UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS GUARAPUAVA
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

VINÍCIUS BAIL ALONSO

EICHEF: SISTEMA PARA AUTOMAÇÃO DE ATENDIMENTO EM BARES E RESTAURANTES

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA

2016

VINÍCIUS BAIL ALONSO

EICHEF: SISTEMA PARA AUTOMAÇÃO DE ATENDIMENTO EM BARES E RESTAURANTES

Monografia de Trabalho de Conclusão de Curso de graduação, apresentado a disciplina de Trabalho de Conclusão de Curso 2 do Curso Superior de Tecnologia em Sistemas para a Internet - TSI da Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para a Internet.

Orientador: Prof. Dr. Diego Marczal

GUARAPUAVA

2016

RESUMO

ALONSO, Vinícius. EICHEF: Sistema para Automação de Atendimento em Bares e Restaurantes. 63 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2016.

Em muitos bares e restaurantes, os pedidos dos clientes são feitos de forma tradicional. O garçom vai até a mesa com papel e caneta, anota o pedido, repassa aos cozinheiros e após estar tudo pronto o pedido é entregue ao consumidor. Embora esse seja um processo simples alguns fatores podem causar um desconforto ao cliente. Como por exemplo, um pedido anotado errado ou que demora para chegar. Com isso, surge a necessidade de automatizar a solicitação de pedidos de clientes de modo a tentar reduzir as falhas humanas. O protótipo desenvolvido apresenta o menu do estabelecimento ao consumidor, possibilitando que ele faça seu pedido sem chamar um garçom e enviar seu pedido diretamente aos funcionários que irão prepará-lo. Permitindo que o cliente acompanhe o preparo do seu pedido. E após o cliente consumar seu pedido pode avaliar a qualidade dos pratos servidos. Nesse cenário, o trabalho do garçom é reduzido a entregar os pedidos. Por meio desse sistema, dados são coletados, como por exemplo, prato mais pedido e melhor avaliado, que posteriormente são dispinibilizados ao gerente na tentativa de auxiliá-lo na tomada de decisões.

Palavras-chave: Atendimento ao cliente. Automatização. Falhas no atendimento. Tomada de decisões.

ABSTRACT

ALONSO, Vinícius. EICHEF: System to Automation of Treatment in Bars and Restaurants. 63 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2016.

In many bars and restaurants, the customer orders are made traditionally. The waiter goes to the table with pen and paper, write down the request, passes the cooks and after being all set the order is delivered to the consumer. Although this is a simple process some factors may cause discomfort to the client. As an example, an application noted wrong or it takes to arrive. With that comes the need to automate the request of customer orders in order to try to reduce human error. The prototype has the establishment consumer menu, enabling it to place your order without calling a waiter and send your order directly to the employees who will prepare it. Allowing the customer to track the preparation of your order. And after the client complete his application can assess the quality of the dishes served. In this scenario, the work of the waiter is reduced to deliver applications. Through this system, data is collected, such as dish more order and better assessed, which are subsequently dispinibilizados the manager in an attempt to assist in decision making.

Keywords: Customer service. Automation. Failures in care. Decision-making.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACID	Atomicity, Consistency, Isolation, Durability (Em português: Atomicidade, Consistência, Isolamento, Durabilidade)
API	Application Programming Interface (Em português: Interface de Programação de Aplicações)
DOM	Document Object Model (Em português: Modelo de Objeto de Documentos)
DRY	Don't Repeat Yourself (Em português: Não se Repita)
HTML	Hypertext Markup Language (Em português: Linguagem de Marcação de Hipertexto)
IHC	Interação Humano Computador
JSON	Javascript Object Notation (Em portugês: Notação de Objeto Javascript)
ORM	Object Relational Mapper (Em português: Mapeamento Objeto Relacional)
POS	Point of Sale (Em português: Ponto de Venda)
QR Code	Quick Response Code (Em portugês: Código de Resposta Rápida)
SGBD	Sistema Gerenciador de Banco de Dados
SPA	Single Page Application (Em português: Aplicação de Página Única)
TCC	Trabalho de Conclusão de Curso
UML	Unified Modeling Language (Em português: Linguagem de Modelagem Unificada)
URL	Uniform Resource Locator (Em português: Localizador Padrão de Recursos)
XML	Extensible Markup Language (Em português: Linguagem de Marcação Extensiva)

SUMÁRIO

1 INTRODUÇÃO

Um dos principais fatores utilizados por estabelecimentos de consumo para fidelizar seus clientes é o atendimento. Ter qualidade nesse aspecto tornou-se um fator crucial para aqueles que almejam satisfazer seus clientes. Isso também pode ser visto como um grande diferencial perante dos concorrentes, pois oferecer um serviço de qualidade é uma forma de obter vantagem competitiva (GARCEZ et al., 2000).

Segundo OLIVEIRA (2002), em estabelecimentos como restaurantes podem ocorrer diversos tipos de falhas que prejudicam o serviço ao cliente. Como exemplo, uma máquina de cartões de crédito pode quebrar, um garçom pode cometer um erro simples ao anotar o pedido ou no momento de entregar o pedido na mesa, confundir com outro pedido.

Essas falhas podem fazer com que o estabelecimento perca clientes, pois estes julgam a credibilidade de um estabelecimento com base em suas experiências anteriores. Alguns dos problemas acima citados são difíceis de prever, porém, os problemas relacionados ao garçom podem ser previstos.

O uso adequado da tecnologia pode ser uma solução, retirando do garçom muitas responsabilidades e com isso evitando que esses profissionais fiquem sobrecarregados. Exemplos de tais são, pedir um prato pode ser passada diretamente a quem vai prepará-lo, pedir uma bebida pode ser passada direto ao bar, evitando assim, que o garçom tenha que andar até a mesa mais vezes do que o necessário.

O principal desafio foi projetar um sistema que automatize essas tarefas sem complicar um processo que já é muito simples. Como também, não alterar muito a rotina dos restaurantes e bares que farão uso desse *software*.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Automatizar os pedidos realizados dentro de bares e restaurantes, por um sistema web acessado por dispositivos móveis dos clientes ou ainda oferecido pelo estabelecimento.

1.1.2 OBJETIVOS ESPECÍFICOS

1. Auxiliar clientes de bares e restaurantes a fazer seus pedidos de forma eletrônica mediante a um dispositivo móvel, sem a necessidade de chamar um garçom;
2. Disponibilizar para os clientes, um menu do estabelecimento, em que será possível avaliar os pratos servidos;
3. Fornecer aos garçons e aos cozinheiros a relação de pedidos realizados pelos clientes;
4. Fornecer ao gerente relatórios para auxiliar a tomada de decisões estratégicas, como por exemplo pratos mais pedidos;
5. Apresentar pratos mais vendidos e melhor avaliados para os clientes, através de um menu interativo.

1.2 DIFERENCIAL TECNOLÓGICO

Neste tópico destacam-se os diferencias tecnológicos do trabalho proposto em relação ao estado da arte. Primeiramente tratando-se de uma aplicação web, o cliente deveria digitar o endereço do site para acessá-lo, porém, isso acabaria gerando desconforto, uma certa demora e dificuldade no acesso ao menu do restaurante. Uma boa alternativa para isso é a utilização de *Quick Response Code* (QR Code).

Segundo Waters (2012), um QR Code é um código de barras bidimensional, em que é possível guardar alguma informação para ser acessada de forma rápida. Essa informação pode ser um vídeo para ser exibido, uma música para ser tocada, entre outras milhares de possibilidades. No sistema EICHEF o QR Code iria abrir o site e informar o número da mesa. Isso, além de possibilitar mais conforto ao cliente, facilita a identificação da mesa que foi feito o pedido.

Outra técnica utilizada foi *Single Page Application* (SPA), que possui algumas vantagens em relação a aplicativos nativos, como por exemplo: é multi-plataforma, dependendo

apenas do *browser*; permite gerenciar o estado do cliente, favorecendo a aplicações *offline*; e, por último não necessita de instalação (FINK; FLATOW, 2014).

Como último diferencial, o sistema também possui um *ranking* com os pratos mais vendidos. Útil para o cliente, na decisão do pedido. Além disso, o sistema visa fornecer alguns relatórios ao gerente, como por exemplo tempo de espera desde o pedido até a entrega do prato. Essas informações podem ajudá-lo, a definir possíveis melhoras no atendimento e na elaboração de pratos.

1.3 ESTRUTURA DO PROJETO

Essa monografia está dividida da seguinte forma: no segundo Capítulo é apresentado o estado da arte e o diferencial tecnológico do protótipo desenvolvido em relação ao mesmo. No terceiro Capítulo são apresentadas as tecnologias estudas e quais foram utilizadas. No quarto Capítulo é descrito a metodologia seguida para desenvolver o protótipo. O quinto Capítulo descreve como o projeto foi desenvolvido. O sexto Capítulo explica a ferramenta desenvolvida. No sétimo Capítulo é apresentado o experimento de aceitação. E no oitavo Capítulo as considerações finais sobre o projeto.

2 ESTADO DA ARTE

Nesta seção serão abordados trabalhos correlatos, sendo apresentado os mais relevantes a essa pesquisa e desenvolvimento.

2.1 EZEE IMENU

O primeiro sistema estudado foi o eZee iMenu (EZEE TECHNOSYS, 2015b), desenvolvido pela empresa eZee Technosys. Este é um aplicativo que possui uma versão para iOS (sistema operacional móvel da Apple, originalmente iPhone OS) e outra para Android (sistema operacional móvel baseado no kernel do Linux). Por esses, é possível que o cliente de um estabelecimento faça pedidos do seu próprio celular. Mas não se limitando apenas aos aparelhos dos clientes, caso seja necessário, o estabelecimento também pode adquirir tablets e quando seus clientes estiverem fisicamente no estabelecimento, um tablet pode ser fornecido como se fosse uma carta de menu. Assim, os clientes podem realizar seus pedidos diretamente por ele. O sistema está ilustrado, na Figura 1.

Figura 1: Ezee iMenu.

Fonte: http://fortle-telecoms.com/products.php

Segundo o site oficial do eZee iMenu[1], este possui as funcionalidades relacionadas à seguir (EZEE TECHNOSYS, 2015a):

[1] Site Official `http://www.ezeeimenu.com/features.php`

- Customização de tema;
- Menu interativo;
- Modificador de items;
- Integração com sistemas *Point of Sale* (POS)[2];
- Modo *offline*;
- *Feedback* do convidado;
- Observações em pedidos ou itens;
- Sincronização;
- Busca rápida;
- Exportação de menu.

Além dessas funcionalidades, o sistema também possui três modos de operação: 1) modo de visualização; 2) modo de pedido do convidado; 3) e, modo do garçom (EZEE TECHNOSYS, 2015b).

O **modo de visualização** é apenas para que o cliente navegue pelo menu interativo, sem poder fazer pedidos. Nesse modo o garçom fica responsável por fazer os pedidos em nome do cliente.

No **modo de pedido**, o cliente pode fazer seus próprios pedidos. Com isso, a responsabilidade do garçom fica apenas em servir a mesa e depois recolher o valor do pagamento da conta.

O **modo garçom** é utilizado pelo garçom para executar seu trabalho da mesma forma que faria utilizando papel e caneta. Porém, quando ele marcar um pedido, o mesmo irá aparecer instantaneamente para as pessoas responsáveis por preparar as refeições na cozinha.

