UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO COMPARATIVO ENTRE FRAMEWORKS DE FRONT-END

SARAH CRISTINA SILVA CRUZ

RECIFE
2017

SARAH CRISTINA SILVA CRUZ

ESTUDO COMPARATIVO ENTRE FRAMEWORKS DE FRONT-END

Monografia apresentada à Universidade Federal de Pernambuco, no curso de Ciência da Computação, apresentado à disciplina Trabalho de Graduação.

Orientador: Sérgio Castelo Branco Soares
Coorientador: Samuel Carlos Romeiro

RECIFE
2017

SUMÁRIO

AGRADECIMENTOS

É com muita alegria que eu escrevo esses agradecimentos. Não faço questão de ser um agradecimento resumido porque penso que esse é o momento de expressar minha gratidão a todas as pessoas importantes que passaram na minha vida durante os anos que passei na UFPE como graduanda em Ciência da Computação.

Primeiro, quero agradecer a Deus por me dar sabedoria e condições de concluir o curso. Sem ele, eu não teria forças durante todos esses anos.

Segundo, quero agradecer minha mãe, Magna do Carmo Silva Cruz, por ser minha inspiração. Uma mulher forte, guerreira, que todos os dias me inspira a melhorar. Obrigada por estar sempre me apoiando, principalmente nos momentos difíceis. Te amo! Também agradeço a meu segundo pai, Sergio Paulo Lemos Monteiro, por todas as longas conversas trocadas durante o jantar e por me presentear com minha meia irmã, Aline Lemos, que me ajudou muito também!.

Gostaria de agradecer a meu pai, Fredd Willham Silveira da Cruz, que, mesmo distante, está sempre me apoiando e me dando palavras de sabedoria. Também agradeço a minha família paterna que mesmo distante torce por mim em todas as minhas realizações. Obrigada pelo carinho, vovó Graça, tias Jane e Meyre, e todos os outros que habitam na terrinha do Acre.

Agradeço a minha família materna, por todos os domingo maravilhosos que recarregam minhas forças para enfrentar a semana. Vovô Braz e Vovó Leni; Tios: Cristina e Hermes, Beto, Fábio e Érika, Junior e Silvania; e meus primos e primas. Vocês são a minha alegria!!!

Um agradecimento mais que especial para meu orientador, Sérgio Soares que, quando nem eu acreditava que podia fazer este TCC, fez com que eu tivesse confiança em mim. Também agradeço a Samuel Carlos, meu coorientador, por estar sempre do meu lado para me ajudar durante o TCC, mesmo que seja sem querer; afinal, você sentava do meu lado no estágio.

Um agradecimento enorme às amizades construídas no Centro de Informática, por todas as horas de estudo e pelos momentos de descontração que me mantiveram viva. Obrigada por sempre estarem disponíveis para me ajudar em minhas dúvidas. Um agradecimento especial a Daniel Oliveira e Túlio Lages, vocês foram a parte mais importante da minha graduação. Um beijo pra essa galera do

PET que eu amo: Pedro Torres, Rafael Nunes, Pedro Diniz, Marlon Reghert e Gustavo Stor.

No CIn, participei de grupos maravilhosos: Ai Ai Ai Kwai, Arestas, Só pra falar e Totalmente Demenos. Vocês foram responsáveis por momentos de risos e muitos *memes*. Daniele Cunha e Renata Andrade, obrigada por todo carinho, conversas e alegrias. Vocês são as amigas mais lindinhas que eu podia ter. Fanny Chien e Déborah Mesquita, quero colocar vocês em um pote e carregar comigo pra sempre. Déborah não vai parar de rir e Fanny não vai entender nada!

Um muito obrigada a todos os membros que participaram comigo no CITi. Foram muitas pessoas que tive a oportunidade de trabalhar, mas agradeço em especial a: Alexandre Cisneiros, José Luciano, Estéfane Oliveira, Leonardo Coutinho, Bruna Cruz, Victor Vernilli, André Carneiro, Vinícius Lira, Caio Calado, Robertson Novelino, Álvaro Conolly, Victor Gutemberg, João Veras, Bruno Barbosa e Victor Leal. Ainda agradeço a Thaís Morais, Gisely Melo, Rayane Martin, João Guilherme, Álef Matheus e Anne Kelly pelos momentos de descontração!

Ainda agradeço a algumas amizades que fiz no CIn que não são do CIn: Eduarda Scharnhorst, melhor blogueira de Recife; Pedro Crivellari, meu colírio; Mateus Bolsoni, meu designer predileto.

Gostaria de agradecer a Elizabeth Karina, minha amiga de infância, por estar ao meu lado durante esses anos e por sempre ter paciência, pois às vezes eu não podia sair. Te amo! Um beijo para minhas amigas do colégio que me ouviram falar bastante do CIn nos últimos anos: Carolina Cadete, Ana Carolina, Mariane Villarouco e Juliana Albuquerque, vocês são maravilhosas e divas. Estamos ficando velhas!

Agradeço do fundo do coração a Rossana Barreira, Sandra Couto e Elizama Santana pelo apoio emocional que tive durante esses anos. Vocês foram responsáveis por uma transformação maravilhosa em mim. Obrigada!

Também gostaria de agradecer aos meus amigos no Instituto SENAI de Inovação: Daniel Alexandro, David Wilson, Sérgio Santos, Hugo Leonardo, Ricardo Robson e Deyvson Lazaro. Foi uma honra trabalhar com vocês.

Sou grata a todos os professores e funcionários do CIn por permitir que eu tivesse acesso a uma formação de qualidade maravilhosa. Fernando Castor, Adriano Sarmento, Ruy de Queiroz, Sílvio Melo, Anjolina Grisi, Carina Frota, vocês

foram grandes inspirações pra mim. Muita gratidão a ‘Seu Edson’ e ‘Dona Hilda’, a presença de vocês alegra o CIn.

Não poderia esquecer de agradecer ao SENAC, seus professores e funcionários, porque o curso de inglês me ajudou a ter êxito nas atividades desenvolvidas na graduação e, principalmente, no meu TCC, quando mais da metade de minha bibliografia foi nesta língua. Um beijo para Rebeca, Gabriel e meu professor Leo. You guys are the best!

Agradecimento especial aos meus amigos da academia (de ginástica mesmo!) que me acompanharam na transformação de meu corpo e reeducação alimentar que passei durante o meu curso. Tudo isso foi importante para que eu continuasse focada e determinada, em meio a ‘loucura do CIn’; o que me fez conseguir finalizar meus estudos com o corpo e mente sã. Stephanie, Junior e Cris, Valeu!!!

Por fim, agradeço a pessoas que estiveram comigo, no dia a dia em minha casa, e me deram um suporte para que eu pudesse ficar mais tranquila para enfrentar as jornadas de estudo e viradas no CIn: Carminha, Bethania e Dona Marli. Obrigada pelos almoços maravilhosos e cuidados comigo.

RESUMO

A área de desenvolvimento web vem crescendo ao longo dos anos e com diversas oportunidades. A cada ano surgem novos frameworks e tecnologias que facilitam a vida do profissional. Sendo assim, parte da função de um bom desenvolvedor Front-End é estar aberto a estudar as inovações na área e se adaptar a essas novas ferramentas ou aperfeiçoá-las. Diante desse cenário, é necessário um levantamento sobre o contexto atual da área, analisando os frameworks e tecnologias em evidência. Este trabalho realiza um estudo comparativo entre dois frameworks de desenvolvimento Front-End. Como prova de conceito, houve uma comparação do estado da prática, com aplicações que utilizam os principais recursos dos frameworks, destacando vantagens e desvantagens. Os sistemas contaram com um design responsivo.

Palavras-chave: Frameworks, Front-end, Estudo Comparativo

ABSTRACT

Web development has been growing over the years with several opportunities. Each year new structures and technologies comes up to make life easier for professionals. Therefore, the role of a good Front End developer includes to be open to innovations in the area and adapting to new tools or improving them. With this scenario, a survey is needed on the current context of the area, analyzing this frameworks and technologies in evidence. This work performs a comparative study between two Front-End frameworks. As proof of concept, there was a state-of-the-practice comparison, with applications that use the main features of those frameworks, highlighting advantages and disadvantages. The systems had a responsive design.

Keywords: Frameworks, Front-end, Comparative Study.

LISTA DE TABELAS

LISTA DE FIGURAS

1 INTRODUÇÃO

1.1 CONTEXTO

A área de desenvolvimento web vem crescendo, ao longo dos anos, e com diversas oportunidades. Segundo Matheus Lima (2017),

> A área de desenvolvimento como um todo ainda é bem fértil, e conta com um mercado que é mais favorável aos desenvolvedores do que as empresas. Ou seja, as empresas disputam os candidatos e não o contrário. Além de ser uma área que passou praticamente ilesa a maior recessão que o Brasil já teve, e conta com salários bastante atrativos. Fora que é totalmente possível trabalhar 100% remoto para empresas americanas, por exemplo, e ganhar em dólar. E pra quem simplesmente não quer mais ficar no Brasil, é possível conseguir vagas internacionais em países como Alemanha, Nova Zelândia, Irlanda, EUA. (2017).

Além de ser uma área estratégica para a economia global, a cada ano surgem novos frameworks e tecnologias que facilitam a vida do desenvolvedor. Sendo assim, parte da função de um bom desenvolvedor Front-End é estar aberto a estudar as inovações na área e se adaptar a essas novas ferramentas ou aperfeiçoá-las.

O desenvolvedor Front-End enfrenta vários desafios, porém, o principal deles é escolher qual dessas tecnologias é a mais adequada ao seu projeto, de acordo com sua necessidade (CAMPOS, 2016). Nesse sentido, é importante que ele saiba trabalhar com design responsivo, pois, hoje em dia, temos uma enorme variedade de tamanhos de telas e, ainda, precisamos considerar que o celular é o maior meio de acesso a internet (VILLELA, 2016).

1.2 OBJETIVO GERAL

Diante do cenário do desenvolvimento de Front-End, é necessário um levantamento sobre o contexto atual da área, analisando os frameworks e tecnologias em evidência.

Este trabalho tem como objetivo principal *realizar um estudo comparativo entre dois frameworks de desenvolvimento Front-End.* Como prova de conceito, haverá uma comparação no estado da prática, com aplicações que utilizarão os

principais recursos dos frameworks, destacando vantagens e desvantagens. Os sistemas contarão com um design responsivo.

1.3 OBJETIVOS ESPECÍFICOS

Como objetivos específicos buscaremos fazer um levantamento dos principais frameworks utilizados recentemente e a escolha de dois deles para avaliação. Será produzido o desenvolvimento front-end de duas aplicações com dois frameworks escolhidos, utilizando Design Responsivo em ambas as aplicações. Após o desenvolvimento dos sistemas, haverá uma comparação dos principais recursos dos frameworks escolhidos a fim de identificar as vantagens e desvantagens de cada framework na prática.

1.4 ESTRUTURA DO TRABALHO

Neste capítulo encontra-se a introdução do trabalho, onde são apresentados os motivos para escolha do tema. Nele também são apresentados os objetivos gerais e específicos do trabalho. Os próximos capítulos estão organizados da seguinte maneira: O Capítulo 2 apresenta a fundamentação teórica. Ou seja, conceitos importantes para o entendimento do trabalho.

O Capítulo 3 faz um levantamento dos frameworks de front-end mais relevantes hoje em dia. É este capítulo que introduz os frameworks selecionados. No Capítulo 4 são apresentados os critérios de comparação selecionados, que são divididos em duas categorias: critérios sociais e critérios técnicos.

No Capítulo 5 é descrita a aplicação que foi desenvolvida. No Capítulo 6 são apresentadas as análise feitas sobre o desenvolvimento do sistema proposto. E por fim, o Capítulo 7 traz as conclusões do trabalho e ideias para possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão discutidos alguns conceitos importantes para o entendimento do trabalho.

2.1 FRAMEWORKS

As respostas para “O que é um framework?” são diferentes em diversos lugares. Gerdessen (2007) apresenta várias definições de framework, onde todas convergem para um ponto principal que é a técnica de reuso do design ou do código. Muitas vezes, os desenvolvedores precisam reescrever várias funções iguais ou similares, causando mais trabalho e perda de tempo para o projeto. O framework veio para diminuir esse esforço, pois ele é, em palavras simples, um suporte com um conjunto de códigos que podem ser reutilizadas. Gerdessen traz uma definição mais específica: “Um framework é um esqueleto para aplicações, visando um domínio específico, abrangendo certas áreas-chave e a interação entre elas dentro desse domínio específico.” (2007, p.12)

Entre as vantagens de se usar um framework, Souza (1998) cita a diminuição de tempo e custo no desenvolvimento de um sistema, afinal os componentes já foram desenvolvidos e testados anteriormente. Além disso, ele aponta a diminuição de manutenção: “Devido a herança, quando um erro em um framework é corrigido, ou uma característica adicionada, os benefícios são imediatamente estendidos às novas classes” (p. 16).

Desta forma, chegamos à conclusão de que framework são importantes para aumentar a produtividade do desenvolvimento de um sistema web, pois são ótimas ferramentas para diminuir o reuso de códigos.

2.2 BIBLIOTECAS

Em um post no site BeCode, Zanette define Biblioteca como um meio de compartilhar funções e métodos:

> Se você tiver que fazer um trabalho de matemática, por exemplo, poderá ir até uma biblioteca física, pegar um livro e utilizar equações desenvolvidas no livro. Então, não será preciso desenvolver as equações desde o início.

> Em outras palavras, você não precisa reinventar a roda! Ainda bem, né? Em programação, a biblioteca possui a mesma função. Desenvolvedores disponibilizam bibliotecas que possuem muitas funções prontas. Assim, outros programadores podem utilizá-las, permitindo que o desenvolvimento seja mais fácil e rápido. (2017)

Zanette (2017) ainda dá como exemplo o jQuey que é uma biblioteca de JavaScript muito reconhecida por desenvolvedores.

Muitas pessoas confundem as definições de biblioteca e framework. A principal diferença entre os dois é o grau de complexidade. Framework são muito mais complexos do que bibliotecas, além de terem uma relação de herança ou dependência entre seus componentes. Souza (1998), ainda, discorre a diferença entre estas tecnologias:

> Um framework corresponde a um projeto de alto nível, consistindo de classes abstratas e concretas que especificam uma arquitetura para aplicações. Uma biblioteca de classes orientada a objetos é apenas um conjunto de classes, não necessariamente relacionadas, que fornece um conjunto de serviços disponibilizado através da interface pública de suas classes. De maneira similar, uma biblioteca de procedimentos fornece uma série de serviços através de chamadas às suas rotinas. Nenhum dos dois tipos de biblioteca permite a reutilização de projeto. O modo como os frameworks operam também é diferente de outras abordagens. Em um framework bem-projetado, o desenvolvedor especifica somente o código de métodos de classes específicas que corresponderão a sua aplicação. O próprio framework irá se encarregar de chamar este código quando for necessário. Em bibliotecas de classes orientadas a objeto ou procedurais, além do código da aplicação propriamente dita, o desenvolvedor deve especificar o fluxo de execução, a estrutura do programa, etc. (1998, p. 13)

Apesar de serem bastante parecidos e confundidos, os frameworks são maiores e mais robustos do que as bibliotecas, proporcionando mais funcionalidades.

2.3 HTML, CSS E JAVASCRIPT

HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) e JAVASCRIPT são três tecnologias importantes que todo desenvolvedor web precisa saber.

O HTML é responsável por definir os componentes de uma página Web através de tags e atributos. Conforme aponta Robbins (2012),

> HTML is not a programming language; it is a markup language, which means it is a system for identifying and describing the various components of a document such as headings, paragraphs, and lists[1]. (2012,p. 12)

Já o CSS é responsável pela aparência desses componentes, adicionando cores e espaçamento, deixando a página mais apresentável e agradável ao usuário. Goodman (2002), ao discorrer sobre o CSS, destaca que

> The Cascading Style recommendation lets authors define style rules that are applied to HTML elements. A rule may apply to a single element, a related group of elements, or all elements. A rule may apply to a single element, a related group of elements, or all elements of a particular type (such as all p elements). Style rules influence the rendering of elements, including their color alignment, border, margins, and padding between borders and the content.[2] (2002, p. 9)

Flanagan (1998) explica de forma detalhada os aspectos do Javascript. Ele afirma que ele é uma linguagem de programação da Web que pode ser usada apenas a linguagem em si e sua API (Interface de Programação de Aplicação) interna mínima, ou em navegadores Web, onde normalmente é chamada de JavaScript do lado do cliente.

