OTÁVIO DE LIMA SAORES

DESENVOLVIMENTO DE UMA BIBLIOTECA DE COMPONENTES WEB PARA EMPRESA PRIVADA

LAVRAS – MG

2022

OTÁVIO DE LIMA SAORES

DESENVOLVIMENTO DE UMA BIBLIOTECA DE COMPONENTES WEB PARA EMPRESA PRIVADA

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Bacharelado em Ciência da Computação para a obtenção do título de Bacharel.

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG

2022

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Soares, Otávio de Lima

Desenvolvimento de uma biblioteca de componentes web para empresa privada / Otávio de Lima Soares. 1^a ed. rev., atual. e ampl. – Lavras : UFLA, 2022.

40 p. : il.

Relatório de estágio(graduação)–Universidade Federal de Lavras, 2022.

Orientador: Prof. Dr. Maurício Ronny de Almeida Souza.

Bibliografia.

1. TCC. 2. Monografia. 3. Dissertação. 4. Tese. 5. Trabalho Científico – Normas. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

OTÁVIO DE LIMA SAORES

DESENVOLVIMENTO DE UMA BIBLIOTECA DE COMPONENTES WEB PARA EMPRESA PRIVADA

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Bacharelado em Ciência da Computação para a obtenção do título de Bacharel.

APROVADA em 25 de Abril de 2022.

Prof. Dr. Paulo Afonso Parreira Junior	UFLA
Bel. Daniel Armando Campos	SYDLE

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG
2022

Dedico à minha família. Em especial aos meus pais, Adriana e Frederico

AGRADECIMENTOS

Agradeço primeiramente a minha mãe Adriana e meu pai Frederico, que sempre me apoiaram e me deram suporte para chegar até aqui.
Ao Prof. DSc. Maurício Ronny de Almeida Souza, pela orientação durante a escrita do presente documento.
A Universidade Federal de Lavras, que forneceu todo o material físico e humano, para que eu pudesse conquistar esse Título.
Aos meus irmãos, Hugo e Alice que sempre estiveram à disposição para me apoiar, principalmente durante a jornada de graduação.
A minha namorada Laura que sempre me manteve motivado e focado no objetivo.
A todos meus amigos que me apoiaram e transformaram esta caminhada mais do que especial.
A toda a minha família que torceram por mim durante esta caminhada.

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.
(Steve Jobs)

RESUMO

O objetivo do projeto relatado foi a implementação de uma interface para um sistema da empresa privada através da construção de uma biblioteca própria de Web Components. Este trabalho serviu de auxílio e nele foi realizado uma parte do projeto de criação de uma biblioteca de componentes, que posteriormente foi utilizada pra desenvolver uma aplicação de marketplace, destinada a disponibilização de aplicativos corporativos. Web components podem ser definidos como um conjunto de tecnologias distintas utilizadas para construir trechos de código HTML que podem ser reutilizados diversas vezes e serem customizados de maneira simples. Essa tecnologia está sendo muito utilizada por empresas, pois garante que as mesmas possam definir um padrão de design nos seus componentes e replicá-los em todos os seus sistemas, o que ajuda a fortalecer sua identidade visual. O estágio reportado nesse relatório foi realizado na empresa SYDLE Sistemas Ltda, que é desenvolvedora de uma plataforma que fornece soluções tecnológicas para que empresas possam lidar com seus processos corporativos de uma maneira mais prática e eficaz. Foi feito uma proposta de criação de um sistema de marketplace onde a empresa poderia publicar aplicativos corporativos, chamados de SYBOX, prontos para serem baixados pelos seus clientes. O trabalho que envolve a implementação da interface desse sistema foi todo baseado no conceito de Web Components. Desta forma, foi criada uma biblioteca de componentes altamente customizáveis para que posteriormente eles pudessem ser reaproveitados em todos os sistemas da empresa. O Trabalho realizado no estágio relacionou-se diretamente com conteúdos vistos em sala de aula e proporcionou uma visão prática dos mesmos.

Palavras-chave: Front-end. Web Components. Marketplace. Corporativo. Reutilização

ABSTRACT

The objective of the reported project was the implementation of an interface for a system of the private company through the construction of its own library of Web Components. This work served as an aid and a part of the project to create a component library was carried out, which was later used to develop a marketplace application, aimed at making corporative apps available. Web components can be defined as a set of distinct technologies used to build HTML code snippets that can be reused several times and be customized in a simple way. This technology is being widely used by companies, as it ensures that they can define a design pattern in their components and replicate them in all their systems, which helps to strengthen their visual identity. The internship reported in this report was carried out at the company SYDLE Sistemas Ltda, which is a developer of a platform that provides technological solutions for companies to deal with their corporate processes in a more practical and effective way. A proposal was made to create a marketplace system where the company could publish corporative apps ready to be downloaded by its customers. The work that involves the implementation of the interface of this system was all based on the concept of Web Components. In this way, a library of highly customizable components was created so that they could later be reused in all the company's systems. The work carried out in the internship was directly related to contents seen in the classroom and provided a practical view of them.

Keywords: Front-end. Web Components. Marketplace. Corporative. Reuse

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A SYDLE é uma empresa de produtos voltados para tecnologia corporativa. A empresa apresenta um conjunto de 9 soluções que visam otimizar a gestão de processos corporativos, sendo eles externos ou internos. Cada solução possui um objetivo específico, porém todas são utilizadas dentro de uma única plataforma denominada SYDLE ONE, que é o seu produto principal. O SYDLE ONE é um sistema web que integra recursos de BPM (Business Process Management), ECM (Enterprise Content Management), CRM (Customer Relationship Management), Analytics, Pagamentos, E-commerce, Gestão de energia, Portal de serviços e SYBOX a uma interface de usuário intuitiva, que busca além de automatizar processos, garantir uma boa experiência de usabilidade aos clientes.

Neste contexto, o objetivo deste documento é descrever as atividades desenvolvidas pelo autor durante o estágio supervisionado realizado na empresa SYDLE. O objetivo do estágio era desempenhar atividades ligadas ao desenvolvimento de software, com um contato direto com o produto da empresa, sendo responsável por realizar o desenvolvimento, modelagem, manutenção e testes de soluções web, atuando na plataforma SYDLE ONE, utilizando tecnologias como: Javascript, Elasticsearch, Angular, entre outras, sendo supervisionado por um profissional capacitado. O estágio foi realizado no período de maio/2021 à dezembro/2021, totalizando 840 horas.

1.1 Sobre a SYDLE

Fundada em 2005 pelo seu CEO Daniel Cataldo, a SYDLE está localizada na região da Savassi na cidade de Belo Horizonte - MG e atua com seus produtos em mais de 80 países. Com mais de 15 anos de atuação no mercado e há mais de 10 vem sendo premiada como uma das melhores empresas de tecnologia para se trabalhar no Brasil e na América Latina, a SYDLE empresa possui seu negócio voltado para produtos de tecnologia corporativa, oferecendo cerca de 9 soluções que auxiliam as empresas a gerenciarem seus processos dentro de uma única plataforma,o SYDLE ONE. Tais soluções consistem em:

- **BPM:** Um sistema de automatização de processos que se baseia na modelagem de diagramas construídos em notação BPMN (Business Process Model and Notation). Para isso, o BPM

conta um editor de diagramas que funciona com base no modelo arrasta e solta e um sistema de importação que permite a montagem de um processo a partir do *upload* de um arquivo contendo um diagrama.

- **ECM:** A solução de ECM é mais voltada para gestão de conteúdos e documentos. Com ela é possível armazenar grandes quantidades de dados em um único local, além de proporcionar um sistema de segurança em que é possível configurar níveis de acesso para os usuários e também a geração de documentos através da criação de formulários personalizados.

- **CRM:** Com o CRM é possível que a empresa consiga gerenciar o relacionamento com seus leads e clientes de uma forma mais prática. O CRM oferece todo um conjunto de ferramentas que auxiliam na captação e classificação de leads, além de automatizar qualquer processo que envolva funil de vendas e contato com o cliente.

- **Analytics:** O analytics é um sistema que realiza a análise e processamento de dados e os sintetiza no formato de gráficos, tabelas, quadros kanban e etc.

