

**UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES -  
CAMPUS DE ERECHIM**

**DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO CURSO DE  
CIÊNCIA DA COMPUTAÇÃO**

**JACKSON FELIPE MAGNABOSCO**

**DESENVOLVIMENTO DE UM APLICATIVO DE CONTROLE AUTOMATIZADO  
NO PROCESSO DE PRODUÇÃO DE CERVEJA ARTESANAL**

**ERECHIM - RS**

**2020**

**JACKSON FELIPE MAGNABOSCO**

**DESENVOLVIMENTO DE UM APLICATIVO DE CONTROLE AUTOMATIZADO  
NO PROCESSO DE PRODUÇÃO DE CERVEJA ARTESANAL**

**Trabalho de conclusão de curso, apresentado  
ao Departamento de Engenharias e Ciências  
da Computação Universidade Regional  
Integrada do Alto Uruguai e das Missões –  
Campus de Erechim.**

**Orientador: Prof. Neilor Avelino Tonin**

**ERECHIM - RS**

**2020**

## **AGRADECIMENTOS**

Agradeço primeiramente aos meus pais Alcione José Magnabosco e Evanir Terezinha Magnabosco, por todo o apoio e esforço realizado por eles para que eu pudesse conquistar a minha graduação, pelo exemplo de caráter, persistência e coragem. Aos meus irmãos Jean Carlo Magnabosco e Jonas Fernando Magnabosco pela parceria em todos os momentos e minha namorada Nágila Zortéa por me ajudar com as revisões e correções. Em especial a meu padrinho Milton Magnabosco pelo auxílio que foi de fundamental importância para que eu chegasse neste momento.

Aos docentes, que além de desempenhar com mérito seus papéis de educadores, mostraram-se profissionais apoiadores e amigos em todas as etapas concluídas. Especial agradecimento ao Prof. Neilor Avelino Tonin, meu orientador, pela preparação e apoio dedicado.

A Compasso Uol, pelos conhecimentos adquiridos na área de desenvolvimento de software.

Aos colegas de graduação, especialmente aos mais presentes, Teyson Lorenzon, Bruno Beltrame, Cristian Abramchuk, entre outros, pelo apoio em momentos de tensão, pelo respeito demonstrado e pela parceria em todas as situações vivenciadas.

Aos amigos, relegados a segundo plano por conta da vida acadêmica, mas que nunca deixaram de estar ao meu lado.

Aos bares que propiciaram as comemorações e sempre tinham uma cerveja boa e gelada para aliviar a tensão.

A todos que contribuíram em qualquer aspecto para a realização deste trabalho, seja pelo apoio moral ou técnico prestado. A todos vocês, muito obrigado.

*"A cerveja e a cachaça são os piores inimigos do homem. Mas o homem que foge dos seus inimigos é um covarde."*

(Zeca Pagodinho)

## RESUMO

A tecnologia digital vem sendo a resposta para os grandes desafios atuais na produção de cerveja artesanal, trazendo soluções inovadoras e agregando bons resultados, facilitando e tornando mais eficientes a execução de tarefas na rotina da fabricação da cerveja. O objetivo deste trabalho foi desenvolver um aplicativo móvel, nomeado como “Velha Guarda”, o qual consiste em apresentar uma solução para o controle automatizado no processo de produção de cerveja artesanal, proporcionando maior segurança, agilidade, produtividade e redução de custos. Este trabalho utilizou a metodologia orientada a objeto e as ferramentas Android Studio com a plataforma de desenvolvimento Dart com o framework Flutter, juntamente com o banco de dados do Firebase, o Astah Community para modelagem dos diagramas UML e o Trello com a metodologia Scrum para gerenciamento do projeto. O presente trabalho aplicou a estrutura do Módulo ESP8266 NodeMCU V3 utilizando a linguagem de programação C no ambiente de desenvolvimento integrado do Arduino, juntamente com o sensor de temperatura DS18B20 a prova d’água para medir a temperatura em todos os processos de produção da cerveja, desde fervura, resfriamento, fermentação até a maturação, avaliando os dados obtidos por meio da temperatura no sensor. Por meio da utilização do aplicativo os cervejeiros independentes, apontaram como um dos resultados à facilidade proporcionada “na palma da mão”, pois o aplicativo propõe o processo de criação da receita, o aferimento de temperatura e ainda a venda ao consumidor. Concluindo assim que o aplicativo é efetivo por tornar a produção facilitada, a gestão dos demais processos envolvidos e ainda a proporciona maior comodidade ao consumidor.

**Palavras-chave:** Automação. Aplicativo. Serviço Web. Cervejeiros Independentes.

## ABSTRACT

Digital technology has been the answer to the great current challenges in the production of craft beer, bringing innovative solutions and adding good results, facilitating and making more efficient the execution of tasks in the routine of brewing. The objective of this work was to develop a mobile application, named “Velha Guarda”, which consists of presenting a solution for the automated control in the craft beer production process, providing greater security, agility, productivity and cost reduction. This work used the object-oriented methodology and Android Studio tools with the Dart development platform with the Flutter framework, together with the Firebase database, the Astah Community for modeling UML diagrams and Trello with the Scrum methodology for management. From the project. This work applied the structure of the ESP8266 NodeMCU V3 Module using the C programming language in the Arduino integrated development environment, together with the waterproof DS18B20 temperature sensor to measure the temperature in all beer production processes, from boiling, cooling, fermentation to maturation, evaluating the data obtained through the temperature in the sensor. Through the use of the application, independent brewers pointed out as one of the results of the ease provided “in the palm of the hand”, as the application proposes the process of creating the recipe, measuring the temperature and even selling it to the consumer. Thus concluding that the application is effective for making production easier, the management of the other processes involved and still provides greater convenience to the consumer.

**Keywords:** Automation. App. Web Service

## LISTA DE FIGURAS

Figura 01 - Widget.....	15
Figura 02 - HTML/CSS análogos em Flutter.....	15
Figura 03 - Cupertino e material.....	16
Figura 04 - NoSQL.....	19
Figura 05 - Provedores de login.....	21
Figura 06 - OneSignal - push notification.....	22
Figura 07 - Sensor DS18B20.....	23
Figura 08 - ESP12.....	25
Figura 09 - Módulo ESP8266 NodeMCU V3.....	25
Figura 10 - Módulo ESP8266 Pinagem.....	26
Figura 11 - BreadBoard.....	26
Figura 12 - Arduino Preferências.....	27
Figura 13 - URL's adicionais de gerenciadores de placa.....	28
Figura 14 - Gerenciador de placas.....	28
Figura 15 - Instalação do pacote da placa.....	29
Figura 16 - Escolha da placa.....	29
Figura 17 - Porta COM.....	30
Figura 18 - Incluir biblioteca.....	31
Figura 19 - Dallas temperature.....	31
Figura 20 - OneWire.....	32
Figura 21 - Instalação Android Studio.....	33
Figura 22 - Tela inicial Android Studio.....	33
Figura 23 - Flutter console.....	34
Figura 24 - Configurando a variável de ambiente.....	35
Figura 25 - Flutter Doctor.....	35

Figura 26 - Android Studio Plugins.....	36
Figura 27 - Plugins Dart e Flutter.....	37
Figura 28 - Montagem dos componentes.....	38
Figura 29 - Estrutura montada.....	38
Figura 30 - Código Arduino.....	39
Figura 31 - Monitor de temperatura.....	42
Figura 32 - Produção da cerveja.....	43
Figura 33 - Produto final.....	43
Figura 34 - Caso de uso geral.....	46
Figura 35 - Caso de uso efetuar login.....	46
Figura 36 - Caso de uso módulo gerente.....	47
Figura 37 - Caso de uso módulo institucional.....	47
Figura 38 - Caso de uso módulo funcionário.....	48
Figura 39 - Diagrama de classe.....	53
Figura 40 - Diagrama de sequência manter login.....	54
Figura 41 - Diagrama de sequência manter produto.....	55
Figura 42 - Diagrama de sequência manter funcionário.....	56
Figura 43 - Diagrama de sequência manter relatório.....	57
Figura 44 - Diagrama de sequência módulo institucional.....	58
Figura 45 - Diagrama de atividade fazer login.....	59
Figura 46 - Diagrama de atividade módulo relatório.....	60
Figura 47 - Diagrama de Componentes e Implantação.....	61
Figura 48 - Abertura.....	62
Figura 49 - Autenticação.....	63
Figura 50 - Inicial.....	64
Figura 51 - Categoria de produto.....	65
Figura 52 - Produto.....	66



Figura 53 - Carrinho de compra.....	67
Figura 54 - Forma de pagamento.....	68
Figura 55- Sobre.....	69
Figura 56 - Automação.....	70
Figura 57 - Temperatura.....	71
Figura 58 - Cronômetro.....	72
Figura 59 - Receitas.....	73

## LISTA DE QUADROS

Quadro 01 – Prós e contras entre desenvolvimento híbrido e nativo.....	13
Quadro 02 - Ferramentas integradas.....	20
Quadro 03 - Descrição do caso de uso efetuar login.....	49
Quadro 04 - Descrição do caso de uso manter funcionário.....	50

## LISTA DE ABREVIATURA E SIGLAS

APP – *Application*  
API – *Application Programming Interface*  
CSS – *Cascading Style Sheet*  
HTML - *HyperText Markup Language*  
HTTP - *Hypertext Transfer Protocol*  
IDE – *Integrated Development Environment*  
IP - *Internet Protocol*  
JSON - *Javascript Object Notation*  
NoSQL - *Not Only SQL*  
OS - *Operating System*  
SDK – *Software Development Kit*  
SQL – *Structured Query Language*  
UML - *Unified Modeling Language*  
URL – *Uniform Resource Locator*  
OS - *Operating System*

## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>09</b>
<b>2 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS.....</b>	<b>11</b>
2.1 METODOLOGIAS DE DESENVOLVIMENTO.....	11
2.1.1 Desenvolvimento nativo.....	11
2.1.2 Desenvolvimento de aplicativo web .....	12
2.1.3 Desenvolvimento de aplicativo híbrido .....	12
2.1.4 Análise das metodologias de desenvolvimento.....	12
2.2 FLUTTER .....	14
2.2.1 Pacotes Flutter.....	14
<b>3 BANCO DE DADOS NÃO RELACIONAL.....</b>	<b>17</b>
3.1 TIPOS DE BANCOS DE DADOS NÃO RELACIONAL.....	17
3.1.1 Análise das metodologias de banco de dados não relacional.....	18
3.2 FIREBASE.....	18
3.2.1 OneSignal.....	21
<b>4 HARDWARE.....</b>	<b>22</b>
4.1 SENSORES INTELIGENTE.....	22
4.1.1 Sensor DS18B20.....	23
4.2 PROTOCÓLO ONE-WIRE.....	24
4.3 MÓDULO WIFI ESP8266 NODEMCU V3 CP2102 ESP-12E.....	24
4.3.1 ESP8266.....	24
4.3.2 NodeMCU V3.....	25
4.3.3 BreadBoard.....	26
<b>5 CONFIGURAÇÕES .....</b>	<b>27</b>

5.1 ARDUINO.....	27
<b>5.1.1 Placa.....</b>	<b>27</b>
<b>5.1.2 Sensor.....</b>	<b>30</b>
<b>5.1.3 OneWire.....</b>	<b>32</b>
5.2 ANDROID STUDIO.....	32
<b>5.2.1 Flutter SDK.....</b>	<b>34</b>
5.3 MONTANDO OS COMPONENTES.....	37
<b>6 LEVANTAMENTO DE REQUISITOS.....</b>	<b>44</b>
6.1 METODOLOGIA DE ANÁLISE.....	44
6.2 DIAGRAMAS UML.....	44
6.3 DESCRIÇÃO DO GERENCIAMENTO.....	44
6.4 DESCRIÇÃO DOS USUÁRIOS.....	45
6.5 MÓDULOS.....	45
<b>6.5.1 Perspectiva do Produto: Módulo Institucional.....</b>	<b>45</b>
<b>6.5.2 Perspectiva do Produto: Módulo Gerente.....</b>	<b>45</b>
<b>6.5.3 Perspectiva do Produto: Módulo Funcionário.....</b>	<b>45</b>
6.6 DIAGRAMAS DE CASO DE USO.....	45
<b>6.6.1 Diagrama de Caso de uso geral.....</b>	<b>46</b>
<b>6.6.2 Efetuar login.....</b>	<b>46</b>
<b>6.6.3 Módulo Gerente.....</b>	<b>47</b>
<b>6.6.4 Módulo Institucional.....</b>	<b>47</b>
<b>6.6.5 Módulo Funcionário.....</b>	<b>48</b>
6.7 DIAGRAMA DE CLASSE.....	53
6.8 DIAGRAMA DE ENTIDADE RELACIONAMENTO.....	53
6.9 DIAGRAMA DE SEQUÊNCIA.....	54
<b>6.9.1 Manter login.....</b>	<b>54</b>
<b>6.9.2 Manter Produto.....</b>	<b>55</b>

<b>6.9.3 Manter funcionário.....</b>	<b>56</b>
<b>6.9.4 Manter relatório.....</b>	<b>57</b>
<b>6.10 DIAGRAMA DE ATIVIDADE.....</b>	<b>58</b>
<b>6.10.1 Módulo Institucional.....</b>	<b>58</b>
<b>6.10.2 Fazer Login.....</b>	<b>59</b>
<b>6.10.3 Relatório.....</b>	<b>60</b>
<b>6.11 DIAGRAMA DE COMPONENTES E IMPLANTAÇÃO.....</b>	<b>61</b>
<b>7 DESENVOLVIMENTO DO APLICATIVO .....</b>	<b>61</b>
<b>7.1 ABERTURA.....</b>	<b>62</b>
<b>7.2 AUTENTICAÇÃO.....</b>	<b>63</b>
<b>7.3 INICIAL.....</b>	<b>64</b>
<b>7.4 CATEGORIA DE PRODUTOS.....</b>	<b>65</b>
<b>7.5 PRODUTO.....</b>	<b>66</b>
<b>7.6 CARRINHO DE COMPRA.....</b>	<b>67</b>
<b>7.7 FORMA DE PAGAMENTO.....</b>	<b>68</b>
<b>7.8 SOBRE.....</b>	<b>69</b>
<b>7.9 AUTOMAÇÃO.....</b>	<b>70</b>
<b>7.10 TEMPERATURA.....</b>	<b>71</b>
<b>7.11 CRONÔMETRO.....</b>	<b>72</b>
<b>7.12 RECEITA.....</b>	<b>73</b>
<b>8 DISCUSSÃO E RESULTADOS.....</b>	<b>74</b>
<b>9 CONCLUSÃO E TRABALHOS FUTUROS.....</b>	<b>75</b>
<b>REFERÊNCIAS .....</b>	<b>77</b>

# 1 INTRODUÇÃO

Devido ao período da atual sociedade, onde as pessoas sentem necessidade de estarem conectadas. Isso porque a informação se tornou uma propriedade muito valiosa e o acesso a ela em qualquer lugar e instante é indispensável. Esse comportamento é percebido constantemente, visto que grande parte das atividades e do tempo das pessoas tem relação com a utilização de uma conexão a internet ou algum dispositivo móvel.

A tecnologia já está no cotidiano das pessoas e o maior desafio é saber aproveitá-la em nosso favor. A inovação na produção de cerveja está evoluindo constantemente e, com isso o produtor também é favorecido, pois através da tecnologia aplicada as cervejarias são possíveis ter o crescimento desejado.

Velha Guarda Cervejas Artesanais é o nome da aplicação apresentada por este trabalho. Ela foi desenvolvida tendo como objetivo auxiliar o pequeno produtor de cerveja artesanal, automatizando sua produção podendo medir a temperatura dos processos da produção da cerveja, desde a fervura, resfriamento, fermentação até a maturação, avaliando os dados obtidos por meio da temperatura no sensor.

Devido ao setor cervejeiro está crescendo cada dia mais, este trabalho propõe desenvolver um aplicativo, compatível para Android e iOS, utilizando o framework Flutter. Com uma ferramenta simples utilização será possível oferecer uma interface de fácil usabilidade para auxiliar o produtor a produzir sua cerveja. Uma vez que estas informações poderão ser facilmente acessadas pelo produtor através do próprio aplicativo.

Outro objetivo do trabalho também é o estudo e aplicação de tecnologias modernas no desenvolvimento de aplicativos móveis conforme é apresentado no início do trabalho. No segundo capítulo são abordadas técnicas de desenvolvimento para dispositivos móveis e tecnologias para implementação de aplicações híbridas.

No terceiro capítulo são abordados conceitos de bancos de dados não relacionais bem como o Firebase estudado e utilizado neste trabalho.

Após o estudo dos *softwares* utilizados para a elaboração da aplicação, será descrito no Capítulo 4 os *hardwares* utilizados no aplicativo através do detalhamento das funcionalidades. A descrição da configuração do sistema, juntamente com a montagem dos componentes é tratado no Capítulo 5.

No Capítulo 6 será descrito o processo de desenvolvimento do sistema através do detalhamento das funcionalidades nos diagramas de modelagem UML.

No Capítulo 7 dá-se início ao desenvolvimento da aplicação com a análise da área em foco e do sistema atual, seguida dos resultados finais do sistema. No oitavo capítulo são abordados as discussões e resultados obtidos.

Finalmente, no ultimo capítulo são apresentadas as conclusões e propostas para futuros trabalhos.



## **2 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS**

Neste capítulo serão abordados métodos de desenvolvimento do aplicativo bem como as tecnologias utilizadas para o desenvolvimento deste trabalho.

### **2.1 METODOLOGIAS DE DESENVOLVIMENTO**

Quando o assunto é desenvolvimento de aplicativos, a primeira pergunta que surge para o desenvolvedor é para qual plataforma ele deve desenvolver. Hoje, as principais e mais utilizadas plataformas são Android e iOS, mas ainda existem outras plataformas como Windows Phone, BlackBerry, entre outras.

A segunda questão é qual abordagem de desenvolvimento utilizar, pois existe mais de uma. Segundo Chede (2013), comenta sobre três abordagens, cada uma com suas vantagens e desvantagens, as técnicas são as seguintes: desenvolvimento nativo, desenvolvimento web e desenvolvimento híbrido. Um aplicativo nativo é focado em uma linguagem específica para cada plataforma. Os aplicativos web são acessados pelo navegador e os modelos híbridos são aplicações multi-plataforma que agrupam características dos aplicativos nativos com o modelo web.

#### **2.1.1 Desenvolvimento nativo**

Chede (2013) explica que um aplicativo nativo é focado em uma plataforma de hardware e software específico. Isto significa que toda a capacidade desta plataforma e seus dispositivos pode ser aproveitada. A aplicação é baixada e executada no dispositivo e como resultado não pode ser aplicada em outra plataforma. De forma comum é encontrado nas lojas de aplicativos como a *Google play* e *AppStore*.

Conforme a aplicação tem acesso direto aos recursos físicos e lógicos como câmera, acelerômetro, contatos entre outros. Desta forma consegue utilizar o máximo de recursos dos dispositivos demonstrando a alta performance do aplicativo. O desenvolvimento nativo utiliza os componentes básicos da plataforma como botões ícones e interfaces (MADUREIRA, 2017).

Em contrapartida, o desenvolvimento nativo aumenta a complexidade e leva mais tempo para ser desenvolvido assim exigindo mais soluções e testes para cada plataforma. Outro inconveniente é o fato do aplicativo utilizar a memória interna do dispositivo para ser

armazenado e precisa de liberação das lojas de aplicativos para ser comercializado (MADUREIRA, 2017).

### **2.1.2 Desenvolvimento de aplicativo web**

Chede (2013) fala que os aplicativos web são entregues via navegadores como muitas aplicações web, mas com algumas diferenças, pois o aplicativo deve reconhecer a plataforma e se ajustar de forma correta, mostrando diferentes telas nas diversas resoluções. Entre os benefícios da técnica pode-se citar a facilidade de desenvolvimento, dado que a aplicação é multiplataforma. Não é necessário realizar a publicação do aplicativo em lojas para ser comercializado e não é preciso fazer download do mesmo.

Chede (2013) também comenta que o HTML5 permite implementar interfaces gestuais e permite utilizar a memória interna do dispositivo assim minimizando o acesso online de dados.

No entanto a diversidade de dispositivos faz com que a etapa de testes ainda seja uma grande barreira no processo, como o aplicativo web não pode ser baixado ele também não pode ser utilizado sem conexão à internet (MADUREIRA, 2017).

### **2.1.3 Desenvolvimento de aplicativo híbrido**

As aplicações híbridas são capazes de combinar aplicativos nativos e web, ele pode ser baseado em HTML5, CSS e Javascript ou Dart trazendo uma incorporação no código que é envelopado em um container que é empacotado, instalado e executado como uma aplicação nativa (MADUREIRA, 2017).

### **2.1.4 Análise das metodologias de desenvolvimento**

Cada abordagem dispõem de exigências próprias, para poder começar a desenvolver aplicativos móveis é necessário ter uma estratégia estabelecida. Para esta definição existem diversas variáveis a serem analisadas, como a competência da equipe desenvolvedora, existência de propósito de monetizar a aplicação, orçamento à disposição para desenvolvimento ou evolução contínua conforme apresenta o quadro 01.

**Quadro 01** - Prós e contras entre desenvolvimento híbrido e nativo

Prós	Contras
<ul style="list-style-type: none"><li>● <b>Escrever o código uma vez:</b> o código é escrito uma vez, e pode ser utilizado em qualquer plataforma que tenha suporte pela ferramenta.</li><li>● <b>Mais rápido, equipes menores:</b> o tempo de desenvolvimento e a quantidade de profissionais diminui consideravelmente pois não é preciso desenvolver aplicações individuais para cada plataforma.</li><li>● <b>Conhecimento e ferramentas:</b> não é necessário conhecimento, nem ferramentas específicas para cada plataforma.</li><li>● <b>Investimento econômico:</b> devido a redução de tempo, da equipe, do conhecimento e das ferramentas, o desenvolvimento do aplicativo requer um investimento menor.</li></ul>	<ul style="list-style-type: none"><li>● <b>Potencial nativo:</b> não é possível utilizar totalmente os recursos específicos de cada plataforma, pois a interface de desenvolvimento é a mesma para qualquer plataforma.</li><li>● <b>Desempenho:</b> existe uma perda de desempenho em relação aos aplicativos nativos devido ao fato da adição de mais camadas de software responsáveis pela transparência entre plataformas.</li><li>● <b>Interface específica:</b> cada plataforma possui algumas linhas guias específicas para a interface de seus aplicativos, com desenvolvimento híbrido o aplicativo terá a mesma interface em todas as plataformas.</li></ul>

Fonte: O próprio Autor (2020)

Após a exploração e verificação das abordagens de desenvolvimento relatados chegou se ao conceito que a melhor abordagem para este trabalho é o desenvolvimento híbrido, pois o aplicativo irá executar de forma nativa, permitindo o usuário utilizar suas funcionalidades sem uma conexão a internet, além das tecnologias utilizadas serem mais conhecidas e simples de manipular.

## 2.2 Flutter

Desenvolvido pela equipe da Google, é um framework de desenvolvimento híbrido que utiliza a linguagem de programação Dart no ambiente de desenvolvimento integrado Android Studio para criação de seus aplicativos. A principal ideia de utilizar ele no trabalho é, ele é totalmente multiplataforma, de código aberto e gratuito, com uma licença limpa e também é um padrão da ECMA com mais de 100 contribuidores externos.

Desenvolve-se apenas um código que vai rodar em diversos dispositivos. Uma tecnologia de fácil aprendizado, especialmente para quem já programou em alguma outra linguagem de programação, pois carrega consigo algumas características similares como classes, orientação a objetos e fortemente tipada.

O Flutter utiliza uma máquina virtual chamada Dart VM que serve de intérprete para imitar a arquitetura do *hardware* da máquina quando o *software* for executado. Esta máquina virtual facilita a portabilidade de um idioma para novas plataformas. Sua arquitetura é feita por uma *engine* que foi totalmente desenvolvida na linguagem de programação C++, convertendo o código para nativo, ou seja binário. (LIMA, 2019).

