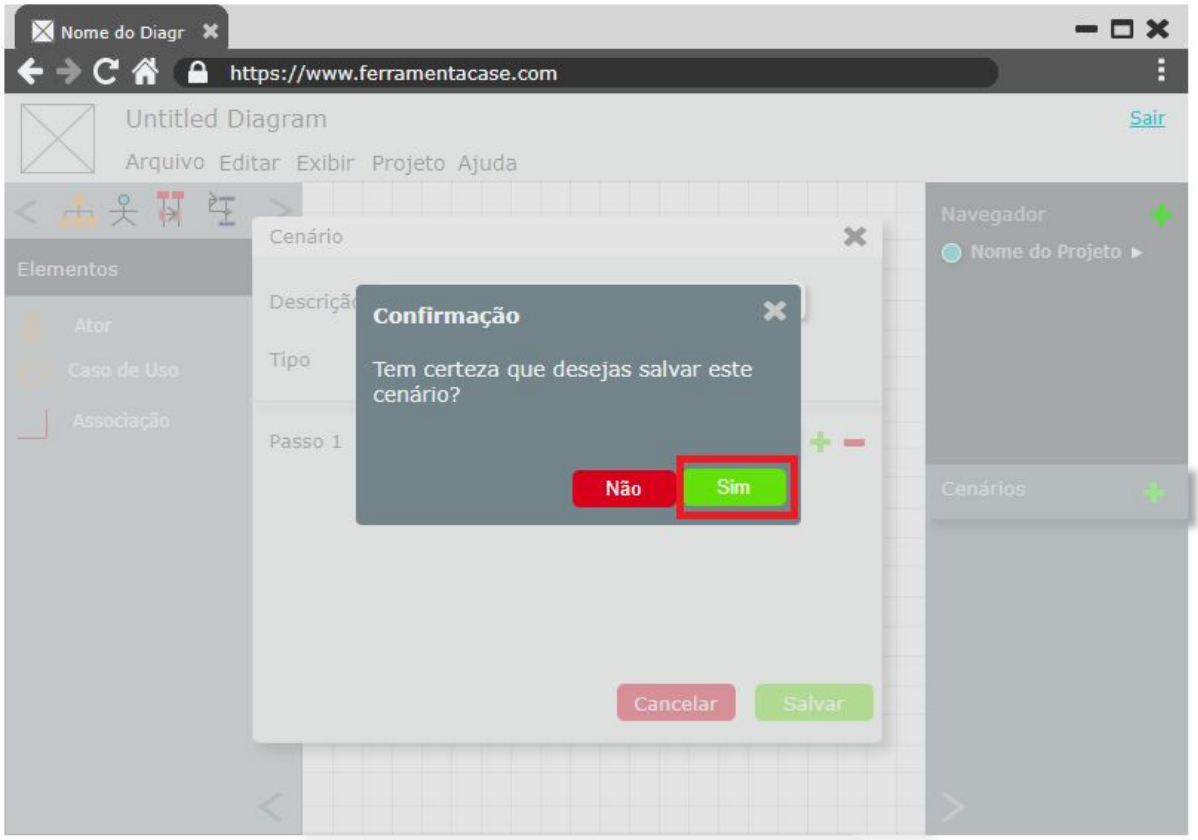

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários
Cenário
Descrição
Pagar à vista
Tipo
Fluxo Alternativo
Passo 1
O usuário seleciona a opção pagamento..
Cancelar
Salvar



