CENTRO UNIVERSITÁRIO SÃO LUCAS JI-PARANÁ

IGOR GUSTAVO RESENDE

DESENVOLVIMENTO DE UM APLICATIVO PARA CONTROLE DE COMANDA ELETRÔNICA UTILIZANDO DESENVOLVIMENTO HIBRIDO

Ji-Paraná
2019

IGOR GUSTAVO RESENDE

DESENVOLVIMENTO DE UM APLICATIVO PARA CONTROLE DE COMANDA ELETRÔNICA UTILIZANDO DESENVOLVIMENTO HIBRIDO

Trabalho de Conclusão de Curso apresentado ao Centro Universitário São Lucas Ji-Paraná, para o título de bacharel em Sistema de Informação, sob orientação do professor Esp. Willian Fachetti.

Ji-Paraná
2019

Dados Internacionais de Catalogação na Publicação
Gerada automaticamente mediante informações fornecidas pelo(a) autor(a)

R433d Resende, Igor Gustavo.

Desenvolvimento de um aplicativo para controle de comanda eletrônica utilizando desenvolvimento híbridos / Igor Gustavo Resende -- Ji-Paraná, RO, 2019.

69 p.

Orientador(a): Prof. William Fachetti

Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Centro Universitário São Lucas

1. Aplicativo. 2. Dispositivos móveis. 3. Atendimento ao cliente. I. Fachetti, William. II. Título.

CDU 004.4:658.6

Bibliotecário(a) Alex Almeida CRB 11.8537

ATA Nº 05/2019 DE TRABALHO DE CONCLUSÃO DE CURSO EM SISTEMAS DE INFORMAÇÃO

No segundo dia do mês de dezembro de 2019, das 17h as 22h reuniram-se na sala de Inovação Tecnológica 7 o(a) professor(a) orientador(a) Willian A. de Olveira Fachetti e os(as) professores(as) Ana Flavia Moreira Camargo e Thyago Bohrer Borges para comporem Banca Examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação sob presidência do(a) primeiro(a), para analisarem a apresentação do trabalho "Desenvolvimento de um aplicativo para controle de comanda eletrônica utilizando desenvolvimento híbrido". Após as arguições e apreciação sore o trabalho exposto foi atribuída à menção como nota do Trabalho e Concluso do curso do Acadêmico(a) Igor Gustavo Resende.

OBS: Trabalho de Conclusão de Curso (X) Aprovado ou (__) Reprovado com nota total de 8,4, atribuídos o valor de 8,0 (oito pontos) para o trabalho escrito e o valor de 8,8 (oito vírgula oito pontos) para a apresentação oral.

Igor Gustavo Resende

Ana Flavia M. Camargo.

Prof. Me. Ana Flavia Moreira Camargo

Prof. Me. Thyago Bohrer Borges

Prof.Esp. Willian A. de Olveira Fachetti
Orientador

Prof. Me. Thyago Bohrer Borges
Coord. Sistemas de Informação

Dedico, todo o esforço para realização deste trabalho, primeiramente a Deus e Senhor Jesus Cristo e a minha família, pai, mãe, irmãos e aos primos. Também dedico esse trabalho a meus amigos e os professores.

AGRADECIMENTOS

Agradeço a Deus, Senhor Jesus Cristo e a minha família, o meu pai João, a minha mãe Ana, meus irmãos Laura, Evelin e Bruno, meus avôs Manoel e Gerson, minhas avós Maria Luiza e Maria Paula, que desde o início foram minha base para tudo, também agradeço aos primos e os amigos, Primos Wagner Maximiano e seu irmão Thiago Maximiano pela convivência de infância, primo Lucas Colombo, ao amigo Daniel De Sousa, meu padrinho José Pereira, minha madrinha Elisabete Colombo e minha afilhada Sulaine Cruz.

Assim como, tenho que agradecer aos meus parentes, tanto aqueles que me julgaram incapaz de alcançar meus objetivos, como também, aqueles que sempre me deram força para continuar na luta.

Bem como, agradeço meus amigos que adquiri no decorrer da faculdade em especial, Marcondes Nilson, Áleff Silva, Hamilton Ventura, Alex Pillon. Da mesma forma agradeço algumas outras pessoas que ao decorrer da minha vida acadêmica contribuiu de alguma forma.

Agradeço a todos os Professores, Willian Fachetti, Diego da Fonseca, Thyago Borges, Maigon Pontuschka, Rodolfo Olivas, Edgar Kaiser, Jones Giacon, Tiago Patrik, Mauro Oliveira e todos outros professores que fizeram parte da minha jornada.

Enfim agradeço a todos que contribuiu de alguma forma por me ajudarem nessa caminhada acadêmica, peço que Deus ilumine o caminho de cada um de vocês, assim como iluminou o meu.

Temos o destino que merecemos. O nosso destino está de acordo com os nossos méritos.

Albert Einstein

RESUMO

Este trabalho teve como objetivo o desenvolvimento de um aplicativo multiplataforma que permitisse aos usuários realizar consulta de comandas eletrônicas através de um dispositivo móvel, trazendo informação de maneira fácil e rápido e agregando valor para uma empresa. No desenvolvimento do aplicativo foi utilizado os frameworks Ionic com Angular e PhoneGap, as linguagens de marcação HTML e CSS, linguagem de programação JavaScript e TypeScript, para o servidor foi utilizada Node.js e o Firebase. A UML foi usada para elaboração da estrutura de projetos de software. O desenvolvimento em Ionic permite que a plataforma seja hibrida onde que um código, pode-se gerar um aplicativo tanto Android como iOS. Para que o usuário conseguisse verificar a comanda eletrônica com seu dispositivo em um estabelecimento, utilizando a tecnologia do QR Code, assim o usuário visualiza em tempo real com transparência e segurança.

Palavras-chave: Aplicativo multiplataforma. Comanda eletrônica. Android. iOS. QR Code.

ABSTRAT

This work aimed to develop a multiplatform application that allows users to query electronic commands through a mobile device, bringing information easily and quickly and adding value to a company. In the development of the application will be used the Ionic frameworks with Angular and PhoneGap, the markup languages HTML and CSS, programming language JavaScript and TypeScript, for the server was used Node.js and Firebase. The UML was used to elaborate the structure of software projects. The development in Ionic allows the platform to be hybridized where that code, it can generate an application both Android and iOS. In order for the user to verify the electronic command with his device in an establishment, the technology of the QR Code was used, so the user visualizes in real time with the greater transparency and security.

Keywords: Multiplatform application. Electronic control. Android. iOS. QR Code.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APP	Aplicativo
CSS	Cascading Stylesheet
HTML	Hyper Text Markup Language
API	Application Programming Interface
SDK	Software Development Kit
MVC	Model-View-Controller
GPS	Global Position System
TCP	Transmission Control Protocol
Npm	Node Package Manager
QR	Code Quick Response Code
DNS	Domain Name System
W3C	World Wide Web
NoSQL	Not Only SQL
XML	Extensible Markup Language
JSON	JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO

Com os avanços da tecnologia, é visível que gradativamente as aplicações em dispositivos móveis estão entrando no mercado, o que nos permite ter uma grande variedade, assim sendo as pessoas podem usufruir de um determinado recurso oferecido como rede de mídia, rede social, podendo ser também de locomoção e numeras aplicações, portanto pode ser determinado para versatilidade de serviços, mais contendo uma variedade de novas oportunidades.

Em estabelecimento como lanchonete, restaurante, boate e algo do gênero, a transparência com o cliente é essencial para fazer com que a empresa cresça e consiga sempre mais formas de atingir seus objetivos (ALVES, 2017). Com esse intuito, precisamos de uma aplicação que traga transparência e modernidade para auxiliar o atendimento, com uma inúmera variedade de tecnologia e sistemas diferentes, necessita de uma aplicação multiplataforma que atenda todos os clientes, com suas variedades.

O trabalho possui como objetivo principal o desenvolvimento de um aplicativo móvel que possibilite aos clientes realizarem a consulta da comanda[1] eletrônica através do seu dispositivo móvel, certificando assim uma maior integridade das informações passadas ao cliente. O aplicativo deve ser utilizado de maneira simples e fácil para que os usuários tenham uma experiencia simples e agradável.

Na construção do aplicativo móvel, foi realizado uma análise sobre as tecnologias que seriam necessárias para desenvolver o projeto, contendo as ferramentas e os modelos utilizado, baseando as principais características inserida no estudo do caso, portanto assim ser capaz de, desenvolver algo que estivesse de acordo com o estudo que foi realizado, métodos de construção de software foram utilizado em todos os processos.

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

1.1 OBJETIVOS

Nessa seção é apresentado o objetivo geral e objetivos específicos do projeto em questão.

1.1.1 **Objetivo geral**

Desenvolver de forma híbrida, um aplicativo de comanda eletrônica.

1.1.2 **Objetivos Específicos**

- Desenvolver uma aplicação de baixo custo;
- Automatizar para o cliente visualizar os seus pedidos, uma comanda eletronica multiplataforma, que retorna os quantidades, valores e descrição;
- Desenvolver um projeto utilizando a abordagem híbrida de desenvolvimento de aplicativos móveis, com tecnologia de identificação QR Code.

1.2 PROBLEMÁTICA

Através dos novos hábitos de consumidores que querem se alimentar fora do lar, o cliente busca sempre um local agradável, transparente e seguro, para fazer sua refeição ou confraternização, observa-se que estas empresas estão se adaptando às novas tecnologias existente para melhor atender ao seu cliente em potencial (SCUADRA, 2018).

Mesmo quando o assunto é venda e atendimento, a transparência é essencial, para que ele saiba qual serviço está comprando e o que pode esperar da empresa ou do atendimento como um todo, através da transparência com o cliente é determinado a fidelidade (ALVES, 2017).

A partir da necessidade da transparência para o cliente de maneira rápidas e práticas, sem que interfira relacionamento entre cliente e funcionário, mais que facilita o cliente o acesso às informações, com o uso da tecnologia. Como identificar o valor consumido em um estabelecimento, sem precisar chamar o garçom ou antes de pedir a conta?

1.3 JUSTIFICATIVA

Através dos dispositivos *mobile* a oportunidade de atingir um grande número de pessoas e, também, objetivo é deixar um estabelecimento mais transparente para o consumidor, propõe o desenvolvimento de um aplicativo para smartphones que possibilita a consulta da comanda eletrônica que pudesse permitir e a visualização de maneira fácil e rápida, tudo que foi consumido no estabelecimento.

Pelo desenvolvimento de um aplicativo com essas funcionalidades por permitir a busca por informações de maneira fácil e prática, tornando mais rápido o acesso à informação para o usuário. Logo, o resultado do trabalho tem como objetivo oferecer uma modernidade para as empresas de pequeno, médio e grande porte, trazendo mais um diferencial para as empresas do ramo.

2 **REFERENCIAL TEÓRICO**

Neste capítulo serão abordados os conceitos essenciais para o desenvolvimento do projeto, da comanda eletrônica. Para o desenvolvimento de aplicativos móveis, há a alternativa de desenvolver a aplicação para a plataforma nativa ou na forma híbrida, contendo vantagem e desvantagem de cada uma das formas.

2.1 DESENVOLVIMENTO NATIVO

A plataforma nativa é desenvolvida com o uso de ferramenta e linguagem de programação especialmente para aquela plataforma, utilizando SDK[2] e *frameworks*[3] providos por ela, os aplicativos móveis ficam vinculados a esse ambiente, executando apenas nos dispositivos da plataforma escolhida (NABUCO; FAÇANHA; ARAÚJO, 2012).

O desenvolvimento nativo contém um desempenho melhor, segue-se os padrões técnicos, de interface e de experiência de usuário definidos pela plataforma, fornecendo um *look and feel*[4] de aplicação nativa ao usuário. Aplicação nativa tem facilidade no acesso, por meios de APIs[5], providos pela plataforma, acesso ao hardware e recursos do dispositivo móvel, como câmeras, contatos, GPS, sensores e e-mail (NABUCO; FAÇANHA; ARAÚJO, 2012).

---

[2] SDK Kit de desenvolvimento de software, é um conjunto de ferramenta de desenvolvimento de software.

[3] Framework de software compreende de um conjunto de classes implementadas em uma linguagem específica, usadas para auxiliar o desenvolvimento de software.

[4] Look and Feel é um termo usado na descrição dos produtos e áreas, como a concepção de produtos, marketing, branding e trademarking, para descrever as principais características da sua aparência.

[5] APIs Application Programming Interface, é um conjunto de rotinas e padrões estabelecidos por um software.

2.1.1 **iOS**

O sistema operacional iOS foi criado em janeiro de 2007 juntamente com o seu primeiro Smartphone, iPhone, no princípio, o sistema era exclusivo, contendo uma grande performance e hardware dos dispositivos foi implantado no iPad, iPod, consequentemente, a comunicação com o hardware do dispositivo se dá por meio de um agrupamento bem estabelecido de interfaces do sistema (APPLE, 2007).

O grande diferencial do sistema operacional iOS não pode ser executado por um dispositivo que não contenha o hardware específico da empresa, sendo assim só será capaz de executar o sistema, os dispositivos que foram fabricados e desenvolvidos pela empresa (MILANI, 2014).

No desenvolvimento de aplicativos para a plataforma iOS requer um computador com o sistema operacional Mac OS e o ambiente de desenvolvimento Xcode, (HEITKOTTER; HANSCHKE; MAJCHRZAK, 2013). Sua linguagem de programação, é o Objective-C e o Swift.s (APPLE, 2014)

2.1.2 **Android**

Android foi criado pela *startup* homônima Android Inc. em outubro de 2003, com o objetivo de criar um padrão em uma plataforma de código aberto com sistema operacional é baseado em Linux. Em agosto de 2005 foi adquirida pela empresa Google (PAPAJORGJI, 2015).

No desenvolvimento de aplicativos Android é recomendado a utilização da linguagem de programação Java, podendo utilizar vários sistemas operacionais, pois não é requisitado um sistema específico e o IDE[6] indicada pelo Google é o Android Studio, oficial para desenvolvimento de aplicativos Android (ANDROID, 2016).

2.2 DESENVOLVIMENTO HÍBRIDO

O desenvolvimento híbrido tem a finalidade de funcionar em qualquer sistema ou dispositivo, utilizando o mesmo código fonte para as diferentes plataformas. Aplicações híbridas permanecem instaladas no aparelho e podem funcionar

[6] IDE é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

independentemente de se ter ou não conexão com internet. Como aplicação é codificada para todas as plataformas, frequentemente as aplicações híbridas são desenvolvidas em linguagens Web que são interpretadas pelo browser nativo do sistema. Portanto apresenta característica nativas como de *web app* [7](LOPES, 2016).

Aplicações híbrida são muitos mais rápidas para serem desenvolvidas, ela utiliza linguagens HTML5, CSS e o JavaScript, que utilizam uma funcionalidade desses sistemas chamada *WebView*[8] para demonstrar o código web como uma aplicação responsiva para qualquer plataforma. Sua padronização pela W3C[9], todas as aplicações que são desenvolvidas e testadas devem funcionar da mesma forma entre uma plataforma e nas outras (LOPES, 2016).

No desenvolvimento nativo existem o acesso aos recursos e o hardware, na aplicação híbrida também existem esse acesso através do framework é possível acessar todos os recursos, já aplicação web é bem limitado (LOPES, 2016).

Tabela 1 – Vantagem e desvantagem plataformas

	Performece	Permissão ao Dispositivo	Play store & App store	Desenvolvimento	Múltipla Plataforma
Web	Sim	Limitado	Não	Bom	Sim
Nativo	Sim	Sim	Sim	Lenta	Não
Híbrido	Sim	Sim	Sim	Bom	Sim

Fonte: O autor

2.2.1 **Ionic**

O Ionic é baseado no Node.js e distribuído via npm[10], é um framework software livre para aplicações híbridos utilizando tecnologias web como HTML, CSS e JavaScript otimizadas para dispositivos móveis (Matos *et al*., 2016 *apud* DRIFTY, 2016d). Criado em 2013 pela empresa Drifty Co. com base na necessidade de criação

[7] Web App pode ser entendido como uma versão híbrida entre aplicativos nativos para celulares e sites na internet.

[8] WebView é um componente do sistema com tecnologia do Google Chrome que permite que apps Android exibam conteúdo da Web.

[9] W3C é a principal organização de padronização da World Wide Web. É um sistema de documentos dispostos na Internet que permitem o acesso às informações apresentadas no formato de hipertexto.

[10] O npm é um gerenciador de pacotes para a linguagem de programação JavaScript.

de aplicativos móveis. Os seus clientes precisavam e uma boa ferramenta de criação de aplicativos, assim sendo projetada para ser bastante performático e trabalhar com parâmetros e tecnologias web modernas (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

O Ionic oferece uma grande vantagem em seu desenvolvimento, pois seus frameworks são sobre o Cordova e Angular. O Ionic para facilitar o uso dos desenvolvedores utiliza um conjunto de ferramentas auxiliar (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

Em um projeto Ionic se dividir em camadas, relacionadas e detalhadas abaixo:

- *Views*: Camada de apresentação como é chamada, é a onde as informações são mostradas para o usuário. Pode ser comparada como o seu homônimo no padrão MVC[11]. O Angular se referem como *templates.* É possível usar *data binding*[12] nas *Views* para poder determinar arquivos HTML separados, que são chamados e introduzidos quando necessário pela conexão com a controladora e compartilhar informações entre as duas camadas (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

- *Controls*: Essa camada fica responsável por controlar o fluxo de dados e lógica da aplicação. no padrão MVC e comparada com a camada homônima. Mostra ao usuário as *Views* e chamar as camadas de dados (Services/Factories) para ligar os dados reais da aplicação, por meio de *data binding*, à interface gráfica. As controladoras possuem uma variável chamada *scope*, que possui todas as informações que são necessárias para a criação da *View* (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

- *Data* (*Services/Factories*): É a onde o encapsula de dados da aplicação e fornece esses dados, frequentemente, por meio de uma web *service*. essa camada é que se aproxima da camada *modelo* do padrão MVC, respondendo às requisições da controladora com os dados a serem utilizados para a criação da *View* e para serem mostrados para o usuário (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

[11] MVC: Arquitetura Modelo-Visão-Controlador é um padrão de arquitetura de software que separa a representação da informação da interação do usuário com ela.
[12] Data binding é um mecanismo pelo qual dados são conectados diretamente à interface do usuário e atualizações feitas nos dados são automaticamente refletidas na interface.

