UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FELIPE BARBOSA KRELLING

RAFAEL DE ANDRADE MAIO

APLICAÇÃO *WEB* PARA GERÊNCIA DE EVENTOS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

FELIPE BARBOSA KRELLING

RAFAEL DE ANDRADE MAIO

APLICAÇÃO *WEB* PARA GERÊNCIA DE EVENTOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Richard Duarte Ribeiro

PONTA GROSSA

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

APLICAÇÃO *WEB* PARA GERÊNCIA DE EVENTOS

por

FELIPE BARBOSA KRELLING
RAFAEL DE ANDRADE MAIO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 9 de novembro 2017 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Dr. Richard Duarte Ribeiro
Prof. Orientador

_____
Dr. Diego Roberto Antunes
Membro titular

_____
Dra. Simone de Almeida
Membro titular

_____
Profª. Dra. Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra. Mauren Louise Sguario
Coordenadora do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Dedicamos este trabalho a todos que
estiveram ao nosso lado, e nos sustentaram
ao passarmos por todo este processo.

AGRADECIMENTOS

Como é impossível mostrar todos os bastidores necessários para realizar qualquer trabalho, gostaríamos de agradecer todas as pessoas que estiveram ao nosso lado e que nos apoiaram desde o início do nosso curso, como nos incentivaram a ir até o seu fim.

Agradecemos a nossa família por sempre ter estado ao nosso lado como também sempre ter sonhado em ver este passo de realizar a nossa graduação sendo realizado. Muita gratidão por todo o apoio, em todos os momentos, independente do ambiente ou das condições.

Por último, mas nunca como menos importante, gostaríamos de agradecer ao prof. Dr. Richard Duarte Ribeiro por ter aceito nos orientar neste ano, onde não só captou muito bem a ideia do projeto, como também sempre se mostrou de mente aberta para nos ajudar.

Nossa sincera gratidão para todos que nos ajudaram para a realização deste trabalho.

RESUMO

KRELLING, Felipe; MAIO, Rafael de Andrade. **Aplicação *web* para gerência de eventos.** 2017. 66 f. Trabalho de Conclusão de Curso Tecnologia em Análise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná - Paraná. Ponta Grossa, 2017.

A quantidade de eventos sociais tem crescido muito na sociedade, como também estão sendo cada vez mais adaptados para as novas tecnologias que estão surgindo atualmente. Com o desenvolvimento de tais tecnologias nas mais variadas áreas, a existência de um *website* que possa unir organizadores de eventos com fornecedores de serviços pode ser visto como uma oportunidade de oferecer auxílio na criação de eventos sociais na região de Ponta Grossa. Os estudos utilizados foram conduzidos em algumas áreas ligadas à realização de eventos. Este trabalho propõe a criação de um *website* visando facilitar a organização de eventos para usuários inexperientes na organização dos mesmos. Este permitirá mais liberdade, facilidade e agilidade para a criação de eventos de tamanhos variados, abrangendo os mais simples, até os de médio porte.

Palavras-chave: Eventos. Organização. Tecnologia na sociedade. Mean *stack.* Node.

ABSTRACT

KRELLING, Felipe; MAIO, Rafael de Andrade. **Events managing web system.** 2017. 66 p. Work of Conclusion Course Graduation in Tecnologia em Análise e Desenvolvimento de Sistemas – Federal Technology University – Paraná. Ponta Grossa, 2017.

The number of social events has grown significantly in society, at the same time that they are being adapted to new technologies which are emerging nowadays. With the development of these technologies, the need for a website which is able to unite event promoters with services providers is seen as necessary to make their work easier in the region of Ponta Grossa. Using some recent studies about the development of social events in Brazil and the overall use of the Internet, the proposed website tend to make the organization of events easier for all of those who are not experienced in this area. It becomes possible to have more freedom, agility and easiness when creating a social event of different sizes, including simple to even medium ones.

Keywords: Events. Organization. Technology in society. MEAN stack. Node.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

ACID	*Atomicity, Consistency, Isolation, Durability*
API	*Application Programming Interface*
CAP	*Consistency, Availability, Partition Tolerance*
CSS	*Cascading Style Sheets*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
JWT	JSON *Web Token*
MIT	Massachusetts Institute of Technology
NPM	*Node Package Manager*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
URI	*Universal Resource Identifier*
URL	*Universal Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Em 2002, o lucro gerado pelas realizações de eventos no Brasil não passava de R$ 4 bilhões. Já em 2013, o número aumentou para R$ 59 bilhões (SEBRAE, 2014). É possível notar o crescimento, visto que no mesmo ano desta pesquisa, registrou-se um aumento de 14%. Em outra pesquisa realizada pela Associação Brasileira de Eventos Sociais, apenas o setor de eventos sociais, como casamentos, formaturas e festas debutantes em 2014 obteve um lucro de R$ 16,8 bilhões (ABEOC, 2015).

O aumento do uso da Internet por empresas para comércio *online* também cresceu. Segundo uma pesquisa realizada pelo Sebrae houve um aumento de 15,3% em 2015 em relação ao ano anterior (SEBRAE, 2015).

Com estas informações percebe-se um espaço no mercado para o desenvolvimento de uma aplicação *web* no formato de um *website*, a fim de facilitar a criação de eventos. Acredita-se que facilitar o caminho do cliente para os serviços utilizados em eventos sociais, como bebidas, local e bandas, irá gerar um custo total menor, seja ele tempo ou dinheiro.

A aplicação tem como objetivo centralizar informações de serviços relacionados a área de eventos e promover um meio para a interação entre clientes e empresas a fim de permitir a organização de eventos sociais em geral. Também terá espaço para empresas anunciarem seus produtos e serviços, possivelmente aumentando sua visibilidade no mercado e estimulando a concorrência e a melhora de preços. Desta forma, espera-se contribuir para a economia do setor de eventos e empresas locais.

O presente trabalho propõe a construção da aplicação mencionada, o levantamento de informações relacionadas à Internet e seu efeito na economia nacional, como também a evolução das aplicações e funcionalidades de *websites.* Também aborda sobre algumas tecnologias existentes para o desenvolvimento de aplicações *web*, a escolha das ferramentas utilizadas e o modelo de desenvolvimento empregado na construção da aplicação.

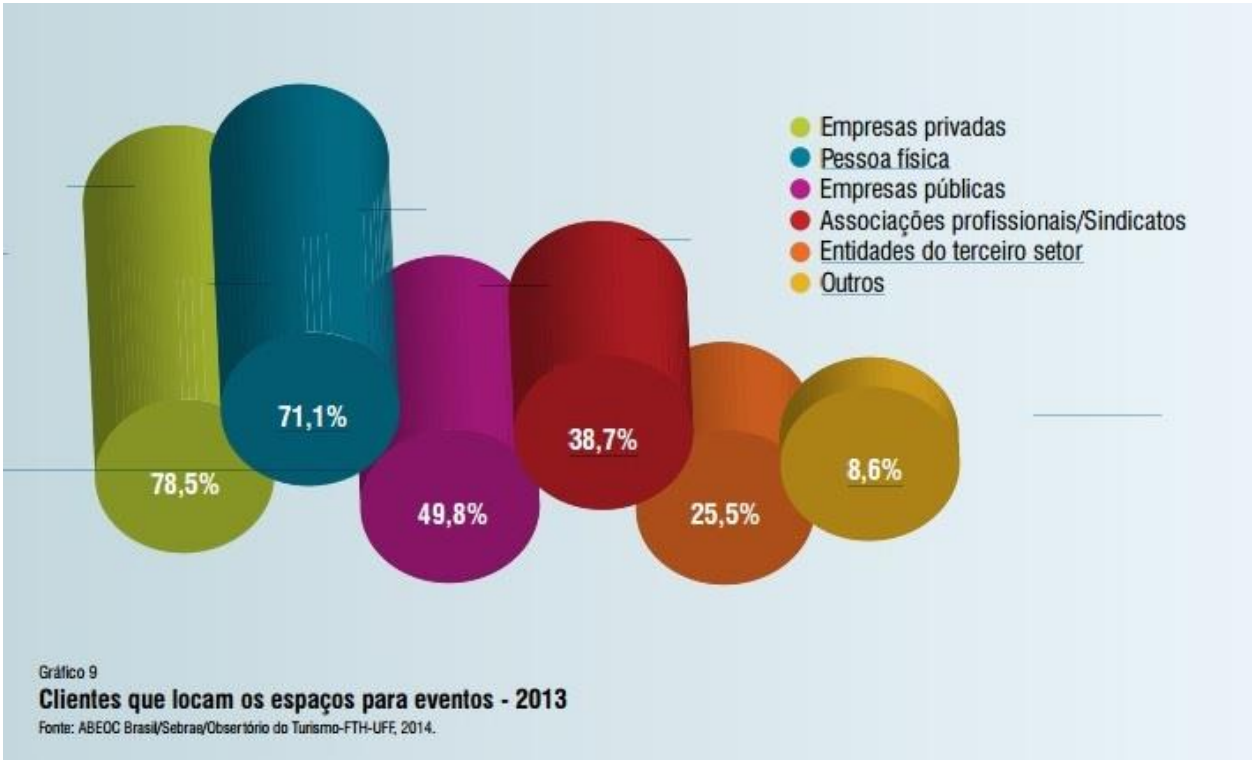
1.1 ESCOPO E MOTIVAÇÃO

A criação de eventos pode ser uma tarefa demorada e árdua para pessoas que não são profissionais na área de criação de eventos. A busca pelos serviços desejados é difícil, pois o organizador deve procurar pelos serviços mais baratos ao mesmo tempo que tenha a qualidade e disponibilidade para seu evento.

De acordo com o relatório da ABEOC (de 2013), grande parte das pessoas que locam espaços para eventos no Brasil são pessoas físicas, mostrando um grande potencial para uso de ferramentas de auxílio na organização de eventos, aproveitando o fato de que elas provavelmente não podem ou não intencionam contratar uma empresa para realizar o evento.

É possível observar a preferência dos clientes quanto a locação de espaços na Figura 1:

Figura 1 – Clientes que locam os espaços para eventos



Fonte: ABEOC (2013)

Além disso, acredita-se que a centralização de informações e listagem de serviços e produtos facilite a busca e possa contribuir com a diminuição do tempo necessário para

a organização de eventos, assim presume-se uma diminuição no custo na realização de tais eventos.

Foi também realizada uma entrevista com um organizador de eventos para auxiliar no levantamento de requisitos para a aplicação. É possível ver a entrevista no Apêndice A.

1.2 OBJETIVOS

O objetivo do trabalho proposto é a construção de um sistema *web* a ser utilizado tanto por empresas relacionadas ao ramo de eventos, quanto por usuários que desejam organizar seus próprios eventos.

Para cumprir estes objetivos gerais, é necessário:

- Desenvolver um componente que permita o cadastro de usuários (físicos e jurídicos) envolvidos com o ramo de eventos;
- Implementar um componente de comunicação que permita a troca de mensagens entre usuários da plataforma;
- Construir um componente que possibilite às pessoas jurídicas anunciar seus produtos e serviços;
- Elaborar um componente de pesquisa de serviços e produtos;
- Desenvolver um componente para o gerenciamento dos produtos relacionados ao evento, contendo os serviços e produtos contratados;
- Desenvolver um componente para gerenciar o processo de contratação dos produtos e serviços (conhecido como “carrinho de compras”).

1.3 ESTRUTURA DO TRABALHO

Este trabalho apresentou a introdução ao tema abordado no primeiro capítulo, como também as motivações que inspiraram sua realização e objetivos determinados para sua conclusão.

O segundo capítulo aborda a revisão literária do conteúdo do trabalho, apresentando tópicos referentes ao impacto da Internet na sociedade, o uso da mesma

na gestão de eventos, além de explicar as tecnologias utilizadas no desenvolvimento da aplicação *web*, discutindo detalhes das linguagens *web*, *frameworks* e banco de dados utilizados.

O terceiro capítulo apresenta a metodologia utilizada para desenvolver o trabalho, contendo também uma análise referente aos sistemas similares existentes no mercado atual.

O quarto capítulo descreve o desenvolvimento do projeto, detalhando os passos que foram necessários para atingir os objetivos desejados, relatando também as dificuldades encontradas em sua realização.

No quinto e último capítulo, são abordados os resultados da aplicação, como também as considerações finais do trabalho, que discute possíveis continuações para o projeto.

2 REFERENCIAL TEÓRICO

Este capítulo visa apresentar as tecnologias necessárias e também os motivos para as escolhas das ferramentas usadas no desenvolvimento do projeto.

Os seguintes assuntos são abordados nesta seção: o impacto da Internet na economia; o histórico da Internet; a evolução das tecnologias com a Internet; as linguagens de servidor; o armazenamento de dados; a escolha das ferramentas; e finalmente os *frameworks* utilizados.

2.1 O CONCEITO DE INTERNET

A Internet Society (2017), organização sem fins lucrativos de associados profissionais que facilita e sustenta a evolução técnica da Internet, define a Internet como:

> A Internet é tanto uma capacidade de difusão mundial, como também um mecanismo de disseminação de informações. Ela também é um meio de colaboração e interação entre indivíduos e seus computadores, independentemente de suas localizações geográficas. (INTERNET SOCIETY, 2017, tradução nossa)

A Internet teve como início a ARPANET (*Advanced Research Projects Agency Network*), a primeira rede baseada em comutação de pacotes[1] (INTERNET SOCIETY, 2017).

A ideia em que foi baseada surgiu da discussão do conceito de Rede Galáctica por J.C.R. Licklider em agosto de 1962. Licklider visava um conjunto de computadores conectados globalmente onde todos poderiam acessar dados e programas independentemente de sua localidade (INTERNET SOCIETY, 2017).