- **Billing:** Billing é a tecnologia que auxilia os clientes fornecendo uma solução completa de gestão de cobranças de produtos, serviços e assinaturas.

- **E-commerce:** A plataforma de e-commerce fornece um sistema que possui front-end e back-end integrados para maximizar conversões de vendas em grande escala.

- **Energy:** O Energy é o sistema que fornece solução de comercialização de energia aplicada à evolução de negócios. Desta forma, a plataforma de Energy possibilita que o cliente gerencie seus contratos, acompanhe o histórico de suas negociações, desenvolva seu portfólio e realize simulações com algoritmos que auxiliam na tomada de decisão para o negócio.

- **Service Desk:** O service Desk é uma plataforma que proporciona maturidade operacional na gestão de demandas internas e externas. O produto conta com um sistema que possibilita o registro e atendimento de tarefas, desenvolvimento de canais de relacionamento com o público, geração de relatório de desempenho baseado nos chamados abertos na fila e os já atendidos e etc.

- **SYBOX:** O SYBOX é uma solução que apresenta automatizações para os processos mais comuns do mundo corporativo, como gestão de Recursos Humanos, gestão de setor Administrativo e Financeiro, Jurídico, Marketing e etc. As soluções SYBOX são sistemas prontos para serem importados e utilizados na plataforma SYDLE ONE.

A empresa possui equipes de desenvolvimento, infraestrutura, marketing, atendimento ao cliente e consultores. Cada equipe de desenvolvimento possui foco em uma das 9 tecnologias apresentadas acima.Além disso há uma equipe de produto que atua no desenvolvimento e manutenção da plataforma que integra todas essas soluções e um equipe de *Growth* que desenvolve as plataformas e conteúdos responsáveis por expor o nome da SYDLE pelo mundo.

Na estrutura, a empresa conta com 3 andares do prédio em que está localizada e disponibiliza notebooks, monitores e internet para os funcionários. A empresa também possui uma sala de refeição que conta com lanches liberados, como frutas, misto quente e outros e uma sala de descompressão, com área gourmet, além de um grande espaço aberto para relaxar ou trabalhar em um ambiente diferente.

Durante o período de estágio o estagiário atuou tanto na equipe de *Growth*, quanto na equipe de usabilidade do produto. Nas duas equipes a metodologia de trabalho é muito parecida. Para o início de cada projeto é realizada uma reunião em que a equipe responsável realiza um detalhamento técnico de tudo o que o projeto precisa atender na entrega final. Com o objetivo final definido, o projeto é quebrado em tarefas menores e a equipe de desenvolvimento escolhe prioriza quais serão trabalhadas de imediato. Para acompanhamento do projeto são marcadas reuniões diárias onde a equipe de desenvolvimento tem a oportunidade de expor em detalhes as tarefas que estão realizando e apresentar possíveis dúvida e impedimentos. Além disso, é feita uma reunião de planejamento a cada 10 dias onde os resultados são apresentados e novas tarefas são planejadas para os próximos 10 dias.

1.2 Organização do Trabalho

O presente trabalho visa relatar atividades desenvolvidas e experiências adquiridas durante o período de estágio. Desta forma o documento encontra-se organizado no sistema de capítulos, onde no Capítulo 1 é feita uma introdução sobre a empresa em que o estágio foi realizado, descrevendo

seu funcionamento e uma breve definição dos objetivos do estágio. O Capítulo 2 traz uma explicação inicial sobre os conceitos e tecnologias utilizadas durante a execução do projeto proposto. O Capítulo 3 faz uma abordagem detalhada sobre as atividades desenvolvidas durante o projeto de estágio, envolvendo a construção de uma biblioteca de Web Components e sua utilização na implementação de um sistema de marketplace. Por fim, o Capítulo 4 apresenta uma conclusão contendo dificuldades enfrentadas, habilidades desenvolvidas e uma comparação entre conceitos adquiridos em sala de aula com a sua utilização na prática dentro do mercado de trabalho.

2 CONCEITOS E TECNOLOGIAS

Este capítulo descreve alguns conceitos e tecnologias a cerca de desenvolvimento web, Web Components, testes automatizados e controle de versões cujo o domínio foi importante para uma boa execução das atividades explicitadas no Capítulo 3.

2.1 Programação Web

O conceito de programação Web pode ser definido pelo conjunto de ações que envolvem o desenvolvimento de sistemas de informações, cuja hospedagem é feita na internet.

Segundo (CALOURIS et al., 2013), aplicações web são divididas em 3 módulos:

- **Interface de usuário:** Módulo que define a parte visível do sistema para o usuário, através da qual, ele se comunica para realizar suas tarefas.
- **Servidor:** Módulo que realiza a ponte de comunicação entre a interface e o banco de dados, realizando as manipulações de dados necessárias antes de transmitir a mensagem entre um e outro.
- **Banco de dados:** Módulo que armazena os dados produzidos dentro do sistema, de maneira a garantir a persistência das informações.

Deste modo a programação web envolve o estudo e a execução das atividades de construção destes 3 módulos. Pra isso, existem diversas tecnologias que possibilitam essa implementação. É possível construir a interface utilizando linguagens como HTML, CSS e JavaScript ou alguma ferramenta de desenvolvimento como o Angular que utiliza de TypeScript como linguagem principal. O servidor pode ser desenvolvido utilizando algum *framework* de desenvolvimento destinado a construção de servidores, como o Laravel que utiliza PHP como linguagem de programação, o Django que utiliza Python, entre outros. Já os bancos de dados, podem ser desenvolvidos através da linguagem SQL utilizando algum sistema de gerenciamento de banco de dados ou através do sistema noSQL onde os SGBDs(sistemas de gerenciamento de banco de dados) armazenam os dados utilizando o formato JSON (JavaScript Object Notation).

2.1.1 Angular

Segundo (GOOGLE, 2022), o Angular é uma plataforma de desenvolvimento, construída através da linguagem TypeScript, que engloba algumas tecnologias que auxiliam no desenvolvimento de aplicações Web. Sendo tais tecnologias:

- Um *framework* orientado a componentes destinado a construção de sistemas web, altamente escaláveis.
- Uma coleção de bibliotecas nativas que proporcionam diversas funcionalidades como roteamento, tratamento de formulários, comunicação com servidor cliente, entre outros.
- Um conjunto de ferramentas de desenvolvimento que ajudam no desenvolvimento, compilação e testes das aplicações.

O desenvolvimento utilizando Angular, costuma ser comparado com o ato de brincar com blocos de montar, onde os blocos são os componentes e a aplicação seria uma das possíveis composições feitas com esses "blocos".

Figura 2.1 – Declaração de componente Angular

```
import { Component } from '@angular/core';

@Component({
  selector: 'meu-componente',
  template: `
    <h2>Clique para gerar um numero</h2>
    <button (click)="generateNumber()"></button>
  `
})
export class MeuComponente {
  number = 0;

  generateNumber(){
    this.number = Math.random() * 10;
    alert(`O numero gerado e: ${this.number}`);
  }
}
```

Fonte: Otávio de Lima Soares

Para construir um componente, que seria um dos blocos que compõe a aplicação é necessário a implementação de uma estrutura com um padrão parecido com o código presente na Figura 2.1. O decorador **@Component** diz para o compilador que a classe declarada logo abaixo é um componente que deve ser construído através da *tag* HTML com nome definido no atributo *selector* e renderizará em tela o conteúdo declarado no atributo *template*. A utilização do **@Component** indica também que o código definido na classe "MeuComponente"é que define os comportamentos do componente que está sendo criado. Desta forma, o exemplo da Figura 2.1 define um componente Angular que ao clicar no botão renderizado é exibido um alerta contendo um número entre 0 e 10, gerado de maneira aleatória.

2.2 Web Components

Como explicado por (YANG; PAPAZOGLOU, 2002), o conceito Web Components é utilizado para definir um conjunto de tecnologias que permite a criação de elementos extensíveis, customizáveis e reutilizáveis, cuja funcionalidade se encontra separada do restante do código da aplicação. Esses componentes encapsulam trechos de código que definem funcionalidades e *templates* HTML que servem como blocos de construção para o desenvolvimento de aplicações complexas, com base em conceitos de reutilização e extensão.

