APLICAÇÃO DE UMA PLATAFORMA WEB DE SERVIÇOS PARA UM CONTEXTO REGIONAL
APPLICATION OF A WEB SERVICE PLATFORM FOR A REGIONAL CONTEXT

Thaiz Izidoro Antunes[1*]

RESUMO
O objetivo deste trabalho é apresentar os elementos que constituem na construção de uma aplicação web usando HTML, CSS, JavaScript, Node.js e MongoDB. A aplicação desenvolvida recebeu o nome de SOSCasa e tem como objetivo o compartilhamento de ofertas de trabalhos regionais e/ou mão de obra local de curto prazo, todos eles relacionados aos cuidados e manutenção de uma casa, que não envolvem vínculo empregatício entre o contratante e o contratado. A plataforma foi construída utilizando o modelo de arquitetura REST e conceitos de programação orientada a objetos. Foram utilizados *frameworks*, bibliotecas e *middlewares* entre eles Bootstrap, Express, Mongoose, Passport e JOI.
Palavras-chave: Aplicação Web. JavaScript. MongoDB. REST.

ABSTRACT
The objective of thispaperis to present the elements that allow the construction of a web application using HTML, CSS, JavaScript, Node.js and MongoDB. The application developed under the name SOSCasa and aims to share short-term regional work and/or labor, all related to the care and maintenance of a home, which does not involve an employment relationship between the employer and the contractor. The platform was built using the REST architecture model and object-oriented programming concepts. Frameworks, libraries and middlewares like Bootstrap, Express, Mongoose, Passport e JOI can be found throughout this paper.
Key words: Web Application. JavaScript. MongoDB. REST.

1 INTRODUÇÃO

Na perspectiva de Neto (2018), a demanda de serviços instantâneos como aplicações que utilizam a web vem aumentando rapidamente e as aplicações estão se tornando cada vez mais complexas devido às regras de negócios envolvidas na implementação dessas aplicações. Por este motivo, diferentes metodologias vêm sendo utilizadas no desenvolvimento de sistemas web.

As características que influenciaram esse crescimento são "[...] Simplicidade aos usuários; informações dinâmicas; facilidade de utilização independente do local que o usuário se encontra. Essas características resultam no crescimento do desenvolvimento de sistemas que utilizam este ambiente para execução, tornando esses sistemas cada vez mais complexos." (NETO, 2018, p. 1).

Para Zaupa, Gimenes, Cowan, Alencar e Lucena (2007), a qualidade das aplicações web não cresce ao mesmo ritmo que a quantidade destas e isto se dá pelo fato de que os métodos tradicionais de desenvolvimento ainda requerem muita modelagem, programação e não tiram muito proveito da reutilização. Neste trabalho o foco foi justamente a limpeza e reutilização de código utilizando as metodologias de desenvolvimento de software como o WIDE"[..] que consiste em um grupo articulado de frameworks de apoio ao desenvolvimento de aplicações Web."(ZAUPA et al, 2007, p. 90).

[1*]Estudante–Ciência e Tecnologia. Universidade Federal de Santa Catarina. E-mail: thaiz.tiz@gmail.com

Neto (2018) ressalta que as Metodologias de desenvolvimento de software buscam guiar os desenvolvedores nas suas atividades e facilitam a construção de sistema através de alguns aspectos:

a) Gerência da complexidade, visualizando de maneira simples o domínio do problema em partes gerenciáveis;
b) Comunicação entre as pessoas envolvidas no projeto, visando um aprofundamento no problema existente antes de seu desenvolvimento;
c) Redução dos custos de desenvolvimento, evitando alocar recursos sem uma prévia análise da aplicação;
d) Manutenção e documentação melhores, através de um desenvolvimento definido.

Este artigo apresenta os passos necessários no desenvolvimento de uma aplicação web de serviços, em particular a metodologia e principalmente as linguagens necessárias para planejar e desenvolver esta aplicação. Este processo consiste em atividades para (i) definir o conjunto de *stacks* a ser utilizado; (ii) definir o escopo e mapear as necessidades da aplicação;(iii) implementar um CRUD (Create, Read, Update, Delete) básico utilizando os conceitos de arquitetura REST; (iv) modelar o banco de dados de acordo com os dados a serem armazenados; (v)implementar os serviços mapeados; (vi) gerar a aplicação com base nos serviços definidos.

A aplicação que foi nomeada como SOSCasa utiliza um banco de dados não relacional e EJS (Embedded JavaScript)para gerar HTML usando JavaScript puro além de utilizar JavaScript no *back-end* da aplicação para especificação das regras de negócio, criação das APIs e conexão com o banco de dados. O desenvolvimento da aplicação foi realizado por meio de um *template* Express básico. Os resultados obtidos mostram uma aplicação rápida, funcional e "[...] vantagens como: reutilização, separação explícita da lógica do negócio e dos serviços, redução de tempo e custo de desenvolvimento, interoperabilidade e manutenibilidade." (ZAUPA et al, 2007, p. 90).

Este artigo está organizado da seguinte forma. A Seção 2 apresenta o conjunto de *stacks* ou conjunto de ferramentas/linguagens de programação utilizado detalhadamente. A Seção 3 apresenta o funcionamento básico da aplicação web desenvolvida. A Seção 4, o processo de concepção, desenvolvimento e metodologia aplicada para construção da aplicação. A Seção 5 apresenta as considerações finais e conclusões.

A seguir será detalhado cada um dos elementos que compõem a estrutura de desenvolvimento de uma aplicação web.

2 **CONJUNTO DE STACK**

O primeiro passo para iniciar a construção de uma aplicação web é geralmente a decisão do conjunto de *stacks* ou conjuntos de soluções. Segundo Thomas (2020) "[...]uma pilha de tecnologia é um ecossistema de dados que relaciona as ferramentas, frameworks e bibliotecas subjacentes usadas para construir e executar os aplicativos."

2.1 HTML

O HyperText Markup Language (HTML) é na verdade uma linguagem de marcação ou "*markup language*" em inglês e não uma linguagem de programação. Foi desenvolvido pelo inglês Tim Berners-Lee no início da década de 90 e seu protótipo foi pensado enquanto

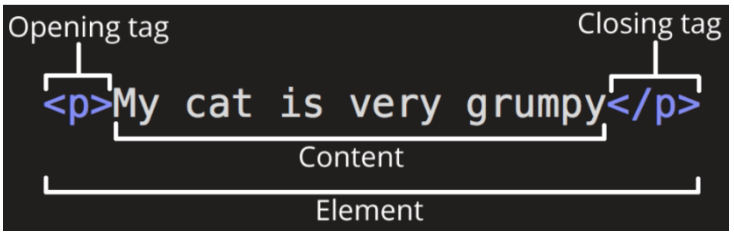
Lee ainda trabalhava para a “Conseil Européen pour la Recherche Nucléaire”(CERN) como uma ferramenta para cientistas e pesquisadores usarem e compartilharem documentos entre si. Berners-Lee terminou de desenvolver tanto o *browser* quanto o software do servidor para a nova linguagem no final do ano de 1990 junto das especificações do HTML, entretanto apesar de todo o esforço o protótipo nunca foi formalmente adotado pela CERN.

