

**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 Nagila Zortea 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 vida é uma peça de teatro que não permite ensaios. Por isso, cante, ria, dance, chore e viva intensamente cada momento de sua vida, antes que a cortina se feche e a peça termine sem aplausos."

(Charlie Chaplin)

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. Definiu-se como objetivo deste trabalho o desenvolvimento de um aplicativo móvel, nomeado Velha Guarda, que consiste em apresentar uma solução para o controle automatizado no processo de produção de cerveja artesanal, levando em conta uma produção de até cem litros, 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 visa aplicar 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 à prova d'água para medir a temperatura em todos os processos de produção da cerveja, desde a fervura, resfriamento, fermentação até a maturação, avaliando os dados obtidos por meio da temperatura no sensor.

Palavras-chave: Automação. Aplicativo. Serviço Web

ABSTRACT

Keywords: Automation. App. Web Service

LISTA DE FIGURAS

Figura 01 - Widget.....	13
Figura 02 - HTML/CSS análogos em Flutter.....	13
Figura 03 - Cupertino e material.....	14
Figura 04 - NoSQL.....	16
Figura 05 - Provedores de login.....	18
Figura 06 - OneSignal - push notification.....	19
Figura 07 - Sensor DS18B20.....	20
Figura 08 - ESP12.....	21
Figura 09 -Módulo ESP8266 NodeMCU V3.....	22
Figura 10 - Módulo ESP8266 Pinagem.....	22
Figura 11 - BreadBoard.....	23
Figura 12 - Arduino Preferências.....	23
Figura 13 - URL's adicionais de gerenciadores de placa.....	24
Figura 14 - Gerenciador de placas.....	24
Figura 15 - Instalação do pacote da placa.....	25
Figura 16 - Escolha da placa.....	25
Figura 17 - Porta COM.....	26
Figura 18 - Incluir biblioteca.....	27
Figura 19 - Dallas temperature.....	27
Figura 20 - OneWire.....	28
Figura 21 - Instalação Android Studio.....	29
Figura 22 - Tela inicial Android Studio.....	29
Figura 23 - Flutter console.....	30
Figura 24 - Configurando a variável de ambiente.....	31
Figura 25 - Flutter Doctor.....	31
Figura 26 - Android Studio Plugins.....	32
Figura 27 - Plugins Dart e Flutter.....	33
Figura 28 - Montagem dos componentes.....	34
Figura 29 - Estrutura montada.....	34
Figura 30 - Código Arduino.....	35
Figura 31 - Monitor de temperatura.....	38
Figura 32 - Produção da cerveja.....	39
Figura 33 - Produto final.....	39
Figura 34 - Caso de uso geral.....	41
Figura 35 - Caso de uso efetuar login.....	42
Figura 36 - Caso de uso módulo gerente.....	43
Figura 37 - Caso de uso módulo institucional.....	43

Figura 38 - Caso de uso módulo funcionário.....	44
Figura 39 - Diagrama de classe.....	55
Figura 40 - Diagrama de sequência manter login.....	56
Figura 41 - Diagrama de sequência manter produto.....	57
Figura 42 - Diagrama de sequência manter funcionário.....	58
Figura 43 - Diagrama de sequência manter relatório.....	59
Figura 44 - Diagrama de sequência manter relatório.....	60
Figura 45 - Diagrama de atividade fazer login.....	61
Figura 46 - Diagrama de atividade módulo funcionário.....	62
Figura 47 - Diagrama de atividade módulo relatório.....	63
Figura 48 - Diagrama de atividade módulo relatório.....	64
Figura 49 - Diagrama de Componentes e Implantação.....	65
Figura 50 - Abertura.....	66
Figura 51 - Autenticação.....	67
Figura 52 - Inicial.....	68
Figura 53 - Categoria de produto.....	69
Figura 54 - Produto.....	70
Figura 55 - Carrinho de compra.....	71
Figura 56 - Forma de pagamento.....	72
Figura 57- Sobre.....	73
Figura 58 - Automação.....	74
Figura 59 - Temperatura.....	75
Figura 60 - Cronômetro.....	76
Figura 61 - Receitas.....	77

LISTA DE TABELAS

Tabela 01 - Ferramentas integradas.....	17
Tabela 02 - Descrição de caso de uso manter produto.....	45
Tabela 03 - Descrição do caso de uso efetuar login.....	47
Tabela 04 - Descrição do caso de uso manter funcionário.....	48
Tabela 05 - Descrição do caso de uso manter venda.....	51
Tabela 06 - Descrição do caso de uso manter relatório.....	53

LISTA DE QUADROS

Quadro 01 – Prós e contras entre desenvolvimento híbrido e nativo.....	11
--	----

LISTA DE ABREVIATURA E SIGLAS

APP – *Application*

API – *Application Programming Interface*

CSS – *Cascading Style Sheet*

HTML - *HyperText Markup Language*

IDE – *Integrated Development Environment*

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*

SUMÁRIO

1 INTRODUÇÃO.....	09
2 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS.....	10
2.1 METODOLOGIAS DE DESENVOLVIMENTO.....	10
2.1.1 Desenvolvimento nativo.....	10
2.1.2 Desenvolvimento de aplicativo web	10
2.1.3 Desenvolvimento de aplicativo híbrido	11
2.1.4 Análise das metodologias de desenvolvimento.....	11
2.2 FLUTTER	11
2.2.1 Pacotes Flutter.....	14
3 BANCO DE DADOS NÃO RELACIONAL.....	14
3.1 TIPOS DE BANCOS DE DADOS NÃO RELACIONAL.....	14
3.1.1 Análise das metodologias de banco de dados não relacional.....	15
3.2 FIREBASE.....	15
3.2.1 OneSignal.....	17
4 HARDWARE.....	20
4.1 SENSOR INTELIGENTE.....	20
4.1.1 Sensor DS18B20.....	20
4.2 PROTOCÓLO ONE-WIRE.....	21
4.3 Módulo Wifi ESP8266 NodeMCU V3 CP2102 ESP-12E.....	22
4.3.1 ESP8266.....	22
4.3.2 NodeMCU V3.....	22
4.3.3 BreadBoard.....	23
5 CONFIGURAÇÕES	23
5.1 ARDUINO	23
5.1.1 Placa	23
5.1.2 Sensor	26
5.1.3 OneWire.....	28
5.2 ANDROID STUDIO.....	28
5.2.1 Flutter SDK.....	30
5.3 MONTANDO OS COMPONENTES.....	33
6 LEVANTAMENTO DE REQUISITOS.....	39
6.1 METODOLOGIA DE ANÁLISE.....	40

6.2 DIAGRAMAS UML.....	40
6.3 Descrição do gerenciamento.....	40
6.4 Descrição dos Usuários.....	40
6.5 Módulos.....	40
6.5.1 Perspectiva do Produto: Módulo Institucional.....	40
6.5.2 Perspectiva do Produto: Módulo Gerente.....	40
6.5.3 Perspectiva do Produto: Módulo Funcionário.....	41
6.6 DIAGRAMAS DE CASO DE USO.....	41
6.6.1 Diagrama de Caso de uso geral.....	41
6.6.2 Efetuar login.....	41
6.6.3 Módulo Gerente.....	43
6.6.4 Módulo Institucional.....	43
6.6.5 Módulo Funcionário.....	44
6.7 Diagrama de classe	55
6.8 Diagrama de entidade relacionamento.....	55
6.9 Diagrama de sequência.....	56
6.9.1 Manter login.....	56
6.9.2 Manter Produto.....	57
6.9.3 Manter funcionário.....	58
6.9.4 Manter relatório.....	59
6.10 Diagrama de atividade.....	60
6.10.1 Módulo Institucional.....	61
6.10.2 Fazer Login.....	61
6.10.3 Módulo Funcionário.....	62
6.10.4 Relatório.....	63
6.10.5 Módulo Gerente.....	64
6.11 Diagrama de Componentes e Implantação.....	65
7 DESENVOLVIMENTO DO APLICATIVO	65
7.1 ABERTURA.....	66
7.2 AUTENTICAÇÃO.....	67
7.3 INICIAL.....	68
7.4 CATEGORIA DE PRODUTOS.....	69
7.5 PRODUTO.....	70
7.6 CARRINHO DE COMPRA.....	71
7.7 FORMA DE PAGAMENTO.....	72
7.8 SOBRE.....	73
7.9 AUTOMAÇÃO.....	74
7.10 TEMPERATURA.....	75
7.11 CRONÔMETRO.....	76
7.12 RECEITA.....	77

8 CONCLUSÃO E TRABALHOS FUTUROS	78
--	-----------

REFERÊNCIAS	79
--------------------------	-----------

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 é possível 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.

Finalmente, no oitavo 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 é 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. O Quadro 1 apresenta algumas vantagens e desvantagens do desenvolvimento híbrido em relação ao desenvolvimento nativo (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"> ● Escrever o código uma vez: o código é escrito uma vez, e pode ser utilizado em qualquer plataforma que tenha suporte pela ferramenta. ● Mais rápido, equipes menores: o tempo de desenvolvimento e a quantidade de profissionais diminui consideravelmente pois não é preciso desenvolver aplicações individuais para cada plataforma. ● Conhecimento e ferramentas: não é necessário conhecimento, nem ferramentas específicas para cada plataforma. ● Investimento econômico: devido a redução de tempo, da equipe, do conhecimento e das ferramentas, o desenvolvimento do aplicativo requer um investimento menor. 	<ul style="list-style-type: none"> ● Potencial nativo: não é possível utilizar totalmente os recursos específicos de cada plataforma, pois a interface de desenvolvimento é a mesma para qualquer plataforma. ● Desempenho: 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. ● Interface específica: 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.

Fonte: Autor

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, o Flutter 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).

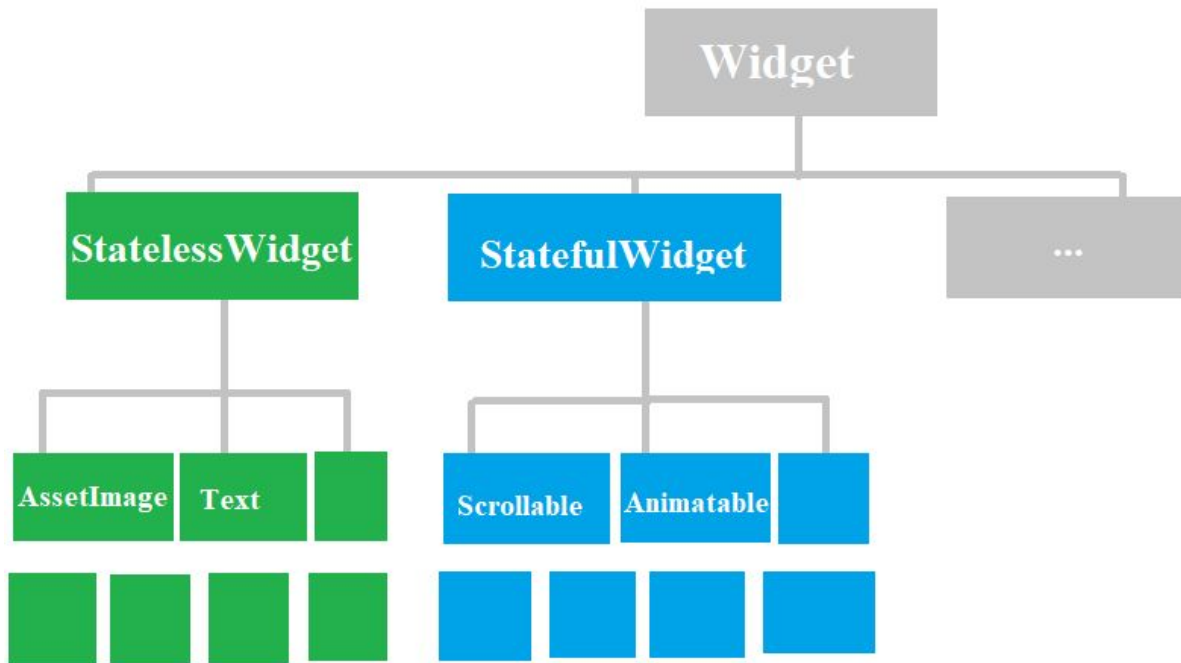
Outra vantagem que torna indispensável é o *Just in time*, a recompilação do código no dispositivo, enquanto o aplicativo está rodando, a aplicação não perde o estado de desenvolvimento. Isso gera um ciclo de desenvolvimento muito rápido e produtivo, possibilitando o recarregamento expresso do aplicativo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

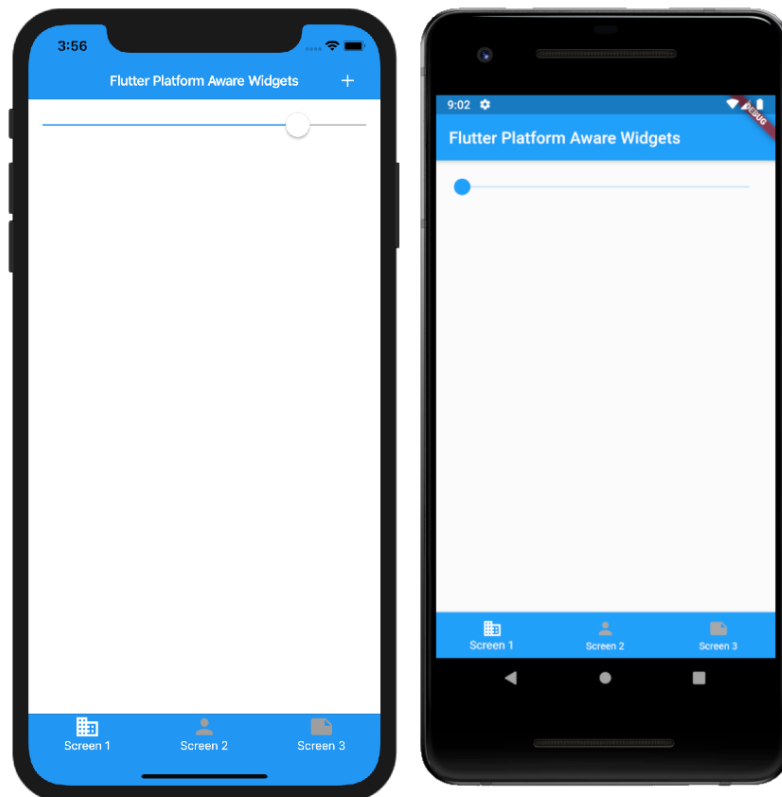
Figura 02 - HTML/CSS análogos em Flutter

<pre> <div class = "greybox" > Lorem ipsum </div> .greybox { background-color: #e0e0e0; width: 320px; height: 240px; font: 900 24px Georgia; } </pre>	<pre> var container = Container(child: Text("Lorem ipsum", style: TextStyle(fontSize: 24.0 fontWeight: FontWeight.w900, fontFamily: "Georgia",),), width: 320.0, height: 240.0, color: Colors.grey[300]); </pre>
--	--

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 dispõem 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

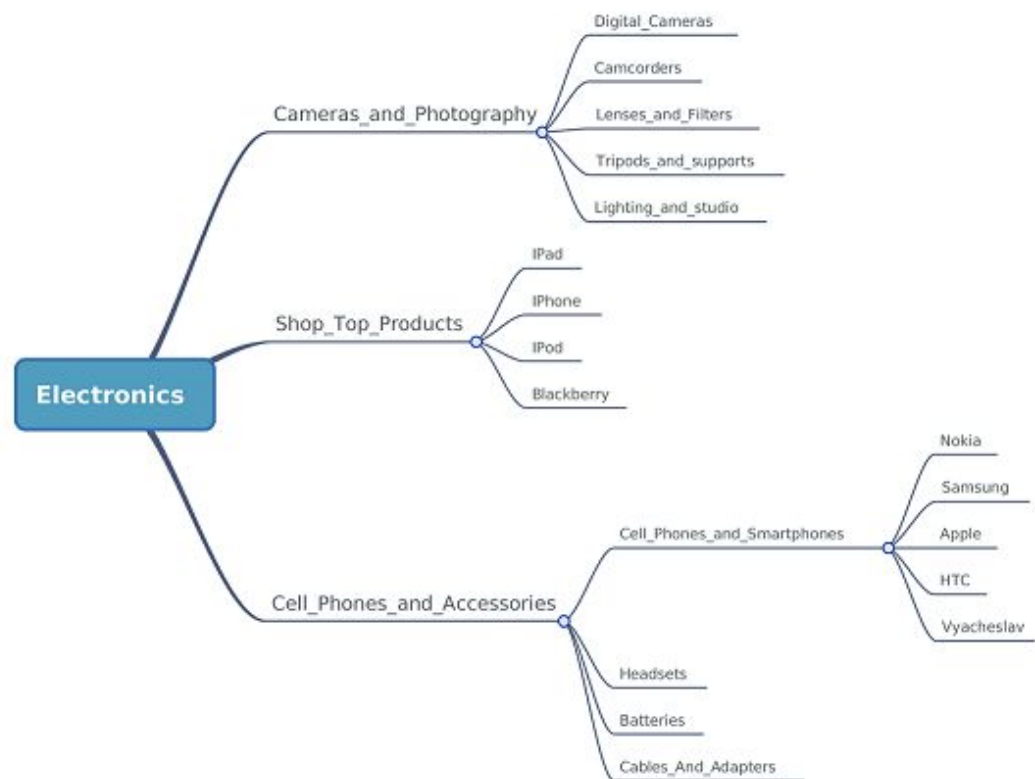
Como qualquer parte da arquitetura de uma aplicação, é preciso avaliar qual é a melhor opção de tecnologia de banco de dados a ser utilizada. Aderindo a um banco de dados não relacional grande parte da responsabilidade da consistência dos dados fica a cargo da aplicação. O banco fica mais simples, escalável 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 fique restrito 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 Tabela 01.









