UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOSÉ CARLOS DE MORAES FILHO

SISTEMA *WEB* GERENCIADOR DE PEDIDOS DE REFEIÇÕES

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2016

JOSÉ CARLOS DE MORAES FILHO

SISTEMA *WEB* GERENCIADOR DE PEDIDOS DE REFEIÇÕES

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina Trabalho de Conclusão de Curso, do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. Me. Alessandro Silveira Duarte

CORNÉLIO PROCÓPIO

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Nome da Diretoria
Nome da Coordenação
Nome do Curso

TERMO DE APROVAÇÃO

Sistema *Web* Gerenciador de Pedidos de Refeições

por

José Carlos de Moraes Filho

Este Trabalho de Conclusão de Curso de graduação foi julgado adequado para obtenção do Título de “Tecnólogo em Análise e Desenvolvimento de Sistemas” e aprovado em sua forma final pelo Programa de Graduação em Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná.

Cornélio Procópio, 28/06/2016

_____

Prof. Me. Alessandro Silveira Duarte

_____

Prof. Me. José Antônio Gonçalves

_____

Prof. José Luiz Vilas Boas

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

Dedico este trabalho à minha família.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Me. Alessandro Silveira Duarte, pela sabedoria com que me guiou nesta trajetória e pela disposição em me ajudar.

Aos meus colegas de sala.

A Secretaria do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização deste trabalho.

RESUMO

MORAES, José Carlos de. **Sistema *Web* Gerenciador de Pedidos de Refeições**. 2016. 53 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

O presente trabalho apresenta o desenvolvimento de um sistema *web* para gerenciar pedidos de refeições. Para seu desenvolvimento foi necessário restringir um cenário para dar sentido às escolhas de tecnologias e estratégias, que foram pesquisadas na literatura para justificar a seleção da abordagem. Ao final do trabalho o *software* foi desenvolvido por meio da abordagem responsiva. Para validar a escolha foi realizado um teste de usabilidade que constatou que para as necessidades e para as regras de negócio desse projeto, a estratégia de desenvolvimento escolhida atendeu ao esperado. Contudo, ainda paira a indagação até onde o uso do desenvolvimento responsivo é assertivo.

Palavras-chave: Responsivo. Abordagem Responsiva. Refeições. Dispositivos Móveis.

ABSTRACT

MORAES, José Carlos de. **Web System to Manage Meals Orders**. 2016. 53 s. Undergraduate Thesis (Graduation) – Technology Analysis and Systems Development. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

This paper presents the development of a web system to manage meals orders. For its development it was necessary to restrict a scenario to make sense of technology choices and strategies that were surveyed in the literature to justify the selection of the approach. At the end of the work the software was developed by responsive approach. To validate the choice was conducted a usability test that found for the needs and business rules of this project , the chosen development strategy has met the expected . However, it was still open to question as far as the use of responsive development is assertive.

Keywords: Responsive. Responsive Approach. Meals. Mobile Devices.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE SIGLAS

ACM	*Association for Computer Machinery*
API	*Application Programming Interface*
CSS	*Cascade Style Sheets*
DBAL	*Database Abstract Layer*
EAP	Estrutura Analítica de Projeto
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Enviroment*
IEEE	*Institute of Electrical and Electronic Engineers*
JDK	Java *Development Kit*
MVC	*Model-View-Controller*
ORM	*Object Relational Mapping*
OO	*Oriented Object*
PHP	*Hypertext Preprocessor*
SEO	*Search Engine Optimization*
SDK	*Software Development Kit*
SM	*Scrum Master*
SMS	*Short Message Service*
SO	Sistema Operacional
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
UX	*User Experience*
V&V	Verificação e Validação
XP	*Extremme Programming*
WPF	*Windows Presentation Foundation*

SUMÁRIO

1 INTRODUÇÃO

O desafio de desenvolver produtos que atendam as necessidades de seu público e que acompanham as constantes mudanças e tendências de seu meio e o que o cerca, é inerente ao mercado tecnológico. Um mercado que é capaz de quebrar e construir novos paradigmas e de redefinir os ditames daquilo que é o mais adequado para o momento, sempre com a premissa de que o novo supera o velho, seja na sua finalidade ou seja pelo meio de atingi-la.

Um exemplo claro pode ser visto na evolução da telefonia móvel, que do seu início até meados da 2ª geração, tinha nos celulares como principal função a comunicação ponta-a-ponta. Com o advento do conceito de *smartphones*, passou a agrupar funções mais complexas tais como: o acesso melhorado à *Internet*; jogos *online*; e assim por diante (JORDÃO, 2009).

No atual contexto, o uso de celulares e *tablets* integra a rotina das pessoas como extensão essencial à realização de atividades diversas, de pagamento de contas à compra *online*, ao entretenimento, não se atendo mais à comunicação pura e simplesmente por voz ou mensagens SMS (*Short Message Service)*. Aludindo ao que foi dito anteriormente, o dinamismo com que o mercado de tecnologia se transforma e se renova, aqui se faz presente, e que para acompanhar as necessidades de seus usuários, tudo o que concerne ao atendimento desses indivíduos deve-se atualizar.

Considerando o exposto acima, o presente trabalho estabelece em seu objetivo o desenvolvimento de um sistema capaz de possibilitar a compra de refeições *online* que, atenda aos consumidores que utilizam *desktops* e os que usam dispositivos móveis. E para decidir qual meio utilizar para alcançar essa finalidade, o autor deste trabalho realizou considerações a respeito das tecnologias e das abordagens existentes por meio de uma revisão na literatura.

Contudo, foi necessário delimitar um cenário para dar sentido às considerações, tendo em vista que as situações e as condições distintas de um projeto, ou problema, requerem abordagens pertinentes, e sem considerar essas restrições, o produto estará fadado ao fracasso, ou no mínimo, a circunstância de desperdício e de alto custo.

No sentido de tornar o assunto claro e objetivar a apresentação dos resultados, o conteúdo deste trabalho foi organizado da seguinte maneira: O

Capítulo 2 traz a fundamentação teórica, para a construção do projeto, e nela podem ser encontradas as três estratégias mais difundidas para o desenvolvimento voltado ao mercado *mobile*, além das bases para a justificativa da inclinação por determinada metodologia de desenvolvimento, e técnicas de testes de *software*.

No Capítulo 3, é esmiuçado o sistema proposto e podem ser observadas as escolhas para sua construção, frente às restrições estabelecidas para sua concepção que também são explicadas nessa seção. Fazem também parte dessa seção a exposição de detalhes da implementação do projeto e o seu cronograma.

Na sequência, o Capítulo 4 apresenta as considerações, deste autor, por meio da soma de resultados obtidos e discorre sobre quais foram as limitações encontradas para a realização do que foi proposto.

2 FUNDAMENTAÇÃO TEÓRICA

O foco deste capítulo é de apresentar uma revisão da literatura dos conteúdos relacionados ao objetivo do trabalho. Nesse ínterim, serão ainda apresentadas as três escolhas mais utilizadas atualmente no desenvolvimento de aplicativos para dispositivos móveis: aplicativos nativos; *mobile web apps*; e finalmente *web sites* responsivos.

Cabe ressaltar que para a pesquisa na literatura, foram utilizadas fontes virtuais como, *sites*, *blogs*, os repositórios da IEEE (*Institute of Electrical and Electronic Engineers*) e da ACM (*Association for Computer Machinery*), além de pesquisas em livros.

O conteúdo será abordado com foco na perspectiva de desenvolvimento e um paralelo será traçado entre as três formas de conceber aplicativos móveis. Os pontos considerados relevantes para a comparação são: complexidade do código; custo de implementação; experiência do usuário; e *performanc*e.

2.1 *MOBILE APP* NATIVO

Gokhale e Singh (2014), dizem que desenvolver aplicativos nativos implica na escolha de uma plataforma específica, haja vista que cada sistema operacional – como por exemplo, o iOS, o Android e o Windows – possui características e necessidades distintas do ponto de vista de implementação, o que segundo os mesmos autores, sugerem que para atender à uma demanda maior de usuários, um grande esforço deve ser despendido, a fim de entregar aplicativos que atendam o maior número de plataformas possíveis.

Nesse sentido, uma empresa que deseja produzir um aplicativo com código nativo deve-se preocupar com seu público-alvo, ao ponto de restringir o alcance a um grupo específico de usuários, como por exemplo, usuários de Android. Ou então, possuir uma equipe multidisciplinar com desenvolvedores capazes de construir projetos em várias plataformas, o que ainda limitaria o escopo de entrega, a não ser que exista uma empresa que detenha profissionais habilitados em todas as plataformas existentes, o que seria um caso muito particular e, sobretudo, custoso (CHARLAND e LEROUX, 2011).

As dificuldades relacionadas a essa escolha são infindáveis. Pense num único projeto com previsão de lançamento mundial em pelo menos três plataformas: Android, iOS e Windows. Os times de desenvolvimento devem entregar o projeto conforme um cronograma que considere todas as especificidades e dificuldades possíveis de cada uma das equipes, para isso, mesmo que seja considerado um cenário ótimo onde nenhum membro da equipe tenha que se ausentar, existiriam ainda como variáveis o nível de habilidade e velocidade de desenvolvimento que cada grupo teria. Enfim, utilizar essa estratégia traria um alto risco de fracasso, tão quanto, um alto custo até o final do projeto.

Contudo, desenvolver código nativo atualmente, possui vantagens quando se trata em acessar recursos nativos, como geolocalização, acelerômetro, câmera, entre outros (GOKHALE e SINGH, 2014). Mesmo em casos onde são consideradas tecnologias híbridas, como PhoneGap, que já possuem acesso a alguns recursos específicos do Sistema Operacional (SO), o código nativo se sobressai em relação ao desempenho. Isso acontece porque geralmente ele é compilado, diferentemente de linguagens interpretadas como o *Javascript*, (CHARLAND e LEROUX, 2011), que são base para aplicativos híbridos.

Além disso, a escolha pelo nativo traz algumas características de interface de usuário que as tecnologias *web* ainda não são capazes de prover. Tenha como exemplo a propriedade *position* do CSS (*Cascade Style Sheets*) que no iOS não suporta o valor “*fixed*”, (CHARLAND e LEROUX, 2011). Contudo, sobre as diferenças de capacidade de acesso a recursos dos dispositivos entre aplicativos nativos e *mobile web apps*, Charland e Leroux (2011) ainda dizem, que se um navegador ainda não possui acesso a algum recurso nativo, não significa a impossibilidade disso acontecer futuramente, apenas não foi desenvolvido ainda.

É amplamente encontrado na literatura que a utilização de um software vai além de suas funções e que usá-lo pode impactar diretamente no estilo de vida de seus usuários, o que implica que a experiência que o usuário tem ao utilizá-lo deve ser agradável e capaz de motivá-lo a fazer uso mais vezes.

Charland e Leroux (2011) modularizam a experiência do usuário (*User Experience – UX*) em duas categorias: Contexto – que são elementos externos ao aplicativo e que não podem ser alterados, ou controlados, como por exemplo, capacidades de *hardware* e o ambiente em que ele é utilizado; e implementação –

que são elementos que estão sob o domínio da aplicação, e que podem ser manipulados, como *design* e características de integração com a plataforma.

A partir desse conhecimento, pode-se considerar como relevante para a análise comparativa entre aplicativos nativos e *mobile web apps* a perspectiva do desempenho, e como variáveis, o tempo de processamento e a largura de banda utilizada. No caso da largura de banda, ambas as implementações sofrem com o mesmo problema e uma solução comum é reduzir o número e a carga de requisições, (ZAKAS, 2013).

A escolha pelo nativo exige que sejam selecionadas tecnologias e linguagens específicas que atendam à plataforma escolhida. *Apps* construídos para Windows Phone, utilizam a *Software Development Kit* (SDK) específica, que suporta o *framework* .NET e linguagens como Visual Basic .NET, C#, C++ e J#, a IDE (*Integrated Development Enviroment*) Visual Studio entre outras tecnologias como, *Siverlight* e .NET, *Windows Presentation Foundation* (WPF) e Silverlight e XAML, (KAMADA *et al*., 2012).

Já os aplicativos desenvolvidos para Android, utilizam a Android SDK que dá suporte para o JDK (Java *Development Kit*) o que garante o desenvolvimento em Java, e pode ser utilizada nas principais IDE's Java, Eclipse e Netbeans, (KAMADA *et al*., 2012).

Os aplicativos desenvolvidos para iOS, são geralmente construídos com a linguagem *Objective-C* que é uma variante entre *Smalltalk* e C, por meio da iOS SDK que provém ao desenvolvedor a IDE XCode, além de outras ferramentas como o *Interface Builder* para a criação de interfaces, o iPhone *Simulator* que simula um ambiente de uso e o *Instruments* que analisa desempenho, consumo de memória, entre outros, (SCHMITT, 2010). Schimitt (2010) ressalta que aplicativos para iOS exigem que o programador possua um computador MAC.

Na subseção seguinte, será abordada a estratégia de desenvolvimento de *mobile web apps* e evidenciadas suas principais características.

2.2 *MOBILE WEB APP*

Segundo Nations (2015), um *web app* pode ser qualquer aplicação que utiliza um navegador como cliente, logo, *mobile web* apps são aqueles desenvolvidos para serem acessados exclusivamente por dispositivos móveis,

(SUMMERFIELD, 2015). Contudo na literatura, alguns autores, como Gokhale e Singh (2014), dividem essa abordagem em duas categorias: a primeira, os *web apps*, que são sistemas executados em *browsers* no lado cliente; a segunda, os híbridos, que são tecnologias que permitem ao desenvolvedor escrever o código da aplicação em HTML, CSS (*Cascade Style Sheet*) e *Javascript*, e no entanto utilizam tradutores de código para acessarem recursos nativos dos dispositivos em que está instalado. O autor deste trabalho abordará o tema de modo abrangente, compreendendo essas tecnologias pela finalidade e, portanto, tratando apenas como *mobile web apps*.

