UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GIANCARLO RODRIGUES

APLICATIVO MÓVEL PARA GESTÃO DE PEDIDOS E CONSULTA DE CARDÁPIOS DE ESTABELECIMENTOS GASTRONÔMICOS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2015

GIANCARLO RODRIGUES

APLICATIVO MÓVEL PARA GESTÃO DE PEDIDOS E CONSULTA DE CARDÁPIOS DE ESTABELECIMENTOS GASTRONÔMICOS

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 / Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Richard Duarte Ribeiro

PONTA GROSSA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Departamento Acadêmico de Informática
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

APLICATIVO MÓVEL PARA GESTÃO DE PEDIDOS E CONSULTA DE CARDÁPIOS DE ESTABELECIMENTOS GASTRONÔMICOS

por

GIANCARLO RODRIGUES

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 20 de Maio de 2015 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Richard Duarte Ribeiro
Prof. Orientador

_____
Luiz Rafael Schmitke
Membro titular

_____
Simone de Almeida
Membro titular

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

AGRADECIMENTOS

Neste passo tão importante da formação acadêmica e para minha vida, gostaria de agradecer primeiramente a Deus por todas as oportunidades e bênçãos obtidas até o momento.

Agradeço ao meu orientador, prof. Dr. Richard Duarte Ribeiro, por todo o apoio prestado, toda a dedicação e confiança empregadas neste trabalho, além da ilustre orientação que colaborou para atingir os objetivos almejados.

Agradeço também a toda minha família e amigos, em especial aos meus pais e minha namorada, que sempre estiveram ao meu lado preocupando-se e colaborando comigo para que atingisse meus objetivos, incentivando quando necessário e sempre ajudando a fazer as escolhas certas.

Por fim, mas não menos importante, agradeço a todas as pessoas que fizeram parte da minha vida nesta longa caminhada realizada na UTFPR, em especial aos professores que sempre nos ajudaram a obter o conhecimento necessário e a estar preparados para os desafios do mercado de trabalho, aos colegas de classe que compartilhei desafios e sempre me ajudaram quando precisava, especialmente na hora da carona para casa, e a todas as empresas que abriram suas portas para a realização do meu estágio, a fim de se obter a prática e o conhecimento ímpar que não são auferidos em sala de aula.

A todos os envolvidos na realização deste trabalho e na minha vida acadêmica, os meus sinceros agradecimentos.

Sabe, se pensar na própria natureza da vida, quero dizer, desde o início, o desenvolvimento da primeira célula que se dividiu em duas células, o único propósito da vida tem sido passar adiante o que foi aprendido. Nada mais importante que isso. Se me pergunta o que fazer com o conhecimento acumulado, digo: passe adiante, como faz qualquer outra célula com o passar do tempo.

(Lucy – Universal Pictures, 2014)

RESUMO

RODRIGUES, Giancarlo. **Aplicativo móvel para gestão de pedidos e consulta de cardápios de estabelecimentos gastronômicos**. 2015. 93 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2015.

Este trabalho fornece uma discussão sobre alguns temas relacionados ao desenvolvimento de aplicativos móveis e apresenta uma solução composta por aplicativos móveis que permite inovar alguns setores dos estabelecimentos gastronômicos da cidade de Ponta Grossa, Paraná. Para o desenvolvimento do servidor da solução foi utilizada a plataforma NodeJS com os *frameworks* Express e Mongoose, além do banco de dados MongoDB. Para a construção do aplicativo móvel utilizou-se o Ionic *Framework* juntamente com o AngularJS, ngCordova e Apache Cordova. Todas as tecnologias utilizadas são *open-source* e o resultado obtido comprova o potencial destas ferramentas quando utilizadas em conjunto.

Palavras-chave: Aplicativos Móveis. Desenvolvimento Híbrido. MEAN. Ionic Framework. Cardápio.

ABSTRACT

RODRIGUES, Giancarlo. **Mobile application for order management and menus query of gastronomic establishments**. 2015. 93 p. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Federal Technology University - Parana. Ponta Grossa, 2015.

This work provides a discussion of some issues related to the development of mobile applications and gives a solution composed of mobile applications to refresh some sectors of the gastronomic establishments in the city of Ponta Grossa, Paraná. To develop the solution's server was used NodeJS platform with frameworks Express and Mongoose, in addition to the MongoDB database. The mobile application construction used the Ionic Framework with the AngularJS, ngCordova and Apache Cordova. All the technologies used are open-source and the results show the potential of these tools when used together.

Keywords: Mobile Applications. Hybrid Development. MEAN. Ionic Framework. Menu.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ACRÔNIMOS

AJAX	*Asynchronous Javascript and* XML
API	*Application Programming Interface*
BRASSCOM	Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação
BSON	*Binary Javascript Object Notation*
DOM	*Document Object Model*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
IDC	*International Data Corporation*
I/O	*Input/Output*
JSON	*Javascript Object Notation*
MEAN	MongoDB, Express, AngularJS, Node.js
MVC	*Model-View-Controller*
NoSQL	*Not Only SQL*
npm	*Node Package Manager*
QR Code	*Quick Response Code*
SDK	*Software Development Kit*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
UI	*User Interface*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Por anos seguidos é possível notar um aumento significativo nas vendas de *smartphones* em vários países do mundo, especialmente no Brasil (BRASSCOM, 2014a). Segundo Corrêa et al. (2014), este dispositivo atrai a atenção dos consumidores porque permite, além das funcionalidades básicas de um telefone celular (receber e efetuar chamadas telefônicas e mensagens de texto), realizar operações que anteriormente eram efetuadas somente em computadores ou *notebooks* (manipular arquivos, acessar a Internet, correio eletrônico, etc.). Isto é possível porque este dispositivo "possui um Sistema Operacional multitarefa, um navegador (tal como o de um *desktop*), rede sem fio e suporte de conexão com a Internet 3G/4G" (CORRÊA et al., 2014, p. 1-2).

Além dos inúmeros recursos disponibilizados, os *smartphones* possuem um preço de compra inferior ao dos computadores e são mais úteis que estes por possuírem recursos de geolocalização e serem mais portáteis (ALLEN; GRAUPERA; LUNDRIGAN, 2010). Outro fator que estimulou a disseminação dos *smartphones* entre os consumidores, segundo Corrêa et al. (2014, p. 3), foi seu baixo valor de aquisição, que possibilitou às pessoas com menor poder de compra o acesso a este dispositivo. O resultado da expansão destes aparelhos é que "hoje em dia os *smartphones* são mais uma necessidade do que um luxo" (PALMIERI; SINGH; CICCHETTI, 2012, p. 179, tradução nossa).

Uma pesquisa realizada pela Google (2013) com mil brasileiros proprietários de *smartphones* relatou que as pessoas estão cada dia mais dependentes de seus aparelhos e que 46% dos entrevistados acessam a Internet todos os dias a partir deles. Um dos resultados desta pesquisa aponta que o comportamento do consumidor está mudando e conclui que as empresas podem se beneficiar desta modificação caso utilizem na organização algum dos recursos oferecidos por estes dispositivos (GOOGLE, 2013).

Borges (2012, p. 23) afirma que um dos setores que recebeu vários benefícios com o avanço tecnológico é o de prestação de serviços relacionados à alimentação e cita algumas das soluções já existentes para este setor:

> Diversos programas e sistemas de gestão e controle de estoque, pedidos, custos, percentual de vendas dos itens do cardápio, recursos humanos, produtividade, etc. Nesse sentido, os donos de restaurantes têm à sua

> disposição uma variedade de soluções de tecnologia da informação. (BORGES, 2012, p. 23-24)

Além desses benefícios, Rigodanzo e Paz (2006, p. 3) destacam a existência de estabelecimentos onde o próprio cliente já realiza seus pedidos por meio de um computador ou então o próprio garçom realiza o pedido utilizando um computador de bolso.

Segundo Borges (2012), todos os avanços em tecnologia da informação colaboraram para a modificação do setor de prestação de serviços, que agora tem como objetivo principal a satisfação dos clientes. Investir em tecnologia possibilita, segundo Rigodanzo e Paz (2006, p. 2), “qualidade e agilidade no processamento dos serviços e consequentemente em um melhor atendimento aos clientes”.

Porter (1989 apud BINDES, 2012, p. 15) defende que “Uma empresa diferencia-se da concorrência, quando oferece alguma coisa singular valiosa para os compradores além de simplesmente oferecer um preço baixo”. Para Bindes (2012, p. 11) “o bom desempenho das instituições passou a depender de sua competência em inovar nas áreas de produtos, serviços e processos”, portanto investir em tecnologia dentro de uma organização pode significar melhoria dos processos da empresa, destaque para a organização e, principalmente, melhoria no atendimento aos clientes.

1.1. MOTIVAÇÃO E ESCOPO

Desfrutando da oportunidade oferecida pelos *smartphones* em atingir um grande número de pessoas e, também, objetivando maximizar ainda mais a vantagem competitiva e a inovação entre as empresas da área de alimentação, este trabalho propõe o desenvolvimento de um aplicativo para *smartphones* que possibilita ao usuário consultar o cardápio de estabelecimentos gastronômicos selecionados da cidade de Ponta Grossa, Paraná. Além disso, o aplicativo possibilita a verificação do valor total da comanda[1] e também a divisão (se necessária) do seu valor total antes de efetuar o pagamento ao estabelecimento.

[1] Comanda é o nome dado às anotações de pedidos feitas pelos garçons nos estabelecimentos comerciais.

Optou-se pelo desenvolvimento de um aplicativo com essas funcionalidades por permitir a busca por informações de locais especializados em refeições e de seus produtos de maneira fácil e prática, tornando mais rápido o acesso à informação para o usuário, e também pelo que é observado por Rigodanzo e Paz (2006, p. 2): “Hoje, é difícil encontrar um restaurante de médio e grande porte que não tenha seu sistema informatizado, principalmente no que tange os setores de compras, estoques, vendas, e folhas de pagamento”.

Assim, o resultado deste trabalho pode oferecer novos atrativos às empresas do setor de alimentação e funcionar em conjunto com os sistemas já existentes nos estabelecimentos, trazendo mais um diferencial para as empresas do ramo e seguindo o que foi proposto por Porter (1989 apud BINDES, 2012, p. 15).

1.2. OBJETIVOS

Para o desenvolvimento deste trabalho teve-se como objetivo geral o desenvolvimento de um aplicativo que integrasse as funcionalidades de comparação de preços, consulta de cardápios, consulta do valor gasto nos estabelecimentos e possibilidade de divisão deste valor antes de efetuar o pagamento nos estabelecimentos gastronômicos de Ponta Grossa, Paraná.

Em conjunto com o objetivo geral do trabalho foram buscados alguns objetivos específicos, citados abaixo:

- Desenvolvimento de uma aplicação de baixo custo e de fácil integração com outros programas de gestão dos estabelecimentos;
- Desenvolvimento do projeto utilizando a abordagem híbrida de desenvolvimento de aplicativos móveis, banco de dados NoSQL (MONGODB, 2015), tecnologia de identificação QR Code (DE LAET, 2010) e um servidor orientado a eventos (NODEJS, 2015a);
- Desenvolvimento de uma comanda eletrônica para permitir o funcionamento completo das funcionalidades do aplicativo;
- Utilização de tecnologias que permitam a expansão da estrutura do aplicativo sem comprometer seu funcionamento;

- Construção de uma solução que integre todas as funcionalidades desejadas em apenas um aplicativo.

1.3. ESTRUTURA DO TRABALHO

Este trabalho apresentou em seu primeiro capítulo uma introdução ao assunto abordado, o que motivou o desenvolvimento do mesmo e os objetivos esperados com o desenvolvimento.

O segundo capítulo é composto pela revisão literária do conteúdo deste trabalho, a qual aborda tópicos referentes à *smartphones*, desenvolvimento de aplicativos móveis, *frameworks* de desenvolvimento, construção de servidores web e bancos de dados.

O terceiro capítulo descreve a metodologia utilizada para elaboração do projeto e as limitações aplicadas ao mesmo, bem como a análise de sistemas similares.

O quarto capítulo apresenta o desenvolvimento do projeto e as tarefas realizadas para atingir os objetivos propostos, bem como discute sobre as dificuldades do mesmo.

Por fim, o quinto capítulo apresenta as considerações finais deste trabalho e sugere alguns trabalhos futuros que utilizam uma abordagem similar a este trabalho.

2 REFERENCIAL TEÓRICO

Nesta seção serão discutidos os seguintes assuntos: o uso da tecnologia de informação em restaurantes; recursos e oportunidades fornecidas pelos *smartphones*; desenvolvimento de aplicativos para *smartphones*; desenvolvimento de servidores *web*; os modelos de bancos de dados existentes e o banco de dados NoSQL MongoDB, além de outros assuntos referentes ao projeto.

2.1. A UTILIZAÇÃO DA TI NO SETOR DE ALIMENTAÇÃO

A Tecnologia da Informação (TI) é definida como a “infraestrutura organizada de *hardware*, *software*, banco de dados e redes de telecomunicações, que permite manipular, gerar e distribuir dados e informações ao longo dos seus usuários (empresas ou pessoas)” (MIGLIOLI, 2007 apud LEITE, 2015). Essa capacidade de obter dados e informações é resultado do processamento do conteúdo armazenado em um sistema de informação. Segundo Menezes (2007, p. 11), “os sistemas de informação contêm informação sobre pessoas, lugares e coisas de interesse, nos ambientes interno e externo de uma organização”.

Segundo Bindes (2012, p. 7-8) a TI é utilizada nas empresas por questões estratégicas, já que “a competitividade requer maior domínio das situações, aptidão para se ajustar às novas condições do mercado e maior competência para saber empregar novas tecnologias”. Todas essas situações são facilitadas com o uso de sistemas de informação, visto que estes auxiliam “os funcionários ou gerentes na tomada de decisões, na análise e visualização de soluções no ambiente organizacional” (MENEZES, 2007, p. 11). A rápida tomada de decisão possibilita “às organizações se transformarem rapidamente e levar essas mudanças ao mercado” (BINDES, 2012, p. 11). Fincotto (2014) apresenta uma justificativa para o uso da TI nas organizações e também alguns benefícios da sua utilização:

> O aumento da concorrência e do consumo faz com que as empresas busquem novas soluções para melhorar e agilizar seu atendimento, exposição de seus produtos e serviços visando aumentar sua competitividade e destaque no mercado. Tendo em vista todas essas mudanças e a constante evolução dos computadores e celulares, as empresas fazem uso cada vez mais intensivo da tecnologia da informação

> (TI) como principal ferramenta de apoio estratégico em seus negócios (FINCOTTO, 2014, p. 1).

Atualmente o mercado de *software* para o setor de alimentação já conta com "sistemas de gestão e controle de estoque, pedidos, custos, percentual de vendas dos itens do cardápio, recursos humanos, produtividade" (BORGES, 2012, p. 23-24). Além disso, é importante observar o que Rigodanzo e Paz apresentam sobre o atual estado da utilização da tecnologia nos estabelecimentos:

> Hoje é comum existirem estabelecimentos onde o cliente pode escolher a sua refeição e sua bebida pelo computador ou pedir seu pedido direto ao atendente, que usando um computador de bolso o faz. (RIGODANZO; PAZ, 2006, p. 3)

A utilização dos sistemas informatizados, segundo Rigodanzo e Paz (2006, p. 3), "tem por objetivo maior proporcionar um melhor atendimento ao cliente, pelos resultados imediatamente oferecidos". Borges (2012) acredita que a tecnologia da informação alterou a prestação de serviços, já que agora o foco é a satisfação dos clientes. Segundo Menezes (2007, p. 12), "Os computadores vêm substituindo gradativamente a tecnologia manual de processamento", pois com o avanço da modernidade é comum o investimento em tecnologia objetivando melhorar os serviços oferecidos (BINDES, 2012).

Em um estudo apresentado pelo SEBRAE (Serviço Brasileiro de Apoio às Micro e Pequenas Empresas) em Abril de 2014, foram geradas algumas oportunidades para o mercado de *software* graças à realização da Copa do Mundo em 2014 e das Olimpíadas em 2016 no Brasil. Segundo o SEBRAE (2015), bares e restaurantes investirão em *software* para atrair o público consumidor, já que estes "se tornaram um grande diferencial para conquistar um público cada vez mais informatizado e conectado" (SEBRAE, 2015, p. 1). Além disso, a pesquisa também apresenta alguns benefícios obtidos com sua utilização, visto que os programas divertem e promovem a interação entre as pessoas, facilitam a escolha dos produtos e podem premiar e dar informações aos clientes, causando uma boa impressão (SEBRAE, 2015).

Segundo Bindes (2012, p. 15), os estabelecimentos "que atingem a capacidade de disponibilizar um produto, serviço, decoração ou qualquer condição de qualidade superior" serão considerados diferenciados e por isso serão colocados em evidência. Para Porter (1989), o destaque no setor é:

> Uma consequência do emprego da TI pelas organizações, as quais acreditam, cada vez mais, que o sucesso em seus empreendimentos se deve, em grande parte, à tecnologia existente nos dias atuais. (PORTER, 1989 apud BINDES, 2012, p. 16-17)

Rigodanzo e Paz (2006, p. 4-5) concluem em seu estudo que a inovação proveniente da informatização é fundamental para os restaurantes, pois promove "um grande avanço na qualidade da gestão, agilizando e estabelecendo maiores condições de atendimento aos clientes". Além dos benefícios para a organização e para os clientes, Borges apresentou em seu estudo que a utilização de tecnologia por parte dos funcionários dos estabelecimentos acarretou em "interesse por cursos de informática e a aquisição de computadores pessoais" (BORGES, 2012, p. 35).

A partir das definições dos autores conclui-se que este setor beneficia-se muito com o investimento em tecnologia. As partes envolvidas – organização e clientes – já são beneficiadas com o conhecimento atual, porém o surgimento de novas tecnologias significa maior destaque e um novo diferencial para a organização. Novas soluções de *software* podem melhorar ainda mais o atendimento e permitir uma maior fidelização dos clientes do estabelecimento, portanto este setor ainda possui grandes oportunidades de inovação.

2.2. *SMARTPHONES* E OPORTUNIDADES

Smartphone é a definição de um dispositivo móvel superior aos telefones celulares comuns – com tamanho de tela maior e de alta resolução, além de outras capacidades – e que possui um teclado QWERTY[2] (ou outros modelos de teclado) (ALLEN et al., 2010, p. 4). Segundo Corrêa et al. (2014) a sua superioridade está ligada às funcionalidades oferecidas, pois segundo os autores:

> Ao invés de apenas armazenar informações de telefones, efetuar e receber ligações, receber e enviar mensagens de texto, os smartphones realizam tarefas mais avançadas: receber e enviar emails [sic], realizar pesquisas na Internet e executar tarefas normalmente associadas a computadores pessoais ou notebooks. (CORRÊA et al., 2014, p. 1-2)

[2] Segundo Couto, QWERTY é um modelo de teclado onde a disposição das teclas é semelhante ao das antigas máquinas de escrever e dos computadores. É um padrão mundial utilizado em *smartphones*, computadores e em extensões de teclados (COUTO, 2012).

Corrêa et al. (2014) afirmam que um *smartphone* possui sistema operacional, um navegador de Internet e suporte à Internet móvel e sem fio (*wireless*). Allen et al. (2010) destacam a presença de recursos de geolocalização, câmera e conectividade embutidos nos dispositivos, tornando-os mais poderosos que os computadores pessoais. Para Silva (2014) outro benefício dos *smartphones* é a sua portabilidade – "pequenos computadores que cabem no seu bolso" (ALLEN et al., 2010, p. 3, tradução nossa) – pois segundo o autor:

> Esses dispositivos são usados normalmente para acessar informações que são processadas por outros dispositivos, como computadores, dispensando a necessidade do usuário de se deslocar até um local especifico para realizar essa tarefa, tornando a atividade mais acessível e prática. (SILVA, 2014, p. 44)

Outra característica desses dispositivos é a capacidade de executar um sistema operacional com aplicações pré-instaladas e permitir ao usuário instalar aplicações de terceiros (XANTHOPOULOS; XINOGALOS, 2013). Essas aplicações – ou aplicativos, *apps* – são pequenos programas (DE MENDONÇA et al., 2011) obtidos de uma loja virtual disponibilizada pelo desenvolvedor do sistema operacional, capazes de acessar os recursos do dispositivo – câmera, GPS – e desempenhar alguma tarefa específica, tornando o dispositivo móvel desejável e popular (RIBEIRO; DA SILVA, 2012).

Segundo Allen et al. (2010, p. 1, tradução nossa), muitas tarefas realizadas nos computadores ou *notebooks* já podem ser efetuadas em *smartphones*, por isso os autores defendem que "o celular é o novo computador pessoal (...) o computador *desktop* não está indo embora, mas o mercado de *smartphones* está crescendo rapidamente". Os autores também afirmam que "já existem mais telefones celulares do que computadores conectados à internet" (ALLEN et al., 2010, p. 1, tradução nossa).

Segundo dados da BRASSCOM (Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação) no ano de 2013 havia 267 milhões de dispositivos móveis no Brasil (BRASSCOM, 2014a). Uma predição feita pela IDC (2013 apud BRASSCOM, 2014b) aponta que no ano de 2016 o Brasil será o quarto maior mercado mundial de dispositivos móveis e que "no Brasil e nos EUA, o tempo gasto em dispositivos móveis superou o consumo de televisão" (BRASSCOM, 2014b, p. 4). Uma pesquisa realizada pela Google (2013, p. 2) concluiu que "a

difusão dos smartphones atinge 26% da população brasileira". Além dessa informação, a pesquisa obteve outras informações importantes sobre o comportamento dos usuários (baseada nas respostas de mil brasileiros entrevistados), a qual aponta que:

- 89% dos usuários de *smartphones* procuram informações locais em seus telefones e 90% tomam decisões em decorrência disso, como fazer uma compra ou entrar em contato com a empresa;
- 82% dos consumidores pesquisaram um produto ou serviço no dispositivo antes da compra;
- Os anúncios para celular são vistos por 96% dos usuários de *smartphones*;
- Os usuários de smartphones estão usando suas mídias para a realização de várias tarefas, e 90% usam o telefone durante outras atividades, como assistir TV (41%) (GOOGLE, 2013, p. 2).

Junto com as informações apresentadas também foram sugeridas algumas opções para auxiliar as empresas a tirar proveito destas informações. Utilizar o sistema de geolocalização para oferecer serviços exclusivos, exibir um número de telefone que pode ser clicado, possuir um site de vendas preparado para dispositivos móveis e fazer publicidade unindo os dispositivos e outras mídias são exemplos que, segundo a Google, podem beneficiar as empresas locais a se aproximar dos clientes, aumentar a visibilidade do negócio, influenciar na decisão do comprador e melhorar a estratégia de *marketing* adotada pela empresa (GOOGLE, 2013).

As informações obtidas pela pesquisa da Google (GOOGLE, 2013) apresentam informações valiosas para empresas de diversos setores. Segundo Corrêa et al., a "expansão tecnológica na área da Computação Móvel, em especial, da telefonia móvel, aumentou a demanda por aplicações para os mais variados setores da sociedade" (CORRÊA et al., 2014, p. 2). Segundo Fincotto:

> Com a utilização dos aplicativos móveis aliados à automação comercial e as tecnologias existentes, as empresas podem se tornar mais competitivas diante de um mercado cada vez mais globalizado e concorrido. (FINCOTTO, 2014, p. 11).

Diante das oportunidades apresentadas, o desenvolvimento de um aplicativo móvel para exibir o cardápio dos estabelecimentos e as informações do total gasto no local mostra-se interessante para as empresas do ramo alimentício, visto que se

aproveita a visibilidade que os aplicativos fornecem e também o potencial de alcance de vários consumidores por meio dos dispositivos móveis, além de criar novos atrativos para os estabelecimentos que fizerem o uso desta tecnologia.

