INTERFACE TECNOLÓGICA

REUSABILIDADE NO DESENVOLVIMENTO DE UM SISTEMA WEB UTILIZANDO O FRAMEWORK ANGULAR

REUSABILITY IN THE DEVELOPMENT OF A WEB SYSTEM USING THE ANGULAR FRAMEWORK

Ian Rotondo Bagliotti – ian.bagliotti@gmail.com

Daniela Gibertoni – daniela.gibertoni@fatectq.edu.br

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

DOI: 10.31510/infa.v17i1.826

RESUMO

No cenário mundial o desenvolvimento de softwares aumentou consideravelmente a sua procura com o foco em solucionar, otimizar problemas ou oportunidades de mercado, garantindo a qualidade do produto, a satisfação do cliente e agregar valor ao negócio, sendo exigidos prazos e tempo de entrega cada vez mais curtos. No meio da vasta concorrência por melhorias e entregas ágeis, surgem mecanismos e tecnologias para aumentar a produtividade na codificação, facilitar a manutenibilidade por meio da reusabilidade e oferecer uma base para iniciar a implementação por meio dos *frameworks*. Este artigo aborda, por meio de pesquisas bibliográficas exploratória e experimental, as técnicas de reuso de código, a partir do qual foi desenvolvido um sistema web para uma entidade filantrópica utilizando o *framework front-end* Angular e Node.js no *back-end*, com a finalidade de investigar e identificar as principais características de reusabilidade neste estudo de caso. Como principais resultados obtidos destacam-se a busca da otimização no processo de criação e a manutenbilidade, por meio de práticas de reusabilidade e por funcionalidades do Angular.

Palavras-chave: Reusabilidade. Angular. Sistema Web. Framework. Desenvolvimento.

ABSTRACT

On the world stage, software development has considerably increased its demand with a focus on solving, optimizing problems or market opportunities, guarantee product quality, customer satisfaction and add value to the business, requiring short deadlines and delivery times being required increasingly short. In the midst of the vast competition for improvements and agile deliveries, mechanisms and technologies emerge to increase productivity in coding, facilitate maintainability through reusability and provides a basis to start implementation through frameworks. This article addresses, through exploratory and experimental bibliographic research, the techniques of code reuse, from which a web system was developed for a philanthropic entity using the framework front-end Angular e Node.js on back-end, with the purpose of investigating and identifying the main characteristics of reusability in this case

study. The main results obtained are the search for optimization in the creation process and maintainability, through reusability practices and Angular functionalities.

Keywords: Reusability. Angular. Web System. Framework. Development.

1 INTRODUÇÃO

Com a expansão da procura de *softwares*, fica evidente a exigência de desenvolvimento de sistemas cada vez mais complexos e com qualidade para garantir a competitividade de mercado. Para suprir essa necessidade, foi desenvolvido inúmeras técnicas, que são abordadas pela a Engenharia de Software em busca de otimizar o processo de implementação, tais como a reusabilidade (FERREIRA e NAVES, 2011).

Para os mesmos autores Ferreira e Naves (2011), desenvolvedores e engenheiros constataram que uma expressiva parte de um código como um todo pode ser reutilizada para o desenvolvimento de uma parte de outro sistema. Para isso, surgiu o reuso (ou conhecido também como reusabilidade) de código, no qual o objetivo é reutilizar uma aplicação, partes de código ou funcionalidades em diversos lugares (FRAKES, 1996), uma vez em que já foi desenvolvido, testado e validado, empregando a filosofia de eliminar a repetição de código. Assim é possível aumentar a qualidade de *software* e diminuição do prazo/tempo da entrega do *software*.

É de suma importância saber o quanto o *framework* (significa em tradução para o português “estrutura” – é a abstração de códigos comuns entre os *softwares* provendo uma funcionalidade genérica) pode ser útil em aplicações web no dia a dia, devido a sua economia e reusabilidade de código, gerando produtividade em projetos. Além disso, segundo Sommerville (2007), gradativamente as fábricas de softwares veem os seus produtos como um ativo de valor, viabilizando o desenvolvimento a partir do reuso de uma aplicação já existente para aumentar o seu retorno sobre os investimentos. Ao longo do tempo, o reúso foi crescendo e se tornando essencial para a elaboração de uma ideia até o desenvolvimento de um software. Essa técnica foi se aperfeiçoando pelos engenheiros de software, possibilitando a reutilização de código de modo mais abrangente e expressiva, tal como o uso globalizado de APIs (Application Programming Interface, que significa em tradução para o português "Interface de Programação de Aplicativos") (BRAUDE, 2005).

