

Desenvolvimento de uma ferramenta para o auxílio na criação de histórias do usuário

João Paulo Cardoso de Souza¹, Vanessa Matias Leite¹

¹Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – CEP 86057-970 – Londrina – PR – Brasil

joao.paulo@uel.br, vanessa.leite@uel.br

Abstract. *The development of software projects is a complex task that demands clear and precise communication among all involved areas, especially during the requirements gathering phase. A failure at this stage can lead to serious issues such as significant delays or even failure to meet client requests. In light of this, this work seeks to develop a tool that assists in the elaboration of these requirements through the creation of user stories, based on usability premises. The proposed tool aims to facilitate the work of developers who use the agile Scrum method by providing detailed user stories and metrics based on Nielsen's heuristics, thereby contributing to the creation of more intuitive and user-friendly digital products.*

Resumo. *O desenvolvimento de projetos de software é uma tarefa complexa que demanda uma comunicação clara e precisa entre todas as áreas envolvidas, especialmente na fase de levantamento de requisitos. Uma falha nessa etapa pode causar sérios problemas, como atrasos significativos ou até mesmo o não cumprimento das solicitações dos clientes. Diante disso, este trabalho busca o desenvolvimento de uma ferramenta que auxilie na elaboração desses requisitos através da criação de histórias de usuário, baseando-se em premissas de usabilidade. A ferramenta proposta tem como objetivo facilitar o trabalho dos desenvolvedores, que utilizam o método ágil Scrum, fornecendo histórias de usuário detalhadas e métricas baseadas nas heurísticas de Nielsen, contribuindo assim para a criação de produtos digitais mais intuitivos e agradáveis para os usuários.*

1. Introdução

O desenvolvimento de projetos requer o trabalho de diversas áreas diferentes e a comunicação entre elas é indispensável. A falta de compreensão de uma solicitação ou requisito pode resultar em sérios problemas no resultado final ou em atrasos significativos, comprometendo a viabilidade do projeto. Dentre os principais fatores que contribuem para esses problemas, destacam-se o planejamento inadequado e a falta de transparência nos requisitos [6].

Para auxiliar no desenvolvimento de software, existem modelos de processos que estruturam e ordenam as tarefas. O modelo de processo clássico, conhecido como cascata, adota uma abordagem sequencial e sistemática para o desenvolvimento de sistemas, mas tem sido criticado por não se adequar bem à natureza não linear do desenvolvimento de software e pela dificuldade em definir todos os requisitos no início do projeto [13]. Como alternativa, os métodos ágeis surgiram para trazer mais dinamismo e menos burocracia

ao desenvolvimento, adotando um desenvolvimento incremental ideal para projetos com requisitos em constante mudança. Esses métodos envolvem o cliente continuamente no processo, permitindo a adição de novos requisitos e a avaliação frequente das interações do sistema. Nos métodos ágeis, a simplicidade é essencial tanto no desenvolvimento quanto no processo, priorizando as pessoas sobre os processos e proporcionando à equipe a liberdade de definir seu próprio modo de trabalho [13].

Dentre os métodos ágeis, um que se destaca, é o Scrum. Esse será o método utilizado nesse trabalho. O Scrum foca na gestão de processos empíricos, com o intuito de entregar valor ao negócio no menor tempo possível [13]. Uma das principais características deste método são as sprints que consiste em um período delimitado durante o qual uma equipe trabalha para completar um conjunto específico de tarefas, visando entregar uma nova versão incremental do produto. Uma das grandes vantagens desse método é que ele possibilita que todos os membros do projeto estejam cientes do estado atual do desenvolvimento.[12].

Outro ponto muito explorado no desenvolvimento ágil, são as histórias de usuário. Essas histórias são constituídas de requisitos de funcionalidades que precisam ser desenvolvidas em um software. A utilização deste recurso facilita a compreensão do que precisa ser feito. Além de possibilitar que testes sejam feitos para encontrar possíveis falhas no projeto [3].

Durante o desenvolvimento de um software, é de suma importância que a equipe de desenvolvimento leve em consideração como os usuários irão interagir com o software. Independentemente de quantos recursos estejam disponíveis, eles podem acabar se tornando inúteis caso o usuário não possua domínio ou conhecimento de como utilizá-los [17].

