INTERFACE TECNOLÓGICA

A UTILIZAÇÃO DO VUE.JS COM UMA API REST EM SPRING BOOT

THE USE OF VUE.JS WITH AN API REST IN SPRING BOOT

William Gabriel de Souza – william007.gabriel@gmail.com
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

Giuliano Scombatti Pinto – giuliano.pinto@fatectq.edu.br
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

DOI: 10.31510/infa.v17i2.904
Data de publicação: 18/12/2020

RESUMO

Aplicações do tipo *single page application* (SPA) são muito utilizadas na atualidade nos mais variados contextos para otimizar as interações com os usuários. Diante disso, o objetivo do presente artigo é apresentar os recursos necessários para o desenvolvimento de uma *single page application* para o gerenciamento de estoques. O *front-end* foi desenvolvido utilizando o *framework* progressivo de JavaScript Vue.js. O desenvolvimento do *back-end* foi realizado o *framework* de Java, Spring Boot Para desenvolver uma API Rest. A metodologia utilizada para o desenvolvimento do presente trabalho foi a pesquisa bibliográfica em livros, artigos, sites, e-books, dentre outros. Conclui-se que o Vue.js junto com o Spring boot possibilita uma aplicação de grande escalabilidade de recursos e desacoplamento entre componentes, além de uma organização mais limpa do código.

Palavras-chave: JavaScript. Vue.js. Java. Spring. Framework. Single Page Application.

ABSTRACT

Single page application (SPA) applications are widely used today in the most varied contexts to optimize interactions with users. Therefore, the objective of this article is to present the necessary resources for the development of a single page application for the management of inventories. The front-end was developed by using the progressive JavaScript framework Vue.js. The development of the back-end was carried out by using the Java framework, Spring Boot to develop a Rest API. The methodology used for the development of this work was bibliographic research in books, articles, websites, e-books, among others. It was concluded that Vue.js together with Spring boot allows for an application of great scalability of resources and decoupling among components, in addition to a cleaner organization of the code.

Keywords: JavaScript. Vue.js. Java. Spring. Framework. Single Page Application.

1 INTRODUÇÃO

A Internet é amplamente utilizada na atualidade para as mais variadas tarefas. O acesso à mesma está presente na maioria das residências brasileiras em grandes regiões: 76,7% no Sudeste, 74,7% no Centro-Oeste, 71,3% no Norte e 56,6% no Nordeste.O mundo conta com mais de 7,6 bilhões de pessoas e cerca de 55% utilizam a Internet, somando o total de 4,1 bilhões de usuários conectados (IBGE,2016). Diante disso, é importante considerar que ela é um ambiente com potencial para os mais variados nichos de mercado.

Na Internet, é muito comum a necessidade de aplicações para realizar o gerenciamento de diversas atividades, tais como um estoque. Um controle de estoque abrange atividades de planejamento, organização e controle do fluxo de matéria prima. Uma vantagem de um sistema *on-line* para gestão de estoque é que o *software* pode ser acessado remotamente em qualquer dispositivo que tenha conexão com a Internet, o que possibilita ao usuário acompanhar o fluxo de seu estoque de qualquer lugar (PDVEND,2018). Nesse sentido, uma *single page application* pode ser utilizada. Esse tipo de aplicação pode ser definido como um software composto por uma única página. De acordo com Oliveira (2017), a página atualiza a interface na medida em que os usuários interagem, sem a necessidade de recarregar todo conteúdo. As aplicações *single page application* foram ganhando espaço por prover uma experiência mais próxima de aplicações nativas.

Basicamente uma *single page application* é muito dependente da linguagem de programação JavaScript que faz toda "mágica" por trás. O JavaScript implementado na aplicação é inteligente o suficiente para renderizar telas dinamicamente e fazer chamadas AJAX, o que possibilita o uso de requisições assíncronas e atualizar as telas de acordo com a interação do usuário com a aplicação (AFONSO; FARIA, 2018, p. 21).

A interação do usuário com uma SPA gera uma experiência mais fluida, pois ele não espera muito tempo para o carregamento entre as telas da aplicação. Isso requer muito código implementado no lado do cliente em JavaScript. Felizmente, existem vários *frameworks* JavaScript para ajudar no desenvolvimento desse tipo de aplicação (AFONSO, A;FARIA, T.A, 2018, p. 22).

