

# **LABORATÓRIO DE DESENVOLVIMENTO**

## **DESENVOLVIMENTO DE UM APLICATIVO DE VENDAS ONLINE PARA UMA CERVEJARIA**

**Jackson F. Magnabosco e Igor Almeida**

Ciência da Computação – Universidade Regional Integrada do Alto Uruguai e das Missões  
Erechim (URI)

Caixa Postal 743 – 99709-910 –Erechim – RS – Brasil

`jacksonmagnabosco@hotmail.com, 046932@aluno.uricer.edu.br`

**Resumo:** A tecnologia digital vem sendo a resposta para os grandes desafios atuais na venda de cerveja artesanal, trazendo soluções inovadoras e agregando bons resultados, facilitando e tornando mais eficientes a venda no país e no exterior. Definiu-se como objetivo deste trabalho o desenvolvimento de um aplicativo móvel, nomeado Velha Guarda, que consiste em apresentar uma solução para a venda de diversos estilos de cervejas artesanais, 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.

**Abstract:** Digital technology has been the answer to the great current challenges in the sale of craft beer, bringing innovative solutions and adding good results, facilitating and making sales more efficient in the country and abroad. The objective of this work was to develop a mobile application, named Velha Guarda, which consists of presenting a solution for the sale of different styles of craft beer, providing greater security, agility, productivity and cost reduction. This work uses 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.

# 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 venda de cerveja está evoluindo constantemente e, com isso o empreendedor 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 empresário cervejeiro, automatizando sua venda trazendo maior segurança, agilidade, produtividade e redução de custos.

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 cliente a comprar a cerveja perfeita. Uma vez que estas informações poderão ser facilmente acessadas pelo cliente 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.

A descrição da configuração do sistema é tratado no Capítulo 4.

No Capítulo 5 será descrito o processo de desenvolvimento do sistema através do detalhamento das funcionalidades nos diagramas de modelagem UML.

No Capítulo 6 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 sétimo 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"><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: 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).

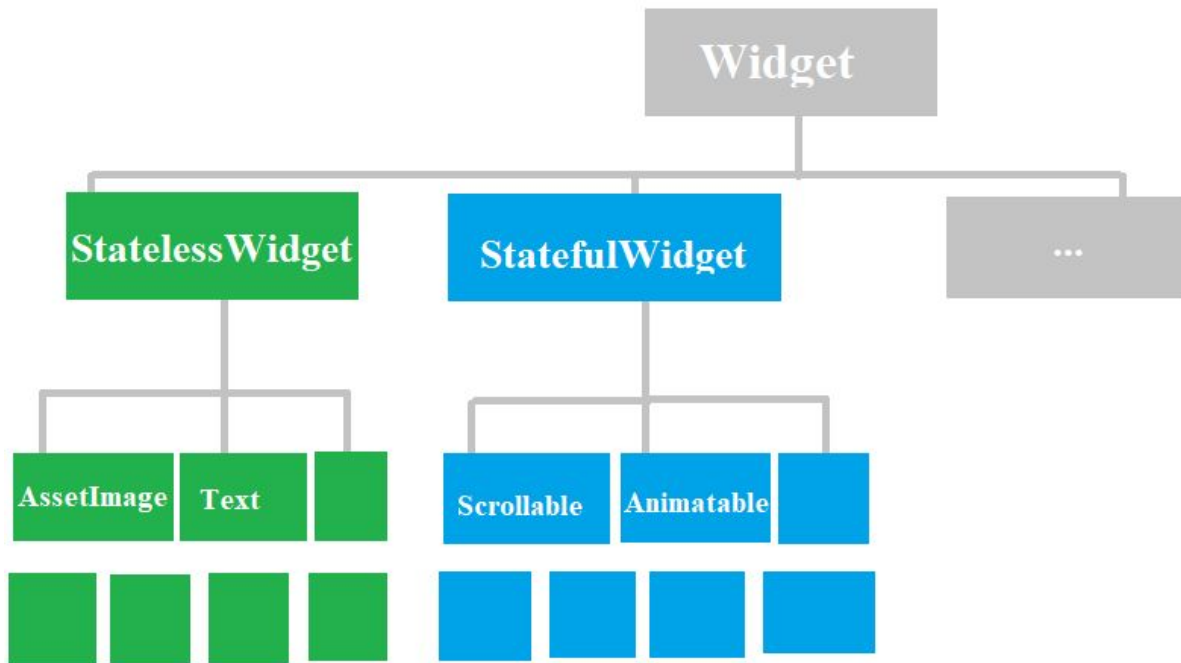
Outra vantagem que torna indispensável é o *Just in time*, a recompilação do código no dispositivo, enquanto o aplicativo está rodando, a aplicação não perde o estado de desenvolvimento. Isso gera um ciclo de desenvolvimento muito rápido e produtivo, possibilitando o recarregamento expresso do aplicativo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

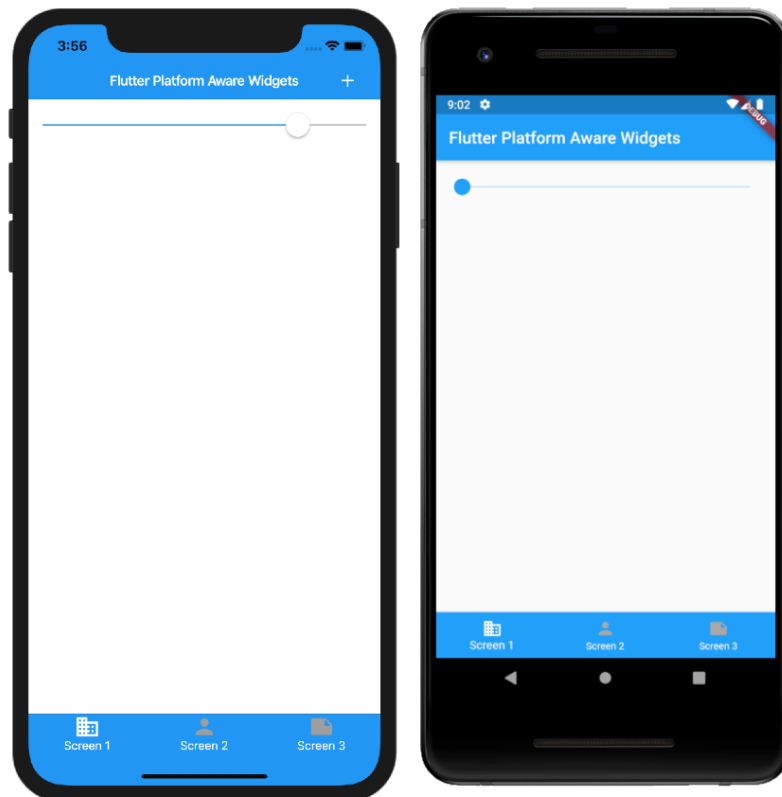
Figura 02 - HTML/CSS análogos em Flutter



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 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

<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)

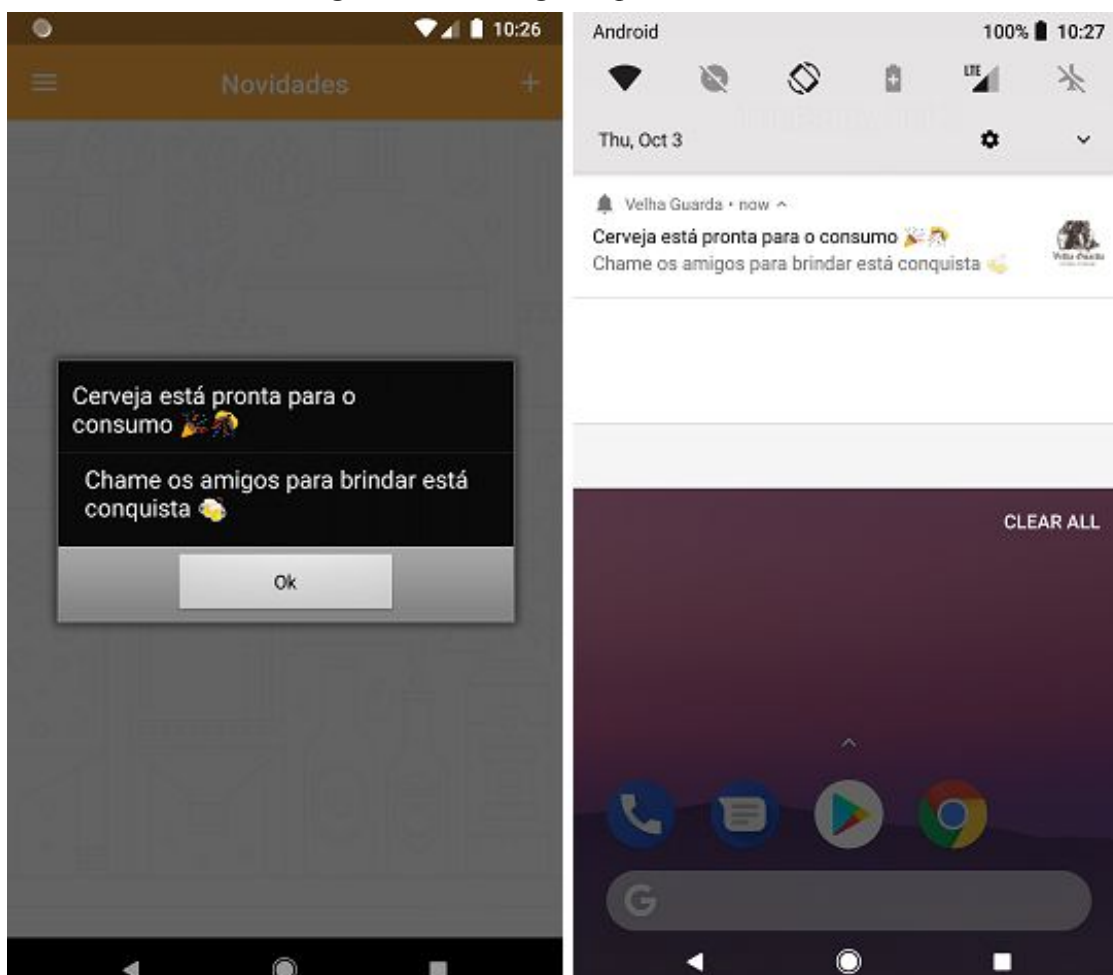
A principal ideia de utilizar o Firebase no projeto é por ser uma plataforma móvel completa e robusta de fácil uso, com a viabilidade de trabalhar *offline* e sincronizar automaticamente ao retornar a conexão, utiliza sincronização em tempo de execução, tem plano gratuito e atende muitos cenários utilizando 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, 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 CONFIGURAÇÕES

### 4.1 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 07.

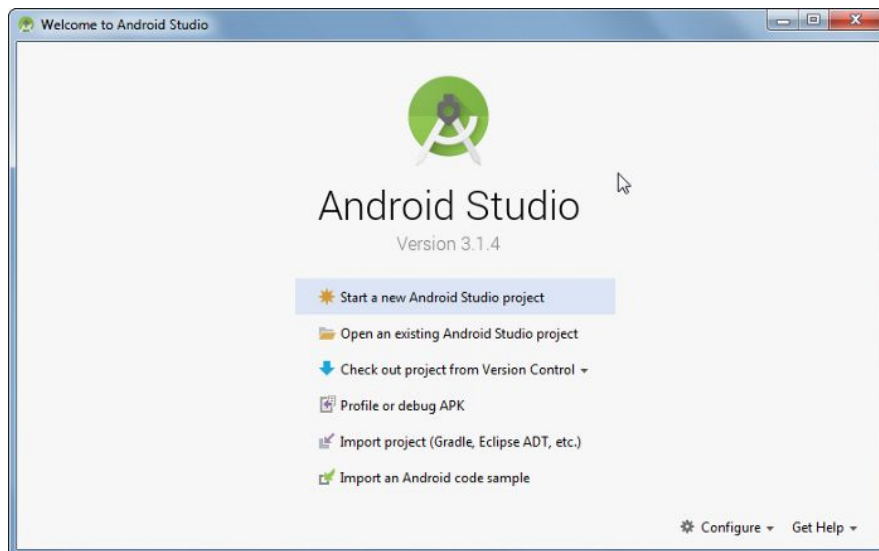
Figura 07 - Instalação Android Studio



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 08.