2.1.1 Como funciona

HTML é a linguagem que os navegadores usam para interpretar textos e imagens em páginas web visuais ou audíveis. A linguagem de marcação define a estrutura do conteúdo mostrado na tela do usuário em que se utilizam uma série de elementos para incluir ou modificar partes do conteúdo e fazê-lo aparecer de uma determinada forma ou até mudar a forma em que o conteúdo aparece. Para isso ser possível a linguagem utiliza as “*enclosing tags*” ou “tags de marcação”. Essas tags são as responsáveis por transformar o simples texto em algo mais legível e visualmente mais atraente. Elas têm a capacidade de transformar uma imagem ou texto em um “hyperlink” e transportar o usuário para uma página diferente, também podem modificar o texto para itálico, negrito, fazer a fonte maior ou menor entre muitas outras possibilidades.

Para uma melhor visualização de como o HTML funciona, um exemplo pode ser observado na figura 1.

Figura 1 – Estrutura de um parágrafo em HTML



Fonte: (MDN Web Docs, 2021, p. 1)

Para especificar que a frase é um parágrafo foi incluído a tag de parágrafo “<p>”. Desta forma é possível incluir inúmeras tags para obter o resultado final desejado.
Segundo a documentação MDN (2021), as principais partes do elemento são as seguintes:

1. Opening tag (tag de abertura): a tag de abertura consiste no nome do elemento, o que neste caso é o p entre os “angle brackets” ou (colchetes angulares em português) e indica onde o elemento começa a ter efeito - neste caso, onde o parágrafo começa.
2. Closing tag (tag de fechamento): é o mesmo que a tag de abertura, exceto que esse inclui a barra antes do nome do elemento. Indica onde o elemento termina - neste caso onde o parágrafo termina.
3. Content (conteúdo): é onde se encontra o conteúdo do elemento, que neste caso, é apenas texto.
4. Element (elemento): A tag de abertura, tag de fechamento e o conteúdo junto formam o elemento.

Elementos também podem ter atributos. Estes atributos possuem informação extra sobre o elemento que não serão visíveis junto do conteúdo. Existem vários tipos de atributos que são utilizados para diferentes designações, mas no geral seguem o padrão da figura 2, com algumas exceções. Primeiro temos o nome do atributo que neste caso é “*class*” seguido do sinal de igual e então entre aspas duplas temos o *value* ou valor o que neste caso é “editor-note”. Se o *value* contiver espaços dentro das aspas duplas isso significará que o

atributo possui mais de um *value* o que neste caso significaria que o atributo possui duas classes distintas.

Figura 2–Atributos em HTML

Fonte: (MDN Web Docs, 2021, p. 1)

Os atributos mais comuns são *class, id, href, type, value, src*, entre muitos outros.

2.1.2 Anatomia do HTML

A anatomia de um documento html pode ser visualizada na figura 3 e conta com os seguintes campos:

- <! DOCTYPE html> - doctype. É uma tag obrigatória. Atualmente, eles são necessários apenas para garantir que o documento se comporte corretamente.
- <html></html> - Este elemento envolve todo o conteúdo da página e é conhecido como o elemento raiz.
- <head></head> - Este elemento atua como um contêiner para todas as coisas que se deseja incluir na página HTML que não seja o conteúdo que será mostrado aos visualizadores da página. Isso inclui palavras-chave ou uma descrição de página que você deseja que apareça nos resultados de pesquisa, CSS para definir o estilo entre outros.
- <meta charset = "utf-8"> - Este elemento define o conjunto de caracteres que o documento deve usar para UTF-8, que inclui a maioria das línguas escritas.
- <title></title> - Isso define o título da sua página, que é o título que aparece na guia do navegador em que a página é carregada. Também é usado para descrever a página quando você a marca / adiciona aos favoritos.
- <body></body> - Ele contém todo o conteúdo que será mostrado aos usuários quando eles visitam uma página, seja texto, imagens, vídeos, jogos, faixas de áudio reproduzíveis ou qualquer outra coisa.

Figura 3 – Anatomia de um documento HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

Fonte: (MDN Web Docs, 2021, p. 1)

2.1.3 HTML 5

A versão de HTML utilizada para desenvolvimento do protótipo foi a HTML5. Essa é a versão mais recente da linguagem e representa a maior quebra comparada com versões anteriores. O motivo das mudanças foi principalmente para padronizar as muitas maneiras em que desenvolvedores estão utilizando a linguagem, bem como para encorajar um único conjunto de melhores práticas em relação ao desenvolvimento web, segundo Adam Wood (2015).

2.2 CSS

CSS que significa Cascading Style Sheets, é uma linguagem que descreve de que maneira documentos escritos em uma linguagem de marcação como HTML serão apresentados visualmente na página web - descreve a forma que serão posicionados e estilizados. Este também não recebe o título de linguagem de programação, mas sim uma linguagem de *stylesheet*, que na tradução literal para o português significa folha de estilo, é um dos alicerces do desenvolvimento web descrito pela World Wide Web junto ao HTML e o JavaScript.

Segundo o W3C (2016) o CSS foi projetado para criar uma separação entre o conteúdo e estilos incluindo layout, cores e fontes. Esta separação é útil para melhorar a acessibilidade e legibilidade do arquivo HTML (ou outro arquivo de marcação). Além disso, fornece mais flexibilidade e controle na especificação das características da apresentação, permite que várias páginas compartilhem a mesma formatação especificando o CSS relevante em um arquivo .css separado, reduzindo a complexidade e a repetição no conteúdo estrutural e possibilitando o arquivo .css ser armazenado em cache para melhorar a velocidade de carregamento entre as páginas que compartilham o arquivo e sua formatação.

2.2.1 Como funciona

É possível utilizar o CSS dentro do próprio documento HTML com uma tag de estilo (<style></style>), ou como a documentação da MDN (2021) sugere, criar uma tag com o endereço do documento .css dentro das tags <head></head> - estas que já foram mencionadas anteriormente na seção 2.1.3 deste documento. Na figura 4 é possível ver um exemplo de como a tag deverá ser criada.

Figura 4. Tag de estilo em CSS

```
<link href="styles/style.css" rel="stylesheet">
```

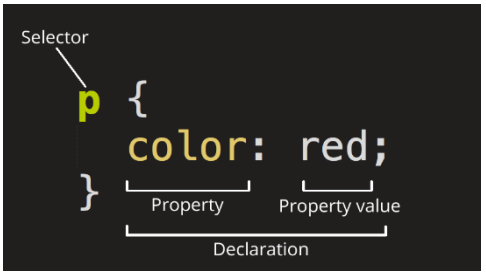
Fonte: (MDN Web Docs, 2021, p. 1)

CSS é o utilizado para estilizar elementos HTML seletivamente. Por exemplo, na figura 5 o CSS seleciona o texto do parágrafo, definindo a cor de todos os parágrafos dentro do HTML para vermelho. Toda a estrutura é chamada de conjunto de regras.
A documentação da MDN Web Docs (2021) define os nomes das partes individuais como:

- Selector (Seletor): Este é o nome do elemento HTML no início do conjunto de regras. Ele define o (s) elemento (s) a serem estilizados (neste exemplo, elementos <p>). Para estilizar um elemento diferente, altere o seletor.

- Declaration (Declaração): Esta é uma regra única, como color: red;. Ele especifica quais propriedades do elemento você deseja estilizar.
- Property (Propriedades): Essas são maneiras de definir o estilo de um elemento HTML. (Neste exemplo, a cor é uma propriedade dos elementos <p>.) Em CSS, pode-se escolher quais propriedades deseja alterar. Caso a propriedade não seja selecionada ela não será alterada e seguirá o padrão definido pelo próprio HTML ou herdará a propriedade de algum outro seletor.
- Property Value (Valor da propriedade): Escolhe uma das muitas aparências possíveis para uma determinada propriedade. (Por exemplo, existem muitos valores de cores além do vermelho.)

