
**ADRIANO OLIVEIRA SILVA**

# IMPLEMENTANDO UM *MICROFRONTEND* COMO TELA DE PAGAMENTOS

**LAVRAS – MG**

**2023**

**ADRIANO OLIVEIRA SILVA**

# IMPLEMENTANDO UM *MICROFRONTEND* COMO TELA DE PAGAMENTOS

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

Prof. Dr. Antônio Maria Pereira de Resende

Orientador

**LAVRAS – MG**

**2023**

**Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos**

**da Biblioteca Universitária da UFLA**

Silva, Adriano Oliveira

Implementando um *microfrontend* como tela de pagamentos / . $2^a$ ed. rev., atual. e ampl. – Lavras : UFLA, 2023.

32 p. : il.

Tcc(graduação)–Universidade Federal de Lavras, 2016.

Orientador: Prof. Dr. Antônio Maria Pereira de Resende.

Bibliografia.

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

CDD-808.066

**ADRIANO OLIVEIRA SILVA**

# IMPLEMENTANDO UM *MICROFRONTEND* COMO TELA DE PAGAMENTOS

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

Aprovado em 07 de Dezembro de 2023.

| | |
|---|---|
| Prof. Dr. Antônio Maria Pereira de Resende | UFLA |
| Profa. Dra. Renata Teles Moreira | UFLA |
| Prof. Dr. Paulo Afonso Parreira Júnior | UFLA |

Prof. Dr. Antônio Maria Pereira de Resende
Orientador

**LAVRAS – MG**
**2023**

*Dedico este trabalho à minha família e amigos.*

**AGRADECIMENTOS**

Agradeço a minha família por ter proporcionado os meios para que fosse possível seguir nesta jornada acadêmica. Agradeço a UFLA, por ter me proporcionado diversas experiências que me transformaram. Agradeço as amizades que construí durante minha vida acadêmica.

*Na dúvida sempre siga seu nariz.*
*(Gandalf, o Cinzento)*

# RESUMO

Atualmente, na indústria de *softwares*, as empresas, especialmente as *startups*, por possuírem menos recursos, procuram evoluir seus produtos de maneira cada vez mais ágil, disponibilizando funcionalidades rapidamente ao mercado. Para obter essa rapidez, muitas vezes isso significa mudar alguma metodologia, processo ou até mesmo a arquitetura do *software* do sistema em questão. Considerando o exposto, foi decidido evoluir a interface do produto de *software* da empresa E-inscrição, que possui uma arquitetura monolítica, para algo mais descentralizado, utilizando a arquitetura de *microfrontends*. Isso torna possível que equipes independentes, ou até mesmo um único desenvolvedor, adicionem novas funcionalidades ao sistema com um esforço reduzido na parte integradora. Esses *microfrontends* devem se comunicar com outras partes do sistema através de APIs REST Rails e JavaScript. Um *microfrontend* foi desenvolvido pelo estagiário de *front-end*, que utilizou a biblioteca React em TypeScript, além de outras tecnologias e serviços, como NPM, CDNs e componentes disponibilizados por um *design system* em construção pela equipe de desenvolvimento.

**Palavras-chave:** Arquitetura de *microfrontend*. Frontend. Javascript. APIs. Typescript. Design System. NPM.

# ABSTRACT

Currently, in the software industry, companies, especially startups, seek to evolve their products in an increasingly agile manner, rapidly releasing features to the market. To achieve this speed, it often means changing methodologies, processes, or even the software architecture of the system in question. With that in mind, it was decided to evolve the interface of the software product for the company E-inscrição, which currently has a monolithic architecture, to a more decentralized approach using *microfrontend*s. This allows independent teams, or even a single developer, to add new features to the system with reduced effort on the integration side. These *microfrontend*s communicate with other parts of the system through Rails and JavaScript REST APIs. A *microfrontend* was developed by the Front-end intern, who utilized the React library in TypeScript, along with other technologies and services such as NPM, CDNs, and components provided by an ongoing Design System being developed by the development team.



**Keywords:** *microfrontend* architecture. Frontend. Javascript. APIs. Typescript. Design System. NPM.

## SUMÁRIO

# 1 INTRODUÇÃO

Diante do cenário de alta competitividade no mercado de *software*, as empresas têm procurado agilizar cada vez mais a evolução de seus produtos para entregar funcionalidades rapidamente. Nessa corrida, no ramo de eventos, a *startup* E-inscrição compete no mercado trazendo um sistema para que organizadores possam gerenciar seus eventos. Seu produto possui uma gama de funcionalidades, desde venda de ingressos até a geração de relatórios de participantes e relatórios financeiros.

Procurando evoluir seu produto e gerar mais valor para seus clientes, a empresa E-inscrição tomou a decisão de recrutar estagiários para especializá-los em *JavaScript*, com o intuito de desenvolver *microfrontends* e microsserviços que possibilitam entrega rápida e desenvolvimento de forma independente, removendo responsabilidades da parte integradora, construída em *Ruby on Rails*.

Este relatório de estágio apresenta o processo de implementação de uma nova funcionalidade utilizando a arquitetura de *microfrontends*, uma interface de alteração de forma de pagamento, na qual o estagiário esteve responsável pela codificação, junto com uma equipe contendo um gestor de projetos, um *designer*, um desenvolvedor *Ruby on Rails*, um outro estagiário *back-end* e um líder técnico. O estágio teve duração de 1 ano, de fevereiro de 2021 até fevereiro de 2022.

## 1.1 Sobre a empresa

No ramo de eventos, a empresa E-inscrição[1] fornece uma ferramenta para organizadores realizarem a gestão de seus eventos. Possui mais de 10 anos de atuação no mercado e já registrou mais de 28 mil organizadores que usam ou já utilizaram o sistema. Na plataforma, os clientes conseguem ter acesso a uma gama de funcionalidades, incluindo a criação de um site personalizado para o evento, credenciamento agilizado, fornecimento de várias formas de pagamento, entre outras.

### 1.1.1 Estrutura organizacional e Gestão

A E-inscricao é uma empresa de médio porte[2] e possui uma estrutura organizacional dividida em vários setores, sendo eles: Vendas, Pré-Vendas, People, Marketing, *Customer Ser-*

[1] Disponível em: https://home.e-inscricao.com/
[2] Disponível em: https://www.linkedin.com/company/e-inscricao/mycompany/

*vice*, Suporte, *Tech* e Produto. Possui uma cultura bastante horizontal, onde há abertura para que todos os colaboradores se expressem e tomem iniciativas diante das tomadas de decisões e desafios. A empresa possui um *Chief Executive Officer* e um *Chief Financial Officer*, e cada área possui um *head* como líder.