Tabela 1 - Ferramentas integradas

Desenvolvimento	Qualidade
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)

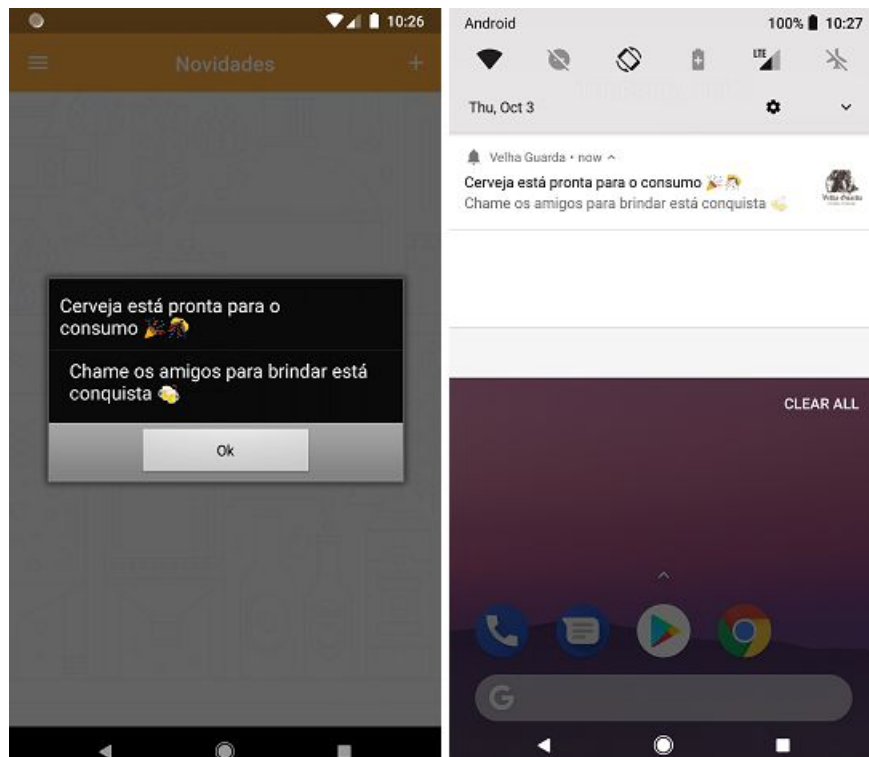
A principal ideia de utilizar o Firebase no projeto é por ser uma plataforma móvel completa e robusta de fácil uso, com a viabilidade de trabalhar *offline* e sincronizar automaticamente ao retornar a conexão, utiliza sincronização em tempo de execução, tem plano gratuito e atende muitos cenários utilizando 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, email 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, umidade, velocidade, entre outros, 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 $0,5^{\circ}\text{C}$. 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.

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

NodeMCU ESP-12 development kit V1.0

PIN DEFINITION

The diagram illustrates the pin definitions for the NodeMCU ESP-12 development kit V1.0. The board is shown with its components and pin headers. The pin definitions are as follows:

Pin Header	Function	GPIO Pin
TOUT	ADC0	A0
RESERVED	RSV	RSV
RESERVED	RSV	RSV
SDD3	GPI010	SD3
SDD2	GPI09	SD2
SDD1	MOSI	SD1
SDCMD	CS	CMD
SDD0	MISO	SD0
SDCLK	SCLK	CLK
GND	GND	GND
3.3V	3V3	3V3
EN	EN	EN
RST	RST	RST
GND	GND	GND
Vin	Vin	Vin
D0	GPI016	USER
D1	GPI05	
D2	GPI04	
D3	GPI00	FLASH
D4	GPI02	TXD1
3V3	3.3V	
GND	GND	
D5	GPI014	HSCLK
D6	GPI012	HMISO
D7	GPI013	RXD2
D8	GPI015	TXD2
RX	GPI03	RXD0
TX	GPI01	TXD0
GND	GND	
3V3	3.3V	

The diagram also shows the physical layout of the board, including the USB Type-C port, USB Type-A port, DC power jack, and reset button. The pin headers are labeled with their functions and corresponding GPIO pins.

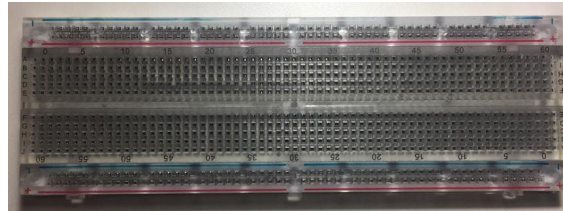
Arduining.com

22

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: Autor

5 CONFIGURAÇÕES

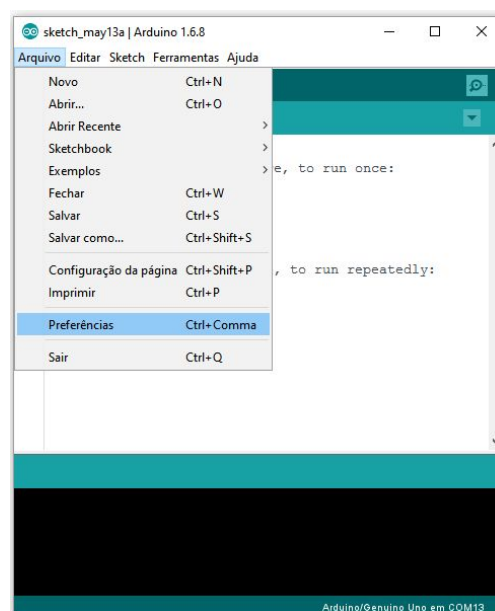
A descrição da configuração do sistema, juntamente com a montagem dos componentes e instalações de softwares é tratado neste capítulo.

5.1 ARDUINO

5.1.1 Placa

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

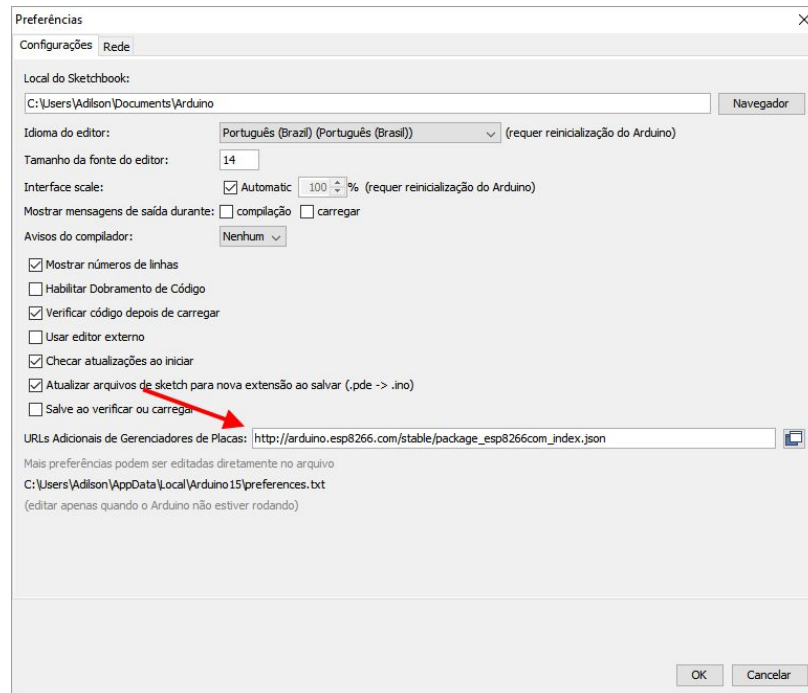
Figura 12 - Arduino Preferências



Fonte: Autor

Procure pelo campo URLs adicionais de Gerenciadores de Placas e passe o LINK: http://arduino.esp8266.com/stable/package_esp8266com_index.json para adicionar as placas da família ESP como demonstra a Figura 13.

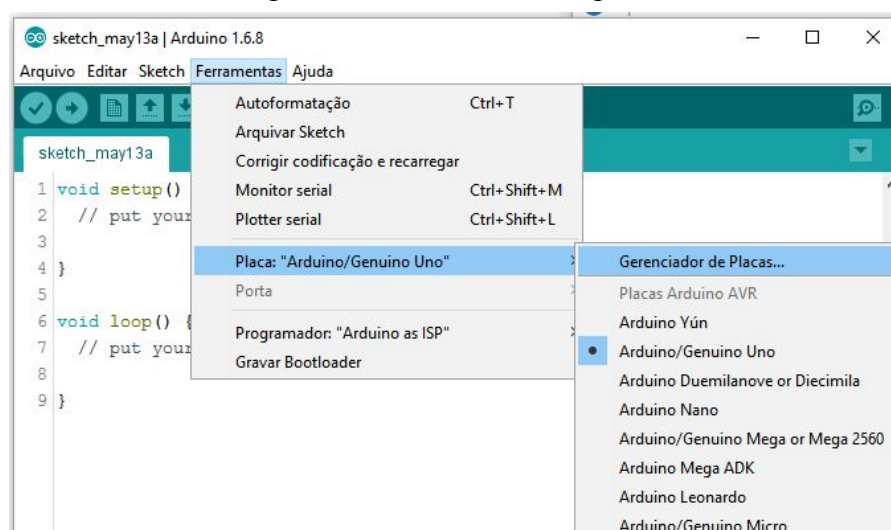
Figura 13 -URL's adicionais de gerenciadores de placas



Fonte: Autor

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

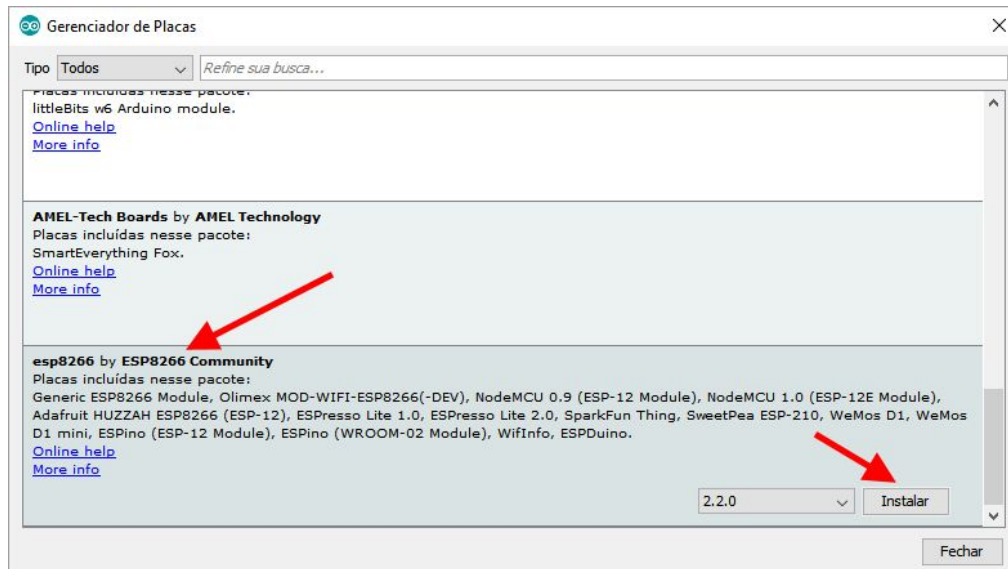
Figura 14 - Gerenciador de placas



Fonte: Autor

Depois 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.

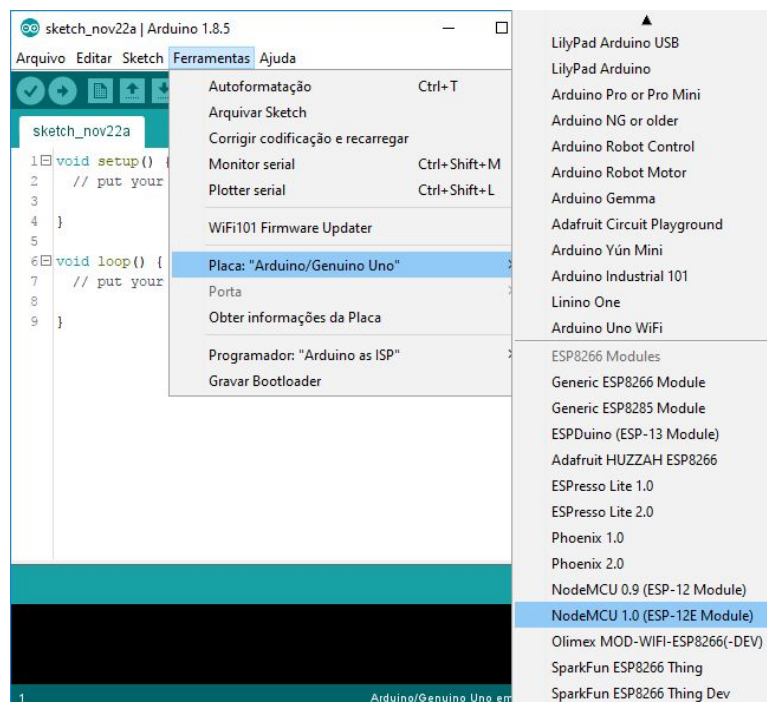
Figura 15 - Instalação do pacote da placa



Fonte: Autor

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

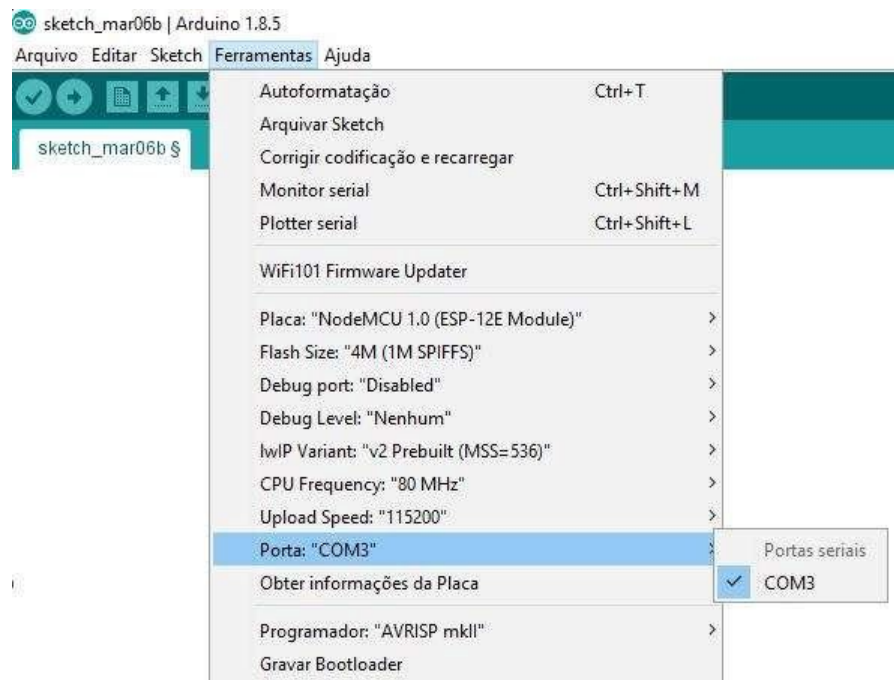
Figura 16 - Escolha da placa



Fonte: Autor

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



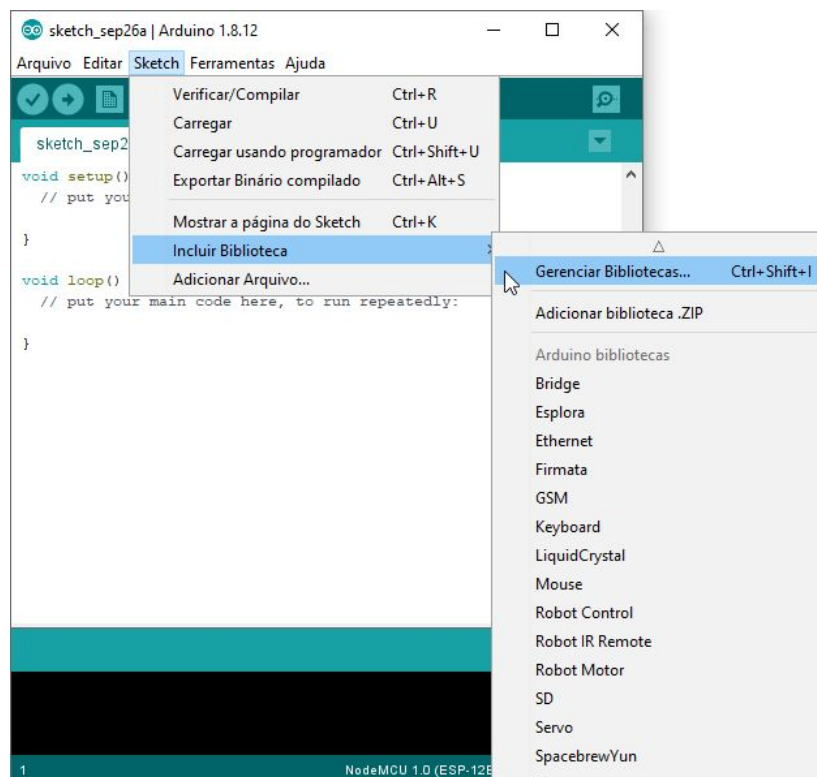
Fonte: Autor

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

Agora selecione no menu superior Sketch, Incluir Biblioteca, Gerenciar Bibliotecas conforme apresenta a Figura 18.

