ARTHUR MOREIRA MAIA

REFATORAÇÃO DO *FRONT-END* DE UMA PLATAFORMA DIGITAL CORPORATIVA *ALL-IN-ONE*

LAVRAS – MG

2023

ARTHUR MOREIRA MAIA

REFATORAÇÃO DO *FRONT-END* DE UMA PLATAFORMA DIGITAL CORPORATIVA *ALL-IN-ONE*

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

2023

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Maia, Arthur Moreira

Refatoração do *front-end* de uma plataforma digital corporativa *all-in-one* / . – Lavras : UFLA, 2023.

42 p. :

Trabalho de Conclusão de Curso–Universidade Federal de Lavras, 2023.

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

Bibliografia.

ARTHUR MOREIRA MAIA

REFATORAÇÃO DO *FRONT-END* DE UMA PLATAFORMA DIGITAL CORPORATIVA *ALL-IN-ONE*

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 06 de Junho de 2023.

Prof. Dr. Maurício Ronny de Almeida Souza	UFLA
Prof. Dr. Eric Fernandes de Mello Araújo	UFLA
Dayvisson Valadão Martins Ferreira	SYDLE

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG
2023

Dedico este trabalho às pessoas mais importantes da minha vida: meus pais, Nilvane e Marcio e minhas tias, Ana e Angela. Sem o amor, a dedicação e o esforço deles, nada disso seria possível. Agradeço às minhas avós, Lerci e Lourdes, que infelizmente não puderam estar presentes para ver minha graduação, mas tenho certeza de que estão orgulhosas. Vocês me inspiraram, me motivaram e me ajudaram a superar todos os desafios. A todos vocês, minha eterna gratidão.

AGRADECIMENTOS

Agradeço ao meu orientador Maurício pela paciência e parceria, a minha família pelo apoio e a mim mesmo pela jornada.

"O amor é uma batalha difícil de vencer, mas vale a pena lutar por ele."
(Aang, Avatar: The Last Airbender.)

RESUMO

A SYDLE é uma empresa brasileira da área de Tecnologia da Informação, com sede em Belo Horizonte, Minas Gerais, que conta com plataforma digital corporativa all-in-one com soluções de BPM, CRM,*E-commerce*, *Billing*, *Service Desk* e mais. Para melhoria tanto desse produto como de todo o *front-end* produzido pela empresa foi desenvolvido um novo *design system* utilizando o conceito de *web-components* para gerar os componentes básicos do *front-end*, tais quais botões, entradas de texto, entre outros. Dando sequência a essa nova abordagem, deu-se início a refatoração de seu produto, o SYDLE ONE. Com o início da refatoração do produto criou-se o projeto SYDLE ONE COMPONENTS, projeto responsável por componentizar toda a parte que seria compartilhada entre essas aplicações mais especialistas (B log, *Service Desk*, SYDLE ONE), contando com formulários, menu de mensagens, entre outros. Quando este projeto começou a maturar, se deu início a refatoração do produto em si, com o projeto SYDLE ONE WORKSPACES.

Palavras-chave: Front-end. Web-components. Design system

ABSTRACT

SYDLE is a Brazilian company in the Information Technology sector, headquartered in Belo Horizonte, Minas Gerais. It offers an all-in-one corporate digital platform with solutions for BPM, CRM, E-commerce, Billing, Service Desk, and more. To improve both this product and the overall front-end developed by the company, a new design system was created using the concept of web components to generate basic front-end elements such as buttons, text inputs, and others. Following this new approach, the refactoring of their flagship product, SYDLE ONE, began. As part of the product refactoring, the SYDLE ONE COMPONENTS project was initiated to componentize shared elements across specialized applications like the Blog, Service Desk, and SYDLE ONE. This included forms, message menus, and other components. As the SYDLE ONE COMPONENTS project matured, the refactoring of the product itself commenced with the WORKSPACES project.

Keywords: Front end. Web-components. Design system.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A SYDLE é uma empresa especializada em tecnologia, com sede em Belo Horizonte, Minas Gerais, que tem como principal atuação o desenvolvimento de software. A empresa oferece soluções em automação e gestão de processos, análise de dados, e-commerce, service desk, entre outras. Sua plataforma principal, a SYDLE ONE, é uma solução integrada que reúne todas as ofertas da empresa em um único lugar. Com um portfólio diversificado de clientes em mais de 80 países, a empresa conta com grandes nomes como Bayer, Carrefour e Serasa.

O SYDLE ONE é uma plataforma corporativa *all-in-one* que tem como base a gestão de processos (BPM), a gestão de conteúdos (ECM) e a gestão à vista (Analytics), de forma a possibilitar a completa automatização dos processos de uma organização. Assim, em conjunto com as demais soluções - CRM [1], *E-commerce*, *Billing* [2], Service Desk [3], ITSM [4] - viabiliza a transformação digital de empresas. Um recurso adicional da plataforma é a capacidade de criar portais de relacionamento que envolvam a empresa e o seu público alvo, como clientes e fornecedores.

Neste contexto, a empresa reconheceu a importância de oferecer uma experiência do usuário moderna, intuitiva e fluida para se manter competitiva no mercado, assim decidindo refatorar o *front-end*, buscando aprimorar a usabilidade, a responsividade e o desempenho geral da plataforma. Isso não apenas melhoraria a satisfação do cliente, mas também atrairia novos usuários e reteria os existentes.

Adicionalmente, a adoção de um novo sistema de design e a utilização de web-components permitiriam uma melhor modularidade, reutilização e escalabilidade dos elementos do *front-end*. Essa

[1] A sigla CRM significa "Gestão de Relacionamento com o Cliente"em inglês (Customer Relationship Management). É uma estratégia de gestão empresarial que visa aumentar a satisfação dos clientes, fidelizá-los e, consequentemente, aumentar as vendas e a lucratividade da empresa. Isso é conseguido através da coleta e análise de dados sobre as interações dos clientes com a empresa, bem como a utilização de tecnologias, como softwares de CRM, para automatizar e integrar processos de marketing, vendas e atendimento ao cliente.

[2] Billing é o processo de cobrança de serviços ou produtos. É um componente fundamental em empresas que oferecem serviços a seus clientes, pois permite aos provedores de serviços manter um registro preciso das transações financeiras e garantir a cobrança adequada. O billing pode ser feito de maneira manual ou automatizada, utilizando sistemas de gestão de cobranças.

[3] Service Desk é uma plataforma de suporte ao usuário, que oferece soluções para problemas técnicos e de outros tipos, ajudando a garantir a satisfação do cliente. O Service Desk da SYDLE pode ser encontrado no seguinte link: <https://servicedesk.sydle.com/portal/home>

[4] Gerenciamento de Serviços de TI (da sigla em inlglês ITSM) é o conjunto de sistemas e processos que as empresas utilizam para aprimorar a maneira como a Tecnologia da Informação é utilizada.

abordagem pode proporcionar uma base sólida para o desenvolvimento futuro de recursos e facilitar a colaboração entre as equipes de desenvolvimento.

Dito isso, o objetivo deste trabalho é descrever as atividades desenvolvidas pelo autor durante o estágio supervisionado realizado na empresa SYDLE, cujo objetivo foi a refatoração do *front-end* do principal produto da empresa, o software SYDLE ONE. O estágio foi realizado no período de Agosto/2021 à Novembro/2022. A nova versão do *front-end* tinha como objetivo modernizar e atualizar a interface do usuário, além de melhorar sua usabilidade e funcionalidades.

O presente trabalho está organizado em capítulos que descrevem as atividades desenvolvidas pelo autor durante o estágio na SYDLE. O Capítulo 2 aborda os conhecimentos técnicos fundamentais para o desenvolvimento do estágio, oferecendo ao leitor uma familiarização com os conceitos relevantes para o trabalho. O Capítulo 3 detalha as atividades realizadas pelo aluno durante o período de estágio na empresa. No Capítulo 4, é feita a conclusão, com o objetivo de identificar quais habilidades e conhecimentos adquiridos previamente foram úteis ao aluno durante o processo, bem como o impacto da experiência para o seu desenvolvimento acadêmico e profissional.

2 CONCEITOS E TECNOLOGIAS

A plataforma *all-in-one*, o SYDLE ONE, da SYDLE é um sistema web que oferece uma ampla gama de funcionalidades aos seus usuários. Entre elas, estão a gestão de projetos, controle de estoque, emissão de notas fiscais, gerenciamento de recursos humanos, entre outras. Com a crescente demanda por uma interface mais moderna e intuitiva, a equipe de desenvolvimento decidiu refatorar todo o *front-end* da plataforma.

Nas subseções dessa sessão, serão abordadas diversas tecnologias e conceitos utilizados na refatoração do *front-end* do SYDLE ONE. Em primeiro lugar, será discutido o conceito de *design system*, sendo ele o sistema de design adotado pela empresa, que envolve a criação de um design consistente e padronizado para garantir uma experiência de usuário coesa e intuitiva em toda a plataforma. Em seguida, será explicado o conceito de *web-components*, que desempenhou um papel fundamental na criação tanto dos componentes básicos do *front-end*, como botões e entradas de texto, como os mais complexos, como menus especializados e até paginas completas. Esse conceito é fundamental para garantir a modularidade, reutilização e escalabilidade dos elementos do sistema.

