INTERFACE TECNOLÓGICA

MICROFRONTEND: um estudo sobre o conceito e aplicação no *frontend*

MICROFRONTEND: a study on the concept and application in the frontend

Charlles Henrique Paiva do Nascimento – charlleshenriquepaiva@hotmail.com

Eder Carlos Salazar Sotto – eder.sotto@fatectq.edu.br

Faculdade de Tecnologia de Taquaritinga (FATEC) – Taquaritinga – São Paulo – Brasil

DOI: 10.31510/infa.v17i1.798

RESUMO

Este artigo tem como objetivo apresentar o conceito e a aplicação da arquitetura de *microfrontend*, demonstrando as vantagens e riscos existentes em sua utilização no desenvolvimento de *software* em comparação ao modelo tradicional monolítico existente hoje. Para isso, foi realizada uma pesquisa bibliográfica utilizando livros, artigos, documentações oficiais e foi desenvolvida uma prova de conceito utilizando essa arquitetura. Entretanto conclui-se que esse tipo de arquitetura de *microfrontend* representa um avanço no desenvolvimento de aplicações para a Web, tendo em vista que traz benefícios de organização dos códigos, independência dos projetos, times e escalabilidade, possibilitando para os desenvolvedores de *software* uma melhoria na criação de soluções, já que diminuem-se as barreiras existentes no modelo monolítico.

Palavras-chave: *Microfrontend*. Aplicações Web. *Desenvolvimento* Web. *Frontend*.

ABSTRACT

This article has the objective to show the concepct and application of microfrontend architecture, demonstrating its advantages and risks within the universe of software development, compared to the traditional monolithic model. For this a bibliographic research was carried out using books and scientific articles and a prof a concept was developed using this architecture model. Though it is concluded that the microfrontend architecture represents a big advance for the future of software development for Web, given that brings benefits of code organization, project independence, teams and scalability, enabling software developers to improve the creation of solutions, since the existing barriers in the monolithic model are reduced.

Keywords: Microfrontend. Web Applications. Web development. Frontend.

1 INTRODUÇÃO

O acesso às tecnologias de informação vem se tornando cada vez mais imprescindível, obrigando as empresas a agirem rápido para se manterem competitivas. O poder e a presença da tecnologia da informação se expandiram, e as instituições passaram a percebê-la como um recurso cada vez mais decisivo para o seu sucesso (CARR, 2009).

Conforme as necessidades surgem, as empresas precisam evoluir seus sistemas, para que estes consigam atender às demandas existentes. Diante disso, alguns aspectos no desenvolvimento de *software* acabam sendo impactados, como a complexidade das regras de negócios incorporadas nos códigos-fontes dos sistemas de informação. De acordo com Newman (2015), quando se adicionam novos recursos, a base de código cresce e com o tempo pode ser difícil realizar-se mudanças, em razão do código ser muito extenso e complexo.

Além disso, adicionar novos recursos e funcionalidades a um sistema pode ser um ponto crítico para as empresas, visto que a escalabilidade do sistema se torna algo complexo. Segundo Jackson (2019), ter um bom desenvolvimento no *frontend* é difícil, e escalar equipes para trabalharem simultaneamente em um produto grande e complexo é ainda mais difícil.

Sendo assim, este trabalho tem como objetivo demonstrar a arquitetura de *microfrontend,* e como dividir o *frontend* em partes menores pode ajudar a aumentar a eficiência e eficácia do trabalho das pessoas que darão manutenção no código-fonte. Outro objetivo é discutir os impactos causados pela evolução dos sistemas de informação, quais são as vantagens e desvantagens da arquitetura *microfrontend*, bem como de sua utilização.

2 ARQUITETURAS E FERRAMENTAS

Este trabalho vista demonstrar uma abordagem que vem sendo discutida para facilitar o desenvolvimento no *frontend* e como diminuir as aplicações monolíticas em várias pequenas aplicações, com o objetivo de tornar o fluxo de trabalho e melhorias em um processo mais rápido e escalável. Além disso, este capítulo apresenta as ferramentas e tecnologias utilizadas no desenvolvimento de uma aplicação *microfrontend*.

2.1 Arquiteturas Monolíticas

Segundo Krishnamurthy (2018), uma arquitetura monolítica descreve uma aplicação de *software* de camada única, onde todos os serviços e componentes estão combinados em um único programa. Isso acaba impactando na complexidade da aplicação, visto que cria-se um alto acoplamento entre estes módulos do sistema.