Com a existência de diversas plataformas e linguagens para desenvolvimento de *softwares* para dispositivos portáteis, é inviável que empresas realizem a distribuição em plataformas específicas, (CHARLAND e LEROUX, 2011), e conforme discutido na seção anterior, as possibilidades de fracasso e custo são altos, capazes de impossibilitarem a concepção de qualquer grande projeto.

No sentido de reduzir o abismo entre as plataformas existentes, começaram a serem consideradas como soluções alternativas as tecnologias *web* supracitadas, por serem amplamente portáveis (GOKHALE e SINGH, 2014). Dentre os pontos que exigem maior atenção ao se desenvolver *web apps*, está o desempenho (ZAKAS, 2013), e como diz Gokhale e Singh (2014), o ponto mais insatisfatório está na experiência do usuário que se comparado à aplicações nativas, ainda deixa a desejar.

Ao julgar pelo fato das limitações presentes no acesso à recursos nativos, no caso de aplicativos híbridos, e quanto os *sites* desenvolvidos apenas para atender o público usuário de celulares e afins, estão ainda relacionadas, complexidades quanto ao tratamento de URL, manutenção (manter 1 versão para *mobile* e uma para *desktops* pode ser custoso), e como circunstância da duplicidade do conteúdo, problemas com SEO (*Search Engine Optmization*) (CAELUM, 2016).

Em geral, *web apps* são aplicações desenvolvidas para serem executadas em *browsers* do lado cliente, (NATIONS, 2015), e são desenvolvidos com a tríade HTML, CSS e *Javascript*, o que, em termos de complexidade de código, portabilidade e custo, constitui uma melhor alternativa se comparada com o desenvolvimento de aplicativos nativos, se e somente se, forem consideradas as três variáveis supracitadas.

Entretanto, não significa também, que se considerarmos a variável usabilidade na comparação, a abordagem para *web* se qualifique inviável, afinal, existem API's (*Application Programming Interface*) *Javascript* que agregam nesse quesito, e num cenário que não exija alto grau de interação, a escolha pela *web* passa a ser interessante em ser cogitada.

No cenário de aplicações híbridas, que são a ponte entre o abismo existente entre os *web apps* e os *apps* nativos, (Gokhale e Singh, 2014), existem ferramentas capazes de oferecer produtos com altos padrões de usabilidade, com um *look and feel* muito próximo do esperado na solução nativa, como por exemplo, o *Phonegap*.

O *Phonegap* é uma tecnologia *open-source* que traduz os códigos de HTML5, CSS3 e *Javascript*, para uma linguagem portável entre as plataformas que suporta, o que garante acesso à recursos nativos como acelerômetro, agenda, câmera, geolocalização, entre outros, (GUINTHER, 2011). Como mostra a Figura 1 apresentada a seguir:

Figura 1 – Acesso à recursos de plataformas nativas pelo *Phonegap*
Fonte: (GUINTHER, 2011)

De um modo geral, *web apps* são construídos essencialmente da tríade HTML, CSS e *Javascript*. No subcapítulo seguinte, será apresentado o desenvolvimento responsivo, suas características, vantagens e desvantagens, a fim de concluir a base levantada para respaldo à escolha decidida para o presente projeto.

2.3 *WEB SITE* RESPONSIVO

A variedade de dispositivos e telas de diversos tamanhos exige da *web* a capacidade de ser flexível e adaptável aos diversos cenários possíveis, e como afirma França (2015), existem aparelhos que possuem características tanto do cenário *mobile* quanto *desktop*, e dentre as dificuldades existentes em entregar ao usuário um *site* que atenda a esses dois extremos, o mesmo autor ainda cita: As medidas fixas; e a caracterização do que é *mobile*, que baseado no que já foi apresentado, pode parecer confuso, como no caso de um computador híbrido que dependendo do modo que é utilizado, pode ser um *notebook*, ou um *tablet*.

Como uma solução de atender inúmeras configurações e formatos de aparelhos, de uma mesma URL (*Uniform Resource Locator*), temos o desenvolvimento responsivo que consiste em utilizar *layouts* fluídos - que não usam medidas fixas, como por exemplo, porcentagem -, *media queries* - um recurso que identifica a resolução da tela e aplica regras de estilização definidas para aquele *breakpoint* específico - e imagens responsivas, (FRANÇA, 2015).

Assim como *mobile web apps*, o cerne da construção de *sites* responsivos é o HTML, o CSS e o *Javascript*, mais a adesão de uma linguagem de programação para conduzir as regras de negócio do lado servidor (CAELUM, 2016). A possibilidade de implementar um único *site* que seja capaz de se adaptar à diversas resoluções de telas, traz consigo vantagens como baixo custo – a manutenção e o desenvolvimento são mais baratos - e manutenção centralizada, e de antemão, são suas desvantagens a *performance* (ZAKAS, 2013), e a usabilidade que passa a possuir necessidades específicas (FRANÇA, 2015), e diferente de *web apps*, ou *native apps*, não são capazes de acessar nenhum recurso do aparelho.

Nesse subcapítulo é concluída a exposição das três estratégias utilizadas atualmente para o atendimento de usuários de dispositivos móveis. A seguinte dá

continuidade à constituição dos fundamentos do trabalho ao abordar métodos de desenvolvimento.

2.4 MÉTODOS DE DESENVOLVIMENTO

Sommerville (2011) relata que processos de desenvolvimento são complexos e exigem rigorosidade em seu cumprimento. Ele ainda diz que a postura atual das empresas e equipes de desenvolvimento está focada no cliente e na capacidade de se adaptar às constantes mudanças, para que as oportunidades de negócio sejam melhores aproveitadas, de modo que algumas organizações se dispõem a substituir “(...) a qualidade e o compromisso com requisitos do software por uma implantação mais rápida do software que necessitam”, (SOMMERVILLE, 2011, p. 38).

O autor ainda diz que a prática de desenvolvimento geralmente segue um modelo de processo, que podem ser expandidos e adaptados. Esses paradigmas de processo podem ser divididos em duas categorias, os métodos tradicionais, e os métodos ágeis.

Dos métodos tradicionais, temos como destaque: O modelo em cascata, que representa distintamente as fases de especificação, desenvolvimento, validação e evolução. Nele, uma etapa só é executada, após o término da anterior; Já o modelo incremental, intercala as atividades das fases de especificação, desenvolvimento e validação, e são entregues incrementos para compor a totalidade do sistema; E o desenvolvimento orientado a reuso, que como o próprio nome denuncia, reutiliza componentes já existentes em novos projetos.

Quanto aos métodos ágeis, o autor define que são baseados no desenvolvimento incremental, contudo os incrementos são pequenos e as entregas ocorrem em maior frequência, (SOMMERVILLE, 2011). Como exemplo de método ágil, temos o *Extreme Programming* (XP), que numa ligeira apresentação, tem os requisitos mapeados por estórias, e os programadores desenvolvem em pares e se orientam por testes, (SOMMERVILLE, 2011).

Ainda sobre metodologias ágeis, temos o *Scrum* que segundo Schwaber e Sutherland (2013) é um *framework* que possibilita a resolução de problemas complexos e adaptativos, com entrega constante de produtos de alto valor, pois se

baseia no conhecimento empírico evoluído iterativamente, como fonte de aperfeiçoamento da previsibilidade e do controle de riscos.

O *Scrum* possui três pilares: Transparência – Todos os membros da equipe devem compartilhar da mesma linguagem e conhecimento geral relacionado ao projeto, inclusive a definição de “pronto”; Inspeção – Os artefatos e o progresso devem ser inspecionados regularmente, a fim de detectar variações; Adaptação – Se o responsável pela inspeção detectar a necessidade de ajustar o que está sendo produzido, a equipe deve ser capaz de se organizar para que os desvios sejam corrigidos, (SCHWABER e SUTHERLAND, 2013).

Os autores explanam que o *framework* é composto de eventos, papéis e artefatos que estruturam como deve ser utilizado. São papéis do *Scrum*: O *Product Owner* (PO)*,* responsável por gerenciar o *Backlog* do Produto, o que significa que apenas ele pode expressar e ordenar os itens do *Backlog* do Produto, e garantir o valor e entendimento dos itens pelo time de desenvolvimento; O Time de Desenvolvimento é uma equipe multifuncional, auto organizável, e são os responsáveis pela entrega das versões que incrementam o produto; E finalmente, o *Scrum Master* (SM), que deve garantir à equipe a compreensão e aplicação do *Scrum*, e resolver impedimentos que estejam atrasando o projeto, (SCHWABER e SUTHERLAND, 2013).

Para que essa estrutura de processo faça sentido, todos os eventos devem acontecer, salvo situações onde existam adaptações dessa estrutura para atender a uma necessidade específica de uma equipe. Todos os eventos possuem um tempo definido e são eles que delineiam o ciclo de vida dessa metodologia. Segundo Schwaber e Sutherland (2013), são eventos do *Scrum*:

- ***Sprint*:** Esse é o principal evento do *framework*, que pode ter a duração de até quatro semanas, e é ao final dela que um incremento deve ser entregue “pronto”. Uma nova *Sprint* só inicia com o término da anterior, e não são realizadas alterações que possam comprometer o objetivo da *Sprint*, nem tão quanto, as metas de qualidade diminuem, e o escopo pode ser renegociado entre o PO e o *Dev Team*; Os autores ainda relatam que uma *Sprint* só pode ser cancelada pelo PO em situações onde o objetivo esteja obsoleto e dadas as circunstâncias do momento, já não representa mais o interesse das partes interessadas;
- **Reunião de Planejamento da *Sprint*:** Nesse evento, o PO, o *Scrum Master* e o *Dev Team*, se reúnem para discutir os objetivos e a duração da *Sprint*. O

SM tem o papel de manter a reunião dentro do tempo previsto, que pode variar conforme a duração da *Sprint*, por exemplo, as de um mês, devem ter duração máxima de oito horas, enquanto o Dono do Produto debate os objetivos do evento, o Time de Desenvolvimento seleciona os itens do *Backlog* do Produto que comporão o *Backlog* da *Sprint*;

- **Reunião Diária:** Esse evento ocorre diariamente e possui duração de no máximo 15 minutos, e recomenda-se que para reduzir a complexidade e evitar divagações, ela acontece no mesmo horário e local, de preferência, em pé. Nela são discutidos a evolução desde a reunião do dia anterior e o que cada membro da equipe pretende realizar durante o dia corrente. Esse é o momento em que os impedimentos são relatados e o *Scrum Master* intervém para solucionar os problemas;
- **Revisão da *Sprint*:** Acontece logo após o término da *Sprint* para inspecionar o incremente produzido e, caso for necessário, adaptar o *Backlog* do Produto. Essa reunião tem caráter informal e possui o intuito de interagir os envolvidos no projeto e motivá-los;
- **Retrospectiva da *Sprint*:** É o momento que o Time se inspeciona e elabora um plano de melhorias para serem aplicadas na próxima *Sprint*; Ela acontece após a Revisão da *Sprint* e antes da próxima *Sprint*; A duração predeterminada para esse evento é de três horas para uma *Sprint* de um mês.

Cabe ressaltar a importância da definição de “pronto” para um *Sprint*, pois é a partir dela, que pode-se observar os resultados do que foi entregue como incremento. A Figura 2 demonstra o ciclo de vida de um *Sprint*:

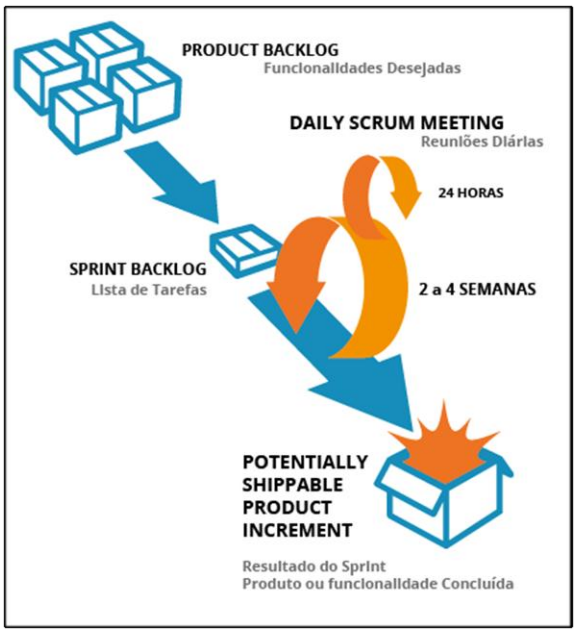

Figura 2 – Metodologias Ágeis – *Scrum*
Fonte: BRQ (2015).