- *App Configuration*: Aqui é definido as rotas, as controladoras são ativas às suas interfaces por meio de rotas. É praticável, do mesmo modo, criar rotas padrão, para o caso de não haver nenhuma rota que encontrasse sendo identificada, o que conseguiria fazer o sistema fragmentar (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

- *Directives*: Especificar comportamentos em elementos de uma página HTML, são um elemento ou um atributo que podem inicializar um comportamento específico definido pelo desenvolvedor. Nesse caso é possível criar uma diretiva para sempre colocar uma imagem padrão caso uma validação retorne um valor falso (Matos *et al*., 2016 *apud* DRIFTY, 2016d).

2.2.2 **PhoneGap**

PhoneGap foi criado em 2009 pela empresa Nitobi, é um *framework* para desenvolvimento de aplicativos híbridos, em 2011 foi adquirida pela Adobe Systems Inc., que teve seu código doado para a Apache Software Foundation para garantir que outras empresas pudessem contribuir (BEZERRA; SCHIMIGUEL, 2016).

O PhoneGap é gratuito e *open source*[13], possibilita a criação de aplicações móveis com acesso do hardware nativo do dispositivo, por meio de API JavaScript. O PhoneGap possui alguns serviços adicionais, ambiente integrado com da Adobe, como o PhoneGap Build (BEZERRA; SCHIMIGUEL, 2016).

Para Santos:

> "O PhoneGap oferece mais um serviço muito útil para testes, principalmente na fase inicial do projeto, onde queremos ver rapidamente um protótipo ou uma aplicação simples. É o **PhoneGap Developer App**. Usar o Build toda hora durante o desenvolvimento não é muito prático, já que você precisa regerar e reinstalar a aplicação a cada mudança" (SANTOS, 2016, p.28).

Entende-se através da citação feita por Santos (2016) que o PhoneGap no princípio do projeto tem várias ferramentas que auxilia nos testes, umas delas PhoneGap *Developer app*, esta ferramenta simula um servidor web hospedando a aplicação e permite executar o app através de um smartphone ou tablet para testes em dispositivos reais.

---

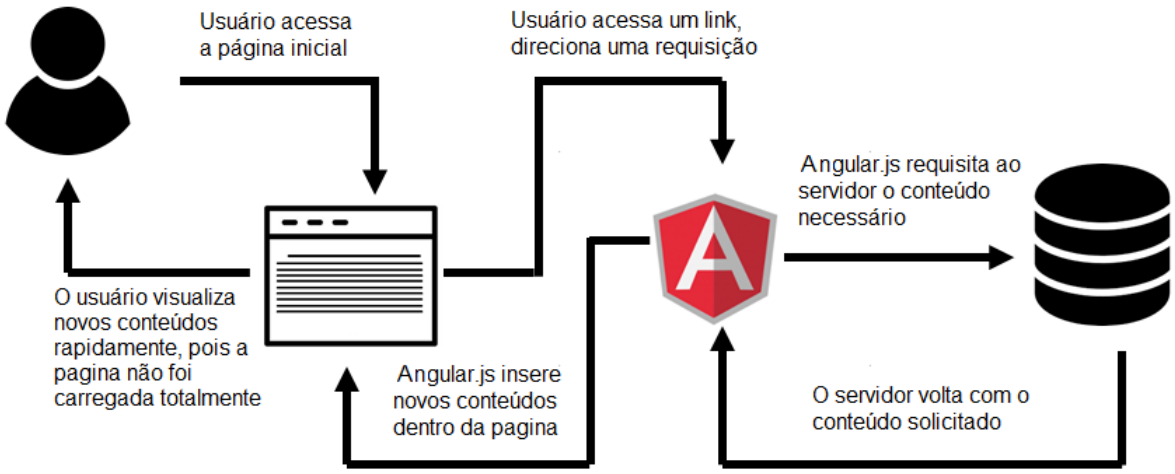
[13] Open source é um termo em inglês que significa código aberto.

2.2.3 **Angular**

Angular é um *framework* de código aberto, JavaScript desenvolvido em TypeScript e JavaScript, sendo lançado em 2016 pela Google, que possibilita estender a sintaxe do HTML para ter um conteúdo dinâmicos web. É uma ferramenta de ponta que permite interagir tanto com o *front-end*[14] quanto o *back-end*[15] (URSINO; CAPANNA, 2015).

Conforme a Figura 1 é possível observar que o servidor retorna o conteúdo específico que, o usuário fez por meio da interface gráfica, o Angular interpreta a ação e faz uma requisição pedindo o conteúdo necessário para o servidor, e o Angular retorna com o atual conteúdo inserindo-o corretamente junto do antigo já apresentado (URSINO; CAPANNA, 2015).

Figura 1 – Ciclo Angular



Fonte: O autor, baseado em (URSINO; CAPANNA, 2015).

2.2.4 **TypeScript**

O código escrito em TypeScript é facilmente compilado no JavaScript padrão. apesar do framework Angular seja um framework JavaScript, a sua estrutura base é na verdade, TypeScript.

---

[14] front-end Interface de interação com o usuário.
[15] back-end Sistema responsável pela regra de negócios, webservices e APIs de uma aplicação.

Vantagem do TypeScript está em sua natureza uma tipagem estática. Em JavaScript, qualquer erro cometido durante a criação do código pode ser descoberto apenas em tempo de execução, pois eles podem ser facilmente ignorados durante a gravação do código. O TypeScript, por outro lado, detecta erros durante a criação do código, permitindo corrigi-los imediatamente (RELEVANT SOFTWARE, 2018).

Como o TypeScript não é como o JavaScript que é mais utilizado pela maioria dos desenvolvedores, é preciso aprender e se preparar para começar a escrever código no TypeScript. Isso faz com que a curva de aprendizado do Angular seja maior; no entanto, depois de aprender o TypeScript (RELEVANT SOFTWARE, 2018).

2.3 RETAGUARDA

Nesta seção serão apresentadas as tecnologias que serão utilizadas no lado do servidor da aplicação.

2.3.1 **Node.js**

Node.js é uma plataforma para arquitetar aplicações web, com o auxílio na interpretação da linguagem JavaScript, produzido na *engine V8*[16], o Node.js também é capaz de fazer requisições assíncronas, não permitindo bloqueios, permitindo que seja mais rápido, para lidar com muitas requisições com o banco de dados (SANTOS, 2016). Não se pode confundir o Node.js com linguagem de programação, e sim um *framework*, que possui algumas vantagens, como: ter um melhorando o desempenho, utilizar da linguagem de JavaScript, ser leve e suportando ser multiplataformas (DUARTE, 2017).

O Node.js não é um servidor, mais pode montar servidores http e https, assim como servidores de DNS, TCP, Media Server, também é possível criar aplicações desktop com o *Node-WebKit* e aplicações de *front-end* (SANTOS, 2016).

[16] *Engine V8* é o nome do interpretador JavaScript, também chamado de máquina virtual JavaScript, desenvolvido pela Google e utilizado em seu navegador Google Chrome. V8 é uma ferramenta desenvolvida na linguagem C++ e distribuída no regime de código aberto.

2.3.1.1 Npm

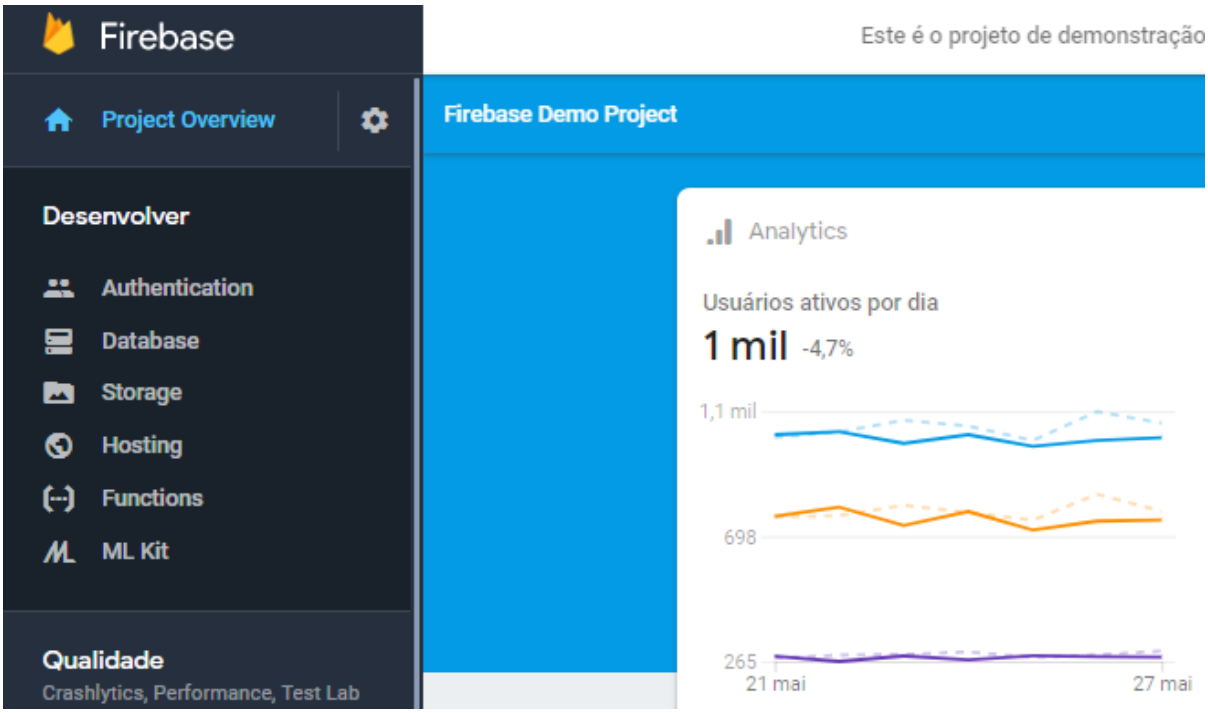
O npm foi criado em 2009, só que sua fundação só veio em 2014. Criado com o projeto de código aberto para auxiliar os desenvolvedores de JavaScript a compartilhar facilmente módulos de código compactados, aplicativos Web *front-end*, aplicativos móveis, robôs, roteadores e inúmeras outras necessidades da comunidade JavaScript (NPM, 2014). O npm permite que os desenvolvedores instalarem e publicarem esses pacotes em um gerenciador de pacotes próprio conhecido como NPM (Node Package Manager), tornando-se muito popular entre os desenvolvedores. Node.js Package Manager é o gerenciador do Node.js, que depois algumas versões ele integrou ao instalador do Node.js, com isso simplificou a utilização dos desenvolvedores, tendo uma maior convergência entre as plataformas (PEREIRA, 2018).

2.3.2 **Firebase**

Firebase é uma plataforma de desenvolvimento mobile (Android e iOS) e *web*, foi comprada pela empresa Google em 2014. O Firebase tem um foco em *back-end* completo e de simplificado na usabilidade, essa ferramenta disponibiliza diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicativos (GASPERIN, 2017). Um *console* web foi criado para simplificar a implantação, nem todos os serviços são gratuitos, tem vários planos, mais para um iniciante é uma excelente o plano gratuito (GASPERIN, 2017).

O Firebase tem diversos serviços, por exemplo: “*Realtime Database, Authentication, Cloud Messaging, Hosting, Test Lab, App Indexing e o Cloud Firestore*” e diversas outras, o Firebase e muito poderoso (GASPERIN, 2017).

Figura 2 – Console web de demonstração do Firebase



Fonte: FIREBASE.

2.3.2.1 Cloud Firestore

O Firebase banco tem uma ferramenta chamada Cloud Firestore que traz um banco de dados flexível e escalonável para desenvolvimento de dispositivos móveis e Web. É um banco de dados NoSQL[17] hospedado na nuvem que os aplicativos do Android, do iOS e Web podem acessar diretamente por meio de Software development kit (SDKs) nativos. Ele também está disponível em SDKs nativos Node.js, Java, Python e Go, bem como em REST e RPC APIs (FIREBASE, 2019).

Cloud Firestore seguindo o modelo NoSQL, o usuário armazena dados em documentos que contêm mapeamento de campos para valores, os documentos são armazenados em coleções, são contêiner de documentos que o usuário pode usar para organizar os dados e assim criar as consultas. Sendo compatíveis com muitos tipos de dados diferentes, desde números simples a objetos complexos e aninhados. É possível criar subcoleções dentro dos documentos e criar estruturas de dados hierárquicas que podem ser escalonadas (FIREBASE, 2019).

[17] NoSQL é um termo genérico que representa os bancos de dados não relacionais.

Figura 3 – Cloud Firestore

Fonte: FIREBASE.

Com isso as consultas no Cloud Firestore são expressivas, eficientes e flexíveis, para cria consulta superficiais até mais complexo, recuperando dados no nível do documento sem precisar recuperar a coleção inteira ou qualquer subcoleção aninhada. Adicionando classificações, filtragem e limites às consultas ou cursores para paginar os resultados (FIREBASE, 2019).

2.3.3 **Web service**

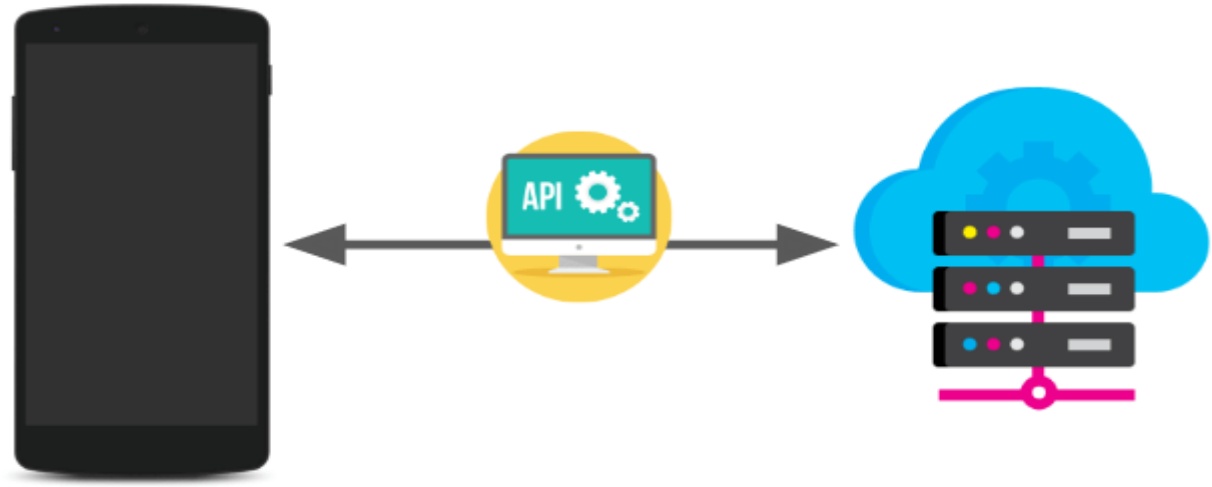
A tecnologia web service tem função de softwares que mostram uma estrutura arquitetural que permitem a comunicação entre aplicações, mesmo que suas linguagens sejam diferentes. Estes componentes que autoriza as aplicações enviem e recebam dados geralmente em formato XML[18] ou JSON[19] (AUGUSTO, 2019).

Web service pode agilizar o processo e a eficiência na comunicação entre cadeias de produção ou de logística. Tornando-se a comunicação entre os sistemas mais dinâmica e principalmente mais seguro, pois não há interferência humana (AUGUSTO, 2019).

Em uma empresa pode-se ter software para desktop desenvolvido em Javascript ou Delft, um para iOS nativo em Objective C, ou Android em Typescript utilizando IONIC, e uma aplicação Web com front-end desenvolvido em PHP, podendo-se conectar todas as aplicações em único Web Service, tem a função de “conversa” com o Banco de Dados e serve como back-end para todas as funcionalidades (AUGUSTO, 2019).

[18] XML é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.
[19] JSON é basicamente um formato leve de troca de informações/dados entre sistemas.

Figura 4 – Aplicação Através Do Web Service



Fonte: AUGUSTO. 2019

A arquitetura de serviço da web baseada em *SOAP* é que define três entidades: prestador de serviços, serviço solicitante de registro e serviço. O provedor de serviços é o serviço, a entidade endereçável da rede que aceita e executa solicitação do consumidor. O consumidor de serviço é um aplicativo, serviço ou algum outro tipo de módulo de software que requer um serviço. Um registro de serviço é um diretório que contém serviços disponíveis. O serviço consumidor encontra a descrição do serviço no registro que é publicado pelo provedor de serviços. Usando esta descrição consumidor começa a interagir com o serviço. A comunicação entre essas entidades é baseada em XML e Protocolo SOAP. Mensagens SOAP compostas por envelope, cabeçalho e corpo. O elemento envelope identifica o XML documento como uma mensagem SOAP. Um elemento de cabeçalho contém chamada e informações de resposta. Mensagens e invocações de métodos são definidos como documentos XML e enviados por um transporte protocolo SMTP, FTP, HTTP, TCP e JMS (ARMIGLIATTO. 2017).