> O uso do JavaScript em documentos Web normalmente deve ser controlado e moderado. O papel apropriado de JavaScript é melhorar a experiência de navegação do usuário, tornando mais fácil obter ou transmitir informações. A experiência do usuário não deve depender de JavaScript, mas ela pode ajudar a facilitar essa experiência. (FLANAGAN, 1998, p. 302)

Com as definições apresentadas, reforçamos que essas tecnologias são a tríade básica que todo desenvolvedor web deve estar familiarizado. Em resumo, o HTML serve para estruturar a informação, o CSS serve para estilizar esse conteúdo, e o Javascript serve para definir o comportamento dos componentes.

[1] Tradução livre: HTML não é uma linguagem de programação. É uma linguagem de marcação, o que significa que é um sistema para identificação e descrição de vários componentes de um documento, como cabeçalho, parágrafos e listas.

[2] Tradução livre: As recomendações do CSS deixam os autores definir regras de estilo que são aplicadas nos elementos HTML. Uma regra pode ser aplicada em um único elemento, a um grupo de elementos, ou a todos os elementos de um tipo particular (como todos os *p*). Regras de estilo influenciam na renderização dos elementos, incluindo suas cores, alinhamentos, bordas, margens e espaço dentre borda e conteúdo.

2.4 FRONTEND & BACKEND

Muitas pessoas ainda se confundem sobre as definições e atividades de “Front-end” e “Back-end”. Robbins (2012, p. 9) define bem essa diferença. Ela afirma, em tradução aberta que “Frontend se refere a qualquer aspecto do processo de design que aparece ou se relaciona diretamente ao navegador” e cita algumas atividades, como por exemplo, documento HTML, desenvolvimento do estilo, código Javascript e design de informações em relação à experiência do usuário no site.

Robbins relata que o backend concentra as atividades que funcionam no servidor, deixando as páginas dinâmicas e interativas. Ela, ainda, cita algumas ações do backend tais como organizar a informação no servidor, processar formulários, organizar o banco de dados e utilizar linguagens server-side.

2.5 DESIGN RESPONSIVO

Silva (2014) afirma que o termo foi apresentada por Ethan Marcotte no dia 25 de maio de 2010, o “pai do design responsivo”, através de uma matéria ao site AlistApart. Na matéria, Ethan alega que a web precisa de um design flexível que “aceite o fluxo e o refluxo das coisas”.

Zemel (2013) apresenta que um site ou sistema com design responsivo pode ser acessado de vários dispositivos eletrônicos com acesso à internet independente de suas configurações. Portanto, mesmo sendo acessado por aparelhos diferentes, o site ou sistema continuará bem organizado, diagramado e terá uma boa distribuição de conteúdos. Neste caso, pode, também, haver uma seleção de assuntos que serão apresentados de acordo com o dispositivo escolhido.

2.6 FRAMEWORKS CSS

Frameworks CSS são ferramentas para auxiliar o desenvolvimento do design responsivo. Eles possuem um padrão CSS feito para facilitar o trabalho dos desenvolvedores que lidam com HTML, CSS e Javascript. Atualmente, existem vários frameworks de CSS, visto que haver uma grande necessidade de sistemas com um design responsivo. Alguns exemplos de Frameworks de CSS: Bootstrap, Materialize, Foundation, Semantic UI, Pure, entre outros.

A seguir, apresentamos uma tabela comparativa entre esses frameworks. As informações foram retiradas de publicações em sites sobre tecnologia[3], pois eles são uma das fontes principais em que os desenvolvedores procuram referências e comparações para tomada de decisão de qual framework usar. Alguns valores foram atualizados, como por exemplo, versão atual e popularidade.

Tabela 1 - Tabela Comparativa entre os Frameworks CSS mais usados atualmente

ITEM	Bootstrap	Foundation	Semantic UI	Pure	Materialize	UIkit
No mercado desde	2011	2011	2013	2013	2014	2013
Versão atual	v4.0.0	v6.4.6	v2.2.13	v1.0.0	1.0 Alpha	3.0.0 - Beta.35
Popularidade (Github)	118,494 estrelas	26,690 estrelas	38,267 estrelas	17,814 estrelas	30,176 estrelas	11,070 estrelas
Pré-processadores	Less e Sass	Sass	Less	Nenhum	Sass	Less, Sass
Responsivo	Sim	Sim	Sim	Sim	Sim	Sim
ícones	Glyphicons Halflings	Foundation Icon Fonts	Font Awesome	Nenhum. É possível usar o Font Awesome.	Material Icons Library do google	Ícones próprios em SVG
Extras/Add-ons	Sim	Sim	Não	Não		Sim
Documentação	Muito boa	Muito boa	Muito boa	Boa	Boa	Boa
Suporte de navegador	Chrome, Safari, Firefox, IE8+ (é preciso Respond.js para IE8)	Chrome, Firefox, Safari, IE9+; iOS, Android, Windows Phone 7+	Chrome, Firefox, Safari, IE10+ (IE9 com o prefixo do navegador apenas), Android 4, Blackberry 10	Últimas versões de Firefox, Chrome, Safari; IE7+; iOS 6.x, 7.x; Android 4.x	Chrome 35+ Firefox 31+ Safari 9+ Opera Edge IE 11+	Chrome, Firefox, Safari, IE9+

Fonte: A autora

[3] Os sites consultados foram: SitePoint, MaterializeCSS, GitHub, GetBootstrap, PureCSS, Foundation, Semantic-UI e GetUIkit. Ao final do trabalho encontram-se as referências completas desses sites.

2.7 MVC

A primeira definição de Padrão de Projeto surgiu em 1979, através de Christopher Alexander em seu livro “A Times Way of Building”. “Cada padrão é uma regra de três partes, que expressa uma relação entre um certo contexto, um problema e uma solução” (p.207). Em outras palavras, um padrão de projeto é uma solução para um problema que já aconteceu diversas vezes em um certo contexto (DevMedia). Existem vários tipos de padrões: Facade, Interpreter, Iterator, Observer, Visitor, Factory Method, entre outros (GAMMA et al, 2000). O padrão de arquitetura MVC (Model-View-Controller) é muito usado em diversos sistemas. Ele é dividido em três camadas: Model, View e Controller. Vinícius Durello explica as camadas do MVC:

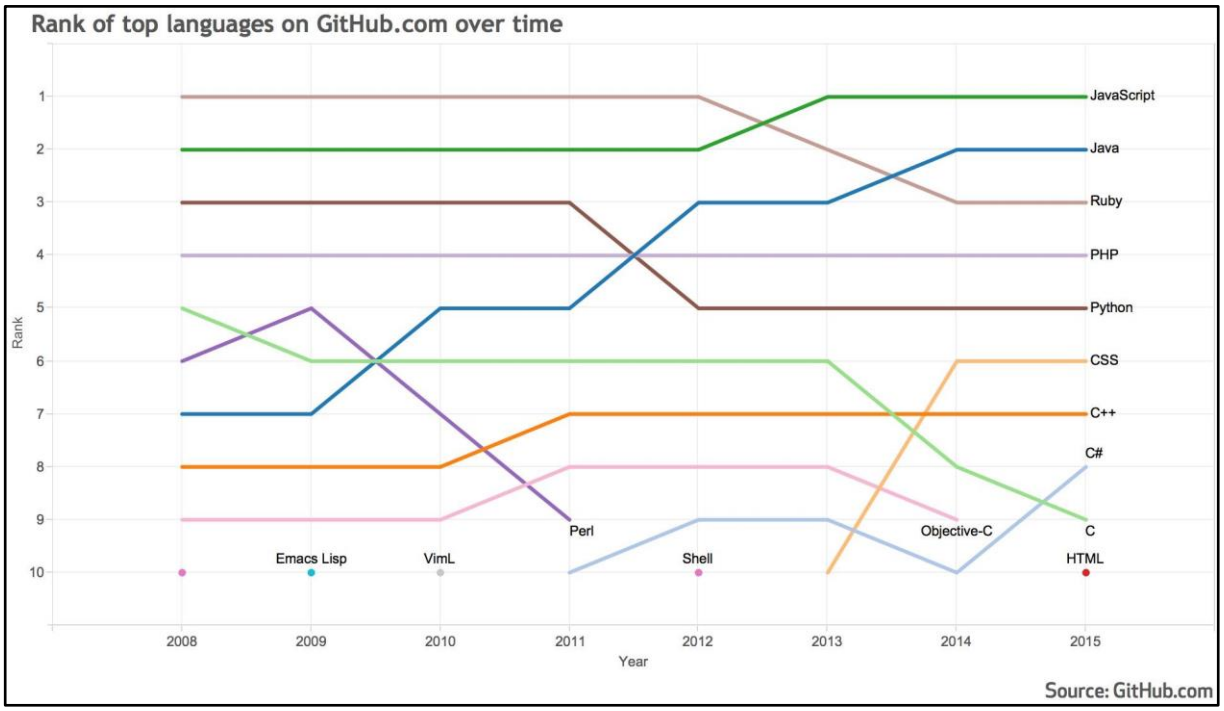
> A camada *Model* mantém as informações relacionadas ao domínio e os objetos que implementam a funcionalidade principal do sistema de informação. Na camada *View* encontram-se os objetos relacionados à apresentação, ou seja, objetos que representam a interface gráfica com o usuário. A camada *Controller* define a maneira como a interface gráfica (*View*) deve agir, a partir das informações fornecidas pelo usuário, além de atualizar as informações e o estado dos objetos da camada *Model*. Assim, a camada *Controller* atua como um mediador, tendo acesso às classes da camada *Model* para realizar as tarefas do sistema, como por exemplo, registrar os dados de um cliente fornecidos através da interface gráfica. A realização dessa tarefa ocorre com a camada de interface gráfica com o usuário (*View*) coletando os dados do usuário e repassando-os para a camada *Controller* que, com os dados fornecidos, instancia um objeto da classe Cliente e persiste as informações desse objeto no banco de dados. (2008, p.302).

Em outras palavras, o *Controller* é a ligação entre a *View* e o *Model*. Ele recebe eventos da *View* e atua sobre o *Model*, ou recebe notificações do *Model* e atua sobre a View.

3 FRAMEWORKS DE FRONT-END

JavaScript é uma linguagem que está sendo muito usada recentemente e, como dito antes, ela é de extrema importância para desenvolvimento web. Em 2015, Alyson LA, integrante da equipe do *GitHub*, mostra que JavaScript era a linguagem mais popular naquele ano, considerando linguagens utilizadas em repositórios públicos e privados. A Figura 1 aponta a popularidade desta linguagem entre 2008 e 2015.

Figura 1 - Popularidade da Linguagem JavaScript (2008 a 2015)



Fonte: GitHub

Pensando nessa popularidade e no foco do nosso trabalho que é front-end do desenvolvimento web, escolhemos estudar os frameworks de JavaScript. A princípio, observamos que existem muitas opções de frameworks de javascript: Meteor, Ember.js, Angular.js, Angular 2, Backbone.js e Vue.js. Neste capítulo, apresentaremos uma breve síntese de como se configura cada um desses frameworks a fim de atender ao nosso objetivo específico: fazer um levantamento dos principais frameworks utilizados recentemente e a escolha de dois deles para avaliação.

3.1 METEOR

Meteor é um framework para desenvolver aplicativos web e mobile. Ele é fullstack, isso significa que ele abrange as atividades do front-end e do back-end, e oferece uma solução toda em JavaScript. Sua filosofia aposta no simples, "Sendo simples você se torna produtivo" (CAVALCANTE, 2015). Segundo o site Tableless, o meteor tem muitas features interessantes que destacam o framework em comparação com outros, como por exemplo sua Reatividade (*Reactivity*). Na documentação da API, a reatividade é explicada de forma resumida:

> Meteor has a simple dependency tracking system which allows it to automatically rerun templates and other computations whenever Session variables, database queries, and other data sources change. Unlike most other systems, you don't have to manually declare these dependencies — it "just works". The mechanism is simple and efficient. When you call a function that supports reactive updates (such as a database query), it automatically saves the current Computation object, if any (representing, for example, the current template being rendered). Later, when the data changes, the function can "invalidate" the Computation, causing it to rerun (rerendering the template).[4] (METEOR)

Nas informações disponibilizadas pelo site do Meteor, percebemos que este framework dá muita importância ao seu time de desenvolvimento: "*Of all our accomplishments at Meteor, what I'm most proud of is the team. It's a privilege and an honor to work with such a talented and caring group of people*[5]." (Nick Martin). Ainda no site é oferecida área especial para explicar as atividades e funções de cada posição da empresa. Os interessados em ingressar na equipe, que até agora está com 28 membros, podem submeter sua inscrição no site oficial.

3.2 BACKBONE

Backbonejs é um framework de front-end lançado em 2010, está hoje em sua versão 1.3.3 e tem como dependência a biblioteca Underscorejs. Diferente dos

[4]Tradução livre: Meteor tem um sistema de rastreamento de dependências que permite a ele repetir a execução de templates e outras computações quando ocorrem mudanças em variáveis de sessão, consultas ao banco de dados e outras fontes de dados. Diferente da maioria de outros sistemas, não é necessário declarar essas dependências manualmente - Meteor apenas funciona. O mecanismo é simples e eficiente. Quando se chama uma função que suporta atualizações reativas (como uma consulta ao banco de dados), o Meteor automaticamente salva o objeto que representa a computação (representando, por exemplo, o template atual sendo renderizado), caso haja. Posteriormente, quando os dados mudam, a função pode "invalidar" a computação, causando a re-execução dela (renderizando novamente o template).

[5] Tradução livre: De todas as realizações do Meteor, a que eu mais sou orgulhoso é o time. É um privilégio e honra trabalhar com um grupo de pessoas talentosas e carinhosas.

outros frameworks citados no trabalho, ela estrutura as aplicações com modelos, visões e coleções. Mike Wilson indica o motivo do Backbone ter sua estrutura diferente:

> A filosofia do Backbone tem modelos responsáveis por armazenar, recuperar e transformar dados. Alguns frameworks distorceram essa intenção do MVC e tornaram o controlador responsável por transformações de dados. Na realidade, a única função do controlador é interpretar solicitações feitas pelo usuário por meio da visão e acessar as seções corretas do modelo. O modelo em si precisa compreender como lidar com os dados que recebe e como buscar a si mesmo a partir do repositório de dados. Wilson. (2013, p. 37)

Em seu site oficial (BackboneJS), podemos achar uma explicação mais completa de sua relação com o padrão MVC. No Backbone até a própria camada ‘Visão’ pode ser um tipo de controlador, pois ela também pode mandar eventos da interface, deixando o HTML ser a verdadeira camada que representa a Visão.

3.3 VUE.JS

Vue é um dos mais novos frameworks dentre os citados neste trabalho. Ele foi lançado em 2016 e já tem cerca de 20 pessoas no seu time de desenvolvimento. Em seu próprio site tem uma comparação com outros frameworks e, como diferencial, ele afirma que é projetado para que, desde o começo, seja adotado de forma incremental.

Vinicius Reis, em seu blog, destaca a simplicidade de usar o Vue: "Vue.js é extremamente amigável para devs com menos experiência, e muito flexível para devs mais ‘avançados’. É uma combinação rara de se ver." (2017). Vinícius Reis ainda destaca outros aspectos positivos do Vue, como uma grande comunidade de apoio, começando pelo fórum muito ativo no site oficial. Ainda, no site do Vue pode-se encontrar um tutorial de uso, documentação da API, guia de estilo e venda de itens da marca.

3.4 ANGULARJS

O AngularJS é um dos frameworks mais populares do JavaScript. Criado em 2009 por Miško Hevery, um integrante do Google, hoje em dia ele é oficialmente do Google (W2schools). Um dos diferenciais desse framework é a extensão dos

atributos do HTML, através de suas diretrizes, que possuem o prefixo ng- (ex: ng-app, ng-init, ng-model). Além disso, ele possui um padrão MV (Model-View-Whatever).

> AngularJS works on MV*, short for Model-View-*Whatever*. The *Whatever* is AngularJS's way of telling that you may create any kind of linking between the Model and the View here.Unlike other frameworks in any programming language, where MVC, the three separate components, each one has to be written and then connected by the programmer, AngularJS helps the programmer by asking him/her to just create these and everything else will be taken care of by AngularJS. [6](Repositório do AngularJS no GitHub)

Ou seja, o AngularJS permite vários arranjos entre o modelo e a visão, possibilitando maior suporte ao programador. Este apenas cria os componentes e cabe ao framework Angular JS resolver os demais problemas.