No ano de 1967 Lawrance G. Roberts publicou seu plano para a ARPANET (INTERNET SOCIETY, 2017). Posteriormente este plano incorporou a pesquisa

[1] Conceito definido inicialmente por Leonard Kleinrock para a comunicação de dados utilizando pacotes (unidade de transferência de informação) enviados individualmente através de nós da rede.

desenvolvida por Donald Davies e Roger Scantlebury desenvolvidos no NPL (National Physics Laboratory) sobre comutação de pacotes (INTERNET SOCIETY, 2017).

A DARPA (*Defense Advanced Research Projects Agency*), junto com Roberts, refinou a estrutura geral e as especificações da ARPANET e em 1968 realizou uma solicitação para cotação (RFQ) para o desenvolvimento do comutador de pacotes chamado *Interface Message Processors*, o IMP (INTERNET SOCIETY, 2017).

Em 1969, na Universidade da Califórnia, em Los Angeles, foi instalado o primeiro IMP e o primeiro computador foi conectado a rede (INTERNET SOCIETY, 2017). O projeto de Doug Engelbart no Instituto de Pesquisa de Standford foi adicionado como segundo nó na rede.

Nos anos que seguiram, vários computadores foram rapidamente adicionados a rede. Esta rede mundial cresceu a ponto de incluir outros tipos de redes como de satélite, rádio, entre outras. A demanda aumentou a ponto de ser necessário o desenvolvimento de novos protocolos (INTERNET SOCIETY, 2017).

A Internet evoluiu e, hoje, é uma rede mundial de computadores que permite a comunicação e a troca de informações.

2.2 O IMPACTO DA INTERNET NA SOCIEDADE

A Internet e as mídias sociais talvez sejam os elementos que mais cresceram em tão pouco tempo na sociedade. Segundo um relatório da organização McKinsey de 2011, o *Facebook* passou a ter 800 milhões de usuários ao redor do mundo, enquanto em 2006 a rede social tinha apenas alguns milhares de estudantes cadastrados (MCKINSEY, 2011). Hoje a mesma empresa conta com mais de 1,86 bilhões de usuários ativos ao redor do mundo, com um crescimento de 17% ao ano (ZEPHORIA, 2017).

É interessante notar que pessoas entre 15 e 35 anos mostram mais interesse pela Internet do que por serviços básicos como água, luz, comida, etc. (COMMSCOPE, 2016). Não somente os jovens perceberam a importância da Internet; alguns governos também já a consideram vital. O Canadá, por exemplo, anunciou em 2016 que liberou fundos para dar acesso à Internet de alta velocidade em todo país, tornando-o um direito humano básico (CRTC, 2016).

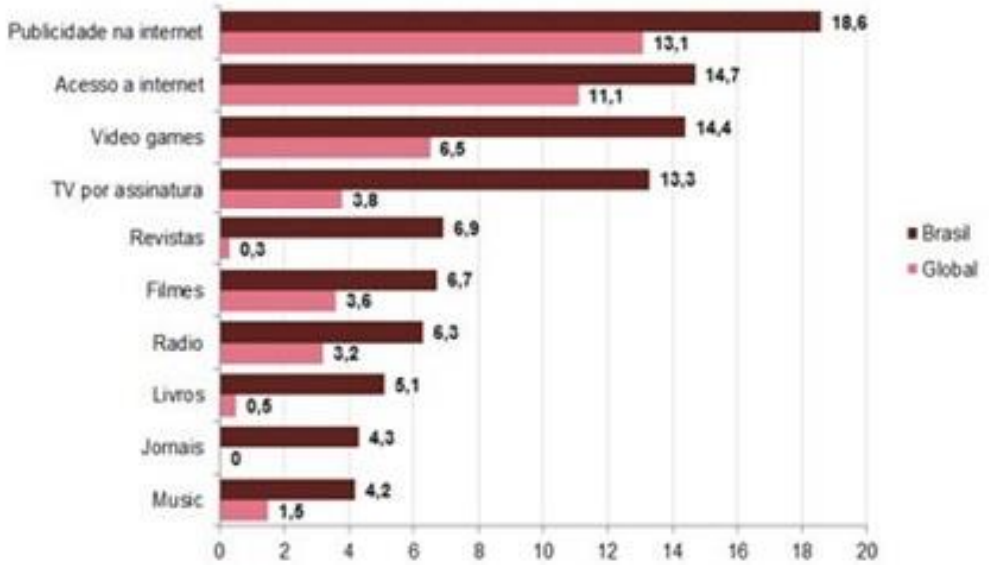
Com as diversas mudanças da rede mundial e na vida social como um todo, também houve uma enorme alteração na economia. McKinsey (2011) mostra a contribuição da Internet no PIB de 13 países desenvolvidos e emergentes no mundo, incluindo o Brasil. Apenas no ano de 2011, 3,4% de todo o PIB de países desenvolvidos foi influenciado pela utilização da Internet (MCKINSEY, 2011).

Com um crescimento constante e ainda com o enorme impacto na economia mundial, é possível imaginar que a Internet está impactando em todos os tipos de serviços que anteriormente não estavam conectados à rede mundial. Sendo assim, serviços que antes necessitavam de profissionais para desenvolvimento, hoje já são possíveis de serem usados por usuários comuns, graças ao uso da rede global.

2.3 O USO DA INTERNET NO BRASIL

O Brasil lidera o uso da Internet na América Latina, como também mostra grandes promessas para o futuro (PWC, 2016). A Figura 2 apresenta o resultado de uma pesquisa mostrando esta tendência. No resultado nota-se que o uso da Internet, nos mais diversos segmentos, está aumentando no Brasil.

Figura 2 - Crescimento por segmento anual da Internet de 2013 – 2017



Fonte: PWC (2016) (Modificada)

Através da Figura 2 (página 19) pode-se também perceber o crescimento considerável da publicidade na Internet, não só no Brasil, mas no mundo. Sendo assim, as empresas parecem estar utilizando a rede global cada vez mais para divulgar seus serviços e encontrar clientes. Esse aumento na divulgação permite inferir duas coisas: que a publicidade na Internet está funcionando e atraindo mais consumidores para as empresas, e que as pessoas estão consumindo mais produtos e serviços ofertados naquele meio.

2.4 O MERCADO DE EVENTOS NO BRASIL

É importante analisar também o impacto positivo direto na economia do país que os eventos sociais causam. Isso pode ser verificado na Figura 3:

Figura 3 – Crescimento no número de eventos e participantes no Brasil

Indicador	2001	2013	Taxa de crescimento do período - 2013/2001	Taxa de crescimento médio anual
Espaços para eventos	1.780	9.445	430,6%	14,9%
Organizadoras e agências de eventos no âmbito da pesquisa	400	2.784	596,0%	17,5%
Número de eventos por ano	327.520	590.913	80,4%	5,0%
Número de participantes por ano (pessoas)	79.849.376	202.171.787	153,2%	8,0%
Receita gerada pelos gastos dos participantes de eventos (R$)	31.429.563.653,00	99.258.344.738,24	215,8%	10,1%
Receita gerada pelos gastos com locações dos espaços para eventos (R$)	1.615.013.187,00	37.810.205.685,52	2241,2%	30,1%
Receita gerada pelos gastos das empresas organizadoras de eventos (R$)	3.986.172.874,00	72.215.303.847,43	1711,6%	27,3%
Empregos diretos	21.784	132.045	506,2%	16,2%
Empregos terceirizados	212.880	1.761.374	727,4%	19,3%
Empregos indiretos	703.992	5.680.257	706,9%	19,0%
RECEITA GERADA (R$)	**37.030.749.714,00**	**209.283.854.271,18**	**465,2%**	**14,0%**
EMPREGOS GERADOS	**938.656**	**7.573.676**	**706,9%**	**19,0%**

Fonte: ABEOC (2013) (Modificada)

Também é possível perceber que o número de empregos diretos aumentou em 2013 quase cinco vezes mais que em 2001. Adicionalmente, é possível notar o aumento dos números de empregos indiretos e terceirizados como decorrência dos diferentes eventos realizados entre 2001 e 2013 (ABEOC, 2013).

Percebe-se que a organização de eventos deixa de ser apenas uma atividade de lazer ou de comemorações (como existe no senso comum da sociedade). Ela passa a ser uma fonte de empregos e de renda. Por estas razões, ferramentas de tecnologia para auxiliar na organização de eventos tornam-se viáveis e possíveis.

2.4 O USO DA INTERNET NA GESTÃO DE EVENTOS NO BRASIL

Como discutido anteriormente, a utilização da Internet é cada vez mais comum na sociedade brasileira, e vem crescendo até mais do que em outros países, considerado um cenário mundial. O número de acessos é alto não apenas de usuários comuns, mas também de empresas. As mesmas estão cada vez mais utilizando a Internet para negócios (ABEOC, 2013).

Ainda de acordo com o relatório da ABEOC, em 2013, o meio de comunicação mais usado como forma de publicidade para a realização de eventos eram os *websites,* como é possivel ver na Figura 4 (página 22). Se esta constatação for considerada junto com a ideia representada pela Figura 2 (página 19), que mostra uma projeção de aumento de publicidade na Internet de quase 19% entre 2013 a 2017, é possível concluir que a melhor opção para construir uma aplicação que facilite a realização de eventos, por questões de popularidade e de adesão, seria através da criação de uma aplicação *web*.

Apesar da Figura 4 ser de um relatório de 2013, acredita-se que *websites* ainda predominem como mais utilizados na comunicação utilizada pelos espaços para eventos. Assim como meios que também utilizem e façam parte da Internet (como *Mobiles* e Redes Sociais) também possuam maiores tendências a crescerem e serem mais utilizados.

Figura 4 – Principais meios de comunicação utilizados pelos espaços para eventos

Fonte: ABEOC (2013)

2.5 A TECNOLOGIA USADA NA *WORLD WIDE WEB*

Nesta seção será abordada a *World Wide Web* (WWW), a plataforma que utiliza a Internet para o compartilhamento de documentos hipermídia[2].

2.5.1 A Proposta Inicial da Internet

Em 1989, Tim Bernes-Lee escreveu uma proposta para o CERN (Organização Europeia para a Pesquisa Nuclear), com o objetivo de demonstrar a importância de seu desenvolvimento de uma maneira focada no compartilhamento de documentos acadêmicos (W3C, 2017).

Tim Bernes-Lee sugeriu uma solução baseada em hipertextos[3] contendo referências para outros documentos. Para que esta solução pudesse ser implementada Tim Bernes-Lee desenvolveu o protocolo HTTP (*Hypertext Transfer Protocol*) para a transferência de documentos de hipertexto e a linguagem de marcação de hipertexto

[2] O conceito de hipermídia é um termo usado para o hipertexto que não está restrito apenas a elementos textuais. A hipermídia permite a integração de elementos como gráficos, vídeos, áudios, entre outros elementos. O termo hipermídia também foi primeiramente utilizado por Ted Nelson (W3C, 2017).
[3] O termo hipertexto foi cunhado por Ted Nelson nos anos 50, como sendo “Informações legíveis por humanos ligadas entre si de forma não restritiva” (W3C, 2017).

chamada HTML (*Hypertext Markup Language*) (INTERNET SOCIETY, 2017).

2.5.2 HTTP 1.1

O HTTP, ou Protocolo de Transferência de Hipertexto é um protocolo baseado na arquitetura cliente-servidor utilizado na *World Wide Web* para troca de informações (TUTORIALS, 2017). A comunicação neste protocolo faz uso do conceito de *request-reply,* o cliente realiza uma requisição ao servidor que por sua vez processa este pedido e envia uma resposta ao cliente (W3C, 2017).

O protocolo HTTP não mantém conexões ativas, assim que uma requisição é realizada, o cliente desconecta do servidor e aguarda uma resposta. Como consequência disso o HTTP não possui estado, tanto o cliente quanto o servidor não guardam informações entre diferentes requisições (TUTORIALS, 2017).

Para realizar uma requisição, é enviada uma mensagem ao servidor composta pelo método da requisição, seguida pela URI (*Universal Resource Indicator*), versão do protocolo e opcionalmente um cabeçalho com informações adicionais que o cliente pode informar (TUTORIALS, 2017).

2.5.3 HTML

Linguagem de Marcação de Hipertexto, ou HTML, é uma linguagem de marcação desenvolvida por Tim Bernes-Lee em 1989 para ser a linguagem padrão do protocolo HTTP. Ela é fortemente baseada no SGML (*Standard Generalized Markup Language*) e adiciona o conceito de âncoras para realizar ligações (W3C, 2017).

O HTML é uma linguagem de marcação, onde seus elementos são os blocos construtores dos *websites.* Atualmente é mantido pela *World Wide Web Consortium* (W3C), comunidade internacional fundada por Tim Bernes-Lee, voltada para o desenvolvimento de padrões de tecnologias *web* (W3C, 2017). A versão do HTML

utilizada no trabalho é a 5.

2.6 TECNOLOGIAS *WEB* SERVIDOR

Nesta seção são apresentadas tecnologias que permitem o desenvolvimento de *websites* dinâmicos.

2.6.1 O Surgimento da *Web* Dinâmica

No início o HTML era utilizado apenas para a criação de documentos estáticos com ligações para outros documentos, permitindo a navegação entre eles. Em 1993, com a criação da *Common Gateway Interface* (CGI), tornou-se possível a geração de conteúdo dinâmico por aplicações executando do lado do servidor.

O conteúdo dinâmico é montado em tempo real por meio do uso de uma linguagem que monta uma página HTML personalizada pela seleção de vários conteúdos armazenados em diferentes tecnologias.

Uma das primeiras linguagens utilizadas para o seu desenvolvimento foi a Perl, seguida por outras linguagens como Java, PHP, Ruby e Python (BRANAS, 2014). A *World Wide Web* utiliza o modelo cliente-servidor para a apresentação das páginas *web* (estáticas e dinâmicas). Neste modelo o navegador (o cliente) mostra uma página *web* que foi requisitada a um servidor (chamado de *servidor web*). Um servidor *web* é um tipo específico de servidor capaz de se comunicar com clientes utilizando o protocolo HTTP (PEARSON, 2017). A Figura 5 (página 25) mostra a evolução de algumas linguagens *Web* servidor.