Sendo assim, o objetivo do presente trabalho é desenvolver uma *single page application* (SPA) para gerenciamento de estoques utilizando as linguagens JavaScript e Java. O projeto visa facilitar o acesso para visualizar e gerenciar os produtos em estoque.

Para o desenvolvimento da SPA, o Vue.js será utilizado. Conforme You (2014), o Vue.js é um *framework* progressivo, presente na linguagem JavaScript, para a construção de interfaces para usuário. Ao contrário de outros *frameworks* monolíticos, o Vue.js foi projetado desde sua concepção para ser adotável incrementalmente. Segundo You (2014), a biblioteca principal é focada exclusivamente na camada visual (*viewlayer*), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes. Por outro lado, Vue também é perfeitamente capaz de dar poder a sofisticadas *Single Page Applications* (SPA) quando usado em conjunto com ferramentas modernas e bibliotecas de apoio.

A escolha do *framework* progressivo Vue.js se deve ao fato dele ter uma curva de aprendizado muito rápida e ser de alta escalabilidade, além de ter uma fácil integração com APIs, na qual uma biblioteca HTTP, tal como o axios, pode ser utilizada.

A escolha do *framework* de Java, Spring Boot, se deve ao fato dos módulos serem de fácil integração. Há a possibilidade de utilizar JPA (Java *Persistence* API) para realizar mapeamentos dos objetos e consequentemente não é necessário utilizar comandos SQL para realizar interações com o banco de dados. O Spring Boot impulsiona o desenvolvimento de microsserviços ajudando nas configurações e também importando módulos e configurando automaticamente todas as dependências no pom.xml (AFONSO, A;FARIA, T.A,2018, p. 42).

O Spring é um *framework* para criação de aplicações Web, incluindo APIs REST. Ele ajuda no desenvolvimento de aplicações Web robustas, flexíveis e com uma clara separação de responsabilidades no tratamento da requisição. Através dele são vários recursos de grande ajuda nas tratativas uma requisição HTTP (AFONSO; FARIA, 2018, p. 39).

2 FUNDAMENTAÇÃO TEÓRICA

2.1 JavaScript

Segundo Dornelles (2018), atualmente, o JavaScript é a linguagem mais popular da Web por ser interpretado pelos navegadores. O JavaScript é uma linguagem orientada a objetos, baseada em eventos e ainda conta com diversos *frameworks* (Angular, React e Vue) e bibliotecas específicas para o uso.

O fato de existirem diversos *frameworks* auxiliares pode ser um problema também, pois fica muito difícil manter um foco em apenas uma ferramenta. Segundo Dornelles (2016), o JavaScript

mudou muito com ao decorrer dos dos anos. Hoje, ele trata-se de uma tecnologia que pode ser utilizada tanto no *client-side* de uma aplicação, quanto no *server-side.*

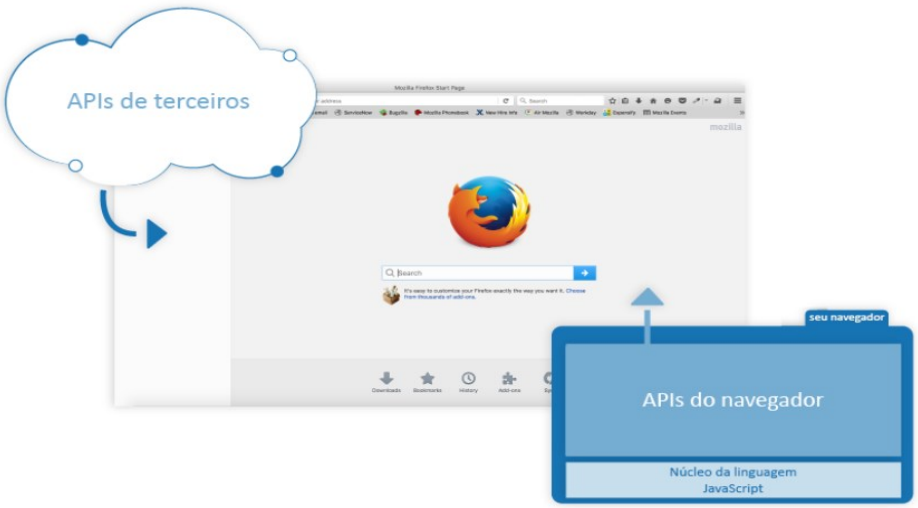
De acordo com Izhudson (2019), o JavaScript é uma linguagem de programação *front-end* que também pode ser usada no *back-end* com a plataforma Node.js. Com o JavaScript é possível manipular elementos da DOM (*Document Object Model*) em páginas Web.