Outro aspecto relevante que será abordado é a escolha das tecnologias e *frameworks* utilizados na refatoração, como HTML, CSS e JavaScript, juntamente com bibliotecas Stencil.JS, a qual foi utilizada para a criação dos *web-components*. O motivo pelo qual essas tecnologias foram selecionadas se deve a sua capacidade de fornecer uma arquitetura moderna, modular e um ecossistema ativo de suporte e desenvolvimento. Por fim, serão apresentados conceitos e práticas de desenvolvimento ágil, como o uso de metodologias como Scrum, que permitiram uma abordagem iterativa e colaborativa no processo de refatoração.

Ao abordar essas tecnologias e conceitos nas subseções, espera-se fornecer uma compreensão abrangente das escolhas feitas na refatoração do *front-end* do SYDLE ONE e como elas contribuíram para a melhoria do produto e do processo de desenvolvimento.

2.1 Desenvolvimento *front-end*

O desenvolvimento *front-end* é a criação da interface do usuário em um *website* ou aplicativo. É responsável por tudo que é visível e interativo na aplicação, incluindo a disposição de elementos

na página, cores, fontes, imagens e animações (FLANAGAN, 2011). O objetivo do desenvolvimento *front-end* é criar uma experiência de usuário agradável e intuitiva.

Existem várias tecnologias que podem ser utilizadas para o desenvolvimento *front-end*, incluindo HTML, CSS e JavaScript (FLANAGAN, 2011). Uma das mais recentes e promissoras tecnologias para o desenvolvimento *front-end* são os *web-components*, a qual é a tecnologia utilizada no projeto. Nas subseções seguintes serão detalhados conceitos como *design system* e *web-components*, os quais protagonizaram grande parte da jornada do estagiário em seu período de estágio.

2.1.1 Design system

O *design system* é uma abordagem essencial para a criação de interfaces coesas e eficientes. Além de definir os elementos e padrões que compõem a interface gráfica de um sistema, o *design system* promove a consistência visual em todas as partes do produto (MARKHAM, 2019). Essa padronização não apenas agiliza o desenvolvimento, uma vez que os componentes não precisam ser construídos do zero, mas também garante uma experiência de usuário unificada.

Dentro do contexto de um *design system*, o conceito de tokenização desempenha um papel crucial. A tokenização envolve a separação das propriedades de estilo em tokens reutilizáveis, que podem ser aplicados a diferentes elementos da interface (MARKHAM, 2019). Essa abordagem permite que alterações de estilo sejam feitas de maneira global, afetando automaticamente todos os elementos que utilizam o token correspondente. Com isso, a manutenção do sistema torna-se mais fácil e consistente, resultando em uma interface coesa.

Um exemplo de implementação da tokenização pode ser observado no Material-UI, um popular framework de interface de usuário para React (MATERIAL-UI, 2021). O Material-UI utiliza tokens para definir as propriedades de estilo dos componentes, como cores, tamanhos e espaçamentos. Dessa forma, é possível criar uma interface visualmente consistente e personalizável, onde os desenvolvedores podem ajustar os tokens para se adequarem ao design específico de seus projetos.

Outro exemplo relevante é o *design system* do governo brasileiro, conhecido como Gov.br. Esse sistema foi criado com o propósito de fornecer diretrizes e recursos para a construção de interfaces consistentes em diferentes órgãos governamentais (BRASILEIRO, 2021). Ele visa promover

a usabilidade e acessibilidade dos serviços públicos online, permitindo uma experiência unificada e intuitiva para os cidadãos.

No caso da SYDLE, a implementação do seu próprio *design system*, a SYDLE UI, possibilitou a criação de interfaces comuns para diversas marcas, preservando a identidade visual individual de cada uma delas. O estágio do aluno na empresa proporcionou a oportunidade de contribuir ativamente nesse projeto, adquirindo valiosos conhecimentos sobre o desenvolvimento de *design systems* e sua importância na criação de produtos de qualidade.

2.1.2 Web-components

Os *web-components* são um conjunto de tecnologias para criação de elementos reutilizáveis e personalizados na web. Eles permitem que desenvolvedores criem seus próprios elementos personalizados, com funcionalidades específicas, que podem ser utilizados em várias páginas e aplicações (W3C, 2021). Dessa forma, os desenvolvedores podem criar componentes personalizados e reutilizáveis, que tornam o desenvolvimento mais rápido e fácil.

Um dos principais benefícios dos *web-components* é a capacidade de encapsular código e estilos dentro do próprio componente (PSR, 221). Isso permite que os componentes sejam facilmente utilizados e personalizados em diferentes projetos, sem que haja conflitos entre os estilos de diferentes componentes na mesma página. Outra vantagem é a capacidade de tokenizar estilos de componentes. Isso significa que é possível definir estilos personalizados para um componente, utilizando variáveis CSS, que podem ser facilmente alteradas para personalizar a aparência do componente em diferentes contextos (W3C, 2021).

A tokenização de estilos de componentes é particularmente útil em projetos de grande escala, como o SYDLE ONE, onde há vários desenvolvedores trabalhando em diferentes partes da aplicação. Ela permite que os desenvolvedores definam e compartilhem uma biblioteca de estilos personalizados, que podem ser utilizados em vários componentes em diferentes partes da aplicação.

A equipe de desenvolvimento *front-end* do SYDLE ONE utiliza o *StencilJS*, desenvolvido pela equipe da Ionic (IONIC, 2021), uma das bibliotecas mais populares para desenvolvimento de *web-components*, possuindo uma sintaxe simples e intuitiva e utilizando uma abordagem de pré-compilação

para melhorar o desempenho e a compatibilidade dos componentes (STENCILJS, 2021a). O StencilJS é baseado em tecnologias web padrão, como HTML, CSS e JavaScript, e é compatível com todos os principais navegadores. Além disso, o *StencilJS* também oferece recursos avançados de estilização, como a tokenização de estilos de componentes (STENCILJS, 2021a; STENCILJS, 2021b).

No contexto da SYDLE, onde existem diversos projetos em andamento, a estrutura de *web-components* trouxe uma padronização e organização para o desenvolvimento *front-end*. Os componentes foram projetados de forma independente, podendo ser reutilizados em diferentes projetos e aplicações. Essa abordagem permitiu uma maior eficiência na implementação de interfaces, uma vez que os componentes já estavam prontos e testados, reduzindo o esforço de desenvolvimento.

Além disso, a utilização de *web-components* promoveu uma arquitetura modular, em que cada projeto possuía seus próprios componentes personalizados, mas também podia aproveitar componentes compartilhados entre os diferentes projetos. Isso resultou em uma melhor organização do código e uma maior facilidade na manutenção e evolução dos projetos. A equipe de desenvolvimento conseguiu trabalhar de forma mais ágil e colaborativa, pois os componentes eram independentes e podiam ser atualizados e aprimorados separadamente.

Em suma, a escolha da SYDLE em utilizar *web-components* se baseou nos benefícios de reutilização, modularidade e padronização que essa abordagem oferece. A estrutura entre os diversos projetos foi estabelecida com base na criação e compartilhamento de componentes personalizados, resultando em um desenvolvimento mais eficiente, maior consistência visual e uma arquitetura escalável e modularizada.

2.2 Metodologias Ágeis

Metodologias ágeis são abordagens de gestão de projetos que priorizam a flexibilidade, colaboração e entrega constante de valor para o cliente. Segundo o manifesto ágil, as metodologias ágeis valorizam "indivíduos e interações mais que processos e ferramentas, software em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos, e resposta a mudanças mais que seguir um plano"(Agile Alliance, 2001). Entre as metodologias ágeis mais conhecidas, destacam-se o Scrum, o Kanban e o Extreme Programming (XP).

O Scrum é uma metodologia de gerenciamento de projetos que se baseia em ciclos de trabalho chamados *sprints*, onde a equipe desenvolve e entrega pequenos incrementos de funcionalidades em um curto período de tempo. O processo é centrado no cliente e enfatiza a comunicação, colaboração e transparência entre a equipe e as partes interessadas (SCHWABER, 2002).

Já o Kanban é uma metodologia que se concentra no fluxo de trabalho, visualização do progresso e limitação do trabalho em progresso. O objetivo é identificar gargalos, reduzir o tempo de ciclo e aumentar a eficiência do processo. O Kanban pode ser aplicado em conjunto com outras metodologias ágeis, como o Scrum (ANDERSON, 2010).

O Extreme Programming (XP) é uma metodologia que valoriza a qualidade do software e a satisfação do cliente. Para alcançar esses objetivos, o XP enfatiza a comunicação constante entre a equipe e o cliente, a entrega constante de valor, o desenvolvimento orientado a testes e a programação em pares (BECK, 2000).

As metodologias ágeis têm sido amplamente adotadas no desenvolvimento de software devido aos seus benefícios em relação à flexibilidade, qualidade e satisfação do cliente. Na subseção seguinte será detalhada a metodologia ágil SCRUM, a qual foi utilizada pela equipe do estagiário durante seu período na empresa.

2.2.1 SCRUM

O SCRUM é uma metodologia ágil que tem como objetivo principal a entrega de valor de forma iterativa e incremental, através da organização e gestão de um time multifuncional e auto-organizável (SUTHERLAND; SCHWABER, 2020). É conhecido por utilizar uma série de artefatos, papéis e eventos para garantir a entrega de valor de forma iterativa e incremental (SUTHERLAND; SCHWABER, 2020).