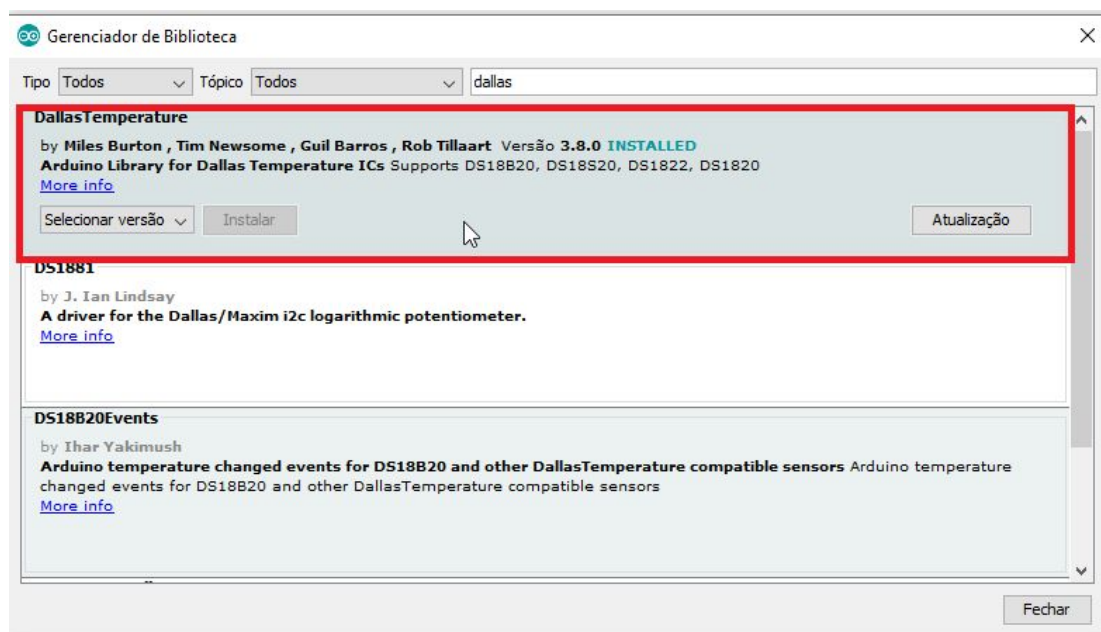
Figura 18 - Incluir biblioteca



Fonte: Autor

Depois 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

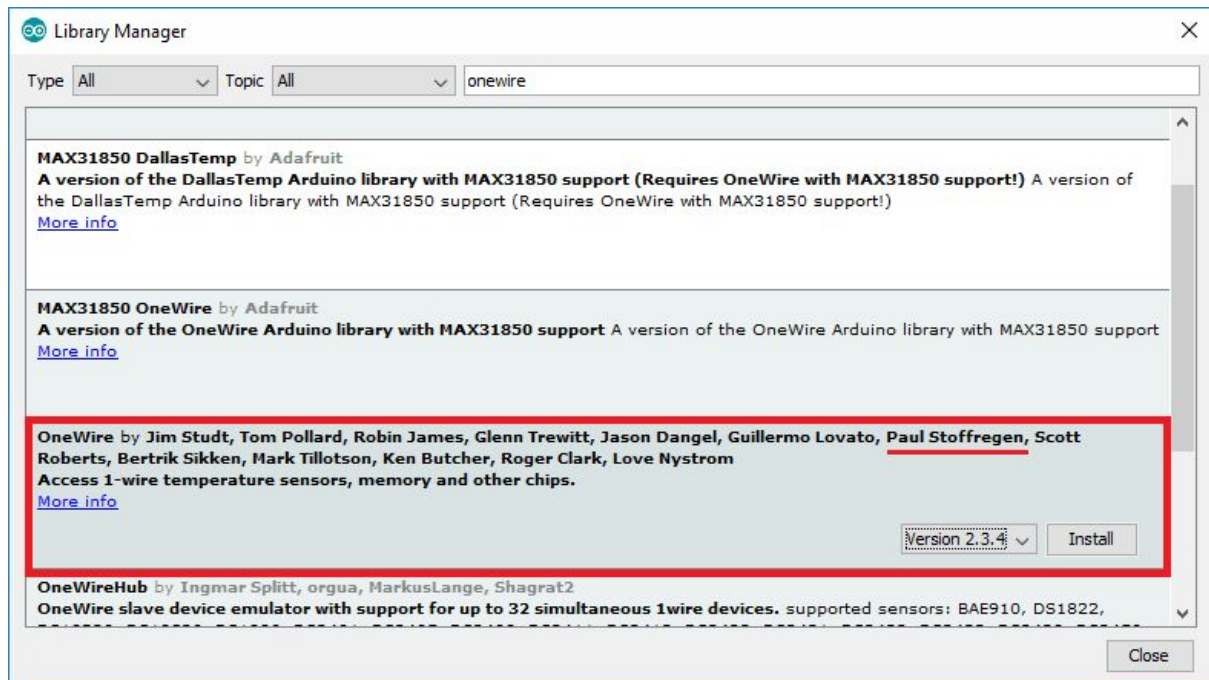


Fonte: Autor

5.1.3 OneWire

Procure 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: Autor

5.2 ANDROID STUDIO

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

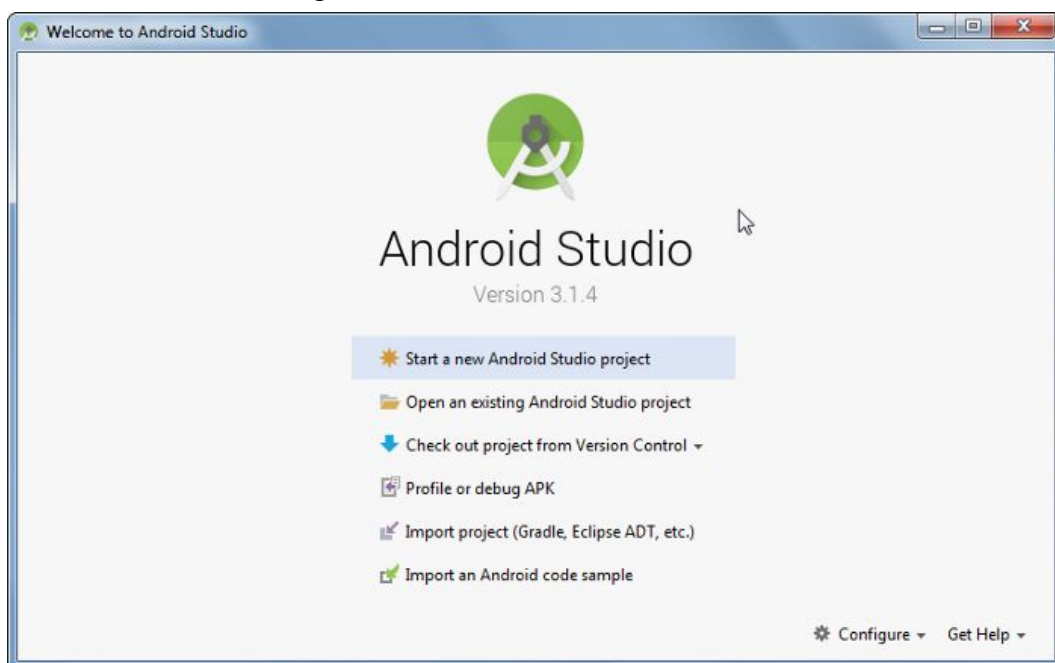
Figura 21 - Instalação Android Studio



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

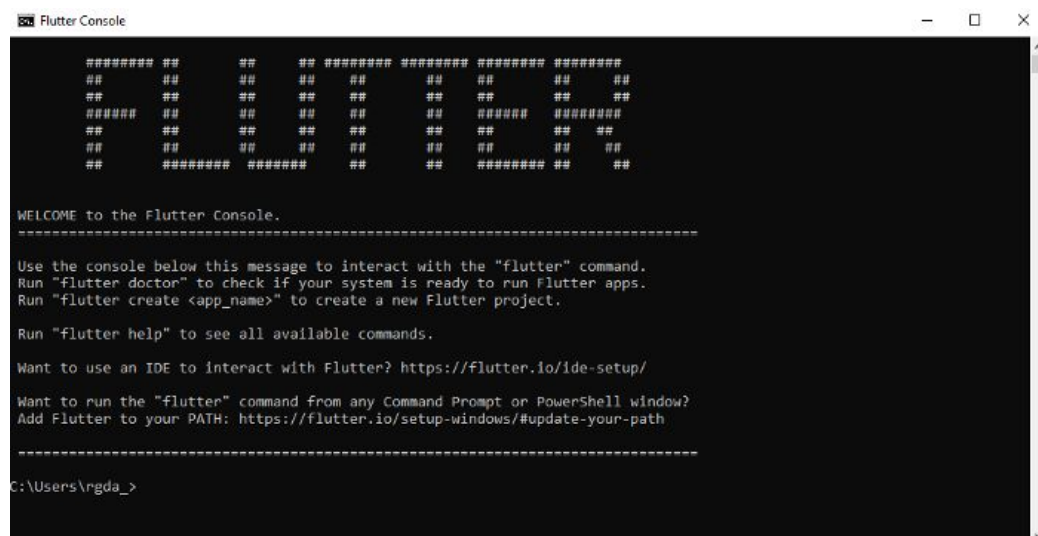


Fonte: Baseada na imagem original de (MUNIZ, 2018)

5.2.1 Flutter SDK

Baixe a SDK do Flutter no link <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\rgda_>
```

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).

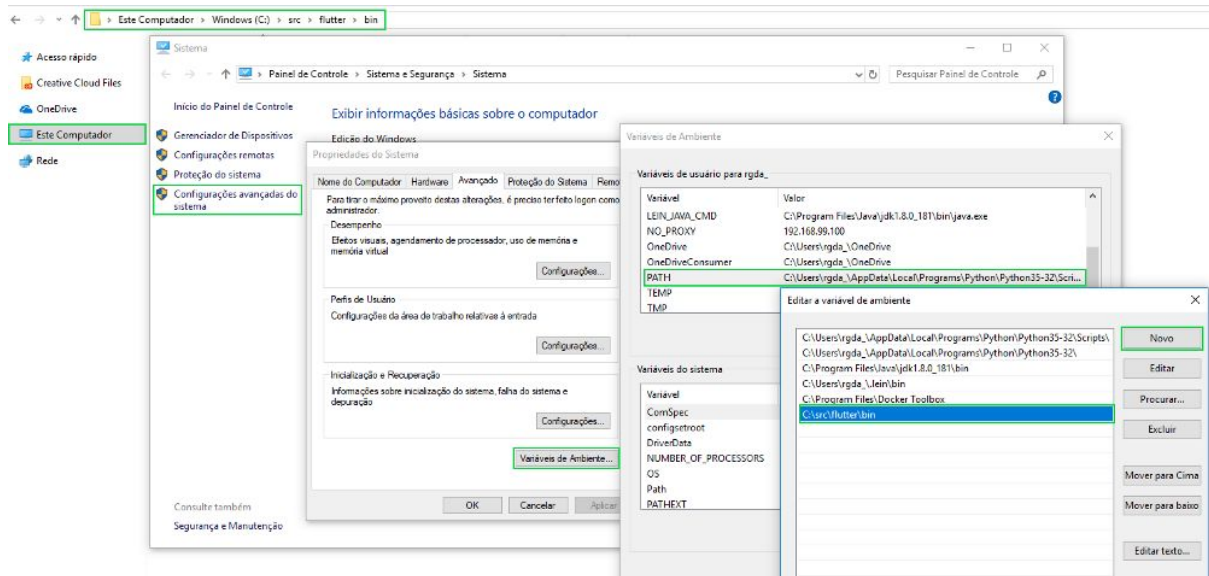
Para isto precisamos seguir um plano, o primeiro passo é copiar o caminho até o diretório `\bin`, presente na pasta do flutter.

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`.

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

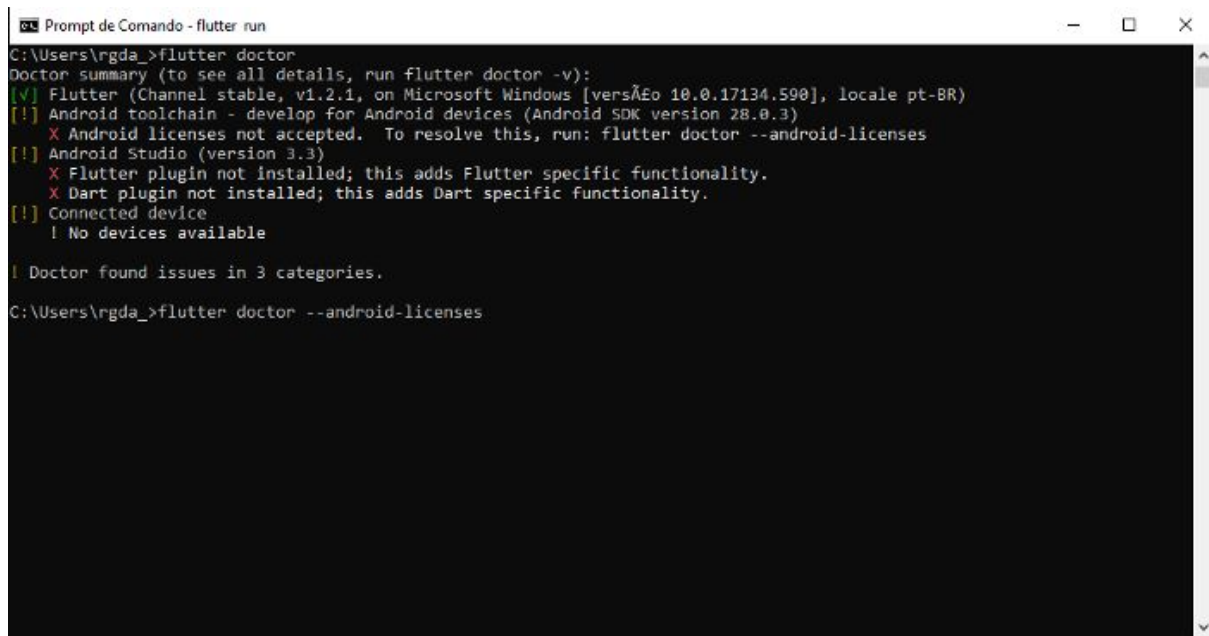
Figura 24 - Configurando a variável de ambiente



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

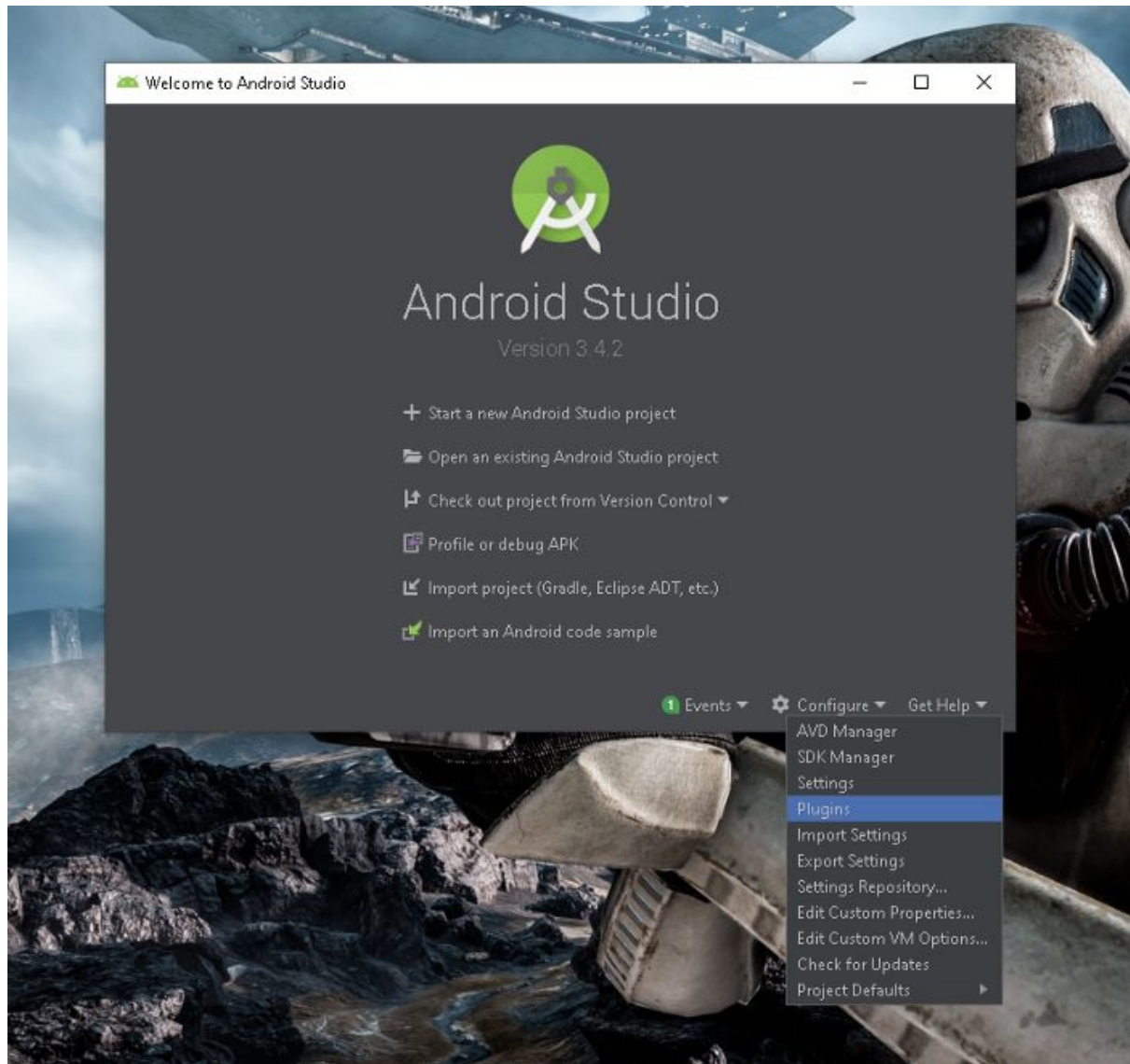


Fonte: Baseada na imagem original de (DIAS, 2019)

O flutter doctor é encarregado por apurar se existem dependências do flutter 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.

Agora vamos acrescentar alguns *plugins* e extensões no Android Studio para que este seja apto a desenvolver em flutter como demonstra a Figura 26.

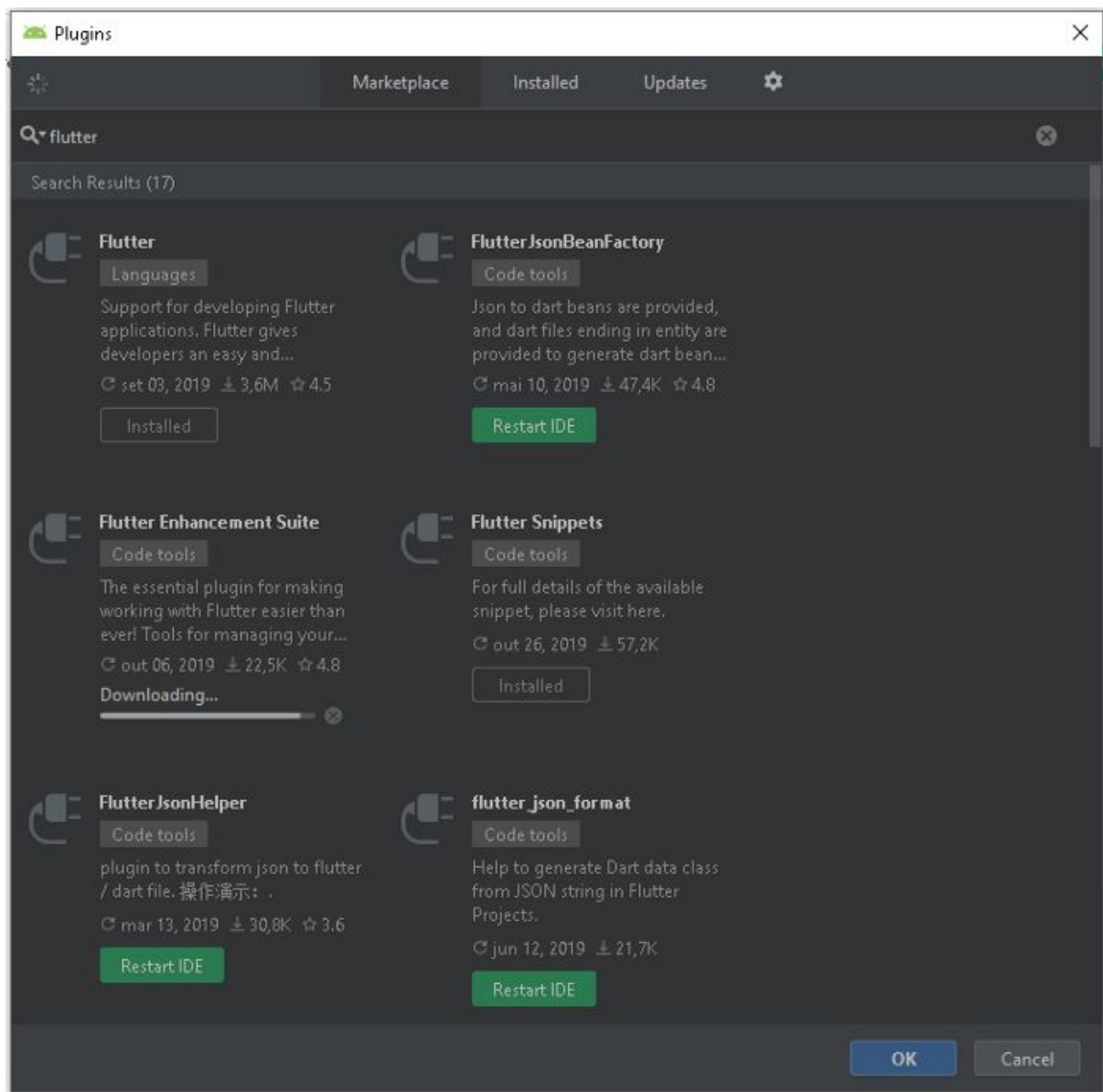
Figura 26 - Android Studio Plugins



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

Instale os plugins do Dart e Flutter para que tudo funcione da forma correta como demonstra a Figura 27.

Figura 27 - Plugins Dart e Flutter

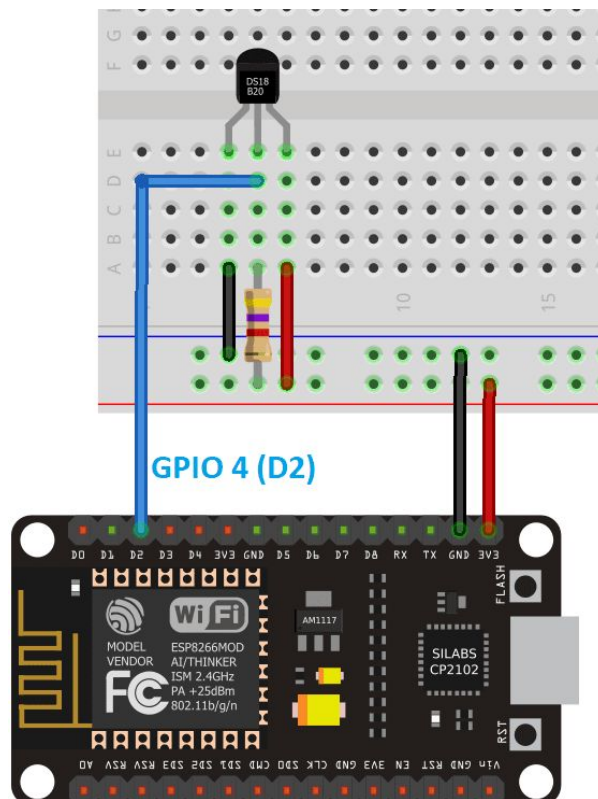


Fonte: Baseada na imagem original de (SESSA, 2020)

5.3 MONTANDO OS COMPONENTES

Primeiramente precisamos 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.

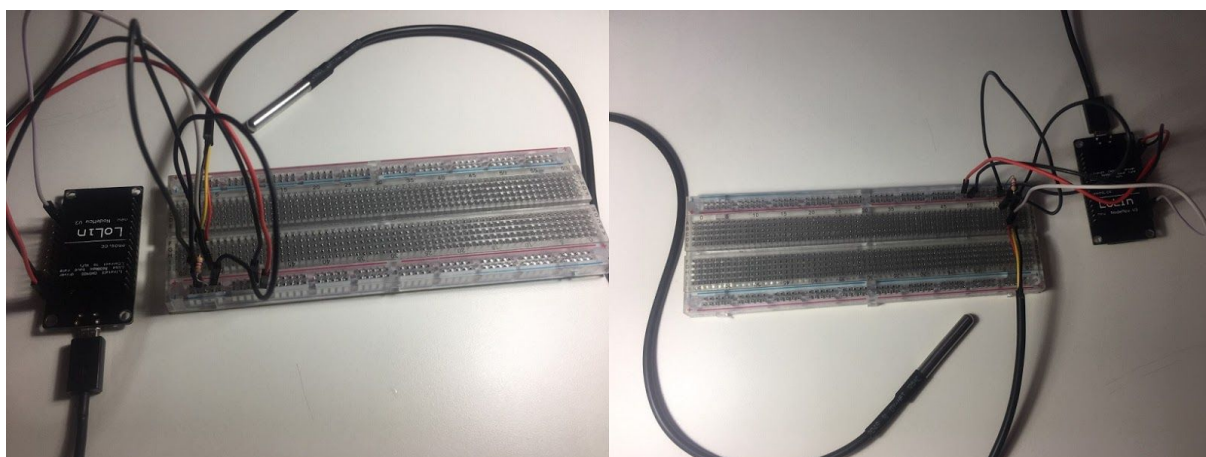
Figura 28 - Montagem dos componentes



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: Baseada na imagem original de (MAGNABOSCO, 2020)

Depois de 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
3  #include <OneWire.h>
4
5  #include <DallasTemperature.h>
6
7  // GPIO where the DS18B20 is connected to
8  const int oneWireBus = 4;
9
10 // Setup a oneWire instance to communicate with any OneWire devices
11 OneWire oneWire(oneWireBus);
12
13 // Pass our oneWire reference to Dallas Temperature sensor
14 DallasTemperature sensors( & oneWire);
15
16 float tempSensor1;
17
18 uint8_t sensor1[8] = {
19     0x28,
20     0xEE,
21     0xD5,
22     0x64,
23     0x1A,
24     0x16,
25     0x02,
26     0xEC
27 };
28 /*Put your SSID & Password*/
29 const char * ssid = "Net Virtua 67"; // Enter SSID here
30 const char * password = "3511598700"; //Enter Password here
31
32 ESP8266WebServer server(80);
33
34 void setup() {
35     // Start the Serial Monitor
36     Serial.begin(115200);
37     // Start the DS18B20 sensor
38     sensors.begin();
39
40     Serial.println("Connecting to ");
41     Serial.println(ssid);
42
43     //connect to your local wi-fi network
44     WiFi.begin(ssid, password);
45
46     //check wi-fi is connected to wi-fi network
47     while (WiFi.status() != WL_CONNECTED) {
48         delay(1000);
49         Serial.print(".");
50     }
51     Serial.println("");
52     Serial.println("WiFi connected..!");
53     Serial.print("Got IP: ");
54     Serial.println(WiFi.localIP());
55
56     server.on("/", handle_OnConnect);
57     server.onNotFound(handle_NotFound);
58
59     server.begin();
60     Serial.println("HTTP server started");
61 }
62