A partir disso, surgiu um conceito essencial na interação humano-computador, a usabilidade. Ela é muitas vezes descrita como a qualidade que determina o quão simples e rápido é para uma pessoa aprender a utilizar um sistema [9]. Jakob Nielsen, nos anos 90, desenvolveu um método para avaliar a eficácia da usabilidade de um sistema. Esse método envolve a aplicação de um teste baseado em 10 heurísticas, as quais são eficientes para detectar falhas na interface de um sistema [10].

Apesar da grande importância da usabilidade, ela não é capaz de ser totalmente precisa com todas as preocupações no desenvolvimento. Por isso, surgiu o conceito de experiência do usuário (UX) [7], que engloba a percepção geral e as reações emocionais dos usuários ao interagir com um produto [6].

Em vista do exposto, esse trabalho tem como objetivo desenvolver uma ferramenta que a partir de palavras chaves, crie histórias do usuário voltadas para requisitos de usabilidade, levando em consideração as heurísticas de Nielsen e os principais conceitos da Experiência do Usuário. Desta forma, essa ferramenta pretende facilitar o trabalho dos desenvolvedores, fornecendo informações e orientações que contribuam para a criação de produtos digitais mais intuitivos e agradáveis para os usuários finais.

2. Fundamentação Teórico-Metodológica e Estado da Arte

2.1. Métodos ágeis

O grande sucesso obtido pelos processos de manufatura da indústria, tornaram-se uma referência para diversos setores produtivos, um deles foi a Engenharia de Software. Esse campo da Tecnologia da Informação que foi desenvolvido a partir da segunda metade do século XX, foi fortemente influenciado pelos modelos de processos desenvolvidos pela indústria. Em especial a indústria automobilística que vivia um momento de grande expansão, graças ao modelo de produção em série criado por *Henry Ford*. O seu modelo influenciou fortemente a criação de padrões de componentes e processos na Engenharia de Software[14].

Esses métodos tradicionais eram considerados muito burocráticos e rigorosos, na década de 90 surgem os conceitos de desenvolvimento ágil, e em 2001 um grupo de especialistas em processos de desenvolvimento de software se reuniram com o intuito de formular maneiras para melhorar o desempenho em seus projetos. A partir disso surgiu o manifesto ágil que é constituído por um conjunto princípios e alguns valores que fundamentam o desenvolvimento ágil [2, 14].

O manifesto ágil, enfatiza a importância da interação humana, a entrega de um software funcional, a colaboração contínua com o cliente e a capacidade de responder a mudanças. Esses valores visam promover um desenvolvimento de software mais eficiente, flexível e centrado nas necessidades reais dos usuários e clientes, sendo eles [2, 14]:

- **Indivíduos e Interação mais que processos e ferramentas** é extremamente necessário valorizar o lado humano, os softwares são desenvolvidos por um grupo de pessoas, e é de suma importância que haja comunicação entre elas para que o trabalho ocorra de forma mais eficiente, apesar dos processos e ferramentas serem muito importantes não se pode excluir a interação entre as pessoas.
- **Software em funcionamento mais que documentação abrangente** não se pode negar a importância da documentação de um projeto, no entanto a entrega de um sistema funcional é melhor do que a de uma documentação extensa e com muitos detalhes desnecessários.
- **Colaboração com o cliente mais que negociação de contratos** os projetos costumam mudar ao longo de seu desenvolvimento, ver o software funcionando deixa muito mais nítida a percepção de funcionalidades que não foram acordadas no início do projeto, apesar de um contrato firmado ser indispensável para proteger tanto o cliente quanto os responsáveis pelo desenvolvimento não se pode excluir a comunicação com o cliente.
- **Responder as mudanças mais que seguir um plano** a flexibilidade no plano de execução de um projeto é algo indispensável, apesar da necessidade seguir o plano inicial, mudanças quase sempre são uma realidade as ferramentas, leis, ideais são atualizadas e isso impacta diretamente na realização de um projeto.

2.1.1. Scrum

O *Scrum* é um método ágil de desenvolvimento de software, idealizado por Jeff Sutherland e sua equipe no início dos anos 1990. Este método popular foca na gestão de processos empíricos, visando entregar valor ao negócio no menor tempo possível [13, 16].

Esse método permite que todas as pessoas envolvidas saibam como está o desenvolvimento do projeto e com isso facilita nas decisões necessárias para alterações que sejam importantes para alcançar o objetivo final. Ele não é capaz de resolver ou apontar todos os problemas que estejam ocorrendo, mas sombra de dúvidas ele torna muito mais fácil identificar esses gargalos [12].