Esta linguagem permite a criação de conteúdo que se atualiza dinamicamente, o controle de multimídias, imagens animadas e tudo o mais que há na DOM (*Document Model Object*).

O núcleo do JavaScript permite armazenar conteúdo em variáveis que podem ser globais ou de escopo, bem como constantes cujo valor é fixo, operações com pedaços de textos, executar códigos em respostas a determinados eventos.

De acordo com Izhudson (2019), as APIs são um conjunto de protocolos e definições constantemente usado no desenvolvimento e integração de aplicações e com isso as APIs dão muito poder para uma aplicação JavaScript elas geralmente se dividem em duas categorias.A Figura 1, apresentada a seguir, ilustra APIs de terceiros e APIs de navegadores.

Figura 1 – APIs de terceiros e APIs de navegadores.



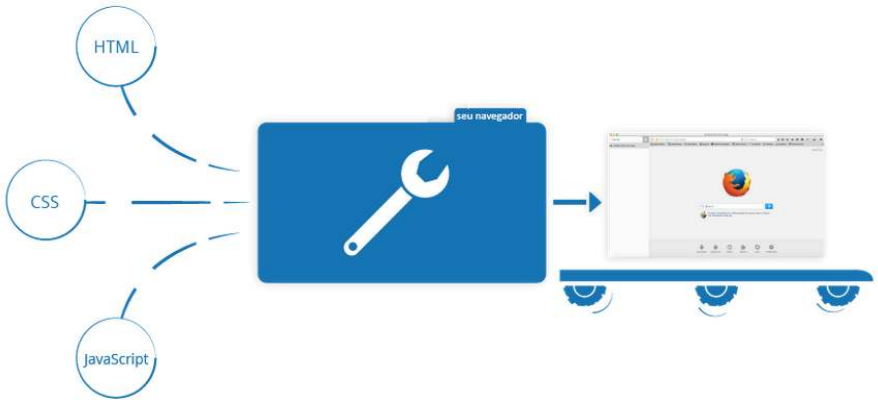
Fonte: Izhudson(2020).

Como é possível notar na Figura 1, cada API apresentada é implementada de um modo diferente. Segundo Izhudson (2019), nesse sentido, as APIs de navegadores já estão implementadas nos próprios *browsers* e são capazes de expor dados do ambiente do computador. Já as APIs de terceiros não são implementadas no navegador

automaticamente e, geralmente, o código e as informações necessitam ser encontradas em algum lugar da Web, tais como servidores.

É possível notar na Figura 2 apresentada a seguir, que quando um site é carregadono navegador, todo o código (HTML, CSS e JavaScript) é renderizado pelo mesmo. De acordo com Izhudson (2019), isso é como uma fábrica que pega a matéria prima (o código) e transforma em um produto (a página Web).

Figura 2 – Renderizando uma pagina Web no navegador.



Fonte: Izhudson(2020).

Como é possível notar na Figura 2, o JavaScript é executado pelo motor do navegador logo após o HTML e CSS forem traduzidos e colocados juntos em uma página Web, isso assegura que a estrutura da página já está carregada.

De acordo com Izhudson (2019), cada guia do navegador tem seu próprio espaço para executar código esses espaços são chamados de "ambientes de execução", isso significa que na maioria dos casos o código em cada guia está sendo executado separadamente, e o código em uma guia não pode afetar diretamente o código de outra guia ou de outro site. Isso é uma boa medida de segurança se esse não fosse o caso, então hackers poderiam começar a escrever código para roubar informações de outros sites, e fazer outras coisas más.

2.2 Vue.js

Segundo Incau (2017), o modo de desenvolver um *front-end* mudou muito ao longo dos últimos anos. Mais do que dar as ferramentas necessárias para o desenvolvedor, os

frameworks baseados em componentes reativos para Web vieram com padrões de reaproveitamento de código.