Arquitetura de estilo REST é servidor cliente arquitetura na qual o cliente envia uma solicitação ao servidor e depois ao servidor processar a solicitação e retornar respostas. Estes pedidos e respostas construídas em torno da transferência de representações de Recursos. Um recurso é algo identificado pelo URI. A representação de recurso é normalmente um documento que captura o estado atual ou pretendido de um recurso. REST é menos fortemente tipado que SOAP. O idioma

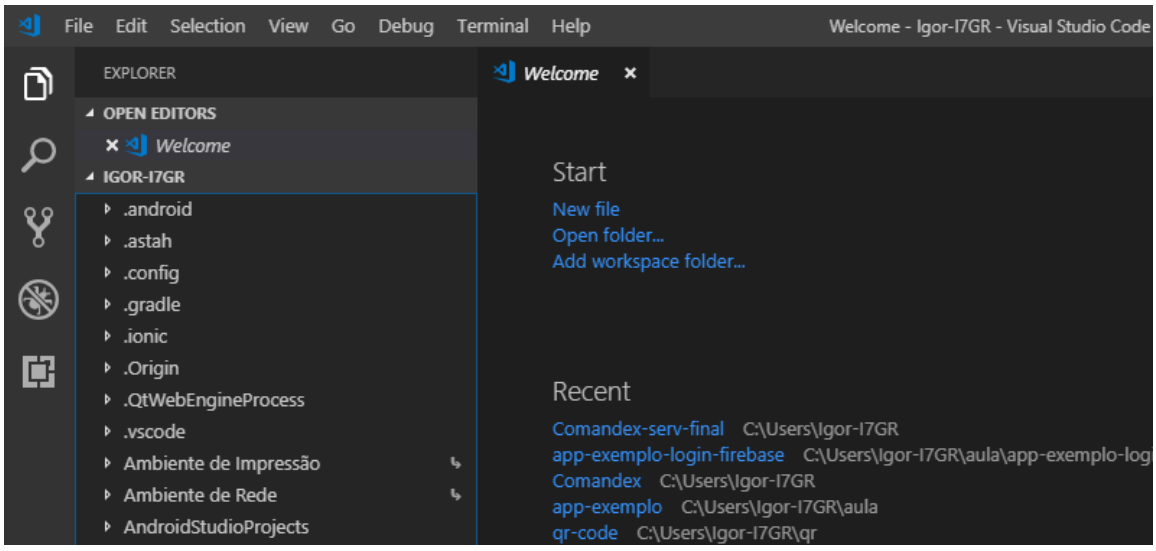
REST é baseado sobre o uso de substantivos e verbos. O REST não requer formato de mensagem, como envelope e cabeçalho, necessário em Mensagens SOAP. Portanto, como a análise XML também não é necessária o requisito de largura de banda é menor. O REST modela os conjuntos de dados para operar como recursos onde os recursos são marcados com URI. O aplicativo da web que segue a arquitetura REST que o chame como serviço da web RESTful. Uso de serviços da Web repousantes Métodos GET, PUT, POST e DELETE http para recuperar, criar, atualizar e excluir os recursos (ARMIGLIATTO. 2017).

2.4 VISUAL STUDIO CODE

O Visual Studio Code foi criado em 2015 pela Microsoft, um editor de código designado ao desenvolvimento de aplicação web, refere-se a uma ferramenta leve e multiplataforma, responde a uma gama enorme de projetos, não apenas ASP.NET, como também Node.js (DIONISIO, 2016).

O Visual Studio Code não é uma versão da ferramenta Visual Studio[20], sua essência é um editor de código leve, com uma instalação bastante simplificada, o Visual Studio Code tem um ambiente simplificado, sendo uma excelente ferramenta para o trabalho (DIONISIO, 2016).

Figura 5 – Tela principal do Visual Studio Code.



Fonte: O autor

[20] Visual Studio é um ambiente de desenvolvimento integrado da Microsoft para desenvolvimento de software especialmente dedicado ao .NET Framework e às linguagens Visual Basic, C, C++, C# e F#.

2.5 QR CODE

O termo QR Code que significa “Código de Resposta Rápida” é parecido com um código de barra tradicional, mas é um código de barra em formato quadrado que trabalha como Código Universal de Produtos (UPC), é avançado e podem ser lidos por máquinas. Seu desenvolvimento ocorreu no Japão e foi inicialmente empregado para a indústria automotiva, ainda assim, ganhou espaço em outras áreas, podendo ser utilizado para armazenar informações, como: detalhes de produtos, números de telefone, mensagens de e-mail entre outros (MOURA, 2018 *apud* TRUSTTHISPRODUCT, 2017).

De acordo com o ECKSCHMIDT, os QR Code têm uma grande capacidade de muito superior aos códigos de barra tradicional, vamos ver algumas capacidades de armazenamento:

- Kanji/Kana – máx. 1817 caracteres
- Binário (8 bits) – máx. 2953 bytes
- Alfanuméricos – máx. 4296 caracteres
- Numéricos – máx. 7089 caracteres

O QR Code é simples para serem gerados e gratuitos, podendo adicionar logos, os códigos podem ser escaneados com a maioria dos aparelhos celulares equipados com câmera. Assim o QR Code está cada vez mais no mercado, sendo utilizado em diversas áreas e trabalhos (ECKSCHMIDT, MORITA, 2014).

3 **METODOLOGIA**

Este projeto tem como objetivo o desenvolvimento de uma aplicação móvel híbrida que atenda na visualização da comanda eletrônica, assim permitindo que o processo de atendimento ao cliente e de realização de pagamento possa se tornar mais ágil.

Antes de abordar o desenvolvimento que será realizado um levantamento de várias concepções que serão utilizados como base para o desenvolvimento da aplicação. Com o levantamento dos conceitos necessários para o desenvolvimento da aplicação, um aprendizado sobre as linguagens de programação que será iniciado para observar o funcionamento das mesmas para assim poder aplicá-las no desenvolvimento do projeto. Para este projeto, foi levado em consideração que o estabelecimento não possui nenhum sistema ou comanda virtual, para mostra o valor da conta, sem necessidade de um atendente informa. Para ser possível será implantado na mesa um QR Code para a leitura através do aplicativo, o cliente poderá fazer seus pedidos normalmente através do atendente, após o pedido aparecerá no aplicativo o item que ele pediu, com sua quantidade, valor de cada item e um valor total.

3.1 UML

Em 1996, a união do entre Grady, Booch e Rumbaugh fortaleceu os três métodos e melhorou o produto final, conhecido como UML (Unified Modeling Language), que resultaram o lançamento dos documentos UML 0.9 e 0.91. Várias organizações adotaram esses métodos como Microsoft, Oracle e IBM, determinaram que a UML é essencial para o desenvolvimento de negócios. Foi estabelecido recursos que poderiam desenvolver uma linguagem de modelagem. Em 1999 foram publicaram “UML: guia do usuário”, foi uma atualização sobre UML 2.0, em sua segunda edição, de 2005 (MARTINEZ, 2019).

A UML significa Linguagem Unificada de Modelagem é uma linguagem padrão para modelagem orientada a objetos. Têm como objetivo auxiliar a visualizar o desenho e a comunicação de objetos. Proporciona que desenvolvedores visualizem

os produtos de seu trabalho em diagramas padronizados, e é muito usada para criar modelos de sistemas de software (MARTINEZ, 2019).

A UML traz tecnologia necessária para auxiliar a prática de engenharia de software orientada a objetos, a UML pode ser linguagem de modelagem padrão para modelar sistemas concorrentes e distribuídos. A UML possui diagramas que são usados em combinação, para facilitar a obter todas as visões e aspectos do sistema, os Diagramas UML é dividido em Estruturais e Comportamentais (MARTINEZ, 2019).

Os Diagramas Estruturais, são divididas em:

- Diagramas De Classe: O diagrama é o mais essencial e o mais utilizado na UML, pois ele serve de apoio as outros Diagramas (MARTINEZ, 2019).
- Diagramas De Objeto: O diagrama de objeto tem uma visão dos valores armazenados pelos objetos de um Diagrama de Classe, mostrando a execução do processo do software (MARTINEZ, 2019).
- Diagramas De Componentes: O diagrama demonstra uma coleção de componentes de software e seus inter-relacionamentos (MARTINEZ, 2019).
- Diagramas De Implantação: O diagrama demonstra na organização do conjunto de elementos de um sistema para a sua execução (MARTINEZ, 2019).
- Diagramas De Pacotes: O diagrama demonstra uma coleção de outros elementos de modelagem e diagramas (MARTINEZ, 2019).
- Diagramas De Estrutura: O diagrama é utilizado para modelar Colaborações descreve uma visão de um conjunto de entidades cooperativas interpretadas por instâncias que cooperam entre si para executar uma função específica (MARTINEZ, 2019).

Já o Diagramas Comportamentais, são divididos em:

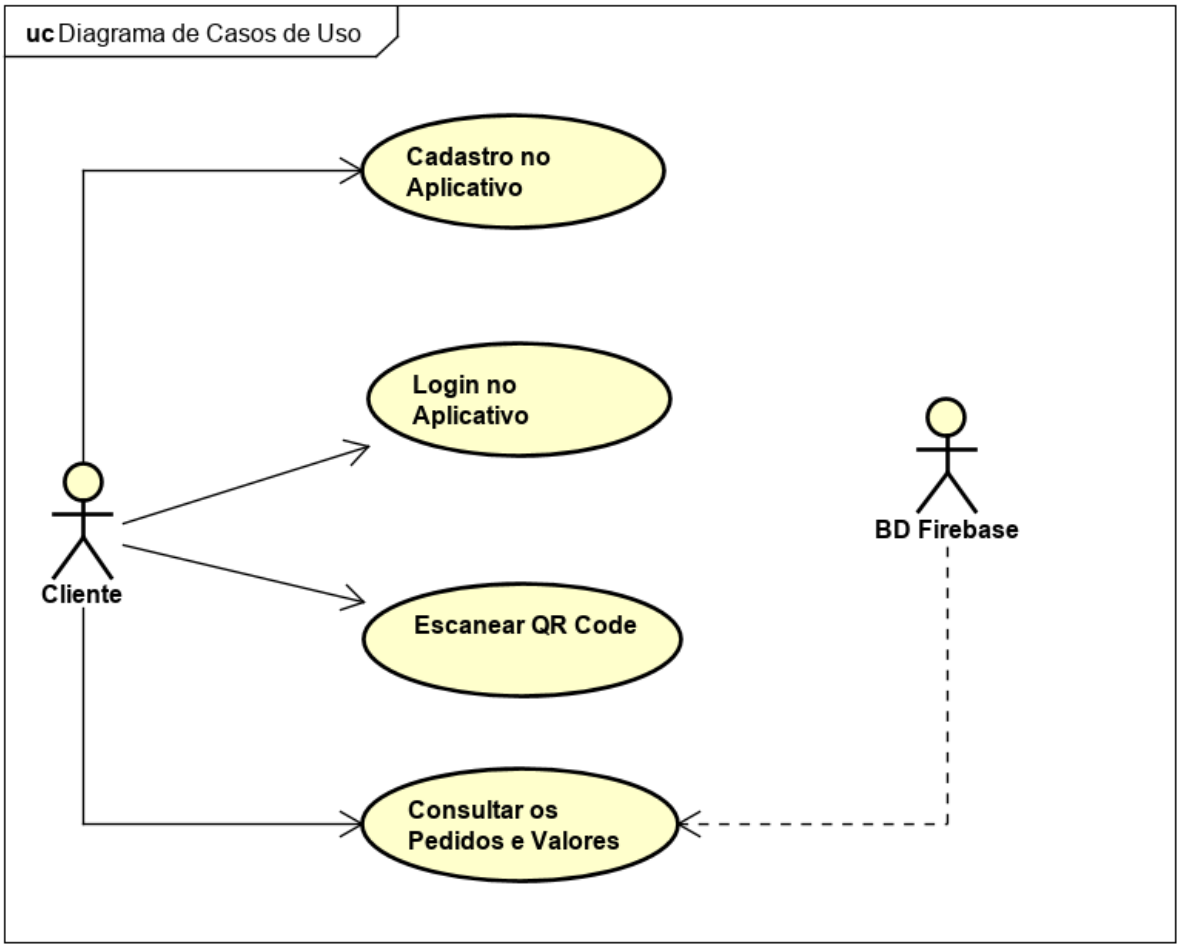
- Diagrama De Caso De Uso: O diagrama de caso de uso mostra uma coleção de casos de uso e atores (um tipo especial de classe) e seus relacionamentos. Aplique esses diagramas para ilustrar a visão estática do caso de uso de um sistema (MARTINEZ, 2019).
- Diagrama De Estados: O diagrama de estados mostra uma máquina de estados, que consiste de estados, transições, eventos e atividades (MARTINEZ, 2019).

- Diagrama De Atividades: O diagrama de atividades mostra o fluxo de uma atividade para outro em um sistema (MARTINEZ, 2019).
- Diagrama De Interação: O diagrama divide em Sequência, Geral interação, comunicação, tempo (MARTINEZ, 2019).

3.2 DIAGRAMA DE CASOS DE USO

Através do Diagrama de Casos de Uso tem como objetivo determinar o comportamento que a aplicação funcionará e apresentar as funcionalidades do que o sistema pode-se disponibilizar no ponto de vista do cliente. Nota-se na Figura 6 o diagrama funcionalidades que serão desenvolvidas.

Figura 6 – Caso de Uso



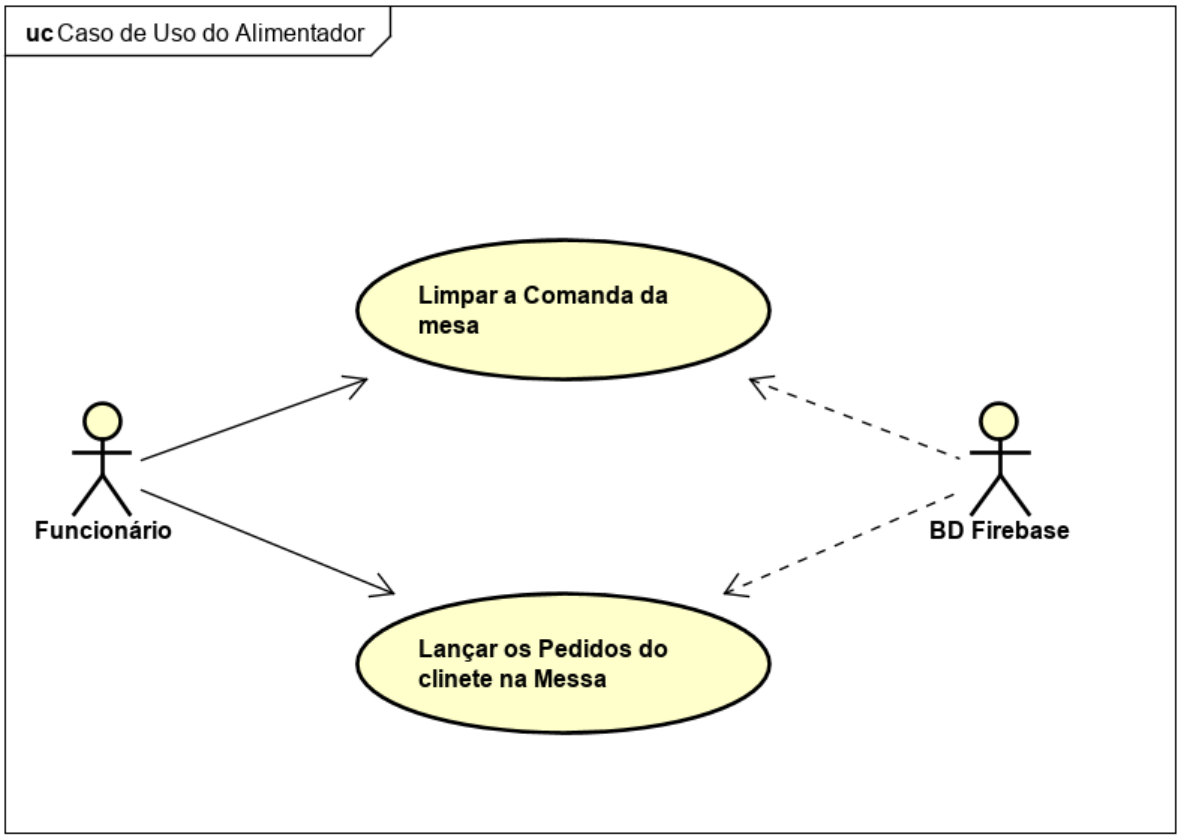
Fonte: O autor

Com o diagrama de caso de uso ver as suas atividades do cliente, podendo ter uma base do funcionamento do aplicativo, demostrando cadastro, login, leitura do QR Code e as consultas. Pode-se ver cada atividades dessas no diagrama de atividades.

O diagrama de atividades e cada uma das ações do cliente, será detalhado qual o procedimento. Assim se comunica com o banco de dados, que é alimentado pro sistema do cliente.

Podemos ver a alimentação do sistema no caso de uso abaixo. Contendo o que o funcionário irá fazer.

Figura 7 – Caso de Uso do Alimentador



Fonte: O autor

Na Figura 7, podemos notar a parte de alimentação feito pelo funcionário no sistema do estabelecimento. Representando o banco de dados Firebase, mais poderia ser qual quer banco, o sistema irá ler através da API.

