UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRENER NOSSE QUEVEDO

ATOMIC DESIGN NO CONTEXTO DE DESENVOLVIMENTO: um Relato de Experiência

CAMPINA GRANDE - PB

2023

BRENER NOSSE QUEVEDO

ATOMIC DESIGN NO CONTEXTO DE DESENVOLVIMENTO: um Relato de Experiência

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. João Arthur Brunet Monteiro

CAMPINA GRANDE - PB

2023

BRENER NOSSE QUEVEDO

ATOMIC DESIGN NO CONTEXTO DE DESENVOLVIMENTO: um Relato de Experiência

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. João Arthur Brunet Monteiro

Orientador – UASC/CEEI/UFCG

Professora Dr. Dalton Dario Serey Guerrero

Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em 14 de fevereiro de 2023.

CAMPINA GRANDE - PB

ABSTRACT

The importance of Atomic Design in software development is vaguely discussed in academic circles, but it is important to emphasize the value of a design system that establishes guidelines for creating user interfaces and facilitates the development of web applications. Atomic Design allows for the creation of a component system that follows a well-defined hierarchy, resulting in the construction of flexible interfaces. However, this flexibility and scalability may not be as simple to implement and have some limitations. In order to observe how Atomic Design can be used as a methodology that facilitates the way of thinking and organizing projects, this work reports the experience of adapting the guidelines of Atomic Design in a confidential e-commerce application; this is an internal project for managing and integrating APIs of a private company. Additionally, some general evaluation criteria are proposed for analyzing the strategy. The overall objective is to highlight success factors, difficulties, and negative aspects of the model in view of its practical application.

Keywords: *Design system; Atomic Design; Interface Design* ; *Web Application*

ATOMIC DESIGN NO CONTEXTO DE DESENVOLVIMENTO: um Relato de Experiência

Brener Nosse Quevedo*
brener.quevedo@ccc.ufcg.edu.br
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

João Arthur Brunet Monteiro
joao.arthur@computacao.ufcg.edu.br
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil

RESUMO

A importância do *Atomic Design* no desenvolvimento de *software* é pouco discutida no meio acadêmico, mas é importante destacar o valor de um sistema de *design* que estabelece diretrizes para criar interfaces de usuário e facilita o desenvolvimento de aplicações *web*. O *Atomic Design* permite construir um sistema de componentes que seguem uma hierarquia bem definida, resultando na construção de interfaces flexíveis. Porém, essa flexibilidade e escalabilidade podem não ser tão simples de serem implementadas e possuem algumas limitações. Diante disto, com o intuito de observar de que maneira o *Atomic Design* pode ser utilizado como uma metodologia que facilita a forma de pensamento e organização dos projetos, este trabalho relata a experiência de adaptar as diretrizes do *Atomic Design* em uma aplicação confidencial de um *e-commerce*; trata-se de um projeto interno de gerenciamento e integração de APIs de uma empresa privada. Além disso, são propostas alguns critérios gerais de avaliação para realização da análise da estratégia. O objetivo geral é destacar fatores de sucesso, dificuldades e pontos negativos do modelo diante da observação de sua aplicação na prática.

KEY WORDS

Design system; *Atomic Design*; *Design* de Interfaces; Aplicação *Web*

ACM Reference Format:
Brener Nosse Quevedo and João Arthur Brunet Monteiro. 2023. *ATOMIC DESIGN* NO CONTEXTO DE DESENVOLVIMENTO: um Relato de Experiência. In *Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação, UFCG).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

A constante aparição de novas tecnologias de software está mudando a percepção dos desenvolvedores sobre padrões de código. Eles buscam reduzir a repetição de tarefas e aumentar a robustez e flexibilidade do código a longo prazo, além de garantir a escalabilidade do sistema de design de um projeto em níveis de abstração mais elevados. A chegada de *framework*s como *React*, *Bootstrap* e *Angular*, de bibliotecas de componentes, como *Material UI*; e ferramentas de prototipação como *Storybook*, reduzem o tempo de desenvolvimento em comparação a um projeto feito do zero [4].

Dentre os problemas enfrentados no desenvolvimento de aplicações *web*, se destacam a manutenção de projetos de grande escala, a responsividade em diferentes navegadores ou dispositivos, e o desempenho. A não adoção de boas práticas de códigos está entre os principais motivos para a ocorrência desses problemas. Uma dessas práticas é a adoção de um *Design System*, que é um guia muito importante para que o sistema seja concebido de forma consistente e padronizada.

A consistência de um *Design System* é um desafio devido ao difícil equilíbrio entre design e funcionalidade, e é comum que essa dificuldade seja causada pela falta de comunicação entre *designers* e programadores. Em contextos de desenvolvimento de sites *web*, a escolha estética de um produto pode afetar a interface do usuário. Se a concepção do design for inadequada, isso pode causar problemas para os desenvolvedores e prejudicar a experiência dos usuários. É importante entender que abordagens que priorizam somente a funcionalidade podem desconsiderar conceitos importantes, como a reutilização de estruturas em todo o projeto, o que é um erro comum entre programadores iniciantes. Sem uma orientação para a criação de novos componentes, como eles se encaixam em novas funcionalidades e se são realmente necessários, pode ser difícil lidar com questões como expansão do projeto, manutenção, documentação e testes.