```

```

63 void loop() {
64     server.handleClient();
65     sensors.requestTemperatures();
66     float temperatureC = sensors.getTempCByIndex(0);
67     Serial.print(temperatureC);
68     Serial.println("°C");
69     delay(5000);
70 }
71
72 void handle_OnConnect() {
73     sensors.requestTemperatures();
74     tempSensor1 = sensors.getTempCByIndex(0); // Gets the values of the
        temperature
75     server.send(200, "text/html", SendHTML(tempSensor1));
76 }
77
78 void handle_NotFound() {
79     server.send(404, "text/plain", "Not found");
80 }
81
82 String SendHTML(float tempSensor1) {
83     String ptr = "<!DOCTYPE html>";
84     ptr += "<html>";
85     ptr += "<head>";
86     ptr += "<title>Monitor de temperatura</title>";
87     ptr += "<meta charset='UTF-8'/>";
88     ptr += "<meta name='viewport' content='width=device-width, initial
        -scale=1 0'>";
89     ptr += "<link href='https://fonts.googleapis.com/css?family=Open+Sans
        :300,400,600' rel='stylesheet'>";
90     ptr += "<link rel='icon' href='favicon.ico' type='image/x-icon'>";
91     ptr += "<style>";
92     ptr += "html { font-family: 'Open Sans', sans-serif; display: block;
        margin: 0px auto; text-align: center; color: #444444; }";
93     ptr += "body { margin-top: 50px; } ";
94     ptr += "h1 { margin: 50px auto 30px; } ";
95     ptr += ".side-by-side { display: table-cell; vertical-align: middle
        ; position: relative; }";
96     ptr += ".text { font-weight: 600; font-size: 19px; width: 200px; }";
97     ptr += ".temperature { font-weight: 300; font-size: 50px; padding-right:
        15px; }";
98     ptr += ".living-room .temperature { color: #F29C1F; }";
99     ptr += ".superscript { font-size: 17px; font-weight: 600; position:
        absolute; right: -5px; top: 15px; }";
100    ptr += ".data { padding: 10px; }";
101    ptr += ".container { display: table; margin: 0 auto; }";
102    ptr += ".icon { width: 82px; }";
103    ptr += "</style>";
104    ptr += "</head>";
105    ptr += "<body>";
106    ptr += "<h1>Monitor de temperatura</h1>";
107    ptr += "<div class='container'>";
108    ptr += "<div class='data living-room'>";
109    ptr += "<div class='side-by-side icon'>";

```



```

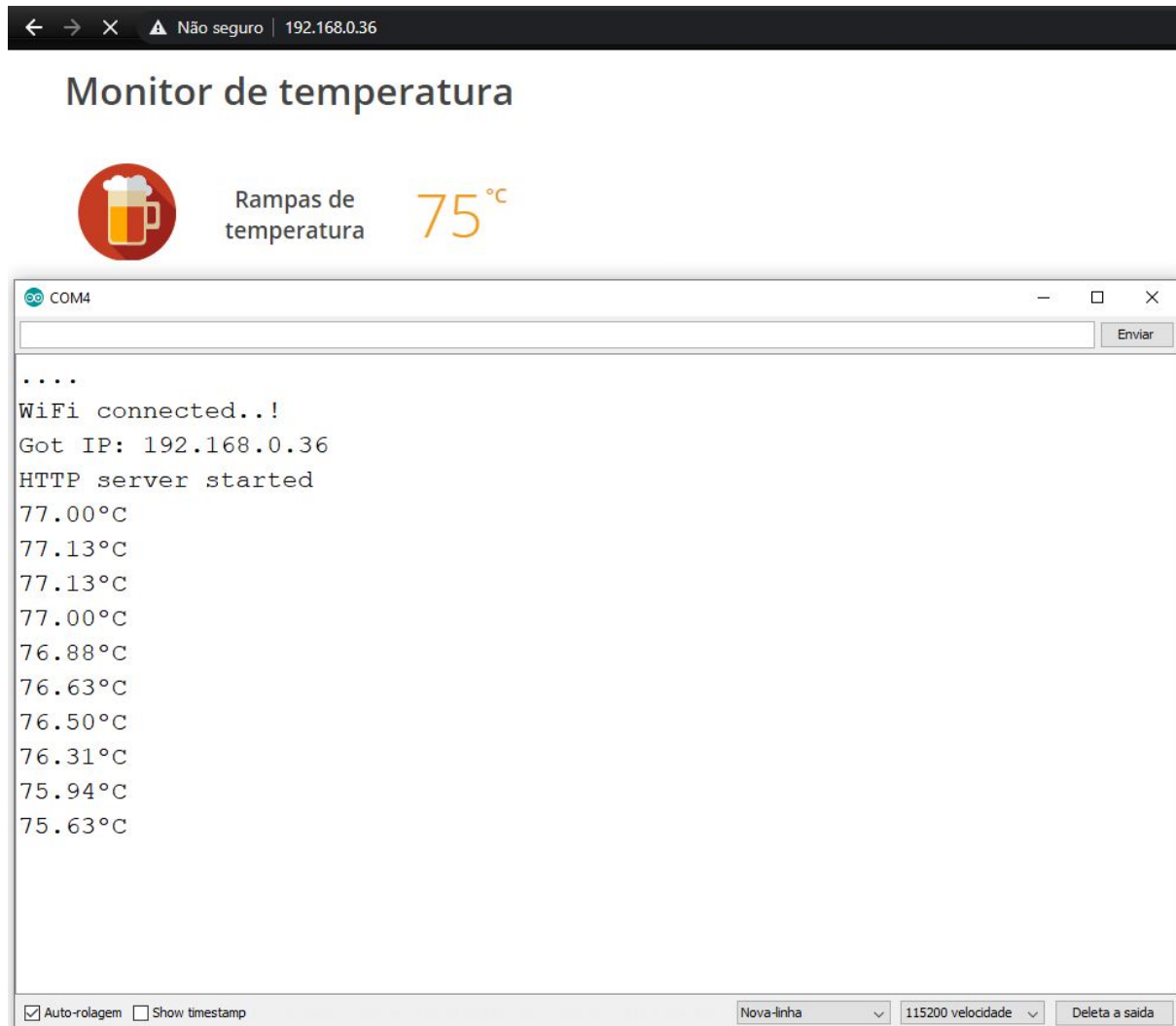
110 ptr += "<svg xmlns='http://www.w3.org/2000/svg' viewBox='0 0 297 297'>"
111 ptr += "<circle cx='148.5' cy='148.5' r='148.5' fill='#c63c22'/>";
112 ptr += "<path d='M296.703 139.216l-96.66-96.678-104.931 199.82 54.626
54.626C231.181 296.318 297 230.1 297 148.5c0-3.119-.108-6.212-.297
-9.284z' fill='#9e231d'/>";
113 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'/>";
114 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
0-1.745.073-2.589.208V90.71158.777 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'/>";
115 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'/>";
116 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>";
117 ptr += "</div>";
118 ptr += "<div class='side-by-side text'>Rampas de temperatura</div>";
119 ptr += "<div class='side-by-side temperature'>";
120 ptr += (int) tempSensor1;
121 ptr += "<meta http-equiv='refresh' content='2'>";
122 ptr += "<span class='superscript'>&deg;C</span></div>";
123 ptr += "</div>";
124 ptr += "</div>";
125 ptr += "</body>";
126 ptr += "</html>";
127 return ptr;
128 }

```

Fonte: Autor

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: Baseada na imagem original de (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: Baseada na imagem original de (MAGNABOSCO, 2020)

E por último foi feita a 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: Baseada na imagem original de (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 (*Unified Modeling Language*).

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.

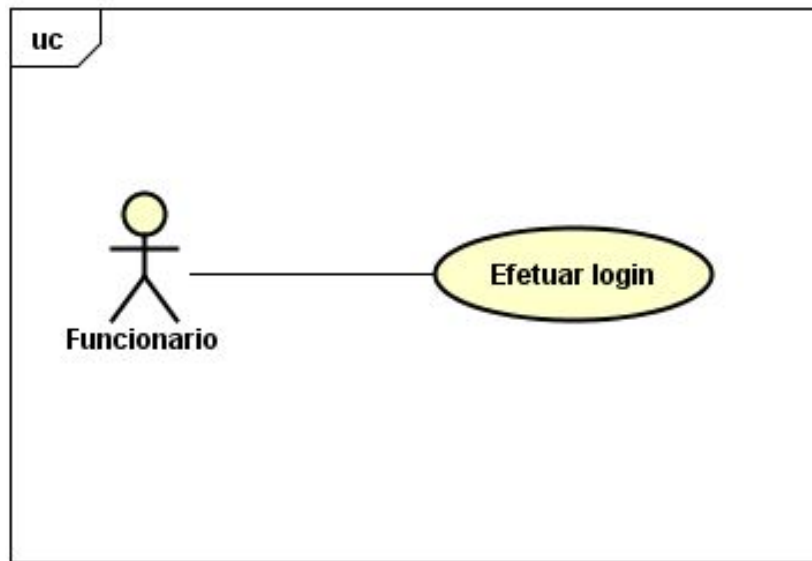
6.5 Módulos

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 etc. 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