3.5 ANGULAR

Apesar de ter o nome parecido com AngularJS e ter a mesma equipe, eles são frameworks distintos. O AngularJs., muito chamado de de Angular1.xS é a primeira versão do Angular. Em setembro de 2016, o Google lançou o Angular, (muito chamado de Angular 2), um framework diferente que utiliza TypeScript e ainda tem seu próprio Command-line Interface, o Angular-CLI. Em um post no MATERA Systems, Otávio Prado (2017) comenta que o código do Angular é mais simples e robusto do que o AngularJS, o que gera uma curva de aprendizado mais curta, fazendo com que os desenvolvedores se voltem para as regras de negócio.

O Angular tem uma documentação detalhada, em seu site, e um tutorial intuitivo, ambos bem fundamentados. Em seu site, podemos ver a seguinte frase: "*We believe that writing beautiful apps should be joyful and fun. We're building a platform for the future."*[7] O Angular está agora em sua versão 5.0.0 e o conteúdo da

[6] Tradução livre: AngularJS trabalha com MV*, abreviatura de "Modelo-Visão-Tanto Faz. Esse Tanto Faz é o jeito do AngularJS dizer que nele você pode criar qualquer tipo de ligação entre o modelo e a visão. Diferente dos outros frameworks em qualquer linguagem de programação, onde o MVC, os 3 componentes separados, cada um deve ser escrito e depois ligados pelo programador, AngularJS ajuda o programador pedindo a ele para apenas criar esses componentes e todo o resto será resolvido pelo AngularJS.

[7] Tradução livre: Nós acreditamos que escrever aplicações bonitas deveria ser alegre e divertido. Nós estamos construindo uma plataforma para o futuro.

mensagem apresentada, no seu site, aponta que este framework pretende unir inovação, prazer e tecnologia nas aplicações que subsidia.

3.6 EMBER.JS

O Ember.js é, conforme a descrição na página oficial do frameworks, “A framework for creating ambitious web applications”[8] (Ember). De graça e com código aberto, ele é usado em várias empresas como Microsoft, Code School, Netflix e Linkedin. Segundo Duarte “tem como principal objetivo ajudar os desenvolvedores que pretendem construir aplicações web escaláveis de uma única página.” (2015, p. 33). O Ember.Js é muito chamado pela comunidade apenas como “Ember”. Por questões de facilidade, iremos nos referir a ele neste projeto como apenas “Ember”.

Criado em 2011 por Yehuda Katz, o Ember passou muito tempo para ser reconhecido, até que em 2015 teve sua versão 2.0 com várias melhorias, o que pode ser tido um dos motivos de seu crescimento (FELIZARDO, 2017). Hoje, o Ember está na versão 2.16.0, com 15 pessoas trabalhando em sua equipe principal (*core time*) e 23 pessoas no "*subteam*".

Em seu site podemos encontrar a documentação bem detalhada deste framework. Além disso, podemos encontrar, também, três tipos de tutorial: "*Quick start*", "*Tutorial*" e "*Guides*". O *Quick Start* é uma breve introdução sobre o ember que ensina a instalar o framework e como funciona sua estrutura. O *Tutorial* serve para ajudar às pessoas a construir uma aplicação rápida e fácil, desde o seu início até o *deploy*. Por último, o *Guides* é feito para pessoas que querem se aprofundar nos detalhes dos componentes oferecidos pelo Ember.

[8]Tradução livre: Framework para criar aplicações web ambiciosas

4 ESCOLHA DE CRITÉRIOS PARA COMPARAÇÃO DOS FRAMEWORKS

Para escolha dos critérios a serem usados na comparação entre os frameworks, realizamos um levantamento de trabalhos de pesquisa que tiveram como foco essa ação. Com base nesse levantamento, identificamos quais critérios foram usados. Assim, a maioria dos critérios foram retirados de trabalho anteriores de comparação entre frameworks. Para cada critério apresentado, faremos referência ao trabalho que o originou.

Em seguida, agrupamos os critérios selecionados e criamos novos critérios com base na junção de outros. Por fim, com base nos agrupamentos realizados, foram consideradas duas categorias para seleção, organização e definição dos critérios: Critérios Sociais e Critérios Técnicos.

4.1 CRITÉRIOS SOCIAIS

Critérios sociais são aqueles que avaliam o impacto do uso do framework na dimensão social. Não estão diretamente relacionados ao código ou estrutura do framework. São eles: Popularidade, Suporte Comunitário, Publicações e Utilização no Mercado.

4.1.1 Popularidade

Este critério tem como objetivo principal analisar o quão popular o framework é. O trabalho de Rebeca Franco (2011) definiu que a popularidade do frameworks poderia ser medida através do *Google Trends*, verificando a frequência de pesquisa de termos. Além disso, iremos comparar a quantidade de estrelas e visualizações (*Watch*) que os dois possuem no GitHub.

O *Google Trends* é uma ótima ferramenta para identificar tendências. Nele, podemos ver o quão um certo termo é popular. Nele, os usuários ainda podem comprar dois termos e aplicar filtros, como o de localização e tempo. (Google Trends).

O *GitHub* é uma plataforma para desenvolvedores hospedarem seus projetos. Nele, a maioria dos projetos de código aberto, incluindo frameworks, estão

hospedados. Esta plataforma é muito popular entre os desenvolvedores, é de extrema importância medir a popularidade dos frameworks por lá. (GitHub).

4.1.2 Suporte comunitário

O suporte comunitário é um dos principais critérios sociais, pois é na comunidade que os desenvolvedores que estão aprendendo uma nova tecnologia vão procurar ajuda, conversar sobre o tema com outros programadores que já passaram por alguma dificuldade ou estão tendo o mesmo problema. Mediremos esse critério através de dados estatísticos no site do StackOverFlow.

O StackOverFlow é um site onde desenvolvedores podem tirar dúvidas. Segundo o site oficial: “Um site de perguntas e respostas para programadores profissionais e entusiastas.” (StackOverFlow).

4.1.3 Publicações

Este critério envolve os trabalhos publicados em artigos, livros, teses e dissertações relacionados aos frameworks. É importante saber se existem muitos artigos publicados envolvendo os temas, pois quanto maior o número de artigos publicados em relação a um certo tema, mais esse tema foi explorado e provavelmente já deve ter suas características e resultados discutidos e validados na comunidade científica. Livros, também, são importante pois são estudos mais estruturados e atendem a um outro público da área.

O trabalho de Rebeca Franco (2011) verifica a quantidade de livros através de uma pesquisa no Google Books, um serviço do Google que oferece livros na Íntegra. (Google Books). Além dessa medida, procuraremos artigos relacionados ao temas no SciELO, no Periódico Capes e no Catálogo de Teses e Dissertações da CAPES. O SciELO (Scientific Eletronic Library Online) é

> É um modelo para publicação eletrônica cooperativa de periódicos científicos na internet. Especialmente desenvolvido para responder às necessidades da comunicação científica nos países em desenvolvimento e particularmente na América Latina e Caribe. O modelo proporciona uma solução eficiente para assegurar a visibilidade e o acesso universal a sua literatura científica, contribuindo para a superação do fenômeno conhecido como ‘ciência perdida’. O Modelo SciELO contém ainda procedimentos integrados para medir o uso e o impacto dos periódicos científicos (SCIELO).

Já o Periódico Capes configura-se como um Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), vinculado ao Ministério da Educação no Brasil. De acordo com a página de apresentação do referido portal, ele funciona como uma biblioteca virtual que contém o melhor da produção científica internacional por reunir e disponibilizar a instituições de ensino e pesquisa no Brasil um acervo de mais de 38 mil títulos com textos completos em 134 bases referenciais e 11 bases dedicadas exclusivamente à patentes. Além disso, possui livros, enciclopédias, obras de referência bem como normas técnicas, estatísticas e conteúdo audiovisual.

Por fim, o Catálogo de Teses e Dissertações da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), anteriormente denominado de Banco de Teses e Dissertações, reúne registros desde 1987 e é um sistema de busca bibliográfica de teses e dissertações produzidas no país, conforme disponível no site da CAPES. Este catálogo é alimentado diretamente pelas informações depositadas pelos Programas de Pós-graduação stricto sensu, reconhecidos pela CAPES, na Plataforma Sucupira. As informações, no catálogo, podem ser acessadas por autor, título e palavra-chave.

4.1.4 Utilização no mercado

Este critério tem como objetivo saber se os frameworks escolhidos estão sendo utilizados no mercado, pois é importante para todos os desenvolvedores saber quais tecnologias aprender para aumentar suas chances de ingresso em alguma empresa. Para este critério, utilizaremos o critério “Vagas de Emprego” do trabalho de Rebeca Franco (2011). Nele, a autora pesquisou no site da Catho as vagas que exigiam conhecimento dos frameworks. Catho é um site onde os usuários podem procurar vagas de emprego através de filtros, como Região, Salário e Perfil. (Catho).

Além da Catho, iremos pesquisar as vagas oferecidas no site LinkedIn. O Linkedin é uma rede social para utilização profissional. Nela, usuário de todo o mundo podem procurar por outros profissionais, empresas e vagas que lhe chamem atenção.

4.2 CRITÉRIOS TÉCNICOS

Critérios técnicos são aqueles relacionados ao código e estrutura do framework. São critérios que obtém respostas através ou da documentação, ou dos programadores que estão utilizando os frameworks, ou do código que foi gerado em uma aplicação. São eles: Implementação do padrão MVC, Documentação, Curva de Aprendizado, Facilidade de Efetuar Testes Automatizados, Validação dos Dados e Métricas de Casos de Uso.

4.2.1 Implementação do padrão MVC

Conforme dito, anteriormente, o MVC é um padrão de arquitetura muito utilizado em vários sistemas. Neste critério será discutido como é feita a implementação do padrão MVC nos frameworks, como eles estruturam as camadas Model, View e Controller.

4.2.2 Testes automatizados

Testes são necessários para garantir a corretude e qualidade do sistema, além de identificar de forma mais rápida os possíveis erros. Paulo C. Bernardo e Fabio Kon apresentam uma definição de testes automatizados.

> Muitos métodos ágeis, como Lean, Scrum e XP (vide referências) recomendam que todas as pessoas de um projeto (programadores, gerentes, equipes de homologação e até mesmo os clientes) trabalhem controlando a qualidade do produto todos os dias e a todo momento, pois acreditam que prevenir defeitos é mais fácil e barato que identificá-los e corrigi-los. [...] Testes automatizados são programas ou scripts simples que exercitam funcionalidades do sistema sendo testado e fazem verificações automáticas nos efeitos colaterais obtidos. (2008, p. 2)

Apesar de saber que testes automatizado são de extrema importância para o desenvolvimento de um sistema, Mosh Hamedani afirma que:

> The reality is: even though automatic tests have a lot of benefits, it doesn't fit every project and every team. For the start, your team needs to have a discipline in writing clean tests and maintaining him. If you don't work in a team like that, writing automatic tests ends up coasting you more than the value you get out of them, because you'll spend a lot of time fixing broken tests, that are hard to read and hard to understand. In those cases, it's better not to write a test at all. (2017)

O sistema proposto foi desenvolvido por uma equipe que ainda não possui uma cultura de testes consolidada na metodologia de desenvolvimento, então, por este motivo, não iremos utilizar testes automatizados em nosso projeto. Entretanto, sabendo da importância dos testes automatizados, iremos avaliar se os frameworks estudados oferecem suporte para testes. Este critério foi apresentado no trabalho de Thiago Roberto (2007).

4.2.3 Data Binding

O sistema analisado possui muitas mudanças de dados e muita visualização dos mesmos, isso significa que ele precisa de sincronia e de uma atualização rápida dos valores. Em um blog da Microsoft é proposta uma definição rápida de Data Binding:

> Data binding is the process that establishes a connection between the application UI and business logic. If the binding has the correct settings and the data provides the proper notifications, then, when the data changes its value, the elements that are bound to the data reflect changes automatically.[9] (MICROSOFT).

Ou seja, é a ligação dos dados apresentados na interface do usuário e os dados no modelo. Existem basicamente dois tipos de data binding: *One Way Data Binding* e *Two Way Data Binding*. No *One Way*, os dados do modelo são refletidos na interface, porém qualquer atualização dos dados na interface não irá afetar os dados do modelo. Ele funciona apenas como leitura. No *Two Way Data Binding*, qualquer alteração de dado, seja na interface ou no modelo, essa alteração será refletida em qualquer lugar.

Neste critério serão avaliados se os frameworks suportam o data binding, qual o tipo e qual o nível de facilidade de usar esse recurso, caso o framework suporte.

4.2.4 Dados e Formulários

Formulários são utilizados em vários sistemas, principalmente no sistema selecionado para este estudo. É importante que o framework dê suporte a validação

[9]Tradução livre: Data binding é o processo que estabelece uma conexão entre uma aplicação UI e a camada lógica. Se a ligação (binding) tem as configurações corretas e os dados oferece as notificações apropriadas, então, quando as dados mudarem de valor, os elementos que estão vinculados a esses dados refletem as mudanças automaticamente.

de dados e formulários. Este critério avalia se os frameworks dão suporte a criação e validação de dados e formulários.

4.2.5 Rotas

Neste critério são avaliados como os frameworks trabalham com rotas, se eles têm abordagens diferentes e, caso sim, quais as vantagens e desvantagens de cada. Além disso, utilizaremos o critério “Routing” do trabalho de Duarte (2015) em que classifica as rotas em “pushState History“ ou “DeepLinking”. O referido autor afirma que “Deep linking consiste em alterar o URL para conter informações específicas sobre um conteúdo para melhorar a pesquisa e indexação” (p. 41) e, também, apresenta que “[...] pushState History é responsável por alterar o URL do navegador sem iniciar nenhum pedido ao servidor” (p. 41).

5 DESENVOLVIMENTO

Temos, ainda, como objetivos específicos *a escolha de dois frameworks de front-end* e a *produção do desenvolvimento front-end de duas aplicações com estes frameworks, utilizando o Design Responsivo.* Portanto, este Capítulo contém informações sobre o sistema desenvolvido e como foi o processo de escolha dos frameworks.

5.1 SISTEMA DESENVOLVIDO

O sistema que será apresentado foi desenvolvido para uma empresa, em caráter de sigilo. Desta forma, as informações aqui apresentadas serão sucintas e direcionadas, exclusivamente, para a avaliação dos critérios escolhidos.

O sistema se inicializa na tela de **Login**, que deve conter validação de dados do formulário, como campo obrigatório, e dados inválidos. Esta tela também pode direcionar o usuário para tela **Esqueci minha senha**. A tela **Esqueci minha senha** também deve conter validação para e-mails não cadastrados. O sistema ainda tem uma tela para **Redefinição de senha** que o usuário tem acesso através de seu e-mail. Esta tela que deve conter medidor de senha e validação para senhas que não são iguais.

O sistema foi feito para três perfis: Administrador, Operador e Fornecedor. O Administrador tem acesso a todas as abas do sistema e a todas informações. Já o Operador e Fornecedor têm acesso a apenas 3 abas, e, dentro dessas abas, eles só podem ver informações relacionadas a eles próprios. O Administrador ainda pode cadastrar e gerenciar usuários e itens, já os outros perfis podem cadastrar e gerenciar usuários, mas apenas observar os itens.

Após logar, o sistema principal tem 6 abas e uma área para visualização e edição de perfil. São elas:

- **Home -** Os perfis podem acompanhar as últimas atualizações dos itens. Essas atualizações são chamadas de alertas. Os alertas possuem: nome, fornecedor, operador e quantidade.

- **Cadastros -** O Administrador pode cadastrar e gerenciar itens.

- **Repositório -** Contém uma tabela contendo os itens cadastrados e filtros de busca.

- **Importar dados -** O Administrador pode importar dados de um excel.

- **Rastreamento -** Os três perfis podem observar, em um mapa, onde estão os itens.

- **Configurações -** Esta tela contém uma grande tabela de dados sigilosos. Ela dá acesso a um formulário que o administrador pode setar algumas configurações gerais, como quantidade máxima e mínima de itens e fornecedores. Este formulário deve conter validação de dados.

- **Perfil -** Nesta tela, os usuário podem editar suas informações, cadastrar usuários e sair do sistema.