2.2 EPED

Outro sistema estudado foi o Eped (PEKUS, 2015), um aplicativo desenvolvido pela empresa Pekus Consultoria para ser utilizado em tablets com o sistema operacional Android.

[2]Sistema por meio do qual ocorre uma transação financeira, tais como, máquinas de cartões de crédito, caixas registradoras, entre outros.

Por esse aplicativo é possível que o cliente faça pedidos por um tablet fixado à mesa ou oferecido por um funcionário.

O Eped é semelhante em alguns aspectos com o eZee iMenu. Ambos oferecem um menu interativo para que o usuário possa fazer pedidos, além disso, o Eped possui algumas funcionalidades como: ser utilizado sem integração com sistema de caixa do estabelecimento e também ser traduzido para três idiomas português, inglês e espanhol. Na Figura 2, uma ilustração do sistema é apresentada.

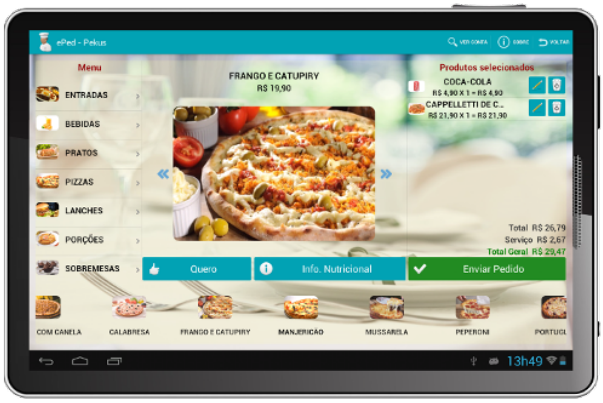

Figura 2: Eped.

Fonte: http://www.pekus.com.br/cardapio.aspx

2.3 PSIU GARÇOM

O último sistema analisado foi o Psiu Garçom (PSIU GARÇOM, 2015). Diferente dos outros dois antes mencionados, ele trabalha com o propósito principal de chamar o garçom "[...] é composto por transmissores individuais sem fio posicionados nas mesas do estabelecimento e por um painel de leds, instalado em um local de fácil visualização [...]" (PSIU GARÇOM, 2015, p. principal). Dessa maneira, a mesa do cliente ganha um botão para chamar o garçom, emitindo um sinal sonoro e exibindo o número de sua mesa em um painel eletrônico. O número só irá sair do painel após o garçom que atendeu pressionar o outro botão que está na mesa do cliente. O sistema é ilustrado na Figura 3

Figura 3: Psiu Garçom.

Fonte: http://gestaoderestaurantes.com.br/blog/index.php/2009/07/17/psiu-garom-lana-nova-mdia-na-fipan-2009/

Além de facilitar o atendimento, o Psiu Garçom também possui outras duas funcionalidades, ele exibe até 99 (noventa e nove) mensagens publicitárias no painel eletrônico e também cronômetra o tempo de atendimento às mesas fornecendo relatórios para o gerente.

2.4 COMPARAÇÃO DOS SISTEMAS ESTUDADOS

O Psiu Gaçom cumpre bem com funcionalidade de chamar os garçons. Porém, com relação ao atendimento, está limitado somente a isso. O eZee iMenu e o Eped são bem completos, porém seu design poderia ser mais simples para o usuário, tornando os sistemas mais intuitivos. Isso pode ser feito removendo conteúdo desnecessário da tela de pedidos e deixando apenas o que é preciso no momento de se fazer um pedido. Na Tabela 1 é possível ver as diferenças mais relevantes a esse projeto.

Tabela 1: Diferenças mais relevantes entre os sistemas

	***Feedback* do Cliente**	**Cronometro da entrega**	**Acompanhamento do pedido**
Ezee iMenu	Possui	Não possui	Não possui
Eped	Não possui	Não possui	Não possui
Psiu Garçom	Não possui	Não possui	Não possui
Eichef	Possui	Possui	Possui

Os sistemas acima citados cumprem seu propósito, que é facilitar o atendimento e como os pedidos são feitos. Por essa razão, o EICHEF considerou as funcionalidades desses

sistemas como base para seu desenvolvimento, como por exemplo, o menu interativo e o *feed-back* do cliente. Porém o EICHEF teve alguns diferencias, tais como possibilidade do cliente acompanhar o estado do seu pedido. Também é possível saber o tempo que o pedido demorou para ser entregue. Com essas informações o gerente pode saber em que ponto o estabelecimento pode melhorar seu atendimento.

3 REFERENCIAL TEÓRICO

3.1 SINGLE PAGE APPLICATION

Segundo Fink e Flatow (2014), o SPA é um conceito que busca melhorar a performance das aplicações *web*. Na abordagem tradicional da *web*, um cliente, que pode ser um navegador ou uma outra aplicação, faz uma requisição ao servidor. Em seguida o servidor deve processar todo o conteúdo estático da página, tais como, *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS), Javascript, entre outros, retornando o cliente. O problema nessa abordagem é que o servidor que deve repetir esse ciclo a cada nova requisição.

A abordagem do SPA é diferente. O servidor deve responder apenas a primeira requisição com todos os arquivos estáticos. Nas outras requisições deve responder utilizando *Javascript Object Notation* (JSON), delegando a responsabilidade de apresentar os dados de uma forma elegante ao cliente (SCOTT, 2015).

Dentre as principais vantagens estudadas, destacam-se:

- **O navegador não precisa recarregar a página**. Ao invés disso, o *framework* do *client-side* pega o conteúdo do JSON utilizando ajax e insere na página;
- **A lógica envolvida na apresentação dos dados fica a cargo do cliente**. Por meio disso, é possível retirar um pouco da carga do servidor;

  Transações são feitas no servidor. O *framework client-side*, deve enviar os dados para a *server-side*, aonde são feitas operações que envolverão banco de dados e regras de negócio da aplicação.

Para se conseguir chegar a isso, foi necessário dividir as responsabilidades entre o lado do servidor e o lado do cliente. No lado do servidor foi necessário utilizar uma API, executando as regras de negócio e respondendo de uma forma adequada. E o lado do cliente, foi responsável por apresentar os dados.

3.2 FRAMEWORK BACKEND

Para o desenvolvimento referente ao *backend*, foi utilizado o *framework* Ruby on Rails. Como o projeto foi desenvolvido seguindo práticas da metodologia SCRUM, foi escolhido uma ferramenta que utiliza conceitos de desenvolvimento ágil.

Ruby on Rails é um *framework* para desenvolvimento web, criado para tornar o desenvolvimento rápido e manter o código limpo. Para isso, o *framework* segue algumas boas práticas, como por exemplo, convenção sobre configuração e *Don't Repeat Yourself* (DRY).

A filosofia do DRY diz respeito a não repetição de código. O Rails é um *framework* escrito com a linguagem Ruby e aproveita-se do poder da linguagem para evitar duplicação de código. Além da linguagem, o padrão *Model-View-Controller* (MVC) implementado no *framework*, permite colocar cada funcionalidade em um lugar especifico evitando duplicações.

A convenção sobre configuração torna o *framework* ágil pelo fato de deixar a configuração para desenvolver uma aplicação miníma. Diferente de outras tecnologias aonde é necessário configurar diversos arquivos *Extensible Markup Language* (XML). O Rails traz várias configurações por *default* e por meio das convenções é possível sobrescrever-las com facilidade (RUBY et al., 2013, p. 21-23).

Na aplicação proposta o *framework* Ruby on Rails, foi responsável pelas regras de negócio da aplicação. Seguindo os princípios do *Single Page Application* (SPA), o Rails é uma *Application Programming Interface* (API) por meio da qual o *framework frontend* utilizado se comunica.

3.3 FRAMEWORK FRONTEND

Tratando-se de uma aplicação que segue o conceito SPA, é importante utilizar um *framework* apropriado para o *client-side*. A função desse *framework* é se comunicar com a API que foi desenvolvida utilizando Ruby on Rails e apresentar os dados em tempo real.

Com esse propósito, foram estudados alguns candidatos, entre eles destacaram-se o Angularjs e o Emberjs.

3.3.1 ANGULARJS 1.0

O Angularjs é um *framework* escrito em Javascript para trabalhar no lado do cliente, tornando o desenvolvimento mais rápido e fácil. Foi desenvolvido pelo Google[1] que o usa em projetos como Gmail, Google Maps e Google Calendar[2].

Até o momento da escrita desse projeto, o AngularJs é mantido pela própria Google e por uma grande comunidade de desenvolvedores. Além disso, ele abstrai diversas funcionalidades comuns a aplicações web modernas. Como por exemplo:

- Separação da lógica, classes e *views*;
- Serviços com Ajax;
- Injeção de dependências;
- Testes automatizados.

Fora as citadas acima, o Angularjs possui muitas funcionalidades que não foram relacionadas (LERNER, 2013, p. 8-9).

Ele possui diversas vantagens de uso, entre elas destacam-se:

- **Modificar o *Document Object Model* (DOM) de forma simples**, sem o uso de métodos observáveis como os *listeners* do Javascript. Para isso o *framework* usa **diretivas**, ***data-binding*** e **expressões** que se misturam ao código HTML. Dessa forma, o desenvolvedor se preocupa com a apresentação dos dados. Deixando o trabalho de como fazer isso com o *framework* (STARZHINSKY, 2015).
- **É organizado**, consegue serapar muito bem a lógica da apresentação dos dados. Isso porque segue o padrão (MVC). Isso torna o aplicação mais fácil para futuras manutenções (LERNER, 2013, p. 8-9).
- **Possui um bom suporte a testes automatizados**. O Angularjs adotou como biblioteca para testes padrão o Jasmine[3]. Graças a isso, é possível testar uma aplicação, desde testes unitários até testes de integração. Além disso, possui suporte a *mocks*[4]. Garantindo que as funcionalidades implementadas estão de acordo com as especificações (KOZLOWSKI, 2013, cap. 2).

[1] Multinacional norte-americana especializada em serviços de internet
[2] Algus dos serviços oferecidos pela empresa Google
[3] `http://jasmine.github.io/`
[4] Objetos dublês, usados para simular objetos reais.

O *framework* também possui algumas desvantagens. Entre as estudadas, a que mais se destaca é o fato do *framework* ser muito grande e oferecer diversas formas de se chegar ao mesmo resultado. Isso pode ser muito confuso e trabalhoso para um desenvolvedor que está inciando com o Angularjs (STARZHINSKY, 2015).

3.3.2 EMBERJS

Segundo Kelonye (2014) o Emberjs é um *framework* divertido e *open source* escrito em Javascript. É usado para criar aplicações ambiciosas e complexas. Foi construido a partir do SproutCore[5] criado por Yehuda Katz e Tom Dale. O Emberjs usa conceitos como MVC e convenção sobre configuração, o que torna o desenvolvimento mais organizado. Tem sido adotado por empresas do mundo todo, entre elas destacam-se a Apple, Groupon, Square, Zendesk e a Tilde.

Utilizar essa ferramenta pode trazer muitos beneficios ao desenvolvedor. Segundo Cravens e Brady (2014, p. 7-8), entre os beneficios destacam-se:

- **Oferece um ótimo recurso de *Object Relational Mapper* (ORM)**, que permite traduzir dados de um tipo de sistema para outro. Isso permite por exemplo, que dados que estão em objetos Javascript, sejam enviado ao *backend* e salvos no Mysql[6] sem complicações.
- **Permite o que é chamado de *developer ergonomics***. Graças as convenções que o *framework* usa, é possível um desenvolvimento ergonômico. Por exemplo, o desenvolvedor não precisa configurar qual template será carregado. Se o template seguir as convenções ele será carregado automaticamente. Além disso, o Emberjs possui muitos comportamentos por *default*, o que tira um pouco da carga de trabalho da equipe de desenvolvimento.