2.3. DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

Fincotto (2014, p. 2) afirma que “ao iniciar um projeto de software para dispositivos móveis, é necessário realizar uma análise criteriosa e estratégica sobre a plataforma, sistemas, produtos e arquiteturas a serem utilizadas”. Isto é necessário porque existem vários sistemas operacionais para *smartphones* e estes determinam a linguagem de programação necessária para construir os aplicativos da plataforma (ALLEN et al., 2010), resultando no que Amatya e Kurti (2014) definem como “fragmentação do sistema operacional”. O quadro 1 apresenta uma lista com alguns sistemas operacionais móveis e suas respectivas linguagens de programação para o desenvolvimento de aplicativos móveis.

Sistema Operacional Móvel	Conjunto de Habilidades Requeridas
Apple iOS	C, Objective C
Google Android	Java, C++, C
RIM Blackberry	Java
Symbian	C, C++, Python
Windows Mobile	.NET
Windows 7 Phone	.NET
HP Palm webOS	HTML/CSS/JS
MeeGo	C, C++
Samsung bada	C++

Quadro 1 – Sistemas operacionais móveis e suas linguagens de programação
Fonte: Adaptado de (CHARLAND; LEROUX, 2011, p. 51)

As diferenças entre linguagens de programação acarretam em SDKs (*Software Development Kits*) e paradigmas de desenvolvimento distintos entre as plataformas (ALLEN et al., 2010). Segundo Ribeiro e da Silva (2012), também existem ambientes de desenvolvimento e lojas de aplicativos específicos conforme o sistema operacional móvel. Essas diferenças são compreendidas pela seguinte afirmação: “enquanto as capacidades do dispositivo são quase idênticas, como

geolocalização, câmera, acesso a contatos e armazenamento *off-line*, as APIs [3] específicas para acessar estas capacidades são diferentes em cada plataforma" (ALLEN et al., 2010, p. 5, tradução nossa).

Os aplicativos desenvolvidos com o SDK da plataforma móvel são denominados aplicativos nativos (KAISER et al., 2011). Um aplicativo nativo, segundo Appcelerator (2012, p. 3, tradução nossa), "é um programa específico para dispositivos projetado para ser executado em um dispositivo móvel e seu sistema operacional associado". Segundo Amatya e Kurti (2014, p. 220, tradução nossa), os aplicativos nativos "garantem a melhor usabilidade, os melhores recursos e a melhor experiência móvel global", já que podem "acessar funcionalidades oferecidas por recursos nativos do sistema operacional, tais como GPS, banco de dados, SMS, e-mail, gerenciador de arquivos, entre outros" (DA SILVA; SANTOS, 2014, p. 163). Kaiser et al. (2011, p. 9, tradução nossa) afirmam que esta é "a maneira normal de desenvolver aplicativos para um celular".

Segundo Appcelerator (2012, p. 3, tradução nossa), os aplicativos nativos "são desenvolvidos para alavancar as capacidades do dispositivo móvel diretamente do seu sistema operacional", por este motivo são tão eficientes. Apesar da eficiência, os aplicativos nativos estão atrelados a uma plataforma específica e não podem ser utilizados em outras plataformas (AMATYA; KURTI, 2014). Segundo Appcelerator (2012, p. 3), "80% dos desenvolvedores expressaram a necessidade de estender suas aplicações em mais de um sistema operacional", visto que "os principais sistemas operacionais móveis são: Android, iOS, Symbian, Windows Mobile e BlackBerry" (DE MENDONÇA et al., 2011, p. 2).

Kaiser et al. (2011, p. 2, tradução nossa) afirmam que "o objetivo principal de várias aplicações móveis é atingir o maior número possível de usuários". Para realizar isso "no mundo do código nativo, a principal escolha é reimplementar para múltiplos destinos" (CHARLAND; LEROUX, 2011, p. 52-53, tradução nossa), o que exige habilidade e familiaridade com diferentes linguagens de programação e suas plataformas – ver quadro 1 –, além de desenvolver um aplicativo distinto para cada uma delas (AMATYA; KURTI, 2014) (JÚNIOR et al., 2013).

Segundo Charland e Leroux (2011), desenvolver um aplicativo nativo para cada plataforma é custoso, pois os "custos da equipe de desenvolvimento precisam

[3] API é o acrônimo de *Application Programming Interface* (Interface de programação de aplicativos). É um padrão de programação que permite o acesso a alguma funcionalidade (CANALTECH, 2014).

ser dobrados ou triplicados para entregar essa experiência nativa para múltiplos sistemas operacionais" (APPCELERATOR, 2012, p. 3, tradução nossa). Segundo Xanthopoulos e Xinogalos (2013), o desenvolvimento de aplicativos nativos é mais difícil e requer um maior nível de conhecimento tecnológico e experiência quando comparado com outras aplicações.

Outro problema enfrentado durante o desenvolvimento nativo para múltiplas plataformas é a fragmentação dos dispositivos. Segundo Amatya e Kurti (2014, p. 220, tradução nossa), "dispositivos com diferentes capacidades de processamento, memória, comunicação e tela são exemplos de fragmentação de dispositivos" [4]. Segundo De Mendonça et al. (2011, p. 5) "o programador precisa levar em conta esses vários fatores na hora do desenvolvimento (...) a fragmentação de um Sistema Operacional é um obstáculo durante o desenvolvimento", pois estas características influenciam na execução do aplicativo desenvolvido.

Com o objetivo de superar o problema da fragmentação no desenvolvimento nativo e construir um aplicativo compatível com o maior número possível de sistemas operacionais móveis, uma nova abordagem ao desenvolvimento de aplicativos foi utilizada (AMATYA; KURTI, 2014). Esta nova abordagem gera um aplicativo conhecido como *web app* – aplicativo *web* –, o qual "é uma aplicação disponível na rede e acessada através de um navegador *Web*, desenvolvida para *tablets*, *smartphones* ou qualquer dispositivo com um *browser*" (JÚNIOR et al., 2013). Segundo Xanthopoulos e Xinogalos (2013), estas aplicações não podem ser instaladas fisicamente no dispositivo, visto que o código fonte da aplicação e seus recursos precisam ser obtidos de um servidor na *web*, por isso não estão disponíveis na loja de aplicativos da plataforma.

Segundo Amatya e Kurti os "*web apps* têm pouco ou nenhum suporte para muitos dos recursos nativos e do dispositivo, como câmera, microfone, agenda de contatos, calendário, gestos compostos e interação, etc" (AMATYA; KURTI, 2014, p. 221, tradução nossa). Isto acontece porque o aplicativo é desenvolvido com as seguintes tecnologias:

> Possui como base os padrões HTML (responsável pela estrutura de páginas Web, onde provê seus componentes essenciais), CSS (define o estilo e a apresentação da página) e JavaScript (define interações do usuário com a página, dando dinâmica a mesma). (JÚNIOR et al., 2013, p. 64)

[4] Charland e Leroux (2011) citam o ambiente Android, visto que este é executado em diferentes dispositivos com variações de tela, teclado, capacidade de processamento e armazenamento.

Segundo Hartmann et al. (2011), o desenvolvimento de *web apps* permite uma maior participação de desenvolvedores *web* na construção de aplicativos móveis. Kaiser et al. (2011, p. II) afirmam que “é fácil de projetar um aplicativo com HTML e Javascript e o maior benefício é a facilidade de implantar o aplicativo em diferentes plataformas e dispositivos”. Júnior et al. (2013) comprovaram por meio de um teste estatístico que o tempo médio gasto para construir um aplicativo em HTML é menor que o tempo médio para construir a mesma aplicação na abordagem nativa para a plataforma Android.

Apesar dos benefícios, Júnior et al. (2013) destacam que o armazenamento completo da aplicação em um servidor pode tornar o acesso lento ou indisponível e Xanthopoulos e Xinogalos (2013) afirmam que a principal desvantagem deste modelo é a impossibilidade de acessar o *hardware* e os dados do dispositivo, limitando as operações que o aplicativo pode realizar. Charland e Leroux (2011) afirmam que diante de todas as diferenças existentes entre sistemas operacionais móveis, a única característica em comum das plataformas é que estas trazem um navegador de Internet (*browser*) que pode ser acessado programaticamente por meio de código nativo e permite a interação com os recursos nativos da plataforma por meio da linguagem Javascript.

Segundo da Silva e Santos (2014, p. 165), “este tipo de abordagem permite a criação de aplicativos Web Híbridos utilizando HTML, CSS e Javascript e ainda utilizar recursos nativos do dispositivo móvel e sensores através de uma API Javascript comum”. Para Hartmann et al. (2011, p. 4, tradução nossa) “esta técnica tenta trazer o melhor de ambos os mundos dentro de uma simples solução integrada: flexibilidade dos web apps com a velocidade e a riqueza de recursos dos aplicativos nativos” e está ligada a utilização de um *framework* que “representa a interface às APIs do sistema operacional” (KAISER et al., 2011, p. 11, tradução nossa).

Hartmann et al. afirmam que a integração do código fonte com o aplicativo “remove a necessidade de uma conexão de dados ativa e melhora sua velocidade e capacidade de resposta” (HARTAMANN et al., 2011, p. 4). Segundo Xanthopoulos e Xinogalos (2013), o desenvolvimento de aplicativos híbridos elimina a necessidade de conhecimento detalhado das plataformas que o aplicativo é destinado. Para Kaiser et al. (2011) o maior benefício é o acesso às APIs nativas do dispositivo e a possibilidade de venda do aplicativo como um aplicativo nativo nas lojas de

aplicativos das plataformas. Segundo Amatya e Kurti, os principais benefícios do desenvolvimento híbrido são:

> Unificação do processo de desenvolvimento para diferentes plataformas. Tudo isso combinado com a economia de tempo do desenvolvedor, portanto afeta diretamente nos custos globais de desenvolvimento. (AMATYA; KURTI, 2014, p. 227, tradução nossa)

Kaiser et al. (2011, p. 4, tradução nossa) afirmam que “a tecnologia híbrida não é a melhor solução para todo tipo de aplicação”, sendo esta a mais indicada para “serviços baseados em localização, trabalhar com gráficos e tabelas ou similares” (KAISER et al., 2011, p. 4, tradução nossa). Segundo Charland e Leroux (2011), a tecnologia nativa deve ser utilizada em aplicações que necessitam de gráficos 3D ou outros processamentos pesados, mas destacam que “a pilha de tecnologia *Web* não atingiu o nível de desempenho que podemos obter com o código nativo, porém está chegando perto” (CHARLAND; LEROUX, 2011, p. 53, tradução nossa). Xanthopoulos e Xinogalos afirmam que

> Aplicativos híbridos têm um desempenho médio segundo a percepção de usuários finais comparados a aplicativos nativos, dado que a interface de usuário é ainda baseada na web, sem o uso de componentes nativos otimizados. (XANTHOPOULOS; XINOGALOS, 2011, p. 216, tradução nossa)

Diante das possibilidades de desenvolvimento de aplicativos apresentadas, para o desenvolvimento deste projeto optou-se pela abordagem híbrida, visto que o aplicativo não exigirá processamento pesado. Este modelo mostrou-se mais interessante que a abordagem nativa porque permite a construção de um aplicativo multiplataforma sem preocupar-se com as capacidades do dispositivo, além de gerar o aplicativo para a loja virtual. Quanto à abordagem *web*, o modelo híbrido elimina a necessidade de um servidor para armazenar o aplicativo e não requer que o usuário acesse a aplicação pelo do navegador do dispositivo, reduzindo o esforço necessário para acessá-la. A utilização de HTML, CSS e Javascript para construção do aplicativo tornam o desenvolvimento mais fácil e rápido, portanto mais interessante para o projeto.

Para desenvolver um aplicativo híbrido é necessário um *framework* que fornece o acesso às capacidades nativas do dispositivo e constrói o aplicativo para as lojas virtuais. Este assunto é discutido na próxima seção.

2.4. *FRAMEWORK* DE APLICATIVOS HÍBRIDOS

Segundo Allen et al. (2010, p. 9, tradução nossa), “a indústria produz múltiplas plataformas que essencialmente fornecem as mesmas soluções para diferentes segmentos de mercado” e por este motivo os programadores que necessitam desenvolver a mesma aplicação para múltiplas plataformas acabam criando “bibliotecas e frameworks que abstraem as diferenças, tornando mais fácil o desenvolvimento de uma aplicação que executa entre as plataformas” (ALLEN et al., 2010, p. 9, tradução nossa). Hartmann et al. definem os *frameworks* de aplicativos híbridos como:

> Um conjunto de bibliotecas, componentes de software e diretrizes de arquitetura que fornecem ao desenvolvedor um abrangente conjunto de ferramentas para construir uma aplicação móvel completa, de cima para baixo. (HARTMANN et al., 2011, p. 6, tradução nossa)

Utilizando um *framework* de aplicativos híbridos os “desenvolvedores precisam escrever um simples código-fonte para qualquer sistema operacional móvel suportado pela ferramenta” (PALMIERI et al., 2012, p. 182, tradução nossa). Segundo Hartmann et al. (2011, p. 9-10, tradução nossa), “você basicamente precisa colocar as bibliotecas no lugar correto e começar a codificar com tecnologias web familiares para produzir rapidamente um aplicativo funcional”, já que “o framework também fornece acessos para parte das funcionalidades nativas dos dispositivos (tais como, acelerômetro e câmera) através de um *wrapper* (normalmente o JavaScript) para a API nativa” (OEHLMAN; BLANC, 2011, p. 2).

Para Palmieri et al. (2012) estas ferramentas tiveram um aumento de popularidade entre os usuários graças à facilidade de utilização, economia de tempo durante o desenvolvimento e devido a “sua característica de compilar o código-fonte da aplicação para vários sistemas operacionais suportados” (PALMIERI et al., 2012, p. 179, tradução nossa), além de afirmar que estas ferramentas trouxeram:

> Redução das habilidades requeridas, redução da codificação, redução do tempo de desenvolvimento e dos custos de manutenção de longo prazo; Decremento do conhecimento de API, maior facilidade de desenvolvimento, incremento na participação do mercado. (PALMIERI et al., 2012, p. 180, tradução nossa)

Segundo Charland e Leroux (2011, p. 50, tradução nossa), “existem mais de 20 frameworks” disponíveis, mas poucos geram aplicativos para um grande número

de plataformas existentes (PALMIERI et al., 2012). Kaiser et al. (2011, p. 11, tradução nossa) afirmam que “as diferenças entre os frameworks são basicamente na interface do usuário, nas animações e na velocidade”. Palmieri et al. (2012) afirmam que durante a escolha de um *framework* é necessário analisar algumas de suas características, como os sistemas operacionais móveis suportados, a licença da ferramenta, a disponibilidade de APIs, a linguagem de programação requerida para desenvolvimento e o ambiente de desenvolvimento necessário.

Partindo destas premissas, para construir o aplicativo híbrido foi utilizado o *framework* Apache Cordova (CORDOVA, 2015a), o qual permite acessar as APIs nativas das plataformas e gerar um aplicativo que pode ser disponibilizado na loja de aplicativos das plataformas.

2.4.1. Apache Cordova

O Apache Cordova é um *framework* de desenvolvimento de aplicativos móveis licenciado sob a Licença Apache 2.0 (WARGO, 2013) – a qual garante que “ele será sempre gratuito e open-source [[5]]” (CORDOVA, 2015a, tradução nossa) – e que permite ao desenvolvedor “utilizar tecnologias web padrão como HTML5, CSS3 e Javascript para o desenvolvimento multiplataforma, evitando a linguagem de desenvolvimento nativo de cada plataforma móvel” (CORDOVA, 2015a, tradução nossa). Segundo Wargo (2013, p. 1, tradução nossa), o *framework* “implementa um conjunto de APIs que estendem as capacidades nativas do dispositivo (como câmera, acelerômetro, aplicativo de contatos, e assim por diante) para uma aplicação web executando dentro de um container nativo”.

Wargo (2013) afirma que o Apache Cordova constrói aplicativos para as plataformas móveis da Google (Android), Apple (iOS), Samsung (bada), BlackBerry (BlackBerry 10), Microsoft (Windows Phone 7 e 8, Windows 8), Firefox (Firefox OS) e Linux Foundation (Tizen). Segundo Cordova (2015c), para criar um aplicativo é necessário ter o SDK da plataforma desejada instalado no computador. Para gerar os aplicativos e testá-los em emuladores ou dispositivos reais é utilizada a interface de linha de comando do Cordova, a Cordova-CLI (CORDOVA, 2015b), a qual é uma

[5] Wargo (2013, p. 9, tradução nossa) destaca que por ser um projeto *open-source,* “as empresas dos sistemas operacionais dos dispositivos estão fortemente envolvidas e têm interesse em fazer este projeto bem sucedido”.

ferramenta de alto nível que abstrai algumas funcionalidades de *scripts* de baixo nível e permite gerar aplicativos para múltiplas plataformas de uma só vez, além de possibilitar a adição de *plug-ins* ao aplicativo (CORDOVA, 2015a).

Um *plug-in* do Apache Cordova é "um conjunto de arquivos que juntos ampliam ou melhoram as capacidades de uma aplicação Cordova" (WARGO, 2013, p. 206, tradução nossa). Segundo Cordova (2015d, tradução nossa), os *plug-ins* "fornecem acesso ao dispositivo e às funcionalidades da plataforma que normalmente estão indisponíveis para aplicativos baseados na web". Estes permitem "chamar código nativo a partir do Javascript" (CORDOVA, 2015a, tradução nossa), já que são compostos por "uma biblioteca Javascript que expõe as capacidades nativas para a aplicação web e o código nativo correspondente, executado dentro do container que implementa a parte nativa da API" (WARGO, 2013, p. 3, tradução nossa).

Quando um *plug-in* é chamado, "uma camada especial de dentro da aplicação traduz a chamada da API do Cordova para a API nativa apropriada" (WARGO, 2013, p. 6, tradução nossa), então o recurso nativo é acessado. Segundo Wargo (2013), o Cordova fornece acesso às APIs nativas de câmera, acelerômetro, bússola, conexão, arquivos, geolocalização, contatos, entre outras. O principal *plug-in* utilizado neste projeto é o *Barcode Scanner*[6], que permite escanear os códigos QR.

Um aplicativo desenvolvido com o Apache Cordova é construído como uma página *web* (CORDOVA, 2015a), por isso qualquer ambiente de desenvolvimento *web* pode ser utilizado (WARGO, 2013), mas distingue-se do modelo tradicional da *web* já que todo o conteúdo da aplicação – HTML, CSS, Javascript, arquivos de mídia – é adicionado ao aplicativo, o qual é embalado em um container nativo que permite a sua disponibilização em lojas de aplicativos (CORDOVA, 2015a) e elimina a necessidade de obter o conteúdo de um servidor externo[7] (WARGO, 2013). Segundo Cordova, a construção de um aplicativo é realizada a partir dos seguintes passos:

> A interface de linha de comando copia um conjunto comum de recursos web dentro de subdiretórios para cada plataforma móvel, faz algumas mudanças

[6] https://github.com/wildabeast/BarcodeScanner

[7] Wargo (2013, p. 10, tradução nossa) define os aplicativos gerados com o Apache Cordova como "aplicações *web* executando dentro de um container de um aplicativo nativo do lado do cliente".

> necessárias para cada uma delas e executa scripts de construção para gerar os binários da aplicação. (CORDOVA, 2015a, tradução nossa)

O binário da aplicação é o aplicativo final que será distribuído entre os usuários. Quando um usuário acessa o aplicativo, um WebView[8] ocupa toda a tela do dispositivo e carrega a página inicial do aplicativo desenvolvido, permitindo ao usuário interagir com o conteúdo adicionado (WARGO, 2013). Segundo Cordova (2015a), o *framework* fornece apenas o ambiente para executar o aplicativo e não fornece nenhum recurso para interação do usuário. Wargo também faz uma observação sobre esta característica:

> O framework Cordova não faz nada para melhorar a interface do usuário (UI) da aplicação Cordova. O framework permite acesso aos recursos específicos do dispositivo e das aplicações e deixa aos desenvolvedores o visual de suas aplicações para como entenderem. (WARGO, 2013, p. 66, tradução nossa)

Diante desta limitação, Wargo (2013) sugere a utilização de *frameworks* de estilização para desenvolver a interface do usuário da aplicação. Segundo Cordova (2015a), é necessário selecionar um material de terceiro para incluir estas características ao aplicativo. Xanthopoulos e Xinogalos (2013, p. 219, tradução nossa) afirmam que "aplicativos híbridos simulam os componentes nativos de interface de usuário através de bibliotecas específicas" e para realizar esta tarefa neste projeto o Ionic *Framework* foi utilizado.

Os *frameworks* multiplataforma, segundo Ribeiro e da Silva (2014, p. 260, tradução nossa), "estão revolucionando a forma como as pessoas desenvolvem aplicativos móveis, reduzindo o esforço requerido para produzi-los e publicá-los em vários mercados móveis". Segundo Palmieri et al. (2012, p. 182, tradução nossa), "adotando uma abordagem multiplataforma, a construção e manutenção das aplicações pode ser melhorada", pois "é mais fácil suportar múltiplos alvos a partir de uma simples base de código orientada à web" (CHARLAND; LEROUX, 2011, p. 50, tradução nossa). Segundo Kaiser et al. (2011) é possível economizar dinheiro utilizando tecnologias web e seus *frameworks* quando comparado ao desenvolvimento nativo para várias plataformas.

[8] "Um WebView é um componente de uma aplicação nativa que é utilizado para processar conteúdo web (normalmente páginas HTML) dentro de uma janela de um aplicativo nativo ou tela" (WARGO, 2013, p. 3, tradução nossa).

Diante de todos os benefícios apresentados, acredita-se que o Apache Cordova facilita o desenvolvimento de aplicativos quando permite o acesso às funcionalidades nativas das plataformas e ao mesmo tempo permite a geração de um binário que pode ser disponibilizado em uma loja virtual de aplicativos. Por ser uma ferramenta *open source*, nenhum custo adicional foi incluído, influenciando na escolha deste *framework* para este projeto. A possibilidade de gerar o aplicativo para múltiplas plataformas também é importante, pois mantém um único código-fonte para múltiplas plataformas, o que contribui diretamente na estrutura e manutenção do projeto. O desenvolvedor preocupa-se apenas com o visual e o funcionamento da aplicação, o restante das tarefas é facilitado com a utilização do Apache Cordova.

2.5. CÓDIGO QR

O Código QR (*Quick Response Code*) é uma matriz de símbolos de duas dimensões desenvolvida pela empresa japonesa Denso Wave no ano de 1994 (DE LAET, 2010). Segundo Soon (2008), o Código QR foi concebido com o objetivo de controlar e identificar a produção de peças automotivas, porém acabou sendo largamente utilizado em outras áreas. Segundo de Laet (2010, p. 3), o Código QR "possui o formato de um mosaico, podendo, por exemplo, armazenar 7.089 dígitos e 4.296 caracteres, numéricos e alfanuméricos, respectivamente". A figura abaixo apresenta um Código QR que contém a sigla UTFPR armazenada.

Figura 1 – Exemplo de Código QR
Fonte: Autoria própria

O código QR possui tolerância a erros, portanto é possível obter os dados mesmo quando o código encontra-se sujo ou danificado (DE LAET, 2010). Além disso, é possível ler os dados de todas as direções (360º), efetuar uma leitura rápida – quando comparado aos códigos de barras convencionais – e criptografar os dados armazenados (SOON, 2008). Segundo de Laet (2010), um sistema de código QR conta com três papéis: Um *software* que gera o código QR, uma impressora ou outro

meio que disponibiliza o código QR e um *scanner* de código QR que efetuará a leitura e tradução dos dados.