A Figura 2 representa o fluxo do sistema. Os retângulos verdes representam telas que podem ser acessadas por todos os perfis. O retângulos vermelhos representam telas que só podem ser acessadas pelo Administrador.

5.2 ESCOLHA DOS FRAMEWORKS

Conforme apontado por Tárcio Zemel (2013), é necessário que os sistemas de hoje em dia possuam um design responsivo a fim de adaptar o conteúdo de acordo com o dispositivo e suas respectivas configurações. Pensando nisso, nosso sistema possuirá design responsivo para atender as necessidade de nossos clientes.

Para o nosso sistema, primeiramente nós tínhamos escolhido o Framework CSS Bootstrap por seu estilo CSS se encaixar melhor com o nosso design e por ter os componentes necessários na aplicação. Além disso, o Bootstrap é um dos mais populares no GitHub e possui uma documentação muito bem detalhada em seu próprio site, onde podemos aprender a usar seus componentes de forma rápida e

fácil. Porém, depois, descobrimos o ng-Bootstrap para Angular e o ember-bootstrap para Ember.

O Ng-Bootstrap, que foi feito especialmente para o Angular. Segundo Sebastian Eschweiler (2015), "*Ng-Bootstrap contains a set of native Angular directives based on Bootstrap's markup and CSS. As a result no dependency on jQuery or Bootstrap's JavaScript is required.*[10]". A documentação do Ng-Bootstrap também é muito bem detalhada, e ele possui uma certa popularidade no stackoverflow e no GitHub, com um total de 528 perguntas no Stack e 4,249 estrelas no GitHub.

O Ember-Bootstrap foi feito especialmente para Ember:

> An ember-cli addon for using Twitter Bootstrap in Ember.js applications. Provides a collection of native Ember components that mimic the original Bootstrap plugins and components in an ember friendly way, replacing the need for bootstrap.js. (Ember-Bootstrap)

Apesar de possuir uma baixa popularidade no GitHub: São 326 estrelas e 22 Watches; o ember-bootstrap possui todos os componentes necessários para o desenvolvimento do sistema.

Para atender ao objetivo geral de nossa pesquisa, r*ealizar um estudo comparativo entre dois frameworks de desenvolvimento Front-End,* a princípio, foi considerada a ideia de utilizar AngularJs por ser um dos frameworks mais famosos de JavaScript. Porém, muito se diz - em diversos posts e entre profissionais da área de Ciência da Computação - que este framework, em breve, pode se tornar obsoleto. Por esse motivo, escolhemos testar uma linguagem mais nova, o Angular.

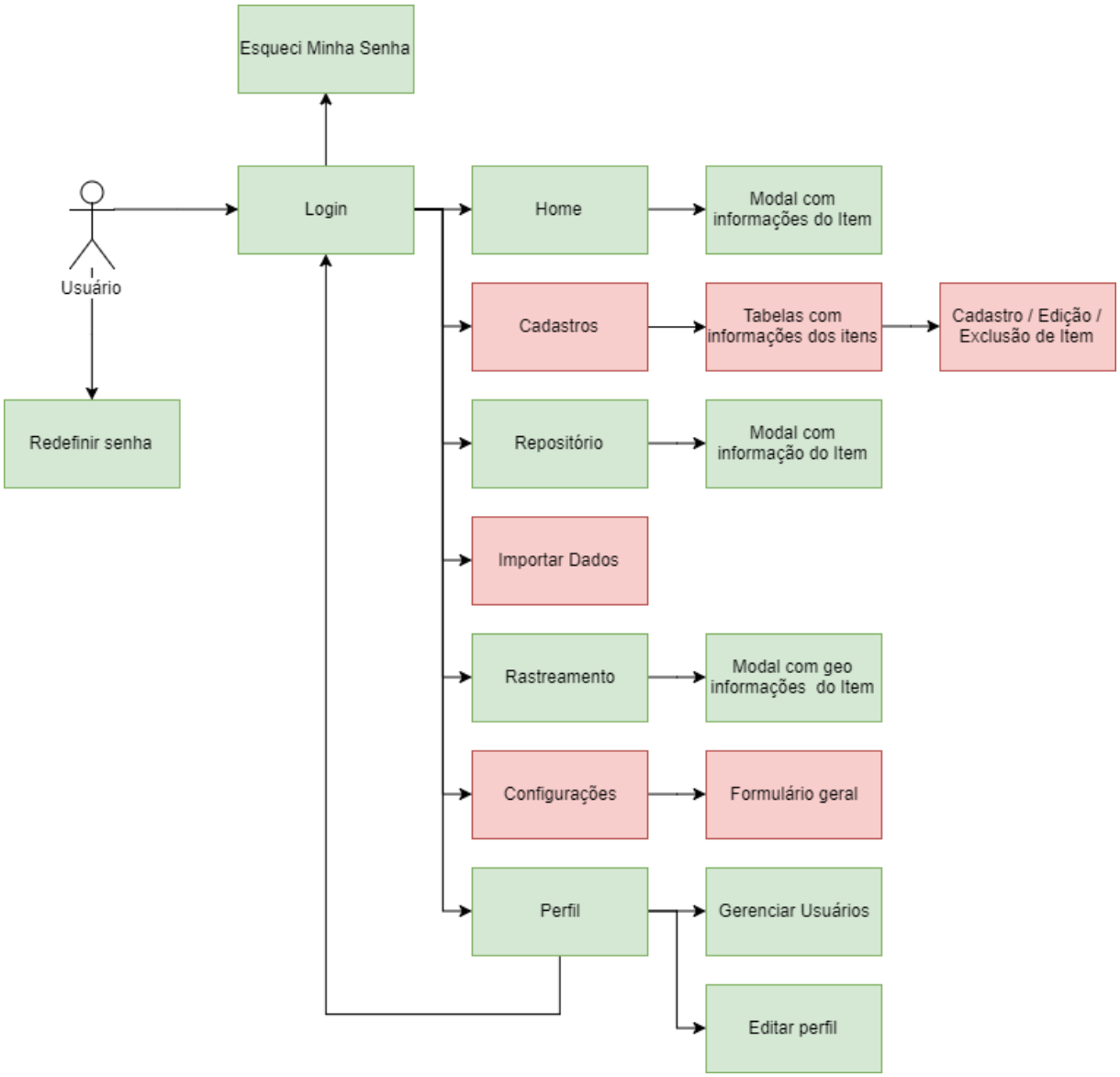
O outro framework que escolhemos foi Ember. No próprio site do Ember, ele afirma que o framework é focado em produtividade. Ele tem como aspectos: escrever menos código, gastar menos tempo com escolhas triviais e ajudar a fazer seu trabalho de forma rápida: *Ember.js is built for productivity. Designed with developer ergonomics in mind, its friendly APIs help you get your job done—fast.*[11] Escolhemos, portanto, o Ember para esse estudo por ele prometer o desenvolvimento rápido de um sistema, .

[10] Tradução livre: O Ng-Bootstrap contém um conjunto de diretrizes nativas do Angular baseadas no CSS do Bootstrap. Como resultado, não há necessidade de dependência do jQuery ou do JavaScript do Bootstrap.

[11] Tradução livre: Ember é construído para produtividade. Projetado com a ergonomia do desenvolvedor em mente, sua API amigável te ajuda a ter seu trabalho feito - de forma rápida.

Assim, para a comparação entre os frameworks, escolhemos um já consolidado no mercado, em sua versão mais avançada, e um outro que tem como prerrogativa a alta produtividade.

Figura 2 - Fluxo do sistema desenvolvido



Fonte: A autora.

6 ANÁLISE COMPARATIVA

Esta pesquisa realizou *um estudo comparativo entre dois Frameworks de desenvolvimento Front-End* e, *como* prova de conceito, *comparamos os principais recursos dos frameworks, destacando vantagens e desvantagens.* A seguir, apresentamos a análise dos critérios sociais e técnicos entre os *Frameworks Angular e Ember.*

6.1 ANÁLISE DOS CRITÉRIOS SOCIAIS

Os critérios sociais analisados nos dois frameworks foram: popularidade, suporte comunitário, publicações e utilização no mercado.

6.1.1 POPULARIDADE

Para investigar a popularidade de cada um desses frameworks foi utilizada a ferramenta Google Trends (disponível no endereço http://www.google.com/trends). Os números do Google Trends funcionam da seguinte maneira:

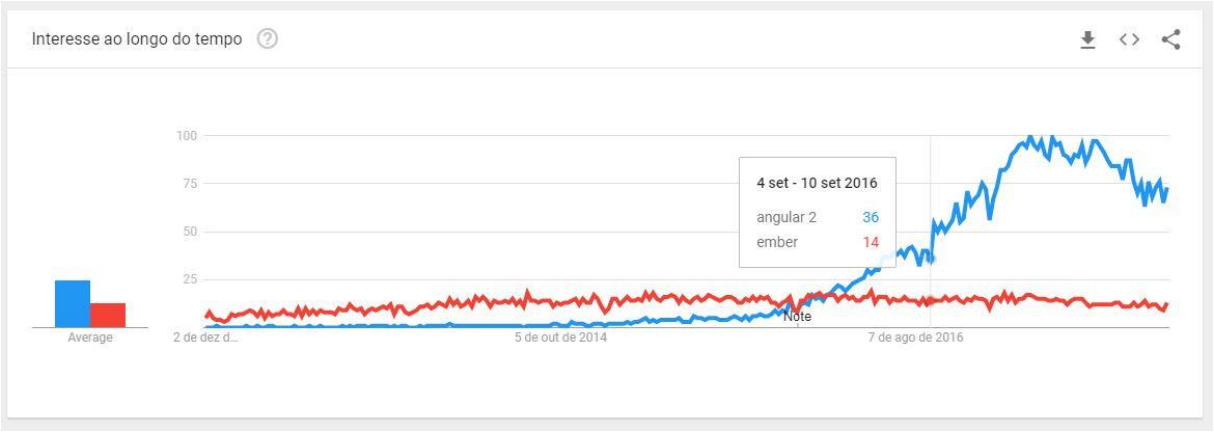
> Os números representam o interesse de pesquisa relativo ao ponto mais alto no gráfico de uma determinada região em um dado período. Um valor de 100 é o pico de popularidade de um termo. Um valor de 50 significa que o termo teve metade da popularidade. Da mesma forma, uma pontuação de 0 significa que o termo teve menos de 1% da popularidade que o pico. (Google)

Nesta ferramenta, foram pesquisados os temas “Angular 2” e “Ember” e aplicados os filtros:

- Geolocalização: Todo o mundo;
- Tempo: Nos últimos 5 anos;
- Categoria: Programação;
- Pesquisa na Web: todas as categorias.

O resultado da busca pode ser analisado na Figura 3 a seguir.

Figura 3 – Gráfico com a popularidade dos frameworks Ember e Angular no Google Trends



Fonte: Google Trends.

Podemos ver que “Angular 2” é um termo consideravelmente mais pesquisado do que ember. O framework ember mantém uma estabilidade em relação à pesquisa do termo, sem aumentar nem diminuir de forma drástica. Já o Angular teve um crescimento considerável em setembro de 2016, provavelmente porque foi o mês de seu lançamento. Entretanto, nos últimos meses, ele vem caindo um pouco.

Acessando os repositórios dos frameworks no GitHub podemos observar que que o Angular tem mais estrelas do que Ember. Além disso, ele tem duas vezes mais visualizações, conforme apresentamos na Tabela 2.

Tabela 2 – Tabela com a popularidade dos frameworks Ember e Angular no GitHub

Critério/Framework		**Angular**	**Ember**
Popularidade	GitHub (Stars)	30,631	18,468
	GitHub (Watch)	2,797	1,054

Fonte: A autora.

Com estes resultados, podemos concluir que tanto no GitHub quanto no Google Trends, o Angular é mais famoso do que Ember.

6.1.2 SUPORTE COMUNITÁRIO

As respostas para este critério foram encontradas de duas formas. A primeira pesquisa foi feita no site mundial do StackOverFlow. Já a segunda pesquisa foi realizada no StackOverFlow em Português. O objetivo era saber como é o suporte comunitário a esses frameworks de forma geral e como é o suporte aqui no Brasil.

No site mundial do StackOverFlow, pesquisando sobre “Angular 2” e “Ember”, podemos encontrar 60.655 perguntas em Angular e 30.624 em Ember. O site ainda oferece vários filtros de pesquisa, entre eles, as publicações que não obtiveram respostas. Angular possui 10.766 perguntas sem respostas, e Ember possui 2.474 publicações sem respostas.

No site em português do StackOverFlow, podemos encontrar 359 publicações em Angular e 26 publicações envolvendo Ember. Das 359 publicações sobre Angular, 84 ainda não foram respondidas. Enquanto que das 26 publicações do Ember, apenas três não foram respondidas.

Nestes sites, ainda, podemos encontrar várias tags sobre temas. Tags funcionam para agrupar perguntas de acordo com uma temática. Por exemplo, Angular podem possuir as seguintes tags: Angular2-http, Angular2-estrutura, Angular2-rotas. A tag “Angular2-rotas” irá listar todas as perguntas que foram relacionadas a como funcionam as rotas em Angular. Da mesma forma, Ember pode possuir as tags: Emberjs, Ember-http, Ember-rotas.

No site mundial, Angular possui 32 tags, enquanto Ember possui 15 tags. Já no site em português, os números são menores. Angular possui cinco tags, enquanto ember possui apenas três. A seguir apresentamos a síntese dos resultados obtidos.

Tabela 3 – Tabela com perguntas sobre os frameworks Ember e Angular no StackOverFlow

Critério/Framework		**Angular**	**Ember**
Suporte Comunitário (StackOverFlow)	Pesquisa “Angular 2” e “Ember”	60,655	30,624
	Sem resposta	10,766	2,474
	Tags relacionadas	32	15

Suporte Comunitário (StackOverFlow em Português)	Pesquisa “Angular 2” e “Ember”	359	26
	Sem resposta	84	3
	Tags relacionadas	5	2

Fonte: A autora

Podemos concluir que as pessoas possuem mais dúvidas em Angular do que em Ember. Por um lado, isso pode estar ligado ao fato de que Angular é mais popular do que Ember, representando que tem mais pessoas usando este frameworks e, consequentemente, tirando mais dúvidas. Por outro lado, isso também pode representar uma falha na documentação do framework, que gera muitas dúvidas e, então, as pessoas precisam de suporte.

Angular possui cerca de 17,749% de perguntas não respondidas, enquanto Ember possui 8,078%. Isso significa uma baixa porcentagem de perguntas não respondidas, o que pode representar comunidades ativas.

Os números no StackOverFlow, em português, são menores pelo fato de ser um grupo menor e esta não ser uma língua universal. Mas, ainda podemos observar que, em português, o número de perguntas em Angular é muito maior do que as perguntas em Ember.

6.1.3 PUBLICAÇÕES

Para realizar a pesquisa sobre as publicações, foi preciso restringir os termos, pois os termos “ember” e “angular” pareceram muito comuns (com vários trabalhos que não tinham relação com nossa temática) o que se tornou inviável filtrar os resultados manualmente.

- **Pesquisa no SciELO**

A busca de termos no banco de dados do SciELO foi realizada tendo como filtros (i) o método “integrada”, (ii) a localização Brasil e (iii) foram considerados artigos e livros publicados no SciELO. Além disso, pesquisamos em todos os

idiomas. Este levantamento foi realizado com base nos seguintes critérios para uso das palavras de busca:

- **Para o framework Ember:** (Ember JS) | (Ember.js)
- **Para o framework Angular:** (Framework Angular 2) | (Angular 2).

No levantamento realizado neste banco de dados, não foram encontrados artigos que mencionaram as palavras mencionadas.

❖ **Pesquisa no Portal de Periódicos da CAPES**

A análise dos artigos disponíveis no Portal de Periódicos da Capes foi realizada tendo como filtro: (i) a procura por título e assunto, (ii) qualquer idioma, (iii) apenas periódicos avaliados por pares, (iv) na subárea temática Ciência da Computação. Utilizamos os seguintes critérios para uso das palavras de busca:

- **Para o framework Ember:** (Framework Ember) | (Ember JS) | (Ember.js)
- **Para o framework Angular:** (Framework **e** Angular 2) | (Angular 2).

Os resultados indicam uma ocorrência de dez artigos relacionados ao Framework Ember: sete destes usam a palavra “ember.js”, dois usaram a palavra “Ember JS” e um usou a palavra “framework ember”.