A evolução do pensamento a respeito do desenvolvimento *web* visa uma interação mais próxima entre produto e consumidor [8]. Como dito por Mark Boulton, a criação de uma boa experiência de usuário é independente do conteúdo da obra, não é possível desenvolver a obra sem que haja uma estrutura bem definida e diretrizes a serem seguidas, pois para que o projeto tenha sucesso, é importante entender como ele foi feito e não somente o que ele é [1]. Para tal, novos métodos de desenvolver interfaces acessíveis e escaláveis vem sendo propostas dentro desse contexto; o *Atomic Design* (AD) se propõe a utilizar modelos da química como forma de estruturação de seus componentes, importando conceitos desta área para o desenvolvimento de *software*.

Enquanto uma abordagem de funcionalidade organiza os componentes baseando-se em sua relação de contexto, o *Atomic Design* é um método de organização de código que se concentra no design e na interface do usuário para construir uma hierarquia de componentes gradativamente complexos. Não se pode afirmar qual abordagem é a melhor, pois o *Atomic Design* pode ser adaptado para trabalhar em conjunto com outros métodos, visando resolver problemas de redundância e má organização de componentes, e estabelecer um sistema de *design* exclusivo para o projeto.

A utilização do *Design* Atômico para o contexto do nosso projeto, se dá pela necessidade de criarmos um sistema consistente e escalável de componentes de interface. Tendo em vista que a hierarquia de elementos proposta pelo *Atomic Design* segue um padrão que procura reduzir a quantidade de trabalho desnecessário, pois os átomos (primeira etapa na hierarquia do AD), como por exemplo

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação, UFCG), 2023. https://doi.org/10.1145/nnnnnnn.nnnnnnn

botões, *containers* e títulos, podem ser rearranjados de forma a criar diferentes entidades (moléculas e organismos).

Uma maneira simples de exemplificar o funcionamento da hierarquia do AD é considerarmos uma uma barra de navegação; esta pode possuir um título (átomo), um agrupamento de *templates* (molécula), dispostos dentro de um *container* (átomo), e a união desses elementos formam um organismo. Tendo isso em mente, é possível repetir esse passo a passo para outros elementos como o rodapé, barra lateral, cabeçalho, etc. A partir do momento que definimos os organismos, podemos estruturar seções de componentes através de *templates*, resultando no esqueleto geral das telas, porém só podemos sair da abstração para um exemplo concreto ao adicionarmos dados reais que preencham as lacunas do *templates*, configurando assim a última etapa do *Atomic Design*, as páginas. Por mais que tenhamos definido isso como etapas, Brad Frost deixa claro que isso não se trata de um passo a passo, mas sim uma maneira de conseguirmos visualizar novos elementos e funcionalidades ao projeto no futuro com mais eficiência [5].

Conforme afirma Frost, "O *Atomic Design* não é um processo linear, mas sim um modelo mental para nos ajudar a pensar em nossas interfaces de usuário como um todo coeso e um conjunto de partes ao mesmo tempo" [5]. Portanto podemos afirmar que a abordagem atômica propõe resolver os problemas listados anteriormente, uma vez que suas ocorrências são de forma pontual no projeto, podem ser isolados e estudados, já que pode ser decomposto em tarefas menores para uma solução mais eficiente, contornando problemas sem prejudicar a estrutura geral do produto.

Este trabalho tem como objetivo apresentar a experiência da aplicação de conceitos do *Design* Atômico no contexto da interface que foi desenvolvida. Para atingir esse objetivo, analisamos a facilidade de implementação; avaliamos as dificuldades apresentadas, e verificamos a como ele lida em um contexto de uma aplicação de médio porte em produção. Deste modo, somos capazes de compreender melhor como a metodologia pode ser empregada para grupos maiores de desenvolvedores. A aplicação foi realizada dentro do sistema interno de uma empresa de *e-commerce*, buscando verificar a aplicabilidade do Design Atômico, analisando suas vantagens e desvantagens para uma aplicação *front-end* de gerenciamento de infraestrutura interna, uma vez que, se bem sucedido, será adotado pelos demais integrantes de outras equipes da empresa. Considerando o contexto da aplicação, levantamos alguns questionamentos que responderemos ao longo da pesquisa: Como é relacionada a flexibilidade e adaptabilidade dos componentes e como isso implica na redução de código escrito? É possível construir muitos componentes com pouco código? Quais as dificuldades relacionadas a curva de aprendizado e a prototipação de telas? Como lidar com alterações bruscas na interface?

A contribuição deste trabalho é relevante para área por apresentar um tema pouco abordado em pesquisas da área de computação, em específico desenvolvimento de *software*, além de trazer um relato de experiência que apresenta as vantagens e desvantagens do uso do *Design* Atômico para aplicações *web*. Os dados foram coletados por experimentos realizados internamente, que serão explicados posteriormente, e informações confidenciais e sensíveis serão respeitadas. Foram observadas percepções mistas sobre a implementação prática da abordagem atômica, bem como problemas na curva de aprendizado sobre a estratégia e na configuração de ferramentas de suporte para testes, documentação e prototipação, assim como adaptações necessárias da metodologia para o projeto. Com relação aos pontos positivos, obtivemos grande reusabilidade e rápida construção de componentes, código legível, e manutenível. A fim de lidar com integrações futuras, vale ressaltar que não é uma abordagem que por mais que seja direta ao ponto, é necessário que o desenvolvedor tenha certa experiência com desenvolvimento *web*, os critérios necessários serão discutidos no decorrer do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Estruturamos a fundamentação teórica de modo a sustentar o desenvolvimento de uma aplicação, utilizando as diretrizes do *Atomic Design*, com foco em uma abordagem que analise as qualidades do modelo como um *Design System*, impactos positivos e negativos dentro da aplicação e quais benefícios e dificuldades essa estratégia trará em longo prazo.