Os artefatos do SCRUM incluem o *Product Backlog*, que contém uma lista ordenada de itens de trabalho que representam as necessidades do produto; o *Sprint Backlog*, que é uma lista de itens de trabalho selecionados do *Product Backlog* para serem desenvolvidos durante uma *Sprint*; e o Incremento do produto, que é a soma de todos os itens concluídos durante um *Sprint* e as versões anteriores do produto (SUTHERLAND; SCHWABER, 2020).

Os papéis do SCRUM incluem o *Scrum Master*, que é responsável por garantir que a equipe SCRUM esteja seguindo as práticas e eventos corretos; o *Product Owner*, que é responsável por definir as necessidades do produto e priorizar o *Product Backlog*; e o time de desenvolvimento, que é responsável por transformar os itens do *Product Backlog* em um Incremento de produto (SUTHERLAND; SCHWABER, 2020).

Finalmente, os eventos do SCRUM incluem a *Sprint*, que é o período de tempo durante o qual o time de desenvolvimento trabalha para entregar o Incremento do produto; a Reunião de Planejamento da *Sprint*, que é realizada no início da *Sprint* e tem como objetivo definir o que será feito durante a *Sprint*; a Reunião Diária SCRUM, que é uma reunião diária de 15 minutos realizada pelo time de desenvolvimento para revisar o progresso e planejar o próximo dia de trabalho; a Revisão da *Sprint*, que é realizada no final da *Sprint* e tem como objetivo revisar o Incremento do produto; e a Retrospectiva da *Sprint*, que é uma reunião realizada após a Revisão da Sprint para discutir como o time pode melhorar seu processo (SUTHERLAND; SCHWABER, 2020).

No contexto da equipe em que o estagiário atuou, adaptou-se alguns conceitos do SCRUM para atender as necessidades específicas do time. Em relação aos papéis, não havia a definição oficial nem de um *Product Owner* nem de um *Scrum Master*, mas sim a figura de um *Tech Lead* (Um *Tech Lead* atua como uma referência técnica nos projetos de TI, na equipe de desenvolvimento) que realizava também esses papéis. Sobre os eventos, o momento de revisão da *Sprint* acontecia na revisão individual dos produtos gerados por cada desenvolvedor, através da plataforma GitLab, por meio de *merge requests*. O restante dos eventos e os artefatos permaneceram inalterados.

3 REFATORAÇÃO DO *FRONT-END* DO SYDLE-ONE

Este capítulo apresenta as atividades desenvolvidas pelo estagiário na empresa SYDLE, durante o período de agosto de 2021 a dezembro de 2022. O objetivo do estágio foi a refatoração do *front-end* da plataforma SYDLE ONE, o qual consistiu na recriação de todo o *front-end* da plataforma, com a finalidade de melhorar sua estrutura de código, atualizar tecnologias e modernizar a aparência.

Como primeiro passo para a efetivação dessa tarefa, foi criada a biblioteca SYDLE UI, que representa o *design system* da empresa e agiliza o desenvolvimento fornecendo componentes web para criação de páginas complexas. Em seguida, o estagiário trabalhou no projeto SYDLE ONE COMPONENTS, que fornece componentes mais complexos com regras de negócio para os principais produtos da empresa, incluindo o SYDLE ONE, *Service Desk* e o Blog [1].

Por fim, o estagiário participou do projeto SYDLE ONE WORKSPACES, que representa o *front-end* da plataforma SYDLE ONE, utilizando tanto a SYDLE UI quanto o projeto SYDLE ONE COMPONENTS.

As seções seguintes detalham a jornada do estagiário, incluindo o treinamento (seção 3.1), participação nos projetos SYDLE UI (Seção 3.3), projeto SYDLE ONE COMPONENTS (Seção 3.3) e projeto SYDLE ONE WORKSPACES (Seção 3.4).

3.1 Treinamento

O estagiário participou de duas etapas de treinamento para atuar de forma efetiva na empresa SYDLE. A primeira consistiu em um minicurso oferecido pela empresa para familiarização com o funcionamento da plataforma SYDLE ONE. A segunda etapa foi composta por estudos autônomos, abrangendo temas técnicos práticos e teóricos das tecnologias utilizadas na equipe.

3.1.1 Treinamento SYDLE ONE

Nesta primeira etapa, o estagiário participou de um curso de treinamento da plataforma SYDLE ONE que, apesar de ser produzido pela SYDLE, foi ministrado por um integrante da empresa

[1] <https://www.sydle.com/br/blog/>

parceira chamada LEVTY [2], empresa essa que também utiliza o SYDLE ONE como principal provedora de suas aplicações. O treinamento durou uma semana e era realizado em conjunto com os novos ingressantes tanto da SYDLE quanto da LEVTY. Para a comunicação entre o ministrante e os participantes do treinamento, foi criado um canal no aplicativo de comunicação Discord [3], onde aconteciam encontros diários em que o instrutor introduzia os temas a serem estudados durante o dia e explicava conceitos e funções da plataforma. Após o encontro se dava início um estudo autônomo individual através do portal Service Desk, que contava com video aulas e material escrito. Nesse material eram demonstrados funcionamentos da plataforma e apresentados atividades práticas com tutoriais a serem realizados pelos alunos. O ambiente de teste da plataforma SYDLE ONE Foundations foi disponibilizado para a prática. O material era dividido da seguinte forma: (i) Introdução, (ii) ECM: Modelagem de dados, (iii) BPM: Modelagem de Processos, (iv) Automação e (v) Conclusão.

Na Introdução (i), foram tratados temas introdutórios sobre os fundamentos e o propósito do SYDLE ONE. Foi apresentado o conceito de CRM, *Service Desk* e *Billing*, com exemplos reais.

Na seção "ECM: Modelagem de dados" (ii) o conteúdo começa informando sobre dados, a importância de modelá-los e apresenta exemplos de aplicações de modelagem com um exercício mental. Em seguida, é aplicado esses conceitos ao escopo da plataforma, explicando o paradigma "Classe-Objeto" e mostrando como eles são constituídos. A classe é descrita como uma entidade modeladora com identificação, conteúdo, relacionamentos e comportamentos, enquanto os objetos são contêineres que armazenam os registros dessas entidades. Há então um vídeo tutorial que mostra como aplicar essa modelagem no ONE. Em seguida, é apresentado como conectar essas entidades, explicando os conceitos internos de referências simples e embutidas e mostrando essas funcionalidades em outro vídeo tutorial. Depois, há um exercício prático que consiste na criação de classes Pessoa, Empresa e Contrato e uma classe conexão para uni-las, com um tutorial descrito em texto e a ser realizado no SYDLE ONE Foundations, e um desafio para testar os conhecimentos adquiridos. O conceito de In-

[2] A LEVTY é uma empresa brasileira com sede em Juiz de Fora, MG, que oferece soluções empresariais inovadoras. Mais informações sobre a empresa podem ser encontradas em seu site oficial, disponível em <https://www.levty.com>.

[3] Discord é uma plataforma de comunicação por voz, vídeo e texto, voltada para comunidades de jogos, mas também é usada para outras finalidades. Disponível para desktop, dispositivos móveis e navegadores, o Discord oferece recursos como canais de texto, voz e vídeo, upload de arquivos, integrações com outros aplicativos e muito mais. Para mais informações, visite o site oficial em <https://discord.com>.

terface é apresentado juntamente com um vídeo de demonstração e o assunto de segurança de dados é abordado, incluindo conceitos como gestão de acessos, gestão de usuários e permissionamento. Também é tratada a visualização de dados, mostrando os tipos de visualização, com vídeos de aplicação. Por fim, o capítulo apresenta a segunda atividade prática do curso, complementando o produto criado na primeira aplicação, incorporando todos os conceitos abordados até então.

A seção dedicada a "BPM: Modelagem de Processos" (iii) é iniciada com uma introdução ao assunto, definindo o que é um processo, como funciona a gestão por processos e o que é gestão de processos de negócio (BPM). Em seguida, é apresentado o tópico principal de BPMN, que é a notação gráfica usada no BPM. Explica-se cada elemento da notação, incluindo linhas, tipos de eventos (simples, mensagem, sinal, conexão, temporal, erro), atividades, eventos acoplados e *gateways*. Depois, é mostrado como identificar regras de negócio através da modelagem de processos e como é feito o versionamento desses processos. Finalmente, a seção inclui uma atividade prática, onde um processo de reembolso de despesas foi criado.

A seção sobre "Automação" (iv) inicia explicando o conceito de automação e sua importância, sem necessidade de conhecimento em programação. São apresentados os métodos padrões disponíveis no SYDLE ONE, seguido por uma atividade prática para validar CPFs. Em seguida, é abordado a criação de métodos customizados, também com uma atividade prática. O próximo tópico são as APIs, incluindo a API do SYDLE ONE e APIs externas, com uma atividade prática para demonstrar como realizar chamadas a elas. Finalmente, o processo anterior é automatizado, com remodelação, adição de atributos, automatização de tarefas, criação de papéis dinâmicos, decisões de fluxo automatizadas, publicação, testes e monitoramento pelo *Analytics*.

A seção de "Conclusão" (v) agradece pela realização do treinamento e oferece conteúdos e exercícios extras, especialmente para desenvolvedores. Tópicos incluem Elasticsearch [4], SYBOX [5], atividades para processos com envio de e-mails, temporizadores e eventos de mensagem, e desafios extras.

[4] Elasticsearch é um motor de busca distribuído de código aberto, baseado em Apache Lucene, que fornece análise de dados em tempo real, visibilidade e gerenciamento de dados de várias fontes. Saiba mais sobre Elasticsearch em <https://www.elastic.co/what-is/elasticsearch>.