A organização de uma equipe de Scrum é algo bastante importante. É necessário que a equipe se mantenha pequena o suficiente para garantir que o processo seja ágil, ao mesmo tempo que é de suma importância que seja grande o suficiente para que tudo seja finalizado antes do *Sprint* final. Essas equipes são formadas por três categorias sendo elas o *Scrum Master*, o *Product Owner* e os *Desenvolvedores*, elas são autogerenciáveis e cada uma dessas equipes tem como responsabilidade agregar valor e algum incremento útil a cada *Sprint*. O papel de cada uma das categorias consistem em [16]:

- O *Scrum Master* é em sua essência o responsável pela eficiência da equipe, ele atua auxiliando os participantes a entenderem o Scrum tanto na prática como na teoria, isso faz com que todos consigam entender e melhorar nas suas atribuições. Ele também ajuda o time a focar nas atividades que precisam ser realizadas, remove empecilhos que estejam atrapalhando o avanço da equipe e garante que tudo ocorra bem dentro do tempo determinado [16, 15].
- O *Product Owner* é o responsável pelo valor final do produto ele precisa entender os requisitos dos projetos, e a partir disso ele cria um Backlog que é uma lista dos itens que precisam ser desenvolvidos [16]. Com essa lista ele busca entender as necessidades dos interessados e define as prioridades da ordem em que os itens precisam ser concluídos. Além disso ele atua como um intermediário ligando a equipe de desenvolvimento ao cliente [16, 1].
- Os *Desenvolvedores* tem como objetivo implementar os itens do Project Backlog, os itens concluídos são apresentados no Sprint Backlog [8]. Essa equipe é multifuncional e autogerenciável, não existe divisão entre membros e áreas, todos trabalham juntos e são responsáveis por finalizar cada atividade antes de cada sprint. Outro ponto muito importante é que essa equipe não é alterada durante o desenvolvimento, isso garante que todo mundo saiba o que precisa ser feito desde o início e que todos estejam cientes de tudo que está ocorrendo [1].

O Scrum também é composto por três artefatos que tem como objetivo tornar mais fácil a compreensão do andamento do projeto como pode ser visto na Figura 1, esses artefatos são divididos em [14]:

- O *Product Backlog* é uma lista de requisitos a serem desenvolvidos no projeto, que pode ser alterada ou reordenada exclusivamente pelo *Product Owner*. As Histórias de Usuário geralmente são o formato mais utilizado para descrever os itens neste artefato.
- O *Sprint Backlog* tem como objetivo tornar mais visível a evolução no desenvolvimento do projeto. Esse artefato consiste em um grupo de itens que devem ser desenvolvidos até o próximo *Sprint* e um plano para que esses itens possam finalmente se tornar um incremento. Um novo Sprint Backlog é criado após cada reunião de Sprint.
- As sprints são períodos intensos e delimitados, geralmente durando entre duas e quatro semanas, em que as equipes de desenvolvimento se concentram em com-

pletar objetivos específicos estabelecidos no início do ciclo. Essa abordagem permite iterar rapidamente e fazer ajustes em tempo real, promovendo uma entrega contínua de valor [13]. Durante uma sprint, são realizadas reuniões diárias, conhecidas como "Daily Scrums", que são essenciais para o sucesso do processo. Essas reuniões breves permitem que a equipe compartilhe progressos, discuta obstáculos e realinhe atividades conforme necessário, garantindo que todos estejam alinhados e focados nos objetivos da sprint.

- Por fim o *Incremento do produto* consiste na entrega de um novo incremento que foi desenvolvido durante a etapa do *Sprint*. Essa etapa é extremamente importante pois aqui o *Product Owner* consegue efetivamente ver o projeto avançando e assim ter uma visão melhor para possíveis melhorias.