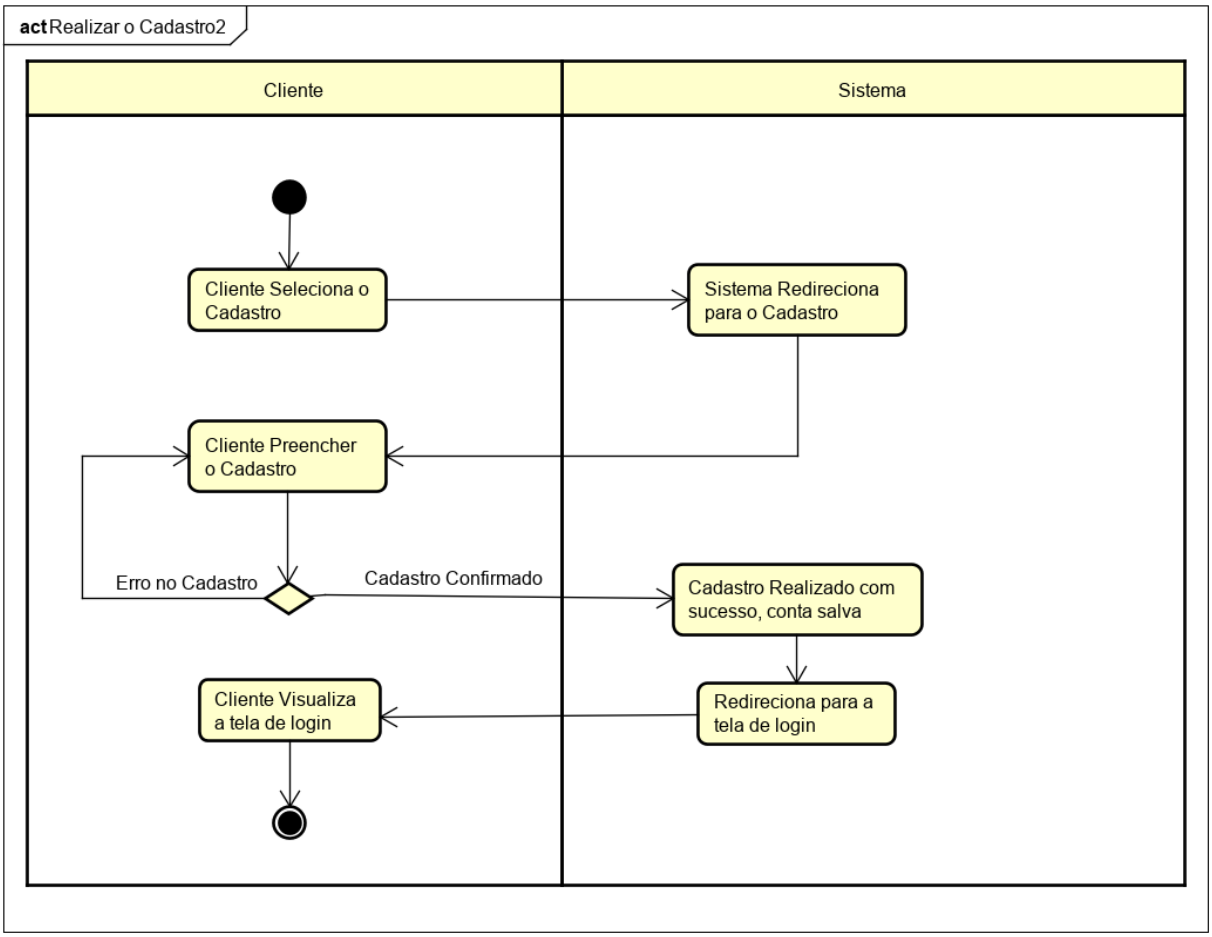
O funcionário limpa a comanda e lança os pedidos, que o banco de dados ler e armazena, o aplicativo ler e apresenta.

3.3 DIAGRAMA DE ATIVIDADES

Os diagramas de atividades são uma técnica para descrever as etapas fundamentais de um determinado processo para que ele atinja o objetivo desejado.

No diagrama a seguir demostrada na Figura 8, observasse como funcionará a realização de cadastro no aplicativo, de uma maneira simples e funcional contendo lado do cliente e o lado do sistema.

Figura 8 – Diagrama de atividade realizar cadastro



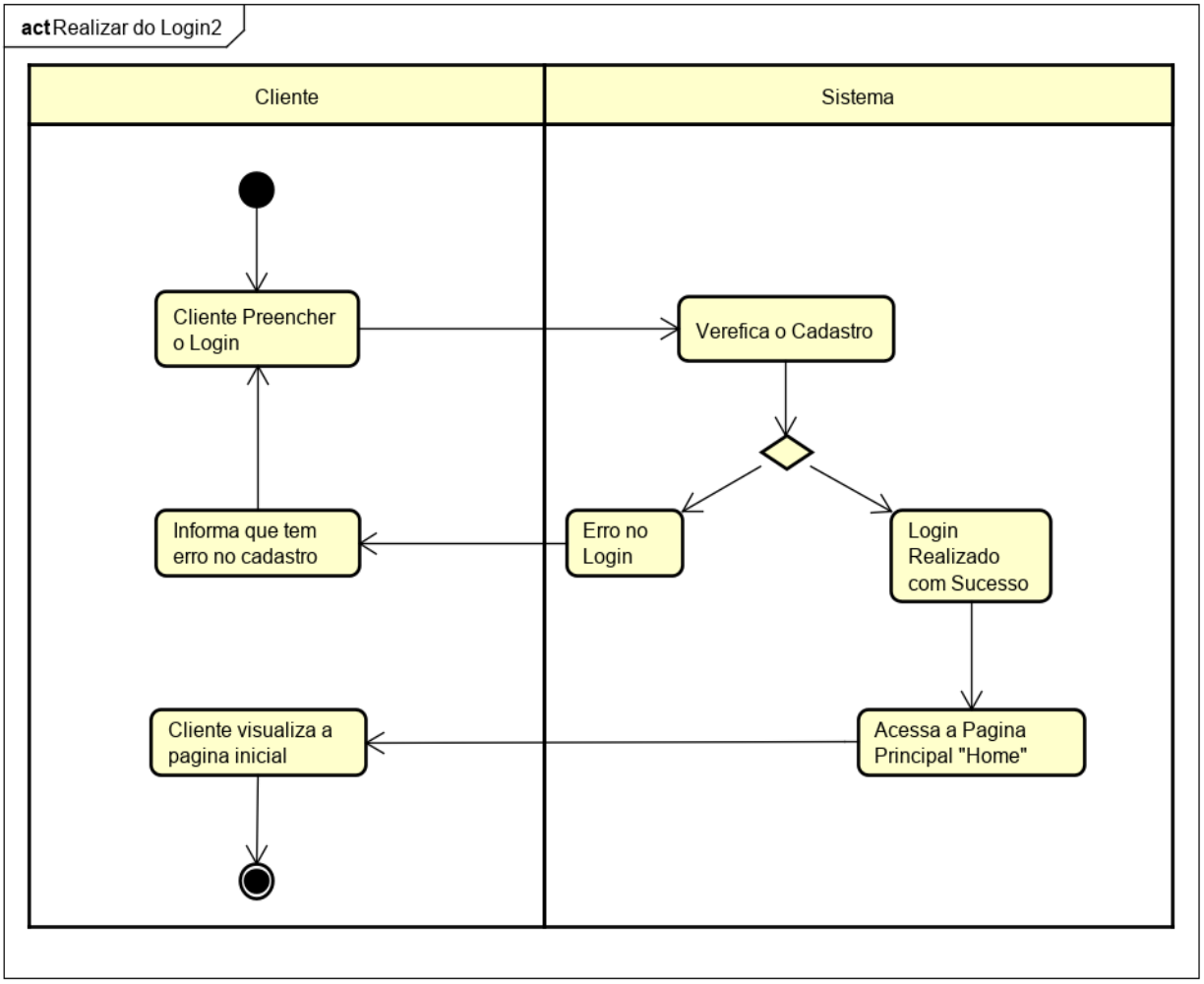
Fonte: O autor

O diagrama demonstra que o usuário seleciona o cadastro, o sistema redireciona para tela de cadastro, onde visualizar campos para preencher, se o cadastro tiver tudo certo, vai para próxima etapa se não o usuário tem que preencher novamente, logo após preencher o sistema armazena-se no banco de dados e confirma o cadastro, e redirecionar o usuário para a tela de login para utilizar o seu cadastro, assim o usuário vai ver a tela de login para acessar o aplicativo com o seus

dados. Com esse diagrama de atividade de cadastro pode-se ter uma base de como funcionará toda a parte de cadastro, do seu início ao fim detalhado.

Próximo diagrama demonstrada na Figura 9**,** nota realização do login e suas funções do usuário e do sistema.

Figura 9 – Diagrama de atividade realizar login

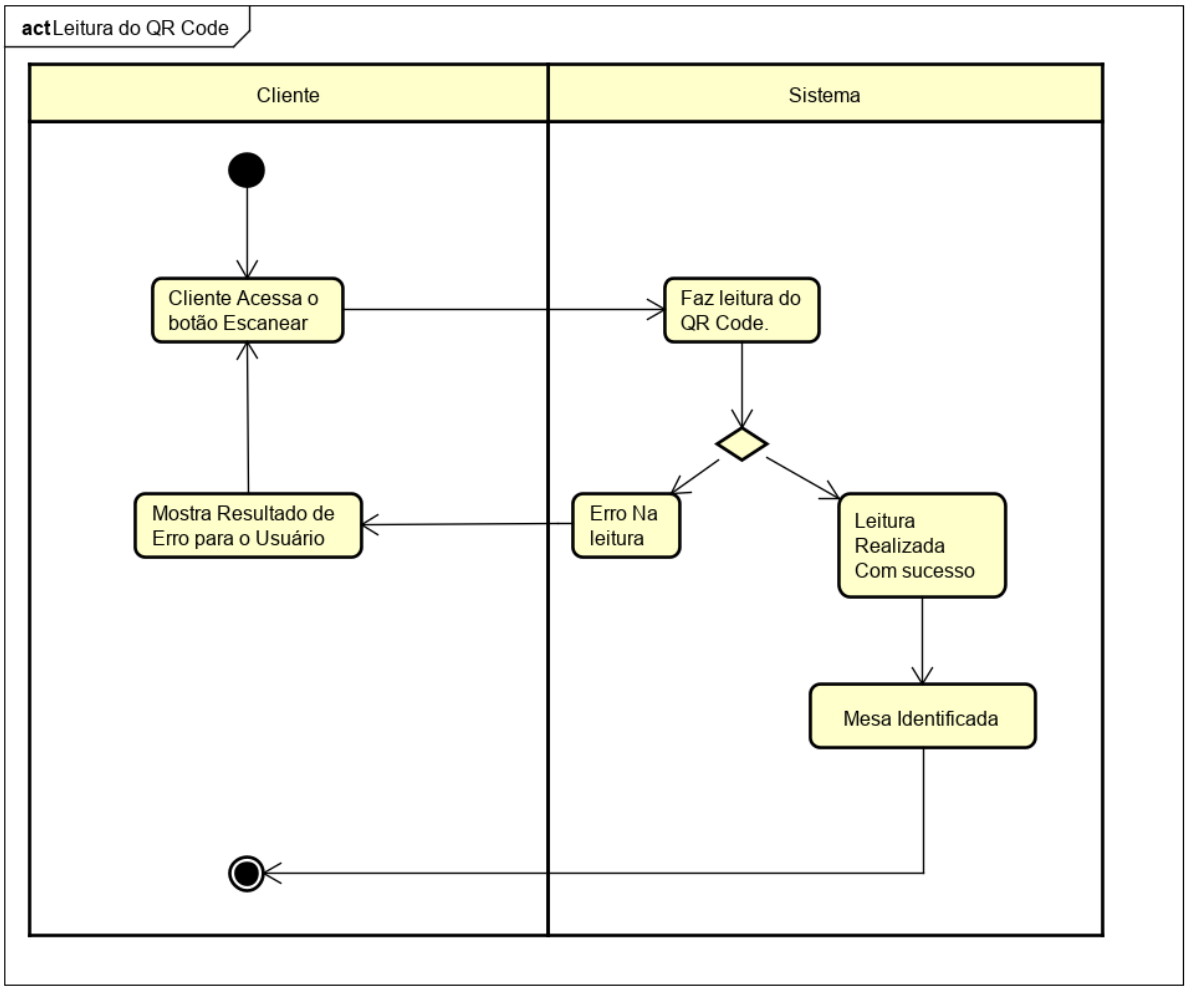


Fonte: O autor

O diagrama da Figura 9 demonstra para usuário a tela de login aonde preencher com sua conta, logo após ele ter preenchido o sistema verifica o cadastro, caso a conta esteja errado, o sistema notificará dizendo que tem um erro no cadastro, e o usuário tentar novamente, mais se o cadastro tiver correto efetuara o login com sucesso, o sistema redireciona para a página principal “*home*” do aplicativo, logo após esse processo o cliente vai visualizar a tela da página principal.

O próximo diagrama de atividade vai demostra as funções da página principal, que tem o botão scanear do QR Code, podendo-se ver na Figura 10**,** as atividades.

Figura 10 – Diagrama de atividade leitura do QR Code

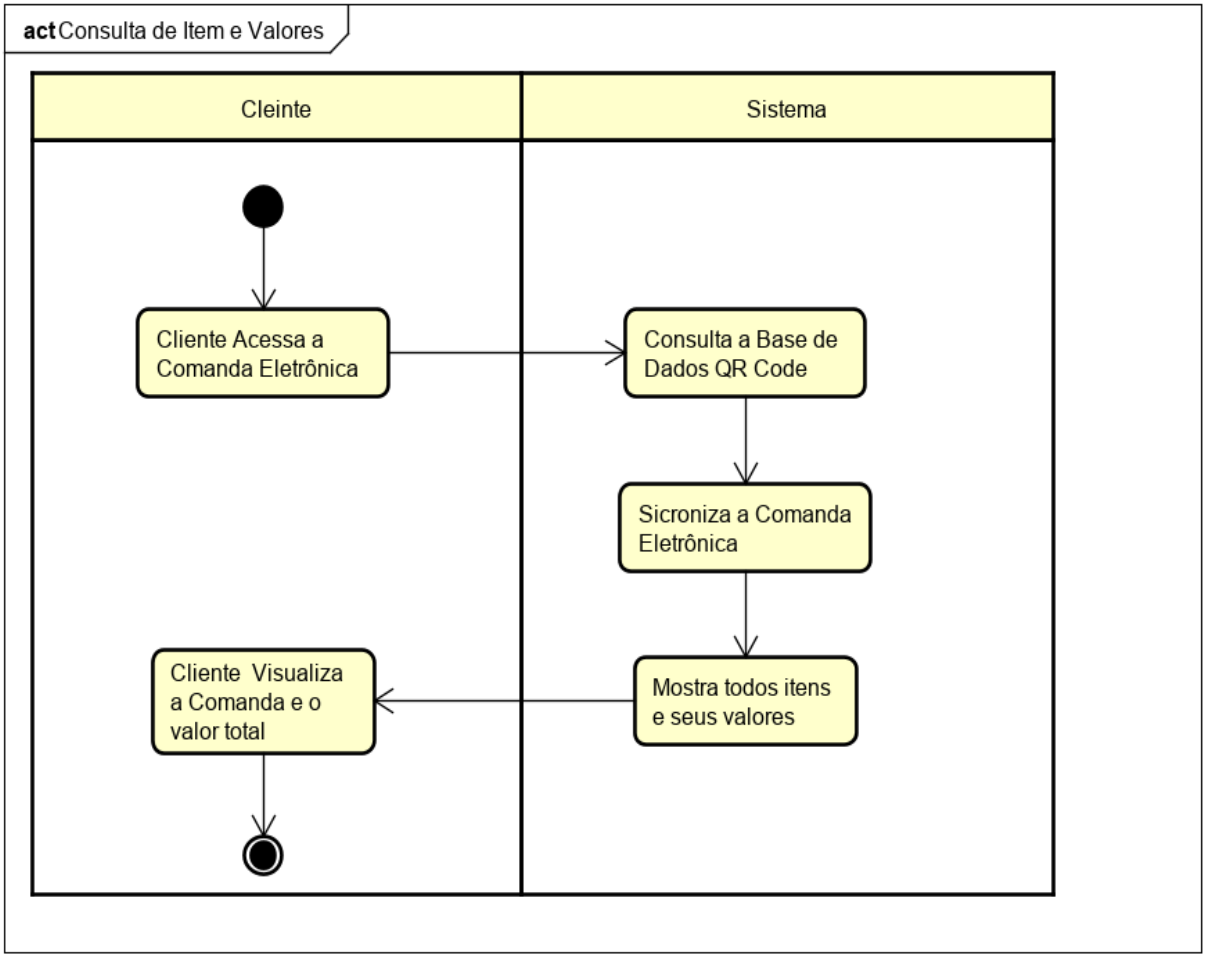


Fonte: O autor

No diagrama de leitura do QR Code, o usuário estará na tela principal, a onde ele visualiza o botão scanear, com isso ele faz a ação, logo após a ação o sistema interpreta a leitura do QR Code, se houver erro na leitura, o sistema volta com um aviso dizendo para o usuário tentar novamente pois ouve um erro, assim terá que fazer o processo novamente, se a leitura for realizada com sucesso ele verifica e identifica a mesa do usuário.

No próximo diagrama pode ser visto na Figura 11, é sobre a consulta da comanda eletrônica, onde obtém os valores e as quantidade e o valor total.