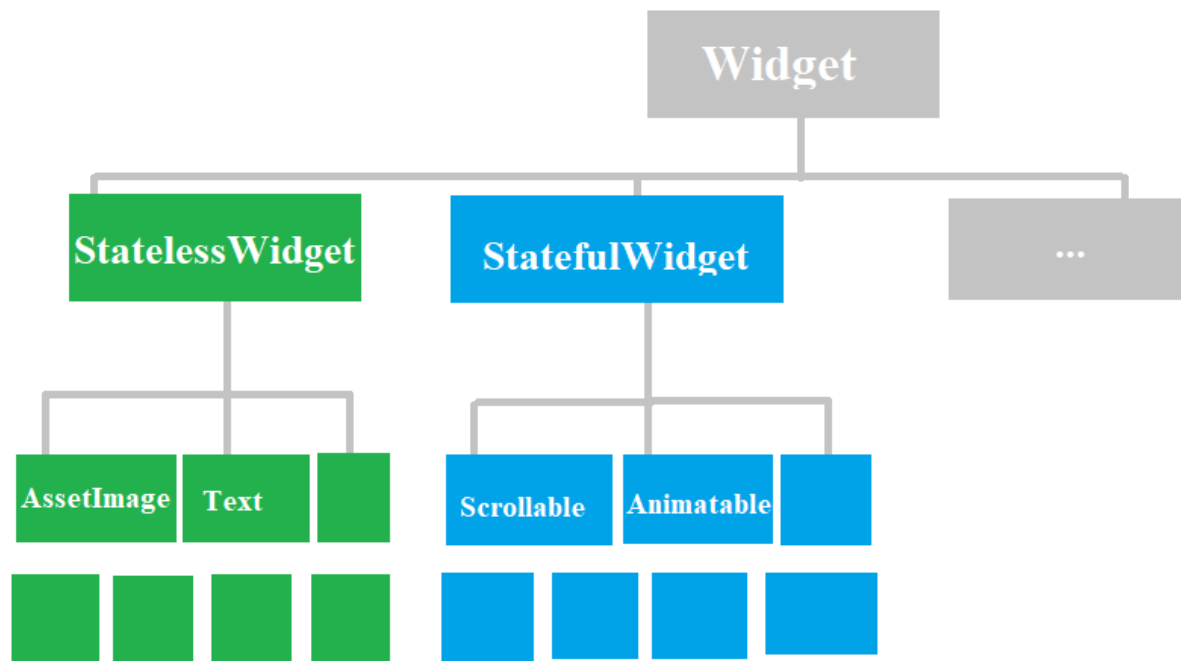
Outra vantagem que torna indispensável é o *Just in time*, a recompilação do código no dispositivo, enquanto o aplicativo está rodando, a aplicação não perde o estado de desenvolvimento. Isso gera um ciclo de desenvolvimento muito rápido e produtivo, possibilitando o recarregamento expresso do aplicativo em tempo de execução. (LIMA, 2019).

O compilador do Dart dispõe de um núcleo, onde a linguagem é processada, o *analyzer*, tem como objetivo realizar a navegação, refatoração e adequação do código, o *Analysis Server*, atua como um servidor local que apresenta o código para os editores. Todo o processo funciona de maneira eficiente. (HONDA, 2019)

O Flutter se preocupa muito com a performance e limitação de *hardware*, assim utiliza 60 quadros por segundos nos aplicativos, Com o uso do código nativo, ele não exibe uma interface lenta, dispõe de um renderizador *Mobile First* apressurado por GPU para que dê-se textura da UI entre as plataformas e o dispositivo.

Esta linguagem de programação é baseada em *widgets*, que são os componentes da aplicação que interagem com o APP. Tudo no Flutter são *widgets*, desde o texto até os botões, ou seja, uma árvore de widgets para o widget conforme a Figura 01 demonstra (DIAS, 2018).

Figura 01 – Widget



Fonte: Adaptado de (DIAS, 2018)

A estrutura do Flutter é similar a estrutura do HTML/CSS na Figura 02 demonstra o HTML/CSS do lado esquerdo e o Flutter do lado direito.

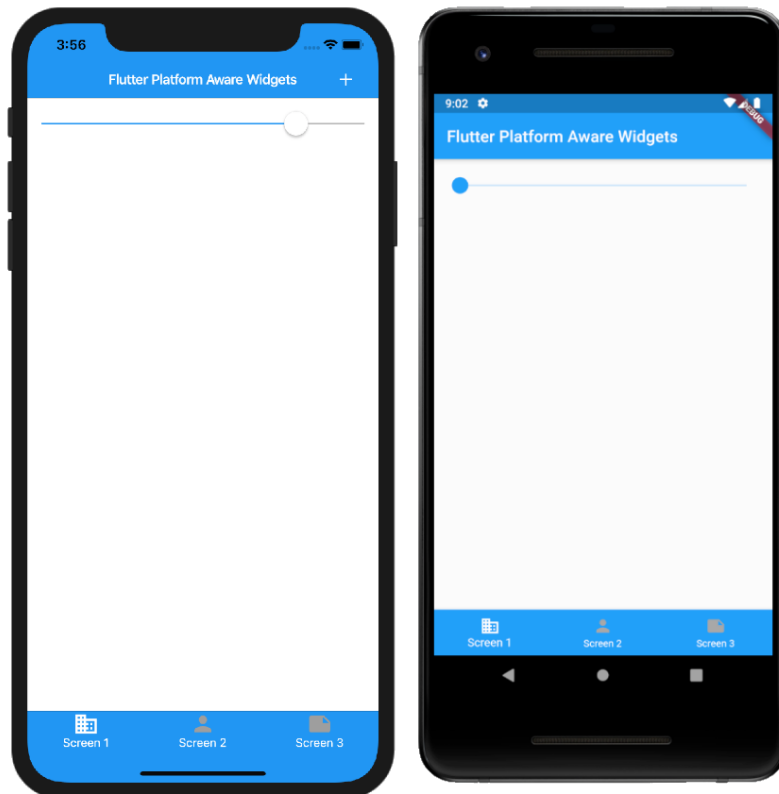
Figura 02 - HTML/CSS análogos em Flutter



Fonte: Adaptado de (DIAS, 2018)

O Flutter tem tanto o sistema de design do Android, que é chamado de material, quanto o sistema de design do iOS, que é chamado de Cupertino, assim simulando as duas interfaces como demonstra a Figura 03.

Figura 03 - Cupertino e material



Fonte: Baseada na imagem original de (MOORE, 2019)

O Flutter permite a utilização de pacotes compartilhados composto por outros desenvolvedores. Isso proporciona construir ligeiramente um aplicativo sem precisar desenvolver tudo do zero.

### 3 BANCO DE DADOS NÃO RELACIONAL

Com o progresso dos dispositivos móveis, a proporção de pessoas online e a simplicidade em se conectar faz com que as aplicações armazenem, analisem uma quantidade cada vez maior de informações (STEPPA, 2009).

De fato que a definição de banco de dados não relacionais se originou em 1998, exclusivamente nos últimos anos vem se transformando mais difundido e usufruído pelas empresas. O grande incentivo para utilizar o *Not Only SQL* (NoSQL) é resolver o impasse de escalabilidade dos bancos tradicionais, pelo fato de ser excessivamente custoso ou complexo escalar um banco de dados SQL relacional (IANNI, 2012).

Este tipo de banco não possui elaboração definida, por isso podem salvar dados de qualquer forma e estrutura, pelo fato de não haver verificação de integridade e de relacionamento, podendo dividir os dados em vários nós chamadas de *shard*, assim tornando possível o processamento destes dados em paralelo, aumentando a quantidade de dados processados e ao mesmo tempo diminuindo o custo do hardware pois não é necessário um grande poder de processamento sendo que cada nó processará uma parte menor de toda a base (SADALAGE, 2014).

#### 3.1 TIPOS DE BANCOS DE DADOS NÃO RELACIONAL

Na atualidade há inúmeras utilizações de gêneros de bancos de dados não relacional, dentre eles podemos citar:

**Banco de dados com chave/valor:** estes bancos armazenam objetos indexados por chaves, estas chaves dispõe de valores associados a elas que podem ser de qualquer tipo. Alguns exemplos de bancos que atuam desta forma: DybaniDb, CouchBase, Riak, Azure Table Storage, Tokyo Cabinet, Berkeley DB, etc (SADALAGE,2014).

**Banco de dados orientados a documentos:** estes bancos armazenam objetos com atributos e valores, referindo-se que os atributos de um objeto podem ser absolutamente diferentes de outro objeto e o valor de uma propriedade de um objeto pode ser de um tipo diferente em outro, ou seja, não possuem nenhuma composição definida. Geralmente são salvos em formato JSON (*Javascript Object Notation*). Exemplos: Cloud Firestore, MongoDB, CouchDb, RavenDb, etc (SADALAGE,2014).

**Banco de dados de famílias de colunas:** diferentes dos bancos de dados tradicionais que salvam cada registro em uma linha, estes bancos armazenam registros por colunas. Esta

abordagem não é boa para leituras ou escritas de porções pequenas na base, mas é ótima quando é preciso escrever ou ler grande parte da base de uma vez. Por este motivo este modelo é bom para aplicações analíticas, pois além de uma leitura mais rápida os dados semelhantes são próximos (SADALAGE,2014). Exemplos: Hadoop, Cassandra, Hypertable, Amazon SimpleDb, etc.

### **3.1.1 Análise das metodologias de banco de dados não relacional**

Como qualquer parte da arquitetura de uma aplicação, é preciso avaliar qual é a melhor opção de tecnologia de banco de dados a ser utilizada. Aderindo a um banco de dados não relacional grande parte da responsabilidade da consistência dos dados fica a cargo da aplicação. O banco fica mais simples, escalável e rápido, mas geralmente perde uma ou outra das conhecidas garantias dos bancos relacionais tradicionais.

Após a análise das abordagens dos gêneros de bancos de dados não relacionais relatados chegou se ao resultado que a melhor abordagem para este trabalho é o orientado a documentos pois ganha flexibilidade, disponibilidade e contém uma linguagem de consulta simples.

## **3.2 FIREBASE**

O Firebase é uma plataforma móvel desenvolvida pela equipe da Google com diversas ferramentas integradas que servem para criar aplicativos de alta produtividade. Utiliza-se um banco de dados não relacional, ou seja, não utiliza SQL e é orientado a documentos, quer dizer que não possui como padrão o sistema de tabelas e relacionamentos entre dados, sobretudo tratando cada informação como uma árvore, com um tronco principal e várias raízes como seus nós, onde cada raiz pode ter outras sub-raízes como demonstra a Figura 04. (PEDRO, 2018).



Figura 04 - NoSQL



Fonte: Baseada na imagem original de (PEDRO, 2018)

Existem vantagens por utilizar um banco de dados NoSQL no lugar do SQL comum, uma delas é o fácil desenvolvimento, utilização e ordenação. Ele é bem simples, com uma breve leitura da documentação todo o projeto estará arrematado para ser realizado, convertendo em um local impecável para propostas em fases introdutórias de validação.

A utilização e ordenação são bem simples de assimilar visto que emprega o padrão JSON para gravar os dados. Outro grande benefício é a agilidade em queries e updates, oposto aos bancos de dados comuns, a aplicação do acesso e nós ao contrário de tabelas amplia a rapidez de resposta da *query* pois ela está sendo realizada pelo próprio cliente.

No contexto, o Firebase não fará uma query em diferentes nós e não vai retornar os dados simplificados, por este motivo é preciso determinar isso tudo manualmente. Assim, fazendo com que seja irrelevante qualquer conversação direta entre nós para criar ligação entre dados pois, basicamente criando essa comunicação no código.

Por outro lado é justamente a ausência de usabilidade desse método pois tudo tem de se a ser feito pelo seu código, logo cabe avaliar os prós e os contras de projeto a projeto (PEDRO, 2018).

Melhor performance para grandes volumes de dados no formato NoSQL são os ideais para um número abundante de dados pois a não presença de comunicação entre os nós diminui o tempo entre uma busca e a inserção, um grande exemplo de onde é utilizado é o BigData, bancos de dados comuns não foram preparado para este tipo de prática.

Uma das principais vantagens é a flexibilidade da estrutura, empregando o JSON possibilita que seus objetos não fiquem restritos a colunas e linhas. Um objeto de um nó pode ter inúmeros outros sub-nós.

As ferramentas integradas podem ser divididas em dois grupos, o primeiro grupo é denominado de desenvolvimento e o segundo grupo é denominado qualidade conforme a Quadro 02.









**Quadro 02** - Ferramentas integradas

<b>Desenvolvimento</b>	<b>Qualidade</b>
Realtime Database	Firebase Analytics
Auth	Invites
Test Lab	Cloud Messaging
Crashlytics	Predictions
Cloud Functions	AdMob
Firestore	Dynamic Links
Cloud Storage	Adwords
Performance Monitoring	Remote Config
Crash Reporting	App Indexing
Hosting	

Fonte: Adaptado de (ADRIANO, 2018)

Ela possibilita um sistema de registro simples e ágil, essa ferramenta possui aplicações integradas através de redes sociais, logins anônimos ou por usuário e senha criado na própria aplicação, a viabilidade de utilizar o sistema de recuperação de senhas interno do Firebase com envio de e-mails como demonstra a Figura 05.

**Figura 05** - Provedores de login

Provedor	Status
 Email/Senha	Ativado
 Smartphone	Desativado
 Google	Desativado
 Play Games	Desativado
 Game Center	Desativado
 Facebook	Desativado
 Twitter	Desativado
 GitHub	Desativado

Fonte: Adaptado de (PEDRO, 2018)

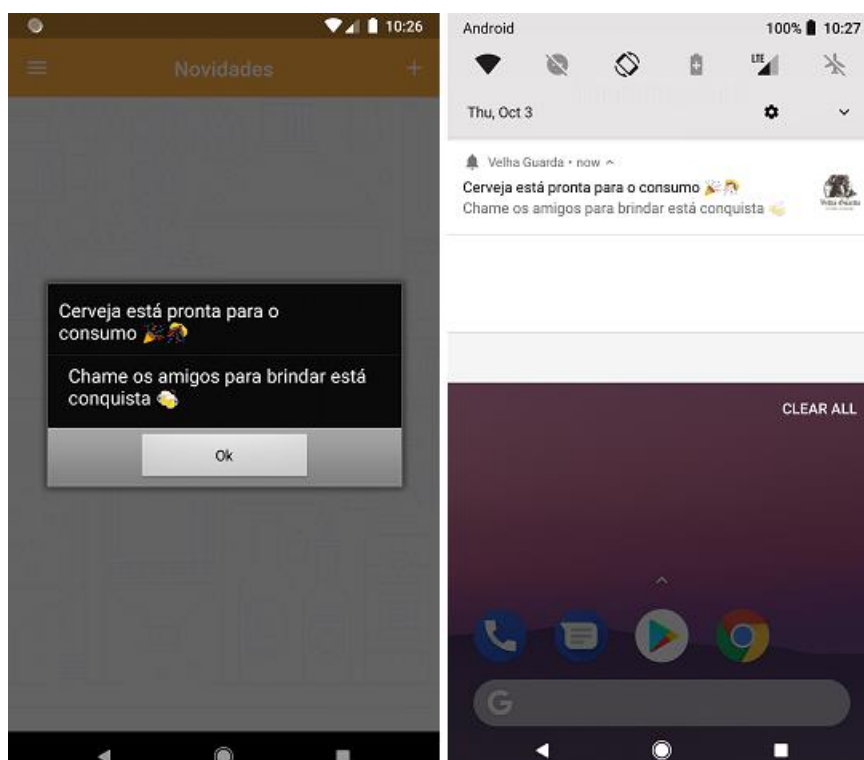
A principal ideia de utilizar o Firebase no projeto é por ser uma plataforma móvel completa e robusta de fácil uso, com a viabilidade de trabalhar *offline* e sincronizar automaticamente ao retornar a conexão, utiliza sincronização em tempo de execução, tem plano gratuito e atende muitos cenários utilizando ferramentas que vão economizar tempo e ajudar a centralizar os recursos do sistema.

### 3.2.1 OneSignal

Segundo (Savio, 2019) o OneSignal é uma plataforma com plano gratuito que se comunica com as ferramentas do Firebase e serve para impulsionar push móvel, web push, e-mail e mensagens no aplicativo. O SDK facilita a integração dos aplicativos no Flutter.

As notificações por push são o principal fator para alertar alguma mudança brusca de temperatura nos processos de produção da cerveja artesanal ou mudança de processo como demonstra a Figura 06.

**Figura 06 - OneSignal - push notification**



Fonte: Baseada na imagem original de (MAGNABOSCO, 2019)

## 4 HARDWARE

Neste capítulo é feito um estudo sucinto sobre os *hardwares*, suas principais características, aplicações e descrições, a comunicação entre o hardware, software e o protocolo de comunicação entre eles.

### 4.1 SENSOR INTELIGENTE

Os sensores inteligentes têm demonstrado o seu papel fundamental no âmbito de automação industrial. Estes sensores são normalmente conectados a um dispositivo computacional utilizando tecnologias sem fio ou com fio com técnicas de comunicação diferentes, como por exemplo, os protocolos de rede.

Os sensores fazem o trabalho crítico dos processos de monitoramento, medições e coleta de dados. Eles são dispositivos usados para detectar e responder a sinais elétricos ou ópticos. Um sensor converte o parâmetro físico como temperatura, em um conjunto de sinais que podem ser medidos eletricamente.

#### **4.1.1 Sensor DS18B20**

O sensor DS18B20 é um termômetro digital à prova d'água fabricado pela Dallas Instruments como demonstra a Figura 07.

**Figura 07** - Sensor DS18B20



**Fonte:** Baseada na imagem original de (ROVAI, 2019)

Este dispositivo auxilia a monitorar o grau da temperatura de um estipulado processo com a finalidade de gerar de forma correta e nas condições propícias para o funcionamento apropriado do processo.

A principal motivação para usar este dispositivo no projeto é baseada no argumento de (MADEIRA, 2018), fundamenta que, o sensor é apto a identificar a situação, interpretá-la

e encaminhar os dados em graus Celsius para o microcontrolador passando por um barramento de apenas um fio utilizando o protocolo de conversação One-Wire.

Este sensor está isolado e é capaz de calcular temperaturas entre -55 °C e 125 °C uma vez que exatidão pode variar 0,5 °C. Existe uma ótima compatibilidade com o módulo ESP8266, pois utilizam unicamente um pino digital, tendo potencial de conectar múltiplos sensores no mesmo pino (ROVAI, 2019).

Uma grande vantagem em utilizar este sensor é o alarme programável, sendo possível disparar um sinal informando sobre situações de alerta, como mudanças bruscas de temperatura. Essa operação é armazenada em uma memória não volátil.

## **4.2 Protocolo One-Wire**

O protocolo de conversação aplicado para realizar com que os valores resultantes do sensor de temperatura DS18B20 se comunique com o módulo ESP8266 é o protocolo One-Wire.

Este protocolo consiste na comunicação pelo meio de um barramento, ou seja, um caminho único para enviar seus dados, na qual, podem ser ligado em diversos dispositivos, de maneira que estes sejam capazes trocar dados com o módulo.

## **4.3 MÓDULO WIFI ESP8266 NODEMCU V3 CP2102 ESP-12E**

Madeira (2018) explica que o módulo ESP8266 NodeMCU é uma placa de desenvolvimento com uma estrutura que pode ser aplicada na geração de uma imensidade de iniciativas de automação diferentes. O que torna este módulo tão notável é a habilidade de conectar-se a uma rede WiFi.

### **4.3.1 ESP8266**

O ESP8266 é um microcontrolador com conexão a Internet. Ou seja, ele é similar a um arduino com integração Wi-Fi como demonstra a Figura 08.

**Figura 08 - ESP-12**



Fonte: Baseada na imagem original de (MORAIS, 2017)

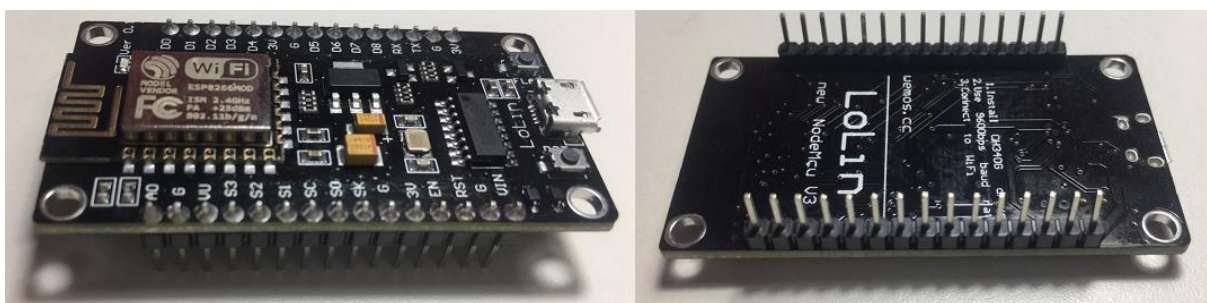
Segundo o argumento de (MORAIS, 2017), esta placa minúscula que também é chamada de módulo tem um potencial muito maior do que aparenta comparando diretamente com o arduino.

#### **4.3.2 NodeMCU V3**

Esta versão do ESP8266-12 inclui uma placa de desenvolvimento com uma estrutura que pode ser utilizada na criação de diversos projetos de automação no ambiente de desenvolvimento integrado Arduino.

Este módulo contém um conversor serial mais um regulador de tensão 3.3V próprio, não sendo necessário ambos como nas outras versões anteriores. Também há pinos próprios para I2C, SPI, Analógico e outros, conforme apresenta a Figura 09.

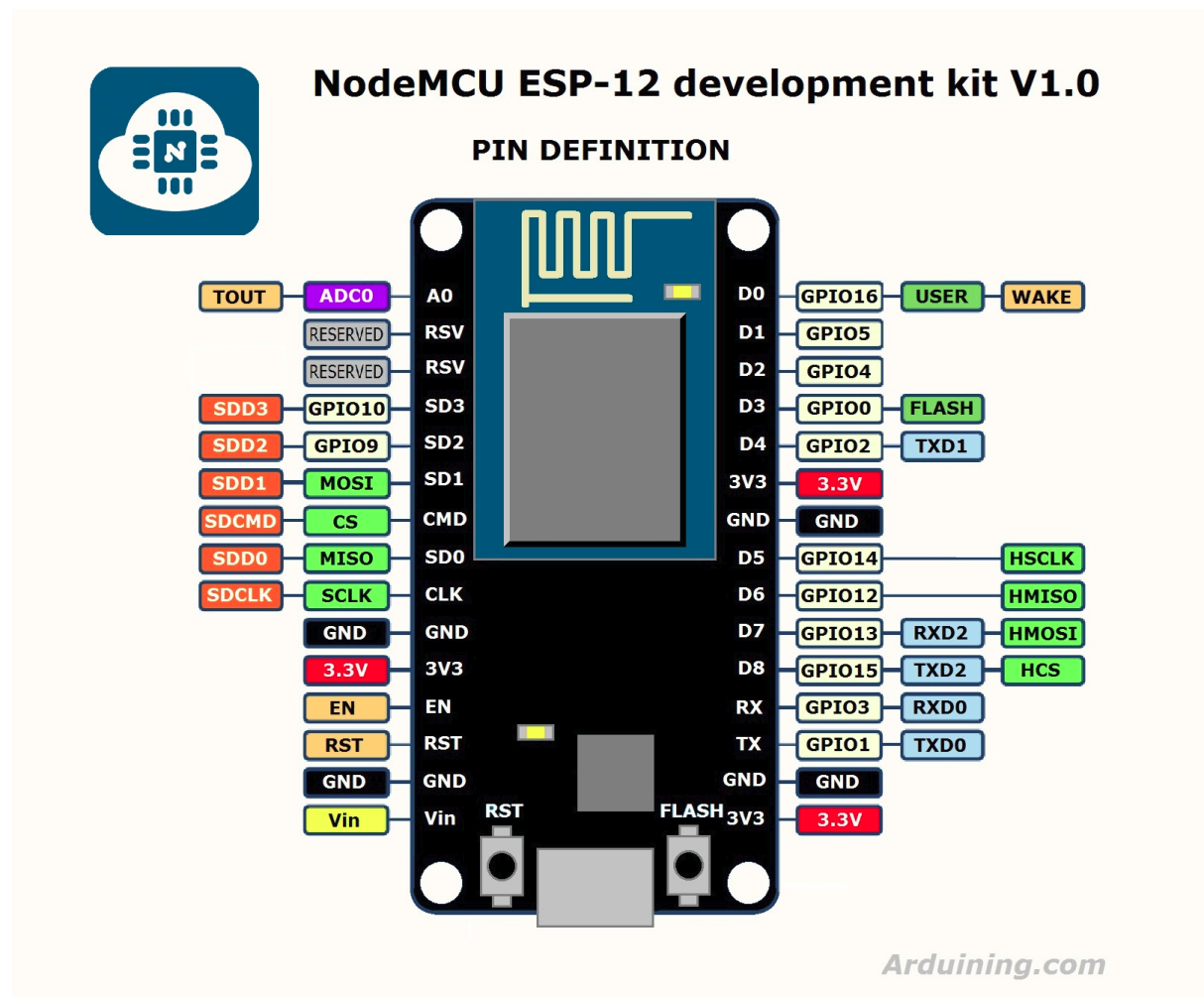
**Figura 09 - Módulo ESP8266 NodeMCU V3**



Fonte: O Próprio Autor (2020)

Na Figura 10 abaixo, podemos ver a pinagem do NodeMcu, e podemos reparar que ele trabalha com 3.3V

**Figura 10** - Módulo ESP8266 Pinagem



**Fonte:** Baseada na imagem original de (THOMSEN, 2016)

### 4.3.3 BreadBoard

Esta pequena placa retangular que possui muitos orifícios, serve para alojar componentes eletrônicos nela. Os furos na placa são interconectados de maneira a facilitar a conexão dos componentes como demonstra a Figura 11.

**Figura 11** -BreadBoard



**Fonte:** O próprio Autor (2020)



## 5 CONFIGURAÇÕES

A descrição da configuração do sistema, juntamente com a montagem dos componentes e instalações de softwares é abordado neste capítulo.