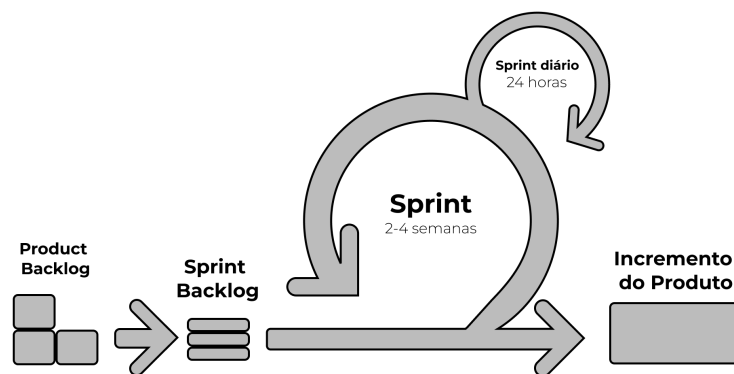


Figura 1. Ciclo de desenvolvimento do SCRUM

2.2. Histórias de Usuário

As histórias de usuário consistem em requisitos de funcionalidades que precisam ser desenvolvidos em um software. Essas histórias são amplamente utilizadas no desenvolvimento ágil. Segundo [3] elas são compostas por três aspectos principais:

- Descrição escrita: Utilizada para planejar e também servir como lembrete.
- Conversas: Discussões sobre a história que ajudam a detalhá-la.
- Testes: Documentam os detalhes e podem ser usados para verificar se a história está completa.

Esses três aspectos são frequentemente chamados de "Cartão, Conversa e Confirmação", refletindo a essência das histórias de usuário no contexto ágil.

Geralmente, as histórias de usuário são criadas utilizando o template ilustrado na Figura 2. Esse modelo é extremamente benéfico, pois permite identificar a classe do usuário mencionada na história e visualizar com mais clareza a justificativa para a ação desejada. Além de fornecer uma descrição concisa do objetivo do usuário, o template ajuda a contextualizar a necessidade, garantindo que todos os membros da equipe compreendam o propósito e o valor da funcionalidade solicitada [19].

As histórias de usuário geralmente são acompanhadas de critérios de aceitação, que servem para definir claramente as especificações e os limites da implementação [4]. Na Figura 3 podemos ver o template criado em 2006, por *Dan North* e *Chris Matts*

COMO UM <Tipo de Usuário>,
EU QUERO <Algum objetivo>,
PARA QUE <Alguma razão>.

Figura 2. Modelo para criação de história de usuário. Adaptado de [19]

para aprimorar a compreensão das histórias de usuário e facilitar a criação de testes de aceitação. Este template é composto por três seções principais [11]:

- **Dado que:** Especifica as condições que devem ser atendidas antes da execução dos testes.
- **Quando:** Descreve o comportamento esperado do sistema sob as condições dadas.
- **Então:** Define as mudanças que devem ocorrer com base no comportamento especificado.

DADO QUE <Uma condição>,
QUANDO <Uma ação>,
ENTÃO <Uma consequência desejada>.

Figura 3. Modelo para criação de critério de aceitação. adaptado de [11]

Esse template ajuda a garantir que a implementação atenda aos requisitos e possa ser testada de forma eficaz [4].

Para verificar a qualidade da história de usuário, foi criado um padrão com 6 fatores conhecidos como INVEST. Eles servem como orientação para quem está escrevendo essas histórias, e estão divididos em [3]:

- **Independência:** As histórias devem ser criadas com cuidado para evitar dependências entre elas. Dependências podem causar problemas de priorização e planejamento, dificultando o progresso do projeto.
- **Negociável:** As histórias de usuário não são contratos rígidos; são descrições concisas das funcionalidades. Os detalhes devem ser discutidos e ajustados entre desenvolvedores e clientes conforme necessário.
- **Valiosa para clientes ou usuários:** É essencial distinguir entre o cliente e o usuário ao criar histórias. Às vezes, uma história pode não parecer relevante para o usuário final, mas pode ser crucial para o cliente. A história deve oferecer valor significativo para o cliente ou usuário final.
- **Estimável:** Os desenvolvedores devem ser capazes de estimar o esforço e o tempo necessários para implementar uma história. Se uma história não pode ser estimada, geralmente é devido a falta de conhecimento sobre o domínio, falta de conhecimento técnico ou porque a história é muito ampla.

- Pequena: A história deve ter um tamanho adequado à capacidade da equipe e ao processo de desenvolvimento. Histórias muito grandes ou muito pequenas podem ser difíceis de planejar e executar eficientemente.
- Testável: A história deve ser passível de teste. Se todos os testes forem concluídos com sucesso, isso indica que a história está correta e pode ser desenvolvida conforme o esperado.

