UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
CIÊNCIA DA COMPUTAÇÃO

MATHEUS VIEIRA DE SOUZA

CHECKLIST DE AVALIAÇÃO HEURÍSTICA DE USABILIDADE PARA COMPARAR OS FRAMEWORKS BOOTSTRAP E TAILWIND EM INTERFACES GRÁFICAS DE E-COMMERCE

CAMPO GRANDE - MS
2025

MATHEUS VIEIRA DE SOUZA

CHECKLIST DE AVALIAÇÃO HEURÍSTICA DE USABILIDADE PARA COMPARAR OS FRAMEWORKS BOOTSTRAP E TAILWIND EM INTERFACES GRÁFICAS DE E-COMMERCE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Federal de Mato Grosso do Sul.
Orientador: Prof. Dr. Ricardo Theis Geraldi

CAMPO GRANDE - MS
2025

AGRADECIMENTOS

Primeiramente, agradeço a minha família, por ter me abençoado, ter tido esperanças em mim durante essa jornada desafiadora de muitas felicidades e muitos atritos. Sou muito grato por ter pessoas amáveis e compreensíveis, que de alguma forma me ajudaram a finalizar este trabalho.

Um agradecimento especial:

A minha mãe Lourdes de Jesus Vieira e meu pai, José Venâncio de Souza Filho, graças a eles eu não estaria no lugar onde estou. Quaisquer palavras de gratidão jamais serão mensuradas pelo que fizeram por mim. Muito obrigado por serem minha inspiração e fortaleza emocional principalmente nos meus momentos de fraqueza, serei eternamente grato por ter vocês dois comigo em minha vida.

Ao meu orientador Ricardo Theis Geraldi, por ter me guiado com este trabalho e por ter tido muita paciência comigo durante esse processo. Obrigado por acreditar em mim, ter me ajudado a ser um estudante mais crítico e por compartilhar seu conhecimento em todos os momentos.

Aos meus amigos mais íntimos que também sempre me apoiaram e me encorajaram durante toda essa caminhada, obrigado a todos pela confiança e suporte.

RESUMO

A usabilidade é um dos principais fatores que influenciam na experiência do usuário, no qual pode gerar maior taxa de conversão devido a facilidade de navegação e agilidade no processo de compras. Esta é uma das razões do porquê e-commerces que têm interfaces gráficas amigáveis, claras e responsivas, isto faz com que clientes tenham uma experiência satisfatória, reduzindo a taxa de abandono e criando uma confiança no sistema.

Neste cenário, a escolha de um *framework CSS* é uma decisão fundamental no desenvolvimento de interfaces gráficas para *e-commerce*, impactando na usabilidade e na experiência do usuário final. Diante da crescente popularidade de abordagens como a baseada em componentes (*component-first*) do *Bootstrap* e a de classes utilitárias (*utility-first*) do *Tailwind*, surge o problema de compreender como diferentes caminhos de desenvolvimento afetam a usabilidade de sistemas de software com as boas práticas de programação e de decidir qual das duas é mais adequada para aplicações robustas como *e-commerces*.

Este trabalho tem como objetivo principal comparar a usabilidade dos *frameworks* CSS *Bootstrap* e *Tailwind* por meio de avaliações heurísticas nas interfaces gráficas de e-commerce, a fim de determinar as suas principais características das interfaces gráficas utilizadas. Para atingir este objetivo, foi adotado um método de pesquisa próximo a uma análise de conteúdo qualitativo, caracterizado como um caso de estudo comparativo entre *frameworks* CSS, que emprega a técnica de avaliação heurística via *checklist*. Esta inspeção foi guiada por um *checklist* adaptado das heurísticas de usabilidade de Nielsen para o contexto de *e-commerce* conforme Kumar (2022), aplicado a dois protótipos funcionais *open-source* que representam cada *framework*.

O principal resultado obtido foi a discussão sobre os dois *checklists* gerados pela avaliação heurística de usabilidade. Durante o processo de avaliação heurística o *Bootstrap* apresentou mais itens/heurísticas implementadas no *checklist* do que o *Tailwind*, porém ambos *frameworks* são viáveis para desenvolvimento de interfaces gráficas para *e-commerce*. O *Bootstrap* sendo ideal para projetos padronizados com prazos curtos e o *Tailwind* sendo mais adequado para projetos onde os desenvolvedores são mais experientes, pois terão mais facilidade em implementar designs personalizados e otimizados.

Palavras-chave: Usabilidade. Avaliação Heurística. E-commerce. Framework CSS. Bootstrap. Tailwind.

ABSTRACT

Usability is one of the main factors that influence user experience, as it can lead to higher conversion rates due to easier navigation and faster purchasing processes. This is one of the reasons why e-commerce platforms that have friendly, clear, and responsive graphical interfaces provide users with a satisfactory experience, reducing abandonment rates and building trust in the system.

In this context, choosing a CSS framework is a key decision in the development of graphical interfaces for e-commerce, as it impacts usability and the final user experience. Given the growing popularity of approaches such as the component-first model of Bootstrap and the utility-first model of Tailwind, the problem arises of understanding how these different development paths affect software usability based on programming best practices, and of deciding which one is more suitable for robust applications such as e-commerce systems.

The main objective of this work is to compare the usability of the CSS frameworks Bootstrap and Tailwind through heuristic evaluations of e-commerce graphical interfaces, in order to determine their main interface characteristics. To achieve this objective, a research method similar to a qualitative content analysis was adopted, characterized as a comparative case study between CSS frameworks, employing heuristic evaluation using a *checklist*. This inspection was guided by a *checklist* adapted from Nielsen's usability heuristics for the e-commerce context, according to Kumar (2022), applied to two open-source functional prototypes representing each framework.

The main result obtained was the discussion of the two *checklists* generated through the heuristic usability evaluation. During the evaluation process, Bootstrap presented more implemented *checklist* items/heuristics than Tailwind; however, both frameworks are viable for the development of graphical interfaces for e-commerce. Bootstrap proves ideal for standardized projects with short deadlines, while Tailwind is more suitable for projects with more experienced developers, as they will find it easier to implement customized and optimized designs.

Keywords: Usability. Heuristic Evaluation. E-commerce. CSS Framework. Bootstrap. Tailwind. Heuristics.

SUMÁRIO

1. Introdução

1.1 Descrição do trabalho

A qualidade do software, especialmente em plataformas de *e-commerce*, ultrapassou o status de diferencial técnico para se tornar um pilar central da estratégia de negócios. Nesse contexto altamente competitivo, onde o consumidor dispõe de inúmeras alternativas de lojas virtuais, a usabilidade emerge como um dos principais atributos de qualidade, impactando diretamente não apenas a experiência do usuário, mas a própria viabilidade econômica da plataforma (GOMES e CORTÊS, 2022).

Estudar a usabilidade em *e-commerce* é fundamental porque a jornada de compra online é intrinsecamente frágil. Problemas que podem parecer simples, como um botão de "*checkout*" mal posicionado, a falta de feedback visual ao adicionar um item ao carrinho, ou uma navegação confusa entre categorias de produtos, geram insatisfações que se acumulam rapidamente (MORAES, 2017). Essas insatisfações levam à frustração, à desconfiança, a erros operacionais e, no cenário mais crítico e comum, ao abandono do carrinho de compras, que é uma das métricas mais temidas pelo setor. Portanto, a usabilidade não é somente uma funcionalidade do sistema, ela gera impacto direto em taxas de conversão, satisfação e fidelização de clientes (PEREIRA, 2013).

Nesse cenário, a adoção de *checklists* padronizados para inspeções de usabilidade, adaptados ao contexto específico de *e-commerce*, torna-se uma ferramenta estratégica. Eles permitem a verificação rápida e sistemática de itens essenciais, ou heurísticas, e padronizam a identificação e descrição de defeitos, garantindo que a interface atenda a um padrão mínimo de qualidade durante a interação do usuário (SANTOS, 2011).

Paralelamente, *frameworks* CSS como *Bootstrap* e *Tailwind*, que são os mais utilizados atualmente no mercado, são essenciais para a construção de uma interface que satisfaça requisitos que um *e-commerce* exige. Eles têm um impacto profundo e direto na aparência, responsividade, consistência, e consequentemente na facilidade de uso das interfaces gráficas. No entanto, a discussão frequentemente se limita à perspectiva do desenvolvedor, focado em velocidade de codificação e manutenção do código, sem analisar como as filosofias distintas de cada ferramenta impactam na experiência do usuário final.

A compreensão das vantagens e desvantagens práticas de cada *framework* é, portanto, essencial para que desenvolvedores e empresas possam tomar decisões informadas, alinhando a escolha tecnológica com os objetivos de negócio. Assim, este trabalho busca explorar essa

lacuna, oferecendo análises que apoiem a escolha e o uso adequado desses *frameworks*, visando melhorar a usabilidade de sistemas de *e-commerce*.

1.2 Objetivo geral

Avaliar os *frameworks* CSS *Bootstrap* e *Tailwind* por meio de uma avaliação heurística baseada em *checklist*, no contexto de plataformas de *e-commerce*, a fim de comparar suas características de usabilidade.

1.2.1 Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Realizar uma revisão de literatura sobre os conceitos de usabilidade, as heurísticas de Nielsen, a inspeção de software, a avaliação heurística por *checklist*, *frameworks*, CSS e as características específicas dos *frameworks Bootstrap* e *Tailwind*.
2. Adotar protótipos *open-source* de páginas de *e-commerce* (página inicial, página de catálogo e página de *checkout*) por meio de projetos *open-source* que utilizam os *frameworks Bootstrap* e *Tailwind*, servindo como instrumentos para a avaliação.
3. Documentar o escopo e as funcionalidades dos protótipos que serão utilizados, incluindo as tecnologias utilizadas e as partes relevantes para a avaliação heurística.
4. Conduzir a avaliação heurística dos protótipos de *e-commerce* utilizando o *checklist* de usabilidade adaptado com as heurísticas para *e-commerce*, a fim de identificar falhas e observações relevantes.
5. Analisar e comparar os resultados das avaliações dos protótipos de maneira qualitativa, detectando erros de usabilidade e pontuando as razões para a efetividade de cada *framework*. Assim, discutir as diferenças práticas entre os frameworks *Bootstrap* e *Tailwind*, suas vantagens e desvantagens para aplicações de *e-commerce*, e também apresentar sugestões como pontos de melhoria.

1.3 Organização do trabalho

A organização deste trabalho está estruturado da seguinte forma:

Capítulo 1: Apresenta a introdução do trabalho descrevendo o contexto no qual este trabalho está inserido e uma descrição geral do projeto, este capítulo é dividido nos subcapítulos: (1) Descrição do trabalho, apresentando o contexto geral no qual este trabalho está inserido, o problema abordado e uma possível solução; (2) Objetivos, elencando o objetivo geral e objetivos específicos deste trabalho; e (3) Organização do trabalho, com uma explicação geral dos capítulos deste trabalho.

Capítulo 2: apresenta a estruturação desta pesquisa com a descrição detalhada de cada etapa de desenvolvimento.

Capítulo 3: apresenta a revisão da literatura com os tópicos listados neste trabalho, dividida pelos subcapítulos: (1) Usabilidade, dissertando sobre sua definição, importância, contexto no qual começou-se a ser estudado e suas principais características; (2) Heurísticas de Nielsen, abordando as 10 heurísticas criadas por Jakob Nielsen utilizadas para realizar a avaliação da usabilidade de interfaces gráficas web; (3) Inspeção de Software, com definições de Avaliação Heurística e a Tabela de *checklist*; (4) CSS, apresentando o panorama geral deste recurso e também sua arquitetura de funcionamento; (5) *Framework web*, contendo a sua definição, o porquê seu uso é tão importante para o desenvolvimento web moderno e uma abordagem sobre *frameworks* CSS; (6) *Bootstrap*, abrangendo seu contexto de criação, características essenciais e exemplos de aplicações no qual ele é utilizado; (7) *Tailwind*, contexto no qual ele é mais aplicado, suas particularidades e diferenças em relação ao *Bootstrap*.

Capítulo 4: apresenta o escopo da avaliação e a documentação dos protótipos utilizados para avaliação heurística detalhando quais são suas funcionalidades, limitações e tecnologias que foram utilizadas.

Capítulo 5: apresenta a avaliação heurística por meio do *checklist* do protótipo que utiliza o *framework Bootstrap* para estilização das interfaces gráficas, contendo comentários relativas à cada heurística que a tabela têm, e no final da avaliação, é discutido brevemente as observações mais relevantes sobre usabilidade.

Capítulo 6: de forma similar ao capítulo anterior, apresenta a avaliação heurística por meio do *checklist* do protótipo que utiliza o *framework Tailwind* para estilização das interfaces gráficas, contendo comentários relativos à cada heurística que a tabela têm, e no final da avaliação, é discutido brevemente as observações mais relevantes sobre usabilidade.

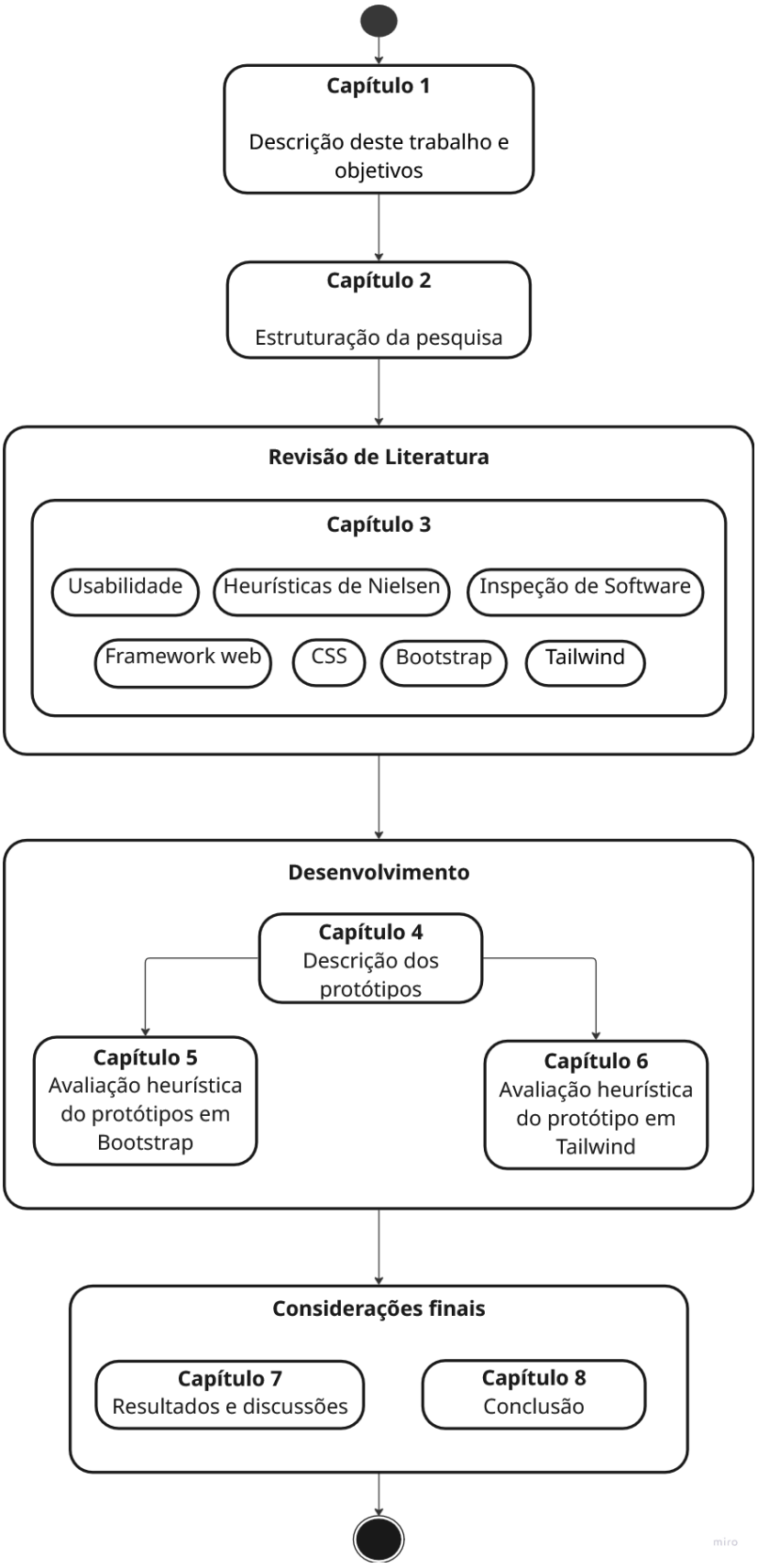
Capítulo 7: apresenta as principais discussões deste trabalho; discute sobre as diferenças práticas dos *frameworks Bootstrap e Tailwind* assim como suas vantagens e desvantagens; alterações feitas nos *checklists*; criação de gráficos para fins de análise; discutir

quais são os melhores casos para se utilizar um *framework* ao invés do outro em aplicações de *e-commerce*; apresenta também comentários, limitações e possíveis melhorias para esta pesquisa.