O Vue.js é um *framework* moderno de desenvolvimento *front-end,* baseado na linguagem JavaScript para componentes reativos. Ele ganhou muita visibilidade no mercado após ser adotado como padrão pelo Laravel**,** um *framework* PHP, tendo a possibilidade de ser aplicado em qualquer projeto que possua um *front-end* independente da linguagem de programação que esteja no *back-end* (INCAU,C;2017,p.10)

Neste artigo, o *framework* progressivo Vue.js será responsável pelo desenvolvimento do *front- end* e a comunicação com o *back-end* foi realizada com o uso de requisições HTTP em formato JSON *(JavaScript Object Notation)*, que é uma representação eficaz para a troca de dados ou informações entre sistemas diferentes, além de ser leve, é muito simples de ler.

2.3 Java

Segundo Pereira (2009), a linguagem Java é orientada a objetos e sua sintaxe se derivada do C++.Sua principal característica é que todo código escrito é em uma *class* e tudo é um objeto. A linguagem possui uma arquitetura neutra e portável, de forma que pode ser utilizada em diversos sistemas operacionais e ter uma alta performance. As aplicações Java podem ser executadas em qualquer plataforma que possua JVM(Java Virtual Machine). O Java utiliza "*Garbage Collector*" para gerenciar o ciclo de vida dos objetos. Ele é a base para todos os tipos de aplicações em rede, sendo assim ele é o padrão global para o desenvolvimento e distribuição de aplicações móveis e incorporadas.

2.4 Spring Framework

De acordo com Afonso e Faria (2018), o Spring *Framework* foi criado para que aplicações possam ter um maior foco em regras de negócios e menos na infraestrutura da aplicação, suas principais funcionalidades são o Spring MVC, suporte para JDBC e JPA e injeção de dependências.

> O Spring MVC é um *framework* para criação de aplicações Web, incluindo APIs RESTful. Ele ajuda no desenvolvimento de aplicações Web robustas, flexíveis e com uma clara separação de responsabilidades no tratamento da requisição.

> Através dele são disponibilizados vários recursos que são de grande ajuda na tratativa de uma requisição HTTP.(AFONSO, A;FARIA, T.A, 2018, p.39).

De acordo com Gentil (2012), o principal *core* do Spring pode ser implementado em qualquer aplicação Java. As principais funcionalidades são: injeção de dependências, programação orientada a aspectos (AOP) e cabe o desenvolvedor dizer para o Spring o que quer usar. O que faz do Spring uma ferramenta poderosa, pois não existe a necessidade de arrastar todas ferramentas do *framework* para uma simples aplicação.

2.5 Spring Boot

Segundo Afonso e Faria (2018) , o Spring Boo**t** trouxe agilidade, o que permite focar nas funcionalidades da aplicação com o mínimo de configuração, porque ele utiliza o conceito de *Convention over Configuration* (CoC). Basicamente o Spring já configura tudo, adotando uma visão opinativa sobre a criações Spring prontas para produção. Ele trabalha com base no Maven e adiciona as dependências no arquivo pom.xml o mesmo fica bastante extenso com várias linhas de configurações.

Segundo Souza (2018), Spring foi criado pela dificuldade que os desenvolvedores apresentavam ao criar determinados tipos de aplicações, precisamente as aplicações corporativas. Na época, a plataforma Java voltada para isso era o J2EE ainda era jovem, com ótimas ideias para a construção de aplicações leves, distribuídas e com um amplo leque de ferramentas, mas com algumas limitações. Essas limitações levaram a uma programação dependente de muitas interfaces e com muitas configurações. Ao final, era comum ter uma solução pesada e que trazia consigo muito mais do que o que realmente era necessário.E para completar, precisávamos utilizar servidores de aplicação pesados, o que tornava a programação e a depuração das aplicações ainda mais lento.

Com o Spring Boot é possível reduzir muito o tamanho do pom.xml com os *starters.* Os *starters* são dependências que agrupam outras dependências e adicionam uma entrada no pom.xml e todas dependências necessárias serão adicionadas ao *classpath.*

No presente trabalho, o Spring Boot será responsável pelo desenvolvimento de uma API RESTe rotas de comunicação entre o *back-end*e *front- end*.