Esses atributos garantem que as histórias de usuário sejam práticas e eficazes, facilitando o desenvolvimento ágil e a entrega de valor ao cliente.

2.3. Interação Humano Computador/ Usabilidade

Os computadores são ferramentas amplamente difundidas na sociedade, após a queda do custo de computador pessoal essas máquinas se tornaram acessíveis, o que antes era algo restrito a um pequeno grupo de pessoas e de uso para algumas tarefas específicas, se tornou algo comum para diversas aplicações diferentes [9]. Dado esse vasto espectro de usuários e usos, é indispensável que um computador seja de fácil uso. A maioria das pessoas não tem interesse em compreender a fundo o funcionamento de um computador, portanto, aprimorar esses aspectos da interface física aumenta significativamente as chances de sucesso do produto no mercado [18].

A forma como os computadores podem ser utilizados para alavancar o desempenho das pessoas tanto no trabalho quanto na vida pessoal, se tornaram foco de diversos pesquisadores. Por volta da década de 1980 esse novo campo foi nomeado de Interação Humano-Computador (IHC) [18].

A IHC pode ser descrita como uma disciplina preocupada com o design, avaliação e implementação de sistemas de computação interativos para uso humano e com estudo dos principais fenômenos que os cercam [17, 18].

Independentemente dos recursos que uma máquina possa oferecer, ela pode se tornar inútil ou ineficaz se o usuário não tiver domínio ou conhecimento sobre suas funcionalidades. Sem esse conhecimento, os recursos disponíveis podem não ser utilizados ou, se forem, é provável que o sejam de forma incorreta. Nesse contexto, surge um conceito essencial na Interação Humano-Computador: a *Usabilidade* [17].

A usabilidade pode ser descrita como a qualidade que determina o quão simples e rápido é para uma pessoa aprender a utilizar um sistema. O objetivo da usabilidade é garantir que o aprendizado do uso de um sistema seja fácil, permitindo que o usuário inicie rapidamente o trabalho a ser realizado. Além disso, o sistema deve ser eficiente para proporcionar alta produtividade, facilitar a memorização de como manuseá-lo, minimizar erros e, caso ocorram, assegurar uma possível recuperação das informações perdidas. Por fim, a usabilidade busca garantir que os usuários fiquem satisfeitos ao utilizar o sistema [9].

2.3.1. Experiência do Usuário - UX

A partir da grande expansão do mercado, alta competitividade entre os produtos e consumidores mais conscientes, a usabilidade apesar de essencial não expressa totalmente as preocupações, uma vez que o foco é dos projetos desenvolvidos são as pessoas, ao invés

de serem centrados na tecnologia, isso deu origem a um novo termo a experiência de usuário (UX) [7].

A *Organização Internacional de Normalização (ISO)*, define a experiência de usuário como sendo as percepções e respostas de um usuário que são resultados da utilização de um sistema. Segundo o ISO 9241-210, as percepções e respostas incluem as emoções, crenças, preferências, percepções, conforto, comportamentos e realizações que surgem antes durante ou após a utilização. A experiência do usuário também é influenciada pela apresentação, funcionalidades do sistema, o comportamento interativo e as suas capacidades assistivas [5].

De acordo com [6] a experiência do usuário refere-se à percepção geral que um produto cria para as pessoas que o utilizam. Em vez de se concentrar apenas no funcionamento interno do produto, a UX foca em como o produto interage com o usuário e como essa interação afeta a experiência geral do usuário.

Garret desenvolveu uma estrutura que pode ser vista na Figura 4, ela tem como finalidade auxiliar no desenvolvimento de projetos, para que eles consigam obter com sucesso uma melhor experiência de usuário. Essa estrutura é dividida em cinco camadas, onde cada camada foca em solucionar os problemas abordados por ela. Dessa forma, os problemas são tratados de forma isolada e com isso é possível trabalhar de uma forma menos abstrata. Conforme o projeto vai progredindo entre as camadas, as decisões vão ficando mais específicas e os detalhes são aperfeiçoados [6].