Capítulo 8: apresenta as principais conclusões deste trabalho.

A Figura 1 ilustra a estruturação deste trabalho:

Figura 1 - Etapas da organização do trabalho



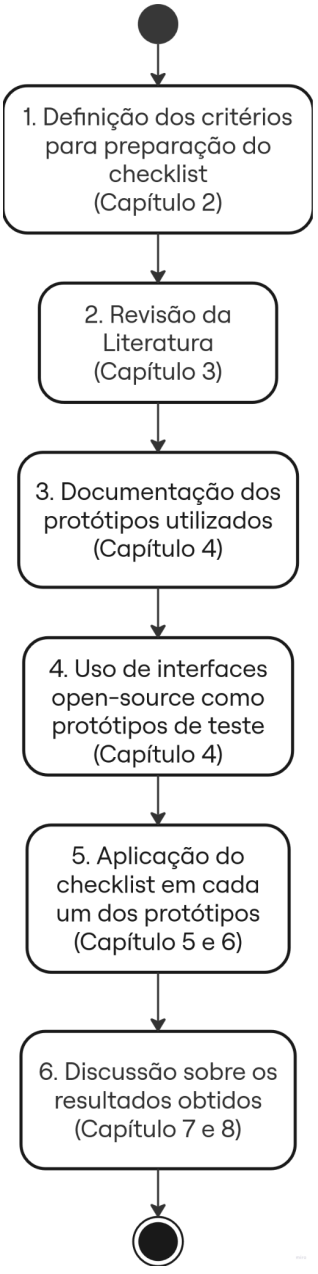
Fonte: Autoria própria (2025)

2. Estruturação da Pesquisa

As etapas do desenvolvimento deste trabalho foram inspiradas pela metodologia de pesquisa qualitativa e descritiva de Sampieri (2017) e adaptações de Creswell (2013). As etapas apresentadas neste capítulo estruturam a condução desta pesquisa. Uma das principais etapas deste trabalho é a avaliação heurística, que tem como objetivo comparar dois *frameworks* CSS, *Bootstrap* e *Tailwind* no contexto de plataformas de *e-commerce*.

A Figura 2 ilustra as etapas desta pesquisa e os próximos parágrafos detalham tais etapas.

Figura 2: Diagrama de atividades referente às etapas da metodologia de pesquisa.



Fonte: Autoria própria (2025).

O trabalho começa com a definição do escopo e dos critérios que serão utilizados nos *checklists* neste mesmo capítulo na Etapa 3 descrita abaixo. Em seguida, é feita a revisão de literatura para a fundamentação teórica dos conceitos, dando sequência com a documentação dos protótipos *open-source* e depois a adoção desses protótipos para realizar as avaliações heurísticas. Após isso, prossegue a execução dos testes de avaliação heurística em cada protótipo por meio dos *checklists*, resultando na análise e discussão dos resultados obtidos para a comparação final dos *frameworks,* juntamente com a conclusão desta pesquisa. A seguir, têm a descrição mais detalhada das etapas realizadas neste trabalho.

Etapa 1: Esta etapa corresponde ao Capítulo 3, no qual foi realizada a revisão de literatura para fundamentar este estudo. Foram abordados os conceitos relacionados à usabilidade, heurísticas de Nielsen, inspeção de software, avaliação heurística por *checklist*, *frameworks web* e as características específicas dos *frameworks Bootstrap* e *Tailwind*. A apresentação desses conceitos serviram de base para o entendimento do *checklist* e posteriormente, para a execução da avaliação heurística e para a criação de protótipos que foram utilizados como instrumento de teste para realizar a avaliação heurística por meio do *checklist*.

Etapa 2: Após a criação dos protótipos, foi estabelecida a definição dos critérios que serão utilizados nas comparações. Serão utilizadas algumas páginas dos protótipos para avaliar fatores como consistência, responsividade, o quão agradável a interface se apresenta, facilidade de uso, curva de aprendizado e prevenção de erros. Para a avaliação heurística, foi utilizado o *checklist* de usabilidade, adaptado para o contexto dos protótipos de *e-commerce*:

- ***Checklist* para *e-commerce*:** Baseado nas heurísticas elaboradas no trabalho de Leavitt e Shneidermans (2007). O *checklist* para *e-commerce* foi desenvolvido por Kumar (2022), com foco em conceitos de design *Mobile-First* e necessidades específicas de *e-commerce*. Assim como foi detalhado no capítulo 3 na seção 3.2, este *checklist* avalia aspectos como navegação em dispositivos móveis, clareza das descrições de produtos, e simplicidade no processo de *checkout*. Este *checklist* de usabilidade tem como prioridade auxiliar o processo de inspeção, as heurísticas de avaliação são divididas em 8 itens subdividido em subitens, no qual é possível analisar se aquela heurística se aplica ou não no protótipo e tecer comentários pertinentes em cada um desses itens.

Etapa 3: Nesta etapa, referente ao Capítulo 4, foram escolhidos dois protótipos *open-source* já existentes com o intuito de evitar viés durante a avaliação heurística. Essa decisão garantiu uma abordagem imparcial, focando apenas no impacto dos *frameworks* CSS na experiência do usuário (UX). Os protótipos selecionados foram:

- **Protótipo 1 (*Bootstrap*):** *Aviato*, um *e-commerce* de roupas desenvolvido com *Bootstrap* 3. Além do *framework* CSS, este protótipo utiliza tecnologias adicionais, como bibliotecas JavaScript (JQuery, Slick.js), *plugins third-party* e Node.js.
- **Protótipo 2 (*Tailwind*):** *FRNTR*, um *e-commerce* de mobílias desenvolvido com *Tailwind* CSS, *Crystallize* (CMS), *Remix* (*framework full-stack*) e Node.js.

Ambos os protótipos apresentam as funcionalidades essenciais de um *e-commerce*, como páginas de catálogo, carrinho de compras, *checkout*, e páginas informativas (contato, sobre, entre outros). A inspeção foi limitada às interfaces gráficas de algumas páginas dos protótipos, com foco em como os *frameworks* CSS interferem na experiência do usuário (UX). As avaliações consideraram a aplicação das heurísticas em páginas representativas, como:

- **Página inicial:** Para avaliar a primeira impressão e navegabilidade do site.
- **Página de catálogo:** Foco na organização dos produtos e da responsividade.
- **Página de *checkout*:** Avaliação da clareza e simplicidade do processo de compra.

Etapa 4: Durante esta etapa, os protótipos foram documentados. É importante para situar exatamente quais tecnologias e funcionalidades de cada protótipo e definir quais partes interessam para serem utilizadas como objetos de avaliação. Este processo é apresentado por meio de capturas de tela dos protótipos, qual é o escopo da avaliação. Também separa o capítulo da documentação dos protótipos, dos capítulos das avaliações heurísticas.

Etapa 5: Nesta etapa, acontecem de fato a avaliação por meio de uma tabela que contém as heurísticas do *checklist* para *e-commerce*. O *checklist* possui sete tópicos divididos em vários subtópicos. Cada item do *checklist* é testado e em seguida os campos foram atualizados com seus respectivos resultados.

Além disso, o *checklist* de usabilidade geral baseado nas heurísticas de Nielsen criado por Caballero et al (2014), serviu como forma de inspiração para adaptar o *checklist* de

e-commerce original feito por Kumar (2022) citado no Capítulo 2, seção 3.2, pois é uma abordagem mais flexível e que permite fazer anotações de cada heurística do *checklist*, gerando assim mais observações e resultados relevantes.

Ressaltando, que foi realizado este processo com ambos os protótipos, para no fim ser possível ter resultados pertinentes para detectar erros, pontuar motivos de cada *framework* ser mais ou menos efetivo em cada um dos casos e perceber suas vantagens e desvantagens gerais. Ao fim deste procedimento, foram geradas duas tabelas de *checklist* com anotações e observações distintas, uma referente ao protótipo *Aviato* e outra para o protótipo *FRNTR* para ser feito as devidas comparações de um *framework* para outro. Este processo está descrito nos Capítulos 5 e 6.

Etapa 6: A última etapa é referente aos resultados obtidos e à discussão detalhada em torno dos protótipos. Durante esta etapa, foi feita a análise e identificação de falhas de usabilidade, para testar se as observações e as constatações fazem sentido ou se correspondem com os bons padrões de usabilidade seguindo os critérios da avaliação heurística baseada em *checklist* e das heurísticas de Nielsen.

3. Revisão da Literatura

3.1 Usabilidade

De acordo com a ISO/IEC 25000 (2014), usabilidade é definida por "*medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso*". A usabilidade é um requisito não funcional crucial no desenvolvimento de software, situado como uma área que integra conceitos e práticas tanto da engenharia de software quanto do design, que abrange aspectos como a experiência do usuário, a interface gráfica, a acessibilidade e a facilidade de aprendizado do sistema. Esses aspectos são essenciais para garantir que o software atenda às especificações técnicas e as necessidades dos usuários finais (REINEHR, 2020).

A usabilidade abrange os seguintes fatores (BASS, CLEMENTS, KAZMAN, 2012):

- **Aprendizagem de *features* do sistema:** Caso o usuário não esteja familiarizado com algum tipo específico de sistema ou algum aspecto dele, o próprio sistema precisa fazer alguma ação ou fornecer *features* para que torne a experiência do usuário mais simples.
- **Uso eficiente do sistema:** O sistema precisa estimular o usuário a ter a capacidade de realizar suas tarefas e outras operações utilizando as funcionalidades do sistema de forma eficiente.
- **Minimização dos impactos de erros:** É preciso pensar em formas no qual o usuário tenha a possibilidade de errar, e que o mesmo não seja "punido" por seus equívocos com funcionalidades de cancelar uma operação.
- **Adaptação do sistema às necessidades do usuário:** O usuário e o sistema precisam ter formas que façam as necessidades do usuário serem atendidas mais facilmente. Como por exemplo, o sistema preenche automaticamente um campo de URL com base em pesquisas anteriores do usuário.
- **Aumento de confiança e satisfação:** Ter avisos e feedbacks sobre as ações que o usuário toma naquele sistema é importante para construir uma relação de confiança entre ambos. Indicar o que está sendo executado para o usuário mostra que as ações que ele toma estão corretas.

Usabilidade não é a única propriedade que é levada em consideração para realizar a avaliação da experiência do usuário. A ISO/IEC 25000 (2014) também divide usabilidade em

alguns componentes que auxiliam a analisar com mais precisão se determinado sistema terá uma experiência satisfatória para o usuário, como os atributos:

- **Acessibilidade (*Acessibility*)**: possibilidade de utilização do software por pessoas com características e capacidades diferenciadas.
- **Apreensibilidade (*Learnability*)**: grau em que o produto possibilita ao usuário aprender a usá-lo com eficiência, eficácia, satisfação e livre de riscos.
- **Estética da interface do usuário (*User interface aesthetics*)**: grau em que uma interface permite uma interação agradável e satisfatória para o usuário.
- **Operabilidade (*Operability*)**: grau em que o produto é fácil de operar, controlar e apropriado ao uso.
- **Reconhecimento apropriado (*Appropriateness recognisability*)**: grau que os usuários conseguem compreender se o produto é adequado às suas necessidades.
- **Proteção contra erros do usuário (*User error protection*)**: grau em que o produto contribui para que erros sejam evitados e tratados durante a sua utilização.

Segundo a norma ISO 9241-210 (2019), é crucial avaliar a usabilidade observando como os usuários se sentem em relação ao sistema e qual é seu nível de eficiência na execução das tarefas. Ainda de acordo com a norma, uma interface que apresenta deficiências em sua usabilidade pode gerar experiências ergonômicas desagradáveis para os usuários, que os impedem de concluir suas tarefas de forma efetiva.

Essa situação pode decorrer de fatores como: falta de feedback visual ou sonoro, excesso de informações na tela ou tempos de resposta lentos do sistema. Estes problemas podem gerar uma desmotivação no usuário em continuar a utilizar o sistema, podendo resultar em impactos negativos, como perda de dados, redução da eficiência operacional e no pior dos casos o abandono completo do sistema (SHNEIDERMAN E PLAISANT, 2016).

É por isto que usabilidade é fundamental para um sistema, ela ajuda a garantir que sistemas sejam eficientes, fáceis de aprender, acessíveis e ofereça uma experiência satisfatória ao usuário. Entretanto, avaliar esses aspectos de forma sistemática pode ser um desafio, especialmente devido à individualidade da experiência de cada usuário (NIELSEN E LORANGER, 2007). Nesse cenário, as heurísticas de Nielsen são uma referência confiável para orientar no processo de análise, permitindo que problemas em interfaces sejam percebidas por meio de heurísticas, antes que causem impactos prejudiciaisnote.

3.2 Heurísticas de Nielsen

As heurísticas de Nielsen são um conjunto de princípios desenvolvidos por Jakob Nielsen, que servem como base para uma abordagem para identificar problemas de usabilidade em interfaces gráficas de usuário. Essas heurísticas foram elaboradas a partir de observações empíricas de problemas comuns de usabilidade encontrados por Nielsen em sistemas de software e interfaces gráficas (NIELSEN E LORANGER, 2007).

As heurísticas criadas por Nielsen consistem em 10 princípios fundamentais que podem ser aplicados durante a avaliação de usabilidade de um sistema (NIELSEN, 1994):

- **Diálogo simples e natural** (*Simple and Natural Dialogue*): A interface deve ser simplificada, uma vez um a cada botão novo em tela ou funcionalidade nova criada pode se tornar um potencial desafio para o usuário aprender. Portanto, quanto mais fluído e natural o sistema for capaz de dialogar com o usuário, melhor será a experiência final.
- **Fale a língua do usuário** (*Speak the User's Language*): A linguagem, os conceitos e a terminologia utilizados no sistema devem refletir o mundo real, de modo que os usuários possam de forma intuitiva entender e relacionar-se com eles.
- **Minimizar o uso da memória do usuário** (*Minimize User Memory Load*): As funções e opções devem ser visíveis e facilmente acessíveis, reduzindo a necessidade de os usuários memorizarem informações.
- **Consistência** (*Consistency*): O sistema deve seguir convenções e padrões de design familiares para os usuários, tornando a experiência mais previsível e intuitiva. Isso encoraja o usuário a aprender mais as funcionalidades da interface, porque ele já têm um conhecimento e experiências anteriores de como interagir com determinados tipos de sistema.
- ***Feedback***: É importante que o sistema sempre informe aos usuários o que está acontecendo, fornecendo feedback adequado tanto positivo como negativo de imediato antes mesmo que algum erro aconteça.
- **Ícones de saída explícitas** (*Clearly Marked Exits*): Os usuários devem ter a liberdade de navegar pelo sistema e desfazer ou cancelar ações indesejadas facilmente, sem ficarem "presos" em um estado indesejado.

- **Atalhos** (*Shortcuts*): O sistema deve apresentar formas mais fáceis e rápidas de executar determinadas funcionalidades, oferecendo opções de navegação (incluindo atalhos) e interação que economizem tempo e esforço do usuário.
- **Mensagens de erro relevantes** (*Good Error Messages*): O sistema deve fornecer mensagens de erro claras e informativas, orientando os usuários sobre como corrigir e resolver problemas.
- **Prevenção de erros** (*Prevent Errors*): Deve ser feito um esforço para evitar que os usuários cometam erros por meio de uma boa organização e clareza na apresentação das informações na interface.
- **Ajuda e documentação** (*Help and Documentation*): O sistema deve oferecer recursos de ajuda e documentação sempre que necessário, permitindo que os usuários encontrem respostas para suas dúvidas ou dificuldades.

Essas heurísticas oferecem uma base sólida para a avaliação da usabilidade geral de interfaces gráficas. Porém, dependendo do contexto ou às vezes o tipo de sistema como por exemplo *e-commerce*, podem demandar adaptações específicas para atender a necessidades particulares, como será discutido na próxima seção.