Figura 08 - Tela inicial Android Studio

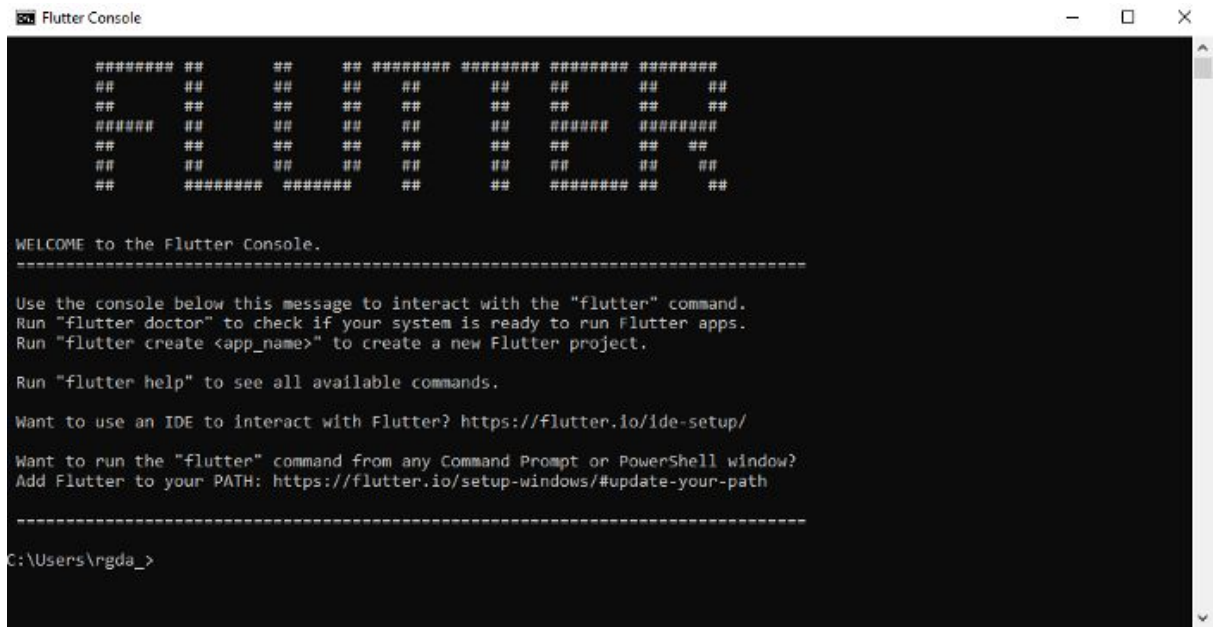


Fonte: Baseada na imagem original de (MUNIZ, 2018)

## 4.2 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 09 (SESSA, 2020).

Figura 09 - 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).

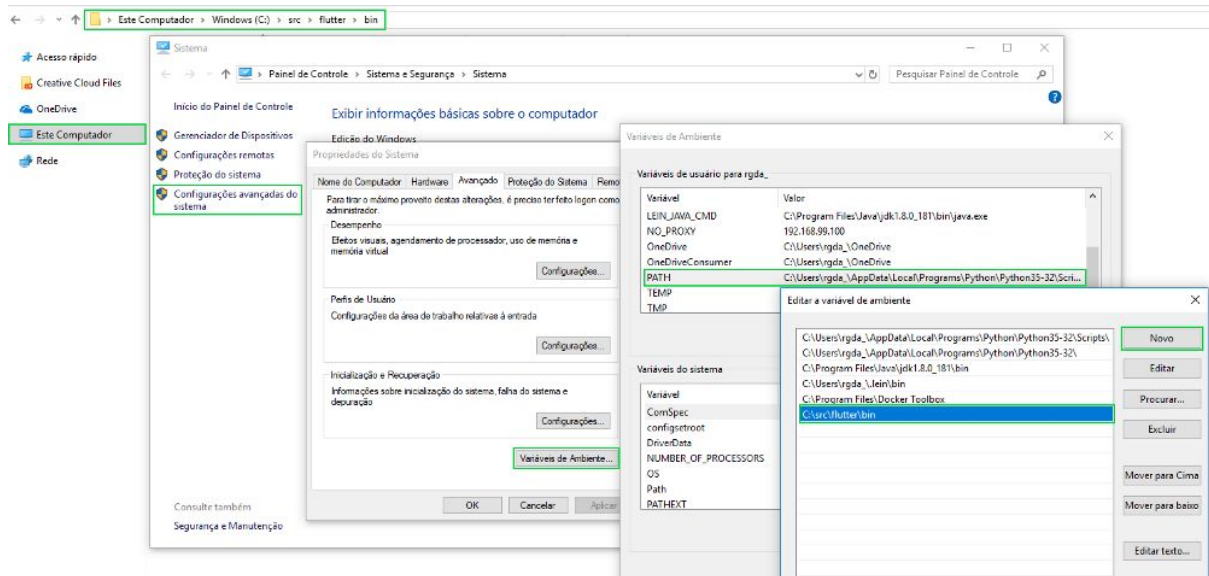
Para isto precisamos seguir um plano, o primeiro passo é copiar o caminho até o diretório \bin, presente na pasta do flutter.

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 como demonstra a Figura 10.

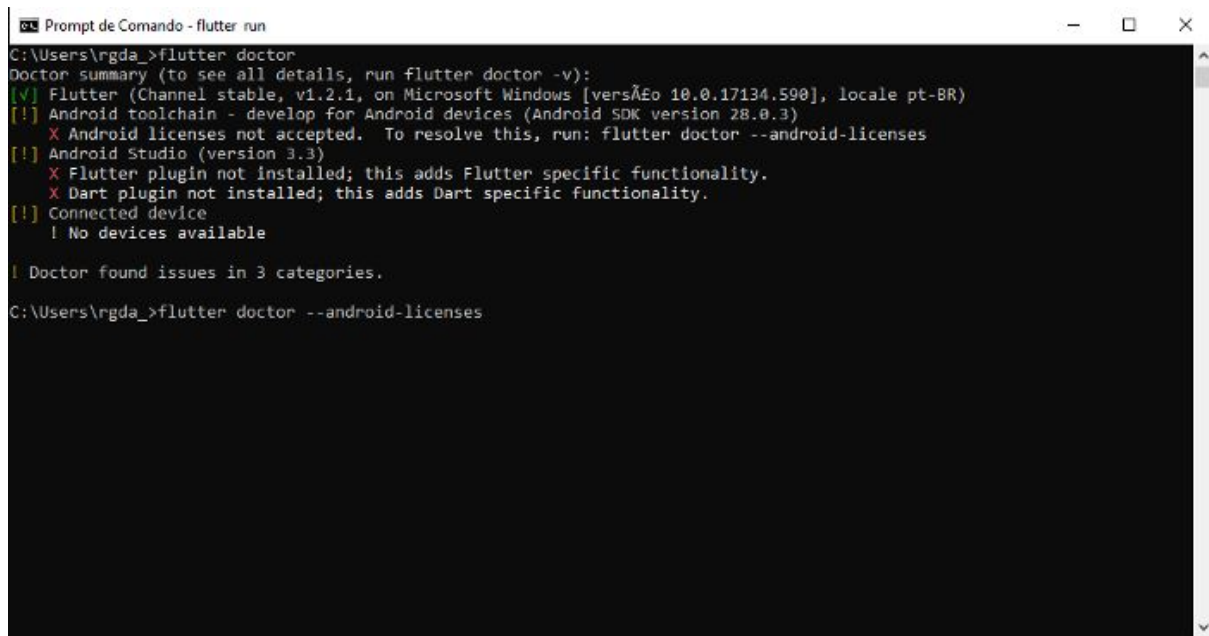
Figura 10 - Configurando a variável de ambiente



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 11.

Figura 11 - Flutter Doctor



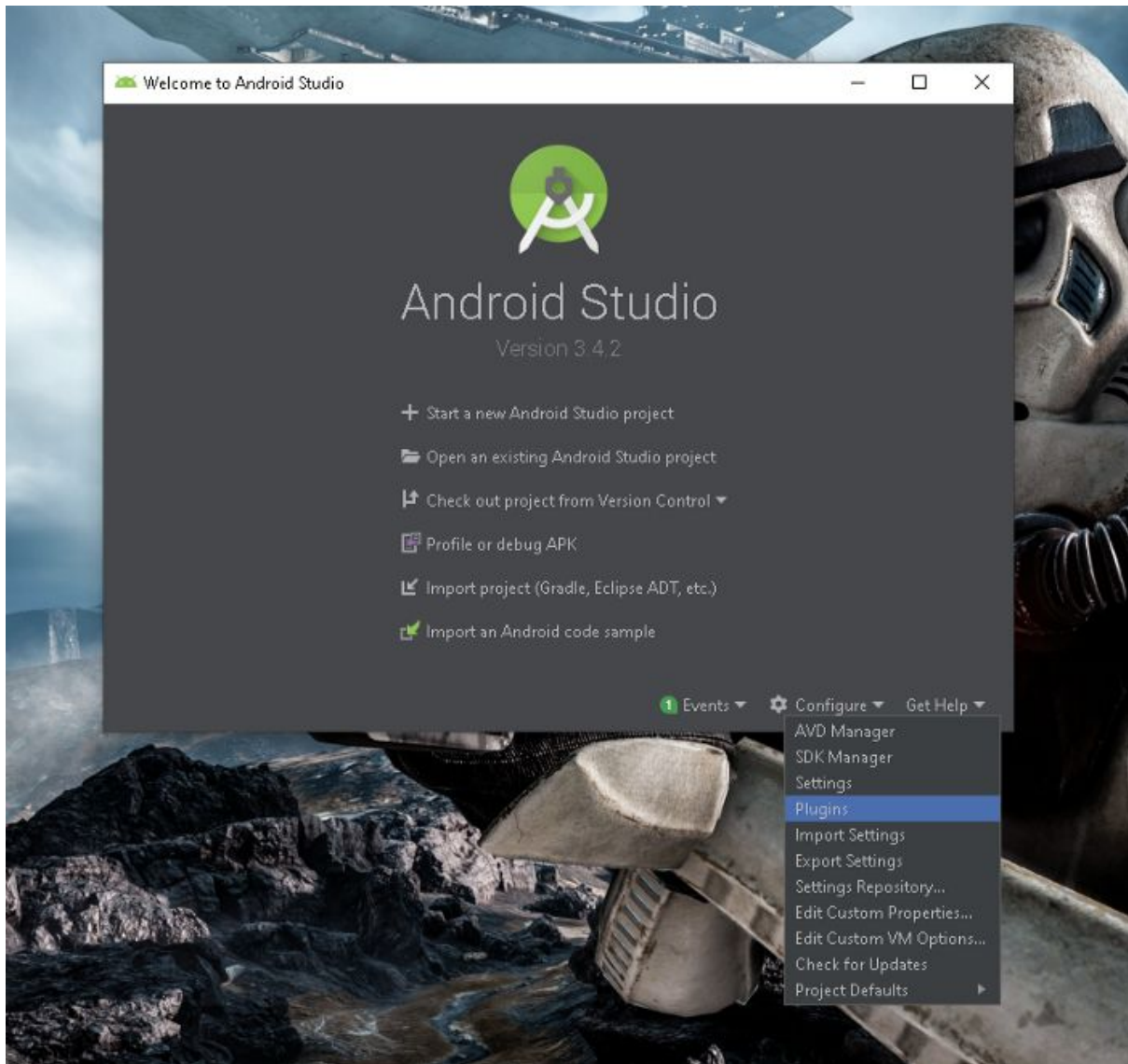
Fonte: Baseada na imagem original de (DIAS, 2019)

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 12.