(PIOVESAN et al., 2021) afirmam que a construção desses "blocos de montar"é baseada em 3 tecnologias básicas:

- **Elementos Customizados:** Um conjunto de APIs JavaScript que possibilitam a construção de elementos customizados, definindo sua forma e comportamento.
- **Shadow DOM:** APIs JavaScript que encapsulam o conteúdo de um elemento em uma árvore DOM "fantasma". Deste modo, seus recursos são renderizados de forma separada do DOM principal da aplicação, evitando conflitos de estilos e de comportamento com outros elementos do sistema.
- **Templates HTML:** Atributo que permite a definição do modelo de estrutura de um elemento customizado, ou seja, define a forma do componente.

2.2.1 Stencil

Segundo (GIULIANO; RILEY; THOMAS, 2022) o Stencil é um compilador de Web Components altamente customizáveis, desenvolvido pela Ionic framework, que combina conceitos dos *frameworks* de desenvolvimento mais populares no mercado em uma única ferramenta.

O Stencil utiliza TypeScript, JSX (tecnologia que permite a escrita de código HTML/XML dentro do JavaScript) e CSS para criar Web Components, que podem ser utilizados para criar aplicações, sistemas de design e bibliotecas de componentes de alta qualidade, que é o caso do projeto relatado no presente documento.

A ferramenta gera componentes compatíveis com os padrões que funcionam com estruturas populares de forma imediata, basta apenas declarar a *tag* HTML do Web Component compilado. Além disso, é possível gerar componentes nativos de qualquer *framework* que serão usados como qualquer outro componente declarado pelo *framework* de escolha. O Stencil faz isso envolvendo seus Web Components por meio do seu recurso Output Target, que mapeia a estrutura de um componente da ferramenta alvo e realiza a tradução em tempo de compilação.

Comparado ao uso direto de elementos personalizados, o Stencil fornece ferramentas adicionais que simplificam a escrita de componentes ágeis. APIs como Virtual DOM, JSX e renderização assíncrona facilitam a criação de componentes poderosos e rápidos, mantendo 100% de compatibilidade com Web Components. O Stencil também habilita alguns recursos importantes em elementos da web, como pré-renderização e objetos como propriedades (em vez de apenas strings).

2.2.2 Storybook

Assim como dito por (STORYBOOK, 2022), o Storybook é uma ferramenta de desenvolvimento de interface de usuário que torna o desenvolvimento mais rápido e fácil isolando componentes. Isso permite que cada componente desenvolvido seja trabalhado individualmente. Desta forma, é possível desenvolver uma interface de usuário inteira sem ter que iniciar uma pilha de desenvolvimento complexa, vincular determinados dados a algum banco de dados ou navegar pelo seu aplicativo.

Além disso, o Storybook permite documentar e testar visualmente os componentes de forma automática para evitar erros. O Storybook possui também um ecossistema de ferramenta complemen-

tares que ajudam em várias tarefas durante o desenvolvimento, como adaptar um layout responsivo ou testar a acessibilidade.

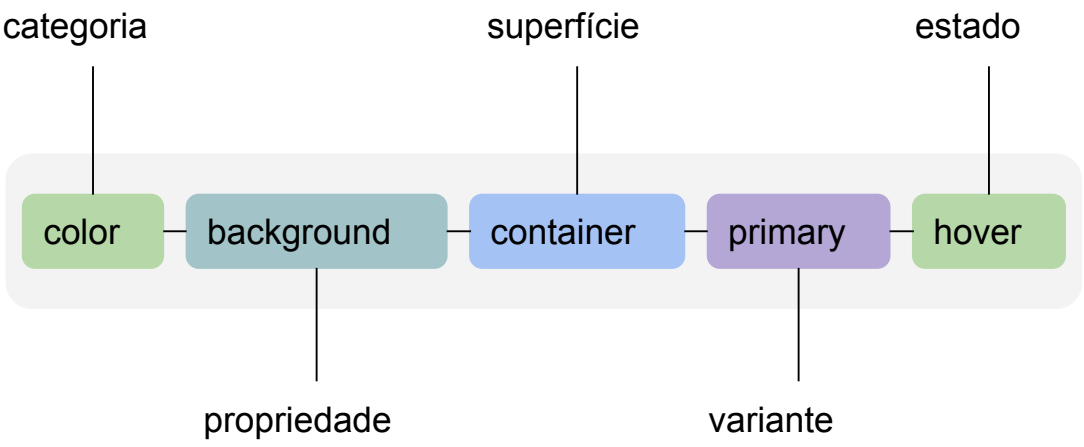
Como o Storybook apresenta ferramentas poderosas que auxiliam no desenvolvimento e documentação, utilizou-se tal ferramenta para realizar a documentação dos componentes presentes na biblioteca desenvolvida no presente projeto. Desta forma além de fazer consulta a respeito da utilização de cada componente o usuário poderia realizar alguns testes em tempo real, além de visualizar informações como o protótipo que inspirou o componente e código HTML que o mesmo encapsula.

2.2.3 Design Tokens

O termo "Design Tokens"surgiu no ano de 2014, durante o desenvolvimento do Design System da empresa Salesforce, implementado pelo time de design no qual a profissional Jina Anne atuava quando propôs essa definição. Este termo foi criado para denominar variáveis semânticas aplicadas ao estilo de um elemento, definindo tipografia, cores e formas de uma maneira livre de tecnologias específicas e que garantem certa praticidade na estilização do mesmo.

Geralmente a nomenclatura dos Design Tokens segue uma anatomia padrão exemplificada pela Figura 2.2 que procura informar seu local de utilização sem que seja necessário verificar seu valor.

Figura 2.2 – Anatomia de um Design Component



Fonte: Otávio de Lima Soares

Desta forma, sempre que preciso modificar o estilo especificado por determinado Design Token é possível modificar seu valor apenas onde foi declarado e a mudança surte efeito em todos os lugares do código onde o *token* foi utilizado.

2.2.4 Style Dictionary

Style Dictionary é um sistema que permite que o usuário defina estilos que podem ser consumidos por qualquer plataforma ou linguagem. Com ele, é possível criar e editar folhas de estilo e um único comando exportar essas regras para qualquer plataforma necessária - iOS, Android, CSS, JS, HTML, arquivos de esboço, documentação de estilo e etc.

De acordo com (AMAZON, 2021), ao gerenciar as experiências do usuário pode ser bastante desafiador manter os estilos consistentes e sincronizados em várias plataformas e dispositivos de desenvolvimento. Ao mesmo tempo, designers, desenvolvedores, gerentes de projeto, entre outros devem ser capazes de ter uma documentação de estilo consistente e atualizada para permitir um trabalho e uma comunicação eficazes. Mesmo assim, erros inevitavelmente acontecem e o design pode não ser implementado com precisão. O Style Dictionary resolve esse problema gerando automaticamente definições de estilo em todas as plataformas a partir de uma única fonte, removendo obstáculos, erros e ineficiências no fluxo de trabalho.

O Style Dictionary está disponível para uso através de uma interface de linhas de comando, mas também pode ser utilizado como um módulo npm convencional, ou seja, é possível realizar sua instalação dentro do projeto via npm e deste modo utilizar suas funcionalidades em meio ao código do projeto. Deste modo, é possível estender suas funcionalidades, adicionando algumas configurações mais específicas do projeta que está sendo desenvolvido.

2.3 Teste de Software

Uma das tarefas mais difíceis durante o desenvolvimento de uma aplicação é garantir que ela esteja funcionando 100% como o esperado. Afinal, a construção do software depende principalmente da habilidade de interpretação e desenvolvimento das pessoas que o desenvolvem, com isso algumas falhas podem surgir durante sua execução.

Para que tais erros não perdurem e que o produto final possa ser entregue com a maior qualidade possível (DELMARO; MALDONADO; JINO, 2016) afirma que existe uma série de atividades, chamadas de "Validação, Verificação e Teste", ou "VV&T", que possuem a finalidade de garantir que tanto o modo pelo qual o software está sendo construído quanto o produto em si estejam em conformidade com o especificado. Essas atividades não se restringem apenas ao produto final e podem ser executadas em tempo de desenvolvimento garantindo que problemas inciais não evoluam para problemas maiores. Hoje em dia existem muitas ferramentas que auxiliam o desenvolvedor na execução de tais atividades. O Jest somado ao Puppeteer é um ótimo exemplo de ferramenta de testes de interface.