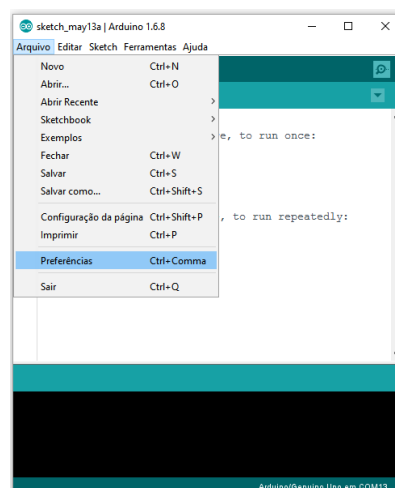
### 5.1 ARDUINO

Segundo o argumento de (TELES, 2016) o arduino é uma IDE de prototipagem eletrônica *open-source* que se embasa em hardware e software moldável e simples de utilizar. É destinado a artistas, designers, programadores ou qualquer pessoa interessada em criar objetos ou ambientes interativos.

#### 5.1.1 Placa

Primeiro passo é configurar a IDE do arduino, selecionando no menu superior Arquivo depois Preferências conforme apresenta a Figura 12.

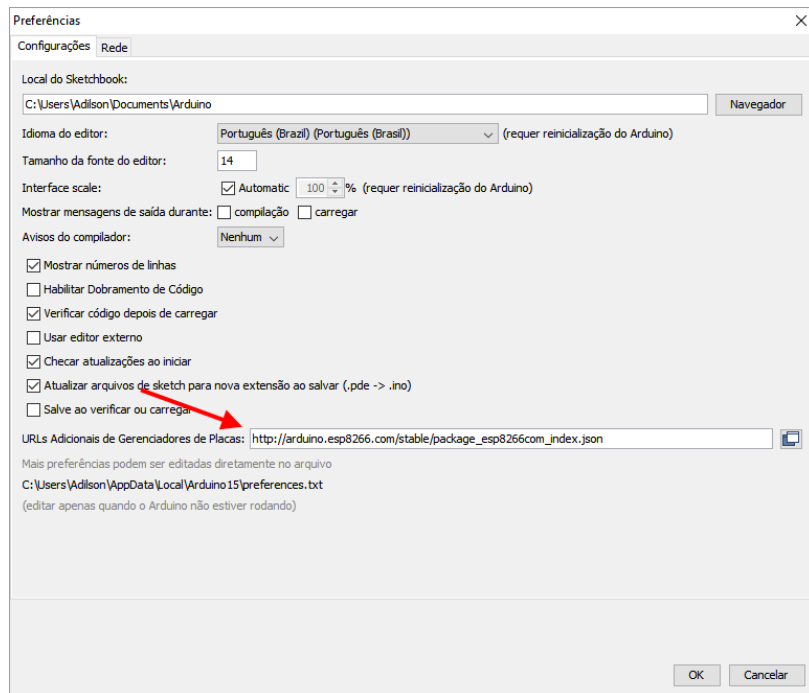
**Figura 12 -** Arduino Preferências



**Fonte:** O Próprio Autor (2020)

Segundo passo procure pelo campo URLs adicionais de Gerenciadores de Placas e adicione o link (ARDUINO) Disponível em: [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) para adicionar as placas da família ESP conforme a Figura 13.

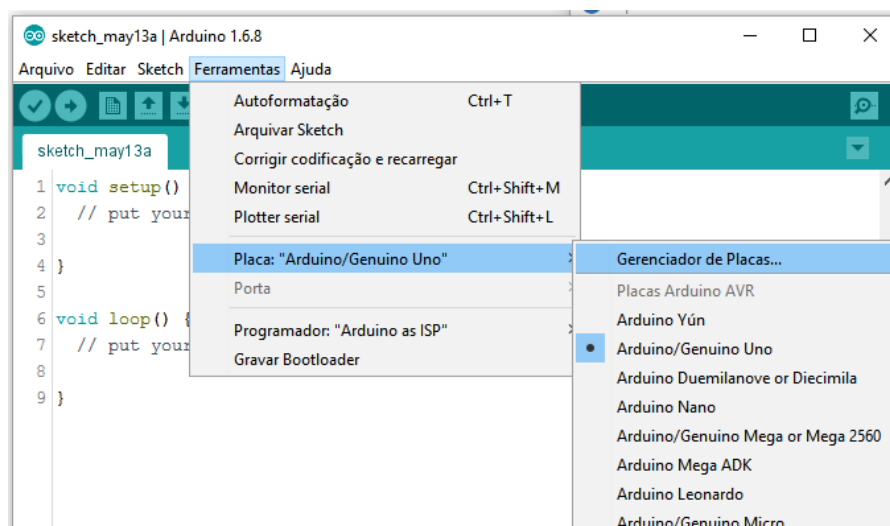
**Figura 13 - URL's adicionais de gerenciadores de placas**



**Fonte:** O Próprio Autor (2020)

Terceiro passo selecione no menu superior Ferramentas, Placa, Gerenciador de Placas conforme apresenta a Figura 14.

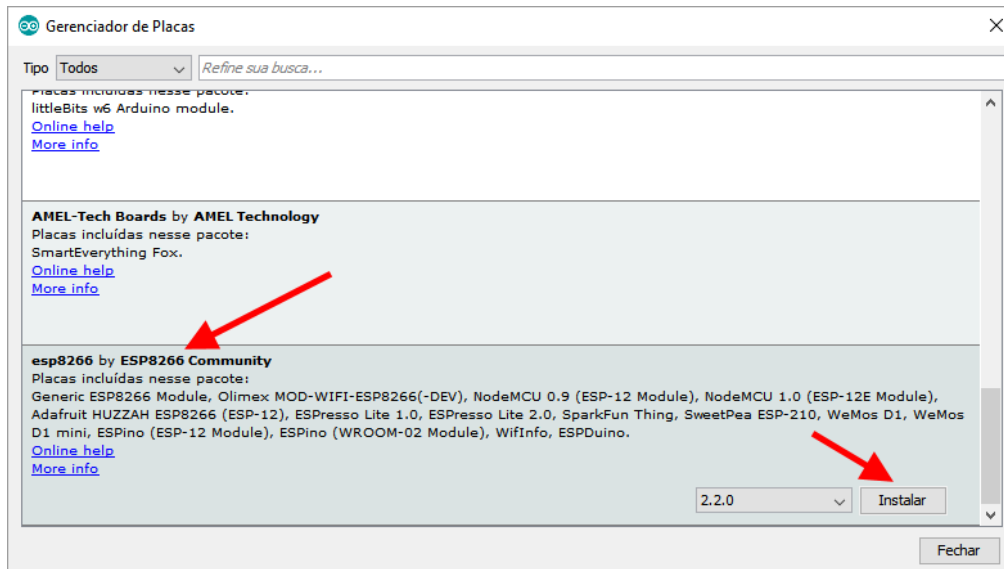
**Figura 14 - Gerenciador de placas**



**Fonte:** O Próprio Autor (2020)

Quarto passo role até o final até encontrar a opção “esp8266 by ESP8266 Community”, clique sobre ela, e instale a última versão do pacote como demonstra a Figura 15.

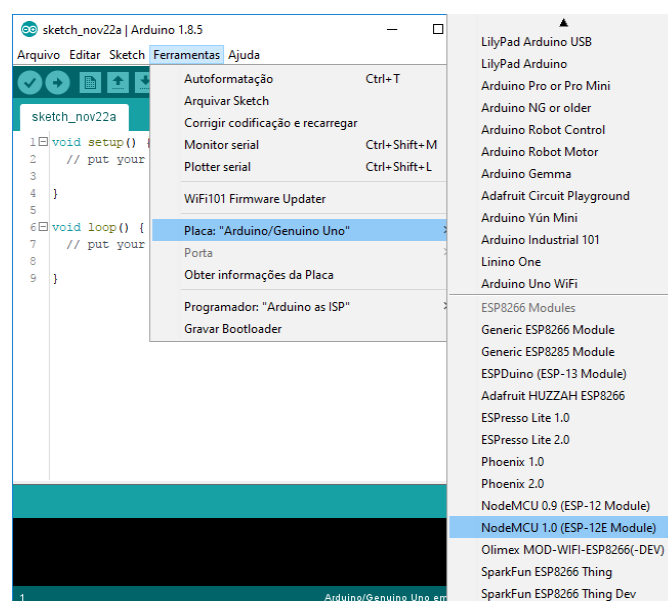
**Figura 15** - Instalação do pacote da placa



**Fonte:** O próprio Autor (2020)

Quinto passo as instalação ser concluída, as placas da linha ESP8266 já estarão disponíveis na lista de placas da sua Arduino IDE conforme apresenta a Figura 16.

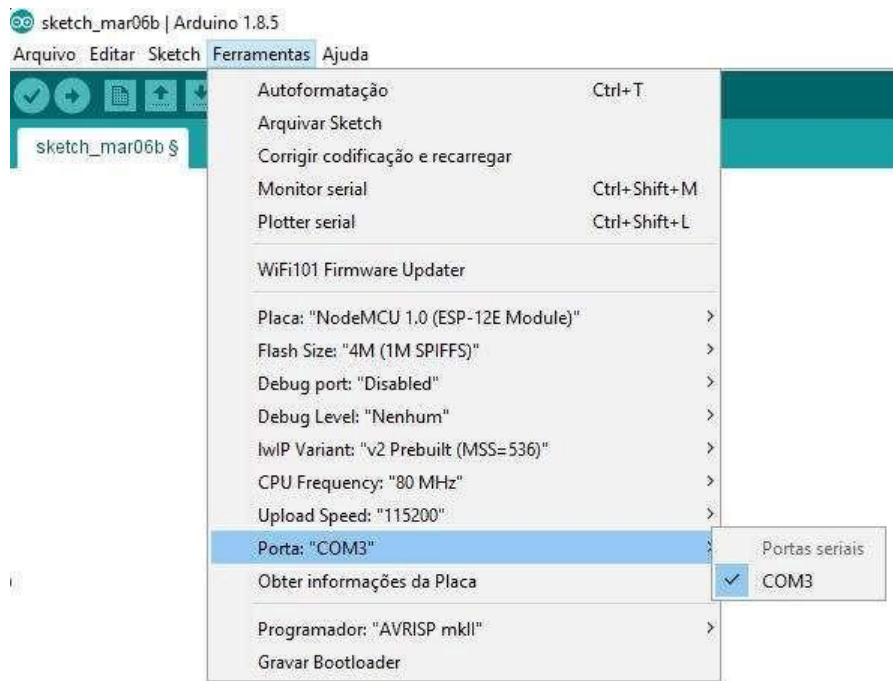
**Figura 16** - Escolha da placa



**Fonte:** O Próprio Autor (2020)

Por fim não altere os outros parâmetros da Placa NodeMCU. Somente altere a porta COM da interface serial-USB. No meu caso a porta é a COM3 como demonstra a Figura 17.

**Figura 17 - Porta COM**



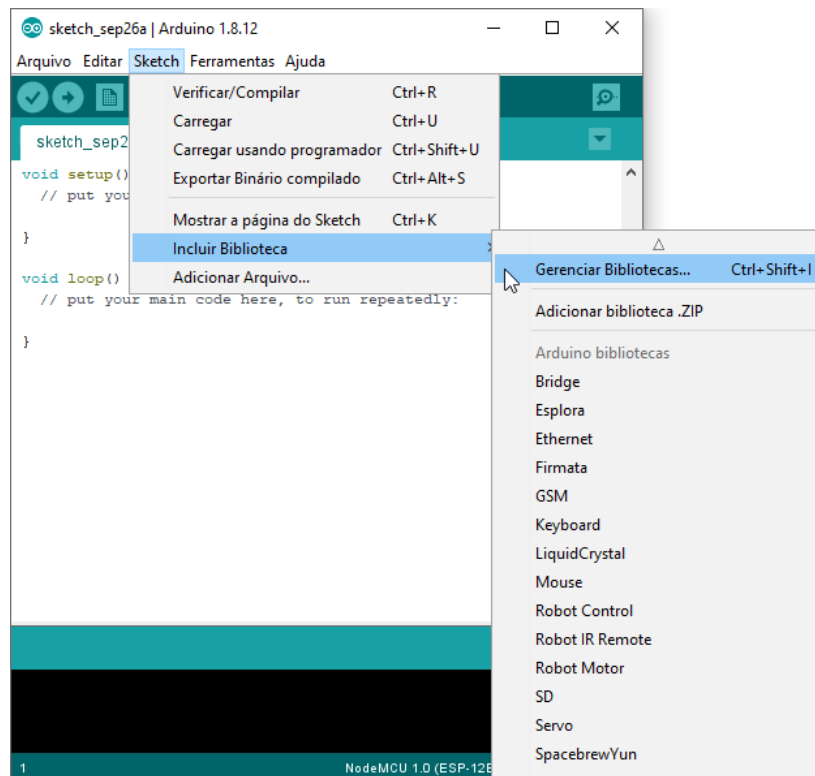
**Fonte:** O Próprio Autor (2020)

**IMPORTANTE:** Saiba que a velocidade padrão da Console da IDE Arduino para o ESP8266 é de 115200 Bps. Após tudo configurado com sucesso, feche e abra o programa Arduino IDE novamente, para que todas as configurações sejam efetivadas.

### 5.1.2 Sensor

Após concluso primeiro passo, selecione no menu superior *Sketch*, Incluir Biblioteca, Gerenciar Bibliotecas conforme apresenta a Figura 18.

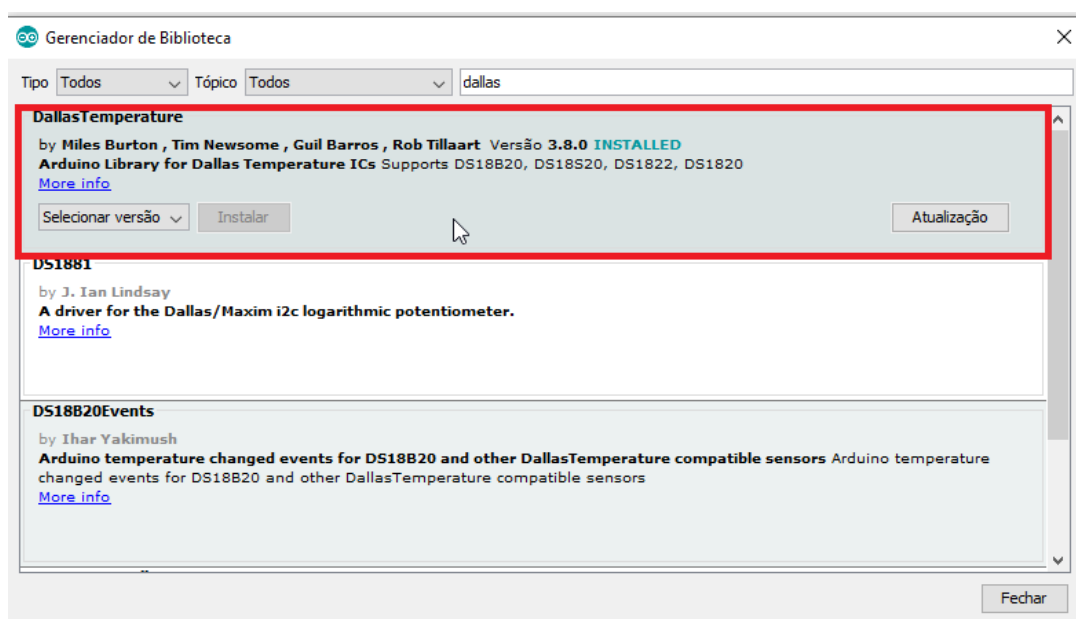
**Figura 18 - Incluir biblioteca**



**Fonte:** O próprio Autor (2020)

Após procure por dallas até encontrar a opção “DallasTemperature”, clique sobre ela, e instale a última versão do pacote como demonstra a Figura 19.

**Figura 19 - Dallas temperature**

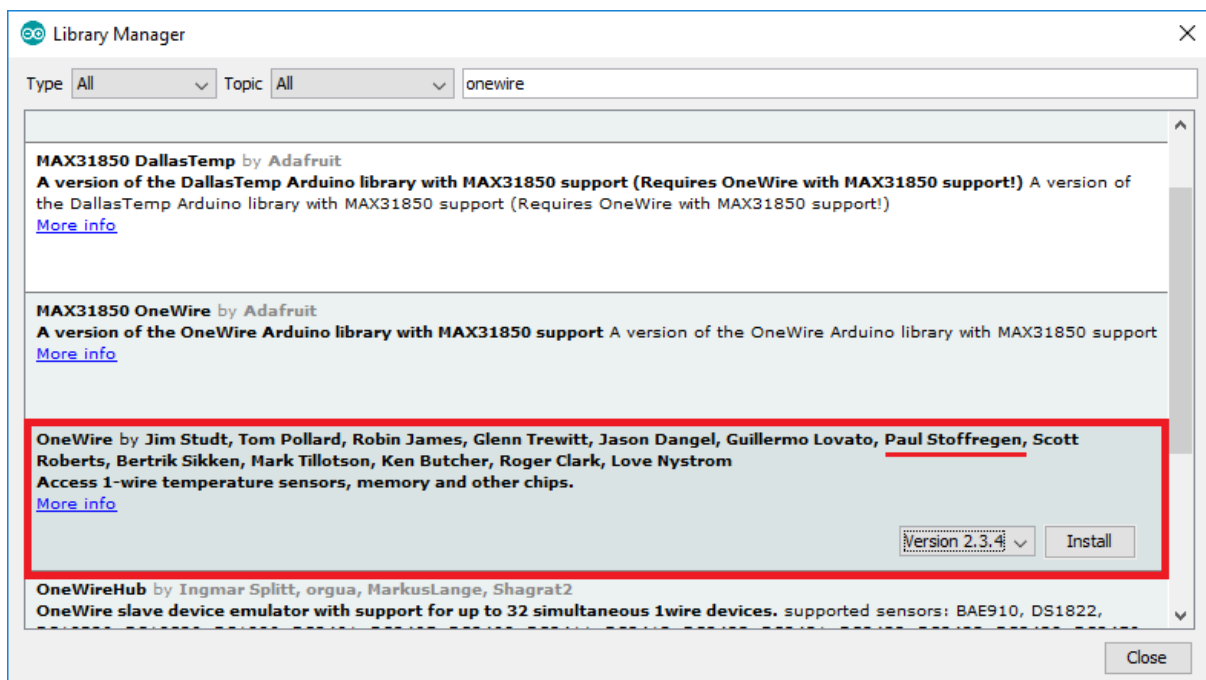


**Fonte:** O Próprio Autor (2020)

### 5.1.3 OneWire

Busque pela biblioteca do OneWire, que é responsável pela comunicação entre o módulo ESP8266 e o sensor DS18B20 demonstra a Figura 20.

**Figura 20 - OneWire**



**Fonte:** Próprio Autor (2020)

## 5.2 ANDROID STUDIO

Baixe o Android Studio no (DEVOLPER) Android Studio, disponível em: <https://developer.android.com/studio/>, após baixar o arquivo, execute-o para iniciar o processo de instalação da IDE conforme apresenta a Figura 21.

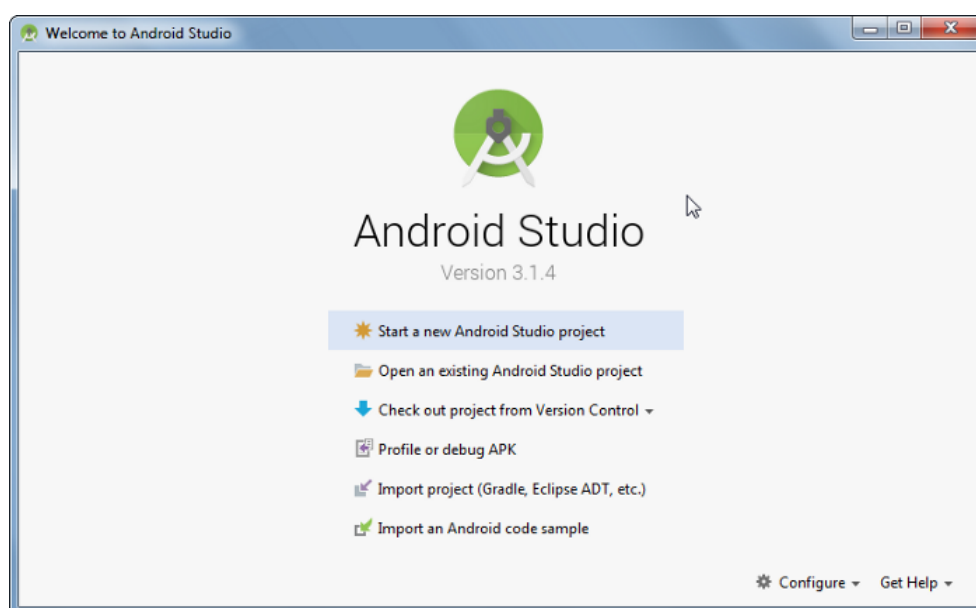
**Figura 21** - Instalação Android Studio



**Fonte:** Baseada na imagem original de (MUNIZ, 2018)

Após o término da instalação, o Android Studio será iniciado e a tela inicial será apresentada como demonstra a Figura 22.

**Figura 22** - Tela inicial Android Studio

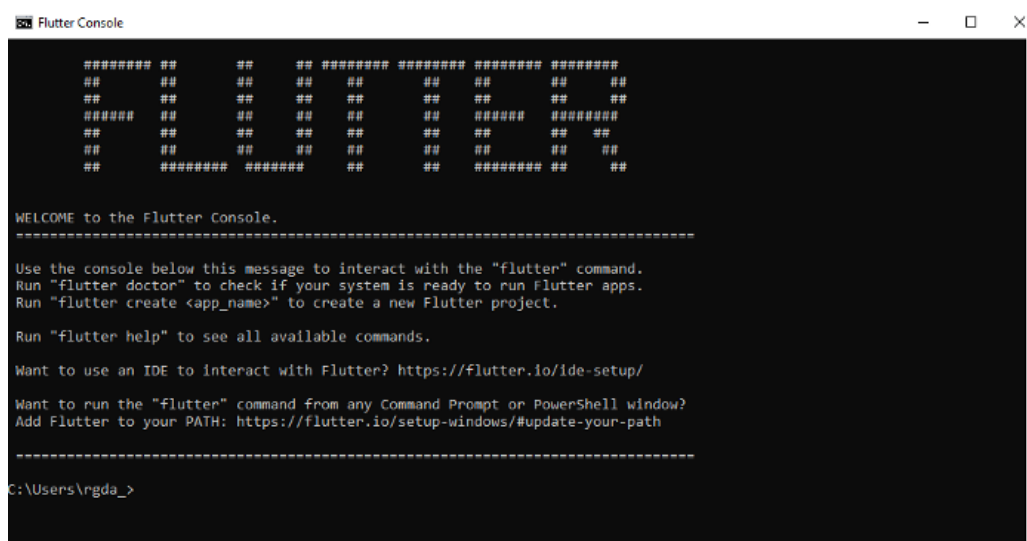


**Fonte:** Baseada na imagem original de (MUNIZ, 2018)

### 5.2.1 Flutter SDK

Baixe a SDK do Flutter (FLUTTER) *Windows install*, disponível em: <https://flutter.dev/docs/get-started/install/windows>, extraia o arquivo no local `c:\src\flutter`. Em seguida, já sendo capaz de acessar todos os comandos do console do Flutter como demonstra a Figura 23 (SESSA, 2020).

**Figura 23** - Flutter console



```
Flutter Console

#####  ##      ##  #####  #####  #####  #####
##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##
#####  ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##
##      ##      ##      ##      ##      ##      ##
##      #####  #####  ##      ##      #####  ##

WELCOME to the Flutter Console.
=====

Use the console below this message to interact with the "flutter" command.
Run "flutter doctor" to check if your system is ready to run Flutter apps.
Run "flutter create <app_name>" to create a new Flutter project.

Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.io/ide-setup/

Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.io/setup-windows/#update-your-path
=====

C:\Users\rngda_>
```

**Fonte:** Baseada na imagem original de (DIAS, 2019)

Porém, para maior proveito, é possível ter acesso ao console do Flutter à partir de qualquer terminal ou prompt de comando, acrescentando o Flutter à variável de ambiente do *path* do seu sistema (DIAS, 2019).

Para isto precisamos seguir um plano, o primeiro passo é copiar o caminho até o diretório `\bin`, presente na pasta do Flutter.