Seu modelo de gestão e alcance de metas é baseado em OKRs[3] - *Objectives and Key Results*, nos quais são estabelecidos objetivos a serem alcançados trimestralmente, cujo cumprimento leva ao avanço em um ou mais objetivos estratégicos maiores que se deseja atingir ao final de cada ano.

Os setores mais relevantes para este relatório são Suporte, Produto e Tech, sendo este último o setor no qual o estagiário atuou. Esses setores possuem bastante interação e interdependência, pois trabalham diretamente e indiretamente na manutenção do sistema, desde sua evolução até a resolução de *bugs*.

## 1.2 Organização do trabalho

Este relatório está organizado no seguinte formato:

O capítulo 1 descreve uma introdução do trabalho e enfatiza informações sobre a empresa, além das atividades exercidas pelo estagiário. O capítulo 2 apresenta os conceitos teóricos das tecnologias e serviços utilizados pelo estagiário durante a execução das atividades. O capítulo 3 detalha o processo de desenvolvimento do projeto no qual o estagiário atuou. O capítulo 4 contém a conclusão do trabalho, apresenta os aprendizados do estagiário e seu ponto de vista em relação às atividades desenvolvidas.

[3] Disponível em: https://endeavor.org.br/estrategia-e-gestao/como-transformar-objetivo-okr-meta/

# 2 TECNOLOGIAS, CONCEITOS E SERVIÇOS

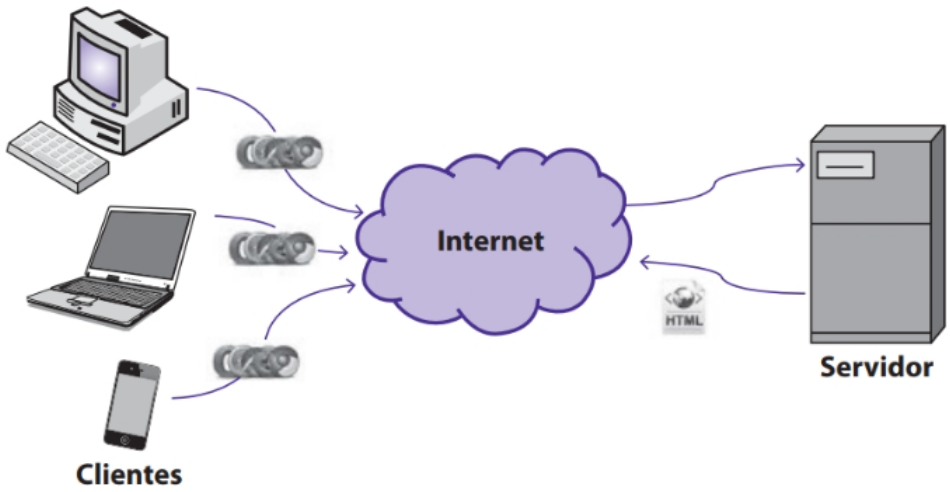
Neste capítulo, apresenta-se sucintamente as tecnologias, conceitos e serviços com os quais o estagiário esteve envolvido ao longo do período de atuação.

## 2.1 Desenvolvimento de Sistemas Web

O Desenvolvimento de Sistemas Web é uma disciplina essencial para a criação de sistemas e aplicações que possam ser acessados através da internet. A arquitetura de um sistema web é distribuída e baseada na organização cliente-servidor, em que através de um navegador em um computador (cliente), um usuário faz requisições para um determinado serviço, que por sua vez está contido em outro computador remoto que realiza todo o processamento e entrega a resposta para este usuário (servidor) (MILETTO; BERTAGNOLLI, 2014).

A Figura 2.1 ilustra um exemplo das camadas que constituem a estutura cliente-servidor.

Figura 2.1 – Estrutura cliente-servidor




Fonte: (MILETTO; BERTAGNOLLI, 2014)

A aplicação do cliente pode conter um *microfrontend* responsável por uma parte específica da interface do usuário que se comunica com o servidor. É comum no servidor utilizar uma API REST, que é uma forma de comunicação que apresenta um conjunto de padrões para funcionar de maneira eficiente e interoperável (SAUDATE, 2021).

## 2.2 *Microfrontends*

Os *microfrontends* são uma abordagem arquitetônica que permite a construção de interfaces de usuário modulares e independentes, facilitando a escalabilidade e a manutenção do código em projetos complexos (JACKSON, 2019).

Os *microfrontends* são uma extensão do conceito de arquitetura de microsserviços aplicados à camada de interface do usuário (GEERS, 2017). Em vez de construir uma única aplicação monolítica, os *microfrontends* dividem a interface do usuário em partes menores e independentes, cada uma delas podendo ser responsável por uma ou mais funcionalidades específicas. Essas partes menores, chamadas de *microfrontends*, podem ser desenvolvidas e implantadas de forma independente, permitindo que equipes diferentes trabalhem em paralelo e com tecnologias diferentes, se necessário. A arquitetura de microfrontends oferece diversos benefícios (GEERS, 2017), tais como:

- Escalabilidade: a divisão da interface do usuário em projetos independentes possibilita que diferentes equipes trabalhem em paralelo, facilitando o processo de evolução da aplicação.
- Manutenibilidade: cada *microfrontend* pode ser desenvolvido e mantido separadamente, o que simplifica a manutenção do código em projetos grandes e complexos.
- Reutilização de código: os *microfrontends* podem ser reutilizados em diferentes contextos, permitindo a criação de componentes padronizados e compartilháveis.
- Flexibilidade tecnológica: cada *microfrontend* pode ser desenvolvido com diferentes tecnologias, desde que sejam compatíveis com a integração no sistema em que será implementado.

## 2.3 Javascript

JavaScript é uma linguagem de programação multi-paradigma, mais conhecida como uma linguagem de *script* para a web, porém também usada em outros ambientes que não são navegadores, como o Node.js. Foi inicialmente projetada em 1995, quando um programador de um famoso navegador da época, o Netscape, trabalhava em validações de campos de formulários no lado do cliente, pois esse processo era feito no servidor em uma época em que a velocidade da internet era muito lenta (FRISBIE, 2019).