Para o acompanhamento e monitoramento do que é desenvolvido, algumas ferramentas como gráficos *burn down*, ou *burn up* e o quadro do *Kanban* – que evidencia os estados de atividades – identificam se a equipe está de acordo com o cronograma, BRQ (2015), veja na Figura 3 um exemplo do quadro *Kanban* e na Figura 4 um exemplo de *burn down*:

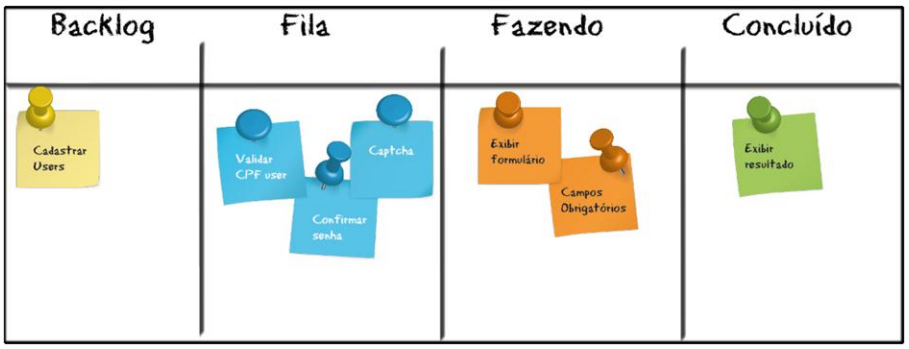

Figura 3 – Quadro do *Kanban*
Fonte: BRQ (2015)

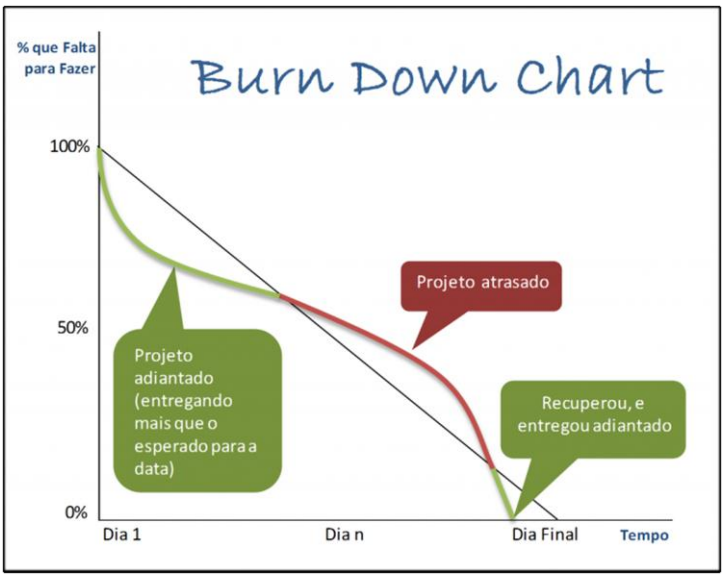

Figura 4 – *Burn Down Chart*
Fonte: BRQ (2015)

O uso do *Scrum* depende fundamentalmente das relações entre as pessoas envolvidas, afinal segue o os princípios do Manifesto Ágil, que segundo Beck *et. al.* (2001), considera indivíduos e interações mais relevantes que processos e ferramentas, onde o foco é o *software* em funcionamento ao invés de documentação abrangente, de modo que o dinamismo da equipe seja capaz de responder às mudanças.

O conteúdo exposto nessa subseção é primordial para compreender a motivação deste autor ao escolher o método de desenvolvimento que foi aplicado para desenvolver este projeto, e que será apresentado no Capítulo de Desenvolvimento. O subcapítulo seguinte, apresenta brevemente o conceito de teste que também auxiliará na justificativa pelas decisões tomadas quanto ao desenvolvimento deste trabalho de diplomação.

2.5 TESTE DE *SOFTWARE*

Pressman (2006, p. 289) diz que “(...) Teste é um conjunto de atividades que podem ser planejadas antecipadamente e conduzidas sistematicamente”, e seu objetivo é identificar defeitos antes de ser utilizado pelo usuário final (PRESSMAN, 2006 e SOMMERVILLE, 2011), e Sommerville (2011) ainda diz que é a garantia de que um *software* atende a seu propósito. Dijkstra *et al.* (1972 *apud* SOMMERVILE,

2011, p. 145) complementa com a afirmação de que “Os testes podem mostrar apenas a presença de erros, e não a sua ausência”.

Segundo os autores Pressman (2006) e Sommerville (2011), a prática de testes é um aspecto de um vasto processo de verificação e validação (V&V). A verificação corresponde a um grupo de atividades que traz garantias de que o programa desenvolvido implementa uma função específica, e validação está atrelada a asserção de que o produto desenvolvido corresponde aos requisitos do cliente. Boehm (1979, *apud* SOMMERVILLE, 2011) ainda sintetiza esse conceito ao dizer:

- ‘Validação: estamos construindo o produto certo?’
- ‘Verificação: estamos construindo o produto da maneira certa?’

A principal finalidade dos processos de verificação e validação é determinar o nível de confiança do produto desenvolvido frente ao cliente (SOMMERVILLE, 2011), onde confiança pode ser entendida “como a probabilidade do programa operar sem apresentar falhas, em um dado contexto de uso e em um intervalo de tempo determinado” (KOSCIANSKI, 2007, p. 338).

Compreender a importância da realização de testes é fundamental para garantir que o produto entregue possua qualidade. A seguir, inicia-se o Capítulo de desenvolvimento, onde poderão ser encontradas as escolhas para esse trabalho de diplomação, e entre essas escolhas, os tipos de testes que foram realizados.

3 DESENVOLVIMENTO DO PROJETO

Da elaboração à implementação do projeto, foi construído um cenário como base para justificar as escolhas de abordagem, tecnologias e metodologia de desenvolvimento. Isso se deve ao fato de que a ampla gama de recursos tecnológicos existentes possui restrições quando se resolve problemas distintos e que não existe uma solução universal. Desconsiderar essa variedade de alternativas seria negligenciar a responsabilidade de desenvolver um produto que atendesse aos objetivos aqui propostos.

Nesse sentido, são delineadas as seguintes restrições para o projeto:

- O sistema deve ser portável tanto à *browsers* de computadores com resolução superior, ou igual à 1280px por 800px, quanto à dispositivos móveis com resolução inferior à supracitada (ABASOV, 2012);
- O custo de implementação deve ser baixo, pois considera tanto aporte financeiro, quanto mão de obra, escassos - se enquadram neste espectro, microempresas e *startups* de tecnologia;
- Por motivos de tempo e custo, deve-se optar por tecnologias que sejam altamente difundidas, a fim de, na necessidade de terceirizar manutenções, a oneração seja a menor possível.

Cabe ressaltar que para compreender a magnitude deste trabalho de diplomação, serão apresentados neste capítulo, os resultados obtidos em testes de usabilidade.

A partir deles, poderão ser observados como a abordagem de implementação escolhida, justifica seus prós e contras já apresentados na literatura. São ainda assuntos deste capítulo, a apresentação de todo o ferramental e método de desenvolvimento utilizados.

Na ordem de apresentação dos assuntos temos: No subcapítulo 3.3 serão apresentados os recursos utilizados, e qual a estratégia de desenvolvimento escolhida; na continuidade, o subcapítulo 3.4 justifica o método de desenvolvimento utilizado a fim de se apropriar da explicação do subcapítulo 2.4 para definir a base da elucidação, sendo responsabilidade desse subcapítulo, apenas expor em como a escolha pôde contribuir durante as etapas de construção do projeto; e finalmente, a subseção 3.5 traz ao cerne do trabalho, a documentação relacionada à análise e

implementação do que foi desenvolvido, como resultados dos testes realizados, diagrama de modelo de domínio, diagrama objeto-relacional, entre outros.

3.1 RECURSOS UTILIZADOS E ESTRATÉGIA DE DESENVOLVIMENTO

Á luz de todo o referencial apresentado no capítulo 2 e com a compreensão do cenário e restrições do projeto, foi decidido que a melhor das alternativas para a construção do sistema é o desenvolvimento responsivo. Embora seja utilizado a arquitetura MVC (*Model-View-Controller*), o paradigma de desenvolvimento adotado neste projeto será o estruturado.

A seleção da abordagem responsiva aconteceu pela abrangência que possibilita ao atender usuários de dispositivos móveis e *desktops*, pelo baixo custo frente às outras estratégias quando considerados os pontos delimitadores do projeto, e pela redução de complexidade e centralização da manutenção.

A centralização do código implementado simplifica correções, melhorias e evolução do sistema, mesmo que isso possibilite um grau de personalização reduzida, e inviabilize o acesso a recursos nativos dos aparelhos (GOKHALE e SINGH, 2014).

A seguir são mostrados os principais recursos utilizados, e aqui não pretende-se trazer à baila a origem e definição de HTML, CSS e *Javascript* (considere também as *toolkits* baseadas em *Javascript*), pois entende-se que aqui devem ser expostas as ferramentas que trouxeram algum diferencial para esse projeto.

3.1.1 *CodeIgniter*

Codeigniter é um *framework* MVC (*Model-View-Controller*) desenvolvido por Rick Ellis em 2006 pela empresa que é fundador, a EllisLab, e que surgiu da *ExpressionEngine* (ELLISLAB, 2016).

Atualmente está na versão 3.0.*, e entre seus recursos e vantagens estão: Tamanho reduzido (atualmente possui 2Mb de tamanho); Compatibilidade com diversos padrões de hospedagem, o que facilita a implantação; Sem regras de código muito restritivas, o que permite ao desenvolvedor usar convenções de nomes que lhe forem mais convenientes; Pouca configuração de ambiente; Possui uma curva de aprendizagem pequena (CODEIGNITER, 2016).

O motivo de escolher essa ferramenta ante inúmeras outras disponíveis no mercado, está relacionado intimamente ao custo de utilizá-la e de hospedar os sistemas desenvolvidos com ela, a julgar pelo fato de que entre as restrições do projeto, a implementação com baixo custo é primordial.

Neste sentido pode-se entender que a economia de investimento se faz tanto na aquisição de mão-de-obra – que devido ao fato de possuir uma pequena curva de aprendizagem, possibilita alterar a equipe de desenvolvimento sem grandes perdas - , quanto na contratação de serviços de hospedagem.

3.1.2 *Doctrine*

Frameworks ORM (*Object Relational Mapping*) consistem no mapeamento entre um modelo de dados relacional, para um modelo orientado a objetos de modo que estabeleçam uma camada intermediária entre a lógica de uma aplicação e seu SGBD (Sistema de Gerenciamento de Banco de Dados) e abstração de acesso aos dados (MAYUMI, 2016).

Mayumi (2016) ainda declara que entre as principais vantagens da utilização dessa técnica estão, o baixo acoplamento, que assegura que ao mudar um SGBD, algumas poucas configurações são necessárias para reaproveitar a estrutura já mapeada, e o aumento de produtividade que é assegurado pelo fato de não ser necessário escrever SQL (*Structured Query Language*) manualmente, nem tão pouco reescrever todo o código no caso de uma mudança de SGBD.

Sua base foi construída sob o DBAL (*Database Abstraction Layer*) e idealmente tem a ambição de utilizar conceitos já existentes em outros ORM’s, como Hibernate para Java (DOCTRINE, 2016). O mapeamento das entidades, e de restrições de atributos é realizada por anotações, conforme exemplifica a imagem a seguir:

```php
<?php
namespace sir\entities;

use Doctrine\ORM\Mapping as ORM;


/**
 * @ORM\Entity
 * @ORM\Table(name="Cardapio")
 */
class Cardapio
{
    /**
     * @ORM\Id @ORM\Column(type="integer")
     * @ORM\GeneratedValue
     * @var integer
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=100, nullable=false)
     * @var string
     */
    private $nome;
```

Figura 5 – Mapeamento de Objeto pelo *Doctrine*
Fonte: Autoria própria

Embora existam outros ORM's no mercado, o *Doctrine* foi escolhido para o mapeamento de objetos, por utilizar um padrão de anotações ao qual o presente autor já estava habituado. Além disso, quando se considera tempo de desenvolvimento, manutenção e custo, *Doctrine* pode ser uma boa escolha por ser estável desde 2006 (DOCTRINE, 2016).

3.1.3 *PhpStorm*

O mínimo necessário para se desenvolver uma aplicação é prover um editor de texto, geralmente com suporte à linguagem utilizada para otimizar o tempo gasto escrevendo linhas de códigos. Em PHP (*Hypertext Preprocessor*) existem inúmeros editores com suporte, contudo, pra produzir esse trabalho foi pensado em algo que otimizasse não apenas a escrita, mas que trouxesse consigo outras facilidades para agilizar a produção.

Após pesquisar algumas IDE's (*Integrated Development Enviroment*) para PHP, foi escolhido o PHPStorm pelas seguintes características: Prevenção de erros *on-the-fly* (no ato da escrita do código aparecem mensagens sugerindo a possibilidade de erro); Dica de código; Refatoração; Possui integração com sistemas de controle de versão; Acesso à linha de comando do Sistema Operacional (SO), à bancos de dados; Inteligente assistente de código que verifica o código escrito, rearranja e formata, e indica ajustes para melhorar o que foi escrito (JETBRAINS, 2016).

3.2 ANÁLISE E IMPLEMENTAÇÃO

Este subcapítulo tem o intento de mostrar os artefatos gerados durante a criação do projeto, da fase de planejamento ao teste do sistema. Sendo assim, é dada a seguinte organização do conteúdo: Inicialmente serão apresentadas as regras de negócio do sistema, e seu diagrama de casos de uso; na sequência, é mostrado o diagrama objeto-relacional e são discutidos os resultados dos testes propostos; em seguida são expostas algumas telas do *software*; e por fim, é apresentado o cronograma do projeto.