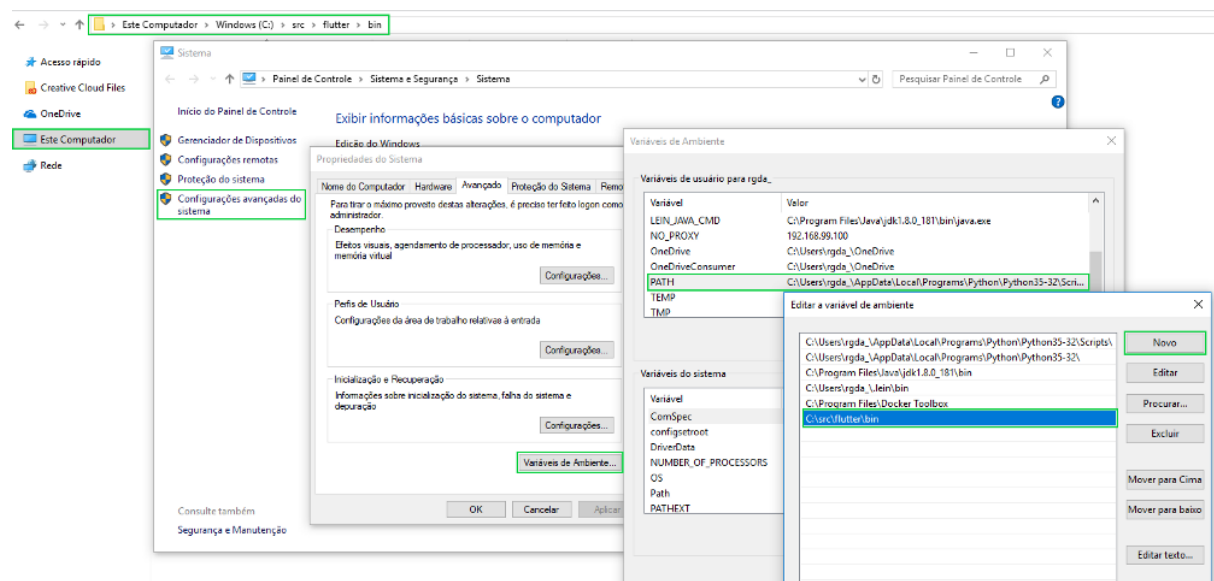
O segundo passo é acessar o *Windows Explorer*, depois clicar com o botão direito em Este computador e acessar as Propriedades.

Próximo passo vá em Configurações avançadas do sistema clique em Variáveis de ambiente. No *textfield* Variáveis de usuário, clique na variável *PATH* depois em Novo, e cole o caminho para o diretório `\bin`.

Neste momento o Flutter já pode ser acessado diretamente do prompt de comando ou outro terminal conforme apresenta a Figura 24.



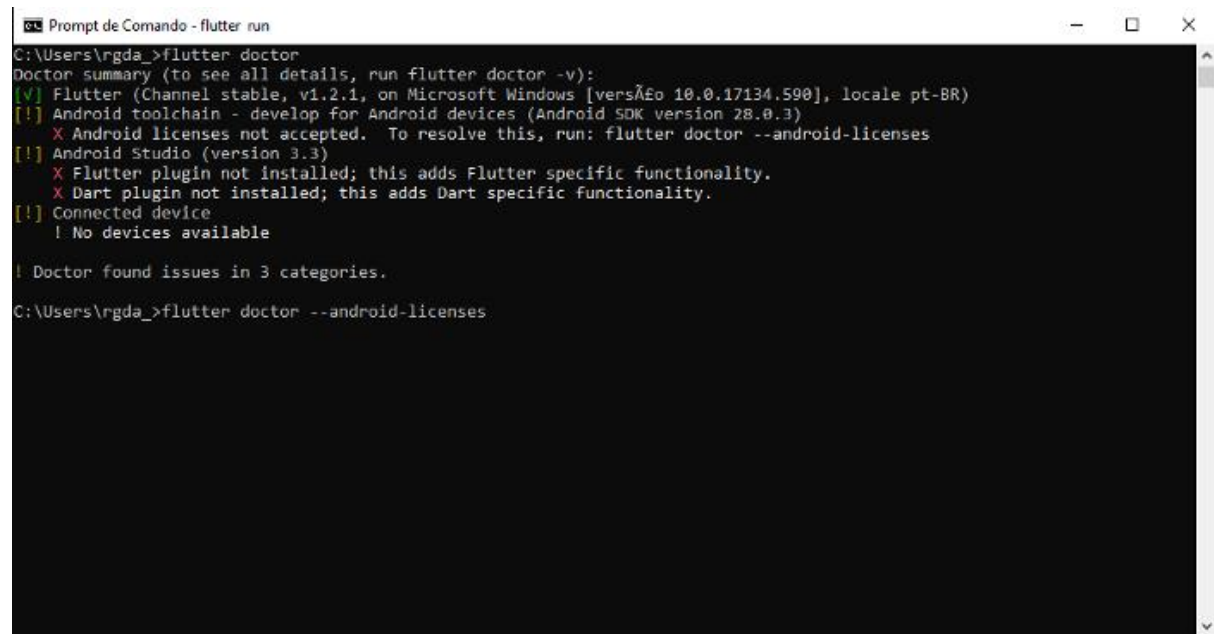
**Figura 24 - Configurando a variável de ambiente**



**Fonte:** Baseada na imagem original de (DIAS, 2019)

Para verificar se tudo ocorreu corretamente, iremos rodar um comando no terminal chamado *Flutter doctor* conforme apresenta a Figura 25.

**Figura 25 - Flutter Doctor**

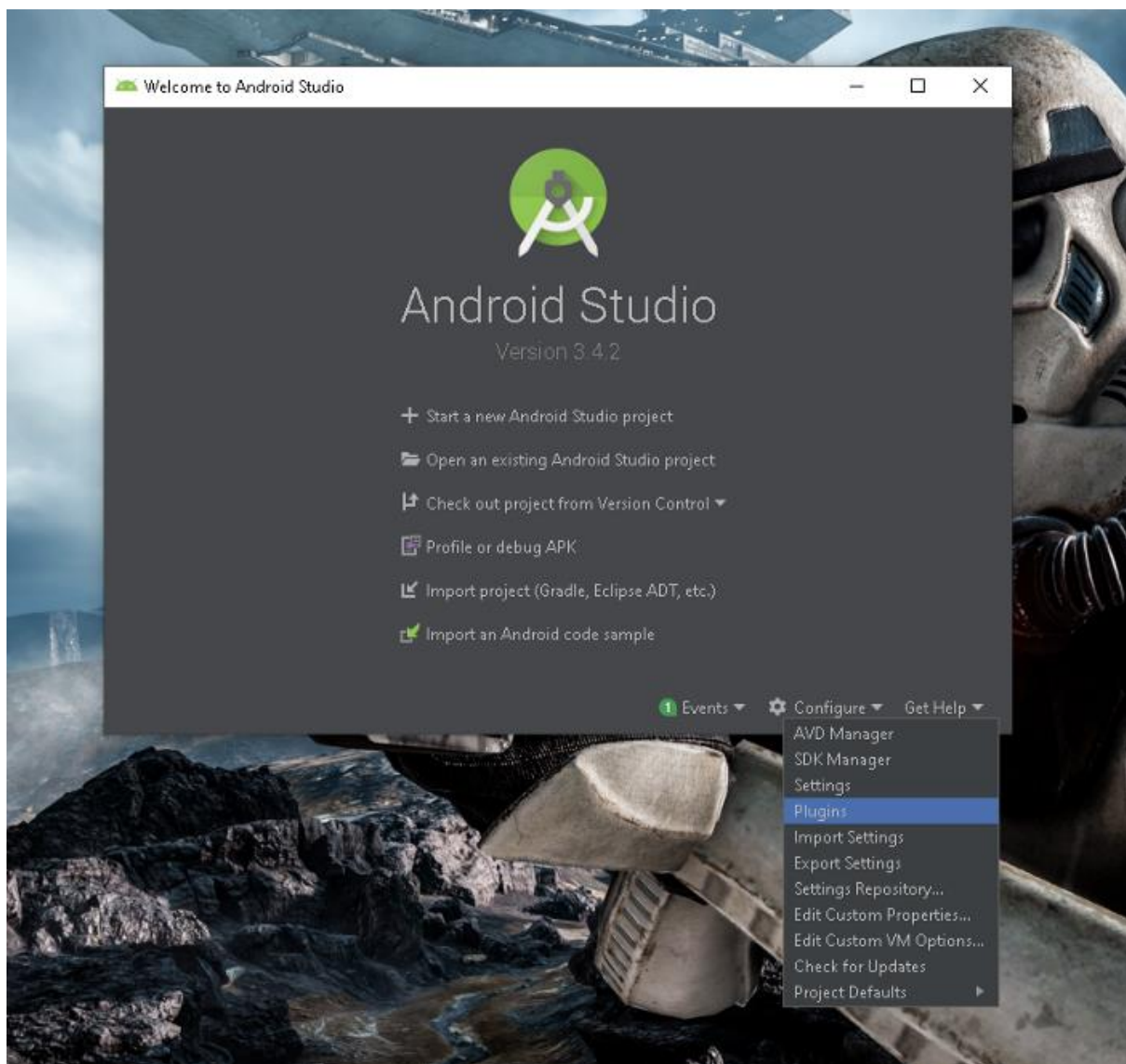


**Fonte:** Baseada na imagem original de (DIAS, 2019)

O Flutter doctor é encarregado por apurar se existem dependências dele mesmo a serem instaladas. Inclusive ele retorna um relatório sobre a condição da instalação mostrando as dependências que faltam, os problemas encontrados e a maneira de resolvê-los.

Após concluídas todas essas etapas deve-se acrescentar alguns *plugins* e extensões no Android Studio para que este seja apto a desenvolver em Flutter como demonstra a Figura 26.

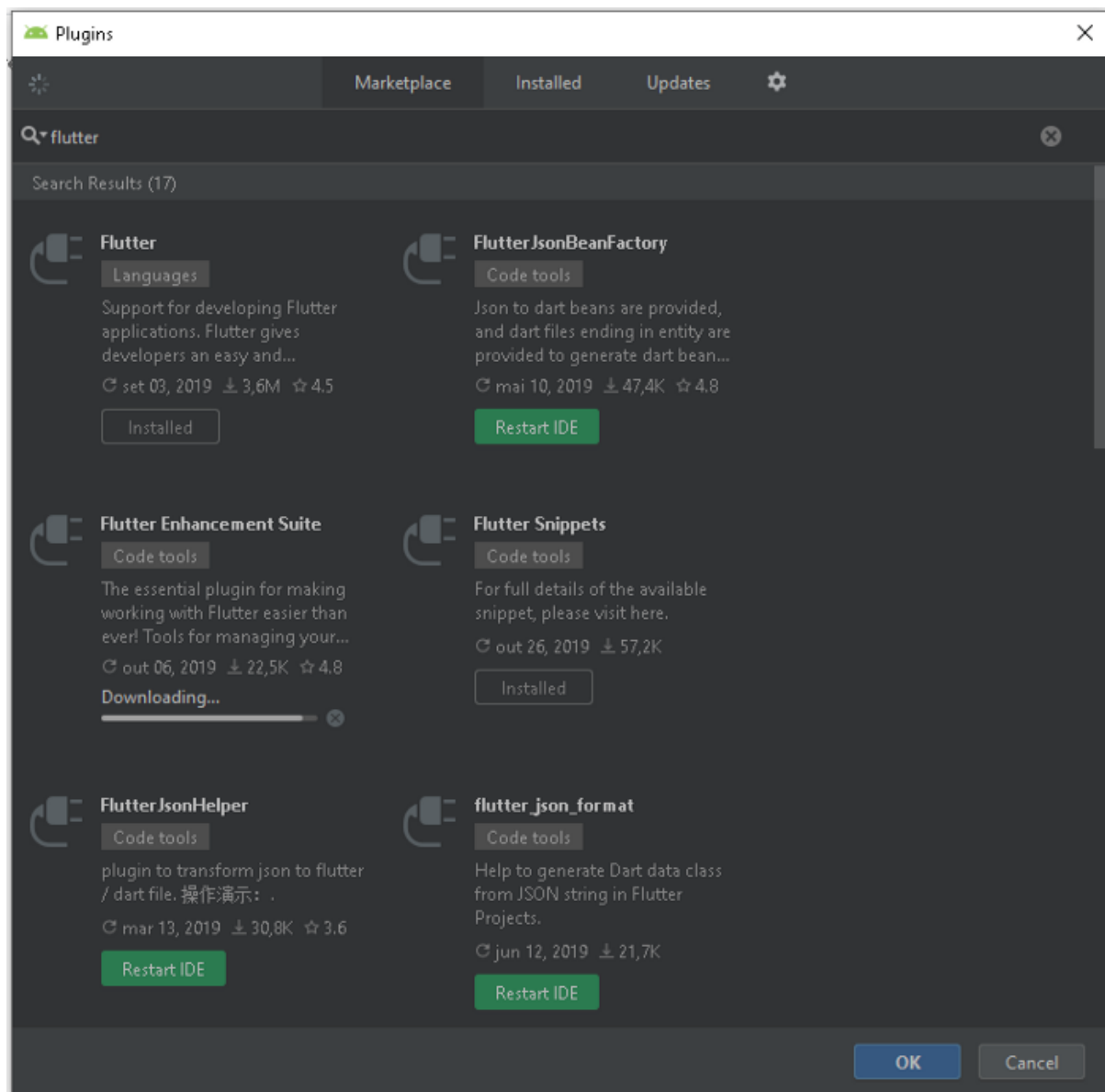
**Figura 26** - Android Studio Plugins



**Fonte:** Baseada na imagem original de (SESSA, 2020)

Em seguida deve-se instalar os plugins do Dart e Flutter para que tudo funcione da forma correta como demonstra a Figura 27.

**Figura 27 - Plugins Dart e Flutter**

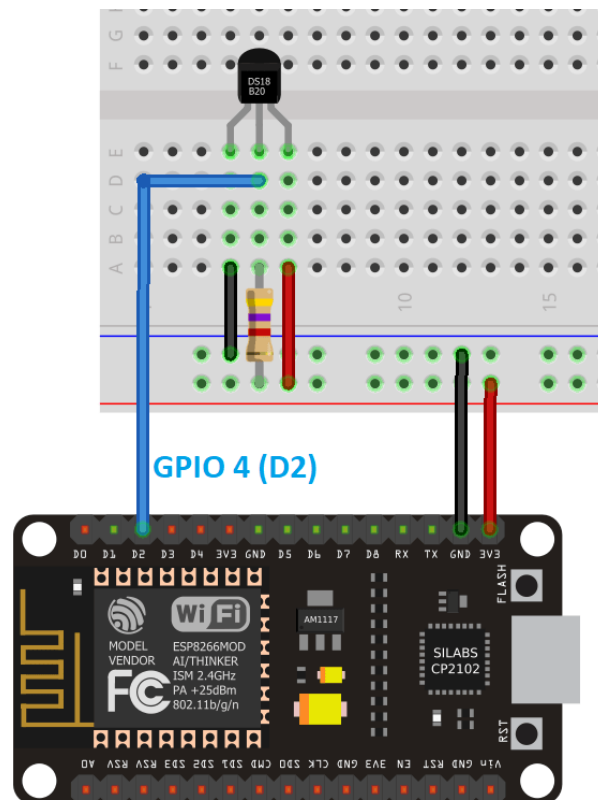


**Fonte:** Baseada na imagem original de (SESSA, 2020)

### 5.3 MONTANDO OS COMPONENTES

Primeiramente precisa-se efetuar as ligações através do módulo Nodemcu V3 Esp8266 ESP-12E e o sensor de temperatura DS18B20. Este sensor dispõe de 3 fios, de modo que, o fio preto deve ser unido ao pino GND do módulo, o vermelho deve ser ligado ao terminal 3V3 do módulo. Enfim, o último fio é o azul que deve ser conectado a um dos pinos digitais, neste caso, foi D2 conforme apresenta a Figura 28.

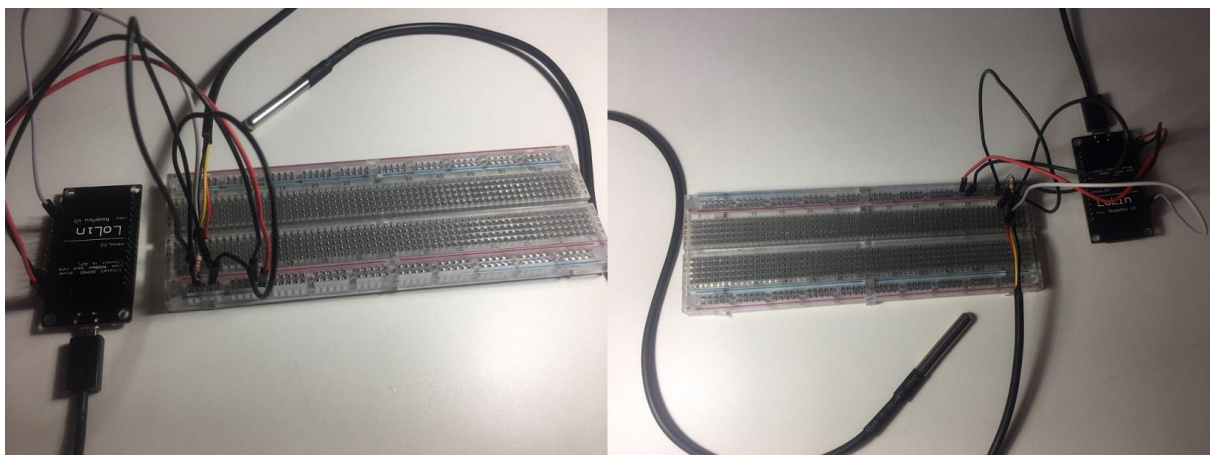
**Figura 28 - Montagem dos componentes**



**Fonte:** Baseada na imagem original de (MADEIRA, 2019)

Lembrando de um detalhe muito relevante que é a necessidade de um resistor de 4,7K Ohms entre o pino de alimentação 3V3 e o de dados D2 conforme apresenta a Figura 29.

**Figura 29 - Estrutura montada**



**Fonte:** (MAGNABOSCO, 2020)

Após instalar as bibliotecas necessárias e montar a estrutura, desenvolveu o seguinte código para o ESP8266 como demonstra a figura 30.

**Figura 30 - Código Arduino**

```
1  #include <ESP8266WebServer.h>
2  #include <OneWire.h>
3  #include <DallasTemperature.h>
4  #include <FirebaseArduino.h>
5
6  const int oneWireBus = 4;
7  OneWire oneWire(oneWireBus);
8  DallasTemperature sensors(&oneWire);
9  #define FIREBASE_HOST "https://velha-guarda.firebaseio.com/"
10 #define FIREBASE_AUTH "SB4ahzKIDOhbPgTtKp7PMQGH10r11"
11
12 float tempSensor1;
13
14 uint8_t sensor1[8] = { 0x28, 0xEE, 0xD5, 0x64, 0x1A, 0x16, 0x02, 0xEC };
15 /*Put your SSID & Password*/
16 const char* ssid = "Nome da rede Wifi"; // Enter SSID here
17 const char* password = "Senha da rede WIFI"; //Enter Password here
18
19 ESP8266WebServer server(80);
20
21 void setup() {
22     // Start the Serial Monitor
23     Serial.begin(115200);
24     // Start the DS18B20 sensor
25     sensors.begin();
26
27     Serial.println("Connecting to ");
28     Serial.println(ssid);
29
30     //connect to your local wi-fi network
31     WiFi.begin(ssid, password);
32
33     //check wi-fi is connected to wi-fi network
34     while (WiFi.status() != WL_CONNECTED) {
35         delay(1000);
36         Serial.print(".");
37     }
38     Serial.println("");
39     Serial.println("WiFi connected..!");
40     Serial.print("Got IP: "); Serial.println(WiFi.localIP());
41
42     server.on("/", handle_OnConnect);
43     server.onNotFound(handle_NotFound);
44
45     Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
46     server.begin();
47     Serial.println("HTTP server started");
48 }
49
50 void loop() {
51     server.handleClient();
52     sensors.requestTemperatures();
53     float temperatureC = sensors.getTempCByIndex(0);
54     Serial.print(Firebase.getString("situation") + "\n");
55     analogWrite(temperatureC, Firebase.getString("situation").toInt());
56     Serial.print(temperatureC);
57     Serial.println("°C");
58     delay(5000);
59 }
60
```



```

61 void handle_OnConnect() {
62     sensors.requestTemperatures();
63     tempSensor1 = sensors.getTempCByIndex(0); // Gets the values of the
        temperature
64     server.send(200, "text/html", SendHTML(tempSensor1));
65 }
66
67 void handle_NotFound(){
68     server.send(404, "text/plain", "Not found");
69 }
70
71 String SendHTML(float tempSensor1){
72     String ptr = "<!DOCTYPE html>";
73     ptr += "<html>";
74     ptr += "<head>";
75     ptr += "<title>Monitor de temperatura</title>";
76     ptr += "<meta charset='UTF-8'/>";
77     ptr += "<meta name='viewport' content='width=device-width, initial-scale
        =1.0'>";
78     ptr += "<link href='https://fonts.googleapis.com/css?family=Open+Sans
        :300,400,600' rel='stylesheet'>";
79     ptr += "<link rel='icon' href='favicon.ico' type='image/x-icon'>";
80     ptr += "<style>";
81     ptr += "html { font-family: 'Open Sans', sans-serif; display: block;
        margin: 0px auto; text-align: center; color: #444444; }";
82     ptr += "body { margin-top: 50px; } ";
83     ptr += "h1 { margin: 50px auto 30px; } ";
84     ptr += ".side-by-side { display: table-cell; vertical-align: middle
        ; position: relative; }";
85     ptr += ".text { font-weight: 600; font-size: 19px; width: 200px; }";
86     ptr += ".temperature { font-weight: 300; font-size: 50px; padding-right:
        15px; }";
87     ptr += ".living-room .temperature { color: #F29C1F; }";
88     ptr += ".superscript { font-size: 17px; font-weight: 600; position: absolute
        ; right: -5px; top: 15px; }";
89     ptr += ".data { padding: 10px; }";
90     ptr += ".container { display: table; margin: 0 auto; }";
91     ptr += ".icon { width: 82px; }";
92     ptr += "</style>";
93     ptr += "</head>";
94     ptr += "<body>";
95     ptr += "<h1>Monitor de temperatura</h1>";
96     ptr += "<div class='container'>";
97     ptr += "<div class='data living-room'>";
98     ptr += "<div class='side-by-side icon'>";
99     ptr += "<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 297 297'>";
100    ptr += "<circle cx='148.5' cy='148.5' r='148.5' fill='#c63c22'/>";
101    ptr += "<path d='M296.703 139.216l-96.66-96.678-104.931 199.82 54.626 54
        .626C231.181 296.318 297 230.1 297 148.5c0-3.119-.108-6.212-.297-9
        .284z' fill='#9e231d'/>";
102    ptr += "<path d='M207.444 99.107c9.143 0 16.556-7.412 16.556-16.556s-7
        .412-16.556-16.556-16.556c-.233 0-.459.025-.689.035.446-1.863.689-3
        .805.689-5.805 0-13.715-11.118-24.833-24.833-24.833-8.456 0-15.919
        4.23-20.404 10.685-2.92-2.582-6.747-4.163-10.951-4.163a16.462 16
        .462 0 00-9.793 3.228c-4.538-5.926-11.683-9.751-19.724-9.751-10.476
        0-19.43 6.491-23.079 15.667a16.457 16.457 0 00-7.106-1.62C82.412 49
        .44 75 56.852 75 65.996s7.412 16.556 16.556 16.556l115.888 16.555z'
        fill='#fff'/>";

```

```

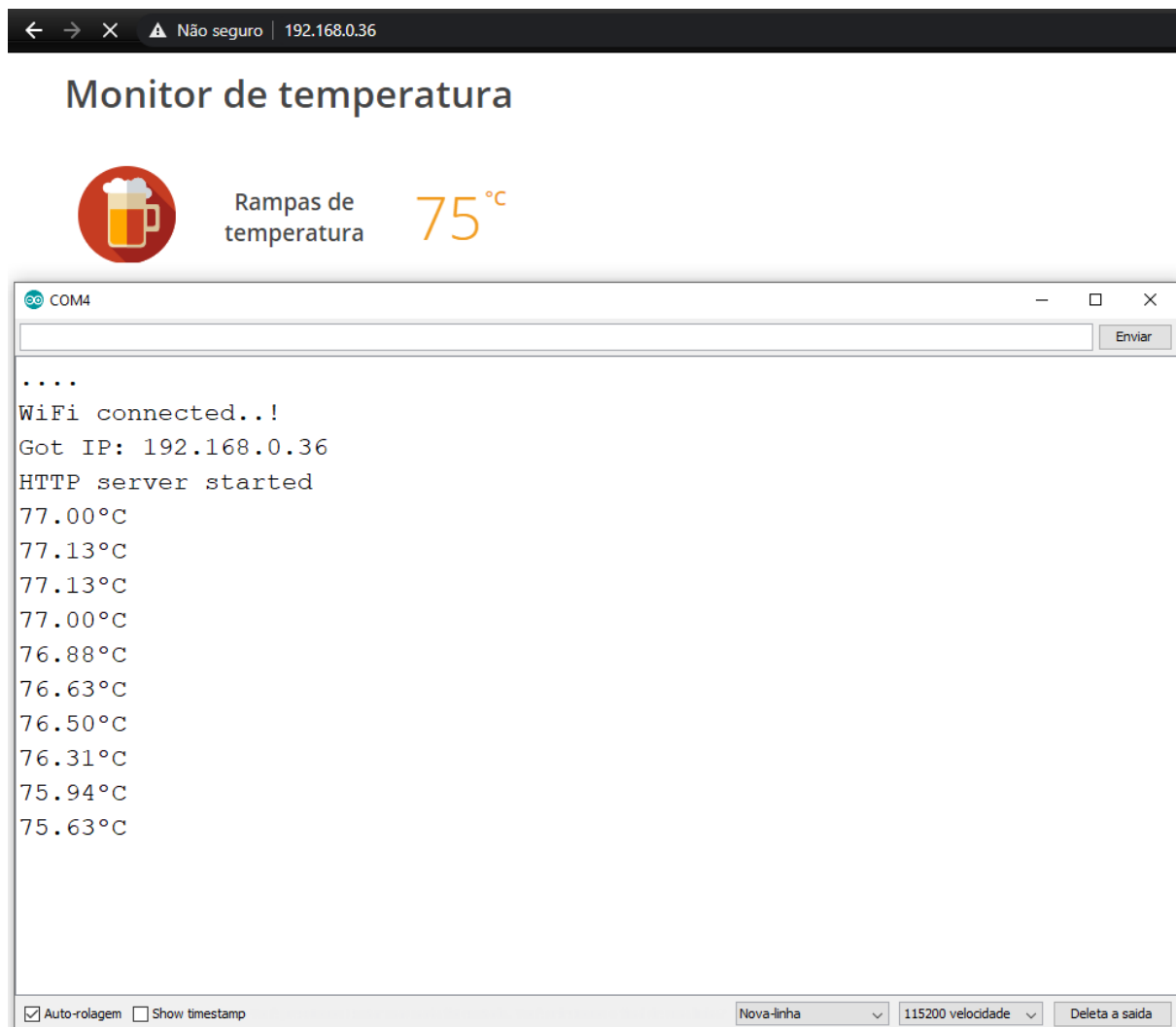
103 ptr += "<path d='M207.444 65.996c-.233 0-.459.025-.689.035.446-1.863.689
-3.805.689-5.804 0-13.715-11.118-24.833-24.833-24.833-8.455 0-15
.919 4.23-20.404 10.685-2.92-2.583-6.747-4.163-10.951-4.163-.882 0
-1.745.073-2.589.208V90.71l58.777 8.397c9.143 0 16.556-7.412 16.556
-16.556s-7.412-16.555-16.556-16.555z' fill='#d0d5d9'/><path d='M238
.282 115.5H206.25v-33H90.75v148.792c0 8.952 7.257 16.208 16.208 16
.208h83.083c8.952 0 16.208-7.257 16.208-16.208V198h32.032a9.218 9
.218 0 009.218-9.218v-64.064a9.216 9.216 0 00-9.217-9.218zm-2.429
61.532a6.718 6.718 0 01-6.718 6.718h-19.488v-54h19.488a6.718 6.718
0 016.718 6.718v40.564z' fill='#f0deb4'/>";
104 ptr += "<path d='M238.282 115.5H206.25v-33h-57.583v165h41.375c8.952 0 16
.208-7.257 16.208-16.208V198h32.032a9.218 9.218 0 009.218-9.218v-64
.064a9.218 9.218 0 00-9.218-9.218zm-2.429 61.532a6.718 6.718 0 01-6
.718 6.718h-19.488v-54h19.488a6.718 6.718 0 016.718 6.718v40.564z'
fill='#d8c49c'/><path d='M111.814 237.857h73.372c7.905 0 14.314-6
.409 14.314-14.314v-92.4h-102v92.4c0 7.906 6.409 14.314 14.314 14
.314z' fill='#ffa800'/>";
105 ptr += "<path d='M148.667 131.143v106.714h36.519c7.905 0 14.314-6.409 14
.314-14.314v-92.4h-50.833z' fill='#c63c22'/></svg>";
106 ptr += "</div>";
107 ptr += "<div class='side-by-side text'>Rampas de temperatura</div>";
108 ptr += "<div class='side-by-side temperature'>";
109 ptr += (int)tempSensor1;
110 ptr += "<meta http-equiv='refresh' content='2'>";
111 ptr += "<span class='superscript'>&deg;C</span></div>";
112 ptr += "</div>";
113 ptr += "</div>";
114 ptr += "</body>";
115 ptr += "</html>";
116 return ptr;
117 }

```

**Fonte:** O Próprio Autor (2020)

Este código monta um servidor web com um monitor serial onde o módulo servirá o servidor, enviando as temperaturas do sensor DS18B20 para o banco de dados do Firebase conforme apresenta a Figura 31.