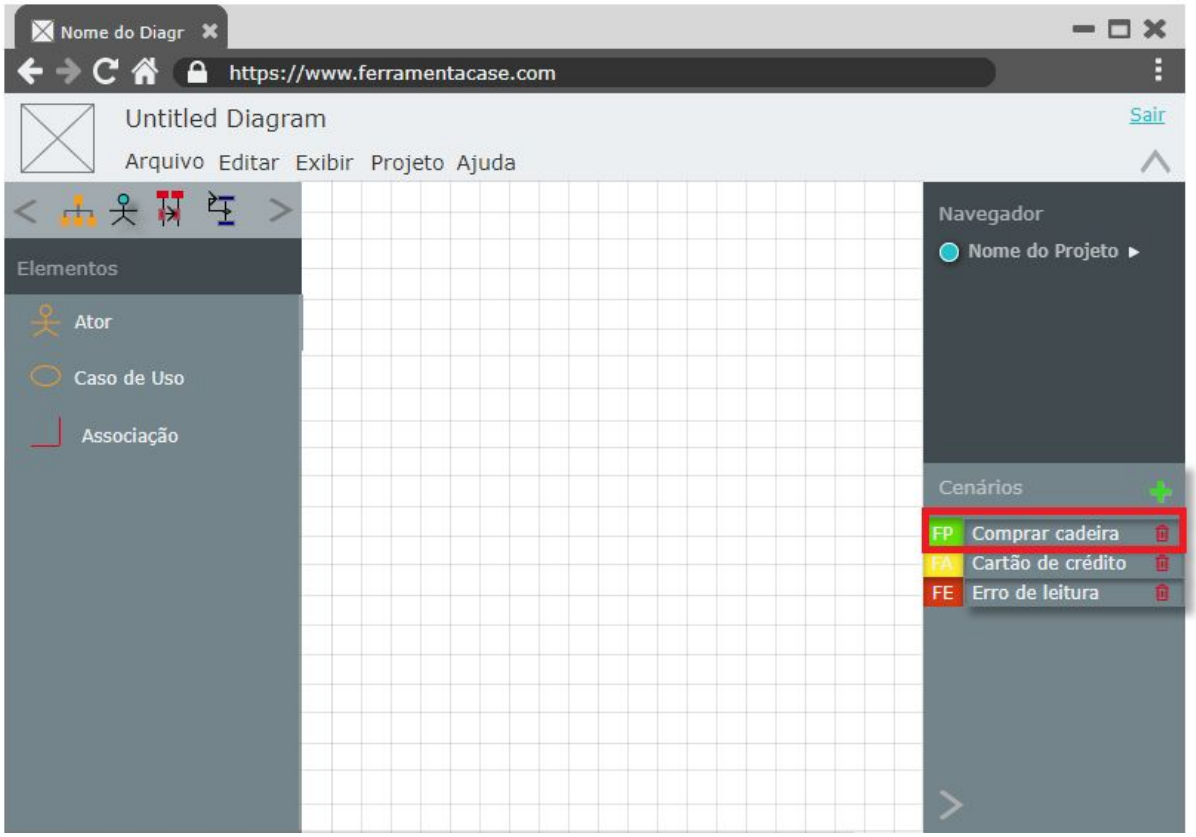
6. Adicionar um novo Cenário com o Fluxo de Exceção
    a) Informar a Descrição
    b) Selecionar o Tipo “Fluxo Alternativo de Exceção”
    c) Adicionar os passos
    d) Clicar no botão “Salvar”
    e) Clicar no botão “Sim” para confirmar