2.1 *D*esign System

Pode-se entender *Design System* como um conjunto de diretrizes, recursos e padrões que orientam a criação de interfaces consistentes e eficientes. Ele inclui elementos visuais, tipografia, cores, espaçamentos, interações e outros aspectos de design. Um *Design System* é criado para ser reutilizável e escalável, ajudando equipes de design a criar interfaces de maneira mais eficiente, sem sacrificar a qualidade ou consistência. Além disso, um *Design System* ajuda a garantir a padronização de elementos em uma marca ou aplicação, melhorando a usabilidade e a experiência do usuário. Um *Design System* é a referência principal internamente em projetos, pois permite que equipes criem produtos melhores de acordo com um escopo pré-definido, tornando o design reutilizável, a capacidade de permitir que o trabalho seja reduzido através da reusabilidade permite que o produto seja mais escalável, sendo esses um dos pilares do sistema de design [7].

2.2 *Atomic Design*

Antes de abordarmos sobre o *Atomic Design*, é necessário entendermos que ele tem como base princípios do *design* modular. Conceitos como reusabilidade, flexibilidade, partes intercambiáveis, são comuns de serem observados na abordagem modular e logo foram adaptadas para o desenvolvimento de *softwares*. O *design* modular apresenta-se como uma estratégia para construir sistemas ou produtos complexos, a partir de subsistemas que podem ser desenvolvidos individualmente, mas que funcionam como um conjunto integrado. Cada módulo representa uma unidade funcional juntamente com os demais módulos, que contribuem para o funcionamento da estrutura maior. O *Atomic Design* (AD) é uma abordagem e metodologia para criar sistemas flexíveis, escaláveis de componentes de interface, que se utiliza de conceitos da química para poder estabelecer uma hierarquia entre seus elementos, classificando-os, em ordem crescente, como átomos, moléculas, organismos, *templates* e páginas [5], a Figura 1 ilustra a hierarquia dos elementos.

A seguir definiremos cada um dos elementos citados anteriormente, conforme apresentado por Frost em [5]. Átomos são a menor unidade funcional da matéria e não podem ser decompostos além disso sem perder seu significado, eles representam componentes que não podem ser decompostos sem perder sua funcionalidade. Por

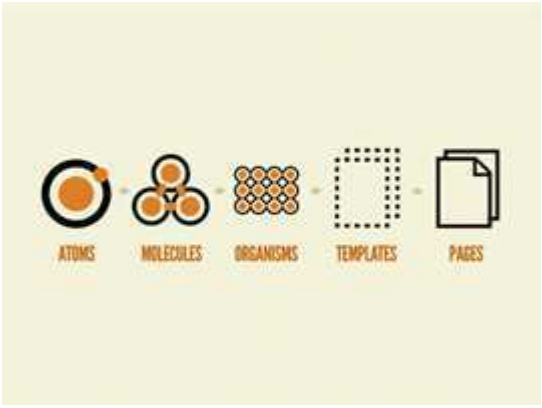

Figura 1: Hierarquia dos Elementos. Fonte: Atomic Design, Brad Frost 2016

serem simples e pequenos, podem ser reutilizados diversas vezes, um exemplo de átomo em um *Design System* são os botões, como podemos ver na Figura 2, cada botão do agrupamento é um átomo. Um botão pode ser utilizado tanto como elemento de submissão de um formulário como de adição de elementos em uma lista;

Figura 2: Exemplo de Átomo: Botão *Default*. Fonte: Elaborado pelo Autor

Moléculas são a união de dois ou mais átomos, são componentes simples de interface que funcionam como uma unidade. Dentre os exemplos de molécula podemos ter um agrupamento de botões, e um campo de texto de pesquisa com um botão de submissão, como podemos ver na Figura 3.

Figura 3: Exemplo de Molécula. Fonte: Elaborado pelo Autor

Organismos são um agrupamento composto por moléculas e átomos, são relativamente mais complexos do que as etapas anteriores por possuir um escopo dentro do contexto que está inserido, elevando seu grau de funcionalidade. Temos como exemplo de organismos as barras de navegação com *links*, botões, título da página, formulários de cadastro de endereços, como visto abaixo na Figura 4.

Figura 4: Exemplo de Organismo. Fonte: Elaborado pelo Autor

Templates são a junção de organismos em seções ou páginas, representando um protótipo da estrutura do conteúdo. Páginas, por fim, são o estágio final, são instâncias específicas dos *templates* a fim de demonstrar como eles se comportam com dados reais de forma a testar a resiliência do *Design System* [5], como um exemplo temos a Figura 5 a seguir

Figura 5: Exemplo de Página da Aplicação. Fonte: Elaborado pelo Autor

A princípio existem variáveis que não são definidas internamente pelo *Atomic Design*, como por exemplo unidades de medida de espaçamento, fonte e cores, que são partes essenciais de uma UI, para isso, o conceito de *Design Token* aparece como resposta. Posteriormente Frost denomina isso como as partículas subatômicas dentro do contexto atômico. Geralmente são variáveis definidas no início do projeto, e podem ser exportadas para diferentes formatos e plataformas, permitindo que as equipes de desenvolvimento e *Design* possam trabalhar de forma colaborativa [5].