Figura 11 – Diagrama de atividade consulta de itens e valores



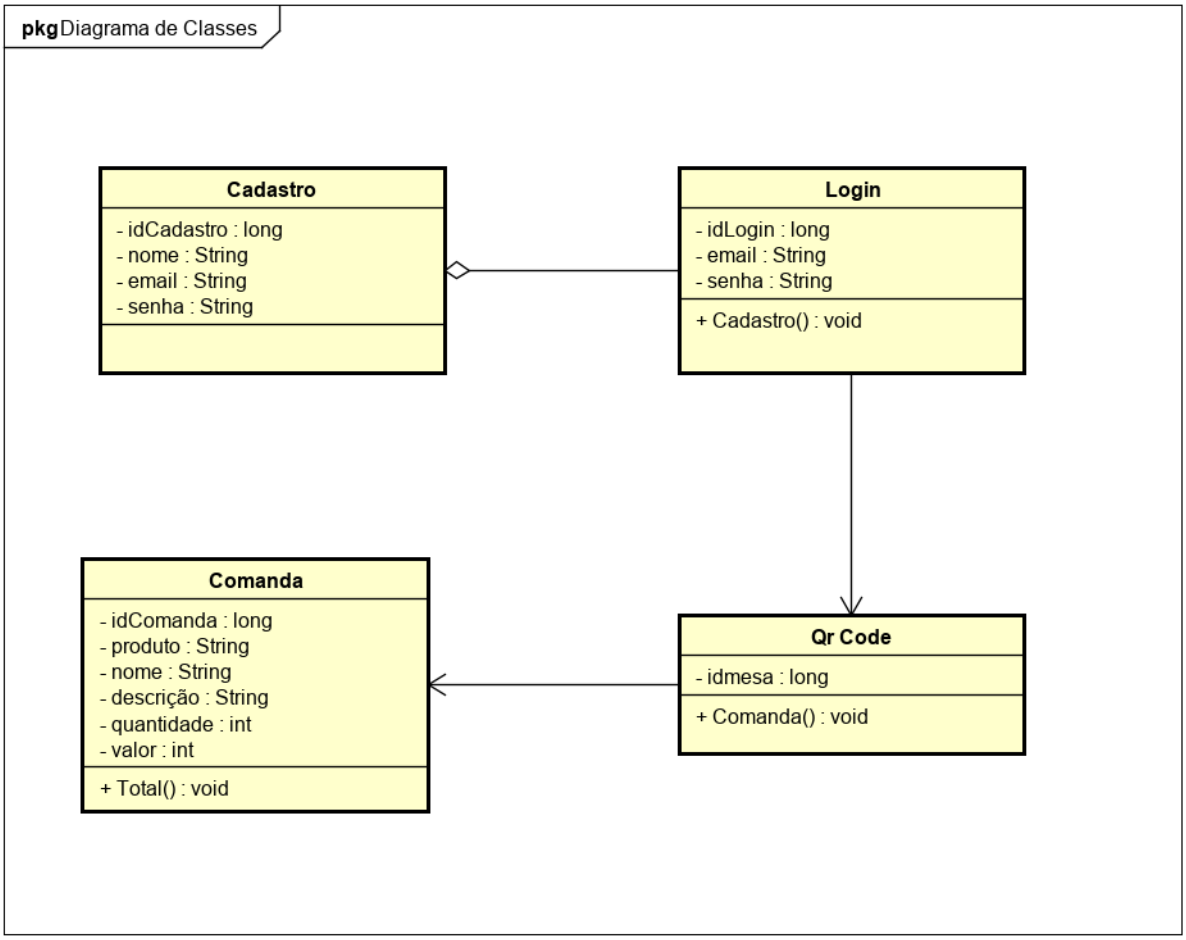
Fonte: O autor

No diagrama da Figura 11, principal função para o usuário, aqui é a onde o cliente ver valores e os itens que consumiu, após o processo anterior o usuário verificar na aba de comanda os seus pedidos, a pois o cliente acessar o sistema rapidamente consulta a base de dados da mesa que foi lida o QR Code, o sistema sincroniza os dados, traz todas as informações necessária para o usuário, e logo após o sistema envia para que o cliente possa visualizar no seu aplicativo.

3.4 DIAGRAMA DE CLASSES

A Figura 12 apresenta o diagrama de classes que será utilizado no desenvolvimento do aplicativo.

Figura 12 – Diagrama de classes



Fonte: O autor

No diagrama de classes podemos notar quatro classes cadastro, login, QR Code e comanda. Podendo ser mais fácil de entender como funciona e quais elementos vai conter.

A classe “Cadastro” vai possuir um cadastro simples com nome e senha e e-mail, será utilizada para armazenar os dados do cliente, para poder acessar a aplicação precisa ser feito o cadastro.

Na classe “Login” é aonde é preenchido o e-mail e senha, assim podendo acessar tela principal aonde vai ter a função scan do QR Code, para funcionar precisa ser feita o cadastro.

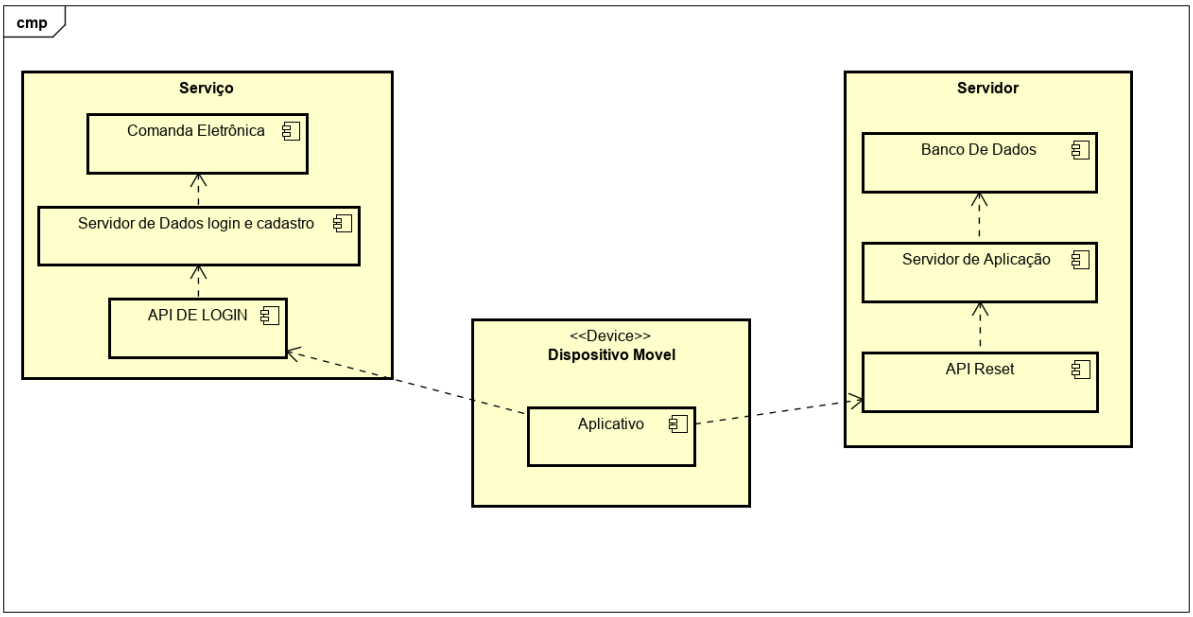
A classe “QR Code” é a classe que reconhece a mesa, para exibir a comanda corretamente para a determinada mesa.

A classe “Comanda” é aonde lista todos os produtos pedidos, assim contendo nome, quantidade, valor e descrição, com isso informa para o cliente tudo sobre o produto e o seu total.

3.5 DIAGRAMA DE IMPLANTAÇÃO

UML os diagramas de implantação possuem como finalidade realizar a modelagem da arquitetura física do sistema, objetivo de demonstra o que ocorre com o relacionamento entre os componentes de software e hardware no sistema e a distribuição física do processamento.

Figura 13 – Diagrama de implantação



Fonte: O autor

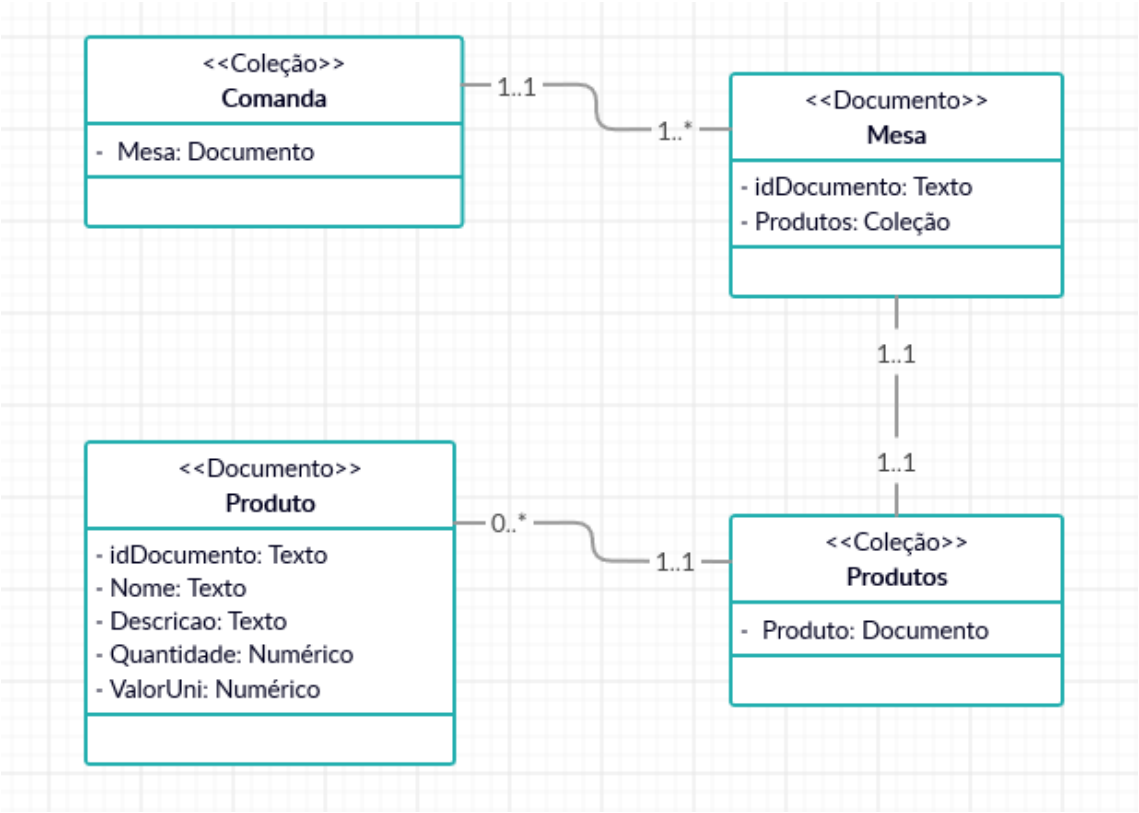
Na Figura 13 o diagrama de implementação nota-se que o aplicativo será instalado no dispositivo móvel, podendo ser do usuário ou do funcionário, A aplicação realizará conexões com a API que será responsável por realizar a comunicação entre o servidor da aplicação e o banco de dados do sistema.

Além de conectar-se com o serviço onde estarão armazenadas as informações da aplicação, o aplicativo também se conectará com uma API.

3.6 BANCO DE DADOS

O Firebase Cloud Firestore trata-se de um banco de dados NoSQL, ou seja, não relacional, o qual trabalha basicamente com Coleções que possuem documentos ou coleções em seu interior. Assim, as coleções e documentos necessários ao sistema encontram-se relacionados na Figura 14.

Figura 14 – Diagrama de Coleções e Documentos do Banco de Dados



Fonte: Autor – Sistema Creatily

3.7 METODOLOGIA DE DESENVOLVIMENTO

A metodologia ágil em desenvolvimento de *software*, e o meto mais utilizado pôs é dinâmico e flexível e com uma ótima produtividade, utiliza uma abordagem de planejamento incremental e muito iterativa. Dessa forma, se define e se documenta detalhadamente todas as fases do início ao fim do projeto, no método ágil isso é feito em pequenas partes, também chamadas iterações (PRESSMAN, 2011).

A metodologia ágil XP (Extreme Programming), definido ao desenvolvimento de softwares, realizado a partir de três pilares: agilidade no desenvolvimento da solução, economia de recursos e qualidade do produto final (PRESSMAN, 2011).

O FDD (Feature Driven Development), reúne as melhores práticas dos outros métodos, focando a funcionalidade, com planejamento de entregas também incremental, ou seja, por fases (PRESSMAN, 2011).

Metodologia MSF (Microsoft Solutions Framework), é mais utilizado para o desenvolvimento de soluções tecnológicas por equipes reduzidas, focando na redução dos riscos e aumento da qualidade final.

Para iniciar a construção de um *software*, é viável a utilização de um roteiro que possibilita a controlar os tópicos adequados ao desenvolvimento, ou seja, como qualidade, controle, estabilidade, organização e cumprimento do prazo estabelecido para a entrega (PRESSMAN, 2011). Em relação a planejamento deste projeto foi escolhido e utilizado o *framework* de desenvolvimento ágil Scrum.

3.7.1 **Scrum**

Scrum “é embasado no empirismo e utiliza uma abordagem iterativa e incremental para entregar valor com frequência e, assim, reduzir os riscos do projeto.” (SABBAGH, 2013, p. 17), assim o Scrum se tornou comum para trabalhar com *software,* tendo uma base para entregar as metas de uma maneira colaborativa agradável. (SABBAGH, 2013). Incluso no Scrum cada membro da equipe de desenvolvimento do projeto possui um papel distinto, contendo as responsabilizar e se comprometer com a execução e o resultado do mesmo (SABBAGH, 2013).

Para Pressman:

> “é um método de desenvolvimento ágil de software concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990. Mais recentemente, foram realizados desenvolvimentos adicionais nos métodos gráficos Scrum por Schwaber e Beedle [Sch01a].” (PRESSMAN, 2011, p.95).

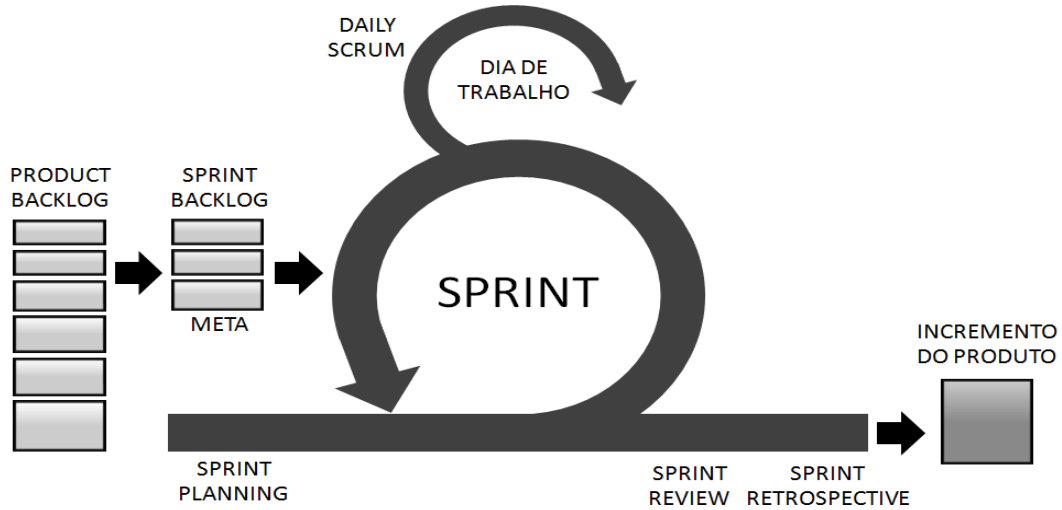
Através de Pressman nota-se dizer que o modelo ágil Scrum já tem mais de duas décadas, e nesse tempo foi realizado desenvolvimentos complementares nos métodos gráficos.

O Scrum pode ser utilizado para diversas áreas de trabalho, mais uma boa vantagem do Scrum que ele atende bem a grupos pequenos, ou seja, que possuem entre 3 a 9 membros (SABBAGH, 2013). Segundo Sabbagh (2013, p. 36), os *Sprint* “acontecem um depois do outro, sem intervalos entre eles, ou seja, tudo em um projeto que utiliza Scrum acontece dentro dos *Sprints*”.

Para realizar o *Sprin*t, o *Product Owner* insere em uma lista designada *Product Backlog* todas as funcionalidades que precisam ser desenvolvidas pela equipe de desenvolvimento para atingir o objetivo final (SABBAGH, 2013). posteriormente a isso, e feito uma reunião denominada *Sprint Planning* o Time de Scrum escolhe alguns itens do *Product Backlog* e estabelece quais tarefas devem ser realizadas para atingir uma meta, se inicia um Sprint que ficará apenas em executar as tarefas definidas no *Sprint Planning* (SABBAGH, 2013). Cada *Sprint* tem uma duração fixa de tempo, será feita uma reunião diária denominada de Daily Scrum, com isso a equipe de Desenvolvimento expõe todos seus trabalhos já realizados e todos os planejamentos do próximo dia (SABBAGH, 2013).

No término do *Sprint* são feitas reuniões, é apresentado o resultado do Sprint para as partes interessadas, conforme isso será realizado adaptações, se necessário planejar melhorias a serem desenvolvida nos próximos etapa do *Sprint* (SABBAGH, 2013). Depois de várias reuniões, um incremento de um produto é gerado, demonstrado o visual para os seus clientes do projeto, (SABBAGH, 2013). A Figura 15 representa o ciclo completo do Scrum.

Figura 15 – Ciclo do Scrum



Fonte: SABBAGH (2013, p. 42)

De acordo com Sabbagh (2013, p. 3) “Scrum é apenas uma ferramenta que pode trazer diversos benefícios em comparação a outras formas de se conduzir projetos, mas somente se bem utilizada.”, mais o Scrum é uma excelente escolha para alguns produtos.

4 **RESULTADOS E DISCUSSÃO**

Esta seção descreve o processo de desenvolvimento que será realizado no decorrer neste capítulo, onde será possível discutir sobre alguns dos principais passos realizados nos *Sprints* para a produção do aplicativo, também apresentar as dificuldades, e alguns resultados obtidos com a utilização das tecnologias.