JavaScript respeita a conformidade ECMAScript (FRISBIE, 2019), que são padrões nos quais as linguagens de *script* devem se basear. De tempos em tempos, novos padrões ECMAScript são lançados para que a linguagem evolua e possua mais funcionalidades. O JavaScript nos navegadores é amplamente utilizado para fazer a manipulação do DOM (*Document Object Model*), uma interface para o HTML que mapeia toda a página como uma árvore hierárquica contendo diferentes tipos de dados.

### 2.3.1 Typescript

Criado pela Microsoft, o TypeScript é uma linguagem de programação que, dentre outras funcionalidades, adiciona tipos de dados ao JavaScript. Portanto, é uma linguagem fortemente tipada que adiciona mais segurança no desenvolvimento ao fazer a checagem de tipos. O TypeScript fornece todas as funcionalidades do JavaScript e adiciona o sistema de tipagem (MICROSOFT, 2023).

Na Figura 2.2 podemos observar um exemplo de um simples código em Typescript, em que a interface *User* é utilizada para definir quais os tipos de parâmetros que serão recebidos pelas funções *deleteUser* e *getAdminUser*.

Figura 2.2 – Exemplo de tipagem com Typescript.

```
interface User {
  name: string;
  id: number;
}

function deleteUser(user: User) {
  // ...
}

function getAdminUser(): User {
  //...
}
```

Fonte: (MICROSOFT, 2023)

## 2.4 HTML

O HTML (HyperText Markup Language) é uma linguagem de marcação que compõe as páginas da *web* (MDN, 2023a). Ela organiza as páginas por meio de elementos, descritos

em *tags*, como <head>, <body>, , entre outras, para exibir dados de forma estruturada. Cada elemento possui atributos que permitem adicionar característas nas *tags*. Um exemplo de elemento com atributo é visto na Figura 2.3, onde o atributo src é utilizado para importar um arquivo externo.

Figura 2.3 – Tag script com atributo src.

```
<script src="jquery-3.6.4.min.js" ></script>
```

Fonte:Autor, 2023

O HTML é fundamental para a *web* e, em conjunto com o CSS (*Cascading Style Sheets*) e o JavaScript, torna possível a criação de interfaces bonitas e interativas.

## 2.5 CSS

CSS (Cascading Style Sheets) é uma linguagem de estilo utilizada principalmente para adicionar estilos ao HTML. Através de seu uso, é possível alterar como os elementos do HTML se apresentam na tela, podendo modificar suas cores, disposição, fontes de textos, entre outros aspectos das páginas da *web*. O CSS é padronizado pelas especificações do W3C, e atualmente sua versão mais recente é a CSS3, que aos poucos está sendo adotada pelos navegadores *web* (MDN, 2022).

Em conjunto, as linguagens JavaScript, CSS e HTML fornecem a possibilidade de criar páginas da *web* dinâmicas e atraentes, melhorando a usabilidade na interação dos usuários nos sistemas da *web*.

A Figura 2.4 ilustra um exemplo de código HTML com CSS e Javascript. O que está dentro da *tag* <style> modifica a fonte do que está dentro da *tag* <body>, o estilo do botão e do *input*. O que está dentro da *tag* <script> adiciona o comportamento de adicionar o nome inserido no *input* na lista, ao clicar no botão.

A Figura 2.5 demonstra o resultado na página *web* do código da Figura 2.4.

## 2.6 SPA

Uma SPA (*Single-page application*) (MDN, 2023b) é uma implementação *web* em que usuários carregam a página apenas uma vez, e seu conteúdo é modificado através de atualizações do estado desta página. Este tipo de aplicação fornece uma experiência mais dinâmica ao usuário, além de significantes melhorias em desempenho pois não necessita a todo momento

Figura 2.4 – Código HTML com CSS e JavaScript.

example_html.html > html > body > script

```html
<!-- Autor: Adriano Oliveira Silva -->
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <title>Exemplo</title>
    <style>
      /* Adiciona estilos ao html. */
      body {
        font-family: Arial, Helvetica, sans-serif;
      }
      button {
        background-color: blueviolet; /* Green */
        border: none;
        color: white;
        padding: 6px 12px;
      }
      input {
        border: 1px solid #ccc;
        padding: 6px 12px;
      }
    </style>
  </head>
  <body>
    <h1>Exemplo de código HTML</h1>
    <div>
      <input id="input" type="text" />
      <button onclick="adicionaNomeNaLista()">Listar</button>
    </div>
    <div>
      <ul id="lista">
        <li>Adriano</li>
      </ul>
    </div>
    <script>
      /* Adiciona um comportamento de adicionar o nome inserido no input na lista do html ao clicar no botão.*/
      const input = document.getElementById('input');
      const lista = document.getElementById('lista');

      function adicionaNomeNaLista() {
        const nome = input.value;
        const li = document.createElement('li');
        li.innerHTML = nome;
        lista.appendChild(li);
      }
    </script>
  </body>
</html>
```

Fonte: Autor, 2023

Figura 2.5 – Página *web* referente ao código da Figura 2.5


Fonte: Autor, 2023

carregar documentos *web* de um servidor. O React é uma biblioteca muito utilizada na construção de SPAs.

### 2.6.1 React

React é uma biblioteca JavaScript para criação de interfaces (META, 2023b). Foi construída para facilitar o desenvolvimento de interfaces interativas, baseando-se na filosofia de componentes encapsulados, que possuem e controlam seu próprio estado. React também pode ser renderizado no servidor e utilizado para a criação de aplicações móveis híbridas, por meio do React Native.

Na Figura 2.6 observa-se como é a anatomia de um componente React. O que se assemelha a uma *tag* html <button>, é na realidade JSX[4], uma sintaxe JavaScript parecida com HTML.

Figura 2.6 – Exemplo de um componente React.

```
function MyButton() {
  return (
    <button>I'm a button</button>
  );
}
```

Fonte:(META, 2023b)

Escrever componentes React é muito semelhante a escrever funções JavaScript. Por esses e outros motivos, é uma das tecnologias de desenvolvimento mais populares atualmente. React foi criada pela empresa Meta, anteriormente conhecida como Facebook (META, 2023a).

## 2.7 Ruby on Rails

Ruby on Rails, também conhecido apenas como Rails, é um *framework* utilizado para criar aplicações *web*, baseado no padrão MVC (*Model-View-Controller*). O Rails é construído em Ruby, uma linguagem de programação focada em simplicidade e produtividade. O Rails possui diversas bibliotecas e outros frameworks incorporados. É conhecido por ser um *framework full stack*, pois fornece ferramentas que abrangem desde o trabalho com bancos de dados até a geração de HTML (BENSON, 2008).