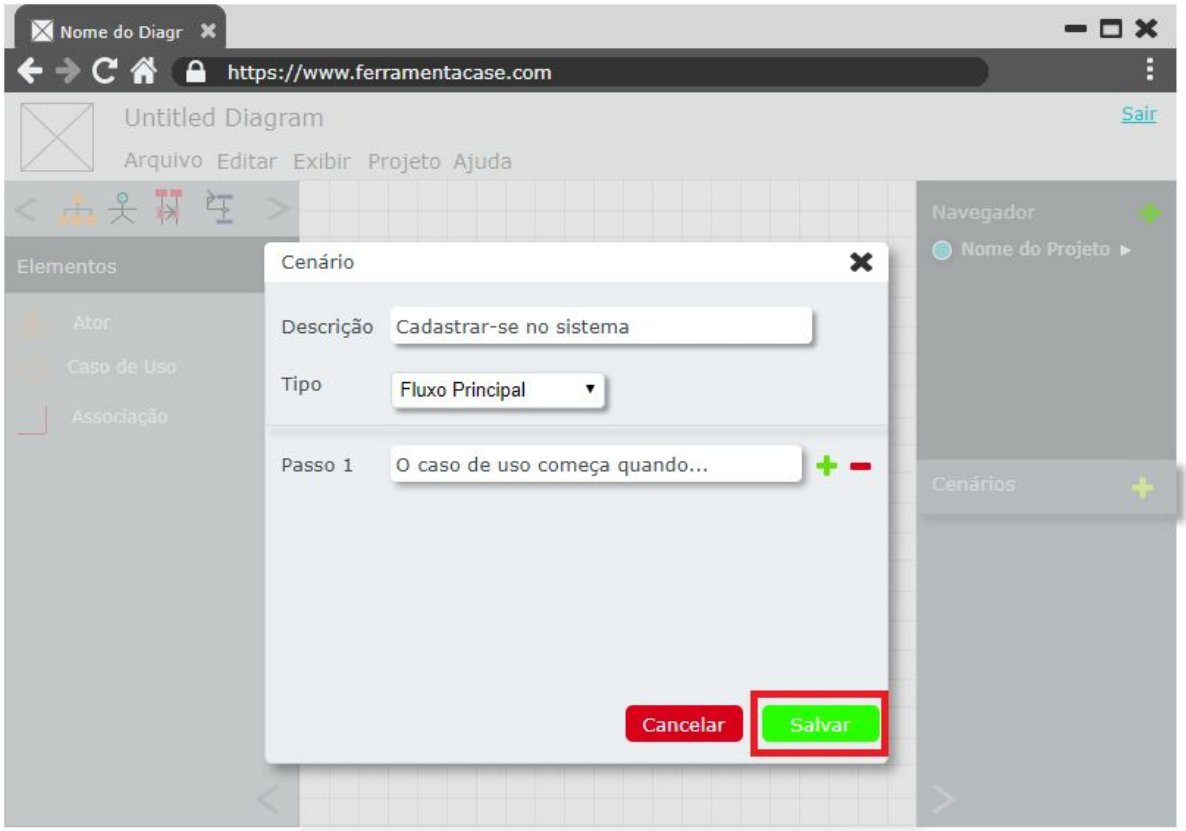

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários

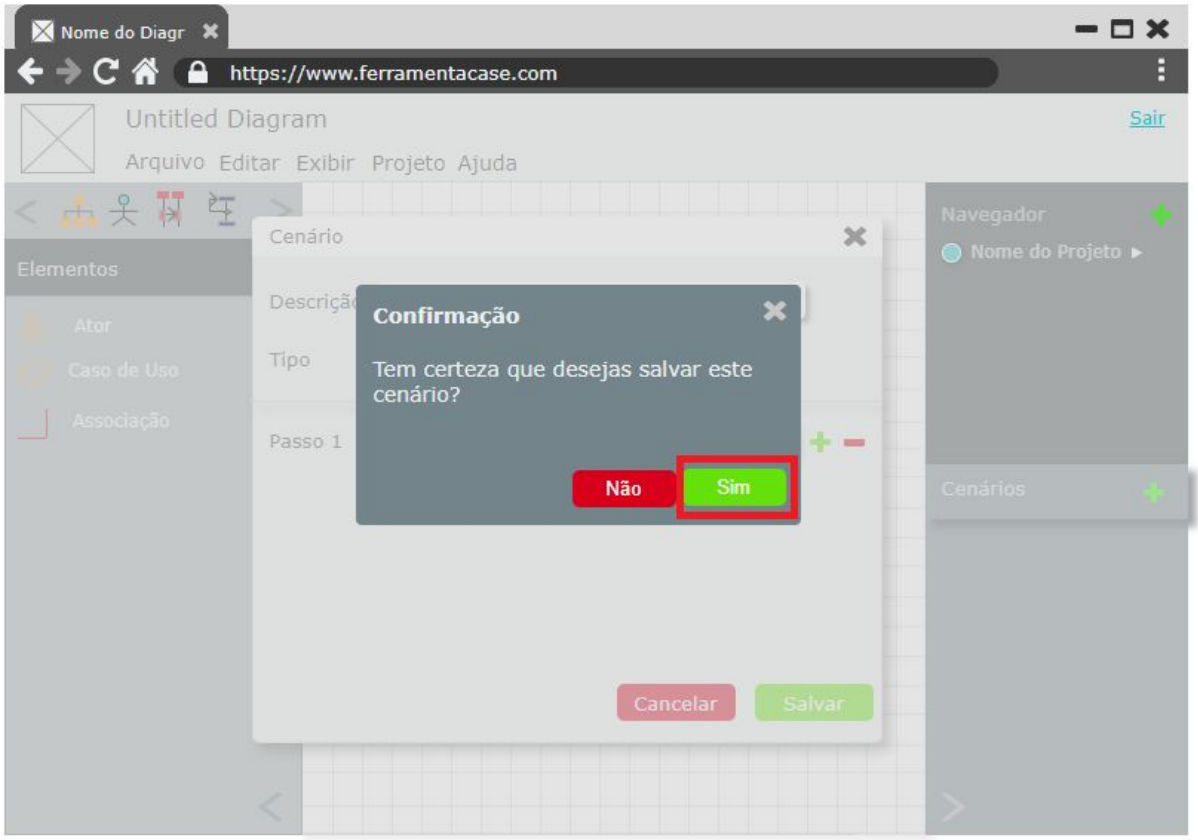

Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários
Cenário
Descrição
Erro do sistema
Tipo
Fluxo de Exceção
Passo 1
O sistema sai fora de ar quando...
Cancelar
Salvar