De acordo com Jackson (2019), em aplicações monolíticas, quando diferentes times desejam trabalhar no mesmo sistema, um dos problemas verificados é que, devido o alto acoplamento e complexidade existentes, podem haver conflitos nos trabalhos realizados, já que estes compartilham da mesma base de código, e isso acaba dificultando o desenvolvimento. Este tipo de arquitetura também dificulta a inclusão de novos recursos, tanto da linguagem de programação quanto de outras tecnologias (como *frameworks)*, pois uma simples alteração pode comprometer o funcionamento de todo o sistema, uma vez que este utiliza uma estrutura monolítica.

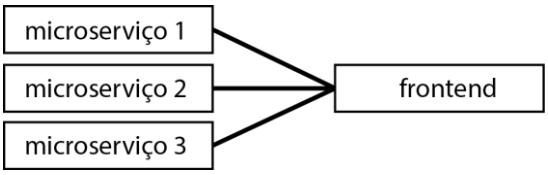
Nas empresas, é comum desenvolverem aplicações Web *frontend* utilizando uma ou mais tecnologias, como *HyperText Markup Language (HTML), Cascading Style Sheets (CSS)* e JavaScript, sendo o *frontend* destes sistemas executados no navegador de internet do usuário. Nessas aplicações, mesmo que seus módulos e serviços estejam estruturalmente organizados de forma separada, no fim tudo fará parte de uma grande aplicação.

Um dos pontos negativos, segundo Almeida (2015) é que, em aplicações monolíticas, quando ocorre um erro em determinada parte do sistema, pode acontecer de todo o restante (mesmo que não tenha nenhuma relação com a parte alterada) do sistema também parar de funcionar. Outro ponto negativo que Almeida (2015) também cita é que, tendo uma base de código muito grande, o entendimento do sistema por novos membros da equipe acaba sendo um processo difícil, impactando na produtividade.

Yang et al. (2018) cita que cada modificação no projeto deverá atualizar o sistema inteiro no ambiente de produção, mesmo que tenha sido alterado ou adiciono algum recurso pequeno, ainda que em uma parte especifica da aplicação, impactando no tempo necessário para realização da atualização.

Na Figura 1 pode-se verificar um exemplo visual de uma arquitetura monolítica no *frontend*.

Figura 1 - Arquitetura monolítica



Fonte: Elaborado pelos autores (2020)

2.2 Arquitetura Microfrontend

De acordo com Jackson (2019), nos últimos anos os microsserviços se popularizaram dentro das organizações, pois esse tipo de arquitetura reduz as limitações existentes em grandes *backends* e monolitos, mas em contrapartida as bases de código no *frontend* continuam complexas e difíceis. Yang et al. (2018) definem que o microsserviços são uma variação do estilo de arquitetura orientada a serviços, onde cria-se uma aplicação como uma coleção de serviços, e estes possuem baixo acoplamento.

Malipense e Zuchi (2018) descrevem microsserviços como um modelo de arquitetura que compõe vários serviços, onde cada um dos serviços são independentes dos demais, sendo cada um responsável por um determinada funcionalidade, onde normalmente comunicam-se uns com os outros via A*pplication Programming Interface* (API) de recursos *HyperText Transfer Protocol* (HTTP).

Com esse tipo de abordagem mostrando benefícios no *backend,* as empresas perceberam uma grande vantagem em sua utilização, possibilitando então surgimento da ideia de replicar este mesmo princípio no *frontend*.

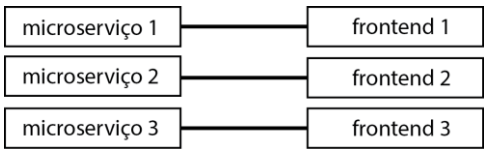
Segundo Yang et al. (2018), a ideia *do microfrontend* é tratar uma aplicação Web como uma combinação de recursos, onde cada time cuida de uma parte desta aplicação, onde cada qual atende a um negócio ou função específicos.

Krishnamurthy (2018) *microfrontend* como uma abordagem de microsserviços, porém aplicado no *frontend,* onde a ideia é decompor a aplicação Web em unidades menores, para que cada uma trate uma determinada funcionalidade, ao invés de desenvolver uma grande aplicação monolítica.

Essa abordagem permite que cada time trabalhe conforme a necessidade, utilizando diferentes tecnologias na criação das aplicações e fazendo implantações de forma independente.