Figura 5. Estrutura do conjunto de regras CSS



Fonte: (MDN Web Docs, 2021, p. 1)

Utilizando seletores é possível alterar basicamente todo o estilo e posição do HTML. Uma tabela com os seletores pode ser vista na tabela 1.

Tabela 1 – CSS Seletores

Nome do seletor	O que ele seleciona	Exemplo
Seletor de elemento (às vezes, chamado tag ou seletor de tipo)	Todos os elementos HTML de determinado tipo.	`p` Seleciona `<p>`
Seletor de ID	O elemento na página com o ID específicado. Em uma determinada página HTML, é uma boa prática usar um elemento por ID (e claro, um ID por elemento) mesmo que seja permitido usar o mesmo ID para vários elementos.	`#my-id` Seleciona `<p id="my-id">` ou `<a id="my-id">`
Seletor de classe	O(s) elemento(s) na página com a classe específicada (várias instâncias de classe podem aparecer em uma página).	`.my-class` Seleciona `<p class="my-class">` e `<a class="my-class">`
Seletor de atributo	O(s) elemento(s) na página com o atributo especificado.	`img[src]` Seleciona `<img src="myimage.png">` mas não `<img>`
Seletor de pseudo-classe	O(s) elemento(s) específicado(s), mas somente quando estiver no estado especificado. Ex.: com o mouse sobre ele.	`a:hover` Seleciona `<a>`, mas somente quando o mouse está em cima do link.

Fonte: (MDN Web Docs, 2021, p. 1)

2.2.2 **CSS 3**

A versão utilizada no desenvolvimento do projeto foi o CSS 3. Este que por sua vez não é a versão mais atual do CSS que possui uma versão CSS 4, porém é a versão recomendada pela W3C e a versão padrão na maioria dos browsers.

Diferente da versão anterior o CSS3 é dividido em vários documentos separados chamados de módulos. Cada módulo adiciona novos recursos ou estende recursos definidos no CSS2, preservando a compatibilidade com versões anteriores.

Devido à modularização, diferentes módulos têm diferentes estabilidades e status. Estes módulos podem ser encontrados em "*CSS current work & how to participate*" no site da W3C.

2.3 JAVASCRIPT

JavaScript ou apenas JS é uma linguagem de programação de alto nível, orientada a objeto, geralmente compilada *just-in-time* pelo próprio navegador e junto ao HTML e CSS é uma linguagem essencial da World Wide Web. Segundo o Web Technology Surveys W³Techs (2021), JS é utilizado no *client-side* para manipular o comportamento da página por mais de 97% dos websites. Apesar de ser mais conhecido por ser uma linguagem de scripting para páginas web, muitos ambientes não web como Node.js, Apache CouchDB e Adobe Acrobat também usam JavaScript.

Como uma linguagem multiparadigma, o JavaScript oferece suporte a estilos de programação orientados a eventos, funcionais e imperativos. Possui interfaces de programação de aplicativos (APIs) para trabalhar com texto, datas, expressões regulares, estruturas de dados padrão e o Document Object Model (DOM).

Como padrão, JavaScript utiliza o padrão ECMAScript ou ES que é um padrão JavaScript, padronizado pela Ecma International, destinado a garantir a interoperabilidade das páginas web em diferentes navegadores. Desde 2012 todos os navegadores modernos possuem suporte total ao ECMAScript e em julho de 2015 a ECMA International publicou a sexta e mais atual versão do ES, o ECMAScript 2015 ou ES6 como é conhecido.

2.3.1 Como é utilizado

JavaScript é uma linguagem extremamente versátil e apesar de ser amplamente utilizada no *client-side* das páginas web, com a ajuda de algumas outras linguagens é possível utilizar o JS no *server-side* das aplicações e também em outras aplicações não web.

2.3.1.1 Client-side ou Front-end

No *cliente-side* das aplicações web, segundo o site W3Tech (2021), mais de 80% das aplicações desenvolvidas em JS utilizam frameworks e bibliotecas de terceiros. Entre essas, a biblioteca mais utilizada é a jQuery, utilizada em mais de 75% dos websites. Outras bibliotecas comuns são o React, criado pelo Facebook,e Angular, criado pelo Google. Ambas as bibliotecas são *open source* e vem se popularizando entre os programadores.

É possível utilizar JS no *cliente-side* sem nenhum framework ou biblioteca adicional uma vez que a linguagem é interpretada diretamente pelo navegador. O uso do JavaScript puro é conhecido pelos programadores como "Vanilla JS".

2.3.1.2 Server-side ou Back-end

Server side JavaScript (SSJS) é uma extensão do JavaScript que habilita o acesso do back-end às bases de dados, sistemas de arquivos e servidores. O JavaScript do lado do servidor é um código JavaScript rodando sobre os recursos locais de um servidor, é como C # ou Java, mas a sintaxe é baseada em JavaScript. Um bom exemplo disso é o Node.JS que é apresentado mais detalhadamente adiante.

2.3.2 **Sintaxe e tipos do JavaScript**

JavaScript tem grande parte de sua sintaxe similar ao Java, mas também é influenciado por Awk, Perl e Python. A linguagem é *case-sensitive*, o que significa que uma palavra escrita com letra maiúscula e minúscula será interpretada de forma diferente. As instruções são chamadas de declaração e são separadas por ponto e vírgula (;). A sintaxe dos comentários em JavaScript é semelhante ao C++ e outras linguagens.

Segundo a MDN (2021), existem 3 tipos de declarações:

1. var: Declara uma variável, opcionalmente, inicializando-a com um valor.
2. let: Declara uma variável local de escopo do bloco, opcionalmente, inicializando-a com um valor.
3. const: Declara uma constante de escopo de bloco, apenas de leitura.

E 7 tipos de dados:

1. Boolean: true e false.
2. Null: Uma palavra-chave que indica valor nulo.
3. Undefined: Uma propriedade superior cujo valor é indefinido.
4. Number: 42 (número inteiro) ou 3.14159 (número decimal).
5. String: "Howdy"– podem ser letras, números ou símbolos que serão entendidos pelo complicador da mesma forma.
6. Symbol: Um tipo de dado cujas instâncias são únicas e imutáveis.
7. Object: Como JavaScript é uma linguagem dinâmica é possível atribuir um tipo de dado para a variável e depois convertê-la para outro tipo apenas atribuindo um tipo diferente.

2.3.4 **JavaScript ES6**

A versão utilizada do JavaScript no desenvolvimento da aplicação foi sua versão mais atual da linguagem,o ES6. Lançada em 2015, essa versão do JS trouxe grandes mudanças e segundo Fawcett (2020) as principais mudanças foram a inclusão das declarações const e let, a inclusão das funções “flecha” ou *Arrow functions,* novo modelo de herança, inclusão de *template literals* entre muitas outras.

2.4NODE.JS

Node.js é um ambiente de execução JavaScript *open-source*, multiplataforma que executa código JavaScript fora de um navegador. Node permite com que desenvolvedores usem JavaScript para todo tipo de aplicativos e ferramentas no *server-side* (back-end). O ambiente de execução omite APIs JavaScript específicas do navegador e adiciona suporte para APIs de sistema operacional mais tradicionais, incluindo bibliotecas de sistemas HTTP e arquivos. Node.js representa um paradigma "JavaScript em todos os lugares", unificando o desenvolvimento de aplicativos da web em torno de uma única linguagem de programação, em vez de diferentes linguagens para scripts do lado do servidor e do lado do cliente.

Nativamente, Node.js apenas suporta a linguagem JavaScript, entretanto muitas linguagens compiladas em JS como TypeScript, CoffeeScript, Dart e ClojureScript são compatíveis com node.

