



**Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"**

CRISTHIAN NUNES DIAS

APLICATIVO PARA CONTROLE FINANCEIRO UTILIZANDO FLUTTER

**Assis/SP
Ano 2020**



Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

CRISTHIAN NUNES DIAS

APLICATIVO PARA CONTROLE FINANCEIRO UTILIZANDO FLUTTER

Exame de Qualificação como requisito para o Trabalho de Conclusão de Curso apresentado ao curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Cristhian Nunes Dias
Orientador(a): Dr. Almir Rogério Camolesi

Assis/SP
Ano 2020

APLICATIVO PARA CONTROLE FINANCEIRO UTILIZANDO FLUTTER

CRISTHIAN NUNES DIAS

Exame de Qualificação como requisito para o Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Dr. Almir Rogério Camolesi

Examinador: _____
Dr. Osmar Aparecido Machado

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus que está sempre me dando forças para continuar apesar de todas as dificuldades, a minha mãe Vera Lúcia Nunes Dias e meu pai Aparecido Ferreira Dias que sempre me cobraram e me ajudaram para estudar e ter uma vida melhor. Da mesma forma dedico-o para meu amigo, professor e orientador Almir Rogério Camolesi por sempre confiar no meu potencial, me ensinar tudo que sei hoje e me dar forças para continuar.

AGRADECIMENTOS

Agradeço a Deus por sempre estar do meu lado, me ajudando e dando forças para continuar e por me dar condições físicas e psicológicas para trabalhar e continuar estudando. Por toda a experiência vivida e a sabedoria que me foi dada.

Agradeço aos meus pais Aparecido Ferreira Dias e Vera Lúcia Nunes Dias por sempre me ajudar a conquistar tudo que tenho hoje, e por me dar privilégios que foram cruciais para o término desse trabalho. Pela apoio incondicional que me proporcionaram desde sempre.

Agradeço aos meus amigos de trabalho e de faculdade que me ajudaram nesse caminho, pela paciência e pela confiança.

Agradeço principalmente ao meu amigo, professor e orientador Dr. Almir Rogério Camolesi, por sempre me ajudar a crescer profissionalmente e individualmente, por estar comigo nessa caminhada estressante, e toda vez utilizando seu vasto conhecimento para nos ensinar e motivar para melhorar. Pois trata-se de uma pessoa que quer o bem de seus alunos acima de tudo.

Agradeço à você que está lendo meu trabalho.

“Não espere o futuro mudar tua vida, porque o futuro é a consequência do presente.”

Racionais Mc's

RESUMO

As empresas hoje precisam de desenvolvimento produtivo e com uma ótima qualidade de software, e isso não é nenhuma novidade. Dito isso, foram lançadas várias tecnologias para auxiliar, porém a que será falada nesse trabalho é o Flutter, que através dele será desenvolvido um aplicativo de controle financeiro.

O objetivo do presente trabalho é o controle financeiro de forma simples e objetiva com ênfase no desenvolvimento com a ferramenta Flutter, de forma que, os leitores inspirem-se por essa nova tecnologia da Google.

Palavras-chave: Flutter, Dart, Firebase, Desenvolvimento Híbrido, UML, Astah Community, Visual Studio Code.

ABSTRACT

Companies today need productive development and excellent software quality, and this is nothing new. That said, several technologies were launched to assist, but the one that will be talked about in this work is the Flutter, which through it will be developed a financial control application.

The objective of the present work is the financial control in a simple and objective way with an emphasis on development with the Flutter tool, so that, be inspired by this new technology from Google.

LISTA DE ILUSTRAÇÕES

Figura 1: Diagrama da visão geral do sistema Flutter	17
Figura 2: Simulação de aplicação (Container vermelho)	18
Figura 3: Simulação de aplicação (Container vermelho + Row)	19
Figura 4: Ilustração do exemplo de Container, Column e Row	20
Figura 5: Simulação de aplicação (Container vermelho + Column + Row)	20
Figura 6: Código inicial	22
Figura 7: Tela do aplicativo	22
Figura 8: Código de inserção da AppBar	23
Figura 9: Tela da AppBar	23
Figura 10: Código de inserção do ícone menu	24
Figura 11: Tela da inserção do ícone	24
Figura 12: Código de alteração de cor do ícone menu	25
Figura 13: Tela da alteração de cor do ícone menu	25
Figura 14: Código de alteração do ícone	26
Figura 15: Tela da alteração do ícone	26
Figura 16: Código da chamada de função de campo de texto	27
Figura 17: Tela da inserção do campo de texto	27
Figura 18: Código da função para inserir campo de texto	27
Figura 19: Código da inserção do botão flutuante	28
Figura 20: Tela da inserção do botão flutuante	28
Figura 21: Código da inserção da barra inferior de navegação	29
Figura 22: Tela da inserção da barra inferior de navegação	29
Figura 23: Código da ancoragem do botão com a barra de navegação	30
Figura 24: Tela da ancoragem do botão com a barra de navegação	30