3 METODOLOGIA

Nesta seção, abordamos a metodologia escolhida para o desenvolvimento e análise da proposta. Tendo em vista os objetivos a serem alcançados, em que pretendemos observar e compreender um fenômeno, optamos por uma pesquisa qualitativa, neste contexto, consideramos pertinente uma pesquisa aplicada, uma vez que buscamos gerar conhecimento para solucionar problemas, nesse caso específico o desenvolvimento da interface, e exploratória, uma vez que esse tipo de pesquisa busca familiaridade com um problema, trazendo o aprimoramento de ideias. [2, 6].

Nosso instrumento de pesquisa é a interface *web* que foi desenvolvida durante o projeto, utilizando o *Atomic Design*. Este artigo

apresenta um relato de experiência que servirá como narrativa que legitima a descoberta do autor sobre o material abordado. Um relato de experiência "Configura-se como narrativa que, simultaneamente, circunscreve experiência, lugar de fala e seu tempo histórico, tudo isso articulado a um robusto arcabouço teórico, legitimador da experiência enquanto fenômeno científico." [3]

3.1 Desenvolvimento da aplicação

Foi desenvolvida uma interface *web* utilizando *React*, uma biblioteca *Javascript* utilizada para construir componentes de interfaces de usuário e gerenciar o estado de aplicações de forma eficiente. O critério de escolha do *framework* se deu pela afinidade do desenvolvedor com a ferramenta e pela popularização do *Atomic Design* pela comunidade. A aplicação é uma *dashboard* de manutenção e criação de entidades que fornece infraestrutura básica para o usuário, sendo uma intermediadora de duas APIs externas, que fornecem os dados necessários para a criação e manutenção desses modelos, e possibilita a realização de todas as operações básicas de um CRUD (criar, ler, atualizar e excluir). Para o desenvolvimento deste projeto, foi utilizado o guia de estilos do *Material UI* como referência para cores, espaçamento e funcionalidades de alguns componentes, pois é uma das bibliotecas de componentes *React* baseada no *Google Material Design* mais populares atualmente e com isso nos aproveitamos do *Atomic Design* durante o processo de estruturação dos componentes. Deste modo, fomos capazes de analisar como a ferramenta se comporta de acordo com a expansão do projeto e quais dificuldades foram apresentadas na implementação.

3.2 Coleta e análise dos dados

Os dados foram analisados a partir de diretrizes que o *Atomic Design* define, com base nos critérios gerais de avaliação definidos pelo autor, antes do processo de desenvolvimento da aplicação; critérios esses que serão indicadas no decorrer da sessão. Como se trata de um relato de experiência, analisamos alguns comportamentos do AD, que serão discutidos na seção seguinte. Por se tratar de uma análise qualitativa, os critérios de avaliação abordados como reusabilidade, consistência, facilidade de manutenção, acessibilidade e flexibilidade foram analisadas a partir de revisões de código e ferramentas de análise de código, como *ESLint*.

Conforme o projeto foi apresentado durante reuniões de demonstração e *code review*, a equipe forneceu alguns *feedbacks* sobre a estrutura do projeto e o *design* da interface, que foram registrados através de atas internas da equipe. O tempo de duração do processo de entregas e revisões durou aproximadamente um mês e meio. O grupo de revisores era constituído por 5 desenvolvedores, sendo 3 programadores sênior, 1 líder técnico e 1 gerente de projeto. Iremos pontuar alguns comentários que achamos pertinentes sobre as revisões feitas na seção a seguir. Por se tratar de uma aplicação que possui acesso restrito, não temos liberdade suficiente para discorrer sobre a especificação interna da aplicação, entretanto traremos um apanhado de comentários feito por desenvolvedores que participaram do processo de revisão do código e *design* na subseção de considerações gerais.

3.2.1 Critérios Gerais de Avaliação. Tendo em vista os objetivos que pretendemos alcançar, após a realização do projeto fomos capazes de determinar critérios que julgamos interessantes para realizar a avaliação do trabalho, são elas:

- **Reusabilidade:** Significa a capacidade de um utilizar componentes, códigos ou funcionalidades já criados em outra parte do projeto, sem precisar recriá-lo. Isso pode ser observado através de bibliotecas de componentes, utilização de padrões de *Design* que permitam a utilização do mesmo código em diferentes contextos. A reusabilidade permite que tenhamos um aumento da eficiência no desenvolvimento, garante a consistência funcional e visual do sistema e reduz a quantidade de código duplicado.
- **Consistência:** Trata-se da capacidade de garantir que a aparência e comportamento dos elementos da interface se mantenham uniforme e coerente em todo sistema.
- **Manutenibilidade:** Trata-se da facilidade de um sistema ser compreendido e modificado sem que isso comprometa o restante da aplicação, o objetivo é garantir que o projeto seja atualizado e corrigido com eficiência e segurança, independente de quem trabalha na modificação, ou quanto tempo se passou desde a concepção do projeto.
- **Acessibilidade:** Um sistema acessível em desenvolvimento de *softwares* é projetado para ser utilizável por todos, incluindo tecnologias de acessibilidade e implementação de padrões de acessibilidade, como WCAG (*web Content Accessibility Guidelines*) [9]. É fácil de usar e navegar, independentemente da habilidade do usuário com as tecnologias. Isso garante acesso igualitário ao conteúdo e funcionalidades da aplicação.
- **Flexibilidade:** Um sistema flexível implica que um site foi concebido com sensatez suficiente para lidar com diversos tipos de dados e adversidades, garantindo robustez e resiliência do sistema.

4 RESULTADOS E DISCUSSÕES

Nesta seção apresentamos os pontos significativos que foram identificados durante o desenvolvimento do projeto com o *Atomic Design*. Discutimos também seus pontos fortes, especificidades e dificuldades durante todo processo, além de perspectivas futuras sobre a utilização da metodologia.