O Emberjs traz várias funcionalidades e conveções. Isso pode facilitar muito o trabalho de um desenvolvedor. Porém, em contrapartida, também existe um problema. Devido a isso, o *framework* acaba tendo uma curva de aprendizado longa. Dependendo do tempo que se tem para desenvolver o projeto, isso pode tornar o seu uso inviável (SKEIE, 2013, cap. 1).

Apesar de os dois *frameworks* estudados conseguirem cumprir bem o papel no lado do cliente. O Emberjs foi escolhido, devido a sua boa integração com Ruby on Rails, isso acontece graças ao fato, do Emberjs seguir filosofias comuns. Como por exemplo, convenção sobre configuração.

[5] *Framework* Javascript criada pela Apple para inovar a experiência do usuário em *web sites*
[6] `https://www.mysql.com`

3.4 TEST DRIVEN DEVELOPMENT

O sistema será desenvolvido seguido os príncipios do *Test Driven Development* (TDD). TDD é um ciclo de desenvolvimento, no qual o desenvolvedor deve escrever um teste antes de escrever um código de produção. Ou seja, primeiro é escrito um código que irá testar a nova funcionalidade, antes de escrever a funcionalidade em si (ANICHE; CORBUCCI, 2014).

O TDD baseia-se em três estágios *red-green-refactor*. No estágio ***red***, o teste é escrito, como a funcionalidade a ser testada não existe, consequentemente o teste irá falhar. Em seguida temos o estágio ***green***, agora a funcionalidade é escrita sem preocupação com conceitos de *clean code*, o importante é apenas passar no teste. Agora com a funcionalidade aprovada no teste, vêem o estágio ***refactor***. Nessa etapa, o desenvolvedor deve refatorar seu código para garantir que fique o melhor escrito possível.

Segundo Aniche e Corbucci (2014, cap. 3) , esse processo pode trazer muitas vantagens para o seu praticante. Entre elas destacam-se:

- **Funcionalidade já fica pronta e testada**. Como o desenvolvedor escreve o teste e depois a funcionalidade, isso implica que quando ela estiver pronta já haverá um teste para ela;
- **O desenvolvedor pensa no teste e não na implementação**. Com isso, ele acaba pensando em cenários possíveis que a nova funcionalidade irá encontrar. Cenários nos quais, ele poderia esquecer se estivesse programando a funcionalidade direto;
- ***Feedback* rápido e constante**. Como os testes estão sempre sendo executados a cada ciclo, caso haja algum problema, poderá ser encontrado e solucionado rapidamente. Graças ao *feedback* fornecido constantemente pelos testes (FREEMAN; PRYCE, 2009).

Embora o TDD traga muitas vantagens, também existem desvantagens. A mais importante delas, considerando o projeto proposto, é a **experiência**. Em alguns momentos a prática de TDD pode atrapalhar. Nem sempre é recomendado o seu uso, como por exemplo, para testar *views* ou lógicas muito simples. Porém, o momento de usar ou não depende da experiência do desenvolvedor (ANICHE; CORBUCCI, 2014).

A importância dessa prática, será oferecer *feedback* constante durante o desenvolvimento do projeto proposto. Isso poderá trazer mais segurança para implementar as funcionalidades propostas. Além de facilitar mudanças, caso sejam necessárias.

3.5 BEHAVIOR DRIVEN DEVELOPMENT

Behavior Driven Development (BDD) é uma metodologia ágil a qual tem o objetivo de desenvolver sofware melhor e mais rapidamente. Foi criado baseado no TDD e no *Domain Driven Development* (DDD). Diferente do TDD, que o objetivo é apenas escrever testes e refatorar o código depois. No BDD, os testes são escritos para descrever como a aplicação deve se comportar, ou seja, é orientado a comportamento.

Sua principal característica é a possibilidade de descrever o software em uma linguagem natural. Por meio da qual, a equipe de desenvolvimento, os testadores e os *stakeholders*[7], podem se entender com mais facilidade (SMART, 2014).

No BDD, as funcionalidades do sistema são descritas em forma de histórias, com participação dos *stakeholders*. Por essa razão são escritas em linguagem natural. Com as histórias montadas, é necessário testá-las, para isso são usados cenários, os quais descrevem as situações as quais a funcionalidade será submetida e quais serão suas respostas (ASTELS; CHELIMSKY, 2010).

Dentre as principais vantagens dessa abordagem, destacam-se:

- A nova funcionalidade desenvolvida fica testada e documentada;
- Promove uma maior interação entre toda a equipe, desenvolvedores, *stakeholders*, entre outros.

Dentre as vantagens acima citadas, a primeira sem dúvidas, é a mais relevante para o projeto proposto. Por meio da descrição, será mais fácil de testar as funcionalidades e manter uma boa documentação para futuras extensões ou mudanças.

3.6 METODOLOGIA SCRUM

Segundo HE Labs (2013), SCRUM é uma metodologia àgil que é utilizada com a finalidade de planejar e gerir projetos de *software*. Nessa metodologia, cada funcionalidade do sistema é chamada de história. Histórias são mantidas em uma lista chamada *product backlog*.

O SCRUM é iterativo, e suas iterações são divididas em intervalos de tempo, esses intervalos são chamados de *sprints*. As *sprints*, são muito importantes porque nelas são divididas,

[7] São as partes interessadas no software, podem ser clientes, usuários, entre outros.

estimadas e desenvolvidas as funcionalidades do sistema. Cada *sprint* é planejada e definida em uma reunião. Essa reunião é chamada *sprint planning*.

A *sprint planning* é composta pela equipe de desenvolvimento, o *product owner*[8] e o *scrum master*[9]. Nessa reunião coordenada pelo *scrum master*, são definidas quais histórias serão implementadas na *sprint* corrente.

Durante uma *sprint*, todos os dias, preferencialmente pela manhã ocorre a *daily scrum meeting*. Trata-se de uma rápida reunião em pé, com duração máxima de 15 (quinze) minutos. Por meio da qual o *scrum master* recebe um *feedback* da execução de cada atividade. Essa reunião é muito importante para o acompanhamento das atividades e também para a equipe de desenvolvimento tirar possíveis dúvidas (RUBIN, 2012).

A cada iteração uma parte do produto final será entregue, até chegar a conclusão do projeto. Isto é demonstrado na Figura 4.

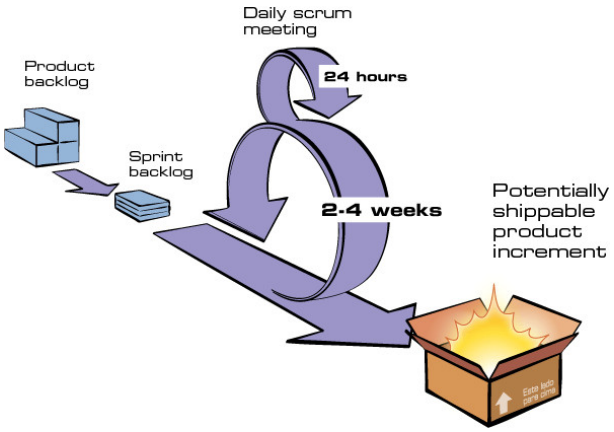

Figura 4: Ciclo do SCRUM.

Fonte: http://www.desenvolvimentoagil.com.br/scrum/

3.7 TECNOLOGIAS APRESENTADAS EM FUNÇÃO DO PROJETO

O projeto proposto seguirá as práticas do *framework* SCRUM, sendo assim, as etapas do projeto serão divididas em *sprints*. Seguindo esse princípio, foi escolhido um *framework* para o *backend* que é apropriado para metogolodias ágeis.

O *backend* será feito utilizando Ruby on Rails, que funcionará como uma API. Sua função será executar as regras de negócio e responder em *Javascript Object Notation* (JSON)

[8]Membro que possui conhecimento do domínio do problema. Responsável por montar o *product backlog* e dar prioridade as histórias na *sprint*.

[9]Membro responsável por ajudar todos os envolvidos no processo do SCRUM. Tem contato direto com a equipe de desenvolvimento os ajudando a entender as funcionalidade e impedindo de receberem interferencias externas.

para o *framework frontend*, que será o Emberjs. A interação do usuário e a apresentação dos dados ficará a cargo do Emberjs, que deverá enviar requisições para a API e depois apresentar suas respostas ao usuário.

As funcionalidades serão dividias em histórias, e cada uma será desenvolvida seguindo o ciclo do TDD. Os testes de aceitação e as histórias serão desenvolvidos utilizando a biblioteca *Cucumber*[10]. Para os outros testes, tais como, unitários e de integração, será utilizada a biblioteca *RSpec*[11].

[10] `https://cucumber.io/`
[11] `http://rspec.info/`

4 PROCEDIMENTOS METODOLÓGICOS

Nesta seção descrevem-se os procedimentos metodológicos da solução proposta para o problema apresentado. Os passos metodológicos estabelecidos inicialmente são relacionados e descritos na sequência.

4.1 LEVANTAMENTO DOS REQUISITOS DO SISTEMA

Seguindo os conceitos do BDD, os requisitos do sistema foram descritos em forma de histórias. Essas histórias descrevem os usuários interagindo com o sistema em alguns cenários, e quais serão as respostas que o sistema deveria retornar em cada cenário. Essas histórias foram escritas em linguagem natural, e não envolveram nenhum conceito de implementação interna.

Ao final dessa etapa, teve-se em mãos todas as funcionalidades do sistema descritas para planejamento e implementação na etapas à diante.

4.2 IDEALIZAÇÃO DA ARQUITETURA DO SISTEMA

Com as histórias em mãos, a próxima etapa foi planejar o funcionamento interno do sistema. Visando ter um código modular, a aplicação foi dividida em módulos. Cada módulo foi responsável por uma parte da aplicação. Conforme a complexidade de cada módulo, poderiam ser usados diagramas da Linguagem de Modelagem Unificada[1] (UML) para ajudar no planejamento.

Após terminada essa etapa, o funcionamento interno da aplicação deveria estar planejado.

[1] Do Inglês: Unified Modeling Language

4.3 ESTUDO DAS TECNOLOGIAS À SEREM UTILIZADAS

Nessa etapa foram estudadas as tecnologias a serem empregadas no projeto. Foram reunidos livros para esta etapa. O objetivo dela foi criar familiaridade com as tecnologias e práticas sugeridas para o projeto.

4.4 DESENVOLVIMENTO DO SISTEMA

Essa etapa foi dividida em *sprints*, planejadas junto com o orientador. As histórias receberiam uma nota durante o planejamento das *sprints*, que representaria sua importância para o projeto. Após isso, foram escolhidas quais histórias seriam implementadas na *sprint* corrente.

Após uma nova funcionalidade ser finalizada ao final de cada *sprint*, as funcionalidades desenvolvidas foram integradas ao projeto. O código fonte da aplicação ficou em um repositório privado no *Bitbucket*[2]. Além disso, uma versão parcial da aplicação estava hospedada no servidor pessoal do orientador.

Seguindo o ciclo do TDD, durante essa etapa foram escritos testes unitários e de integração. Os testes de aceitação, seriam deixados para a próxima etapa.