A metodologia utilizada neste artigo é a pesquisa bibliográfica exploratória e experimental, de modo que se realizou uma abordagem dos temas de uma maneira prática e

dinâmica das metodologias e técnicas de reusabilidade de *software*, framework Angular, Node.js, capaz de elaborar um estudo de caso ao desenvolver um sistema web para uma entidade filantrópica.

O objetivo desse artigo é investigar e identificar as principais características de reusabilidade no desenvolvimento de um sistema web utilizando o *framework* Angular para uma entidade filantrópica.

Este artigo está estruturado em quatro sessões: a primeira delas é a introdução que traz a contextualização proposta deste trabalho, sendo que na sessão dois é apresentada a fundamentação teórica que contem as informações sobre o tema chave reusabilidade e as tecnologias utilizadas para o desenvolvimento do estudo de caso (sessão quatro), pois na sessão três contem a metodologia utilizada nesta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção dedica-se a apresentar os conceitos referentes ao tema central deste artigo que é reusabilidade bem como apresentar as tecnologias que foram utilizadas no estudo de caso no qual foi desenvolvido um sistema web.

2.1 Reusabilidade

A aplicação da reusabilidade é apontado por muitos pesquisadores o principal propósito da Engenharia de Software (FERREIRA e NAVES, 2011). A reusabilidade tem como propósito fundamental manter a qualidade, a eficácia e a efetividade na elaboração e manutenção de um *software*. No momento em que se utiliza fragmentos ou partes de *software* já avaliados e prontos, não será necessário a codificação e a verificação de erros, garantido a qualidade e o funcionamento total do *software* (REZENDE, 2005).

Para Pressman (1995), o reuso de *software* é um aspecto fundamental de um componente de alta qualidade, sendo necessária a projeção e implementação do componente de modo que possa ser reaproveitado em diversos programas.

O reuso é uma abordagem de desenvolvimento que se tornou dominante para o desenvolvimento de *softwares*. Ao construir sistemas, são procurados meios de integrar

componentes, sistemas, métodos e *frameworks* já existentes, em vez do desenvolvimento de um sistema do zero (SOMMERVILLE, 2011).

Segundo Hirama (2012), o desenvolvimento de um *software* melhor, mais rapidamente e com o menor custo, é necessário assumir um procedimento que se baseia no reuso de sistemas.

Sommerville (2011) enfatiza que, a Engenharia de Software orientada a reuso possibilita reduzir o desenvolvimento do *software*, e consequentemente, os custos do projeto e os seus riscos, onde uma vez o componente a ser utilizado já foi validado, favorecendo uma entrega mais rápida do *software*.

Poulin (1994) expõe um modelo comum para reusabilidade, na qual os módulos reusáveis devem possuir: uma baixa complexidade, boa documentação (comentários no código e código intuitivo), poucas dependências externas (poucos parâmetros de entrada e saída) e confiabilidade comprovada (por meio de testes e validações).

Salientando a manutenibilidade e a flexibilidade de sistemas, uma abordagem com reuso de código pode ser usada para reduzir custos e tempo de desenvolvimento de *software*, e além da intensa manutenção relacionada ao suporte e a atualizações consideráveis do *software* (BROWN, 1996).

Umas das formas mais visíveis de reusabilidade são os *frameworks*. Um *framework* pode ser considerado como um esqueleto de componentes de *software*, oferecendo uma base para auxiliar o desenvolvimento de funcionalidades mais específicas para atender diretamente os requisitos de um sistema (MAAN, 2005). Assim, na sessão seguinte é apresentado o *framework* Angular.

2.2 *Framework* Angular

O Angular.js é uma estrutura feita em JavaScript de *open source* (código aberto), criado por Miško Hevery e Adam Abrons em 2009, com finalidade de promover uma alta produtividade de desenvolvimento, permitindo utilizar a sintaxe *HTML* (abreviação de HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto em português) para manipular elementos de forma eficiente. Mantido e desenvolvido pela equipe do Google, na qual Misko participa, juntamente com os inúmeros e severos contribuidores do código aberto. Com o AngularJS, o desenvolvimento de 17.000 linhas de código inicial pode

ser reduzido em 1.500 linhas de código, aumentando drasticamente a produtividade do desenvolvedor (BRANAS, 2014).