Diferente da aplicação cliente, o lado do servidor nunca entra em contato com o usuário, e realiza tarefas como validações, analisa e processa dados fornecidos pelo cliente, notificações e as atividades dentro do banco de dados (BRANAS, 2014). Além das ferramentas mostradas na Figura 5 (página 25), uma nova tecnologia no desenvolvimento de aplicações *Web* tem chamado atenção nos últimos anos: NodeJS, uma plataforma para desenvolvimento de aplicações do lado servidor utilizando

JavaScript (NODESOURCE, 2017).

Figura 5 – Histórico de linguagens de *Web* servidor

Fonte: PINGDOM (2007)

2.6.2 NodeJs

NodeJs é uma plataforma escrita sobre o motor[4] JavaScript Chrome V8[5]. Utiliza um modelo orientado a eventos e I/O[6] não bloqueante, o que segundo sua documentação

[4] Um motor JavaScript é um programa ou interpretador que executa código JavaScript.
[5] Motor JavaScript de alto desempenho e *Open Source* utilizado pelo navegador Google Chrome, responsável por compilar e executar código-fonte JavaScript (GOOGLE, 2017).
[6] I/O refere-se à entrada/saída (*Input/Output*), termo é utilizado para designar as operações de requisição e resposta no servidor.

o torna leve e eficiente (NODEJS, 2017). Cantelon (2014) afirma que o motor V8 aumenta o desempenho do Node consideravelmente por conta da eliminação de intermediários e compila diretamente o código para linguagem de máquina (CANTELON, 2014).

NodeJs permite utilizar a linguagem *JavaScript* para o servidor em um modelo assíncrono, e foi projetado para criar aplicativos escaláveis e de alta *performance* (NODEJS, 2017). Publicado por Ryan Dahl em 2009, desde então, o NodeJs vem conquistando espaço como ferramenta para desenvolvimento de aplicações *web* (NODEJS, 2017). A versão do NodeJs utilizada no trabalho é a 6.10.2.

Uma aplicação Node é executada como um laço contínuo em um único processo e aguarda por eventos, como requisições, por exemplo (CANTELON, 2014). Tal característica o difere de outros modelos de programação onde cada conexão é tipicamente executada em um novo processo (CANTELON, 2014).

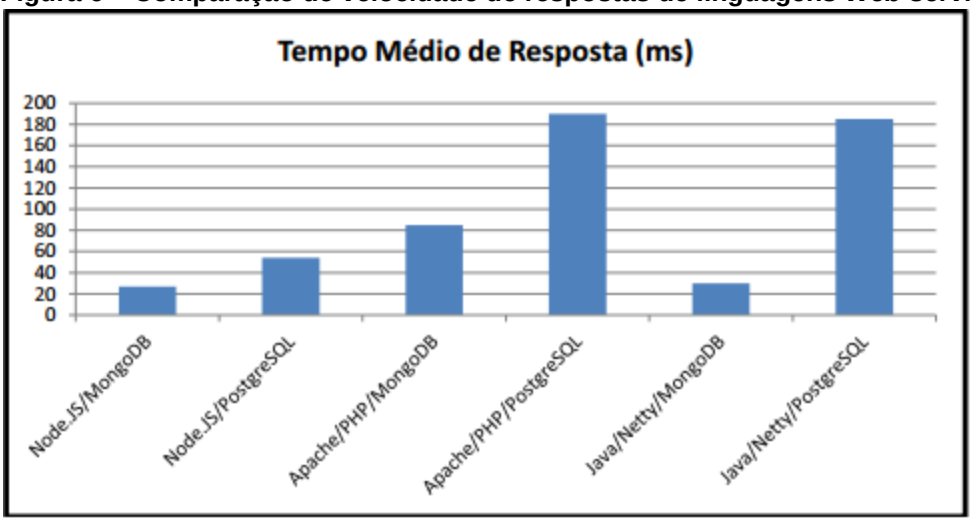
Outro aspecto relevante para a escolha da ferramenta é a grande comunidade envolvida no projeto, sendo o segundo projeto mais acompanhado no *website* GitHub (CANTELON, 2014). Além disso, apenas o NPM[7] (*Node Package Manager*) possui um repositório com mais de 450 mil módulos disponíveis para *download,* fornecendo soluções para problemas comuns no desenvolvimento de aplicações (NPMJS, 2017).

Em um estudo de desempenho realizado por Schroeder (2014), a combinação das tecnologias NodeJs e MongoDB (página 30) se mostraram mais performáticas se comparadas com outras combinações.

Para um melhor entendimento, a Figura 6 (página 27) exemplifica por meio de um gráfico o tempo médio de resposta das tecnologias NodeJS e MongoDB com relação a outras, como o uso de Apache, PHP e PostgreSQL:

[7] NPM é um gerenciador de dependências da plataforma NodeJs.

Figura 6 – Comparação de velocidade de respostas de linguagens *Web* servidor



Fonte: SCHROEDER (2014)

2.7 ARMAZENAMENTO DE DADOS

Esta seção aborda tanto o modelo relacional para o armazenamento de dados como também o modelo não relacional. Também é apresentado o modelo orientado a documentos, que é o tipo de banco da tecnologia MongoDB.

2.7.1 Modelo Relacional

Este modelo já existe há mais de quarenta anos, e é composto por tabelas e relações (DARWEN, 2010). Utilizando conceitos da álgebra relacional para tratar conjuntos de objetos e suas relações, foi criado com o propósito de satisfazer as necessidades de dados na época.

A linguagem *Structured Query Language*[8] (SQL) é utilizada para consultar, manipular e definir bancos relacionais com o uso de declarações (USN, 2016), podendo ser resumida em *Data Definition Language*[9] (DDL) e *Data Manipulation Language*[10] (DML).

[8] Criada pela IBM na década de 1970, e muito difundida desde então (USN,2016).
[9] Linguagem de Definição de Dados (*USN*, 2016).
[10] Linguagem de Manipulação de Dados (*USN*, 2016).

Na DDL, é possível encontrar comandos para criar objetos, deletar e modificar a estrutura de um objeto já existente (USN, 2016). Portanto, a criação de tabelas e de *index*[11] encontram-se nesta linguagem. Enquanto na *DML*, comandos como adicionar, ler, atualizar e deletar dados (*CRUD*[12]) fazem parte da linguagem.

2.7.2 Modelo Não Relacional

Quando existiam apenas os Modelos Relacionais, não eram necessários bancos com tanto foco na agilidade e flexibilidade, devido à pequena quantidade de dados utilizados (NTNU, 2014). Nos dias de hoje, havendo a necessidade de manipular dados de proporções de *Big Data*[13], são também necessários bancos que tenham mais escalabilidade, e sejam mais ágeis e flexíveis (STRAUCH, 2009).

Modelos não relacionais não foram feitos para substituírem totalmente os relacionais, pois ainda há a preferência para modelos desse tipo em projetos que visam manter os padrões *ACID*[14], e sem muita necessidade de expansão (NTNU, 2014).

Como o aplicativo foi planejado para ser flexível e aberto para futuros melhoramentos, optou-se por um Banco de Dados não relacional que foge dos padrões *ACID* determinados por bancos relacionais tradicionais. Entre os demais bancos *NoSQL*[15], foi escolhido o *MongoDB* que é um banco não relacional orientado a documentos.

2.7.3 O Modelo Orientado a Documentos

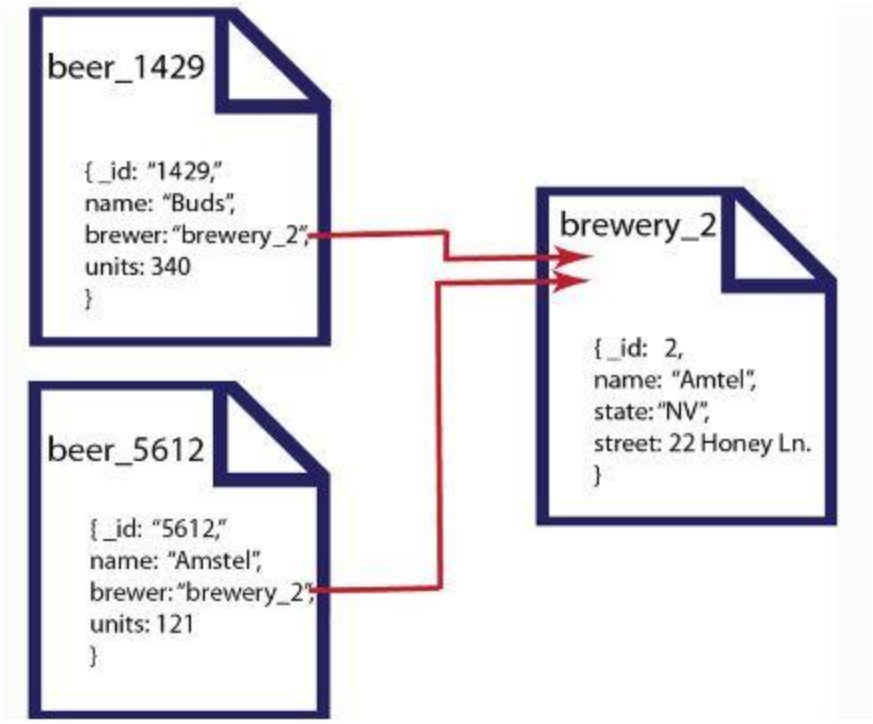
Cada registro armazenado nesse modelo é tratado como um documento, sendo estes documentos armazenados em coleções (MongoDB, 2017). Cada documento é

[11] Indicadores utilizados nas diversas linguagens de programação.
[12] Acrônimo para *Create*, *Read, Update, Delete* (Criar, Ler, Atualizar, Deletar).
[13] Grande quantidade de dados, estruturados e não-estruturados, exigindo mais capacidades de armazenamento e processamento.
[14] Acrônimo de transações que seguem padrões de Atomicidade, Consistência, Isolação e Durabilidade.
[15] Bancos que não são apenas *SQL* (METADATA, 2011).

tratado de forma independente, sem separação do esquema de dados dos campos em si. Assim há a possibilidade de utilização de referências e relações a outros documentos, pois estes já vão estar referenciados.

A Figura 7 mostra um exemplo de armazenamento em documentos. É possível avaliar na prática os documentos '*Beer*' fazendo parte de uma coleção de cervejas, no caso, '*Brewery*'. Sendo assim, todas as cervejas já terão relação, possuindo uma chave estrangeira *'brewery_2'*, fazendo referência à sua coleção.

Figura 7 – Exemplo de armazenamento em documentos



Fonte: COUCHBASE (2017)

Ao armazenar os dados dessa maneira, existem diferenças significativas em aspectos de desempenho. Ao tratar documentos como unidades independentes, a implementação de armazenamento distribuído se torna mais fácil. Outro aspecto importante é a linguagem de consulta ser mais natural, em contraste com as sentenças complexas de *SQL* (ULB, 2015).

Diferente das propriedades ACID de bancos relacionais, bancos orientados a documentos seguem o teorema *CAP - Consistency, Availability, Partition Tolerance* (ULB, 2015), onde:

- *Consistency* (consistência)*:* Padrão que exige que todos os dados do banco estejam consistentes e independentes de sua localização.
- *Availability* (disponibilidade): Busca tornar todos os serviços disponíveis.
- *Partition Tolerance* (tolerância a falhas): Foca no tratamento e cuidados com possíveis falhas que passam ocorrer no banco.

Aumentando disponibilidade e tolerância a falhas*,* se perde em consistência. Enquanto que aumentando a tolerância a falhas e consistência, se perde em disponibilidade (ULB, 2015).

2.7.4 MongoDB

Escolhido para este projeto por ter uma curva de aprendizagem pequena, ser gratuito, e ser o banco de dados orientado a documentos mais utilizado atualmente (MONGODB, 2017). O MongoDB é multiplataforma e de código aberto e foi desenvolvido em C++ em 2007, tendo sua primeira versão lançada em 2009 (NAYAK, 2013).

Seus documentos são armazenados em BSON*,* que são a versão binária de documentos JSON (BANNER, 2012)*.* BSON é capaz de armazenar vetores, objetos embutidos e JSON. Ele também contém uma lista ordenada de elementos, que consiste no nome do campo, tipo e valor destes (NAYAK, 2013).

É possível usar referências para outros documentos, com dados normalizados, ou manter todas as informações agrupadas. Existem também ferramentas para validação de documentos. Isto é necessário caso uma implementação necessite que documentos de uma determinada coleção possuam obrigatoriamente algum campo específico (BANNER, 2012).

2.8 *FRAMEWORKS* UTILIZADOS

Com o objetivo de facilitar e diminuir o tempo necessário para o desenvolvimento da aplicação proposta, foi escolhido o conjunto de *frameworks* conhecido como MEAN *Stack* (HOLMES, 2013). O nome deste conjunto deriva de suas ferramentas: Mongoose,

Express, Angular e NodeJS.

Frameworks prometem maior produtividade e menor tempo de desenvolvimento de aplicativos por meio do *design* e reutilização de código (RIEHLE, 2000, p2). Para Riehle, *frameworks* representam o domínio da aplicação em um *design* abstraído e possuem implementações reutilizáveis (RIEHLE, 2000, p8). As próximas subseções abordam separadamente cada um dos *frameworks* utilizados neste projeto.

2.8.1 Mongoose

Mongoose é um ODM[16] do NodeJS desenvolvido sob a licença MIT (Massachusetts Institute of Technology) (MONGOOSE, 2017). Este módulo possui uma abstração da conexão com o banco de dados MongoDB e adiciona funcionalidades de validação e conversão de objetos para documentos, além de permitir a definição de esquemas flexíveis para documentos a serem utilizados na aplicação (MONGOOSE, 2017).

A ferramenta Mongoose fornece aos desenvolvedores a capacidade de modelar objetos e armazená-los como documentos MongoDB. Apesar do MongoDB utilizar estrutura de dados de esquema livre[17], a ferramenta Mongoose permite a possibilidade tanto da utilização de um esquema de dados mais definido como um esquema flexível (HAVIV, 2014). A versão do Mongoose utilizada é a 4.10.8.