4.1 PREPARAÇÃO

Para iniciar o trabalho, faz-se necessário definir o projeto em suas metas para cada *Sprint* a serem executado (*Product Backlog*). Logo após de possuir todas as metas definidas e anteriormente do início do *Sprint*, portanto foram relacionadas as tarefas fundamentais para atingir a sua meta (*Sprint Backlog*), e assim depois desse processo que foi iniciada a execução do *Sprint*. A tabela 2 demonstra o *Product Backlog* do projeto, no qual abrange a meta de cada item, tendo a tempo estimada de dias de desenvolvimento e em qual *Sprint* o item deveria ser desenvolvido.

Tabela 2 – Product Backlog do projeto.

Product Backlog				
Nº	Sprint	Tempo (em Semanas)	Item	Meta
1	1	1	Levantamento Do Negócio	Construção da ideia, planejamento do negócio e o que terá na aplicação.
2	1	1	Construção Das Telas	Construção de todas as telas possível.
3	1	1	Levantamento Do Banco De Dados	Levantamento do banco de dados, que será utilizado no aplicativo.
4	2	2	Construções Das telas De Login e Cadastro	Construções das telas de login e cadastro, com os campos necessários.
5	2	1	Configuração Do Firebase	Configuração do Firebase para ter acesso ao cadastro.
6	2	1	Acesso ao Aplicativo	ter acesso ao aplicativo através do cadastro.

7	3	2	Construção Das Telas Do Aplicativo	Construção de todas as telas possível do aplicativo
8	3	3	Implementação Do Botão Scan	Implementação do botão Scan, com a função do QR Code.
9	3	1	Personalização Do Botão Scan	Personalização do botão Scan, para tornar mais adequado ao aplicativo.
10	4	3	Configuração Do Banco De Dados	Configuração do banco de dados, para acessar no aplicativo.
11	4	7	Configuração Da Comanda	Configuração para exibir a comanda.
12	4	2	Personalização Da Comanda	Personalização da comanda.
13	4	4	Teste e Reparos	Testa o aplicativo, fazer reparos e personalização final.

Fonte: Do Autor.

Para iniciar o desenvolvimento do aplicativo precisamos configurar as ferramentas utilizada para a aplicação, para o editor de código-fonte foi utilizado o Visual Studio Code, na versão 1.32.3, 64bits. O sistema operacional utilizado durante o desenvolvimento e testes foi o Windows 10 Enterprise LTSC, foi inserido também ferramentas com node na sua versão 11.10.0, a ferramenta PhoneGap na sua versão 8.2.2, a ferramenta o npm na versão 6.7.0. Foi utilizado o Google Chrome, para simular o desempenho da interface dos aplicativos em dispositivos com inúmeros tamanhos e capacidades de tela e configurações, a qual beneficiou e tornou mais rápida a visualização e o desenvolvimento dos aplicativos. O teste dos aplicativos foi realizado no dispositivo da marca Asus, modelo ASUS_Z00AD, que possui o sistema operacional Android (versão 5.0) e um dispositivo da marca Xiaomi, modelo Redmi Note 6 Pro, que possui o sistema operacional MIUI Global (versão 10.3), baseado no Android, nessa versão do Android (versão 9 PKQ1.180904.001).

Após todas as instalações configurações, são apresentadas as tarefas realizadas em cada *Sprint*, todas dificuldades encontradas no decorrer do processo de desenvolvimento e os resultados obtidos logo após a realização de cada Sprint.

4.2 PRIMEIRO *SPRINT*

No primeiro Sprint iniciou-se o levantamento de negócio, construção das telas, levantamento do banco de dado, para determinar no que será feito, as telas que será desenvolvida, e os elemento do banco de dados que será utilizado, para agilizar no processo de desenvolvimento para o aplicativo.

4.2.1 **Sprint Backlog**

Para atingir todas as metas que foi enumerada no *Sprint* número 1, foram descritas e realizadas as seguintes tarefas, exibida na Tabela 3:

Tabela 3 - 1º Sprint Backlog.

1º *Sprint Backlog*	
Item do Product *Backlog*	**Tarefas**
Levantamento Do Negócio	Compreensão da Aplicação
	Modelo de Negócio
Construção Das Telas	Definições de todas as telas
	Esboços de todas as telas
Levantamento Do Banco De Dados	Definições do banco de dados
	Esboços do banco de dados

Fonte: Do Autor.

4.2.1.1 Canvas

Para fazer o levantamento de negócio, para a aplicação do Negócio foi feita o modelo Canvas, que levanta alguns elementos para o projeto.

O modelo Canvas é uma metodologia elaborada por Alex Osterwalder, separado em 9 elementos essenciais para todas as organizações, na configuração de um quadro, possibilita que o negócio seja analisado de forma completa, deixando substanciado as possíveis modificações para várias ideias ou situações (MOTA, 2018).

Com essa metodologia, temos potencial denominar a aplicação do modelo em negócio, De acordo com (MOTA, 2018), esses elementos do modelo Canvas

envolvem as 4 áreas de um negócio, como: consumidores, oferta, infraestrutura e viabilidade financeira, podendo qualquer empresa pode criar modelos de negócio de suas empresas ou até recriar de outras.

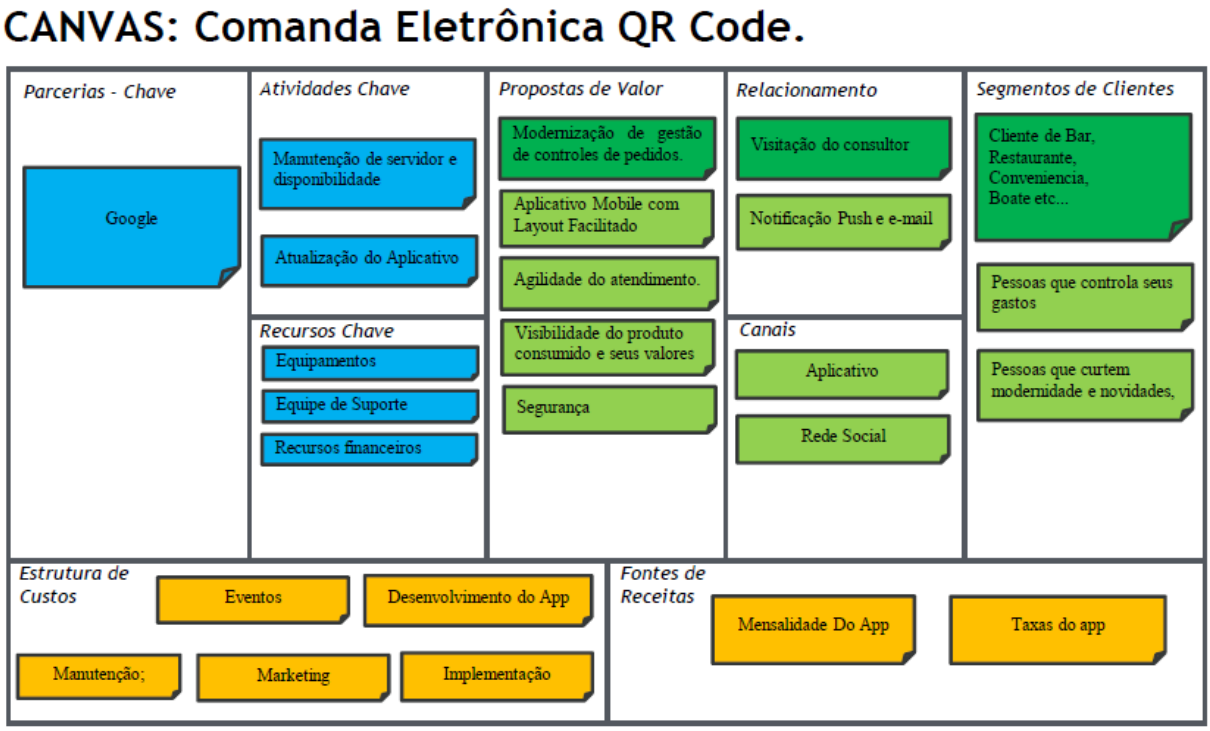
Separado em 9 elementos o modelo Canvas são descritos a seguir:

- Segmentação do Cliente: é considerado o mais importante é o primeiro, a ser feito, pois nesse bloco o segmento do mercado afetará os clientes, deve ser definido os principais clientes para o projeto (PIMENTA, 2019).
- Proposta de Valor: Esse componente é correspondente ao porquê os clientes escolherem sua empresa. A proposta de valor deve solucionar ou satisfazer alguma necessidade do cliente. É neste momento em que a empresa pode refletir e descobrir seus diferenciais diante dos concorrentes (PIMENTA, 2019).
- Canais: Onde terá interação com o cliente, como que ele saberá do seu produto. É determinado por onde a empresa vai trabalhar o marketing do produto na presença dos seus clientes (PIMENTA, 2019).
- Relacionamento: nessa etapa é feito interação com o cliente, através de e-mail, ligações e ir no local demonstrando um relacionamento. Descreve sua estratégia para evitar que seus clientes substituem pela sua concorrência (PIMENTA, 2019).
- Parcerias Chave: São os fornecedores e parceiros terceirizadas, é qualquer tipo de funcionamento ou matéria-prima fornecida por diferente empresa, é importante para você entregar sua proposta de valor. (PIMENTA, 2019).
- Recursos Chave: O que é necessário para fazer o seu negócio funcionar. Os recursos fundamentais para que sua empresa obtenha o sucesso na produção da aplicação ou produto final (PIMENTA, 2019).
- Atividades Chave: Parecido com recursos chave, as atividades essenciais para você realizar o seu negócio, devem ser realizadas de forma estável para que a aplicação funcione de maneira adequada (PIMENTA, 2019).
- Estrutura de Custo: É determinado os todos os principais custos, para a aplicação e a operação do negócio (PIMENTA, 2019).
- Fontes de Renda: É aonde determinamos o meio de renda da aplicação. Alguns exemplos são: mensalidade, venda, assinatura, licença, entre outros (PIMENTA, 2019).

Através dessas premissas foi construído o modelo canvas, para a aplicação do negócio, respeitando todas as medidas de cada elemento. Podendo ser levantada quem são os clientes, proposta de valor do aplicativo, relacionamento com o cliente, meios de canais, parcerias chave, atividades chave, recursos chave, os gastos e as receitas.

.

Figura 16 – Modelo Canvas da Comanda Eletrônica.

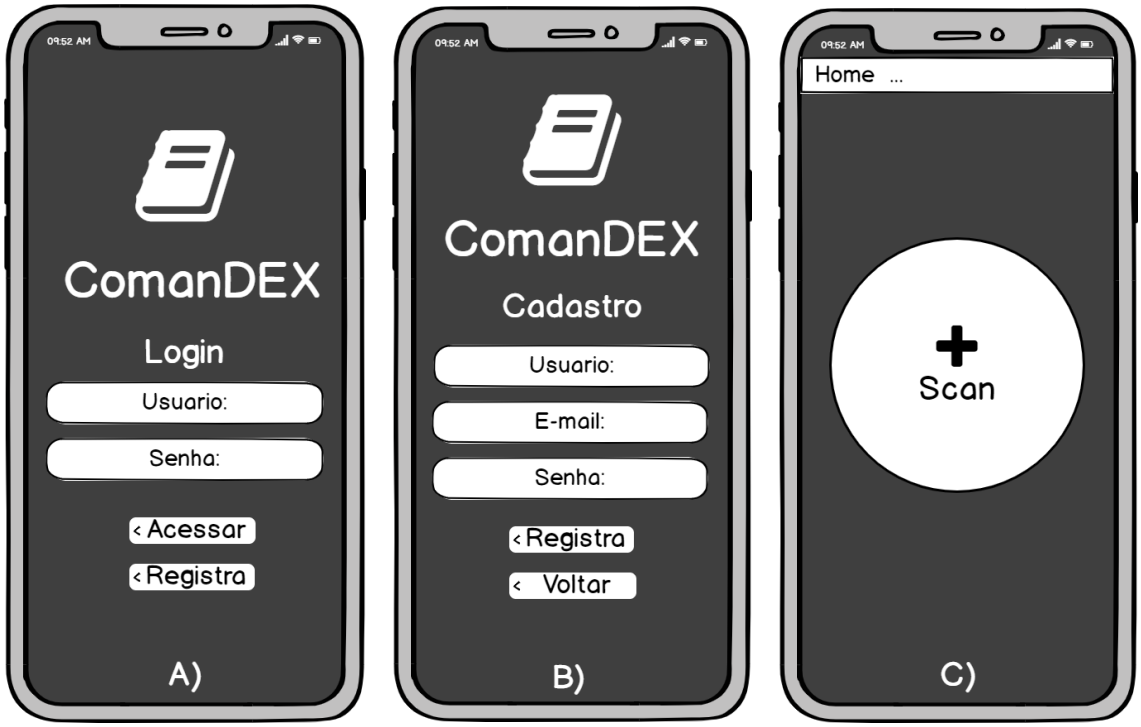


Fonte: Do Autor

4.2.1.2 Esboço Das Telas

Para construção foi necessário ter a ideia definida, através do modelo Canvas, deixou bem acentuado o formato do negócio que seria aplicado, portanto facilitou a criação dos esboços das telas, que facilitará no desenvolvimento do aplicativo. Para criação do esboço foi utilizado Balsamiq uma plataforma web para criar esboços, desenhos e algumas outras ferramentas. Tivemos o seguinte resultado, cinco telas como, tela de login, cadastro, home, lateral, comanda. Podemos ver abaixo, as seguintes telas, login, cadastro, home:

Figura 17 – Esboços, Tela de Login (a), Tela de Cadastro (b) Tela Home (c).



Fonte: Do Autor.

O resultado inicial do esboço das telas, pode ser visualizado na Figura 17, a qual apresenta a Tela de login do aplicativo dos clientes (a), a onde eles irão acessar com uma conta, caso não tenha, tem o botão de cadastro, para acessar a tela de cadastro, que podemos visualizar na (b), podendo criar sua conta facilmente, se caso já tenha uma conta, tem o botão já tenho, assim o cliente pode volta para a tela de login, quando o cliente acessa o aplicativo através da sua conta, o cliente acessa a tela principal (home), podemos ver no (c), que demonstra um botão scan, onde o cliente acessar o QR Code. Logo abaixo podem ver duas telas, lateral e comanda:

Figura 18 – Esboços, Tela Lateral (d), Tela de Comanda (e).

Fonte: Do Autor.

Pode ser visualizado na Figura 18, no qual podemos notar a tela lateral (d), que consta o acesso home, que tem a função do scan, e acesso a comanda, que podemos ver ao lado (e), podemos notar que, tem uma comanda, com item, valor, quantidade e logo abaixo valor total.

Portanto podemos ver na Figura 18 e na Figura 17, as funções completas do que será a aplicação, e de como será desenvolvido.

4.2.1.3 Levantamento Do Banco De Dados

Após esboço das telas, podemos fazer um levantamento profundo do banco de dados, o que será utilizado, para que essa aplicação funcione de maneira correta sem faltar algo, com vários análise, foi possível chegar em um esboço completo de como será o banco de dados.

O banco de dados terá uma sequência definida com coleção e documentos no Firebase, que será alimentado pelo o sistema do cliente.

Depois dessas etapas concluídas, do modelo canvas, as telas da aplicação e do banco de dados, só assim é possível iniciar a segunda *Sprint,* são determinantes cada processo feito.

4.3 SEGUNDO *SPRINT*

No segundo *Sprint* iniciou-se o desenvolvimento da tela de login e cadastro do aplicativo, configuração no Firebase para possibilitar o cadastro, por final com as telas feitas e Firebase configurado iremos ter acesso a tela principal do aplicativo.

4.3.1 **Sprint Backlog**

Na Tabela 4 demonstra as tarefas realizadas para atingir a meta do segundo *Sprint*:

Tabela 4 – 2º Sprint Backlog.

2º Sprint Backlog	
Item do Product *Backlog*	**Tarefas**
Construção Das telas, Login e Cadastro.	Gerar estrutura inicial do aplicativo (pastas, bibliotecas)
	Adicionar elementos de design da página inicial
	Construir página inicial com as opções de cadastro e login
Configuração Do Firebase	Configuração Do Firebase (criação de projeto)
Acesso ao Aplicativo	fazer ligação do Firebase com o cadastro e acessar o app

Fonte: Do Autor.

4.3.1.1 Construção Das Telas

Neste *Sprint* foram realizadas as configuração inicial do aplicativo, assim teve o início ao aplicativo da comanda eletrônica, depois de fazer a estrutura básica do aplicativo, começou a ser adicionado um design para a página inicial, e logo em seguida colocamos o código título, que consta na parte superior, exemplo na Figura 18, que mostra a parcial do código.

Figura 19 **–** Código da tela com título.

```
<ion-header>
  <ion-toolbar color="dark">
    <ion-title>Comanda Eletronica</ion-title>
  </ion-toolbar>
</ion-header>
```

Fonte: Autor

Depois da alteração do título foi incrementado posicionamento, cor e várias substância, que foi ganhando forma a tela inicial, vale ressaltar o campo que faz o login, inserimos os campos de e-mail e senha, demonstrado na Figura 20, podemos ver o código, e a estrutura utilizada.

Figura 20 – Código dos campo, e-mail e senha.

```
  </ion-item>
    <h1>Comanda Eletronica</h1>
    <h6>Acessso Ao APK</h6>
<ion-grid>
  <ion-row class="inputStyle">
      <ion-icon name="mail" color="white"></ion-icon>
      <ion-input type="email" [(ngModel)]="email"></ion-input>
  </ion-row>
  <ion-row class="inputStyle">
  <ion-icon name="key" color="white"></ion-icon>
  <ion-input type="password" [(ngModel)]="password"></ion-input>
</ion-row>
</ion-grid>
```

Fonte: Autor

Abaixo do código da Figura 20, foi adicionada os comando para inserir os botão, podemos notar na Figura 21, dois botões, um deles e para acessar na tela principal, nomeado o botão de “Entrar”, no segundo botão, foi configurado para fazer o cadastro, clicando no botão ele acessa a página de cadastro, nomeado de “Registrar”.