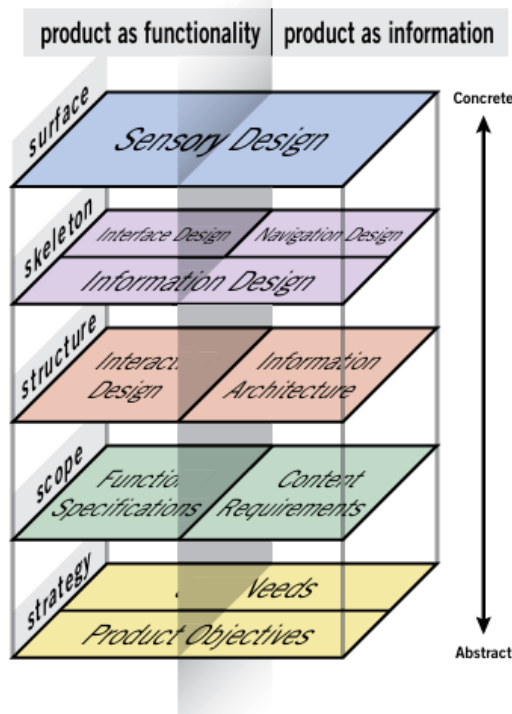


Figura 4. Elementos UX [6]

Essa estrutura é construída de baixo para cima, onde cada camada acima depende das camadas anteriores. Caso essa ordem não seja seguida é possível que os projetos não sejam bem-sucedidos. As camadas estão divididas em [6]:

- **Estratégia:** Um dos principais motivos para a falha no desenvolvimento de um projeto, está no mal planejamento do que se deseja obter ao final do projeto. A camada de estratégia tem como finalidade encontrar a resposta para duas questões principais, o que o usuário espera do produto e quais são os objetivos finais do produto. Com essas duas perguntas respondidas as próximas etapas compreender melhor o que precisa ser feito.
- **Escopo:** Com base nas conclusões obtidas na camada anterior, a camada de escopo aborda dois tópicos principais: as especificações funcionais e os requisitos de conteúdo. As especificações funcionais detalham as funcionalidades que o sistema deve incluir para atender às necessidades dos usuários, enquanto os requisitos de conteúdo definem os elementos de conteúdo necessárias para o funcionamento do sistema.
- **Estrutura:** Na terceira camada, o foco se desloca das questões mais abstratas de escopo e estratégia para aspectos mais concretos e próximos da experiência do usuário. Nesta camada, abordam-se o design de interação e a arquitetura da informação. O design de interação visa criar fluxos de sistema que simplifiquem a execução de tarefas pelos usuários. Já a arquitetura da informação se concentra em estruturar o espaço das informações de forma a garantir um acesso mais intuitivo ao conteúdo.
- **Esqueleto:** A camada anterior define o funcionamento do sistema. Já a camada de esqueleto visa definir a forma que essas funcionalidades iram assumir, essa camada também é responsável por aperfeiçoar os detalhes. Ela é dividida em três partes o design de interface, o design de navegação e o design de informação.
- **Superfície:** Por fim a camada do topo da estrutura herda todo o trabalho realizado nas camadas anteriores. O foco dessa última camada é trabalhar o design sensorial, esse trabalho consiste em organizar os arranjos que compõe o esqueleto do projeto.

2.3.2. Heurísticas de Nielsen

Nos anos 90, *Jakob Nielsen* e *Rolf Molich* formularam juntos um conjunto de heurísticas para avaliação de interfaces de usuário que verificavam a quão boa era a usabilidade de um sistema [10]. Alguns anos após isso *Nielsen* aprimorou essas heurísticas, resultando em suas 10 heurísticas, elas são utilizadas para identificar problemas na usabilidade das interfaces de usuários. As heurísticas de *Nielsen* são divididas em [10, 9]:

1. **Visibilidade do status do sistema:** a interface do sistema deve sempre garantir que o usuário tenha ciência de tudo que está acontecendo no momento, idealmente esse feedback deve ser exibido no menor tempo possível. Essa rápida resposta auxilia os usuários a definirem suas próximas ações dentro do sistema;
2. **Correspondência entre o sistema e o mundo real:** é crucial que os usuários de um determinado sistema consigam compreender de forma simples e direta as informações disponibilizadas para ele. Dessa forma os sistemas precisam desenvolvidos usando palavras, símbolos e conceitos que são familiares ao público-alvo. Quando esse design segue essas convenções se torna muito mais fácil e intuitiva a utilização do sistema e a memorização de como utilizá-lo;
3. **Controle e liberdade do usuário:** é muito comum que as pessoas executem uma ação sem a intenção de executá-la ou até mesmo executem uma ação e logo em

seguida mudem de ideia. Por isso, é necessário que os sistemas disponibilizem uma forma para que o usuário consiga reverter os resultados de suas ações. Quanto mais simples e fácil forem as formas de desfazer uma ação maior será a sensação de liberdade para o usuário;