Esta ferramenta possui funcionalidades adicionais ao *driver* nativo do Node.Js para MongoDB, e é utilizada para acessar as bases de dados do MongoDB e realizar operações de inserção, alteração, remoção e busca (DAYLEY, 2014). Foi idealizado para trabalhar num ambiente assíncrono, e fica mais fácil o entendimento do modelo ao analisar um exemplo da definição de um de objeto qualquer na Figura 8 (MONGOOSE, 2017).

Na Figura 8, são criadas as variáveis básicas de um *e-mail*, como exemplo *'title'*

[16] *Object-Document Mapper*, é responsável pelo mapeamento objeto documento.

[17] Esquemas livres são mais flexíveis do que esquemas de banco de dados, que são estruturas formais para um determinado sistema de gerenciamento de banco de dados (HAVIV, 2014).

sendo o título do mesmo, '*body'* sendo o corpo da mensagem, e *'date'* sendo a data de envio, sendo '*BlogPost'* uma instância de '*Schema'*:

Figura 8 – Exemplo de modelagem de um objeto utilizando Mongoose

```
var Comments = new Schema({
    title     : String
  , body      : String
  , date      : Date
});

var BlogPost = new Schema({
    author    : ObjectId
  , title     : String
  , body      : String
  , buf       : Buffer
  , date      : Date
  , comments  : [Comments]
  , meta      : {
      votes : Number
    , favs  : Number
  }
});

var Post = mongoose.model('BlogPost', BlogPost);
```

Fonte: MONGOOSE (2017)

Uma das razões para o MongoDB ter sido escolhido como banco de dados no MEAN stack é porque o Moongoose consegue interpretar e enviar arquivos JSON, fazendo com que ele se adapte ao Express e Node.Js (DAYLEY, 2014).

2.8.2 Express

A página oficial do Express o descreve como "um *framework* NodeJs minimalista e flexível, provendo um conjunto de recursos robustos para a construção de aplicações web" (EXPRESS, 2017). Foi lançado em novembro de 2010, desenvolvido inicialmente por TJ Holowaychuk, é atualmente mantido pela StrongLoop, estando na versão 4 (EXPRESS, 2017).

Segundo Brown (2014), uma das características mais atraentes do *framework* Express é sua natureza minimalista. Ao contrário da maioria dos *frameworks* em que são

acoplados vários módulos que muitas vezes não são utilizados, o Express utiliza a ideia de “menos é mais”, oferecendo um ambiente enxuto, porém com a capacidade de adicionar módulos para necessidades específicas (BROWN, 2014, p2).

O *framework* possui implementações para a manipulação de rotas, *cache*, *cookies* e outros recursos que podem ser adicionados de acordo com a necessidade (HANH, 2016). Este *framework* vem sendo muito utilizado devido à facilidade de uso e a possibilidade de criação de servidores *web* de forma rápida. Sendo minimalista, muitos desenvolvedores criaram pacotes compatíveis para resolver dificuldades encontrado no desenvolvimento de alguma aplicação *web* caso não seja possível apenas com o Express, como o auxílio ao utilizar parâmetros por URL[18], dados por POST[19] e sessões (EXPRESS, 2017). A versão do Express utilizada é a 4.15.1.

2.8.3 Angular

Desenvolvido em 2009 por Misko Hevery e Adam Abrons, Angular é um *framework JavasScript* de código aberto do lado cliente que busca promover alta produtividade e experiência no desenvolvimento *web* (ANGULAR, 2017). Segundo a página oficial, o Angular é o que seria o HTML caso este fosse criado para o desenvolvimento de páginas dinâmicas (ANGULAR, 2017).

Para alcançar este objetivo, o *framework* estende o vocabulário HTML para facilitar o trabalho do desenvolvedor (BRANAS, 2014, p. 8). Desta forma são criadas aplicações com componentes reutilizáveis e de fácil manutenção, reduzindo de forma significativa o tamanho do código final (BRANAS, 2014). Trabalhando apenas no lado cliente, ele também se torna uma ferramenta muito flexível, sendo possível se adaptar com qualquer tecnologia de servidor (ANGULAR, 2017).

O *framework* é composto por diversas bibliotecas, sendo algumas parte do seu núcleo e obrigatórias, e outras opcionais. De forma simplificada, para desenvolver

[18] A URL (*Universal Resource Locator)* se refere ao nome do endereço de um *website* qualquer (BALANCE, 2017).
[19] Método de resposta que envia dados a serem processados para um recurso especificado (W3SCHOOLS, 2017).

aplicações Angular são criados documentos HTML chamados *templates* utilizando marcações nativas do Angular. Para cada *template* criado deve-se implementar uma classe com o decorador *component* e adicionar a lógica por trás do *template*. Pode se criar também *services*, classes com funções bem definidas que não necessitam de uma representação visual (ANGULAR, 2017). A versão Angular utilizada é a 4.

2.8.4 Bootstrap

Criado em 2010 por desenvolvedores do Twitter, bootstrap é um *framework* gratuito de código aberto voltado para o desenvolvimento de páginas estáticas e dinâmicas (BOOTSTRAP, 2017). O *framework* contém estilos pré-definidos em HTML e CSS (*Cascading Style Sheets)* para tipografia, formulários, botões e outros componentes de interface, assim como extensões opcionais em JavaScript (CIMO, 2015). A versão utilizada no trabalho é a 3.3.7.

Ele permite a estilização rápida de documentos HTML ao fornecer modelos pré-definidos que podem ser atribuídos a elementos através de classes. É muito utilizado por permitir um desenvolvimento ágil e possuir um desenvolvimento com foco inicial em aparelhos celulares (*mobile first*)*,* conceito que permite facilmente a implementação de conteúdos responsivos, de forma a poderem ser visualizados tanto em dispositivos portáteis com telas pequenas, como computadores pessoais com telas maiores (BOOTSTRAP, 2017).

A instalação do *framework* consiste em referenciar os arquivos de estilo e *JavaScript* em um documento HTML, após isso basta atribuir aos elementos da página classes pré-definidas que podem ser encontradas na documentação do *framework.*

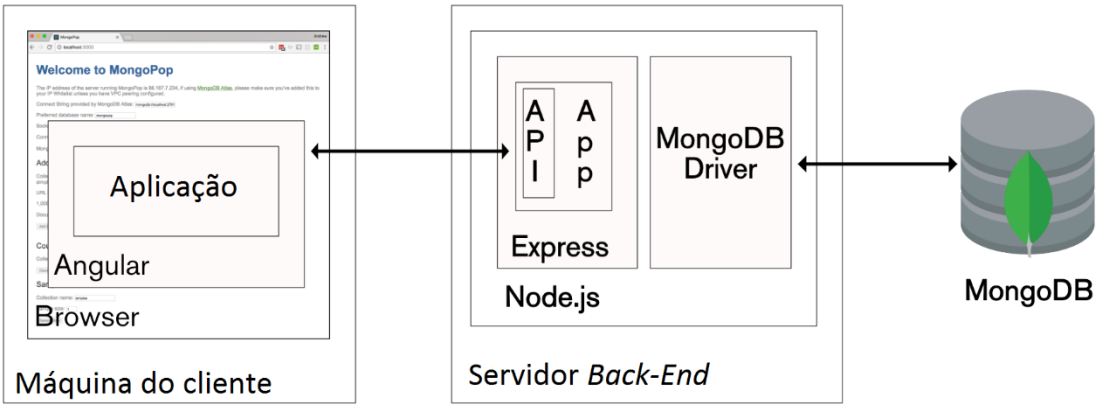
2.9 INTEGRAÇÃO DAS FERRAMENTAS

Como relatado anteriormente, *framework*s promovem maior agilidade para o desenvolvimento de aplicações. O MEAN *stack*, conjunto de ferramentas escolhido para implementação da aplicação proposta, utiliza a linguagem *JavaScript* tanto no lado

servidor quanto no lado cliente, tornando a curva de aprendizagem ainda menor (HOLMES, 2013).

A Figura 9 exemplifica como ocorre a integração entre as ferramentas utilizadas. O servidor da aplicação é construído utilizando a plataforma NodeJS e os *framework*s Express e Mongoose, enquanto que aplicação cliente é desenvolvida em Angular.

Figura 9 – Exemplificação do MEAN *Stack*



Fonte: MONGODB (2017)

2.9.1 API e Arquitetura REST

Como foi observado na Figura 9, uma API (*Application Programming Interface*) construída com o *framework* Express é utilizada para comunicação entre a aplicação cliente (Angular) e aplicação servidor (NodeJS).

Segundo Barlett, uma API é um conjunto de pontos de acesso públicos e endereçáveis que permitem uma aplicação interagir com outros sistemas ou aplicações (BARLETT, 2016).

Bretz aconselha a utilização do estilo arquitetural REST (*Representational State Transfer*) ao implementar uma API sob o MEAN *stack*, com o objetivo de facilitar o processo de integração com o Angular, além de aproveitar as vantagens oferecidas pela arquitetura (BRETZ, 2014).

O termo REST foi cunhado por Roy Fielding em 2000 na sua tese de doutorado

ao propor um estilo arquitetural para sistemas distribuídos com o objetivo de melhorar a escalabilidade, desempenho, portabilidade, entre outros fatores (DOGLIO, 2015). Aplicações que desejam adotar o estilo arquitetural REST devem seguir as seguintes restrições:

- *Client-server* – fazer uso de uma arquitetura cliente servidor, como o HTTP por exemplo.
- *Stateless* – toda requisição feita ao servidor deve conter toda informação necessária para seu processamento, não armazenando informações entre requisições.
- *Cacheable* – respostas a uma requisição devem ser explicitamente definidas como cacheable quanto aplicável (recursos como imagens ou arquivos estáticos podem ser reaproveitados, dispensando a necessidade de uma nova transferência).
- *Uniform Interface* – uma interface uniforme de acesso, independente do tipo de dispositivo que a acessa.
- *Layered System* – restringe o acesso dos componentes ao seu escopo, possibilitando escalabilidade futura e simplificando a complexidade geral do sistema.
- *Code-on-demand* – restrição considerada opcional, referente a disponibilização de código de acordo com a necessidade.

Aplicações REST comumente fazem uso do protocolo HTTP, embora não restrito a este, como camada de transporte (DOGLIO, 2015). Os pontos de acesso são definidos através de uma URI em conjunto com os métodos HTTP, de acordo com a funcionalidade de cada ponto de acesso e podem retornar como resposta documentos JSON (DOGLIO, 2015).

3 METODOLOGIA

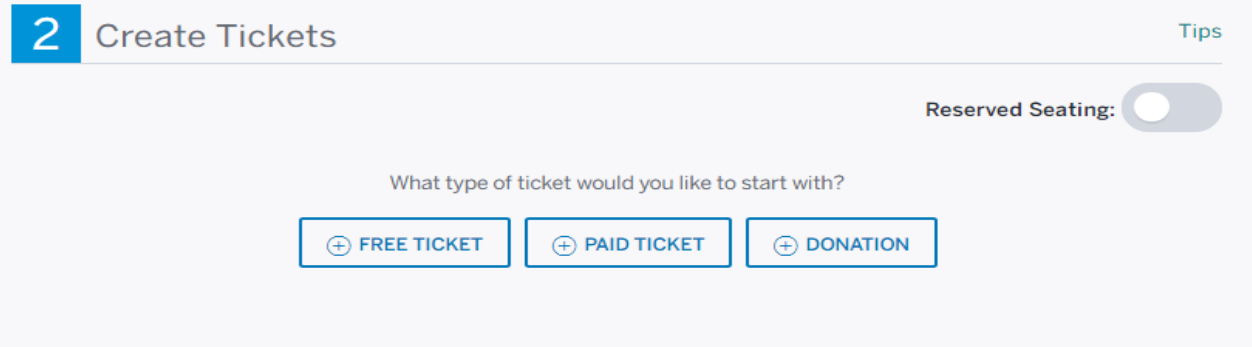
Este capítulo apresenta uma breve análise de outros *software* similares, certas limitações que o projeto não busca resolver, como também a metodologia de desenvolvimento escolhida.

3.2 SISTEMAS SIMILARES

Após pesquisas por sistemas que poderiam solucionar o que este projeto busca resolver, foram encontradas duas ferramentas similares à ideia proposta. A primeira foi o *website* EventBrite[20], que possibilita a criação, gerenciamento e busca por eventos em qualquer cidade. Com base na cidade em que o usuário acessar, serão obtidos eventos próximos com informações básicas (como preço e horário), além de onde o evento ocorrerá.

Na sessão de criação e gerenciamento de eventos, é possível criar desde sua apresentação, horário e local, como criar os tipos de ingressos a serem vendidos (grátis, pagos, doações). A Figura 10 mostra um exemplo dos ingressos na criação de um evento.