3.2.1 Heurísticas para *e-commerce*

As heurísticas de Nielsen, originalmente desenvolvidas para avaliar a usabilidade de interfaces gráficas gerais (NIELSEN, 1993), foram adaptadas por Leavitt e Shneidermans (2007) para atender as necessidades do *e-commerce*. Nesse contexto, destacam-se fatores como a importância de manter o usuário informado sobre o progresso de suas ações, a confirmação de um pedido ou o status de entrega, bem como a utilização de termos e imagens familiares, que tornam a compreensão e a navegação mais intuitivas e acessíveis.

A aplicabilidade das heurísticas nesse contexto específico, ressalta a necessidade de ferramentas, abordagens ou técnicas que possibilitem avaliar e melhorar a qualidade dos sistemas. Ao integrar práticas de inspeção de software com as diretrizes das heurísticas para *e-commerce*, é possível criar um meio para verificar se as funcionalidades e usabilidade estão alinhadas às expectativas do público-alvo, fortalecendo a qualidade geral do produto (SHNEIDERMAN E PLAISANT, 2016).

3.3 Inspeção de Software

A Verificação e Validação (V&V), são processos que ocorrem de forma independente e que são utilizadas para garantir que o produto final atenda aos requisitos especificados e às expectativas do usuário. A verificação concentra-se em assegurar que o software está sendo construído corretamente, ou seja, que os artefatos produzidos em cada fase do ciclo de vida estão em conformidade com os requisitos e padrões estabelecidos. Já a validação busca confirmar que o software construído atende às necessidades reais do usuário e ao propósito para o qual foi desenvolvido. Essas atividades são essenciais para identificar e corrigir defeitos precocemente, reduzindo custos e esforços associados à manutenção e garantindo a qualidade do produto final (BOEHM, 2001).

A inspeção de software é uma técnica formal e sistemática de revisão de artefatos, como documentos de requisitos, código-fonte e modelos de design, que desempenha um papel fundamental no processo de V&V (NAIK E TRIPATHY, 2008). Durante as inspeções, equipes multidisciplinares analisam os artefatos em busca de defeitos, inconsistências e desvios em relação aos requisitos, permitindo a identificação precoce de problemas antes que eles se propaguem para fases posteriores do desenvolvimento (KALINOWSKI, 2008).

Ainda durante o processo de Verificação e Validação de software, é comum encontrar erros e defeitos (CAROCA, 2010). Esse cenário ocorre porque há tomadas de decisões equivocadas nas etapas do ciclo de vida de desenvolvimento, que podem acarretar em defeitos e, consequentemente, aumentar o custo de desenvolvimento e manutenção do software (BOEHM, 2001). Uma dessas consequências é o aumento no esforço para corrigir defeitos nas atividades finais do processo de desenvolvimento (KALINOWSKI, 2008).

Conforme a terminologia padrão da norma IEEE 1044-2009 - *Classification for Software Anomalies* (IEEE, 2010), são definidos os conceitos de erro, defeito, falha e problema, da seguinte forma:

- **Erro**: A diferença entre o comportamento esperado e aquele que ocorre de fato é denominado como erro. O erro humano é um resultado que faz parte do processo de resolução de problemas, assim como é a execução correta de uma ação (SANDHOF, FILGUEIRAS, 2006).

- **Defeito**: Um defeito é uma manifestação visível de um erro em um artefato de software, surgindo quando algo no sistema não funciona conforme foi planejado (GERALDI, 2015). O defeito representa, principalmente, uma vulnerabilidade concreta no código, na lógica ou na interface gráfica, que afeta diretamente a experiência do usuário (SOMMERVILLE, 2013).
- **Falha**: Uma falha é o comportamento do software que se desvia das expectativas do usuário durante sua operação. Ela pode ser desencadeada por uma combinação de diferentes defeitos presentes no sistema. Porém, não são todos os defeitos que resultarão em falhas visíveis, alguns defeitos podem permanecer latentes sem impactar diretamente a funcionalidade na percepção do usuário (ALSHAZLY, 2014).
- **Problema**: Experiência de dificuldade ou frustração enfrentada pelo usuário como resultado de uma interação insatisfatória com o sistema em uso.

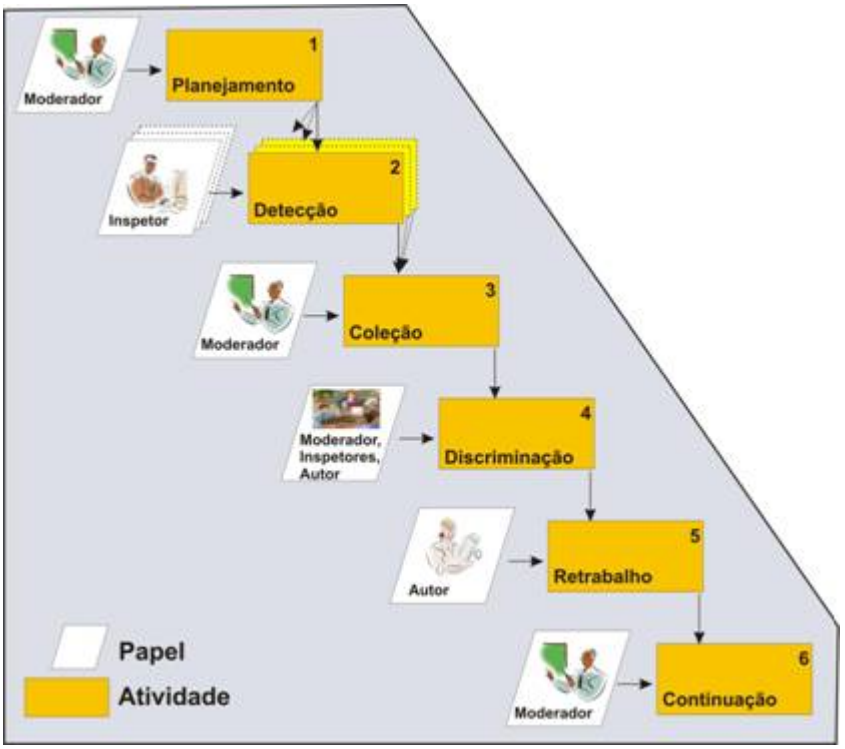
"*Os benefícios da inspeção de software cresceram ao longo do tempo e englobam: redução do nível de defeitos; aumento na satisfação do cliente; aumento na produtividade de desenvolvimento, reduzindo a entrega do software; redução de custos de gerenciamento e do ciclo de vida do software; melhora contínua dos processos por meio da remoção de defeitos sistemáticos; melhor programação de horários para as reuniões de inspeção; treinamento mais rápido para desenvolvedores de novos produtos; e melhor experiência dos inspetores ao longo do tempo*". (BRYKCZYNSKI et al., 1994; FAGAN, 2002, 1986; GILB E GRAHAM, 1993; JONES, 1996; WELLER, 1993)

Conforme a Figura 3, a inspeção de software é um processo tradicional de detecção de defeitos, sendo definido por meio das etapas: (traduzido na íntegra de FAGAN, 1999):

- **Planejamento**: Um usuário, desempenhando o papel de moderador da inspeção, define o contexto da inspeção, seleciona os inspetores e distribui o material a ser inspecionado.
- **Apresentação**: Os autores dos artefatos a serem inspecionados apresentam as características destes. Esta fase pode ser omitida se os inspetores possuem conhecimento sobre o projeto e os artefatos que devem ser inspecionados.
- **Preparação**: Os inspetores estudam os artefatos individualmente, e eventualmente fazem anotações sobre estes produzindo uma lista de discrepâncias. O fornecimento de técnicas de leitura pode facilitar a execução desta tarefa.

- **Reunião**: Uma reunião em equipe ocorre, envolvendo o moderador, os inspetores e os autores do documento. Discrepâncias são discutidas, e classificadas como defeito ou falso positivo. A decisão final sobre a classificação de uma discrepância sendo discutida é do moderador.
- **Retrabalho**: O autor corrige os defeitos encontrados pelos inspetores e confirmados pelo moderador.
- **Continuação**: O material corrigido pelos autores é repassado para o moderador, que faz uma análise da inspeção como um todo e re-avalia a qualidade do artefato inspecionado. Ele tem a liberdade de decidir se uma nova inspeção deve ocorrer ou não.

Figura 3: Etapas do processo de inspeção de software.



Fonte:: KALINOWSKI et al., 2004.

Diante da importância da inspeção de software para identificar e corrigir defeitos mais rápido, é crucial compreender as diferentes técnicas de inspeção de software que podem ser empregadas nesse processo. As principais técnicas são:

- Técnica de leitura baseada em perspectiva, que consiste em orientar os inspetores a considerarem diferentes perspectivas, como a do usuário,

desenvolvedor ou testador, para analisar os documentos (GERALDI e OLIVEIRAJR, 2017)

- Inspeção baseada em cenários, que utiliza cenários de uso ou instruções detalhadas que orientam o inspetor a procurar defeitos em situações específicas, simulando o uso real do sistema (GERALDI E OLIVEIRAJR, 2017)
- Técnica de leitura Ad-Hoc, de acordo com (MELO, 2019) “Essa técnica dispensa o uso de métodos formais de leitura, permitindo que cada leitor analise o documento conforme sua própria abordagem. Por isso, sua eficácia depende da experiência individual do inspetor e apresenta como principal limitação o fato de não ser repetível nem passível de aprimoramento, já que não há um procedimento padronizado a ser seguido.”
- Inspeção baseada em *checklist*, onde os inspetores utilizam listas de verificação com itens específicos para guiar a análise dos artefatos. (GERALDI, 2015) Lista de itens, que por sua vez auxiliam os inspetores a encontrar defeitos no produto durante o processo de inspeção, como por exemplo enumerando defeitos característicos, priorizar diferentes defeitos ou ainda em contribuir com ideias para os revisores descobrirem o defeito. (MONTEBELO, 2006)

Diante da importância da inspeção de software para a identificação e correção precoce de defeitos técnicos nos artefatos do desenvolvimento, é fundamental também considerar abordagens que avaliem a qualidade do sistema sob a perspectiva do usuário (BARBOSA, SILVA, 2010). Nesse contexto, a avaliação heurística de usabilidade surge como uma técnica complementar, voltada para identificar problemas na interação do usuário com a interface do sistema.

Essa forma de avaliação consiste em um processo sistemático no qual especialistas analisam a interface de um sistema com base em princípios consolidados de usabilidade, conhecidos como heurísticas. O intuito é identificar problemas que atrapalhem a interação do usuário, propondo melhorias que tornem o sistema mais intuitivo e eficiente. Trata-se de uma técnica que prioriza a observação crítica dos elementos de interface em relação às boas práticas de design e interação (NIELSEN, LORANGER, 2007).

Enquanto a inspeção de software foca na conformidade com requisitos e padrões técnicos, a avaliação heurística utiliza um conjunto de princípios ou heurísticas para detectar dificuldades e falhas na usabilidade, promovendo melhorias na experiência do usuário. Assim, a combinação dessas técnicas contribui para uma garantia de qualidade mais abrangente, contemplando tanto aspectos funcionais quanto a satisfação e eficiência na utilização do software (MOREIRA, 2022).

3.3.1 Avaliação Heurística de Usabilidade

A avaliação heurística é uma forma de reconhecer problemas de usabilidade que possibilita a identificação de diversos problemas em interfaces gráficas de sistemas. Essa abordagem baseia-se em um conjunto de heurísticas, que delineiam características comuns das interfaces gráficas e são derivadas do conhecimento em aspectos psicológicos, computacionais e sociológicos, sendo conduzidas por avaliadores (SCHERER, 2018).

Como outras técnicas de usabilidade, a avaliação heurística é um tipo de avaliação que utiliza alguma referência como guia para a inspeção como, por exemplo, um *checklist*. Porém, deve-se levar em conta o fato de que a usabilidade é algo subjetivo, por estar essencialmente conectado com o conhecimento e interpretação pessoal do avaliador sobre o produto e sua interação com o mesmo, o que faz com que a avaliação heurística seja frequentemente criticada, por não fornecer uma maneira sistemática de gerar soluções para os problemas de usabilidade ou uma forma de avaliar a qualidade provável de quaisquer redesigns (BROWN, 2013).

A adoção de um *checklist* no processo de avaliação heurística é fundamental. Ele atua como uma ferramenta estruturada para organizar e guiar a inspeção, reduzindo a influência de vieses individuais e garantindo que aspectos essenciais da usabilidade sejam analisados de forma sistemática. Além disso, o *checklist* facilita a identificação de problemas ao alinhar os critérios de avaliação a um padrão claro e replicável (SOUSA, 2022).

A utilização de *checklists* na avaliação heurística não apenas complementa a análise, mas também aprimora sua consistência e abrangência, como será discutido a seguir.

3.3.2 Inspeção baseada em *checklist*

Os *checklists* têm o objetivo geral de permitir ao usuário visualizar rapidamente um conjunto de tarefas ou elementos a serem observados. Esse tipo de inspeção é útil em diversas áreas, como em plataformas ou sites de *e-commerce*. Na Figura 4 desta seção, é apresentado um esboço de um *checklist* com itens associados às heurísticas, sendo tais itens organizados e categorizados para cada etapa do processo da avaliação heurística. Esse formato permite que os avaliadores, que realizam a inspeção, marquem cada etapa completada, funcionando como itens a serem verificados ou "itens de verificação" (SOUSA, 2022).

#	Review Checklist	Yes / No / Not applied	Comments
Visibility if system state			
1,1	Does every interface begin with a title or header that describes page contents?		
1,2	Is there a consistent icon design scheme and has it been used consistancy thoughout the design?		
1,3	Is selected icon / element is highlighted from other icons / elements which are not selected in the application		
1,4	Error promots at the very place the action takes place and it is consistance through out the application		
1,5	In user onboarding does the interface tell where the user is at any given moment?		
1,6	If pop-up windows are used to display error messages, do they allow the user to see the field in error? or does it overlay the the fields		
1,7	Dose the feedback change according to the type of error or notification intended?		
1,8	When click on a action button or completed a process dose the system provides a feedback to the user?		
1,9	Does the user get notified when a group of actions completed successfully and indicate the next set of actions to be taken in order to complete the task / achive the goal		
1,10	Does the elements clearly show which can be clicked and which can not be clicked?		
1,11	Does the element shape / color change when the element being selected or cursor on top the element		
1,12	If multiple options can be selected in a multiselect menu, is there visual feedback about which options are already selected?		

Figura 4: Captura parcial da tela da tabela de *checklist* adaptada de Caballero *et al*. (2014).

Além disso, a utilização do *checklist* é um guia que não segue um protocolo estritamente sistemático como pode ser observado na Figura 5. Por meio deste método de avaliação, não é determinado quais são as orientações de como a inspeção deve ser realizada, somente apresenta aos inspetores quais itens devem ser inspecionados por meio de instruções em um documento no formato de um *checklist* . (GERALDI, 2015).

Sousa (2022) complementa: "*No entanto, outras abordagens trazem diferentes definições para essa nomenclatura. Hermawati e Lawson (2016), por exemplo, apresentam o termo "checklist de usabilidade" como um conjunto básico de heurísticas de usabilidade — semelhante às 10 heurísticas de Nielsen (NIELSEN, 1994) — sem o acréscimo de itens específicos de verificação.*"

No trabalho de Caballero *et al*. (2014), foi criado um *checklist* de usabilidade geral que expande as heurísticas originais de Jakob Nielsen de dez para treze, dividido em diversos subitens. Este *checklist* foca em aspectos gerais da usabilidade, como consistência, prevenção de erros, e feedback ao usuário. Entretanto, este *checklist* acaba se tornando genérico quando trata-se de realizar avaliação heurística em *e-commerce*, por não atender com precisão os itens que são importantes para serem avaliados nestes cenários como, por exemplo, as páginas de catálogo, de checkout, de pagamento e a navegabilidade das interfaces gráficas em *e-commerce*.

Kumar (2022) criou um *checklist* organizado das heurísticas de Nielsen a fim de facilitar futuras inspeções de software. O *checklist*, por sua vez, é uma adaptação sucinta da contribuição realizada por Leavitt e Shneidermans (2007), que elaboraram heurísticas específicas para *e-commerce*. A Figura 5 apresenta o *checklist* de usabilidade para *e-commerce* utilizando o princípio do *Mobile-First*.