Figura 12 - Android Studio Plugins

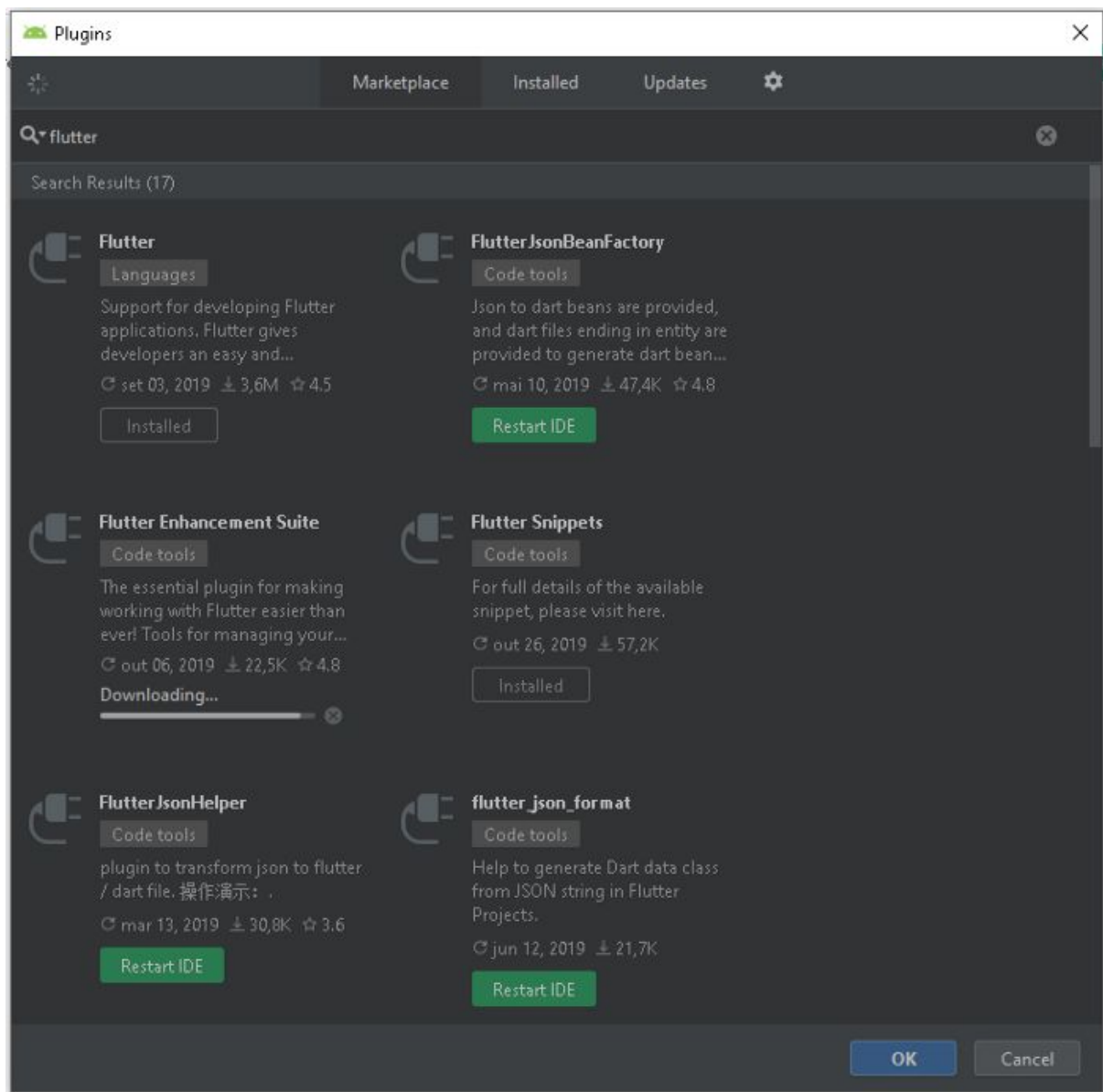


Fonte: Baseada na imagem original de (SESSA, 2020)

Instale os plugins do Dart e Flutter para que tudo funcione da forma correta como demonstra a imagem 13.



Figura 13 - Plugins Dart e Flutter



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

O Módulo do Gerente é responsável por controlar todas as operações do sistema.

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

O Módulo do funcionário permitirá que a mesma 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.