## 2.8 NPM

O NPM (*Node Package Manager*) é um gerenciador de pacotes para o Node.js, utilizado pelos desenvolvedores para compartilhar códigos JavaScript. É composto por um repositório

[4] https://facebook.github.io/jsx/

(*npm registry*), onde ficam os códigos compartilhados, e também por uma interface de linha de comando (npm CLI), que os desenvolvedores podem usar para instalar, desinstalar ou publicar pacotes Node.js.

O NPM é uma ferramenta importante para a comunidade JavaScript e de código aberto, pois oferece uma maneira fácil e rápida de compartilhar códigos e criar aplicações Node.js. Isso torna o processo de desenvolvimento mais ágil, uma vez que existem muitas ferramentas prontas e problemas resolvidos por outros desenvolvedores que podem ser úteis em um determinado projeto (NPM, 2023).

## 2.9 CDNs

CDN (*Content Delivery Network*) é um conjunto de servidores distribuídos pelo mundo, utilizados para fornecimento de arquivos pela *internet* de forma rápida e confiável (CLOUDFLARE, 2023). A maior parte do tráfego na *internet* é composta por esses serviços, que são fundamentais para o bom funcionamento de diversos *sites* na *internet*. Um dos grandes serviços de CDN atualmente é o Cloudflare[5].

## 2.10 *Design Systems*

Um *Design System* é um conjunto de documentações utilizadas para guiar o desenvolvimento de aplicações, tanto no aspecto de *design* como na codificação de interfaces. Essa documentação contém exemplos de código e uso de componentes que irão compor a aplicação, e seu objetivo principal é manter padronização, objetivos e visão de um determinado produto.

Através de um *Design System*, designers e desenvolvedores têm uma direção de como criar protótipos e construir *front-ends*, respectivamente, permitindo que os membros da equipe de um produto possam trabalhar com mais rapidez e confiança, diminuindo a necessidade de recriar soluções para problemas já resolvidos (VESSELOV; DAVIS, 2019).

## 2.11 Controle de versões com Git e Github

Para controlar versões de um *software*, existem Sistemas de Controle de Versões como o Git. Um Sistema de Controle de Versões é uma ferramenta para gerenciar a adição, remoção ou edição de uma base de código, rastreando seus arquivos e as modificações realizadas.

[5] Disponível em: https://www.cloudflare.com/pt-br/cdn/

Além de ferramentas como o Git, serviços como o GitHub possibilitam melhorar ainda mais o trabalho colaborativo, além de fornecer outras funcionalidades que ajudam na documentação, gerenciamento e lançamento de produtos de *software* (TSITOARA, 2019).

### 2.11.1 Git

Git é um Sistema de Controle de Versões distribuído, gratuito e de código aberto (CHACON, 2023). O Git utiliza um sistema de ramificações que fornece aos desenvolvedores a possibilidade de trabalhar em contextos diferentes sem nenhum atrito, assim podendo controlar qual o objetivo de cada ramificação, separando o que está pronto para produção, o que está sendo testado, entre outros cenários.

O Git foi construído na linguagem C e é extremamente rápido. Uma das formas de se trabalhar com o Git é ter um repositório central que é clonado para a máquina do desenvolvedor, e assim as operações são realizadas diretamente nesta máquina local, o que fornece uma vantagem em relação a sistemas centralizados que precisam comunicar com o servidor várias vezes. Sendo distribuído, significa que cada desenvolvedor possui um *backup* do repositório remoto, o que essencialmente elimina as chances de perda total da base de código, se alguém já clonou o repositório.

Na Figura 2.7 observa-se uma ilustração de ramificações que representa um fluxo de desenvolvimento. A *branch master* é a *branch* de produção, correspondente a versão do produto que chega até os clientes. A *branch development* pode ser utilizada para que desenvolvedores testem novas funcionalidades. E a *branch topic* pode ser uma branch para testar uma funcionalidade em específico.

### 2.11.2 Github

O GitHub é um serviço que fornece diversas ferramentas para desenvolvedores. Não é apenas um repositório de códigos *online*, é também uma plataforma onde é possível revisar código, gerenciar projetos e compartilhar com a comunidade de desenvolvedores do mundo. O GitHub tem um amplo portfólio de ferramentas. É possível criar soluções para CI/CD (*Continuous Integration/Continuous Deployment*), verificar se seu código é seguro, automatizar fluxos de trabalho, publicar pacotes, ter um ambiente de desenvolvimento na nuvem, planejar projetos e interagir com a comunidade (GITHUB, 2023).

Figura 2.7 – Exemplo de ramificações do git.


Fonte:(CHACON, 2023)

## 2.12 Basecamp

O Basecamp é um serviço de gestão de projetos que possui uma filosofia *all-in-one* (BASECAMP, 2023). Muitas ferramentas de gestão de projetos não possuem funcionalidades para armazenar documentos, planejar calendários, realizar *chat*, entre outras. Para suprir essa lacuna, elas dependem de *links* e integrações com outros serviços, o que descentraliza a gestão. O Basecamp evita essa estratégia, reunindo todas as funcionalidades mencionadas em um único lugar, proporcionando um ambiente mais focado e menos complexo.

O Basecamp é uma ferramenta paga, porém fornece um período de 30 dias gratuito para experimentação [5].

## 2.13 Figma

A ferramenta Figma é utilizada para o desenvolvimento colaborativo de interfaces (FIGMA, 2023). Por meio do Figma, é possível criar protótipos e compartilhá-los em tempo real, proporcionando um ambiente de trabalho que aproxima *designers* e desenvolvedores. Além disso, o Figma permite a criação de recursos reutilizáveis e facilita o planejamento de projetos. A plataforma também oferece ferramentas específicas para a criação de sistemas de design. O Figma está disponível gratuitamente, mas também oferece planos pagos[6].

[5] Disponível em: https://basecamp.com/pricing
[6] Disponível em: https://www.figma.com/pricing/

## 2.14 Scrum

O Scrum é uma metodologia ágil utilizada para gestão e acompanhamento de projetos, de forma colaborativa (SCRUM.ORG, 2023). Se baseia no princípio de geração de valor com pequenas entregas, em que a cada ciclo ocorrem contínuas melhorias, experimentação e *feedbacks*. A metodologia possui vasta documentação, e para ter total domínio desta, existem até certificações que são emitidas por empresas de autoridade que implementam seu treinamento.