4.1 Processo de Desenvolvimento da aplicação

O principal desafio encontrado durante o desenvolvimento do projeto foi montar uma interface que seria utilizada internamente entre as equipes de infraestrutura e que possuiria grande potencial de expansão para o futuro, como o projeto foi desenvolvido por apenas uma pessoa, foram encontradas certas dificuldades em definir por onde começar o projeto. Entretanto, como possuía uma interface datada, que era utilizada anteriormente, foi possível obter algumas referências de telas, que auxiliaram no processo de criação.

A aplicação oferece uma *dashboard* para manutenção e criação de uma infraestrutura básica para o usuário, e atua como intermediária entre duas APIs externas, que fornecem os dados necessários para a criação e manutenção dos modelos. Ela permite que o usuário realize todas as operações básicas de um CRUD, a Figura 6 exemplifica de forma geral a arquitetura do projeto, sendo apenas a interface *front-end* o alvo do nosso estudo.

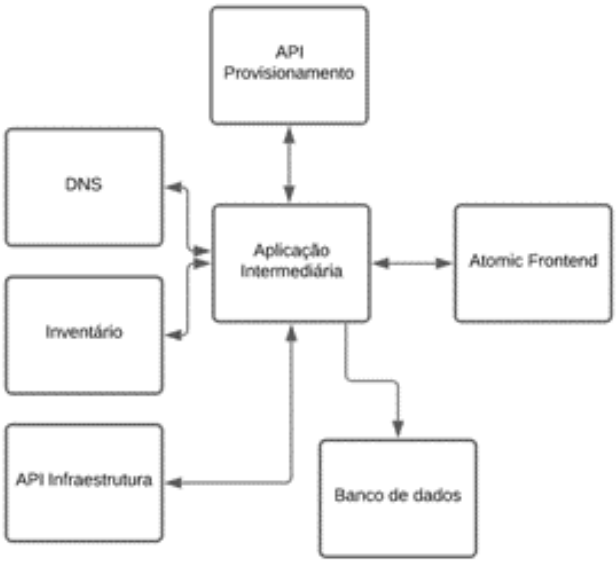

Figura 6: Exemplo de Página da Aplicação. Fonte: Elaborado pelo Autor

Considerando que esse projeto foi desenvolvido para uma empresa privada, e que há um contrato de sigilo com relação ao material desenvolvido e o código fonte, apresentamos algumas telas modificadas ao longo do texto, a fim de preservar informações sensíveis. Levando em consideração que o objetivo do artigo é mostrar como *Atomic Design* se comportou diante a especificação e a construção da interface, essas telas serão suficientes para a exemplificação.

A teoria por trás do *Atomic Design* é de fácil compreensão, porém sua aplicação prática requer tempo e atenção a detalhes, como ferramentas externas, bibliotecas e *frameworks* que precisam seguir os passos propostos pela metodologia, a abordagem segue o modelo de construção baseado no princípio DRY (*Don't Repeat Yourself*), isso implica que durante a fase de criação dos átomos, os componentes que possuem variações são pensados de antemão, de forma a evitar que haja a duplicação de código.

Todos os componentes foram feitos com a ajuda da biblioteca de estilização chamada *styled-components*, ela permite que possamos escrever códigos CSS em arquivos *Javascript*. Porém não foram utilizadas bibliotecas específicas de componentes como *Material UI* ou *Ant Design* para criarmos telas, o uso dessas bibliotecas se deu apenas como guia de referência de estilos e funcionalidades, o que resultou em mais tempo despendido no início do projeto, já que as primeiras duas etapas de codificação de átomos e moléculas são a base da aplicação porque fornecem a estrutura básica para a construção de componentes mais complexos.

Levando em consideração a acessibilidade da plataforma para pessoas com deficiência, foi necessário realizar um estudo a fim de seguir as normas de acessibilidade e importá-las para dentro da nossa aplicação, sendo algumas delas o uso de *tags* HTML semanticamente e sintaticamente corretas e acesso às funcionalidades da plataforma através do teclado. Sobre a construção de organismos e unidades mais complexas, um exemplo que pode ser mencionado foi um organismo requisitado pelos usuários, no qual um *input* do tipo *select* permitia a pesquisa, seleção e visualização de países, de um campo para outro, conforme podemos ver na Figura 7.

Figura 7: componente do tipo Organismo que funciona com seleção e filtragem de dados. Fonte: Elaborado pelo Autor

Durante o processo de desenvolvimento foi possível observar que, por mais que utilizar uma biblioteca de componentes como *Material UI* seja ótima alternativa para entregas rápidas, o processo de teste, documentação e estilização em cima dos estilos padrões da biblioteca, se tornam tarefas difíceis, uma vez que é necessário dominar os recursos que a biblioteca provê, bem como utilizá-los a fim de que eles possuam o comportamento desejado. Por mais que a abordagem atômica adicione uma camada extra de complexidade para nossa aplicação, não houve um aumento na quantidade de código gerado, pois como podemos definir variantes de átomos em nível de estilização, como visto na Figura 8, podemos alternar entre estilos através de propriedades, como vimos nas Figura 2, Figura 9 e Figura 10 e a partir dessa decomposição em elementos menores e independentes, conseguimos isolar as dificuldades de abstração em outros componentes.

```
const ButtonWrapper = (props) => {
    switch (props.variant) {
        case ButtonsVariants.outlined:
            return <ButtonOutlined {...props} />;
        case ButtonsVariants.link:
            return <ButtonLink {...props} />;
        default:
            return <Button {...props} />;
    }
};
```

Figura 8: Exemplo do Código do Botão atômico. Fonte: Elaborado pelo Autor

Uma das limitações que foram encontradas é o fato de que o projeto foi desenvolvido por apenas uma pessoa, logo não houve observação sobre como outros desenvolvedores lidariam com a metodologia empregada, talvez em um trabalho futuro, e na expansão do projeto para outras equipes, isso possa ser observado com mais clareza.