Na Figura 2 é demonstrado um exemplo visual de uma aplicação construída em uma arquitetura *microfrontend.*

Figura 2: Arquitetura microfrontend



Fonte: Elaborada pelos autores (2020)

2.3 *Single*-SPA

De acordo com a documentação oficial *Single*-SPA (2020), o *Single*-SPA é um *framework* para reunir vários *microfrontends* JavaScript em uma aplicação *frontend,* tendo em vista auxiliar em atividades tais como:

- Utilizar diferentes *frameworks* no *frontend,* como *ReactJS*, *Angular*, *Vuejs* entre outros.
- Implantar os *microfrontends* de forma independente.
- Melhorar o tempo de carregamento inicial.

2.4 ReactJS

De acordo com Copes (2019), o React foi desenvolvido no *Facebook* e lançado no mundo em 2013, sendo utilizado em alguns dos aplicativos mais utilizados do mundo, como o próprio *Facebook* e o *Instagram*.

Segundo Pandit (2018), o React é uma biblioteca *JavaScript* de código aberto utilizado para criar interface com usuários, especificamente *Single Page Application* (SPA*)*, sendo uma biblioteca capaz de criar grandes aplicações Web, com a capacidade de alterar seu estado e seus dados sem a necessidade de recarregar a página, além de ser rápida, escalável e simples.

A documentação oficial em REACT (2013), afirma que o React não é um *framework Model, View, Controller* (MVC) e sim uma biblioteca para criação de componentes de interfaces de usuário, que exibem dados que podem ou não serem alterados a qualquer momento.

2.5 Angular

Segundo Branas (2014), o AngularJS foi criado por Miško Hevery e Adam Abron em 2009, sendo este um *framework* JavaScript de código aberto que é executado do lado do cliente (*client-side),* e que auxilia na produtividade do desenvolvimento Web. Em um dos projetos que Hevery trabalhou, notou-se a existência de cerca de 17 mil linhas de código, quando decidiu reescrever esse software utilizando seu *framework,* o projeto passou a possuir apenas 1.500 linhas de código, fazendo com que ganhasse grande aceitação em outros projetos dentro da empresa.

Angular (2020) descreve o Angular como um *framework* de código aberto para criação de SPAs, criada e mantida pelo Google. Freeman (2018) descreve que o Angular explora aspectos do desenvolvimento *backend* no *frontend,* e que a biblioteca foi construído com o *design pattern* MVC, o que torna os projetos construídos com angular extensíveis, de fácil manutenção, testáveis e padronizados.

Conforme Booth (2017) cita, o *framework* passou por modificações em sua segunda versão, que na realidade foram uma reformulação completa do *framework,* onde verificou-se um ganho de inúmeras funcionalidades novas, inclusive com compatibilidade das versões recentes do TypeScript.

2.6 Vue.js

Conforme a documentação Vue.js (2020), o vue.js é um *framework* para construção de interfaces com o usuário focado na camada de visualização e com fácil integração a outras bibliotecas e projetos existentes, sendo capaz de ser utilizado para criação de aplicações SPA sofisticadas.

A função da lib reativa Vue é observar um objeto JavaScript e refletir qualquer mudança do seu estado no DOM do HTML. (LACERDA, 2017).

Filipova (2016) diz que a ideia do Vue surgiu quando Evan You trabalhava na Google Creative Labs, com o objetivo de criar um protótipo que oferecesse de forma fácil e flexível o *data binding* reativo e componentes reutilizáveis.

Kyriakidis (2016) descreve o Vue como um excelente ecossistema de *plugins* e ferramentas que estende em qualquer serviço, mesmo que básico e com a possibilidade de ser

incluído em qualquer projeto com poucas linhas de código. Ele ainda descreve o Vue como sendo rápido, leve e o futuro do desenvolvimento *frontend*.

3 METODOLOGIA

Para realização deste trabalho, foi desenvolvida uma pequena aplicação utilizando cada um dos *frameworks* citados. Este trabalho também se baseou em revisões bibliográficas dos assuntos abordados, focada em artigos, livros e, devido à ausência de conteúdos acadêmicos sobre alguns assuntos abordados, foi utilizada também a documentação oficial de tais tecnologias como base de estudo.

4 CONSIDERAÇÕES SOBRE O USO DE MICROFRONTEND

Conforme os sistemas vão escalando e consequentemente a base de código vai acompanhando esse crescimento, manter uma arquitetura escalável torna-se cada vez mais necessária, porém é um processo complicado e desafiador. As aplicações grandes trazem naturalmente uma complexidade maior, o que muitas vezes dificulta o entendimento do funcionamento de toda a aplicação, seja para os novos desenvolvedores de *software* e até mesmo para os mais experientes, impactando diretamente na produtividade, no trabalho em equipe, qualidade de entrega entre outros.