[5] SYBOX é uma solução oferecida pela SYDLE, com o objetivo de automatizar processos de negócios. Para mais informações, visite o site <sydle.com/br/sybox/>.

3.1.2 *Roadmap* de aprendizado

Depois de concluir o treinamento na plataforma, o estagiário conheceu seu time e foi contextualizado sobre os projetos da equipe e indicado temas importantes para estudo, a fim de adquirir conhecimentos essenciais para seu desenvolvimento dentro do time. Seu líder apresentou um *roadmap* [6] (Figura 3.1) de aprendizado para desenvolvimento *front-end*, incluindo diversos tópicos da área, como HTML, CSS, SASS, JavaScript, TypeScript, NPM, *Web-Components*, Stencil, Git/GitHub/GitLab, Bootstrap, Angular, React, Yarn e DevTools. O *roadmap* fornecido pelo líder incluía referências a materiais online gratuitos para estudo independente, o que ajudou o estagiário a se familiarizar com os tópicos. A principal referência utilizada foi o canal do YouTube "Traversy Media" [7], que mostrava como utilizar as tecnologias de *front-end*. O estudo dos tópicos selecionados demandou cerca de duas semanas.

3.2 SYDLE UI

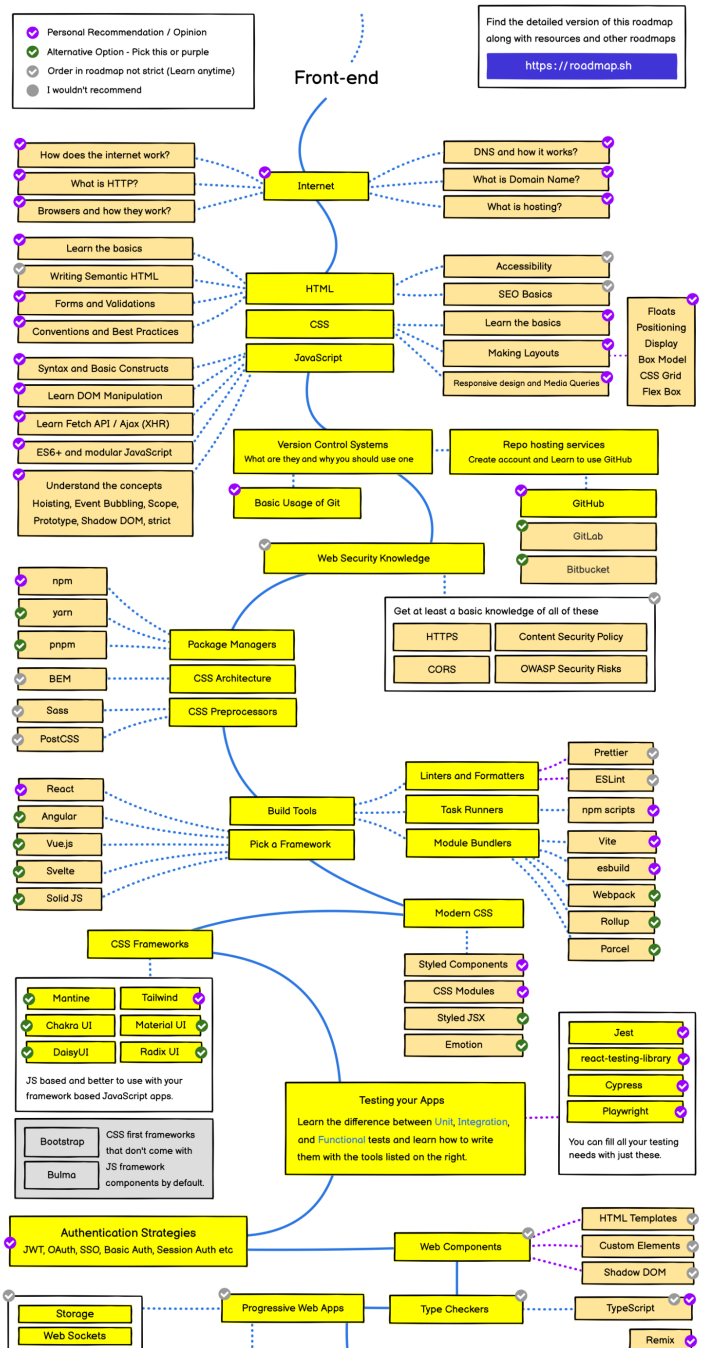
Após a realização de todo o treinamento, o estagiário começou a atuar como desenvolvedor no projeto SYDLE UI, o *design system* da SYDLE. Devido ao caráter multimarca da SYDLE (pois além do SYDLE ONE, a empresa conta com outros produtos, como uma plataforma de *e-commerce* personalizável para clientes e um ambiente virtual de aprendizagem (AVA), entre outros), a biblioteca SYDLE UI oferece componentes visualmente flexíveis, que podem ser facilmente modificados através de *design tokens*, além de serem compatíveis com qualquer marca ou *framework*, além primorar e agilizar o desenvolvimento de aplicações da empresa. A trajetória do estagiário neste projeto se deu desde a criação de componentes, tokenização de regras de CSS e correções de *bugs* (atividade que é denominada 'Operação').

3.2.1 Criação de componentes

A participação do estagiário no time de desenvolvimento começou com uma sessão de programação em par junto com o líder de equipe. O objetivo era criar um componente para exibir textos

[6] <https://roadmap.sh/frontend>
[7] <https://www.youtube.com/@TraversyMedia>

Figura 3.1 – Trecho do *roadmap* de desenvolvimento *front end*.



Fonte: <https://roadmap.sh/frontend>

de maneira dinâmica e estilizada. Para isso, o primeiro passo foi criar a estrutura de pastas e arquivos necessários para armazenar o código do componente. Isso foi realizado usando o comando `npm run generate [nome da componente]`, que é fornecido pela biblioteca StencilJS e permite gerar um arquivo com as informações relevantes para a renderização de um componente.

Após a criação da estrutura, o próximo passo foi implementar a lógica do componente. O componente precisava ter a capacidade de receber um tipo de texto através de um parâmetro, incluindo opções como `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `body`, `small`, `code` e `lead`. Além disso, o componente deveria aceitar o texto a ser exibido como um slot, permitindo que o usuário insira o conteúdo entre as tags HTML que representam a componente, como pode ser observado na Figura 3.2. A componente deveria então renderizar o texto de acordo com os estilos definidos pelas diretrizes de design da empresa. A Figura 3.3 representa o resultado gerado pelo componente de texto.

Figura 3.2 – Pseudocódigo do componente de renderização de texto.

Fonte: Do autor

Figura 3.3 – Captura de tela do resultado gerado pelo componente de texto, renderizando o texto 'Sample text' no estilo 'h1'.

Como o projeto já estava em estado avançado, o estagiário não participou da criação de nenhum outro componente, porém teve participação efetiva na aplicação de tokens de CSS nos componentes existentes e na melhoria e correção de*bugs*. Ambos os temas serão abordados nos tópicos a seguir.

3.2.2 Tokenização

Como a SYDLE é uma empresa com caráter multimarca, um dos requisitos para a criação da SYDLE UI era que a biblioteca deveria ser adaptável aos estilos de marca de clientes. Para atingir esse objetivo, aplicou-se o conceito de *design tokens* e *component tokens*. Essa abordagem flexibiliza a aparência dos componentes, definindo variáveis de CSS através de tokens de estilo que podem ser alterados facilmente.

Os tokens base do projeto são divididos nas seguintes categorias:

- *Breakpoints*: Tokens utilizados para melhorar a responsividade dos elementos. Por exemplo, se um componente deve ter uma aparência diferente em telas menores, podemos definir um ponto de quebra com tamanho específico e ajustar o comportamento do componente nesse ponto.

- *Color*: São tokens de cores que contém uma paleta definida para trazer consistência e reconhecimento de marca nos produtos da empresa. Por exemplo, se o tom de verde da empresa é definido como `00B28B`, esse token pode ser utilizado em todo o projeto para garantir a consistência da cor.

- *Effect*: Define valores para propriedades de efeito, como sombras. Por exemplo, podemos definir o tamanho e a opacidade da sombra para um componente específico.

- *Grid*: Aqui estão definidos os valores para o posicionamento dos elementos em grid, como o número de colunas disponíveis, o espaço entre linhas e colunas e a distância da grade em relação as bordas da página. Por exemplo, se a grade tem 12 colunas, podemos usar os tokens para posicionar um componente em 3 colunas ou 9 colunas, segundo as necessidades.

- *Radius*: Usado para definir a curvatura dos cantos das bordas de componentes, como quinas bem agudas ou bem arredondadas. Por exemplo, se queremos que um botão tenha cantos arredondados, podemos usar o token para definir o valor do raio.

- *Spacing*: Trata-se de unidades de espaçamento, ou seja, o espaço negativo entre elementos e componentes, geralmente utilizado em regras de *padding* e *margin*. Por exemplo, se precisamos

de 16 pixels de espaçamento entre dois elementos, podemos usar o token correspondente para garantir a consistência do espaçamento em todo o projeto.

- *Typography*: Tokens para a tipografia, definindo fonte, tamanho, altura da linha, espaçamento entre letras e palavras e hierarquia textual utilizados.