2.6 SOA

Segundo a IBM (2017), SOA é um padrão de mercado definido que apresenta os processos de negócios de uma maneira voltada para serviços. O objetivo da arquitetura orientada a serviços (SOA) é separar a lógica da implementação para que o desenvolvedor possa focar na montagem de um aplicativo. Para alcançar o objetivo, todos componentes de serviços que foram implementados individualmente pelos processos de negócios são criados e o resultado é uma arquitetura de três camadas (lógica de integração de negócios, componentes de serviços e implementação).

Segundo Boaglio (2018), a arquitetura orientada a serviços (SOA), abrange uma solução para o REST com o objetivo de criar uma solução separada e independente para um problema. O ideal de uma aplicação é separar suas funcionalidades se se adequar de acordo com a demanda, com o conceito de microsserviços, cada funcionalidade é independente.

2.7 REST

De acordo com Afonso e Faria (2018), o padrão REST deixa aplicações mais simples, até mesmo pela organização que ele impõe. Essa simplicidade acaba dando uma segunda vantagem, que é adequar o software para evoluir de forma incremental, ou seja, evoluir sem impactos aos elementos que já existem. Como esse padrão preza por uma aplicação *back-end stateless*(sem estado), é possível escalar de maneira muito mais fácil o *software*.

O REST vem de um arquitetura moderna, que traz flexibilidade e desacoplamento. É colocado em prática por meio de uma aplicação *front-end* separada de outra aplicação *back-end*. O intuito é a formalização de um conjunto de melhores práticas se denominando de *constraints*, essas *constraints* tem como objetivo determinar a forma de padrões como HTTP e URL devem ser modeladas, aproveitando o fato que todas as características oferecidas pelo REST (AFONSO, A;FARIA, T.A,2018, p.26).

2.8 Axios

De acordo com Bernardes (2015) o axios é um cliente HTTP que funciona tanto no navegador quando no servidor. A biblioteca é uma API que interage com XMLHTTPRequest ou usando o padrão HTTP. Utilizando o axios no presente projeto para

que ocorra a comunicação entre o *front-end* e *back-end* utilizando o padrão HTTP com a biblioteca JavaScript Axios.

3 PROCEDIMENTOS METODOLÓGICOS

A metodologia utilizada no presente trabalho é a pesquisa bibliográfica, realizada em livros, artigos, sites, e-books, dentre outros. Baseado em pesquisas realizadas foi possível realizar o desenvolvimento uma API REST em Spring Boot e consumi-la utilizando o *framework* JavaScript Vue.js.

Na primeira etapa do trabalho, foi desenvolvida o *back-end* do projeto a API REST utilizando os devidos módulos do Spring: Spring Boot DevTolls para carregar atualizações feitas sem precisar para a execução do servidor, Spring Data JPA para mapear os objetos e abstrair a camada dao, MYSQL Driver para conexão da base de dados e Spring Web é o subprojeto do Spring Framework que se concentra em fornecer a infraestrutura para a criação e execução de aplicativos da Web avançados.

Na segunda etapa do trabalho , foi desenvolvido o *front-end* utilizando o *framework* Vue.js com algumas bibliotecas auxiliares para o desenvolvimento de componentes como o Vuetify e Bootstrap Vue, para conseguir se comunicar com o *back-end* a API foi necessário uma biblioteca HTTP como o Axios que foi utilizado.

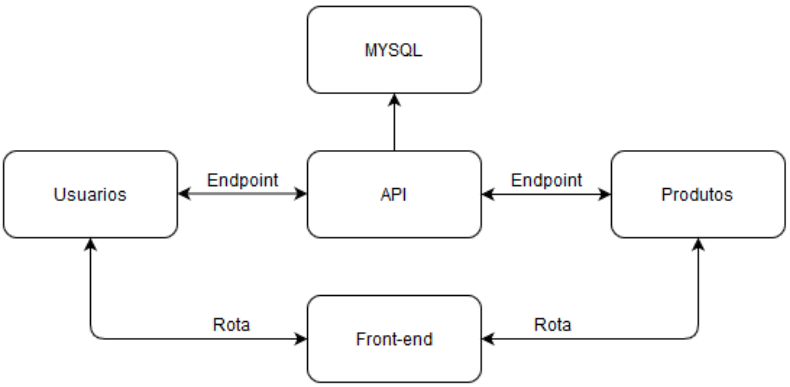
4 RESULTADOS E DISCUSSÃO

Com o intuito de exemplificar a arquitetura da aplicação, foi desenvolvido um serviço REST API responsável por efetuar cadastros de usuários e produtos, onde os usuários podem visualizar os produtos que obtêm em estoque. A API foi desenvolvida utilizando a linguagem Java e o *framework* Spring Boot já mencionado na fundamentação teórica, para base de dados foi utilizado MYSQL.