### 6.6.1 Diagrama de Caso de uso geral

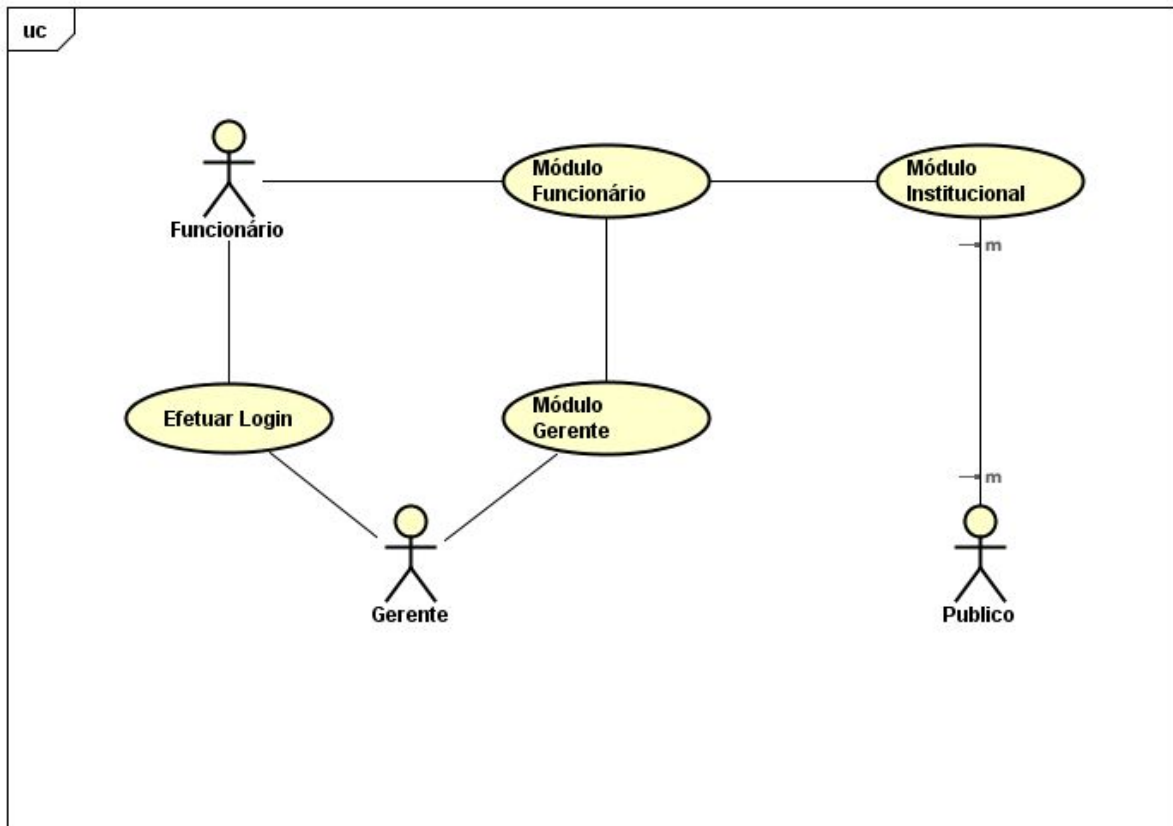


Figura 14 - Diagrama geral de caso de uso

### 6.6.2 Efetuar login

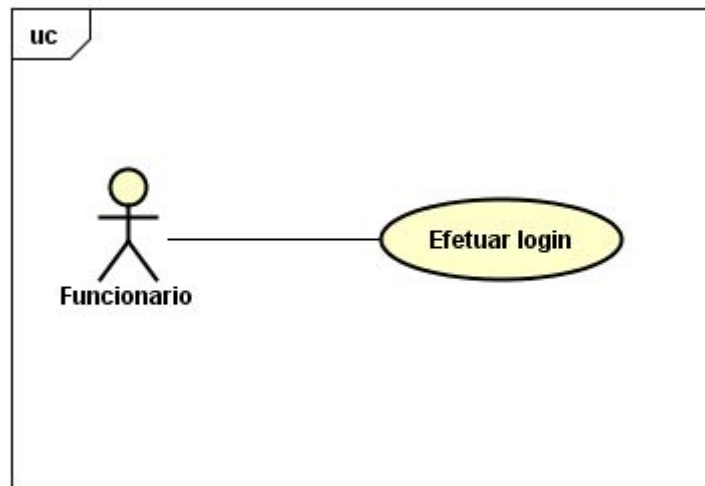


Figura 15 - Diagrama de caso de uso efetuar login

### 6.6.3 Módulo Gerente

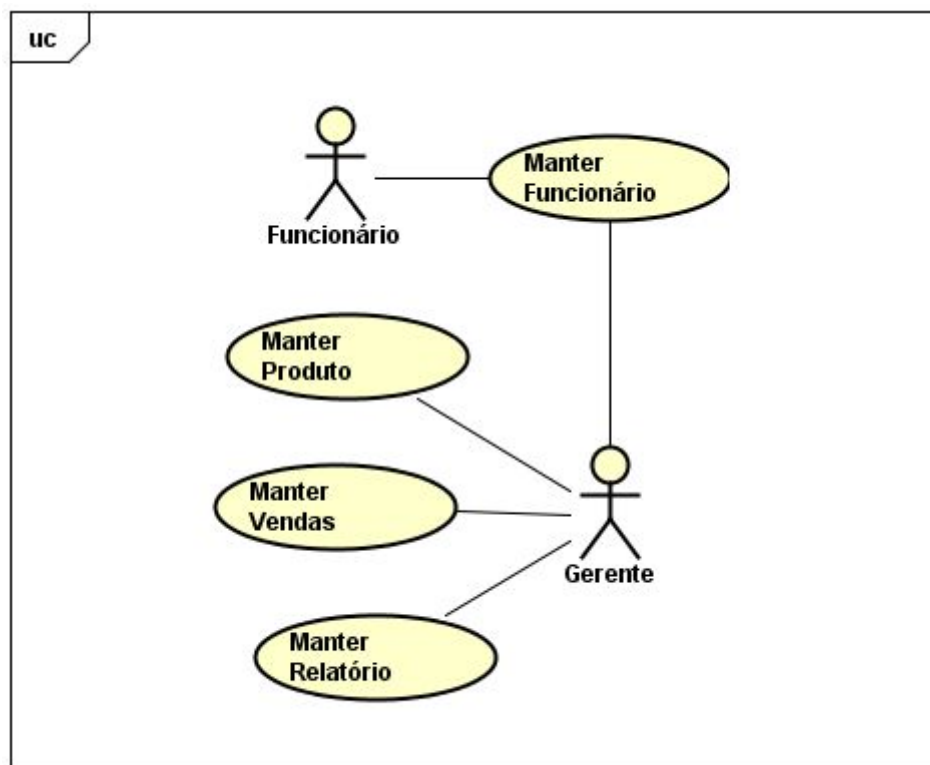


Figura 16- Diagrama de caso de uso módulo gerente

#### 6.6.4 Módulo Institucional

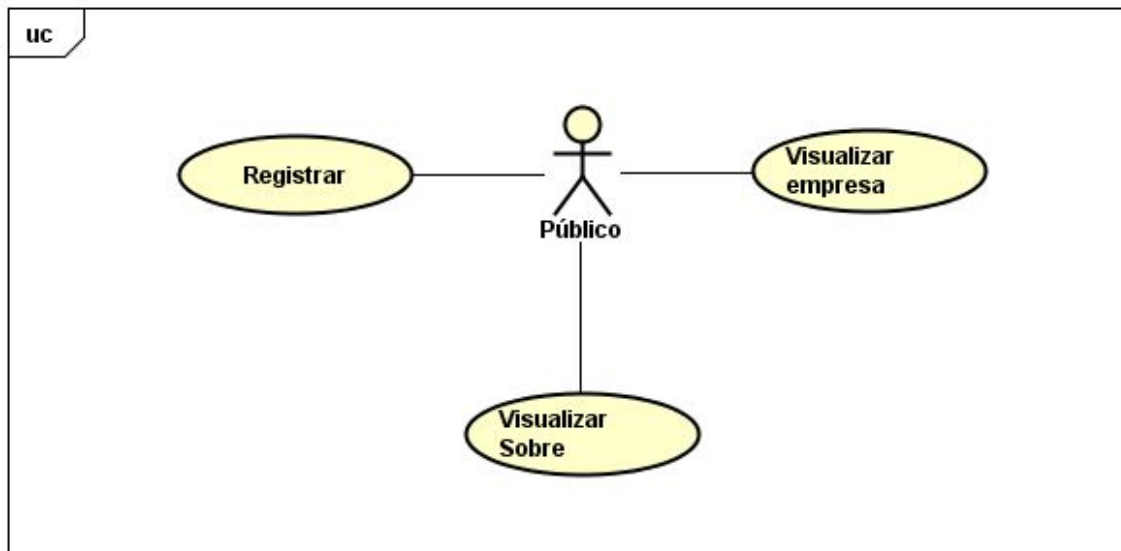


Figura 17- Diagrama de caso de uso módulo institucional

#### 6.6.5 Módulo Funcionário

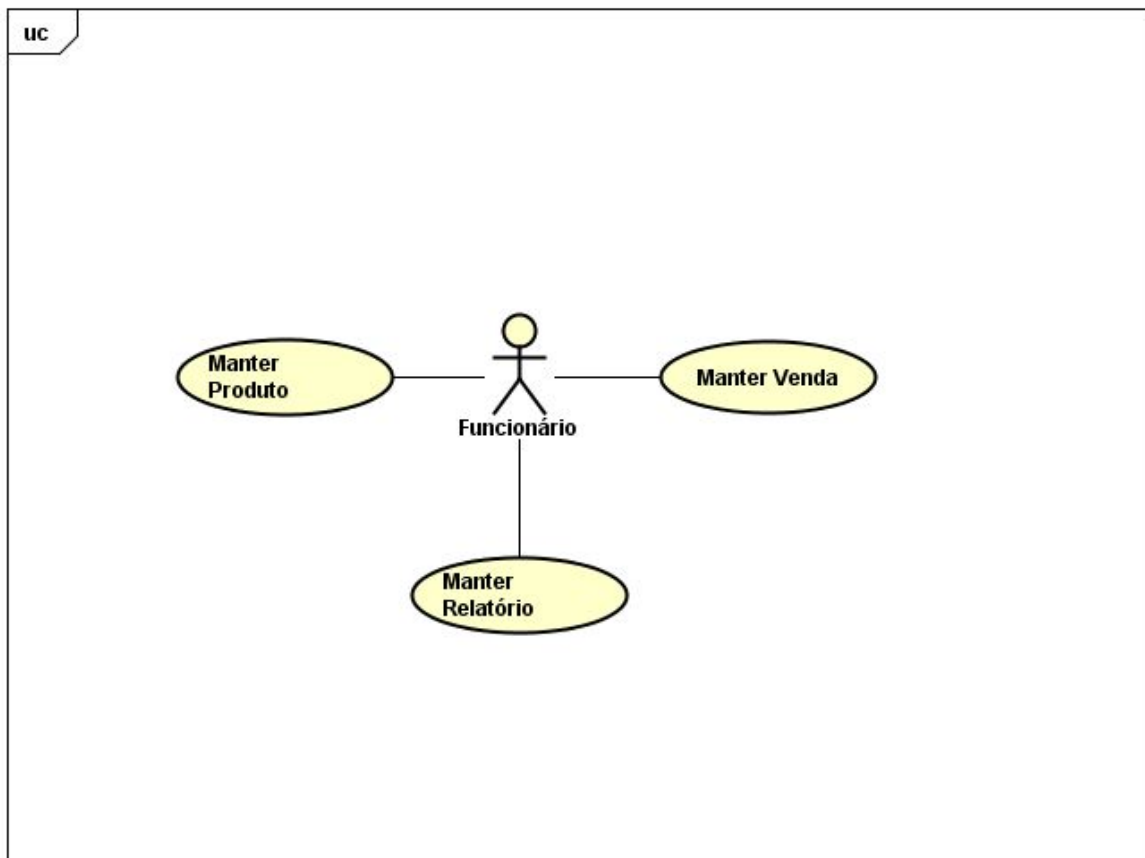


Figura 18- Diagrama de caso de uso módulo funcionário

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

	<b>Ator</b>		<b>Sistema</b>	
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 é possível excluir um registro que seja feito por um Gerente.

Tabela 02- Descrição de caso de uso manter produto

<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

Tabela 03- Descrição do caso de uso efetuar login



<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 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

<b>Caso de Uso:</b>	Manter Venda
<b>Ator(es):</b>	Gerente, Funcionário
<b>Pré-condições:</b>	O usuário deve estar logado no sistema.
<b>Pós-condições:</b>	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 é 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.	

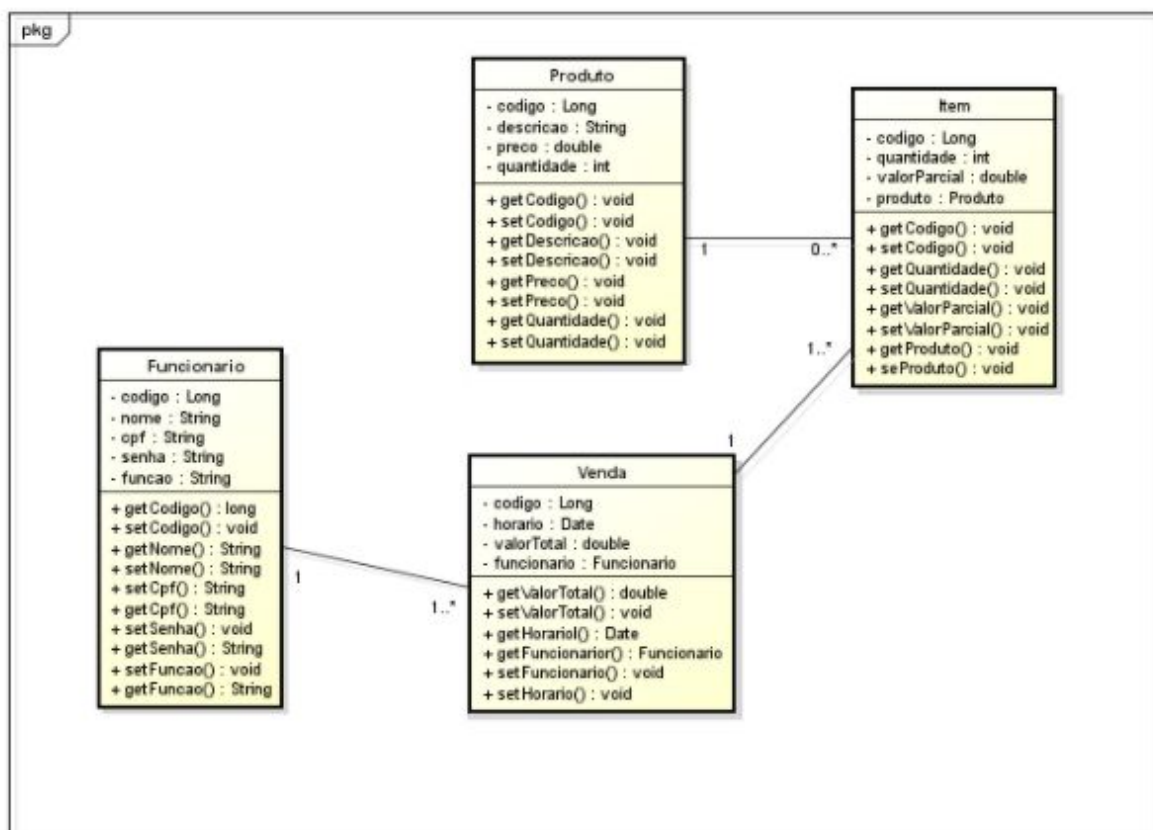
A1	O Ator seleciona a operação de alteração. Utiliza “Alterar relatório” (linha 8). / seleciona a operação. Utiliza “Buscar relatório” (linha 14), seleciona a operação de exclusão. Utiliza “Excluir relatório (linha 17)”.
A2	Cancela caso de uso.
A3	Cancela caso de uso.
E1	O Ator não preenche corretamente todos os campo.
R1	Só é 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 19, mostra a estrutura básica das classes do sistema.