O objetivo a ser alcançado nessa etapa é ter a aplicação proposta implementada com as funcionalidades descritas na Seção 4.1.

4.5 VALIDAÇÃO DOS REQUISITOS IMPLEMENTADOS

Os requisitos já implementados foram testados *online*. Esses testes foram feitos visando achar possíveis *bugs*[3]. Além disso, foi testado como o *layout* ficaria em diferentes tamanhos de tela e *browsers*.

4.6 TESTES DE USABILIDADE

Com as funcionalidades implementadas, o sistema foi testado com usuários reais, para verificar sua usabilidade. Porém, foi realizado um experimento de caráter mais geral, sem seguir profundamente as diretivas de teste de usabilidade de *Interação Humano Computador* (IHC).

[2] `https://bitbucket.org/`

[3] Um comportamento errado da aplicação, causado por algum erro de lógica do desenvolvedor

Nessa etapa foram reunidos alguns voluntários para usar o sistema e depois deixarem um *feedback*. Isso foi feito na universidade com alguns alunos. O objetivo dessa etapa foi analisar *feedbacks*, por meio disso foi possível detectar dificuldades de uso, não percebidas pelo desenvolvedor.

4.7 UTILIZAR A APLICAÇÃO EM AMBIENTE DE PRODUÇÃO REAL

Após desenvolver o sistema pretendeu-se utilizá-lo em um restaurante ou bar. Para isso seria necessário entrar em contato com alguns proprietários e verificar se seria possível.

O objetivo dessa etapa seria utilizar a aplicação em um bar ou restaurante real, para assim, analisar seu comportamente no ambiente para o qual foi desenvolvida. Porém, essa etapa não foi executada, devido a problemas com a *internet* do bar escolhido.

4.8 ANÁLISE DOS RESULTADOS OBTIDOS

Após um tempo de acompanhamento foi necessário verificar se o sistema cumpriu corretamente sua proposta inicial. O objetivo dessa etapa foi transcrever os resultados obtidos nas etapas anteriores, para relatar o sucesso ou o fracasso da aplicação desenvolvida.

5 DESENVOLVIMENTO

Nesse capítulo será abordado o desenvolvimento da aplicação, contento detalhes da implementação.

5.1 ARQUITETURA DA SOLUÇÃO PROPOSTA

Com o intuito de modularizar o código da aplicação, foi tirado proveito dos *namespaces* [1] que o Ruby On Rails permite criar. Cada *namespace* implementado é um módulo do sistema. Com isso, o protótipo foi dividido nos módulos à seguir:

- **Admin**: Esse módulo contém a área do gerente, por meio da qual esse ator deve executar suas ações no sistema, tais como: cadastrar novos produtos, marcar itens como indisponíveis, entre outras;
- **Cook**: Esse módulo é responsável pela interação do cozinheiro com o sistema. Por meio desse módulo o cozinheiro tem acesso a sua tela, pela qual altera o estado do pedido;
- **Barman**: Módulo responsável pela interação do *barman*. Por meio desse módulo, o *barman* consegue controlar seus pedidos e alterar seus estados;
- **Cashier**: Módulo responsável pela interação do caixa. Por meio desse módulo, o caixa tem acesso a sua área aonde pode fechar os pedidos;
- **API**: Módulo responsável pela interação do Emberjs com o *backend* da aplicação. Por meio desse módulo, o Emberjs consegue consultar dados do *backend*, colocando-os disponiveis para o cliente.

Cada módulo mantém de forma isolada o código responsável por cumprir seus determinados requisitos.

[1] `http://guides.rubyonrails.org/routing.html#controller-namespaces-and-routing`

5.1.1 ATORES E SEUS PAPÉIS NA APLICAÇÃO

A aplicação proposta deve atuar em bares e restaurantes. Então, com base nesse escopo, os atores que interagem com o sistema são os seguintes:

- Cliente;
- Garçom;
- *Barman*;
- Caixa;
- Cozinheiro;
- Gerente.

Seus papéis estão ilustrados na Figura 5:

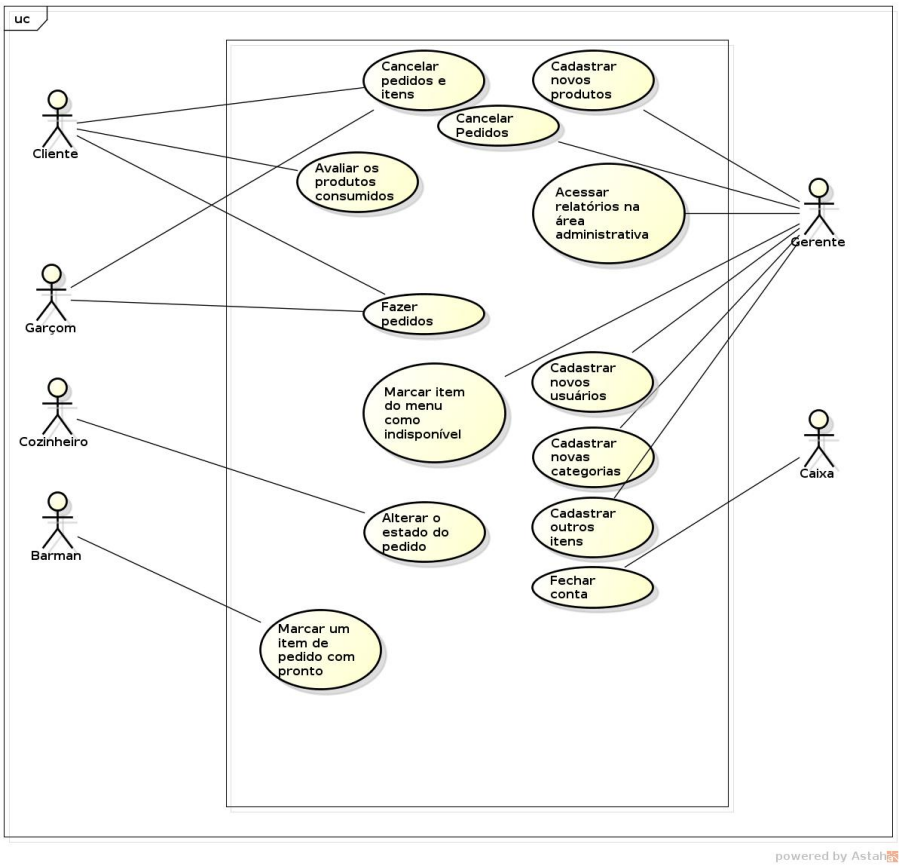

Figura 5: Diagrama de casos de uso.

Fonte: O autor

O papel do **cliente** permite a ele fazer os pedidos, avaliar os produtos consumidos e também cancelar pedidos e itens.

O **garçom** pode fazer pedidos, cancelar itens e pedidos, conforme solicitação do cliente. Ao ***barman*** é permitido marcar as bebidas como prontas, para que o cliente possa acompanhar seu pedido. O papel de **caixa** é de cuidar dos pagamentos dos pedidos por parte dos clientes.

O **cozinheiro** deve marcar a comida do pedido como pronta. Com isso, o cliente é notificado de que seu pedido está pronto por completo ou parcialmente, e o garçom sabe que deve entregar o pedido.

O **gerente** tem um papel importante na área administrativa. Por meio da qual, pode cadastrar produtos, usuários e categorias. Também pode acessar os relatórios da aplicação que estão disponíveis apenas na área administrativa.

5.2 SCRUM NA PRÁTICA

A metodologia SCRUM foi empregada para o desenvolvimento do protótipo, servindo como um guia no processo de desenvolvimento, embora nem todos os seus recursos e premissas tenham sido utilizadas.

Foram escritas as história que se encontram na Tabela 2 e colocadas em uma aplicação *online* chamada Trello[2]. Essas histórias foram foram utilizadas como base para o desenvolvimento das funcionalidades na aplicação.

[2] `https://trello.com`

Tabela 2: Histórias do Protótipo Desenvolvido

História	Descrição
HS 1	O sistema deve permitir o cadastro e a manutenção de novos usuários
HS 2	O sistema deve permitir o cadastro e a manutenção de novas categorias
HS 3	O sistema deve permitir o cadastro e a manutenção de novos produtos
HS 4	O sistema deve possuir uma interface para fazer novos pedidos
HS 5	O sistema deve permitir que um pedido seja cancelado por completo
HS 6	O sistema deve permitir que apenas alguns itens do pedido sejam cancelados
HS 7	O sistema deve oferecer uma interface para avaliação dos produtos consumidos
HS 8	O sistema deve permitir retirar itens do menu temporariamente
HS 9	O sitema deve permitir alterar o estado do pedido
HS 10	O sistema deve oferecer uma interface de acesso aos relatórios gerenciais
HS 11	O sistema deve oferecer uma interface para marcar uma conta como paga
HS 12	O sistema deve permitir o cadastro e manutenção de itens que não estaram no menu

O Trello possui um Kanban[3], que foi muito utilizado para armazenar as histórias do *product backlog*. E controlar as que estavam sendo desenvolvidas e as que já estavam implementadas por completo.

Não houveram *sprints*, em seu lugar foram utilizadas reuniões informais com o orientador. Afim de mostrar as novas funcionalidades implementadas, fazer pequenos testes e discutir melhorias ou mudanças.

5.3 TECNOLOGIAS ENVOLVIDAS

Foram utilizadas diversas tecnologias para a conclusão da aplicação proposta. Dentre elas destacam-se:

- *Websockets*;
- Ruby On Rails;
- Emberjs;

[3] `https://www.versionone.com/what-is-kanban/`

- Postgresql.

Segundo (IETF, 2011) *websocket* é um protocolo de comunicação, que tem como objetivo, estabelecer uma forma de comunicação que atue nas duas direções entre um *browser* e um servidor remoto. Foram utilizados *websockets* para fazer a comunicação em tempo real, quando o cliente finaliza o pedido, quando o pedido tem seu estado alterado pelo cozinheiro ou *barman*, e quando é entregue pelo garçom.

Os *websockets* fazem a comunicação por meio de canais. Para cada uso acima citado, foi criado um canal, no qual o Ruby On Rails publica uma informação sempre que necessário. O Emberjs receberá essa informação em formato JSON, e assim poderá informar o cliente automaticamente. Graças aos *websockets* o cliente não precisa carregar novamente o *browser* para acompanhar seu pedido. Isso facilita o uso do sistema, por exigir menos ações do usuário.

O *framework* Ruby On Rails, foi utilizado para fazer o lado do servidor da aplicação. Nele estam localizados os módulos citados na seção 5.1. Além de cuidar das regras de negócio o *framework* também é responsável por persistir dados no banco de dados.

O *framework* Emberjs, foi utilizado para fazer a interface com o cliente. Seu papel é trabalhar no lado do cliente das interfaces do cliente, se comunicando com o servidor por meio do módulo **API**.

Segundo (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2016), o Postgresql é um banco de dados relacional *open source*. Suporta os principais tipos de dados e possui as propriedades *Atomicity, Consistency, Isolation, Durability* (ACID)[4].

Devido as razões acima citadas, foi escolhido como Sistema Gerenciador de Banco de Dados (SGBD) da aplicação desenvolvida.

5.4 BANCO DE DADOS DA APLICAÇÃO

A seguir é apresentado o diagrama do banco de dados do projeto. Por meio dele, é possível armazenar e posteriormente extrair as informações necessárias para cumprir os objetivos propostos na Seção 1.1.2. Como é apresentado na Figura 6.