3.2.1 Regras de Negócio

O sistema de pedidos de refeições *online* desenvolvido para esse trabalho deve propiciar aos usuários as seguintes operações relatadas no Quadro 1:

USUÁRIO	REGRAS DE NEGÓCIO
Cliente	• A capacidade de selecionar estabelecimentos e filtrá-los por cidade ou por tipo de estabelecimento; • Visualizar os pratos e cardápios dos estabelecimentos e filtrá-los por tipo de prato ou pelos cardápios; • Adicionar e excluir itens do carrinho de compras; • Fechar pedidos; e • Visualizar os pedidos já realizados, e os que estão em andamento;
Atendente	• Atender pedidos.
Gerente	• Atender pedidos; e • Adicionar, editar ou excluir cardápios.
Proprietário	• Atender pedidos; • Adicionar, editar ou excluir cardápios; • Adicionar, editar ou excluir estabelecimentos; e • Adicionar, editar ou excluir usuários;

Quadro 1 – Resumo das Regras de Negócio
Fonte: Autoria própria

Para mais detalhes, consulte o Apêndice E.

A subseção seguinte dá continuidade na explicação do projeto e nela poderão ser vistas o diagrama de casos de uso, o objeto-relacional e o de domínio das classes.

3.2.2 Análise do Sistema

Sobre as regras de negócio, a Figura 6 ilustra os casos de uso do sistema, e em comparação com a proposta de implementação mostrada na defesa da proposta, o módulo de pagamento foi retirado. Entende-se que nesse tipo de negócio, o pagamento *online* traz consigo alguns problemas como: Caso o pedido não seja atendido, o grau de complexidade para estorno do valor pago é muito grande, para um negócio que exige muita agilidade e lida com transações de baixo valor:

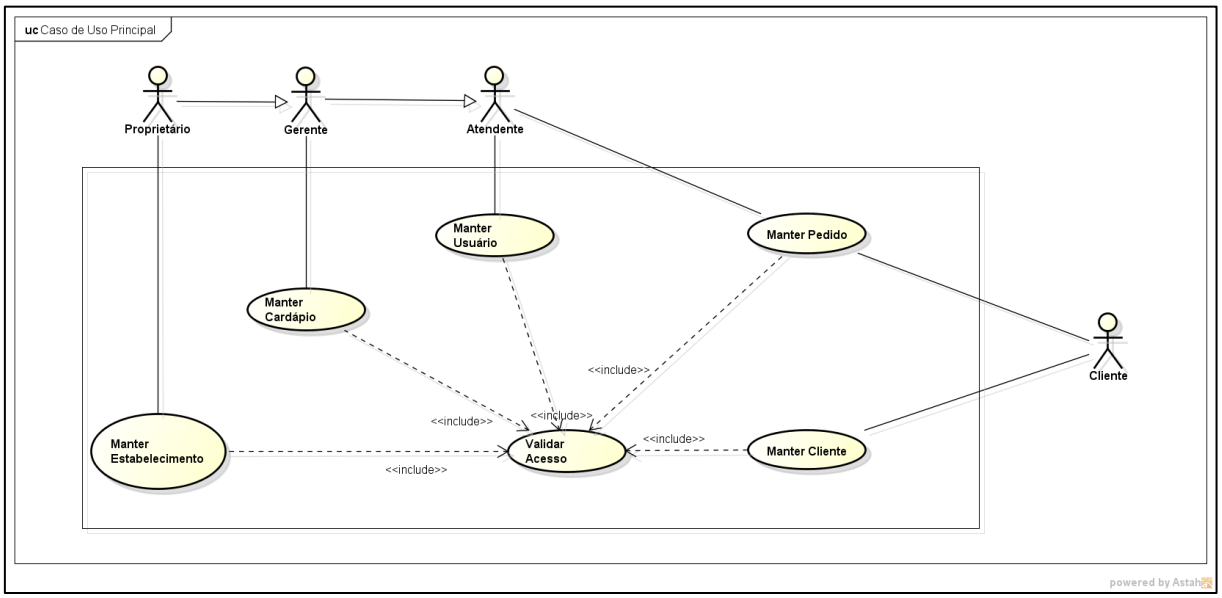

Figura 6 – Diagrama de Caso de Uso Principal
Fonte: Autoria própria

Dentre os casos de uso expostos na Figura 6, o “Manter Pedido” é o principal, a julgar pelo fato de relacionar todos os tipos de usuário e de ser o mecanismo que justifica a existência do sistema. Para melhor apresenta-lo, foi elaborado um caso de uso estendido, que é ilustrado na Figura 7:

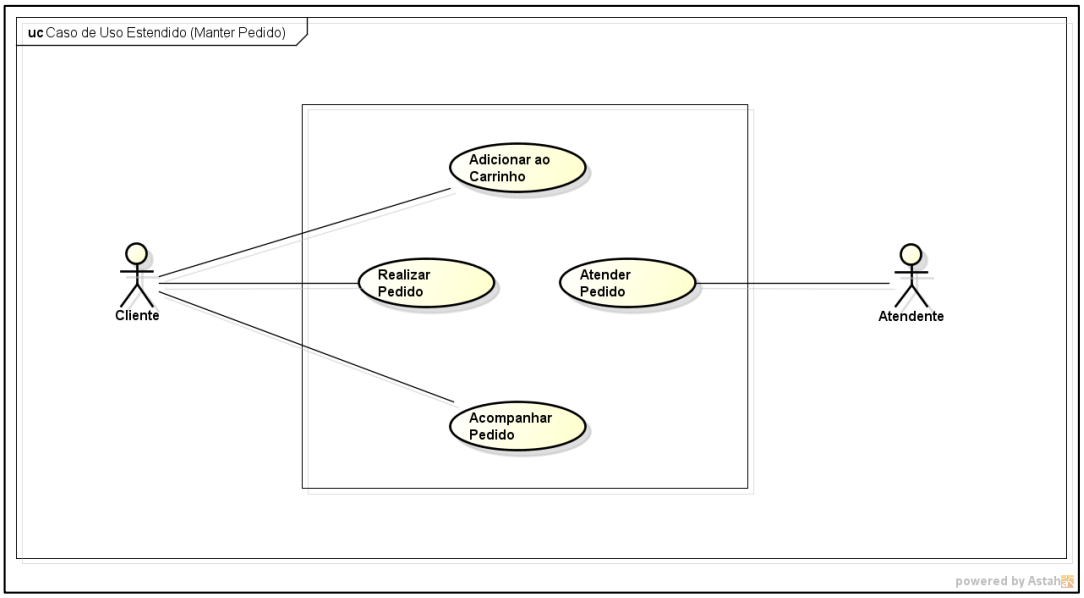

Figura 7 – Caso de Uso Estendido (Manter Pedido)
Fonte: Autoria própria

Observe que nesse caso de uso estendido, apresentado na Figura 7, não aparecem os usuários do tipo gerente e proprietário. Isso acontece devido ao fato de que dentre os usuários administrativos, o atendente é o que possui menos responsabilidades, e como evidenciado na Figura 6 todos os demais usuários herdam suas responsabilidades.

Baseado nas necessidades levantadas foi elaborado o Diagrama Objeto-Relacional para a construção da camada de dados, apresentado na Figura 8:

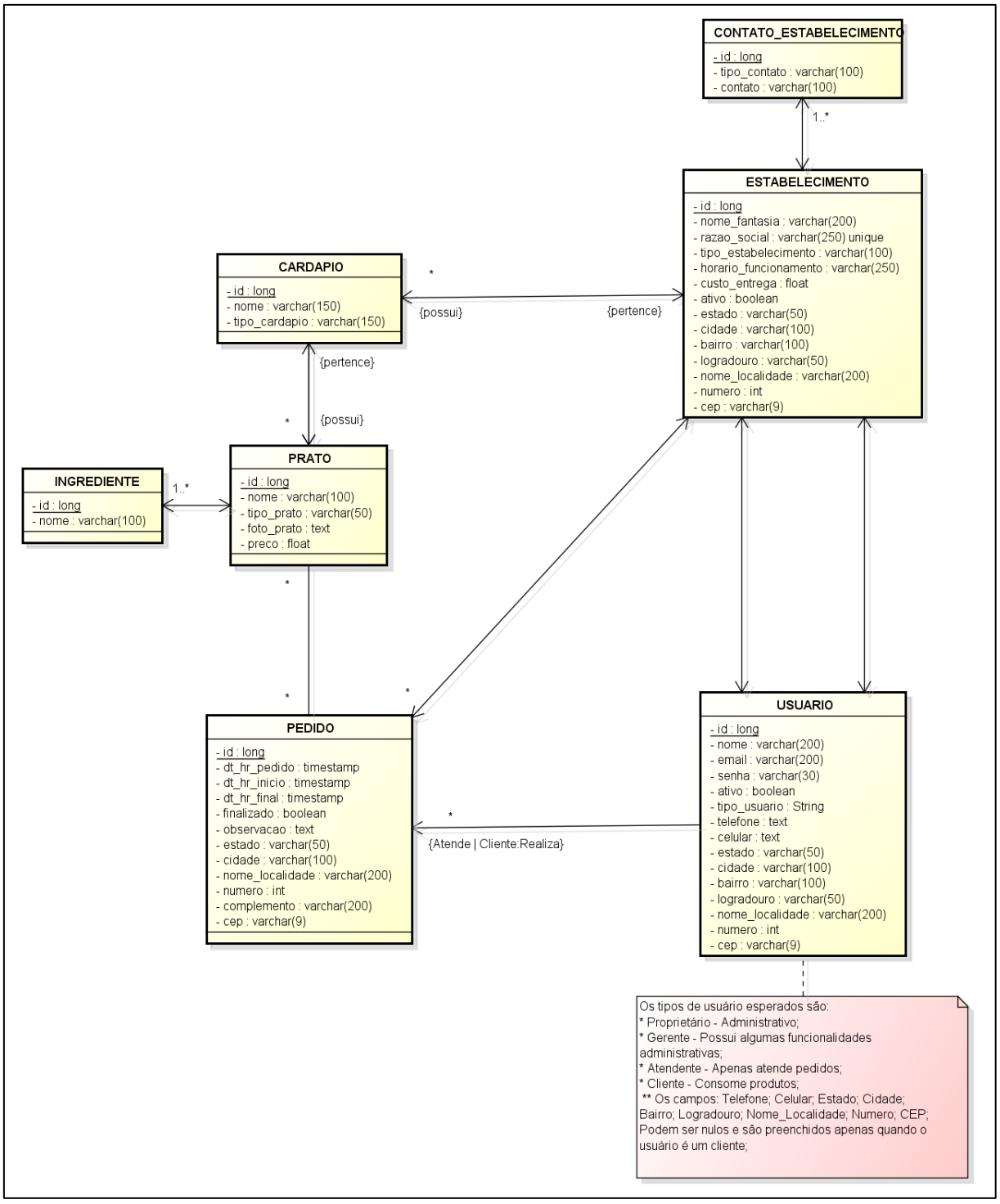

Figura 8 – Diagrama de Objeto-Relacional
Fonte: Autoria própria

No intuito de melhor guiar o desenvolvimento do projeto, foi gerado como artefato da análise do sistema um diagrama de modelos de domínio que segundo Larman (2007) é um dos mais importantes modelos em análise Orientada a Objeto (OO) por esclarecer alguns conceitos em um domínio e por servir de base para outros modelos. O diagrama gerado pode ser visto na Figura 9:

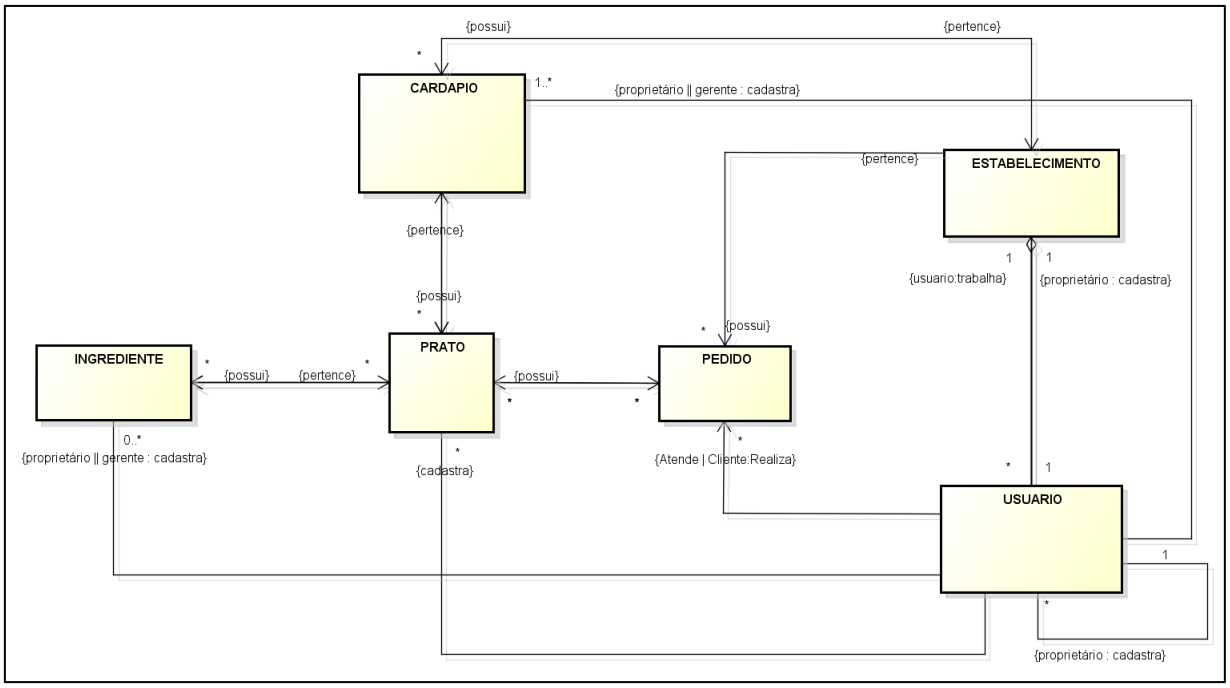

Figura 9 – Diagrama de Modelo de Domínio
Fonte: Autoria própria

Como pode-se perceber, o diagrama exposto na Figura 9, oferece uma perspectiva conceitual (LARMAN, 2007) e evidencia as classes conceituais e suas associações. Isso em um processo que exige agilidade, possibilita enfoque apenas em como se relacionam as entidades e quais são elas. Essa abordagem permite nortear o desenvolvimento e reduz o tempo gasto com documentação.