Soon (2008) cita a utilização de códigos QR em *tickets* de ônibus, controle de passageiros de embarcações, identificação de pacientes em hospitais do Japão, rastreamento de animais em fazendas da Austrália, entre outros. De Laet (2010) em sua obra utilizou o Código QR para a identificação de trabalhadores das usinas sucroalcooleiras. No Brasil, o Conselho Nacional de Trânsito (Contran) desenvolveu novos modelos da Carteira Nacional de Habilitação (CNH), do Certificado de Registro de Veículo (CRV) e do Certificado de Registro e Licenciamento de Veículo (CRLV) que possuirão, a partir de Julho de 2015, as informações do condutor e do veículo – conforme o documento – codificadas em um Código QR impresso nos documentos a fim de dificultar a falsificação dos mesmos (CARTEIRAS..., 2014).

Fincotto (2014, p. 3) afirma que o código QR é “mais utilizado para a leitura através de dispositivos móveis”, portanto é uma excelente escolha para este projeto devido à necessidade de se obter as informações da comanda.

2.6. IONIC FRAMEWORK

O Ionic é um “poderoso framework de desenvolvimento de aplicativos nativos em HTML5 que ajuda a construir aplicativos móveis com impressão nativa, tudo com tecnologias como HTML, CSS e Javascript” (IONIC, 2015b, tradução nossa). Segundo Ionic (2015a), este *framework* não foi construído com a intenção de ser executado em navegadores *desktop*, como o Google Chrome ou Safari, mas sim em navegadores de baixo nível, como o WebView, utilizados em aplicativos híbridos. O Ionic é um *framework open-source* licenciado sob a licença do MIT[9], a qual garante que “você pode utilizar o Ionic nos seus próprios projetos comerciais ou pessoais gratuitamente” (IONIC, 2015b, tradução nossa).

Segundo Ionic (2015a), este *framework* pode ser caracterizado como um *framework* de interface de usuário (UI), pois gerencia toda a aparência e as interações da interface do aplicativo construído. Além disso, Bradley (2013) destaca a presença de padrões de design Javascript, os quais permitem ao desenvolvedor

[9] http://opensource.org/licenses/MIT

“criar os mesmos tipos de interações poderosas de interface de usuário vistas em aplicativos nativos, como menus laterais e controles de navegação” (BRADLEY, 2013, tradução nossa).

O Ionic fornece “elementos e layouts de interface de usuário com estilo nativo que você obteria com um SDK nativo no iOS ou Android, mas realmente não existiam antes na web” (IONIC, 2015a, tradução nossa). Bradley (2013) afirma que o *framework* possui um design similar ao do iOS – sistema operacional móvel da Apple[10] – porém destaca que o desenvolvedor é livre para alterar ou sobrescrever os modelos conforme a necessidade, visto que os estilos são todos desenvolvidos com CSS.

Apesar do funcionamento multiplataforma dos aplicativos híbridos, Ionic (2015b) destaca que o *framework* funciona nas versões superiores do iOS 7 e do Android 4.1, visto que estas versões já contam com um WebView com desempenho superior aos anteriores e que pode ser acessado programaticamente. Segundo Bradley (2013, tradução nossa) o “Ionic é aquela peça que faltava quando se cria aplicativos nativos com padrões web. Apenas insira algum CSS e Javascript e você vai rapidamente desenvolver aplicações nativas com HTML5”.

A utilização de um *framework* de interface de usuário – como o Ionic – permite construir uma aplicação com uma estrutura bem definida e com padrões de *design* semelhantes àqueles de aplicativos nativos. Segundo Bradley (2013), os *frameworks* UI permitem desenvolver uma aplicação de alto nível e fornecem um bom ponto de partida para os projetos. Com a utilização do Ionic *Framework* criou-se uma aplicação com um visual padronizado e semelhante ao de um aplicativo nativo.

Ionic (2015b) afirma que algumas tarefas desempenhadas pelo Ionic *Framework* requerem o AngularJS, portanto é necessário entender como ele funciona para explorar todo o potencial da ferramenta.

2.7. ANGULARJS

O AngularJS é um *framework* estrutural Javascript de código-fonte aberto (*open source*) para criação de aplicações dinâmicas da *web* (ANGULARJS, 2015). É

[10] http://www.apple.com/br/

patrocinado e mantido pela Google (FREEMAN, 2014) e, segundo Freeman (2014, p. 3, tradução nossa), “tem sido utilizado em algumas das maiores e mais complexas aplicações web” já que “drena alguns dos melhores aspectos de desenvolvimento do lado servidor e utiliza-os para melhorar o HTML no navegador, criando uma fundação que torna simples e fácil a construção de aplicações custosas” (FREEMAN, 2014, p. 3, tradução nossa)

Dayley (2014, p. 397, tradução nossa) afirma que o AngularJS proporciona um método estruturado e uma abordagem clara para a criação de aplicações *web* e também “fornece o melhor do Javascript e jQuery”, já que foi construído a partir de uma pequena versão da biblioteca jQuery[11] chamada jqLite (FREEMAN, 2014). Com AngularJS é possível utilizar o HTML como linguagem de marcação (ANGULARJS, 2015) e estendê-lo “adicionando novos elementos, atributos, classes, e (embora raramente usado) comentários especiais” (FREEMAN, 2014, p. 19, tradução nossa). Segundo Freeman (2014), qualquer editor de texto pode ser utilizado para construir a aplicação e quando a biblioteca do AngularJS é carregada na aplicação o seguinte processo ocorre:

> A biblioteca AngularJS dinamicamente compila o HTML em um documento, a fim de localizar e processar estas adições e criar uma aplicação. Você pode complementar as funcionalidades embarcadas com código Javascript para customizar o comportamento da aplicação e definir suas próprias adições para o HTML. (FREEMAN, 2014, p. 19, tradução nossa)

As aplicações desenvolvidas com AngularJS “são construídas em torno de um padrão de design chamado *Model-View-Controller* (MVC)” (FREEMAN, 2014, p. 3, tradução nossa), o qual parte da premissa da separação de interesses. Em um projeto *web* significa “separar os dados [model], a lógica que opera sobre os dados [controller] e os elementos HTML usados para mostrar os dados [view]” (FREEMAN, 2014, p. 47, tradução nossa). Este padrão vem sendo utilizado desde o final da década de 1970 e resulta em maior facilidade de desenvolvimento, manutenção e testes, além contribuir para a criação de uma aplicação padronizada e extensível (FREEMAN, 2014). Dayley (2014) afirma que uma aplicação Javascript sem este padrão pode facilmente sair do controle quando se tenta gerenciar todo o projeto.

[11] jQuery é uma biblioteca da linguagem Javascript utilizada explicitamente para manipular os elementos da página HTML e então construir um aplicativo, sendo mais indicada para aplicações de baixo nível e que requerem resultados imediatos (FREEMAN, 2014).

Segundo Dayley (2014), o AngularJS força e facilita a utilização correta do padrão MVC, o que colabora para o desenvolvimento de uma aplicação organizada. Além disso, o autor afirma que com o *framework* é possível desenvolver um código HTML mais intuitivo e de fácil manutenção, utilizar filtros que facilmente formatam os dados, focar apenas na lógica da aplicação e não em pequenos detalhes e modelar uma aplicação a partir dos dados e não a partir dos elementos da página (DAYLEY, 2014). Segundo o seu mantenedor, o "Angular simplifica o desenvolvimento de aplicações apresentando um nível maior de abstração para o desenvolvedor" (ANGULARJS, 2015), manipula os elementos da página (DOM) e as requisições que são feitas para o servidor e também automatiza muitas tarefas necessárias para uma aplicação *web* funcionar. Para Freeman,

> O objetivo do AngularJS é trazer ferramentas e recursos que estão disponíveis somente no desenvolvimento do lado servidor para o cliente web e, fazendo isso, tornar mais fácil desenvolver, testar e manter complexas e custosas aplicações web. (FREEMAN, 2014, p. 45, tradução nossa)

Diante de todos os benefícios apresentados e devido a necessidade do Ionic de utilizar o AngularJS, utilizou-se este *framework* para facilitar o desenvolvimento da aplicação, tanto na criação de uma aplicação padronizada quanto na obtenção dos dados do servidor, assunto que será abordado na próxima seção.

2.8. OBTENDO OS DADOS DA APLICAÇÃO

Um aplicativo móvel pode obter dados para a aplicação comunicando-se com um servidor *web* disponível na rede. Segundo Dayley (2014, p. 10, tradução nossa), "o foco principal do servidor *web* é manipular as requisições dos navegadores", portanto ele pode receber dados, solicitações de documentos e requisições AJAX[12]. Como um aplicativo híbrido é executado dentro de um pequeno navegador do dispositivo móvel, o mesmo formato de troca de mensagens de um navegador comum pode ser utilizado para receber os dados na aplicação móvel. Por

[12] O AJAX (Asynchronous Javascript and XML) é "apenas uma requisição GET ou POST que é feita diretamente pelo Javascript em execução no navegador" (DAYLEY, 2014, p. 9, tradução nossa).

este motivo, para definir a comunicação entre o aplicativo e o servidor *web* foi utilizado o protocolo HTTP (DAYLEY, 2014).

2.8.1. Protocolo HTTP

O termo protocolo define "um conjunto bem conhecido de regras e formatos a serem usados na comunicação entre processos a fim de realizar uma determinada tarefa" (COULOURIS et al., 2001, p. 79). O Protocolo de Transferência de Hipertexto (tradução livre de HTTP – *Hypertext Transfer Protocol*) "define quais tipos de requisições podem ser feitas, bem como o formato destas requisições e a resposta HTTP" (DAYLEY, 2014, p. 8, tradução nossa). Segundo Coulouris et al. (2001, p. 25), o HTTP "define as maneiras pelas quais os navegadores e outros tipos de cliente interagem com os servidores web". Ihrig (2013) afirma que este protocolo é baseado em texto, funciona sobre o protocolo TCP[13] e é responsável por conduzir a *web*.

Quando uma requisição é realizada, o cliente aponta para um URL[14] (*Uniform Resource Locator*) e faz uma requisição HTTP, então o servidor que armazena este URL processa a requisição e responde ao cliente retornando uma mensagem (IHRIG, 2013). Coulouris et al. citam os processos de uma requisição HTTP de maneira mais aprofundada, que se segue:

> Cada requisição especifica o nome de um método a ser aplicado em um recurso no servidor e o URL desse recurso. A resposta fornece o status da requisição. As mensagens de requisição e resposta também podem conter dados de recurso, como o conteúdo de um formulário ou a saída de programa executado no servidor web. (COULOURIS et al., 2001, p. 152)

Os métodos do protocolo HTTP – ou verbos HTTP – determinam a ação que será realizada sobre o recurso da URL requisitada (IHRIG, 2013). Segundo Coulouris et al. (2001, p. 151), estes métodos "são aplicáveis a todos os seus recursos" e são fixos ao protocolo. O quadro 2 apresenta os principais métodos HTTP, segundo Freeman (2014), e uma breve descrição de suas respectivas funções.

---

[13] TCP (*Transmission Control Protocol*) é um protocolo de transporte que oferece um serviço para transportar os *bytes* pela rede (COULOURIS et al., 2001).

[14] Os URLs "Identificam os documentos e outros recursos armazenados como parte da web" (COULOURIS et al., 2001)

Método	Descrição
GET	Operação de leitura utilizada para obter um recurso, a qual não deve modificar o estado do servidor. Segundo Coulouris et al., “Se o URL se referir a dados, então o servidor web responderá retornando os dados identificados por esse URL. Se o URL se referir a um programa, então o servidor web executará o programa e retornará sua saída para o cliente” (COULOURIS et al., 2001, p. 152).
POST	Operação utilizada para criação de novos recursos dentro do servidor, como “submissão de formulários HTML e adição de dados a uma base de dados” (IHRIG, 2013, p. 169, tradução nossa).
PUT	Similar ao método POST, pode ser utilizado para alterar um recurso existente ou então inserir um novo recurso no servidor, caso ele ainda não exista.
DELETE	Utilizado para remover um recurso já armazenado no servidor.

Quadro 2 – Métodos HTTP e suas respectivas funções
Fonte: Adaptado de (IHRIG, 2013) (COULOURIS et al., 2001).

Após o recebimento de uma requisição HTTP o servidor retorna uma mensagem informando o *status* da requisição, a qual contém um valor inteiro de três dígitos para a identificação do resultado no programa e uma frase textual para o entendimento do usuário (COULOURIS et al., 2001). O quadro 3 apresenta alguns dos códigos de resposta HTTP e sua descrição.

Código	Descrição
200	Indica que a requisição foi realizada com sucesso.
301	Indica que o recurso está em outra localização – URL – portanto será feito um redirecionamento para a correta localização.
404	O servidor não localizou o recurso solicitado na URL.
500	O servidor encontrou um erro durante a execução da requisição.

Quadro 3 – Códigos de resposta HTTP e sua descrição
Fonte: Adaptado de (IHRIG, 2013).

Segundo Freeman (2014), a maneira mais utilizada para obter dados de um servidor é utilizando AJAX para chamar os códigos do servidor, o qual retorna, por exemplo, um arquivo JSON com as informações para alterar os dados de um formulário HTML. Este é o método utilizado neste projeto para obter os dados armazenados no servidor.

2.8.2. JSON

JSON (*Javascript Object Notation*) é um formato de troca de dados em texto puro (IHRIG, 2013) completamente independente de linguagem de programação (JSON, 2015). O JSON “é baseado em um subconjunto da linguagem de

programação Javascript” (JSON, 2015, tradução nossa), portanto “sintaticamente o JSON é muito similar à sintaxe de objetos literais do Javascript” (IHRIG, 2013, p. 263, tradução nossa). Segundo Freeman (2014), é fácil manipular o JSON na linguagem Javascript, o que colaborou para o sucesso deste formato de troca de dados.

Segundo Ihrig (2013), o “JSON é usado como um mecanismo para serializar estruturas de dados em Strings” (IHRIG, 2013, p. 263, tradução nossa), as quais podem ser enviadas pela rede ou gravadas em arquivos. Dayley (2014, p. 73, tradução nossa) afirma que o “JSON é um método muito leve para converter objetos Javascript em um formato de *String* e depois voltar novamente”. Freeman (2014, p. 116, tradução nossa) afirma que o “JSON tornou-se o formato de troca de dados para aplicações *web*”, já que suporta alguns dos tipos de dados nativos da linguagem de programação Javascript (IHRIG, 2013): “*Number, String, Boolean, Array, Object,* e o tipo especial *null*” (FREEMAN, 2014, p. 116, tradução nossa). Um objeto JSON é definido com a seguinte estrutura:

> Objetos JSON começam com uma chave de abertura, {, e terminam com uma chave de fechamento, }. Entre as chaves estão zero ou mais pares chave/valor, conhecidos como membros. Os membros são delimitados por vírgulas, enquanto dois pontos são usados para separar a chave de um membro do seu valor correspondente. A chave deve ser uma string entre aspas. (IHRIG, 2013, p. 263, tradução nossa)

A estrutura utilizada nos objetos JSON é “familiar para programadores das linguagens da família C, incluindo C, C++, C#, Java, Javascript, Perl, Python, e muitas outras” (JSON, 2015, tradução nossa), portanto torna-o um formato de troca de dados ideal para diferentes plataformas (JSON, 2015). A figura 2 apresenta um exemplo de um arquivo JSON, no qual a chave1 e a chaveN remetem a um *String*, a chave2 a um valor numérico e a chave3 a um valor booleano.

```
{
   “chave1”:”valor1”,
   “chave2”:”2,
   “chave3”:true,
   “chaveN”:”valorN”
}
```

Figura 2 – Estrutura de um arquivo JSON
Fonte: Autoria própria.

Segundo Dayley (2014), o JSON utiliza poucos caracteres para gerar o objeto, possui um desempenho elevado e permite ao desenvolvedor visualizar as

informações do objeto, já que o seu formato permite a leitura e escrita por humanos (JSON, 2015). Ihrig (2013) afirma que o “JSON é simples para ler, escrever e entender” (IHRIG, 2013, p. 270, tradução nossa) e facilita a integração entre as aplicações.

A próxima seção trata de assuntos referentes ao servidor da aplicação.

2.9. SERVIDOR DA APLICAÇÃO

Segundo Coulouris et al. (2001, p. 21), o termo servidor remete a um programa sendo executado em um computador ligado na rede, capaz de aceitar “pedidos de programas em execução em outros computadores para efetuar um serviço e responder apropriadamente”. Segundo os autores, o termo serviço é utilizado para referenciar a parte responsável por gerenciar os recursos[15] e disponibilizá-los aos usuários e a outros programas (COULOURIS et al., 2001). Dentro da *web*, o servidor responsável por receber os pedidos dos clientes – envio de dados, solicitação de documentos, requisições AJAX – é denominado servidor *web*, que determina qual tarefa deve ser realizada por meio dos cabeçalhos HTTP e pela informação da URL (*Uniform Resource Locator*) (DAYLEY, 2014, p. 10).

Segundo Almeida et al. (2013), o aumento do número de usuários da Internet exige que servidores *web* manipulem cada vez mais requisições simultâneas. Para Almeida et al. (2013), investir em infraestrutura e processamento pode ser uma solução para manipular mais requisições, porém destacam que “do ponto de vista do *software* é necessário que a aplicação esteja preparada para assimilar uma carga crescente de conexões e processamentos, ou seja, escalabilidade” (ALMEIDA et al., 2013, p. 101). Além da escalabilidade, quando um servidor *web* recebe requisições simultâneas para um mesmo recurso (página *web*, imagem, arquivo), um problema denominado concorrência pode ocorrer.

Segundo Coulouris et al. (2001, p. 34), o termo concorrência refere-se ao acesso simultâneo de clientes a um recurso compartilhado. Se a concorrência não for gerenciada, o recurso compartilhado pode assumir resultados inconsistentes ou até mesmo ficar indisponível (COULOURIS et al., 2001). Junior (2012, p. 1) afirma

[15] Recursos de *hardware* (discos, impressoras) e recursos de *software* (arquivos, bancos de dados) (COULOURIS et al., 2001).

que para gerenciar este tipo de problema, “muitos *softwares* se utilizam do modelo de threads para alcançar concorrência de I/O[16] e outros recursos”.

Em servidores *web* que trabalham com o modelo baseado em *threads*, “cada requisição é atendida por uma thread separada” (JUNIOR, 2012, p. 1). Segundo Dayley (2014), cada requisição recebida no servidor é designada para uma *thread* disponível que realiza todas as tarefas necessárias, envia a resposta ao cliente e depois é destruída. Welsh et al. (2001) destacam que cada *thread* conta com operações sincronizadas que protegem os recursos compartilhados. Neste modelo, “quanto maior o número de conexões simultâneas, maior será o número de threads abertas” (JUNIOR, 2012, p. 1). Kegel (2011 apud JUNIOR, 2012, p.1) cita que “o limite de conexões simultâneas suportadas pela maioria dos *web servers* é de 10.000 (dez mil)” e, segundo Welsh et al. (2001), o modelo de *thread* por requisição é utilizado na maioria dos servidores.

A figura 3 esboça a manipulação de duas requisições dentro de um servidor *web* baseado em *threads*. Na Thread-1 é buscado e enviado o conteúdo de um arquivo e na Thread-2 são buscadas as informações em um banco de dados e depois enviadas para quem fez a requisição. Vale lembrar que após o término das operações as *threads* são destruídas.

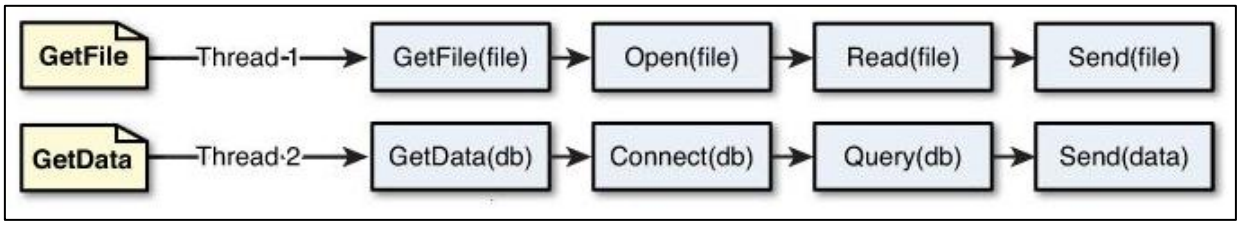

Figura 3 – Manipulação de requisições em servidores web baseados em *threads*
Fonte: Dayley (2014, p. 55).

Welsh et al. (2001, p. 231) citam que um alto número de *threads* pode levar à diminuição do desempenho do servidor, portanto um elevado número de requisições simultâneas pode comprometer o funcionamento do mesmo. Além da redução de desempenho, Welsh et al. (2001) destacam que *threads* são projetadas para apoiar multiprogramação e por este motivo os recursos de *hardware* são virtualizados pelo sistema operacional, o qual encarrega-se de gerenciar os recursos e processos, limitando a aplicação. Além disso, Ihrig (2013) destaca o alto consumo

[16] I/O refere-se à entrada/saída (I*nput/Output*). Este termo é utilizado para designar as operações de requisição e resposta presentes no servidor.

de recursos computacionais durante a realização das operações de criação e destruição de uma *thread*.

Diante de alguns limites impostos pelo modelo baseado em *threads*, Almeida et al. (2013, p. 102) defendem que “muitos desenvolvedores têm optado por um modelo baseado em eventos para gerenciar concorrência”, pois segundo Junior (2012, p. 1) “um modelo baseado em eventos tem se mostrado mais eficiente para situações onde há várias solicitações simultâneas e grande demanda por operações de I/O”. Segundo Welsh et al. (2001, p. 232, tradução nossa), um servidor com a abordagem orientada a eventos “consiste de um pequeno número de threads (tipicamente uma por CPU) que repetem continuamente, processando eventos [17] de diferentes tipos a partir de uma fila”.

No modelo baseado em eventos, a *thread* que executa o processo de repetição – planejador de eventos – captura o evento superior da fila de eventos, executa-o e depois captura o próximo item da fila (DAYLEY, 2014). Welsh et al. destacam que neste modelo “a aplicação é responsável por decidir quando processar cada evento de entrada” (WELSH et al., 2001, p. 232), ao contrário do modelo de *threads* onde o sistema operacional gerencia a execução dos processos. Além disso, Welsh et al. fazem uma observação a respeito deste modelo:

> Uma importante limitação deste modelo é que ele assume que threads de manipulação de eventos não bloqueiam e por esta razão mecanismos de I/O não bloqueantes devem ser empregados. (WELSH et al., 2001, p. 232, tradução nossa)

A utilização de mecanismos de I/O não bloqueantes afeta a execução dos eventos manipulados pelo planejador de eventos, pois se um evento bloqueante é executado, o planejador de eventos só vai poder executar o próximo evento após o encerramento do evento bloqueante. Em alguns casos o evento bloqueante pode comprometer o funcionamento de todo o sistema caso este demore demais para ser finalizado, principalmente no funcionamento de um servidor *web*, pois se o servidor ficar bloqueado as novas requisições podem ficar esperando ou não ser atendidas. A figura 4 ilustra o modelo de um servidor orientado a eventos, no qual a *thread* principal – planejador de eventos – manipula os eventos da rede, do disco e das

[17] Segundo Junior (2012, p. 2), “evento é uma indicação de algo que aconteceu” e estes “podem ser gerados pelo sistema operacional ou internamente pela aplicação” (WELSH et al., 2001, p. 232, tradução nossa).

requisições que chegam para o servidor e fica responsável por escolher qual evento processar.