Em relação ao Framework Angular, observamos que há um alto índice de artigos no Portal de Periódicos Capes. Foram identificados 570 artigos que citam a palavra “Angular 2”. Para refinar nossos dados, utilizamos simultaneamente os descritores “Angular 2” e “framework”. Nesta última busca, encontramos 91 artigos. Como esse trabalho não se propôs a realizar a leitura de todos os artigos, diante dos dados obtidos consideramos que o framework Angular é o mais citado em artigos no portal de periódicos Capes.

❖ **Pesquisa no Catálogo de Teses e Dissertações da Capes**

Realizamos a busca no Catálogo de Teses e Dissertações da Capes tendo como filtro: (i) a procura pelos termos em destaque no título, resumo e palavras

chaves, (ii) o período de tempo de 2011 a 2017, (iii) a *Grande Área de conhecimento* como Ciências Exatas e da Natureza, (iv) a *Área de Conhecimento* como Ciência da Computação e (v) a *Área de Avaliação* como Ciência da Computação. Utilizamos os seguintes critérios para uso das palavras de busca:

- **Para Framework Ember:** (Framework Ember) | (Ember JS) | (Ember.js)
- **Para Framework Angular:** (Framework Angular 2).

Para os descritores acima não foram encontradas dissertações e teses. O fato de uma dissertação e uma tese ter o tempo de produção, respectivamente, de dois e quatro anos pode ter gerado a ausência de publicações neste suporte. O framework Ember foi lançado em 2011 e o framework Angular em 2016, sendo, portanto, ainda muito recente as duas temáticas no meio acadêmico e nas pesquisas científicas.

❖ **Pesquisa no Google Books**

No Google Books, utilizamos os seguintes critérios para uso das palavras de busca:

- **Para Framework Ember:** (Ember.js)
- **Para Framework Angular:** (Angular 2)

Os termos foram pesquisados com aspas para forçar uma procurar pelos termos específicos. Foi aplicado o filtro “Livros” como documento. Como resultados, obtivemos 11.900 resultados para Angular e 2.150 resultados para Ember. Segundo os resultados do Google Books, existe um número considerável de livros disponíveis sobre ambos frameworks; porém, em uma rápida análise sobre o título dos livros, chegamos a conclusão de que muitos desses livros não tem ligação com os frameworks.

Para refinar os resultados, utilizamos o filtro de tempo para a busca: de 2011 a 2017 para “Ember.js” e de 2014 a 2017 para “Angular 2”, considerando o período de lançamento de cada framework. Nesta última tentativa, encontramos 16 livros que estão relacionados ao Framework Ember e 26 livros relacionados ao Angular. Concluímos, portanto, que a temática do “Angular 2” é mais frequente no Google Books do que “Ember.js”.

Em linhas gerais, os resultados encontrados quanto à publicação estão dispostos na Tabela 4.

Tabela 4 – Tabela com publicações sobre os frameworks Ember e Angular

Critério/Framework		Angular	Ember
Publicações (SciELO)	**Artigos**	0	0
Publicações (Periódico CAPES)	**Artigos**	91	10
Publicações (Catálogo de Dissertações e Teses da CAPES)	**Tese**	0	0
	Dissertação	0	0
Publicações (Google Books)	**Livros**	26	16

Fonte: A autora

6.1.4 UTILIZAÇÃO NO MERCADO

- **Pesquisa no Catho**

Utilizamos os seguintes critérios para uso das palavras de busca:

- **Para o framework Ember:** (Ember.js)
- **Para o framework Angular:** (Angular 2).

A pesquisa com a palavra “Angular 2” apontou 34 vagas de emprego; contudo, uma análise refinada dos dados indicou que apenas 15 estavam relacionadas de fato ao Framework Angular. O resto das vagas tinham como pré-requisito apenas o AngularJs, que aparecia nos resultados por ter o nome muito parecido com Angular. Já para o framework Ember.js, foi encontrada apenas uma vaga para a palavra “Ember.js”.

❖ **Pesquisa no LinkedIn**

Para analisar a utilização dos frameworks no site Linkedin, utilizamos o filtro com objetivo “Vagas”, “Tecnologia da Informação” como função e “No último mês” como data do anúncio. Utilizamos os seguintes critérios para uso das palavras de busca:

- **Para o framework Ember:** (Ember)
- **Para o framework Angular:** (Angular 2).

Pesquisando “Angular 2” com essas configurações e adicionando o filtro de localização “Brasil”, obtivemos 78 resultados. Já pesquisando “Ember” com o filtro de localização “Brasil”, obtivemos 49 resultados. Utilizando o filtro “Mundialmente”, obtivemos 2143 resultados para Angular, e 2.228 resultados para Ember. A Tabela 5, a seguir, sintetiza todos os resultados.

Tabela 5 – Tabela com a utilização no mercado sobre os frameworks Ember.JS e Angular 2 no Catho e no LinkedIn

Critério/Framework		Angular	Ember
Utilização no mercado	Catho	15	1
	LinkedIn (Brasil)	78	49
	LinkedIn (Mundo)	2.143	2.228

Fonte: A autora

Com os dados obtidos, podemos concluir que existem mais vagas que tem como pré-requisitos o framework Angular do que o Ember no site da Catho. Porém, no linkedin, os dois possuem quase a mesma quantidade de vagas. Concluímos que os dois frameworks são muito utilizados no mercado, e que é importante aprendê-los pois pode ser um diferencial em uma candidatura. A seguir, apresentamos a Tabela 6 que resume de forma comparativa todas as informações coletadas quanto aos critérios sociais.

Tabela 6 – Tabela comparativa dos critérios sociais dos frameworks Ember e Angular

Critério/Framework		Angular	Ember
Popularidade	GitHub (Stars)	Mais (30,631)	Menos (18,468)
	GitHub (Watch)	Mais (2,797)	Menos (1,054)
	Google Trends (Últimos 5 anos)	Muito interesse	Pouco interesse
Suporte Comunitário (StackOverFlow)	Pesquisa "Angular 2" e "Ember"	Mais perguntas (60,655)	Menos perguntas (30,624)
	Sem resposta	Mais perguntas sem resposta (10,766)	Menos perguntas sem resposta (2,474)
	Tags relacionadas	Maior quantidade (32)	Menor quantidade (15)
Suporte Comunitário (StackOverFlow em Português)	Pesquisa "Angular 2" e "Ember"	Mais perguntas (359)	Menos perguntas (26)
	Sem resposta	Mais perguntas sem resposta (84)	Menos perguntas sem resposta (3)
	Tags relacionadas	Poucas tags (5)	Poucas tags (2)
Publicações (SciELO)	Artigos	0	0
Publicações (Periódico CAPES)	Artigos	Mais (91)	Menos (10)
Publicações (Catálogo de Dissertações e Teses da CAPES)	Teses	0	0
	Dissertação	0	0
Publicações (Google Books)	Livros	Mais (26)	Menos (16)
Utilização no mercado	Catho	Mais (15)	Menos (1)
	LinkedIn (Brasil)	Mais (78)	Menos (46)
	LinkedIn (Mundo)	Menos (2.143)	Mais (2.226)

Fonte: A autora

6.2 ANÁLISE DOS CRITÉRIOS TÉCNICOS

Neste Capítulo serão apresentadas as análise dos critérios técnicos dos Frameworks Angular e Ember. Como foi dito no Capítulo 4, a análise dos critério técnico é obtida através de da documentação dos frameworks, dos programadores que estão utilizando os frameworks, ou do código que foi gerado em uma aplicação. Para análise desses critérios, utilizaremos a documentação oficial oferecida nos sites do Angular e do Ember, o desenvolvimento do sistema proposto e a percepção analítica da autora.

6.2.1 Implementação do Padrão MVC

Para entender como funciona o padrão MVC em ambos frameworks, primeiro precisamos compreender como estão estruturadas suas arquiteturas.

- **Angular**

O Angular possui várias características e serviços importantes para seu desenvolvimento, os mais importantes entre eles são; Módulos, Componentes, Templates, Diretivas e Serviços. A seguir, uma breve descrição de cada serviço:

- **Módulos:** Servem para agrupar vários Componentes de acordo com a estrutura da aplicação. Uma aplicação tem obrigatoriamente pelo menos um Módulo em sua estrutura.

- **Componentes:** Um componente é um arquivo que controla um estado de uma View. Uma aplicação em Angular é composta por vários Componentes.

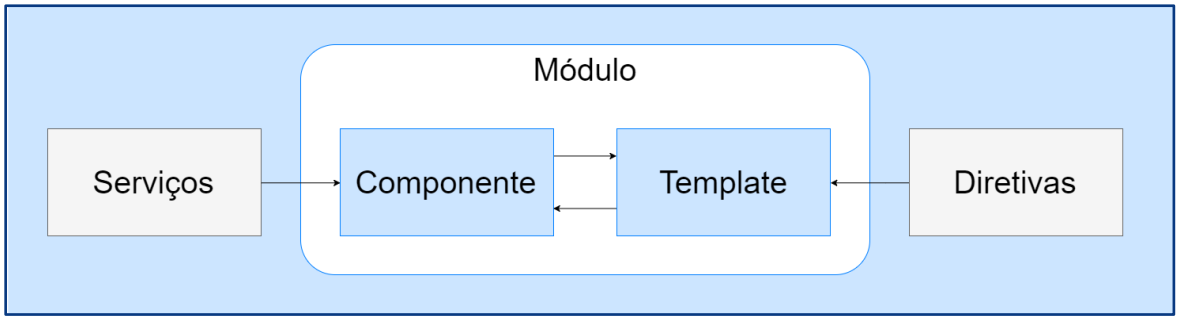
- **Templates:** Renderiza a View do Componente. Ela é composta por HTML, alguns elementos do Template do Angular, e representa como o componente será renderizado (Angular).

- **Diretivas:** Segundo o site oficial, são componentes sem um template. O usuário pode criar suas próprias diretivas. (Angular)

- **Serviços:** Serviços podem ser valores, funções ou recursos de uma aplicação precisa. Os componentes consomem os serviços em uma aplicação, pois os componentes devem ser classes “limpas” que não possuem muito código e deixam o trabalho para os serviços.

A Figura 4 simula a estrutura de uma aplicação utilizando os componentes citados acima:

Figura 4 - Estrutura básica de uma aplicação em Angular.



Fonte: A autora

Existem outros componentes importantes para serem estudados em Angular, porém vamos nos restringir aos conceitos básicos para entender a arquitetura do Framework.

❖ **Ember**

O Ember possui vários blocos importantes para seu desenvolvimento, são eles: Modelos, Template, Componentes, Rotas e Controladores. A seguir, uma breve descrição de cada um.

- **Modelos (Models):** São objetos que representam os dados que a aplicação irá apresentar para o usuário (Ember).

- **Templates:** Templates são arquivos .hbs (Handlebars) que possuem código HTML com chaves duplas {{}} que na maioria das vezes representam o modelo relacionado àquele template. Através dos templates podemos estruturar como a interface irá ficar.

- **Componentes (Components):** De acordo com o site oficial, Componentes definem como as interfaces irão funcionar.

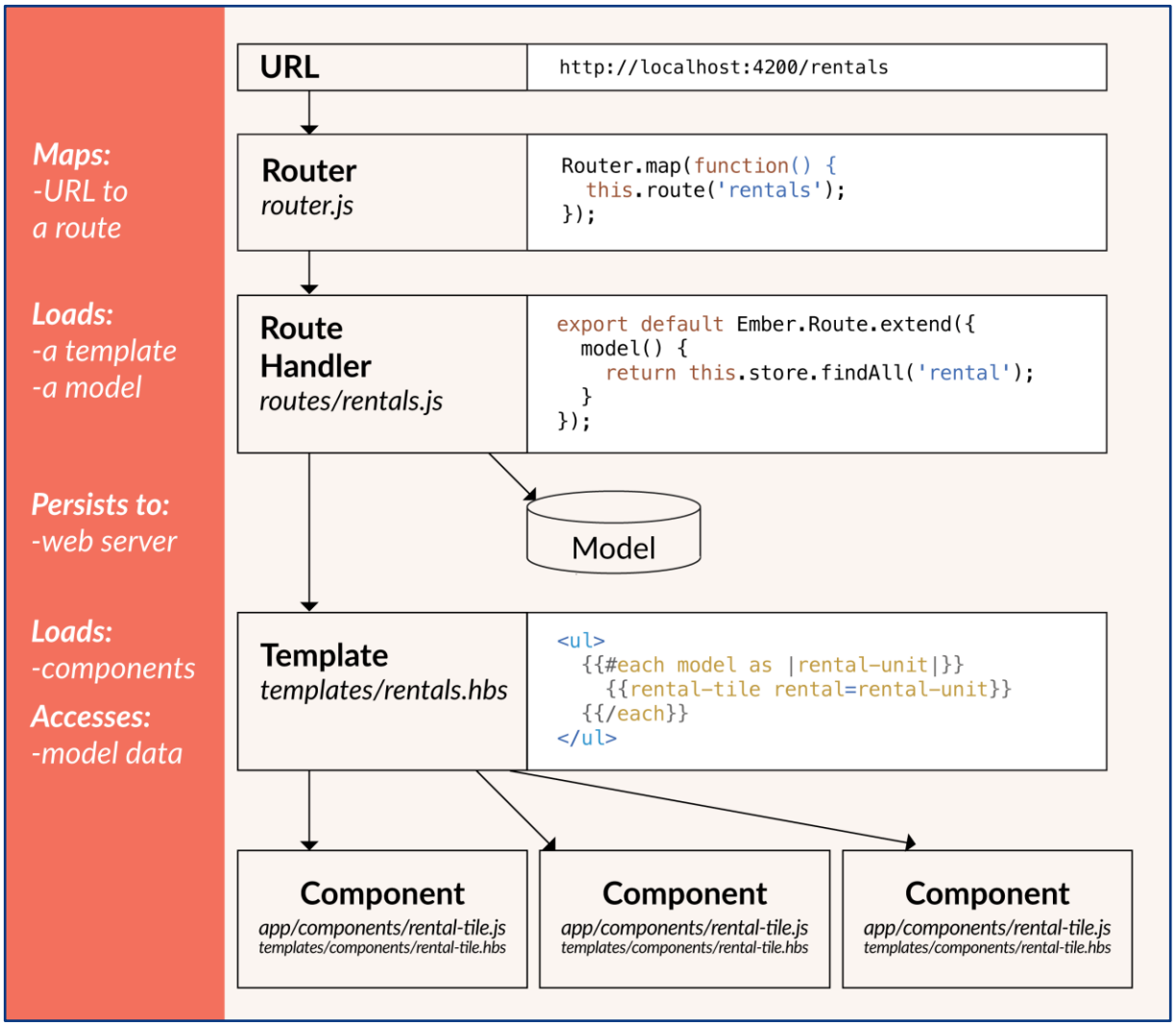
- **Rotas (Routes):** Especificam a URL que irá representar um certo Template. Elas, também, especificam quais dados (Models) estão conectados àquele Template.

- **Controladores (Controllers) :** O tutorial do Ember em seu site oficial afirma que os Controladores têm um comportamento igual ao de um Componente especializado. "*The controller receives a single property from the Route – model – which is the return value of the Route's model() method*.[12]" (Ember). Os Controllers são parte do Ember, porém eles são usados em grande parte para quem quer customizar o Controlador. Como eles mesmo dizem em seu site, um módulo Controlador já é gerado automaticamente, mesmo se o usuário não o criar explicitamente. (Ember)

A Figura 5 representa como funciona a estrutura de uma aplicação em Ember:

[12] Tradução livre: O controlador recebe uma única propriedade do Route - Modelo - que é o valor retornado do método model() do Route.

Figura 5 - Estrutura básica de uma aplicação em Ember



Fonte: https://guides.emberjs.com/v2.17.0/getting-started/core-concepts/

A Figura 5 mostra como as Rotas são responsáveis por carregar os dados em seu respectivo template, e como os Componentes são responsáveis pelo comportamento dos Templates.