3.2.3 Metodologia e Cronograma

Partindo das ponderações da Seção 2.4 sobre a importância do uso de processos de *software*, muito foi discutido sobre metodologias ágeis. Nesse ínterim, um enfoque foi dado ao *Scrum*, pois foi a metodologia de desenvolvimento selecionada para a realização deste trabalho. Ressalta-se que para adequar às condições e recursos disponíveis para o projeto, houve uma inclinação pela sugestão de Pagotto *et al* (2016).

Pagotto *et al* (2016) propuseram algumas alterações na utilização do *framework*, que consideram características do PSP (*Personal Software Process*) e retira algumas atividades do processo, como as reuniões diárias. A finalidade dessa adaptação, sobretudo, está em reduzir o tempo de construção de um projeto com apenas um desenvolvedor, e não reduzir a qualidade do que é entregue. Na Figura 10 é apresentada a configuração da estrutura do *Scrum* Solo:

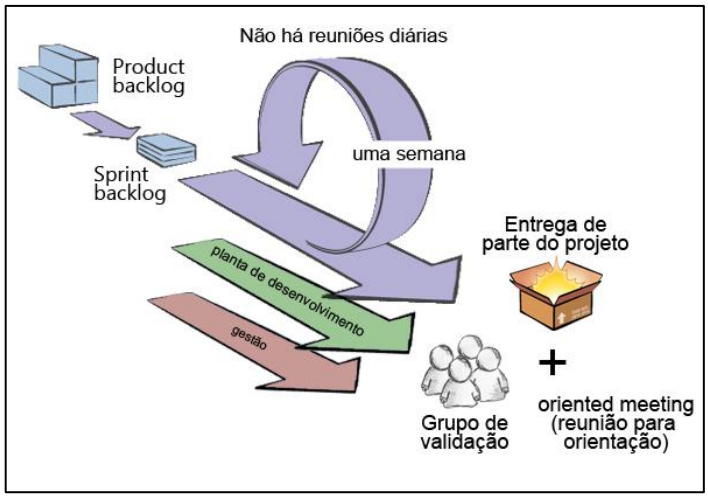

Figura 10 – *Scrum Solo*
Fonte: PAGOTTO *et al* (2016, p. 3)

É possível perceber na Figura 10, que o tempo máximo de desenvolvimento para essa abordagem, é de uma semana. Sua estrutura ainda conta com documentações essenciais previstas na planta de desenvolvimento e na gestão: Diagrama de entidade-relacionamento, de sequência, de casos de uso, de classes; E estrutura analítica do projeto (EAP), cronograma e planilha de custos, respectivamente (PAGOTTO *et al*, 2016). A validação fica sob a responsabilidade do grupo de validação, formada pelo orientador e, ou cliente.

Considerando as premissas de agilidade e qualidade propostas pela adesão do *Scrum* Solo, seu uso foi amplamente explorado na realização deste projeto de desenvolvimento.

Para mapear as atividades que deveriam ser executadas, foi utilizada uma EAP (Estrutura Analítica de Projeto), mostrada na Figura 11.

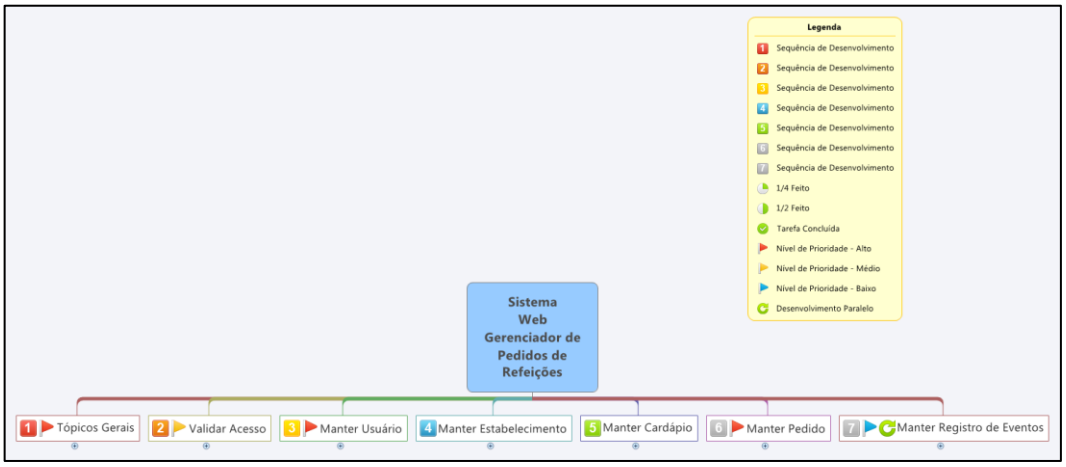

Figura 11 – Estrutura Analítica do Projeto
Fonte: Autoria própria

Como pode ser visto na Figura 11, os requisitos foram identificados em uma EAP que usou de base os casos de uso – identificados nos itens 2 ao 7. A partir dessa agregação foi gerado um *product backlog*, o qual pode ser visto em um fragmento no Quadro 2 a seguir:

PRODUCT BACKLOG				
Id	**Itens**	**Módulo**	**Front / Back - End**	**Feito**
1	Formulário de Login - Front –Estrutura	Validar Acesso	Front	Ok
2	Formulário de Login - Front - Validações (js)	Validar Acesso	Front	Ok
3	Formulário de Login - Front – CSS	Validar Acesso	Front	Ok
4	Formulário de Login - Back – Model	Validar Acesso	Back	Ok
5	Formulário de Login - Back – Controller	Validar Acesso	Back	Ok
6	Cadastro de Usuários Adm. - Front – Estrutura	Manter Usuário	Front	Ok
7	Cadastro de Usuários Adm. - Front - Validação (js)	Manter Usuário	Front	Ok
8	Cadastro de Usuários Adm. - Front – CSS	Manter Usuário	Front	Ok
9	Cadastro de Clientes - Front – Estrutura	Manter Usuário	Front	Ok
10	Cadastro de Clientes - Front - Validação (js)	Manter Usuário	Front	Ok

Quadro 2 – *Product Backlog* do Projeto
Fonte: Autoria própria

Cabe observar que este acompanhamento ainda permitia identificar quais itens já foram atendidos. Embora tenham sido expostos dez itens, o *product backlog* era composto por cento e sete deles, que foram distribuídos em *Sprints* na ordem do Quadro 3:

Sprint	Módulo	Total Tarefas	Tarefas
1	Validar Acesso	5	1 – 5
2	Manter Usuário	8	6 - 11; 15; 16
3	Manter Usuário	8	12 - 14; 17 – 21
4	Manter Usuário	10	22 – 31
5	Manter Usuário	10	32 – 41
6	Manter Estabelecimento	8	42 – 49
7	Manter Estabelecimento	7	50 – 56
8	Manter Cardápios	8	57-64
9	Manter Cardápios	7	65 – 71
10	Manter Pedido	3	72 – 74
11	Manter Pedido	4	75 – 78
12	Manter Pedido	3	79 – 81
13	Manter Pedido	5	82 – 86
14	Manter Pedido	2	87 – 88
15	Manter Pedido	2	89 – 90
16	Manter Pedido	3	91 – 93
17	Manter Pedido	3	94 – 96
18	Manter Pedido	6	97 – 102
19	Manter Pedido	4	103 – 106
20	Manter Registro Eventos	1	107

Quadro 3 – Divisão das *Sprints*
Fonte: Autoria própria

O Quadro 3 deixa claro que cada *Sprint* estava focada em um determinado caso de uso e seus respectivos itens mapeados pela EAP e pelo *product backlog* – consulte o Apêndice F para maiores informações.

Observa-se também um total de 20 ciclos de desenvolvimento. Ademais, cabe para o momento evidenciar que a cada início de um novo ciclo esse quadro era preenchido com o objetivo de determinar quais itens deveriam ser atendidos nele.

Para um melhor acompanhamento do projeto, foi utilizado um quadro para receber a entrada de quantos itens haviam sido desenvolvidos em uma *Sprint*, que pode ser vista no Quadro 4:

T. Tarefas	Prioridade	Tarefas Faltantes	Tarefas Feitas	Previsto
107	0	107	0	0
102	1	102	5	5
94	2	97	5	8
86	3	88	9	8
76	4	77	11	10
66	5	69	8	10
58	6	57	12	8
51	7	47	10	7
43	8	42	5	8
36	9	38	4	7
33	10	35	3	3
29	11	32	3	4
26	12	27	5	3
21	13	32	-5	5
19	14	34	-2	2
17	15	36	-2	2
14	16	39	-3	3
11	17	42	-3	3
5	18	30	12	6
1	19	21	9	4
0	20	8	13	1

Quadro 4 – Lançamento de Atividades
Fonte: Autoria própria

A alimentação dessa planilha tinha ainda a responsabilidade de manter o gráfico de *burndown* atualizado, movendo a linha de tarefas faltantes que pode ser analisada no Gráfico 1:

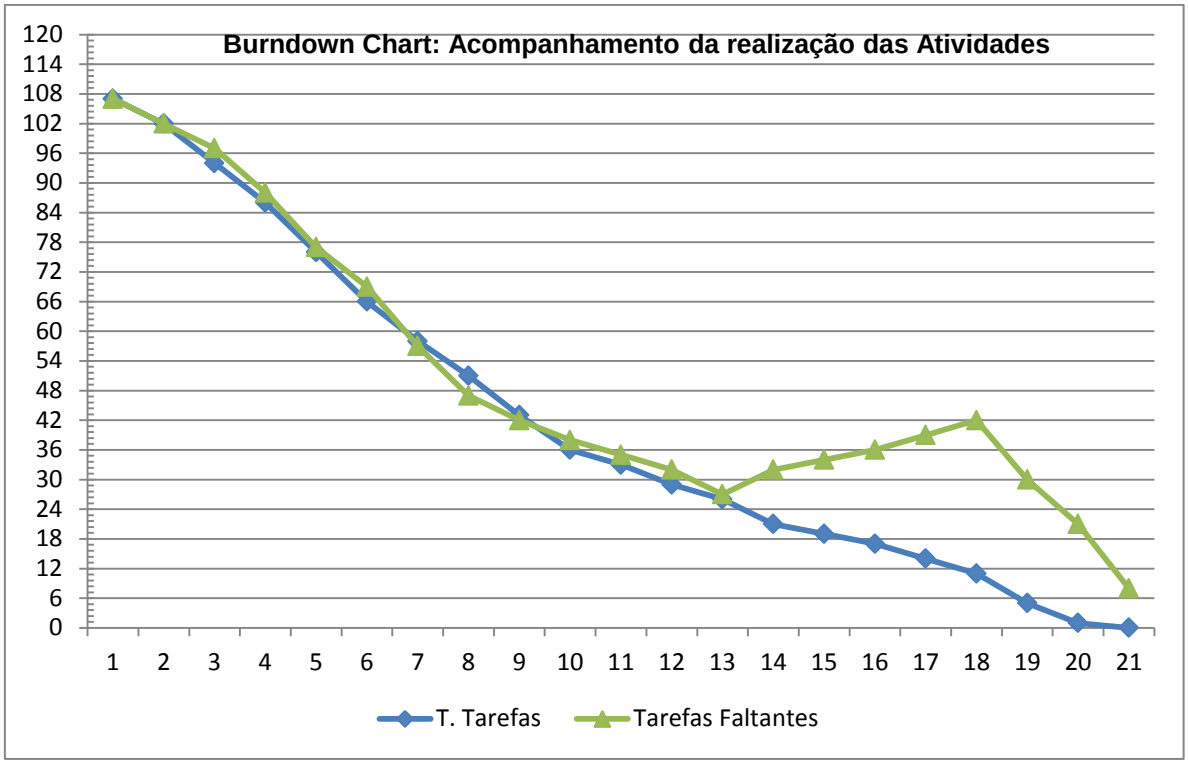

Gráfico 1 – *Burndown Chart* do projeto
Fonte: Autoria própria

Observe que no gráfico existem duas linhas: Uma verde que corresponde às tarefas faltantes, e é essa linha que exibe o rendimento real; E uma azul que define uma previsão de realização. É possível perceber que da *Sprint* treze a dezoito - que são justamente as que possuem atividades relacionadas ao caso de uso “Manter Pedido” - houve um atraso, que pode ser justificado pela quantidade de atividades com alto grau de complexidade em um mesmo incremento.