[4]Características que tornam as transações de um banco de dados confiáveis.

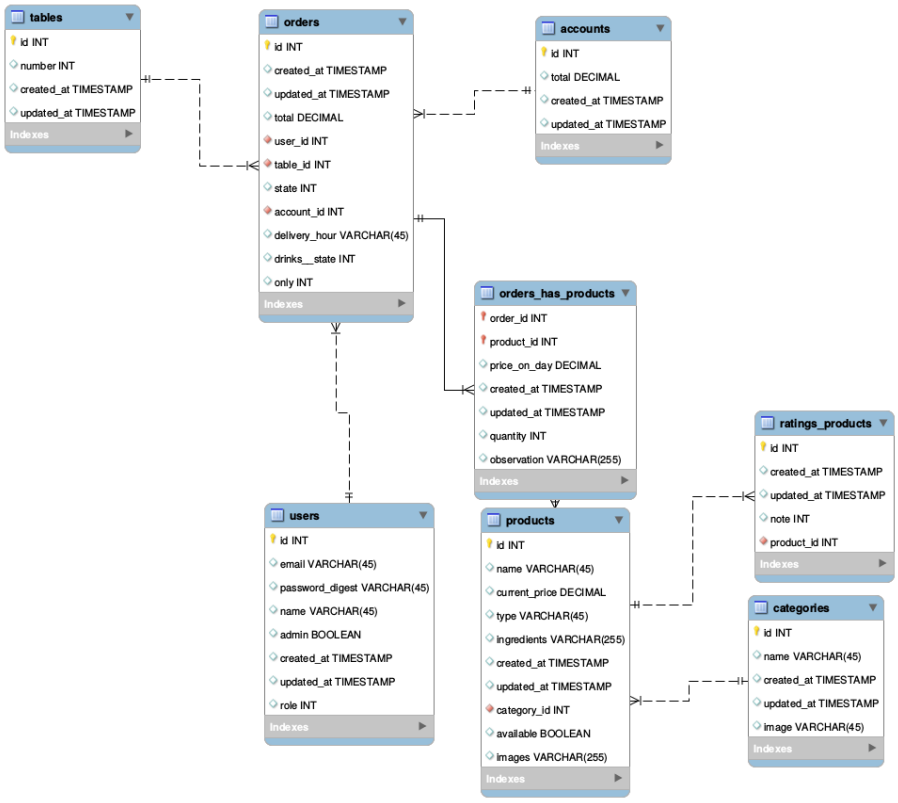

Figura 6: Banco de dados.

Fonte: O autor

O primeiro objetivo à ser cumprido que é o número **2**. O cliente deveria visualizar o menu com os produtos para então escolher e fazer o pedido. Os produtos seriam separados por categorias no menu.

Para cumprir esse objetivo, foram criadas as tabelas: *products* e *categories*. Na tabela *products*, são armazenados os dados dos produtos.

O relacionamento entre categoria e produto, pemitire separar os produtos por categoria no menu. Na tabela *categories* que salva as categorias, é armazenado o nome da categoria.

O cliente também pode avaliar os produtos, essa avaliação consiste em uma nota de 1 a 5. Para fazer essa avaliação foi criada a tabela *rattings_products*. Que armazena uma identificação do produto e a nota recebida.

Para cumprir o objetivo número **3**, foi necessário armazenar: os pedidos feitos, quem fez cada pedido e os itens de cada pedido. Para armazenar os pedidos foi criada a tabela *orders*.

Para identificar quem fez o pedido foi criada a tabela *tables*. E nela são guardados o número da mesa.

Os itens de cada pedido são salvos na tabela *orders_has_products*, que guarda a identificação do pedido e do produto, e também o preço do produto no dia em que foi pedido e uma observação que o cliente pode enviar a quem prepara seu pedido.

Com o pedido salvo em uma tabela, é possível exibi-los em uma tela para os cozinheiros. Cumprindo assim o objetivo número **3**.

Graças a tabela *orders_has_products* o objetivo número **4** e a parte dos produtos mais vendidos do objetivo **5** são cumpridos. É necessário, fazer consultas a base de dados. E o *framework* Ruby on Rails possuiu um módulo chamado *ActiveRecord*[5] que fornece uma forma simples para fazer consultas. E para mostrar os pratos mais bem avaliados, foi preciso consultar a tabela *rattings_products*.

Para tornar a aplicação funcional, foi adicionada a funcionalidade da conta. Por meio da qual é possível receber os pagamentos dos clientes. As contas são representadas no banco pela tabela *accounts*, que salvam o valor total da conta.

5.5 PROTÓTIPO DA INTERFACE GRÁFICA

Nessa Seção são detalhadas alguns protótipos de telas para a aplicação proposta.

5.5.1 TELA DO PRODUTO

A tela por meio da qual o cliente pode ver os produtos, é ilustrada na Figura 3.

[5] `http://guides.rubyonrails.org/active_record_basics.html`

Figura 7: Tela de visualização do produto.

Fonte: O autor

Como é possível ver na Figura 7 a tela do produto teve foco na simplicidade. Contendo as informações básicas como nome, preço e uma foto do produto. Para visualizar mais informações e fotos do produto, o cliente deveria clicar no botão com o sinal de adição, situado no canto superior direito. A tela também possui um botão para voltar e adicionar o produto ao pedido.

5.5.2 TELA DO COZINHEIRO E BARMAN

As telas que são utilizas pelo cozinheiro e *barman* são ilustradas, nas Figuras 4 e 5.

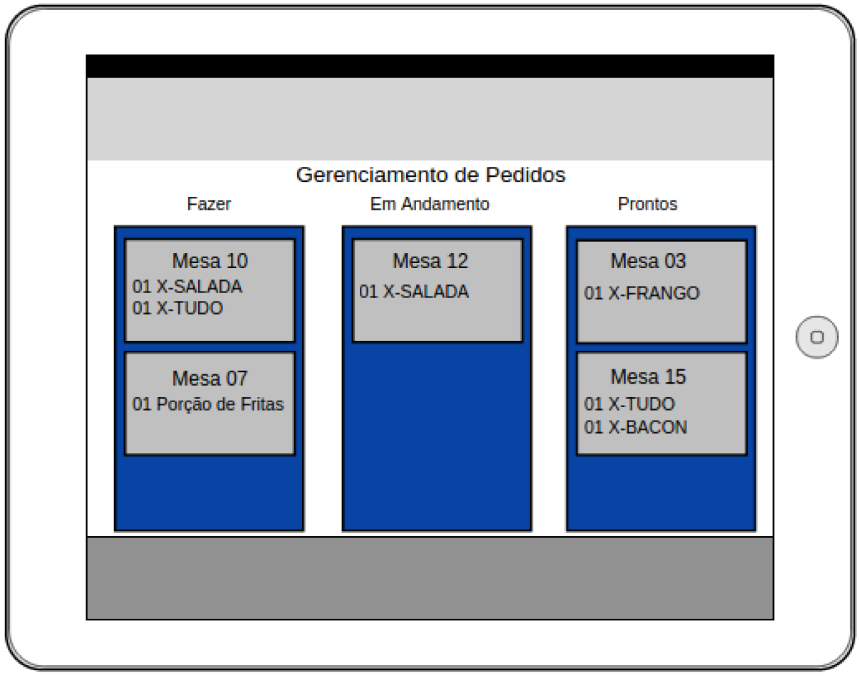

Figura 8: Tela de visualização do cozinheiro.

Fonte: O autor

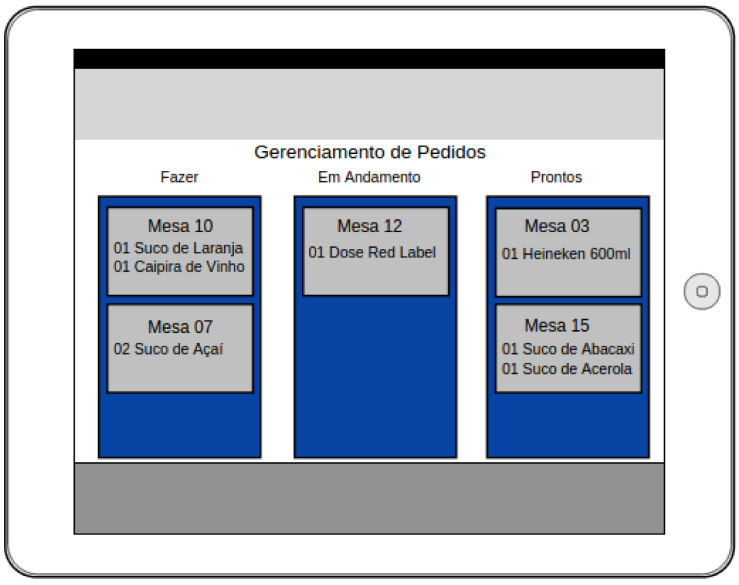

Figura 9: Tela de visualização do *barman*.

Fonte: O autor

As duas telas apresentadas são idênticas. A única diferença está no conteúdo, a do cozinheiro tem apenas comida e a do *barman* apenas bebida. As telas possuem 3 partes, pedidos

para fazer, pedidos em andamento e pedidos prontos.

Quando o pedido é confirmado pelo cliente ele se junta aos pedidos a fazer. E quando o cozinheiro ou *barman* começar a prepará-lo, deve tocar na tela e arrastar o pedido para junto dos pedidos em andamento.

Quando o pedido está pronto, quem o preparou deve arrastá-lo para junto dos pedidos prontos. Isso é importante, porque quando o pedido estiver nessa área o garçom será notificado de que está pronto.

5.5.3 TELA DO GARÇOM

Na Figura 10, é ilustrada a tela pela qual o garçom interage com o sistema.

Figura 10: Tela de visualização do garçom.

Fonte: O autor

No topo da tela do garçom é possível visualizar o nome do garçom e um *link* para sair do sistema. Logo abaixo, tem uma listagem dos pedidos prontos, que já podem ser entregues aos clientes. Quando o garçom entregar um pedido em sua mesa correspondente, deve pressionar o botão entregue. Dessa forma é possível marcar o tempo que o pedido demorou desde seu

preparo até a entrega.

O garçom também poderia fazer buscas nos pedidos que estão em andamento e para fazer. Isso poderia ser feito por meio do campo para buscas, situado no canto superior esquerdo da tela.

6 O SISTEMA EICHEF

Eichef é o nome do protótipo desenvolvido nesse projeto. Trata-se de uma ferramenta de auxílio do atendimento com foco em bares e restaurantes. Que visa automatizar os pedidos realizados dentro de bares e restaurantes, por um sistema *web* acessado por dispositivos móveis dos clientes ou ainda oferecido pelo estabelecimento.

6.1 DINÂMICA DE FUNCIONAMENTO

Nessa seção será descrita brevemente a dinâmica de funcionamento do protótipo desenvolvido. Ilustrada abaixo na Figura 11.