Figura 10 – Tipos de ingressos na criação de um evento no EventBrite



Fonte: EVENTBRITE (2017)

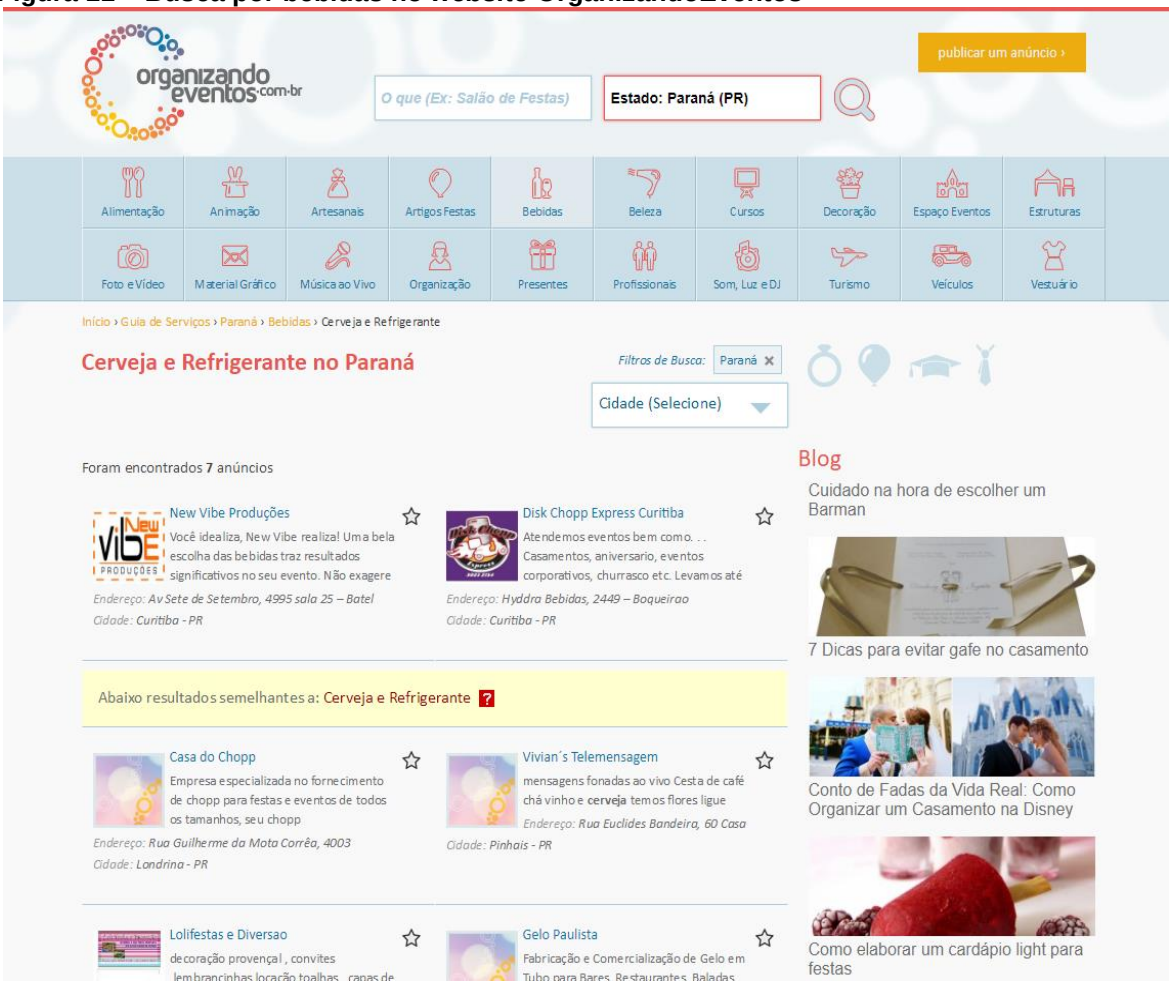
Depois do evento ser criado, é possível gerenciar informações como: os dados da página *web* criada para o evento após sua criação, o número de ingressos vendidos, e as visitações que o evento recebeu de interessados.

[20] https://www.eventbrite.com.br

O *website* EventBrite busca ajudar na criação de eventos e gerenciá-los dentro de sua hospedagem, enquanto a proposta deste trabalho é conectar fornecedores para se criar um evento.

Porém, a segunda ferramenta encontrada, o *website* OrganizandoEventos[21], possui uma abordagem mais parecida com o pretendido neste projeto. A Figura 11 mostra o resultado de uma busca por bebidas no estado do Paraná.

Figura 11 – Busca por bebidas no *website* OrganizandoEventos



Fonte: EVENTOS (2017)

A abordagem oferecida possui similaridades já que o *website* conecta qualquer pessoa que queira criar um evento com fornecedores em sua região. Contudo, existe um grande foco nos fornecedores e não nos produtos que eles vendem. Por exemplo, ao

[21] https://www.organizandoeventos.com.br/

invés de haver imagens das bebidas oferecidas, as que aparecem são dos próprios fornecedores, com suas logos e informações de contato.

Em resumo, essa proposta apenas conecta fornecedores com criadores de eventos, e divulga serviços.

3.3 RESTRIÇÕES DO PROJETO

A primeira e original proposta deste trabalho visava não apenas conectar criadores de eventos com fornecedores de produtos e serviços, mas também faria a criação e gerenciamento dos eventos. Seria possível escolher todos os produtos, locais e serviços necessários para a realização, e no final haveriam transações diretas entre fornecedores e criadores do evento.

Tal proposta seria mais completa, porém houveram muitas complicações para se colocar em prática e o tempo necessário para o desenvolvimento seria mais longo. Itens como um sistema de avaliação entre criadores de eventos e fornecedores de serviço também foram previstos para esta proposta, mas também não foram desenvolvidos pelo curto prazo de tempo.

Seria necessário garantir que todos os serviços listados pelos fornecedores no *website* fossem entregues, assim como todos criadores de eventos concluíssem suas compras. Porém seria muito difícil gerar uma logística ou prover recursos, caso problemas como esse surgissem nas transações entre os usuários. Portanto, foi colocada uma plataforma mais simples, onde tais tarefas não dependem exclusivamente do *website* para serem concluídas, e apenas lida com a conexão entre os fornecedores de serviços e os criadores de eventos.

Existe a possibilidade da continuação da aplicação depois de ser testada diretamente com futuros usuários, e que prove ser útil conseguindo realizar todas as propostas idealizadas nas motivações para o desenvolvimento do projeto.

3.4 METODOLOGIA DE DESENVOLVIMENTO

Dedicar mais tempo na metodologia de um projeto pode dividir os custos finais, como também economizar tempo valioso no desenvolvimento, o qual poderia ser utilizado no desenvolvimento do *software* em questão (SUTHERLAND, 2014).

> A maioria dos desenvolvimentos de *software* eram feitos no método Cascata, onde o projeto era completamente distinto em diferentes estágios e movia-se passo a passo em direção ao produto final para os consumidores de *software*. O processo era devagar, imprevisível, e frequentemente se tornava um produto que os consumidores não queriam ou nem pagariam para utilizar (SUTHERLAND, 2014, tradução nossa)

Considerando o conceito acima no planejamento, gerenciamento e desenvolvimento deste projeto, foi utilizado o método de desenvolvimento Scrum (SUTHERLAND, 2014).

Criado por Jeff Sutherland em 1990 foi idealizado como um método coletivo, dinâmico, e que buscasse facilitar o trabalho de desenvolvedores (LAYTON, 2015). Todo o projeto que usa Scrum é realizado de forma incremental, dentro de curtos períodos de tempo e com tarefas definidas focadas em objetivos com mais prioridades, denominadas *Sprint* (SUTHERLAND, 2014).

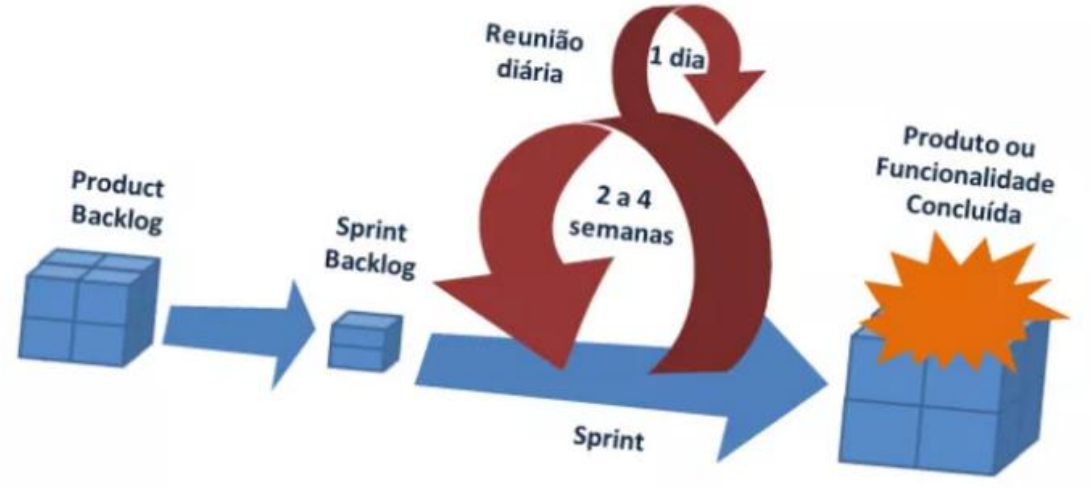
O grupo do projeto é dividido entre três principais atores, sendo eles: *Product Owner*, *Scrum Master* e Time *Scrum* (LAYTON, 2015). O *Product Owner* é encarregado de definir e gerenciar o projeto como um todo. O *Scrum Master* gerencia e ensina a utilização do *Scrum* dentro do Time *Scrum*, que são representados pelos grupos de programadores que são responsáveis pela codificação e manutenção técnica do projeto (SUTHERLAND, 2014).

Ao iniciar o projeto, o *Product Owner* é encarregado de fazer o *Product Backlog*, que possui todas as funcionalidades que precisarão existir no projeto (LAYTON, 2015). Depois da determinação do produto, o Time *Scrum* realiza uma *Sprint Planning*, onde observam e selecionam itens do *Product Backlog* a serem implementados na *Sprint*. Esta focará na realização de todos os itens definidos de acordo com suas prioridades definidas (LAYTON, 2015).

São realizadas reuniões diárias chamadas de *Daily Scrum*, onde o Time *Scrum* descreve o que já foi feito. Nestas é possível redefinir prioridades na meta de acordo com a experiência adquirida na *Sprint* anterior, como também definir as próximas tarefas da *Sprint* (LAYTON, 2015).

Após o término de uma *Sprint*, é realizada uma *Sprint Review*, onde o resultado parcial é possível de ser apresentado para clientes ou usuários que informaram os requisitos para o *Product Owner*. Na *Sprint Review* podem ser realizadas até mudanças mais profundas, como remover ou adicionar outras funcionalidades (LAYTON, 2015). É possível analisar na Figura 12 uma exemplificação do ciclo completo do *Scrum*:

Figura 12 – Ilustração do ciclo do Scrum



Fonte: MINDMASTER (2014)

4 DESENVOLVIMENTO DA APLICAÇÃO

O capítulo aborda algumas etapas elaboradas na realização do projeto, sendo possível observar mais coesão nas ferramentas utilizadas. É apresentado também os resultados obtidos com o desenvolvimento do projeto, utilizando telas do mesmo para mostrar como ficaram algumas das funcionalidades criadas.

4.1 FUNCIONAMENTO DAS FERRAMENTAS

O desenvolvimento da aplicação foi iniciado com uma análise dos sistemas similares encontrados, mencionados no capítulo da Metodologia (página 37). Com base nesses sistemas e no tempo disponível para o desenvolvimento foram definidas as funcionalidades básicas da aplicação, que por sua vez foram adicionadas ao *Product Backlog*. Após o levantamento das funcionalidades a serem implementadas, estas foram subdivididas em *sprints*, que foram implementadas de forma incremental.

Como relatado no Referencial Teórico (página 35), uma aplicação *Mean stack* é constituída por uma API no lado servidor e uma aplicação cliente em angular. Para alcançar este objetivo foi construída uma API REST em NodeJS utilizando os *framework* Express e Mongoose. A aplicação Angular realiza requisições a API desenvolvida, que por sua vez processa a requisição e retorna um documento JSON.

Algumas etapas básicas são repetidas para o desenvolvimento de cada funcionalidade implementada. Estas são a definição de rotas como ponto de acesso a API e a codificação de *controllers* e *models* no lado servidor, além da criação de *services* e *components* para a aplicação Angular.

Os *controllers* são *scripts* que possuem funções agrupadas de acordo com seu contexto com o objetivo de se manter uma aplicação em camadas. Os *models* objetos definidos a partir de esquemas de dados utilizando o *framework* Mongoose, permitindo a utilização de funcionalidades atreladas ao banco de dados (MongoDB) como inserção, atualização e remoção de elementos.

4.2 *PRODUCT BACKLOG*

Como foi visto no capítulo 3, o primeiro passo do Scrum é o *Product Backlog*, no qual são acatadas e colocadas em um documento as funcionalidades a serem implementadas nos *Sprints*. É possível analisar o *Product Backlog* no Quadro 1.

Quadro 1 – *Product Backlog* do projeto

Product Backlog				
N°	Item	Meta	Estimativa (dias)	*Sprint*
1	Preparar repositório, ferramentas principais e implantação	Criar repositório utilizando o Github, instalação das principais ferramentas Mongoose, Express, Angular, Node e Bootstrap, implantação no Heroku	2	1
2	Desenvolver páginas estáticas	Desenvolvimento de páginas estáticas iniciais, quem somos, e identidade da página	1	1
3	Cadastro de usuário	Implementar funcionalidade para cadastro de usuários	1	1
4	Autenticação	Implementar funcionalidade para login de usuários e restrição de conteúdo	1	2
5	Alteração dos dados cadastrais	Desenvolver funcionalidade para alterar os dados cadastrais de usuários	1	2
6	Formulário de contato	Implementar formulário de contato com envio de e-mail	1	2
7	Recuperação de senha	Desenvolver funcionalidade para recuperação de senha de usuários	1	2
8	Confirmação de conta	Alterar o cadastro de usuários para forçar a ativação da conta.	1	2
9	Cadastro de anunciante	Implementar funcionalidade para atualização de conta de usuário comum para anunciante	2	3
10	Edição de informação dos anunciantes	Criar uma opção para alterar os dados cadastrais de pessoas jurídicas	1	3
11	Página de anunciante	Desenvolver página de modelo para anunciantes	1	3

(Continuação)

Product Backlog				
N°	Item	Meta	Estimativa (dias)	*Sprint*
12	Troca de mensagens	Adicionar a funcionalidade para troca de mensagens entre usuários (envio, remoção e respostas)	2	4
13	Paginação das mensagens	Implementar a paginação de mensagens	1	4
14	QR Code	Desenvolver um modo de cada anunciante possuir seu código QR para seus endereços	1	4
15	Controle de produtos	Implementar funcionalidade para anunciantes adicionarem, listar, alterar e apagar produtos e serviços	2	5
16	Paginação de itens para anunciantes	Adicionar a paginação de itens para anunciantes	1	5
17	Paginação de itens para usuários	Implementar listagem, busca e paginação de produtos e serviços como cards por usuários	1	5
18	Página de Eventos	Desenvolver página modelo para criação e listagem de eventos	1	6
19	Criação de Eventos	Implementar funcionalidade para criação de eventos com data e título	2	6
20	Inserção e exclusão de itens ao evento	Desenvolver funcionalidade que permita o usuário inserir e remover itens anunciados aos seus eventos	2	6
21	Verificação disponibilidade de item	Implementar funcionalidade para enviar pedido de disponibilidade de item ao anunciante	2	7
22	Página para pedidos	Desenvolver página modelo para listagem de pedidos de clientes a fornecedores	1	7
23	Confirmação e Rejeição de Item	Desenvolver funcionalidade para que os fornecedores tenham acesso aos itens pedidos e possam confirmar ou rejeitar pedidos	2	7

Fonte: Autoria própria

Para armazenar os arquivos fonte do projeto, foi utilizado o Github[22], uma plataforma de desenvolvimento de código aberto, onde é possível revisar, gerenciar e construir um projeto compartilhado com outras pessoas. O Heroku[23] foi utilizado como hospedagem e implementação do *website*, descartando possíveis problemas de infraestrutura.