4. **Consistência e padrões:** garantir que o sistema esteja de acordo com as convenções que são utilizadas na plataforma ou no setor em que o sistema se encaixa, torna muito mais agradável à experiência para o usuário. Quando o sistema foge dessas convenções o usuário é forçado a aprender algo novo, enquanto se o sistema utilizasse tais convenções, o usuário conseguiria intuitivamente com base em suas experiências anteriores com outros sistemas ser mais direto para alcançar seu objetivo, ao invés de ter que gastar um tempo e esforço para aprender algo novo;
5. **Prevenções de erros:** apesar de serem importantes boas mensagens de erro não são a melhor forma de tratar problemas. É necessário que o sistema evite ao máximo que os erros ocorram, garantir que não existam condições que possibilitem erros e exibir ações para confirmação antes de um usuário efetivamente realizar uma ação é indispensável;
6. **Reconhecimento em vez de lembrança:** forçar o usuário a lembrar de informações vistas anteriormente em ações futuras, faz com que os usuários precisem fazer muito mais esforço cognitivo enquanto utilizam o sistema, isso faz com que a experiência de usuário não seja tão agradável. Dessa forma, o sistema deve permitir que os usuários tenham acesso as informações na interface, ao invés de obrigar o usuário decorar informações;
7. **Flexibilidade e eficiência de uso:** a flexibilização na forma como os processos podem ser realizados, permite que os usuários consigam adaptar o sistema a uma forma de utilização que funciona melhor para elas. Um exemplo disso são os atalhos ocultos, enquanto usuários mais experientes potencializam o seu desempenho utilizando esses atalhos, os usuários inexperientes acabam tendo o seu aprendizado prejudicado. Dessa forma, a flexibilidade traz inúmeros benefícios para os diversos tipos de usuários;
8. **Design estético e minimalista:** o foco da interface deve ser garantir que o conteúdo que o usuário necessita, seja exibido de forma clara. Sendo assim, é importante que informações irrelevantes ou que não sejam muito utilizadas não recebam tanto destaque quanto as informações que realmente precisam ser visualizadas. Isso garante que o usuário consiga focar e ser mais direto no que ele realmente precisa;
9. **Ajude os usuários a reconhecer, diagnosticar e recuperar erros:** os erros precisam ser facilmente compreendidos pelos usuários. Isso cria a necessidade de que as mensagens de erro sejam escritas de forma simples e objetiva. Além disso, é de suma importância que sugerir uma forma de como solucionar tais erros;
10. **Ajuda e documentação:** apesar de ser preferível que o sistema seja de fácil compreensão e não seja necessário instruções de como utilizá-lo. Fornecer uma documentação clara que auxilie os usuários a entenderem as funcionalidades e a utilização de sistema, pode ser necessário. Também, é importante garantir que a documentação seja facilmente encontrada e que ela seja de fácil compreensão;

3. Objetivos

O objetivo principal deste trabalho é desenvolver uma ferramenta que, a partir de informações detalhadas fornecidas pelo usuário, gere histórias de usuário. Essas histórias são projetadas para melhorar a experiência dos usuários e são especificamente adaptadas para o método ágil Scrum. Este estudo é baseado em metas específicas que definem a direção e o escopo da pesquisa:

- Desenvolver um aplicativo capaz de receber uma entrada do usuário e, com base nessa entrada, criar histórias de usuário detalhadas e voltadas para UX.
- Gerar métricas a partir das heurísticas de Nielsen para ajudar os desenvolvedores a validar suas aplicações com base nas histórias de usuário criadas.
- Produzir um documento contendo instruções para auxiliar no desenvolvimento, a partir das histórias do usuário.

4. Procedimentos metodológicos/Métodos e técnicas

Durante a fase inicial do trabalho, será realizada uma revisão bibliográfica com o objetivo de construir uma base sólida de conhecimento sobre métodos ágeis, em particular o Scrum. Essa revisão incluirá a compreensão das melhores práticas para a criação de histórias de usuário, bem como a análise das técnicas mais eficazes para aprimorar a usabilidade e a experiência do usuário.