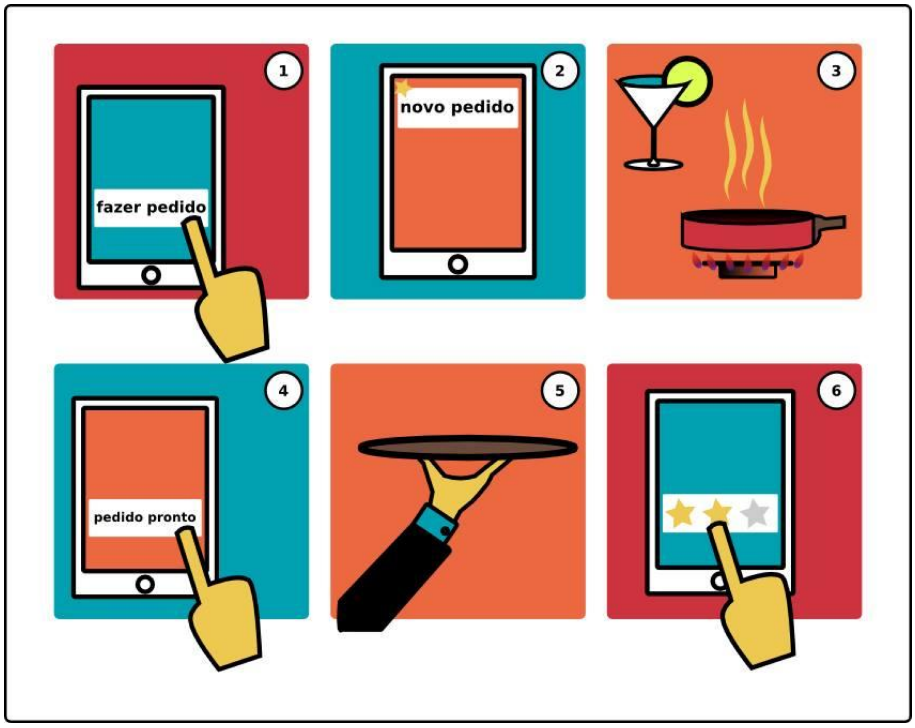

Figura 11: Dinâmica do funcionamento da protótipo desenvolvido.

Fonte: O autor

Na ação número **1** o cliente interage com a aplicação para fazer seu pedido. Após o

pedido ser confirmado, o mesmo é separado. As bebidas são enviadas ao *barman* e a comida ao cozinheiro. Ambos recebem notificações em suas telas, como é demonstrando na etapa **2**.

Após serem notificados, eles devem preparar o que foi enviado. Assim como ilustra a etapa **3**. Quando terminarem seu trabalho, eles devem comunicar no sistema que o produto está pronto para ser entregue ao cliente. Isso é demonstrado na etapa **4**.

Com isso, o garçom recebe uma notificação em seu celular, avisando que o pedido ou parte dele está pronto. Então ele faz a entrega ao cliente, como demonstra a etapa **5**.

Depois que o pedido é entregue por completo, o cliente pode avaliar os produtos que consumiu e o atendimento prestado. Isso é ilustrado na etapa **6**.

6.2 INTERAÇÃO DO CLIENTE COM O SISTEMA

Para que o cliente possa acessar a aplicação deve fazer a leitura de um *QR Code*. O *QR Code* estaria na mesa do cliente, assim como ilustrado na Figura 12.

Figura 12: Exemplo de QR Code utilizado para acessar a aplicação.
Fonte: O autor

Após fazer a leitura o cliente é redirecionado para a página inicial do Eichef. Mostrada na Figura 13.

Figura 13: Página inicial de acesso do cliente.

Fonte: O autor

Nessa tela o cliente pode ir para o menu, que é mostrado na Figura 14.

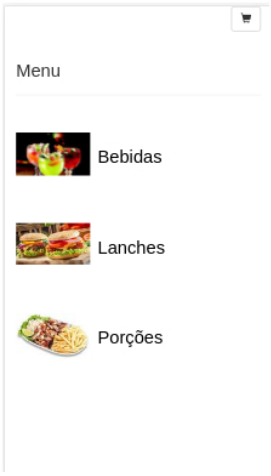

Figura 14: Página do menu de acesso do cliente.

Fonte: O autor

Na tela do menu o cliente pode escolher uma categoria de produtos para pedir. Supondo que o cliente escolhesse a categoria **bebidas**, ele é redirecionado para a tela apresentada na Figura 15.

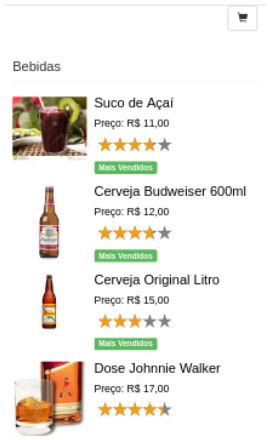

Figura 15: Página da categoria de acesso do cliente.

Fonte: O autor

Nota-se que nessa tela são exibidos os produtos da categoria escolhida anteriormente. Também é possível observersar a média das avaliações anteriores dos produtos. Assim como uma indicação dos produtos mais vendidos. Nesse caso, supondo que o cliente escolha um produto, é redirecionado para a tela ilustrda na Figura 16.

Figura 16: Página da produto de acesso do cliente.

Fonte: O autor

Como é possível observar na ilustração o cliente pode visualizar uma foto do produto, seu nome e preço. No exemplo observado na Figura 16, não foram cadastrados ingredientes

pelo gerente. Por essa razão não aparece o sinal de mais no canto superior direito.

Caso o cliente decidice, clicar em **Adicionar ao Pedido**, seria redirecionado ao carrinho. Como é mostrado na Figura 17.

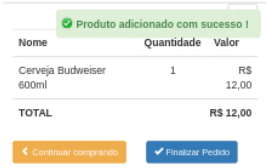

Figura 17: Página do carrinho de acesso do cliente.

Fonte: O autor

Nessa tela do carrinho o cliente pode voltar para o menu, editar ou excluir algum item do pedido ou finalizasse o pedido. Quando o cliente finalizasse o pedido, seria redirecionado para uma tela de espera. Ilustrada abaixo na Figura 18.

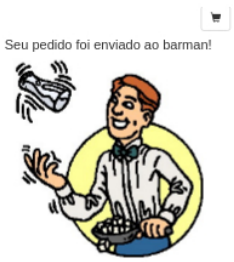

Figura 18: Página de espera após enviar o pedido ao *barman*.

Fonte: O autor

Por meio dessa tela, o cliente pode acompanhar o andamento do seu pedido. Quando o *barman* começar a preparar o pedido do cliente, a tela é atualizada. Como mostrado na Figura 19.

Figura 19: Página de espera, enquanto *barman* prepara o pedido.

Fonte: O autor

Quando o *barman* finaliza o pedido, o cliente é notificado. Assim como mostrado na Figura 20.

Figura 20: Página de espera, após o *barman* finalizar o pedido.

Fonte: O autor

Após isso o garçom teria que entregar o pedido. Quando ele entregasse o pedido, o cliente seria redirecionado para a tela de avaliação dos produtos. Como mostrado na Figura 21.

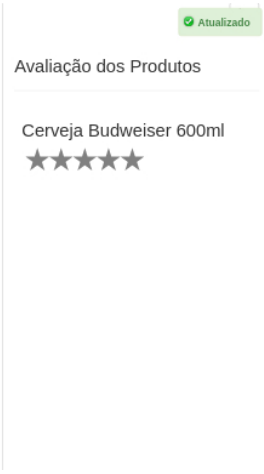

Figura 21: Página de avaliação dos produtos.

Fonte: O autor

Nessa tela o cliente pode avaliar os itens, com a nota de 1 a 5. Após avaliar todos os itens o cliente é redirecionado novamente para a página inicial.

6.3 INTERAÇÃO DO GARÇOM COM O SISTEMA

Para que o garçom comece a utilizar o sistema primeiro é necessário fazer sua autenticação. Como é visto, na tela de autenticação da Figura 22.

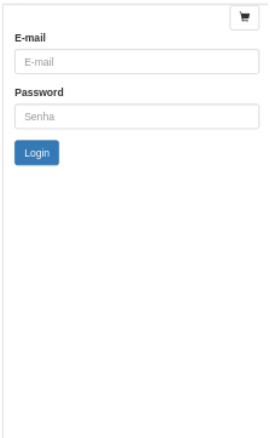

Figura 22: Página de autenticação do garçom.

Fonte: O autor

Após sua autenticação, o garçom é redirecionado para a tela por meio da qual acompanha os pedidos finalizados. Como ilustra a Figura 23.

Figura 23: Página de pedidos do garçom.

Fonte: O autor

Por meio dessa tela o garçom sabe quais pedidos estam prontos. E pode entrega-los aos clientes. Após a entrega precisa apenas clicar em **Entregar**. O garçom também pode fazer um pedido em nome de uma mesa. Basta clicar em **Novo Pedido**, que assim é aberto um modal com as mesas do estabelecimento. Como mostra a Figura 24.

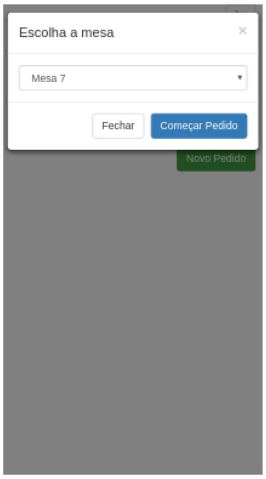

Figura 24: Página de mesas do estabelecimento na área do garçom.

Fonte: O autor

Por meio dessa tela, o garçom pode escolher uma mesa. E logo após, fazer um pedido da mesma forma como uma cliente faria, como foi demonstrado na Seção 6.2. Com exceção de que ao invés de ser enviado à tela de espera, retorna a tela de pedidos do garçom.

6.4 INTERAÇÃO DO BARMAN COM O SISTEMA

Para que o *barman* acesse o sistema, deve estar autenticado. Por meio da página de autenticação, ilustrada na Figura 25.

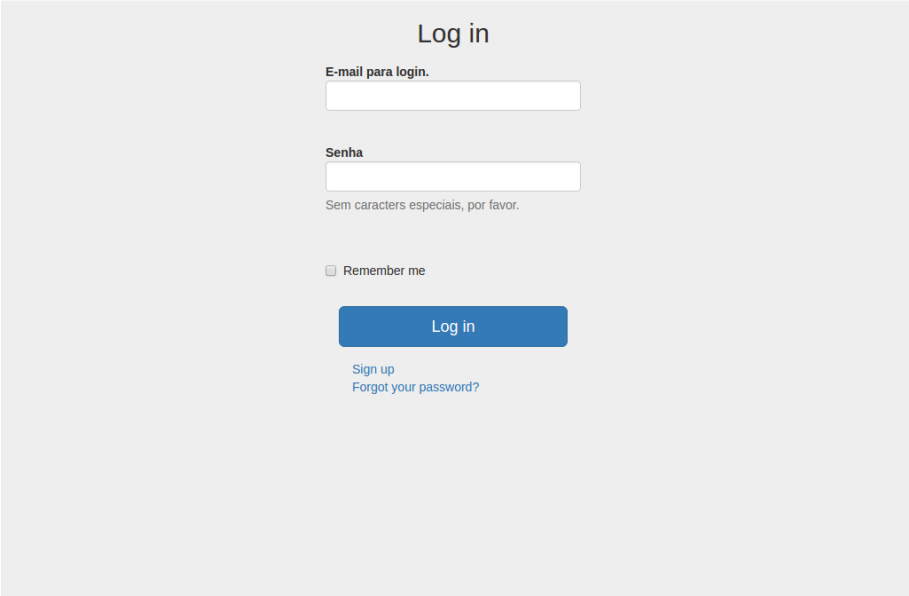

Figura 25: Página de login para o gerente, *barman* e cozinheiro.