Figura 19 - Diagrama de classe



Fonte: Autor

## 6.7 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.



## 6.7 Diagrama de sequência

### 6.7.1 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.

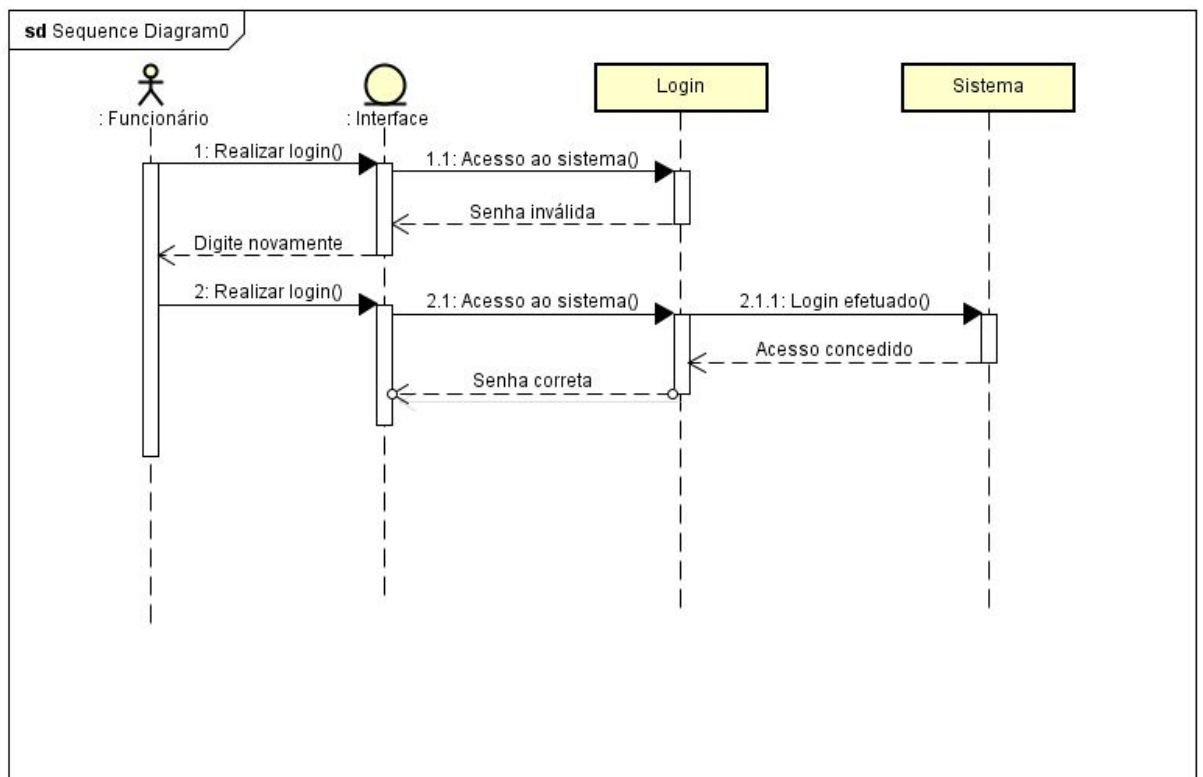


Figura 20 - Diagrama de sequência manter login

### 6.7.2 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.

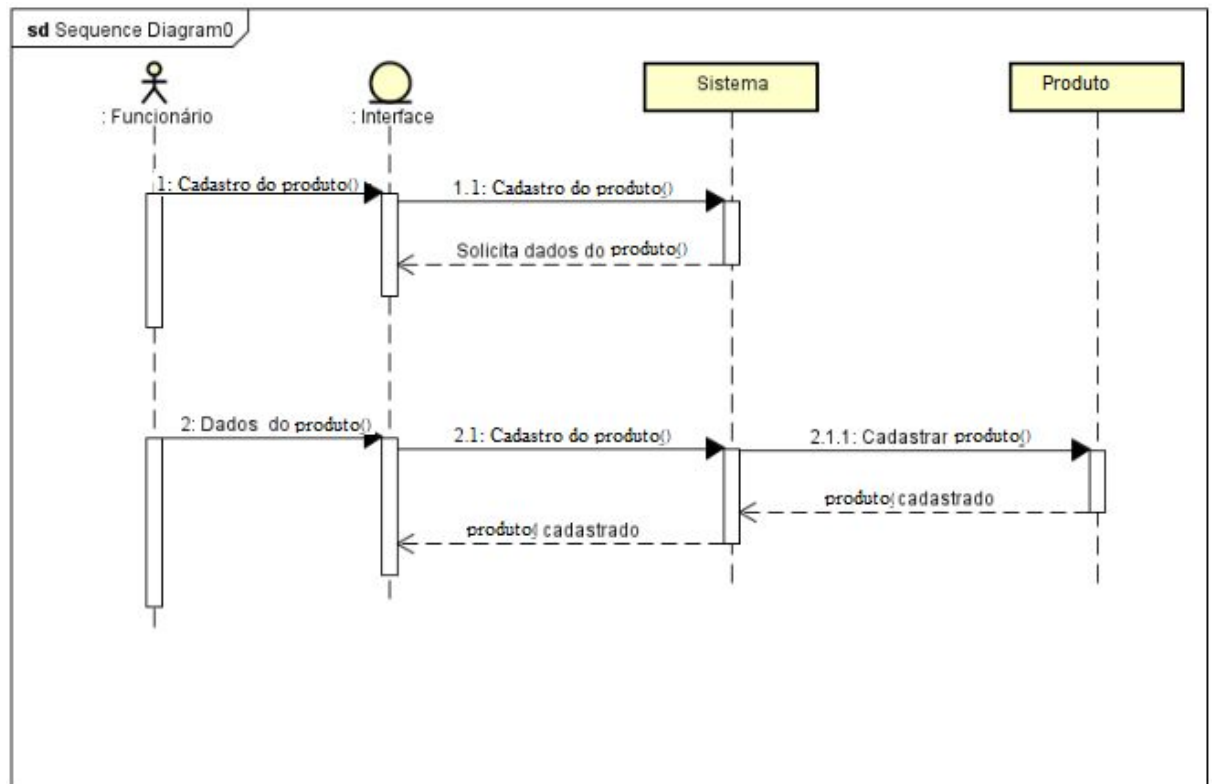


Figura 21 - Diagrama de sequência manter produto

### 6.7.3 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

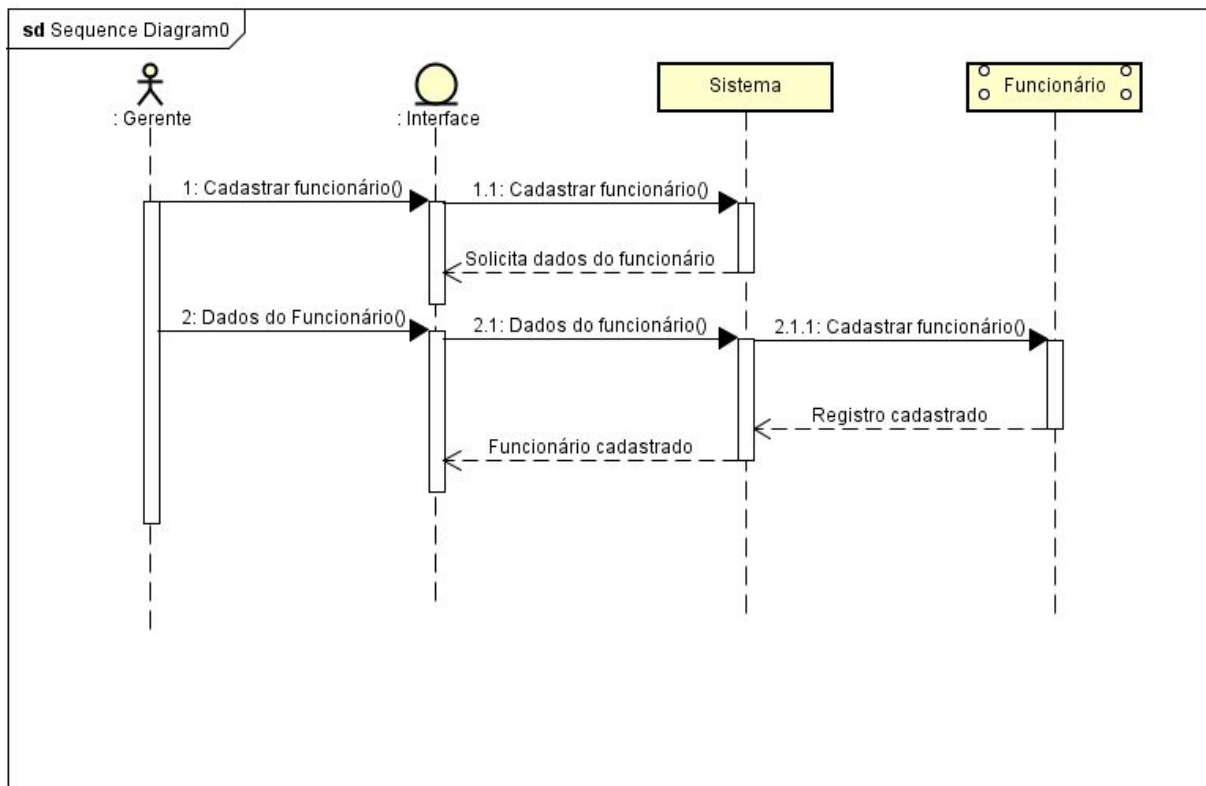


Figura 22 - Diagrama de sequência manter funcionário

### 6.7.4 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.

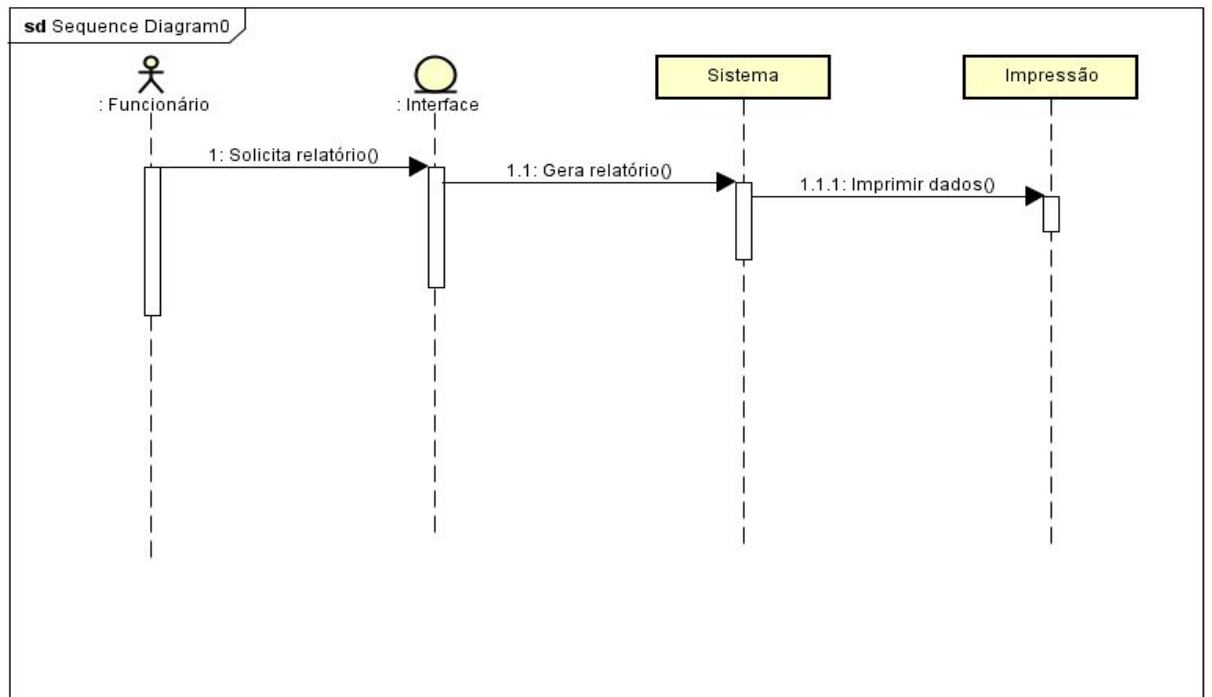


Figura 23 - Diagrama de sequência manter relatório

## 6.8 Diagrama de atividade

### 6.8.1 Módulo Institucional

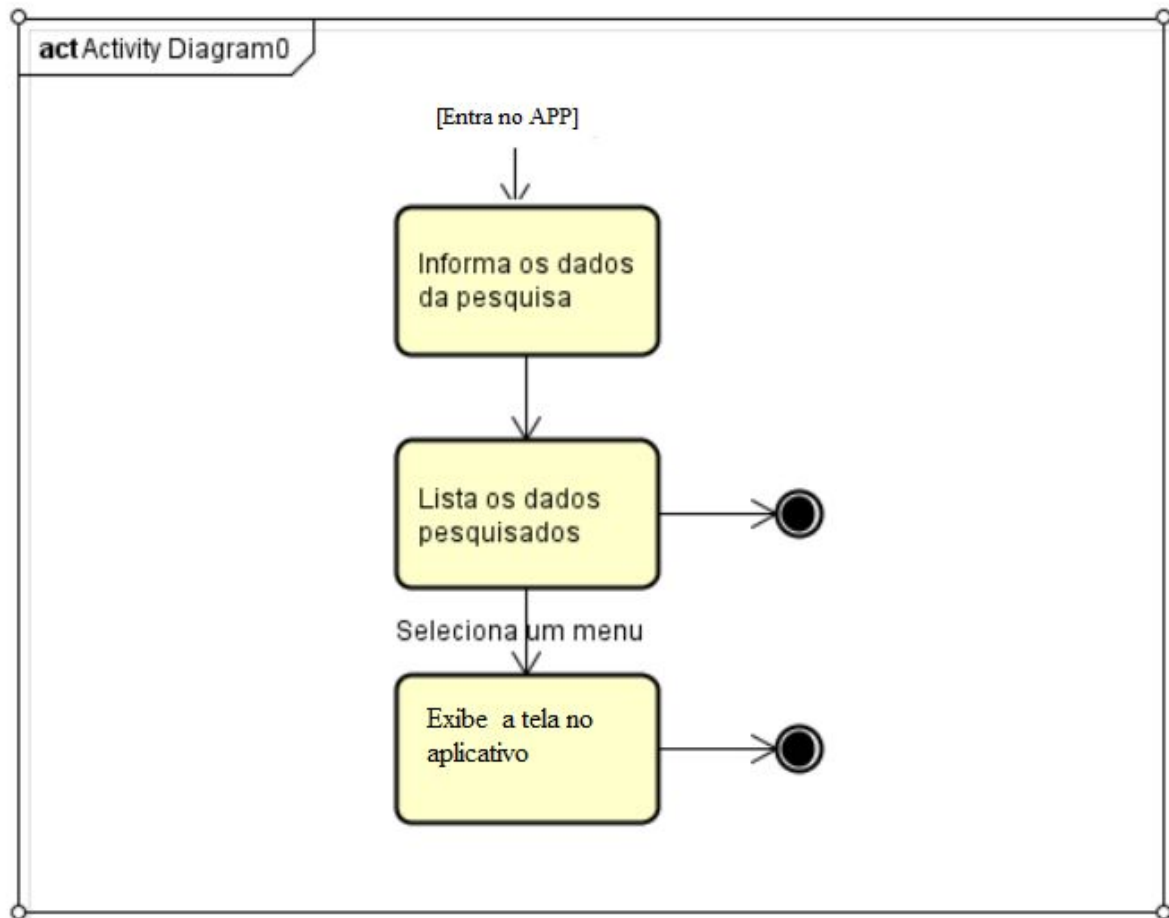


Figura 24 - Diagrama de atividade módulo institucional

### 6.8.2 Fazer Login

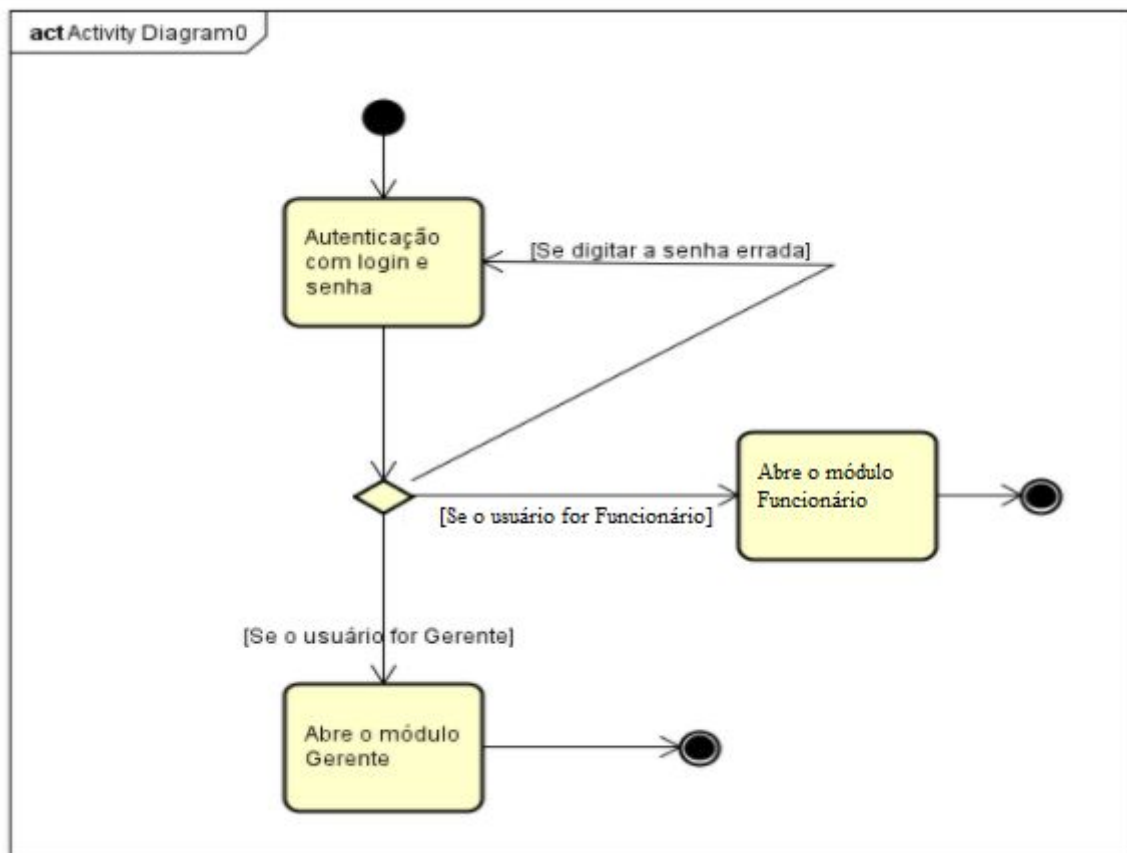


Figura 25 - Diagrama de atividade fazer login

### 6.8.3 Módulo Funcionário

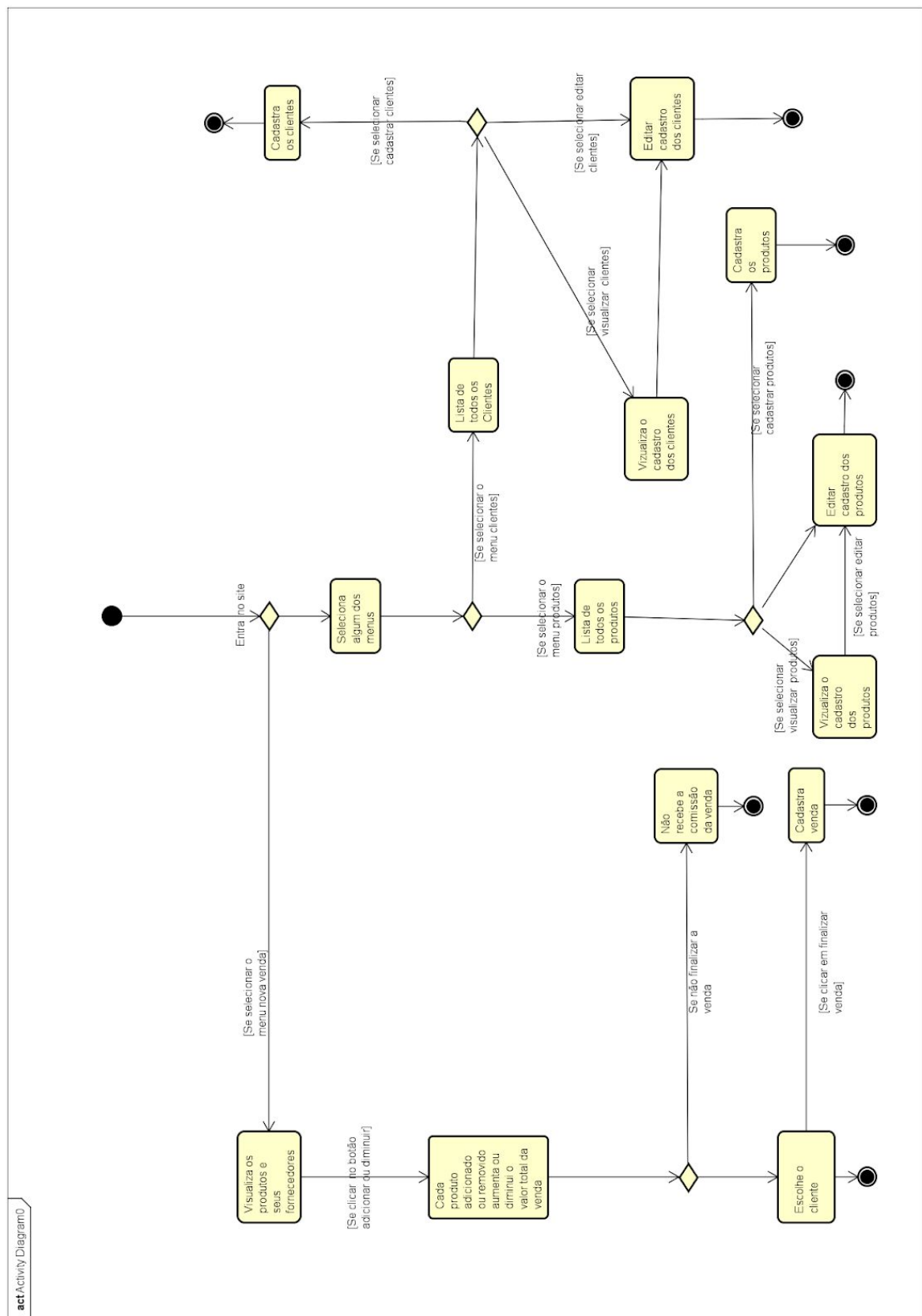