Node.js surgiu em novembro de 2009 apresentado pela primeira vez na conferência Internacional *European JSConf* pelo Americano Ryan Dahl. O engenheiro de software criticava as possibilidades limitadas do então mais popular servidor web para JavaScript, o Apache HTTP Server, em lidar com mais de 10.000 conexões. O Node.js combinava o mecanismo V8 JavaScript do Google, um loop de eventos e uma API de Entrada/Saída de baixo nível.

Em 2010 o gerenciador de pacotes NPM foi incluído no Node.js. O NPM fez com que a publicação e compartilhamentos de pacotes fosse mais fácil e ainda simplificava a instalação, atualização e desinstalação destes pacotes. Hoje NPM hospeda mais de 1.000.000 de pacotes *open source* que podem ser utilizados sem custo.

Inicialmente a plataforma era suportada apenas por Linux e Mac OS X. Entretanto, segundo Dawl (2011), em 2011 a Microsoft e Joyent implementaram uma versão do Node para Windows.

2.4.1 Como funciona

Node.js é principalmente utilizado para construir programas de rede, como servidores web. Ao contrário do PHP, por exemplo, Node.js usa uma estrutura assíncrona o que significa que seu código é executado simultaneamente ou mesmo em paralelo e *callbacks* são usados para sinalizar conclusão ou falha de execução.

Segundo a documentação do Node.js (2021) "[...] Um aplicativo Node.js é executado em um único processo, sem criar um novo thread para cada solicitação. O Node.js fornece um conjunto de primitivas de E/S assíncronas em sua biblioteca padrão que evita o bloqueio do código JavaScript e, geralmente, as bibliotecas em Node.js são escritas usando paradigmas sem bloqueio, tornando o comportamento de bloqueio uma exceção em vez da regra."

Sendo assim, Node.js é capaz de lidar com milhares de conexões simultaneamente com um único servidor sem ter que gerenciar simultaneamente as threads, que podem ser uma fonte significativa de erros.

Figura 7: Servidor Node.js

```
// Carrega o modulo HTTP do Node
var http = require("http");

// Cria um servidor HTTP e uma escuta de requisições para a porta 8000
http.createServer(function(request, response) {

  // Configura o cabeçalho da resposta com um status HTTP e um Tipo de Conteúdo
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Manda o corpo da resposta "Olá Mundo"
   response.end('Olá Mundo\n');
}).listen(8000, '127.0.0.1');

// Imprime no console a URL de acesso ao servidor
console.log('Servidor executando em http://127.0.0.1:8000/');
```

Fonte: (MDN Web Docs, 2021, p. 1)

Existem diferentes argumentos para utilizar o Node.js ou não, entretanto a possibilidade de introduzir um *server-side* ou *back-end* sem precisar introduzir uma nova linguagem de programação torna cômodo o uso do Node.js na construção de aplicações.

2.4.2 **Node.js v14**

A versão utilizada do Node.js foi a v14 que segundo a documentação do Node encontrada no website da empresa (2021) “[...] O status de lançamento do LTS é "suporte de longo prazo", o que normalmente garante que bugs críticos sejam corrigidos por um total de 30 meses. Os aplicativos de produção devem usar apenas versões *Active* LTS ou *Maintenance* LTS.”

2.5 MONGODB

MongoDB ou apenas Mongo é um de banco de dados orientado a documento, de código aberto e noSQL ou não relacional, construído em uma arquitetura escalável que armazena dados estruturados em Java Script Object Notation (JSON) como documentos com esquemas dinâmicos. Isso permite um desenvolvimento rápido, pois os esquemas do banco de dados podem ser modificados rapidamente à medida que os aplicativos evoluem.

O MongoDB é popularmente usado em colaboração com AWS, Azure e outras fontes de dados para desenvolvimento e funcionamento de aplicativos. Por permitir o armazenamento e a consulta de grandes volumes de dados, MongoDB oferece melhores execuções de consulta com indexação adequada e recursos de processamento, análise em tempo real e manipulação de dados otimizada com o uso de consultas ad-hoc, maior disponibilidade e flexibilidade de dados com recursos de replicação robustos. O compartilhamento de dados permite a divisão de grandes blocos de dados para um processo de execução de consulta distribuído e mais rápido.

2.5.1 Como funciona

MongoDB oferece tanto a versão Community gratuita e livre para uso com algumas limitações e a versão Enterprise que possui alguns recursos extras e precisa ser paga.

Em MongoDB, dados são armazenados como documentos que contêm uma estrutura de dados composta por campo (*field*) e valor (*value*). Os valores podem incluir outros documentos, matrizes e matrizes de documentos. Estes documentos são semelhantes aos objetos JSON como mostrado na figura 8.

Figura 8 –Estrutura de dados

Fonte: (MongoDB, 2021, p.1)

Segundo o MongoDB (2021), as vantagens de usar documentos são:

1. Documentos (ou seja, objetos) correspondem a tipos de dados nativos em muitas linguagens de programação.
2. Documentos e matrizes incorporados reduzem a necessidade de junções caras.
3. O esquema dinâmico suporta polimorfismo fluente.

MongoDB guarda esses documentos em coleções ou *collections em* inglês. Essas coleções são análogas as tabelas dos bancos de dados relacionais. Além disso, Mongo possui suporte para *views* de leitura e *views* materializadas.

Alguns dos principais recursos do MongoDB são a persistência de dados de alta performance, suporte para várias linguagens, replica set (grupo de servidores que mantém o mesmo conjunto de dados, fornecendo redundância e aumentando a disponibilidade dos dados), escalabilidade horizontal e suporte para vários mecanismos de armazenamento como *In-memory Storage Engine* e ferramentas API de armazenamento conectáveis que permite que terceiros desenvolvam mecanismos de armazenamento para MongoDB.

2.6 FRAMEWORKS E BIBLIOTECAS

2.6.1 Frameworks

Clark (2020) diz que o *Framework* é uma abstração na qual o software pode ser alterado por código adicional escrito pelo usuário fornecendo uma maneira padrão de construir e implementar aplicativos. Visam facilitar o desenvolvimento de software permitindo que desenvolvedores dediquem seu tempo para atender requisitos de software reduzindo o tempo geral de desenvolvimento. Entretanto, isso costuma trazer um custo adicional de processamento o que tende a deixar os programas maiores e em alguns casos mais lentos.

As estruturas de software podem incluir programas de suporte, compiladores, bibliotecas de código, conjuntos de ferramentas e interfaces de programação de aplicativos (APIs) que reúnem todos os diferentes componentes para permitir o desenvolvimento de um projeto ou sistema.

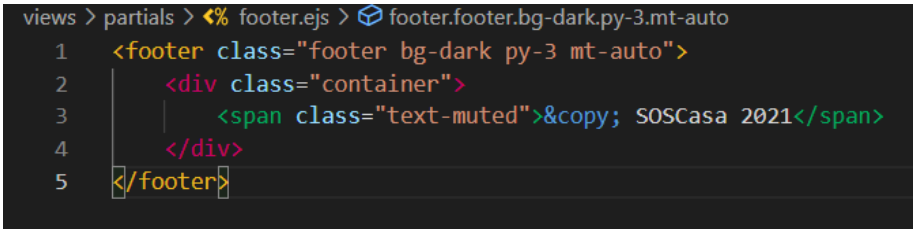
2.6.1.1 Bootstrap

Bootstrap é um CSS framework, *open-source* e gratuito utilizado no *client-side* ou front-end, que pode ou não conter código JavaScript. Os *templates* facilitam a criação de formulários, botões, tipografia e outros componentes visuais.