**Figura 31 - Monitor de temperatura**



**Fonte:** (MAGNABOSCO, 2020)

Este monitor de temperatura foi empregado durante a produção da cerveja, essa cerveja é do tipo *belgian dark strong ale* trazendo notas maltadas bem presentes, além de notas de frutas secas, ameixa, condimentos adocicados ela contém alta carbonatação, mas não aguda. conforme demonstra a figura 32.



**Figura 32 - Produção da cerveja**



**Fonte:** (MAGNABOSCO, 2020)

Concluindo a última etapa de pasteurização da cerveja que é nada mais que o processo de aquecimento da cerveja, a fim de eliminar as atividades microbiológicas existentes no líquido para poder ser engarrafada conforme mostra a figura 33.

**Figura 33 - Produto final**



**Fonte:** (MAGNABOSCO, 2020)

## **6 LEVANTAMENTO DE REQUISITOS**

Levantamento de requisitos é primordial para um bom planejamento, pois dele deriva as demais etapas na construção de um sistema. Nenhuma outra parte do trabalho inutiliza o sistema se for feita de forma errada.

Todo o sistema é baseado nos requisitos levantados. Os requisitos levantados são divididos em funcionais que descrevem explicitamente os serviços do sistema e não funcionais que estão relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade e tecnologias envolvidas.

### **6.1 METODOLOGIA DE ANÁLISE**

A metodologia do sistema foi modelada através da análise orientada a objetos, que constitui uma linguagem de análise chamada UML

### **6.2 DIAGRAMAS UML**

Em português (linguagem de modelagem unificada), é uma linguagem visual para modelagem de softwares baseada no paradigma da orientação a objetos, usada para especificar, construir, visualizar e documentar um sistema de software. A UML permite que os desenvolvedores visualizem os seus trabalhos em diagramas.

### **6.3 DESCRIÇÃO DO GERENCIAMENTO**

Para a gestão e o gerenciamento do trabalho foi utilizado o método ágil *Scrum*, o projeto foi dividido em ciclos (com duração máxima de 2 a 4 semanas) chamados de *Sprints*.

As funcionalidades a serem implementadas no projeto são mantidas em uma lista de requisições abertas que se encontram na fila de atendimento chamada de *Backlog*.

As funcionalidades a serem finalizadas vão para a Aprovação onde é a parte da revisão, logo após a aprovação as funcionalidades vão para o Concluído.

## **6.4 DESCRIÇÃO DOS USUÁRIOS**

Os usuários do sistema proposto serão os gerentes e funcionários os quais serão responsáveis por grande parte do acompanhamento e atualização dos dados do sistema.

## **6.5 MÓDULOS**

### **6.5.1 Perspectiva do Produto: Módulo Institucional**

O Módulo Institucional nada mais é do que a microcervejaria em um todo. Cada setor tem seu papel, que irá conter alguns dados sobre o mesmo, como nome, endereço, telefone. Este módulo visa permitir que clientes atuais ou futuros clientes entrem facilmente em contato com a microempresa.

### **6.5.2 Perspectiva do Produto: Módulo Gerente**

O Módulo do Gerente é responsável por controlar todas as operações do sistema, tais como: Controle de vendas, produtos, gestão de equipe.

### **6.5.3 Perspectiva do Produto: Módulo Funcionário**

O Módulo do funcionário permitirá que o mesmo faça vendas, cadastre e atualize os dados dos produtos, consulte relatórios.

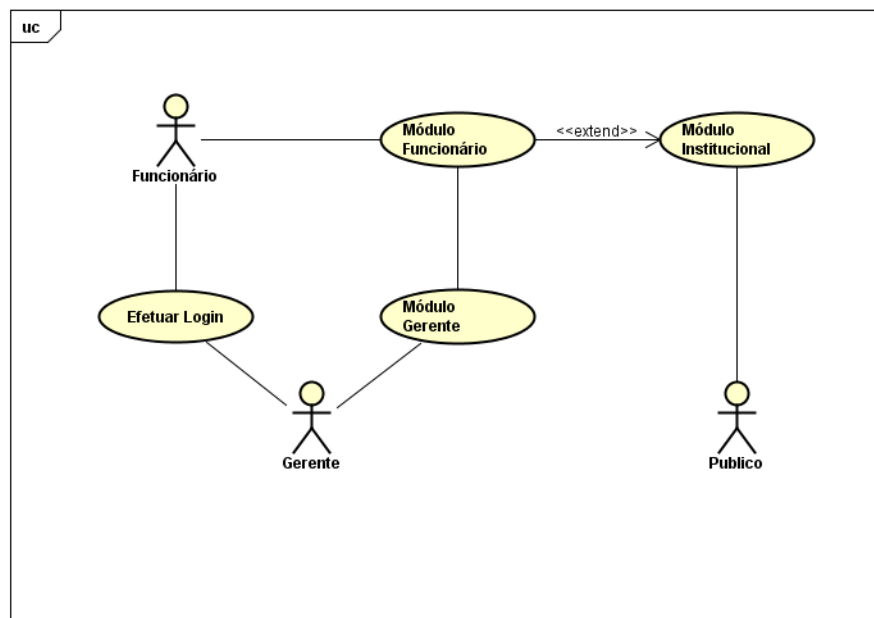
## **6.6 DIAGRAMAS DE CASO DE USO**

Segundo SOUSA (2009) o diagrama de caso de uso é um segmento de ações entre o ator e o sistema, que acontece de forma atômica, na perspectiva do ator. Ele é utilizado normalmente nas fases de levantamento e análise de requisitos do sistema. Na figura 34 é demonstrado o caso de uso geral.

### 6.6.1 Diagrama de Caso de uso geral

O diagrama de caso de uso geral é apresentado na Figura 34.

**Figura 34** - Caso de uso geral

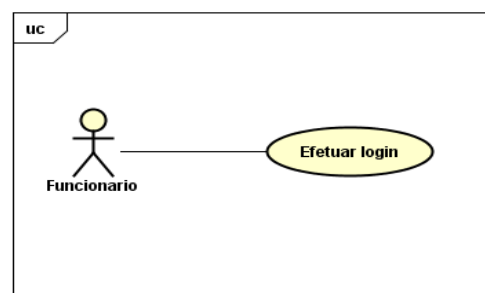


**Fonte:** O Próprio Autor (2020)

### 6.6.2 Efetuar login

O diagrama de caso de uso do módulo de efetuar login é apresentado na Figura 35 e a descrição deste caso de uso é apresentada no quadro 03.

**Figura 35** - Caso de uso efetuar login

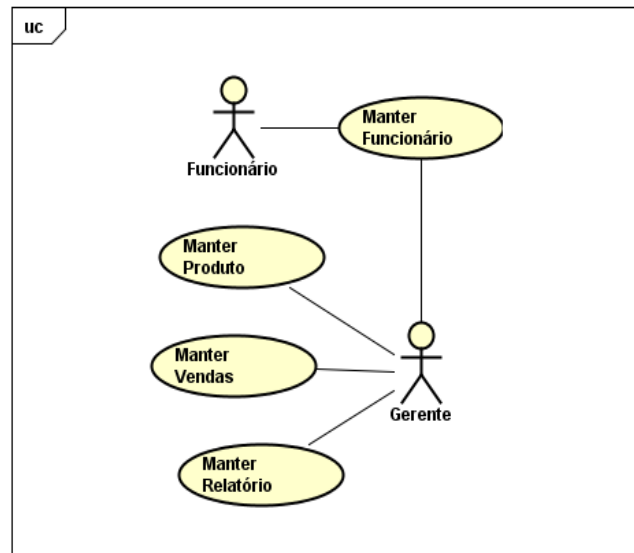


**Fonte:** O Próprio Autor (2020)

### 6.6.3 Módulo Gerente

O diagrama de caso de uso do módulo gerente é apresentado na Figura 36.

**Figura 36** - Caso de uso módulo gerente

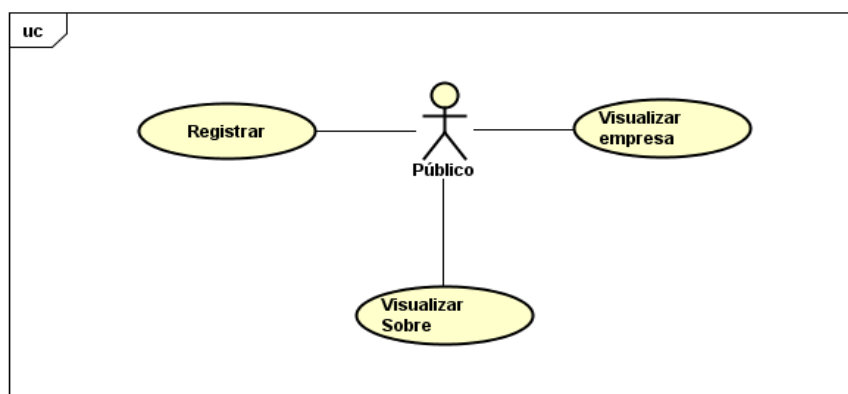


**Fonte:** O Próprio Autor (2020)

### 6.6.4 Módulo Institucional

O diagrama de caso de uso do módulo institucional é apresentado na Figura 37.

**Figura 37** - Caso de uso módulo institucional

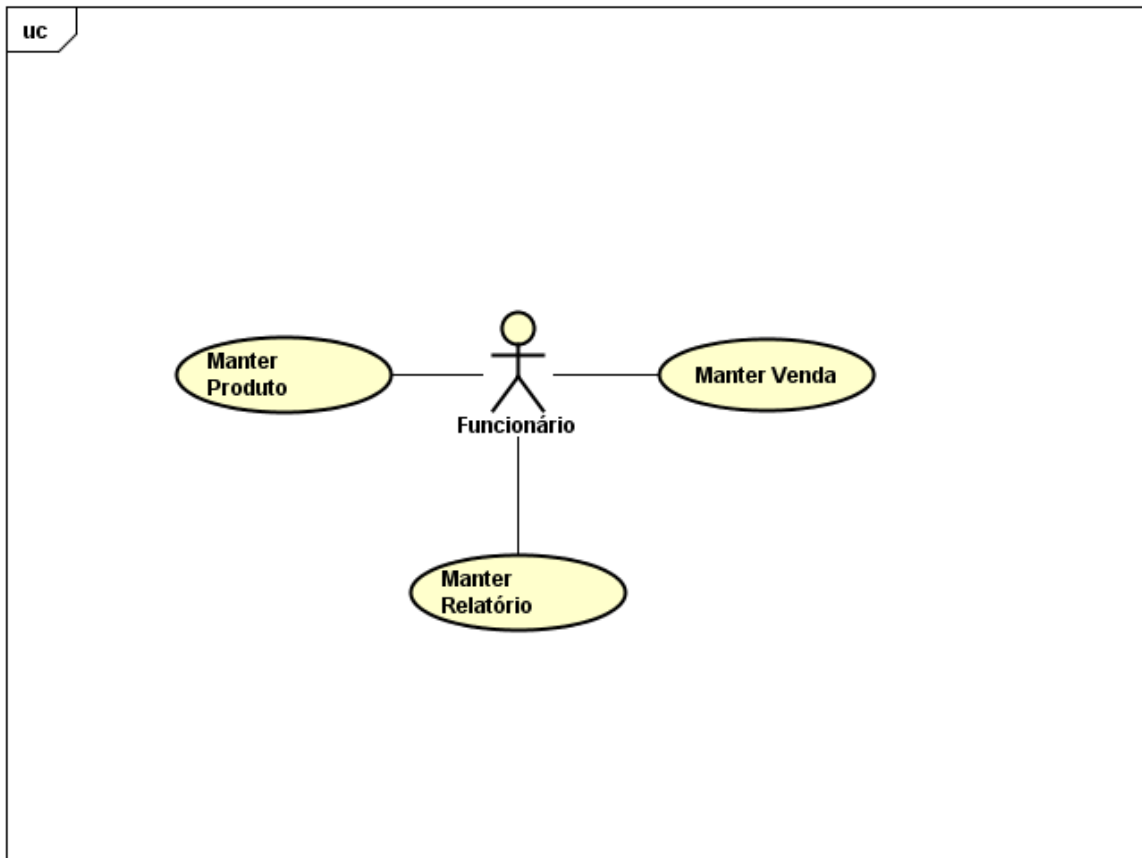


**Fonte:** O Próprio Autor (2020)

### 6.6.5 Módulo Funcionário

O diagrama de caso de uso do módulo funcionário é apresentado na Figura 38 e a descrição deste caso de uso é apresentada no quadro 04.

**Figura 38** - Caso de uso módulo funcionário



**Fonte:** O Próprio Autor (2020)

**Quadro 03:** Descrição Efetuar *Login*.

<b>Caso de Uso:</b>	Efetuar <i>Login</i> .
<b>Ator(es):</b>	Funcionário, Gerente.
<b>Pré-condições:</b>	O usuário deve estar cadastrado no sistema.
<b>Pós-condições:</b>	Usuário logado.

	Ator		Sistema	
1	Abre interface de <i>login</i>			
		2	Usuário informa login e senha	A1
3	Preenche e seleciona dados			
		4	Verifica os registros selecionados	E1 E2
		5	Efetuar <i>login</i>	A2
		6	Encerra caso de uso	

E1	O usuário digita <i>login</i> ou senha incorreta, volta ao passo 3
E2	O Sistema emite mensagem de usuário não existente, volta ao passo 3
A1	O usuário cancela a entrada ao sistema.
A2	O sistema encerra

Fonte: O Próprio Autor (2020)

**Quadro 04:** Manter Funcionário

<b>Caso de Uso:</b>	Manter Funcionário
<b>Ator(es):</b>	Gerente, Funcionário.
<b>Pré-condições:</b>	O usuário (Gerente, Funcionário) deve estar logado no sistema.
<b>Pós-condições:</b>	O usuário deve ser cadastrado, consultado, alterado ou excluído

	Ator		Sistema	
1	O caso de uso inicia quando o ator solicita “Manter Funcionário”.			
		2	Sistema oferece a interface de manutenção de funcionário.	
3	O Ator seleciona as operações de novo registro.			A1 R1 R2
		4	Ativa o formulário para registro.	
5	Preenche os dados do funcionário.			E1 E2
		6	Grava os dados do funcionário	
		7	Encerra o caso de uso.	
8	Informa os dados para a busca do funcionário.			R3
		9	Busca e mostra os dados do	



			funcionário.	
10	Atualiza os dados previamente cadastrados.			
11	Confirma alteração.			A2
		12	Grava os dados.	
		13	Encerra o caso de uso.	
14	Informa os dados para busca do funcionário.			R4
		15	Busca e mostra os dados do funcionário.	
		16	Encerra o caso de uso.	
17	Informa os dados para busca do funcionário.			R5
		18	Busca e mostra os dados do funcionário.	
19	Ator seleciona a operação de exclusão. Utiliza “Excluir funcionário”.			A3
		20	Excluir o funcionário.	
		21	Encerra o caso de uso.	

A1	O Ator seleciona a operação de alteração. Utiliza “Alterar funcionário” (linha 8). / seleciona a operação de busca. Utiliza “Buscar funcionário” (linha 14), seleciona a operação de exclusão. Utiliza “Excluir funcionário (linha 17).
A2	Cancela caso de uso.

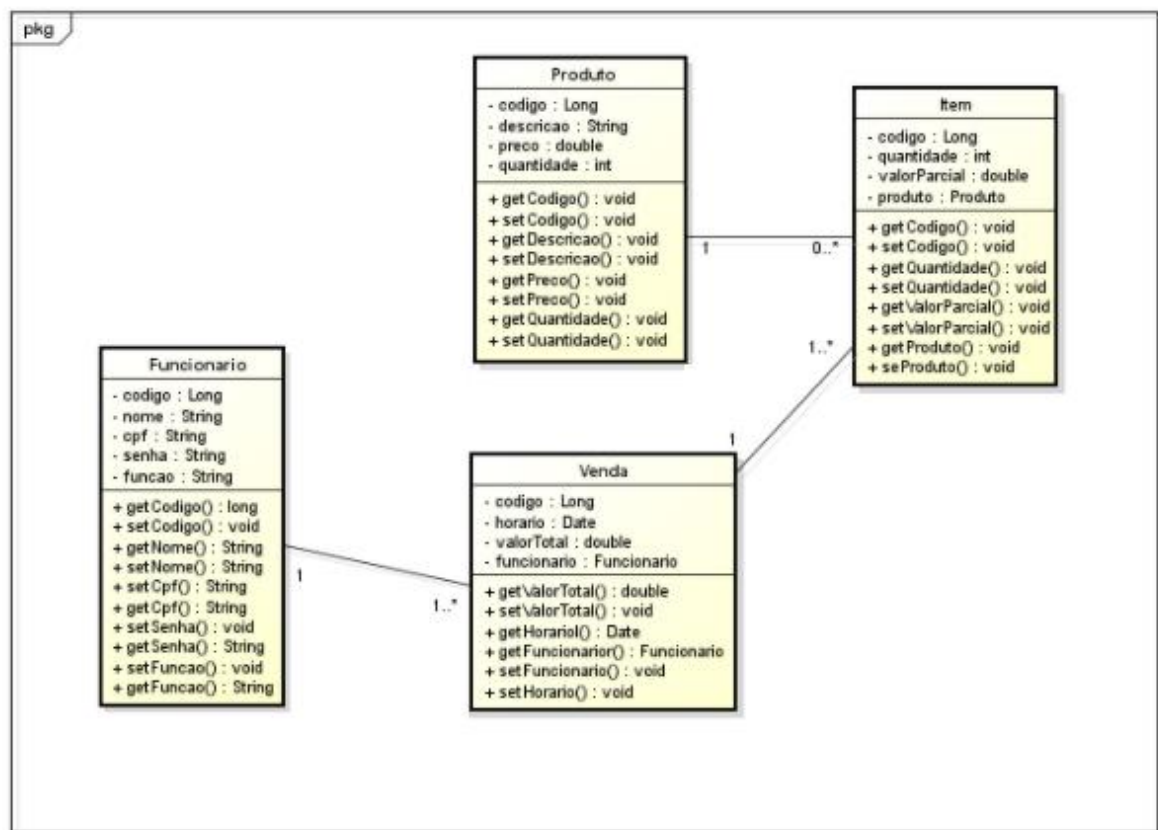
A3	Cancela caso de uso.
E1	O Ator não preenche corretamente todos os campos.
E2	Sistema emite uma mensagem de falha.
R1	Gerente pode cadastrar novos funcionários do tipo funcionário e Gerente. Funcionário não podem cadastrar novos Funcionários.
R2	Somente se o usuário for do tipo Gerente poderá adicionar novos Funcionários.
R3	Gerente pode alterar dados dos seus Funcionários.
R4	Gerente pode buscar dados dos seus Funcionários.
R5	Gerente pode excluir seus Funcionário. Os Funcionários não podem excluir nenhum usuário.

**Fonte:** O Próprio Autor (2020)

## 6.7 DIAGRAMA DE CLASSE

O diagrama de classe tem como propósito proporcionar uma visão além das principais classes do sistema e como elas se relacionam. A figura 39, mostra a estrutura básica das classes do sistema.

**Figura 39** - Diagrama de classe



**Fonte:** O Próprio Autor (2020)

## 6.8 DIAGRAMA DE ENTIDADE RELACIONAMENTO

È a exibição gráfica do modelo conceitual. Como explicado anteriormente, a modelagem deste trabalho emprega banco de dados não relacional, assim, toda a relação entre as classes bem como os dados que serão gravados, foi representada no diagrama de classe.

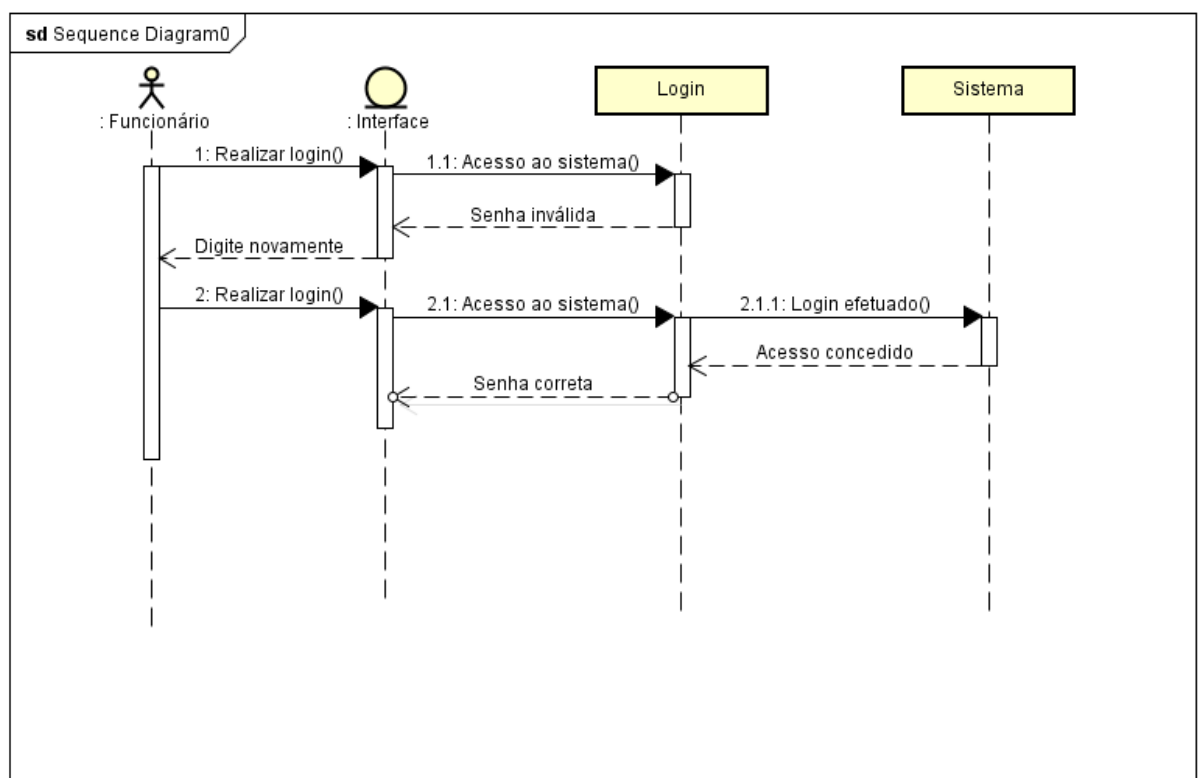
## 6.9 DIAGRAMA DE SEQUÊNCIA

Procópio (2015) explica que o diagrama de sequência procura determinar a sequência de eventos que ocorrem em um determinado processo, ele baseia-se nos diagramas de caso de uso.