2.3.1 Jest

Jest é uma estrutura de teste JavaScript projetada para garantir a exatidão de qualquer base de código JavaScript. Ele permite a escrita de testes através de uma API acessível, familiar e enriquecida com recursos que fornecem resultados precisos de forma rápida. Como dito por (FACEBOOK, 2022), o Jest possui uma documentação completa de todas as funcionalidades de sua API, garantindo um maior auxílio ao desenvolvedor durante o desenvolvimento.

O Jest possui uma tecnologia de paralelização que realiza os testes em processos isolados, garantindo um estado global exclusivo para cada um. Além disso o *framework* possui uma politica de priorização de testes que executa primeiro os processos que falharam durante a última execução e depois reorganiza a prioridade de execução com base nos tempos de processamento anteriores de cada um. Deste modo a execução de testes consegue ser mais precisa e rápida.

2.3.2 Puppeteer

Puppeteer é uma biblioteca que implementa uma API hierárquica de alto nível, que permite controlar navegadores do tipo Chromium ou Chrome, através do protocolo DevTools.

A comunicação do Puppeteer com o Chromium, ou Chrome é estruturada através de estruturas análogas as de um navegador do navegador registradas na Figura 2.3. Ao se conectar a uma instância Chromium é criada uma instância do tipo *Browser* e é através dela que toda comunicação é orquestrada, fazendo assim o papel de um navegador propriamente dito. Logo que criada, a instância

Browser necessita de uma instância de contexto, denominada *BrowserContext* que armazena informações relativas as sessões do usuário naquele "navegador", bem como as páginas, representadas pela entidade *Page* e seu conteúdo, representado por *Frame*. Desta forma, o Puppeteer consegue através de comandos do protocolo DevTools manipular essas entidades da forma desejada, podendo renderizar por exemplo trechos de código a serem testados e realizar algumas interações para checar seu comportamento.

2.4 Controle de Versões

Assim como explicado por (CHACON; STRAUB, 2022), o controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo tornando possível recuperar versões específicas posteriormente. Para profissionais da área de tecnologia da informação em geral, entre outros, esse é um sistema que auxilia de maneira significativa o fluxo de trabalho, uma vez que mantém uma linha do tempo de alterações nos códigos fonte de um software ou de imagens desenhadas para compor uma tela. Desta forma, é possível retroceder o arquivo ou projeto todo a um estágio anterior, comparar as mudanças com o passar do tempo, ver quem foi que modificou determinadas partes do arquivo, entre outras funcionalidades interessantes.

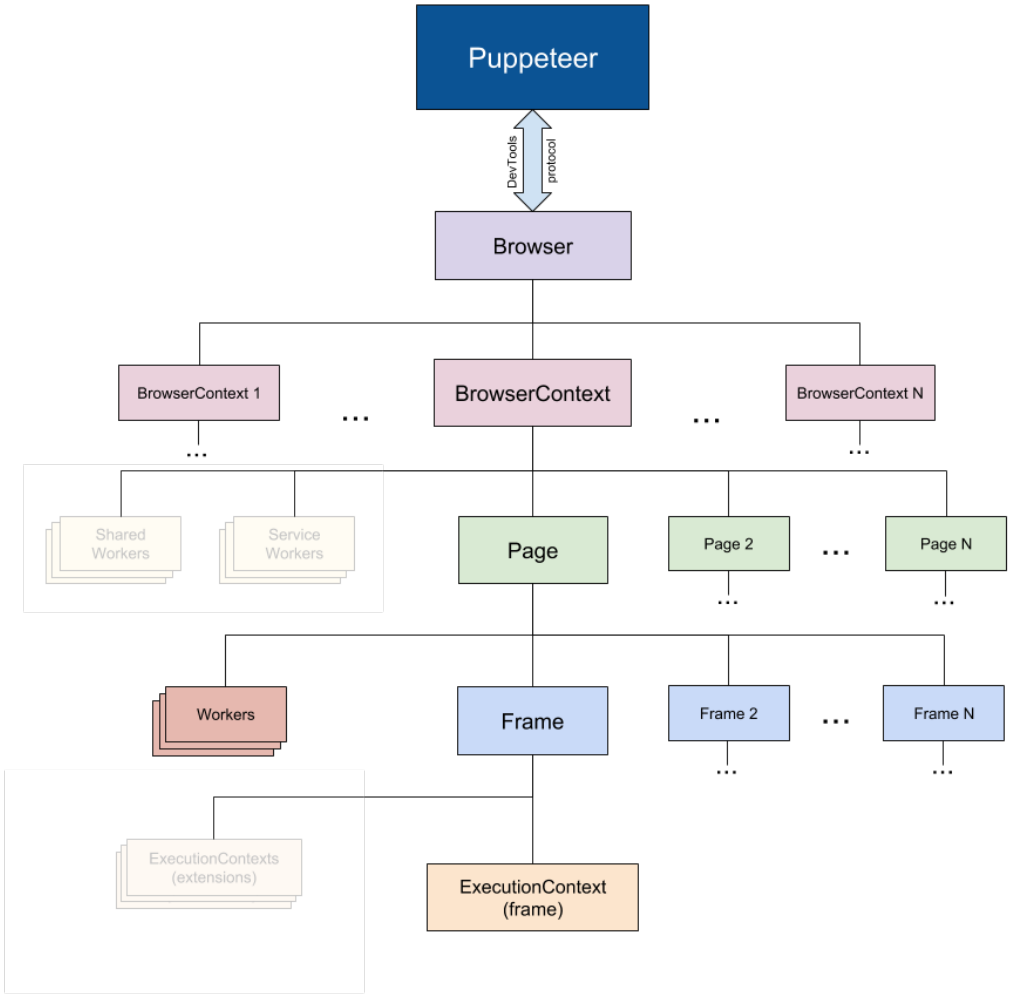
Há diferentes tipos de sistemas de controle de versões, que utilizam de técnicas distintas para realizar o gerenciamento dos arquivos. Tais variações consistem em:

- **Sistema de controle de versão local:** Consiste em um banco de dados local na máquina do usuário que armazena as versões dos arquivos a medida que alterações vão sendo feitas.

- **Controle de versão centralizado:** Criado como uma alternativa ao sistema local, para possibilitar que usuários diferentes pudessem ter acesso a mesma base de arquivos. Com isso é criado um repositório central onde os usuários se conectam para ter acesso aos arquivos, fazerem alterações e etc.

- **Controle de versões distribuído:** É uma fusão do sistema local com o centralizado. Há um repositório central que contem o versionamento dos arquivos, porém cada usuário tem sua cópia local que permite que eles trabalhem localmente e depois façam *upload* dessas alterações para

Figura 2.3 – Estrtura de comunicação do Puppeteer



Fonte: (PUPETEER, 2022)

o servidor, evitando assim problemas acarretados por falta de rede, falhas no servidor, entre outros.

2.4.1 Git & Gitlab

Git é o sistema de controle de versão mais utilizado no mercado atualmente. Isto se deve ao fato de ser um sistema eficiente e de fácil manipulação. Segundo (CHACON; STRAUB, 2022) o que difere o Git dos demais sistemas de controle de versão é a maneira que ele analisa e armazena seus dados. A maioria dos SCVs enxerga os dados como uma lista de mudanças ao longo do tempo, desta forma cada arquivo contém uma lista e cada nó dessa lista armazena as mudanças feitas no mesmo, já o Git ele armazena os dados no formato de *snapshot*, ou seja, toda vez que uma alteração é submetida é tirada como se fosse uma foto do estado de cada arquivo e uma referência a esse *snapshot* é armazenada. Para fins de otimização, arquivos que não tiveram alteração não são armazenados novamente e sim uma referência ao último estado alterado do mesmo. Desta forma, cada *snapshot* gerado pelo Git, chamado de *commit*, possui um código identificador e é através dele que é possível localizar os estados dos arquivos e navegar sobre a linha do tempo de alterações.