Figura 25: Imagem ilustrativa Firebase com uma lista compartilhada	34
Figura 26: Imagem ilustrativa Firebase com vários dispositivos	34
Figura 27: Imagem ilustrativa Firebase com uma lista compartilhada e vários dispositivos	35
Figura 28: Diagrama de Caso de Uso – Geral.....	37
Figura 29: Diagrama de Caso de Uso – Fazer login.....	38
Figura 30: Diagrama de Caso de Uso – Inserir receita ou despesa	39
Figura 31: Diagrama de Caso de Uso – Exibir transações.....	40
Figura 32: Diagrama de Caso de Uso – Notifica pagamento.....	41
Figura 33: Diagrama de Caso de Uso – Notifica vencimento	42
Figura 34: Diagrama de Classes da aplicação	43
Figura 35: Diagrama de Atividade para inserir despesa ou receita	44

SUMÁRIO

1. INTRODUÇÃO	12
1.1. OBJETIVOS	13
1.2. JUSTIFICATIVA	13
1.3. MOTIVAÇÃO	14
1.4. PERSPECTIVA DE CONTRIBUIÇÃO	14
1.5. METODOLOGIA	15
1.6. PÚBLICO ALVO	15
1.7. ESTRUTURA DO TRABALHO	16
2. ARQUITETURA FLUTTER	17
2.1. WIDGETS	18
3. DESENVOLVIMENTO COM FLUTTER	22
4. CONCEITOS E TECNOLOGIAS COMPLEMENTARES UTILIZADAS NO TRABALHO	31
4.1. UML	31
4.2. ASTAH COMMUNITY	32
4.3. DART	32
4.4. FIREBASE	33
4.5. VISUAL STUDIO CODE	35
5. DESENVOLVIMENTO DO PROJETO	36
5.1. DIAGRAMA DE CASO DE USO	36
5.2. DIAGRAMA DE CLASSES	42
5.3. DIAGRAMA DE ATIVIDADE	44
6. CRONOGRAMA	45
7. CONSIDERAÇÕES FINAIS	46
8. REFERÊNCIAS	47

1. INTRODUÇÃO

A tecnologia aprimora-se cada vez mais com o passar dos anos. O que antes era feito de forma manuscrita, hoje temos ferramentas na palma da mão que suprem nossas necessidades de maneira mais rápida e eficiente. Com essa grande evolução, conforme Paula (2017) tanto pessoas quanto empresas estão investindo em aplicações móveis para facilitarem o seu dia a dia.

Dito isso, a procura de aplicações fluídas para controle de gastos e planejamento financeiro está cada vez maior. Com base nessas observações, é proposto o desenvolvimento de uma aplicação para gerenciar tais transações.

No desenvolvimento, a administração do tempo é muito importante, pois se feito de maneira correta, a produtividade aumenta significativamente. Hoje no mercado há várias ferramentas para nos auxiliarem, e uma delas é o Flutter¹, que traz uma facilidade enorme em desenvolver aplicativos móveis de uma forma híbrida sem precisar retrabalhar no código.

Segundo Magalhães (2019) o Flutter tem seu próprio framework de processamento, o que quer dizer que ele não dependerá de nada específico de cada plataforma. Todos os efeitos que há no sistema operacional IOS² e Android³ estão incluídos na ferramenta. Nele também contém uma opção de visualização em segundos de todas alterações feitas sem precisar recompilar o aplicativo no celular, isso tudo de forma rápida e simples para o desenvolvimento da aplicação móvel de finanças.

Os aplicativos que se sobressaem no mercado hoje, são os com design mais trabalhados, com animações que chamam atenção e de uma usabilidade rápida. A Tecnologia Flutter contém tudo isso, sendo escalável, pois permite uma fácil manutenção de código fonte.

¹ <https://flutter.dev/> Acessado em 19 nov. 2019.

² É um sistema operacional de dispositivos móveis da Apple.

³ É um sistema operacional baseado no núcleo Linux, sendo seu maior colaborador o Google.