### 6.9.1 Manter login

Os funcionários realizam o login se a senha estiver correta a interface do usuário realiza o acesso ao sistema, se a senha estiver errada o sistema informa senha inválida, conforme demonstra a figura 40.

**Figura 40** - Diagrama de sequência manter login

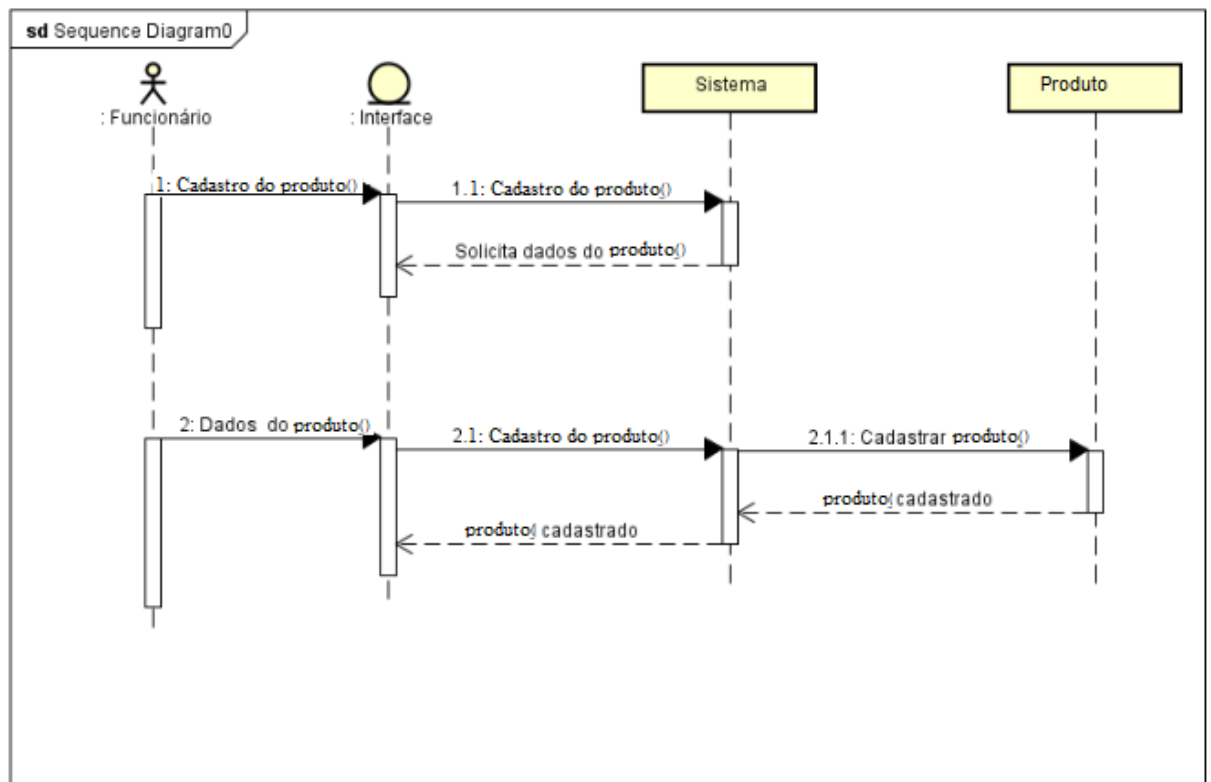


**Fonte:** O Próprio Autor (2020)

### 6.9.2 Manter Produto

O funcionário acessa cadastrar novo produto, a interface do sistema habilita as atualizações, o sistema informa ao usuário que o cadastro foi concluído com sucesso, conforme demonstra a figura 41.

**Figura 41** - Diagrama de sequência manter produto

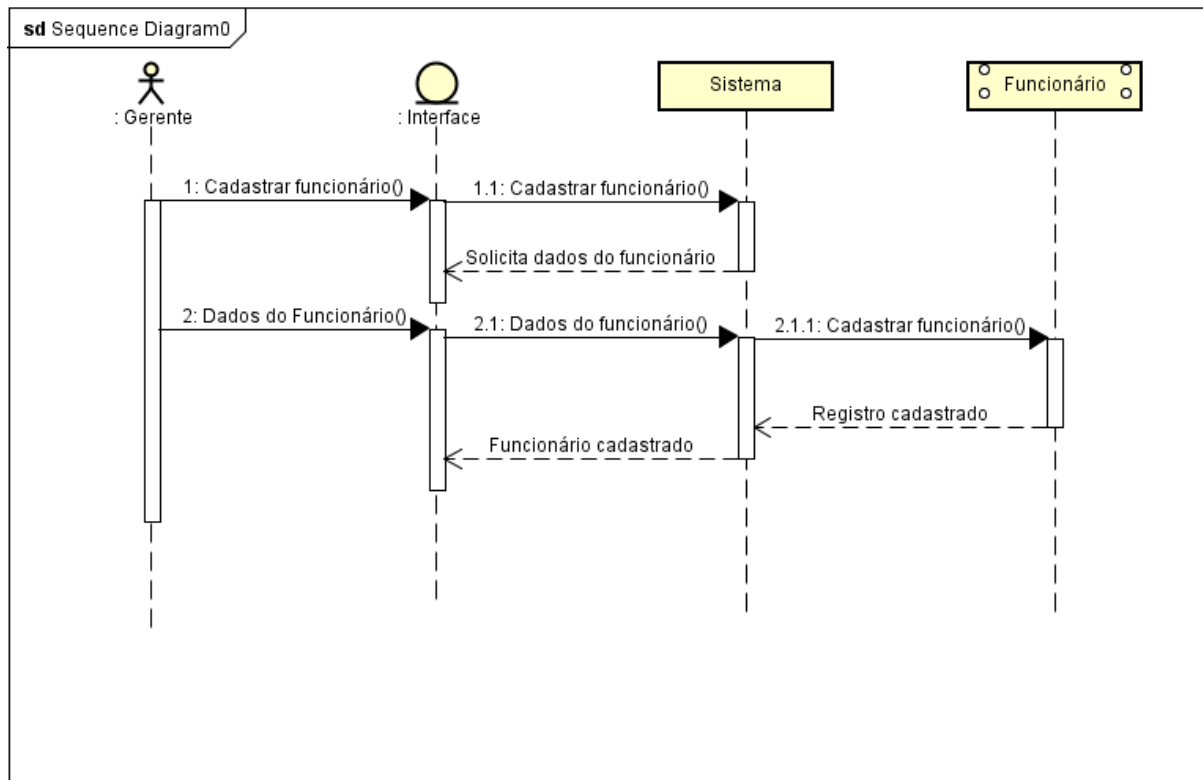


**Fonte:** O Próprio Autor (2020)

### 6.9.3 Manter funcionário

O gerente realiza o login, a interface do sistema habilita a opção de editar ou cadastrar novo usuário, o sistema informa que o usuário foi cadastrado com sucesso, conforme demonstra a figura 42.

**Figura 42** - Diagrama de sequência manter funcionário

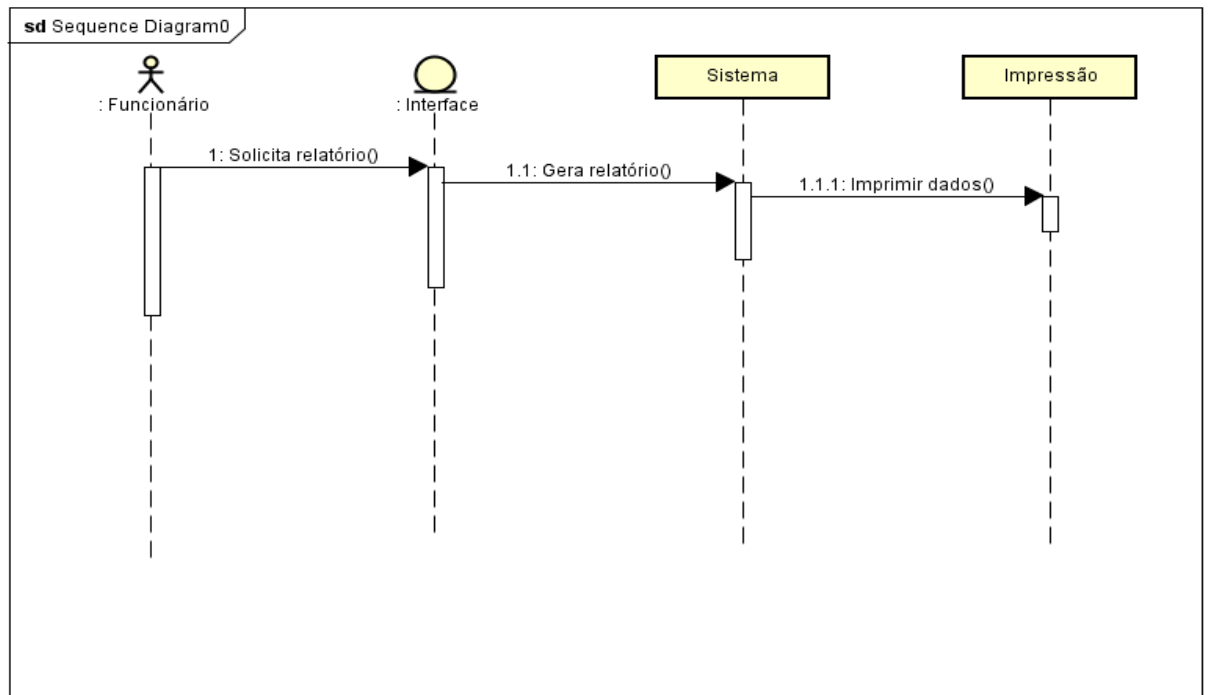


**Fonte:** O Próprio Autor (2020)

#### 6.9.4 Manter relatório

O gerente ou funcionário acessa a interface do sistema e solicita emitir relatórios o sistema gera o relatório e o usuário imprime os dados desejados, conforme demonstra a figura 43.

**Figura 43** - Diagrama de sequência manter relatório



**Fonte:** O Próprio Autor (2020)

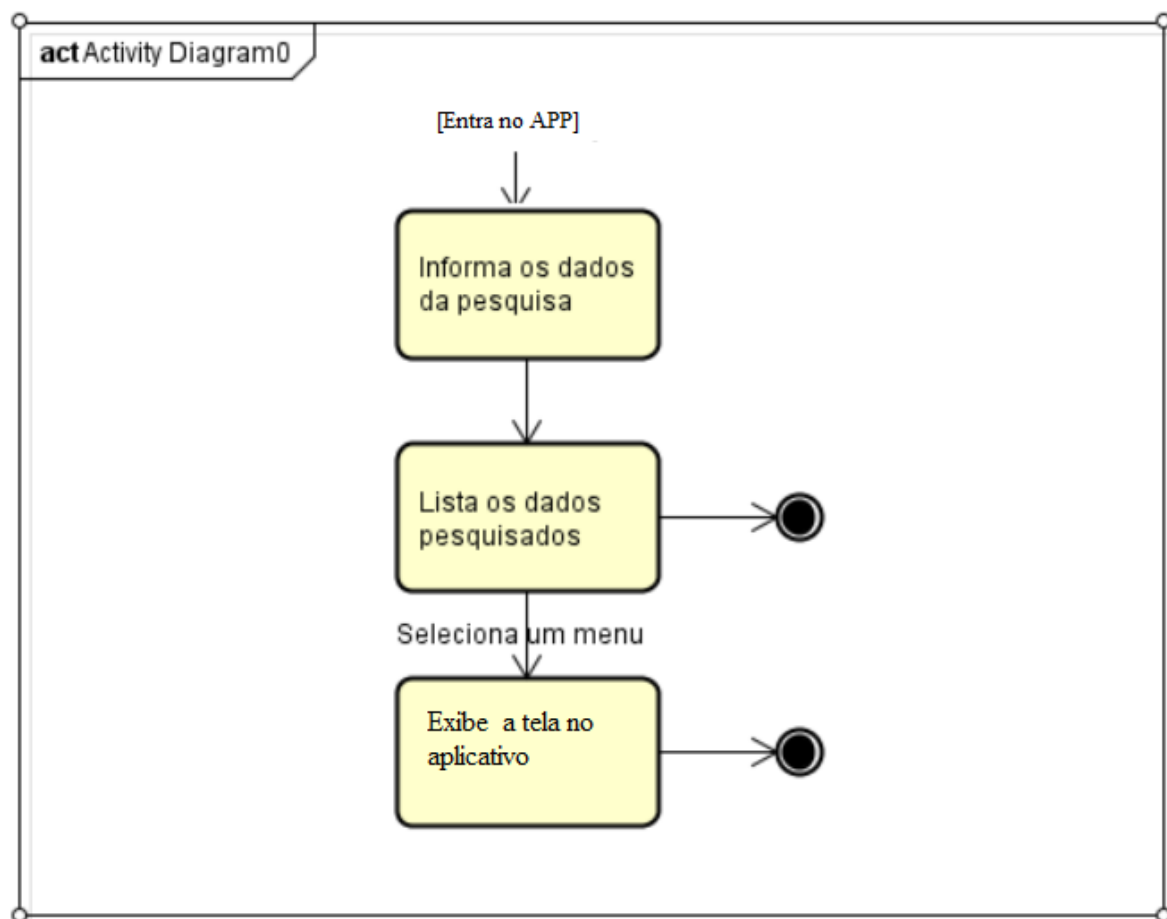
## 6.10 DIAGRAMA DE ATIVIDADE

Segundo Silva (2015) o diagrama de atividade mostra o fluxo de uma atividade para outra.

### 6.10.1 Módulo Institucional

O Módulo Institucional nada mais é do que a microcervejaria em um todo. Na figura 44 é demonstrado o módulo institucional.

**Figura 44** - Diagrama de sequência módulo institucional



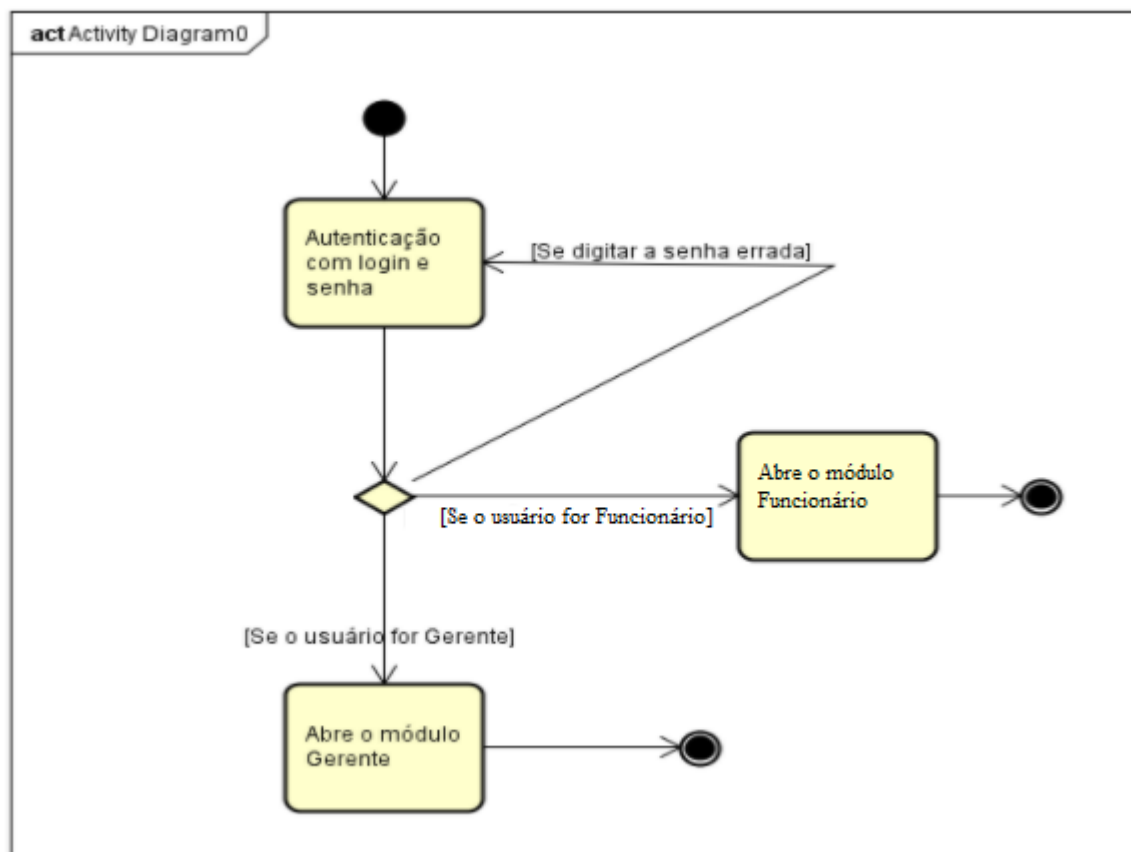
**Fonte:** O Próprio Autor (2020)



### 6.10.2 Fazer Login

O diagrama de atividade do módulo fazer login é apresentado na Figura 45.

**Figura 45** - Diagrama de atividade fazer login

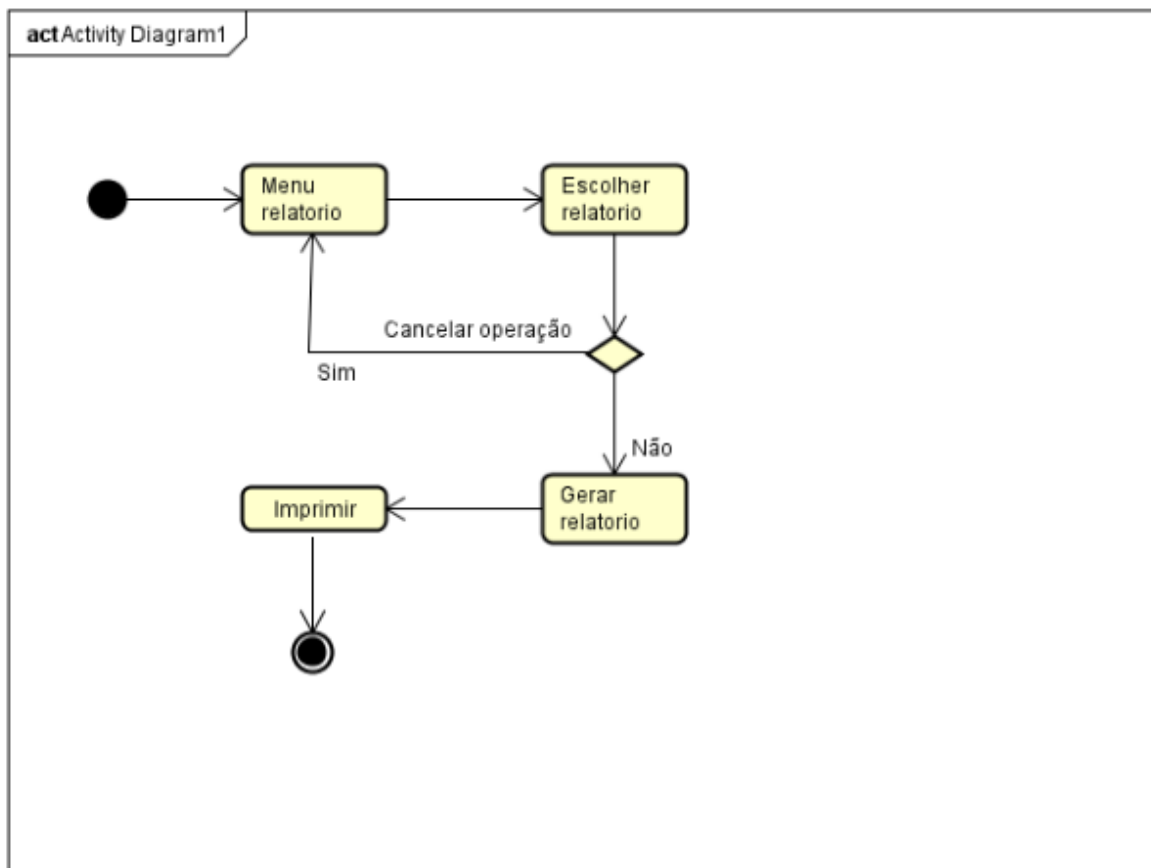


**Fonte:** O Próprio Autor (2020)

### 6.10.3 Relatório

O diagrama de atividade do módulo relatório é apresentado na Figura 46.

**Figura 46** - Diagrama de atividade módulo relatório

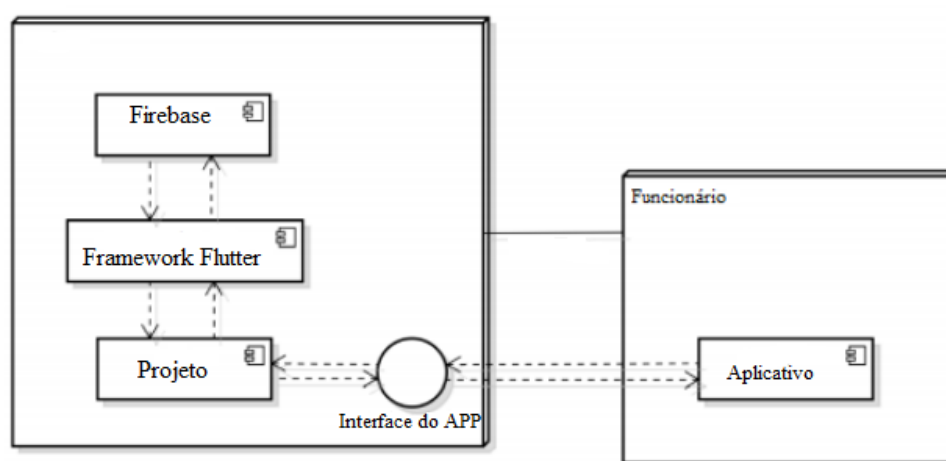


**Fonte:** O Próprio Autor (2020)

## 6.11 DIAGRAMA DE COMPONENTES E IMPLANTAÇÃO

Segundo Silva (2015) o diagrama de componentes e implantação representa um modelamento físico dos componentes de software de um determinado sistema. O diagrama de componentes e implantação é apresentado na Figura 47.

**Figura 47** - Diagrama de Componentes e Implantação



**Fonte:** O Próprio Autor (2020)

## 7 DESENVOLVIMENTO DO APLICATIVO

Neste capítulo será apresentada a análise sobre as áreas foco do trabalho como sobre o ambiente e sistema atual utilizado pelos produtores de cerveja artesanal. A análise inicia-se no comportamento da sociedade atual e serão apresentadas as funcionalidades do sistema desenvolvido no trabalho.

Na sociedade contemporânea, os amantes cervejeiros tendem a buscar diferentes notas, aromas e sabores, com isso as cervejas artesanais estão ganhando cada vez mais espaço no mercado brasileiro.

A busca pela cerveja artesanal cresceu, gerando demanda e consequentemente uma maior movimentação financeira no setor e participação na economia do país.

O Brasil é considerado o terceiro maior produtor de cerveja artesanal do mundo, estimulando uma indústria que fatura R\$ 100 bilhões por ano, atrás apenas dos Estados Unidos e da China (SARIS, 2019).

No mês de março, o Ministério da agricultura, pecuária e abastecimento lançou o anuário da cerveja 2019, o qual traz dados sobre a atividade cervejeira no Brasil nos últimos anos. Segundo o estudo, o crescimento no setor vem avançando de forma sustentada e traz números registrados de cervejarias e de cervejas que confirmam essa tendência. O país atingiu a marca de 1209 cervejarias registradas em 26 estados. Só em 2019 foram 320 novas cervejarias, ou seja, quase uma nova marca foi aberta por dia no país (SALVADOR, 2020).

## 7.1 ABERTURA

Ao abrir o aplicativo o mesmo exibirá por alguns segundos a tela de abertura com a logomarca e o nome do aplicativo, conforme é apresentado na Figura 48.