Entretanto, pessoas e organizações podem implementar o Scrum seguindo o seu guia. Também é muito comum que as organizações utilizem de alguns princípios da metodologia para adaptar o seu fluxo de trabalho de acordo com suas necessidades.

# 3 DESENVOLVIMENTO DO PROJETO

Neste capítulo, são explicitadas as atividades desenvolvidas pelo estagiário no desenvolvimento do *microfrontend* de alteração de forma de pagamento da empresa E-inscrição.

O capítulo está organizado da seguinte forma: A Seção 3.1 fala sobre os problemas e objetivos que motivaram o projeto. A Seção 3.2 detalha aspectos do projeto, do funcionamento do *microfrontend* e sua incorporação. A Seção 3.3 descreve os processos da equipe na gestão do projeto, desde a concepção da ideia até a liberação para o cliente e manutenções futuras. E por fim, a Seção 3.4 trata as considerações finais.

## 3.1 Gestão e acompanhamento do projeto

A empresa adota uma versão adaptada do SCRUM [7], no qual a equipe realiza uma série de reuniões ao longo da semana. Cada semana representando uma *sprint*. As reuniões não ocorrem diariamente, mas sim nas segundas, quartas e sextas-feiras. Nas segundas-feiras, é realizado o planejamento da semana, enquanto nas quartas e sextas ocorrem reuniões para atualização e acompanhamento do progresso.

Durante o planejamento das segundas-feiras, são elencadas tarefas que devem ser concluídas até o final da sprint. Nas reuniões de acompanhamento, os desenvolvedores compartilham o progresso realizado durante os dias com o gestor de projetos. Além disso, outras reuniões para esclarecimento de dúvidas, alinhamento e resolução de obstáculos são realizadas de forma pontual e assíncrona, entre os membros da equipe. Esse é o processo de desenvolvimento adotado no projeto.

## 3.2 Os problemas e objetivos

Um dos problemas que motiva a realização deste projeto está relacionado as modificações no monolito, que são onerosas. Para implementar qualquer atualização, é necessário realizar um *deploy* de toda a aplicação. Como a empresa tem objetivos de longo prazo substituir a interface com um novo *Design System*, continuar com a estratégia atual é inviável. A mão de obra Rails também é mais cara e escassa, o que agrava o exposto acima.

[7] Scrum. https://www.scrum.org/

Outro ponto a ser considerado é que o sistema já possui a implementação do pagamento via Pix, no entanto, essa funcionalidade não está presente na tela de alteração de pagamentos. A Figura 3.1 demonstra a tela desenvolvida em Rails, que precisa ser substituída.

A arquitetura foi proposta pelo líder técnico da equipe, em colaboração com o gestor de projetos, que traçam uma estratégia para identificar quais telas devem ser substituídas por *microfrontends*.

O estagiário não participa na escolha da arquitetura. No entanto, tem autonomia para tomar decisões técnicas relacionadas à troca de informações entre as APIs e bibliotecas utilizadas no projeto, visando a resolução de problemas, sempre com a aprovação do líder técnico.

Considerando os problemas mencionados, elencam-se dois objetivos principais: (1) substituir uma tela antiga, codificada em Ruby on Rails, por um *microfrontend* JavaScript com componentes do *Design System* da empresa; e (2) adicionar a opção de pagamento via Pix.

## 3.3 O projeto

O projeto é documentado no Basecamp, onde todas as informações relacionadas a ele são centralizadas. A Figura 3.2 mostra uma captura de tela do cabeçalho do projeto no Basecamp, onde são apresentadas diversas informações relevantes, como os responsáveis pelo projeto, notas de observação, datas importantes, entre outras.

A solução é projetada e documentada pela equipe de produto na ferramenta Figma. O link para o projeto no Figma é disponibilizado no Basecamp, conforme exibido na Figura 3.2. Na Figura 3.3, observa-se uma captura de tela dos *designs* no Figma, onde são apresentados exemplos de telas e textos que descrevem os comportamentos desejados. O estagiário, sob a supervisão do líder técnico, pôde iniciar o desenvolvimento com base na documentação disponibilizada. O controle de versões é realizado por meio das ferramentas Git e GitHub.

### 3.3.1 O *microfrontend*

O *microfrontend* é construído utilizando a biblioteca React e codificada em TypeScript. Ele consiste em uma tela que permite aos participantes escolherem ou alterarem a forma de pagamento de sua inscrição em um evento específico.

Na Figura 3.4, observa-se o *microfronted* feito e incorporado na parte integradora. À esquerda, há uma lista de formas de pagamento que é renderizada de acordo com as opções configuradas para o evento. No canto superior direito, estão localizados botões que redirecionam

Figura 3.1 – Tela antiga construída em Rails.

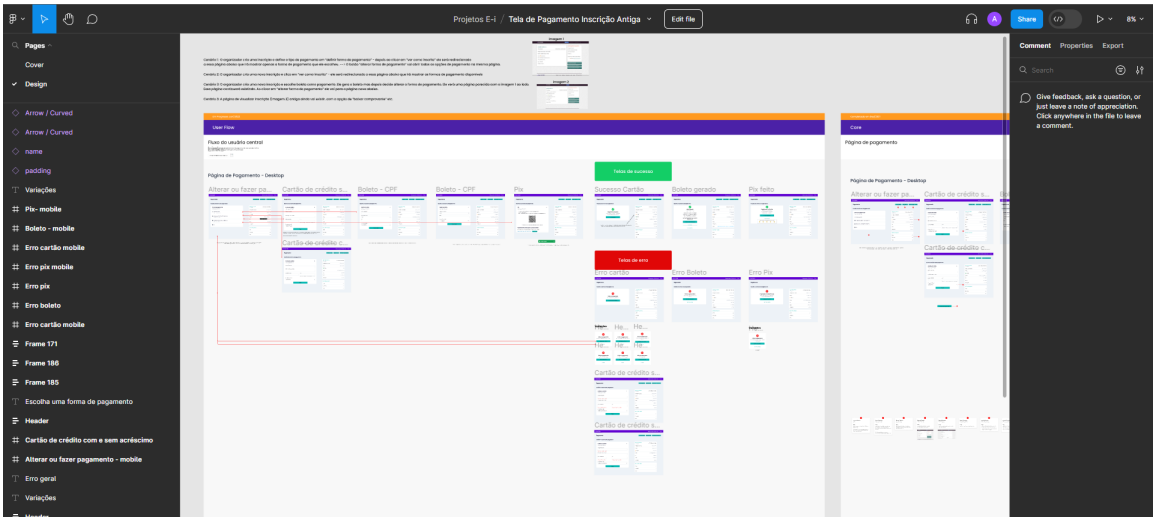

Fonte: Autor, 2023.