O framework foi desenvolvido por Mark Otto e Jacob Thornton enquanto trabalhavam no Twitter como uma ferramenta para encorajar a consistência visual e dinâmica entre os ambientes desenvolvidos internamente. Segundo as palavras de Mark Otto (2012) “[...] Um grupo super pequeno de desenvolvedores e eu nos reunimos para projetar e construir uma nova ferramenta interna e vimos uma oportunidade de fazer algo mais. Por meio desse processo, nos vimos construindo algo muito mais substancial do que outra ferramenta interna. Meses depois, terminamos com uma versão inicial do Bootstrap como uma forma de documentar e compartilhar padrões de design e ativos comuns dentro da empresa.”(Mark Otto, 2012, p.1)

Bootstrap foi usado no desenvolvimento para estilizar a aplicação, deixá-la uniforme e agilizar no desenvolvimento uma vez que as classes com os estilos já estão prontas é necessário apenas usá-las no HTML como mostra a figura 9, desta forma o tempo gasto na construção do CSS diminui consideravelmente.

Figura 9 – Bootstrap classes



Fonte: Elaborado pela autora (2021)

2.6.1.2 Express.js

Express.js ou apenas Express é um framework *open-source* para Node.js. Em geral é utilizado no *back-end* para construir aplicações web e web APIs de forma simples e eficiente. Express.js fornece um roteamento simples e um middleware que é responsável por tomar decisões para dar as respostas corretas às solicitações feitas pelo cliente. Além disso, Express oferece muitos recursos adicionais para Node.js e utiliza apenas código JavaScript, o que simplifica para desenvolvedores que já conhecem a linguagem usarem a aplicação. Foi utilizado na aplicação para simplificar e facilitar a criação das API's.

2.6.2 Bibliotecas

Enquanto frameworks são muito mais abrangentes e contém códigos reutilizáveis e genéricos, as bibliotecas executam operações específicas e bem definidas. As bibliotecas se conectam ao código e o código se conecta a um framework.

Uma biblioteca é uma coleção de implementações escritas em uma determinada linguagem na qual é chamada pelo código para executar funções específicas que podem ser implementadas em programas totalmente diferentes. Em geral são menores que os frameworks e contém código não-volátil como configuração de data, documentação, classes, valores, especificação de tipo, entre outros.

2.6.2.1 Mongoose

Segundo a documentação da NPM (2021), "[...] Mongoose é uma ferramenta de modelagem de objetos MongoDB projetada para funcionar em um ambiente assíncrono. O Mongoose oferece suporte para *promises* e *callbacks*." (NPM, 2021, p. 1). Em geral, Mongoose gerencia relacionamento entre dados, converte tipos, fornece validação de esquema e é usado para mapear esses objetos no código. Para usá-lo é necessário fazer o download e instalar a biblioteca usando NPM. Na figura 10 é possível ver como montar um esquema utilizando Mongoose.

Figura 10 – Mongoose Schema

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const reviewSchema = new Schema({
    body: String,
    rating: Number,
    author: {
        type: Schema.Types.ObjectId,
        ref: 'User'
    }
});