Figura 26 - Diagrama de atividade Módulo Funcionário

#### 6.6.4 Relatório

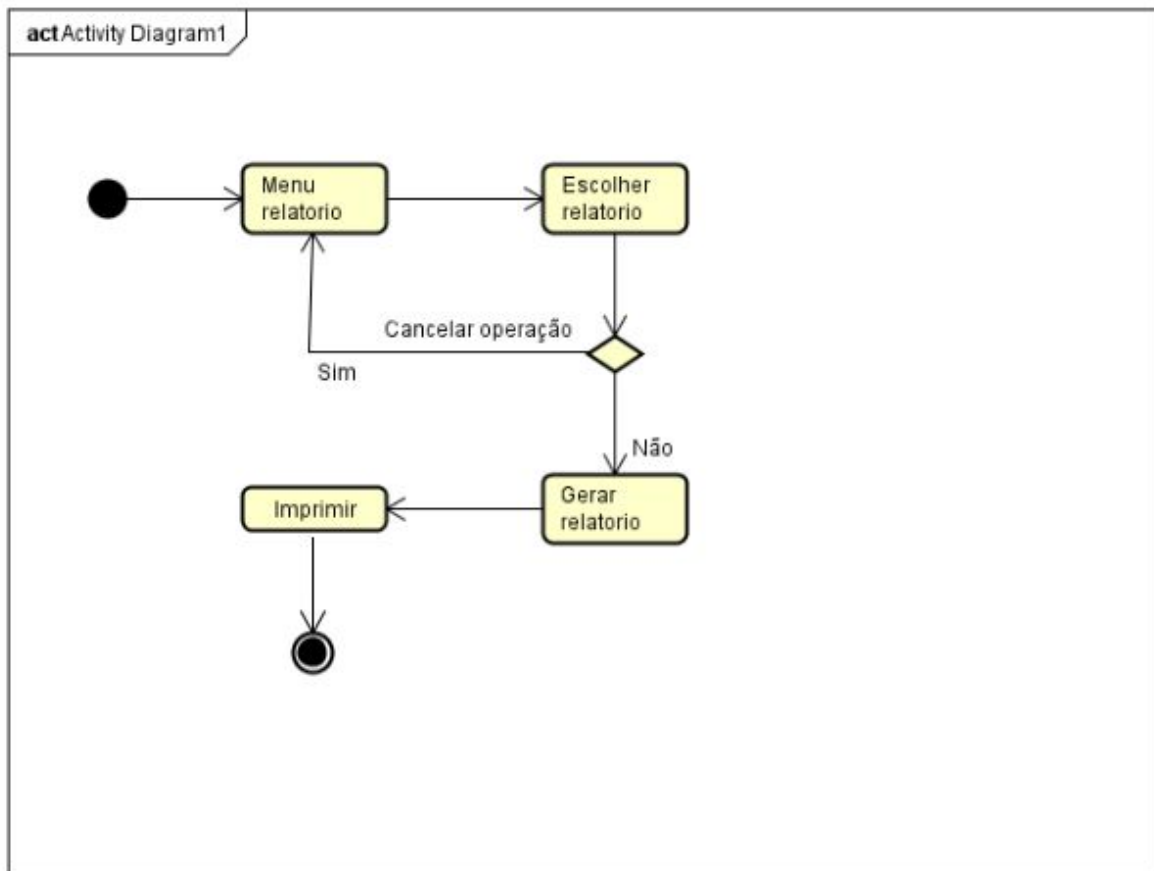


Figura 27 - Diagrama de atividade relatório



### 6.6.5 Módulo Gerente

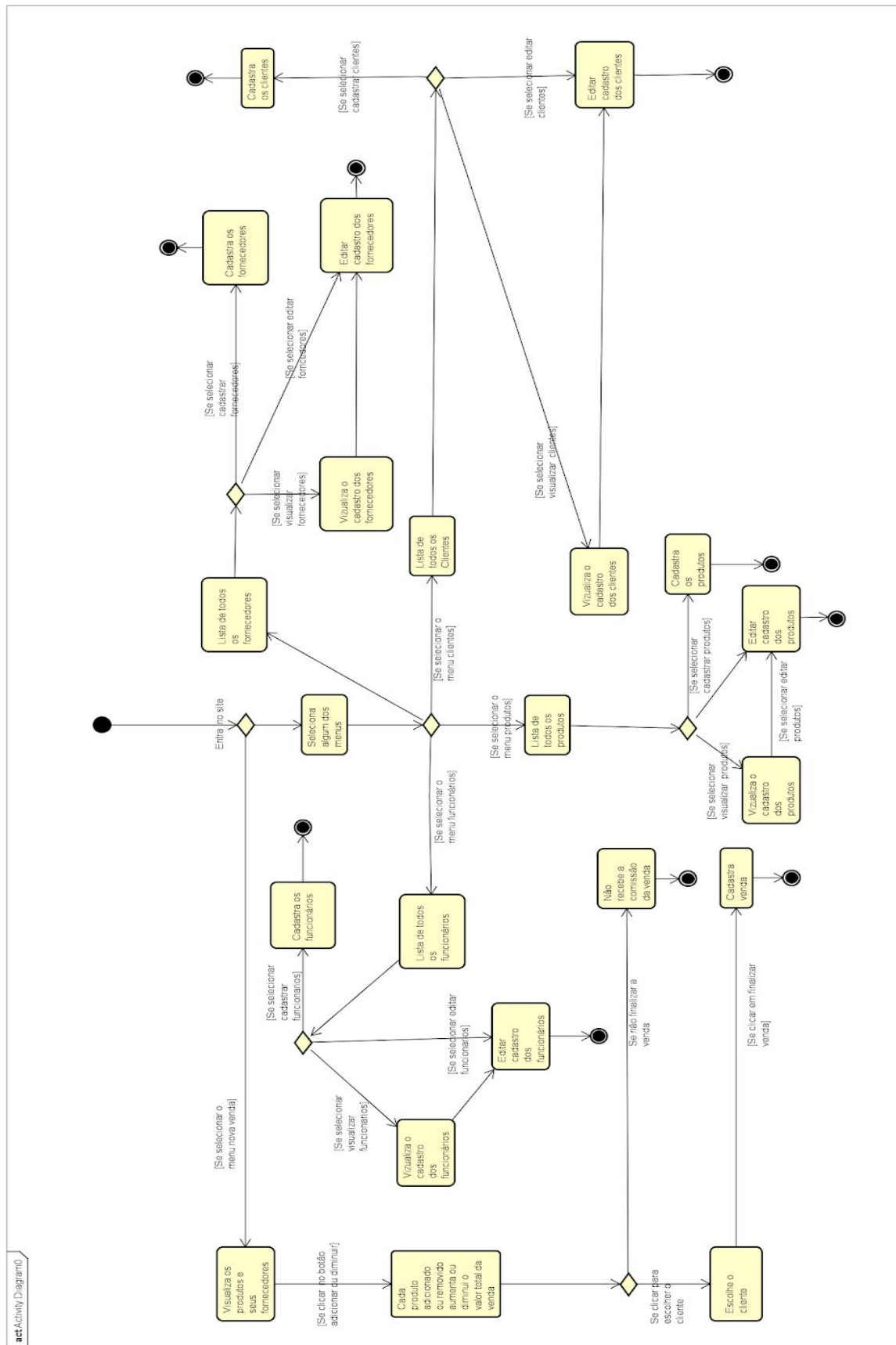


Figura 28 - Diagrama de atividade Módulo Gerente

## 6.7 Diagrama de Componentes e Implantação

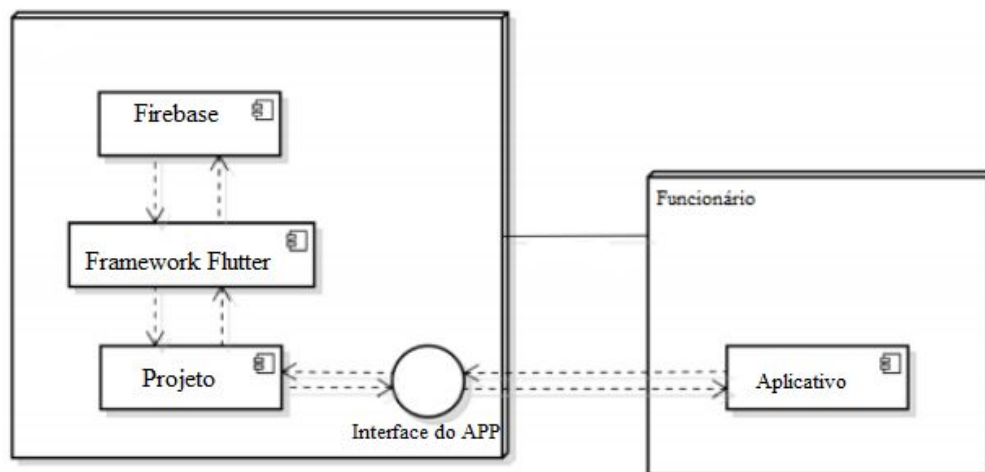


Figura 29 - Diagrama de Componentes e Implantação

## 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

Figura 30 - Abertura

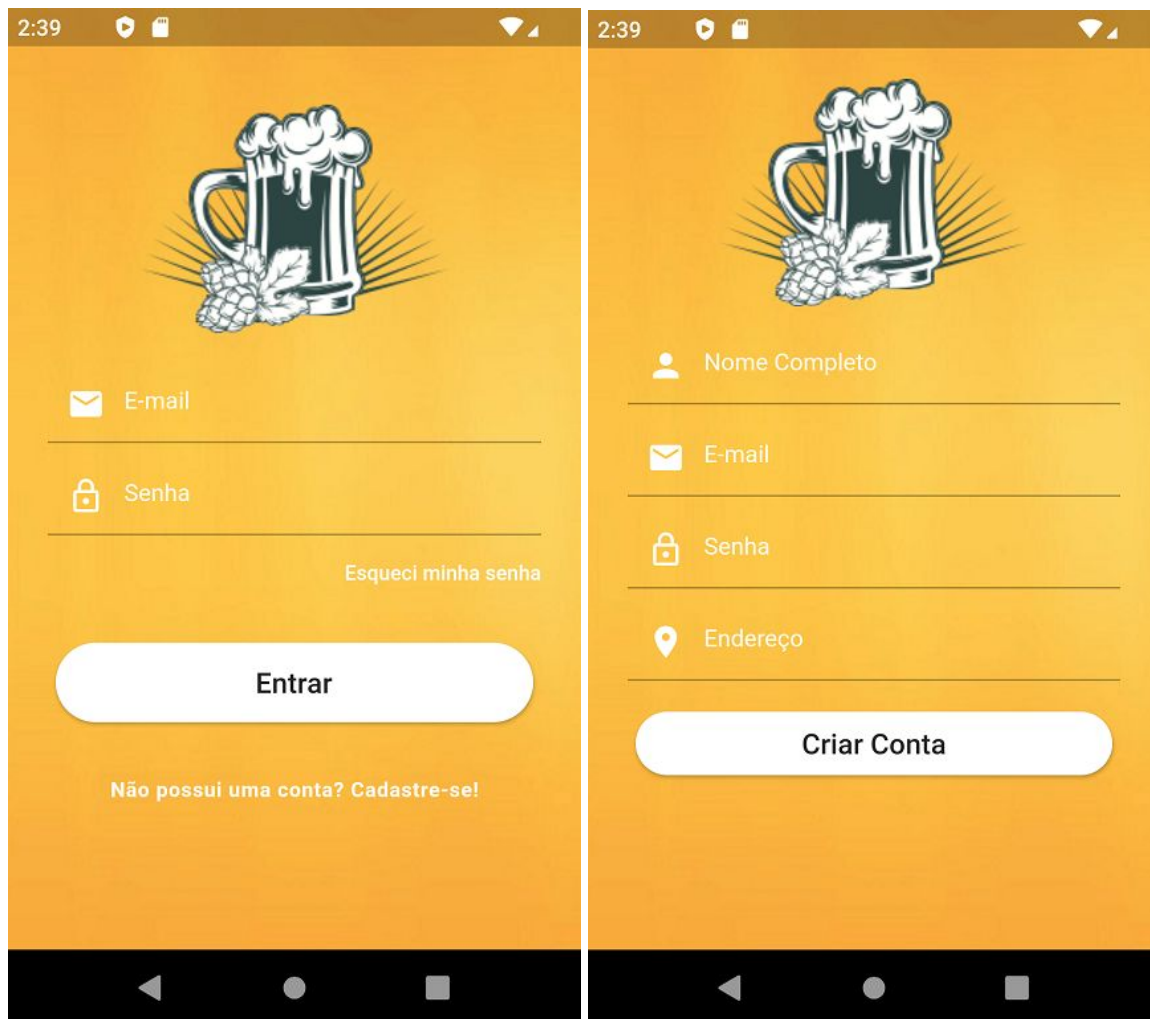


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

## 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.

Figura 31 - Autenticação

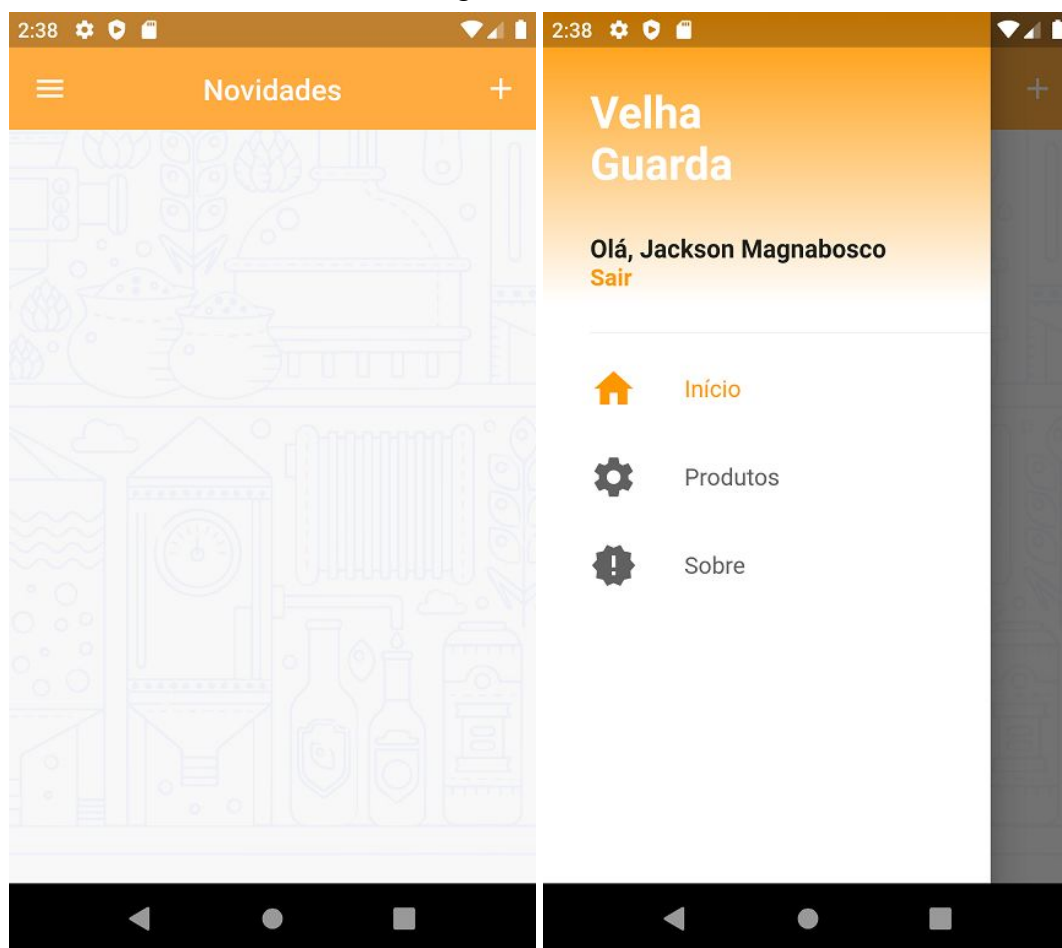


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

### 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.

Figura 32 -Inicial

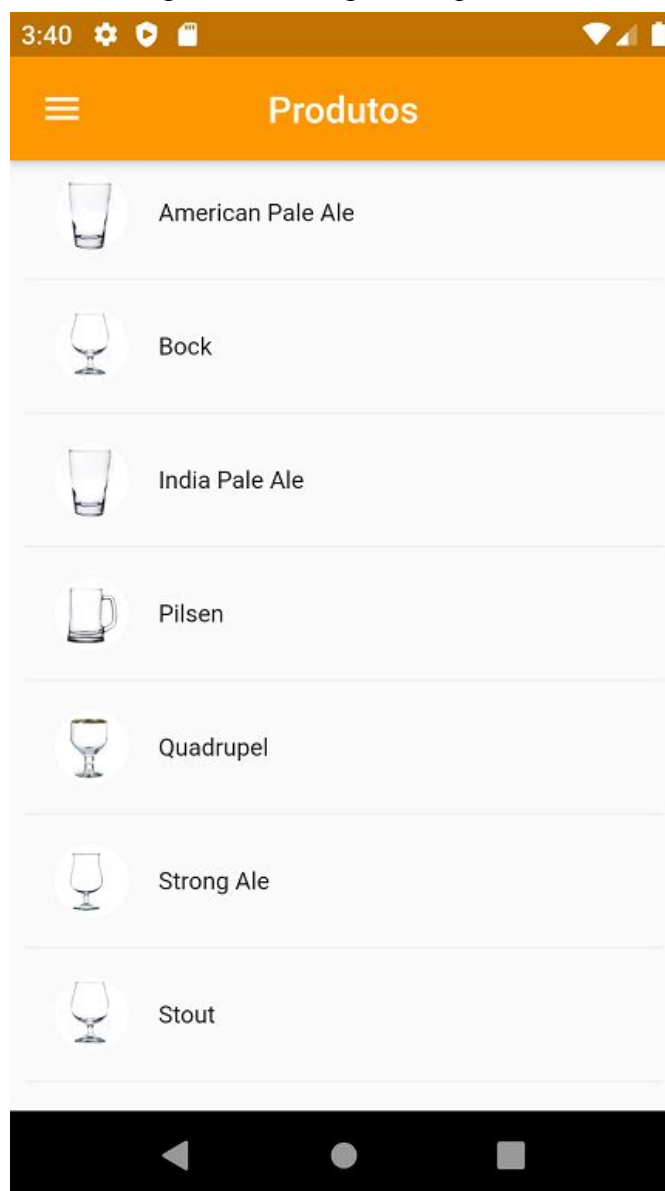


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

## 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.

Figura 33 - Categoria de produto

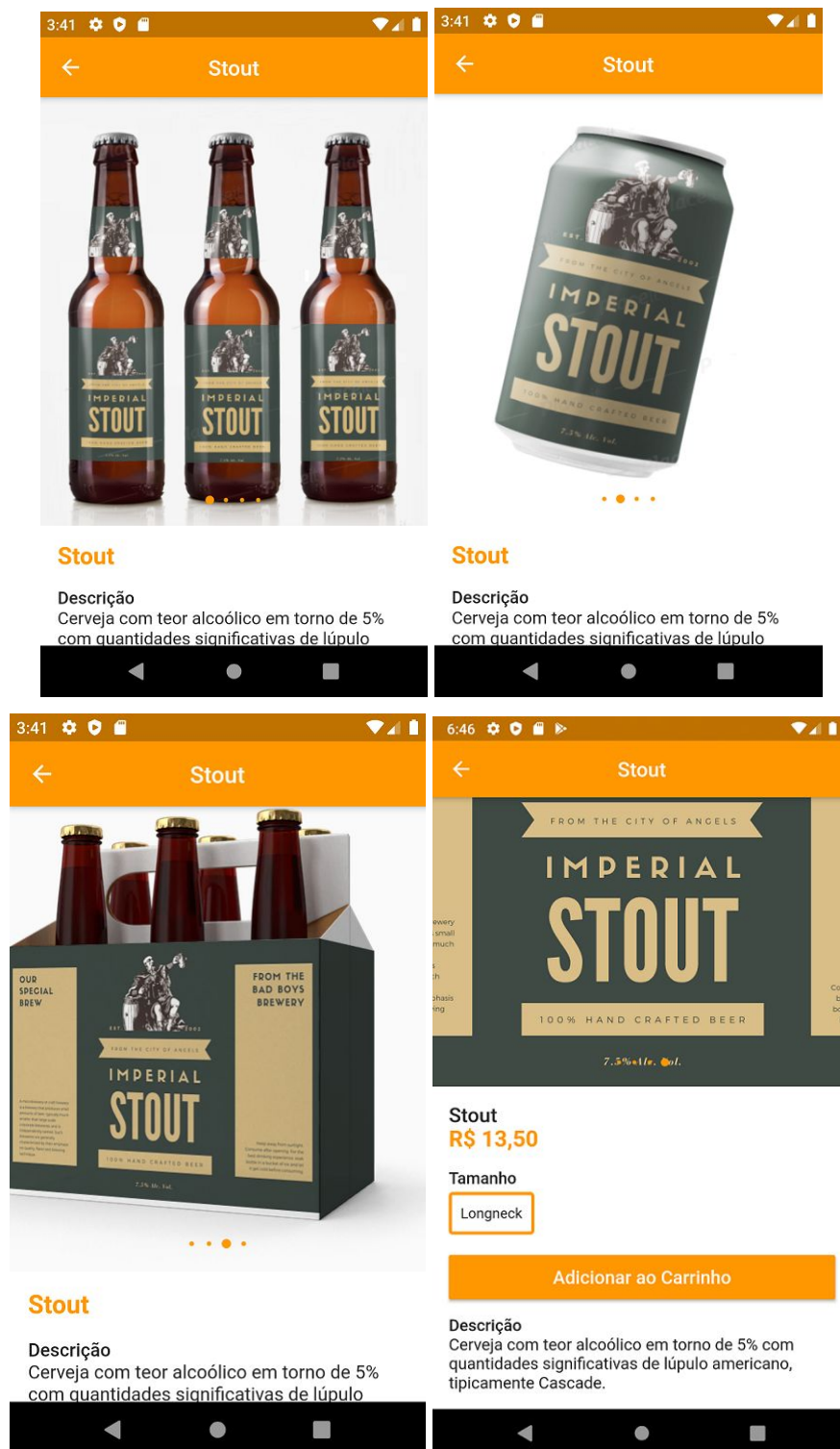


Fonte: Baseada na imagem original de (MAGNABOSCO, 2020)