Figura 3.2 – Captura de tela do projeto no Basecamp.


Fonte: Autor, 2023.

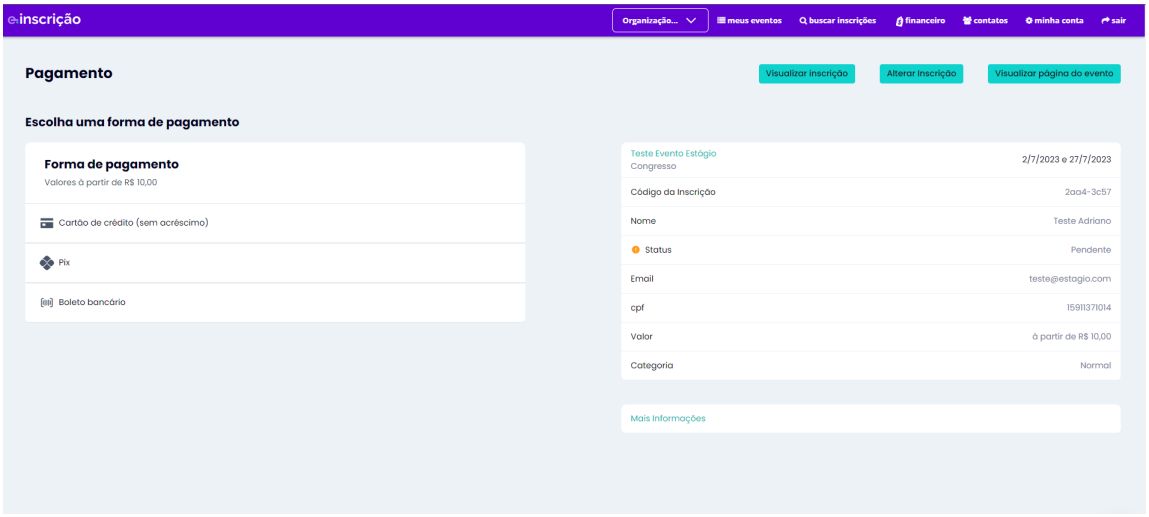
para outras partes do sistema, conforme descrito. Logo abaixo, há duas tabelas com informações dos participantes. A barra de navegação em roxo pertence à parte integradora do sistema e não faz parte do *microfrontend*. O *microfrontend* recebe as informações do participante e do

Figura 3.3 – Captura de tela da ferramenta Figma com protótipos e descrições de cenários.




Fonte: Autor, 2023.

Figura 3.4 – O microfrontend desenvolvido.



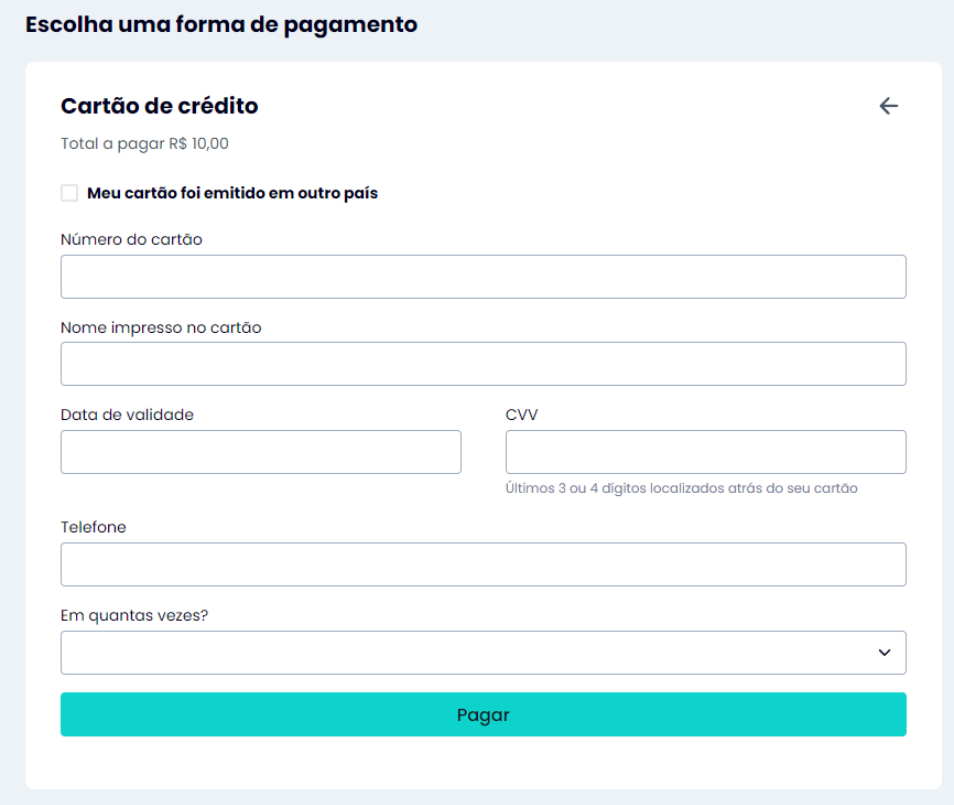

Fonte: Autor, 2023.

evento de uma API REST da parte integradora, que é solicitada assim que a página é carregada. A parte mais dinâmica da aplicação é onde está contida a listagem de formas de pagamento. Cada interação do usuário com uma forma de pagamento irá gerar um cenário diferente.

No cenário do cartão de crédito, é apresentado um formulário no qual o usuário insere os seus dados e, ao clicar em "Pagar", os dados são enviados para outra API REST JavaScript, que retorna uma situação de sucesso ou erro. O formulário pode ser visto na Figura 3.5.

Na Figura 3.6, observa-se o cenário de pagamento via Pix. No momento em que o usuário clica na opção, é feita uma requisição na mesma API REST JavaScript que traz as informações de pagamento do Pix. Assim, o participante pode ler o QRCode no seu aplicativo

Figura 3.5 – O formulário de cartão de crédito.