O conceito de micro aplicações no *frontend* traz uma abordagem que divide uma grande aplicação em outras diversas aplicações menores, com times dedicados, permitindo a realização de entregas de novos recursos e atualizações de forma independente, dando assim mais autonomia para os desenvolvedores, gerando mais valor e otimizando a entrega do produto ou serviço atendidos pelo sistema.

4.1 Vantagens do *microfrontend*

Segundo Jackson (2019), uma das vantagens da abordagem de *microfrontend* é a capacidade de ter bases de códigos mais simples e desacopladas. Isso permite que o código fonte seja menor, tendendo a ser mais simples e fácil para os desenvolvedores trabalharem. Também vale ressaltar que, tendo as funcionalidades desacopladas, é possível que estas

funcionalidades trabalhem de forma isolada, fazendo com que, em caso de falha, somente a aplicação inoperante ficará indisponível, mantendo todo o restante em funcionamento.

O *microfrontend* ainda tem outros benefícios como a independência dos times e a diminuição do tempo necessário para liberação de novos recursos e atualizações nas aplicações. De acordo com Krishnamurthy (2018), ao dividir as aplicações monolíticas em micro aplicações separadas, as equipes têm uma maior autonomia e flexibilidade, tanto para escolher qual tecnologia a ser utilizada, como no tempo necessário para lançar novos produtos ou serviços.

As bases de códigos independentes permitem que as equipes integrem seus recursos e realizem entregas com maior velocidade, permitindo testar novas tecnologias sem se preocupar com o fato de que estas alterações possam impactar recursos de outros times. Newman (2018) afirma que, em uma aplicação monolítica, caso seja testada uma nova linguagem de programação, base de dados ou qualquer outro tipo de tecnologia, a mudança pode impactar em uma grande parte do sistema.

Sendo assim, a arquitetura de *microfrontend* ajuda a explorar novos meios para que o desenvolvimento possa ser cada vez mais inovador, mostrando que as aplicações monolíticas podem vir ser um grande risco com tempo, por serem complexos, difíceis de manter e atualizar.

4.2 Desvantagens do *microfrontend*

Mesmo com as grandes vantagens existentes na abordagem do *microfrontend*, é importante conhecer as dificuldades que podem surgir quando se decide utilizar essa arquitetura. Segundo Jackson (2019) e Krishnamurthy (2018), o *microfrontend* possui desvantagens como:

- **Tamanho dos arquivos:** Os pacotes das aplicações independentes podem ter duplicações de dependências, aumentando significativamente o tamanho dos arquivos baixados pelo navegador do usuário.
- **Complexidade operacional:** Desvantagem que também existe com a arquitetura de microsserviços. Tendo uma arquitetura mais distribuída, as aplicações terão mais coisas para serem gerenciadas, ou seja, mais repositórios, ferramentas, compilações e pipelines de entrega. Sendo assim, é necessário uma certa maturidade para conseguir gerenciar tudo isso.

- **Estratégia de Testes:** Exige que as equipes escrevam os melhores testes para que, caso algum recurso esteja quebrado, não seja disponibilizado para o usuário final.
- **Experiencia do usuário:** Manter um padrão de *User Experience (*UX) e *User Interface* (UI) pode ser difícil, já que cada equipe trabalha de forma independente e podem construir os sistemas como acharem melhor.

4.3 Criando uma aplicação com *microfrontend*

Com o objetivo de exemplificar a arquitetura de *microfrontend*, foram desenvolvidas pequenas aplicações que utilizam a API do sistema de repositórios Github para pesquisar e listar usuários e repositórios. Foi criado também um *microfrontend* responsável por um menu lateral, para possibilitar a navegação entre os outros *microfrontend*s. O projeto foi desenvolvido utilizando os *frameworks* e bibliotecas *Single*-SPA, Angular, React e Vue.js, todos eles já mencionados anteriormente.