O Git possui um conjunto de comandos que são utilizados para realizar as ações desejadas com os arquivos observados. Porém utilizar apenas linhas de comando sem uma interface gráfica para organizar os dados pode ficar um pouco cansativo com o passar do tempo. Por isso, algumas ferramentas podem auxiliar no uso do Git, o Gitlab é uma delas. O Gitlab é uma plataforma DevOps completa que auxilia os desenvolvedores durante todo o processo de desenvolvimento de softwares. Com o Gitlab é possível gerenciar repositórios do Git através de uma interface gráfica intuitiva, que permite criar repositórios, visualizar *commits*, organizar o fluxo de trabalho com permissionamento de usuário para realizar determinadas tarefas no repositório e etc. Além disso o Gitlab conta com outras funcionalidades, como a criação de quadros do tipo kanBan para gerenciar as tarefas do time, configuração de ferramentas de integração e *deploy* contínuo, gerenciamento de pacotes Node, entre outras funcionalidades que garantem uma maior automatização do processo de gestão do desenvolvimento.

3 SYDLE-UI: UMA BIBLIOTECA DE WEB COMPONENTS

Este capítulo descreve as atividades que foram realizadas em um projeto de estágio, na empresa SYDLE, durante o período de maio à dezembro de de 2021. O projeto teve como principal objetivo a criação de uma biblioteca de Web Components que pudesse ser utilizada durante o desenvolvimento dos projetos da empresa, de maneira a garantir uma padronização no design dos componentes utilizados na construção das interfaces de todos os seus produtos. O desenvolvimento dessa biblioteca seguiu uma estrutura muito bem definida que será detalhada nas seções seguintes deste capítulo.

A Seção 3.1 trás a motivação para a criação da biblioteca e como se estruturaram as etapas de desenvolvimento. A Seção 3.2 realiza um passo a passo de como os componentes foram desenvolvidos e publicados. A Seção 3.3 demonstra a construção de uma aplicação de marketplace utilizando a biblioteca desenvolvida. E por fim a Seção 3.4 contém um relato de toda a experiencia de desenvolvimento de sistemas com Web Components bem como suas vantagens e dificuldades.

3.1 Concepção da SYDLE-UI

O projeto surgiu através de uma vontade que a empresa tinha de criar uma identidade visual que fosse forte e única da empresa, para que pudesse ser aplicada em todos os seus produtos. O objetivo além de garantir uma maior consistência de interface nos sistemas da organização era posicionar a empresa na rede de maneira que as pessoas remetessem à SYDLE sempre que vissem alguma aplicação que seguisse tais padrões.

Para isso, foi necessário pensar em uma maneira de criar essa padronização de uma forma que fosse fácil, rápida, prática e sem dependência com nenhuma tecnologia muito específica, tendo em vista que os produtos da organização utilizam tecnologias distintas. Além disso, também foi de extrema importância pensar em uma forma de garantir que essa padronização pudesse ser flexível a algumas customizações, principalmente de mudanças de paleta de cores e estilização de textos, visto que a SYDLE faz parte de um grupo de empresas parceiras como a LEVTY e a Dog Life que viriam a usufruir dessa solução futuramente, adaptando ao seu próprio estilo de design.

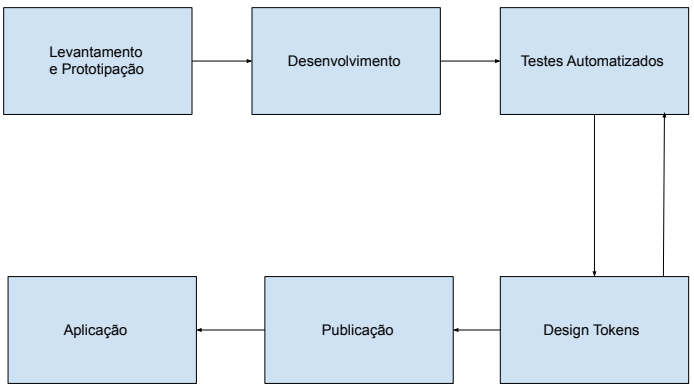
Levando em consideração todos esses requisitos sugeriu-se o desenvolvimento de uma biblioteca de Web Components altamente customizáveis, denominada SYDLE-UI. A escolha pelos Web

Components foi feita pois quando instalados no projeto eles podem ser utilizados como se fossem *tags* nativas do HTML, suprindo a necessidade de liberdade tecnológica nos projetos e também seu alto nível customização através da utilização de Design Tokens.

3.2 Construção da SYDLE-UI

Com a tecnologia definida, criou-se um time que envolvia colaboradores da área de usabilidade e um ou dois representantes dos demais times que viriam a utilizar essa biblioteca tanto na construção de novas aplicações quanto na refatoração de outras já existentes. Após a definição do time foi necessário estabelecer um padrão de trabalho que também garantisse que os componentes criados tivessem a mesma qualidade e seguissem o mesmo estilo de funcionamento. Com isso, estabeleceu-se o fluxo de trabalho ilustrado pela Figura 3.1.

Figura 3.1 – Fluxo de desenvolvimento do projeto



Fonte: Otávio de Lima Soares

De uma forma geral a SYDLE-UI se inicia por uma fase de coleta de ideias de componentes através do estudo de outras bibliotecas de referência, juntamente com a criação de protótipos que seriam o esboço inicial de como ficaria o sistema de design da biblioteca. Logo após iniciou-se então a fase de construção, trazendo para o código o que time de design desenhou. Na sequência testes

automatizados são aplicados para garantirem a qualidade de desenvolvimento dos Web Components. Após desenvolvimento e testes inicia-se então a fase de construção dos Design Tokens, com o objetivo de permitir a customização dos componentes, passando por uma nova rodada de testes e seguindo para fase de publicação que permite sua disponibilização em massa para a utilização em aplicações mais genéricas.

3.2.1 Levantamento e prototipação dos componentes

Para dar início a construção da biblioteca foi feito um levantamento dos componentes que seriam interessantes estar nessa biblioteca mais geral como diferentes tipos de botões, caixas de texto, componentes de *layout* e etc. O levantamento de tais componentes foi inspirado pela biblioteca de componentes do Ionic, um kit de ferramentas de código aberto que facilitam o desenvolvimento de aplicativos móveis e de *desktop* de alta qualidade e desempenho usando tecnologias da Web — HTML, CSS e JavaScript. Além de servir como base para o levantamento de componentes, o Ionic serviu como ferramenta de consulta para eventuais dúvidas sobre quais comportamentos conferir a cada componente durante o tempo de desenvolvimento dos mesmos. Após esse levantamento, os membros do time responsáveis pelo design do projeto desenvolveram um protótipo, utilizando o Figma como ferramenta de prototipação, de cada um dos componentes elencados, detalhando cada comportamento nas mais variadas situações que componente web pode se encontrar, sendo algumas delas: ativado, desativado e *hover* (ato de passar o mouse sobre o componente), para que então o trabalho de construção pudesse ser iniciado.

3.2.2 Desenvolvimento dos componentes

Com o protótipo em mãos utilizou-se o compilador de Web Components Stencil, onde eram escritos os códigos que davam comportamento de desenho aos web components. O Stencil foi escolhido, pois é fácil de aprender, principalmente quando o desenvolvedor já está acostumado com tecnologias como React e similares e garante maior produtividade, encurtando caminhos que seriam necessários na forma "natural".

O Stencil propõe uma estrutura muito bem definida para o desenvolvimento de componentes, o que facilita muito a sua construção, através da utilização de decoradores ao longo do código que automatizam alguns processos. Como explicado por (HANISCH; THOMAS, 2021), para se iniciar um componente como é necessário criar um arquivo de extensão .tsx como meu-componente.tsx e inseri-lo no diretório src/components do projeto, contendo algo parecido com o conteúdo da Figura 3.2. A extensão .tsx é necessária pois o Stencil utiliza a tecnologia JSX juntamente com TypeScript para construir os componentes.