Fonte: Autor, 2023.

de banco ou copiar o link do Pix para realizar o pagamento. Uma contagem regressiva é iniciada e, durante esse tempo, o *microfrontend* realiza chamadas à API REST JavaScript verificando o status do pagamento. Se o pagamento for identificado, é exposto uma tela de sucesso. Caso o tempo seja esgotado, é exibida uma tela em que o usuário pode tentar novamente.

O último cenário é o de pagamento com Boleto. É fornecido um formulário no qual o usuário preenche seus dados. Ao clicar em "Gerar boleto", é feita uma requisição para a API REST JavaScript que gera um boleto pendente de pagamento. O *microfrontend* apresenta uma tela de sucesso caso o boleto tenha sido gerado ou uma de erro, caso não tenha sido possível fazer a transação. O formulário pode ser visto na Figura 3.7.

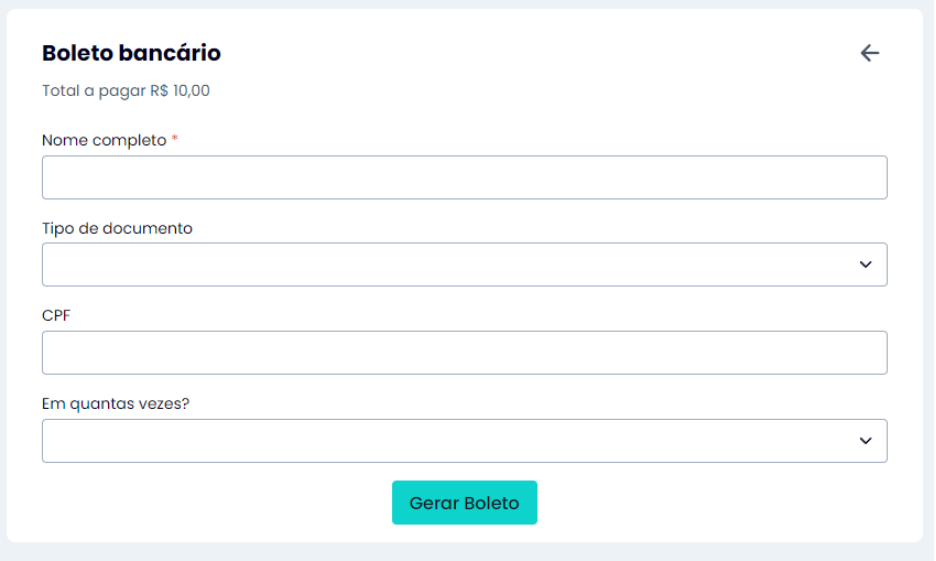
### 3.3.2 Incorporando o *microfrontend*

A incorporação é realizada por meio da injeção de um código *bundled* JavaScript na parte integradora, utilizando a tag <script> em um arquivo .erb. O código *bundled* é gerado pelo projeto do *microfrontend* por meio da funcionalidade de compilação do NPM e é publicado no

Figura 3.6 – O cenário de Pix.


Fonte: Autor, 2023.

Figura 3.7 – O formulário do Boleto.




Fonte: Autor, 2023.

registro npm com o comando *publish*. Dessa forma, o código fica publicamente disponível e é fornecido por meio de CDNs.

Com o código disponível na web, basta utilizar o atributo src da tag script, que importa arquivos de *script* externos, para ter acesso ao *microfrontend* dentro da parte integradora. Uma vez que o *microfrontend* está integrado, ainda é necessário ter uma forma de obter informa-

ções sobre o evento para fazer o download dos dados fornecidos pelas APIs. Para isso, são compartilhados dados por meio de *datasets*, que são atributos HTML que permitem armazenar informações definidas pela parte integradora e lidas pelo *microfrontend*.

### 3.3.3 Testes e liberação para clientes

Após a conclusão do desenvolvimento, o produto está completo, mas a funcionalidade ainda não é disponibilizada ao público antes de passar por uma etapa rigorosa de testes ponta-a-ponta. O processo de testes ponta-a-ponta é realizado por meio de uma planilha contendo casos de uso, elaborados pelo gestor do projeto, uma vez que a equipe não possui um analista de qualidade dedicado, devido a falta de recursos financeiros.

Durante o período de testes, participam desenvolvedores, membros da equipe de suporte e o gestor de projeto. Os testes ponta-a-ponta são realizados em um ambiente controlado. Os membros da equipe de suporte são responsáveis por seguir os casos de uso, registrando os resultados na planilha. Caso um teste falhe, o desenvolvedor responsável realiza as correções necessárias e o teste é repetido. Durante esse processo, o gestor analisa os *feedbacks* registrados na planilha, que servem como base para aprimorar a funcionalidade.

Devido à metodologia de testes adotada e à quantidade de pessoas envolvidas, esta etapa do projeto é bastante custosa e lenta, e isso evidencia a necessidade de implementação de testes automatizados. No entanto, os testes automatizados não são incluídos no planejamento do projeto e a equipe não possui experiência na sua implementação.

Quando todos os testes são aprovados, a equipe adquire confiança suficiente para disponibilizar a funcionalidade ao cliente final. Após a liberação, é realizado o acompanhamento por meio da equipe de suporte. Quaisquer bugs encontrados pelos clientes tornam-se chamados para a equipe de suporte, que por sua vez cria tarefas no Basecamp para sinalizar aos desenvolvedores a necessidade de correção.

## 3.4 Considerações finais

O estagiário participou de vários projetos durante seu período de estágio, todos atuando como desenvolvedor *frontend*. Neste relatório, mostra-se uma fração dessa atuação, focando apenas no projeto de alteração de forma de pagamento. Sua conclusão permite que um fluxo do sistema incorpore uma funcionalidade crucial para a empresa: o pagamento por Pix, que se torna a principal forma de pagamento na plataforma em termos de volume. Além disso, a

arquitetura adotada possibilita agilidade no desenvolvimento, e resulta na finalização do projeto em menos de três meses.

Para o estagiário, o projeto representa um significativo amadurecimento em termos de autonomia no trabalho. São tomadas diversas decisões técnicas que impactam na entrega, sempre buscando *feedbacks* com o líder técnico (que também atua como supervisor de estágio) e com o restante da equipe. Vale ressaltar que a experiência prévia do estagiário em empresa júnior tem um impacto positivo nas atividades realizadas no projeto.

Durante o desenvolvimento, foi encontrado o desafio de alinhar as especificações da interface com o protótipo criado pelo *designer*. Isso resultou em várias reuniões entre o estagiário e o *designer* para aprimorar a compreensão dos fluxos de funcionamento da interface. Embora demandem tempo, essas reuniões também possibilitam a identificação de cenários que não são percebidos anteriormente, permitindo a análise e a implementação de soluções adequadas.