Table 2. Mobile-first usability guideline

Web Site Organization
G1.Provide Index page
G2.Minimize the number of clicks
G3.Open new page in new window
G4.Eliminate horizontal scrolling
Web Site Navigation
G5.Make navigation evident from the first page
G6.Navigation should be easy to find
G7.Provide navigate option to move between pages
G8.Use standard navigation Patten
Web Site Controls
G9. Use short names for buttons
G10.Minimize the number of buttons on a page
G11.Controls must be large enough for touch
G12.Visullay align controls.
G13. Use consistent and meaningful icons
Images
G14.Use optimized images
G15.Provide zoom in and zoom out features
Feedback
G16.Provide feedback while users are waiting
G17.Use help to assist customers
Task
G18.All tasks must be sequentially organized
G19.Standardize common task sequence
Item Display
G20.Ensure only necessary information is displayed
G21.Product information should be in homogeneous chunk
G22.Visited products denoted by color change
Item search
G23.Provide search field
G24.Search field should be on the top
G25.Provides search history
G26.Provide auto-complete option

Figura 5: *Checklist* de heurísticas para ecommerce de KUMAR (2022).

A Tabela 2 refere-se ao *checklist* de heurísticas proposto por Kumar (2022). O *checklist* é focado no princípio *mobile-first* e avalia aspectos como a navegação, clareza das descrições de produtos e a simplicidade no processo de *checkout*. Ele divide os princípios de avaliação em oito itens e vinte e sete subitens com a finalidade de orientar o processo de inspeção de software.

A eficácia de uma avaliação heurística em interfaces gráficas de *e-commerce* depende não apenas da aplicação rigorosa de *checklists*, mas da compreensão dos elementos técnicos que sustentam a implementação da usabilidade. Nesse sentido, o CSS *(Cascading Style Sheets)* é um recurso relevante a ser considerado durante a avaliação, pois define a apresentação visual, a responsividade e a consistência de estilo.

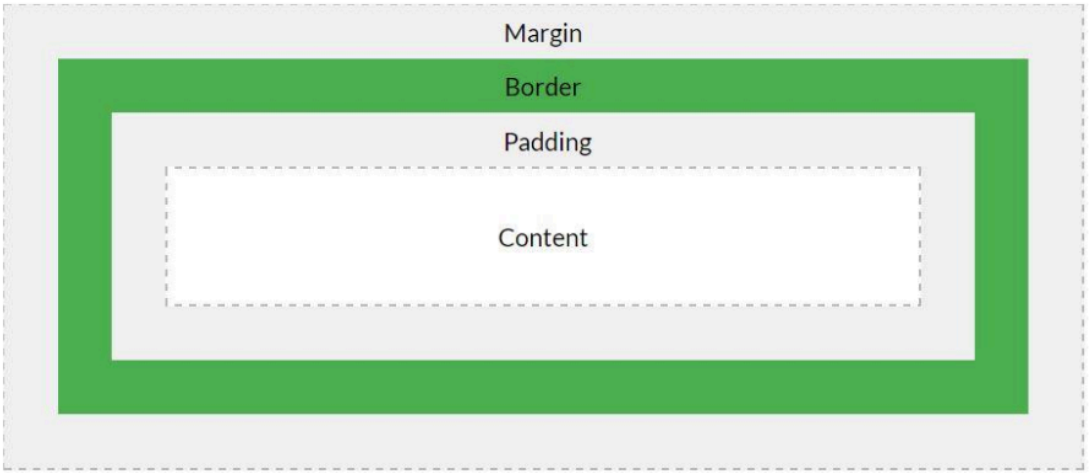
3.4 CSS

CSS (*Cascading Style Sheets*) é uma linguagem de *stylesheets* utilizada para descrever a apresentação e o estilo do conteúdo da web estruturado em HTML (*Hypertext Markup Language*) ou XML. O CSS define o formato das páginas HTML e como são exibidas na página web, sendo responsável pela aparência e sensação que as páginas transmitem, incluindo: o fundo da página; tamanho da fonte; cor e textura; animação e efeitos especiais (MAZZA, 2014).

De acordo com Salmi (2023), o CSS é definido por seus componentes principais.

- *Selectors*: O CSS utiliza seletores para selecionar elementos HTML a serem estilizados. Os seletore*s* atribuem controle preciso sobre suas regras de estilo, direcionando elementos específicos, classes, identificadores ou combinações complexas.
- *Cascading*: A cascata se refere à forma no qual os estilos são aplicados quando há regras que conflitam. Os estilos podem ser herdados de elementos pai, substituídos por regras mais específicas ou depender da ordem em que são declarados.
- *Box model*: O modelo de caixa é fundamental para o CSS e define como os elementos são renderizados em termos de conteúdo, preenchimento, bordas e margens. Ele fornece uma maneira estruturada de controlar o posicionamento e o espaçamento dos elementos em uma página da web como vista na Figura 6.

Figura 6: Diagrama ilustrando o Box Model do CSS



Fonte: https://www.w3schools.com/css/css_boxmodel.asp

A Figura 6 ilustra como estão organizadas as camadas do *Box Model,* abrangendo:

- ***Content****:* O espaço onde o texto e as imagens estão localizados, normalmente representado no centro da caixa mostrando o real conteúdo.
- ***Padding****:* A área ao redor do conteúdo, que cria espaço entre o conteúdo e a borda. É mostrado como uma camada ao redor do conteúdo.
- ***Border****:* Uma linha que envolve o padding, delimitando a caixa. Geralmente representada como uma linha ao redor do *padding*.
- ***Margin****:* A área externa à borda, que separa a caixa de outros elementos na página. É exibida como um espaço ao redor da borda.

- Layout e Posicionamento: simplifica o layout e a disposição dos elementos. Por exemplo, Flexbox e Grid são recursos avançados do CSS. Com tais recursos os desenvolvedores podem criar layouts de página responsivos e complexos.

O CSS possui duas ferramentas importantes que possibilitam aos desenvolvedores web economizar tempo na construção de sites, que são os pré-processadores e *frameworks* CSS. Os *frameworks* serão detalhados na próxima seção.

3.5 Framework web

De acordo com Meirt, "*"Framework"* é um termo amplo, frequentemente mal compreendido. Conceitualmente, um *framework* no sentido de desenvolvimento web, pode

ser comparado a uma biblioteca: não uma biblioteca de livros, mas uma biblioteca de padrões de design, completa com toda a funcionalidade necessária*"*.

É muito comum a utilização de *frameworks* no contexto de desenvolvimento web para facilitar e agilizar o desenvolvimento de novas funcionalidades, além de auxiliar na reutilização de código. Diferentemente de uma biblioteca, que o desenvolvedor chama a função no código quando necessário, o *framework* fornece padrões para a estrutura de um projeto inteiro mesmo antes do código começar a ser desenvolvido (MEIRT, 2015).

Os *frameworks* oferecem vantagens aos desenvolvedores, eles permitem uma redução no tempo de desenvolvimento, graças ao uso de estruturas prontas. São de fácil utilização, principalmente por causa da comunidade ativa de colaboradores que se ajudam atuando com diversas contribuições. Além disso, a existência de documentação organizada e detalhada, para orientar na instalação e utilização do *framework*, é uma das maiores vantagens para quem irá desenvolver os projetos (SALMI, 2023).

Portanto, o *framework* tem um papel crucial na arquitetura do sistema, bem como irá definir os parâmetros do projeto e determinar como irá funcionar o fluxo de controle do sistema (CARNEIRO e NETO, 2003). No contexto de desenvolvimento web, *frameworks* CSS como *Bootstrap* e *Tailwind* são predominantes no mercado de desenvolvimento *front-end,* principalmente quando o foco é especificamente a criação de interfaces gráficas web com HTML e CSS. Os *frameworks* podem ser ferramentas com curva de aprendizado relativamente baixa, com documentação detalhada e ser possível desenvolver interfaces gráficas responsivas e dinâmicas.

3.5.1 Framework CSS

O uso de *frameworks* CSS têm crescido no mercado de desenvolvimento web, os mesmos são responsáveis por otimizar a performance de interfaces gráficas assim como economizar tempo dos desenvolvedores na criação de páginas com designs responsivos (BHAT, 2023).

Os *Frameworks* CSS são conjuntos de bibliotecas e classes CSS que facilitam a criação de *layouts* de páginas, sendo formado por arquivos e pastas de código organizados de forma estruturada. Além de simplificar, estes *frameworks* auxiliam no processo de design do site, oferecendo uma base inicial para o desenvolvimento (SANTO, SOUZA, 2020).

A escolha do *framework* CSS ideal depende de alguns fatores, como: (i) a velocidade de instalação, já que alguns *frameworks* são mais rápidos e fáceis de instalar; (ii) a facilidade de aprendizado, determinados *frameworks* são complexos; (iii) se tem uma curva de aprendizado alto, a facilidade de configuração e integração com outros sistemas; e (iv) a disponibilidade de fazer a manutenção e suporte a longo prazo (SANTO, SOUZA, 2020).

Diante da variedade de *frameworks* CSS existentes e de suas respectivas características, tem-se a importância de avaliar seus pontos fortes e fracos, e compará-los para auxiliar os desenvolvedores a identificar o *framework* mais apropriado, considerando as exigências do projeto e a experiência do time de desenvolvimento (SALMI, 2023).

Cada *framework* CSS possui suas próprias particularidades, além de prós e contras específicos, o que gera um dilema para os desenvolvedores: "Qual *framework* melhor se adequa ao projeto em um determinado contexto?*"* Para chegar nesta conclusão, será realizado e mostrado nos capítulos 5 e 6 um estudo comparativo entre os *frameworks* Bootstrap e *Tailwind*, que são *frameworks* CSS mais utilizados e conhecidos por sua facilidade de uso, personalização e responsividade, com base no modelo de inspeção adotado por uma *checklist* heurística.

3.6 *Bootstrap*

Bootstrap é um *framework CSS open source* originalmente criado para o site do Twitter. É usado para desenvolvimento de componentes de interface e front-end para sites e aplicações web utilizando HTML, CSS e JavaScript (Site oficial do *Bootstrap*).

De acordo com Tomazini e Lopes (2015), a definição do *Bootstrap* é elaborada como: "*É uma coleção de vários elementos e funções personalizáveis para projetos da web de código aberto (opensource), empacotados previamente em uma única ferramenta. Em palavras simples, é um conjunto de ferramentas para facilitar o desenvolvimento de sites e sistemas web. Ao projetar um site com o Bootstrap, os desenvolvedores podem escolher quais elementos querem usar.*"

Atualmente, o *Bootstrap* não é apenas um *framework* com design responsivo eficaz, ele oferece também suporte para diversas opções de funcionalidade e estilo. Seus arquivos CSS e JavaScript podem ser incluídos em um projeto para ajudar na criação de elementos como por exemplo pop ups, menus e slideshows (SILVA, 2014). E devido às características próprias dos dispositivos móveis atuais o *Bootstrap* é projetado para a adaptação das telas nos diversos dispositivos, fornecendo uma ampla biblioteca de componentes que permitem

aplicações e desenvolvimento nas diversas linguagens e técnicas de web design disponíveis (SILVA, 2014).

Dentre as diversas características do *Bootstrap*, para SPURLOCK (2013) e FRAIN (2022), as que mais se destacam são:

- ***Grid System*:** O *Bootstrap* possui um sistema de grid responsivo baseado em 12 colunas, permitindo que os desenvolvedores criem layouts flexíveis e adaptáveis a diferentes tamanhos de tela.
- **Componentização:** O *Bootstrap* fornece uma ampla variedade de componentes pré-construídos, como classes de botões, formulários, navegação, alertas, entre outros, o que facilita a construção de interfaces gráficas de usuário.
- **Tema Padrão**: O tema padrão do *Bootstrap* pode ser facilmente personalizado, permitindo que os desenvolvedores criem designs consistentes e elegantes.
- **Responsividade**: É projetado para ser responsivo por padrão, adaptando-se automaticamente a diferentes tamanhos de tela, desde dispositivos móveis até desktops.
- **Suporte a Navegadores**: Possui suporte a uma ampla gama de navegadores modernos, garantindo que o seu site funcione corretamente em diferentes plataformas.
- **Modularidade**: A construção do *Bootstrap* é feita de forma modular, permitindo que os desenvolvedores selecionem apenas os componentes necessários para o seu projeto, reduzindo o tamanho do código final.
- **Comunidade e Documentação**: O *Bootstrap* possui uma grande e ativa comunidade de desenvolvedores que contribuem com plugins, temas e soluções, além de uma documentação abrangente.

Embora o *Bootstrap* seja amplamente reconhecido por sua abordagem simplificada ao desenvolvimento de interfaces gráficas e pela vasta gama de componentes pré-construídos, sua natureza padronizada pode, em certos casos, limitar a personalização e a flexibilidade do design. Isso ocorre porque os temas e estilos fornecidos pelo *Bootstrap* tendem a gerar interfaces gráficas visivelmente similares, o que pode não atender a projetos que demandam uma estética altamente customizada.

3.7 *Tailwind*

Segundo Adam Wathan, Jonathan Reinink, David Hemphill e Steve Schoger, criadores do *Tailwind* CSS: "*Assim como o Bootstrap, o Tailwind CSS é um framework CSS open source baseado em classes utilitárias que permite a construção rápida e eficiente de interfaces de usuário diretamente no HTML. Diferente de frameworks tradicionais como Bootstrap, que oferecem componentes prontos e estilizados, o Tailwind adota uma abordagem minimalista e personalizável, em que o desenvolvedor define as estilizações.*" (Site oficial do *Tailwind*)

Enquanto o *Bootstrap* se utiliza de classes semânticas, isto é, onde os nomes das classes refletem o propósito do elemento, o *Tailwind* CSS adota uma abordagem *utility-first*, com classes CSS são atômicas e imutáveis. A flexibilidade é alcançada por meio da combinação dessas classes. Atualmente, o *Tailwind* é muito utilizado em novos projetos justamente por esta abordagem *utility-first*, que pode acelerar o desenvolvimento, fornecer uma personalização mais flexível (BHAT, 2023).

O código do *Tailwind* é explícito e permite a compreensão da exibição apenas olhando o código HTML. Ele funciona bem com *frameworks* de *front-end* como *React e Vue* que têm uma estética de inserir muitos dados CSS ou JavaScript no HTML. Além disso, ele têm a função de agrupar coleções de classes para reutilização e fornece a diretiva *@apply*, para construir novas classes CSS a partir das utilidades do próprio *Tailwind* (RAPPIN, 2022).

Ainda segundo RAPPIN (2022), outra vantagem de se utilizar *Tailwind* é a sua facilidade para fazer alterações incrementais e visualizar rapidamente os resultados. Isso torna o *framework* especialmente útil durante a fase de prototipagem de um novo site, permitindo aos desenvolvedores entender com clareza o escopo e o impacto das modificações realizadas. Além disso, o *Tailwind* contribui com a redução do volume de CSS a ser escrito, já que a maior parte do design vem da composição das utilidades do *Tailwind*.

Dentre a gama de funcionalidades que o *framework* disponibiliza, alguns dos principais fatores que fazem do *Tailwind* ser uma opção de *framework* CSS ser utilizado no mercado, segundo BHAT (2023) são:

- **Abordagem *Utility-first***: O *Tailwind* CSS adota uma abordagem "*utility-first*", onde os estilos são aplicados diretamente nas classes HTML, em vez de usar classes semânticas.

- **Customização Flexível**: O *Tailwind* CSS é altamente customizável, permitindo que os desenvolvedores personalize facilmente a paleta de cores, espaçamento, tipografia e outros aspectos
- **Responsividade Integrada**: O *Tailwind* CSS possui suporte nativo para responsividade, permitindo que os desenvolvedores apliquem estilos específicos para diferentes tamanhos de tela.
- **CSS Reduzido**: O *Tailwind* CSS gera apenas os estilos CSS necessários para o seu projeto, resultando em um tamanho de arquivo menor em comparação a outras bibliotecas.
- **Suporte a plugins**: O *Tailwind* CSS pode ser estendido com plugins, permitindo que os desenvolvedores adicionem funcionalidades personalizadas.
- **Comunidade Ativa**: Por ser um *framework* relativamente novo e bem popular, o *Tailwind* possui uma comunidade ativa de desenvolvedores que compartilham recursos, plugins e soluções.

4. Descrição dos protótipos

As heurísticas de Nielsen ou até mesmo o *checklist* de usabilidade geral do estudo de Caballero et al (2014), são boas formas de realizar a avaliação heurística. Contudo, no contexto de *e-commerce* as heurísticas podem se tornar pouco proveitosas como abordagens padronizadas porque não são todas as heurísticas que coincidem com alguma característica que existe em sites de *e-commerce*. Nesse cenário faz com seja necessário uma adaptação das heurísticas e do *checklist*, como pode ser observado no Capítulo 2, feito por KUMAR (2022), e definir qual será o escopo da avaliação a ser realizada.