Com os *design tokens* definidos, começou-se a aplicação deles nos componentes, uma vez que eles haviam sido desenvolvidos sem sua utilização. Para isso, foi necessário revisitar todos os componentes para atualizar suas regras de CSS e aplicar os *tokens*.

Para garantir que tudo fosse realizado de maneira eficiente, foi criado um *backlog* de atividades, incluindo todos os componentes. Cada desenvolvedor escolheu então qual componente gostaria de adicionar os *tokens* e assumiu a tarefa.

A aplicação dos *tokens* um componente começava com a criação dos *component tokens* para esse elemento. Para isso, era criado um arquivo chamado "tokens.js" na pasta da componente e preenchido com os dados relevantes parametrizados pelo *design system*, assim como podemos conferir na Figura 3.4. Esse arquivo servia como base para o motor de *tokens* do projeto gerar os *component tokens* específicos daquele componente e ser utilizados nas regras de CSS. Após a execução do comando `npm build component-tokens [nome da componente]` eram gerado os *component tokens*, que, quando aplicados ao componente, faziam com que o conteúdo do arquivo de CSS de um componente, como o demonstrado na Figura 3.5, pudesse ser modificado em um arquivo semelhante ao retratado na Figura 3.6.

:

3.2.3 Operação

Depois da aplicação de *tokens* nos componentes, a versão de lançamento foi finalmente publicada. Esta versão permitiu que os times começassem a usar a biblioteca em seus respectivos projetos, incluindo o Blog, o Portal de Relacionamentos e o SYDLE ONE. No entanto, assim que a SYDLE UI começou a ser utilizada em diferentes ambientes, foram identificados vários erros e oportunidades de melhoria, uma vez que o projeto era novo e estava sendo testado pela primeira vez em produção.

Figura 3.4 – Exemplo do arquivo “tokens.js” do componente ’Avatar’

```
module.exports = {
  avatar: {
    'size':
       {
        value: '{design-system-spacer.7.value}',
        doc: 'Avatar regular size.',
      },
    },
    'border': {
      'radius': {
        value: '{design-system-border-radius.value}',
        doc: 'Default border radius.',
      },
    },
    'color': {
      value: '{design-system-color-emphasis.value}',
      doc: 'Avatar font color',
    },
    'background-color': {
      value: '{design-system-color-base.value}',
      doc: 'Avatar background color',
    },
  };
```

Fonte: Do autor

No contexto SYDLE, o termo “Operação” se refere aos esforços para corrigir erros e adicionar melhorias aos sistemas. Para realizar a operação na SYDLE UI, foi necessário estabelecer uma colaboração entre os times para garantir a continuidade das manutenções no projeto e evitar sua obsolescência. O objetivo era fazer com que o projeto evoluísse ainda mais, já que o foco principal já não estava mais na biblioteca. Sempre que um erro era identificado ou uma melhoria solicitada, uma *issue*

Figura 3.5 – Regras de CSS do componente “Avatar” antes da aplicação dos *component tokens*

Fonte: Do autor

era criada no *backlog* do projeto para que os membros dos times pudessem atuar. A criação de *issues*, mesmo para correções pontuais, é importante para manter o controle de atividades e documentação do projeto, já que a biblioteca conta com versionamentos e atualizações. Desta forma, é possível ter uma nota de lançamento com as melhorias e correções de *bugs* em cada nova versão.

Com o decorrer do tempo, um comportamento foi sendo adotado naturalmente pelos times. Quando um componente apresentava problemas ou requeria melhorias, o desenvolvedor, ao qual as atividades dependiam desses problemas ou melhorias para serem realizadas, criava uma *issue* e a resolvia imediatamente ou em um momento oportuno. Por exemplo, quando um *bug* simples na componente de entrada de texto foi identificado pelo estagiário que atrapalhava o funcionamento da componente que estava sendo criada por ele no projeto SYDLE ONE WORKSPACES. O *bug* acontecia pois o componente de entrada de texto possuía a possibilidade de mostrar um botão interno que era capaz de limpar o conteúdo ali inserido, entretanto quando tal botão era clicado a componente não disponibilizava nenhuma informação de que a ação teria sido feita. Portanto, assim que identificou o mal funcionamento, ele criou uma *issue* e a corrigiu.

Figura 3.6 – Regras de CSS do componente “Avatar” depois da aplicação dos *component tokens*.

```
:host {
  ...
  width: map-get($tokens, avatar-size);
  height: map-get($tokens, avatar-size);
  color: map-get($tokens, avatar-color);
  background-color: map-get($tokens, avatar-background-
color);
  border-radius: map-get($tokens, avatar-border-
radius);
}
```

Fonte: Do autor

Durante o período de um ano e três meses de atuação do estagiário na empresa, ele realizou inúmeras atividades de operação que tornam impossível mencioná-las todas. Este trabalho ajudou o estagiário a compreender a estrutura dos componentes existentes e a desenvolver habilidades para detectar erros no código, aprimorando sua capacidade de rastreamento de erros e a compreensão do fluxo de funcionamento dos componentes. Além disso, contribuiu significativamente para a melhoria do código e para a otimização da performance através da refatoração.

Na próxima seção, será discutido em detalhes o projeto SYDLE ONE COMPONENTS. Durante o desenvolvimento deste projeto, foram identificados vários problemas de comportamento e identificadas várias oportunidades de melhoria na SYDLE-UI, o que exigiu ampla atuação do estagiário na manutenção dos componentes do projeto. Essa atuação permitiu que o estagiário compreendesse completamente a estrutura do projeto, se familiarizasse com a estrutura de componentes web e adquirisse habilidade tanto na linguagem TypeScript quanto na biblioteca StencilJS.

3.3 Projeto SYDLE ONE COMPONENTS

O projeto SYDLE ONE COMPONENTS é uma iniciativa que visa fornecer acesso a componentes básicos da plataforma, como formulários e menus especializados (por exemplo, menus de criação, de mensagens, de usuários), com informações obtidas do banco de dados. Durante o projeto, o estagiário desempenhou um papel importante tanto na criação da arquitetura do projeto quanto na criação de diversos componentes, destacando o campo de coordenada geográfica que enriqueceu o conhecimento sobre utilização de APIs externas (Google Maps API), o *card* de objeto (*object card*), que teve grande papel no aprendizado de manipulação de HTML e CSS, o menu de mensagens que uniu tanto habilidades de CSS quanto a utilização de requisições a API interna da plataforma e o campo de YouTube que, além de exercitar os fundamentos sobre desenvolvimento de *web-components*, foi importante para o exercício de criação de documentação técnica.

3.3.1 Campo de coordenada geográfica

Um formulário é um elemento crucial para a interação do usuário com uma plataforma online. Ele é composto por diversos campos que permitem que o usuário insira informações relevantes. Entre esses campos, destaca-se o campo de coordenada geográfica. Esse campo é responsável por receber as coordenadas de latitude e longitude do usuário e renderizar um mapa com um marcador que representa a localização inserida.

No modo de edição, o campo de coordenada geográfica é representado por dois inputs de texto e um botão, como pode ser visualizado na Figura 3.7. A primeira entrada de texto é responsável pelo valor de latitude e o segundo pelo valor de longitude. Os inputs só permitem a adição de números e caracteres especiais, como "-" e ".", para representar as coordenadas. O botão aciona um modal com um mapa do Google Maps, onde o usuário pode selecionar qualquer local na Terra. Ao fechar a janela modal, os campos de texto são preenchidos automaticamente com as coordenadas selecionadas no mapa. Se as coordenadas preenchidas forem alteradas em um dos campos de entrada, a seleção do mapa também muda. Além disso, o mapa em modo de edição apresenta uma barra de pesquisa para facilitar a busca por locais específicos.

No modo de leitura, o campo de coordenada geográfica apresenta apenas um botão com o ícone de um mapa com alfinete. Ao ser clicado, esse botão abre um mapa navegável que não pode ter o seu marcador alterado. O marcador é exibido com um *tooltip* contendo o *card* do objeto ao qual está vinculado. Em modo de leitura, o mapa não apresenta uma barra de pesquisa. O mapa em modo de leitura oferece controles de visualização (Mapa e Satélite), controle de zoom, *Street View* e a possibilidade de maximizar para ocupar toda a tela do monitor.

Figura 3.7 – Campo de coordenada geográfica em modo de edição.

Fonte: Do autor

O estagiário adquiriu conhecimento sobre a API do Google Maps para construir o campo navegável, o que foi possível graças à documentação completa disponível [8]. O processo de aprendizado foi direto, uma vez que o estagiário seguiu os tutoriais disponíveis e adaptou para os padrões requeridos pelo componente. As principais funcionalidades da API utilizadas pelo estagiário foram a manipulação de marcadores no mapa, modificação do layout dos componentes da interface do mapa e comunicação de coordenadas entre os inputs do campo e o mapa. Isso significa que o mapa precisava estar ciente de quando uma coordenada mudava na entrada de texto e precisava comunicar aos campos de texto se alguma coordenada era modificada no mapa e vice-versa.