Neste projeto, foram criadas quatro aplicações, com diferentes tecnologias e responsabilidades:

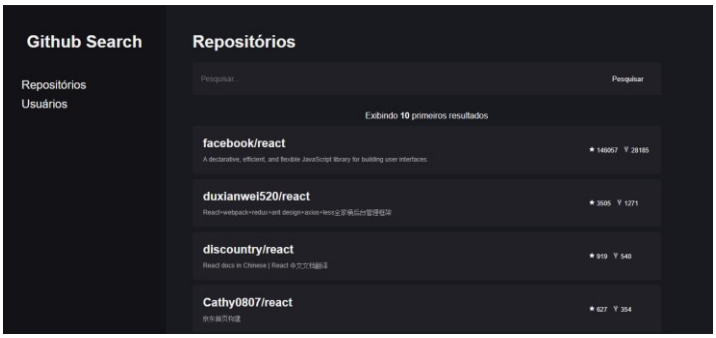
- ***App-client-github***: responsável por carregar as micro aplicações e exibi-las quando solicitadas.
- ***Sidebar***: criado com Vue.js, tem como finalidade exibir um menu para navegação entre as aplicações.
- ***searchRepositories***: criado com React, exibe um campo de busca e uma listagem dos repositórios.
- ***searchUsers***: criado com Angular, tem como responsabilidade listar e pesquisar os perfis de usuários do Github.

O código fonte do projeto encontra-se disponível para consulta no Github no endereço **https://github.com/charllespaiva/microfrontend-github-search**.

Na tela inicial da aplicação é exibido o menu lateral referente ao *microfrontend Sidebar,* com a listagem dos apps disponíveis para acesso. Por padrão é carregado também a aplicação *searchRepositories,* onde é exibida uma listagem com os 10 primeiros repositórios que possuam o termo “react” em seu nome. É possível também realizar uma busca pelo campo de pesquisa exibido na tela, onde a aplicação fará uma busca na API do Github e atualizará a listagem para exibir os 10 primeiros resultados. Em casa resultado é possível

visualizar o nome do repositório, que quando clicado, redireciona para o repositório original no site do Github, sendo exibidas também a descrição, quantidade de pessoas que curtiram aquele repositório e número de pessoas que fizeram um *fork (*ramificação ou cópia). Na Figura 3 pode-se visualizar os dois *microfrontends* sendo exibidos.

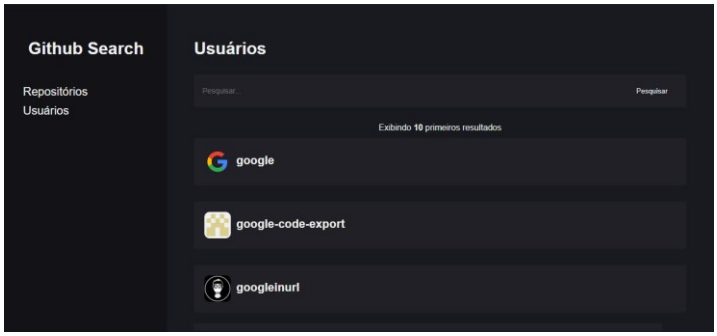
Figura 3 - Tela de repositórios



Fonte: Elaborada pelos autores (2020)

Após o usuário clicar na opção “Usuários” no menu lateral, é apresentada uma página que traz um campo de busca e uma listagem de usuários do Github. Por padrão, estão sendo exibido perfis de usuários que possuam o termo “google” em seu nome e que, quando clicado, é aberta uma nova aba no navegador, exibindo o perfil no site do Github, conforme pode ser visto na Figura 4.

Figura 4 – Tela de Usuários



Fonte: Elaborada pelos autores (2020)

Ao a arquitetura de *microfrontend* dentro da aplicação, é possível ter algumas vantagens como o desacoplamento entre as apps. Neste cenário, caso ocorra de o *microfrontend searchRepositories* falhar ou apresentar qualquer tipo de instabilidade, a aplicação *Sidebar* e *searchUsers* continuarão funcionando normalmente, sem qualquer tipo de

impacto. Caso novos *microfrontends* sejam adicionados ao projeto, somente o que estiver com problema ficará indisponível na tentativa de acesso.

5 CONSIDERAÇÕES FINAIS

Com base no estudo realizado, conclui-se que o uso de *microfrontend* possibilita um grande avanço no desenvolvimento de *software para a Web*, principalmente em aplicações que utilizam microsserviços, já que seu uso possibilita a diminuição da complexidade de grandes monolitos e também a diminuição do acoplamento e consequente melhora na escalabilidade dos componentes, melhorando também a organização das bases de código e gerando ganho para as empresas na hora de projetar suas soluções de forma flexível, ao contrário do modelo monolítico.