Para isso foram criados *endpoints disponibilizados* na API chamados de usuários e produtos. Para realizar a autenticação e cadastros de usuários é chamado o *endpoint* de usuários. Para o cadastro, alteração, listagem e exclusão de produtos é chamado o *endpoint* de produtos, que é chamado com a finalidade de gerenciar os produtos em estoques do usuário que realizou o *login*.

A Figura 3 representa a arquitetura desenvolvida para o presente projeto.

Figura 3 – Arquitetura desenvolvida.



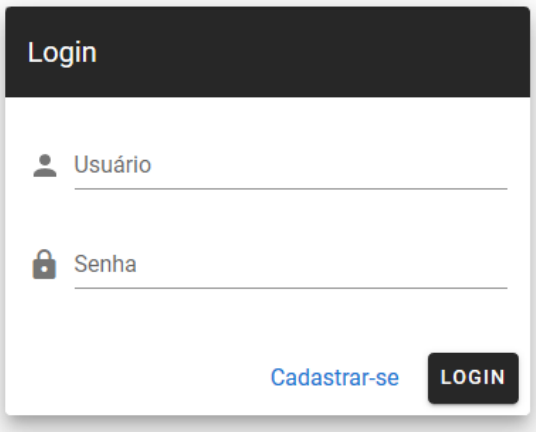
Fonte: Elaborada pelo autor (2020).

O codigo fonte do presente projeto desenvolvido está disponível no GitHub nos seguintes links:

- *Back-end*: https://github.com/Williams25/estoque-api
- *Front-end*: https://github.com/Williams25/estoque-ui

Na primeira tela da aplicação, é possível realizar o *login* ou se cadastrar, caso ainda não tenha um usuário cadastrado, informando os campos usuário e senha no formulário para realizar o *login.* O *front-end* utiliza de uma biblioteca HTTP o Axios que faz uma requisição HTTP utilizando o método POST para o *back-end* a API persiste os dados da requisição no banco de dados e retorna uma resposta podendo ser usuário autenticado ou não autenticado. A Figura 4 ilustra a tela de *login.*

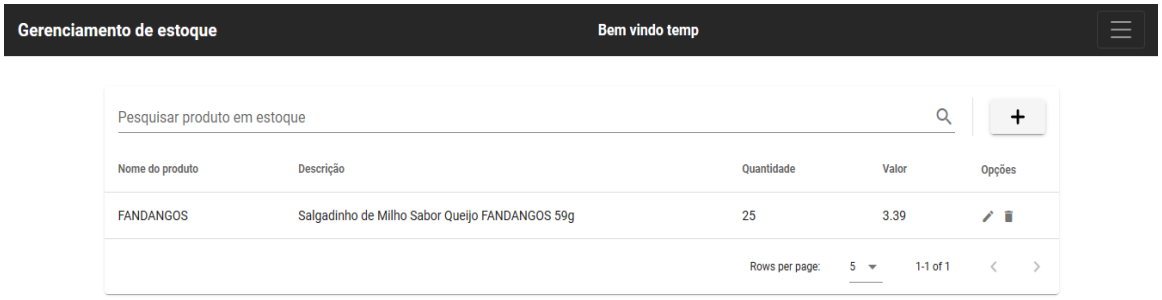
Figura 4 – Tela de *login.*



Fonte: Elaborada pelo autor (2020).

Após o usuário realizar o *login*, é apresentado a página de Gerenciamento de estoque é apresentada, que consta o CRUD(*create, update, read, delete*) para gerenciar os produtos em estoque. Também é apresentado um *component data-table* para a visualização das informações dos produtos com as opções de editar e apagar. A Figura 5 ilustra a página de Gerenciamento de estoque.

Figura 5 - Gerenciamento de estoque.



Fonte: Elaborada pelo autor (2020).