Figura 21 – Código do botão de acesso.

```
<ion-button style="margin-top: 20px"
            fill="outline"
            shape="round"
            expand="block" (click)="entrar()"
            color="light" (click)="onSubmitLogin()">Entrar</ion-button>

<ion-button
            fill="outline"
            shape="round"
            expand="block" routerLink="/registro"
            color="light">REGISTRAR</ion-button>
</ion-content>
```

Fonte: Autor

Com isso foi finalizado a tela de login, e iniciado o desenvolvimento da tela de cadastro, que é bem parecido com a tela de login, só adicionado um campo, para fazer o cadastro, e faz algumas alterações nos, títulos e nos botões. Logo a pois precisamos configurar os botões da tela de login com a tela de cadastro, assim eles acessar as funções principais entre as páginas.

4.3.1.2 Configuração do Firebase.

Para conseguir fazer um cadastro temos que configurar o Firebase, para que ele armazene os registros, para isso temos que acessar o console web, criar novo projeto, selecione parte web, com isso ele gera um código que você vai inserir, no seu projeto.

Portanto logo após, temos que incrementar na estrutura do projeto, para que funciona corretamente o cadastro e o acesso ao aplicativo, irá pegar uma parte do código, que o Firebase passou e adiciona em dentro *environments* no *environment.ts,* podemos ver na Figura 22, o código inserido.

Figura 22 – Código do Firebase adicionado no aplicativo.

```
export const environment = {
  production: false,
  firebase: {
    apiKey: "AIzaSyALrccZjmp4z2QIojz_IFPzf0dQmq-gEeM",
    authDomain: "comandex-f45a2.firebaseapp.com",
    databaseURL: "https://comandex-f45a2.firebaseio.com",
    projectId: "comandex-f45a2",
    storageBucket: "comandex-f45a2.appspot.com",
    messagingSenderId: "836470530972"
  }
};
```

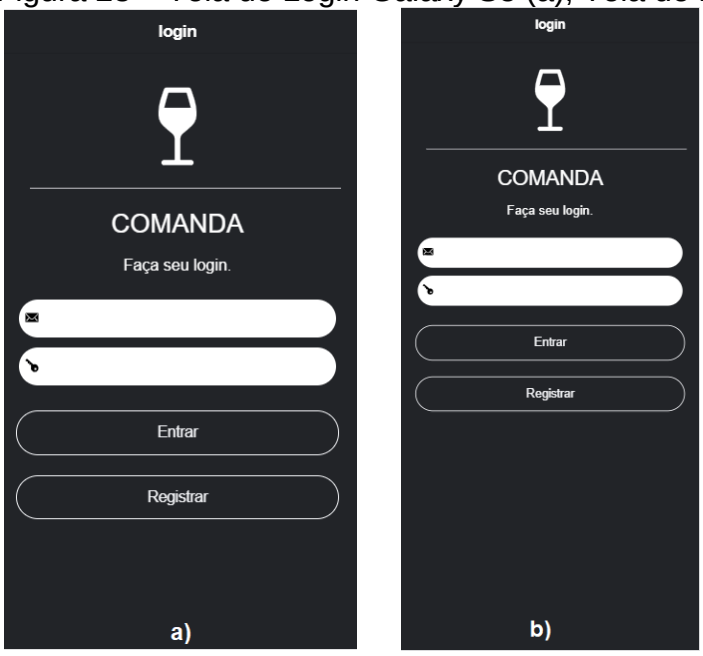
Fonte: Autor

Logo após todos os processos concluímos o segundo *Sprint*, e teve um bom resultado, foi comprido as etapas com êxito. Portanto o aplicativo já e possível acessar e cadastra uma conta.

4.3.1.3 Resultados obtidos

O resultado obtido na segunda *Sprint,* com isso temos a tela de login e a tela de cadastro funciona perfeitamente. Com isso a tela de login consegue acessar a página home ou página principal, para isso precisa fazer o cadastro, com a configuração no Firebase podemos fazer o cadastro facilmente sem erro. Podemos ver nas figuras abaixo, o resultado de como ficou o desenvolvimento da segunda *Sprint*.

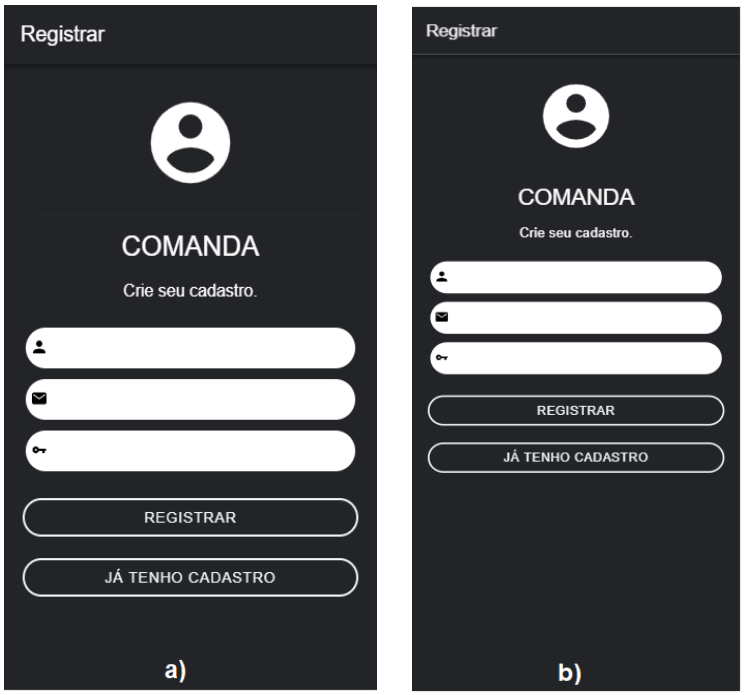
Figura 23 – Tela de Login Galaxy S5 (a), Tela de login iPhone X (b).



Fonte: Do Autor

Na Figura 23 podemos ver duas telas de login uma do Galaxy S5 (a), que roda o sistema Android, a outra tela e do iPhone X (b), que rosa o sistema do iOS, podemos ver que eles se comportam muito bem nos dois sistemas. Nas figuras abaixo podemos notar as telas de cadastro.

Figura 24 – Tela de Cadastro Galaxy S5 (a), Tela de Cadastro iPhone X (b).



Fonte: Do Autor

Na Figura 24, demonstra as telas de cadastro, a representação do Android (a), e temos do iOS (b). Diferente da Figura 23, telas de login temos um campo a mais e diferença na logo.

Para a conclusão da segunda *Sprint*, foi necessária toda a realização das etapas, só assim poder iniciar a terceira *Sprint* sem dificuldade, pois seriam necessários esses recursos.

4.4 TERCEIRO *SPRINT*

No Terceiro *Sprint* iniciou-se o desenvolvimento da tela que faltava para o aplicativo, como tela inicial, tela lateral e tela de comanda. Depois passamos para etapa de implementação do botão Scan, foi bem complicado, teve vários dificuldade para essa implementação com o funcionamento correto, concluído foi feito a personalização.

4.4.1 **Sprint Backlog**

A Tabela 5 apresenta todas as tarefas para realização do terceiro *Sprint*, para atingir a meta.

Tabela 5 – 3° *Sprint Backlog*

3º *Sprint Backlog*	
Item do Product *Backlog*	**Tarefas**
Construção Das Telas Do Aplicativo	Construção da tela inicial (home)
	Construção da tela lateral menu
	Construção da tela de comanda
Implementação Do Botão Scan	Implementação do botão Scan
	Função do QR Code
Personalização Do Botão Scan	Personalização do botão Scan

Fonte: Do Autor.

4.4.1.1 Desenvolvimento

Nesse *Sprint backlog* foi iniciado pela construção das telas que faltava no aplicativo, em termos práticos foi criado uma nova tela comanda e configuração da tela home, que vem como padrão básico. Podendo notar na Figura 25 as alterações na tela home (tela principal).

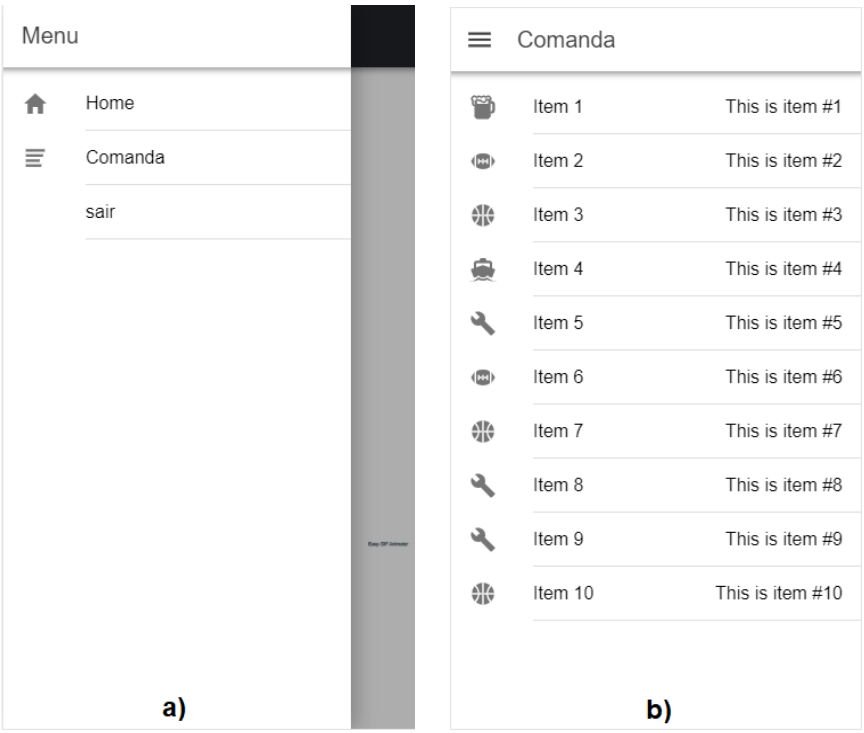
Figura 25 – Código tela Home

```
<ion-header>
  <ion-toolbar color="dark">
    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>
      Início
    </ion-title>
  </ion-toolbar>
</ion-header>
```

Fonte: Do Autor.

Foi criado a tela lateral (menu), e foi criada a tela comanda, a onde irá mostra o resultado. Com isso foi criado todas as telas projetadas, faltando as configurações.

Figura 26 – Tela Lateral (a), Tela de Comanda (b).



Fonte: do Autor

Podemos notar na Figura 26 como ficou as duas telas, a lateral e a de comanda. Na tela de lateral (MENU) podemos ver tela home e a tela de comanda, e o botão de sair. Já na tela de comanda podemos ver uma lista de itens. Com o Desenvolvimento das telas, foi possível iniciar a construção do botão de Scan do QR Code.

Essa parte foi muito trabalhoso, pois teve que ter uma atenção especial, pois foi utilizado uma biblioteca especial para que funcionasse, foi adicionado na tela principal (home) um código que o botão funcionasse.

Figura 27 – Botão Scan

```
scanCode() {
  this.barcodeScanner.scan().then(barcodeData => {
    alert('Barcode data ' + JSON.stringify(barcodeData));

    this.router.navigate(['/comanda', barcodeData['text']])
    this.scannedData = barcodeData;

  }).catch(err => {
    console.log('Error', err);
  });
}
```

Fonte: do Autor

A biblioteca utilizada chamada de ZBar, tem umas características especiais e muito popular, este é um plugin muito bom para digitalizar uma variedade de tipos de código de barras, incluindo EAN-13 / UPC-A, UPC-E, EAN-8, Código 128, Código 39, Interleaved 2 de 5 e QR Code. Precisamos incluir o plug-in no arquivo do módulo principal do aplicativo. Substitua o seguinte código no arquivo app.module.ts (JOLLY.EXE, 2019), podemos ver um trecho do código usado na Figura 28.

Figura 28 – Código ZBar

```
  providers: [
    StatusBar,
    SplashScreen,
    BarcodeScanner,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
```

Fonte: do Autor

Depois de incluir a biblioteca ZBar, foi realizado vários testes diretamente nos dispositivos, pra ver se funcionava corretamente, depois de vários testes foi dado a continuidade, na personalização do botão, pois o botão quem vem é muito simples e fora de estética.

Para incluir um botão, teve que escolher um que fosse moderno e que fosse de agrado com a aplicação.

Figura 29 – Código do botão scan

```
13  <ion-content  class="teste">
14    <button class="button-clear"
15
16        (click)="scanCode()"  block>      <p class="h1large">Escanear</p>
17      </button>
18
19    <div *ngIf="scannedData">
20
21      <p>Identificado: <b>{{scannedData['text']}}</b></p>
22      <p>Formato Escaneado: <b>{{scannedData['format']}}</b></p>
23    </div>
24    </ion-content>
```

Fonte: do Autor

Com a implementação desse código e outros, foi possível deixar o botão mais de acordo com o aplicativo, tivemos um resultado muito bom agradável, podemos ver na Figura 30 logo abaixo:

Figura 30 – Tela principal (home).

Fonte: do Autor

Podemos notar que o botão fica centralizado, uma proporção boa e animado. com esses resultados terminamos o terceiro *Sprint*, que foi fundamental para a próxima etapa.

4.5 QUARTO SPRINT

Chegamos ao quarto e último *Sprint* a onde foi desenvolvido a configuração do banco de dados, configuração da comanda, personalização da comanda e o teste e reparos, para chegar nessa etapa foi fundamental passar por todos os processos.

4.5.1 **Sprint Backlog**

Na Tabela 6 podemos ver a quarta *sprint* e a última, e suas atividades que precisa ser desenvolvido para encerrar o projeto.

Tabela 6 – 4º *Sprint Backlog*

4º *Sprint Backlog*	
Item do Product *Backlog*	**Tarefas**
Configuração Do Banco De Dados	Alimentar o banco de dados.
Configuração Da Comanda	Configuração para exibir a comanda.
Personalização Da Comanda	Personalização da comanda.
Teste e Reparos	Testa o aplicativo
	Fazer reparos
	Personalização final

Fonte: do Autor

4.5.1.1 Desenvolvimento

Para que os dados sejam exibidos de forma correta precisamos de configurar para exibir de maneira correta, assim puxando a API do banco de dados do sistema.

Pra chega nesse resultado foi necessária uma implementação no código, na tela de comanda, uma etapa de nível médio mais que teve um bom grau de dificuldade. Na Figura 31 podemos notar o código que foi adicionado para puxar os dados da comanda.

Figura 31– Puxando a Coleção

```
15    comandas: ComandaI[];
16    xyz:"";
17    ngOnInit() {
18      //  a mesa que vem da rota
19      this.xyz = this.route.snapshot.params['mesa'];
20      console.log(this.xyz);
21      // mandar listar
22      this.comandaService.listar();
23      //this.comandas = this.comandaService.getComandas()
24      this.comandaService.getComandas().subscribe(res => {
25        this.comandas = res
```

Fonte: do Autor

No código acima podemos notar que ele puxa a coleção comanda, que logo tem uma rota chamado “mesa”, depois o código lista e puxa os dados.

Depois de puxar os dados precisamos organizar para exibir corretamente de maneira agradável, pra isso foi preciso muito tempo de edição, para ter um resultado aceitável da maneira que se comporta a tela de comanda.

Figura 32 – *Layout* da comanda

```
1   .border-img {
2       border: 2px solid #702C91;
3       width: 45px !important;
4       height: 45px !important;
5   }
6
7   .item-title {
8       color: rgb(255, 0, 0);
9       font-weight: 700;
10      font-size: 18px;
11      width: 100%
```

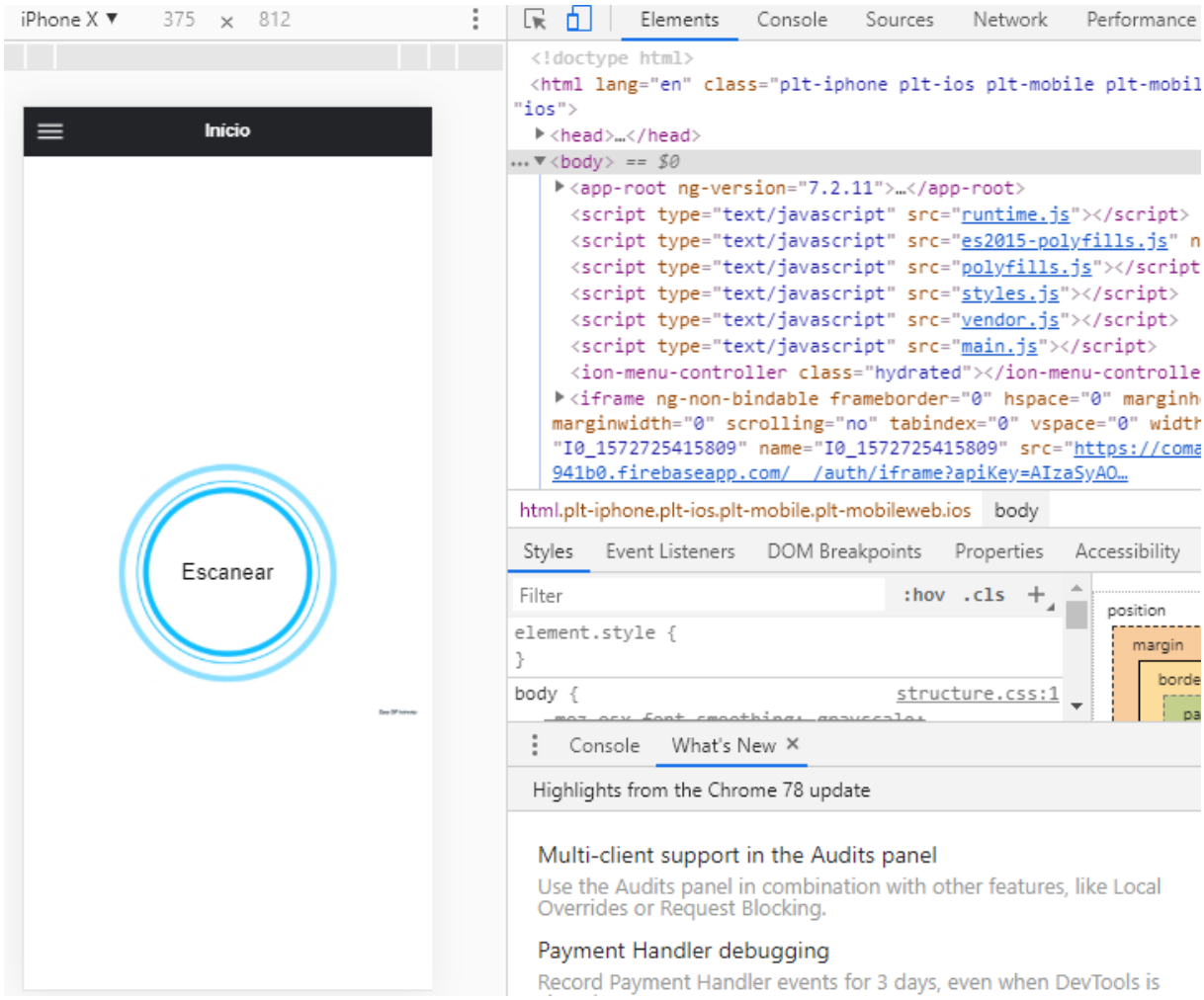
Fonte: do Autor

O código da Figura 32 é o *layout*[21] da comanda é personalizado, onde é determinado o tamanho, letra, fonte e várias outras edições de visual, pode-se notar ajuste na imagem no “.border-img” e ajuste no título no “.item-title”.