Pode ser inviável utilizar o *checklist* de heurísticas de Nielsen para cada interface gráfica dos sistemas. Considerando que muitas dessas heurísticas propostas por NIELSEN (1993) simplesmente não se aplicam ou não têm como ser testadas em determinadas interfaces gráficas. O foco é avaliar a usabilidade dos protótipos por meio do *checklist*, e sobretudo, avaliar os *frameworks* que são utilizados neles, o quanto eles impactam na experiência final do usuário.

Portanto, os critérios que serão definidos para avaliação heurística são: adaptar o *checklist* proposto por KUMAR (2022) acrescentando mais colunas no *checklist* para realizar os testes para cada protótipo nas paǵinas inicial, de catálogo e de checkout/pagamento, e também realizar a tradução livre das heurísticas. Abaixo segue uma breve descrição dos protótipos, e uma pequena documentação dos mesmos.

4.1 Aviato (*Bootstrap*)

O projeto *Aviato*, disponível no GitHub, serve como protótipo para avaliar o *Bootstrap*. Segundo os desenvolvedores (tradução própria): "*Aviato* é uma solução *front-end* de *e-commerce* baseada no *Bootstrap* 3, que estiliza componentes padrão e adiciona mais de 25 elementos flexíveis. Inclui páginas como Home, Loja, Carrinho, *Checkout*, Blog e *Dashboard*, além de elementos como botões, alertas e tipografia."

O projeto combina tecnologias *front-end* e automação, com foco em:

- **jQuery**: Biblioteca JavaScript para manipular HTML, gerenciar eventos, animações e interações Ajax, agilizando o desenvolvimento web.
- **Gulp**: Ferramenta JavaScript para automatizar tarefas como compilação SASS, inclusão de arquivos, auto-prefixação CSS e sincronização do navegador.

- **Node.js**: Ambiente de execução JavaScript fora do navegador, essencial para executar Gulp e ferramentas similares.
- **npm**: Gerenciador de pacotes para dependências, scripts de build e desenvolvimento.

A estrutura do projeto está na pasta 'theme' com arquivos HTML e CSS editáveis. Para uso básico, edite e abra index.html no navegador. Para executar a aplicação, instale Node.js e Gulp CLI globalmente, execute “**npm install**” para dependências. Use ”**npm run dev**” para executar no servidor local. Testes em dispositivos móveis foram feitos acessando localhost via IP da máquina no celular. Nas Figuras 7, 8 e 9 são apresentadas as capturas de tela da avaliação heurística.

Figura 7: Página inicial do Aviato.

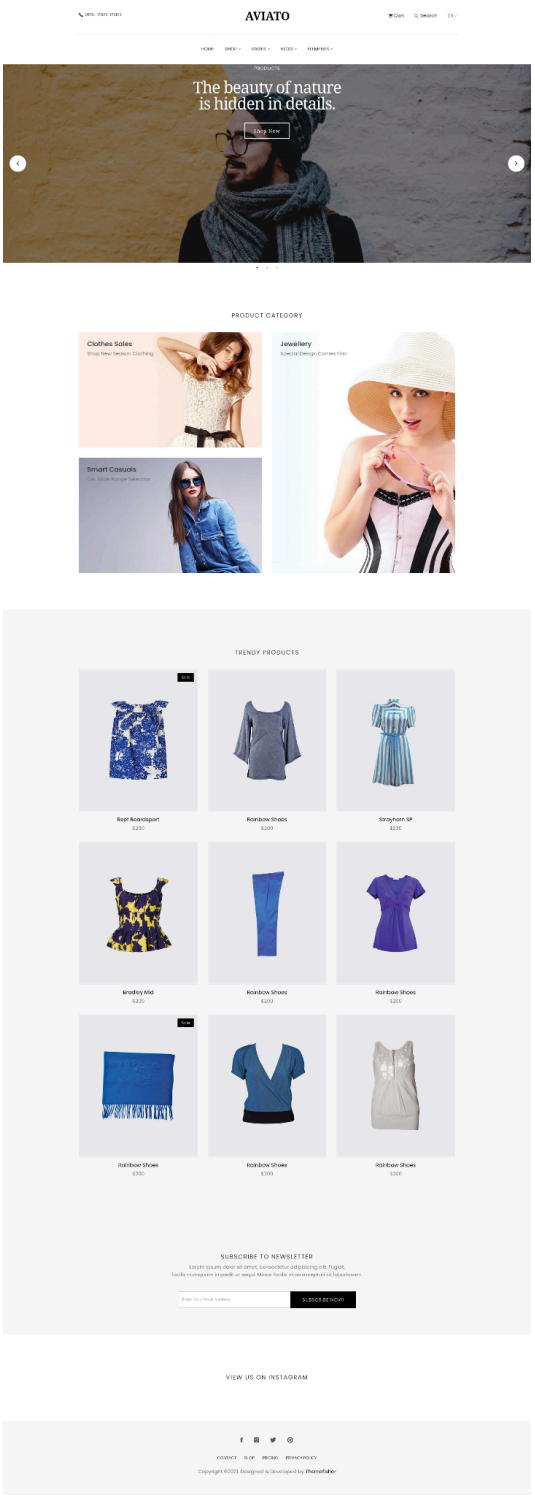


Figura 8: Página de pagamento.

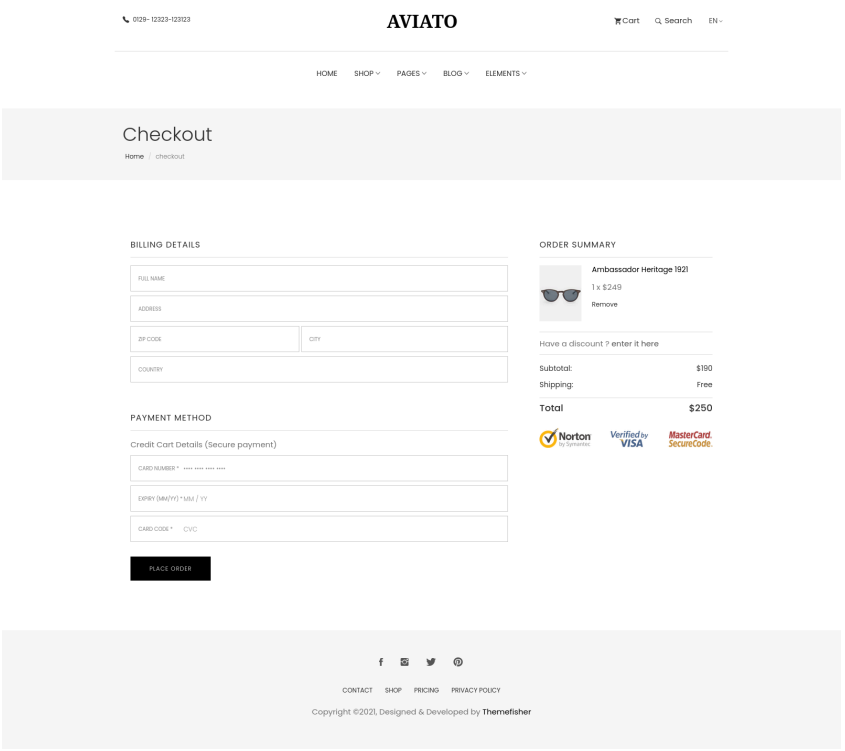
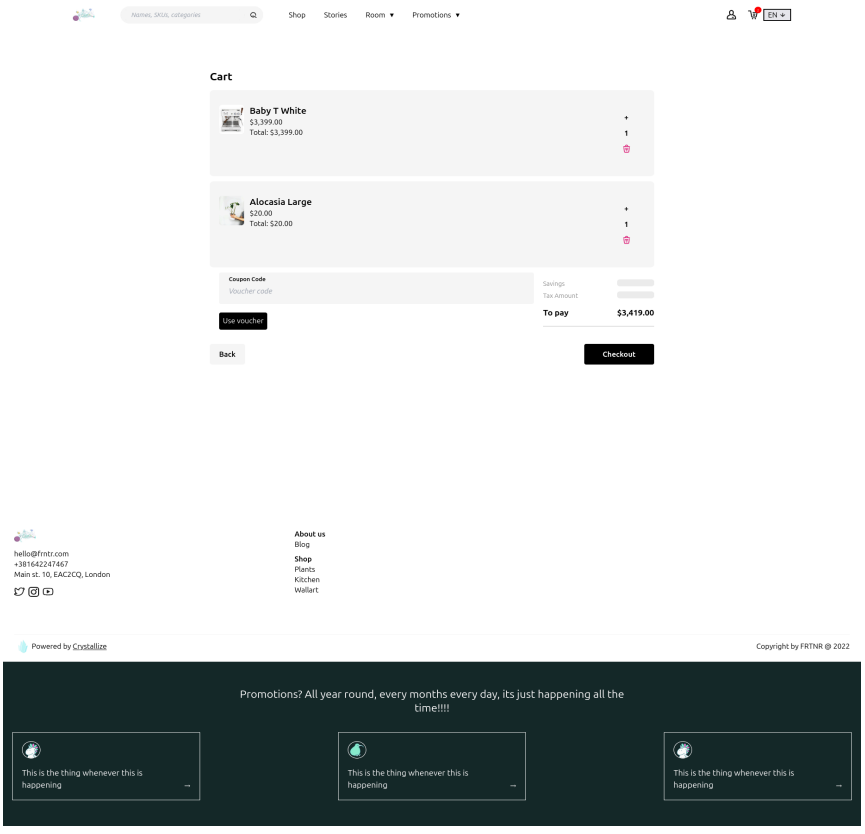


Figura 9: Página de carrinho.



4.2 FRTNR (*Tailwind*)

O FRTNR é o protótipo que utiliza o *Tailwind* como *framework* CSS encontrado no site da Vercel. O site é uma loja online, voltado para venda de mobílias, seja itens de decoração, sofás, mesas e lâmpadas personalizadas. Além do do próprio *Tailwind* CSS, outras tecnologias utilizadas no projeto são (VERCEL, s.d):

- **Remix**: Um *framework* de desenvolvimento web *full-stack* que permite construir interfaces gráficas de usuário com React e renderização do lado do servidor ou geração de site estático.
- **React**: Uma biblioteca JavaScript para construir interfaces gráficas de usuário interativas e reativas.
- **Crystallize**: Um CMS (*Content Management System*) de *e-commerce* que fornece uma API flexível para gerenciar produtos, conteúdo e dados de *e-commerce*. Ele permite que os desenvolvedores tenham total controle sobre a apresentação da loja, enquanto o conteúdo é gerenciado de forma desacoplada.
- **Node.js**: Ambiente de tempo de execução JavaScript necessário para executar o Remix e as ferramentas de desenvolvimento.
- **npm (*Node Package Manager*)**: Utilizado para gerenciar as dependências do projeto e executar scripts de desenvolvimento e build.

Assim como no protótipo anterior, neste aqui existem ferramentas em comum como o node.js e o gerenciamento dos pacotes com npm, além da questão dos testes no celular ser feito da mesma forma acessando a porta do *localhost* pelo IP da minha máquina local, e também que para a aplicação rodar de forma simples, basta utilizar o comando **npm run dev**. Além disso, este projeto conta com mais funcionalidades como suporte a conteinerização via Docker, envio de *emails*, armazenamento em cache com Redis, possibilidade de *deploy* na Vercel, dentre outras funcionalidades de infraestrutura que não foram consideradas por não serem relevantes para o processo de inspeção. Nas Figuras 10, 11 e 12 são apresentadas asseguem as capturas de tela das interfaces gráficas que foram feitas a avaliação heurística.

Figura 10: Página inicial do site.

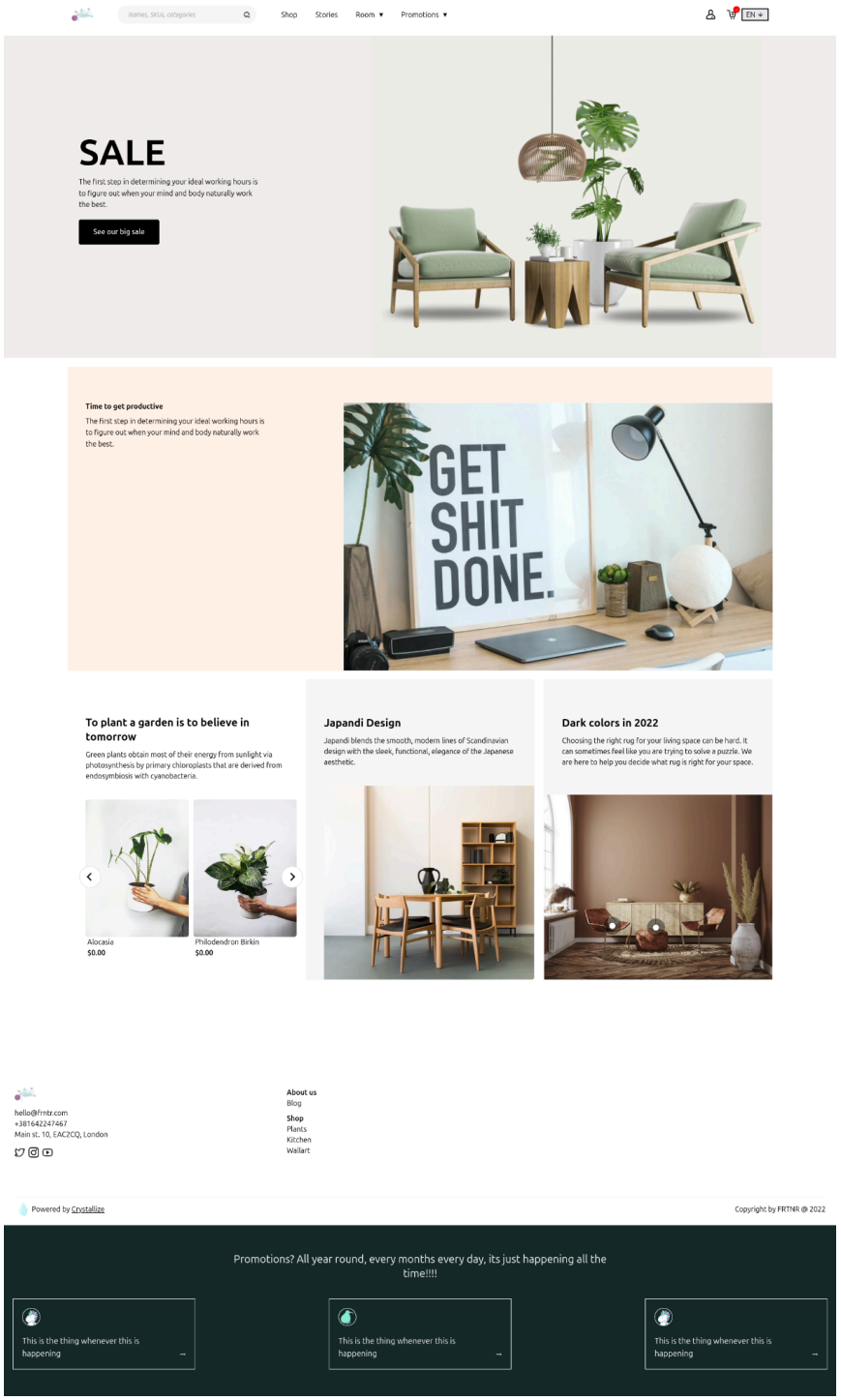


Figura 11: Página de carrinho de compras.