Em 2014, foi lançado o Angular 2, sendo amplamente reformulado e não compatível com a sua versão anterior. Foi escrito em cima do TypeScript, por ser um superconjunto de JavaScript, desenvolvido e mantido pela Microsoft, com a funcionalidade de acrescentar novos recursos ao JavaScript (BOOTH, 2017). Seshadri e Green (2014) acrescentam que a partir dessa ideia, os engenheiros se dedicaram a criar um *framework* simplificado para o desenvolvimento de aplicações Web. Dando início com o projeto Google Feedback, consistindo o primeiro a utilizar o framework *Angular* em sua versão 2, sendo a referência para estudos de funcionamento e uso de um framework JavaScript da visão de desenvolvedores.

Singh (2017) lista alguns benefícios para se utilizar o Angular em vez do Angular.js no desenvolvimento:

- O Angular é baseado em componentes e possui uma melhor detecção de alterações no projeto;
- Angular possui um melhor desempenho e conta com um sistema de *templates* mais completo;
- fornece APIs mais simples, carregamento por demanda (conhecido como *lazzy load*) e uma depuração mais fácil;
- Possui um maior suporte para testes; fornece componentes aninhados;
- Conta com compilação *Ahead of Time* (possui a sigla *AOT* – traduzida como compilação antecipada) que melhora a velocidade de carregamento (compila o código para JavaScript antes da execução/exibição da aplicação);
- O Angular.js é usado com um controlador e com escopo, mas o Angular é baseado em componentes.

Booth (2017) diz que o Angular 2 ou superior utiliza a vantagem do *Web Components*, que são elementos *HTML* encapsulados que o navegador conhece como exibir, e do *Shadow DOM*(*Document Object Model* - que significa traduzido ao português Modelo de Documento por Objetos), que pode ser definido como uma representação do código-fonte *HTML* como uma estrutura de árvore aninhada. O *Shadow DOM* é um objeto *DOM* encapsulado que pode ser criado a partir de qualquer elemento *DOM* existente, possibilitando assim, manipular os elementos dinamicamente a partir de um evento ou de um gatilho. Complementa ainda que o

Angular utiliza esses dois mecanismos para suportar o desenvolvimento orientado a componentes, dando a possibilidade de criar componentes reutilizáveis para as aplicações.

No entendimento de Nayrolles, Gunasundaram e Rao (2017) o principal foco do Angular é os dispositivos móveis, já que deve se considerar o desempenho de carregamento da aplicação. Com isso, existem inúmeros módulos separados do seu núcleo, mantendo apenas os módulos centrais, removendo os indesejados e aumentando a sua performance. Já para Gechev (2017), o Angular é uma estrutura para desenvolvimento de aplicativos: seu foco principal é fornecer uma base sólida para o desenvolvimento de interfaces dos usuários da aplicação. Seshadri e Green (2014) explicam que esse resultado se deve à utilização de módulos e a reusabilidade na aplicação, aplicando todas a vantagens, benefícios e conceitos apresentados na sessão 2.1.

2.3 Node.js

Os sistemas webs desenvolvidos em .NET, PHP, Java, Python possuem uma característica familiares: suspendem todo o processamento que está sendo realizado enquanto esperam por uma entrada/saída de dados no servidor, conhecido como Modelo Bloqueante (*Blocking-Thread*). A partir desse problema, foi desenvolvido o NodeJS em 2009, por Ryan Dahl com ajuda de 14 colaboradores, em busca de uma tecnologia inovadora com uma arquitetura não bloqueante (*non-blocking thread*) (PEREIRA, 2014).

Moraes (2018) complementa que o Node.js possui uma estrutura orientada a eventos e uma forma de I/O que faz que seja leve e eficiente. Essas características são essenciais para um melhor desempenho de intenso tráfego de rede e para aplicações em tempo real, que são os maiores obstáculos da web.