Depois desse processo concluído fiz alguns testes manuais, através do smartphone especificado, e foi determinado que ocorreu naturalmente, já no iphone foi feito o teste online através do navegador a onde faz uma simulação, tudo ocorreu de acordo com o esperado, de maneiras simples e fácil.

[21] *Layout* é uma palavra inglesa, muitas vezes usada na forma portuguesa "leiaute", que significa plano, arranjo, esquema, design, projeto.

Figura 33 – Emulador online



Fonte: Google

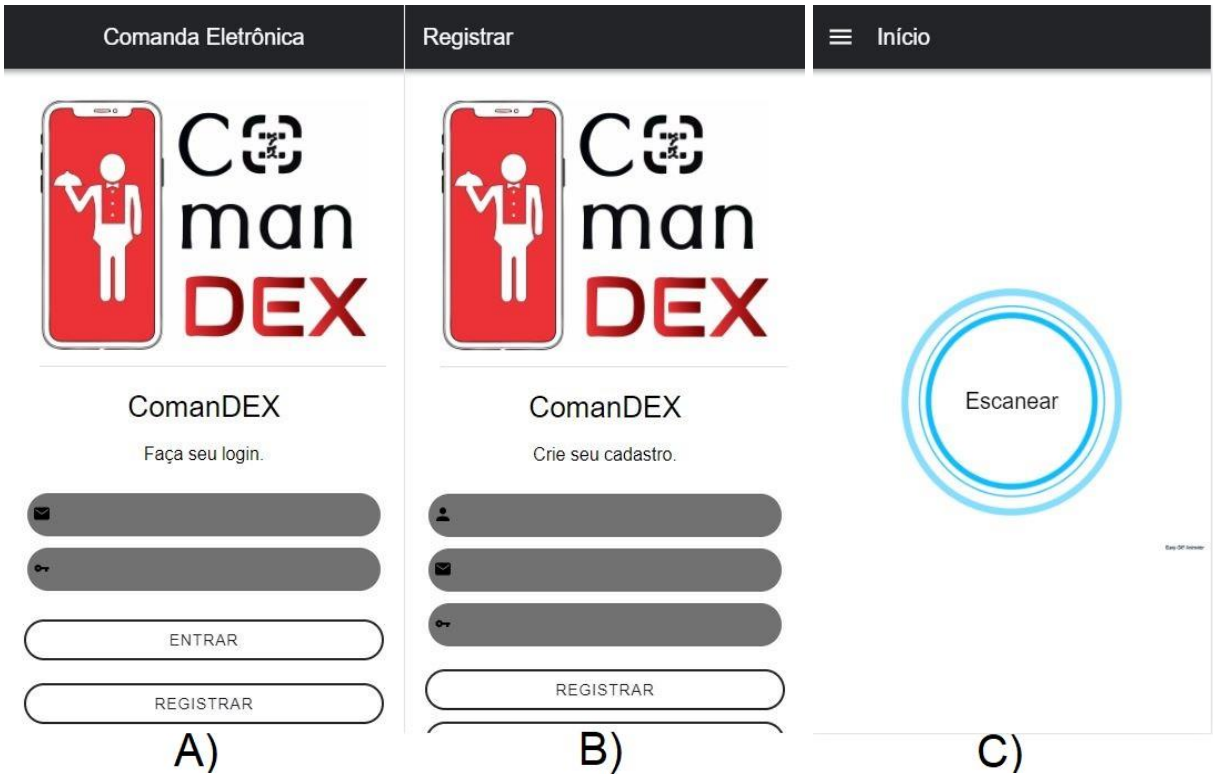
Com isso chegamos no resultado final, com sucesso e um bom desempenho, pois todas as aplicações usadas são tudo de alto nível, assim agregando valor a aplicativo.

4.6 RESULTADO FINAL

Chegamos no resultado final do aplicativo, após todas as *Sprint* podemos notar um aplicativo funcional, bonito e de fácil manuseio, o seu resultado estava dentro do previsto, o aplicativo foi renomeado para “ComanDex”, depois de ter colocado o nome foi criado uma logo.

O resultado das alterações pode ser verificado na Figura 34, onde são exibidas as informações da tela A) login, tela B) Cadastro e tela C) home.

Figura 34 – Resultado final das telas A) Login, B) Cadastro e C) home

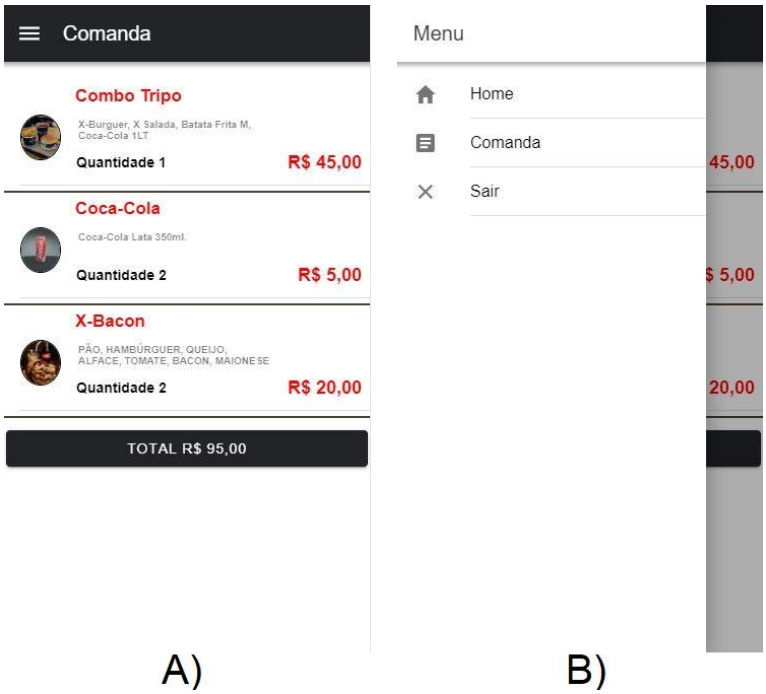


Fonte: do Autor

Podemos notar que ele ficou simples, na tela A) de login temos a logo nome do aplicativo, dois campos para acessar o aplicativo e-mail e senha, logo em baixo tem dois botões um de entra e outro de registrar. Na tela b) muito parecido com a tela A), podemos ver logo e o nome, diferentemente temos 3 campos um a mais a onde colocamos o nome, um botão para registar e outro para voltar. A tela C) é a tela principal home, a onde tem a o scaner, que vai identificar a mesa, não teve alteração nessa tela pois já foi personalizado anteriormente.

Chegamos nas duas últimas telas, podemos notar na Figura 35, temos a tela mais complexas do aplicativo a tela A) comanda, nessa tela temos a informação do pedido como: nome do produto, descrição, quantidade, valor e valor total, para chegar nesse resultado foi preciso fazer um banco de dados, puxar e personalizar as informações. Na tela B) menu lateral, é aonde conseguimos navegar para as outras telas como home, comanda e sair do aplicativo, voltando pra tela login.

Figura 35 – Resultado final das telas A) Comanda, B) Menu Lateral.



Fonte: do Autor

Chegamos ao fim do trabalho, o aplicativo está funcional, leve e flexível, podendo ser utilizado através do sistema do cliente puxando o banco de dados através da API. O aplicativo mostra a informação sobre os produtos e serviços consumido, e o seu valor total.

5 **CONCLUSÃO**

Este trabalho teve como objetivo de oferecer criar um software para estabelecimentos como restaurante ou lanchonete, aumentar sua competitividade, através da transparência com o cliente, proposto por Alves (2017). O trabalho é voltado para uma aplicação de comanda eletrônica, que permite o usuário consultar seus valores e seus produtos consumidos.

Foi escolhida o potencial oferecido pelos dispositivos móveis, que estão muito difundidos entre a população, então para atingir a meta foi necessário o desenvolvimento hibrida, que permite clientes do Android e do iOS acessar a aplicação.

Para o desenvolvimento foi utilizado os *frameworks* Ionic com Angular e o PhoneGap, as linguagens de marcação HTML e CSS, linguagem de programação JavaScript e TypeScript, para o servidor foi utilizada Node.js e o Firebase. A tecnologia QR Code, traz uma grande vantagem e versatilidade para a aplicação.

Foi utilizado a UML, com base no problema foi construído os diagramas que foi necessária para que permitir a visualização das ações dos usuários e dos funcionários, suas classes, e a implantação. Utilizando o modelo Scrum, que permite o detalhamento do que se pode feito, com uma grande facilidade e versatilidade.

Através do modelo Scrum foi feito a tabela *Product Backlog,* conseguimos concluir as etapas feitas, no tempo determinado, pelas tabelas *Sprint Backlog,* foi determinado cada *sprint* após concluir uma foi feito outra até chegar no resultado final.

Resultado final foi um aplicativo leve, implementação pode ser utilizado em sistema já existente, que torna um aplicativo barato, para que um comerciante adquire o app e agrega valor ao estabelecimento e sua marca.

REFERÊNCIA

ALVES, Felipe: **5 Maneiras de ser transparente com Seu Cliente**- Milvus, 2017 Disponível em: <https://milvus.com.br/blog/5-maneiras-de-ser-transparente-com-seu-cliente/>. Acessado em: 14 de maio de 2019

ANDROID. **Meet Android Studio - Android Studio**. 2016. Disponível em: <https: //developer.android.com/studio/intro/index.html>. Acesso em: 01 de maio de 2019.

APPLE. **About Objective-C.** 2014. Disponível em: <https://developer.apple. com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>. Acesso em: 02 de maio de 2019.

APPLE. **News, Apple - Hot News**. 2007. Disponível em: <http://web.archive.org/web/ 20071020040652/http://www.apple.com/hotnews>. Acesso em: 02 de maio de 2019.

ARMIGLIATTO, G. **Lest's crate the Future - REST usa JSON e SOAP usa XML, certo?**. 2017. Disponível em: < http://www.matera.com/blog/post/rest-usa-json-e-soap-usa-xml-certo>. Acesso em: 29 de outubro de 2019.

AUGUSTO, C. **Web Services – O que é, como funciona e os protocolos SOAP e REST**. 2019. Disponível em: < http://ninjadolinux.com.br/web-services-soap-e-rest/>. Acesso em: 29 de outubro de 2019.

BEZERRA, P. T.; SCHIMIGUEL, J. **Desenvolvimento de aplicaçõeses mobile crossplatform utilizando phonegap**. 2016. Disponível em: <http://eumed.net/cursecon/ecolat/ br/16/phonegap.html>. Acesso em: 01 de maio de 2019.

DIONISIO, E. J. D. **Artigo Introdução ao Visual Studio Code,** 2016 Disponível em: <https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418> Acessa em: 08 de maio de 2019.

DUARTE, L. **O que é Node.js e outras 5 dúvidas fundamentais**. 2017. Disponível em: <http://www.luiztools.com.br/post/o-que-e-nodejs-e-outras-5-duvidas-fundamentais/> Acesso em: 05 de maio de 2019.

ECKSCHMIDT, T.; MORITA, S, S. **QR CODE – COMUNICAÇÃO E ENGAJAMENTO NA ERA DIGITAL** – Estados Unidos: MODERATTUS Publicações, 2014.

FIREBASE. **Firebase helps mobile and web app teams succeed**. 2019. Disponível em: <https://firebase.google.com/?hl=pt-BR> Acessado em: 8 de maio de 2019.

FIREBASE. **Firebase - Cloud Firestore**. 2019. Disponível em: < https://firebase.google.com/docs/firestore/?hl=pt-br> Acessado em: 29 de outubro de 2019.

GASPERIN, C. A. **Firebase: O Que é e Como Funciona,** 2017 Disponível em: <http://micreiros.com/firebase-o-que-e-e-como-funciona/> Acessado em: 08 de maio de 2019.

HEITKOTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. **Evaluating crossplatform development approaches for mobile applications. In: Web information systems and technologies**. Springer, 2013. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-36608-6_8>. Acesso em: 02 de maio de 2019.

JOLLY.EXE, 2019: **Ionic 4 | Adding ZBar Barcode / QR Code Scanner in Ionic 4 Application using Cordova and Native Plugin-** Disponivel em:<https://www.freakyjolly.com/ionic-4-adding-barcode-qr-code-scanner-in-ionic-4-application-using-cordova-and-native-plugin/> Acesso em: 12 de maio de 2019.

LOPES, Sérgio. **Aplicações mobile híbridas com Cordova e PhoneGap**, São Paulo: Série Caelum., 2016.

MARTINEZ, M: **UML**. 2019 Disponível em: <https://www.infoescola.com/engenharia-de-software/uml/> Acesso em: 03 de dezembro de 2019.

MATOS, B. R. D.; SILVA, J. G. B. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataforma**. 2016. 63f. Universidade de Brasília - UnB, Brasília .2016.

MILANI, A. **Programando para iPhone e iPad: Aprenda a construir aplicativos para o iOS**. Novatec Editora, 2014. 472 p. ISBN: 978-85-7522-394-9.

MOTA, Gleison. **CANVAS: O que é e para que serve?**. Disponível em: <http://www.administradores.com.br/artigos/empreendedorismo/canvas-o-que-e-e-para-que-serve/109236/>. Acesso em: 08 de maio de 2019.

MOURA, C, A. **FASTER-FOOD: SISTEMA DE GERENCIAMENTO DE PEDIDOS EM FAST-FOODS POR MEIO DE UM DISPOSITIVO MÓVEL**. 2018. 96f. INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIOGRANDENSE - CÂMPUS PASSO FUNDO CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET, Passo Fundo. 2018.

NABUCO, T. S.; FAÇANHA, A. R.; ARAÚJO, M. C. C. **Impactos da diversidade de um ecossistema móvel no desenvolvimento de softwares**. VII CONNEPI (Congresso Norte Nordeste de Pesquisa e Inovação), 2012. Palmas.

NPM. **About npm.** 2014. Disponível em: <https://www.npmjs.com/about> Acesso em 06 de maio de 2019.

PAPAJORGJI, P. **Automated Enterprise Systems for Maximizing Business Performance**. 1 edition. ed. Hershey, PA: IGI Global, 2015. ISBN 978-1-4666-8841-4.

PEREIRA, C. R. **Node.js - Aplicações web real-time com Node.js**, Casa Do Codigo, 2014.

PIMENTA, Marcelo Severo. **O quadro de modelo de negócios: um caminho para criar, recriar e inovar em modelos de negócios**, 2015. Disponível em: < http://www.bibliotecas.sebrae.com.br/chronus/ARQUIVOS_CHRONUS/bds/bds.nsf/be606c09f2e9502c51b09634badd2821/$File/4439.pdf>. Acesso em 28 de agosto de 2019

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

RELEVANT SOFTWARE**. top 6 reasons to use angular 5 framework for your project**. Disponivel em: <https://relevant.software/blog/top-6-reasons-to-use-angular-5-framework-for-yourproject/ >. Acesso em: 05 de maio de 2019.

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

SANTOS, G. **Node.js — O que é, por que usar e primeiros passos**. 2014. Disponível em: <https://medium.com/thdesenvolvedores/node-js-o-que-%C3%A9-por-que-usar-e-primeiros-passos-1118f771b889> Acesso em: 05 de maio de 2019.

SCUADRA. **Conheça 4 tendências de Food Service para 2018**. 2018. Disponível em: <https://www.scuadra.com.br/blog/conheca-4-tendencias-de-food-service-para-2018/>. Acesso em: 14 de maio de 2019.

URSINO, D.; CAPANNA, C. A. **AngularJS - un framework di frontiera per la realizzazione**. 2015. Disponível em: <http://www.barbiana20.unirc.it/wp-content/uploads/2015/07/Tesi-Cristiano-finale-2.pdf>. Acesso em: 02 de maio de 2019.