module.exports = mongoose.model("Review", reviewSchema);
```

Fonte: Elaborado pela autora (2021)

2.6.2.2 Passaport

Passaport não é na verdade uma biblioteca e sim um *middleware* de autenticação para Node.js. Dentre as várias definições de bibliotecas disponíveis, um middleware é dito como uma biblioteca de middleware. Middlewares são softwares que permitem a comunicação e o gerenciamento de dados entre o sistema operacional e os aplicativos, funcionando de forma oculta no sistema.

A ferramenta é conhecida por ser flexível e modular e pode ser inserida em qualquer aplicativo que use Express. O Passaport faz com que o processo de autenticação, que pode vir a ser complicado, se torne mais simples e rápido para o desenvolvedor. Permite a implementação de login utilizando Facebook e Twitter.

3 A APLICAÇÃO WEB DESENVOLVIDA

A plataforma desenvolvida usando as linguagens de programação e ferramentas citadas no capítulo 2 tem como objetivo criar um ambiente onde pessoas comuns possam ofertar serviços ou buscar por candidatos que desejem realizar serviços rápidos relacionados às necessidades de uma casa, como limpeza, manutenção e organização, na cidade de Joinville e região. Esses trabalhos podem envolver atividades como montagem de móveis, jardinagem, passear com cachorros, cuidar de crianças, entre outras necessidades que possam vir a surgir enquanto se administra um lar. Todas essas atividades postadas visam um trabalho de curto período e que não envolve vínculo empregatício entre o prestador de serviço e o contratante.

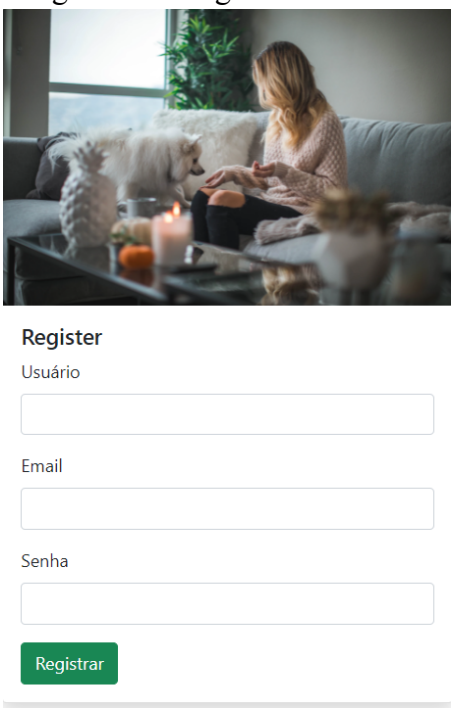
Desde o início da pandemia de COVID-19 no Brasil em 2020 muitas pessoas perderam o emprego e buscam maneiras alternativas de sustentar a família. Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE) (2021), a taxa de desemprego chegou a 14,7% até fevereiro de 2021. Visto esse cenário a plataforma pretende trazer uma possibilidade de renda para essas pessoas e para profissionais que já atuavam nos determinados ramos a crescer seu negócio e conseguir mais exposição a possíveis clientes.

A aplicação é simples e intuitiva e recebe o nome de SOSCasa. A princípio não é necessário efetuar o login ou ter uma conta na plataforma para visualizar as postagens de outros usuários, entretanto é necessário realizar o login para que se possa criar novas postagens e fazer comentários nas postagens existentes. A SOSCasa ainda possui um campo para avaliações onde é possível avaliar o prestador de serviço ou o serviço prestado.

O registro de usuário é simples e é necessário apenas que o indivíduo conceda o nome de usuário, um e-mail que ainda não esteja cadastrado no banco de dados da aplicação e uma senha de sua escolha e pode ser visto na figura 11, caso o usuário deixe algum campo em

branco ou ocorra algum erro durante a criação da conta o sistema retornará um *feedback* ao usuário esclarecendo o erro para que este possa vir a consertá-lo e reenviar a requisição. Se o usuário já possuir uma conta ele será redirecionado para a página de login onde precisará entrar apenas o nome de usuário e senha para entrar na aplicação.

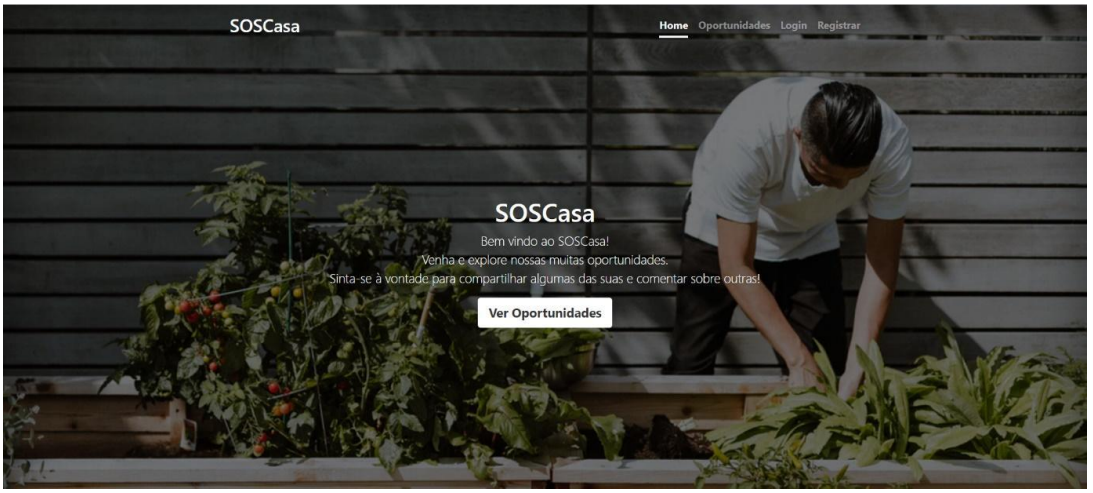
Figura 11 – Registro de usuário



Fonte: Elaborado pela autora (2021)

A página inicial, como mostra a figura 12, é composta por uma imagem, uma breve explicação do propósito da SOSCasa e um botão de redirecionamento que, caso o usuário deseje clicar, o levará para a página principal onde encontrará as postagens dos outros usuários.

Figura 12 – Página inicial SOSCasa

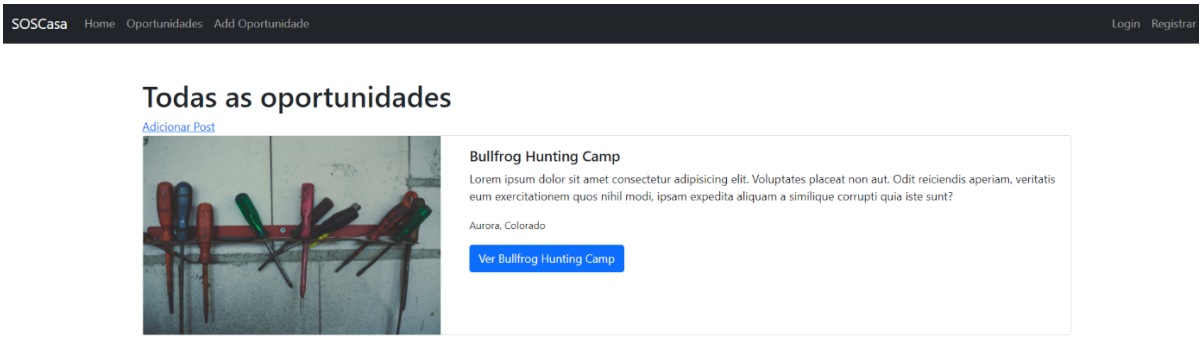


Fonte: Elaborado pela autora (2021)

Na página principal estão visíveis todos os *posts,*indiferente se o usuário estiver cadastrado no sistema ou não. As oportunidades são organizadas por ordem de postagem sendo que o último post a entrar no banco de dados do sistema será o primeiro post que o

usuário irá visualizar ao entrar na página. Nesta página os posts ficam dentro de *cards*, e estão visíveis o título, a descrição, a localização e a imagem sendo que destes, os as imagens apenas serão mostradas caso o autor tenha decidido enviá-las na criação do post como mostra a figura 13. O usuário pode usar a barra de rolagem para ver os diversos post que estiverem disponíveis dentro da aplicação ou ainda usar a barra de pesquisa que se encontra na parte superior da página para buscar posts com títulos específicos.

Figura 13 – Exibição das oportunidades



Fonte: Elaborado pela autora (2021)

A oportunidade conta com os campos de título, localização, preço, descrição e imagem. Os campos de título, preço e descrição são obrigatórios, já os campos de localização e imagem são opcionais e serão preenchidos de acordo com a vontade do usuário. O usuário poderá fazer o upload de uma ou múltiplas fotos como mostra a figura 14. Este upload de imagens é feio usando imagens salvas no computador do usuário e estas imagens são armazenadas em uma nuvem chamada Cloudinary*. É possível editar todos os campos de um post após sua criação, todavia o mesmo só poderá ser feito pelo autor do mesmo, que precisa estar logado para realizar a edição. Além disso, se for da vontade do usuário, ele também poderá deletar as postagens criadas que contém sua autoria.

Figura 14 – Página de criação de uma nova oportunidade

Fonte: Elaborado pela autora (2021)

* Cloudinary – Nuvem utilizada para armazenamento de imagens. https://cloudinary.com/.

Na página de visualização da oportunidade são encontradas as informações relacionadas à oportunidade, nas quais são a imagem, o título, a localização, o preço e a descrição como visto na figura 15. Estão visíveis na mesma página as avaliações com estrelas de 1 a 5 nome de quem avaliou e descrição da avaliação. Também, caso o usuário esteja logado, é possível adicionar uma avaliação.

Figura 15 – Página de visualização de uma oportunidade

Fonte: Elaborado pela autora (2021)

A barra de navegação é fixa no topo da página e está presente em todas as rotas existentes na aplicação. Nela são encontrados os botões:

a) Home - direciona o usuário para a página inicial;
b) Posts - direciona o usuário para a página principal;
c) Novo Post - caso o usuário esteja logado redireciona o usuário para a página de criação de posts. De forma contrária o usuário será redirecionado para a página de login;
d) Login - direciona o usuário para a página de login. Estará visível apenas se o usuário não estiver logado;
e) Registrar - direciona o usuário para a página de registro de novo usuário. Estará visível apenas se o usuário não estiver logado;
f) Logout - realizar o logout do usuário no sistema e redirecionar o mesmo para a página principal. Estará visível apenas se o usuário estiver logado;

Assim como a barra de navegação, toda a aplicação é responsiva e, sendo assim, é possível ter acesso usando o computador, tablet ou celular sendo que o dimensionamento da página e imagens contidas na mesma será redimensionado de acordo com o tamanho da tela do usuário.

4 METODOLOGIA

O objetivo deste trabalho foi a criação de um aplicativo web para oferta de serviços e/ou mão de obra usando tecnologias e conceitos já existentes visando utilizá-los da melhor maneira possível. Segundo Alves (2017) “[...] para inovar não é necessário inventar a roda,

mas adotar soluções que possam sanar problemas presentes.” (ALVES, 2017, Saúde Business). Partindo desta premissa procurou-se tecnologias e conceitos de arquitetura e desenvolvimento de softwares que proporcionam soluções inteligentes e que se encaixem às necessidades da aplicação como arquitetura REST, a escolha de um banco de dados não relacional e utilizar apenas uma linguagem de programação para desenvolver o *front-end* e o *back-end* da aplicação.

Tudo isso faz sentido quando utilizados em conjunto pois segundo W3Tech (2021) JavaScript é a linguagem mais utilizada por desenvolvedores para o *front-end*, logo não é necessário aprender uma segunda sintaxe de programação quando utilizado com o Node.js para o *back-end*. Tendo isto em vista, MongoDB possui uma biblioteca construída em cima do node.js chamada Mongoose o que facilita a sua implementação e compatibilidade. Além disso, como são linguagens e frameworks comumente utilizados juntos, é relativamente fácil encontrar documentação e exemplos de códigos em livros ou na internet.

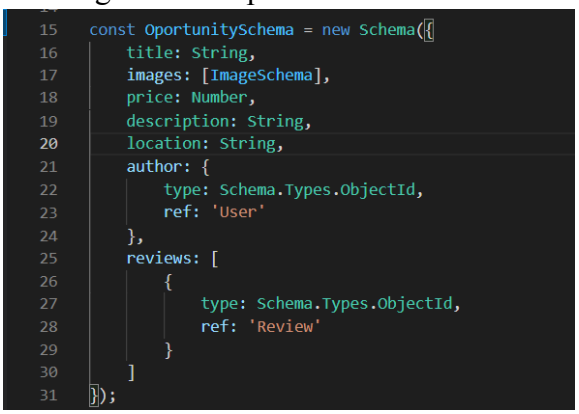
Para desenvolvimento da aplicação web foram necessários alguns passos distintos. Depois da escolha de arquitetura e conjunto de ferramentas a serem utilizadas, foi dado início a construção da aplicação criando um app Express básico como mostrado na figura 16. Este app foi a base para a criação das rotas e desenvolvimento do *back-end* da aplicação.

Figura 16–Express app básico

Fonte: Elaborado pela autora (2021)

O próximo passo foi criar um modelo de post utilizado pelo banco de dados para armazenar as informações contidas na oportunidade. Este modelo continha inicialmente o título, preço, descrição e localização como mostrado da figura 17. Mais tarde foram acrescentados os campos de autor e avaliação.

Figura 17 – Oportunidade Schema



Fonte: Elaborado pela autora (2021)

Depois foi necessário criar algumas funcionalidades de CRUD (Create, Read, Update e Delete) básicas. Foram criadas as rotas para a página inicial (/home), página principal (/index), ver um post (/show), criar um novo post (/new), editar um post (/edit) e por fim excluir um post (delete). Essas rotas foram criadas utilizando o padrão REST de arquitetura e pode ser visualizado na figura 18.

Figura 18 – Modelo CRUD: Criar oportunidade