O Node.js é um ambiente em JavaScript, que fica do lado do servidor, sendo orientado a eventos. Ele executa o JavaScript usando o mecanismo V8 feito pelo Google para utilizar em seu navegador Google Chrome. A utilização desse mecanismo permite que forneça um ambiente de tempo de execução do lado do servidor que compila e executa o JavaScript em uma alta velocidade (NANDAA, 2018).

Para Rauch (2012), os benefícios são que os programadores podem utilizar a mesma linguagem de programação no lado do cliente e do servidor. Com isso, o JavaScript se tornou muito dinâmico, e o Node.js um sucesso instantâneo, por conta de sua simplicidade,

produtividade aprimorada de programação e alto desempenho, gerando uma grande comunidade de código aberto, apoiando empresas e seu próprio desenvolvimento.

3 PROCEDIMENTOS METODOLÓGICOS

Para o desenvolvimento deste trabalho, foi realizada a pesquisa bibliográfica exploratória experimental, composta por dados obtidos através de pesquisas em livros, teses, artigos e dissertações que abordam os temas associados ao assunto tratado (GIL, 2008). Além disso, com os conhecimentos teóricos obtidos durante a pesquisa, foi possível realizar um estudo de caso ao implementar um sistema que utiliza técnicas de reusabilidade, Angular e Node.js, para atender os requisitos da aplicação de maneira que se torne um desenvolvimento ágil e reutilizável, com facilidade de manutenibilidade.

4 ESTUDO DE CASO

O projeto desenvolvido foi um sistema web, nomeado Doutor Vida, para o Lar São Vicente de Paula, uma instituição de Longa Permanência para Idosos (ILPI) mantida pela Associação Senhor Bom Jesus, situada na cidade de Ibitinga, no interior do Estado de São Paulo. Trata-se de uma organização sem fins lucrativos, no qual todos os funcionários dão o máximo suporte para os residentes com o objetivo de suprir as necessidades de cada um. Além disso, promover interações, atividades e consultas para uma boa estadia do residente.

O problema enfrentado pela instituição era o fato do sistema de gerenciamento usado até então ser muito complexo, e com isso os funcionários da instituição deixaram de usar e voltaram a documentar a mão.

Como integrante do Grupo de Pesquisa em Engenharia de Software, da Fatec Taquaritinga, que possui foco em pesquisas de Interação Humano-Computador, englobando desde a avaliação de usabilidade até as metodologias de desenvolvimento de *software* e sua implementação, foi realizado a proposta sistêmica para o problema enfrentado pelo Lar São Vicente de Paula. O planejamento foi o desenvolvimento de outro sistema, no qual não deve existir muita complexidade e ter usabilidade para que não ocorra o mesmo problema enfrentado anteriormente. A equipe de desenvolvimento do sistema é responsável por fazer o

levantamento dos requisitos, documentar os requisitos, garantir que todo o sistema esteja funcionando de acordo com o planejado.

Para desenvolver esse sistema que entregue uma boa usabilidade e traga, consequentemente, uma experiência de usuário positiva, reduzindo o tempo de desenvolvimento, foi escolhido utilizar a tecnologia Angular com o superconjunto de JavaScript (TypeScript) e com o Node.js.

A escolha das tecnologias Angular e Node.js foi justamente pela baixa curva de aprendizagem oferecida pelo conjunto, visando que ambos compartilham a mesma linguagem, embora seja para diferentes bases de desenvolvimento (lado do cliente e lado servidor), sendo necessário aprender apenas uma linguagem de programação para *front-end* e *back-end*, além de possuírem um dos melhores desempenhos da aplicação e na produtividade no desenvolvimento. Os resultados de seu uso podem ser encontrados no próximo item desta sessão.

O desenvolvimento do *client-side* foi baseado no *framework* Angular, por ser focado na lógica da aplicação, proporcionando um maior resultado em entregas, possibilita o desenvolvimento baseado em módulos, feito por componentes, de modo que possa ser reutilizado o código durante a implementação do sistema, além de facilitar futuras manutenções e atualizações. Além disso, o Angular possui *Web Componentes*, que permite criar *tags* customizadas, possibilitando uma maior reutilização do código, a verbalização de aplicações e consequentemente uma maior organização da estrutura da aplicação como um todo.

4.1 Resultados obtidos

O Angular é um *framework* complexo e completo, sendo necessário ter um conhecimento básico sobre *NPM* (*Node Packet Manegement*), um gerenciador de pacotes do Node.js, que se resume em um repositório online para a publicação de projetos de código aberto. Além disso, é necessário ter conhecimento de JavaScript e TypeScript, na qual o Angular foi projetado e desenvolvido.