- **Padrão MVC (*Model-View-Controller*)**

No Angular, a **View** é composta pelos Templates, enquanto o Componente se encaixa na camada **Controller**, pois ele controla a View. Porém, como foi dito na descrição do serviço, ele não deve conter a lógica do negócio em si, a lógica deve ser feita nos serviços, deixando os Serviços na camada do **Controller**. A camada

Model é composta pelos Serviços, que podem oferecer dados através de um Sistema Web, armazenamento local ou até uma fonte de dados simulada. (Angular)

No Ember, existe uma certa confusão quando são definidas as camadas do MVC. A **View** pode ser representada através dos Templates e das Rotas (Routes). Porém, essas mesmas Rotas, através do Route Handler, também podem ser consideradas como parte da camada **Controller**, pois são os Route Handlers que fazem a ligação a camada View e a Model. Além da Route Handle, a camada **Controller** também pode ser representada através dos Controladores (Controllers) e dos Componentes (Components), pois eles são responsáveis pelo comportamento da Interface. Por fim, a camada **Model** pode ser representada através dos Modelos (Models).

Considerando os dados apresentados e as análises feitas, podemos concluir que ambos Frameworks implementam o padrão MVC.

6.2.2 Facilidade de Efetuar Testes Automatizados

Em Angular, testes podem ser realizados através de 4 tecnologias: Jasmine, Angular Testing Utilities, Karma e Protractor. Os arquivos de testes são criados de forma automática junto com os componentes, e ficam na pasta que os componentes ficam. Da mesma forma, um teste de módulo fica junto do módulo criado. Ember oferece em seu site oficial um bom tutorial para testes automatizados. A cada novo componente oferecido em seu tutorial, ele explica como o programador pode realizar testes. Todos os arquivos de testes ficam juntos em uma pasta “Tests”.

Ambos frameworks possuem uma documentação muito bem detalhada de como utilizar esses recursos, e ambos oferecem suporte para os três tipos de testes automatizados: Teste unitário, Teste de Integração e Teste End-to-End.

6.2.3 Data Binding

Como foi dito no Capítulo 4, Data Binding é importante para estabelecer uma conexão entre os dados que estão sendo guardados no sistema e os dados que estão sendo escritos na interface pelo usuário. De extrema importância para formulários e representação dos dados, existem dois tipos de ligações: Two-Way Data Binding e o One-Way Data Binding.

❖ **Angular**

O Angular possui todos os tipos de *Data Binding*. O *one-way data binding* na direção *Source -> View* pode ser usado através da notação **{{valor}}** ou da notação **[propriedade] = "valor"**. *O one-way data binding* na direção *View -> Source* pode ser usado através da notação **(evento)="manipulação"** (Angular). A seguir, uma representação do uso do *one-way data binding* no sistema desenvolvido.

Figura 6 - Uso do One-Way Data Binding em Angular no sistema criado

```
22    <tr *ngFor="let user of data.data | paginate: {   id:'user',
23                  itemsPerPage: data.meta.limit,
24                  currentPage: data.meta.page,
25                  totalItems: data.meta.total_docs}">
26      <td>{{user.company}}</td>
27      <td>{{user.url}}</td>
28      <td>{{user.port}}</td>
29      <td class="teste">
30        <button class="editar" [routerLink]="['editar/', user._id]">Editar</button>
31        <button class="excluir" (click)="removeUser(user)" >Excluir</button>
```

Fonte: a autora

Na Figura 6, nas linhas 26, 27 e 28 foram utilizados o *One-Way Data Binding* na direção *Source -> View,* onde está sendo apresentado na tela as características do usuário: {{user.company}} imprime a companhia daquele usário. Na linha 31, podemos observar o *one-way data binding* na direção oposta. Quando o usuário clicar nesse botão, o evento (click) dispara, a função "*removeUser(user)*" será chamada e aquele usuário será removido.

O *two-day data binding* pode ser usado através da notação **[(ngModel)] = "property"**. O *Two-way data binding* foi muito usado nos formulários do projeto que criamos. A seguir, um exemplo do uso do ngModel no sistema desenvolvido.

Figura 7 - Uso do Two-Way Data Binding em Angular no sistema criado

```
35    <input
36      type="password"
37      placeholder="senha"
38      aria-describedby="basic-addon2"
39      name="senha" id="senha"
40      required [(ngModel)]="entrada2"
41      #senha="ngModel"
42      (input)="senhasIguais=false"
43      (input)="senhasDiferentes=false"
44      (blur)="senhasIguais=false"
45      (blur)="senhasDiferentes=false"
46      (keyup.enter)="checar()"
47      (blur)="checar()"
48    >
49    <p class="senhaForte" *ngIf="(senhasIguais)">Senhas iguais :)</p>
50    <p class="senhaFraca" *ngIf="(senhasDiferentes)">As senhas não estão iguais.</p>
```

Fonte: a autora

Na Figura 7, podemos observar na linha 40 que o [(ngModel)] faz a ligação entre o conteúdo que o usuário escreve no *input* e a variável “entrada2”. É desta forma que o Angular oferece o *Two-Way Data Binding*.

- **Ember**

O One-way data binding pode ser executado através de chave dupla **{{valor}}**. A Figura 8 a seguir mostra o uso do two-way data binding em nosso sistema.

Figura 8 - Uso do One-Way Data Binding em Ember no sistema desenvolvido

```
<div class="row">
  <div class="col-xs-6">
    <p>{{item.name}}</p>
    <p>{{item.cor}}</p>
  </div>
  <div class="col-xs-6">
    <p>{{item.marca}}</p>
    <p>{{item.quantidade}}</p>
  </div>
</div>
```

Fonte: a autora

O Two-Way Data Binding pode ser executado, também, através de chaves duplas. A Figura 9, a seguir, mostra a utilização do Two-Way Data Binding em nosso sistema.

Figura 9 - Uso do Two-Way Data Binding em Ember no sistema desenvolvido

```
2       <p>Digite aqui o item que você procura:</p>
3       <input>
4       {{input value=search}}
5       <h4>Buscando por: <b>{{search}}</b></h4>
```

Fonte: A autora

A Figura 9 mostra o *Two-Way Data Binding* através de uma busca. Na linha 4, o input atribui o valor escrito a variável "**search**", e na linha 5, as chaves duplas mostram o valor recebido. Existem outras maneiras de trabalhar com Data Binding no Ember. Além das opções apresentadas, o programador também pode utilizar Computed Properties, aplicando computed.alias() para *Two-Way* e computed.oneWay() para o *One-Way*. Porém, como não foram usadas durante a execução do projeto, não iremos nos aprofundar neste assunto.

- **Data Binding**

Com as informações apresentadas, podemos concluir que tanto o Angular quanto o Ember oferecem maneiras de utilizar o tanto o One-Way quanto o Two-Way Data Binding, ambos de forma bem fácil e rápida.

6.2.4 Dados e Formulários

De acordo com o site oficial do Angular:

> Forms are the mainstay of business applications. You use forms to log in, submit a help request, place an order, book a flight, schedule a meeting, and perform countless other data-entry tasks. In developing a form, it's important to create a data-entry experience that guides the user efficiently and effectively through the workflow.[13] (Angular)

[13] Tradução livre: Formulários são o suporte principal de aplicações empresariais. Usuários usam formulários para login, submeter uma solicitação de ajuda, fazer um pedido, agendar um voo, agendar uma reunião e executar incontáveis outras tarefas com entrada de dados. No desenvolvimento de um formulário, é importante criar uma experiência de entrada de dados que guie o usuário eficientemente e efetivamente através do fluxo de trabalho;

Pensando na importância de um formulário, nós iremos avaliar como funciona o suporte a validação de dados em um formulário em ambos frameworks.

- **Angular**

Em Angular, um formulário é composto por um Template com código HTML e um Componente para lidar com dados e as interações. Através da diretiva ngModel, podemos ter acesso a um conjunto de classes do Angular CSS. As classes *ng-touched* e *ng-untouched* são utilizadas para identificar se o usuário já visitou aquele *input*, as classes *ng-dirty* e *ng-pristine* são utilizadas para identificar se o valor daquele input já mudou, ou seja, se o usuário já escreveu algo, e as classes *ng-valid* e *ng-invalid* são utilizadas para saber se o valor escrito naquele input é válido. (Angular). A Figura 10, a seguir, simula a utilização do Angular adicionando as classes CSS nos arquivos de acordo com as interações do usuário.

Figura 10 - Uso das classes CSS do Angular no sistema desenvolvido

Fonte: A autora

A figura ilustra uma interação do usuário. Em um primeiro momento, o usuário ainda não tinha colocado o mouse dentro do input. Como é possível notar no código,

o input tem a classe “*ng-untouched*”. Após o usuário tocar no input e tirar o mouse, a classe “*ng-untouched*” é retirada e a classe “*ng-touched*” é adicionada. A Figura 11, a seguir, mostra como foi configurada a lógica da Figura 10.

Figura 11 - Uso das classes Angular CSS no sistema desenvolvido

```
50    <input
51      [textMask]="{mask: maskCep}" class="form-control" (blur)="getAddress()"
52      [(ngModel)]="cep" type="text" formControlName="cep"
53      [ngClass]="{'required': (staff.get('cep').hasError('required') && staff.get('cep').touched)
54      || (staff.get('cep').hasError('required') && formDir.submitted)}" />
55    <p class="requiredText" *ngIf="(staff.get('cep').hasError('required') && staff.get('cep').touched)
56      || (staff.get('cep').hasError('required') && formDir.submitted)">Campo Obrigatório</p>
```

Fonte: A autora.

Na linha 53, a classe ‘*required*’ será adicionada ao input caso o mesmo tenha sido tocado e não tenha dados erros. Do mesmo modo, na linha 55, o aviso “Campo Obrigatório” só será exibido quando o campo tiver sido tocado e conter erros.

O Angular ainda tem uma classe chamada **Validators** que permite o programador a definir tamanho mínimo e máximo de uma string, tamanho mínimo e máximo de um número, validar e-mails, entre outros. Ele retorna um erro caso a validação não tenha passado. A Figura 12 a seguir mostra a utilização do Validator no sistema desenvolvido.

Figura 12 - Uso de Validators do Angular no sistema desenvolvido

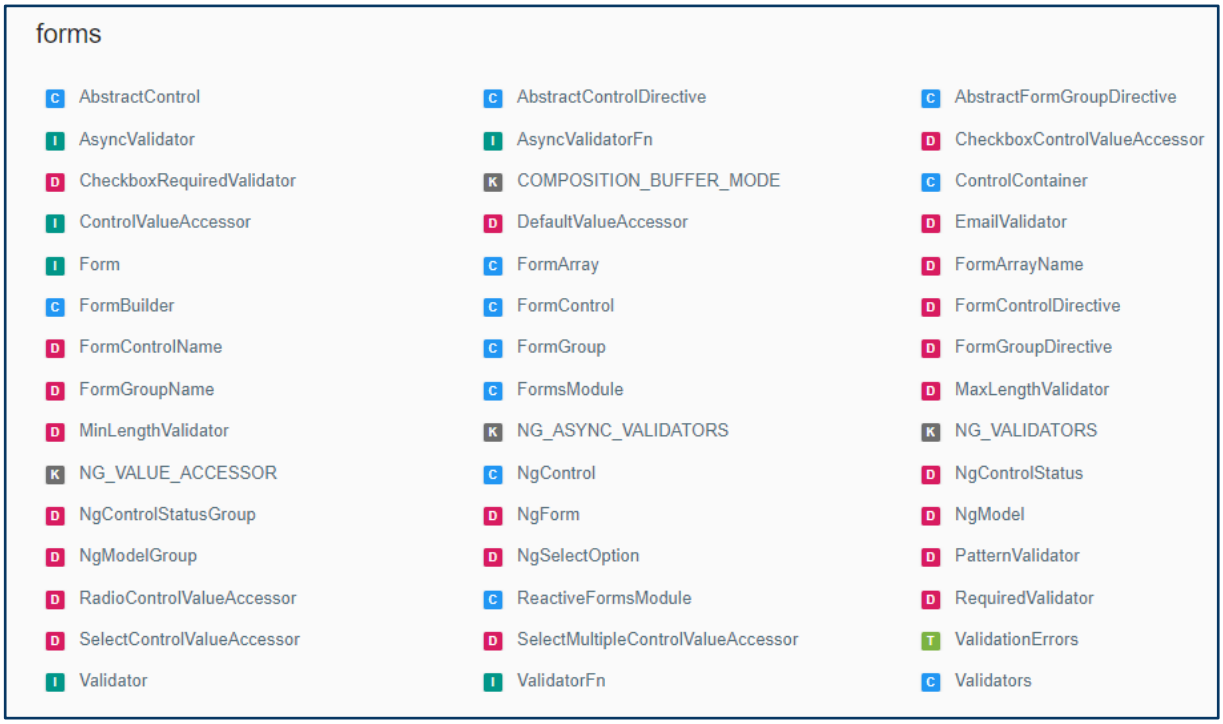
```
36    ngOnInit() {
37      this.user = this.fb.group({
38      url: ['', [Validators.required]],
39      email: ['', [Validators.required, Validators.email]],
40      port: ['', [Validators.required]],
41      company: ['', [Validators.required]],
42      profile: ['AdminFactory', [Validators.required]]
43          });
44      }
```

Fonte: A autora

Na imagem (Figura 12), foi utilizado o Validator que exigem campos não vazios (Validator.Required), e também o Validator que performa uma validação de email (Validator.email). Também é possível criar novos Validators para validar o formulário. O Angular ainda oferece outras classes que podem ajudar o usuário a

melhorar a estrutura e a performance do seu formulário, como FormArray, FormBuilder, FormGroup, ReactiveFormsModule, entre outros. A Figura 13 mostra os diversos elementos relacionados aos formulários em Angular.

Figura 13 - Recursos do Angular para formulários



Fonte: www.angular.io/api

Devido a não estar relacionado ao objetivo do sistema implementado, não iremos entrar em detalhes em todos os elementos oferecidos pelo Angular.

❖ Ember

Em Ember, podemos utilizar os Helpers **{{input}}** e **{{textarea}}** nos formulários. Helpers são funções do Ember que podem ser usadas nos Templates. O programador também pode criar novos helpers e helpers podem ficar dentro de outros helpers. No caso de {{input}}, podemos utilizar vários atributos html dentro deste helper para ajudar na validação do formulário. Como por exemplo: **min, max, size, required, type**, entre outros. Utilizando {{Textarea}} podemos utilizar vários atributos html também: **wrap, maxlength, rows, cols, readonly**, entre outros. A Figura 14 a, seguir, mostra como foram utilizados os helpers no sistema desenvolvido:

Figura 14 - Uso do helper {{input}} no sistema desenvolvido

```
8       <div class="row">
9         <div class="col-sm-4">
10          <label>Mínimo de Embalagens</label>
11          {{input type="number" min="0" max="1000"}}
12        </div>
13        <div class="col-sm-4">
14          <label>Máximo de Embalagens</label>
15          {{input type="number" min="0" max="1000"}}
16        </div>
17        <div class="col-sm-4"></div>
18      </div>
```

Fonte: A autora

Como podemos observar na Figura 14 há inserção dos atributos HTML dentro dos helpers. Ember possui vários Addons para serem utilizados. De acordo com o site oficial, "*Addons provide a wide range of functionality to projects, often saving time and letting you focus on your project*.[14]" (Ember). Foram utilizados vários addons em nosso sistema, como por exemplo, ember-validations. A Figura 15 simula a utilização do ember-validations em nosso sistema:

Figura 15 - Uso do Addon ember-validation do Ember no sistema desenvolvido

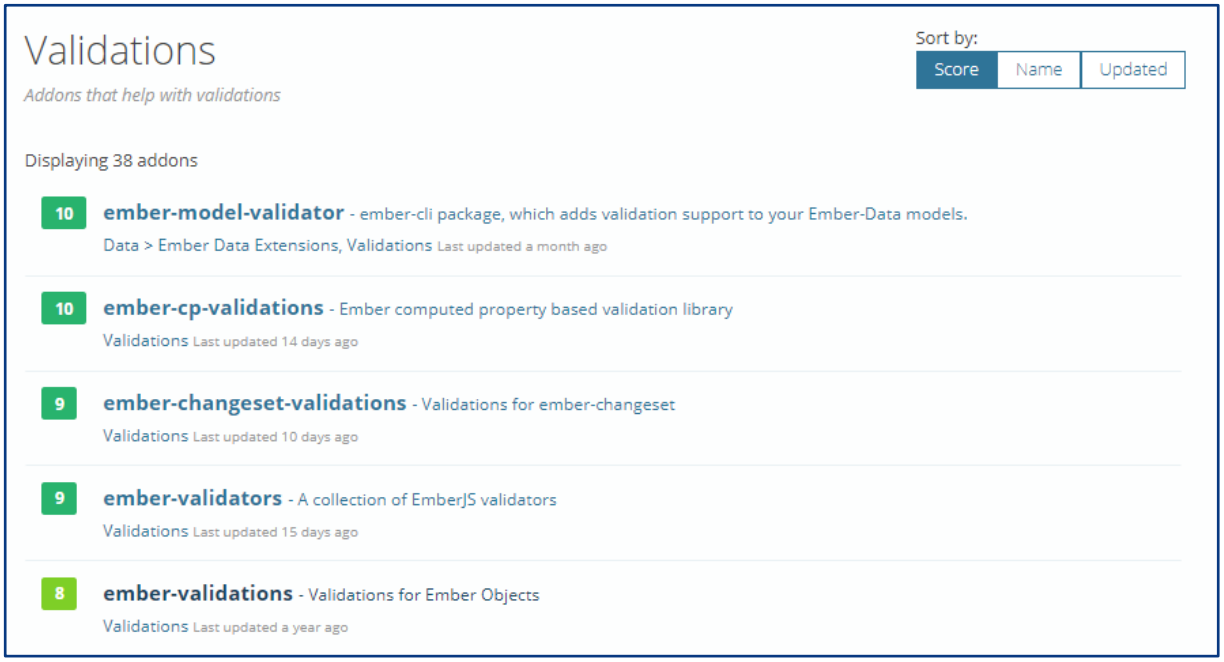
```
5       validations: {
6         model.code: {
7           presence: true,
8           lenght: {minimun: 1, maximum: 32}
9         },
10        model.name: {
11          presence: true,
12          lenght: {minimun: 10, maximum: 100}
13        },
14        model.description: {
15          lenght: {minimun: 20, maximum: 300}
16        }
```

Fonte: A autora

A Figura 15 mostra o uso do validators no cadastro de um item. O Ember possui vários outros addons disponível para uso. A Figura 16 mostra os addons mais bem pontuados na área de validação:

[14] Tradução livre: Addons oferecem uma ampla gama de funcionalidades aos projetos, muitas vezes economizando tempo e deixando você focar em seu projeto.