## 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.

Figura 34 - 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.



Figura 35 - Carrinho de compra

2:17


← Stout

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

— 3 +

 **Calcular Frete** 

99711-278

 **Cupom de Desconto** +

Resumo do Pedido	
Subtotal	40.50
Desconto	0.00
Entrega	15.00
<b>Total</b>	<b>55.50</b>

**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 página inicial.

Figura 36 - Forma de pagamento

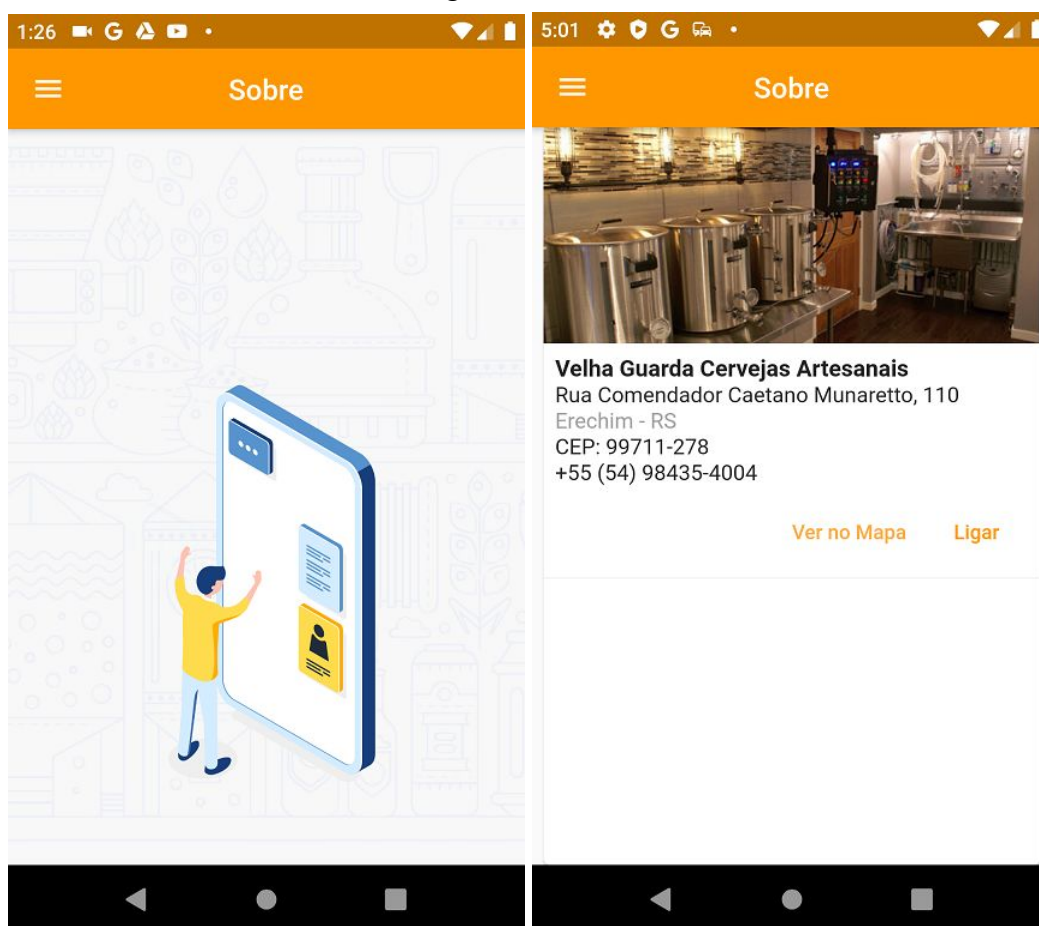
The figure displays four sequential screenshots of a mobile application's payment interface. The first two screenshots (top left and top right) show the initial card registration screen. The top half features a large orange card graphic with a masked number (\*\*\*\* \* 0000), the name 'NOME / SOBRENOME', and the validity date 'VALIDADE / DATA'. Below this, there are input fields for 'Número do cartão' and 'Validade', each with a 'Próximo' button. The bottom half contains a numeric keypad with digits 1-9, 0, a decimal point, and a confirmation checkmark. The third screenshot (bottom left) shows the next step where the CVV is entered. The card graphic now displays the name 'Jackson Magnabosco' and the CVV '999'. Below the CVV input field are 'Voltar' and 'Finalizar' buttons. The bottom half still has the numeric keypad. The fourth screenshot (bottom right) shows the final confirmation screen. It features a large green checkmark icon and a brown wallet icon, indicating successful payment. The background has a faint pattern of various food items.

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.

Figura 37 - Sobre



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.

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, 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.

## REFERÊNCIA

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

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

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

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

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

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

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).> 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.