Figura 3.2 – Estrutura de um componente no Stencil

```
import {Component, h, Host, Prop} from "@stencil/core";

@Component({
    tag: "meu-componente",
    styleUrl: ["scss/index.scss"]
})
export class MeuComponente {
    /**
     * Conteudo textual do botao
     */
    @Prop() conteudo : string;

    render() {
        return (
            <Host>
                <button>{this.conteudo}<button>
            </Host>
        )
    }
}
```

Fonte: Otávio de Lima Soares

O *decorator* **@Component** encontrado logo no início do código é responsável por informar ao compilador algumas informações como o nome da *tag* HTML que vai ser gerada para o componente, e o caminho de onde se encontra a folha de estilos do mesmo. Logo abaixo temos uma classe tradicional que implementa de fato o componente, definindo seus atributos, métodos, escutadores de eventos e contém também o código JSX que é renderizado em tela através da declaração da função render.

As propriedades declaradas na classe que contém o *decorator* **@Prop** transmitem ao compilador a informação de que são atributos públicos que podem ter seu valor definido pelo usuário. Por isso possuem também um observador de mudanças que faz com que o componente seja atualizado na tela toda vez que houver alguma alteração.

Uma vez compilado o componente pode ser declarado através da *tag* HTML criada, definindo o valor de seus atributos. No caso do exemplo a declaração pode ser feita como na Figura 3.3, que renderizará em tela um botão que contém o texto "Enviar".

Figura 3.3 – Declaração da Tag HTML

```
<meu-componente conteudo="Enviar"></meu-componente>
```

Fonte: Otávio de Lima Soares

Atrelado ao Stencil utilizou-se uma ferramenta chamada Storybook que é uma biblioteca de visualização de modelos que apresenta os componentes compilados pelo Stencil em uma interface onde é possível visualizá-los, realizar testes manuais e gerar uma documentação dos mesmos. Neste Projeto, utilizou-se esta ferramenta para documentar informações a respeito de tudo o que foi desenvolvido na biblioteca, desde informações necessárias para o desenvolvimento de componentes, até informações de publicação e instalação da biblioteca para a utilização em projetos. Assim, o conteúdo dividido em:

1. Desenvolvimento: Informações a respeito de configuração de ambiente, escrita de testes, publicação de versão, padrões de codificação e etc.

2. Projetos: Conteúdos que auxiliam na instalação da biblioteca para ser utilizada em projetos Stencil, AngularJs e ReactJs.

3. Fundamentos: Documentação da definição da paleta de cores, tamanhos de espaçamento, utilização de ícones e tipografia.

4. Componentes: Informações sobre os componentes desenvolvidos.

Neste projeto, a categorização dos componentes no Storybook foi feita através da funcionalidade que cada um desenvolvia, visando facilitar a busca por componentes na documentação. Desta

forma, os componentes foram agrupados em: Layout, Controle, Entrada, Visualização e Utilidades. Para inserir a documentação de um componente no projeto é necessário a inserção de um arquivo meu-componente.stories.tsx no diretório /src/components/meu-componente/ contendo a declaração de uma classe do tipo **SyStoryConfig** que conta com o atributo de título que mostra para o compilador em qual sessão a história deve ser criada, atributo de design contendo o link para o protótipo do componente, e atributo args que define os atributos do componente. A Figura 3.4 faz uma ilustração do conteúdo do arquivo meu-componente.stories.tsx.

Figura 3.4 – Estrutura do arquivo de história do Storybook

```
import {SyStoriesConfig} from "../../../.storybook/stories/automatedStories";
import readme from "./readme.md";

const storyConfig: SyStoriesConfig = {
    title: 'Components/Controls/MeuComponente',
    design: {
        type: "figma",
        url: "https://www.figma.com/file/meu-componente"
    }
    args: {
        conteudo: "Enviar"
    }
}
```

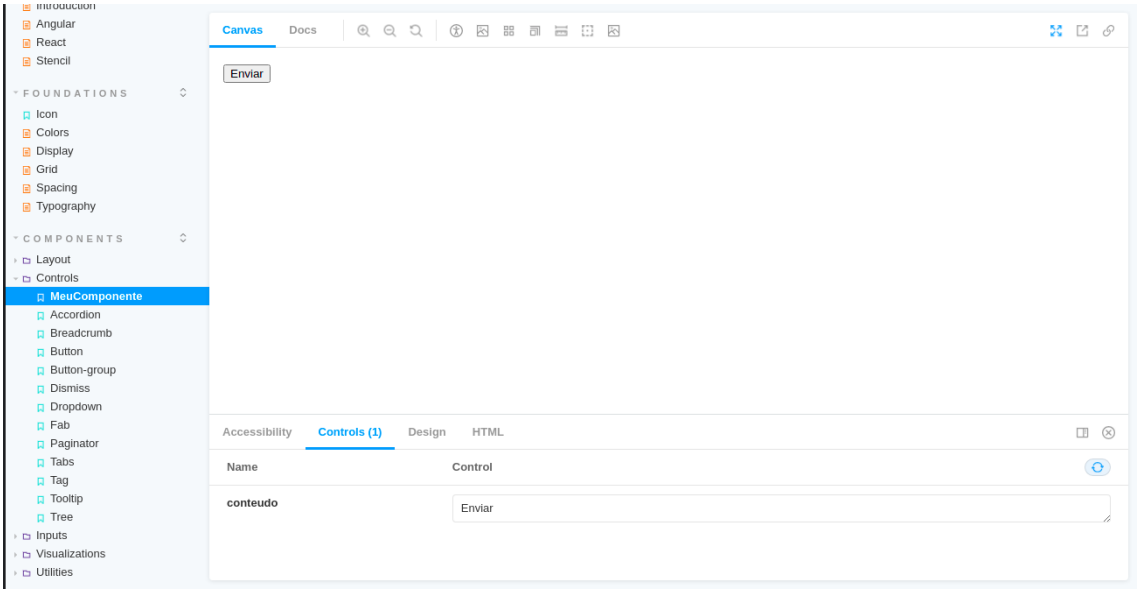
Fonte: Otávio de Lima Soares

Desta forma, como exemplifica a Figura 3.5, o compilador organiza essas propriedades e cria uma sessão no Storybook, contendo uma renderização do componente, uma aba mostrando o nível de acessibilidade do componente, uma aba de controle com os valores dos atributos definidos, uma aba de design exibindo o protótipo e uma aba contendo o código HTML gerado.

3.2.3 Testes automatizados

Durante o desenvolvimento dos componentes, foi aplicada uma ferramenta para realizar testes automatizados, de maneira a garantir que modificações futuras não alterem o desenho e os comportamentos estabelecidos para cada um. A ferramenta de testes utilizada foi o Jest, pois fornece toda uma

Figura 3.5 – Exibição do componente no Storybook



Fonte: Otávio de Lima Soares

estrutura de funções que auxiliam na escrita de tais testes e permitem o armazenamento dos resultados através de capturas de tela.

Em adição ao Jest foi necessário usar também o Puppeteer, uma biblioteca que possui uma API que possibilita controlar o Chrome ou Chromium através das ferramenta de desenvolvimento. Utilizou-se o Pupeteer para que o Jest pudesse renderizar e manipular os componentes presentes nos testes escritos através de uma janela web, a fim de realizar as capturas de tela que armazenam os resultados e posteriormente fazer uma comparação pixel a pixel dos mesmos para sempre garantir que o comportamento principal do componente foi mantido a cada mudança ou manutenção.

Para se realizar um teste utilizando o Jest com o Pupeteer basta adicionar um arquivo HTML contendo a declaração do componente a ser testado,como se encontra na Figura 3.6, e um arquivo de extensão .e2e.ts, que contém o código que irá manipular o componente, ao diretório src/components/meu-componente/test.