4.3 *SPRINTS*

Nesta seção, as *Sprints* realizadas no projeto são apresentadas, bem com algumas explicações de como foram feitas tais tarefas com exemplos de códigos.

4.3.1 *Sprint* 1

Como já explicado no funcionamento do Scrum, é realizada uma *Sprint Backlog* antes da realização das *Sprints*. O primeiro passo do projeto foi a inicialização do repositório, instalação das ferramentas e implementação da funcionalidade para cadastro de usuários. O Quadro 2 demonstra como foi a primeira *sprint*:

Quadro 2 – *Sprint 1 Backlog*

***Sprint 1* Backlog**	
Item	**Tarefas**
Criação do repositório utilizando o Github, instalação das principais ferramentas Mongoose, Express, Angular, Node e Bootstrap, implantação no Heroku	Criar um repositório no Github
	Instalar ferramentas principais
	Implantar sistema no Heroku
Desenvolver páginas estáticas	Desenvolver páginas estáticas iniciais, quem somos, e identidade da página

[22] https://github.com/
[23] https://www.heroku.com/what

(Continuação)

***Sprint 1* Backlog**	
Item	**Tarefas**
Cadastro de usuário	Implementar funcionalidade para cadastro de usuários

Fonte: Autoria própria

Nesta *sprint* foi necessária a instalação da biblioteca bcryptjs, uma implementação do algoritmo de hash *bcrypt* para NodeJs com o objetivo de realizar o hash nas senhas armazenadas. Isso possibilita haver mais segurança caso alguém consiga ter acesso ao banco de dados, pois as senhas estarão protegidas.

4.3.2 *Sprint* 2

O segundo *Sprint* foi responsável pela realização da maior parte das funcionalidades de gerenciamento de conta dos usuários. Também foram implementadas formas para usuários recuperarem suas senhas e confirmarem suas contas por *e-mail*. Como o desenvolvimento desta *Sprint* utilizou funcionalidades relacionadas ao envio de *e-mails*, também foi desenvolvida a seção para permitir o contato com o responsável pela aplicação.

Quadro 3 – *Sprint 2 Backlog*

***Sprint 2* Backlog**	
Item	**Tarefas**
Autenticação	Preparar servidor para autenticar usuários
Alteração dos dados cadastrais	Criar opção de alterar informações de usuários
E-mail para contato	Implementar o envio de e-mails para contato
Recuperação de senha	Realizar a recuperação de senhas por e-mail
Confirmação de conta	Criar modo de usuários confirmarem suas contas por e-mail
	Fazer com que conteúdos restritos sejam apenas vistos pelo usuário responsável proprietário das informações

Fonte: Autoria própria

Foi necessária a utilização do JWT (*JSON Web Token*) como forma de autenticação de usuários. JWT é um padrão aberto (RFC 7519) que define um método para representar reivindicações de forma segura entre duas partes.

Para isso foram utilizados os pacotes *passport* e *passport-jwt*, através de uma estratégia de autenticação, como pode ser observada na Figura 13. Um ponto de acesso foi criado para a autenticação de usuário recebendo o nome de usuário e a sua senha. Caso sejam válidos, a API retorna um *token*. A aplicação angular armazena este *token*, que por sua vez é incluído em cada requisição autenticada realizada.

Figura 13 – Exemplo de estratégia de autenticação

```
module.exports = function(passport){
    let opts = {};
    opts.jwtFromRequest = ExtractJwt.fromAuthHeaderWithScheme('jwt');
    opts.secretOrKey = config.secret;
    passport.use(new EstrategiaAutenticacao(opts, (jwt_payload, done) => {
    Usuario.usuarioPortId(jwt_payload.data._id, (err,usuario) => {
    If(err){
        return done(err, false);
    }
    If(usuario){
    return done(null, usuario);
    }    else {
        return done(null, false);
    }
    });
}));
}
```

Fonte: Autoria própria

A Figura 14 possui um exemplo de uma rota protegida, onde é necessário que o usuário envie um *token* para que seja verificado e autenticado.

Figura 14 – Rota autenticada

```
Router.post('/:id',passport.authenticate('jwt', {session:false}), ItemController.alterar);
```

Fonte: Autoria própria

Para o envio de *e-mails* dentro da aplicação foi utilizado o módulo do *Node*

Nodemailer. A Figura 15 exibe a função de exemplo responsável pelo envio de *e-mail* e as configurações necessárias para tal.

Figura 15 – E-mail *Model*

```
module.exports.enviarEmail = function(info, callback){
    let transporter = nodemailer.createTransport({
    service: 'gmail',
    auth:    {
            user: 'exsm2017@gmail.com',
            pass: 'klapaucius',
    },
            tls:    { rejectUnauthorized: false }
    });
   var mailOptions = {
        from :  'Tunts <exsm2017@gmail.com>',
        to : info.destinatario,
        subject : info.assunto,
        text: info.mensagem
    }
    transporter.sendMail(mailOptions, callback);
}
```

Fonte: Autoria própria

Na Figura 16 pode ser observada uma função do *controller* para envio de *e-mail*.

Figura 16 – E-mail *Controller*

```
module.exports = {
    enviar : (req,res,next) => {
    let email = new mail.Email(req.body.destinatario, req.body.assunto,
                                    req.body.mensagem);
    mail.enviarEmail(email, function(err,r){
        if(err){
            console.log(err);
            res.json({success:false,msg: 'Falha ao enviar e-mail! '});
        }    else {
            Res.json({success:true,msg: 'E-mail enviado com sucesso!!'});
        });
    }
```

Fonte: Autoria própria

4.3.3 *Sprint* 3

Este *Sprint* teve como foco a implementação de funcionalidades básicas para anunciantes, visto que as mesmas possuem conteúdos diferentes do que usuários comuns. O Quadro 4 demonstra como ficou o *Backlog* do *Sprint*.

Quadro 4 – *Sprint 3 Backlog*

***Sprint 3* Backlog**	
Item	**Tarefas**
Cadastro de anunciante	Implementar o cadastro de pessoas jurídicas
Edição de informação dos anunciantes	Criar a opção de alterar dados cadastrais de pessoas jurídicas
Página de anunciante	Criar *templates* para as páginas dos anunciantes
	Desenvolver regras de negócio para as partes de cliente e servidor

Fonte: Autoria própria

4.3.4 *Sprint* 4

Como *Backlog* do *Sprint* 4, foi decidido pela implementação de funcionalidades adicionais ao sistema, como maneiras dos usuários trocarem mensagens, e utilizar QR code[24] para possibilitar o acesso de qualquer página de algum anunciante desejado.

Quadro 5 – *Sprint 4 Backlog*

***Sprint 4* Backlog**	
Item	**Tarefas**
Troca de mensagens	Adicionar funcionalidade de envio, remoção e respostas de mensagens
Paginação de mensagens	Implementar as páginas das mensagens
QR code	Gerar QR codes para o endereço da página de anunciantes

Fonte: Autoria própria

[24] http://www.qr-code-generator.com/

Foi instalado durante a *Sprint* 4 o componente ‘ng2-bs3’[25], que permite a integração das funcionalidades do *framework* Bootstrap com o Angular. O QR *code* foi adicionado com o intuito de facilitar o compartilhamento das páginas dos anunciantes, pois assim as páginas poderão ser acessadas via código QR gerado.

4.3.5 *Sprint* 5

Durante o desenvolvimento da *Sprint* 5 foi realizada a instalação do *DataTables*[26], um *plug-in*[27] *JavaScript* para paginação de dados tabulados. Foram desenvolvidas também regras CSS para a alteração do estilo de tabelas, possibilitando a exibição de cartões na pesquisa de itens. A *Sprint 5* é demonstrada no Quadro 6:

Quadro 6 – *Sprint 5 Backlog*

***Sprint 5* Backlog**	
Item	**Tarefas**
Controle de produtos	Implementar funcionalidades para criar, editar, listar e excluir serviços e produtos
Paginação de itens para anunciantes	Adicionar a paginação de itens para anunciantes controlarem seus serviços e produtos
Paginação de itens para usuários	Implementar paginação para usuários possuírem controle de seus serviços e produtos requisitados no sistema

Fonte: Autoria própria

4.3.6 *Sprint* 6

O *Sprint* 6 contou com a implementação do gerenciamento de eventos da aplicação. É possível analisar o *backlog* na Quadro 7.

[25] https://www.npmjs.com/package/ng2-bs3-modal
[26] https://datatables.net/manual/styling/bootstrap
[27] http://whatis.techtarget.com/definition/plug-in

Quadro 7 – *Sprint 6 Backlog*

Sprint 6 Backlog	
Item	**Tarefas**
Página de Eventos	Desenvolver página modelo para criação e listagem de eventos
Criação de Eventos	Permitir usuários criarem eventos definido uma data e titulo para o evento
Inserção e exclusão de itens ao evento	Permitir o usuário inserir e remover itens anunciados aos seus eventos

Fonte: Autoria própria

Como nos outros *Sprints*, foi necessário desenvolver as funcionalidades básicas tanto na parte do cliente como na parte de servidor. Ao invés de fazer referência a outros documentos, foi utilizada a flexibilidade de modelos orientados a documentos no banco de dados e realizado a conexão de informações aos documentos de eventos.

4.3.7 *Sprint* 7

O *Sprint* 7 foi responsável pelas funcionalidades de verificação de disponibilidade dos produtos e serviços da aplicação, como mostra o Quadro 8.

Quadro 8 – *Sprint 7 Backlog*

Sprint 7 Backlog	
Item	**Tarefas**
Verificação disponibilidade de item	Implementar funcionalidade para enviar pedido de disponibilidade de item ao fornecedor
Página para pedidos	Desenvolver página modelo para listagem de pedidos de clientes a fornecedores
Confirmação e Rejeição de Item	Desenvolver funcionalidade para que os fornecedores tenham acesso aos itens pedidos e possam confirmar ou rejeitar pedidos

Fonte: Autoria própria

Este *Sprint* foi mais extenso, pois contou com muitas regras de negócio. Todos os itens já listados podem ser vistos por todos os usuários, mas apenas o usuário anunciante e proprietário do item tem permissão para confirmar ou rejeitá-lo caso ele esteja sem

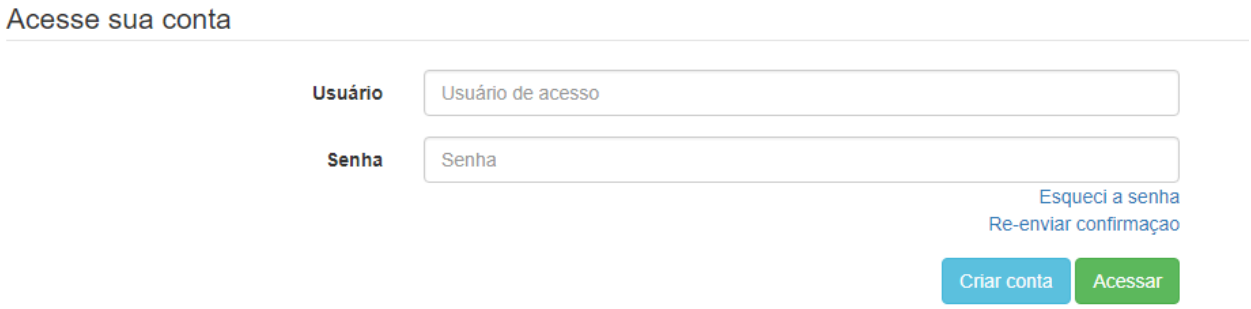
estoque ou não esteja disponível por outro motivo. Caso aceito, é enviado a confirmação para o usuário que requisitou o produto para confirmar o pedido.