**Figura 48 - Abertura**

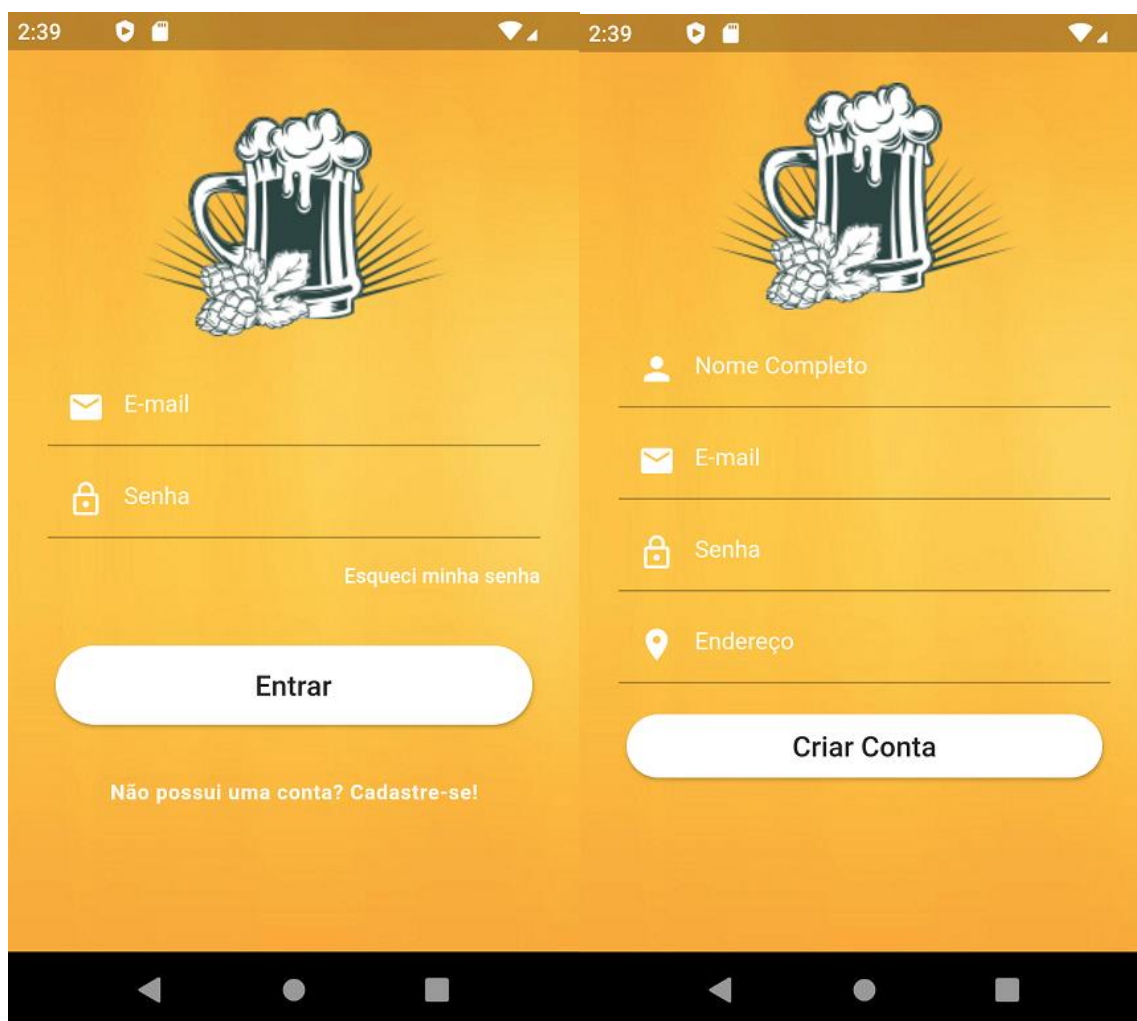


**Fonte:** (MAGNABOSCO, 2020)

## 7.2 AUTENTICAÇÃO

Após passar a tela de abertura o mesmo exibirá a tela de autenticação, o usuário informará seus dados para login e deve clicar em entrar sendo redirecionado para tela inicial, caso o usuário não seja cadastrado ele deve clicar em cadastre-se e informar os dados necessários para o cadastro e depois clicar em criar conta, conforme é apresentado na Figura 49.

**Figura 49** - Autenticação

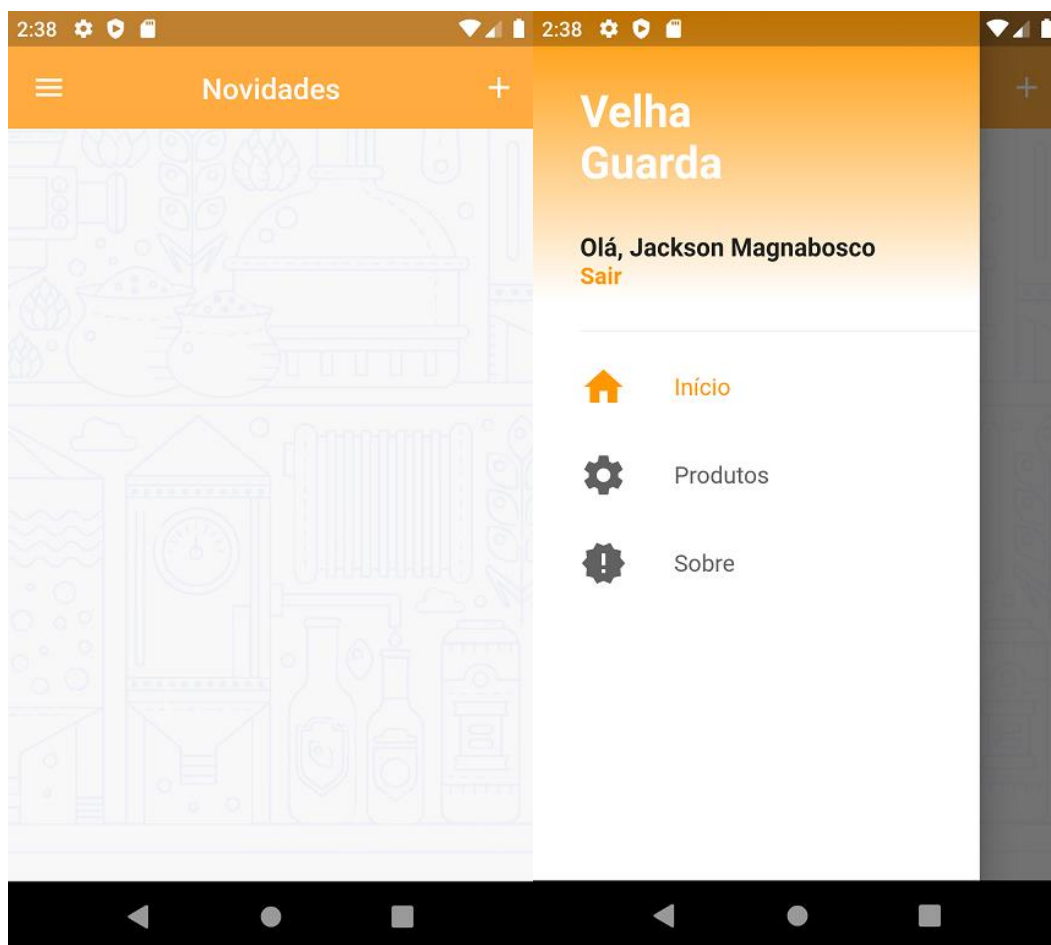


**Fonte:** (MAGNABOSCO, 2020)

### 7.3 INICIAL (DESTINADO AO PÚBLICO DE CONSUMIDORES)

Após a autenticação ser feita vai acontecer um redirecionamento para a tela inicial e o usuário poderá acessar o menu inicial que é responsável pela navegação entre as telas do aplicativo, conforme é apresentado na Figura 50.

**Figura 50** - Inicial

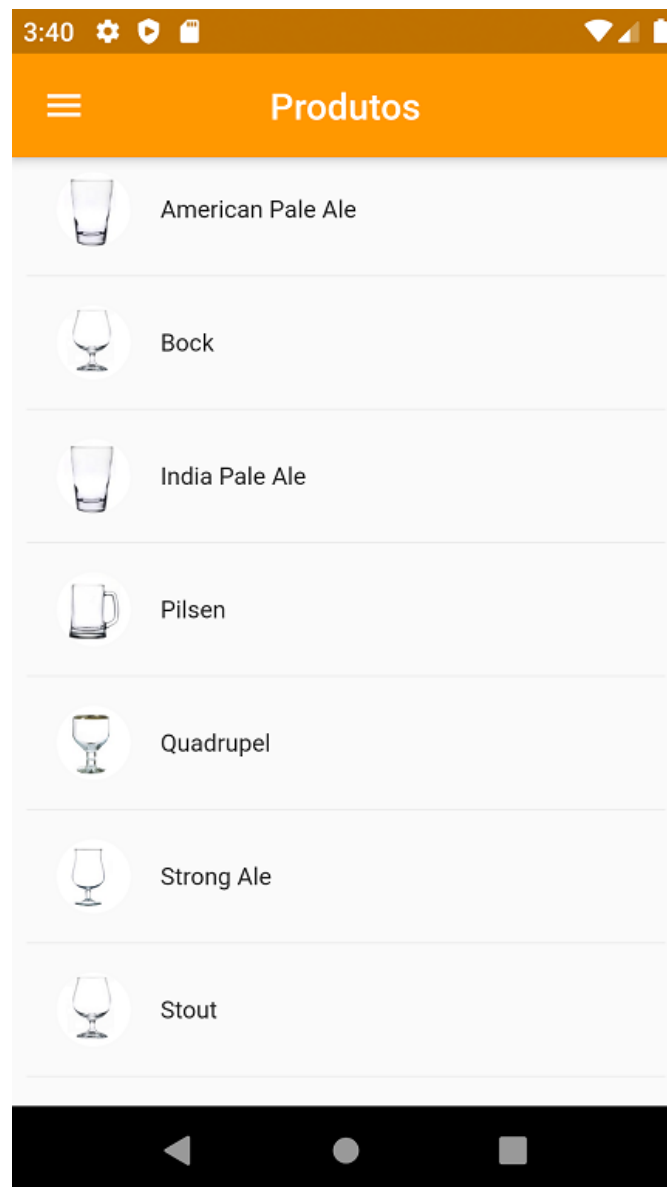


**Fonte:** (MAGNABOSCO, 2020)

## 7.4 CATEGORIA DE PRODUTOS

Após acessar o menu inicial no canto superior esquerdo e clicar na categoria de produtos vai aparecer uma lista com os mais diversos tipos de cervejas trazendo diferentes notas, aromas e sabores, conforme é apresentado na Figura 51.

**Figura 51** - Categoria de produto

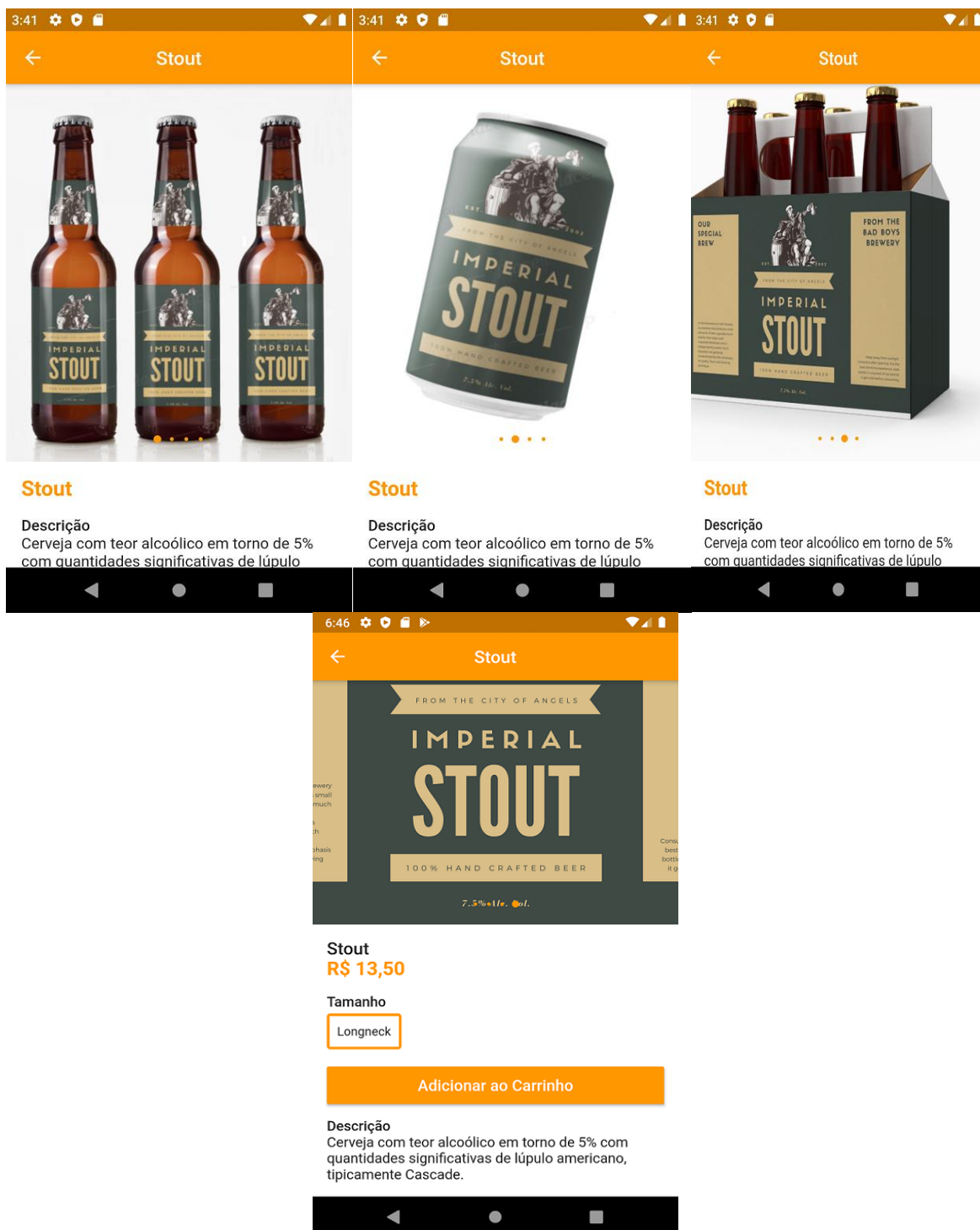


**Fonte:** (MAGNABOSCO, 2020)

## 7.5 PRODUTO

Após o usuário escolher a categoria do produto, vai abrir a tela do produto onde vai ter imagens e uma breve descrição sobre ele, conforme é apresentado na Figura 52.

**Figura 52 - Produto**



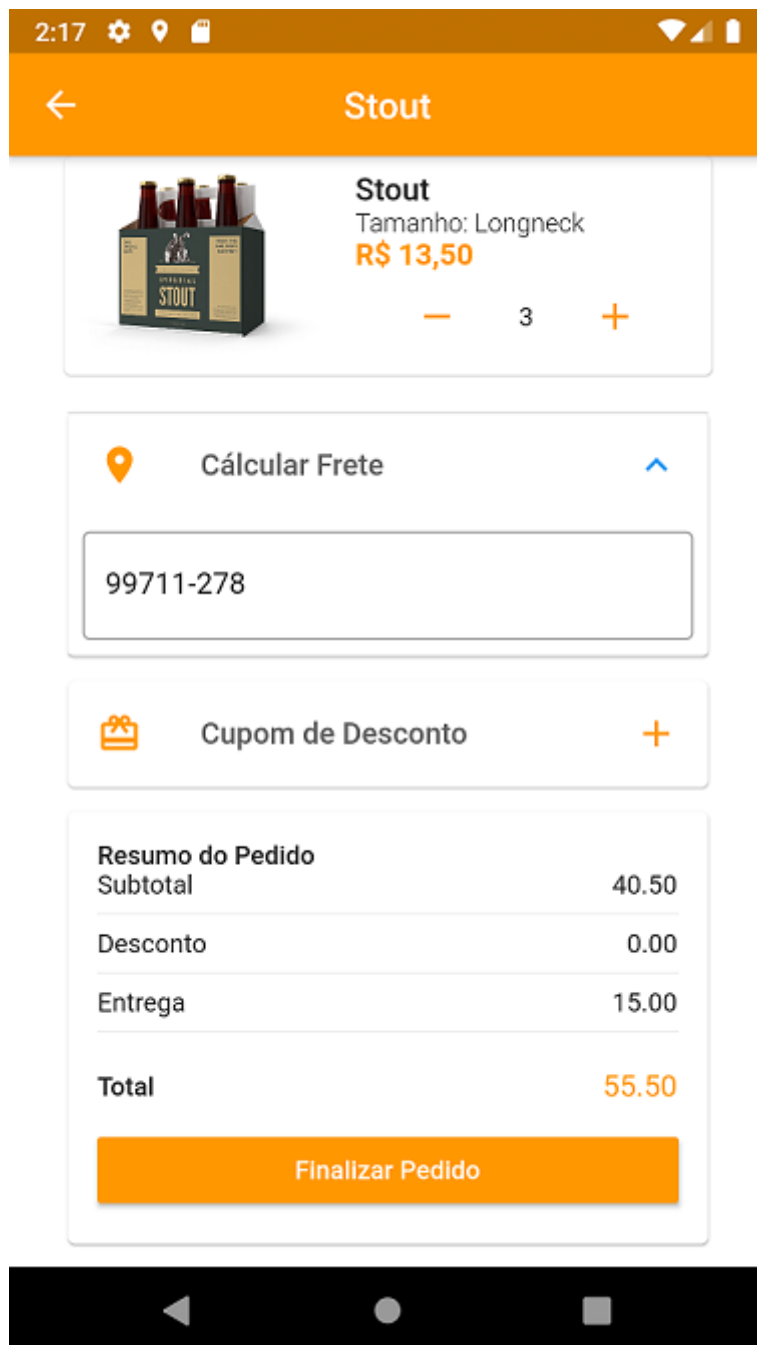
Fonte: (MAGNABOSCO, 2020)



## 7.6 CARRINHO DE COMPRAS

Após o usuário adicionar ao carrinho vai abrir a tela de compra, onde o usuário vai escolher quantas unidades vai comprar, calcular o valor do frete, adicionar cupom de desconto se tiver e finalizar o pedido, conforme é apresentado na Figura 53.

**Figura 53** - Carrinho de compra



**Fonte:** (MAGNABOSCO, 2020)

## 7.7 FORMA DE PAGAMENTO

Após o pedido ser finalizado o usuário vai ser direcionado a tela de pagamento onde vai cadastrar seu cartão de crédito e pagar o produto, após a verificação do pagamento o usuário vai ser redirecionado para a página inicial, conforme é apresentado na Figura 54.

**Figura 54** - Forma de pagamento

The figure consists of three screenshots of a mobile application interface for payment.

The top-left screenshot shows a card registration screen. It features a large orange card graphic with a masked number (\*\*\*\* \* \* \* \*), the name "NOME / SOBRENOME", and the validity date "VALIDADE MM/AA". Below the card, there are input fields for "Número do cartão" and "Validade", followed by a "Próximo" button and a numeric keypad.

The top-right screenshot shows the same card registration screen but with the card number "5519 3917 0923 0078" and the name "JACKSON MAGNABOSCO" entered. The validity date is "02/22". It includes "Voltar" and "Próximo" buttons.

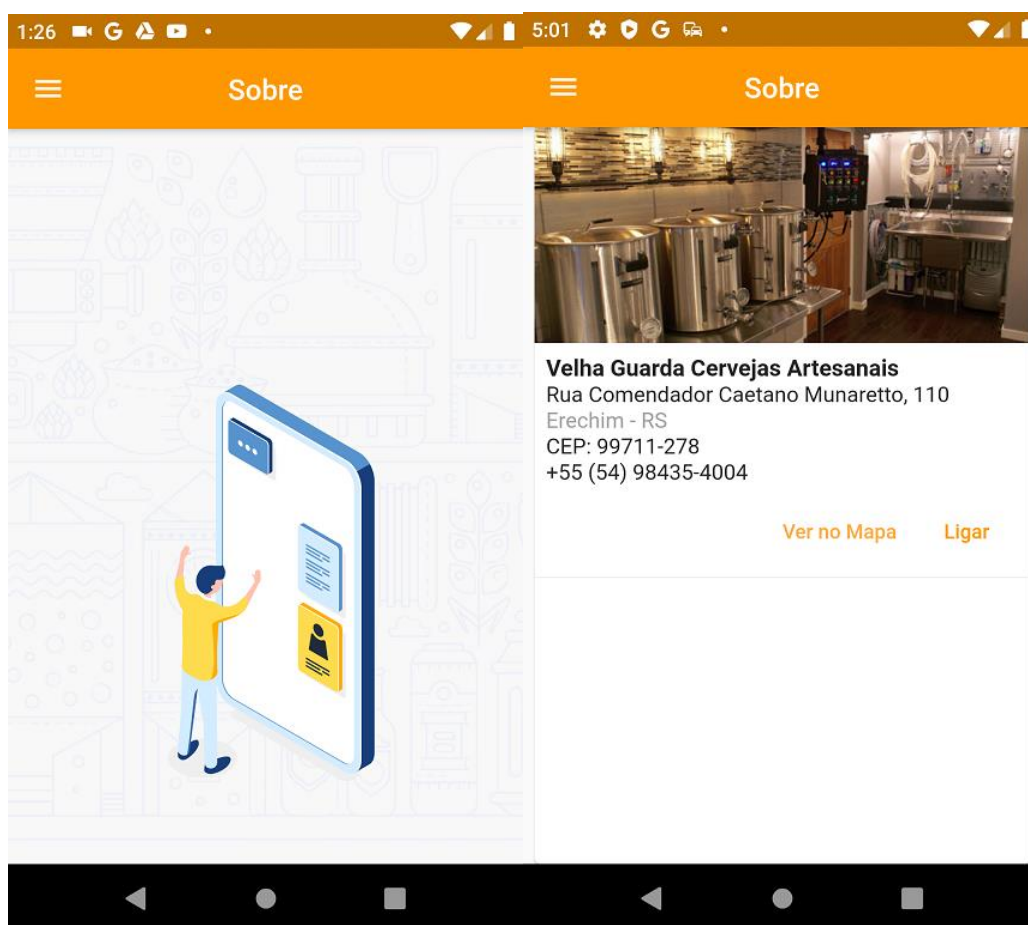
The bottom screenshot shows a payment confirmation screen. It displays a card with the name "Jackson Magnabosco" and the number "999". Below the card is a "CVV" field with "999" entered. There are "Voltar" and "Finalizar" buttons. A large green checkmark icon is overlaid on the screen, indicating successful payment. A numeric keypad is also visible at the bottom.

**Fonte:** (MAGNABOSCO, 2020)

## 7.8 SOBRE

Após acessar o menu inicial no canto superior esquerdo e clicar na categoria sobre vai aparecer a tela que descreve sobre a empresa e o time que faz parte dela, serve para tirar as dúvidas ou contatar algum problema encontrado no aplicativo, conforme é apresentado na Figura 55.

**Figura 55** - Sobre

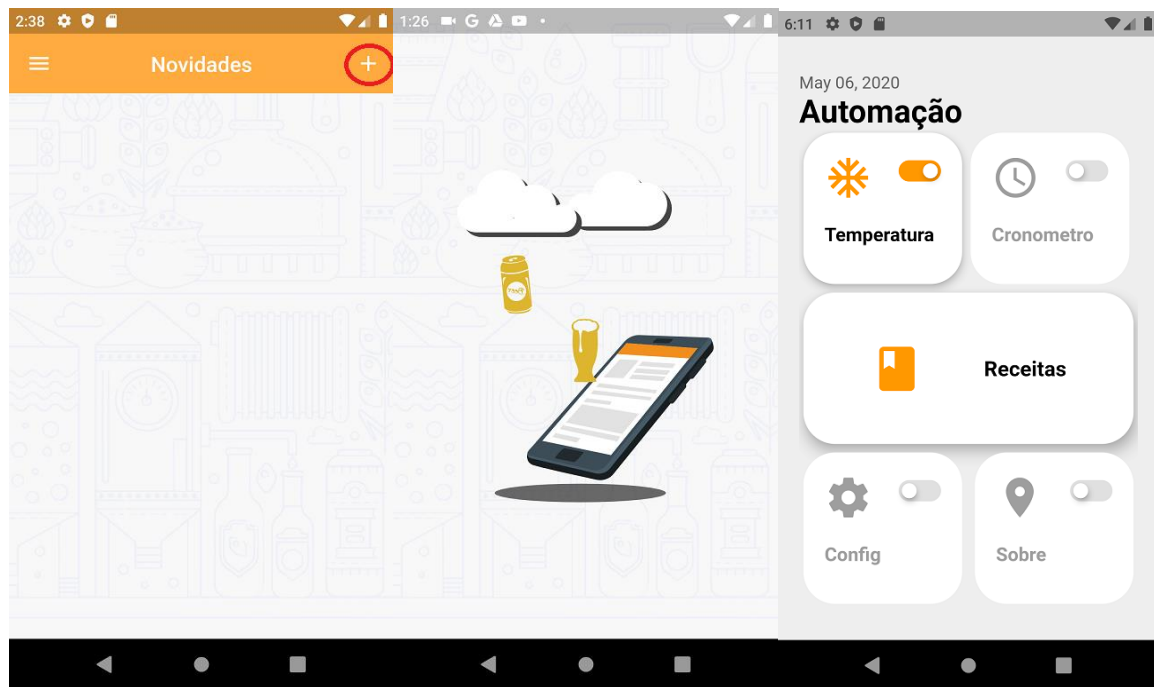


**Fonte:** (MAGNABOSCO, 2020)

## 7.9 AUTOMAÇÃO (PARA O PROCESSO DE CRIAÇÃO - DESTINADO AOS CERVEJEIROS INDEPENDENTES)

Após acessar o menu automação no canto superior direito, vai aparecer o menu com os itens que vão colaborar para automatizar os processos da cervejaria, conforme é apresentado na Figura 56.