Foi observado que, após a primeira entrega, o projeto respondeu bem em ambiente de produção. Embora seja somente utilizado em monitores *widescreen* (1920:1080), ele foi codificado de forma que

Figura 9: Botão tipo *Link*. Fonte: Elaborado pelo Autor

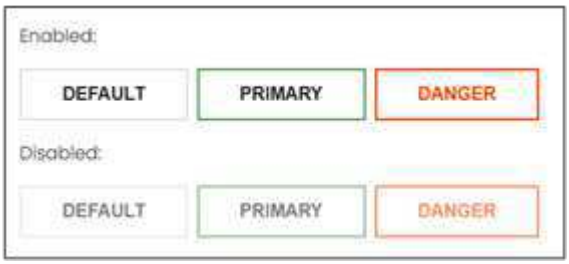

Figura 10: Botão *Outlined* Fonte: Elaborado pelo Autor

suporte a responsividade para celulares também. Vale ressaltar que a estrutura de responsividade utiliza, em sua maioria, o *Display Grid*, que foi o que mais se encaixou dentro da proposta do projeto, pois ele se caracteriza como um átomo dentro de nossa hierarquia.

Apesar do escopo da interface ser simples, foram necessárias a produção de cerca de 20 telas. Com exceção da página de *login* do usuário, todas as outras páginas seguem um padrão de *layout* com uma sidebar e o conteúdo da página, como mostra a Figura 5. As páginas de formulário, de forma geral, tem componentes em comum de *input* do tipo *text*, *select* e *checkbox*, a diferenciação delas se dá pelo contexto e pela disposição dos elementos em tela.

Levando em consideração que muitas das telas possuem os mesmos componentes, e apresentam pouca variação em estilização, o processo de desenvolvimento dessas telas foi mais rápido, o fluxo de criação seguia o fluxo de coleta de requisitos sobre a tela, seleção e estruturação na árvore de elementos do HTML, dos componentes necessários, integração com serviços externos e utilidades, e por fim gerenciamento de estados da página como do contexto *React*.

4.2 As desvantagens

Levando em consideração que todos os problemas citados a seguir podem ser resolvidos através de planejamentos e capacitações das partes diante da metodologia e ferramentas utilizadas, a metodologia de forma geral não apresenta problemas graves que impeçam a escalabilidade do design do projeto.

O *Atomic Design* adiciona uma camada de complexidade ao projeto e consome muito tempo: decompor elementos em átomos, moléculas e organismos é uma tarefa trabalhosa e demanda tempo, tanto de entendimento, quanto de gerenciamento. É muito importante que os desenvolvedores estejam a par do guia de estilo que está sendo seguido, tal como, ele deve ser atualizado para que não entre em desuso e se torne obsoleto. A seguir será elencado quais foram as desvantagens observadas ao utilizarmos o *Atomic Design* no projeto:

- **Complexidade:** Criar componentes atômicos e compostos pode ser complexo e demorado, especialmente se a equipe de desenvolvimento não estiver familiarizada com a metodologia, bem como a adaptação da estratégia para projetos com requisitos específicos.
- **Sobrecarga em átomos e moléculas:** adicionar suporte a vários tipos de variantes para elementos como botões e *input*s, pode afetar negativamente a legibilidade dos componentes, pois isso implica em uma sobrecarga de propriedades e estilos aos átomos e moléculas, levando a dificuldades de manutenção.
- **Manutenção adicional:** por mais que *Atomic Design* permita modificações na sua estrutura, ele deve ser consistente durante toda a fase de desenvolvimento, sua manutenção requer atenção especial, já que uma modificação generalizada terá consequências em outros componentes de maior hierarquia. Porém isso não é um ponto totalmente ruim, visto que modificações na camada dos átomos podem ser feitas de maneira que não haja efeito colateral no restante do *layout*, sendo assim, é importante que a documentação se mantenha atualizada a fim de não gerar inconsistências para novos componentes adicionados no futuro.
- **Uso excessivo de componentes:** O *Atomic Design* pode levar a reutilização excessiva de componentes, implicando em uma interface monótona e repetitiva. Podemos contornar isso com a utilização de um guia de estilo definido pelo time de design e desenvolvedores.
- **Importação de componentes:** como temos uma estrutura de pastas com mais etapas, a navegação pela subpastas para a importação de componentes dificulta o desenvolvimento e prejudica a legibilidade do código, isso é contornável através da utilização de *webpacks* e padronização das nomenclaturas de componentes.

4.3 As vantagens

Antes de definirmos as vantagens é importante ressaltar que algumas tomadas de decisões em relação a aplicação da estratégia são baseadas em experiências anteriores e também por recomendações feitas pela comunidade de desenvolvedores *web* que utilizam a abordagem atômica no cotidiano, com base nisso, iremos discorrer a seguir sobre os benefícios que foram apresentados durante sua aplicação:

- **Responsividade:** A estruturação em átomos, moléculas, organismos e páginas torna mais fácil a construção de interfaces adaptáveis a diferentes tamanhos de tela, sem que seja necessário refazer o design do zero para cada tamanho. Além disso, o *Atomic Design* incentiva a criação de componentes

genéricos e independentes do tamanho de tela, o que facilita a criação de *layouts* responsivos.