4.4 RESULTADO DO PROJETO

O resultado final da aplicação pode ser visto como exemplo nas Figuras 17, 18, 19, 20, 21, 22 e 23:

a) É necessário criar uma conta, tanto para buscar produtos e criar eventos, como para quem quer anunciar seus produtos e serviços. A primeira tela é igual para qualquer tipo de usuário, mas anunciantes devem inserir seu CNPj após já possuir uma conta básica criada. A Figura 17 demonstra a primeira tela de *login* de qualquer usuário:

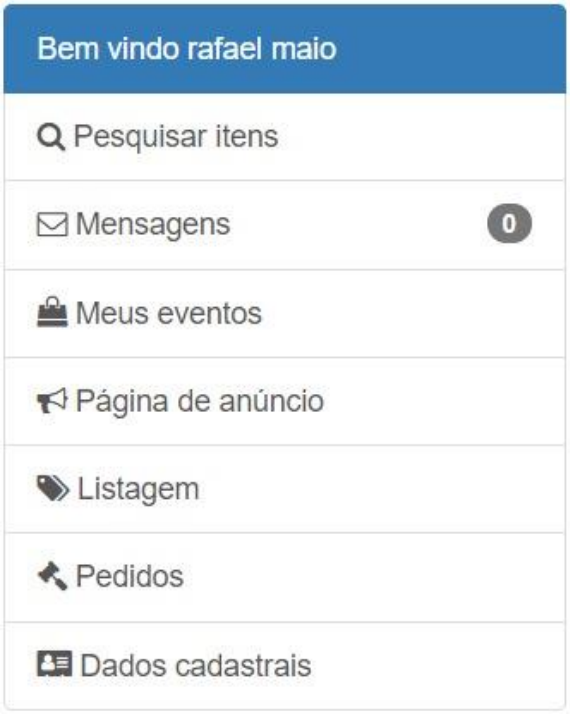
Figura 17 – Exemplo de *login*



Fonte: Autoria própria

b) Após entrar, o usuário tem acesso ao menu principal no canto esquerdo da página, como pode ser visto na Figura 18 (págin:

Figura 18 – *Menu* principal



Fonte: Autoria própria

c) É possível buscar produtos e serviços ao clicar em “Pesquisar itens”, como o exemplo da Figura 19:

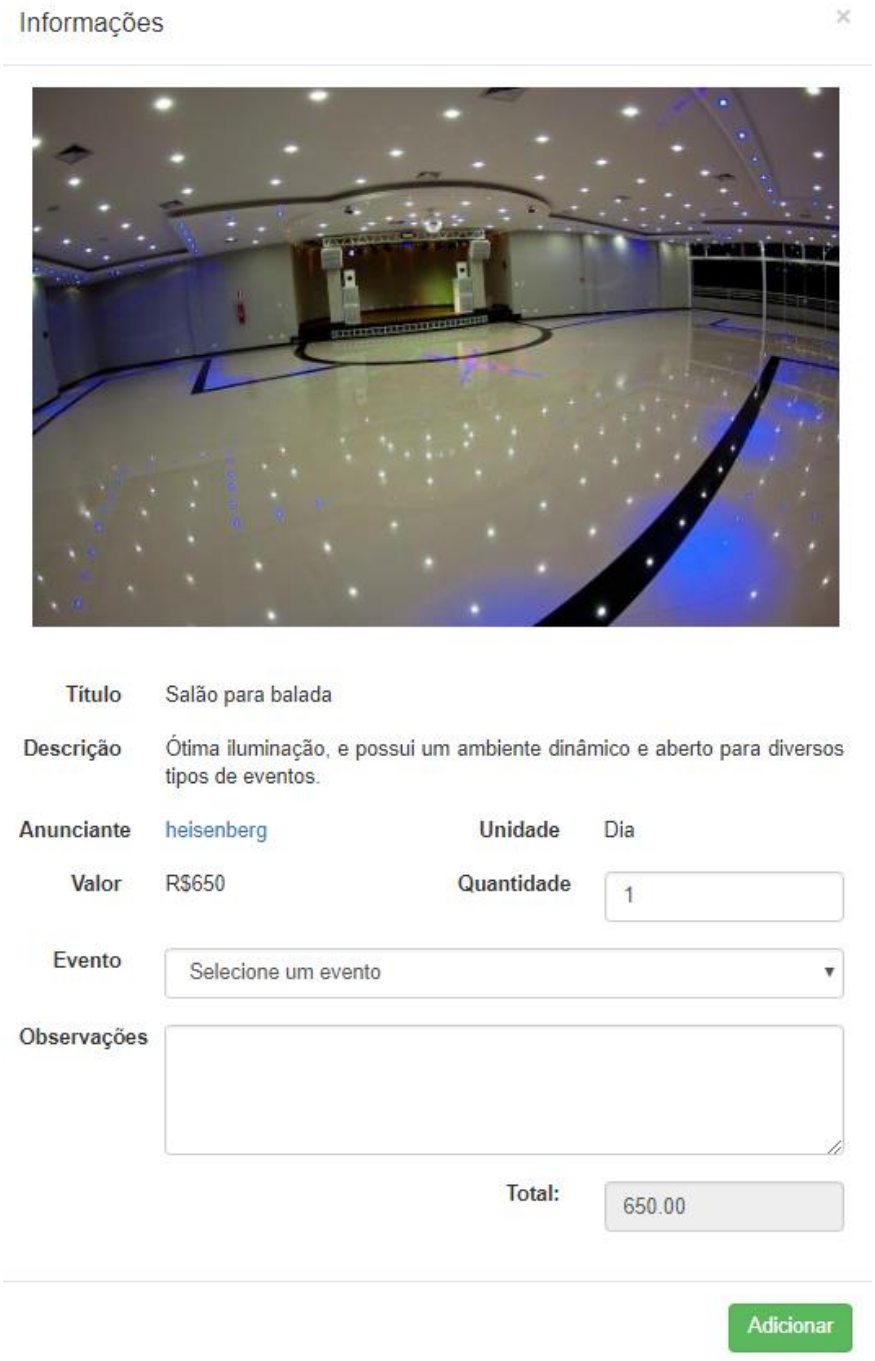
Figura 19 – Exemplo de busca de produtos

Fonte: Autoria própria

d) É possível clicar em algum item e especificar o evento que ele já criou, adicionando o item para determinado evento selecionado, e aguardar a confirmação do

anunciante. Como também é possível enviar uma mensagem com alguma observação, como visto na Figura 20:

Figura 20 – Exemplo de seleção de produto ou serviço

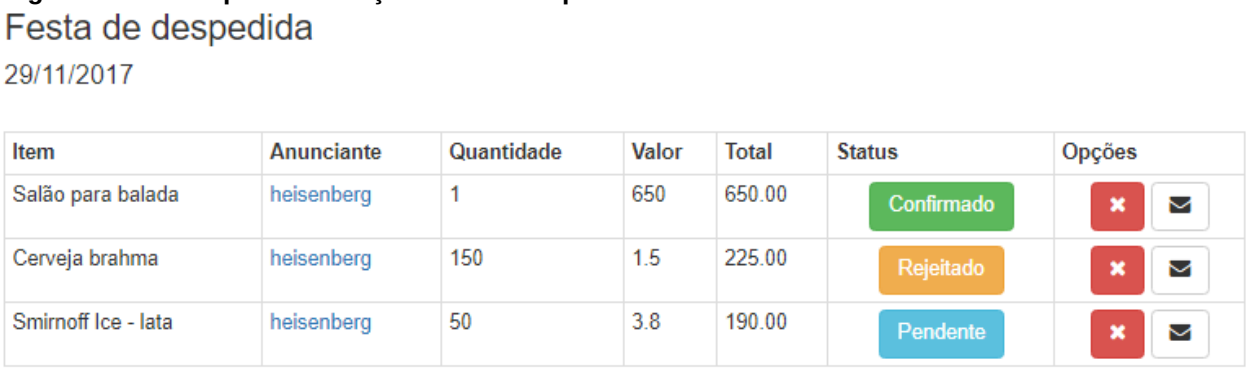


Fonte: Autoria própria

e) Após confirmar o item na página de seus eventos, é necessário que o anunciante confirme a disponibilidade e possibilidade de entrega do serviço ou produto. Caso

aceito, o *Status* é atualizado na página de eventos como “Confirmado”, como mostrado na Figura 21:

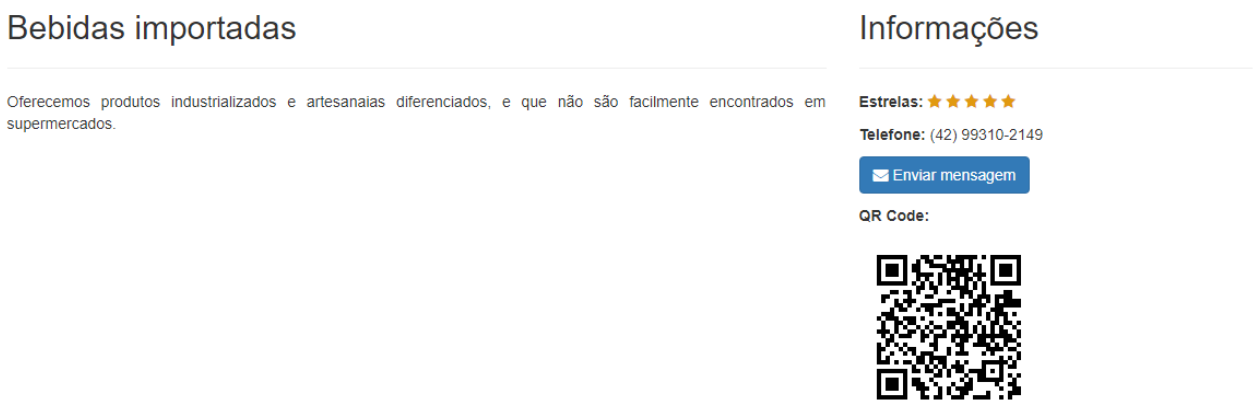
Figura 21 – Exemplo de serviço confirmado para evento



Fonte: Autoria própria

f) Na Figura 22, há um exemplo da página de um anunciante, onde é possível adicionar uma imagem acima do texto para ilustrar sua loja ou serviço:

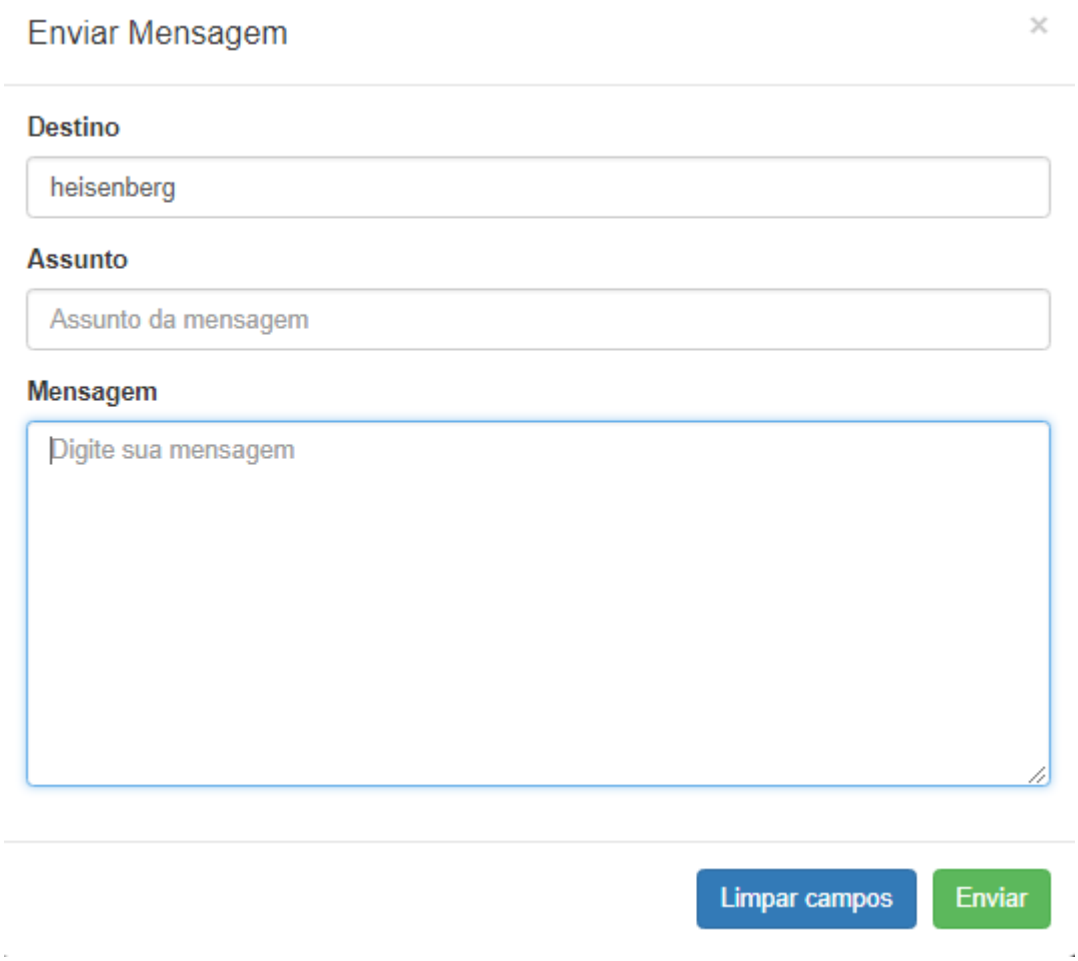
Figura 22 – Exemplo da página de um anunciante



Fonte: Autoria própria

g) A Figura 23 exibe um exemplo de tela para envio de mensagens (página 56):

Figura 23 – Exemplo de envio de mensagens



Fonte: Autoria própria

5 CONCLUSÃO

Neste capítulo são apresentadas algumas observações finais quanto ao projeto, relatando também algumas dificuldades encontradas nas pesquisas e desenvolvimento, e possibilidades de trabalhos futuros.

5.1 CONSIDERAÇÕES FINAIS

O objetivo principal do projeto é facilitar as comunicações que ocorrem entre pessoas que participam na criação de eventos em Ponta Grossa. Uma tarefa árdua para quem não possui experiência ou muitos contatos já existentes, a busca por melhores preços e disponibilidade acabam acarretando em produtos finais que foram agilizados por conta de prazos, sendo que poderiam ser melhores caso houvesse mais facilidade no processo.

As ferramentas escolhidas são livres, e foram escolhidas utilizando critérios como: curva de aprendizagem e desempenho. Isso leva além de uma economia no desenvolvimento do aplicativo, a uma aplicação prática do uso de programas de código aberto.