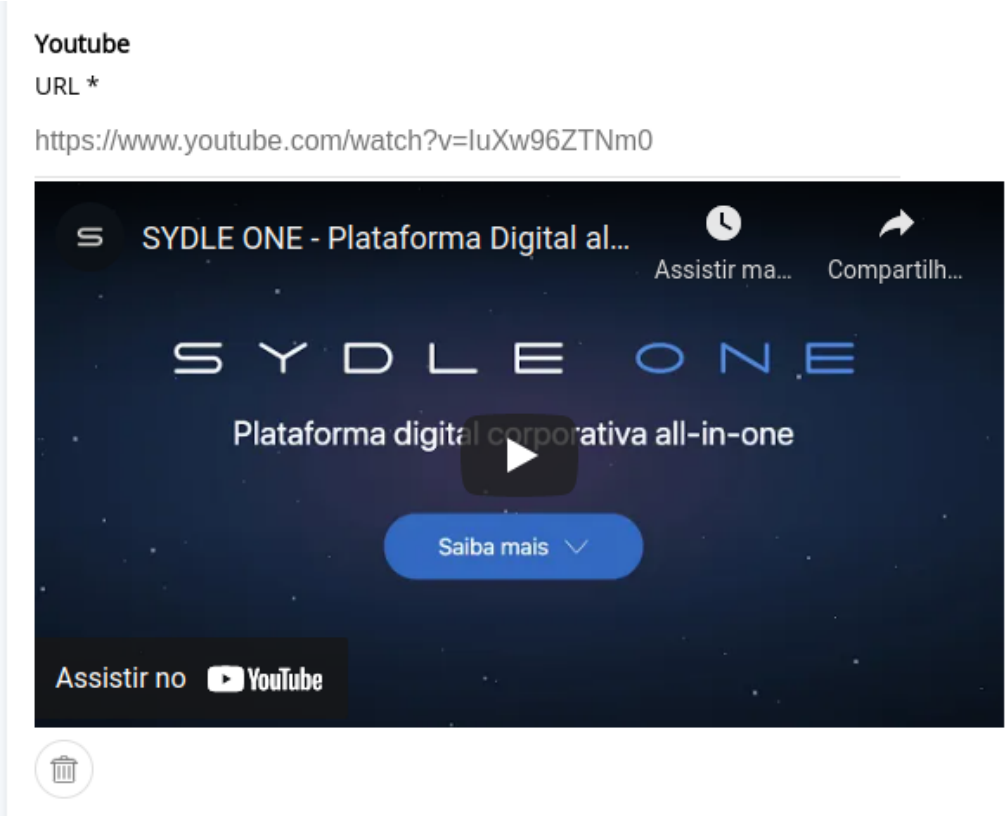
3.3.2 YouTube Field

O *YouTube Field* é uma implementação que serve como exemplo para a documentação de uma nova funcionalidade da plataforma que permite a criação de campos customizados pelo usuário. O papel do estagiário foi documentar essa funcionalidade, e a documentação resultante foi disponibilizada no portal *Service Desk*.

[8] <https://developers.google.com/maps/documentation/javascript?hl=pt-br>

A documentação começa listando os requisitos que a componente que representaria aquele campo deveria ter, incluindo as propriedades [9] "*field*", "*value*" e "*path*", bem como os métodos "*changeValue*" e "*disconnectedCallback*".

Figura 3.8 – Captura de tela do formulário com o campo de "Youtube" no modo de edição.

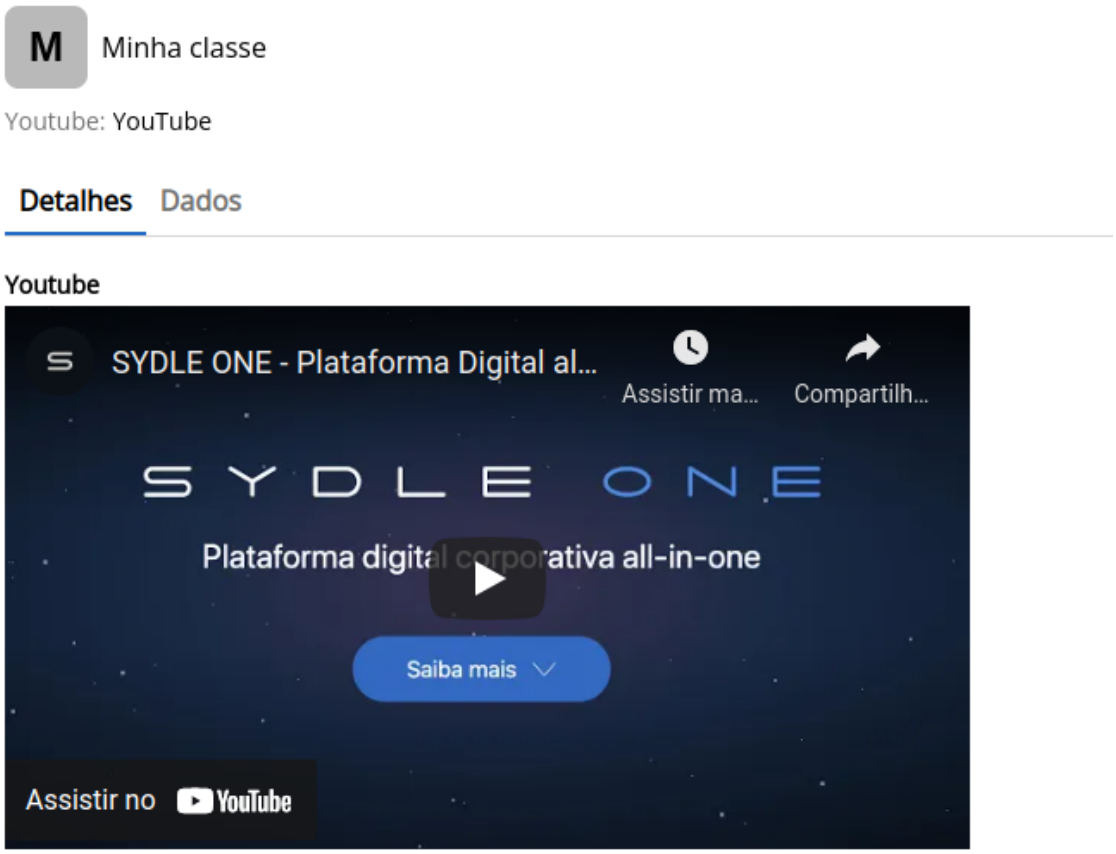


Na seção da documentação "Exemplo de uso", é apresentado o objetivo que se quer atingir e como chegar lá em dois passos essenciais: modelagem no ECM da plataforma e criação de um *web-component* responsável. Um exemplo específico é criado para a renderização de vídeos do YouTube, com um campo que, em modo de exibição, contém uma entrada de texto para receber um *link* de vídeo. Se o *link* for válido, o campo renderiza um iFrame com o reprodutor de vídeo, como mostrado

[9] O *decorator* @Prop() é uma funcionalidade do StencilJS que permite que um componente receba dados externos na forma de propriedades. Isso permite que os componentes sejam reutilizáveis e modulares, uma vez que as propriedades podem ser definidas externamente e passadas para o componente de maneira dinâmica. As propriedades podem ser configuradas para serem mutáveis ou imutáveis e o StencilJS irá gerenciar a atualização do componente quando uma propriedade é alterada.

na Figura 3.8. Em modo de leitura, o campo renderiza apenas o iFrame [10] com o vídeo, como pode ser observado na Figura 3.9.

Figura 3.9 – Captura de tela do formulário com o campo de "Youtube" no modo de leitura.



A atividade foi realizada pelo estagiário e seu colega do time e consistiu em duas etapas: a criação do componente e a elaboração da documentação. A primeira etapa foi executada com facilidade pelo estagiário, já que o componente em questão não era complexo. No entanto, a segunda etapa demandou mais ajuda, uma vez que o estagiário não tinha muita experiência na elaboração de

[10] iFrame é uma abreviação para *"inline frame"* e é uma tecnologia web que permite a incorporação de um documento HTML dentro de outro documento HTML, permitindo assim a exibição de conteúdo externo em uma página web, como vídeos, mapas, etc.

documentações técnicas. Com a ajuda de seu colega, eles conseguiram produzir uma documentação completa e didática, atendendo às necessidades de forma eficiente.

3.3.3 *Card* de objeto

Uma das principais funções de um *card* de objeto é representar visualmente uma instância de um objeto, mostrando as principais informações de identificação e destaque associadas a ele, como mostra a Figura 3.10. A relevância dessas informações é definida na modelagem da classe daquele objeto, ou seja, ao modelar uma classe o usuário escolhe quais informações serão consideradas identidade do objeto, quais terão destaque e quais são informações comuns. Não há um limite para a quantidade de cada uma dessas categorias de informações.

Além das identidades e dos destaques, o *card* do objeto também contém um avatar, que pode ser uma imagem cadastrada ao criar o objeto ou, caso não haja uma imagem, a primeira letra da primeira identidade, com um fundo colorido gerado aleatoriamente.

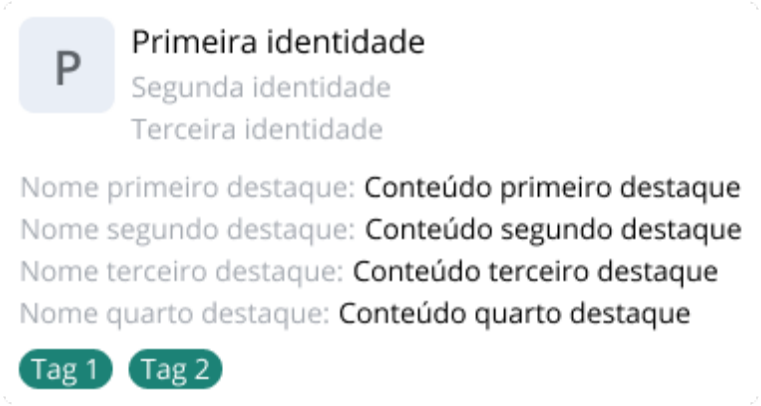
Figura 3.10 – Modelo de um *card* de objeto no tamanho *default*

Fonte: Do autor

As informações de identidade incluem os detalhes mais importantes, como o nome do objeto, por exemplo. As informações de destaque, por sua vez, são as informações que são consideradas mais importantes sobre o objeto mas não são únicas, como a data de criação, o status ou outras informações relevantes. O *card* de objeto também apresenta variações de tamanho, tendo uma variação "*small*" que tem o visual reduzido para ocupar menos espaço de tela (Figura 3.11).