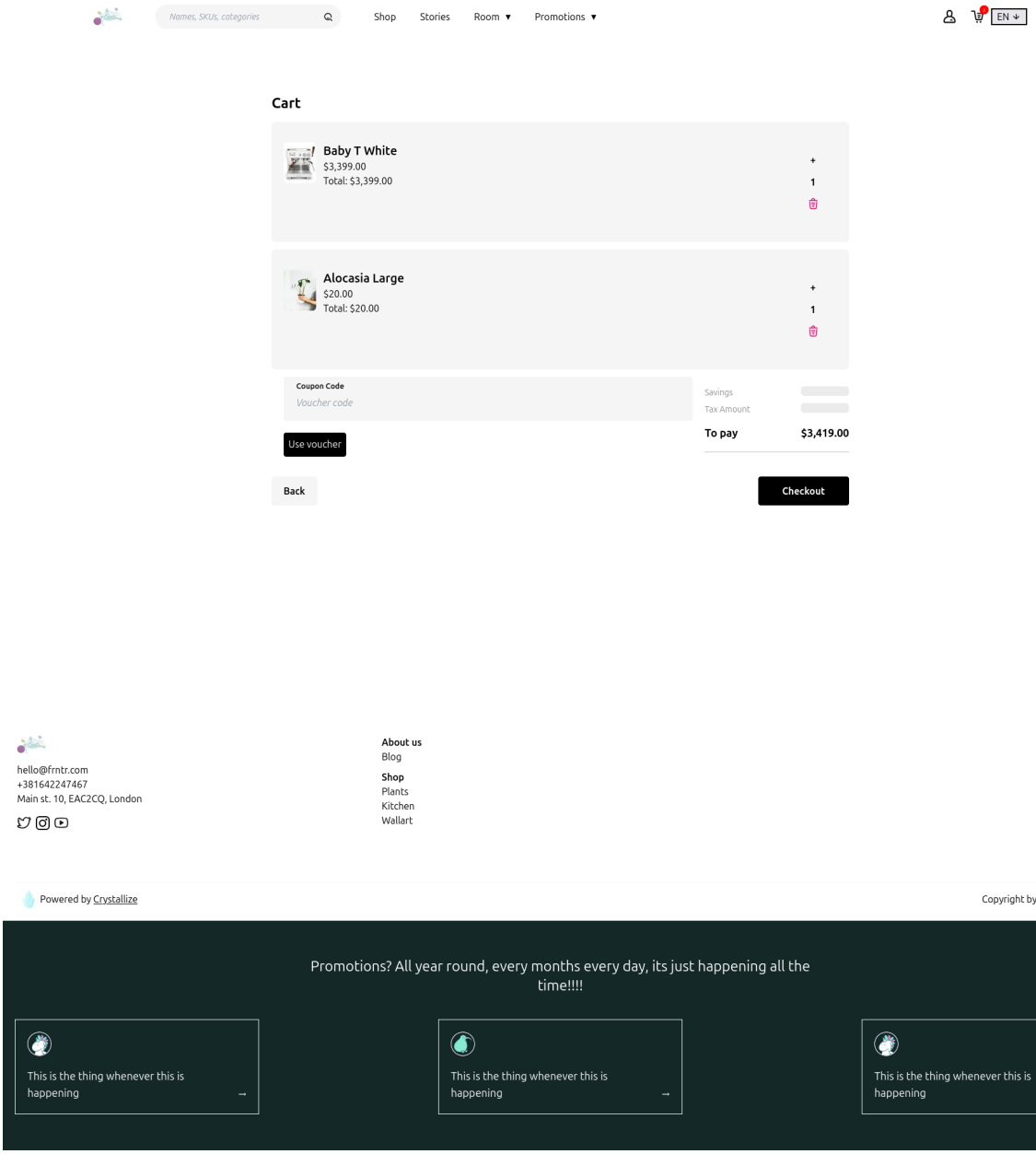
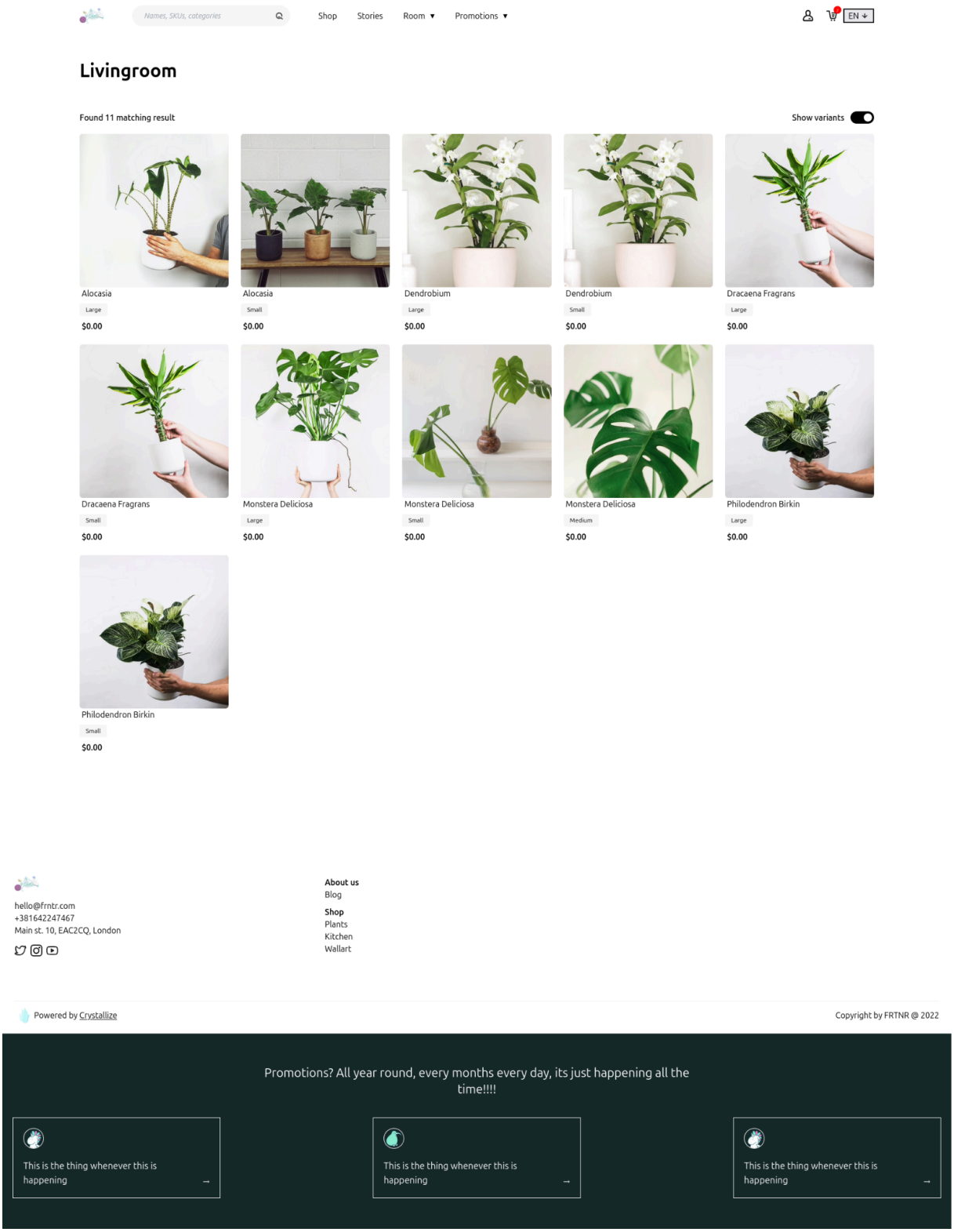


Figura 12: Página de catálogo dos produtos.



5. Avaliação do *Bootstrap* no protótipo Aviato

Neste capítulo, é feito a avaliação heurística de usabilidade via *checklist*, conforme descrito no Capítulo 4, foi adicionado as colunas de "Sim/Não/Não se aplica" e de "Comentários". Estes itens do *checklist*, foram preenchidos de acordo com a execução do projeto e análise das interfaces gráficas dentro do escopo proposto (página inicial, página de catálogo e página de checkout).

Além disso, o perfil do inspetor responsável por esta avaliação é o de um estudante do curso de Ciências da Computação, cursando o último ano de graduação, com entendimento básico sobre usabilidade e com 1 ano de experiência de mercado como desenvolvedor web. Essa informação é pertinente, para explicitar a persona que fez as avaliações dos capítulos 5 e 6 e deixar estabelecidos seus vieses. A Tabela 3 apresenta o *checklist* com os campos preenchidos após o processo de avaliação**.**

Tabela 3: *Checklist* de heurísticas de usabilidade para *e-commerce* preenchido do protótipo Aviato.

#	Review Checklist	Sim/ Não/ Não se aplica	Comentários
1. Organização do site			
1.1	Fornecer página de índice	Sim	A página de índice existe que é a página inicial do site, facilitando a navegação para as diferentes seções do site.
1.2	Minimizar o número de cliques	Sim	O número de cliques necessários para acessar as principais áreas do site do catálogo, carrinho e pagamentos é eficiente.
1.3	Abrir nova página em nova janela	Não se aplica	A abertura de novas páginas em uma nova janela não é usada neste site, pois não é necessário para o fluxo de navegação.
1.4	Eliminar a rolagem horizontal	Sim	O site elimina a rolagem horizontal, garantindo que o conteúdo caiba na tela sem exigir rolagem lateral.
2. Navegação do site			

2.1	Tornar a navegação evidente desde a primeira página	Sim	Dá para perceber que desde a página inicial, é intuitivo a navegação, olhar a primeira página percebe-se rapidamente onde se deve clicar.
2.2	A navegação deve ser fácil de encontrar	Sim	Os menus de navegação são facilmente localizáveis, com menus suspensos no topo da página e botões no corpo das paǵinas em lugares visíveis.
2.3	Fornecer opção de navegação para mover-se entre páginas	Sim	O site oferece botões ou links que permitem a movimentação fácil entre páginas relacionadas.
2.4	Usar padrão de navegação padrão	Sim	O padrão de navegação segue convenções conhecidas e intuitivas assim como em *e-commerces* conhecidos.
3. Controles do site			
3.1	Usar nomes curtos para os botões	Sim	Os botões do site possuem nomes curtos e claros, permitindo uma fácil compreensão de suas funções.
3.2	Minimizar o número de botões em uma página	Sim	O número de botões em cada página é limitado, evitando sobrecarregar o usuário com opções desnecessárias.
3.3	Os controles devem ser grandes o suficiente para toque	Sim	Os botões e *links* são grandes o suficiente para interação por toque em dispositivos móveis.
3.4	Alinhar visualmente os elementos interativos	Sim	Os elementos de interação da página como botões e *links* estão visualmente alinhados, criando uma interface limpa e organizada.
3.5	Usar ícones consistentes e significativos	Sim	Ícones são usados de forma consistente e possuem significados claros, facilitando a interação do usuário.
4. Imagens			
4.1	Usar imagens otimizadas	Sim	As imagens são otimizadas, o que melhora o tempo de carregamento e a experiência do usuário.

4.2	Fornecer funções de zoom in e zoom out	Não	O site não oferece uma função de *zoom* nas imagens, o que poderia ser uma adição útil, especialmente para visualizar detalhes de produtos.
4.3	Fornecer feedback enquanto os usuários aguardam	Sim	Durante o carregamento, o site oferece *feedback* ao usuário, como ícones de carregamento, indicando que a ação está sendo processada.
4.4	Usar ajuda para auxiliar os clientes	Sim	Há uma seção de ajuda ou FAQ que oferece suporte para os usuários em caso de dúvidas.
5. Tarefas			
5.1	Todas as tarefas devem ser organizadas sequencialmente	Sim	As tarefas são organizadas de forma sequencial e lógica, facilitando o progresso do usuário em cada etapa.
5.2	Padronizar a sequência comum de tarefas	Sim	As sequências de tarefas seguem um padrão, permitindo que os usuários se familiarizem com o fluxo de trabalho ao longo do uso.
6. Exibição de itens			
6.1	Garantir que apenas informações necessárias sejam exibidas	Sim	Apenas as informações essenciais são exibidas, evitando sobrecarregar a página com detalhes irrelevantes.
6.2	As informações do produto devem estar em blocos homogêneos	Sim	As informações do produto são organizadas em blocos homogêneos, facilitando a leitura e comparação.
6.3	Produtos visitados indicados por mudança de cor	Não	Produtos já visitados não são marcados com uma alteração de cor.
7. Busca de itens			
7.1	Fornecer campo de busca	Sim	O site oferece um campo de busca, facilitando a localização de produtos ou informações específicas.
7.2	O campo de busca deve	Sim	O campo de busca está posicionado no topo da

	estar no topo		página, porém talvez fosse melhor ter uma barra já à mostra do que passar o mouse na lupa para a barra de pesquisa aparecer.
7.3	Fornece histórico de busca	Não	O site não armazena e nem exibe o histórico de busca do usuário, o que poderia melhorar a experiência de pesquisa.
7.4	Fornecer opção de autocompletar	Não	O site não oferece a opção de autocompletar durante a digitação no campo de busca.

Tabela 3: *Checklist* de heurísticas de usabilidade para *e-commerce* preenchido do protótipo Aviato.

5.1 Considerações

Neste protótipo, foi percebido que a interface é bem estruturada e alinhada às principais heurísticas de usabilidade para *e-commerce*, especialmente pela navegação intuitiva, tamanhos adequadamente dimensionados para celulares, por exemplo e, organização dos elementos de forma harmônica. Cinco critérios foram avaliados como “Não” ou “Não se aplica” durante a avaliação heurística, sendo os itens: Abrir nova página em nova janela (1.3), Fornecer funções de *zoom in* e *zoom out* (4.2), Produtos visitados indicados por mudança de cor (6.3), Fornecer histórico de busca (7.3) e Fornecer opção de autocompletar (7.4).

Também foi observado, a otimização do carregamento de páginas e oferecimento de feedback claro durante a interação. A organização dos elementos da interface e a exibição de informações de forma clara contribuem para uma experiência de usuário satisfatória. Algumas dessas heurísticas talvez não foram atendidas no *checklist* iguais às citadas anteriormente, por ausência da implementação ou talvez por decisão dos próprios desenvolvedores.

6. Avaliação do *Tailwind* no protótipo FRTNR

Da mesma forma que no capítulo anterior, neste capítulo é feita a avaliação heurística de usabilidade via *checklist* do protótipo FRTNR, que como descrito no capítulo 4 foram adicionadas as colunas de “Sim/Não/Não se aplica” e de “Comentários”. Estes itens do *checklist*, foram preenchidos de acordo com a execução do projeto e análise das interfaces gráficas dentro do escopo proposto (página inicial, página de catálogo e página de checkout). A Tabela 4 apresenta o *checklist* com os campos preenchidos após o processo de avaliação**.**

Tabela 4: *Checklist* de heurísticas de usabilidade para *e-commerce* preenchido do protótipo FRNTR.

#	Review Checklist	Sim / Não / Não se aplica	Comentários
1. Organização do site			
1.1	Fornecer página de índice	Sim	A navegação principal está presente com links para "*Shop*", "*Stories*", "*Room*" e "*Promotions*".
1.2	Minimizar o número de cliques	Sim	As principais áreas podem ser acessadas com poucos cliques a partir da barra de navegação.
1.3	Abrir nova página em nova janela	Não se aplica	Não há necessidade de abrir novas janelas para o fluxo de navegação atual.
1.4	Eliminar rolagem horizontal	Sim	Não há rolagem horizontal nas três interfaces gráficas apresentadas.
2. Navegação do site			
2.1	Tornar a navegação evidente	Sim	A navegação é clara e visível desde a página inicial, com a barra de navegação superior em todas as telas.
2.2	A navegação deve ser fácil de encontrar	Sim	A barra de navegação está localizada na parte superior e é fácil de localizar.
2.3	Fornecer opções para	Sim	O site permite fácil movimentação entre

	mover-se entre as páginas		páginas e categorias usando a barra de navegação.
2.4	Usar padrão de navegação padrão	Sim	O padrão de navegação segue convenções comuns de *e-commerce*, o que facilita o uso.
3. Controles do site			
3.1	Usar nomes curtos para botões	Sim	Os botões, como "*Checkout*", "*Use voucher*", "*Back*", e os de navegação, têm nomes curtos e objetivos.
3.2	Minimizar o número de botões em uma página	Sim	O número de botões é limitado e condizente com a função da página.
3.3	Controles devem ser grandes o suficiente para toque	Sim	Os controles, como botões de adição de itens e "*Checkout*", são grandes o suficiente para interação em dispositivos móveis.
3.4	Alinhar visualmente os controles	Sim	Os controles estão visualmente alinhados, mantendo uma interface organizada.
3.5	Usar ícones consistentes e significativos	Sim	Ícones são usados de forma consistente, como o carrinho e as setas de rotação de produtos.
4. Imagens			
4.1	Usar imagens otimizadas	Sim	As imagens são de alta qualidade e parecem bem otimizadas em termos de carregamento.
4.2	Fornecer função de zoom para imagens	Não	Não há função de zoom para os itens nas páginas de produto.
4.3	Fornecer feedback enquanto os usuários esperam	Não se aplica	Não foi possível identificar comportamentos de carregamento.
4.4	Usar ajuda para auxiliar os	Não se	Não foi identificada uma seção de ajuda

	clientes	aplica	visível nas interfaces gráficas fornecidas.
5. Tarefas			
5.1	Todas as tarefas devem ser organizadas sequencialmente	Sim	O fluxo de *checkout* e navegação segue uma ordem lógica e sequencial.
5.2	Padronizar a sequência de tarefas comuns	Sim	O processo de adicionar ao carrinho e realizar o *checkout* segue um padrão comum em sites de *e-commerce*.
6. Exibição de itens			
6.1	Garantir que apenas as informações necessárias sejam exibidas	Sim	As interfaces gráficas mostram apenas as informações essenciais de produtos e ações.
6.2	Informações do produto devem ser organizadas em blocos homogêneos	Sim	As informações de produto são exibidas em blocos, como nome, preço e imagem do item.
6.3	Produtos visitados devem ser indicados por mudança de cor	Não se aplica	Não há indicação de produtos visitados entre as interfaces gráficas.
7. Busca de itens			
7.1	Fornecer campo de busca	Sim	O campo de busca está presente no topo da página.
7.2	O campo de busca deve estar no topo	Sim	O campo de busca está posicionado corretamente no topo.
7.3	Fornecer histórico de busca	Não	Não há uma funcionalidade visível de histórico de busca.
7.4	Fornecer opção de autocompletar	Não se aplica	Não foi possível avaliar a função de autocompletar.