Figura 35 - Caso de uso efetuar login

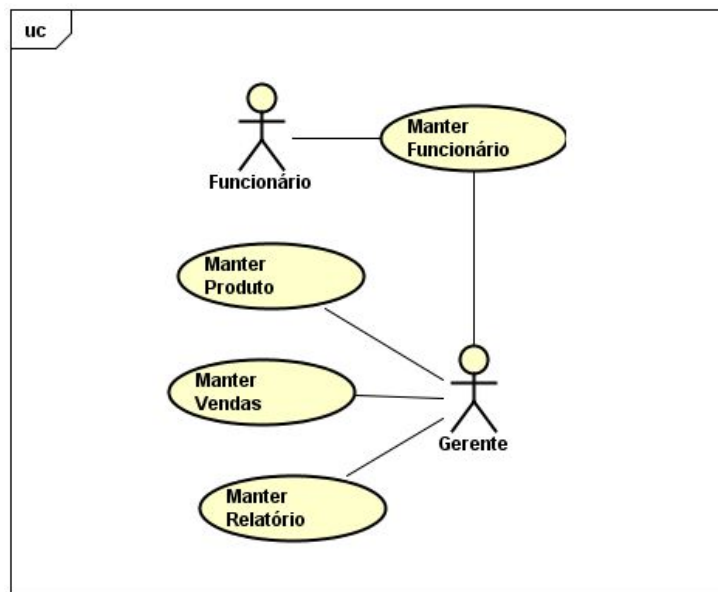


Fonte: Autor

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

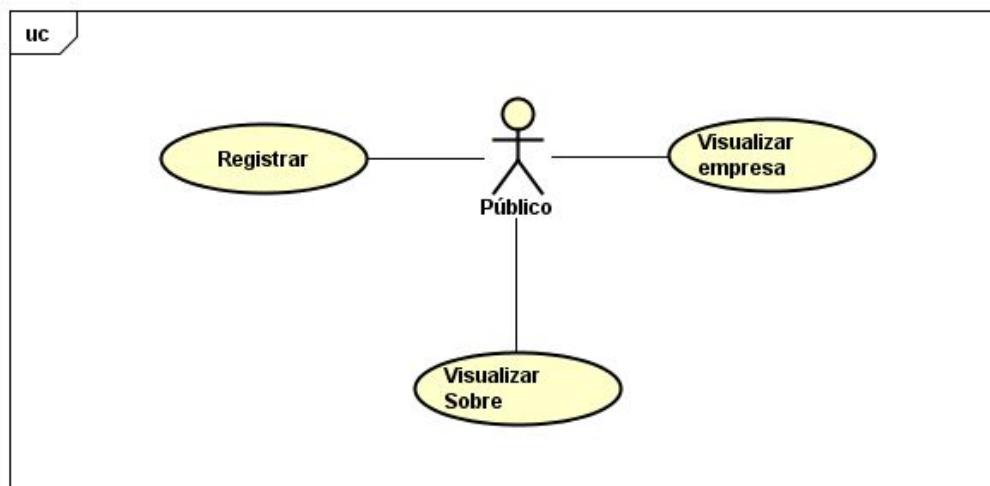


Fonte: Autor

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

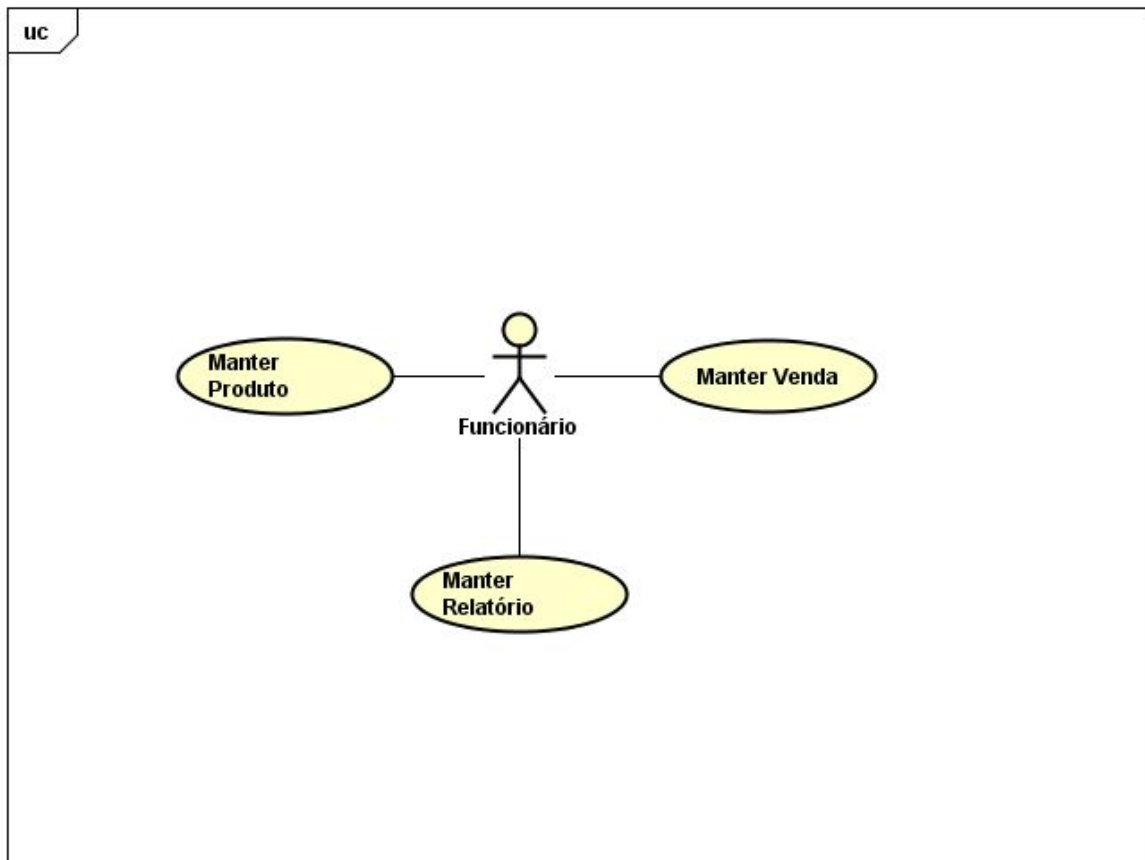


Fonte: Autor

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 na tabela 04

Figura 38 - Caso de uso módulo funcionário



Fonte: Autor

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

	Ator		Sistema	
1	O caso de uso inicia quando o ator solicita a manutenção de produtos			
		2	Sistema oferece a interface de manutenção de produtos.	
3	O Ator seleciona as operações de novo registro.			A1
		4	Ativa o formulário para registro.	
5	Preenche os dados do produto			
		6	Grava os dados.	
		7	Encerra o caso de uso.	
8	Informa os dados para a busca dos produtos.			
		9	Busca e mostra os dados dos produtos.	
10	Atualiza os dados previamente cadastrados.			
11	Confirma alteração.			A2
		12	Grava os dados.	
		13	Encerra o caso de uso.	

14	Ativa a interface para visualização dos produtos.			
		15	Busca e mostra os dados dos produtos.	
		16	Encerra o caso de uso.	
17	Informa os dados para busca dos produtos.			
		18	Busca e mostra os dados dos produtos.	
19	Ator seleciona a operação de exclusão. Utiliza “Excluir produtos”.			R1
		20	Excluir os produtos.	A3
		21	Encerra o caso de uso.	

A1	O Ator seleciona a operação de alteração. Utiliza “Alterar produtos” (linha 8). / seleciona a operação. Utiliza “Buscar produtos” (linha 14), seleciona a operação de exclusão. Utiliza “Excluir produtos (linha 17).
A2	Cancela caso de uso.
A3	Cancela caso de uso.
E1	O Ator não preenche corretamente todos os campo.
R1	Só e possível excluir um registro que seja feito por um Gerente.

Tabela 02- Descrição de caso de uso manter produto

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

Tabela 03- Descrição do caso de uso efetuar login

Caso de Uso:	Manter Funcionário
Ator(es):	Gerente, Funcionário.
Pré-condições:	O usuário (Gerente, Funcionário) deve estar logado no sistema.
Pós-condições:	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 campo.
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.
----	--

Tabela 04- Descrição do caso de uso manter funcionário

Caso de Uso:	Manter Venda
Ator(es):	Gerente, Funcionário
Pré-condições:	O usuário deve estar logado no sistema.
Pós-condições:	A venda deve ser cadastrada, consultada, alterada ou excluída

	Ator		Sistema	
1	O caso de uso inicia quando o ator solicita “Vender um produto”.			
		2	Sistema oferece a interface de venda do produto.	
3	O Ator seleciona as operações de novo registro.			A1
		4	Ativa o formulário para registro.	
5	Preenche os dados da venda.			
		6	Grava os dados.	
		7	Encerra o caso de uso.	
8	Informa os dados para a busca da venda.			
		9	Busca e mostra os dados da venda.	
10	Atualiza os dados previamente cadastrados.			
11	Confirma alteração.			A2
		12	Grava os dados.	
		13	Encerra o caso de uso.	

14	Ativa a interface para visualização das vendas.			
		15	Busca e mostra os dados da vendas.	
		16	Encerra o caso de uso.	
17	Informa os dados para busca das vendas.			
		18	Busca e mostra os dados das vendas.	
19	Ator seleciona a operação de exclusão. Utiliza “Excluir venda”.			R1
		20	Excluir a venda	A3
		21	Encerra o caso de uso.	

A1	O Ator seleciona a operação de alteração. Utiliza “Alterar venda” (linha 8). / seleciona a operação. Utiliza “Buscar venda” (linha 14), seleciona a operação de exclusão. Utiliza “Excluir venda (linha 17).
A2	Cancela caso de uso.
A3	Cancela caso de uso.
E1	O Ator não preenche corretamente todos os campo.
R1	Só e possível excluir um registro que seja feito por um Gerente.

Tabela 05- Descrição do caso de uso manter venda.

Caso de Uso:	Manter relatório
Ator(es):	Gerente, Funcionário
Pré-condições:	O usuário deve estar <i>logado</i> no sistema.
Pós-condições :	O usuário deve alterar, consultar e excluir os relatórios

	Ator		Sistema	
1	O caso de uso inicia quando o ator solicita “Manter relatório”.			
		2	Sistema oferece a interface dos relatórios.	
3	O Ator seleciona as operações de novo relatório.			A1
		4	Ativa o formulário para o relatório.	
5	Preenche os dados do relatório.			
		6	Grava os dados.	
		7	Encerra o caso de uso.	
8	Informa os dados para a busca do relatório			
		9	Busca e mostra os dados do relatório	
10	Atualiza os dados previamente cadastrados.			
11	Confirma alteração.			A2
		12	Grava os dados.	
		13	Encerra o caso de uso.	

14	Ativa a interface para visualização dos relatório.			
		15	Busca e mostra os dados do relatório.	
		16	Encerra o caso de uso.	
17	Informa os dados para busca do relatório.			
		18	Busca e mostra os dados do relatório.	
19	Ator seleciona a operação de exclusão. Utiliza “Excluir relatório”.			R1
		20	Excluir a relatório	A3
		21	Encerra o caso de uso.	

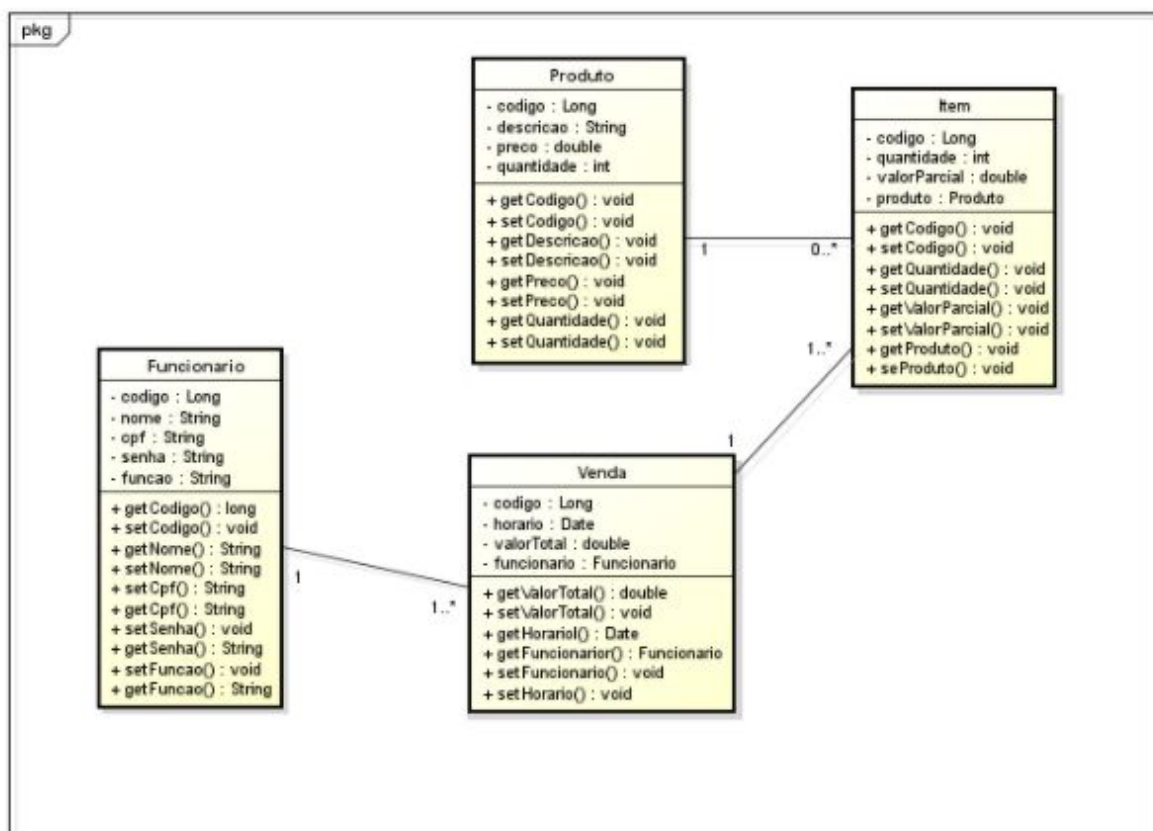
A1	O Ator seleciona a operação de alteração. Utiliza “Alterar relatório” (linha 8). / seleciona a operação. Utiliza “Buscar relatório” (linha 14), seleciona a operação de exclusão. Utiliza “Excluir relatório (linha 17)”.
A2	Cancela caso de uso.
A3	Cancela caso de uso.
E1	O Ator não preenche corretamente todos os campo.
R1	Só e possível excluir um relatório com o nível de acesso Gerente.

Tabela 06- Descrição do caso de uso Manter Relatório.

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: Autor

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.