Todas as ações referentes a produto são realizadas pelo *endpoint* de produtos que é o responsável por gerenciar as informações, portanto a listagem de produtos e ações de exclusão, edição e inserção são realizadas por ele, onde é realizada uma requisição HTTP para o serviço de produtos que é submetidos as informações descritas no formulário, quando a requisição HTTP é bem sucedida o produto é persistido no banco de dados, caso ao contrário é retornado um erro pelo *endpoint* e o *front-end* trata a exceção e apresentando uma mensagem do erro ocorrido ao usuário.

5 CONCLUSÃO

Com base nos estudos realizados, foi possível concluir que o ecossistema Spring tem um grande potencial de escalabilidade e de fácil implementação. Com o Spring Boot, foi realizada a criação da API Rest e por meio do módulo JPA, foi possível realizar o mapeamento objeto-relacional, visto que o mesmo ajuda a ganhar produtividade, pois não é necessário criar as tabelas manualmente. Também foi possível identificar que, para atender as necessidades do projeto, a API de estoque deveria ser constituída de dois *endpoints* ,Usuários e Produtos, e ambos implementados utilizando o módulo JPA.

Por fim, constatou-se que o objetivo do trabalho foi bem atendido pelas linguagens

e tecnologias utilizadas. Elas ajudaram a implementação da API e aplicação *front-end* de estoque, que foi bem sucedida.

REFERÊNCIAS

AFONSO,A.**Funcionamento do Spring Boot**.Disponívelem<https://blog.algaworks.com/spring-boot/> Acesso em 8 de Out 2019.

AFONSO.A, FARIA.T.A. **E-Book Fullstack Angular e SpringGuia para se tornar um desenvolvedor moderno**. Ed.1. São Paulo: AlgaWorks 10 de Set 2018.

BERNARDES, M.**Como usar o Axios comoclienteHTTP**.Disponível em <http://codeheaven.io/how-to-use-axios-as-your-http-client-pt/ > Acesso 19 de Fev 2020.

BOAGLIO, F. **Spring Boot acelere o desenvolvimento de micro serviços.** Ed. 1. São Paulo: Casa do Código 2018.

DORNELLES. N. **JavaScript para iniciantes.**Disponível em<https://becode.com.br/javascript-para-iniciantes-origens-o-que-e-para-que-serve/> Acesso em 9 de Mar 2020.

GENTIL, Efraim. **Introdução ao Spring Framework**. Disponívelem<https://www.devmedia.com.br/introducao-ao-spring-framework/26212> Acesso em 3 Abr 2020.

IBM. **Arquitetura orientada a Serviços**.Disponível em<https://www.ibm.com/support/knowledgecenter/pt-br/SSV2LR/com.ibm.wbpm.wid.main.doc/prodoverview/topics/csoa.html> Acesso em 6 de Mar 2020.

INCAU, C. **Vue.js com suas aplicações incríveis**. Ed. 1. São Paulo: Casa do Código 20 de Abr 2017.

IBGE. **PNAD Contínua TIC 2016: 94,2% das pessoas que utilizaram a Internet o fizeram para trocar mensagens**. Disponível em <noticias/releases/20073-pnad-continua-tic- 2016-94-2-das-pessoas-que-utilizaram-a-internet-o-fizeram-para-trocar-mensagens> Acesso em 8 de Out 2019.

MDNWEBDOCS. **O que é JavaScript**. Disponível em< https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/O_que_e_JavaScript> Acesso em 20 de Mar 2020.

OLIVEIRA, D. De J. **Uma proposta de arquitetura paraSingle-Page Applications.** Disponivel em <https://www.cin.ufpe.br/~tg/2017-2/djo-tg.pdf> Acesso em 3 de Abr 2020.

PAULA, A, PEREIRA. **O que é Java**. Disponívelem<https://www.tecmundo.com.br/programacao/2710-o-que-e-java-.htm> Acesso em 20 de Mar 2020.

PDVEND. **Controle de estoque online: quais as vantagens e por que é importante?**. Disponível em<https://blog.pdvend.com.br/controle-de-estoque-online-quais-as-vantagens-e-por-que-e-importante/ > Acesso em 6 de Mar 2020.

SOUZA,M. **Como começar com Spring**.Disponível em<https://www.devmedia.com.br/exemplo/como-comecar-com-spring/73> Acesso em 6 de Mar2020.

YOU.E. **O que é Vue.js**. Disponível em < https://vuejs.org/v2/guide/> Acesso em 8 de Out 2019.