```
13  module.exports.createOportunity = async(req, res, next) => {
14      const oportunity = new Oportunity(req.body.oportunity);
15      oportunity.images = req.files.map(f => ({ url: f.path, filename: f.filename }));
16      oportunity.author = req.user._id;
17      await oportunity.save();
18      req.flash('success', 'Uma nova oportunidade criada com sucesso!');
19      res.redirect(`/oportunities/${oportunity._id}`)
20  }
```

Fonte: Elaborado pela autora (2021)

Depois de criar as rotas foi criado um HTML básico para cada uma das rotas que posteriormente veio a ser estilizado usando Bootstrap 5. Também foram criadas neste passo a barra de navegação e rodapé utilizando o modelo básico retirado do website do Bootstrap (2021) e adaptado para o contexto da aplicação. Foi adicionado o estilo e formatação das páginas criadas utilizando as classes e grid system do Bootstrap.

O próximo passo foi criar uma validação do formulário principal no lado do cliente para caso o cliente possa vir a entrar alguma informação inválida, ou submeter algum campo necessário em branco, o sistema não permitirá a submissão dele. Tendo em vista que erros podem acontecer, é necessário comunicá-lo ao usuário e para isso um erro *handler* foi adicionado junto às rotas para redirecionar o usuário e informá-lo que algum erro aconteceu. Um exemplo das rotas de erro pode ser visto na figura 19, assim como o *template* de mensagem de erro.

Figura 19 – Erro *Handler*

```
app.all('*', (req, res, next) => {
    next(new ExpressError('Página não encontrada', 404))
});

app.use((err, req, res, next) => {
    const {statusCode = 500} = err;
    if(!err.message) err.message = 'Alguma coisa deu errado'
    res.status(statusCode).render('error', {err} );
});
```

Fonte: Elaborado pela autora (2021)

Por questões de segurança é necessário também acrescentar uma validação do formulário no *back-end* da aplicação uma vez que *post requests* podem ser enviados utilizando Postman* ou outras formas. Para essa validação foi utilizado JOI** que é uma ferramenta popular de validação JavaScript que percorre o formulário validando os campos e retorna uma mensagem caso tenha acontecido algum erro durante a postagem. Um exemplo de como a ferramenta foi implementada pode ser visto na imagem 20.

---

* Postman – Aplicação em que é possível executar requisições no servidor. https://www.postman.com/.
**JOI – Ferramenta popular para validação de formulários. https://joi.dev/

Figura 20 – Modelo de implementação da ferramenta JOI

```
schemas.js > ...
1   const Joi = require('joi');
2   const { number } = require('joi');
3
4   module.exports.oportunitySchema =  Joi.object({
5       oportunity: Joi.object({
6           title: Joi.string().required(),
7           price: Joi.number().required().min(0),
8           location: Joi.string().required(),
9           description: Joi.string().required(),
10      }).required(),
11      deleteImages: Joi.array()
12  });
```

Fonte: Elaborado pela autora (2021)

Para criar o formulário de avaliação o processo foi similar a criação do formulário principal onde primeiro cria-se o modelo que será utilizado pelo banco de dados para armazenamento e posteriormente consulta dessas informações. Este modelo contém o corpo que será onde o usuário escreverá a descrição da avaliação (*string*) e o valor que é onde o usuário quantifica o serviço prestado (*number*). Em seguida foi criado o formulário de avaliação HTML, adicionado validação para este formulário, adicionado os estilos, criada as rotas para criar nova avaliação, editar avaliação existente ou excluir avaliação. Este formulário foi incluído na página onde as oportunidades são exibidas e podem ser vistos por qualquer usuário que esteja na página e o HTML pode ser visto na figura 21. Foi acrescentado o esquema de estrelas para uma melhor usabilidade do usuário onde o mesmo pode escolher 1 estrela para quando considerar o serviço péssimo e 5 estrelas quando considerar o serviço ótimo.

Figura 21–HTML de exibição das avaliações

```
<% for(let review of oportunity.reviews) { %>
    <div class="card mb-3 ">
        <div class="card-body">
            <h5 class="card-title">Avalição:</h5>
            <p class="starability-result" data-rating="<%= review.rating %>">
              Rated: <%= review.rating %> stars
            </p>
            <h6 class="card-subtitle mb-2 text-muted">Por <%= review.author.username %></h6>
            <p class="card-text"> <%= review.body %></p>
            <% if(currentUser && review.author.equals(currentUser._id)) { %>
            <form action="/oportunities/<%=oportunity._id%>/reviews/<%=review._id%>?_method=DELETE" method="POST">
                <button class="btn btn-sm btn-danger">Apagar</button>
            </form>
            <% } %>
        </div>
    </div>
<% } %>
```

Fonte: Elaborado pela autora (2021)

A autenticação é uma etapa importante na criação de uma aplicação web e foi feita usando a biblioteca Passaport.js. Após a configuração os passos foram similares a criação do formulário principal. Sendo assim o primeiro passo, assim como para o formulário principal e o formulário de avaliação, foi criar um modelo que é utilizado pelo banco de dados.

Após foram criados o formulário de registro e o de login, as rotas para estes formulários que são "/register" e "/login" respectivamente que podem ser vistas na figura 22. Para saber se o usuário está logado foi utilizado o Middleware isLoggedIn*. Por fim, foi adicionada a rota de

* isLoggedIn – *Middleware* para verificar se o usuário está logado.
https://docs.oracle.com/cd/E14571_01/doc.1111/e10726/c06_core_ref116.htm#CSIDO439.

logout. Além de criar as rotas de login e registro, também foi adicionado a lógica de redirecionar o usuário para a página principal após o registro ou o login ser concluído.

Figura 22– Rotas de Registro e Login