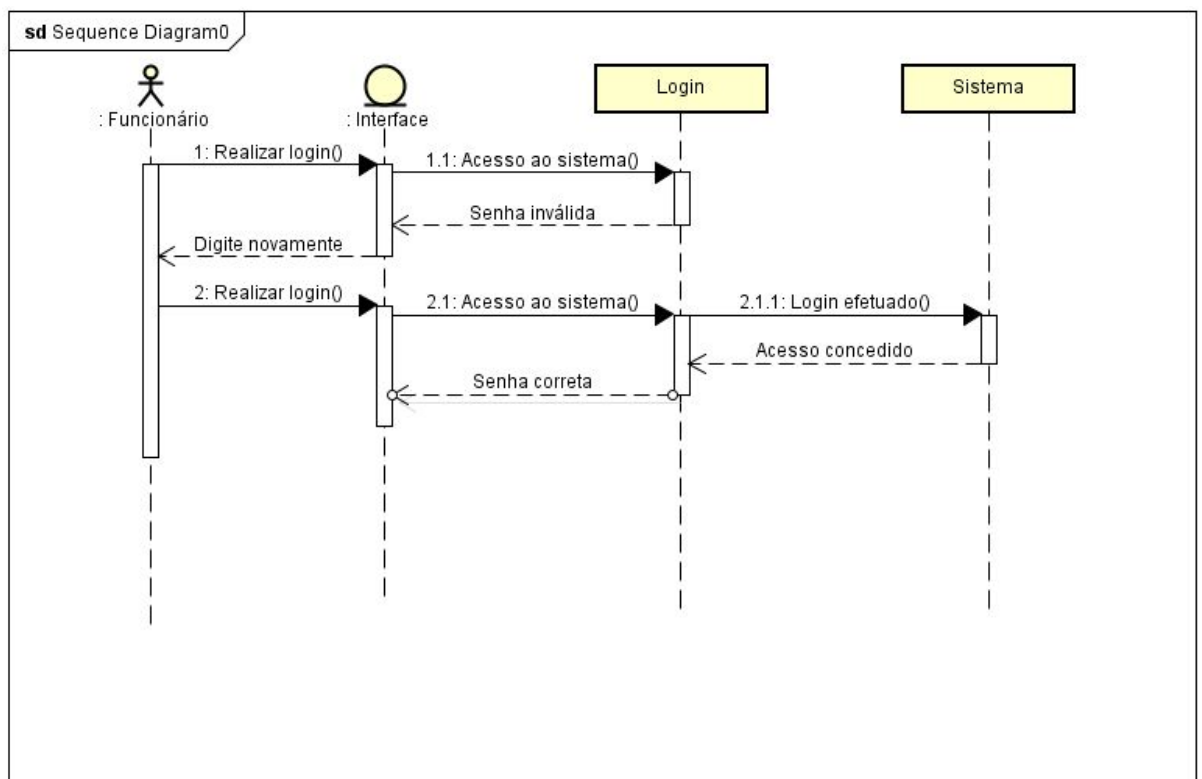
6.9 Diagrama de sequência

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

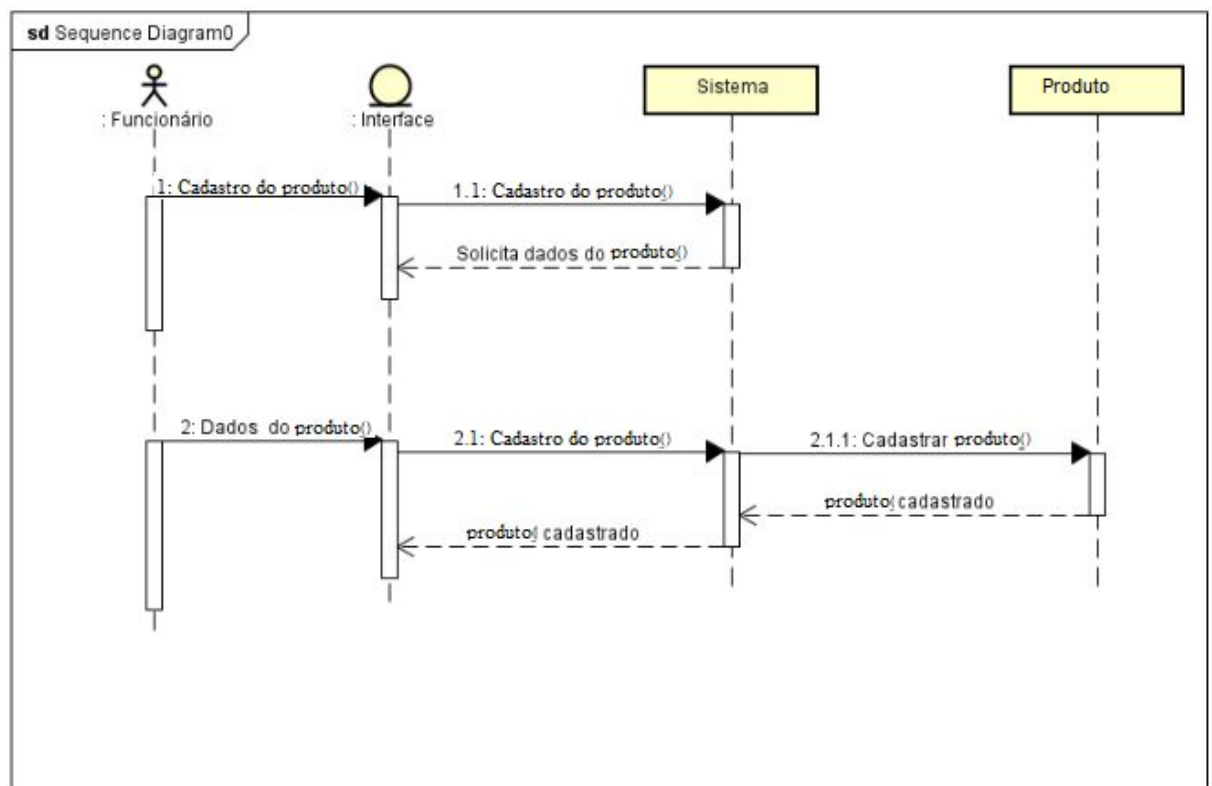


Fonte: Autor

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

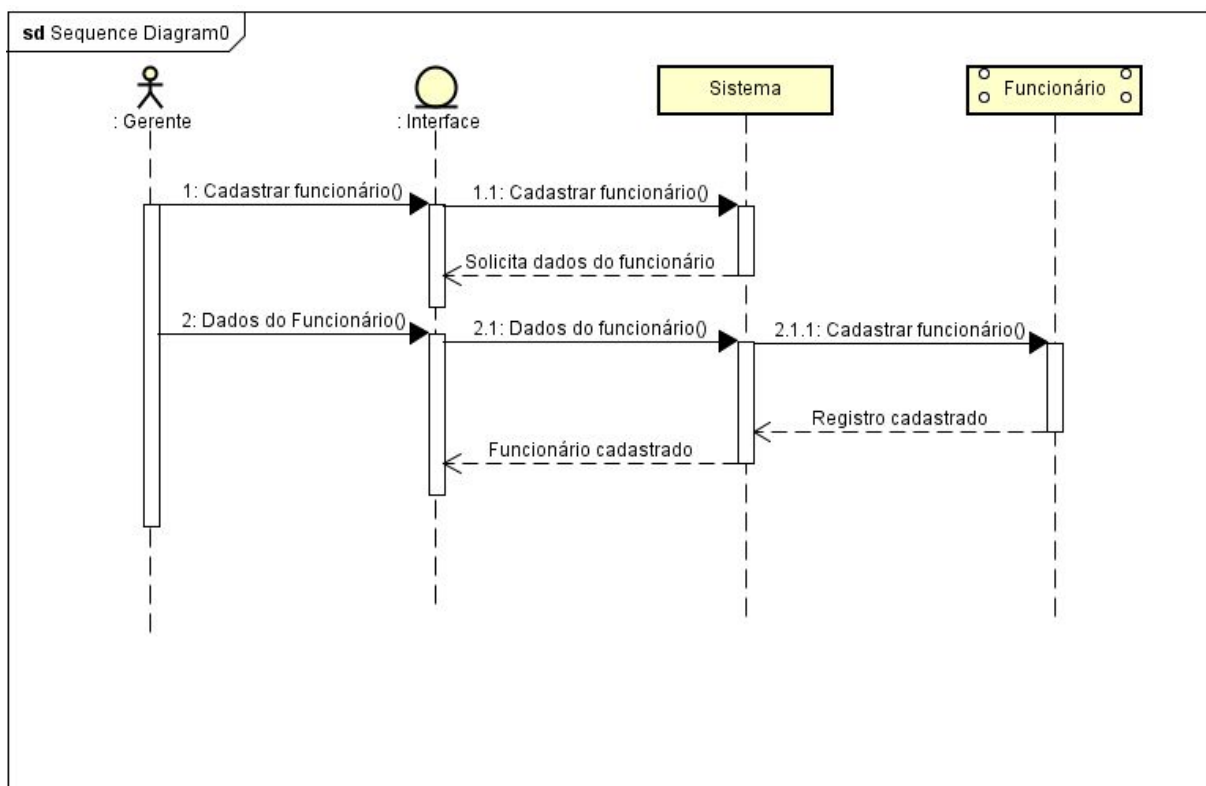


Fonte: Autor

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

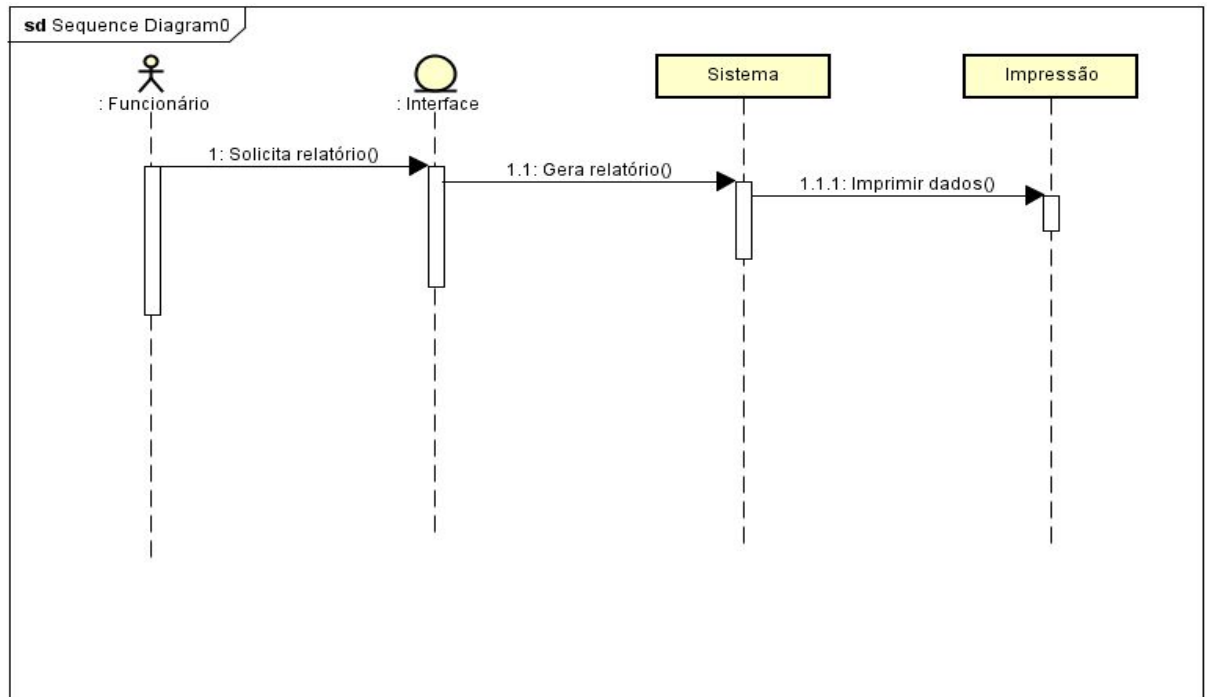


Fonte: Autor

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: Autor

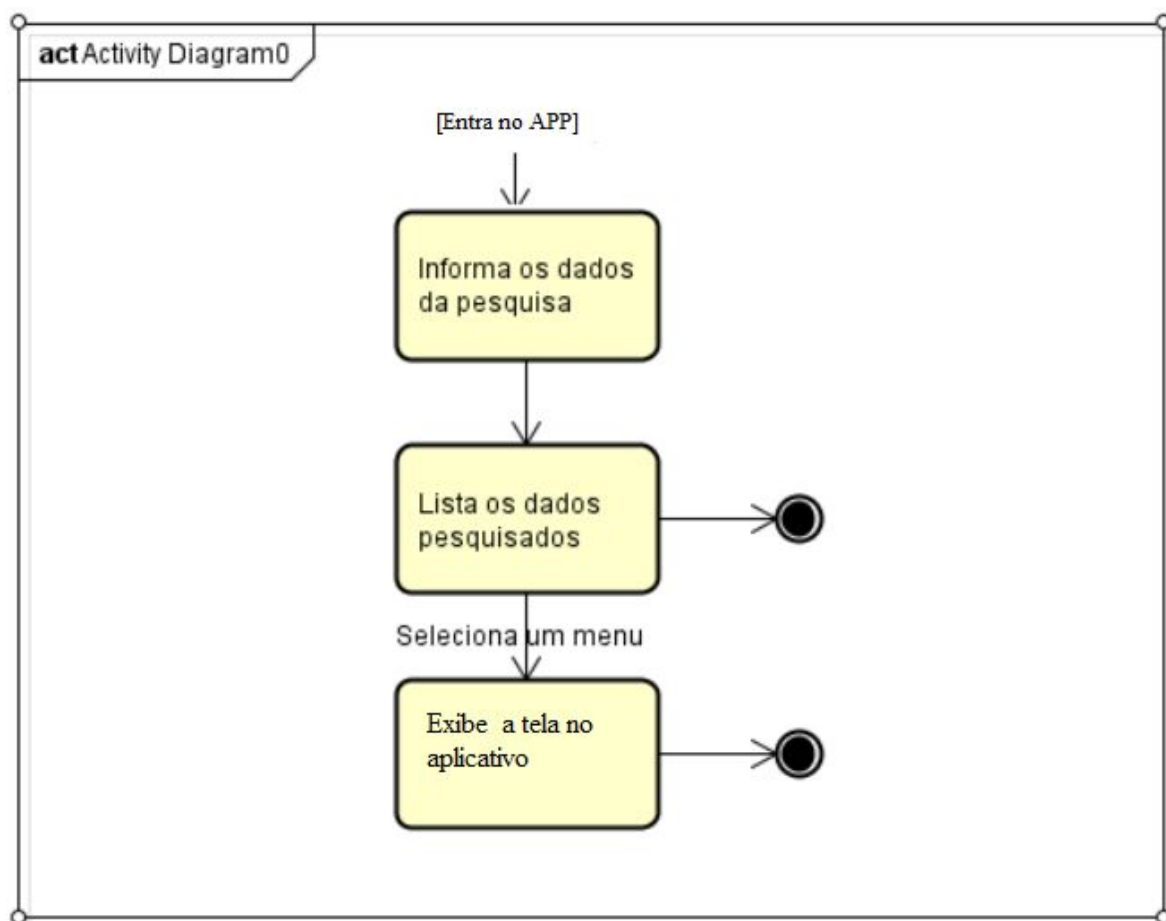
6.10 Diagrama de atividade

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 manter relatório

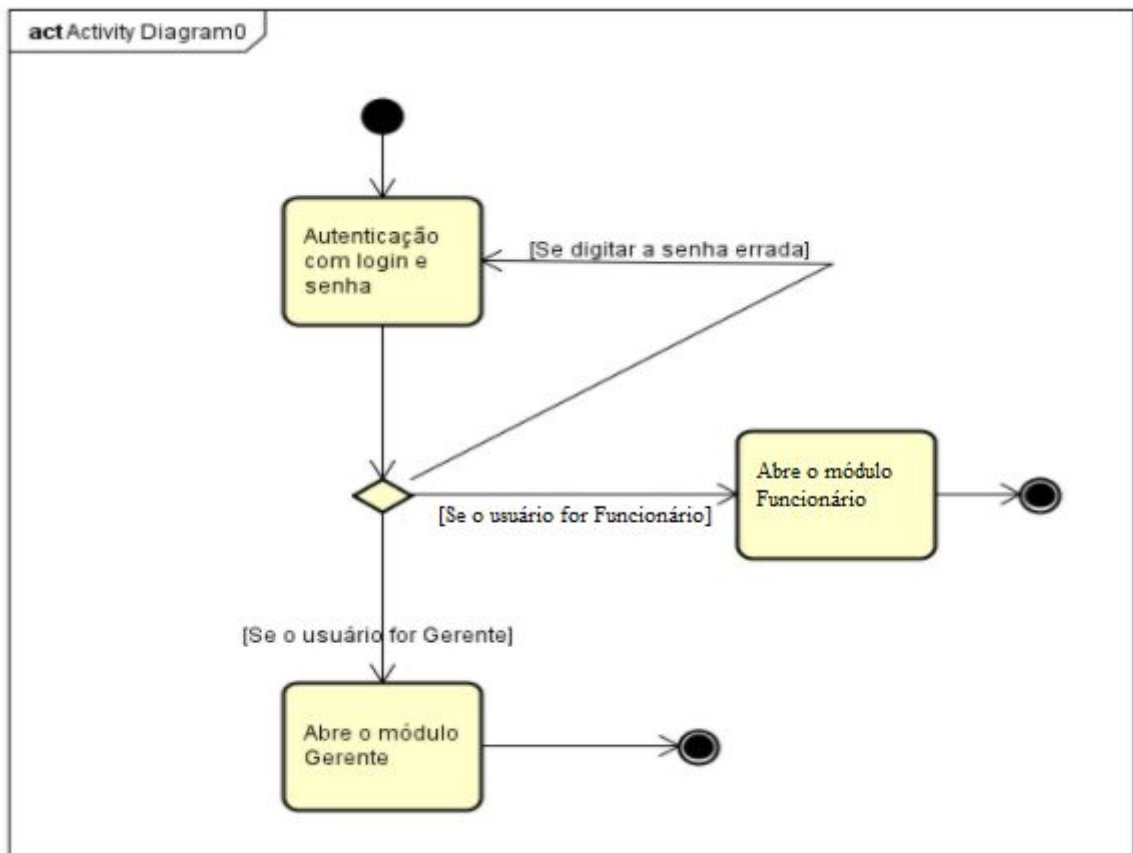


Fonte: Autor

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

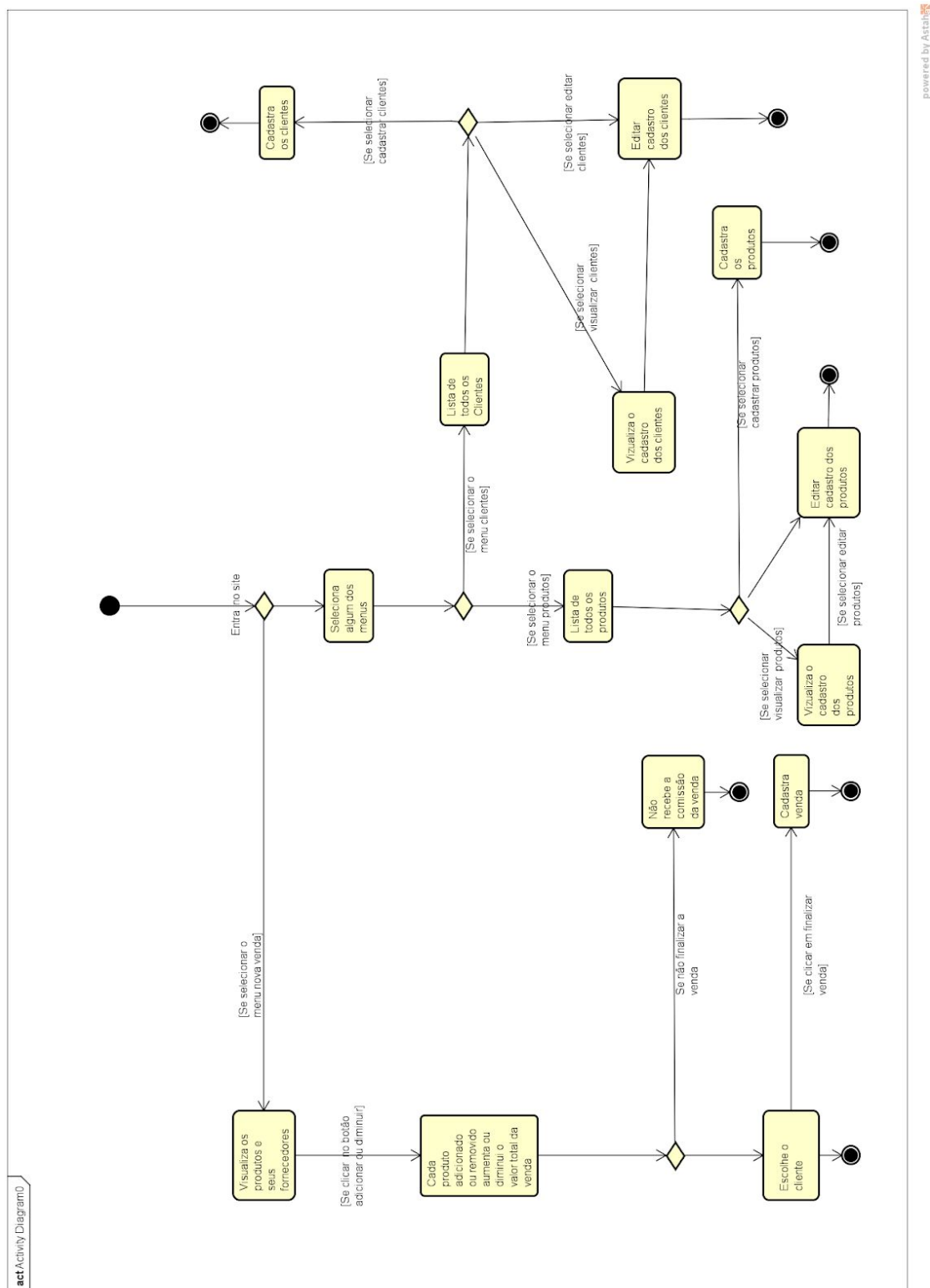


Fonte: Autor

6.10.3 Módulo Funcionário

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

Figura 46 - Diagrama de atividade módulo funcionário

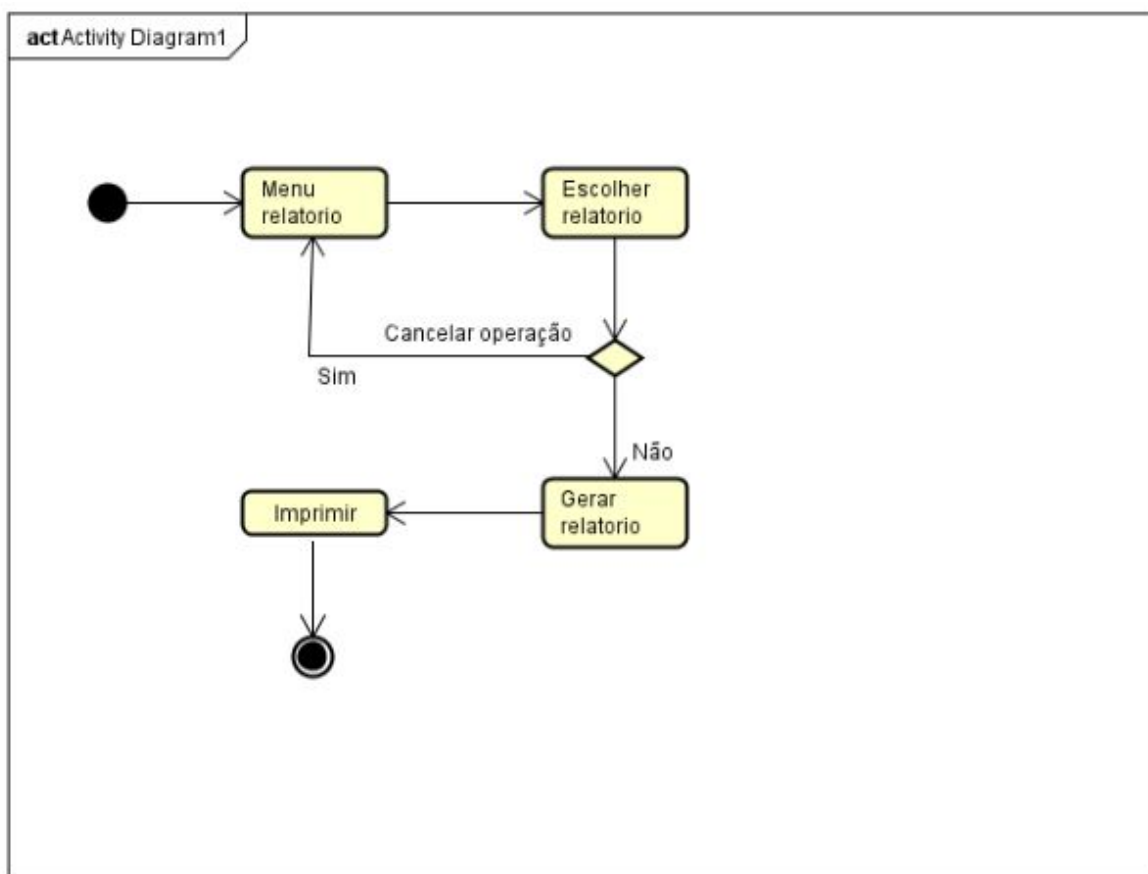


Fonte: Autor

6.10.4 Relatório

O diagrama de atividade do módulo relatório é apresentado na Figura 47.