6.1 Considerações

O protótipo FRTNR atende à maioria das heurísticas, com boa organização do site, navegação clara e intuitiva, controles bem projetados e exibição de informações de forma limpa e sequencial. A presença de imagens otimizadas, a consistência no uso de ícones e escolha da disposição dos elementos no *layout* são pontos positivos.

Assim como no protótipo anterior, algumas heurísticas de *zoom* nas imagens (4.2), indicação de produtos visitados (6.3), histórico de busca (7.3) e uma seção de ajuda visível (4.4) estão ausentes. A falta de avaliação de *feedback* durante carregamento (4.3) e autocompletar (7.4) também são pontos que requerem uma atenção para serem desenvolvidas.

A heurísticas de *zoom* nas imagens e histórico de busca devem ser prioridade, porque são funcionalidades básicas esperadas em *e-commerce*, juntamente com a adição de uma seção de ajuda visível, pois é crucial para oferecer suporte ao usuário.

7. Resultados e discussão

7.1 Observações em comum dos protótipos

Durante o processo de avaliação heurística dos protótipos, foi observado que alguns dos itens avaliados tiveram resultados semelhantes, seja por decisão dos desenvolvedores que criaram as interfaces gráficas ou por desconhecimento de bons padrões de usabilidade. Este resultado foi obtido por meio da observação direta dada pela comparação de ambos os *checklists*, no qual algumas dessas heurísticas tiveram o mesmo comportamento quando testadas nos dois projetos, que serão mais detalhadas a seguir.

7.1.1 Falta de Função de *Zoom* nas Imagens

Os sites não oferecem funcionalidade de *zoom in* e *zoom out* nas imagens dos produtos, o que pode limitar a capacidade dos usuários de visualizar detalhes importantes, especialmente em um contexto de *e-commerce* onde a visualização do produto é crucial. Pode ser interessante implementar uma função de *zoom* para imagens de produtos, permitindo que os usuários ampliem as imagens ao passar o mouse ou tocar, em dispositivos móveis. Isso pode melhorar a experiência do usuário e aumentar a confiança na compra.

7.1.2 Ausência de Histórico de Busca

Os sites não armazenam ou exibem o histórico de busca, o que poderia facilitar a navegação para usuários que desejam repetir pesquisas anteriores ou refinar buscas. Uma solução poderia ser a adição de uma funcionalidade de histórico de busca, exibindo pesquisas recentes em um menu suspenso no campo de busca. Isso pode melhorar a usabilidade, especialmente para usuários frequentes.

7.1.3 Abertura de Novas Páginas em Nova Janela

A ausência de uma funcionalidade para abrir novas páginas pode ser um ponto a revisar, dependendo do contexto. Em alguns casos, abrir links externos (como informações adicionais ou promoções) em uma nova janela pode evitar que o usuário saia do fluxo

principal do site. Uma possível melhoria seria avaliar se links externos abrem em novas janelas seria benéfico para manter o usuário no fluxo de compra. Se implementado, garantir que isso seja feito de forma consistente e com *feedback* claro, por exemplo : um ícone indicando que o link abrirá em uma nova aba.

O protótipo desenvolvido com *Bootstrap*, apresenta uma interface bem estruturada e alinhada às principais heurísticas de usabilidade para *e-commerce*, destacando-se pela navegação intuitiva, botões com tamanhos adequados para dispositivos móveis e organização visual eficiente. A conformidade com a maioria dos critérios avaliados (apenas dois "Não" em 24) reflete a robustez da base criada com o *framework*, que otimiza o carregamento de páginas e oferece *feedback* claro durante a interação. A organização sequencial de tarefas e a exibição de informações de forma clara contribuem para uma experiência de usuário positiva, com destaque para a acessibilidade e o design adaptado a diferentes dispositivos.

Apesar dos pontos fortes, algumas lacunas limitam a profundidade da interação. A ausência de funcionalidade de *zoom* nas imagens de produtos é uma deficiência significativa, especialmente em *e-commerce*, onde a visualização de detalhes é crucial para a decisão de compra. Implementar essa função, permitindo ampliar imagens ao passar o mouse ou tocar em dispositivos móveis, pode aumentar a confiança do usuário e melhorar a experiência. Outro ponto crítico é a falta de histórico de busca, o que poderia facilitar a navegação ao permitir que usuários revisitem pesquisas anteriores ou refiram buscas com mais facilidade. Adicionar um menu suspenso no campo de busca para exibir pesquisas recentes seria uma solução prática para melhorar a usabilidade, especialmente para usuários frequentes.

A seção de ajuda, embora presente, pode não ser suficientemente visível, o que representa uma oportunidade de melhoria. Incluir um ícone fixo de "?" no canto inferior direito ou um *link* destacado no cabeçalho tornaria o acesso mais intuitivo. Além disso, a estratégia para abertura de novas páginas merece atenção. Embora marcada como "Não se aplica", avaliar a possibilidade de abrir links externos em novas janelas, com ícones que sinalizam essa ação, pode ajudar a manter o usuário no fluxo principal de compra, desde que implementado de forma consistente e com feedback claro.

Em resumo, o protótipo *Aviato* demonstra uma base sólida para *e-commerce*, com navegação clara, controles bem projetados e otimização eficiente. No entanto, a implementação de zoom nas imagens e histórico de busca é prioritária para atender às expectativas de plataformas modernas e potencializar a conversão. Realizar testes de usabilidade com usuários reais também é recomendado para validar as heurísticas atendidas e

identificar possíveis ajustes, garantindo que a experiência do usuário seja ainda mais fluida e satisfatória. Dos 26 itens do *checklist*, 20 se enquadram em “Sim”, 4 em “Não” e 2 em “Não se aplica”.

Já o protótipo desenvolvido com *Tailwind*, apresenta uma interface bem estruturada que atende à maioria das heurísticas de usabilidade para *e-commerce*, destacando-se pela organização clara do site, navegação intuitiva e controles bem projetados, adequados para dispositivos móveis. A exibição sequencial de informações, a otimização de imagens e a consistência no uso de ícones contribuem para uma experiência de usuário limpa e eficiente, alinhada às convenções de plataformas de comércio eletrônico.

No entanto, algumas limitações impactam a usabilidade. A ausência de funcionalidade de *zoom* nas imagens de produtos é um ponto crítico, especialmente em um contexto de *e-commerce* onde a visualização de detalhes visuais é essencial para a decisão de compra. Implementar uma função que permita ampliar imagens ao passar o mouse ou tocar em dispositivos móveis pode aumentar a confiança do usuário. Da mesma forma, a falta de um histórico de busca dificulta a repetição de pesquisas ou a navegação eficiente para usuários recorrentes. Adicionar um menu com sugestões baseadas em buscas anteriores no campo de busca seria uma melhoria prática para otimizar a experiência, especialmente em sessões prolongadas.

Outro aspecto a ser aprimorado é a visibilidade da seção de ajuda. A ausência de uma FAQ ou suporte claramente acessível pode dificultar a resolução de dúvidas, comprometendo o suporte ao cliente. Incluir uma seção de ajuda no menu principal ou rodapé, com informações sobre políticas de compra, devoluções e suporte, além de considerar um ícone de chat ao vivo, tornaria a assistência mais acessível. A falta de indicação visual para produtos visitados também pode dificultar a navegação em catálogos extensos, sendo recomendável implementar marcadores visuais, como mudanças de cor, para ajudar os usuários a rastrearem sua interação com o site.

Além disso, a ausência de avaliação de feedback durante o carregamento e da funcionalidade de autocompletar no campo de busca sugere lacunas no design ou na análise. Garantir *feedback* visual, como barras de progresso ou ícones de carregamento, durante ações que requerem espera, pode melhorar a percepção de desempenho e reduzir a frustração. Da

mesma forma, implementar e testar uma função de autocompletar que sugira termos ou produtos relevantes durante a digitação aceleraria o processo de busca e aprimoraria a usabilidade.

Em resumo, o protótipo FRTNR oferece uma base sólida para *e-commerce*, com navegação clara e design consistente, mas a implementação de *zoom* nas imagens, histórico de busca e uma seção de ajuda visível é prioritária para atender às expectativas dos usuários. Alguns testes adicionais podem ser relevantes, para avaliar o *feedback* durante carregamento e a funcionalidade de autocompletar, aliados a testes de usabilidade com usuários reais, são essenciais para validar as heurísticas atendidas e garantir uma experiência mais fluida e satisfatória. Dos 26 itens do *checklist*, 19 se enquadram em "Sim", 2 em "Não" e 5 em "Não se aplica".

7.2 Comparação dos frameworks

A comparação entre os frameworks foi realizada a partir dos resultados obtidos da avaliação heurística de cada *checklist* e das referências da literatura abordadas no Capítulo 2. Após fazer a inspeção das interfaces gráficas, os dados foram organizados de forma a identificar semelhanças, diferenças e padrões em relação às heurísticas avaliadas. As principais vantagens e desvantagens percebidas foram as seguintes:

7.2.1 *Bootstrap*

As principais vantagens são:

1. **Componentes prontos**: O *Bootstrap* oferece uma diversidade de componentes pré-estilizados (botões, *dropdowns*, modais, formulários), que aceleram a implementação de heurísticas como a navegação clara (2.1–2.4), controles consistentes (3.1–3.5) e feedback durante carregamento (4.3).
2. **Consistência visual**: Pelo fato de se utilizar os sempre os mesmos componentes disponibilizados pelo framework, a estrutura padrão do *Bootstrap* ajuda a garantir alinhamento visual (3.4), eliminação de rolagem horizontal (1.4), e uma interface coesa, facilitando a aderência às heurísticas de organização (1.1–1.4).

3. **Agilidade no desenvolvimento**: A facilidade de uso de classes pré-definidas reduz o tempo necessário para criar uma interface funcional, permitindo foco em funcionalidades básicas de *e-commerce*.
4. **Suporte a interações simples**: Nas heurísticas de inclusão de autocompletar (7.4) e feedback durante carregamento (4.3), é percebido que o *Bootstrap* lida bem integrando JavaScript para interações deste tipo.

Desvantagens:

1. **Menor flexibilidade de design**: O *Bootstrap* pode gerar interfaces gráficas que parecem "genéricas" devido à dependência de estilos padrão, o que pode ter limitado a implementação de funcionalidades mais customizadas, como zoom nas imagens (4.2).
2. **Limitação em funcionalidades avançadas**: A ausência de histórico de busca (7.3) pode indicar que o *Bootstrap* não facilita interações dinâmicas complexas sem bibliotecas adicionais ou até mesmo algum outro framework front-end mais robusto.

Tailwind

Vantagens:

1. **Flexibilidade de design**: O *Tailwind* permitiu a criação de uma interface visualmente moderna e personalizada, com controles bem estilizados (3.1–3.5) e layouts responsivos (1.1–1.4, 2.1–2.4), atendendo às heurísticas de organização e navegação.
2. **Tamanho otimizado**: Por compilar dinamicamente as classes utilitárias no modo JIT (*Just-In-Time*), é gerado apenas o CSS necessário para o projeto, portanto as imagens otimizadas (4.1) e a interface podem implicar que o *Tailwind* oferece um código mais otimizado.
3. **Customização fácil**: A abordagem *utility-first* pode ter facilitado nos ajustes no posicionamento do campo de busca (7.2) e no alinhamento de controles (3.4), sem a limitação de componentes prontos.

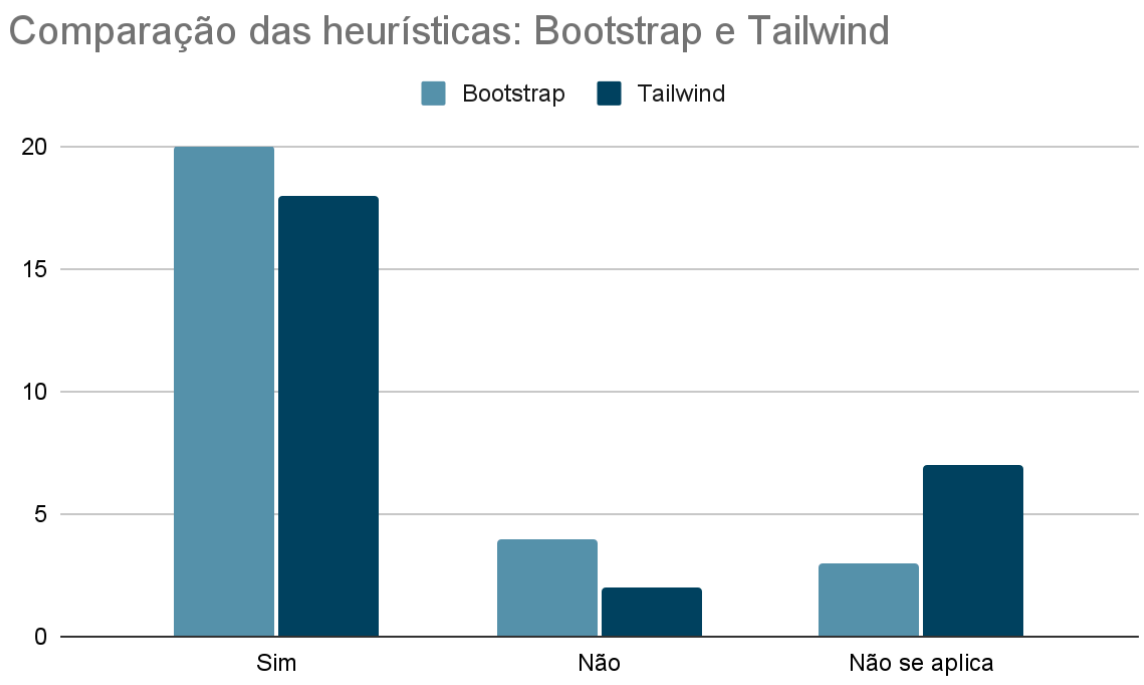
4. **Escrever CSS é dispensável**: Pelo *Tailwind* permitir a construção de interfaces gráficas aplicando diretamente classes utilitárias no HTML, é possível desenvolver o projeto sem a necessidade de escrever CSS manualmente, uma vez que o desenvolvedor entenda bem as classes utilitárias e saiba como utilizá-las corretamente

Desvantagens:

1. **Falta de componentes prontos**: A ausência de componentes pré-construídos exigiu mais esforço para implementar funcionalidades como seção de ajuda (4.4), feedback durante carregamento (4.3), e autocompletar (7.4), que não foram avaliados ou implementados.
2. **Curva de aprendizado**: A abordagem utilitária do *Tailwind* pode ser mais trabalhosa para equipes menos experientes, resultando em lacunas como a falta de indicação de produtos visitados (6.3) ou histórico de busca (7.3).
3. **Maior tempo de desenvolvimento**: Atrelado ao motivo anterior de curva de aprendizado, se o time de desenvolvedores forem iniciantes, a necessidade de estilizar cada componente manualmente pode ter contribuído para a não implementação por exemplo, de funcionalidades dos protótipos como zoom (4.2) e seção de ajuda (4.4).
4. **Dificuldade de manutenção:** Em contexto de aplicações comerciais, a necessidade de realizar entregas com prazo limitado e mais uma vez dependendo da experiência do time de desenvolvedores, pode ser que aplicar boas práticas de programação não sejam realizadas, contribuindo para um código *Tailwind* menos semântico e um código HTML mais verboso.