Fonte: O autor

Por essa mesma página, o gerente e cozinheiro também fazem suas autenticações. Após fazer sua autenticação o *barman* é redirecionado para sua tela por onde acompanha os pedidos. Tela ilustrada na Figura 26.

Figura 26: Página de acompanhamento dos pedidos do *barman*.

Fonte: O autor

Por meio dessa tela, o *barman* consegue ver os novos pedidos na aba **Fazer**. Também consegue controlar os pedidos que esta fazendo e os que já terminou, pelas abas **Em Andamento** e **Pronto**. Alternando os pedidos de lugar utilizando as setas. Diferente do planejado

anteriormente na Seção 5.5.2, foram usadas setas no lugar de arrastar os pedidos. Para tentar dimunuiu o esforço do usuário.

6.5 INTERAÇÃO DO COZINHEIRO COM O SISTEMA

O cozinheiro também precisa fazer a autenticação assim como o *barman*. E para isso utiliza a mesma tela mostrada na Figura 25. Sua tela de acompanhamento dos pedidos, também é muito semelhante. Com a diferença de que ao invés de bebida os pedidos teriam comida. Assim, como mostrado na Figura 27.

Figura 27: Página de acompanhamento dos pedidos do cozinheiro.

Fonte: O autor

6.6 INTERAÇÃO DO GERENTE COM O SISTEMA

Para que o gerente acesse o sistema é necessário fazer a autenticação utilizando a mesma tela que o *barman* e cozinheiro. Ilustrada na Figura 25. Após isso, é redirecionado a sua página inicial. Na qual pode observar os gráficos com informações para tomada de decisão.

A Figura 28 mostra o tempo médio mensal de entrega dos pedidos. Com esse gráfico o gerente pode acompanhar o tempo médio em minutos que seus clientes tem esperado, do momento em que fizeram seus pedidos até a hora em que o pedido foi entregue. Com essa informação o gerente pode analisar o desempenho de sua equipe.

Figura 28: Gráfico com o tempo médio de entrega dos pedidos.

Fonte: O autor

A Figura 29 mostra à esquerda os produtos mais vendidos do estabelecimento. E a direita os produtos mais bem avaliados pelos clientes.

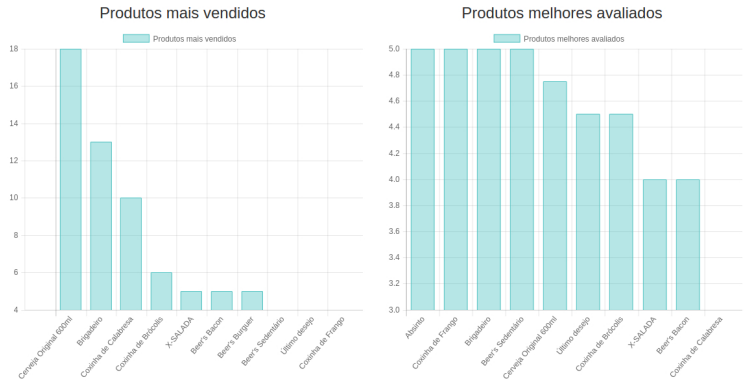

Figura 29: Gráficos com os produtos mais vendidos e mais bem avaliados.

Fonte: O autor

A Figura 30 mostra à esquerda o *ranking* dos produtos menos vendidos e à direita os produtos com as piores avaliações por partes dos clientes. Essa informação pode ser de grande utilizadade para o gerente. Porque assim, ele pode por exemplo, remover os produtos que não estão sendo vendidos. E trocar por novos produtos na tentativa de satisfazer seus clientes

Figura 30: *Ranking* com os produtos menos vendidos e com piores avaliações.

Fonte: O autor

7 EXPERIMENTO DE ACEITAÇÃO

Foi realizado um experimento com o protótipo desenvolvido para validar a pesquisa. O experimento foi realizado com alunos da UTFPR - Campûs Guarapuava em uma simulação após o professor da disciplina ceder sua aula.

7.1 SUJEITOS

O experimento contou com alunos do 4° período do curso de Tecnologia em Sistemas para a Internet e outros alunos voluntários, totalizando 12 participantes.

7.2 AMBIENTE DE APLICAÇÃO

O experimento foi realizado no laboratório de informática aonde os alunos estavam assistindo a aula. Antes da definição do roteiro, foram cadastrados categorias, produtos e usuários, necessários para o experimento. O roteiro do experimento seguiu a sequência a seguir:

1. Inicialmente foi feita uma apresentação pessoal. Logo após foi explicado o propósito do experimento, que era validar uma pesquisa de Trabalho de Conclusão de Curso (TCC);
2. Então, foi realizada uma demonstração do protótipo em funcionamento. Foi exibido no *data show* todo o processo de fazer um pedido e apresentado como todos os atores envolvidos deveriam atuar no sistema;
3. Foram designados os seguintes papéis um aluno como gerente, um aluno como *barman*, um aluno como garçom, um aluno como cozinheiro e o restante foram os clientes;
4. Após a divisão dos papéis, os clientes receberam seus *QR codes* e começaram a fazer os pedidos, simulando o funcionamento de um bar;
5. A cada intervalo de tempo eram trocados os papéis para que todos pudessem conhecer todo o sistema;

6. Foi realizada uma conversa com os alunos sobre sua opnião geral sobre o sistema, se tiveram alguma dificuldade com o uso;

7. Foram enviados os questionários que se encontram em anexo para avaliação do Eichef.

7.3 EXPERIMENTO

O experimento foi realizado com consentimento do Prof. Eleandro Maschio Krynski. Que já havia informado os alunos um dia antes sobre o experimento e pediu a eles que instalassem um aplicativo para leitura de *QR Code* em seus *smartphones*. A turma possuía 12 alunos regularmente matriculados, porém apenas 8 estavam presentes naquela dia, somando com os 4 voluntários que chegaram logo em seguida.

O experimento correu de uma forma muito positiva. Os alunos foram muito participativos e não tiveram problemas para utilizar o sistema. Eles fizeram diversos pedidos com diferentes produtos e aguardaram sua entrega, avaliando depois. O alunos também cadastraram novos produtos, categorias e removeram produtos do menu.

Houveram diversos elogios sobre a ideia do sistema, sobre o funcionamento e da facilidade de uso. Também houveram boas sugestões de melhoras. Ao final, houve uma conversa com os alunos e após o envio dos questionários. Em seguida o experimento foi encerrado.

7.4 RESULTADOS

Além do experimento relatado acima, foi consultado o proprietário de um bar local. O mesmo fez uma análise do sistema, elogiando pela facilidade de uso e pela organização que o mesmo oferece.

Pode-se observar que de um ponto de vista geral o Eichef teve uma boa aceitação. Analisando os dados da Tabela 3. Pode-se concluir que a área administrativa cumpriu seu papel, fornecendo informações para tomada de decisão, controlar os pedidos e produtos do menu. O proprietário do bar gostou dos gráficos fornecidos, ressaltando que ajudaria a saber quais produtos poderia remover do cardápio.

Observando os dados da Tabela 4, pode-se concluir que o sistema ajudaria os cozinheiros e *barmans*. O proprietário do bar, ressaltou que em alguns momentos a tela do *barman* pudesse não ser tão útil no caso de uma bebida rápida, como por exemplo, pegar uma cerveja para o cliente. Porém, essa situação poderia ser contornada com facilidade.

Em relação aos garçons, olhando para a Tabela 5. Pode-se conlcuir que sua área facilitaria seu trabalho de uma forma geral. Pelo fato de nem todas as respostas serem positivas, conclui-se que seria necessário algumas melhorias, tais como, alerta sonoro para o garçom quando um pedido novo chegasse.

Já na área do cliente, olhando para a Tabela 6, pode-se notar que o menu interativo exibiria os itens de forma clara. Porém algumas melhorias na navegação seriam necessárias. Segundo algumas sugestões fornecidas pelos participantes, como por exemplo, adicionar um filtro de busca e adicionar subcategorias. A tela de acompanhamento do pedido e o carrinho foram satisfatórios.

A avaliação dos produtos expressou bem a satisfação do cliente e pode ser melhorada adicionando a possibilidade de comentar o produto e oferecer sugestões.

De uma forma geral o Eichef foi bem sucedido em seus objetivos. E haveria uma melhora em sua usabilidade, após algumas melhorias sugeriadas pelos participantes.

Tabela 3: Questionário sobre a área do gerente.

Questões Área Administrativa	Alternativas	N° Votos
	Péssimo	0
	Ruim	0
As informações dispostas nos gráficos foram úteis para tomada de decisão ?	Regular	1
	Ótimo	2
	Abstenções	9
	Péssimo	0
	Ruim	0
O controle de pedidos em sua área ajudou você a monitorar os mesmos ?	Regular	1
	Ótimo	2
	Abstenções	9
	Péssimo	0
	Ruim	0
O controle de produtos foi útil para controlar e apresentar os produtos aos clientes ?	Regular	0
	Ótimo	3
	Abstenções	9

Tabela 4: Questionário sobre as áreas do cozinheiro e *barman*.

Questões Cozinheiro e *Barman*	Alternativas	N° Votos
	Péssimo	0
	Ruim	0
O sistema trouxe facilidade para seu trabalho ?	Regular	1
	Ótimo	3
	Abstenções	8
	Péssimo	0
	Ruim	0
Você achou complicado gerenciar os pedidos pela tela ?	Regular	1
	Ótimo	3
	Abstenções	8

Tabela 5: Questionário sobre a área do garçom.

Questões Cozinheiro e *Barman*	Alternativas	N° Votos
	Facilitou	4
O sistema facilitou ou complicou suas tarefas de rotina ?	Complicou	0
	Abstenções	8
	Péssimo	0
	Ruim	0
Conforme a pergunta anterior, quanto facilitou ou complicou ?	Regular	3
	Ótimo	1
	Abstenções	8
	Péssimo	0
	Ruim	0
A tela do garçom facilitou seu controle sobre os pedidos já preparados ?	Regular	3
	Ótimo	1
	Abstenções	8
	Péssimo	0
	Ruim	0
O modo de pedido facilitou seu trabalho na hora de fazer os pedidos pelos clientes ?	Regular	2
	Ótimo	2
	Abstenções	8

Tabela 6: Questionário sobre a área do cliente.

Questões Cliente	Alternativas	N° Votos
	Péssimo	0
	Ruim	1
O menu interativo ofereceu facilidade na hora de fazer o pedido ?	Regular	3
	Ótimo	3
	Abstenções	5
	Péssimo	0
	Ruim	1
O menu interativo mostrou de forma clara os itens disponíveis para consumo no estabelecimento ?	Regular	1
	Ótimo	5
	Abstenções	5
	Péssimo	1
	Ruim	0
A tela de acompanhamento do pedido, ajudou a monitorar o andamento de seu pedido	Regular	1
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
O carrinho de compras facilitou seu controle dos itens que foram pedidos ?	Regular	2
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
O carrinho de compras facilitou seu controle de quanto iria gastar ?	Regular	0
	Ótimo	7
	Abstenções	5
	Péssimo	0
	Ruim	0
A média de avaliações mostradas em cada produto, foram uteis na sua tomada de decisão sobre quais produtos pedir ?	Regular	2
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
A avaliação do produto expressou bem seu grau de satisfação quanto ao mesmo ?	Regular	1
	Ótimo	6
	Abstenções	5

8 CONSIDERAÇÕES FINAIS

8.1 REAFIRMAÇÃO DA CONTRIBUIÇÃO

O objetivo do Eichef foi ajudar os clientes no processo de fazer pedidos dentro de estabelecimentos como bares e restaurantes. Na tentativa de diminuir o esforço tanto do cliente, quanto dos garçons, *barmans* e cozinheiros. E ainda apresentar as informações guardados em meio a esse processo para auxiliar o gerente na tomada de decisões.