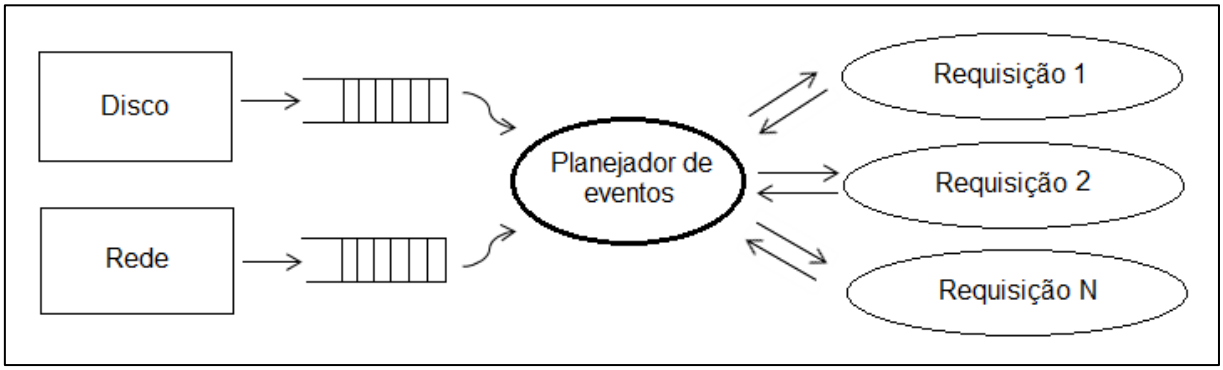

Figura 4 – Manipulação de eventos em um servidor orientado a eventos
Fonte: Adaptado de Welsh et al. (2001, p. 232)

Teixeira (2011, p. 8, tradução nossa) afirma que “por algum tempo, a comunidade ‘hacker’ de programação de sistemas C tem reconhecido que a programação orientada a eventos é a melhor forma de escalar um servidor para manipular muitas conexões simultâneas”. Tilkov e Vinoski (2010, p. 80, tradução nossa) afirmam que a “programação orientada a eventos oferece uma alternativa mais eficiente e escalável que fornece aos desenvolvedores mais controles sobre a alternância entre as atividades da aplicação”. Junior (2012, p. 1) afirma que “há vários sistemas desenvolvidos utilizando um modelo baseado em eventos”, porém para a realização deste trabalho foi desenvolvido um servidor *web* sobre a plataforma Node.js, abordada na sequência.

2.9.1. Node.js

Node.js[18] é uma plataforma para a fácil construção de rápidas e escaláveis aplicações de rede (NODEJS, 2015a). Foi construído sobre a plataforma de execução de Javascript Chrome V8[19] (NODEJS, 2015a), que segundo Schroeder e Dos Santos (2014, p. 4) aumentou seu desempenho. Segundo Dayley (2014, p. 39, tradução nossa), o “ambiente do Node.js é extremamente limpo e fácil de instalar,

[18] (NODEJS, 2015a)
[19] O Chrome V8 é o “motor Javascript de alto desempenho e *Open Source* da Google” (CHROME..., 2015, tradução nossa). É responsável por “compilar e executar código-fonte Javascript, manipular a alocação de memória para objetos” (CHROME..., 2015, tradução nossa) e eliminar os objetos da memória quando não são necessários, além de possibilitar que qualquer aplicação escrita em C++ exponha seus objetos e funções para acesso por meio de código-fonte Javascript (CHROME..., 2015).

configurar e implantar", além de ser um "ambiente Javascript do lado servidor" (TILKOV; VINOSKI, 2010, p. 80, tradução nossa). Dayley (2014, p. 39, tradução nossa) afirma que o "Node.js foi desenvolvido em 2009 por Ryan Dahl como uma resposta à frustação causada por questões de concorrência, especialmente quando se tratava de serviços *web*". Hoje o *framework* é "usado principalmente para criar servidores altamente escaláveis para aplicações *web*" (IHRIG, 2013, p. 1, tradução nossa).

A plataforma utiliza um modelo orientado a eventos com I/O não bloqueante, tornando-o leve e eficiente (NODEJS, 2015a), além de "executar dentro de uma simples *thread*" (IHRIG, 2013, p. 2, tradução nossa). Essa combinação torna-o "perfeito para aplicações em tempo real com uso intenso de dados executados sobre dispositivos distribuídos" (NODEJS, 2015a, tradução nossa). A execução em uma única *thread* faz com que o "consumo de memória alocado por requisição seja menor" (SCHROEDER; DOS SANTOS, 2014, p. 3-4). Além disso, a utilização de uma *thread* e I/O não bloqueante (ou assíncrono) faz com que "muitas requisições possam ser atendidas simultaneamente" (SCHROEDER; DOS SANTOS, 2014, p. 5).

Dayley (2014, p. 55, tradução nossa) explica que "em vez de executar todas as tarefas de cada requisição sob threads individuais, o Node.js adiciona as tarefas para uma fila de eventos e em seguida há uma simples *thread* executando um ciclo de eventos". Segundo Ihrig (2013), em cada iteração do ciclo de eventos um evento é retirado da fila e processado. Segundo Teixeira (2011), os eventos são executados um por vez e suas respectivas funções de *callback*[20] são executadas. Vale ressaltar que estas funções de *callback* são um novo evento, portanto "são simplesmente adicionadas para o fim da fila" (IHRIG, 2013, p. 29, tradução nossa) para posterior processamento. Junior (2012) destaca que pela característica assíncrona, não é possível garantir a ordem de execução dos eventos.

A figura 5 representa a execução das mesmas tarefas retratadas anteriormente na Figura 1 deste documento, porém sobre o modelo de eventos do Node.js. Cada função executada é um evento, o qual chama uma nova função – *callback* – que é colocada no fim da fila. Este procedimento é executado até não existirem eventos na fila de eventos.

[20] Um *callback* é "uma função que é executada quando alguma coisa significante acontece" (TEIXEIRA, 2011, p. 8, tradução nossa).

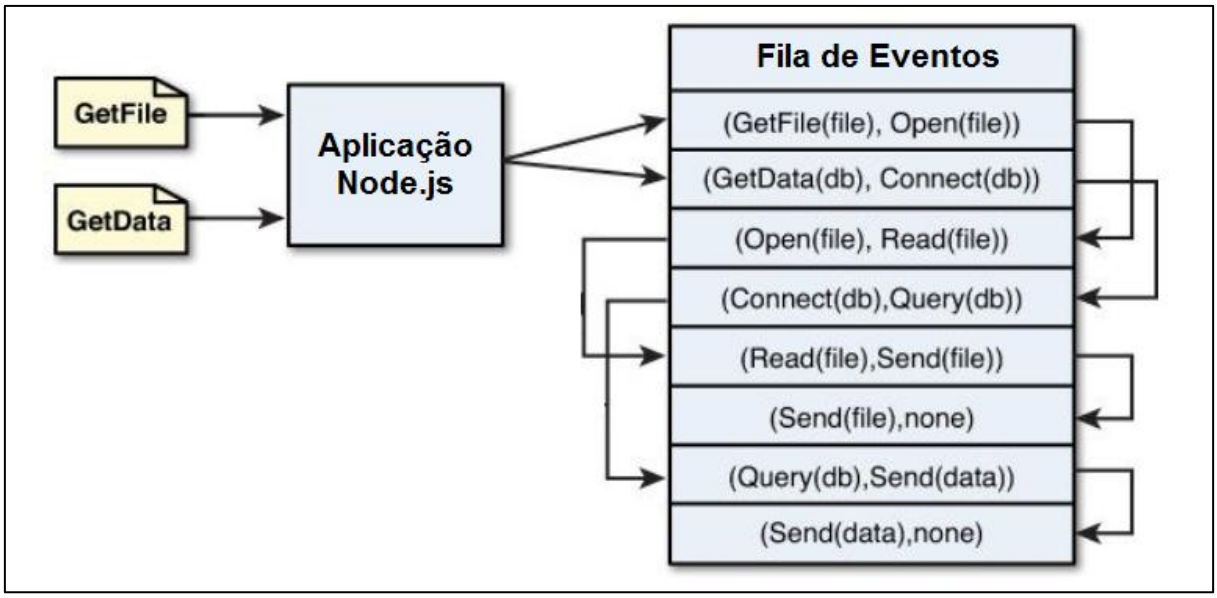

Figura 5 – Manipulação de requisições em servidores *web* orientados a eventos
Fonte: Adaptado de Dayley (2014, p. 56).

Dayley (2014) afirma que o Node.js é uma plataforma modular, portando muitas funcionalidades "são fornecidas por módulos externos em vez de serem incorporadas à plataforma" (DAYLEY, 2014, p. 39, tradução nossa). Os módulos são gerenciados por uma ferramenta de linha de comando chamada npm – Node Package Manager – que é instalada junto com o Node.js (IHRIG, 2013) (TEIXEIRA, 2011) e que utiliza um arquivo chamado package.json para registrar os módulos instalados e outras configurações do projeto (DAYLEY, 2014).

O desenvolvimento de aplicações no Node.js é feito a partir da linguagem Javascript e para utilizar um módulo existente é necessário utilizar a função 'require()' dentro do código da aplicação desenvolvida (TEIXEIRA, 2011), a qual recebe um *string* como argumento – nome do módulo a ser carregado – e retorna um objeto que permite acessar as funcionalidades do mesmo (IHRIG, 2013).

Alguns módulos fazem parte do núcleo do Node.js, portanto já vêm com o programa, outros são desenvolvidos e publicados por terceiros. Para obter um módulo de terceiro é necessário utilizar o npm e efetuar o *download* do mesmo. Segundo Dayley (2014), existem módulos para diversos propósitos, pois segundo o autor, "a cultura Node.js é muito ativa na criação e publicação de módulos para quase todas as necessidades imagináveis" (DAYLEY, 2014, p. 39, tradução nossa). Dentre os módulos utilizados no desenvolvimento do servidor, dois módulos de terceiros merecem destaque: Express e Mongoose.

2.9.2. Express

O Express é um *framework* criado por TJ Holowaychuk (IHRIG, 2013) e é definido como “um leve e flexível *framework* de aplicações web que disponibiliza um robusto conjunto de recursos para aplicações web e mobile” (EXPRESS..., 2015, tradução nossa). Segundo Cross et al. (2013b, p. 5), é o *framework* mais utilizado no ambiente Node.js para desenvolver aplicações web já que é possível construir simples aplicações com o padrão *Model-View-Controller* (MVC) em pouco tempo. Yaapa (2013) afirma que com o Express é possível criar desde aplicativos simples até os mais complexos, pois o Express faz uma abstração de outros módulos – *http* e *connect* – e disponibiliza uma interface para a fácil utilização dos mesmos (DAYLEY, 2014).

O módulo *http* já vem compilado (ou instalado) com o Node.js e é responsável por suportar as características do protocolo HTTP, portanto é uma API de baixo nível (NODEJS, 2015b). Segundo Dayley (2014), o módulo *http* disponibiliza um *framework* que permite a construção de um servidor HTTP básico. Segundo Nodejs (2015b), para suportar o desenvolvimento da maioria das aplicações HTTP possíveis o módulo *http* “trata-se apenas de manipulação de fluxo e análise de mensagem. Ele analisa uma mensagem em cabeçalhos e corpo, mas não analisa os cabeçalhos reais ou corpo” (NODEJS, 2015b, tradução nossa). Por este motivo o módulo *http* não suporta manipulação de rotas, *cache* e *cookies* (DAYLEY, 2014, p. 119).

Para suportar outros recursos no servidor *web* e facilitar a manipulação das requisições são utilizadas funções de *middleware*. Estas funções “são executadas entre o ponto onde a requisição é recebida pelo Node.js e o ponto onde a resposta é enviada” (DAYLEY, 2014, p. 381, tradução nossa) e possibilitam a reutilização do código (IHRIG, 2013). Um *middleware* “pode processar a requisição completamente ou desempenhar uma operação sobre a requisição e, em seguida, passá-la para outro pedaço de *middleware* para processamento adicional” (IHRIG, 2013, p. 176, tradução nossa), e é neste contexto que o módulo *connect* é utilizado, já que este “fornece uma estrutura de middleware que permite facilmente inserir funções

middleware em um nível global, de caminhos ou para uma simples rota [[21]]" (IHRIG, 2013, p. 177, tradução nossa) dentro das aplicações desenvolvidas.

Segundo Yaapa (2013, p. 7, tradução nossa), os *middlewares* do módulo *connect* "cuidam de muitas das funcionalidades comumente requeridas em aplicações *web* para o Node". Além disso, Yaapa (2013) destaca que qualquer pessoa pode escrever suas próprias funções de *middleware* e incluí-las na aplicação. Dayley (2014, p. 381, tradução nossa) afirma que os *middlewares* suportados pelo Express permitem "rapidamente servir arquivos estáticos, implementar *cookies*, suportar seções, processar dados POST, e muito mais".

A facilidade obtida com a utilização do *framework* Express é um dos principais motivos da sua utilização. A possibilidade de rápida criação de servidores *web* com vários recursos potenciais e poucas linhas de código aumenta o potencial do *framework*, que combinado com o modelo orientado a eventos do Node.js permite criar uma estrutura de alto desempenho.

2.9.3. Mongoose

O Mongoose é uma biblioteca "que fornece funcionalidades adicionais ao *driver* nativo do Node.js para MongoDB" (DAYLEY, 2014, p. 297, tradução nossa). Com ele é possível acessar as bases de dados do MongoDB – que será explicado na sequência – e desempenhar operações de busca, remoção, alteração, gravação, agregação e outras de uma forma simplificada (DAYLEY, 2014). Segundo Cross et al. (2013b, p. 7, tradução nossa), o "Mongoose fornece uma abordagem declarativa familiar para a modelagem de dados". Para Dayley o Mongoose é utilizado, principalmente, "para aplicar um esquema estruturado em uma coleção do MongoDB, o qual fornece os benefícios de validação e de conversão de tipos" (DAYLEY, 2014, p. 297, tradução nossa).

Para a criação de um esquema estruturado, o Mongoose utiliza um objeto denominado *schema* para estruturar os dados a serem armazenados (IHRIG, 2013), no qual são definidos os campos e o tipo do dado armazenado[22], seus índices e a validação dos dados (DAYLEY, 2014). Após a definição do *schema* é necessário

[21] Segundo Ihrig (2013), uma rota é a combinação entre os verbos HTTP (GET, POST) e a URL.
[22] Segundo Dayley (2014, p. 300, tradução nossa), os tipos de dados podem ser "String, Number, Boolean ou Bool, Array, Buffer, Date, ObjectId ou Oid, Mixed".

associá-lo a uma conexão com a base de dados, a qual é definida como *Model* (IHRIG, 2013). Segundo Dayley (2014, p. 297, tradução nossa), “o objeto Model atua como uma representação de todos os documentos em uma coleção”.

A figura 6 apresenta um esquema de dados do Mongoose sobre logradouros, onde na primeira linha é requerido o módulo Mongoose, na segunda linha é solicitada a funcionalidade de criação de *schemas* do módulo requerido anteriormente, da terceira à décima primeira linha é feita as configurações do tipo de dado que será armazenado e na décima segunda linha é vinculada o *Schema* criado a um *Model* intitulado Endereço, dentro da tabela endereços.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var EnderecoSchema = new Schema({
    cep: {type: String, index: 1, required:true},
    uf: {type: String, uppercase:true, required:true},
    cidade: {type: String, uppercase:true, required:true},
    bairro: {type: String, uppercase:true, required:true},
    logradouro: {type: String, uppercase:true, required:true},
    inserido_em: {type: Date, default: Date.now, select: false},
    alterado_em: {type: Date, default: Date.now, select: false}
});
module.exports = mongoose.model('Endereco', EnderecoSchema, 'enderecos');
```

Figura 6 – Exemplo de um esquema estruturado do Mongoose
Fonte: Autoria própria.

Quando um *Model* retorna os documentos da base de dados, cada documento retornado é passado à função de *callback* como um *Document* do Mongoose (DAYLEY, 2014). Segundo Dayley (2014), o objeto *Document* representa um documento individual de uma coleção do MongoDB que “permite-lhe interagir com um documento a partir da perspectiva do seu modelo de esquema, fornecendo um número de métodos e propriedades extras que suportam validação, modificações, etc” (DAYLEY, 2014, p. 310, tradução nossa).

Dayley (2014) destaca que dentre os benefícios da utilização do Mongoose está à possibilidade de criação de um esquema estruturado[23] para os documentos, pode-se realizar validação e conversão dos dados dentro de um objeto, é mais fácil utilizá-lo do que usar apenas o *driver* nativo do MongoDB para Node.js e possibilita a utilização de *middleware* sobre os dados antes da realização das operações no banco de dados. Entretanto Dayley (2014) alerta sobre algumas desvantagens na utilização do Mongoose, pois segundo o autor algumas operações não são tão

[23] Ihrig (2013, p. 226, tradução nossa) afirma que o “MongoDB não tem um esquema predefinido”, ou seja, estruturado.

eficientes quanto àquelas realizadas pelo *driver* nativo e a obrigatoriedade em definir um esquema pode não ser interessante quando não é necessário utilizar um esquema estruturado no armazenamento dos dados.

A utilização do Mongoose neste projeto se deve à estruturação dos dados armazenados, visto que a estrutura do projeto conta com um esquema estruturado. Outro aspecto importante colocado por Ihrig (2013) é que a base de dados NoSQL mais utilizada com a plataforma Node.js é o MongoDB, portanto trabalhar com uma biblioteca que padroniza as tarefas entre as duas plataformas – Node.js e MongoDB – significa trabalhar com uma estrutura bem definida, de fácil manutenção e que resulta em um projeto bem organizado.

2.10. BANCO DE DADOS

Dayley (2014, p. 195, tradução nossa) cita que “boas aplicações web precisam ser capazes de armazenar e obter dados com precisão, velocidade e confiabilidade” e afirma que na maioria das aplicações ou serviços *web* há uma solução de armazenamento de dados de alto desempenho, responsável por atender a demanda dos usuários (DAYLEY, 2014).

Segundo Heuser (2001), uma solução de armazenamento de dados – ou banco/base de dados – é responsável por agrupar e armazenar um conjunto de dados e também atender a demanda dos usuários ou serviços sobre eles, ou seja, fornecer o acesso aos dados. Ihrig destaca que as bases de dados fornecem um rápido acesso a um grande volume de dados e que “geralmente existem dois tipos de bases de dados – bases de dados relacionais e bases de dados NoSQL” (IHRIG, 2013, p. 217, tradução nossa). Ambas são discutidas a seguir.

2.10.1. Bases de dados relacionais

Uma base de dados construída sobre o modelo relacional é composta por tabelas ou relações (HEUSER, 2001, p. 87) que pressupõe que os dados estão

estruturados[24] e por isso armazena-os em linhas e colunas de uma tabela com campos ou atributos semelhantes (PARKER et al., 2013, p. 1). Para consultar e manipular os dados armazenados é utilizado uma linguagem que fornece “uma maneira simples de coletar estes dados” (POLITOWSKI; MARAN, 2014, p. 1), denominada SQL – *Structured Query Language* (NAYAK et al, 2013) – a qual é “uma linguagem padrão de definição e manipulação do banco de dados” (HEUSER, 2001, p. 85).

Dentro das bases de dados de modelo relacional, um processo conhecido como normalização é responsável por armazenar os dados repetidos ou redundantes em uma nova tabela da base de dados (IHRIG, 2013, p. 218). Segundo Parker et al. (2013, p. 1), este processo eleva o número de tabelas da base de dados e exige que durante uma consulta vários dados sejam buscados e combinados para exibir o resultado esperado, aumentando o tempo gasto pela base de dados para processar os resultados. Além disso, Nance et al. (2013, p. 113, tradução nossa) destacam o aumento da complexidade de código e um possível atraso no desenvolvimento da aplicação quando utiliza-se uma base de dados de modelo relacional, pois neste modelo os dados “precisam ser transformados de um lado para outro entre a base de dados e os objetos da linguagem de programação”.

Segundo Politowski e Maran (2014, p. 1), “Bancos de dados Relacionais são os mais utilizados para armazenamento de dados [,] a [sic] décadas nos permitindo gravar grandes porções de dados no disco”, porém Nayak et al. (2013) afirmam que um dos maiores problemas do modelo relacional está na quantidade de dados armazenados, pois o desempenho do banco de dados é reduzido quando cresce o volume de dados armazenados. Segundo Nance et al. (2013, p. 111, tradução nossa), “bancos de dados relacionais não são adequados para aplicações *web* modernas que podem suportar milhões de usuários simultâneos**”**, pois:

> Bancos de Dados Relacionais foram projetados para serem executados em uma máquina apenas, onde para escalar, é necessário comprar uma máquina melhor (ou fazer um upgrade) com mais recursos, necessários para lidar com o alto tráfego de dados na web (big data). (POLITOWSKI; MARAN, 2014, p. 3)

Diante desta afirmação, Nance et al. (2013, p. 111, tradução nossa) explicam que “tentar melhorar a escalabilidade em bases de dados relacionais

[24] Segundo Parker et al. (2013, p. 1, tradução nossa), os “objetos na base de dados com o mesmo número de características, tipo e formato são agrupados juntos, tornando-se dados estruturados”.

significa adicionar servidores maiores. Servidores maiores são altamente complexos, proprietários e desproporcionalmente caros". Diante dos problemas enfrentados pelas bases de dados relacionais, uma nova solução de armazenamento de dados é necessária para não comprometer o funcionamento das aplicações *web* que exigem uma resposta rápida do servidor.

2.10.2. Bases de dados NoSQL

O termo NoSQL é abreviação de *Not Only SQL* (DAYLEY, 2014) – Não só SQL em tradução livre – e "consiste de tecnologias que fornecem armazenamento e recuperação [de dados] sem os modelos fortemente restritos de tradicionais bases de dados SQL relacionais" (DAYLEY, 2014, p. 195, tradução nossa). Segundo Nance et al. (2013, p. 111, tradução nossa), "o objetivo do NoSQL não é rejeitar o [sic] SQL, mas ser utilizado como um modelo de dados alternativo para aplicações que não funcionam bem com o modelo de base de dados relacional". Além disso, Politowski e Maran (2014, p. 9) afirmam que os bancos de dados NoSQL foram desenvolvidos para obter um desempenho maior devido ao que era fornecido pelos bancos de dados relacionais.

Uma base de dados NoSQL não utiliza nem possui uma linguagem padrão de consulta – como a SQL dos modelos relacionais – pois cada fornecedor de base de dados NoSQL cria uma linguagem de consulta exclusiva à plataforma (NAYAK et al., 2014). Ihrig (2013, p. 225, tradução nossa) afirma que "a única coisa que bases de dados NoSQL têm em comum é que elas abandonam o modelo de dados relacional", o que segundo Dayley (2014, p. 195) permite desenvolver um modelo de dados[25] que se aproxima das necessidades da aplicação, possibilitando maior controle dos dados e simplificação do projeto, o que nunca poderia ser feito com bases de dados relacionais.

Segundo Parker et al. (2013, p. 1, tradução nossa), os dados armazenados e manipulados não são estruturados e por isso "podem ser de qualquer tipo e ter qualquer formato". Os dados não estruturados "podem ser documentos de texto, e-mails, áudio, vídeo e até dados das redes sociais" (NANCE et al., 2013, p. 112, tradução nossa) e contam com várias técnicas de modelagem, conforme a base de

[25] Modelo de dados é a "descrição formal da estrutura de um banco de dados" (HEUSER, 2001, p. 5).

dados utilizada (NANCE et al., 2013). Ihrig (2013, p. 225, tradução nossa) afirma que “existem muitos tipos de bases de dados NoSQL disponíveis”, cada uma com um modelo diferente de armazenamento dos dados – chave/valor, orientado a documentos, orientado a grafos (NANCE et al., 2013) – onde “cada um tem seus próprios benefícios e fraquezas” (NANCE et al., 2013, p. 113).

Nance et al. (2013) afirmam que as bases de dados NoSQL têm sido fortemente utilizadas em cenários de *web sites* onde os recursos dos bancos de dados relacionais não são importantes e o desempenho obtido com o NoSQL é mais interessante para a aplicação. Além disso, os autores destacam que em aplicações com “grandes volumes de transações, necessidade de baixa latência no acesso a grandes volumes de dados e necessidade de disponibilidade quase perfeita enquanto opera em um ambiente não confiável” (NANCE et al., 2013, p. 111, tradução nossa), a utilização da tecnologia NoSQL é fundamental. Schroeder e dos Santos também destacam alguns pontos positivos da tecnologia NoSQL:

> Grande parte destes bancos são projetos “open-source” e que já nasceram com foco na escalabilidade, tendo como foco a alta performance, bem como capacidade básica a de rodar em cluster, garantindo mais facilmente a alta disponibilidade quando comparado com banco de dados relacionais. (SCHROEDER; DOS SANTOS, 2014, p. 3)

Apesar dos inúmeros benefícios desta tecnologia, Nance et al. (2013, p. 112) afirmam que estas bases de dados não possuem suporte para encriptação de arquivos de dados, falta criptografia na comunicação com o cliente, existe fraca autenticação entre cliente e servidor e possui vulnerabilidade a ataques de negação de serviço e injeção de SQL. Diante do que foi apresentado, conclui-se que o desenvolvedor precisa escolher qual tecnologia utilizar, visto que cada uma tem seus benefícios e malefícios.

2.10.3. Comparação entre modelo relacional e NoSQL

Diante das diferenças entre o modelo relacional e o NoSQL, o projetista de bancos de dados deve escolher qual modelo se adapta às suas necessidades (PARKER et al., 2013), pois “o mecanismo de armazenamento de dados (...) deve ser capaz de executar em um nível satisfatório para a demanda dos usuários” (DAYLEY, 2014, p. 195, tradução nossa). Segundo Politowski e Maran (2014,p. 9),

“Bancos de dados NoSQL e Relacionais utilizam paradigmas diferentes e, por sua vez, possuem finalidades diferentes, mas com o mesmo propósito: persistir dados”.

Segundo Parker et al. (2013, p. 1, tradução nossa), “se a base de dados não é estruturada e extremamente grande, então uma base de dados NoSQL é uma boa escolha”. Nance et al. (2013, p. 112) afirmam que bases de dados NoSQL possuem uma boa escalabilidade independente do hardware utilizado, o que não acontece com algumas bases de dados relacionais. Segundo Politowski e Maran (2014), os bancos de dados relacionais foram desenvolvidos para executar em apenas uma máquina e integram várias aplicações em uma única base de dados, fornecendo um controle de concorrência que “assegura a consistência dos dados e os mantêm sempre atualizados” (POLITOWSKI; MARAN, 2014, p. 2).

As bases de dados NoSQL não possuem uma linguagem padrão entre seus diferentes modelos e as operações desempenhadas são limitadas, tornando complexo o desenvolvimento de consultas nas bases de dados (PARKER et al., 2013, p. 1). Nayak et al. (2013, p. 19) afirmam que o modelo NoSQL ainda é imaturo, não tem interface e linguagem de consulta padronizadas e possui manutenção difícil, portanto ainda possui desvantagens sobre o modelo relacional.

Segundo Nayak et al. (2013, p. 19), as bases de dados NoSQL estão atrás das bases de dados relacionais somente no número de usuários, pois como não há uma linguagem padrão entre as bases NoSQL os usuários não estão familiarizados com esta tecnologia e ainda preferem o padrão estabelecido pelo SQL. Segundo Nance et al. (2013, p. 113-114), as bases de dados relacionais estão a 40 anos sendo utilizadas, portanto é mais fácil encontrar desenvolvedores para esse modelo de banco de dados, ao contrário das bases de dados NoSQL.

Apesar das dificuldades, as bases de dados NoSQL se sobressaem perante as bases de dados do modelo relacional pois “fornecem uma variedade de modelo de dados para escolha, são facilmente escaláveis, não requerem administradores de bases de dados, são rápidas, mais eficientes e flexíveis” (NAYAK et al., 2013, p. 19, tradução nossa), por este motivo Nance et al. afirmam que “aplicações grandes, públicas e centradas em conteúdo tendem a ser melhor servidas por bancos de dados NoSQL” (NANCE et al., 2013, p. 115, tradução nossa).

Para escolher entre bancos de dados relacionais e NoSQL é necessário analisar o tipo de aplicação a ser desenvolvida, a viabilidade e regularidade da estrutura dos dados e também o tipo de consulta desejada, bem como o custo

envolvido e o desempenho desejado (NANCE et al., 2013, p. 112). Para a realização deste projeto, optou-se por utilizar um banco de dados NoSQL orientado à documentos, portanto o MongoDB será responsável por armazenar as informações da aplicação.

2.10.4. MongoDB

MongoDB é um banco de dados NoSQL orientado a documentos[26] (IHRIG, 2013, p. 225) desenvolvido pela empresa 10gen[27] e lançado no ano de 2009 (NAYAK et al., 2013, p. 17). É uma base de dados que fornece alto desempenho, escalabilidade automática e alta disponibilidade, além de ser simples de instalar e manipular (DAYLEY, 2014, p. 3). Foi construída utilizando a linguagem de programação C++ (NAYAK et al., 2013, p. 17) e armazena seus dados no formato BSON[28], que é um JSON em formato binário (IHRIG, 2013, p. 225), porém superior a este quando comparado ao espaço de armazenamento e a velocidade de busca (NAYAK et al., 2013, p. 17).

Os dados armazenados no MongoDB são agrupados dentro de uma coleção – *Collection* – que equivale às tabelas de bases de dados relacionais, porém sem um esquema fixo (DAYLEY, 2014, p. 196). Dentro das coleções existem os documentos – *Documents* – que são a "representação de uma simples entidade de dados em uma base de dados MongoDB" (DAYLEY, 2014, p. 196, tradução nossa) e que equivalem às linhas de uma tabela das bases de dados relacionais, porém podem ter qualquer campo ou valor sem a existência de restrições para armazenar os dados (DAYLEY, 2014, p. 196).

Cross et al. (2013a, p. 6, tradução nossa) afirmam que a habilidade de armazenar os dados no formato de objetos JSON economiza processamento e tempo, além de ser "uma melhor abordagem do que reduzir a estrutura semi-complexa do JSON em múltiplas relações em uma base de dados relacional". A

[26] Bases de dados orientadas a documentos armazenam seus dados no formato de documentos, os quais são semelhantes aos registros das bases relacionais, porém mais flexíveis já que não seguem uma estrutura (NAYAK et al., 2013).
[27] (MONGODB, 2015)
[28] O BSON "suporta estruturas como arrays e embedded objects assim como JSON" (POLITOWSKI; MARAN, 2014, p. 3) e "contém uma lista ordenada de elementos que consistem do nome do campo, tipo e valor" (NAYAK et al., 2013, p. 17).

figura 7 apresenta um exemplo da estrutura de um documento do MongoDB com os campos ‘name’, ‘version’, ‘languages’, ‘admin’ e ‘paths’.