- **Colaboração:** *Atomic Design* proporciona uma linguagem comum entre *designers* e programadores, isso resulta na aproximação entre essas partes, o que permite uma tomada de decisão mais coerente sobre o futuro da aplicação, evitando a ocorrência de conflitos de *design* e funcionalidade.
- **Estrutura hierárquica:** A maneira como é realizada a estruturação torna o processo de *redesign* mais simples, é possível continuar desenvolvendo em camadas de alto nível e mesmo assim refatorar a base (átomos e moléculas), sem gerar inconsistências graves nas páginas.
- **Documentação:** Por estabelecer uma estrutura clara e organizada de componentes, dividido em etapas, a documentação desses elementos se torna uma tarefa mais simples, principalmente quando integramos o projeto com ferramentas de suporte como *Storybook*.
- **Legibilidade:** por termos uma estrutura de pastas claramente definida, tornando o processo de familiarização com o projeto mais intuitivo. É comum utilizar bibliotecas de tipagem, como *PropTypes* ou *Typescript*, que adicionam tipagem estática ao *Javascript*. Com a utilização dessas bibliotecas para construir componentes, temos mais segurança ao atribuir propriedades a estes componentes e facilitamos a documentação. Assim permitimos que novos desenvolvedores entendam como os componentes devem ser usados de forma consistente.
- **Prototipação:** O *Atomic Design* permite que seja mais fácil a prototipação das telas, porque ele fornece uma estrutura de componentes modular. Cada componente pode ser criado e testado independentemente, tornando possível o trabalho colaborativo entre os *designers* e desenvolvedores para a criação de protótipos de telas rapidamente. Com uma estrutura modular e documentação adequada, é possível identificar onde a mudança deve ser feita e avaliar o impacto dela.
- **Interação entre *Atomic Design* e *Storybook*:** O *Storybook* é uma poderosa ferramenta para o desenvolvimento de componentes, oferecendo diversas funcionalidades, como visualizador de componentes, ferramentas de documentação e testes. Durante o processo de desenvolvimento, ele facilita a visualização dos componentes antes de serem implementados na tela, economizando tempo e esforço, pois não é necessário *mockar* os componentes ou criar páginas de teste para visualizá-los. A Figura 11 mostra como o arquivo com a extensão .stories deve ser disposto na pasta, sua estrutura é semelhante a de um arquivo nativo em que devemos retornar os componentes igual uma função.
- **Testes:** O *Atomic Design* torna os testes de componentes mais fáceis, graças à decomposição em elementos menores e independentes, com menor complexidade e interações. A estrutura hierárquica também permite testar componentes em diferentes níveis de abstração, identificando problemas em níveis inferiores antes que afetem níveis superiores. Além disso, a manutenção dos testes é facilitada com a reutilização de testes de componentes menores em componentes maiores

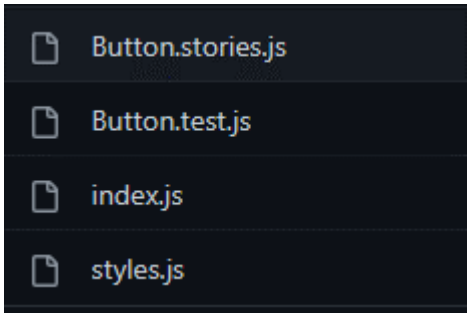

Figura 11: Estrutura interna de uma pasta do componente atômico Botão. Fonte: Elaborado pelo Autor

4.4 Considerações Gerais

Durante o desenvolvimento foram feitas observações que não se categorizam como vantagens ou desvantagens. A primeira delas o fato de que o *Atomic Design* é uma metodologia para criação de componentes modulares e escaláveis para projetos de grande escala, mas pode ser considerado uma sobrecarga desnecessária em projetos pequenos, onde a escalabilidade não é tão importante. A estratégia introduz complexidade adicional no desenvolvimento e manutenção, e não justifica o investimento de tempo e recursos. Em projetos de escopo pequeno, ou que não possuem o objetivo de criar um guia de estilo interno entre os desenvolvedores, pode ser mais fácil e rápido usar uma abordagem tradicional para construir componentes.

Durante as entregas, os desenvolvedores da equipe, que possuem acesso à ferramenta, fizeram alguns comentários de revisão do projeto. Dos relatos elaborados acerca do *design* da interface e estrutura do código, todos se mostraram positivos à ideia da metodologia atômica para o projeto, mas com ressalvas sobre o tempo necessário para que novas funcionalidades fossem implementadas, e como elas se encaixariam dentro dos prazos de próximas entregas. Das avaliações negativas, podemos listar a sobrecarga de componentes de baixo nível, como átomos e moléculas, e como isso poderia ser um impeditivo para o avanço da hierarquia do *design*, dado que a manutenção desses componentes se torna complexa de ser gerenciada, isso se expande para todo o projeto, visto que adicionamos uma camada de complexidade nos componentes. Outro ponto abordado foi sobre como a UI tinha a mesma estilização, e que seria interessante que houvesse uma expansão da paleta de cores, espaçamento e mais variedades de átomos.

Sobre a relação entre prazo de entrega e complexidade da metodologia, inicialmente observamos que houve uma curva de aprendizado necessária para o entendimento da estratégia, porém ao dominar os conceitos da mesma, o processo de entregas se mostrou mais eficiente.