Após a conclusão da revisão bibliográfica, iniciará a fase de planejamento, que definirá como a ferramenta será desenvolvida. Esse planejamento abrangerá o desenvolvimento da base de dados e a forma como ela será utilizada para criar as histórias de usuário e gerar as métricas.

Com a base de dados concluída, o desenvolvimento da ferramenta terá início. Uma vez finalizado o desenvolvimento, a aplicação será submetida a testes rigorosos. Finalmente, um relatório detalhado sobre o desempenho da ferramenta será elaborado.

5. Cronograma de Execução

1. Revisão bibliográfica;
2. Planejamento de desenvolvimento;
3. Trabalhar base de dados;
4. Escrita da versão preliminar do TCC;
5. Desenvolvimento da ferramenta;
6. Testes e elaboração de relatório do desempenho;
7. Escrita versão final do TCC;
8. Revisão ortográfica;

6. Contribuições e/ou Resultados esperados


Ao fim do projeto, espera-se obter uma ferramenta robusta e eficiente que simplifique significativamente o trabalho dos desenvolvedores. A ferramenta deverá criar histórias de usuário que sejam claras e de fácil compreensão, além de fornecer orientações práticas e detalhadas que contribuam para o desenvolvimento de projetos mais intuitivos, agradáveis e eficazes para os usuários finais.

Tabela 1. Cronograma de Execução

	jul	ago	set	out	nov	dez	jan	fev
Atividade 1	x	x						
Atividade 2		x						
Atividade 3		x	x					
Atividade 4			x	x				
Atividade 5				x	x			
Atividade 6					x	x		
Atividade 7				x	x	x	x	
Atividade 8							x	x

7. Espaço para assinaturas

Londrina, 29/07/2024.



Aluno



Orientador

Referências

- [1] Dairton Luiz Bassi Filho. Experiências com desenvolvimento ágil. *São Paulo*, 2008.
- [2] Kent Beck, W Cunningham, A Hunt, R Martin, D Thomas, M Beedle, and J Sutherland. Manifesto ágil. manifesto para desenvolvimento ágil de software, 2001.
- [3] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
- [4] António MS Ferreira, Alberto Rodrigues da Silva, and Ana CR Paiva. Towards the art of writing agile requirements with user stories, acceptance criteria, and related constructs. In *ENASE*, pages 477–484, 2022.
- [5] International Organization for Standardization. Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems. Standard, International Organization for Standardization, Geneva, CH, July 2019.
- [6] Jessie James Garrett. *The elements of user experience*. New Riders, 2010.
- [7] Rex Hartson and Pardha S Pyla. *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier, 2012.
- [8] Marcos Machado and Sérgio Gustavo Medina. Scrum–método ágil: uma mudança cultural na gestão de projetos de desenvolvimento de software. *Revista Científica Intraciência, Faculdade do Guarujá–UNIEESP*, 1(1):58–71, 2009.
- [9] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

- [10] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.
- [11] Dan North. Introducing bdd, better software magazine. *Better Software Magazine*. [Online]. Available: <http://www.stickyminds.com/BetterSoftware/magazine.asp>, 2006.
- [12] Paulo Pereira, Paula Torreão, and Ana Sofia Marçal. Entendendo scrum para gerenciar projetos de forma ágil. *Mundo PM*, 1(14):64–71, 2007.
- [13] Roger S Pressman and Bruce R Maxim. *Engenharia de software-9*. McGraw Hill Brasil, 2021.
- [14] Rafael Prikladnicki, Renato Willi, and Fabiano Milani. *Métodos ágeis para desenvolvimento de software*. Bookman Editora, 2014.
- [15] JHTC Sbrocco and Paulo Cesar de MACEDO. Metodologias ágeis: engenharia de software sob medida. *São Paulo: Érica*, 8:9, 2012.
- [16] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 21(1):1–38, 2011.
- [17] Gaurav Sinha, Rahul Shahi, and Mani Shankar. Human computer interaction. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 1–4. IEEE, 2010.
- [18] HCR Vieira and Maria Cecília C Baranauskas. Design e avaliação de interfaces humano-computador. *Campinas: Unicamp*, 2003.
- [19] Karl E Wieggers and Joy Beatty. *Software requirements*. Pearson Education, 2013.