```
const express = require('express');
const router = express.Router();
const users = require('../controllers/users');
const catchAsync = require('../utils/catchAsync')
const passport = require('passport');

router.route('/register')
    .get(users.renderRegister)
    .post(catchAsync(users.register));

router.route('/login')
    .get(users.renderLogin)
    .post(passport.authenticate('local', {
        failureFlash: 'Usuário ou senha incorreta!',
        failureRedirect: '/login'
    }), users.login);

router.get('/logout', users.logout)

module.exports = router;
```

Fonte: Elaborado pela autora (2021)

Para poder fazer o upload de imagens usando arquivos salvos no computador do usuário foi utilizado Cloudinary para armazenar as imagens que os usuários fizerem o *upload* no SOSCasa, uma vez que o banco de dados Mongo não suporta arquivos com tamanhos acima de 16MB, desta forma MongoDB apenas armazenará as URLs das imagens armazenadas no Cloudinary e fará uma requisição ao Cloudinary quando necessário. Para ser possível fazer o upload de imagens é necessário mudar o método de envio do formulário para *enctype="multipart/form-data"* entretanto para este formulário funcionar foi necessário adicionar ao projeto um middleware chamado Multer que basicamente lida com formulários do tipo *multipart/form-data*e segundo a documentação NPM (2021) "Multer adiciona um objeto de corpo e um arquivo ou objeto de arquivos ao objeto de solicitação. O objeto de corpo contém os valores dos campos de texto do formulário, o arquivo ou objeto de arquivos contém os arquivos carregados por meio do formulário." Desta forma é possível fazer o upload de imagens e texto ao mesmo tempo.

Por fim foi feito o *deploy* (implementação ou lançamento) da aplicação utilizando Heroku – Heroku é uma plataforma que permite que os desenvolvedores criem, executem e operem aplicativos inteiramente na nuvem. A aplicação pode ser acessada de qualquer lugar pelo link: https://infinite-reef-58067.herokuapp.com/.

Foram executados testes de usuário para testar o funcionamento da aplicação, porém testes unitários, integrados ou testes automatizados não foram aplicados.

5 CONSIDERAÇÕES FINAIS E CONCLUSÃO

Este artigo teve como objetivo apresentar as ferramentas e métodos disponíveis na construção de uma aplicação web. As opções na escolha destas ferramentas são imensas e é necessário levar em contas as necessidades da aplicação, o tempo disponível para o desenvolvimento e a quantidade de recursos disponíveis seja eles financeiros, de máquinas ou de pessoas e por isso a escolha das linguagens, assim como dos frameworks e bibliotecas

apresentados neste artigo são tão importantes pois eles impactam diretamente o tempo, custo e facilidade de desenvolvimento. Além da escolha das ferramentas também foram apresentados a arquitetura, os métodos, e os passos utilizados para desenvolver a aplicação. Também foi apresentado o modelo final da aplicação web desenvolvida. Como sugestão para trabalhos futuros fica o aprimoramento da segurança da aplicação utilizando Mongo *injections* ou alternativas que façam sentido para a aplicação e/ou criar uma política de segurança de conteúdo. Além desta, outra sugestão importante é a aplicação de testes unitários e testes integrados que garantam o funcionamento correto da aplicação.

REFERÊNCIAS

DIAS NETO, Arilo Carlos. **Metodologia de desenvolvimento de aplicações web**. 2008. Disponível em: https://www.devmedia.com.br/metodologias-de-desenvolvimento-de-aplicacoes-web-parte-01/9816 Acesso em: 31 jul. 2021.

ZAUPA, Fábio. GIMENES, Itana M. S. COWAN, Don. ALENCAR, Paulo. LUCENA, Carlos. **Um Processo de Desenvolvimento de Aplicações Web baseado em Serviços.** 2007**.** Disponível em: http://www.ic.unicamp.br/sbcars2007/tecnicas/files/sbcars2007-zaupa-processo.pdf Acesso em: 31 jul. 2021.

THOMAS, Aran. FINGENT. 2020.“Top 6 tech stacks that reign software development in 2020”. 2020. Disponível em: https://www.fingent.com/blog/top-6-tech-stacks-that-reign-software-development-in-2020/ Acesso em: 25 jul. 2021.

MDN WEB DOCS. **HTML basics**. 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics Acesso em: 24 jun. 2021.

WOOD, Adam. **Learn HTML code, tags e CSS**. 2018. Disponível em: https://html.com/html5/ Acesso em: 24 jun. 2021.

W3C. **HTML & CSS. What is CSS**. 2016. Disponível em: https://www.w3.org/standards/webdesign/htmlcss#whatcss. Acesso em: 25 jun. 2021.

MDN WEB DOCS. **CSS basics**. 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics Acesso em: 25 jun. 2021.

W³TECHS WEB TECHNOLOGY SURVEYS. **Usage statistics of JavaScript as client-side programming language on websites**. 2021. Disponível em: https://w3techs.com/technologies/details/cp-javascript/. Acesso em: 28 jun. 2021.

W³TECHS WEB TECHNOLOGY SURVEYS. **Usage statistics of JavaScript libraries for websites**. 2021. Disponível em: https://w3techs.com/technologies/overview/javascript_library. Acesso em: 28 jun. 2021.

FAWCETT, Amanda. EDUCATIVE. **JavaScript ES6 Tutorial: A complete crash course on modern JS.** 2020. Disponível em: https://www.educative.io/blog/javascript-es6-tutorial-a-complete-crash-courseAcesso em: 28 jun. 2021.

DAHL, Ryan. NODE.JS. **Porting Node to Windows With Microsoft 's Help**. 2011. Disponível em: https://nodejs.org/en/blog/uncategorized/porting-node-to-windows-with-microsofts-help/ Acesso em**:** 28 jun. 2021.

NODE.JS. **Introduction To node.js.** Disponível em: https://nodejs.dev/learn. Acesso em: 01 ago. 2021.

MDN WEB DOCS. **Introdução Express/Node**. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introduction Acesso em: 30 jun. 2021.

NODE.JS. **Releases**. 2021. Disponível em: https://nodejs.org/en/about/releases/ Acesso em:30 jun. 2021.

CLARK, Mariana. **Back4App**. 2020. Dez principais estruturas de front-end e back-end. Disponível em: https://blog.back4app.com/pt/dez-principais-estruturas-de-front-end-e-back-end/. Acesso em: 28 jun. 2021.

MONGODB | DOCUMENTATION. **Introduction to MongoDB**. Disponível em: https://docs.mongodb.com/manual/introduction/. Acesso em: 1 jul. 2021.

OTTO, Mark. **Bootstrap in A List Apart No. 342** . 2012. Disponível em: https://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/ Acesso em:5 jul. 2021.

NETWORK PACKAGE MANAGER - NPM. **Mongoose**. 2021. Disponível em: https://www.npmjs.com/package/mongoose. Acesso em: 6 jul. 2021.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA.**Desemprego**. Rio de Janeiro: IBGE, 2021. Disponível em: https://www.ibge.gov.br/explica/desemprego.php. Acesso em: 1 ago. 2021.

ALVES, Maicon. Saúde Business. **Para inovar não é necessário inventar a roda**. 2017. Disponível em: https://www.saudebusiness.com/empreendedorismo/%E2%80%9Cpara-inovar-no-necessrio-inventar-roda%E2%80%9D-diz-maicon-alves. Acesso em: 14 jul. 2021.