O código pertencente a Figura 3.7 então abre o código HTML da pasta de testes e cria um escopo de testes através da função **describe** indicando qual é o componente a ser testado e uma função que contém os cenários de teste. Dentro desse cenário de testes, cada caso é definido pela função **it**

Figura 3.6 – HTML de teste

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Meu componente</title>

    <style>
      body {
        padding: 1rem;
      }
    </style>
  </head>

  <body>
    <meu-componente conteudo="Enviar"></meu-componente>
  </body>
</html>
```

Fonte: Otávio de Lima Soares

Figura 3.7 – Código do teste

```
import { newE2EPage } from '@stencil/core/testing';
import fs from 'fs';
import path from 'path';

const html = fs.readFileSync(path.resolve(__dirname, 'index.html'), 'utf-8');

describe('meu-componente', () => {
  it('renders', async () => {
    const page = await newE2EPage({
      html: html,
    });

    const comp = await page.find('meu-componente');
    expect(comp).toBeTruthy();

    await page.waitForTimeout(3000);

    const compare = await page.compareScreenshot();
    expect(compare).toMatchScreenshot();
  });
})
```

Fonte: Otávio de Lima Soares

que define um título com o que deve ser feito e função que executa o teste. No cenário de exemplo o teste então renderiza o componente em tela através da criação de uma página web de testes com o Puppeteer utilizando a função **newE2EPage**, dentro dessa página ele procura pelo componente declarado, verifica se a busca foi bem sucedida e então faz uma captura de tela que é comparada com a imagem de referência, exemplificada pela Figura 3.8.

Figura 3.8 – Captura de tela de referência

Fonte: Otávio de Lima Soares

3.2.4 Design Tokens

Após a implementação inicial dos componentes e o ambiente de testes propriamente configurado a terceira etapa do projeto foi a definição de Design Tokens. A finalidade desta etapa foi garantir aos componentes algumas possibilidades de customização, bem como adaptação ao tema escuro ou claro do navegador do usuário. Novamente o time de design atuou definindo o design dessas customizações para que os *tokens* pudessem ser implementados. Para organizá-los melhor esses *tokens* foram divididos entre:

1. *Design Tokens*: Esses eram mais genéricos, definiam a paleta de cores a ser utilizada, tipografia dos textos, tamanhos de espaçamento e etc.

2. *Component Tokens*: Esses já eram mais específicos para cada componente em si e muitas vezes utilizavam valores definidos nos Design Tokens.

Com essa configuração de design é possível que os componentes sejam customizados de forma mais fácil seja usando os valores dos *tokens* pré-definidos ou alterando o valor de alguns *tokens* através da folha de estilos do projeto.

Para a criação destes Design Tokens foi utilizado o Style Dictionary, pois a ferramenta permite que com a criação de um único arquivo JavaScript contendo a definição dos *tokens*, seja possível exportá-los para qualquer tipo de plataforma, sendo ela CSS, IOS, Android ou HTML, o que evita retrabalho.

A definição de Design Tokens para um componente na SYDLE-UI é feita através da criação de um arquivo de nome *tokens*.js, na raíz do diretório que contém os arquivos do componente, contendo a seguinte estrutura: Com a estrutura declarada na Figura 3.9 o Style Dictionary compila este objeto

Figura 3.9 – Definição de Design Tokens

```
const { autoDoc } = require('../../../.style-dictionary/token.helper');

const TOKENS = {
  "meu-componente": {
    "background-color": {
      value: "#0000"
    },
    "color": {
      value: "#fffff"
    }
  },
};

module.exports = autoDoc(TOKENS);
```

Fonte: Otávio de Lima Soares

e cria um arquivo tokens.scss (3.10) na pasta de estilos do componente que contém a declaração dos Design Tokens em variáveis Sass, com suas respectivas documentações criadas pelo *helper* autoDoc. Para utilizar o token basta declará-lo como valor em alguma regra css através da função map-get, como na Figura 3.11, que pega o token e verifica se ele teve um valor definido em alguma outra regra, caso contrário é utilizado o valor padrão definido no arquivo *tokens*.js.

Figura 3.10 – Design Tokens compilados

```
/**
 * Do not edit directly
 * Generated on Wed, 30 Mar 2022 18:07:27 GMT
 */

 $\textit{tokens}: (
  meu-componente-background-color: var(--meu-componente-background-color, #0000),
  meu-componente-color: var(--meu-componente-color, #ffff),
);

/**
 * @prop --meu-componente-background-color: Meu-componente background-color
 * @prop --meu-componente-color: Paginator button border-radius
 */
```

Fonte: Otávio de Lima Soares

Figura 3.11 – Utilização de Design Tokens

```
:host {
  display: block;
  background-color: map-get($\textit{tokens}, meu-componente-background-color);
  color: map-get($\textit{tokens}, meu-componente-color);
}
```

Fonte: Otávio de Lima Soares

3.2.5 Publicação

Esta etapa do projeto teve como objetivo estruturar como ficaria o esquema de disponibilização de versões da biblioteca e definir a forma como essa biblioteca seria distribuída, para que outros colaboradores pudessem utilizá-las em seus projetos. Como a SYDLE já utilizava o Gitlab para gerenciar os repositórios do projeto optou-se por utilizar o Gitlab Package Registry para publicar os pacotes da SYDLE-UI, já que a ferramenta se encontra configurada com verificações de segurança, a fim de garantir que apenas colabores da SYDLE conseguissem instalar o pacote e utilizá-lo em seus projetos.

A publicação de pacotes pelo GitLab Package Registry é muito simples, para isso, basta criar um projeto com repositório no Gitlab e em seguida configurar o npm para realizar publicações neste projeto específico através dos comandos contidos na Figura 3.12.

Figura 3.12 – Comandos de configuração do npm

```
npm config set @minhaBiblioteca:registry https://gitlab.dominio.com/api
/v4/packages/npm/

npm config set -- '//gitlab.dominio.com/api/v4/packages/npm/:_authToken'
"<seu_token>"