Entretanto, apesar das vantagens existentes nessa arquitetura, é importante que haja uma maturidade e consciência na escolha desse modelo, visto que em aplicações menores pode haver uma complexidade desnecessária, já que será preciso gerenciar os processos de *deploy,* além de uma atenção especial para a experiencia do usuário, para que se mantenha uma uniformidade entre os sistemas. O uso inadequado do *microfrontend* pode gerar mais problemas que benefícios, como visto nas desvantagens citadas anteriormente, fazendo então com que seu uso seja uma solução adequada a depender do cenário onde será utilizado. Desta maneira, caso seja utilizado de forma correta, a arquitetura de *microfrontend* se mostra uma solução muito vantajosa.

Por meio da aplicação que utiliza a arquitetura de *microfrontend* desenvolvida neste artigo, foi possível exemplificar o seu funcionamento, aplicando os conceitos apresentados na prática.

REFERÊNCIAS

ALMEIDA**,** Adriano**. Arquitetura de microserviços ou monolítica?**. Disponível em <http://blog.caelum.com.br/arquitetura-de-microservicos-ou-monolitica/>. Acesso em: 27 mar. 2020.

ANGULAR(2020). **Angular:** Introduction to Angular concepts. Angular Docs, 2020. Disponível em: <https://angular.io/guide/architecture>. Acesso em: 09 de março de 2020.

BOOTH J. D. **Angular 2 Succinctly**. Morrisville,United States of America: Syncfusion, 2017.

BRANAS, Rodrigo. **AngularJS essentials**: Design and construct reusable, maintainable, and modular web applications with AngularJS. Birmingham, United Kingdom: Packt Publishing, 2014.

CARR, Nicholas G. **Será que TI é tudo?:** Repensando o papel da tecnologia da informação. 1 ed. São Paulo: Editora Gente, 2009.

COPES, Flavio. **The React Handbook.** 2019. Disponivel em: < https://www.freecodecamp.org/news/the-react-handbook-b71c27b0a795/>. Acesso em: 27 de março de 2020.

FILIPOVA, Olga. **Learning Vue.js 2**. Birmingham, United Kingdom:Packt Publishing. 2016.

FREEMAN, Adam. **Pro Angular 6.** 3 ed. London: Apress, 2018

JACKSON, Cam. **Micro Frontends.** Disponível em <https://www.martinfowler.com/articles/micro-frontends.html>. Acesso em: 27 de mar. 2020

KRISHNAMURTHY, Santhosh. What are Micro Frontends. **IEEE India Info**, v. 14, n. 4, p. 105-109, out/dez 2019.

KYRIAKIDIS A.; MANIATIS K.; YOU E. **The Majesty of Vue.js**. Leanpub, 2016.

LACERDA, Milene. **No mar de libs e frameworks: conhecendo o Vue.js –Parte I.** [s.l.]:BrazilJS, 2017. Disponível em:<https://braziljs.org/blog/no-mar-de-libs-e-frameworks-conhecendo-o-vue-js-parte>. Acesso em: 23 mar. 2020.

MALIPENSE, L. M.; ZUCHI, J. D. UM ESTUDO SOBRE O CONCEITO E A APLICAÇÃO DA ARQUITETURA DE MICROSERVIÇOS. **Revista Interface Tecnológica**, v. 15, n. 1, p. 122-134, 30 jun. 2018.

NEWMAN, Sam. **Building Microservices**: Designing fine-grained systems. 1. ed. Califórnia: O'Reilly Media, 2015.

PANDIT, Nitin. **What Is ReactJS and Why Should We Use It?** 2018. Disponível em: <www.c-sharpcorner.com/article/what-and-why-reactjs>. Acesso em: 12 março 2019.

REACT (2013). **React**: Why did we build React?. React Blog, 2013. Disponível em: <https://reactjs.org/blog/2013/06/05/why-react.html >. Acesso em: 09 de março de 2020.

SINGLE-SPA (2020). **Single-SPA:** Getting Started with single-spa, Docs, 2020. Disponivel em: <https://single-spa.js.org/docs/getting-started-overview>. Acesso em: 27 de março de 2020

VUEJS(2020). **Vue.js:** What is Vue.js?. Vue.js Docs, 2020. Disponível em: <https://vuejs.org/v2/guide/>. Acesso em: 09 de março de 2020.

YANG C., CHUANCHANG L., ZHIYUAN S. **Research and Application of Micro Frontends.** IOP Conf. Ser.: Mater. Sci. Eng, Beijing, ano 19, n 490, abr. 2019. Disponível em: <https://iopscience.iop.org/article/10.1088/1757-899X/490/6/062082/pdf> Acesso em: 27 de mar. 2020.