Figura 16 - Addons do Ember que ajudam validação de formulários



Fonte: A autora

A Figura 16 mostra que existem, até o momento, 38 addons apenas na área de validação. No entanto, existem, ainda, outros addons na área de formulários.

❖ **Comparação**

Tanto o Angular quanto o Ember oferecem muitos recursos para o programador desenvolver bem seus formulários. Pelo nosso sistema ser simples, não utilizamos tantos recursos dos frameworks, apenas o necessário para nosso sistema. Porém, pelo estudo feito, podemos concluir que ambos frameworks oferecem um bom suporte para formulários.

6.2.5 Rotas

Ambos os frameworks possuem uma implementação simples e de fácil utilização das rotas e ambos, também, explicam muito bem como elas funcionam em seus tutoriais.

Em Angular, o usuário administra suas rotas através do **RouterModule**, que é um módulo dedicado a definir como funcionarão as rotas daquele módulo. Segundo a definição no tutorial do Angular, RouterModule é "*A separate NgModule that*

provides the necessary service providers and directives for navigating through application views[15].” (Angular). O usuário pode gerenciar todas as rotas em apenas um RouterModule raiz (Root) ou ele pode dividir essas rotas em um RouterModule filho (Child). A Figura 17 representa o RoutingModule Root do sistema desenvolvido.

Figura 17 - Configuração do RouterModule Root em Angular no sistema desenvolvido

```
10    const appRoutes: Routes = [
11      { path: 'esqueciMinhaSenha', component: EsqueciMinhaSenhaComponent },
12      { path: 'redefinirSenha', component RedefinirSenhaComponent },
13      {path: 'login', component LoginComponent},
14      {path: 'sistema', loadChildren: 'app/sistema/sistema.module#SistemaModule'},
15      { path: '', component LoginComponent},
16    ];
```

Fonte: a autora

Como “sistema” é possui uma ramificação muito grande, foi criado um RouterModule filho (*Child*) dentro dele, conforme apresentado na Figura 18.

Figura 18 - Configuração de RouterModule Child em Angular no sistema desenvolvido

```
12    const sistemaRoutes = [
13      {path: 'sistema', component: SistemaComponent, children: [
14              {path: '', redirectTo: '/sistema/home', pathMatch: 'full'},
15              {path: 'cadastros', loadChildren:
16                'app/sistema/cadastros/cadastros.module#CadastrosModule'},
17              {path: 'home', loadChildren:
18                'app/sistema/home/home.module#HomeModule'},
19              {path: 'inventario', component: AlertasComponent},
20              {path: 'importar', component: AlertasComponent},
21              {path: 'rastreamento',  component: AlertasComponent},
22              {path: 'configuracoes', loadChildren:
23                'app/sistema/configuracoes/configuracoes.module#ConfiguracoesModule'},
24        ]}
25    ];
```

Fonte: a autora

A Figura 18 mostra como funciona o RouterModule Child, que é filho do RouterModule Root. Dentro deste módulo de rotas, existe outros módulos de rotas filhas. A navegação das rotas através do template é feita através da diretiva RouterLink. A Figura 19 mostra a utilização desta directiva no sistema desenvolvido.

[15]Tradução livre: Um NgModule que fornece os serviços e diretivas necessários para navegação através das views de uma aplicação.

Figura 19 - Uso do RouterLink do Angular no sistema desenvolvido

```
<li><a routerLinkActive="active"
        routerLink="/home" >HOME</a></li>
<li><a routerLinkActive="active"
        routerLink="/cadastros">CADASTROS</a></li>
<li><a routerLinkActive="active"
        routerLink="/inventario">INVENTÁRIO</a></li>
<li><a routerLinkActive="active"
        routerLink="/importar">IMPORTAR DADOS</a></li>
<li><a routerLinkActive="active"
        routerLink="/rastreamento">RASTREAMENTOAÇÃO</a></li>
<li><a routerLinkActive="active"
  routerLink="/configuracoes">CONFIGURACOES</a></li>
```

Fonte: a autora

Além de poder criar rotas filhas em Angular e usar a diretiva RouterLink nos templates, ainda podemos criar guardas para essas rotas, passar parâmetros, trabalhar com Lazy Loading, dentre outras possibilidades.

Utilizando o Ember, o Ember-CLI gera um arquivo em *JavaScript,* **router.js**, e nele serão definidas todas as rotas do sistema. Além disso, também é criada uma pasta com todas as rotas do sistema. A Figura 20 representa como ficaria a configuração do sistema desenvolvido em ember.

Figura 20 - Organização das rotas do sistema no router.js utilizando o framework Ember

```
9    Router.map(function() {
10     this.route('sistema', function() {
11       this.route('home', function() {
12         this.route('alarmes');
13         this.route('alarme:id');
14       });
15       this.route('cadastros', function() {
16         this.route('tags');
17         this.route('tags-adicionar');
18         this.route('tags-editar');
19         this.route('plantas');
20         this.route('plantas-adicionar');
21         this.route('plantas-editar');
22         this.route('projetos');
23         this.route('projetos-adicionar');
24         this.route('projetos-editar');
25         this.route('embalagens');
26         this.route('embalagens-adicionar');
27         this.route('embalagens-editar');
28         this.route('setor');
29         this.route('setor-adicionar');
30         this.route('setor-editar');
31         this.route('rotas');
32         this.route('rotas-adicionar');
33         this.route('rotas-editar');
34       });
```

Fonte: A autora

A Figura 20 mostra como um arquivo router.js pode ficar grande caso o sistema seja complexo. É necessário que ele seja dividido em subníveis, pois isso passa uma impressão de o código ser mais limpo, além de ser uma boa prática de programação.

Em Ember, a navegação das rotas se dá pelo helper **{{link-to}}.** A Figura 21 mostra o uso do helper nos templates.

Figura 21 - Uso do helper {{link-to}} do Ember no sistema desenvolvido

```
{{#navbar.content}}
  {{#navbar.nav as |nav|}}
    {{#nav.item}}{{#nav.link-to "index"}}HOME{{/nav.link-to}}{{/nav.item}}
    {{#nav.item}}{{#nav.link-to "cadastros"}}CADASTROS{{/nav.link-to}}{{/nav.item}}
     {{#nav.item}}{{#nav.link-to "inventario"}}INVENTÁRIO{{/nav.link-to}}{{/nav.item}}
    {{#nav.item}}{{#nav.link-to "importar"}}IMPORTAR DADOS{{/nav.link-to}}{{/nav.item}}
    {{#nav.item}}{{#nav.link-to "rastreamento"}}RASTREAMENTO{{/nav.link-to}}{{/nav.item}}
    {{#nav.item}}{{#nav.link-to "configuracoes"}}GC16{{/nav.link-to}}{{/nav.item}}
  {{/navbar.nav}}
{{/navbar.content}}
```

Fonte: a autora

Utilizando o Ember, nós podemos criar rotas filhas, passar parâmetros, apresentar feedback do carregamento da rota, entre outras possibilidades.

- **Comparação**

Ambos frameworks possuem diretivas e helpers de fácil utilização para gerar a navegação do sistema nos templates. A escolha de qual a melhor maneira de gerenciar as rotas varia de acordo com o perfil do programador. Particularmente, prefiro a maneira como o Angular gerencia suas rotas. Pois, no Ember o arquivo *router.js* pode ficar muito longo a medida que o sistema cresce; enquanto em Angular, nós temos a opção de dividir essas rotas em outros *RouterModules* menores. O sistema desenvolvido não era tão complexo, então as duas configurações de navegação conseguiram satisfazer o desenvolvimento das rotas.

6.3 VANTAGENS E DESVANTAGENS

- **Framework CSS**

Durante o desenvolvimento usando Angular, percebemos que existiam muitos conflitos no ng-bootstrap. Foi preciso utilizar o Bootstrap em alguns momentos e, além disso, precisamos de código jQuery em algumas situações do projeto. O código jquery foi inserido no componente. O ember-bootstrap se mostrou excelente durante o desenvolvimento do sistema, pois ele continha vários componentes que foram utilizados no projeto e não tivemos muita dificuldade.

Ambos frameworks oferecem um bom padrão CSS para ajudar ao desenvolvedor.

- **Command-line Interface**

Os dois frameworks possuem uma "Command-line Interface" (CLI), onde é possível executar vários comandos. O Ember utiliza o Ember-CLI, e o Angular utiliza Angular-CLI. Através deles, a criação de um projeto se torna bem mais fácil, visto que eles já criam os arquivos seguindo a estrutura do respectivo framework.

- **Estrutura dos arquivos**

Um dos bloco mais importantes para o desenvolvimento front-end no Angular é o Componente. Ele é conectado ao template HTML, onde podemos construir a camada view. Quando um componente é criado no Angular-CLI, são criados quatro arquivos: um TypeScript representando o componente em si, um arquivo HTML que representa o template daquele componente, um arquivo CSS, e um arquivo TypeScript de teste. Todos os arquivos são salvos dentro de uma pasta com o nome do componente criado. No framework Ember, todos os arquivos são agrupados de acordo com seu tipo. Todos os testes ficam na pasta de testes, todos os templates ficam na pasta de templates. Esse padrão segue para todos os arquivos do ember.

A análise comparativa entre as estruturas de ambos frameworks indica que o Angular é organizado pelos seus componentes enquanto o Ember é organizado de acordo com o tipo de arquivo. Portanto, a escolha sobre qual seria a melhor estrutura de arquivos pode variar para cada programador, pois depende da forma como este trabalha. Particularmente, penso que a forma do Angular estruturar os arquivos seja mais intuitiva e organizada em relação ao Ember, pois desta forma, os arquivos de um mesmo componente não ficam separados.

- **CSS**

Como citado anteriormente, quando criamos um componente no angular-cli, ele cria automaticamente uma pasta com 4 arquivos. Um deles, é um arquivo .css, onde lá podemos colocar o código CSS utilizado naquele determinado componente. Em Ember, quando criamos uma rota, ele cria automaticamente um template para aquela rota; porém, ele não cria um arquivo css, deixando apenas um arquivo app.css para o programador colocar todo o código CSS.

Ter apenas um lugar para colocar todo o código CSS de um sistema é uma ideia não muito interessante, pois o arquivo poderá ficar longo, fazendo com que o programador se perca em seu código, dificultando a edição do mesmo. O *ember-Cli,*

ainda, possui um addon chamado *ember-component-css* que permite que o programador defina um css para um componente único, porém este addon não resolve completamente o problema. Para dividir o código css de acordo com a rota, foi usado o método “@import url(“nome-da-rota.css”)” em app.css. Esta foi uma solução que funcionou para dividir o código css de acordo com a rota, porém este método ainda não restringe o código a apenas aquela rota.

Pelos motivos citados acima, Angular 2 foi considerado mais interessante em questão de organização do código css.

- **Feedback**

Muitas vezes, durante o desenvolvimento utilizando Ember, o sistema continha erros que faziam com que alguns itens não fossem exibidos na tela. Apesar de ser um erro de código, ou seja, um erro do programador, o ember em muitas vezes não dava nenhum feedback. Não havia mensagem no console, ou no terminal, ele simplesmente apagava aquele item sem mostrar a justificativa para que o desenvolvedor achasse o erro. Essa falta de feedback dificultou, na maioria das vezes, o desenvolvimento do sistema.

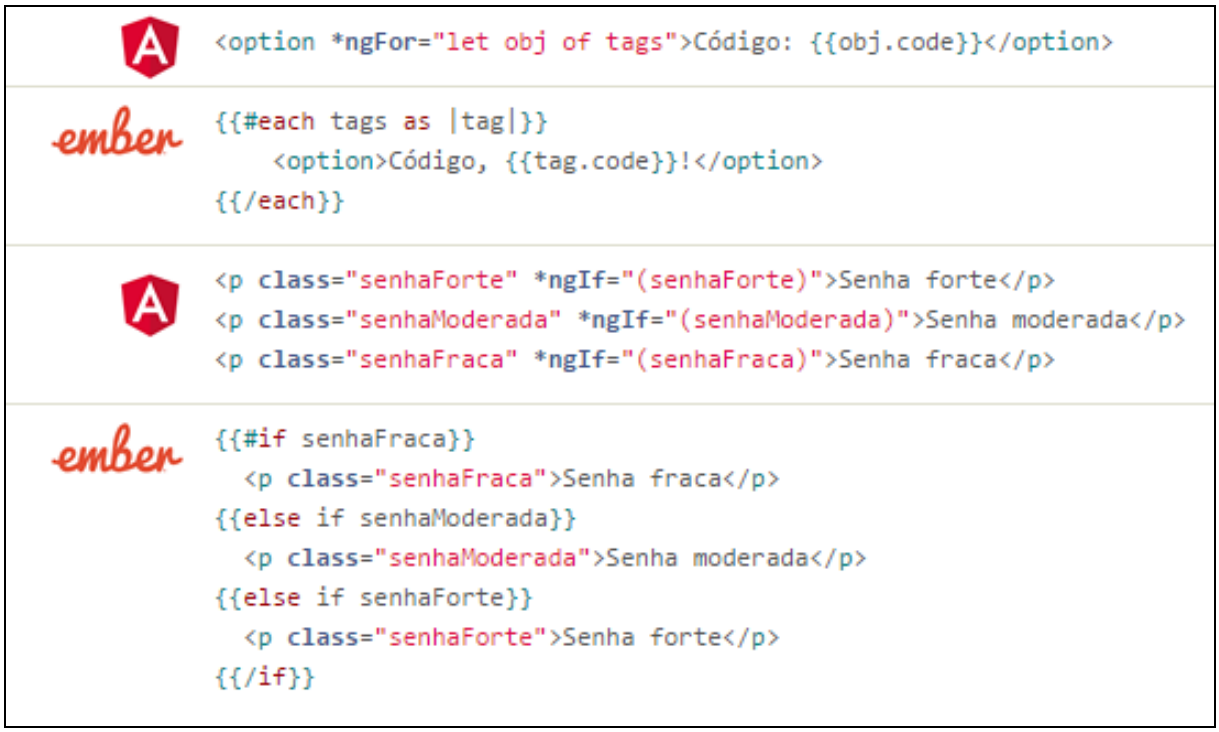
- **Documentação e Suporte Comunitário**

Ambos Frameworks possuem uma boa documentação em seus sites oficiais. Eles também possuem bons tutoriais, e, apesar de observar que o Ember tem menos popularidade do que o Angular , tanto o Ember quanto o Angular possuem um bom suporte comunitário na internet.