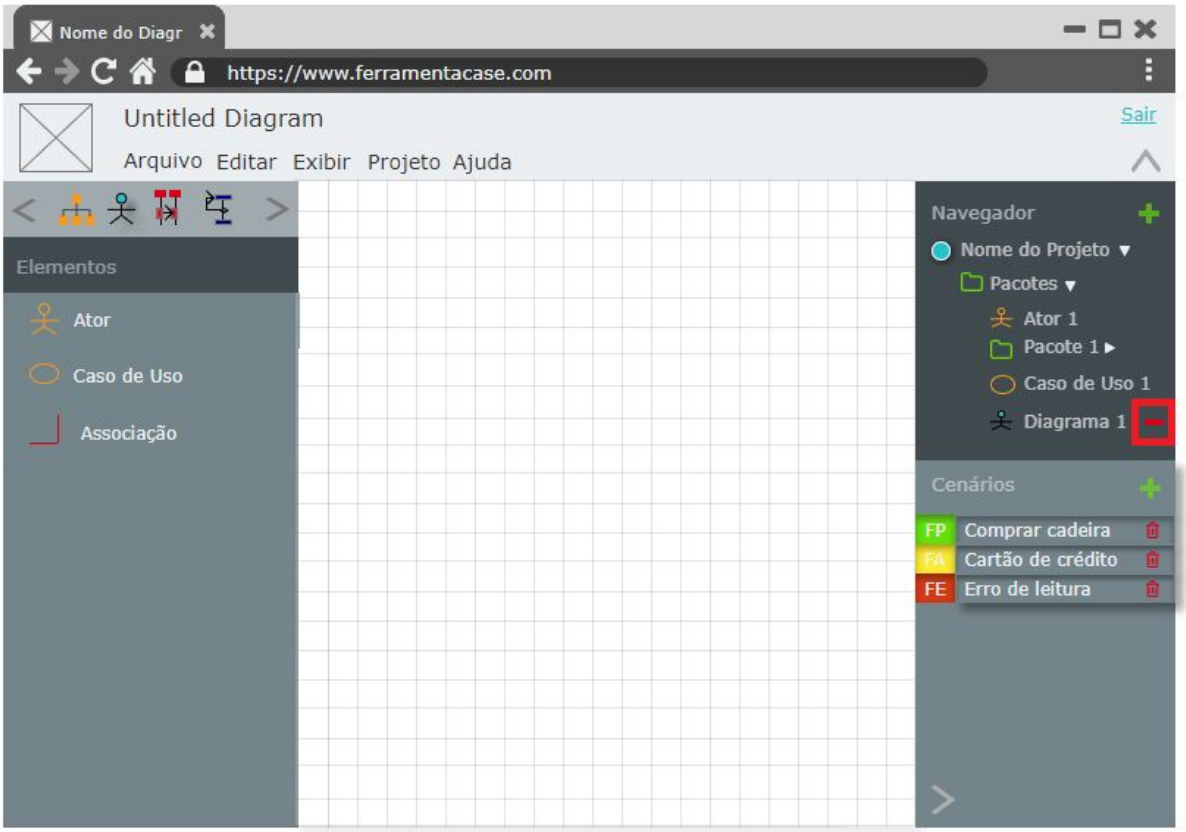
7. Editar o primeiro Fluxo Principal
    a) Clicar em “Comprar Cadeira”
    b) Editar o Nome, o Tipo ou os Passos.
    c) Clicar no botão “Salvar”
    d) Clicar no botão “Sim” para confirmar


Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários
FP Comprar cadeira
FA Cartão de crédito
FE Erro de leitura


Nome do Diagr
https://www.ferramentacase.com
Untitled Diagram
Sair
Arquivo Editar Exibir Projeto Ajuda
Elementos
Ator
Caso de Uso
Associação
Navegador
Nome do Projeto
Cenários
Cenário
Descrição
Cadastrar-se no sistema
Tipo
Fluxo Principal
Passo 1
O caso de uso começa quando...
Cancelar
Salvar



8. Excluir o Diagrama 1
    a) Clicar em “Nome do Projeto”
    b) Clicar em “Pacotes”
    c) Clicar em “Diagrama 1”
    d) Clicar no “-”



9. Sair do sistema
    a) Clicar em “Sair”