Esse resultado qualificaria uma nova fragmentação das tarefas. Além do mais, veja que o gráfico supracitado não evidencia intervalos de tempos por data, mas considera a entrega conforme esperado na *Sprint*. Para suprir essa necessidade, o autor deste trabalho se guiou por um cronograma mapeado nos moldes de uma EAP, veja a Figura 12:

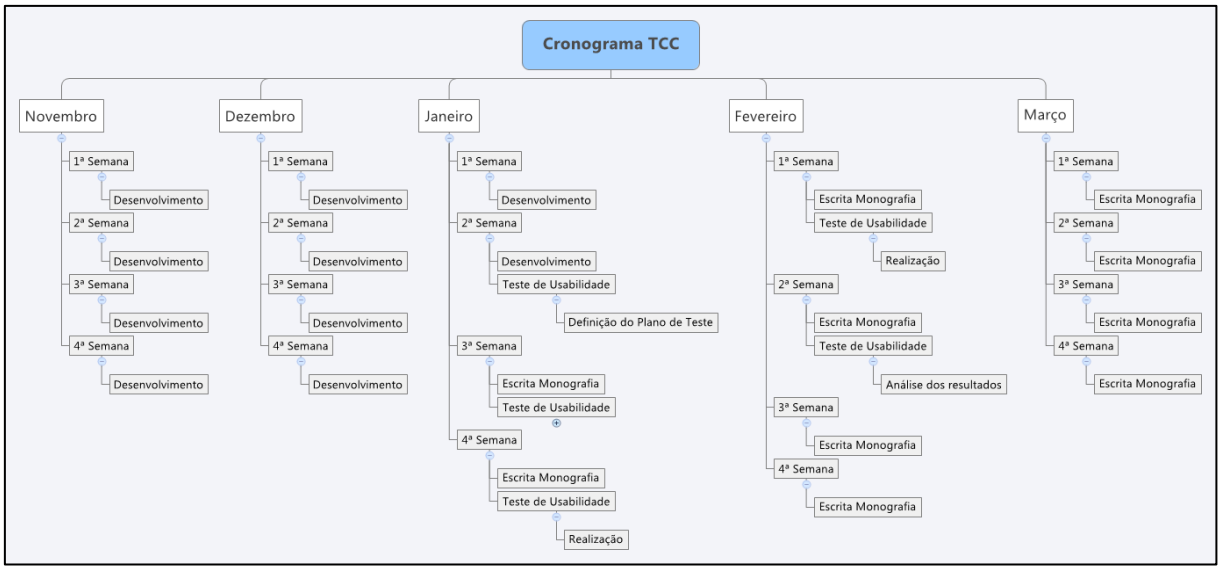

Figura 12 – Cronograma do Projeto
Fonte: Autoria própria

No subcapítulo seguinte, serão relatados os testes realizados no sistema, e quais os resultados obtidos deles.

3.2.4 Teste de *Software*

O conteúdo apresentado na seção 2.5 nos mostrou que um projeto de *software* de qualidade, deve ser submetido a etapas de testes, haja vista que não existe um meio de testar todas as possibilidades por conta da complexidade e magnitude compreendidas (DIJKSTRA, 1972 *apud* SOMMERVILLE, 2011), pode se entender como responsabilidade da equipe, ou responsável pelos testes, de que sejam capazes de compreender em quais cenários o produto é seguro (perceber quais domínios são devidamente atendidos), e onde pode ser corrigido, ou melhorado.

Com a intencionalidade de atender a essas expectativas, mas atento para a restrição de custo do projeto (apresentada no início do capítulo 3), e à premissa de que o sistema deve ser acessível por dispositivos móveis com resolução inferior a 1280px por 800px – resolução de 12 polegadas, comum em *netbooks* – (ABASOV, 2012), foram pensados e utilizados duas modalidades de testes: O teste unitário; e o teste de usabilidade.

3.2.4.1 Teste Unitário

A aplicação do teste unitário foi dada pela utilização de uma biblioteca do CodeIgniter que possibilita a criação de uma classe Controller que implementa um

objeto da classe de teste unitário. Para realizar a verificação do resultado é necessário acessar a URL (*Uniform Resource* Locator) que invoca o método principal (que testa os retornos). Veja na Figura 13, um exemplo de teste realizado neste projeto que testou a alteração de um pedido em estado de espera, para o estado de produção:

```php
class Test_Controller extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->library('unit_test');
        $this->load->model(array('estabelecimento_model', 'pedido_model'));
    }

    public function proccessUnitTest()
    {
        $id = 87;
        $test = $this->testEnviarPedidoProducao($id);
        $this->unit->run($test, 'is_true', 'Teste de envio de pedido para produção.');
        echo $this->unit->report();
    }

    /*@PASSED*/
    public function testEnviarPedidoProducao($idPedido = false){
        if(!$idPedido){
            return false;
        }

        $date = date('Y/m/d H:i:s');

        return $this->pedido_model->atualizar(array('data_hora_pedido_inicio'=>$date), $idPedido);
    }
```

Figura 13 – Realização de teste unitário
Fonte: Autoria própria

Observe que foi desenvolvido um método chamado "*processUnitTest*" que realiza a chamada ao método que altera o estado de um pedido de "não atendido" para "em produção". Veja também que o método "*run*" do objeto da classe de teste, exige que sejam passados três parâmetros formais. O primeiro parâmetro é o retorno da execução do teste a ser testado; O segundo é a indicação do valor esperado; e o terceiro é uma descrição do teste.

Durante o andamento do projeto, todos os métodos das classes *Models* foram cobertos pelos testes. Nessa etapa foram criados os testes primeiro, e quando o resultado era positivo, integravam a totalidade do programa (PRESSMAN, 2011).

3.2.4.1 Teste de Usabilidade

Para a realização do teste de usabilidade, foram utilizados um computador *notebook*, e dois dispositivos móveis que estão enquadrados nos *breakpoints* mais comuns (ABASOV, 2012), sendo respectivamente:

- *Notebook* Lenovo B490: Windows 7; Processador Intel *Core* I3 2.40GHz; e tela de 14 polegadas com resolução de 1366px por 768px;
- *Smartphone* Motorola Moto G: Android 5.1.1; Processador *Quad-Core* 1.4GHz; e tela de 5 polegadas com resolução de 1280px por 720px;
- Samsung *Galaxy Pocket Neo*: Android 4.1.2; Processador 1 Core 850 MHz; tela de 3 polegadas com resolução de 320px por 240px.

Figura 14 – Lenovo B490
Fonte: Lenovo (2016)

Figura 15 – Moto G 16GB
Fonte: Motorola (2016)

Figura 16 – Samsung Pocket
Fonte: Samsung (2016)

As máquinas foram conectadas a um servidor local, por intermédio de um roteador – veja o Apêndice A. Os cinco participantes que realizaram a simulação do ambiente utilizaram todos os dispositivos para avaliar o sistema de uma perspectiva mais geral.

Após a simulação, cada um dos testadores foi direcionado a responder um questionário qualitativo que enumera os valores das respostas em uma escala de 0 (zero) a 5 (cinco), sendo zero a nota mais baixa possível e cinco que indicava a nota mais alta possível (Apêndice C). A seguir é apresentado no Quadro 5, o resultado das respostas dos usuários e pode-se ainda, visualizar uma média para o resultado de cada questão:

Questão	Usuário					Média
	1	2	3	4	5	
Nível de satisfação quanto ao uso do sistema	3	2	4	3	3	3
Atendimento a proposta do sistema	5	4	4	3	5	4,2
Layouts são intuitivos	3	3	4	3	3	3,2
Organização da informação	4	2	4	3	4	3,4
Rótulos e textos explicativos	5	3	5	5	4	4,4
Mensagens de retorno	3	3	4	3	3	3,2
Facilidade em operar o sistema	3	3	4	4	3	3,4
Presença de falhas (0 é alta, e 5 é baixa)	4	4	3	5	5	4,2
Qualidade da navegação	4	3	3	4	3	3,2
Nível de confiança (você utilizaria esse sistema em um cenário real?)	4	2	4	3	3	3,2

Quadro 5 – Resultados do teste de usabilidade
Fonte: Autoria própria

Com a realização dos testes pôde ser constatado que em nenhum dos pontos avaliados, a média de valores foi inferior a três pontos, mesmo assim, se atentando para a avaliação de cada indivíduo é capaz de perceber que o participante número 2 apresentou um nível de satisfação abaixo do valor médio, e analisando cada uma de suas respostas identifica-se uma inclinação negativa da qualidade da navegação que resulta em um nível de confiança inferior à média.

A existência dessa dissonância leva a crer que isso pode ser resultado da falta de domínio do participante e que há uma abertura para futuros testes com aumento do espaço amostral pra confirmar se esse desvio é restrito, ou se sua frequência é grande o suficiente para atestar uma baixa qualidade do produto.

A seguir são demonstradas algumas telas do sistema as quais os participantes navegaram durante o processo de testes. Na Figura 17, podemos ver a página inicial com todas as disposições possíveis dentro do cenário de testes e na Figura 18, o painel administrativo. Na ordem da esquerda pra direita são exibidas as resoluções do Samsung *Galaxy Pocket Neo*, do Moto G na orientação retrato, e finalmente a representação da tela no *notebook*:

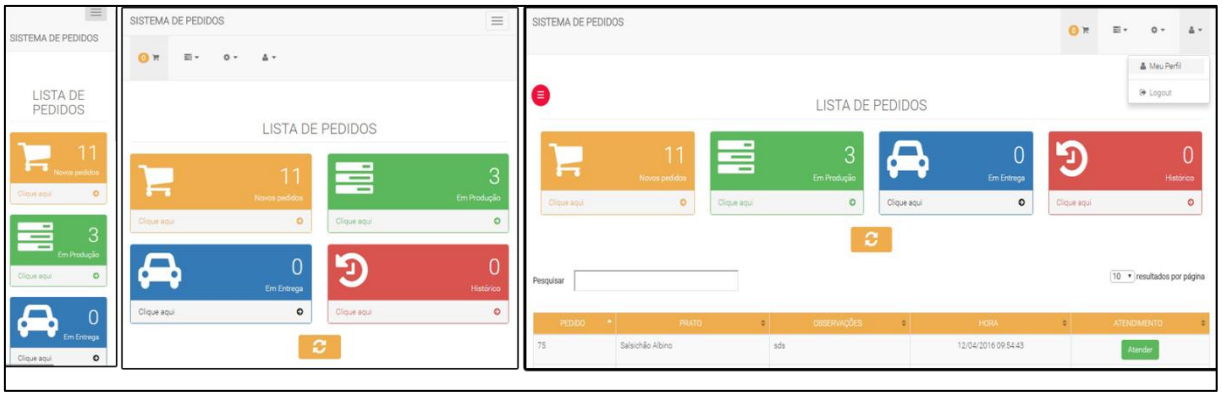

Figura 17 – Painel administrativo em três dimensões
Fonte: Autoria própria

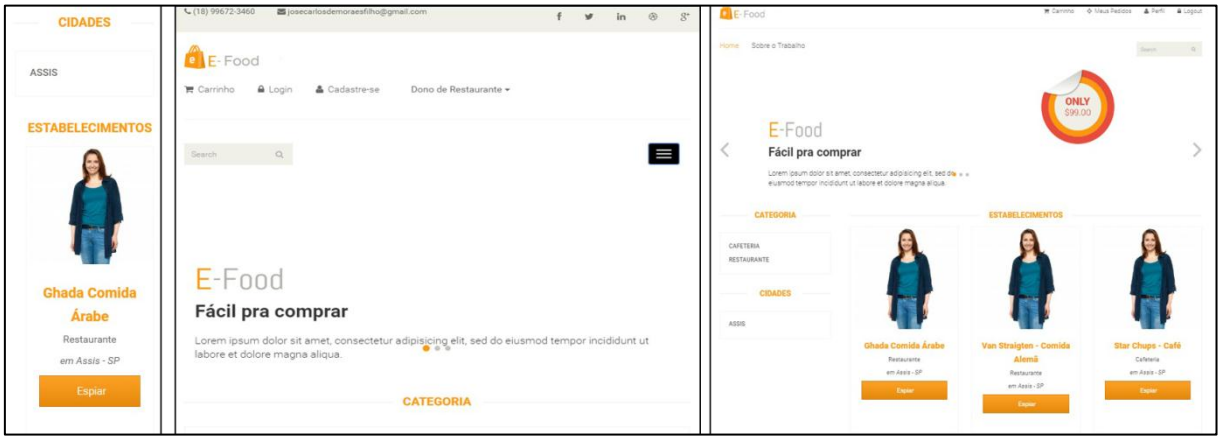

Figura 18 – Página Inicial em Três Dimensões
Fonte: Autoria própria

A seguir serão realizadas reflexões a respeito do trabalho realizado, e dos resultados obtidos.

4 CONSIDERAÇÕES FINAIS

A muito que se discutir a respeito de desenvolvimento para atender ao mercado *mobile*, pois a problemática da escolha de abordagem, presente nesse projeto, é recorrente em outros. Além disso, a existência de várias estratégias, tecnologias e métodos para a resolução de um problema implicam na necessidade de não assumir um determinado meio como universal.

Analisando o caminho percorrido para obter o produto proposto nesse trabalho, observa-se que as variáveis do ambiente, se relacionam intimamente com as estratégias e processos para a construção deste, e que uma escolha equivocada pode tornar o fracasso a possibilidade mais iminente. Por isso, como processo de desenvolvimento, foi experimentado o *Scrum* Solo, que simplificou e tornou mais ágil a entrega dos incrementos, mesmo com as dificuldades relacionadas a eles.

Dentre as principais dificuldades existentes, a organização e utilização do processo foi a mais marcante, pois entende-se que um único projeto, é insuficiente para amadurecê-lo aos moldes de seus usuários. Cabe ressaltar que nesse mesmo sentido, determinar um cronograma sem uma base histórica, acaba por trazer riscos quanto a entrega dos incrementos.

Como a proposta desse trabalho foi desenvolver um sistema de compra de refeições *online* que pudesse atender tanto usuários *desktop* quanto de dispositivos móveis, e especificamente para as restrições do cenário utilizado, o desenvolvimento responsivo foi capaz de atender ao que foi proposto. Contudo, as ressalvas de limitações de usabilidade que foram constatadas na subseção 3.2.4.1, deixam ainda a indagação de até onde a abordagem responsiva é capaz de ser uma solução assertiva.