```
{
        name: "New Project",
        version: 1,
        languages: ["JavaScript", "HTML", "CSS"],
        admin: {name: "Brad", password: "****"},
        paths: {"/tmp", project:"/opt/project", html: "/opt/project/html"}
}
```

Figura 7 - Estrutura de um documento do MongoDB
Fonte: (DAYLEY, 2014, p. 196)

Para manipular os dados armazenados o MongoDB utiliza uma interface Javascript (CROSS et al., 2013ª, p. 5) com comandos de manipulação construídos utilizando a sintaxe de um objeto JSON, enquanto que bancos de dados relacionais utilizam a sintaxe SQL (PARKER et al., 2013, p. 3). Segundo Politowski e Maran (2014, p. 3), o “MongoDB permite [que] usuários realizem modificações de apenas um atributo em um documento sem a necessidade de interação com o restante da estrutura”.

O MongoDB “procura manter a maioria dos dados na memória para que simples consultas tomem menos tempo, eliminando a necessidade de recuperar os dados do disco rígido” (PARKER et al., 2013, p. 2, tradução nossa), porém quando a quantidade de dados é maior que a quantidade de memória RAM disponível, o MongoDB passa a recuperar os dados do disco rígido (PARKER et al., 2013, p. 2). Apesar do aumento de desempenho, “existem desvantagens com o MongoDB, como seu fraco desempenho com funções agregadas e consultas baseadas em valores não-chave” (PARKER et al., 2013, p. 6, tradução nossa).

Segundo Parker et al. (2013, p. 6), as decisões tomadas durante o desenvolvimento afetam o desempenho do MongoDB e este requer maior esforço de implementação quando comparado às bases de dados relacionais. Além disso, Nayak et al. (2013, p. 17) afirmam que a indexação[29] utiliza muita memória RAM e que a base de dados pode não ser confiável, já que o

> MongoDB não possui controle de concorrência e gerenciamento de transações. Então, se um usuário lê um documento, escreve uma modificação e devolve ao banco de dados, pode acontecer de outro usuário escrever uma nova versão do documento entre o processo de leitura e escrita do primeiro. (POLITOWSKI; MARAN, 2014, p. 4)

[29] Indexação é a inserção de índices nos campos de uma tabela ou coleção, o qual aumenta a velocidade de busca nos dados armazenados.

Nance et al. (2013, p. 115, tradução nossa) afirmam que “toda empresa tem uma necessidade diferente de como planejam utilizar uma base de dados e as necessidades dependem do que uma empresa quer fora da sua base de dados”. No estudo comparativo entre a base de dados MongoDB e bases de dados SQL realizado por Parker et al., os autores concluíram que:

> MongoDB é definitivamente a escolha para usuários que precisam de uma estrutura de base de dados menos rígida. MongoDB pode ser uma boa solução para grandes conjuntos de dados no qual o esquema está constantemente mudando ou no caso de as consultas realizadas serem menos complexas. Para aqueles usuários que tem um esquema rigoroso e uma modesta quantidade de dados estruturados, nós também encontramos que o MongoDB desempenha melhor que o SQL, em geral. (PARKER et al., 2013, p. 6, tradução nossa)

Nayak et al. (2013, p. 17, tradução nossa) afirmam que o “MongoDB é bem adequado para aplicações como sistemas de gerenciamento de conteúdo, arquivamento, análises em tempo real, etc”. Segundo Cross et al. (2013ª, p. 5), o banco de dados MongoDB foi projetado para facilitar o desenvolvimento e a escalabilidade e já é utilizado por grandes corporações, como eBay, Foursquare, MTV Networks e The Guardian (IHRIG, 2013, p. 225) (NAYAK et al., 2013, p. 17). Dayley (2014, p. 195, tradução nossa) afirma que o “MongoDB é, de longe, o banco de dados NoSQL mais popular e bem apoiado atualmente disponível”.

No teste de desempenho realizado por Politowski e Maran (2014, p. 9), os autores concluíram que “para uma aplicação com alta carga de consultas à base de dados, como serviços web, por exemplo, o banco MongoDB é uma ótima alternativa”, sendo este o principal motivo da escolha desta base de dados para este projeto. No cenário de um aplicativo móvel, o número de consultas à base de dados é elevado e a mesma deve ser rápida e suportar toda a carga depositada sobre a mesma.

2.11. INTEGRAÇÃO DAS TECNOLOGIAS

Algumas tecnologias utilizadas neste projeto fazem parte de uma popular combinação – ou pilha – de tecnologias conhecida pelo acrônimo MEAN, que se refere à MongoDB, Express, AngularJS e Node.js (IHRIG, 2013). Dentro dela o MongoDB é o banco de dados responsável pelo armazenamento dos dados, o

Express pelo servidor *web*, o AngularJS controla a obtenção dos dados no lado do cliente e o Node.js fornece a plataforma de desenvolvimento (DAYLEY, 2014). A lista abaixo apresenta a tecnologia utilizada e a função que a mesma desempenhou no projeto:

- **NodeJS (NODEJS, 2015a):** Plataforma de desenvolvimento onde os *scripts* desenvolvidos foram executados;
- **Express (EXPRESS, 2015):** Módulo do Node.js responsável pelo gerenciamento do servidor *web* e das rotas utilizadas para acessá-lo;
- **Mongoose (MONGOOSE, 2015):** Módulo do Node.js responsável pela estruturação dos dados e acesso ao banco de dados MongoDB para manipulação dos mesmos;
- **MongoDB (MONGODB, 2015):** Base de dados responsável pelo armazenamento dos dados da aplicação;
- **AngularJS (ANGULARJS, 2015):** Responsável pela estruturação do código do aplicativo, requisitar dados do servidor *web* e manipulação das ações da interface do usuário;
- **Ionic *Framework* (IONIC, 2015a):** Responsável pela construção da interface do usuário e padronização do visual da aplicação;
- **Apache Cordova (CORDOVA, 2015a):** Responsável por permitir o acesso às funcionalidades nativas dos dispositivos por meio da API Javascript disponibilizada, empacotar todos os arquivos de mídia, HTML, CSS e Javascript desenvolvidos e por construir um arquivo binário que pode ser enviado à loja de aplicativos das plataformas.

Para complementar o entendimento das tecnologias citadas, a figura 8 apresenta de forma visual os itens da lista anterior:

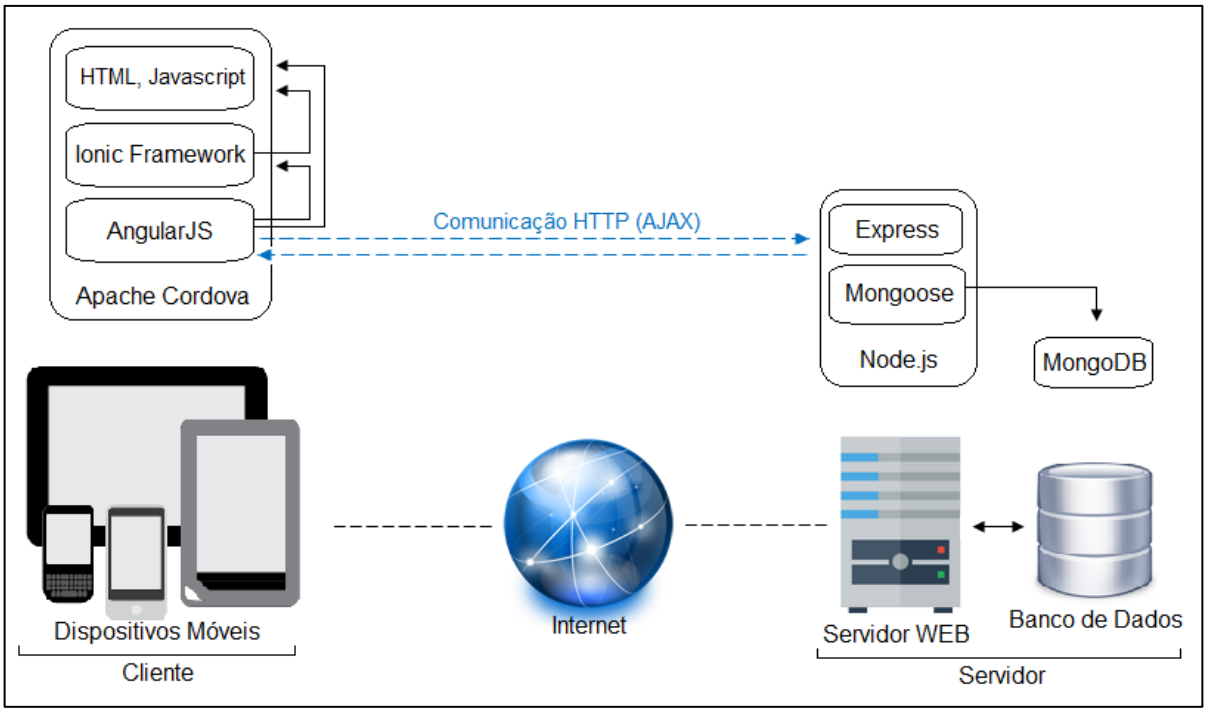

Figura 8 – Resumo visual das tecnologias utilizadas no projeto
Fonte: Autoria própria.

Em um cenário de busca por um produto utilizando o aplicativo, o usuário acessa a aplicação que está instalada no seu dispositivo móvel e interage com as interfaces de usuário geradas pelo Ionic, AngularJS e os arquivos HTML que encontram-se dentro do arquivo binário gerado pelo Apache Cordova. Quando o usuário solicita os resultados, o AngularJS efetua uma requisição HTTP que será manipulada no servidor pelo Express, o qual solicita ao Mongoose os dados armazenados no MongoDB e retorna para o usuário as informações solicitadas. Após receber a resposta o AngularJS a armazena na sua estrutura e o Ionic exibe para o usuário os dados formatados e estilizados, permitindo novamente a interação do usuário.

3 METODOLOGIA

Neste capítulo são apresentadas outras soluções de *software* com propósitos similares, as limitações e a metodologia utilizada para o desenvolvimento deste projeto.

3.1. ANÁLISE DE SISTEMAS SIMILARES

Foi efetuada uma pesquisa na loja virtual de aplicativos da Google, a Google Play[30], na tentativa de encontrar outras soluções de *software* que possuíssem funcionalidades similares às buscadas neste projeto. A pesquisa retornou alguns aplicativos com funcionalidades semelhantes, porém estas não se encontravam no mesmo aplicativo, portanto a ideia de apresentar uma solução integrada foi fortalecida.

O aplicativo Restaurantes[31] efetua a busca por estabelecimentos a partir do nome do estabelecimento ou pela localização do dispositivo móvel, o qual exibe apenas os estabelecimentos mais próximos. Além disso, esta solução também permite visualizar os itens do cardápio, detalhes, avaliação e fotos do local. É uma solução simples e com uma interface de usuário simplificada.

O aplicativo Zomato - Restaurante & Bar[32] apresenta as mesmas funcionalidades que o aplicativo Restaurantes, mas também permite fazer *check-in* no local e efetuar uma ligação para o estabelecimento diretamente do aplicativo, além de contar com uma interface visualmente agradável.

O aplicativo e-Cardápios *Online*[33] possui algumas funcionalidades semelhantes aos aplicativos anteriores, porém não apresenta a funcionalidade de fazer *check-in* e efetuar uma ligação para o local, mas conta com as funcionalidades de fornecer cupons de desconto para seus usuários e de buscar por promoções contidas na plataforma, além da própria exibição de anúncios. A figura 9 apresenta a

[30] https://play.google.com/
[31] (RESTAURANTES..., 2015)
[32] (ZOMATO..., 2015)
[33] (ECARDAPIOS..., 2015)

tela de busca por estabelecimentos dos três aplicativos, conforme a ordem apresentada neste texto.

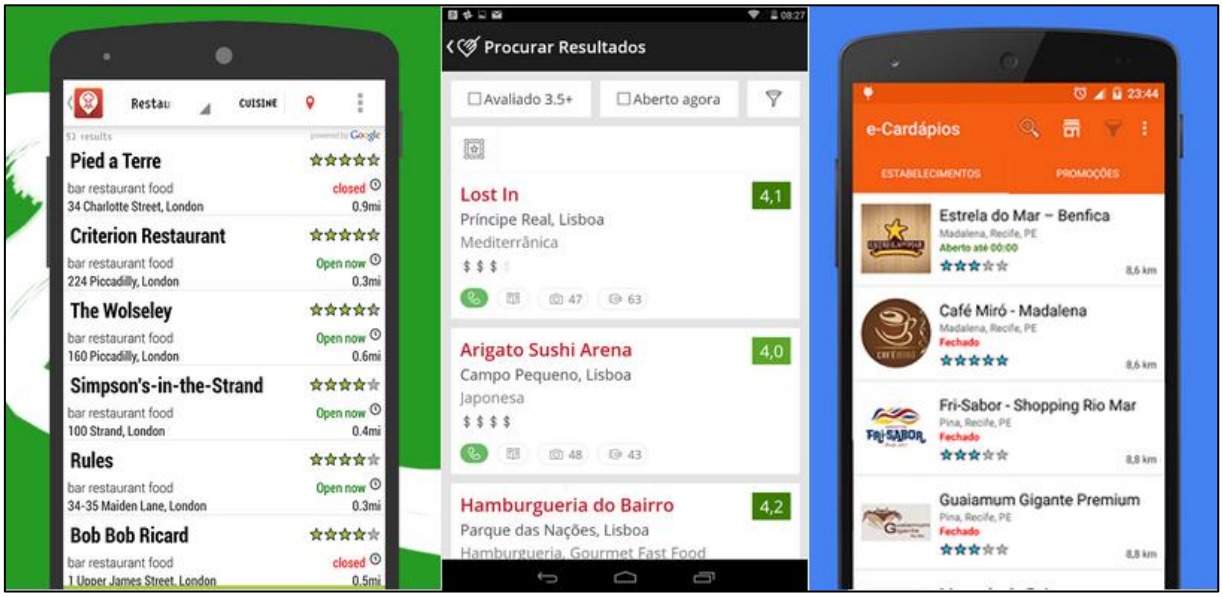

Figura 9 – Tela de busca de estabelecimentos
Fonte: Adaptado de (RESTAURANTES..., 2015), (ZOMATO..., 2015) e (ECARDAPIOS..., 2015).

Os aplicativos anteriores foram encontrados a partir da busca pelo termo “Restaurante” e “Cardápio”. Como nenhum resultado referente à conta ou comanda foi encontrado, uma nova busca com o termo “Comanda” foi realizada. A única solução encontrada que permitia ao cliente verificar o valor da comanda foi a do aplicativo Minha Comanda[34], porém este não possui comunicação com o estabelecimento e para o cliente verificar o valor gasto, o mesmo deveria inserir os dados na aplicação.

Diante do que foi encontrado nas pesquisas, concluiu-se que desenvolver uma solução que integre as diversas funcionalidades em apenas um aplicativo é útil para o usuário da aplicação, já que além de encontrar as informações do local, também é possível interagir com a plataforma do estabelecimento e consultar o valor gasto, não sendo necessário digitar as informações. As funcionalidades propostas tornam a utilização da aplicação mais fácil quando comparada com as já existentes, visto que se procurou a automatização da maioria das tarefas. Após a definição dos pontos de melhoria, algumas restrições ao projeto foram impostas.

[34] (MINHACOMANDA..., 2015)

3.2. RESTRIÇÕES DO PROJETO

Apesar de a ferramenta Apache Cordova permitir a geração de aplicativos para diversas plataformas móveis, este projeto é restrito à construção de aplicativos para a plataforma Android[35]. Esta decisão foi tomada devido à natureza deste sistema operacional móvel, o qual é de código fonte aberto (*open source*) e não exige uma licença de desenvolvedor para executar aplicativos na plataforma (KASPERBAUER, 2013). Além disso, o SDK do Android pode ser utilizado em qualquer sistema operacional tradicional – Windows, Mac OS X e distribuições do Linux – ao contrário do desenvolvimento para iOS (sistema operacional móvel da Apple), onde o SDK é executado somente em computadores Mac com Mac OS X (DE MENDONÇA; BITTAR; DE SOUZA DIAS, 2011).

Além de não ser necessária uma licença de desenvolvedor, a plataforma Android conta com algumas ferramentas gratuitas que facilitam o teste e o desenvolvimento de aplicativos. O SDK do Android possui um emulador – Android Virtual Device (AVD)[36] – que simula algumas capacidades dos dispositivos físicos e a empresa Genymobile também disponibiliza um emulador – Genymotion[37] – para a plataforma Android, o qual é mais rápido que o AVD e possui versões gratuitas e pagas.

Quanto aos objetivos deste projeto, a comunicação com o servidor para obter o valor da comanda somente é feita a partir de requisições HTTP, a qual retorna um arquivo JSON com as informações requisitadas. Esta restrição foi aplicada devido ao tempo disponível, portanto outros formatos de resposta – como o XML – não são suportados. A possibilidade de divisão do valor da comanda restringe-se ao número de parcelas selecionadas, não podendo ser escolhidas outras opções de divisão. Por fim, os itens selecionados na comanda eletrônica são exibidos somente para os usuários do estabelecimento, portanto nenhuma outra funcionalidade – métodos de pagamento, controle de estoque, por exemplo – foi adicionada na estrutura do projeto. Definidas as limitações do projeto, foi selecionado um método para gerenciar seu desenvolvimento.

---

[35] O Android foi projetado pela Google para ser executado em *smartphones* e *tablets* e "baseia-se em uma versão modificada do Linux" (KASPERBAUER, 2013, p. 87).
[36] http://developer.android.com/tools/devices/index.html
[37] https://www.genymotion.com/#!/product

3.3. METODOLOGIA DE DESENVOLVIMENTO

Durante a construção de um *software*, indica-se a utilização de um roteiro que permita gerenciar tópicos pertinentes ao seu desenvolvimento, como qualidade, controle, estabilidade, organização e cumprimento do prazo estabelecido para a entrega (PRESSMAN, 2011). Para gestão e planejamento deste projeto foi utilizado o *framework* de desenvolvimento ágil Scrum[38].

O Scrum "é uma metodologia de desenvolvimento ágil de software concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990" (PRESSMAN, 2011, p. 95) que utiliza "uma abordagem iterativa [[39]] e incremental [[40]] para entregar valor com frequência e, assim, reduzir os riscos do projeto" (SABBAGH, 2013, p. 17), portanto "fornece uma base e um caminho para entregar metas de negócios de uma maneira colaborativa, razoável e agradável" (CORE..., 2015, tradução nossa).

Segundo Sabbagh (2013, p. 17), o "Scrum é um framework Ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos" e não é utilizado apenas no desenvolvimento de *software* – apesar de ter sido concebido com este objetivo – pois muitas empresas de diversos segmentos[41] já desfrutam de seus benefícios (CORE..., 2015). Dentro do Scrum cada pessoa envolvida com o projeto possui um papel distinto, a fim de se responsabilizar e se comprometer com a execução e o resultado final do mesmo (SABBAGH, 2013).

Sabbagh (2013, p. 265) afirma que o "Scrum foi concebido para times pequenos, ou seja, que possuem entre 3 e 9 membros". Segundo (CORE..., 2015), o grupo que executa o Scrum é denominado Time de Scrum e é composto, principalmente, pelo *Product Owner* – responsável por definir o projeto e garantir retorno financeiro sobre o mesmo (SABBAGH, 2013) –, Scrum *Master* – responsável por ensinar e garantir a utilização do Scrum no Time de Scrum (SABBAGH, 2013) –

[38] (SABBAGH, 2013)
[39] O termo iterativo refere-se ao desenvolvimento em ciclos, os quais são repetidos ao longo de um processo para atingir um objetivo.
[40] O termo incremental refere-se ao desenvolvimento em partes, onde cada parte gera uma nova funcionalidade que é integrada sucessivamente a fim de construir o produto final.
[41] "Educação, marketing, operações e outros estão adotando o Scrum" (CORE..., 2015, tradução nossa).

e pela própria equipe de desenvolvimento – grupo de programadores responsáveis pela codificação do projeto (SABBAGH, 2013).

O time de Scrum constrói o produto de forma incremental, a partir de um curto período de tempo denominado *Sprint* (CORE..., 2015), o qual é um "ciclo de desenvolvimento de duração fixa" (SABBAGH, 2013, p. 273) executado várias vezes durante o projeto a fim de atingir uma meta. Segundo Sabbagh (2013, p. 36), os *Sprint* "acontecem um depois do outro, sem intervalos entre eles, ou seja, tudo em um projeto que utiliza Scrum acontece dentro dos Sprints".

Antes de realizar o *Sprint*, o *Product Owner* insere em uma lista denominada *Product Backlog* todas as funcionalidades que necessitam ser desenvolvidas pela equipe de desenvolvimento para atingir o objetivo final (SABBAGH, 2013). Após isso, em uma reunião denominada *Sprint Planning* o Time de Scrum seleciona alguns itens do *Product Backlog* e determina quais tarefas devem ser realizadas para atingir uma meta – a meta do *Sprint* – e então se inicia um *Sprint* que focará apenas em executar as tarefas definidas no *Sprint Planning* (SABBAGH, 2013). Cada *Sprint* tem uma duração fixa de tempo – uma semana, por exemplo – e conta com reuniões diárias denominadas *Daily* Scrum, onde o time de desenvolvimento expõe o trabalho já realizado e planeja as ações do próximo dia, sempre focando na meta do *Sprint* (SABBAGH, 2013).

Após o término do *Sprint* são realizadas reuniões de *Sprint Review* – onde é apresentado o resultado do *Sprint* para as partes interessadas[42] e realizado adaptações na mesma, se necessário – e a reunião de *Sprint Retrospective* – onde o time analisa seus processos de trabalho e planeja melhorias a serem executadas nos próximos *Sprint* (SABBAGH, 2013). Após essas reuniões, um Incremento de Produto é gerado, o qual "representa valor visível para os clientes do projeto" (SABBAGH, 2013, p. 46) e nada mais é que "um incremento de funcionalidades do produto prontas" (SABBAGH, 2013, p. 272). A figura 10 apresenta o ciclo completo do Scrum.

[42] "Além dos clientes, as partes interessados no projeto são os usuários, patrocinadores e quaisquer pessoas que tenham algum tipo de influência nas definições do projeto ou tenham interesse direto no seu andamento e sucesso" (SABBAGH, 2013, p. 82).

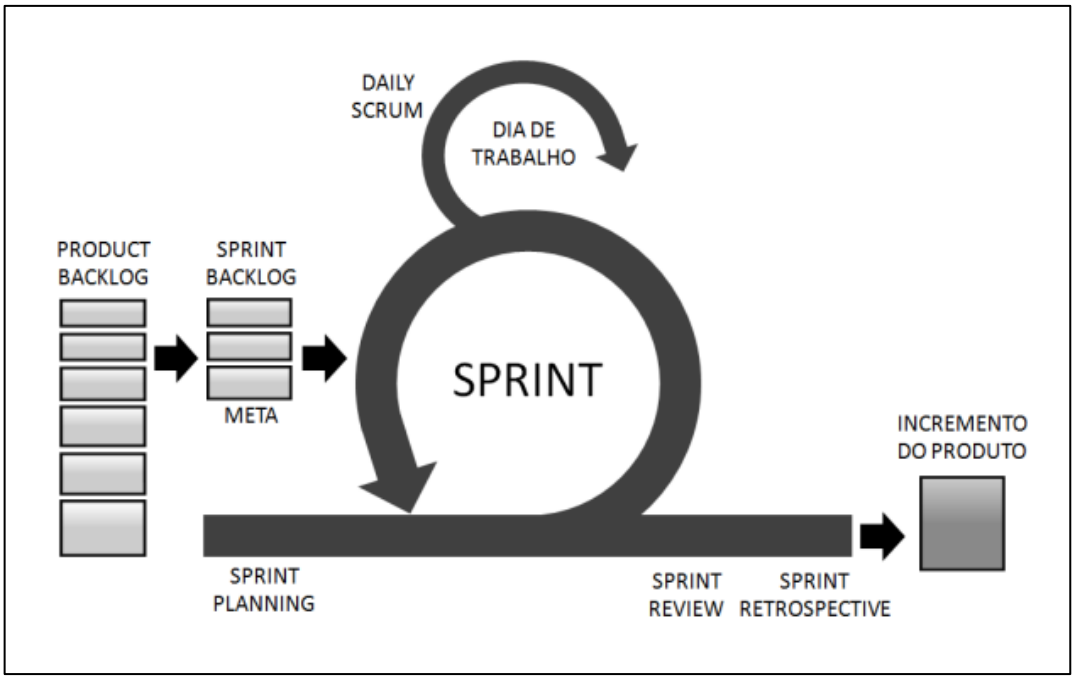

Figura 10 – Ciclo do Scrum
Fonte: (SCHWABER; BEEDLE, 2002; SCHWABER,2004 apud. SABBAGH, 2013, p. 42).

Segundo Pressman (2011, p. 95), “O Scrum enfatiza o uso de um conjunto de padrões de processos de software que provaram ser eficazes para projetos com prazos de entrega apertados, requisitos mutáveis e críticos de negócio“. Sabbagh (2013, p. 3) afirma que o Scrum é “a forma mais comum de se trabalhar em projetos de desenvolvimento de software”, pois segundo o autor a sua utilização permite:

> Reduzir os riscos de insucesso, entregar valor mais rápido e desde cedo, e lidar com as inevitáveis mudanças de escopo, transformando-as em vantagem competitiva. Seu uso pode também aumentar a qualidade do produto entregue e melhorar a produtividade das equipes. (SABBAGH, 2013, p. 3)

Segundo (CORE..., 2015) não é necessário modificar o *framework* para desenvolver alguns produtos, mas algumas restrições deste projeto – o tamanho da equipe, o prazo de desenvolvimento e os requisitos incertos – forçaram a adaptação do Scrum, portanto algumas funcionalidades não foram utilizadas. O processo de desenvolvimento deste projeto é descrito na seção a seguir.

4 DESENVOLVIMENTO

Este capítulo discute alguns dos principais passos realizados nos *Sprints* para a realização do projeto, além de apresentar as dificuldades encontradas e os resultados obtidos com a utilização das tecnologias.

4.1. PREPARAÇÃO

Antes do início da codificação do projeto foram definidas as metas para cada *Sprint* a ser realizado (*Product Backlog*). Depois de possuir todas as metas definidas e antes do inicio de um *Sprint*, foram elencadas as tarefas necessárias para atingir a sua meta (*Sprint Backlog*), e só depois foi iniciada a execução do *Sprint*. O quadro 4 apresenta o *Product Backlog* do projeto, o qual contém a meta de cada item, a duração estimada de dias de desenvolvimento e em qual *Sprint* o item deveria ser construído.

Product Backlog				
Nº	**Item**	**Meta**	**Estimativa (em dias)**	**Sprint**
1	Construção do servidor HTTP (servidor)	Desenvolver a estrutura básica do servidor para que seja possível iniciar a comunicação entre cliente e servidor.	1	1
2	Módulo endereço (servidor)	Permitir o gerenciamento dos itens de endereço (CEP, UF, Cidade, Bairro, Logradouro) na plataforma.	1	1
3	Módulo estabelecimento (servidor)	Permitir o gerenciamento dos estabelecimentos cadastrados na plataforma.	2	1
4	Módulo produto (servidor)	Permitir o gerenciamento dos produtos de cada estabelecimento na plataforma.	2	2
5	Página inicial do aplicativo (aplicativo)	Desenvolvimento da página inicial do aplicativo para que seja possível acessar as funcionalidades que serão incrementadas no futuro.	1	2
6	Busca de produtos (aplicativo)	Possibilitar a busca e exibição dos produtos que estão cadastrados na plataforma. Exibir apenas as informações básicas.	1	2
7	Detalhes de produto da busca (aplicativo)	Permitir que quando o usuário selecionar um produto busque e exiba as informações adicionais do mesmo. Deve exibir no mesmo item da busca.	2	2
8	Exibir estabelecimento (aplicativo)	Buscar e exibir todas as informações referentes ao estabelecimento selecionado em uma nova página, quando selecionado.	2	3

Product Backlog				
Nº	**Item**	**Meta**	**Estimativa (em dias)**	**Sprint**
9	Comanda eletrônica 1 (aplicativo do garçom)	Exibir os produtos do estabelecimento.	1	3
10	Comanda eletrônica 2 (aplicativo do garçom)	Vincular os produtos escolhidos a uma comanda.	1	3
11	Consultar comanda (aplicativo)	Buscar e exibir as informações do valor gasto em uma comanda.	2	3
12	Dividir comanda (aplicativo)	Transmitir para o servidor o número de parcelas escolhidas para o pagamento da comanda.	1	4

Quadro 4 – Product Backlog do projeto
Fonte: Autoria própria.

Para o desenvolvimento dos arquivos de código-fonte foi utilizado o editor de códigos Atom[43], o qual é um editor *open-source* desenvolvido pela empresa GitHub Inc.[44]. O sistema operacional utilizado durante o desenvolvimento e testes foi o ElementaryOS[45] versão 0.2.1(Luna), o qual é uma distribuição Linux baseada na versão 12.04 do Ubuntu. A ferramenta Chrome Dev Tools[46] também foi utilizada para simular o comportamento da interface dos aplicativos em dispositivos com diferentes tamanhos e capacidades de tela, a qual facilitou e tornou mais rápida a visualização e o desenvolvimento dos aplicativos. O teste dos aplicativos foi realizado no emulador Android Virtual Device e em um dispositivo da marca Motorola, modelo Moto G (primeira geração), que possui o sistema operacional Android (versão 5.0.2).

A seguir são apresentadas as tarefas realizadas em cada *Sprint*, as dificuldades encontradas durante o desenvolvimento e o resultado obtido após a realização de cada *Sprint*.

4.2. *SPRINT* Nº 1

No primeiro *Sprint* iniciou-se a construção do servidor com a criação da estrutura básica para permitir a comunicação entre cliente e servidor, além do desenvolvimento dos módulos *Endereço* e *Estabelecimento*.

[43] https://atom.io/
[44] https://github.com/
[45] http://elementary.io/
[46] https://developer.chrome.com/devtools

4.2.1. *Sprint Backlog*

Para atingir as metas do *Sprint* número 1 foram realizadas as seguintes tarefas, apresentadas no quadro 5:

1º Sprint Backlog	
Item do Product Backlog	**Tarefas**
Construção do servidor HTTP (servidor)	Incluir os módulos principais da aplicação (pastas)
	Preparar a estrutura de arquivos do servidor
	Preparar uma rota para responder aos testes
	Teste de disponibilidade do servidor
Módulo endereço (servidor)	Desenvolver o modelo dos dados
	Desenvolver o *controller*
	Desenvolver as rotas para responder as requisições
	Construir uma interface com CRUD de gerenciamento
	Teste das operações do CRUD
Módulo estabelecimento (servidor)	Desenvolver o modelo dos dados
	Desenvolver o *controller*
	Desenvolver as rotas para responder as requisições
	Construir uma interface com CRUD de gerenciamento
	Desenvolver a busca de endereços por CEP
	Teste das operações do CRUD

Quadro 5 – Primeiro *Sprint Backlog*
Fonte: Autoria própria.

4.2.2. Desenvolvimento

Neste *Sprint* foram desenvolvidas as principais configurações do servidor, as quais foram utilizadas em todo o projeto. Primeiramente foram programados os arquivos do Express (servidor HTTP) e do Mongoose (acesso à base de dados MongoDB) para testar a disponibilidade do servidor e, juntamente com estas atividades, elaborada uma estrutura de pastas para armazenar os arquivos e facilitar a organização do servidor, onde cada pasta armazenava os arquivos conforme a função desempenhada. A figura 11 apresenta a estrutura final de pastas do servidor do projeto.

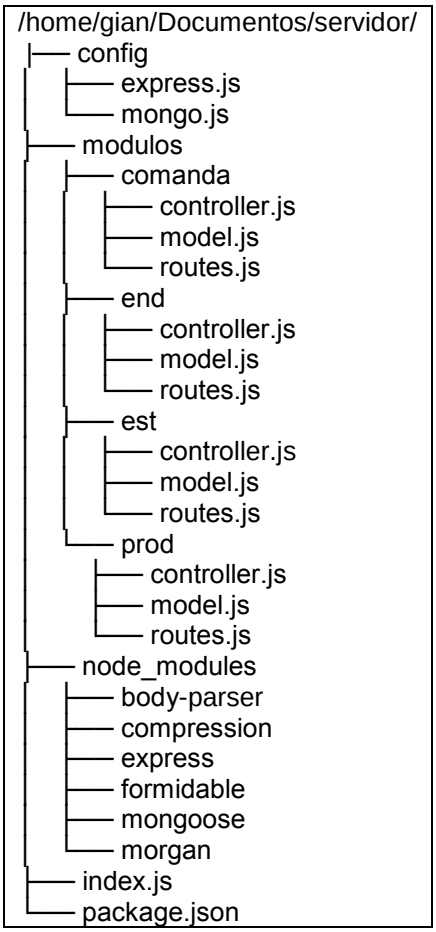

Figura 11 – Árvore de diretórios do servidor do projeto
Fonte: Autoria própria.

O diretório *config* possui dois arquivos de configuração do servidor, o arquivo *express.js*[47] com as configurações do servidor *web* executado sob a plataforma Node.js (inclusão e configuração de módulos, rotas, endereço IP e porta) e o arquivo *mongo.js*[48] com as configurações necessárias para o módulo Mongoose estabelecer uma conexão com a base de dados MongoDB.

O diretório *modulos* possui subdiretórios que correspondem aos módulos do servidor, cada um com o nome do respectivo módulo e seus arquivos de código-fonte relacionados: o modelo (*model.js*) dos dados armazenados, o controlador (*controller.js*) das tarefas desempenhadas e o arquivo com as rotas (*routes.js*) do módulo. A estrutura dos arquivos dos módulos foi elaborada seguindo o padrão *Model-View-Controller* (MVC) e permitiu maior controle e organização do projeto.

O diretório *node_modules* é gerado automaticamente pelo npm quando os módulos são adicionados ao projeto e conta com subdiretórios que armazenam os

[47] O arquivo *express.js* está disponível no Apêndice A deste documento.
[48] O arquivo *mongo.js* está disponível no Apêndice B deste documento.

arquivos dos mesmos. Nenhuma alteração foi efetuada neste diretório, pois ele é gerenciado pelo npm.

O arquivo *index.js* é o arquivo principal do servidor. Aqui são incluídos os arquivos de configuração do diretório *config* e inicializado o servidor a partir das configurações definidas. Quando o servidor precisa ser iniciado, basta executar este arquivo e o servidor web estará disponível, caso as configurações estejam corretas. Também há o arquivo *package.json*, o qual é o arquivo de configuração do npm para gerenciar o projeto (módulos necessários, nome e versão da aplicação, entre outros), gerado automaticamente pelo comando *npm init* e requerido por todas as aplicações Node.js.

Após a organização da estrutura do projeto foi construído o módulo *Endereço*, responsável por gerenciar o cadastro de logradouros de algumas cidades. Este módulo foi necessário para implementar a funcionalidade de preenchimento automático de informações de localização, presente no módulo *Estabelecimento*. Depois de finalizar os testes deste módulo, deu-se inicio à construção do módulo *Estabelecimento*.

O módulo *Estabelecimento* gerenciava as informações dos estabelecimentos cadastrados na plataforma e contava com operações de inserção, alteração, remoção e busca de dados, todas configuradas no arquivo *controller.js*. Cada tarefa a ser desempenhada era executada conforme a rota acessada, a qual foi pré-configurada no arquivo *routes.js* do módulo. A figura 12 apresenta um exemplo de rota do módulo *Estabelecimento*, a qual atende por uma requisição HTTP do tipo PUT, recebe um parâmetro (*id*) por meio da rota e chama a função *alt* do objeto *estCtrl*, o qual é o controlador deste módulo .