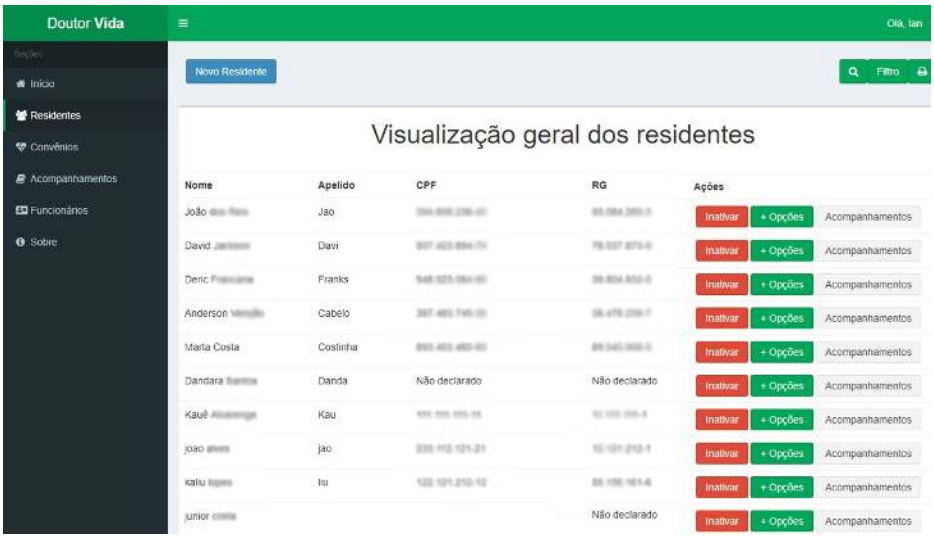
O uso do framework Angular traz alguns benefícios no desenvolvimento do projeto nos quais: diminui os bugs no desenvolvimento, já que passou por vários testes e está validado por milhares de desenvolvedores ao redor do mundo, possui uma documentação extensa e

completa, no qual auxiliou no desenvolvimento e no aprendizado; possui um guia de estilo amplo para a convenção e estrutura da aplicação e do código desenvolvido, que possibilitou uma maior legibilidade e uma uniformização de todo código desenvolvido pela equipe.

Com o desenvolvimento deste projeto em Angular foi possível aplicar a reusabilidade em grande parte de sua estrutura, desde a reutilização da interface do painel de controle em todas as telas que necessitam autenticação do usuário até o compartilhamento e reutilização dos módulos baseados em componentes, além da grande utilização de *Web Components*, que possibilita incorporar novas *tags* ao *HTML* em foco de reutilizar grande parte do projeto, sendo possível validar os conceitos abordados na sessão dois, no desenvolvimento da aplicação utilizando a reusabilidade do *framework* Angular.

Está exibida na Figura 1 a estrutura de interface compartilhada entre as telas do sistema.

Figura 1. Captura de tela da visualização geral dos residentes do sistema Doutor Vida.