Espera-se que esse trabalho contribua de alguma forma para compreender a magnitude na escolha da melhor abordagem para o atendimento de usuários *mobile*, bem como, sirva como um norte para projetos que venham lidar com premissas e restrições similares àquelas apresentadas aqui.

REFERÊNCIAS

ABASOV, Mike. **Global Screen Size Diversity Infographic: sizing up man's new best friend**. Publicado em: 20 dez 2012. Disponível em:< http://dev.mobify.com/blog/global-screen-size-diversity/>. Acesso em: 10 jan. 2016.

BECK, Kent; BEEDLE, Mike; BENNEKUM, Arie Van; COCKBURN, Alistair; CUNNINGHAM, Ward; FOWLER, Martin; GRENNING, James; HIGHSMITH, Jim; HUNT, Andrew; JEFFRIES, Ron; KERN, Jon; MARICK, Brian; MARTIN, Robert C.; MELLOR, Steve; SCHWABER, Ken; SUTHERLAND, Jeff; THOMAS, Dave. Disponível em: <http://www.agilemanifesto.org/iso/ptbr/>. Acesso em: 18 set. 2015.

BRQ. **Metodologias Ágeis – Scrum**. Disponível em:< http://www.brq.com/metodologias-ageis/>. Acesso em: 20 set. 2015.

CAELUM. **Desenvolvimento *Web* com HTML, CSS e Javascript**: curso WD-43. Disponível em: < www.caelum.com.br/apostilas>. Acesso em: 10 fev. 2016.

CODEIGNITER. *Why codeigniter?.* Disponível em: <https://www.codeigniter.com/>. Acesso em: 10 fev. 2016.

DOCTRINE. *About.* Disponível em: < http://www.doctrine-project.org/about.html>. Acesso em: 10 fev. 2016.

ELLISLAB. *A Brief History of CodeIgniter.* Disponível em: < https://ellislab.com/codeigniter>. Acesso em: 10 fev. 2016.

FRANÇA, Fabiano dos Santos. Web Design Responsivo: caminhos para um site adaptável. **Interfaces Científicas – Exatas e Tecnológicas**. Aracaju, v. 1, n. 2, p. 75 – 84, jun. 2015. Disponível em: <https://periodicos.set.edu.br/index.php/exatas/article/view/2220>. Acesso em: 20 fev. 2016.

GUINTHER, Marcel. **Desenvolvimento Mobile Multiplataforma com Phonegap**. 17 Outubro 2011. Disponível em: <http://www.mobiltec.com.br/blog/index.php/desenvolvimento-mobile-multiplataforma-com-phonegap/>. Acesso em 22 set. 2015.

KAMADA, Terumi P. B.; CARPEJANI Jayson; ISHIDA, Celso Yoshikazu; GOMES, Márcio Luiz Rossato; NEVES, Luiz Antônio Pereira. **Análise das Plataformas de**

Desenvolvimento Mobile Aplicados na Área Educacional, Usando Android e Windows Phone. Estudo de Caso: Aplicativo Planetas no Windows Phone. v. 10, Nº1, Julho, 2012. Disponível em: <http://seer.ufrgs.br/renote/article/viewFile/30916/19896>. Acesso em: 20 set. 2015.

JETBRAINS. PHPStorm. Disponível em: <https://www.jetbrains.com/phpstorm/>. Acesso em: 11 fev 2016.

JORDÃO, Fábio. **História: A Evolução do Celular**. TECHTUDO, 22 mai. 2009. Disponível em: http://www.gta.ufrj.br/grad/10_1/movel/evolucao.html. Acesso em: 12/03/2016.

KOSCIANSKI, André. Qualidade de *software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de *software* – 2. ed. – São Paulo: Novatec, 2007.

LARMAN, Craig. Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo; tradução Rosana Vaccare Braga ... [*et al.*] – 3. ed. – Porto Alegre: Bookman, 2007.

LENOVO. B490 (Laptop). Disponível em: < http://support.lenovo.com/br/en/products/laptops-and-netbooks/lenovo-b-series-laptops/lenovo-b490-notebook?tabName=Downloads&linkTrack=Mast:SubNav:Support:Drivers%20and%20Software|Drivers%20and%20Software&beta=false>. Acesso em: 18 fev. 2016.

MAYUMI, Priscila. ORM – *Object Relational Mapping.* **Revista Easy .Net Magazine**. n. 28. Disponível em: < http://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158>. Acesso em: 21 fev. 2016.

MOTOROLA. Motorola Moto G 16GB. Disponível em: < http://www.motorola.com.br/Moto-G-da-Motorola/moto-g-2nd-gen-br.html>. Acesso em: 18 fev. 2016.

NATIONS, Daniel. *Web Applications: What is a Web Application?*. Disponível em: <http://webtrends.about.com/od/webapplications/a/web_application.htm>. Acesso em 22 set. 2015.

PAGOTTO, Tiago; FABRI, José Augusto; L'ERARIO, Alexandre; GONÇALVES, José Antônio. 2016. Disponível em:

<https://engenhariasoftware.files.wordpress.com/2016/04/scrum-solo.pdf> . Acesso em: 12 Jan. 2016.

PRESSMAN, Roger S. Engenharia de Software; tradução Rosângela Delloso Penteado – 6. ed. – São Paulo: McGraw-Hill, 2006.

PRESSMAN, Roger S. Engenharia de Software: uma abordagem profissional; tradução Ariovaldo Griesi – 7. ed. – Porto Alegre: McGraw-Hill, 2011.

SAMSUNG. Samsung *Galaxy Pocket* Neo. Disponível em: < http://www.samsung.com/br/support/model/GT-S5312RWBZTA>. Acesso em: 18 fev. 2016.

SCHIMITT, Rodrigo. Os Primeiros Passos para Desenvolver IOs. 22 ago 2010. Disponível em: <http://www.devmac.com.br/2010/08/os-primeiros-passos-para-desenvolver-para-ios/>. Acesso em: 22 set. 2015.

SOMMERVILLE, Ian. Engenharia de Software; tradução Ivan Bosnic e Kalinga G. de O. Gonçalves – 9. ed. – São Paulo: Pearson Prentice Hall, 2011.

SUMMERFIELD, Jason. *Mobile Web Sites vs. Mobile App (Application): Which is Best for Your Organization?*. Disponível em: <http://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/>. Acesso em: 21 set. 2015.

APÊNDICE A – Plano de Teste

1 PROPÓSITO DO TESTE

A realização do teste de usabilidade para esse trabalho, tem a pretensão de entender se, para o cenário definido na proposta do projeto, a abordagem de desenvolvimento de um sistema *web* responsivo tem o potencial de suprir a demanda de usuários de dispositivos móveis, e desse modo, identificar os pontos falhos nessa estratégia.

2 DECLARAÇÃO DOS PROBLEMAS

1. A navegação foi fluída (não houveram dificuldades, ou entraves que dificultassem a realização das tarefas)?
2. Para a finalidade do sistema proposto, a estratégia responsiva supre as necessidades de seus usuários?
3. A *performance* é aceitável, ou é um impedimento para o uso do sistema?

3 DEFINIÇÃO DO PERFIL DO USUÁRIO

Os participantes devem ter entre 18 a 30 anos de idade, e possuir experiência em utilizar computadores e celulares para a realização de compras *online*.

4 METODOLOGIA

O teste será organizado da seguinte forma:

1. O usuário preencherá o formulário de Definição de Perfil do Usuário (Apêndice B);

2. Realizará as tarefas propostas no Apêndice C a partir do *notebook*, seguido do Moto G, e finalmente o *Samsung Galaxy Pocket Neo*;

3. Responderá ao questionário sobre o sistema que pode ser encontrado no Apêndice D.

5 AMBIENTE DE TESTE

Será utilizado um servidor de aplicação local conectado a um roteador que permitirá o acesso pelas máquinas. Sendo elas:

- *Notebook* Lenovo B490: Windows 7; Processador Intel *Core* I3 2.40GHz; e tela de 14 polegadas com resolução de 1366px por 768px;
- *Smartphone* Motorola Moto G: Android 5.1.1; Processador *Quad-Core* 1.4GHz; e tela de 5 polegadas com resolução de 1280px por 720px;
- Samsung *Galaxy Pocket Neo*: Android 4.1.2; Processador 1 Core 850 MHz; tela de 3 polegadas com resolução de 320px por 240px.

Cabe ressaltar que não caberá ao usuário que realizará o teste, a configuração do ambiente acessado.

APÊNDICE B – Questionário para Identificação do Perfil do Participante

Este questionário tem como único objetivo, identificar sua aptidão para realizar o teste, e desse modo, não existem respostas certas, ou erradas. Ressalta-se que a questão 4 pode ser assinalada em mais de um item.

1. **Qual a sua idade? _____ anos.**
2. **A quanto tempo utiliza computadores? _____**
3. **Já comprou pela *Internet*? ( ) Sim ( ) Não**
4. **Comprou por qual dispositivo? ( ) Computador ( ) Celular ( ) Tablet**

APÊNDICE C – Lista de Tarefas

A seguir, são apresentadas as tarefas que devem ser realizadas para fundamentar as respostas que serão dadas para avaliar o *software*.

1. Cadastre-se
2. Procure algo para comprar
3. Adicione 2 itens ao carrinho
4. Acesse a página do carrinho
5. Retire 1 item do carrinho
6. Finalize a compra (adicione uma observação)
7. Vá para meus pedidos
8. Faça *logout*

Após a realização desta lista de tarefas, responda ao questionário de avaliação.

APÊNDICE D – Questionário de Avaliação

Este questionário de avaliação tem o intento de compreender quais são suas percepções sobre o sistema, como por exemplo, o nível de confiança passado e a fluidez na navegação. Para preenche-lo assinale o campo corresponde com a sua nota para o ponto em questão, onde 0 (zero) corresponde a menor nota possível, e 5 (cinco) a maior nota possível.

Questão	0	1	2	3	4	5
Nível de satisfação quanto ao uso do sistema						
Atendimento a proposta do sistema						
Layouts são intuitivos						
Organização da informação						
Rótulos e textos explicativos						
Mensagens de retorno						
Facilidade em operar o sistema						
Quantidade de falhas do sistema (0 é alta e 5 é baixa)						
Qualidade da navegação						
Nível de confiança (você utilizaria esse sistema em um cenário real?)						

Quadro 6 – Questionário Qualitativo
Fonte: Autoria Própria

APÊNDICE E – Detalhamento das Regras de Negócio

Proprietário

1. Deve se cadastrar e aceitar os termos de uso sistema;
    a. Registrar se o usuário está ativo; data e horário que se cadastrou;
2. Cadastrar:
    a. Nome;
    b. Telefone fixo, celular (necessário cadastrar pelo menos um telefone pra contato);
    c. E-mail;
    d. CPF;
    e. Senha;
    f. Status (ativo | inativo);
    g. Se o usuário cancelar seu cadastro, seu registro se qualifica como inativo e automaticamente todos os estabelecimentos associados a ele e seus respectivos usuários, são qualificados como inativos;
        i. É registrada a operação em um *log* de eventos;
    h. O acesso dos usuários será validado através do e-mail e senha;

Estabelecimento

1. Nome do estabelecimento;
2. Tipo de estabelecimento (pub; lanchonete; pizzaria; etc.);
3. Telefone (fixo, ou celular);
   Endereço;
    a. Estado;
    b. Cidade;
    c. Tipo de logradouro (Rua; Av.; etc.);
    d. Nome da localidade;
    e. Número;
    f. Cep;
    g. Horário de funcionamento;
    h. Custo pela entrega (opcional);
4. Usuário para administrar o estabelecimento;
    a. Deve haver no mínimo um usuário capaz de cadastrar cardápio (Proprietário, ou Gerente);
    b. Podem haver vários usuários trabalhando em um estabelecimento, porém um usuário não pode trabalhar em mais de um estabelecimento;
    c. Um Proprietário pode cadastrar apenas um estabelecimento, e um estabelecimento deve pertencer apenas à um proprietário;
    d. Um usuário só está ativo, quando vinculado a um estabelecimento, exceto se ele for um cliente;
5. Status (ativo | inativo);

6. Quando o Contratante “excluir” um estabelecimento, o mesmo é qualificado como “inativo” e os usuários (Gerente | Atendente) associados a ele serão, automaticamente, qualificados como “inativos”;
7. Mensagens (*log* das mensagens enviadas no atendimento dos Pedidos);

Usuários

1. Um Usuário Proprietário pode cadastrar outro Usuário de nível menor;
    a. Existem três tipos de usuário do sistema: Proprietário; Gerente; Atendente;
2. Dados cadastrais;
    a. Nome;
    b. Telefone (fixo, ou celular);
    c. E-mail;
    d. Senha de acesso;
    e. Status (ativo | inativo);
    f. Quando o Contratante “excluir” um Usuário (Gerente | Atendente), o mesmo será automaticamente qualificado como inativo;
        i. Usuários inativos podem ser “ativados” novamente quando associados a um estabelecimento; Compõe uma lista de usuários inativos;
        ii. Caso o Contratante queira excluir permanentemente um usuário, ele deve acessar a lista de Usuários inativos e excluir o Usuário selecionado;
3. Um Proprietário possui uma página para gerenciamento dos usuários e dos estabelecimentos e seus cardápios;
    a. Um Gerente possui uma página para gerenciamento de cardápios do estabelecimento o qual é responsável;
    b. Um Proprietário também pode atualizar seus dados;