```
var express = require('express')
    , router = express.Router()
    , estCtrl = require('./controller.js');

router.put('/alt/:id', function(req, res) {
    estCtrl.alt(req, res);
});

// todas as outras rotas são incluídas abaixo
module.exports = router;
```

Figura 12 – Exemplo de rota do módulo Estabelecimento
Fonte: Autoria própria.

4.2.3. Resultados obtidos

Com a finalização deste *Sprint* foi possível comunicar-se com o servidor e desempenhar testes de manipulação dos módulos *Endereço* e *Estabelecimento*. Durante o cadastro de um novo estabelecimento e após o preenchimento do campo CEP, o sistema efetuava uma busca pelas informações cadastradas no módulo *Endereço* e preenchia automaticamente os campos com as informações encontradas, facilitando para o usuário o preenchimento dos campos.

4.3. *SPRINT* Nº 2

No segundo *Sprint* foi desenvolvido o módulo *Produto* do servidor, responsável por gerenciar o registro de produtos na plataforma, e alterado o módulo *Estabelecimento* construído no *Sprint* anterior, a fim de permitir a inclusão de usuários de um estabelecimento. Além disso, deu-se inicio à construção do aplicativo de busca e disponibilização dos dados da plataforma para o cliente.

4.3.1. Sprint Backlog

O quadro 6 apresenta as tarefas realizadas para atingir a meta do *Sprint* número 2:

2º Sprint Backlog	
Item do Product Backlog	**Tarefas**
Módulo estabelecimento (servidor)	Alteração no modelo de dados para permitir cadastro de usuário e senha
	Alterar cadastro de estabelecimento para permitir o cadastro de usuários
	Criar nova rota para validar a autenticação de usuário
Módulo produto (servidor)	Desenvolver o *model* dos dados
	Desenvolver o *controller*
	Desenvolver as rotas para responder as requisições
	Construir uma interface com CRUD de gerenciamento
	Teste das operações do CRUD
Página inicial do aplicativo (aplicativo)	Gerar estrutura inicial do aplicativo (pastas, bibliotecas)
	Adicionar elementos de *design* da página inicial (imagens, logo, animação)
	Construir página inicial com as opções de menu

2º Sprint Backlog	
Item do Product Backlog	**Tarefas**
	Preparar o arquivo com as rotas de navegação do aplicativo (*ng-router*)
Busca de produtos (aplicativo)	Preparar a rota que leva até a busca de produtos
	Preparar a lista que exibe os produtos
	Estabelecer conexão com o servidor e receber os dados
	Teste da exibição dos itens na tela
Detalhes de produto da busca (aplicativo)	Acrescentar a exibição de detalhes na lista anterior
	Estabelecer conexão com o servidor e receber os dados
	Testar funcionalidades integradas

Quadro 6 – Segundo *Sprint Backlog*
Fonte: Autoria própria.

4.3.2. Desenvolvimento

Este *Sprint* contou com uma alteração no módulo *Estabelecimento*, já finalizado no *Sprint* anterior. Esta alteração não gerou problemas para o desenvolvimento, pois a metodologia Scrum permite alterar a execução e os requisitos do projeto de maneira flexível. A figura 13 apresenta o trecho de código adicionado ao modelo de dados (*model.js*) do módulo *Estabelecimento* para permitir o vínculo de usuários a um estabelecimento, o qual corresponde a um vetor de usuários onde cada usuário possui um *login*, uma senha e um campo do tipo Booleano informando se o usuário é administrador ou não do sistema.

```
usuarios: [
  {
    login: {type:String, index:true, required:true, unique:true, lowercase:true},
    senha: {type:String, required:true},
    adm: {type:Boolean, default:false}
  }
]
```

Figura 13 – Trecho de código inserido no modelo de dados do módulo Estabelecimento
Fonte: Autoria própria.

Após concluir esta tarefa, prosseguiu-se com a construção do módulo *Produto* e do aplicativo para os clientes.

O módulo *Produto* armazena informações de produtos dos estabelecimentos cadastrados na plataforma, os quais seriam requisitados pelo aplicativo durante a busca por cardápios. Para buscar os dados na base de dados MongoDB, o controlador deste módulo efetua uma busca pelo nome do produto requisitado pelo

usuário. A figura 14 apresenta o trecho de código do arquivo *controller.js* deste módulo, responsável por buscar os dados e retorná-los para o usuário, onde *buscar_nome* é o nome da função do controlador, a variável *query* contém o termo para a busca dos dados e *Produto* é o nome do modelo dos dados onde será efetuada a busca, conforme as opções configuradas. Após o Mongoose retornar os dados, os mesmos são repassados à função *responder*, a qual retorna ao usuário a resposta.