- **Esforço**

Em questão de esforço, as diretivas do Angular e os Helpers do Ember, que foram muito muito utilizados no sistema, oferecem uma diferença na quantidade de código usado. O ember, normalmente, obtém uma ou duas linhas de código a mais do que o Angular para fechar o helper. Contudo, as diretivas e helpers possuem o mesmo nível de dificuldade para serem implementadas. A Figura 22 mostra uma diferença entre diretivas e Helpers.

Figura 22: Comparação entre Diretivas e Helpers.



```
<option *ngFor="let obj of tags">Código: {{obj.code}}</option>
```

```
{{#each tags as |tag|}}
    <option>Código, {{tag.code}}!</option>
{{/each}}
```

```
<p class="senhaForte" *ngIf="(senhaForte)">Senha forte</p>
<p class="senhaModerada" *ngIf="(senhaModerada)">Senha moderada</p>
<p class="senhaFraca" *ngIf="(senhaFraca)">Senha fraca</p>
```

```
{{#if senhaFraca}}
  <p class="senhaFraca">Senha fraca</p>
{{else if senhaModerada}}
  <p class="senhaModerada">Senha moderada</p>
{{else if senhaForte}}
  <p class="senhaForte">Senha forte</p>
{{/if}}
```

Fonte: a autora.

Como podermos ver na Figura 22, existe uma diferença de quantidade de linhas de código. Em Angular, o código é mais curto e limpo do que em Ember. Se considerarmos um sistema muito grande, a diferença de linhas de código ficará mais aparente do que o exemplo retirado do sistema. Em nossa análise, o Angular foi considerado mais fácil de implementar por exigir um menor esforço do programador.

7. CONCLUSÃO

Este trabalho teve como objetivo fazer um estudo comparativo entre dois frameworks de front-end. Para isso, fizemos um levantamento dos principais frameworks utilizados, recentemente, e escolhemos dois deles para avaliação: Angular e Ember.js. Em seguida, produzimos o desenvolvimento front-end de duas aplicações com os dois frameworks escolhidos, utilizando Design Responsivo em ambas as aplicações. Após o desenvolvimento dos sistemas, comparamos os principais recursos dos frameworks escolhidos e identificamos as vantagens e desvantagens de cada framework na prática.

Após análise dos resultados obtidos, podemos chegar a conclusão de que o Angular é mais popular do que Ember. Em questão de publicações, ambos não possuem artigos no SciELO, nem teses e dissertações da CAPES, mas possuem livros disponíveis para estudo no Google Book. Apesar de saber que Angular possui um suporte comunitário maior, Ember também possui um bom suporte comunitário no GitHub e no Stack. Angular é mais utilizado no mercado do que Ember, apesar de existirem muitas vagas para as duas tecnologias.

Em vários aspectos, tanto o framework Angular quanto o framework Ember conseguem satisfazer as necessidades do sistema. Ambos ofereceram um bom gerenciamento de rotas, suporte para os dois tipos de Data Binding, suporte para Formulários, possuem uma Command-line Interface e sabemos que ambos oferecem um bom suporte para testes automatizados.

Para o sistema desenvolvido, o framework Angular foi mais adequado devido a estrutura dos arquivos ser modularizada e organizada, diferentemente do Ember.

7.1 TRABALHOS FUTUROS

Em trabalhos futuros, seria interessante implementar a cultura de testes automatizados no desenvolvimento de um novo sistema para diminuir seus erros e otimizar o tempo de desenvolvimento. Após o estudo sobre o tema no critério “Testes Automatizados”, seria interessante ter uma análise na prática.

Além de testes automatizados, poderíamos explorar outros recursos oferecidos pelos frameworks e que não foram explorados em nosso sistema. No caso do Angular, existem muitos módulos que ainda podem ser explorados. No caso

do ember, existem muitos addons que não foram utilizados em nosso sistema. Em contribuições futuras, o sistema poderia oferecer diferentes features que explorassem o que não foi desenvolvido. Basta observar a documentação do Angular e a lista de addons disponíveis do Ember, existem muitos recursos a serem explorados.

Além disso, seria interessante fazer esta análise com todos os outros frameworks citados no trabalho, como Vue.js. Vue.js foi muito considerado para ser estudado durante este trabalho pelo seu diferencial ser a sua simplicidade. Outro framework/biblioteca que não foi citado no trabalho, no entanto tem uma certa importância, é o React.js. React.js não é considerado um framework, e sim apenas uma biblioteca, e por este motivo, ele não foi citado neste trabalho. React está tendo uma boa aceitação no mundo, basta observar sua popularidade no GitHub, no StackOverFlow e no GoogleTrends. Apesar de ser uma biblioteca que não chega a ser um framework completo, React deveria ser considerado em comparações futuras.

8 REFERÊNCIAS

ALYSON LA. **Language Trends on GitHub. GitHub.** 19/08/2015. Acesso em: 29/11/2017. Disponível em: <https://github.com/blog/2047-language-trends-on-github>

BACKBONE.Js. Site: BACKBONE.Js.. Acesso em: 30/11/2017. Disponível em: <http://backbonejs.org/#>

BARROS, Tiago; SILVA, Mauro; ESPÍNOLA, Emerson. **State MVC:** Estendendo o padrão MVC para uso no desenvolvimento de aplicações para dispositivos móveis. C.E.S.A.R - Centro de Estudos e Sistemas Avançados do Recife. Acesso em: 24/11/2017. Disponível em: <http://www.tiagobarros.org/docs/SMVC.pdf>

BERNARDO, Paulo Cheque., KON, Fabio. A Importância dos Testes Automatizados: Controle ágil, rápido e confiável de qualidade. **Engenharia de Software Magazine**, 1(3), pp. 54-57. 2008. Acesso em: 30/11/2017. Disponível em: <https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>

SOUZA, Marcos Vinícius Bittencourt de. Estudo Comparativo entre Frameworks Java Para Construção de Aplicações Web. **Trabalho de Graduação.** Nº 191. Santa Maria, RS, Brasil, 2004. Disponível em: <www-app.inf.ufsm.br/bdtg/arquivo.php?id=20&download= >. Acesso em: 23 de out 2017.

CAMPOS, Ana Cristina. IBGE: celular se consolida como o principal meio de acesso à internet no Brasil. **Agência Brasil.** 2016. Acesso em: 11/09/2017. Disponível em: http://agenciabrasil.ebc.com.br/geral/noticia/2016-12/ibge-celular-se-consolida-como-o-principal-meio-de-acesso-internet-no-brasil

CATHO**. Empregos e Vagas de emprego**. Acesso em: 30/11/2017. Disponível em: <https://www.catho.com.br >

CAVALCANTI, Leo. Site TABLELESS. **Apresentando:** Meteor! . 06/08/2015. Acesso em: 30/11/2017. Disponível em: <https://tableless.com.br/apresentando-meteor/ >

DEVMEDIA. **Conheça os Padrões de Projeto**. Acesso em: 24/11/2017. Disponível em: <https://www.devmedia.com.br/conheca-os-padroes-de-projeto/957 >.

DUARTE, NUNO FILIPE BRANDAO. Frameworks e Bibliotecas Javascript**. Dissertação em Engenharia Informática.** Área de Especialização em Arquitecturas Sistemas e Redes. Instituto Superior de Engenharia do Porto (ISEP), 2015. Acesso em: 29/11/2017. isponível em: http://recipp.ipp.pt/bitstream/10400.22/8223/1/DM_NunoDuarte_2015_MEI.pdf

DURELLI, V. H. S., VIANA, M. C. e PENTEADO, R. A. D. (2008). Uma Proposta de Reúso de Interface Gráfica com o Usuário Baseado no Padrão Arquitetural MVC. In: **IV Simpósio Brasileiro de Sistemas de Informação,** Rio de Janeiro - RJ. Anais SBS 2008.

EMBER.JS. Site Ember.js. **A framework for creating ambitious web applications**. Acesso em: 24/11/2017. Disponível em: <https://emberjs.com/ >

IHIMG, Simon. **Ember-bootstrap**. Acesso em: 3/11/3017. Disponível em: <http://www.ember-bootstrap.com/ >

ESCHWEILER, Sebastian. **Using Bootstrap with Angular.** Coding The Smart Way.com**.** 15/05/2015. Acesso em: 30/11/2017. Disponível em: <https://medium.com/codingthesmartway-com-blog/using-bootstrap-with-angular-c83c3cee3f4a >

FELIZARDO, André. **O que é Ember.Js**. Blog do André Felizardo. 30/07/2017. Acesso em: 29/11/2017. Disponível em: <http://www.andrefelizardo.com.br/blog/o-que-e-ember-js/ >

FLANAGAN, David. **JavaScript:** o guia definitivo. 4. ed. Porto Alegre: Bookman, 2004.

FOUNDATION. **The most advanced responsive front-end framework in the world.** Zourb Foundation. Acesso em: 24/11/2017. Disponível em: <https://foundation.zurb.com/ >

FRANCO, Rebeca Such Tobias. Estudo comparativo entre frameworks Java para desenvolvimento de aplicações web: JSF 2.0, grails e spring web MVC. 2011. 89 f. **Trabalho de Conclusão de Curso** (Especialização) – Universidade Tecnológica Federal do Paraná, Curitiba, 2011. Acesso em: 23/10/2017. Disponível em: <http://www2.dainf.ct.utfpr.edu.br/esp/monografias-de-especializacao-da-turma-vi-2010-2011/CT_JAVA_VI_2010_16.PDF/view >.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projeto**: Solução Reutilizáveis de Software Orientado a Objeto, 1 ed., São Paulo: Bookman, 2000.

GERCHEV, Ivaylo. **The 5 Most Popular Frontend Frameworks of 2017 Compared.** SitePoint. 2017. Acesso em: 26/11/2017. Disponível em: <https://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/ >

GERDESSEN, Anton. **Framework comparison method: Comparing two frameworks based on technical domains, focussing on customisability and modifiability**. Master's thesis. UvA: University of Amsterdam, 2007.

GETBOOTSTRAP**. Site GetBootstrap.** Acesso em: 24/11/2017. Disponível em: <https://getbootstrap.com/ >

GETULKIT. Site GetUIkit. A **lightweight and modular front-end framework for developing fast and powerful web interfaces.** Acesso em: 24/11/2017. Disponível em: <https://getuikit.com/ >

GITHUB. **Site GitHub.** Acesso em: 30/11/2017. Disponível em:<https://github.com/ >

GITHUB.COM. Site: Github.com. **Repositório do AngularJS no GitHub**. Acesso em: 29/11/2017. Disponível em: <https://github.com/angular/angular.js >

GOODMAN, Danny, **Dynamic HTML:** The Definitive Guide. O'Reilly, 2002.

GOOGLE TRENDS. **Site Google Trends**. Acesso em: 30/11/2017. Disponível em: <https://trends.google.com.br/trends/ >

HOMEDANI, Mosh. The Complete Angular Couse: Beginner to Advanced. Udemy. 09/2017. Acesso em: 6/12/2017. Disponível em: <https://www.udemy.com/the-complete-angular-master-class/>

JÚNIOR, Ismael de Souza Oliveira. Comparação entre Frameworks Java para Desenvolvimento de Web Services: Axis2 e CXF. **Trabalho de Conclusão de Curso** (Monografia). Universidade FUMEC, Faculdade de Ciências Empresariais – FACE. Belo Horizonte, 2013. Acesso em: 23/10/2017. Disponível em: <http://professores.dcc.ufla.br/~terra/publications_files/students/2013_fumec_oliveirajunior.pdf >.

LIMA, Matheus. **Como é trabalhar como Front-End Developer.** 2017. Acesso em: 11/07/2017. Disponível em: https://medium.com/trainingcenter/como-%C3%A9-trabalhar-como-front-end-developer-por-matheus-lima-7404017a9703

METEOR. Site Meteor. **Introducing Meteor API Docs.** Acesso em: 30/11/2017 Disponível em: <http://docs.meteor.com/#/full/ >

METEOR. Site Meteor. **Meteor Development Group.** The open source company for JavaScript and GraphQ. Acesso em: 30/11/2017. Disponível em: <https://www.meteor.io/ >

MICROSOFT. Site Microsoft. **Data Binding Overview.** Acesso em: 30/11/2017. Disponível em: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/data-binding-overview>

MATERIALIZE. Site Materialize. **Um framework front-end moderno e responsivo baseado em Material Design.** Acesso em: 24/11/2017. Disponível em: <http://materializecss.com/ >

NG-BOOTSTRAP. Site Ng-Bootstrap. **Repositório do Ng-Bootstrap no GitHub.** Acesso em: 30/11/2017. Disponível em: https://github.com/ng-bootstrap/ng-bootstrap

BRASIL. Ministérios da Educação. **PERIÓDICOS CAPES.** Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes). Acesso em: 26/11/2017. Disponível em: https://www.periodicos.capes.gov.br/?option=com_pcontent&view=pcontent&alias=missao-objetivos&mn=69&smn=74.

PURECSS. Site PureCSS. **A set of small, responsive CSS modules that you can use in every web project.** Acesso em: 24/11/2017. Disponível em: <https://purecss.io/ >

PRADO, Olavo Filipe. **Começando com Angular 2.** MATERA Systems. 09/02/2017. Acesso em: 29/11/2017. Disponível em: <http://www.matera.com/br/2017/02/09/comecando-com-angular-2/ >

REIS,Vinícius. **Por que Vue.js e não React?** 16/03/2017. Acessado em: 27/11/2017.
Disponível em: https://blog.codecasts.com.br/por-que-vue-js-e-nao-react-d5b58c09d193.

ROBBINS, Jennifer Niederst. **Learning web design:** A beginner's guide to HTML, CSS, JavaScript, and web graphics. Cambridge, UK: O'Reilly Media, 2012.

ROBERTO, Thiago. Análise e Comparação de Frameworks para Desenvolvimento Web em Java. **Trabalho de Conclusão de Curso**. Universidade Federal de Santa Catarina. Florianópolis, SC. 2007. Acessado em: 23/10/2017. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_669/TCC-ThiagoRobertoSantos-final.pdf >

SCIELO. Site **SciELO** - Scientific Electronic Library Online. Acesso em: 26/11/2017. Disponível em: http://www.scielo.org/php/level.php?component=56&item=1&lang=pt

SEMATIC-UI. Site Semantic-UI. **User Interface is the language of the web.** Acesso em: 24/11/2017. Disponível em: <https://semantic-ui.com/ >

SHOEMAKER, Craig. **HTML5 History:** Clean URLs for Deep-linking Ajax Applications**.** Acesso em: 30/11/2017. Disponível em: <http://www.codemag.com/article/1301091 >

SILVA, Maurício Sammy. **Web Design Responsivo.** Primeira edição. São Paulo: Novatec, 2014.

SOUZA, C. R. B. **Um Framework para Editores de Diagramas Cooperativos baseados em Anotações**. Campinas: Unicamp, 1998. Acesso em: 27/11/2017. Disponível em: <http://repositorio.unicamp.br/bitstream/REPOSIP/275954/1/Souza_CleidsonRonaldBotelhode_M.pdf >

STACKOVERFLOW. **Site StackOverFlow.** Acesso em: 30/11/2017. Disponível em: <https://pt.stackoverflow.com/>

VILLELA, Flávia. Celular é principal meio de acesso à internet no Brasil, mostra IBGE. **Agência Brasil.** 2016. Acesso em: 11/09/2017. Disponível em: http://agenciabrasil.ebc.com.br/economia/noticia/2016-04/celular-e-principal-meio-de-acesso-internet-na-maioria-dos-lares

W3SCHOOLS. Site. W3Schools. **AngularJs Tutorial.** Acesso em: 24/11/2017. Disponível em: <https://www.w3schools.com/angular/default.asp >

WILSON, Mike. **Construindo Aplicações Node com MongoDB e Backbone:** Prototipação rápida e implantação escalável. Novatec Editora, 2013. Acesso em: 30/11/2017. Disponível em: <https://books.google.com.br/books?id=fKqjAwAAQBAJ>

ZANETTE, Alysson. **Framework x Biblioteca x API.** Entenda as diferenças! BeCode. Acesso em: 27/11/2017. Disponível em: <https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas/ >

ZEMEL, Tárcio. **Web Design Responsivo:** páginas adaptáveis para todos os dispositivos. São Paulo: Casa do Código, 2013.