Cardápio

1. Nome;
2. Tipo de cardápio (ex.: comida oriental; sucos);
3. Um estabelecimento pode possuir muitos cardápios, um cardápio deve pertencer a um estabelecimento;
4. Um cardápio possui muitos pratos, e um prato pertence a um cardápio;

Prato

1. Nome do Prato;
2. Tipo do Prato (pizza; frutos do mar; comida oriental; etc.);
3. Foto do Prato;
4. Tamanho (ex.: P, M, G);
5. Preço;
6. Ingredientes;

    a. Um prato tem muitos Ingredientes e um Ingrediente pode estar presente em muitos pratos;
7. Um estabelecimento pode ter muitos Cardápios e um Cardápio pode pertencer a muitos estabelecimentos;
    a. Um Usuário (Proprietário | Gerente) pode copiar um Cardápio de outro Estabelecimento desde que esteja associado ao Contratante;

Ingredientes

1. Nome;
2. Quando um ingrediente for excluído, deve ser registrado o evento em um *log* de eventos;

Cliente

1. O cliente pode utilizar o sistema livremente, contudo, só será necessário validar o acesso nas etapas de confirmação do pedido;
    a. O cliente que possuir cadastro deve fazer o *login*;
    b. O cliente que não possuir cadastro deve se cadastrar para concluir o pedido;
2. Dados cadastrais;
    a. Nome;
    b. E-mail;
    c. Telefone (fixo, ou celular);
    d. Senha de acesso;
    e. Endereço;
3. O Cliente possui um perfil onde pode acompanhar seus pedidos, ou visualizar o histórico de pedidos;

Pedido

1. Um pedido acontece quando;
    a. O Cliente acessa o sistema e seleciona o Estabelecimento;
    b. Seleciona o Cardápio;
    c. Seleciona o Produto;
    d. Insere alguma observação no pedido (Ex. Pedido sem cebola);
    e. Clica em <<Fechar Pedido>>;
    f. Aparece o custo do pedido e um botão para confirmação;
    g. Ao confirmar o sistema redireciona o Usuário a página de de pedidos realizados.
2. Quando um pedido é confirmado, é feito um “lançamento” no painel administrativo que compete ao respectivo Estabelecimento o qual foi feito o Pedido;
    a. Quando o Prato começar a ser feito o Usuário seleciona o Pedido e clica em <<Em Execução>>;
        i. O Pedido passa a compor uma lista de Pedidos em execução;

   ii. É enviada uma mensagem no perfil do Cliente que o Pedido começou a ser feito (exibir a data e o horário);
   iii. A mensagem enviada é registrada no perfil do Estabelecimento;
b. Quando o Pedido for concluído o Usuário seleciona o respectivo Pedido no painel administrativo e clica em <<Pedido Concluído>>;
   i. É enviada uma mensagem ao Cliente comunicando que seu Pedido foi concluído e em breve será entregue (exibir data e horário);
   ii. A mensagem enviada é registrada no perfil do Estabelecimento;

APÊNDICE F – *Product Backlog* do Projeto

PRODUCT BACKLOG				
Id	**Tarefas**	**Módulo**	**Front / Back - End**	**Feito**
1	Formulário de Login - Front -Estrutura	Validar Acesso	Front	Ok
2	Formulário de Login - Front - Validações (js)	Validar Acesso	Front	Ok
3	Formulário de Login - Front – CSS	Validar Acesso	Front	Ok
4	Formulário de Login - Back – Model	Validar Acesso	Back	Ok
5	Formulário de Login - Back - Controller	Validar Acesso	Back	Ok
6	Cadastro de Usuários Adm. - Front - Estrutura	Manter Usuário	Front	Ok
7	Cadastro de Usuários Adm. - Front - Validação (js)	Manter Usuário	Front	Ok
8	Cadastro de Usuários Adm. - Front - CSS	Manter Usuário	Front	Ok
9	Cadastro de Clientes - Front - Estrutura	Manter Usuário	Front	Ok
10	Cadastro de Clientes - Front - Validação (js)	Manter Usuário	Front	Ok
11	Cadastro de Clientes - Front – CSS	Manter Usuário	Front	Ok
12	Portal de Compras - Exibição dos Pratos - Estrutura	Manter Pedido	Front	Ok
13	Portal de Compras - Exibição dos Pratos - Efeitos (js)	Manter Pedido	Front	Ok
14	Cliente - Ger. Perfil – Estrutura	Manter Usuário	Front	Ok
15	Cliente - Ger. Perfil - Validação (js)	Manter Usuário	Front	Ok
16	Cliente - Ger. Perfil – CSS	Manter Usuário	Front	Ok
17	Cadastro Usuários – Model	Manter Usuário	Back	Ok
18	Cadastro Usuários – Controller	Manter Usuário	Back	Ok
19	Proprietário - Ger. Perfil – Estrutura	Manter Usuário	Front	Ok
20	Proprietário - Ger. Perfil - Validação (js)	Manter Usuário	Front	Ok
21	Proprietário - Ger. Perfil – CSS	Manter Usuário	Front	Ok
22	Gerenciamento Perfil – Model	Manter Usuário	Back	Ok
23	Gerenciamento Perfil – Controller	Manter Usuário	Back	Ok
24	Proprietário - Ger. Usuários - Cadastro - Estrutura	Manter Usuário	Front	Ok
25	Proprietário - Ger. Usuários - Cadastro - Validação (js)	Manter Usuário	Front	Ok
26	Proprietário - Ger. Usuários - Cadastro - CSS	Manter Usuário	Front	Ok
27	Proprietário - Ger. Usuários - Cadastro - Model	Manter Usuário	Back	Ok
28	Proprietário - Ger. Usuários - Cadastro - Controller	Manter Usuário	Back	Ok
29	Proprietário - Ger. Usuários - Atualização - Estrutura	Manter Usuário	Front	Ok
30	Proprietário - Ger. Usuários - Atualização - Validação (js)	Manter Usuário	Front	Ok
31	Proprietário - Ger. Usuários - Atualização - CSS	Manter Usuário	Front	Ok
32	Proprietário - Ger. Usuários - Atualização - Model	Manter Usuário	Back	Ok
33	Proprietário - Ger. Usuários - Atualização - Controller	Manter Usuário	Back	Ok
34	Proprietário - Ger. Usuários - Alterar Estado - Estrutura	Manter Usuário	Front	Ok

35	Proprietário - Ger. Usuários - Alterar Estado - CSS	Manter Usuário	Front	Ok
36	Proprietário - Ger. Usuários - Alterar Estado - Mensagem Confirmação (js)	Manter Usuário	Front	Ok
37	Proprietário - Ger. Usuários - Alterar Estado - Model	Manter Usuário	Back	Ok
38	Proprietário - Ger. Usuários - Alterar Estado - Controller	Manter Usuário	Back	Ok
39	Proprietário - Ger. Usuários - Exclusão - Estrutura	Manter Usuário	Front	Ok
40	Proprietário - Ger. Usuários - Exclusão - Validação (js)	Manter Usuário	Front	Ok
41	Proprietário - Ger. Usuários - Exclusão - Mensagem Confirmação (js)	Manter Usuário	Front	Ok
42	Proprietário - Ger. Usuários - Exclusão - Model	Manter Usuário	Back	Ok
43	Proprietário - Ger. Usuários - Exclusão - Controller	Manter Usuário	Back	Ok
44	Ger. Est. - Cadastro - Estrutura	Manter Estabelecimento	Front	Ok
45	Ger. Est. - Cadastro - Validação	Manter Estabelecimento	Front	Ok
46	Ger. Est. - Cadastro - CSS	Manter Estabelecimento	Front	Ok
47	Ger. Est. - Cadastro - Model	Manter Estabelecimento	Back	Ok
48	Ger. Est. - Cadastro - Controller	Manter Estabelecimento	Back	Ok
49	Ger. Est. - Atualização - Estrutura	Manter Estabelecimento	Front	Ok
50	Ger. Est. - Atualização - Validação (js)	Manter Estabelecimento	Front	Ok
51	Ger. Est. - Atualização - CSS	Manter Estabelecimento	Front	Ok
52	Ger. Est. - Atualização - Model	Manter Estabelecimento	Back	Ok
53	Ger. Est. - Atualização - Controller	Manter Estabelecimento	Back	Ok
54	Ger. Est. - Alterar Estado - Estrutura	Manter Estabelecimento	Front	Ok
55	Ger. Est. - Alterar Estado - Msg Confirm. (js)	Manter Estabelecimento	Front	Ok
56	Ger. Est. - Alterar Estado - CSS	Manter Estabelecimento	Front	Ok
57	Ger. Est. - Alterar Estado - Model	Manter Estabelecimento	Back	Ok
58	Ger. Est. - Alterar Estado - Controller	Manter Estabelecimento	Back	Ok
59	Ger. Cardápios - Cadastro - Estrutura	Manter Cardápio	Front	Ok
60	Ger. Cardápios - Cadastro - Validação (js)	Manter Cardápio	Front	Ok
61	Ger. Cardápios - Cadastro - CSS	Manter Cardápio	Front	Ok
62	Ger. Cardápios - Cadastro - Model	Manter Cardápio	Back	Ok
63	Ger. Cardápios - Cadastro - Controller	Manter Cardápio	Back	Ok
64	Ger. Cardápios - Atualização - Estrutura	Manter Cardápio	Front	Ok
65	Ger. Cardápios - Atualização - Validação (js)	Manter Cardápio	Front	Ok
66	Ger. Cardápios - Atualização - CSS	Manter Cardápio	Front	Ok
67	Ger. Cardápios - Atualização - Model	Manter Cardápio	Back	Ok
68	Ger. Cardápios - Atualização - Controller	Manter Cardápio	Back	Ok

69	Ger. Cardápios - Exclusão - Estrutura	Manter Cardápio	Front	Ok
70	Ger. Cardápios - Exclusão - Validação (js)	Manter Cardápio	Front	Ok
71	Ger. Cardápios - Exclusão - CSS	Manter Cardápio	Front	Ok
72	Ger. Cardápios - Exclusão - Model	Manter Cardápio	Back	Ok
73	Ger. Cardápios - Exclusão - Controller	Manter Cardápio	Back	Ok
74	Portal de Compras - Exibição dos Pratos - CSS	Manter Pedido	Front	Ok
75	Compra da Refeição - Ajustes da Compra - Estrutura	Manter Pedido	Front	Ok
76	Compra da Refeição - Ajustes da Compra - js	Manter Pedido	Front	Ok
77	Compra da Refeição - Ajustes da Compra - CSS	Manter Pedido	Front	Ok
78	Compra da Refeição - Ajustes da Compra - Botão add "carrinho de compras"	Manter Pedido	Front	Ok
79	Página Carrinho Compras - Remover Item - Estrutura / CSS	Manter Pedido	Front	Ok
80	Página Carrinho Compras - Remover Item - Msg Confirmação (js)	Manter Pedido	Front	Ok
81	Página Carrinho Compras - Fechar Pedido - Botão para pg fechamento pedido	Manter Pedido	Front	Ok
82	Página Fechamento Pedido - Inserir Endereço - Estrutura / CSS	Manter Pedido	Front	Ok
83	Página Fechamento Pedido - Inserir Endereço - Validações (js)	Manter Pedido	Front	Ok
84	Página Fechamento Pedido - Definir forma pgto - Estrutura / CSS	Manter Pedido	Front	Ok
85	Página Fechamento Pedido - Definir forma pgto - Validações (js)	Manter Pedido	Front	Ok
86	Página Fechamento Pedido - Btn Fechar Pedido	Manter Pedido	Front	Ok
87	Cliente - Acomp. Pedidos - Estrutura / CSS	Manter Pedido	Front	Ok
88	Cliente - Acomp. Pedidos - js	Manter Pedido	Front	Ok
89	Usuários Operacionais - Acomp. Pedidos - Estrutura / CSS	Manter Pedido	Front	Ok
90	Usuários Operacionais - Acomp. Pedidos - js	Manter Pedido	Front	Ok
91	Usuários Operacionais - Histórico de Pedidos Atendidos - Data	Manter Pedido	Front	Ok
92	Usuários Operacionais - Histórico de Pedidos Atendidos - Período	Manter Pedido	Front	Ok
93	Usuários Operacionais - Histórico de Pedidos Atendidos - Est. [Proprietário]	Manter Pedido	Front	Ok
94	Usuários Operacionais - Histórico de Pedidos Cancelados - Data	Manter Pedido	Front	Ok
95	Usuários Operacionais - Histórico de Pedidos Cancelados - Período	Manter Pedido	Front	Ok
96	Usuários Operacionais - Histórico de Pedidos Cancelados - Est. [Proprietário]	Manter Pedido	Front	Ok
97	Portal Compras - Busca - Pratos	Manter Pedido	Front	Ok
98	Portal Compras - Busca - Estabelecimentos	Manter Pedido	Front	Ok
99	Exibição dos Pratos - Model	Manter Pedido	Back	Ok
100	Filtros de Busca - Model	Manter Pedido	Back	Ok
101	Exibição dos Pratos - Controller	Manter Pedido	Back	Ok
102	Filtros de Busca - Controller	Manter Pedido	Back	Ok
103	Compra da Refeição - Model	Manter Pedido	Back	Ok

104	Compra Refeição - Controller	Manter Pedido	Back	Ok
105	Carrinho de Compras - Model	Manter Pedido	Back	Ok
106	Carrinho de Compras - Controller	Manter Pedido	Back	Ok
107	Registro Eventos - Model	Manter Registro Eventos	Back	Ok

Quadro 7 – *Product Backlog* Completo
Fonte: Autoria própria