Figura 47 - Diagrama de atividade módulo relatório

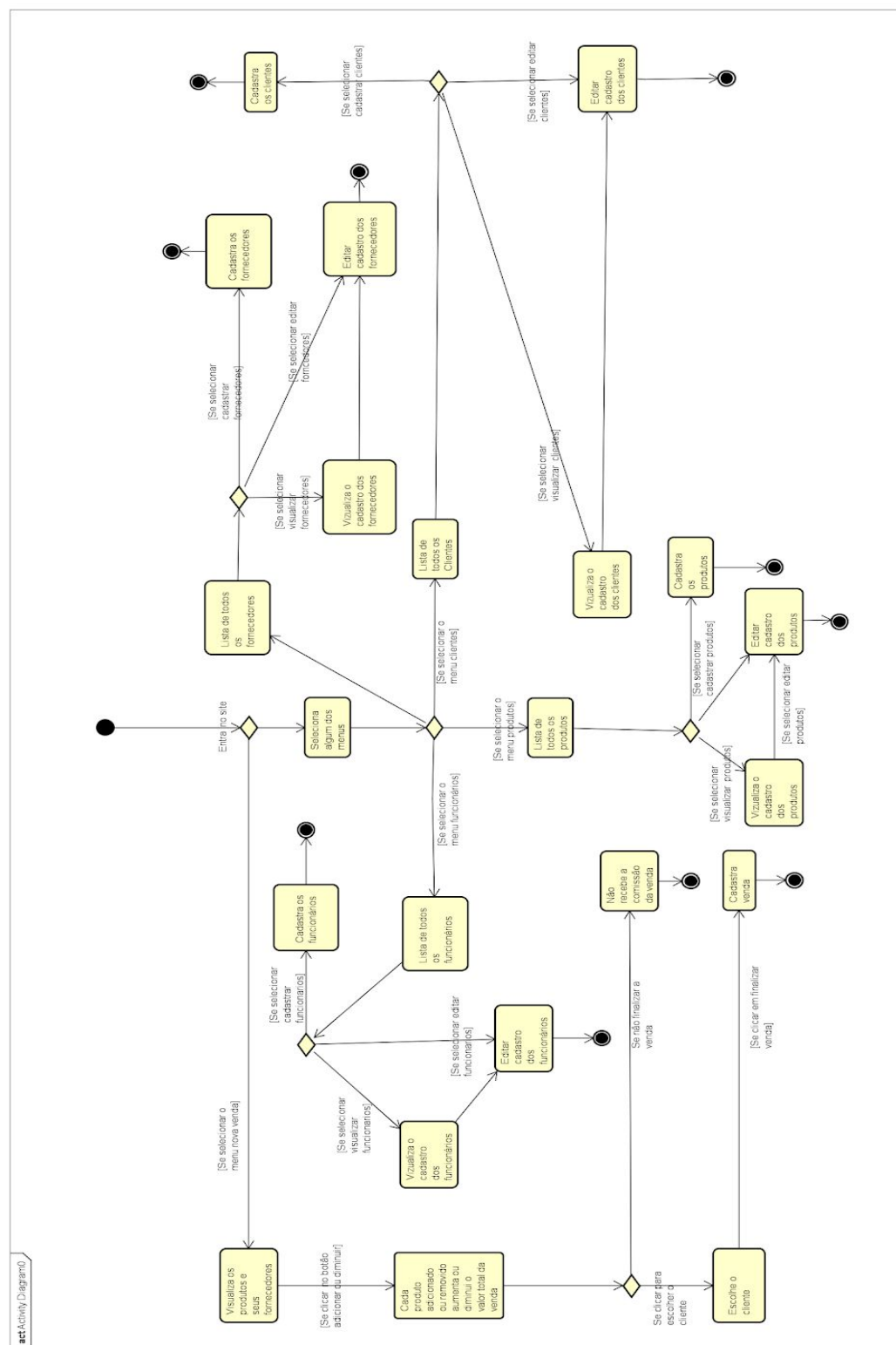


Fonte: Autor

6.10.5 Módulo Gerente

O diagrama de atividade do módulo relatório é apresentado na Figura 48.

Figura 48 - Diagrama de atividade módulo relatório

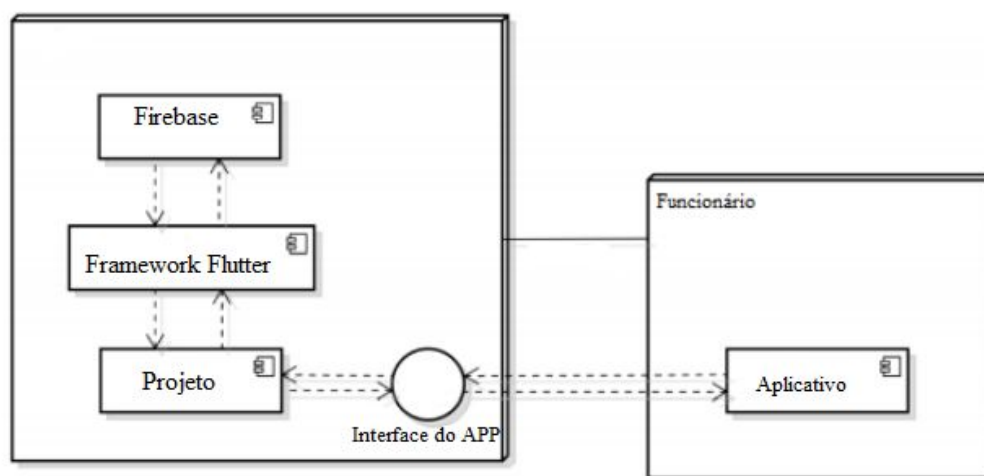


Fonte: Autor

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 o sistema. O diagrama de componentes e implantação é apresentado na Figura 49.

Figura 49 - Diagrama de Componentes e Implantação



Fonte: Autor

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 50.

Figura 50 - Abertura

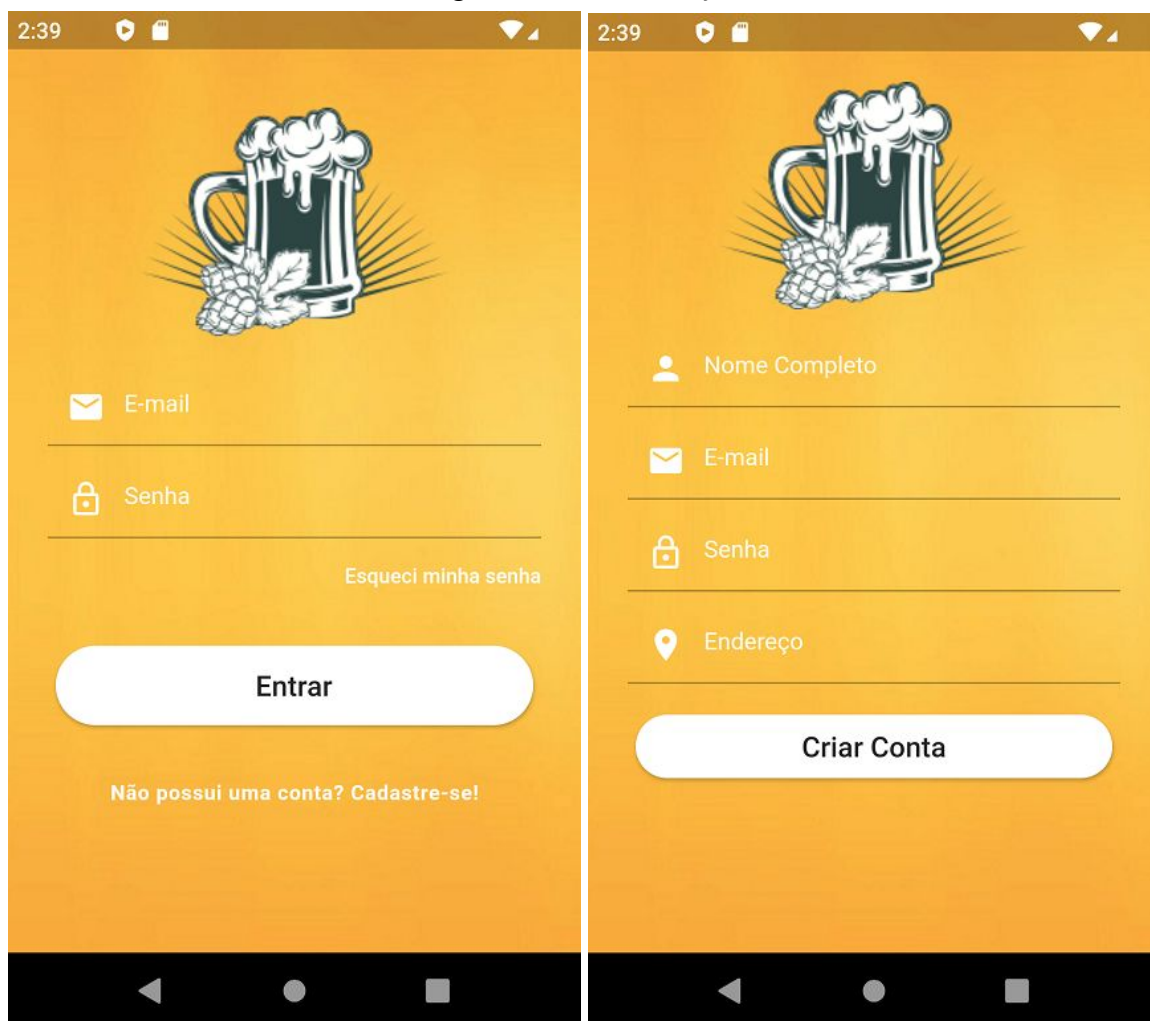


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 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 51.

Figura 51 - Autenticação

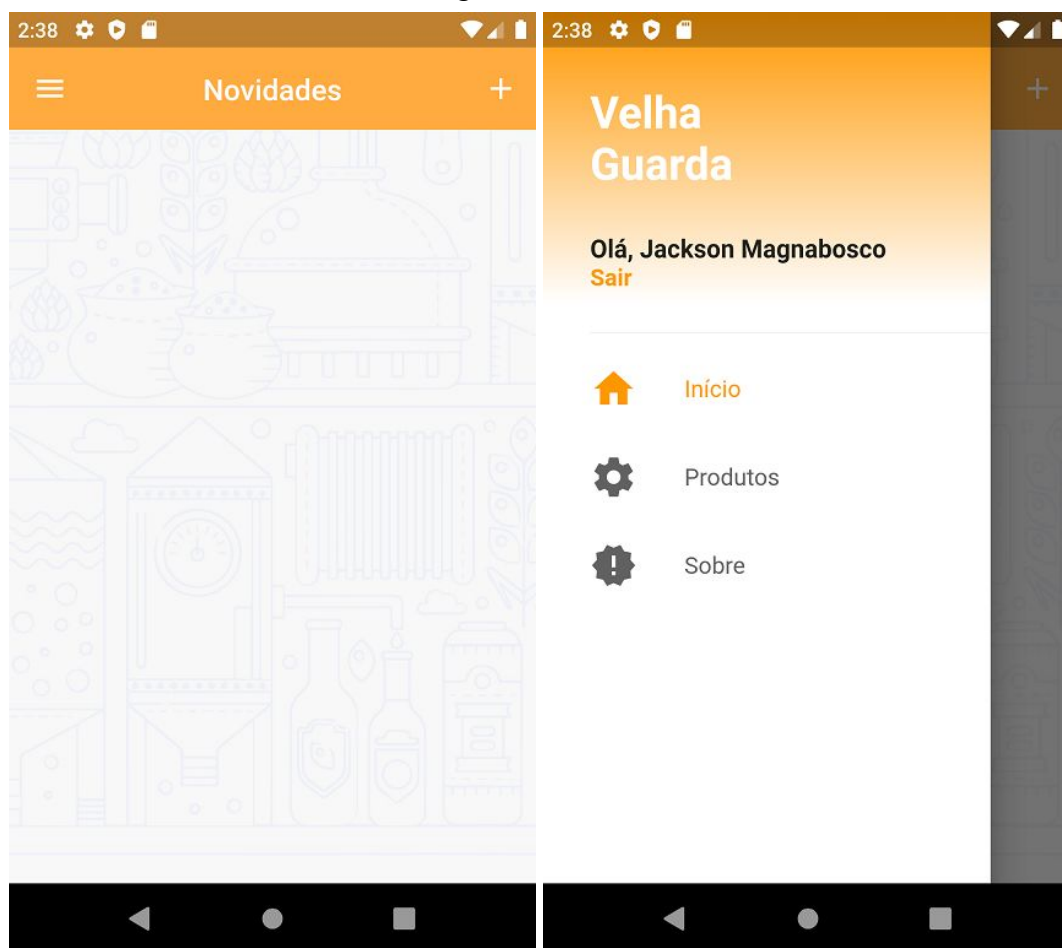


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

7.3 INICIAL

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 52.

Figura 52 -Inicial

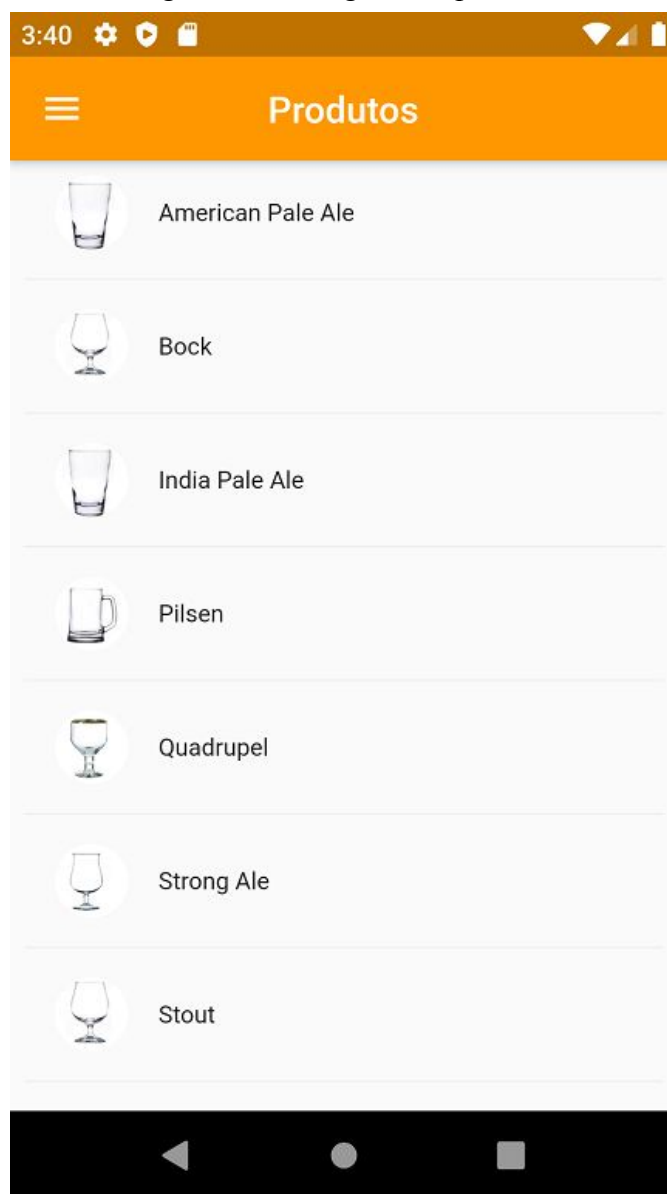


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 53.

Figura 53 - Categoria de produto

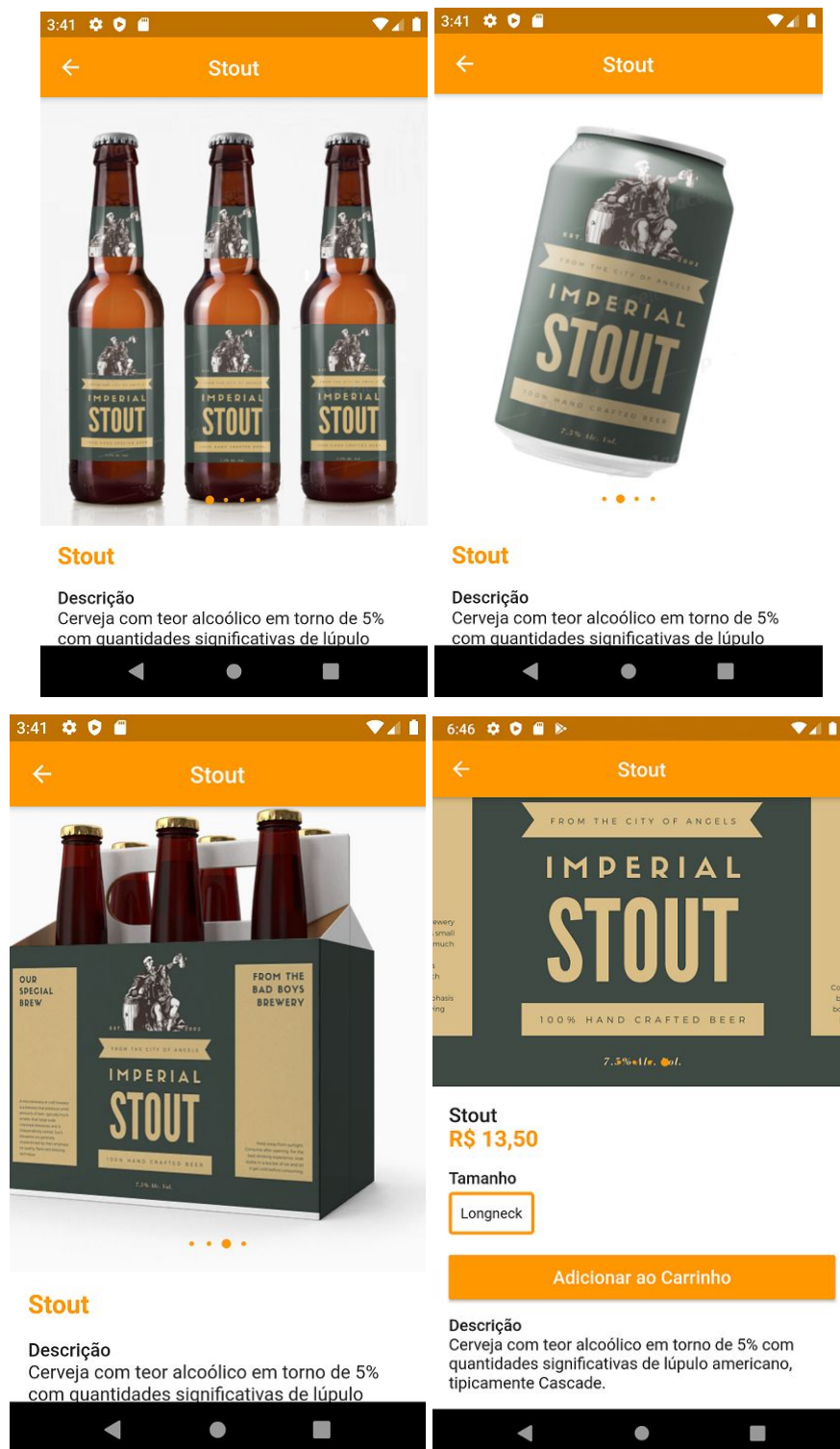


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 54.

Figura 54 - Produto



Fonte: Baseada na imagem original de (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 55.