Outro fator observado é que é necessário ter conhecimento prévio de uma série de conceitos e tecnologias, dentre elas estão:

- **HTML:** é a base para a construção de componentes, portanto é necessário saber como estruturar e nomear elementos HTML de forma acessível e semântica.
- **CSS:** é utilizado para estilizar os componentes. Há a necessidade de conhecer como usar a cascata, herança e seletores

para estilizar elementos e como isso pode ser dinamicamente ajustável através de funções internas utilizando bibliotecas como *styled-components*.

- ***Javascript*:** é utilizado para criar interações e comportamentos dinâmicos nos componentes. É importante conhecer como manipular o DOM e estruturar código de forma modular.
- **Design:** conhecer princípios de *design* como hierarquia, alinhamento, contraste, *design tokens* e consistência para aplicar no processo de criação de componentes.
- ***frameworks*:** Entender como *React*, *Vue*, *Angular*, funcionam, como a estruturação de rotas, gerenciamento de estados e integração com outras bibliotecas agilizam a construção de componentes.
- **Conhecimento de boas práticas de acessibilidade:** tais como padrões de navegação, rótulos de formulário, acessibilidade a todas as funcionalidades da página através do teclado, entre outros, para garantir que o sistema seja acessível a todos os usuários.
- **Prototipação de telas:** ferramentas como *Storybook* agilizam a criação de documentações e prototipação de telas, relevante para que novos integrantes da equipe se situem em relação aos padrões.

5 CONSIDERAÇÕES FINAIS

Foi posto em prática, durante o desenvolvimento do projeto, a abordagem Atômica, esta se mostrou eficiente ao ser trabalhada junto com outras ferramentas. Notou-se que, para um projeto de pequeno porte com poucos *templates*, essa estratégia não é ideal, uma vez que o esforço para codificação é muito alto para pouco resultado. Mas, para o caso apresentado no artigo, a abordagem se mostrou eficaz e será sugerida como proposta de estudo para outros times de desenvolvimento *front-end* da empresa. Observou-se que a estratégia avaliada possui uma curva de aprendizado para sua utilização, como no caso apresentado o projeto foi desenvolvido por apenas uma pessoa, não podemos observar o comportamento do design atômico em uma equipe, e como ele seria mantido durante as entregas para os clientes internos. Porém, como há previsão para a entrada de novos integrantes na equipe, foi provida uma documentação interna que engloba os principais métodos de criação e manutenção de componentes e seus testes.

Em trabalhos futuros pretendemos expandir o projeto para outras frentes, com uma equipe maior e observar de que forma o *Atomic Design* se comporta em um ambiente de produção de larga escala e como novas funcionalidades afetam o projeto, tal como disponibilizar um guia de estilo próprio do projeto, definindo uma lista de componentes e regras de estilização próprias. Como apontamos no artigo, uma estrutura de componentes atômicos tem seus prós e contras de maneira geral, o *Atomic Design* é uma excelente abordagem, mas não é a solução ideal para todos os projetos. É importante compreender o escopo do projeto antes de aplicá-lo, com base nas considerações mencionadas anteriormente. Ele é especialmente útil para projetos que envolvam a criação de bibliotecas de componentes ou para situações em que várias interfaces compartilham os mesmos componentes, e de forma geral é importante que como estratégia ela pode ser moldada de acordo com as necessidades da aplicação. Por fim, a pesquisa desenvolvida mostra que o *Atomic Design* possui um grande potencial de sucesso, uma vez que seus pontos negativos podem ser pautados em falhas de gerência e organização interna da equipe ou até mesmo não ser condizente com o escopo do projeto. Deste modo, *Design System*s, em particular, *Atomic Design*, se mostram um campo com potencial para análises futuras.

REFERÊNCIAS

[1] Mark Boulton. 2012. *Structure First. Content Always.* Retrieved 22 de jan. de 2023 from https://markboulton.co.uk/journal/structure-first-content-always/

[2] Marco Antonio F. da Costa and Maria de Fátima B. da Costa. 2017. *Projeto de Pesquisa: entenda e faça.* Editora Vozes Limitada.

[3] Mônica Ramos Daltro and Anna Amélia de Faria. 2019. Relato de experiência: Uma narrativa cientifica na pós-modernidade. *Estudos e pesquisas em psicologia* 19, 1 (2019), 223–237. https://doi.org/1808-4281

[4] Njeru Mwendi Edwin. 2014. Software frameworks, architectural and design patterns. *Journal of Software Engineering and Applications* 7 (June 2014), 670–678. https://doi.org/10.4236/jsea.2014.78061

[5] Brad Frost. 2016. *Atomic Design* (1st. ed.). Chapter Atomic Design Methodology. https://atomicdesign.bradfrost.com/chapter-2/.

[6] Antônio Carlos Gil. 2002. *Como elaborar projetos de pesquisa* (4 ed.). Atlas, São Paulo.

[7] Katie Sylor-Miller Diana Mounter Marco Suarez, Jina Anne and Roy Stanfield. 2018. *Design Systems Handbook.* https://www.designbetter.co/design-systems-handbook.

[8] Vendly Rick Gunawan, Geraldi Anthony and Maria Susan Anggreainy. 2021. The Effect of Design User Interface (UI) E-Commerce on User Experience (UX). *2021 6th International Conference on New Media Studies (CONMEDIA)* (2021), 95–98. https://doi.org/10.1109/CONMEDIA53104.2021.9617199

[9] W3C. 2018. *Diretrizes de Acessibilidade para Conteúdo Web.* Retrieved 29 de jan. de 2023 from https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/