Outro aspecto importante é que a nova interface construída possui componentes que ainda não foram criados no *design system*. Esse fato gera discussões sobre a evolução do *design system* da empresa, que posteriormente é descontinuado para dar lugar à criação de um novo.

Após a liberação para o cliente, o projeto passa por uma série de melhorias, principalmente em relação à comunicação com as APIs e ao aprimoramento do tratamento de erros. Este e outros projetos foram entregues durante a atuação de 1 ano do estagiário na empresa E-inscrição, projetos que impactam na evolução e melhoria do produto, e que entram efetivamente em uso pelos clientes.

# 4 CONCLUSÃO

A experiência vivenciada durante a realização do estágio supervisionado configura o início da carreira dos profissionais no mercado de trabalho. Apesar de se tratar de um estágio, é de suma importância que o aprendiz assuma responsabilidades que efetivamente gerem valor para a empresa e seus clientes, conferindo maior significado ao período inicial da carreira profissional. Nesse sentido, o presente relatório apresenta uma parcela das atribuições desempenhadas, por meio de um projeto que impactou os clientes da empresa e proporcionou valiosa experiência ao estagiário.

O projeto consistiu na implementação de um *microfrontend* que incorporou uma funcionalidade de alteração da forma de pagamento, visando aprimorar a arquitetura do sistema e adicionar a opção de pagamento via Pix. Contudo, a realização desse projeto não foi a única experiência adquirida durante o estágio. Dentre outras experiências relevantes, destacam-se: compreensão das regras de negócio de um sistema complexo; familiarização com diversas arquiteturas de *software*; compreensão da comunicação entre serviços web; interação com uma equipe multidisciplinar ágil; familiarização com a cultura organizacional horizontal de uma *startup*; ampla experiência em desenvolvimento *front-end* com JavaScript, entre outras.

Os desafios enfrentados durante o estágio na empresa E-inscrição foram variados. A adaptação ao trabalho remoto em conjunto com os estudos universitários durante o período da pandemia de Covid-19 foi um dos obstáculos enfrentados pelo estagiário. No entanto, essa situação proporcionou uma oportunidade para aprimorar a gestão pessoal, cuja evolução foi notável. O estágio também proporcionou um ambiente propício para o exercício da comunicação e das relações interpessoais, o que também foi outro ponto de melhoria observado. Houve diversas situações desafiadoras e, de modo geral, a vivência no estágio proporcionou um avanço significativo na carreira profissional, o que foi comprovado pela efetivação como desenvolvedor júnior ao término do estágio.

Além disso, é importante mencionar o papel da UFLA (Universidade Federal de Lavras). O curso de Sistemas de Informação oferece uma sólida base e fundamentos para preparar os indivíduos para ingressarem no mercado de trabalho, especialmente nos campos de gestão e desenvolvimento. Além disso, a universidade conta com entidades, como as empresas juniores, que preparam os alunos de forma excepcional para o mercado de trabalho.

No que diz respeito aos pontos de melhoria a serem mencionados, em relação à empresa, sugere-se investir mais na capacitação de seus desenvolvedores, proporcionando mais oportu-

nidades e recursos para enriquecer o aprendizado de todos. Além disso, é importante investir em profissionais com maior nível de senioridade para garantir a estabilidade de um sistema complexo, e se atentar aos quesitos de qualidade, implementando testes automatizados.

Quanto à universidade, sugere-se que sejam oferecidos mais projetos que proporcionem uma vivência mais corporativa aos alunos, como as empresas juniores. Isso permitirá que os estudantes tenham uma experiência prática que se assemelhe ao ambiente empresarial.

# REFERÊNCIAS


BASECAMP. Refreshingly simple project management. 2023. Disponível em: <https://basecamp.com/>.

BENSON, E. **The Art of Rails®**. [S.l.]: Wiley Publishing, 2008.

CHACON, S. git –fast-version-control. 2023. Disponível em: <https://git-scm.com/about>.

CLOUDFLARE. O que é rede de distribuição de conteúdo (cdn)? como a cdn funciona? 2023. Disponível em: <https://www.cloudflare.com/pt-br/learning/cdn/what-is-a-cdn/>.

FIGMA. How you design , align , and build matters. do it together with figma. 2023. Disponível em: <https://www.figma.com/?fuid=>.

FRISBIE, M. **Professional JavaScript for Web Developers**. [s.n.], 2019. Disponível em: <https://www.google.com.br/books/edition/Professional_JavaScript_for_Web_Develope/3GOuDwAAQBAJ>.

GEERS, M. Micro frontends. 2017. Disponível em: <https://micro-frontends.org/>.

GITHUB. Let's build from here. 2023. Disponível em: <https://github.com/>.

JACKSON, C. Micro frontends. 2019. Disponível em: <https://martinfowler.com/articles/micro-frontends.html>.

MDN. Css. 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.

MDN. Html: Linguagem de marcação de hipertexto. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.

MDN. Spa (single-page application). 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>.

META. React. a javascript library for building user interfaces. 2023. Disponível em: <https://legacy.reactjs.org/>.

META. React docs. 2023. Disponível em: <https://react.dev/>.

MICROSOFT. Typescript documentation. 2023. Disponível em: <https://www.typescriptlang.org/docs/handbook/>.

MILETTO, E. M.; BERTAGNOLLI, S. D. C. **Desenvolvimento de Software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP.** 1. ed. Disponível em: Minha Biblioteca: Grupo A, 2014: BOOKMAN®, 2014.

NPM. Npm. 2023. Disponível em: <https://www.npmjs.com/>.

SAUDATE, A. **APIs REST**. [s.n.], 2021. Disponível em: <https://www.google.com.br/books/edition/APIs_REST/FVkXEAAAQBAJ>.

SCRUM.ORG. What is scrum?. 2023. Disponível em: <https://www.scrum.org/learning-series/what-is-scrum>.

TSITOARA, M. **Beginning Git and GitHub**. [s.n.], 2019. Disponível em: <https://www.google.com.br/books/edition/Beginning_Git_and_GitHub/3xfBDwAAQBAJ>.

VESSELOV, S.; DAVIS, T. **Building Design Systems**. [s.n.], 2019. Disponível em: <https://www.google.com.br/books/edition/Building_Design_Systems/ZjuSDwAAQBAJ>.