A Figura 13 mostra a quantidade de heurísticas que nos testes tiveram feedback de “Sim”, “Não” e “Não se aplica” por *framework*.

Figura 13: Gráfico de comparação de heurísticas por framework



Fonte: Autoria própria (2025)

Heurísticas atendidas ("Sim"): O *Bootstrap* demonstrou um número ligeiramente maior de heurísticas atendidas (20) em comparação com o *Tailwind* (18). Isso reforça a ideia de que o *Bootstrap*, com seus componentes prontos, facilita a conformidade com um conjunto básico de boas práticas de usabilidade de forma mais imediata.

Heurísticas não atendidas ("Não"): Ambos os frameworks apresentaram um número baixo de heurísticas não atendidas (*Bootstrap*: 4, *Tailwind*: 2). As heurísticas não atendidas geralmente indicam falhas ou ausências de funcionalidades importantes, como a falta de *zoom* nas imagens ou histórico de busca.

Heurísticas "Não se aplica": Na categoria "Não se aplica" o resultado é contraintuitivo. O *Tailwind* apresentou um número significativamente maior (7) de heurísticas nessa categoria em comparação com o *Bootstrap* (3). Isso pode indicar que a abordagem *utility-first* do *Tailwind*, por não oferecer componentes pré-construídos para certas funcionalidades (como feedback de carregamento ou autocompletar), exige que as mesmas sejam implementadas manualmente, ou que não foram consideradas no escopo dos protótipos avaliados. No caso do *Bootstrap*, o menor número de "Não se aplica" sugere que seus componentes já contemplam mais cenários por padrão.

Em resumo, o gráfico ilustra que, embora ambos os frameworks sejam capazes de produzir interfaces gráficas com boa usabilidade, o *Bootstrap* tende a entregar um nível de conformidade básico rapidamente devido à sua natureza *component-first*, enquanto o *Tailwind*, apesar de sua flexibilidade, pode exigir mais esforço para cobrir todas as heurísticas, especialmente aquelas que dependem de funcionalidades pré-construídas.

7.3 Considerações finais

Os resultados obtidos e as limitações identificadas na análise dos protótipos Aviato (*Bootstrap*) e FRTNR (*Tailwind*) revelam informações cruciais sobre o impacto de cada framework no desenvolvimento de interfaces gráficas para *e-commerce*. No caso do *Bootstrap*, seu impacto foi predominantemente positivo, facilitando a criação de uma interface consistente e funcional, seguindo apropriadamente as heurísticas de usabilidade. No entanto, essa abordagem apresentou limitações na implementação de funcionalidades mais customizadas, como a função de *zoom* em imagens e o histórico de busca.

As principais vantagens do *Bootstrap* estão na rapidez de desenvolvimento, na consistência visual proporcionada por seus componentes prontos e na facilidade de uso para equipes menos experientes. Por outro lado, suas desvantagens incluem uma menor flexibilidade de design, o que pode resultar em interfaces gráficas genéricas, e um peso potencialmente maior da aplicação devido à dependência de JavaScript. Diante disso, o *Bootstrap* é mais indicado para projetos com prazos curtos, que necessitam de interfaces gráficas padronizadas e contam com equipes menos experientes.

Já o *Tailwind*, permitiu a criação de uma interface moderna e altamente flexível. Contudo, a ausência de componentes prontos resultou em mais lacunas no atendimento às heurísticas, como a falta de uma seção de ajuda, de *feedback* durante o carregamento e da funcionalidade de autocompletar. As vantagens do *Tailwind* incluem a flexibilidade de design, que possibilita a criação de interfaces gráficas únicas, e um melhor desempenho, uma vez que apenas o CSS necessário é gerado. Suas desvantagens, por sua vez, englobam um maior tempo de desenvolvimento e uma curva de aprendizado mais acentuada. Portanto, o *Tailwind* é mais adequado para projetos que exigem designs personalizados, contam com equipes experientes e têm o desempenho como foco principal.

Em suma, ambos os *frameworks* são viáveis para o desenvolvimento de *e-commerces*, mas a escolha ideal dependerá do contexto específico de cada projeto. O *Bootstrap* se destaca em cenários onde a rapidez e a consistência são prioritárias, enquanto o *Tailwind* é superior para projetos que demandam designs únicos e otimizados. Para maximizar a usabilidade, é fundamental que ambos os protótipos implementem funcionalidades ausentes, como *zoom*, histórico de busca e uma seção de ajuda, independentemente do *framework* utilizado. Além disso, a realização de testes de usabilidade com usuários reais é recomendada para avaliar o impacto prático das escolhas de *framework* na experiência do usuário, especialmente em áreas como *feedback* durante o carregamento e suporte ao cliente.

8. Conclusão

Os resultados indicam que, embora ambos os *frameworks* permitam a criação de interfaces gráficas funcionais, o *Tailwind* demonstrou maior flexibilidade para atender a heurísticas específicas de *feedback* visual e customização, resultando em uma interface com maior potencial para uma experiência de usuário otimizada e mais adequada para *e-commerce*. Em contrapartida, mesmo apresentando maior rigidez, tendo pequenas falhas de usabilidade relacionadas à falta de diferenciação visual e à adaptação de componentes padronizados, durante a avaliação heurística o *Bootstrap* atendeu quase todos os itens das heurísticas do *checklist*, o que mostrou-se uma opção muito viável para desenvolver rapidamente interfaces gráficas responsivas e agradáveis.

É fundamental reconhecer as limitações desta pesquisa para uma interpretação precisa dos resultados. A avaliação baseou-se em protótipos *open-source* existentes, o que pode ter introduzido vieses relacionados às decisões de design e implementação originais. O foco primário na avaliação heurística, embora eficaz, não substitui testes com usuários reais, que poderiam fornecer *insights* mais aprofundados sobre a experiência prática e a satisfação do usuário. O escopo limitado de páginas avaliadas e a ausência de dados quantitativos também representam limitações.

Para aprimorar a compreensão sobre o impacto dos *frameworks* CSS na usabilidade de *e-commerce*, sugere-se a realização de testes de usabilidade com usuários reais, o desenvolvimento de protótipos controlados para eliminar vieses de design, uma análise de desempenho detalhada e a exploração de outras heurísticas e métricas de experiência do usuário. Para trabalhos futuros, pode-se incluir especialistas em UI/UX para avaliar esses frameworks com o *checklist* apresentado neste trabalho e contribuir com uma visão ainda mais abrangente sobre a escolha de frameworks CSS interfaces gráficas para *e-commerce.*

REFERÊNCIAS

ALSHAZLY, Amira. A. ELFATATRY, Ahmed. S. ABOUGABAL, Mohamed. S. “Detecting defects in software requirements specification”. Alexandria Engineering Journal”. 2014.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ISO 9241-210:2019: Ergonomia da interação humano-sistema - Parte 110: Princípios de diálogo. Rio de Janeiro: ABNT, 2019.

AZIZL, Maslina Abdul, et al. “Heuristic Evaluation Of E-Commerce Marketplace”. Universiti Teknologi MARA Cawangan Kedah. 2022.

BHAT, Kartik. “Ultimate Tailwind CSS Handbook”. Orange Education Pvt Ltd, AVATM. 2023.

BOEHM, Barry. BASILI, Victor R. “Software Defect Reduction Top 10 List”. University of Southern California, University of Maryland. 2001.

BROWN, Dan. “Designing Together: The collaboration and conflict management handbook for creative professionals”. New Riders, 2013.

BRYKCZYNSKI, B.; MEESON, R.; WHEELER, D. A. “Software Inspection: Eliminating Software Defects.” In: Proc. Annual Software Technology Conf., 1994.

CABALLERO, Rosa Yáñez Gómez. CASCADO, Daniel. SEVILLANO, José-Luis. “Heuristic Evaluation on Mobile Interfaces: A New Checklist”. Department of Computer Technology and Architecture, ETS Ingenieria Informatica, Universidad de Sevilla. 2014.

CARNEIRO, C. M. P. S. NETO, M. G. M.. “Frameworks de Aplicações Orientadas a Objetos – Uma Abordagem Iterativa e Incremental”. 2003.

CAROCA, Caio Regis. “Um Processo De Verificação E Validação Para Os Componentes Do Núcleo Comum Do Middleware Ginga”. 2010.

Documentação oficial do Bootstrap. Disponível no site: https://getbootstrap.com/docs/5.3/getting-started/introduction/. Acesso em: 4 dez. 2025

Documentação oficial do Tailwind. Disponível no site: https://tailwindcss.com/. Acesso em: 4 dez. 2025

FAGAN, M. Software pioneers. c´ap. “A History of Software Inspections”, New York, NY, USA: Springer-Verlag Inc., 2002.

FAGAN, Michael E. “Design and Code Inspections to Reduce Errors in Software Development”. IBM Systems Journal, 15, 182-211. 1976.

FERREIRA, A. “Usabilidade e Acessibilidade no design para a Web”. Faculdade de Belas Artes Universidade do Porto. 2008.

FONSECA, Ivan Francisco. “Como Melhorar A Experiência Do Usuário Com E-Commerce Através Da Usabilidade”. Faculdade de Tecnologia de Americana. 2014.

FRAIN, Ben. “Responsive Web Design with HTML5 and CSS3”. Packt Publishing, 4th Edition, 2022.

GERALDI, R. T. “SMartyCheck : uma técnica de inspeção baseada em checklist para diagramas de casos de uso e de classes da abordagem SMarty”. 2015.

GERALDI, R. T. , JUNIOR, E. O. “Defect Types and Software Inspection Techniques: A Systematic Mapping Study”. Journal of Computer Science. Universidade Federal de Maringá. 2017.

GIL, Antônio Carlos. “Como Elaborar Projetos de Pesquisa”. 2002.

GILB, T.; GRAHAM, D. “Software Inspection”. Boston, MA, USA: Addison-Wesley, 471 p., 1993.

GOMES, Luiz A. F. CORTÊS, Mario L. “Comércio Eletrônico: Uma Análise da Aplicabilidade de Modelos de Qualidade de Software”. UNICAMP . 2002.

Hatim Al Salmi. “Comparative CSS frameworks”. Umm Al-Qura University. 2023.

IEEE Standard Classification for Software Anomalies. 2009. Disponível em: https://ieeexplore.ieee.org/document/5234341 . Acesso em: 8 dez. 2025

ISO/IEC. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE. ISO/IEC 25000:2014*. Geneva: International Organization for Standardization, 2014. Disponível em : https://www.iso.org/standard/64764.html . Acesso em: 8 dez. 2025

JAIN, Ronak. SHRIVASTAVA, Vishal. PANDEY, Akhil. SHARMA, Aarti. “Modern Web Development using CSS & HTML”. 2024.

JAKOB, N. “Usabilidade na web: Projetando Websites com Qualidade”. GEN LTC. 2007.

JAKOB, N. “Usability Engineering”. Morgan Kaufmann. 1993.

JONES, C. Software defect-removal efficiency. IEEE Computer, v. 29, 1996.

KALINOWSKI, Marcos. SPINOLA, Rodrigo. “Introdução à Inspeção de Software - Aumentando a Qualidade Através de Verificações Intermediárias”. 2008.

KALINOWSKI, M. SPÍNOLA, R.O. TRAVASSOS, G.H. “In Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software”. On Simpósio Brasileiro de Qualidade de Software, Brasília, 2004.

KUMAR, Bimal Aklesh. “Mobile-First Usability Guideline for Responsive E-Commerce Websites”. Fiji National University. 2022.

LEAVITT, Michael O. SHNEIDERMAN, Ben. “Research-Based Web Design & Usability Guidelines”. 2006.

Len Bass, Paul Clements, Rick Kazman. “Software Architecture in Practice third edition. Addison-Wesley Professional”. 2012.

MATOS, Christine Martins De; Oliveira. et al. “Utilização de frameworks front-end CSS no desenvolvimento de sistemas web”. FEPEG. 2016.

MAZZA, Lucas. “Desenvolvimento Web HTML e CSS”. Casa do código. 2021.

MAZZA, Lucas. “HTML5 E CSS3 - Domine a Web do futuro”. Casa do Código. 2014.

MEIRT, J. O. . “The Little Book of HTML/CSS Frameworks”. O’Reilly Media, Inc. 2015.

MELO, Vagner Claudino De. “Assinatura De Documentos Por Meio Da Autenticação Digital”. Centro Universitário Atenas. 2019.

MOHD, Tauheed Khan. THOMPSON, Jordan. CARMINE, Amedeo. REUTER, Grant . “Comparative Analysis on Various CSS and JavaScript Frameworks”. Journal of Software. 2022.

MORAES, João Tiago Telles de. “E-commerce: Um estudo exploratório da percepção dos consumidores sobre a qualidade de serviço”. Universidade de Brasília. 2017.

NAIK, Kshirasagar. TRIPATHY, Priyadarshi. “Software Testing And Quality Assurance: Theory and Practice”. 2008.

NIELSEN, Jakob. “Heuristic Evaluation Of User Interfaces”. Technical University of Denmark. 1990.

NIELSEN, Jakob. MACK, Robert L. “Usability Inspection Method”, New York, Wiley, 1994.

PEREIRA, Geremias. “Sistemas De Informação Em Comércio Eletrônico”. Faculdade de Tecnologia de Americana. 2013.

PERGENTINO, Ana Carolina dos Santos. “Avaliação Heurística em um Contexto Real”. Universidade de Brasília. 2019.

PRESSMAN, R. S. MAXIM, B. R. “Engenharia de Software: uma abordagem profissional”. AMGH Editora Ltda. 2021.

SANDHOF, Karen. FILGUEIRAS, L. V. L. “Defeitos de software como erros humanos”. In II Workshop Um Olhar Sociotécnico sobre a Engenharia de Software WOSES. 2006.

SANTOS, Fábio Henrique Oliveira dos. “Apoio ao Processo De Inspeção De Usabilidade Para Aplicações De Software”. 2011.

SANTO, Felipe do Espirito. SOUZA, Leonardo Patrocínio. “Comparativo Entre Frameworks De Css Bootstrap E Bulma Para Desenvolvimento De Projetos Web”. Faculdade de Tecnologia de Taquaritinga (Fatec). 2020.

SCANDURA, Leonardo. OLIVEIRA, Lígia. “Avaliação Heurística De Sites De Ecommerce Estudo De Caso - Sites Amazon E Kabum”. Instituto Federal Catarinense. 2020.

SCHERER, Noemi Pereira. “Avaliação Heurística E Teste De Usabilidade Para Softwares De Design De Interiores”. Universidade Tecnológica Federal Do Paraná. 2018.

SHNEIDERMAN, Ben; PLAISANT, Catherine. “Designing the User Interface: Strategies for Effective Human-Computer Interaction”. 6. ed. Pearson, 2016.

SILVA, AA. de A. P. “Design Responsivo: Técnicas, Frameworks e Ferramentas”. UFRJ. Rio de Janeiro – RJ. 2014.

SILVA, Jean Michel Galindo da. “Avaliação Da Usabilidade De Sites De E-Commerce Com Análise Envoltória De Dados”. PUC Rio Pontifícia Universidade Católica do Rio de Janeiro. 2021.

SOMMERVILLE, Ian. “Engenharia de Software”. Escola Politécnica da Universidade de São Paulo (EPUSP). 2013.

SOUSA, Arley Gomes de. “Avaliaí: Um Acervo De Checklists De Heurísticas Para Facilitação Da Avaliação Do Design De Interfaces Digitais”. Universidade Federal Do Ceará. 2022.

SPURLOCK, Jake. ”Bootstrap”. O’Reilly Media, Inc. 2013.

TOMAZINI, Marcos. LOPES, Luiz Fernando Braga. “Web design responsivo - Bootstrap”. Faculdade Cidade Verde (FCV). 2015.

TURAN, Bülent Onur. ŞAHİN, Kemal. “Responsive Web Design And Comparative Analysis Of Development Frameworks”. Mimar Sinan Fine Arts University. 2017.

VERCEL. *Ecommerce Template with Crystallize and Remix*. Disponível em: https://vercel.com/templates/ecommerce/ecommerce-crystalize-remix. Acesso em: 7 nov. 2025

WELLER, E. F. “Lessons from Three Years of Inspection Data (software development).” IEEE Software, v. 10, 1993.

WOHLIN, Claes. “Experimentation in Software Engineering”. 2024.

YOUNG, Michal. “Software Inspections”. John Wiley & Sons. 1999.