Para criar esse componente, o estagiário enfrentou diversos desafios, dentre eles a necessidade de ler arquivos JSON e manipular as informações recebidas para renderizá-las de forma organizada e responsiva na tela. Para alcançar esse objetivo, ele precisou aprimorar seu conhecimento em HTML para agrupar elementos semelhantes e em CSS para garantir que o componente fosse responsivo e capaz de lidar com diferentes tamanhos de texto e quantidades de informações. Embora o estagiário

Figura 3.11 – Modelo de um *card* de objeto no tamanho *small*



Fonte: Do autor

executado com facilidade as tarefas de leitura de arquivos JSON e de separação das informações, ele enfrentou mais dificuldades na parte de CSS devido à complexidade das regras. No entanto, com esforço e ajuda, ele foi capaz de superar esses obstáculos e entregar um componente de qualidade. Na Figura 3.12 esta demonstrado uma aplicação de um *card* de objeto com informações reais.

Figura 3.12 – Utilização de um card de objeto no tamanho *default*, com simulação de um uso real

Fonte: Do autor

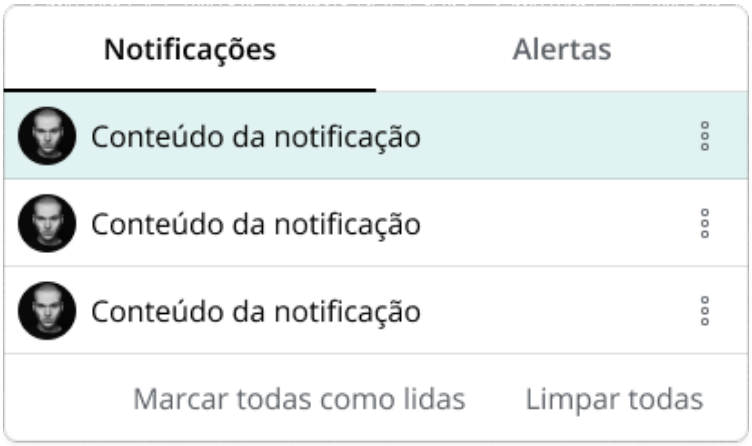
3.3.4 Menu de Mensagens

O menu de mensagens é uma componente que permite ao usuário visualizar notificações e alertas em uma única interface. Este menu é composto por duas abas: uma para notificações e outra para alertas.

O menu de notificações (Figura 3.13) é responsável por exibir informações importantes para o usuário. As notificações são apresentadas na forma de *cards*, que contém informações pertinentes so-

bre um evento ou ação na plataforma. O usuário pode clicar em uma notificação para ser redirecionado para a página referente àquela notificação.

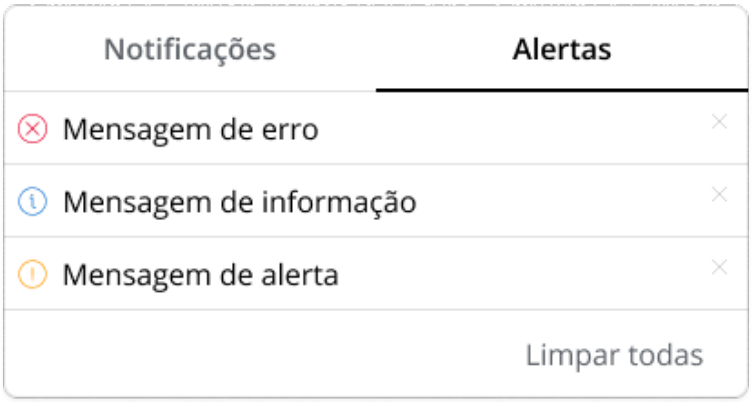
Figura 3.13 – Menu de notificações.



Fonte: Do autor

O menu de alertas (Figura 3.14) é responsável por apresentar alertas emitidos na plataforma. Os alertas são exibidos em formato de *toast* na tela por alguns segundos. No entanto, para permitir ao usuário um acesso mais prolongado a estes alertas, eles também são armazenados no menu de alertas. Isso significa que o usuário pode visualizar todos os alertas emitidos no *workspace* naquele momento.

Figura 3.14 – Menu de alertas.



Fonte: Do autor

Para criar o menu de mensagens, o estagiário precisou criar dois subcomponentes: o menu de notificações e o menu de alertas. Neles precisou implementar o recurso de *Streaming Updates* do Redux [11], o qual funciona como um *listener* [12]que "escuta" a API da plataforma em busca de alertas e notificações, para que esses menus possam exibir as mensagens em tempo real. Quando uma mensagem é emitida, o *listener* captura os dados, e os componentes filtram as informações relevantes e as exibem nos menus apropriados.

3.4 Projeto SYDLE ONE WORKSPACES

Essa seção descreve o projeto SYDLE ONE WORKSPACES, que representa a interface de usuário da plataforma. Ao contrário dos outros projetos ao qual atuou, o SYDLE ONE WORKSPACES é o projeto final e não é consumido por nenhum outro. Durante o desenvolvimento deste projeto, o estagiário desempenhou um papel ativo, contribuindo para diversas funcionalidades, destacando a criação da *app bar*, do menu de criação e da *view* de listagem de classes. Neste capítulo, será apresentada uma descrição detalhada de cada uma dessas funcionalidades, destacando as principais decisões de design e os desafios encontrados durante a implementação.

3.4.1 *App bar*

A *App bar* (Figura 3.15) é a barra superior do *workspace*, lá estão contidos os menus de mensagem, menu de criação e menu de usuário, além de permitir acesso ao menu lateral, o qual lista todos os *workspaces* disponíveis ao usuário e os permite acesso.

Figura 3.15 – App Bar

Fonte: Do autor

[11] Redux é uma biblioteca JavaScript para gerenciamento de estados em aplicações web. <https://redux.js.org/.>

[12] Em programação, um *listener* é um objeto ou função que "escuta" eventos em uma aplicação e responde a eles. Em JavaScript, os *listeners* são frequentemente usados para "escutar"eventos de mouse, teclado e outros eventos do usuário, bem como para se comunicar com outras partes do código que precisam saber quando um evento específico ocorreu.

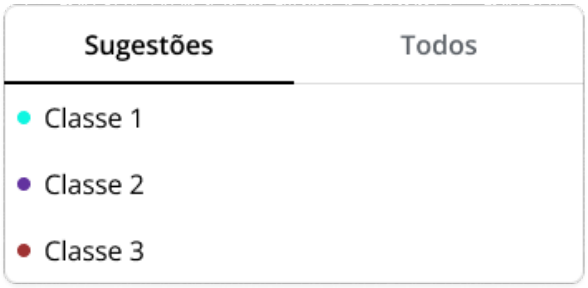
A criação desse componente foi bastante simples, uma vez que seu objetivo principal era envolver outros componentes e exibir a barra lateral com os respectivos menus quando solicitado. Para isso, era necessário implementar a lógica que exibia a barra lateral quando o ícone de menu à esquerda era clicado, bem como a exibição dos menus de criação, mensagens e usuário quando os ícones correspondentes eram selecionados. Além disso, o componente também fazia uma requisição simples para obter o nome do *workspace* atual. No entanto, embora a funcionalidade em si fosse relativamente simples, foi necessário prestar muita atenção para evitar conflitos nos eventos de clique e garantir que apenas um menu fosse aberto por vez.

3.4.2 Menu de criação

O menu de criação é responsável por permitir a criação de novos objetos a partir das classes contidas naquele *workspace*. Composto por duas abas, "Sugeridos" e "Todos", o menu é estruturado de forma a facilitar a escolha da classe adequada para cada contexto.

A aba "Sugestões" (Figura 3.16) apresenta as classes mais relevantes para o *workspace* ao qual o usuário está inserido, com base em lógicas de *back-end*. Por exemplo, se o usuário está trabalhando em um objeto da classe "Issue", ao abrir o menu de criação, a classe "Issue" será sugerida como uma das opções.