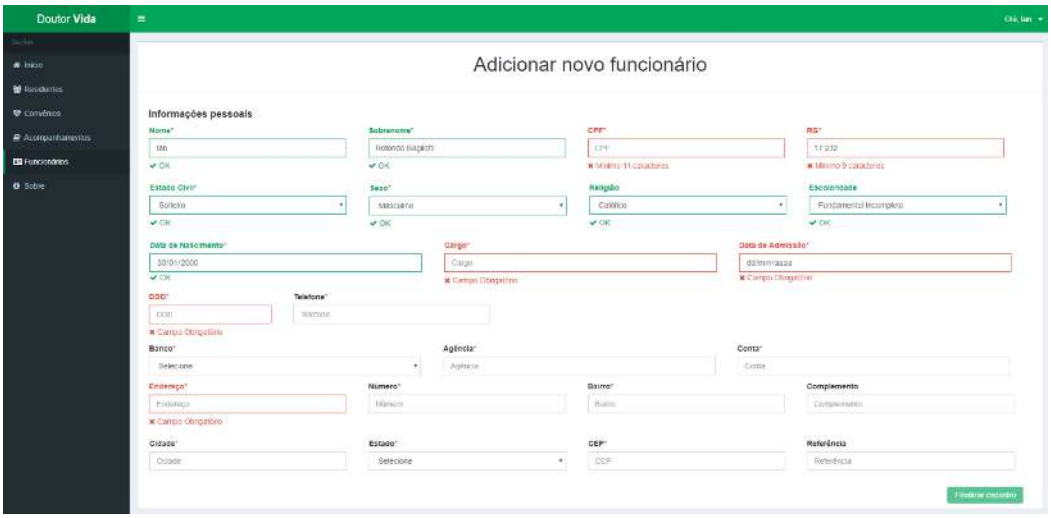


Fonte: Elaborado pelo autor.

Ainda com o avanço do sistema e a necessidade de replicar o código em vários componentes do Angular, foi possível reutilizar validações de formulários e formatações de campos, como pode ser observado exibido na Figura 2, os campos que são necessários serem validados compartilham um mesmo serviço que gerencia e aplica as mensagens na tela do usuário, não sendo necessário refazer toda a regra a cada campo que necessita de uma validação.

Este serviço de validação está sendo utilizado em toda aplicação que possui um formulário para ser submetido e possui campo(s) a ser(em) validado(s), sendo assim, possui o compartilhamento e o reuso de código de maneira global na aplicação, tornando muito mais enxuto os componentes.

Figura 2. Captura de tela de adicionar novo funcionário do sistema Doutor Vida.



Fonte: Elaborado pelo autor.

Com esse princípio de reutilizar os componentes, validações e interfaces, a quantidade de código desenvolvido foi reduzida bruscamente, incluindo o seu tempo de desenvolvimento, já que não foi necessário replicar linhas de código desnecessárias. Além disso, para efetuar manutenções no sistema, tanto como melhorias, o processo não será árduo, pois o desenvolvimento foi baseado em componentes, com foco em reusabilidade de código, e assim que necessário uma alteração, será feito em apenas um arquivo e será aplicado a todos os métodos que a utilizam.

Entretanto, quando se desenvolve um componente para ser reutilizado em outros módulos, requer um conhecimento mais abrangente do *framework* Angular e ser capaz de abstrair o negócio para trabalhar com formas mais genéricas, incluindo uma maior dedicação para transformar o componente reutilizável, de forma que consiga atender todos os requisitos necessários. Por isso, o desenvolvimento baseado em reuso, não é algo tão simples de se colocar em prática, por demandar um conhecimento mais abrangente e consolidado.

5 CONSIDERAÇÕES FINAIS

Com a implementação do sistema, foi possível atingir uma melhor usabilidade, consequentemente, facilitar o cotidiano dos funcionários que por lá trabalham, ressaltando que o sistema antigo parou de ser utilizado por ser muito complexo. Além disso, o sistema atendeu todas as necessidades para a instituição, dessa forma não tornando a experiência dos usuários negativa, possibilitando que haja um gerenciamento adequado dos residentes, excluindo as possibilidades de eventuais perdas de documentos e informações, tendo em vista que estavam mantendo os registros de maneira manual (em papeis), com dificuldades de gerenciar as informações e estando sujeito a perdas de registros.

Uns dos maiores desafios no desenvolvimento de sistemas é buscar otimizações em seu processo de criação, através disso, a reusabilidade no desenvolvimento com Angular pode ser indicada como uma das melhores escolhas, para aqueles que já possuem um breve conhecimento de *frameworks* similares ou tecnologias por *client-side* (lado do cliente). Ressalta-se, entretanto, que quando entendido o seu funcionamento e o desenvolvedor passar a ser detentor de conhecimento de forma parcial, a sua utilização no dia a dia se torna algo muito mais prático e eficiente, capaz de reduzir drasticamente o tempo de desenvolvimento de uma aplicação.

Desta forma, a reusabilidade no desenvolvimento com Angular se tornou imprescindível durante o desenvolvimento do projeto, pois a prática e a experiência trouxe uma excelente facilidade durante a codificação da aplicação, resultando em entregas mais eficazes, com uma fácil manutenção de todo código, visando uma única padronização do projeto utilizando o conceito do guia de estilo do Angular.