**Figura 56 - Automação**

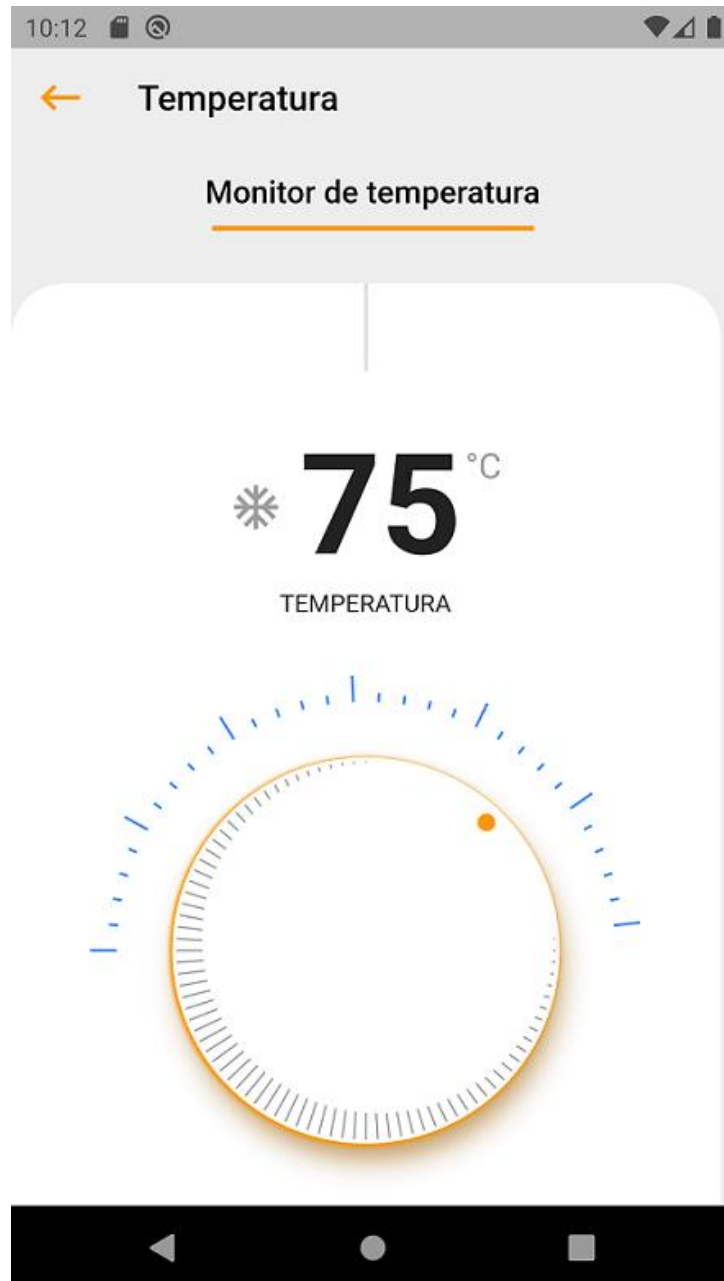


**Fonte:** (MAGNABOSCO, 2020)

## 7.10 TEMPERATURA

Após acessar o menu temperatura o usuário vai conseguir visualizar a temperatura do processo da cerveja em tempo real ou a última atualização dos dados no banco de dados do Firebase, conforme é apresentado na Figura 57.

**Figura 57** - Temperatura

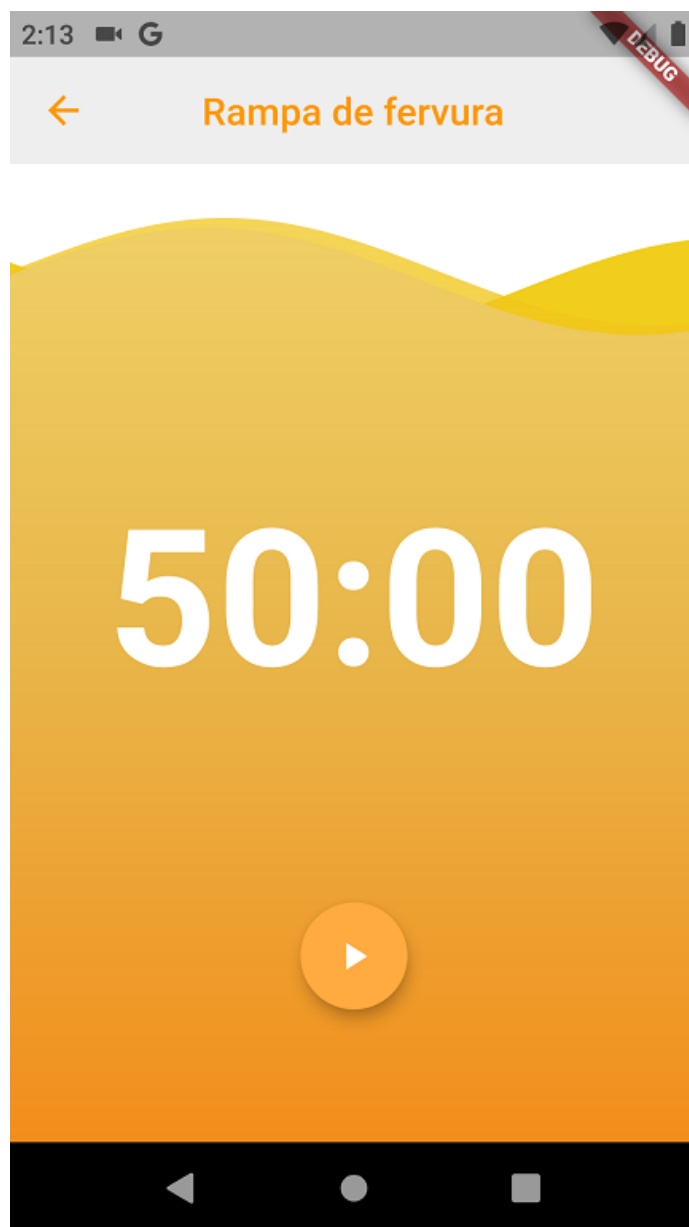


**Fonte:** (MAGNABOSCO, 2020)

## 7.11 CRONÔMETRO

Após acessar o menu do cronômetro, esta tela serve para avisar a troca de cada rampa no processo de fervura enviando notificações por push no celular, conforme é apresentado na Figura 58.

**Figura 58** - Cronômetro

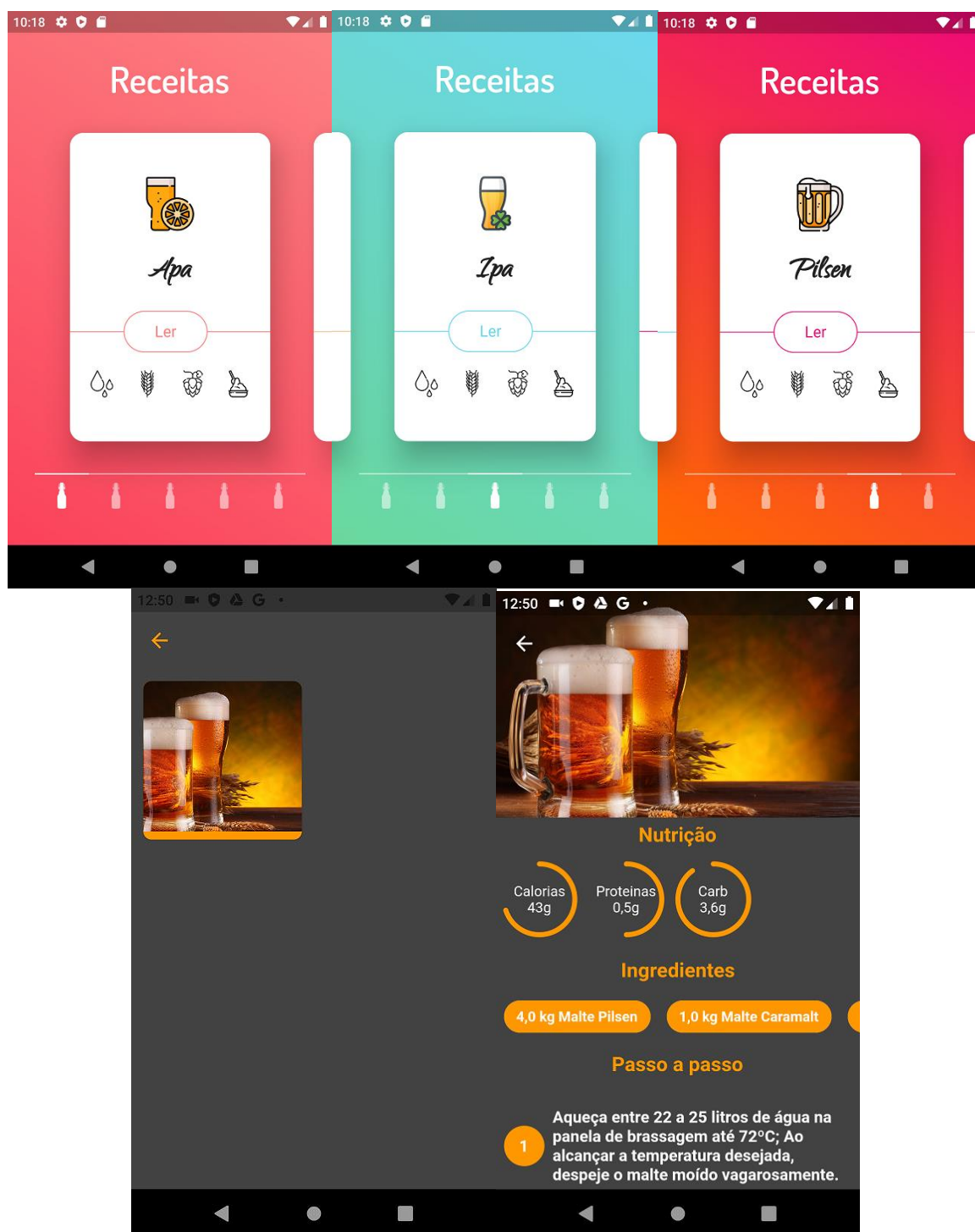


**Fonte:** (MAGNABOSCO, 2020)

## 7.12 RECEITAS

Após acessar o menu de receitas, vai aparecer uma tela com diversas receitas com o passo a passo de como serem preparadas, conforme é apresentado na Figura 59.

**Figura 59-** Receitas



Fonte: (MAGNABOSCO, 2020)

## 8 RESULTADOS E DISCUSSÕES

O primeiro resultado apontado é que o aplicativo, foi conclusivo na parte de desenvolvimento assim como na parte de teste com cervejeiros independentes. Em uma segunda etapa, foi realizado teste com cervejeiros, os quais enfatizaram que o termômetro digital e o acompanhamento realizado no processo de produção “na palma de sua mão” facilitou e ainda proporcionou uma melhor experiência aos mesmos.

Ainda diante de um relato com os cervejeiros eles apontaram que o aplicativo se tornou uma facilidade para o processo de produção, pois ficou mais fácil a avaliação de temperatura, assim como, o aplicativo pode ser utilizado desde a fase inicial da produção, por conter o passo a passo de ingredientes e produção da cerveja.

Com tudo, ainda os cervejeiros em seu relato concluíram que preferiram utilizar o aplicativo, até pelo seu aspecto interativo, com telas coloridas e animadas.

Um segundo resultado obtido neste estudo, foi que a parte de vendas ao público tornou-se facilitada diante a possibilidade que o aplicativo fornece, com entrega na casa direto do consumidor, um mercado apontado em constante crescimento, por diversas pesquisas econômicas no país em 2020.

Um terceiro resultado foi a gestão de equipe, quando realizado o processo de produção de cerveja, por mais de um cervejeiro independente, a organização de funções e responsabilidades de cada um, ainda pode ser gerida pelo aplicativo, sendo dividido em funções como: Receita, gestão de tempo de rampas de fervura, temperatura, entre demais funções realizadas.

Por fim é evidente que o uso do aplicativo facilita desde o processo de criação da receita, realização, gestão da equipe, vendas e consumo.



## 9 CONCLUSÃO E TRABALHOS FUTUROS

O desenvolvimento da aplicação permitiu explorar um grande acervo tecnológico, começando pelo paradigma de desenvolvimento móvel híbrido, gerando automaticamente aplicativos multiplataforma. Foi possível esclarecer também que, na maioria dos casos, é uma grande vantagem optar por esta metodologia, pois o investimento financeiro e o prazo acabam sendo menores, além do fato de existirem *frameworks* como o Flutter que acelerarem muito mais este processo.

Um tópico que requer atenção foi o banco de dados do Firebase, pelo fato de ser um banco de dados não relacional, um conceito diferente do utilizado no desenvolvimento mais tradicional. Percebeu-se que estes bancos vêm ganhando espaço nas grandes aplicações atuais. Para aqueles acostumados com o paradigma relacional, em primeiro momento o funcionamento do banco parece confuso, porém assim que seus conceitos são absorvidos, a impressão é que ele se torna mais simples de trabalhar. Além disso, o banco encaixou-se perfeitamente nos requisitos da aplicação desenvolvida, mantendo o aplicativo funcional mesmo sem uma conexão com a internet, sincronizando os dados em tempo real.

Por intermédio das pesquisas realizadas, e a partir delas a realização da modelagem foi possível implementar uma ferramenta que atenda os cervejeiros independentes.

No processo de desenvolvimento deste trabalho ocorreu um grande enriquecimento em relação à modelagem orientada a objetos, e de um modo geral a realização deste trabalho, permitiu ampliar conhecimentos adquiridos durante o decorrer do curso.

Em meio ao cenário de crescimento da tecnologia no meio industrial, o aplicativo Velha guarda é uma ferramenta com grande possibilidade de aceitação por produtores de cerveja artesanal.

Conforme o desenvolvimento deste trabalho, percebeu-se que as áreas referentes à gastronomia cervejeira e de dispositivos móveis cresceu muito nos últimos tempos e continua crescendo, fazendo delas ótimos alvos para investimentos. Com esta motivação, foi possível desenvolver um artefato tecnológico para dispositivos móveis que atende diretamente ao público cervejeiro.

De acordo com a análise feita em relato com pequenos produtores da região, foi possível entender a necessidade de fazer este tipo de automação utilizando uma aplicação *mobile*. Tornando possível a consulta da temperatura em tempo real de produção de maneira informatizada e organizada na palma da mão foi o que mais agradou os produtores.

Além dos conceitos cervejeiros, foi possível explorar as tecnologias utilizadas para o desenvolvimento deste projeto e entender a referência de desenvolvimento móvel utilizando as principais tecnologias voltadas para o ramo mobile. A vantagem da utilização do Flutter Framework, juntamente com o Firebase, foi a facilidade de utilização e a aceleração do processo de desenvolvimento, fornecendo layout organizado, diminuindo tempo e custo de desenvolvimento, além de possuir uma curva de aprendizado muito pequena, o Flutter oferece praticidade de interação com o código pré-existente.

Para empreendimentos futuros pretendemos implantar um controle de vendas a prazo, para dar mais comodidade a cada consumidor, e um fluxo de caixa completo com parcelamento e lançamento de nota fiscal. O banco de dados receberá mais réplicas, aumentando a disponibilidade da aplicação, buscando trazer para a microcervejaria e para seus colaboradores controle e segurança sobre cada movimentação melhorando o desempenho de sistema e a satisfação dos clientes.

Espera-se que estes desenvolvimentos futuros da aplicação possam auxiliar ainda mais o trabalho dos produtores de cerveja artesanal, aumentando a eficiência e a praticidade nos processos e garantindo um registro permanente das informações pertinentes a cada lote de cerveja produzida.

## REFERÊNCIAS

ADRIANO, Thiago. **Introdução ao Firebase**. Disponível em: <<https://medium.com/@programadriano/introdu%C3%A7%C3%A3o-ao-firebase-bd59bfd03f29>> . Acesso em: 25 abril. 2020

ANDRADE, Kleber. **Instalando e Configurando Flutter no Windows**. Disponível em: <<https://medium.com/flutter-comunidade-br/instalando-e-configurando-flutter-no-windows-cae74711df1e>> . Acesso em: 26 setembro. 2020

CHEDE, C. **Desenvolvimento de apps – Parte 2: híbrido, nativo ou web? 2013**. Disponível em: <[https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/desenvolvimento\\_de\\_apps-parte\\_2\\_hibrido\\_nativo\\_ou\\_web](https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/desenvolvimento_de_apps-parte_2_hibrido_nativo_ou_web)>. Disponível em: 28 de Setembro de 2019.

DIAS, Diego. **Por que usar Flutter**. Disponível em: <[https://www.flutterbrasil.com/read-blog/1\\_1-por-que-usar-flutter.html](https://www.flutterbrasil.com/read-blog/1_1-por-que-usar-flutter.html)>. Acesso em: 24 abril. 2020

DIAS, Rafael. **Instalando o Flutter no Windows**. Disponível em: <<https://medium.com/sysvale/instalando-o-flutter-no-windows-7d19cfdae1b8>> . Acesso em: 26 setembro. 2020

ELECFREAKS. **Display Values of Multiple DS18B20s on ESP8266 NodeMCU Web Server**. Disponível em: <<https://lastminuteengineers.com/multiple-ds18b20-esp8266-nodemcu-tutorial/>>. Acesso em: 26 setembro. 2020

HIPSTER PONTO TECH: **Flutter** – Hipsters #183. Entrevistadores: Paulo Silveira, Igor Borges, Alexandre Freire, Guilherme Silveira, Alex Vieira, Fausto Blanco, Gabriel Sávio, Roberta Arcoverde. Produtora: Alura, 14 jan. 2020. Podcast. Disponível em: <<https://open.spotify.com/episode/08VJEPbd6EmQhyle3ktT9n>>. Acesso em: 24 abril. 2020.

HONDA, Rafael. **Nossa reunião sobre Flutter e Dart**. Disponível em: <<https://medium.com/mega-senior/google-i-o-19-nosso-resum%C3%A3o-sobre-flutter-e-dart-402611573b23>>. Acesso em: 24 abril. 2020

IANNI, V. **Introdução aos bancos de dados NoSQL**. 2012. Disponível em: <<https://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044>>. Acesso em: 07 set. 2015.

LIMA, Alexandre. **Por Flutter usa dart**. Disponível em: <<https://alexandredslima.com/2019/10/06/por-que-flutter-usa-dart/>>. Acesso em: 24 abril. 2020

MADEIRA, Daniel. **DS18B20 – Sensor de temperatura inteligente**. Disponível em: <<https://portal.vidadesilicio.com.br/sensor-de-temperatura-ds18b20/>> . Acesso em: 27 abril. 2020

MADEIRA, Daniel. **DS18B20 – GUIA RÁPIDO#16 — Utilizando sensor de temperatura DS18B20 com ESP8266 NodeMCU**. Disponível em: <<https://medium.com/@automacaoem5minutos/guia-r%C3%A1pido-16-utilizando-sensor-de-temperatura-ds18b20-com-esp8266-nodemcu-403c74a6238f>> . Acesso em: 08 setembro. 2020

MADUREIRA, D. **Aplicativo Nativo, web app ou aplicativo híbrido?**. Disponível em: <<https://usemobile.com.br/aplicativo-nativo-web-hibrido/#o-que-e-app-nativo>>. Acesso em: 14 de outubro de 2020.

MAGNABOSCO, Jackson. **Aplicativo para automação de uma micro cervejaria**. Disponível em: <<https://github.com/jacksonn455/automacao-cervejaria>>. Acesso em: 04 agosto. 2020

MAGNABOSCO, Jackson. **Monitor de temperatura**. Disponível em: <<https://github.com/jacksonn455/Monitor-de-temperatura>>. Acesso em: 15 outubro. 2020

MARCUSSO, Eduardo. **Anuário da cerveja 2019**. Disponível em: <<https://www.cervesia.com.br/noticias/noticias-de-mercado-cervejeiro/7759-anuario-da-cerveja-2019.html>> . Acesso em: 29 abril. 2020

MOORE, Kevin. **Platform-Aware Widgets in Flutter**. Disponível em: <<https://www.raywenderlich.com/4968762-platform-aware-widgets-in-flutter>> . Acesso em: 25 abril. 2020

MORAIS, José. **O QUE É ESP8266 - A FAMÍLIA ESP E O NODEMCU**. Disponível em: <<https://portal.vidadesilicio.com.br/o-que-esp8266-nodemcu/>>. Acesso em: 27 abril. 2020

MUNIZ, Luiz. **Parte 2 — Instalando o Android Studio no Windows**. Disponível em: <<https://medium.com/@lcmuniz/parte-2-instalando-o-android-studio-no-windows-4e9f28c9a84#:~:text=Ap%C3%B3s%20o%20arquivo%20ser%20baixado,os%20componentes%20que%20ser%C3%A3o%20instalados.>>. Acesso em: 27 abril. 2020

PEDRO, João. **Firestore — como, quando e porque utilizar esse Banco de Dados do Google**. Disponível em: <<https://medium.com/@dezembro/firebase-como-quando-e-porque-utilizar-esse-banco-de-dados-do-google-f65ab5ae182a>> . Acesso em: 25 abril. 2020

PROCÓPIO, Fábio. **Diagrama de Sequência**. Disponível em: <<https://docente.ifrn.edu.br/givanaldorochoa/disciplinas/engenharia-de-software-licenciatura-em-informatica/diagrama-de-sequencia>> . Acesso em: 25 abril. 2020

ROVAI, Marcelo. **O IOT FEITO SIMPLES: MONITORANDO MÚLTIPLOS SENSORES.** Disponível em: <<https://tudosobreiot.com.br/o-iot-feito-simples-monitorando-multiplos-sensores/>> . Acesso em: 27 abril. 2020

ROVAI, Marcelo. **O IoT feito simples: Monitorando a temperatura desde qualquer lugar.** Disponível em: <<http://labdegaragem.com/profiles/blogs/o-iot-feito-simples-monitorando-a-temperatura-desde-qualquer>> . Acesso em: 27 abril. 2020

SADALAGE, P. J.; FOWLER, Martin. **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence.** Crawfordsville: Addison-Wesley, 2013.

SALVADOR, Douglas. **Novidades do mercado.** Cerveja de todos os jeitos, Curitiba, v. 86, n. 7, p. 3-3, abril. 2020.

SARIS, Simoni. **Um olhar mais econômico para a cerveja artesanal.** Disponível em: <<https://www.folhadelondrina.com.br/economia/um-olhar-mais-economico-para-a-cerveja-artesanal-2971424e.html>> . Acesso em: 24 abril. 2020

SAVIO, Gabriel. Push Notifications - **Flutter com OneSignal.** Disponível em: <<https://medium.com/@gbrlsavio2/push-notifications-flutter-com-onesignal-fd5f68ead24d>> . Acesso em: 05 maio. 2020

SESSA, Claudiney. **Iniciando no Flutter: Configurando o ambiente de desenvolvimento.** Disponível em: <<https://medium.com/flutter-comunidade-br/iniciando-no-flutter-parte1-52e120e007d7>> . Acesso em: 26 setembro. 2020

SILVA, Flávio. **UML – Diagramas Comportamentais Diagrama de Atividades** Disponível em: <<http://www.facom.ufu.br/~flavio/swmod-files/files/2015-02/10-UML-Diagramas-Atividades.pdf>> . Acesso em: 26 setembro. 2020

SILVA, Flávio. **UML – Diagramas Estruturais Diagrama de Componentes** Disponível em: <<http://www.facom.ufu.br/~flavio/swmod-files/files/2015-02/11-UML-Diagramas-Compoentes-e-Implantacao-Pacotes.pdf>> . Acesso em: 26 setembro. 2020

SOUSA, Vinicius. **Desenvolvimento de Software Dirigido por Caso de Uso.** Disponível em: . Acesso em: <[https://www.devmedia.com.br/desenvolvimento-de-software-dirigido-por-caso-de-uso/9148#:~:text=Atualmente%2C%20existe%20um%20artefato%20muito,de%20Uso%20\(Use%20Case\).](https://www.devmedia.com.br/desenvolvimento-de-software-dirigido-por-caso-de-uso/9148#:~:text=Atualmente%2C%20existe%20um%20artefato%20muito,de%20Uso%20(Use%20Case).>)> 03 maio. 2020

STEPPAT, N. **Bancos de dados não relacionais e o movimento NoSQL.** 2009. Disponível em: <<https://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/>> . Acesso em: 07 set. 2019.

TELES, Elaine. **Arduino: O que é? Pra que serve? Quais as possibilidades?.** Disponível em: <<https://medium.com/nossa-coletividad/arduino-o-que-%C3%A9-pra-que-serve-quais-as-possibilidades-efbd59d33491>> . Acesso em: 26 setembro. 2020

THOMSEN, Adilson. **Como programar o módulo ESP8266 NodeMCU**. Disponível em: <<https://www.filipeflop.com/blog/esp8266-nodemcu-como-programar/>> . Acesso em: 26 setembro. 2020