Foi constatado durante o desenvolvimento da aplicação uma integração uniforme e sem dificuldades na utilização das ferramentas escolhidas, facilitando o uso das mesmas. Além disso o MEAN *stack* possui uma grande comunidade de desenvolvimento, contando com vasto conteúdo em fóruns pela Internet e livros voltados para seu desenvolvimento.

O trabalho não apresentou um teste real realizado entre lojas reais e criadores de eventos pela falta de tempo e recursos para apresentação. Porém, acredita-se que o projeto atendeu os principais requisitos para quem busca facilidade na criação de eventos como foram levantados nas necessidades do projeto, como: conectar criadores de eventos com os fornecedores de serviços, e trazer mais dinamicidade e vantagens para criadores de eventos, como encontrar preços melhores.

5.2 DIFICULDADES ENCONTRADAS

Entre as dificuldades encontradas durante o processo de desenvolvimento do projeto, pode-se citar uma grande quantidade de informações defasadas encontradas ao realizar busca por informações sobre o MEAN *stack,* além de informações atuais sobre o setor de eventos no Brasil.

Outro aspecto que afetou a construção do projeto foi a atualização constante de bibliotecas. Tais atualizações acarretaram na necessidade de reescrita de partes de código por não serem compatíveis com as versões mais recentes, mais especificamente no caso da biblioteca passport-jwt e Mongoose.

5.3 TRABALHOS FUTUROS

A proposta inicial do projeto constituía tanto em realizar a comunicação entre criadores de eventos e anunciantes, como também de fazer a entrega final dos eventos pela própria aplicação. Tal tarefa exigiria transações monetárias, contratos mais complexos, além de um acompanhamento jurídico. Devido a esses aspectos não foram implementadas soluções para este caso de uso. Porém, o projeto foi desenvolvido de uma forma a possibilitar desenvolvimentos futuros para esses casos, ou como forma de implementar características adicionais.

Algumas funcionalidades não foram incluídas pelo prazo necessário para a entrega do Trabalho de Conclusão de Curso, como a avaliação de criadores de eventos para os anunciantes, por exemplo. Tal funcionalidade é importante para a confiabilidade do sistema, pois ela pode ditar quais anunciantes são confiáveis.

Levando-se em conta que aplicação servidor já foi construída, seria também possível o desenvolvimento de um aplicativo móvel que fizesse uso da API já desenvolvida, o que tornaria o trabalho menos complexo e aumentaria a visibilidade da aplicação.

REFERÊNCIAS

ADMINISTRADORES. **O papel da internet:** como ferramenta da *TI*, e seus impactos na economia. 08 abri. 2011. Disponível em: <http://www.administradores.com.br/artigos/economia-e-financas/o-papel-da-internet-como-ferramanta-da-ti-e-seus-impactos-nos-setores-da-economia/54024/>. Acesso em: 17 mai. 2017.

ANGULAR. ***One framework:*** Mobile & Desktop. 20 mar. 2017. Disponível em: <https://angular.io/>. Acesso em: 17 mai. 2017.

ABEOC (ASSOCIAÇÃO BRASILEIRA DE EMPRESAS E EVENTOS) (Brasil) (Org.). **Pesquisa da Associação Brasileira de Eventos Sociais mostra que o mercado de festas e cerimônias atingiu R$ 16,8 bi no ano passado**. 20 mai. 2015. Disponível em: <http://www.abeoc.org.br/2015/05/pesquisa-da-associacao-brasileira-de-eventos-sociais-abrafesta-mostra-que-o-mercado-de-festas-e-cerimonias-atingiu-r-168-bi-no-ano-passado/>. Acesso em: 29 mar. 2017.

BALANCE. ***A Simple Guide to Understanding* URLs*:*** URLs*: Their meaning, Definition and Usage*. 15 mai. 2017. Disponível em: < https://www.thebalance.com/what-does-url-mean-897078>. Acesso em: 13 ago. 2017.

BARLETT, Martin. ***History of* APIs**. 2016. Disponível em: <https://history.apievangelist.com/>. Acesso em: 20 ago. 2017.

BRANAS, Rodrigo. ***AngularJS Essentials:*** *Design and construct reusable, maintainable, and modular web applications with AngularJS*. 1.ed. Mumbai: Packt Publishing, 2014.

BRETZ, Adam; IHRIG J., Colin. ***Full Stack* JavaScript *Development with* MEAN*.*** Disponível em: <http://pepa.holla.cz/wp-content/uploads/2016/11/mean1.pdf>. Acesso em: 11 set. 2017.

BROWN, Ethan. ***Web Development with Node & Express:*** *Leveraging the Javascript Stack*. 1.ed. O'Reilly Media Inc., 2014.

CRTC (*CANADIAN RADIO-TELEVISION AND TELECOMMUNICATIONS COMMISSION*). ***CRTC establishes fund to attain new high-speed Internet targets.*** Disponível em: <http://news.gc.ca/web/article-en.do?nid=1172599>. Acesso em: 06 mai. 2017.

COMMSCOPE. **Relatório da pesquisa sobre a geração Y 2016 da *CommScope*.** Disponível em: <pt.commscope.com/millennials>. Acesso em: 04 mai. 2017.

COUCHBASE. ***Comparing document-oriented and relational data*.** Disponível em: <https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/compare-docs-vs-relational.html>. Acesso em: 04 mai. 2017.

DARWEN, Hugh. ***An Introduction to Relational Database Theory***. 1.ed. Bookboon, 2009

DAYLEY, Brad. ***Node.js, MongoDB and Angular.js:*** *Web Development*. 1.ed. Addison-Wesley Professional, 2014.

DOGLIO, Fernando. ***Pro* REST API *Development with* Node.js.** p. 3-7. 2015.

EVENTBRITE. ***Descubra eventos fantásticos, ou crie seus próprios eventos e venda de ingressos***. Disponível em: < https://www.eventbrite.com.br/>. Acesso em: 03 ago. 2017.

EVENTOS. ***Guia para Festas e Eventos***. Disponível em: <https://www.organizandoeventos.com.br/>. Acesso em: 04 ago. 2017.

EXAME.**Google ajudou a movimentar até 37 R$ bi na economia brasileira**. 12 dez. 2016. Disponível em: <http://exame.abril.com.br/negocios/google-ajudou-a-movimentar-ate-r-37-bi-na-economia-brasileira/>. Acesso em: 17 fev. 2017.

EXPRESS. **Express: Node.js *web application framework***. Disponível em: < https://expressjs.com/>. Acesso em: 20 mar. 2017.

HAHN, Evan. ***Express in Action:*** *Writing, building, and testing Node.js applications*.

1.ed. Manning Publications, 2016.

HOLMES, Simon. ***Getting MEAN with Mongo, Express, Angular and Node.*** Disponível em: <http://salesmanagement.org/web/uploads/pdf/af4610022ea794e97eee259432519a92.pdf>. Acesso em: 07 mai. 2017.

INTERNET SOCIETY. ***Brief History of the Internet***. Disponível em: <https://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet#LGR67>. Acesso em: 03 mai. 2017.

LAYTON, Mark. **Scrum *for dummies***. p. 32-48. 2015.

MCKINSEY&COMPANY(Org.). ***The great transformer: The impact of the Internet on economic growth and prosperity***. out. 2011. Disponível em: <http://www.mckinsey.com/industries/high-tech/our-insights/the-great-transformer>. Acesso em: 04 mai. 2017.

METADATA STANDARDS(Org.). *Information Technology Standards.* ***A Comparison of SQL and NoSQL Databases***. 13 mai. 2011 Disponível em: <http://metadata-standards.org/Document-library/Documents-by-number/WG2-N1501-N1550/WG2_N1537_SQL_Standard_and_NoSQL_Databases%202011-05.pdf>. Acesso em: 04 mai. 2017.

MINDMASTER. 2014. Disponível em: <http://www.mindmaster.com.br/scrum/>. Acesso em: 06 ago. 2017.

MONGODB, INC. ***mongoDB***. Disponível em: <https://www.mongodb.com>. Acesso em: 20 mar. 2017.

MONGOOSE. ***Mongoose: elegant mongodb object modeling for node.js***.. Disponível em: <http://mongoosejs.com/>. Acesso em: 20 mar. 2017.

NAYAK, Ameya; PORIYA, Anil; POOJARY, Dikshay. *Type of NOSQL Databases and its Comparison with Relational Databases*. **International Journal of Applied Information Systems**, v. 5, n. 4, 2013.

NODE.JS FOUNDATION. ***NODE.JS Evented I/O for V8 JavaScript***. Disponível em: <http://nodejs.org/>. Acesso em: 20 mar. 2017.

NODESOURCE. ***Node.JS usage and adoption statistics 2017***. 10 mar. 2017. Disponível em: <https://nodesource.com/node-by-numbers>. Acesso em: 20 mai. 2017.

NTNU (Norwegian University of Science and Technology). Advanced Process Simulation. ***SQL vs NoSQL***. 08 dez. 2014. Disponível em: <http://folk.ntnu.no/preisig/HAP_Specials/AdvancedSimulation_files/2014/AdvSim-2014__Birgen_Cansu_Databases.pdf>. Acesso em: 07 mai. 2017.

PEARSON. ***What is a Web Server?*****.** 2 dez. 1999. Disponível em: <http://ptgmedia.pearsoncmg.com/images/0130225347/samplechapter/0130225347.pdf>. Acesso em: 20 mar. 2017.

PINGDOM.***A history of the dynamic web*****.** 7 dez. 2007. Disponível em: <http://royal.pingdom.com/2007/12/07/a-history-of-the-dynamic-web/>. Acesso em: 20 fev. 2017.

PWC (PRICEWATERHOUSECOOPERS). ***Brazil - leading the digital media revolution in Latin America.*** *2014.* Disponível em: <http://www.pwc.com/gx/en/global-entertainment-media-outlook/assets/brazil-summary.pdf>. Acesso em: 06 mai. 2017.

PUC (Pontifícia Universidade Católica do Paraná). ***e-Economics:*** O Impacto da Internet na Economia***.*** 2000. Disponível em: <https://www.assija.com.br/downloads/economics.pdf>. Acesso em: 03 mai. 2017.

RIEHLE. ***Framework Design:*** *A Role Modeling Approach*. 2000. Disponível em: <http://dirkriehle.com/computer-science/research/dissertation/diss-a4.pdf>. Acesso em: 20 mar. 2017.

SCHROEDER, R.; SANTOS, F. **Arquitetura e Testes de Serviços Web de alto desempenho com *NODE.JS* e *MongoDB***. p. 1-17. 2014.

SEBRAE (SERVIÇO BRASILEIRO DE APOIO ÀS MICRO E PEQUENAS EMPRESAS) (Brasil) (Org.). **Empresas utilizam a internet para fazer negócios e aumentar**

vendas. 04 mar. 2016. Disponível em: <http://www.al.agenciasebrae.com.br/sites/asn/uf/AL/empresas-utilizam-a-internet-para-fazer-negocios-e-aumentar-vendas>. Acesso em: 20 mar. 2017.

SEBRAE (SERVIÇO BRASILEIRO DE APOIO ÀS MICRO E PEQUENAS EMPRESAS) (Brasil) (Org.). **Estudo mostra a expansão da área de eventos no Brasil.** 22 jan. 2016. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/bis/estudo-mostra-a-expansao-da-area-de-eventos-no-brasil>. Acesso em: 20 mar. 2017.

STRAUCH, Christof. ***Lecture on NoSQL Databases*.** 10 jun. 2009. Disponível em: <http://www.christof-strauch.de/nosqldbs.pdf>. Acesso em: 07 mai. 2017.

SUTHERLAND, Jeff. **SCRUM:** *The art of doing twice the work in half the time*. p. 3-11. 2014.

TUTORIALS. ***Simple Easy Learning.*** 2017. Disponível em: < https://www.tutorialspoint.com/http/index.htm>. Acesso em: 11 ago. 2017.

ULB (Université Libre de Bruxelles). ***Document Oriented Databases***. 12 out. 2015. Disponível em: <http://cs.ulb.ac.be/public/_media/teaching/infoh415/student_projects/couchdb.pdf>. Acesso em: 07 mai. 2017.

USN (University College of Southeast Norway). ***Structured Query Language.*** 08 jan. 2016. Disponível em: <http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf>. Acesso em: 03 mai. 2017.

WEBCODE. ***Bootstrap Programming Cookbook.*** 21 jan. 2016. Disponível em: <https://www.webcodegeeks.com/wp-content/uploads/2015/12/Bootstrap-Programming-Cookbook.pdf>. Acesso em: 03 mai. 2017.

W3C (WORLD WIDE WEB CONSORTIUM). ***Information Management: A Proposal.*** mar. 1989. Disponível em: <https://www.w3.org/History/1989/proposal.html>. Acesso em: 03 mai. 2017.

W3SCHOOLS. 2017. Disponível em: <https://www.w3schools.com/tags/ref_httpmethods.asp>. Acesso em: 13 ago. 2017.

ZEPHORIA. ***The Top 20 Value Facebook Statistics.*** 22 mar. 2017. Disponível em: <https://zephoria.com/top-15-valuable-facebook-statistics/>. Acesso em: 03 mai. 2017.

APÊNDICE A - Entrevista realizada com criador de eventos

Entrevista com criador de eventos

1- **Você realizou algum evento? Se sim, quais?**

R: Sim. Um show nacional (Armandinho), festas open bar, como: Batucada, Cervejada da Atlética XV de Outubro, JOIA, Miss Joia, Pré Jogos, Santa Backy, entre outros!

2- **Você utilizaria um *website* para auxiliar a busca de serviços e produtos para eventos?**

R: Sim.

3- **Caso sim, pode listar alguns itens que seria importante no *website?***

R: Possibilidade de gerenciar o evento no *website,* encontrar produtos (como bebidas) com seus devidos preços e disponibilidade, encontrar serviços (como bandas e limpeza) com seus devidos preços e disponibilidade, encontrar locais disponíveis na cidade para locação, entrar em contato com os fornecedores de produtos e serviços pelo *website*, e poder ver avaliações e avaliar fornecedores e seus serviços.