Um dos principais pontos observados de reuso no projeto foi a utilização de módulos por meio de componentes partilhados por *Web Components* em toda a aplicação, sendo desenvolvido apenas uma vez e atingindo a aplicação como um todo, resultando em uma entrega muito mais rápida, não sendo necessário desenvolver novamente ou replicar o código já desenvolvido, uma vez já validado. Além disso, a manutenção e correções realizadas no projeto se tornaram extremamente rápida, já que toda a estrutura segue um padrão único de boas práticas, definido pelo guia de estilo do Angular, sendo capaz qualquer desenvolvedor em Angular entender toda a estrutura do projeto e as funções desenvolvidas.

Com o conhecimento de funcionamento de *frameworks* tal como o citado neste estudo, pode-se aplicar em projetos, sejam pequenos ou gigantescos, para tornar o trabalho mais prático, eficiente e garantir os esforços na implementação dos requisitos oferecida pela base do *framework*, graças a reusabilidade. Além disso, permite reduzir consideravelmente o tempo de desenvolvimento de uma aplicação. É afinal, uma excelente escolha para programadores de todos os níveis de conhecimento.

REFERÊNCIAS

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

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

BRAUDE, Eric J. **Projeto de Software**: Da Programação À Arquitetura: Uma Abordagem baseada em Java. Porto Alegre, RS: Bookman, 2005.

BROWN, Alan W. **Component-Based Software Engineering**, Selected Papers from the Software Engineering Institute. IEEE Computer Society Press, 1996.

CHACON, Scott. **Pro Git**. 1. ed. Apress, 2009.

FERREIRA, H.N.M.; NAVES, T.F. **Reuso de software: suas vantagens, técnicas e práticas.** In: Anais do Encontro Anual de Computação de 2011 - ENACOMP 2011. Catalão, GO, Brasil. Disponível em: <https://www.enacomp.com.br/biblioteca-de-publicacoes/publication-details.php?id=51>. Acesso em 18 Mai.2020.

FRAKES, W; TERRY, C. **Software Reuse and Reusability Metrics and Models**. Blacksburg, VA: Virginia Polytechnic Institute & State University, 1995.

GECHEV, M. **Switching to Angular**: Align with Angular version 5 and Google's long-term vision for Angular. 3. ed. United Kingdom: Packt Publishing, 2017.

GIL, Antonio C. **Métodos e Técnicas de Pesquisa Social.** 6. ed. São Paulo: Editora Atlas S.A., 2008.

HIRAMA, K. **Engenharia de Software**: Qualidade e Produtividade com Tecnologia. 1. ed. Elsevier Brasil, 2012.

MANN, Kito D. **JavaServer Face in Action**. 1. Ed. Greenwich: Manning Publications

MORAES, Wiliam B. **Construindo aplicações com NodeJS**. 2. ed. São Paulo: Novatec Editora, 2018.

NANDAA, A. **Beginning API Development with Node.js**. 1. ed. United Kingdom: Packt Publishing, 2018.

NAYROLLES, M.; GUNASUNDARAM, R; RAO, S. **Expert Angular.** 1. ed. Packt Publishing, 2017.

PEREIRA, Caio R. **Node.js**: Aplicações web real-time com Node.js. 1. ed. Editora Casa do Código, 2014.

POULIN, Jeffrey S., **Measuring Software Reusability, Third International Conference on Software Reuse**, Rio de Janeiro, Brazil, 1-4 November 1994.

PRESSMAN, Roger S. **Engenharia de Software**: uma abordagem profissional. 7. ed. São Paulo: Pearson Makron Books, 2011.

RAUCH, Guillermo. **Smashing Node.js**: JavaScript Everywhere. 2 ed. Wiley, 2017.

REZENDE, Denis A. **Engenharia de Software e Sistemas de Informação.** 3. ed. Rio de Janeiro: Brasport, 2005.

SESHADRI, S.; GREEN, B. **Desenvolvendo com AngularJS.** Novatec Editora, São Paulo, SP, 2014.

SILVERMAN, Richard E. **Git**: guia prático. 1. ed. São Paulo: Novatec Editora, 2013.

SINGH, Anil. **Angular 2 Interview Questions and Answers**: With Typescript and Angular 4. 1. ed. Educreation Publishing, 2017.

SOMMERVILLE, Ian. **Engenharia de Software.** 8. ed. São Paulo: Pearson Addison Wesley, 2007.