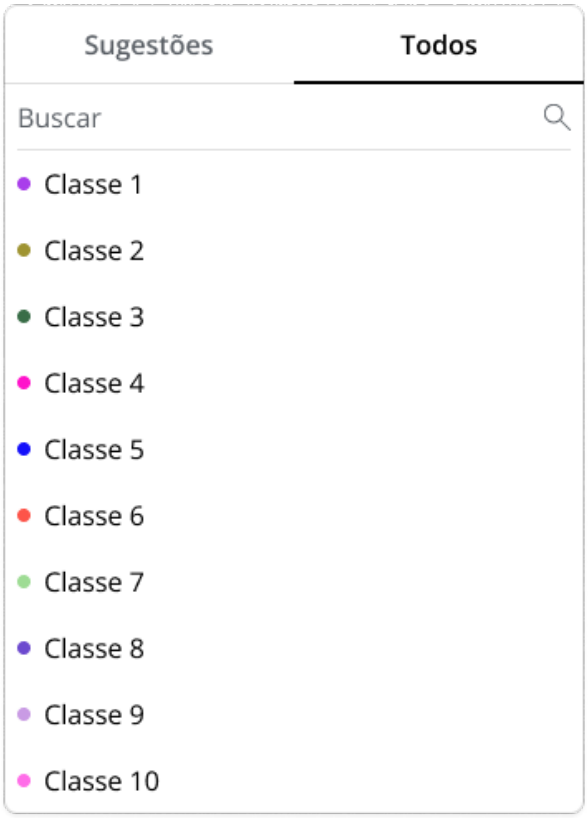
Figura 3.16 – Aba "Sugestões" do menu de criação.



Fonte: Do autor

Já a aba "Todos" (Figura 3.17) exibe todas as classes disponíveis para o usuário criar novos objetos. Como a plataforma pode contar com uma grande quantidade de classes, essa aba possui uma barra de pesquisa para facilitar a busca pelas classes desejadas.

Figura 3.17 – Aba “Todos” do menu de criação.



Fonte: Do autor

O visual do menu é bem simples e direto, estando contido em um menu suspenso. Cada classe é representada por um item de lista com um *badge* colorido e seu nome.

A primeira parte desse componente consiste em fazer uma requisição à API para obter as classes a serem exibidas no menu. Essa requisição é feita toda vez que o menu é aberto, a fim de economizar espaço em cache, já que as informações obtidas podem mudar de acordo com o estado atual do *workspace*. Depois de filtrar as classes, elas precisam ser exibidas em itens selecionáveis dentro de abas, que só são renderizadas se houverem classes disponíveis. Se não houver, essa aba não aparece. Na listagem de classes, é necessário incluir um pequeno círculo colorido, cuja cor é calculada pelo mesmo mecanismo que determina a cor dos *cards* de objeto. Quando uma das classes é selecionada, uma requisição é enviada para abrir o formulário de criação daquela classe para que o usuário possa preenchê-lo.

3.4.3 *View* de listagem de classes

A *view* de listagem de classes (Figura 3.18) é uma área do *workspace* dedicada a explorar todas as classes ali contidas. Ela é disposta por uma visualização em árvore, onde os nós pais representam os pacotes aos quais cada classe está inserida e, ao clicar no pacote, é exibido como nós folhas as classes em si, acompanhadas de um *badge* colorido. Ao clicar em algum dos nós folhas, é emitido um evento na API requerendo que os objetos daquela classe sejam listados na *view* de listagem de objetos. Além da representação dos pacotes e das classes, a *view* conta também com uma barra de pesquisa, a qual faz uma pesquisa tanto em pacotes quanto em classes, e um botão para colapsar todos os pacotes, reduzindo o espaço ocupado pelos itens na *view*.

Figura 3.18 – Listagem de classes com alguns pacotes "abertos"mostrando suas respectivas classes.

Fonte: Do autor

Essa *view* tinha como objetivo exibir as classes relacionadas a um objeto, para isso, era necessário fazer uma requisição que retornava um arquivo JSON contendo todos os pacotes e suas respectivas classes. Os pacotes eram representados como itens de lista, ao serem clicados, exibiam as classes contidas neles. Caso um pacote não possuísse nenhuma classe, ele não era exibido na lista. A barra de pesquisa era capaz de buscar tanto por pacotes quanto por classes e realizava uma busca exata, retornando apenas os resultados que começavam exatamente iguais ao termo de pesquisa inserido. Quando o resultado da pesquisa correspondia a um pacote, ele era exibido aberto, mostrando todas as suas classes, independentemente de os nomes das classes corresponderem à pesquisa ou não. Se a pesquisa correspondia a uma classe, o pacote correspondente era exibido aberto, mostrando apenas as classes compatíveis com a pesquisa.

3.5 Considerações Finais

Durante o período de atuação na empresa SYDLE, o estagiário desempenhou diversas atividades que foram abordadas nas seções anteriores, desde o treinamento inicial até a concretização da nova interface *front-end* da plataforma SYDLE ONE. Ao longo desse processo, o estagiário obteve um vasto conhecimento técnico e aprimorou suas habilidades colaborativas junto à equipe. Apesar do aprendizado significativo, também enfrentou desafios que despertaram seu interesse pelo *Product Design* e resultaram em uma transição de carreira para essa área.

Embora atualmente não esteja mais atuando como desenvolvedor, o conhecimento adquirido durante esse período continua sendo extremamente valioso em sua função como designer e teve um papel fundamental em sua trajetória profissional. As experiências vivenciadas como desenvolvedor forneceram uma base sólida para compreender as necessidades técnicas e colaborar de forma efetiva com a equipe de desenvolvimento. Essa combinação de conhecimentos em desenvolvimento e design permite que o estagiário aborde os projetos com uma perspectiva ampla e integrada, buscando soluções que sejam viáveis tecnicamente e também atendam às expectativas de usabilidade e experiência do usuário.

A jornada pelo desenvolvimento de software proporcionou uma compreensão aprofundada das práticas ágeis, das metodologias de desenvolvimento e dos desafios enfrentados no processo de

criação de produtos digitais. Essa experiência contribuiu para a formação de uma mentalidade colaborativa, adaptável e orientada para resultados, que se reflete na abordagem do estagiário como designer. O conhecimento técnico adquirido também permite uma melhor comunicação e colaboração com a equipe de desenvolvimento, facilitando a criação de soluções eficientes e eficazes.

4 CONCLUSÃO

Esse trabalho teve como objetivo apresentar um relatório das experiências como engenheiro de software que o aluno teve em seu tempo como estagiário na empresa SYDLE. O principal objetivo durante o estágio foi atuar na renovação e modernização do principal produto da empresa, o SYDLE ONE. Para atingir esse objetivo o estagiário atuou em projetos incrementais aos quais formaram a base para a obtenção do objetivo principal, sendo eles o projeto SYDLE UI e o projeto COMPONENTS, além de atuar no projeto WORKSPACES, o qual já representava a nova interface da plataforma.

Ao final do período de estágio na empresa SYDLE, o estagiário obteve uma experiência enriquecedora em sua formação acadêmica e profissional. A participação em projetos de renovação do *front-end* do principal software da empresa permitiu que ele desenvolvesse habilidades em desenvolvimento web e *web-components*, além de despertar seu interesse em design de interface e experiência de usuário, o que o levou a migrar para a área de *Product Design*.

As habilidades e conhecimentos aprendidos durante o curso de Ciência da Computação foram essenciais para o desempenho do estagiário no desenvolvimento dos projetos, em especial os conceitos de Programação Orientada a Objetos e Linguagens Formais e Autômatos. Apesar da falta do aprofundamento em tecnologias atuais utilizadas na industria, a base lógica de programação adquirida durante o curso foi fundamental para que o estagiário pudesse aprender novas tecnologias com facilidade.

A empresa SYDLE ofereceu um ambiente acolhedor e propício para o desenvolvimento do estagiário, permitindo que ele explorasse diferentes áreas e se sentisse motivado a desbravar novas carreiras na empresa. Além disso, a experiência proporcionou uma imersão no ambiente empresarial, possibilitando o desenvolvimento de habilidades colaborativas, organizacionais e de cooperação.

Em resumo, o período de estágio na SYDLE foi uma experiência altamente positiva para o estagiário, contribuindo significativamente para sua formação acadêmica e profissional e despertando seu interesse em novas áreas de atuação. O conhecimento adquirido durante o estágio será fundamental para o seu futuro desempenho como profissional de Ciência da Computação.

REFERÊNCIAS

Agile Alliance. **Manifesto for Agile Software Development**. 2001. <http://agilemanifesto.org/>.

ANDERSON, D. J. **Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results**. [S.l.]: Prentice Hall, 2010. ISBN 978-0-13-137842-8.

BECK, K. **Extreme Programming Explained: Embrace Change**. Boston, MA: Addison-Wesley Professional, 2000.

BRASILEIRO, P. do G. **Design System do Governo Federal**. 2021. <https://www.gov.br/governodigital/pt-br/design-system>.

FLANAGAN, D. **JavaScript: The Definitive Guide**. [S.l.]: O'Reilly Media, 2011.

IONIC. **Stencil**. 2021. Disponível em: <https://stenciljs.com/>.

MARKHAM, M. **Building a Design System: A Practical Guide to Creating Design Languages for Digital Products**. [S.l.]: O'Reilly Media, 2019. ISBN 9781492052540.

MATERIAL-UI. **Material-UI: A popular React UI framework**. 2021. <https://material-ui.com/>.

PSR, V. **Web Components**. 221. <https://medium.com/@valdeirpsr/web-components-9cdcad1f87b9>.

SCHWABER, K. **Agile Project Management with Scrum**. 1st. ed. Redmond, WA: Microsoft Press, 2002.

STENCILJS. **Getting Started with Stencil**. 2021. Disponível em: <https://stenciljs.com/docs/getting-started>.

STENCILJS. **Styling Components**. 2021. Disponível em: <https://stenciljs.com/docs/styling>.

SUTHERLAND, J.; SCHWABER, K. **Scrum Guide**. 2020. Disponível em: <https://scrumguides.org/>.

W3C. **Web Components**. 2021. Disponível em: <https://www.w3.org/standards/techs/components>.