```
buscar_nome: function(req, res) {
    var query = {nome: new RegExp(req.params.nome, "i")};
    Produto
        .find(query)
        .select('_id nome preco estabelecimento')
        .populate('estabelecimento', '_id nome')
        .exec(function(err, data) {
            responder(err, data, res);
        });
},
```

Figura 14 – Função de busca de produtos do módulo *Produtos*
Fonte: Autoria própria.

Após disponibilizar as informações dos produtos, deu-se início à construção do aplicativo dos clientes dos estabelecimentos. Esta tarefa exigiu muito tempo no início do desenvolvimento, pois era necessário definir a estrutura adequada para o aplicativo, porém a programação foi facilitada pelo Ionic *Framework* já que ele fornece os itens prontos da interface de usuário, bastando o desenvolvedor copiá-los e inseri-los no seu projeto.

Apesar de o Ionic *Framework* fornecer os itens prontos, a animação da página inicial foi feita utilizando CSS. O *framework* AngularJS possui um módulo adicional responsável pela animação de itens de interface, porém a sua utilização no projeto não foi bem sucedida porque gastou-se muito tempo para compreender seu funcionamento e a continuação com o mesmo poderia comprometer o prazo do projeto, portanto este módulo foi descartado e optou-se pelo CSS para dar movimento aos itens da página inicial. A tarefa de animar os itens da interface com CSS não foi tão simples e tomou grande parte do tempo.

Neste *Sprint* também se iniciou a manipulação das interfaces do aplicativo e esta tarefa exigiu tempo e esforço de desenvolvimento, visto que era preciso inserir um item e verificar como o mesmo apresentava-se na interface. Apesar de ser uma tarefa tediosa, o Ionic *Framework* disponibiliza uma ferramenta de visualização de interfaces chamada *Live Reload* que exibe no navegador Google Chrome o estado

atual das interfaces. Dentro do Google Chrome há o *Chrome Dev Tools* que permite simular o tamanho da tela de vários dispositivos móveis, então a união destas ferramentas permitiu verificar o comportamento da interface dos aplicativos em vários dispositivos móveis e trouxe agilidade para o desenvolvimento e os testes.

4.3.3. Resultados obtidos

Ao final deste *Sprint* o cadastro de produtos já estava habilitado e era possível buscar informações e detalhes dos produtos dos estabelecimentos utilizando o aplicativo dos clientes. Também era possível vincular usuários a um estabelecimento cadastrado na plataforma, já que esta funcionalidade seria necessária no próximo *Sprint* a ser realizado.

O resultado inicial do desenvolvimento do aplicativo dos clientes pode ser visualizado na figura 15, a qual apresenta a Tela inicial do aplicativo dos clientes (a), intitulado Cardápio de Bolso, seguido da tela de busca de produtos (b) que exibe dois resultados e a exibição de mais detalhes de um produto (c), respectivamente.

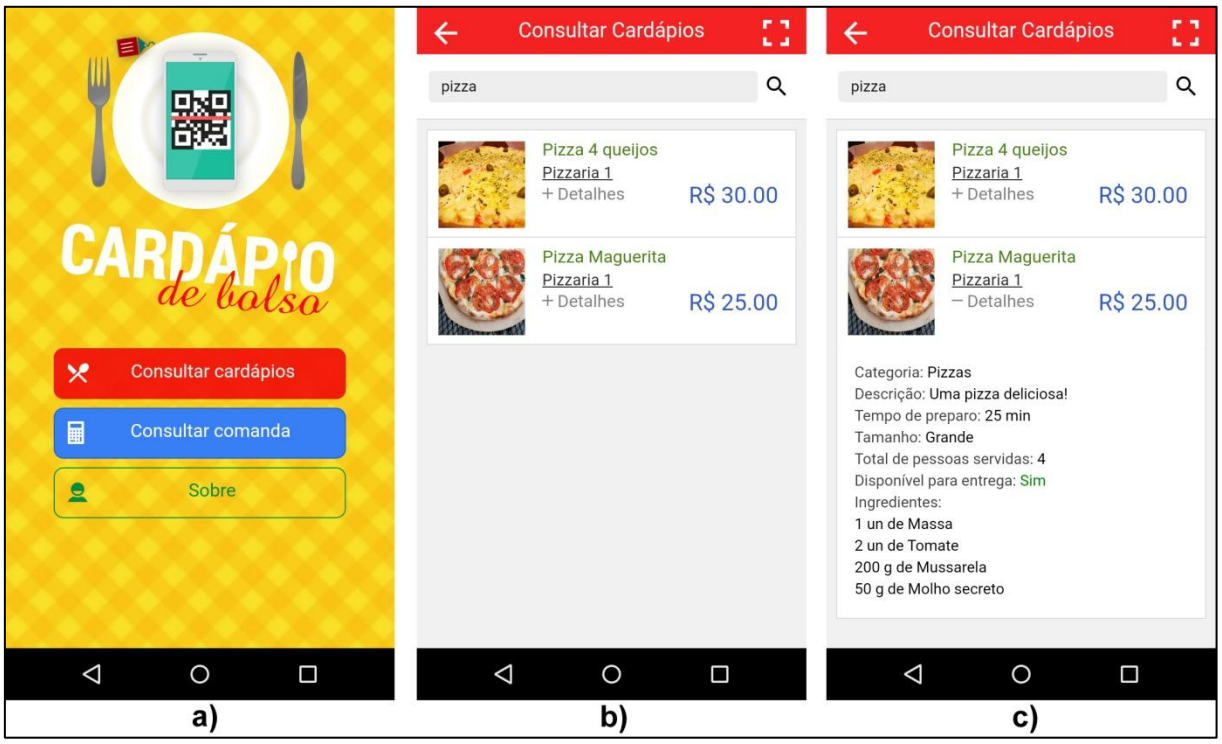

Figura 15 – Tela inicial do aplicativo (a), busca de produtos (b) e detalhes de um produto (c)
Fonte: Autoria própria.

4.4. *SPRINT* Nº 3

Durante este *Sprint* foram adicionadas novas funcionalidades ao aplicativo dos clientes e iniciou-se a construção da comanda eletrônica para uso do estabelecimento, a qual gerenciava o pedido dos clientes e o transmitia para o servidor, permitindo o gerenciamento e a consulta do valor da comanda por ambos os aplicativos.

4.4.1. Sprint Backlog

O quadro 7 apresenta as tarefas realizadas para atingir a meta do *Sprint* número 3:

3º Sprint Backlog	
Item do Product Backlog	**Tarefas**
Exibir estabelecimento (aplicativo)	Criar nova rota para levar à página de detalhes do estabelecimento
	Criar no arquivo *services.js* um item para buscar as informações do estabelecimento
	Criar a página que exibe as informações do estabelecimento
	Alterar pesquisa de produtos para que o nome do restaurante leve até a página com as informações do mesmo
	Testar a exibição das informações
Comanda eletrônica 1 (aplicativo do garçom)	Criar a estrutura do novo aplicativo
	Criar sistema de login no aplicativo (deve verificar se é administrador ou garçom)
	Exibir as opções de menu conforme o nível de usuário (adm., garçom)
	Exibir os itens (produtos) do estabelecimento
	Testar os itens desenvolvidos
Comanda eletrônica 2 (aplicativo do garçom)	Permitir adicionar, alterar e remover os itens de uma comanda.
	Enviar para o servidor a comanda. Deve ser possível gravar, alterar e remover a comanda e seus itens.
	Testar as funcionalidades desenvolvidas
Consultar comanda (aplicativo)	Desenvolver nova rota para a página de consulta
	Criar página inicial da consulta de comanda com as instruções de como utilizar e um botão para prosseguir
	Integrar a funcionalidade de leitor de QR Code para obter os dados da comanda.
	Preparar página que exibe os resultados ao usuário
	Teste das funcionalidades
Dividir comanda (aplicativo)	Criar novo campo no modelo dos dados para armazenar o total de parcelas de pagamento (servidor).
	Criar nova rota para persistir o total de parcelas do pagamento (servidor).
	Alterar o controller para realizar as tarefas necessárias (servidor).

3º Sprint Backlog	
Item do Product Backlog	**Tarefas**
	Alterar aplicativo para solicitar a divisão do valor da comanda.
	Teste das funcionalidades.

Quadro 7 – Terceiro *Sprint Backlog*
Fonte: Autoria própria.

4.4.2. Desenvolvimento

A construção da funcionalidade de exibição de informações de um estabelecimento não exigiu esforço de desenvolvimento e foi simples de ser adicionada ao aplicativo dos clientes, bastando requisitar os dados do estabelecimento que já estavam disponíveis no servidor e exibir na tela. O desenvolvimento da comanda eletrônica exigiu a construção de um novo aplicativo, intitulado Comanda de Bolso, utilizado pelos garçons do estabelecimento para gerenciar os pedidos e as informações cadastradas na plataforma (produtos, informações do estabelecimento, entre outros), conforme o nível de acesso do usuário (baseando-se no campo *adm* do cadastro de usuários, criado no *Sprint* número 2).

Neste *Sprint* também foi incluída nos dois aplicativos as funcionalidades para acessar os recursos nativos dos dispositivos. Foram utilizados *plug-ins* que acessavam a câmera, verificavam a conectividade do aparelho, faziam o dispositivo vibrar e exibiam mensagens na tela. Além disso, foi utilizada uma biblioteca de código Javascript chamada ngCordova[49] para acessar os *plug-ins*, já que esta disponibiliza algumas funções que permitem o rápido e fácil acesso aos recursos nativos. Para utilizar algum *plug-in* bastava chamar a função fornecida pelo ngCordova e aguardar o resultado retornado pela função. A biblioteca ngCordova merece destaque, pois facilita o desenvolvimento, já que simplifica o acesso aos *plug-ins*, e colabora para a organização do código, visto que sua utilização requer poucas linhas de código.

A figura 16 apresenta um trecho de código do aplicativo Cardápio de Bolso que, a partir da função do ngCordova e do *plug-in* BarcodeScanner, acessa a câmera do dispositivo e efetua a leitura do Código QR, retornando os dados obtidos.

[49] http://ngcordova.com/

O item *checkComanda* é a função Javascript chamada na interface do usuário e *$cordovaBarcodeScanner.scan()* é o objeto e a função, respectivamente, da biblioteca ngCordova que desempenha a tarefa de acessar o *plug-in*. Depois de efetuar a leitura, um objeto com as informações obtidas (*barcodeData*) é retornado para o código Javascript que realiza as tarefas necessárias. Neste caso é convertido para um objeto Javascript e algumas informações são passadas para a função *config* do objeto *ViewComandaService*.

```
$scope.checkComanda = function() {
      $cordovaBarcodeScanner.scan()
         .then(function(barcodeData) {
            try {
               var obj = angular.fromJson(barcodeData.text);
               ViewComandaService.config(obj.lnk,obj.mesa);
               $state.go('viewComanda');