Após realizado o experimento e consultado um proprietário de um bar local, pode-se concluir que o protótipo teve êxito em seus objetivos. Oferecendo aos clientes uma interface simples para que não fosse necessário muito esforço para conhecer os produtos, as categorias e fazer um pedido. Além de melhorar a comunicação entre os clientes e o gerente do estabelecimento, pois, o protótipo oferece ao cliente a possibilidade de avaliar os produtos, demonstrando sua satisfação ou insatisfação com os mesmos.

Outro aspecto a ser levado em consideração foi o uso QR Code, que facilitou o acesso a aplicação por parte do cliente. Tornando viável o uso de SPA pelo fato de o cliente não precisar digitar uma *Uniform Resource Locator* (URL) em seu navegador.

A combinação de uso do SPA com o QR Code foi sem dúvidas o maior diferencial em relação ao estado da arte. Além de demonstrar uma forma alternativa aos aplicativos nativos, também pode servir como uma referencia a futuros trabalhos correlatos.

8.2 TRABALHOS FUTUROS

Esta seção tem por objetivo apresentar melhorias futuras para o protótipo desenvolvido.

1. **Melhorias na segurança:** como tratava-se de um protótipo que não necessariamente seria utilizado em ambiente real. A segurança acabou sendo deixada em segundo plano. Seria necessário criptografar os dados enviados do Emberjs para a API. A API atualmente está pública, seria necessário criar permissões por *token* e criptografar dados transmitidos

pelos *websockets*;

2. **Criação de uma identidade visual:** o protótipo possui um *layout* desenvolvido em *bootstrap*, seria necessário desenvolver um *layout* novo especialmente para a aplicação. Além de criar um logotipo personalizado para o mesmo;

3. **Notificações para o cliente e o garçom:** quando o cliente está na área de espera e o garçom está em sua de acompanhamento de pedidos. A única forma de aviso que recebem é uma pequena notificação no canto da tela. Para melhorar isso poderia ser adicionado um alerta sonoro juntamente com a janela;

4. **Identificação do cliente:** para identicar o cliente que fez o pedido e assim no futuro manter um histório de sua preferência. Para isso, seria necessário utilizar alguma rede social, como por exemplo o *facebook*. Assim o cliente poderia fazer login pelo *facebook*.

5. **Repetir o pedido:** adicionar a funcionalidade de o cliente poder repetir seu último pedido;

6. **Busca de itens:** adicionar um filtro de buscas, por meio do qual, o cliente digitaria um termo e o sistema buscaria por suas ocorrências;

7. **Adicionar subcategorias:** para facilitar a navegação do cliente no menu interativo, adicionar subcategorias para ajudar os clientes na seleção dos produtos melhorando a usabilidade.

REFERÊNCIAS

ANICHE, M.; CORBUCCI, H. **Test-Driven Development: Test e Design no Mundo Real com Ruby**. [S.l.]: CASA DO CODIGO, 2014. ISBN 9788566250435.

ASTELS, D.; CHELIMSKY, D. **The RSpec Book: Behaviour Driven Development with RSpec, Cucumber, and Friends**. Pragmatic Bookshelf, 2010. (Pragmatic Bookshelf Series). ISBN 9781934356371. Disponível em: <https://books.google.com.br/books?id=0rxoPgAACAAJ>.

CRAVENS, J.; BRADY, T. **Building Web Apps with Ember.js**. O'Reilly Media, 2014. ISBN 9781449370909. Disponível em: <https://books.google.com.br/books?id=dNr-AwAAQBAJ>.

EZEE TECHNOSYS. **Página das features**. 2015. Disponível em: <http://www.ezeeimenu.com/features.php>.

EZEE TECHNOSYS. **Página inicial**. 2015. Disponível em: <http://www.ezeeimenu.com/>.

FINK, G.; FLATOW, I. **Pro Single Page Application Development: Using Backbone.js and ASP.NET**. Apress, 2014. (EBL-Schweitzer). ISBN 9781430266747. Disponível em: <https://books.google.com.br/books?id=ayuLAwAAQBAJ>.

FREEMAN, S.; PRYCE, N. **Growing Object-Oriented Software, Guided by Tests**. Pearson Education, 2009. (Addison-Wesley Signature Series (Beck)). ISBN 9780321699763. Disponível em: <https://books.google.com.br/books?id=QJA3dM8Uix0C>.

GARCEZ, E. M. S.; FACHIN, G. R. B.; JUNIOR, P. P. A. Indicadores da qualidade em restaurantes: Um estudo de caso. **Revista de Ciências da Admnistração**, n. 3, Abril 2000.

HE LABS. **Desenvolvimento Àgil**. 2013. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>.

IETF. **The WebSocket Protocol**. 2011. Disponível em: <https://tools.ietf.org/html/rfc6455>.

KELONYE, M. **Mastering Ember.js**. Packt Publishing, 2014. (Community experience distilled). ISBN 9781783981991. Disponível em: <https://books.google.com.br/books?id=dY3jBAAAQBAJ>.

KOZLOWSKI, P. **Mastering Web Application Development with AngularJS**. Packt Publishing, 2013. (Community experience distilled). ISBN 9781782161837. Disponível em: <https://books.google.com.br/books?id=mZXjwz5X08EC>.

LERNER, A. **Ng-Book - the Complete Book on Angularjs**. Fullstack.io, 2013. ISBN 9780991344604. Disponível em: <https://books.google.com.br/books?id=I7UpnwEACAAJ>.

OLIVEIRA, K. D. **Recuperação de serviço no processo de atendimento em restaurante: Estudo de caso em Porto Alegre**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2002.

PEKUS. **Página inicial**. 2015. Disponível em: <http://www.pekus.com.br/cardapio.aspx>.

PSIU GARÇOM. **Página inicial**. 2015. Disponível em: <http://www.chamagarcom.ind.br/>.

RUBIN, K. **Essential Scrum: A Practical Guide to the Most Popular Agile Process**. Pearson Education, 2012. (Addison-Wesley Signature Series (Cohn)). ISBN 9780321700377. Disponível em: <https://books.google.com.br/books?id=3vGEcOfCkdwC>.

RUBY, S.; THOMAS, D.; HANSSON, D. **Agile Web Development with Rails 4**. Pragmatic Bookshelf, 2013. (Pragmatic Programmers). ISBN 9781937785567. Disponível em: <https://books.google.com.br/books?id=ymiqnAEACAAJ>.

SCOTT, E. **SPA Design and Architecture: Understanding Single Page Web Applications**. Manning Publications Company, 2015. ISBN 9781617292439. Disponível em: <https://books.google.com.br/books?id=v_oXrgEACAAJ>.

SKEIE, J. **Ember.js in Action**. Manning Publications Company, 2013. ISBN 9781617291456. Disponível em: <https://books.google.com.br/books?id=LYxRmwEACAAJ>.

SMART, J. **BDD in Action: Behavior-Driven Development for the Whole Software Lifecycle**. Manning Publications Company, 2014. ISBN 9781617291654. Disponível em: <https://books.google.com.br/books?id=2BGxngEACAAJ>.

STARZHINSKY, A. Blog, **AngularJS advantages and limitations**. 2015. Disponível em: <http://blog.softelegance.com/angularjs/angularjs-advantages-and-limitations/>.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **Página sobre**. 2016. Disponível em: <https://www.postgresql.org/about/>.

WATERS, J. **QR Codes For Dummies**. Wiley, 2012. (For Dummies). ISBN 9781118370711. Disponível em: <https://books.google.com.br/books?id=Nd_50ehL5EIC>.

ANEXO A – QUESTIONÁRIO SOBRE A ÁREA DO GERENTE

Tabela 7: Questionário sobre a área do gerente.

Questões Área Administrativa	Alternativas	N° Votos
	Péssimo	0
	Ruim	0
As informações dispostas nos gráficos foram úteis para tomada de decisão ?	Regular	1
	Ótimo	2
	Abstenções	9
	Péssimo	0
	Ruim	0
O controle de pedidos em sua área ajudou você a monitorar os mesmos ?	Regular	1
	Ótimo	2
	Abstenções	9
	Péssimo	0
	Ruim	0
O controle de produtos foi útil para controlar e apresentar os produtos aos clientes ?	Regular	0
	Ótimo	3
	Abstenções	9

ANEXO B – QUESTIONÁRIO SOBRE AS ÁREAS DO COZINHEIRO E *BARMAN*

Tabela 8: Questionário sobre as áreas do cozinheiro e *barman*.

Questões Cozinheiro e *Barman*	Alternativas	N° Votos
	Péssimo	0
	Ruim	0
O sistema trouxe facilidade para seu trabalho ?	Regular	1
	Ótimo	3
	Abstenções	8
	Péssimo	0
	Ruim	0
Você achou complicado gerenciar os pedidos pela tela ?	Regular	1
	Ótimo	3
	Abstenções	8

ANEXO C – QUESTIONÁRIO SOBRE A ÁREA DO GARÇOM

Tabela 9: Questionário sobre a área do garçom.

Questões Cozinheiro e *Barman*	Alternativas	N° Votos
	Facilitou	4
O sistema facilitou ou complicou suas tarefas de rotina ?	Complicou	0
	Abstenções	8
	Péssimo	0
	Ruim	0
Conforme a pergunta anterior, quanto facilitou ou complicou ?	Regular	3
	Ótimo	1
	Abstenções	8
	Péssimo	0
	Ruim	0
A tela do garçom facilitou seu controle sobre os pedidos já preparados ?	Regular	3
	Ótimo	1
	Abstenções	8
	Péssimo	0
	Ruim	0
O modo de pedido facilitou seu trabalho na hora de fazer os pedidos pelos clientes ?	Regular	2
	Ótimo	2
	Abstenções	8

ANEXO D – QUESTIONÁRIO SOBRE A ÁREA DO CLIENTE

Tabela 10: Questionário sobre a área do cliente.

Questões Cliente	Alternativas	N° Votos
	Péssimo	0
	Ruim	1
O menu interativo ofereceu facilidade na hora de fazer o pedido ?	Regular	3
	Ótimo	3
	Abstenções	5
	Péssimo	0
	Ruim	1
O menu interativo mostrou de forma clara os itens disponíveis para consumo no estabelecimento ?	Regular	1
	Ótimo	5
	Abstenções	5
	Péssimo	1
	Ruim	0
A tela de acompanhamento do pedido, ajudou a monitorar o andamento de seu pedido	Regular	1
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
O carrinho de compras facilitou seu controle dos itens que foram pedidos ?	Regular	2
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
O carrinho de compras facilitou seu controle de quanto iria gastar ?	Regular	0
	Ótimo	7
	Abstenções	5
	Péssimo	0
	Ruim	0
A média de avaliações mostradas em cada produto, foram uteis na sua tomada de decisão sobre quais produtos pedir ?	Regular	2
	Ótimo	5
	Abstenções	5
	Péssimo	0
	Ruim	0
A avaliação do produto expressou bem seu grau de satisfação quanto ao mesmo ?	Regular	1
	Ótimo	6
	Abstenções	5