Figura 55 - Carrinho de compra

2:17


← Stout

 **Stout**
Tamanho: Longneck
R\$ 13,50

— 3 +

 **Cálculo Frete** 

99711-278

 **Cupom de Desconto** +

Resumo do Pedido	
Subtotal	40.50
Desconto	0.00
Entrega	15.00
Total	55.50

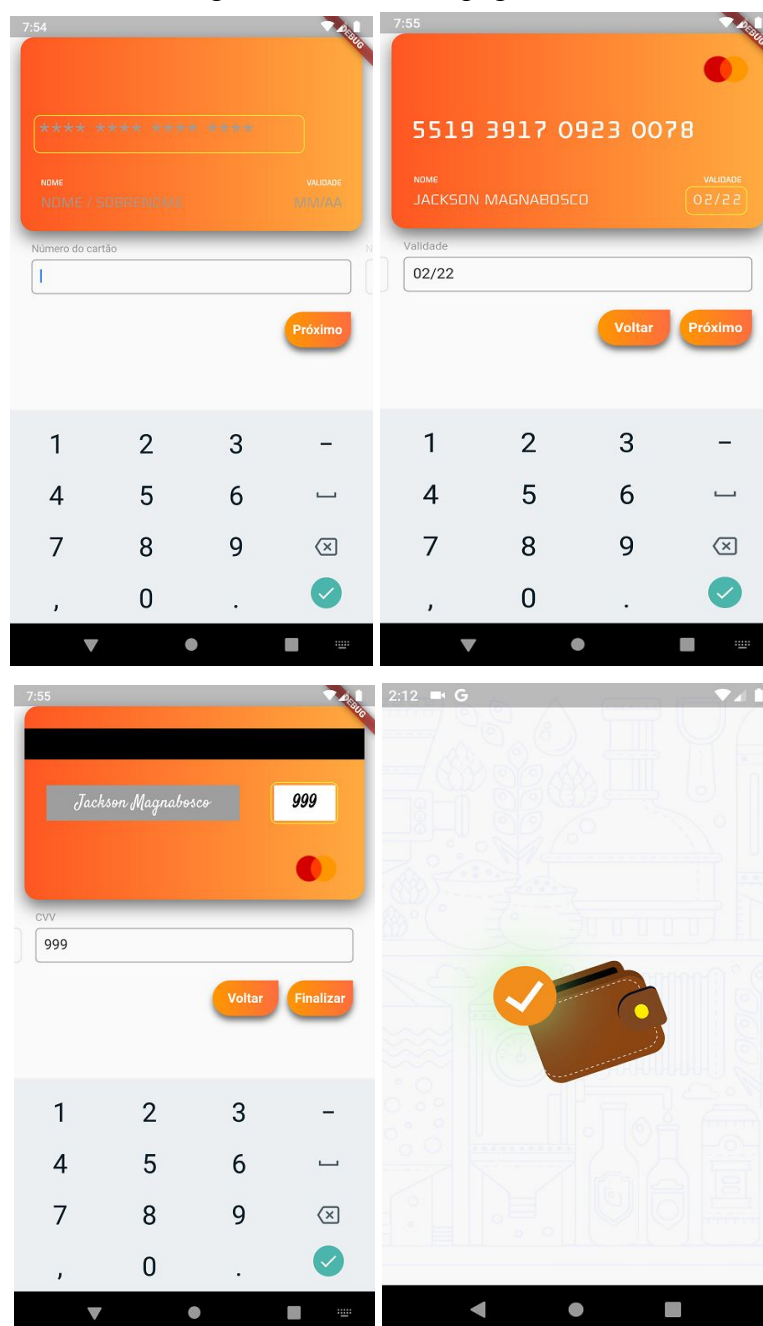
Finalizar Pedido

Fonte: Baseada na imagem original de (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 56.

Figura 56 - Forma de pagamento

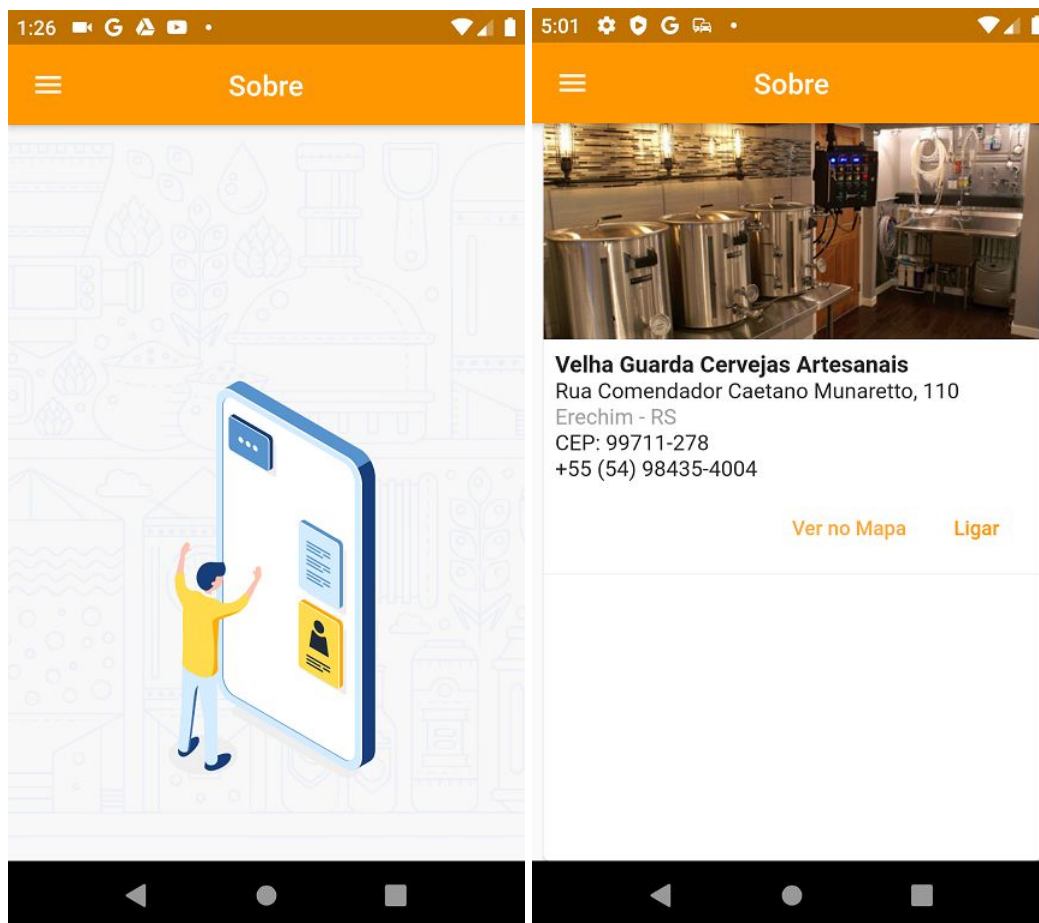


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 57.

Figura 57 - Sobre

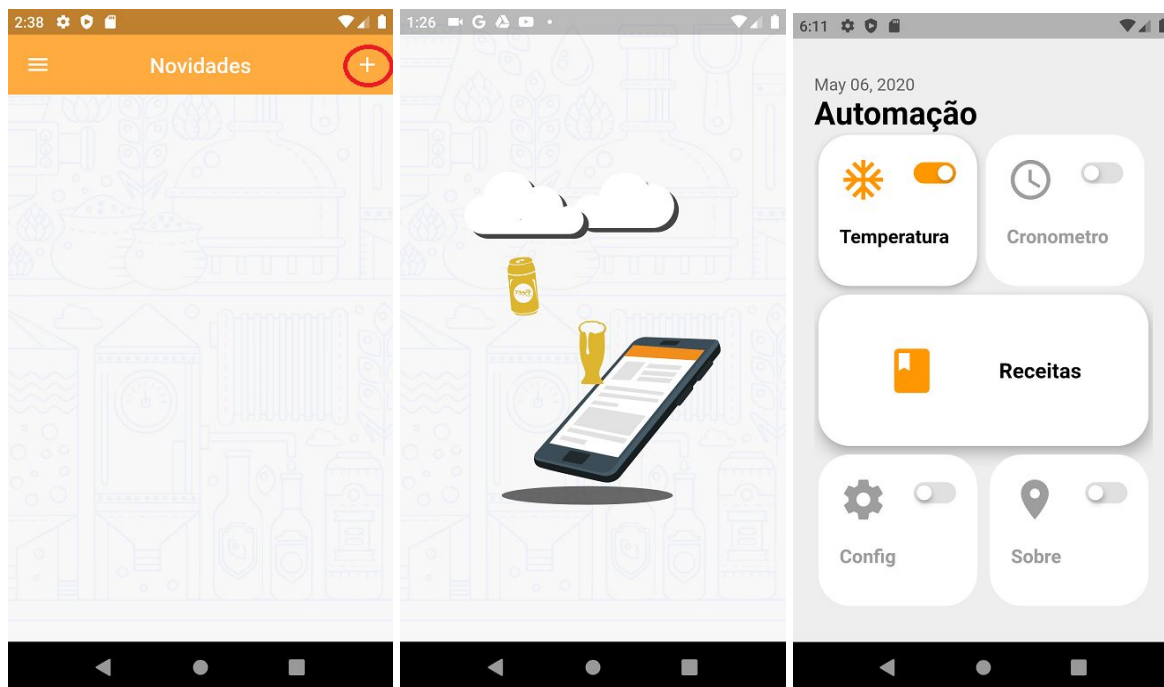


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

7.9 AUTOMAÇÃO

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 58.

Figura 58 - Automação

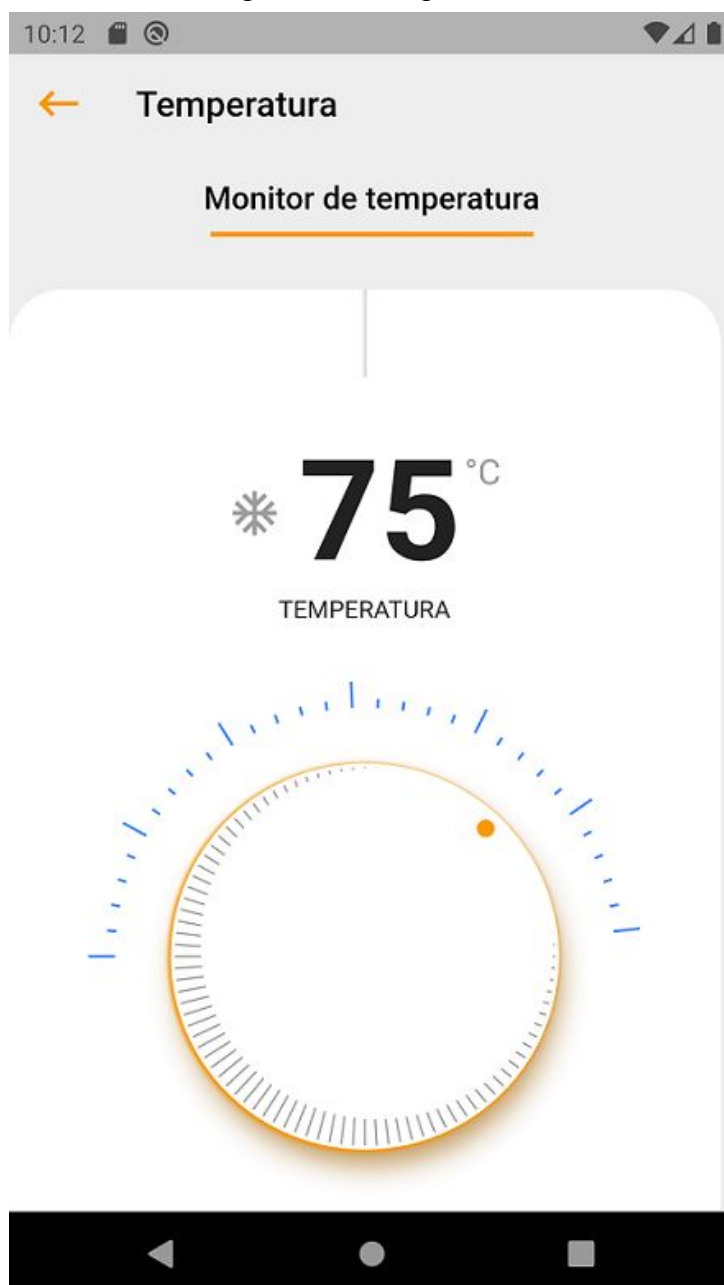


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 59.

Figura 59 - Temperatura

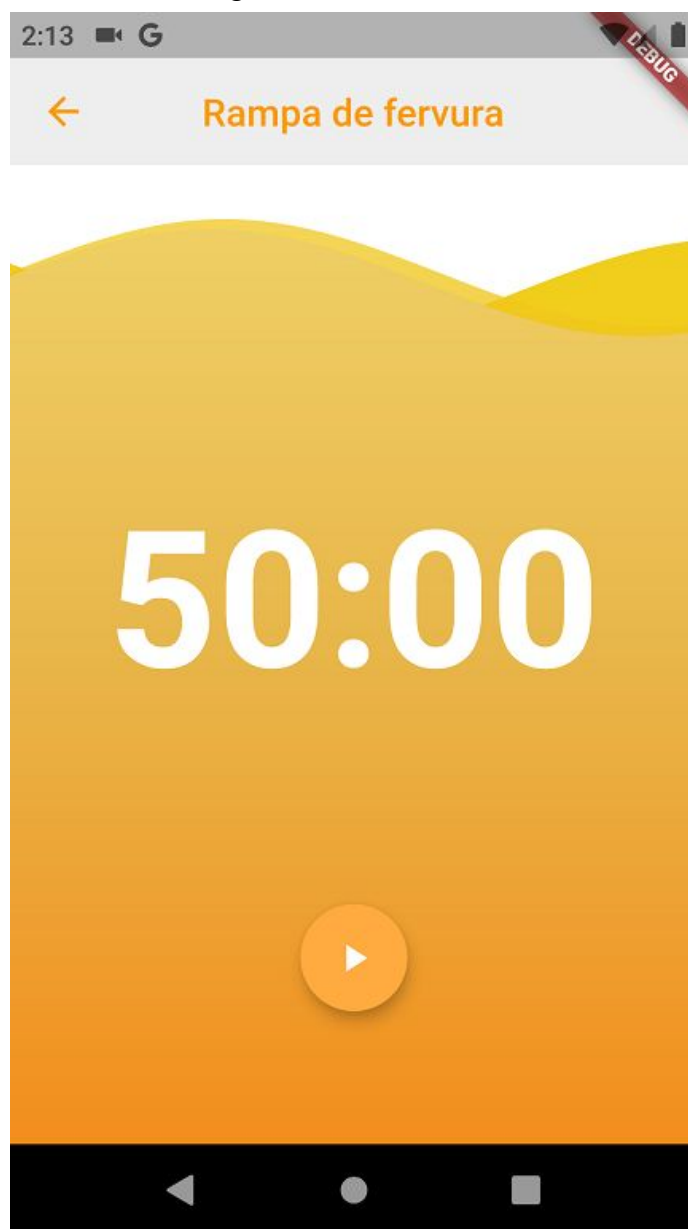


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 60.

Figura 60 - Cronômetro

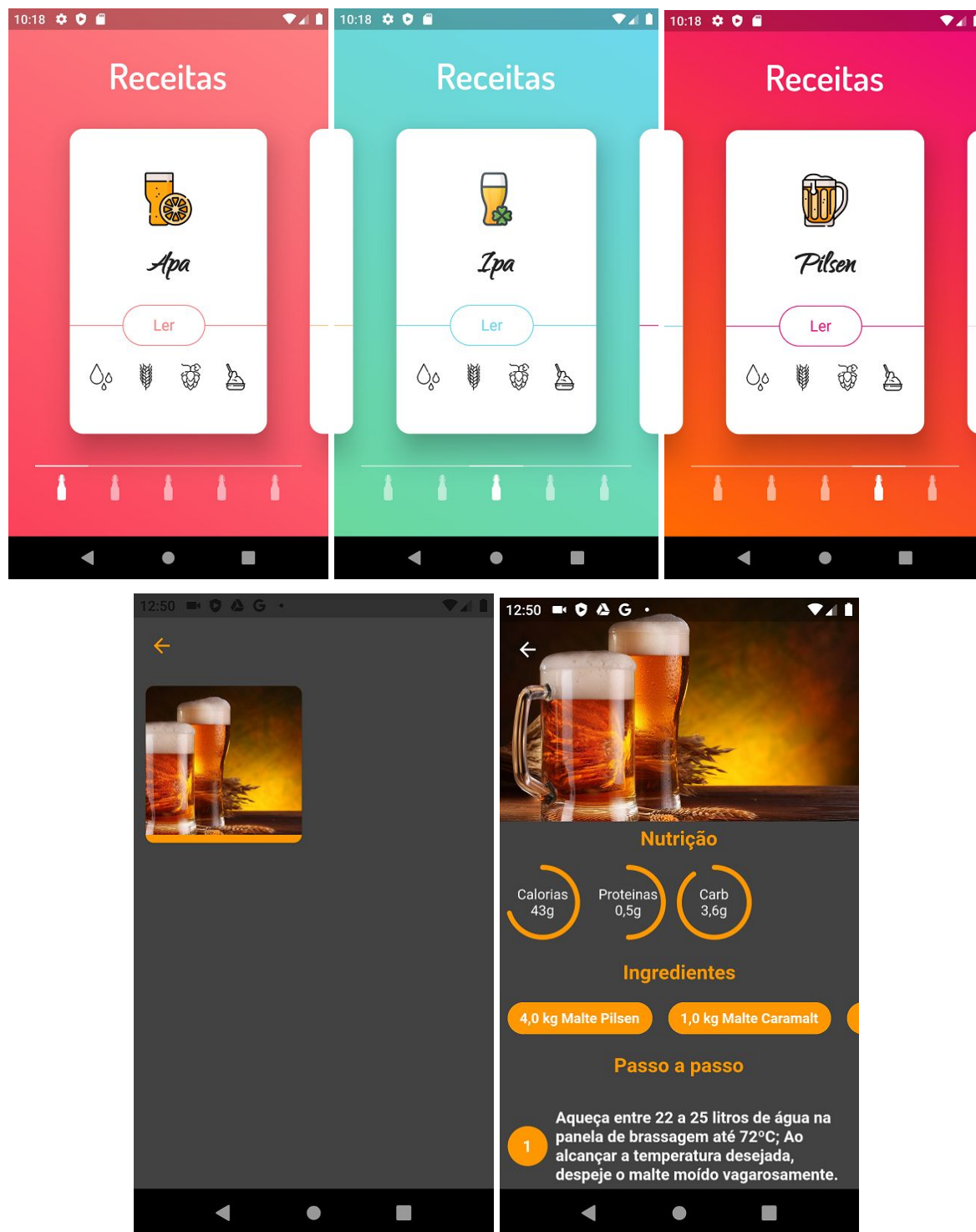


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

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 61.

Figura 61- Receitas



Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

8 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 perceber 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 precisou bastante atenção foi o bancos 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 microcervejeiros.

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 conversa com pequenos produtores da região, foi possível entender a necessidade de fazer este tipo de automação utilizando uma aplicação mobile. Tornar 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 cliente, 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>. 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>. 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-ervejeiro/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. **Firebase — 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-Componentes-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.

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