            } catch(err) {
               if(!barcodeData.cancelled) {
                  var alertPopup = $ionicPopup.alert({
                     title: 'Erro',
                     template: 'O Código QR é inválido.'
                  });
               }
            }
         }, function(error) {
            var alertPopup = $ionicPopup.alert({
               title: 'Erro!',
               template: 'Ocorreu um erro durante a leitura do Código QR. Tente novamente.'
            });
         });
}
```

Figura 16 – Função do controlador para leitura de Código QR
Fonte: Autoria própria.

O desenvolvimento do item “Comanda eletrônica 2 (aplicativo do garçom)” do *Product Backlog* exigiu a construção de um novo módulo no servidor, denominado *Comanda*. A construção deste módulo não exigiu tanto esforço porque a estrutura escolhida para o servidor facilitou a inclusão de novas funcionalidades. Bastou criar uma pasta com os arquivos de rota do Express, o modelo dos dados do Mongoose, o arquivo controlador (*controller*) com as funções e incluir a rota no arquivo de configuração do Express e o novo módulo já estava em funcionamento.

O item “Dividir comanda (aplicativo)” do *Product Backlog* estava relacionado ao *Sprint* de número 4, porém por possuir uma meta fácil de ser atingida foi incluído ao terceiro *Sprint.* Vale destacar novamente que essa alteração na execução do projeto é facilitada pelo Scrum e não compromete o desenvolvimento do projeto,

porém a adição de várias tarefas no mesmo *Sprint* pode comprometer o prazo estabelecido para entrega do mesmo.

4.4.3. Resultados obtidos

Neste Sprint foi adicionada ao aplicativo dos clientes a opção de visualizar as informações de um estabelecimento após efetuar uma busca de produtos. O resultado desta alteração pode ser verificado na figura 17, onde são exibidas as informações gerais (a), de localização (b) e todos os produtos (c) de um estabelecimento.

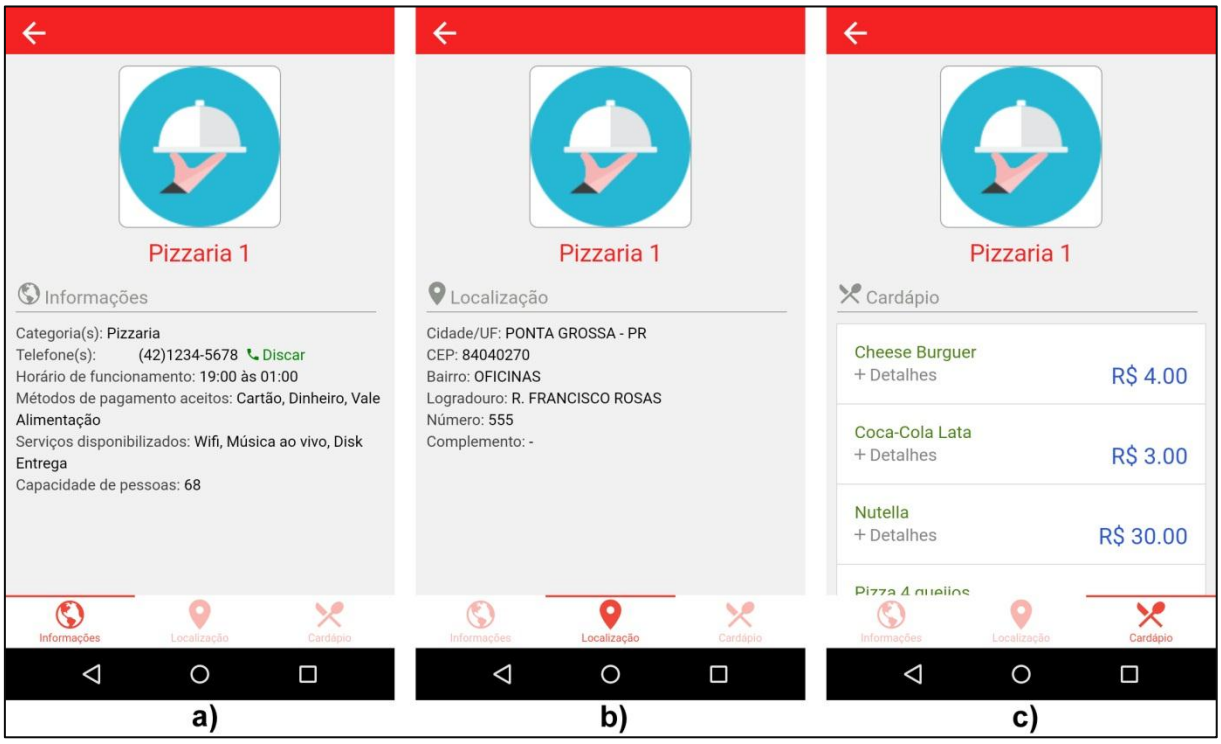

Figura 17 – Informações adicionais de um estabelecimento
Fonte: Autoria própria.

A figura 18 apresenta a tela de *login* (a), os itens de um estabelecimento que podem ser adicionados a uma comanda (b) e a tela de seleção da quantidade de itens de um produto (c), e a figura 19 apresenta a tela de exibição de uma comanda com alguns itens (a), uma mensagem de alerta após o envio da comanda para o servidor (b) e as opções de alteração de cadastro disponíveis (c), todas do aplicativo Comanda de Bolso. Vale destacar que as opções de alteração de cadastro só estão disponíveis quando o usuário tem privilégios de Administrador do sistema.

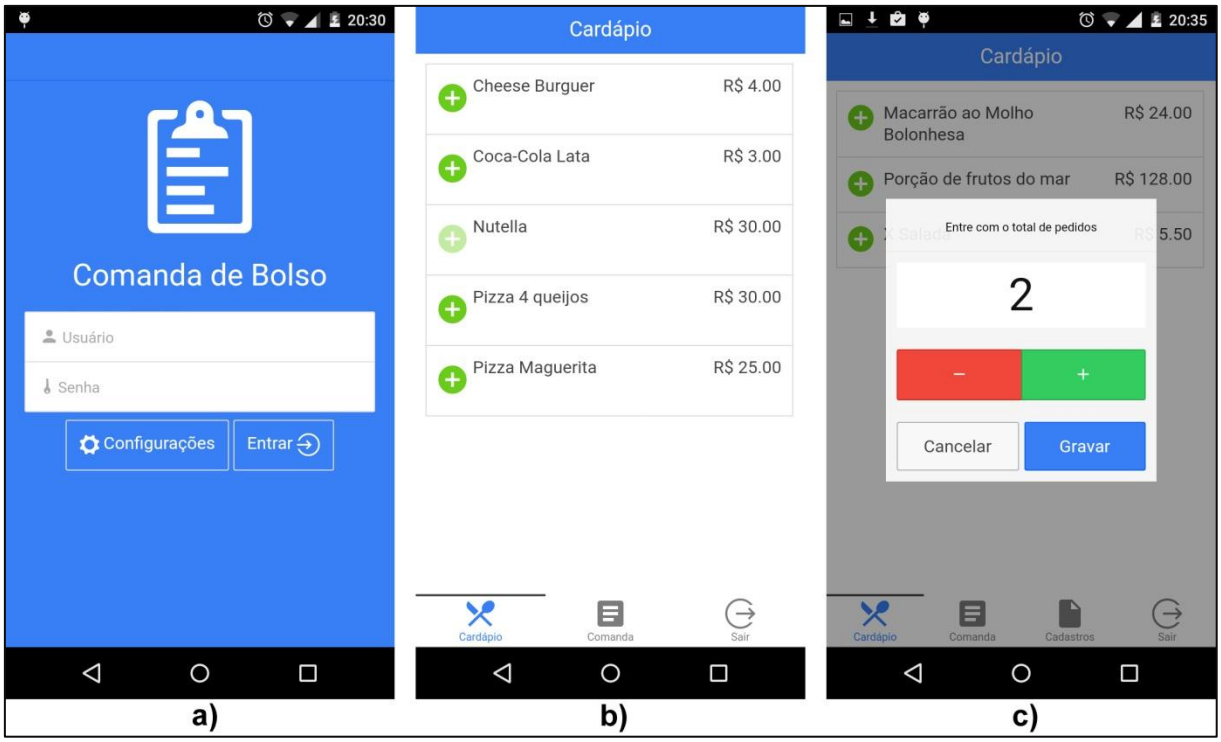

Figura 18 – Tela de *login* (a), itens de estabelecimento (b) e seleção de quantidade (c)
Fonte: Autoria própria.

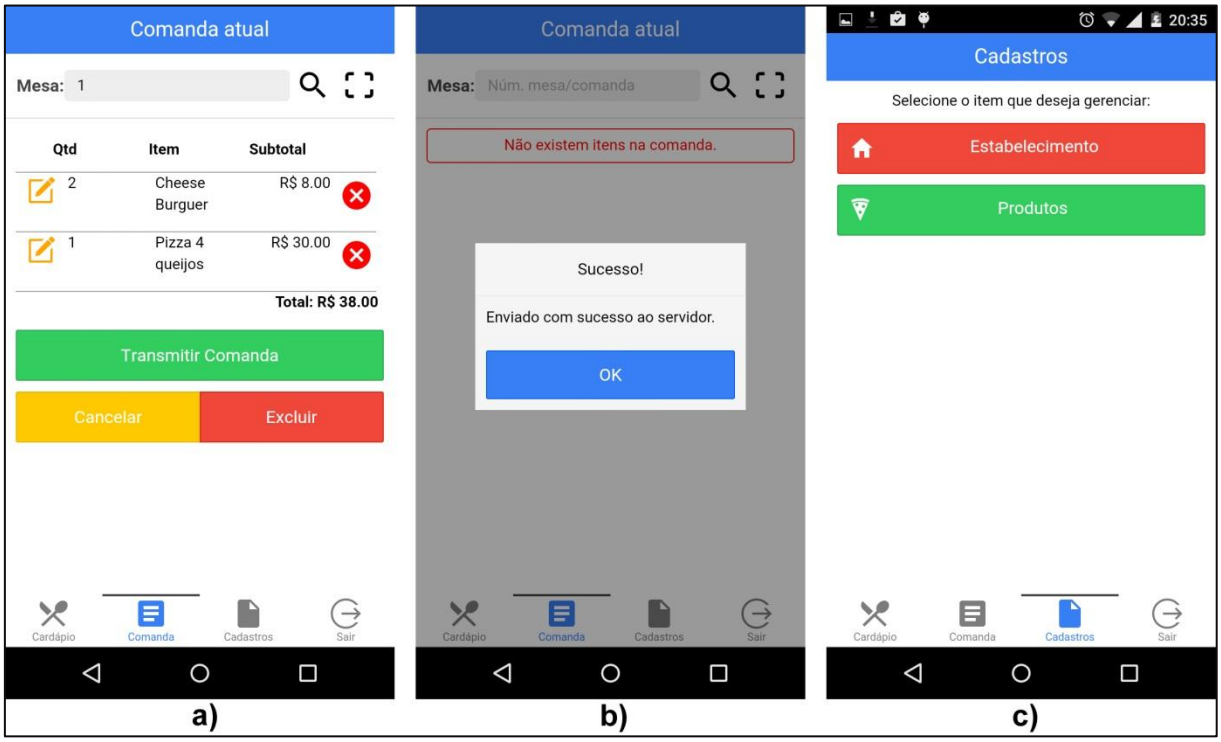

Figura 19 – Comanda atual (a), mensagem de sucesso (b) e opções de cadastro (c)
Fonte: Autoria própria.

A figura 20 apresenta a tela com as informações de cadastro de um estabelecimento (a), cadastro de produto (b) e alteração de imagem de um produto (c) do aplicativo Comanda de Bolso.

Figura 20 - Cadastros de estabelecimento (a), produto (b) e alteração de imagem (c)
Fonte: Autoria própria.

A figura 21 apresenta as imagens do aplicativo Cardápio de Bolso, onde apresenta as telas de instruções para escaneamento do código QR com a identificação da comanda (a), os itens de uma comanda (b) e a opção de divisão do valor da comanda (c).

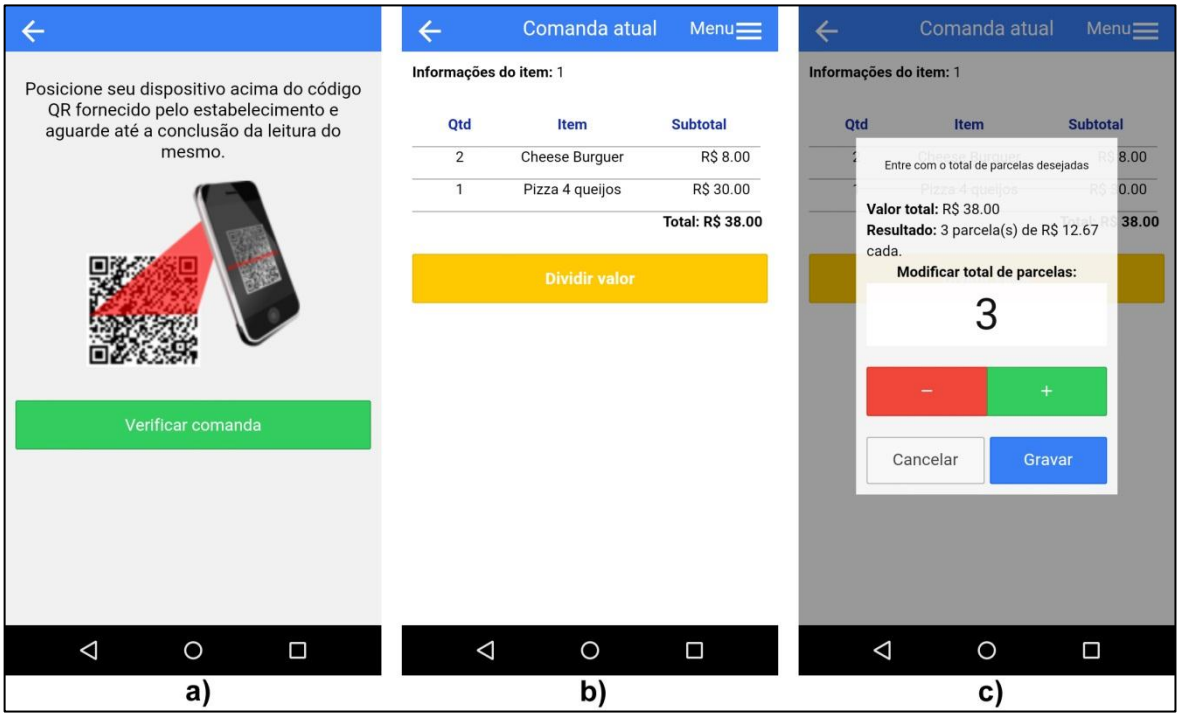

Figura 21 – Instruções de escaneamento (a), itens da comanda (b) e divisão de valor (c)
Fonte: Autoria própria.

5 DISCUSSÕES FINAIS

Nesta seção são apresentadas as considerações finais obtidas a partir deste trabalho, bem como sugestões para trabalhos futuros.

5.1. CONSIDERAÇÕES FINAIS

Este trabalho propôs o desenvolvimento de uma nova solução para os estabelecimentos gastronômicos da cidade de Ponta Grossa, Paraná, com o objetivo de oferecer um novo atrativo aos estabelecimentos e consequentemente aumentar sua competitividade através de uma melhoria do atendimento aos clientes, conforme proposto por Bindes (2012), Rigodanzo e Paz (2006) e Porter (1989 apud BINDES, 2012).

Foi utilizado o potencial oferecido pelos dispositivos móveis – que estão muito difundidos entre a população – a partir de um aplicativo móvel para os dispositivos. O primeiro aplicativo – voltado aos clientes – permite a consulta de cardápios e informações dos estabelecimentos, bem como a verificação do total gasto em um estabelecimento. O segundo aplicativo – voltado aos garçons – permite gravar os pedidos dos clientes e transmiti-los ao servidor, além de gerenciar o cadastro dos produtos na plataforma (caso o usuário seja administrador do sistema).

O desenvolvimento dos aplicativos utilizou a abordagem híbrida de construção de aplicativos, a qual permite construir um aplicativo utilizando HTML, CSS e Javascript e acessar funcionalidades nativas dos dispositivos – câmera, conectividade – que não podem ser acessadas apenas por HTML e Javascript. Esta abordagem permitiu construir um aplicativo de qualidade e bem apresentável visualmente, fazendo uso de recursos nativos dos dispositivos móveis e que consequentemente colaboram para uma melhor experiência de uso aos usuários. O maior benefício desta abordagem é o curto tempo para o desenvolvimento de uma aplicação estruturada, funcional e bem organizada, que pode utilizar recursos nativos dos *smartphones* e entregar ao usuário uma experiência satisfatória durante seu uso.

Todas as ferramentas utilizadas no projeto são *open-source*, ou seja, não exigiram licença de desenvolvedor ou pagamento para utilização, portanto não houve gasto durante o desenvolvimento. Este fator é importante para as empresas de desenvolvimento de *software*, já que a redução de custos é almejada em várias organizações. Além disso, a construção dos aplicativos ocorreu de maneira fácil e tranquila, visto que as ferramentas e tecnologias utilizadas possuem uma baixa curva de aprendizado e tornaram o trabalho mais produtivo.

Apesar deste projeto não ter sido aplicado em um estabelecimento real para comprovar seu desempenho, todos os objetivos estabelecidos foram alcançados. Além de o seu desenvolvimento avaliar o potencial de utilização das tecnologias escolhidas e da abordagem híbrida de construção de aplicativos móveis, MongoDB, Express, Mongoose e NodeJS mostraram ser excelentes tecnologias para a construção de servidores *web*, já que permitiram construir uma solução estruturada, organizada e com alto desempenho em pouco tempo. Ionic *Framework*, AngularJS, ngCordova e Apache Cordova mostraram ser um conjunto de tecnologias que permitem construir bons aplicativos com pouco tempo de desenvolvimento, além de facilitar a programação e manutenção do código.

Os aplicativos obtiveram um resultado muito bom quanto ao desempenho e a qualidade da interface, além de o tempo necessário para atingir esses resultados ser baixo. Todo o projeto apresentou-se organizado na mesma estrutura, portanto a manutenção e as alterações no código-fonte tornaram-se muito mais fáceis. Grande parte do sucesso deste projeto deve-se, principalmente, às ferramentas utilizadas que facilitaram o desenvolvimento e a organização do mesmo.

5.2. TRABALHOS FUTUROS

Este projeto permite a expansão das funcionalidades oferecidas, bem como o desenvolvimento de novas soluções com as tecnologias aqui utilizadas.

Acredita-se que o mesmo pode se beneficiar com a inclusão de funcionalidades que permitam, por exemplo, chamar o garçom até a mesa, compartilhar com os amigos a experiência sobre os pratos degustados em um estabelecimento, convidar amigos para ir a um estabelecimento utilizando o

aplicativo, exibir promoções do estabelecimento para o usuário, premiar usuários com cupons de desconto entre outras funcionalidades.

As tecnologias utilizadas permitem a construção de várias soluções de *software*, já que oferecem inúmeras possibilidades de aplicação e utilização dos recursos nativos dos dispositivos móveis. Com os *plug-ins* do Apache Cordova e com a biblioteca ngCordova já é possível acessar os recursos nativos de conectividade, câmera, estado da bateria, geolocalização, contatos, calendário, informações gerais do dispositivo, orientação do dispositivo, arquivos, lanterna, bússola, entre outros[50] recursos já presentes nos dispositivos móveis. Todas essas facilidades permitem desenvolver soluções com diversas áreas de aplicação.

Uma nova tecnologia chamada NW.js[51] – anteriormente conhecida por NodeWebKit – combina NodeJS e Chromium[52] e permite a construção de *software* instalável em computadores utilizando tecnologias *web*. Os *softwares* são desenvolvidos com HTML, CSS e Javascript e não requerem conexão com a Internet para funcionar, portanto esta nova tecnologia, que também faz uso do NodeJS, apresenta-se interessante para o desenvolvimento de novas soluções a partir das tecnologias *web*.

[50] Os *plug-ins* disponíveis podem ser verificados em http://plugins.cordova.io/#/ e http://ngcordova.com/docs/plugins/.
[51] http://nwjs.io/
[52] Chromium é o projeto *open-source* do navegador Google Chrome.

REFERÊNCIAS

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Pro smartphone cross-platform development:** iPhone, blackberry, windows mobile and android development and distribution. [S.l.]: Apress, 2010.

ALMEIDA, Will Ribamar Mendes; DE ARAÚJO FILHO, Patrício Moreira; QUIXABEIRA, Rafael Freitas. Utilização da tecnologia Node JS para consumo eficiente dos recursos de servidores web. In: **Acta Brazilian Science**, São Luiz, ano I, vol.1, p. 100-111, abr. 2013.

AMATYA, Suyesh; KURTI, Arianit. Cross-platform mobile development: challenges and opportunities. In: **ICT Innovations 2013**. Springer International Publishing, 2014. p. 219-229.

ANGULARJS. **What Is Angular?** Disponível em: <https://docs.angularjs.org/guide/introduction> . Acesso em: 05-Mar-2015.

APPCELERATOR. **Native vs. HTML5 Mobile App Development:** Which option is best? 2012. Disponível em: <http://www.appcelerator.com/enterprise/resource-center/white-papers/native-vs-html5-mobile-app-development/>. Acesso em: 29-Dez-2014.

BINDES, Rodrigo dos Santos. **A tecnologia da informação no auxílio à gestão do grupo alfa de restaurantes**. 2012. Disponível em: < http://repositorio.uniceub.br/handle/123456789/1010>. Acesso em: 29-Set-2014.

BORGES, Daniele Meira. **Fatores que influenciam a aceitação de tecnologia:** a percepção de gestores e funcionários em uma rede de restaurantes. 2012. Disponível em: <http://bdm.unb.br/handle/10483/4348>. Acesso em: 04-Set-2014.

BRADLEY, Adam. **Where does the Ionic Framework fit in?** 2013. Disponível em: <http://ionicframework.com/blog/where-does-the-ionic-framework-fit-in/>. Acesso em: 19-fev-2015.

BRASSCOM. **Mapa de Conectividade.** Disponível em <http://www.brasscom.org.br/brasscom/Portugues/detInstitucional.php?codArea=3&codCategoria=48>. Acesso em 14 nov. 2014a.

_____. **Mobilidade.** Disponível em: <http://www.brasscom.org.br/brasscom/Portugues/detInstitucional.php?codArea=3&codCategoria=48>. Acesso em: 14 nov. 2014b.

CANALTECH. **O que é API?** 2014. Disponível em: <http://canaltech.com.br/o-que-e/software/O-que-e-API/>. Acesso em: 06-fev-2015.

CARTEIRAS de habilitação terão itens de segurança em 2015. 2014. Disponível em: <http://www.brasil.gov.br/infraestrutura/2014/12/carteiras-de-habilitacao-terao-itens-de-seguranca-em-2015>. Acesso em: 25-mar-2015.

CHARLAND, Andre; LEROUX, Brian. Mobile application development: web vs. native. In: **Communications of the ACM**, v. 54, n. 5, p. 49-53, 2011. Disponível em: < https://queue.acm.org/detail.cfm?id=1968203> . Acesso em: 18-Nov-2014.

CHROME V8. Disponível em: <https://developers.google.com/v8/intro>. Acesso em: 28-jan-2015.

CORDOVA. **Overview.** Disponível em: <http://cordova.apache.org/docs/en/4.0.0/guide_overview_index.md.html#Overview> . Acesso em: 17-fev-2015a.

_____. **The Command-Line Interface.** Disponível em: <http://cordova.apache.org/docs/en/4.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface>. Acesso em: 17-fev-2015b.

_____. **Platform Guides.** Disponível em: <http://cordova.apache.org/docs/en/4.0.0/guide_platforms_index.md.html#Platform%20Guides>. Acesso em: 17-fev-2015c.

_____. **Plugin Development Guide.** Disponível em: <http://cordova.apache.org/docs/en/4.0.0/guide_platforms_index.md.html#Platform%20Guides>. Acesso em: 18-fev-2015d.

CORE Scrum. Disponível em: <https://www.scrumalliance.org/why-scrum/core-scrum-values-roles>. Acesso em 24-mar-2015.

CORRÊA, Ana Grasielle Dionisio, et al. Design Centrado no Usuário no Contexto Móvel: estudo de caso com um aplicativo para gerenciamento de contas de bares e restaurantes. In: **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação**, v. 1, n. 1, 2014. Disponível em: < http://revistas.setrem.com.br/index.php/reabtic/article/view/17>. Acesso em: 22-Set-2014.

COULOURIS, George, et al. **Sistemas Distribuídos:** Conceitos e Projeto. 4. ed. São Paulo: Bookman Editora, 2001.

COUTO, Erick. **Quais as diferenças entre os teclados Qwerty e Swype?** 2012. Disponível em: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2012/08/quais-diferencas-entre-os-teclados-qwerty-e-swype.html>. Acesso em: 06-fev-2015.

CROSS, Zach; POCHEC, Aga; SANTIAGO, Daniel; SINGH, Divit. **Developing mobile apps with Node.js and MongoDB, Part 1**: A team's methods and results. 2013. Disponível em <http://www.ibm.com/developerworks/library/mo-nodejs-1/>. Acesso em 24 dez. 2014a.

_____. **Developing mobile apps with Node.js and MongoDB, Part 2:** Hints and tips. 2013. Disponível em: <http://www.ibm.com/developerworks/mobile/library/mo-nodejs-2/index.html>. Acesso em: 24 dez. 2014b.

DAYLEY, Brad. **Node. js, MongoDB, and AngularJS Web Development**. Nova Jersey: Pearson Education, 2014.

DA SILVA, Marcelo Moro; SANTOS, Marilde Terezinha Prado. Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares. In: **Revista TIS**, v. 3, n. 2, 2014. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/86>. Acesso em: 18-Nov-2014.

DE LAET, Luis Gabriel Lins. **Utilização da Plataforma Android no desenvolvimento de um aplicativo para o setor sucroalcooleiro.** 2010. Disponível em: <http://www.uems.br/portal/biblioteca/repositorio/2012-06-26_18-08-13.pdf>. Acesso em: 9-set-2014.

DE MENDONÇA, Vinıcius Rafael Lobo; BITTAR, Thiago Jabur; DE SOUZA DIAS, Márcio. **Um estudo dos Sistemas Operacionais Android e iOS para o desenvolvimento de aplicativos**. 2011. Disponível em: < http://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submission_54.pdf>. Acesso em: 03-Nov-2014.

ECARDAPIOS. Disponível em <http://www.e-cardapios.com.br/web/pt-br>. Acesso em: 19-mar-2015.

EXPRESS. Disponível em: <http://expressjs.com/>. Acesso em: 29-jan-2015.

FINCOTTO, Marcos Apolinário. Automação Comercial utilizando Aplicativos Móveis-Um Foco na Plataforma Android. In: **Revista TIS**, v. 3, n. 2, 2014. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/85>. Acesso em: 18-Nov-2014.

FREEMAN, Adam. **Pro AngularJS**. [S.l.]: Apress, 2014.

GOOGLE. **Nosso Planeta Mobile:** Brasil. 2013. Disponível em: <http://services.google.com/fh/files/misc/omp-2013-br-local.pdf>. Acesso em: 19 nov. 2014.

HARTMANN, Gustavo; STEAD, Geoff; DEGANI, Asi. Cross-platform mobile development. In: **Tribal, Lincoln House, The Paddocks**, Tech. Rep, 2011.

HEUSER, Carlos Alberto. **Projeto de banco de dados.** 4. ed. Porto Alegre: Sagra Luzzatto, 2001. xvi, 204 p. (Livros didáticos; 4) ISBN 85-241-0590-9.

IHRIG, Colin J. **Pro Node. js for Developers**. Nova Iorque: Apress, 2013.

IONIC. **Chapter 1:** All about Ionic. Disponível em: <http://ionicframework.com/docs/guide/preface.html>. Acesso em: 19-fev-2015a.

_____. **Ionic Documentation Overview**. Disponível em: < http://ionicframework.com/docs/overview/>. Acesso em: 19-fev-2015b.

JSON. **Introducing JSON.** Disponível em: <http://www.json.org/>. Acesso em: 06-Mar-2015.

JUNIOR, Francisco de Assis Ribeiro. **Programação Orientada a Eventos no lado do servidor utilizando Node. js**. 2012. Disponível em: < http://www.infobrasil.inf.br/userfiles/16-S3-3-97136-Programa%C3%A7%C3%A3o%20Orientada___.pdf >. Acesso em: 15-Jan-2015.

JÚNIOR, Venilton Falvo et al. Uma comparação do tempo de implementação: Android vs. HTML5. In: **X Workshop Latinoamericano Ingeniería de Software Experimental**, 2013. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/eselaw/2013/005.pdf>. Acesso em: 22-Out-2014.

KAISER, Christian; MEIER, Andreas; TERÁN, Luis. How to Develop Mobile Applications with Web-Technologies. In: **Universitas Friburgensis**, v. 201, n. 1, 2011.

KASPERBAUER, Marcelo et al. Chronos Mobi: uma aplicação móvel multiplataforma para o gerenciamento de projetos. In: **Revista Brasileira de Computação Aplicada**, v. 5, n. 1, p. 84-97, 2013. Disponível em: < http://www.upf.br/seer/index.php/rbca/article/view/2774 >. Acesso em: 18-Out-2014.

LEITE, Luis Marcos. **Tecnologia da Informação:** Qual o melhor conceito? Disponível em: <http://ogestor.eti.br/tecnologia-da-informacao-melhor-conceito/>. Acesso em: 03 jan. 2015.

MENEZES, Marcus Vinicius Ruas. **Sistemas de Informação e Funcionalidades de Softwares Gerenciadores de Bares e Restaurantes.** 2007. Disponível em: < http://www.repositorio.uniceub.br/handle/123456789/831 >. Acesso em: 04-Set-2014.

MINHACOMANDA. Disponível em: < https://play.google.com/store/apps/details?id=br.com.newgt.mybill >. Acesso em: 19-mar-2015.

MONGODB. Disponível em <http://www.mongodb.org/>. Acesso em: 20-mar-2015.

MONGOOSE. Disponível em <http://mongoosejs.com/>. Acesso em: 20-mar-2015.

NANCE, Cory et al. Nosql vs rdbms-why there is room for both. In: **Proceedings of the Southern Association for Information Systems Conference**. 2013. p. 111-116.

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.

NODEJS. Disponível em: <http://nodejs.org/>. Acesso em: 27-jan-2015a.

_____. Disponível em: <http://nodejs.org/api/http.html#http_http>. Acesso em: 03-fev-2015b.

OEHLMAN, Damon; BLANC, Sebastien. **Pro Android Web Apps:** Develope for Android using HTML5, CSS3, & Java script, Apress, 2011.

PALMIERI, Manuel; SINGH, Inderjeet; CICCHETTI, Antonio. Comparison of cross-platform mobile development tools. In: **Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on**. IEEE, 2012. p. 179-186.

PARKER, Zachary; POE, Scott; VRBSKY, Susan V. Comparing nosql mongodb to an sql db. In: **Proceedings of the 51st ACM Southeast Conference**. ACM, 2013. p. 5.

POLITOWSKI, Cristiano; MARAN, Vinícius. **Comparação de Performance entre PostgreSQL e MongoDB**. 2014. Disponível em: < http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=21132 >. Acesso em: 09-Mar-2015.

PRESSMAN, Roger S. **Engenharia de software:** uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. 780 p. ISBN 9788563308337.

RESTAURANTES. Disponível em: <https://play.google.com/store/apps/details?id=com.akasoft.topplaces>. Acesso em: 19-mar-2015.

RIBEIRO, André; DA SILVA, Alberto Rodrigues. Survey on Cross-Platforms and Languages for Mobile Apps. In: **Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the**. IEEE, 2012. p. 255-260.

RIGODANZO, Alexandro Vicente; PAZ, Carolina. **A importância da informatização em restaurantes.** 2006**.** Disponível em: < http://assesc.edu.br/download/3_jornada_cientifica/importancia_informatizacao_restaurantes.pdf>. Acesso em: 04-Set-2014.

SABBAGH, Rafael. **Scrum:** Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013. 280 p. ISBN 9788566250107.

SCHROEDER, Ricardo; DOS SANTOS, Fernando. **Arquitetura e testes de serviços web de alto desempenho com Node.js e Mongodb**. 2014. Disponível em:< http://www.ceavi.udesc.br/arquivos/id_submenu/787/ricardo_schroeder_versao_final_.pdf >. Acesso em: 13-Jan-2015.

SEBRAE. **TI EM BARES E RESTAURANTES.** Disponível em: <http://sustentabilidade.sebrae.com.br/Sebrae/Sebrae%202014/Estudos%20e%20Pesquisas/2014_04_22_RT_Fev_TIC_TIBaresRestaurantes%20%281%29.pdf>. Acesso em: 02 jan. 2015.

SILVA, Uellisson Lopes da. **Uma revisão sistêmica da literatura sobre desenvolvimento de aplicativos para dispositivos móveis:** Tendências e desafios. 2014. Disponível em: < http://dspace.bc.uepb.edu.br:8080/jspui/bitstream/123456789/3990/1/PDF%20-%20Uellisson%20Lopes%20da%20Silva.pdf >. Acesso em: 13-Out-2014.

SOON, Tan Jin. **QR Code**. 2008. Disponível em: <https://foxdesignsstudio.com/uploads/pdf/Three_QR_Code.pdf>. Acesso em: 25-mar-2015.

TEIXEIRA, Pedro. **Hands-on Node. js.** [S.l.]: Leanpub, 2011.

TILKOV, Stefan; VINOSKI, Steve. Node. js: Using JavaScript to build high-performance network programs. In: **IEEE Internet Computing**, v. 14, n. 6, p. 80-83, 2010.

WARGO, John M. **Apache Cordova 3 Programming**. [S.l.]: Pearson Education, 2013.

WELSH, Matt; CULLER, David; BREWER, Eric. SEDA: An architecture for well-conditioned, scalable internet services. In: **ACM SIGOPS Operating Systems Review**. ACM, 2001. p. 230-243.

XANTHOPOULOS, Spyros; XINOGALOS, Stelios. A comparative analysis of cross-platform development approaches for mobile applications. In: **Proceedings of the 6th Balkan Conference in Informatics**. ACM, 2013. p. 213-220.

YAAPA, Hage. **Express web application development**. [S.l.]: Packt Publishing Ltd, 2013.

ZOMATO. Disponível em: <https://play.google.com/store/apps/details?id=com.application.zomato>. Acesso em: 19-mar-2015.

APÊNDICE A – Arquivo de configuração do Express.js

```
var compression = require('compression')
  , bodyParser = require('body-parser')
  , path = require('path')
  , morgan = require('morgan')
  , rotaEndereco = require('../modulos/end/routes.js')
  , rotaEstabelecimento = require('../modulos/est/routes.js')
  , rotaProduto = require('../modulos/prod/routes.js')
  , rotaComanda = require('../modulos/comanda/routes.js');

module.exports = function(express) {
    var app = express();

    app.use(compression()); // utiliza a compressão de arquivos para
                            //diminuir o tamanho das respostas HTTP.

    app.use(bodyParser.json());
    app.use(morgan('short'));
    app.set('x-powered-by', false); // Desabilita o X-Powered-By das requisições HTTP
                              // por motivos de segurança

    app.set('port', 8000); // configurando a porta que o servidor será executado
    app.set('end', '192.168.1.7'); // endereço ip que o servidor será executado

    /*
    * Abaixo são carregadas as rotas da aplicação
    */
    app.use('/api/end', rotaEndereco);
    app.use('/api/est', rotaEstabelecimento);
    app.use('/api/prod', rotaProduto);
    app.use('/api/com', rotaComanda);

    // rota para os arquivos publicos (site, imagens, etc)
    app.use(express.static(path.join(__dirname, '../www')));

    return app;
}
```

APÊNDICE B – Arquivo de configuração de acesso ao MongoDB utilizando Mongoose

```
var mongoose = require('mongoose')
    , end = 'mongodb://localhost/teste'; // teste é o nome da base de dados

mongoose.connect(end);

// informações ao administrador do sistema sobre o status da conexão
mongoose.connection.once('open', function () {
    console.log('> Conexão do Mongoose foi aberta.')
});

mongoose.connection.on('connected', function () {
    console.log('> Mongoose conectado em: ' + end);
});

mongoose.connection.on('disconnected', function () {
    console.log('> Mongoose disconectado.');
});

mongoose.connection.on('error',function (err) {
    console.error('Ocorreu um erro com o Mongoose. Mensagem: ' + err);
});

process.on('SIGINT', function() {
     mongoose.connection.close(function () {
         console.info('> Mongoose foi encerrado devido ao encerramento da
aplicação!');
        process.exit(0);
     });
});
// fim das informações ao administrador
```