npm config set always-auth true
```

Fonte: Otávio de Lima Soares

O primeiro comando é responsável por definir a url de download de pacotes pertencentes ao grupo "minhaBiblioteca".O segundo comando adiciona o token de acesso do usuário, que pode ser obtido acessando a aba "Tokens de Acesso"no perfil do usuário no gitlab e concatenado ao final da url. Este token é necessário para garantir o controle de instalção de pacotes privados. Por fim o terceiro comando define que é necessário autenticação de usuário para qualquer ação feita no Package Registry. Feito isso os pacotes podem ser publicados seguindo os passos padrão de publicação de pacotes npm.

3.3 Usando a SYDLE-UI no desenvolvimento de um Marketplace

A última etapa do projeto teve como objetivo realizar a implementação de um sistema web utilizando a SYDLE-UI. A aplicação desenvolvida foi uma plataforma de marketplace, onde a SYDLE pudesse disponibilizar alguns aplicativos, denominados SYBOX, prontos para serem baixados e aplicados à plataforma SYDLE ONE dos clientes.

O desenvolvimento do marketplace foi dividido em três etapas. A primeira etapa envolveu a criação de uma POC (Proof of concept), para validar a viabilidade da aplicação. Esta construção inicial foi feita com o desenvolvimento de componentes em Angular que utilizavam a SYDLE-UI em sua composição e seguiam o protótipo construído pelo time de UI/UX. Desta forma, foi possível testar a usabilidade da biblioteca, bem como a sua integração com projetos criados pelo Angular e detectar alguns pontos de melhoria nos componentes.

A segunda etapa teve como principais atividades a correção de todos os erros apontados durante o desenvolvimento da etapa anterior e criar alguns componentes específicos do marketplace, que passam a compor então a biblioteca SYDLE-UI MARKETPLACE. Com a criação desses novos

componentes houve um retorno à primeira etapa do projeto onde os componentes criados no AngularJS foram reescritos em Stencil, utilizando toda a estrutura apresentada na Sessão 3.2.1. Logo após configurou-se um ambiente de testes similar ao da SYDLE-UI e por fim foram desenvolvidos Design Tokens para os mesmos.

A terceira e última etapa foi a conclusão de todo o projeto, através da construção de fato do sistema web, utilizando a ferramenta de desenvolvimento AngularJS e os componentes implementados tanto na SYDLE-UI, quanto na SYDLE-UI MARKETPLACE. Como os componentes já estavam prontos e fiéis ao protótipo fornecido o desenvolvimento da interface da aplicação consistiu em combinar os componentes desenvolvidos ao longo do projeto para dar forma às páginas do Marketplace. A Figura 3.13 exemplifica como ficou a página inicial.

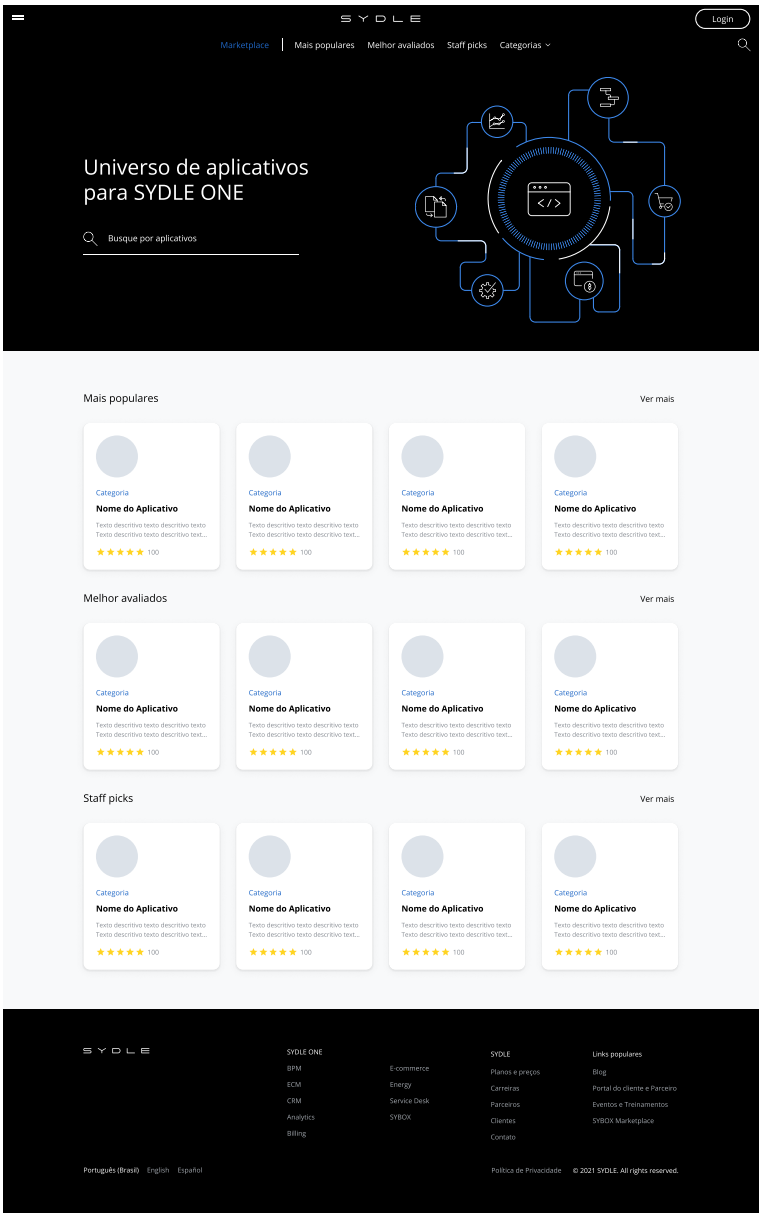
3.4 Considerações finais

Todo o processo de desenvolvimento da biblioteca da SYDLE-UI seguiu um sistema de etapas muito bem definido, como foi descrito nas Seções acima. Essa padronização garantiu um ótimo fluxo de trabalho e permitiu que todo o time estivesse em sintonia durante a construção dos componentes. A quebra das tarefas em etapas permitiu um desenvolvimento limpo e de extrema qualidade, uma vez que a cada avanço a fase anterior era validada, corrigida e incrementada.

Em contrapartida o desenvolvimento de Web Components foi um grande desafio, principalmente na parte de mapear os comportamento de cada componente, bem como a definição de métodos, escuta e emissão de eventos e definição de propriedades. Outro grande desafio na construção da SYDLE-UI foi a criação de Design Tokens o suficiente para garantir uma boa customização sem que os componentes fujam muito da identidade visual pré-estabelecida.

Foi um período de aprendizado muito intenso, a cada etapa era um tecnologia nova a se dominar, o que contribuiu muito para a minha formação na obtenção do título de bacharel em ciência da computação.

Figura 3.13 – Página inicial Marketplace



Fonte: Otávio de Lima Soares

4 CONCLUSÃO

O estágio realizado na SYDLE, como Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação proporcionou a aplicação dos conhecimentos adquiridos no período acadêmico, em um contexto de mercado de trabalho. As experiências adquiridas em cada etapa do trabalho foram as seguintes:

- A concepção da ideia da SYDLE-UI possibilitou um entendimento de como as empresas transformam suas necessidades em um produto que traga soluções para as mesmas e como o planejamento estratégico de um projeto funciona.
- A etapa de construção dos componentes da biblioteca proporcionou o desenvolvimento de habilidades de desenvolvimento de Web Components utilizando o Stencil e de documentação de componentes e interfaces no Storybook.
- Durante os testes dos componentes foram adquiridas habilidades a respeito de teste de software, principalmente de teste de interfaces. Além da familiarização com tecnologias como o Jest e o Puppeteer.
- O desenvolvimento de Design Tokens permitiu o conhecimento do conceito de desenvolvimento customizável e da geração de estilos multiplataforma com o Style Dictionary.
- A etapa de publicação da SYDLE-UI possibilitou o aprendizado sobre criação e distribuição de pacotes npm utilizando o Gitlab Package Registry.
- A etapa de construção do Marketplace possibilitou a ampliação do conhecimento de tecnologias web, como por exemplo o aprendizado do *framework* Angular.

Durante a execução do projeto algumas relações com conceitos adquiridos durante a graduação foram feitas. Os conhecimentos obtidos na disciplina de Programação Web foram essenciais para a realização do trabalho descrito neste documento. Conceitos como padrões de projeto e desenvolvimento com HTML e CSS foram vistos durante a disciplina e aplicados no estágio. A disciplina de Teste de Software proporcionou uma noção de como funciona os testes de ponta a ponta. A disciplina

de engenharia de software proporcionou o entendimento teórico das etapas de criação de um produto, o que ajudou na realização prática.

4.1 Trabalhos futuros

Os trabalhos futuros envolvem o desenvolvimento de novas aplicação utilizando a SYDLE-UI. Para este desenvolvimento será necessário o levantamento da necessidade de componentes específicos para o contexto da nova aplicação. Desta forma será necessário novamente a realização de todas as etapas descritas neste documento porém no contexto dos novos componentes levantados.

REFERÊNCIAS

AMAZON. **Style Dictionary**. [S.l.], 2021. Disponível em: <https://amzn.github.io/style-dictionary/#/README>. Acesso em: 6 abr. 2022.

CALOURIS, G. et al. **Sistemas Distribuídos**. 5. ed. São Paulo: bookman, 2013.

CHACON, S.; STRAUB, B. **Git Pro**. 2. ed. [S.l.]: Apress, 2022.

DELMARO, M. E.; MALDONADO, j. C.; JINO, M. **Introdução ao teste de software**. 2. ed. Rio de Janeiro: Elsevier, 2016.

FACEBOOK. **Jest**. [S.l.], 2022. Disponível em: <https://jestjs.io/>. Acesso em: 9 abr. 2022.

GIULIANO, A.; RILEY, W. M.; THOMAS, J. **Stencil: A Web Components Compiler**. [S.l.], 2022. Disponível em: <https://stenciljs.com/docs/introduction>. Acesso em: 4 abr. 2022.

GOOGLE. **What is Angular?** [S.l.], 2022. Disponível em: <https://angular.io/guide/what-is-angular>. Acesso em: 10 abr. 2022.

HANISCH, S.; THOMAS, J. **My First Component**. [S.l.], 2021. Disponível em: <https://stenciljs.com/docs/my-first-component>. Acesso em: 4 abr. 2022.

PIOVESAN, G. C. et al. **Web Components**. [S.l.], 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Web_Components>. Acesso em: 11 abr. 2022.

PUPETEER. **Puppeteer**. [S.l.], 2022. Disponível em: <https://pptr.dev/#?product=Puppeteer&version=v13.5.2&show=api-overview>. Acesso em: 9 abr. 2022.

STORYBOOK. **Introduction to Storybook for React**. [S.l.], 2022. Disponível em: <https://storybook.js.org/docs/react/get-started/introduction>. Acesso em: 5 abr. 2022.

YANG, J.; PAPAZOGLOU, M. P. Web component: A substrate for web service reuse and composition. In: PIDDUCK, A. B. et al. (Ed.). **Advanced Information Systems Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 21–36. ISBN 978-3-540-47961-1.