1.1. OBJETIVOS

Um dos objetivos com o desenvolvimento e implementação desse trabalho é produzir por meio de um aplicativo híbrido, informações que auxiliará o usuário a fazer o controle monetário, pois servirá como suporte financeiro pessoal de diversas despesas pessoais, pois nele será contido as funcionalidades de inserção, remoção, exclusão e edição de dados de uma maneira simplificada para que não seja difícil de utilizar e muito menos demorado. Conterá também uma interface fluída, elegante, sempre priorizando o desempenho.

Além da aplicação móvel, também terá como um dos focos expor mais sobre a ferramenta Flutter, pois trata-se de uma tecnologia de fácil aprendizagem e com um ótimo desempenho por se utilizar comunicação nativa com o dispositivo.

1.2. JUSTIFICATIVA

Este trabalho se justifica pela necessidade de aprimorar a produtividade em programação móvel híbrida, utilizando todo o conhecimento adquirido pelo curso de Análise e Desenvolvimento de Sistema e juntamente com a indispensabilidade da aplicação de controle monetário pessoal. Como mencionado anteriormente, a aplicação será para o controle e planejamento financeiro, auxiliando no desenvolvimento rápido e fluído com um melhor aproveitamento de códigos.

Outra defesa para esse trabalho, dá-se pela ausência de informações para essa nova tecnologia, que está crescendo pelo decorrer dos anos. Essa ferramenta mencionada tem como finalidade principal o desenvolvimento híbrido de aplicações, que só necessita de uma linguagem de programação para ser feita, dito isso, com apenas um código, é possível instalar e executar em qualquer dispositivo móvel, que conseqüentemente acaba sendo obtido um aumento significativo na produtividade e ao mesmo tempo expondo seus pontos positivos e negativos para futuros desenvolvedores. Essa ferramenta será o Flutter que

utiliza a linguagem de programação Dart⁴ para fins de obter ótimos resultados de desempenho na execução da aplicação e velocidade no seu desenvolvimento.

1.3. MOTIVAÇÃO

A motivação para a produção desse trabalho surgiu da necessidade de gerar aplicações rápidas, com design elegante, com uma altíssima fluidez de animações, com código híbrido, executando tanto em IOS quanto em Android e para conclusão do curso de Análise e Desenvolvimento de Sistemas.

Além disso, toda a experiência que foi aprendida em engenharia de software teve um papel primordial para definição de todo o projeto, pois por se tratar de tecnologias que não usamos no curso, consegue-se desenvolver a aplicação, pois o conceito utilizado foi o mesmo.

E por fim, a maior motivação é aprender uma nova tecnologia e desenvolver uma aplicação que me ajudará a crescer profissionalmente, com a finalidade de não ser um trabalho em vão.

1.4. PERSPECTIVA DE CONTRIBUIÇÃO

Com esse trabalho, pretende-se mostrar como é rápido gerar interfaces gráficas profissionais, animações nativas da própria ferramenta, sem precisar digitar muitos códigos comparando com ferramentas concorrentes para geração do aplicativo financeiro.

Será contribuído para comunidade também as vantagens e desvantagens de utilizar a ferramenta Flutter para desenvolver aplicações.

Como a ferramenta é nova, a agregação para a área de estudo é alta, pois não há muito material disponível (artigos, livros, tutoriais, cursos, etc.) sobre o Flutter.

⁴ <https://dart.dev/> Acessado em 07 Abr. 2020.

1.5. METODOLOGIA

O projeto será desenvolvido com uma análise realizada por diversas pessoas que tem interesse de fazer o controle financeiro. A metodologia da análise e a própria implementação será feita orientada a objetos. Será feito todo em cima de uma engenharia de software. Este estudo de caso será feito com base em pessoas que precisam se planejar financeiramente, com agregação de um levantamento de requisitos, ou seja, as funcionalidades que necessitam em uma aplicação para os ajudar.

Para fazer os casos de usos, diagrama de atividades, diagrama de classes será usado a linguagem UML com base no programa Astah⁵, nele consegue-se realizar trabalhos com uma agilidade maior por ser simples e objetivo. Para o desenvolvimento do aplicativo em geral será utilizado Flutter que tem Dart como a linguagem principal. O banco de dados será feito no Firebase⁶, que é em nuvem e pode ser local ao mesmo tempo, pois se o aplicativo não tiver nenhum tipo de conexão com a internet, o banco se encarregará de salvar os dados locais até que seja feita uma conexão. E para escrever os códigos será utilizado o programa Visual Studio Code⁷.

1.6. PÚBLICO ALVO

O aplicativo que foi desenvolvido nesse trabalho tem como público principal qualquer pessoa que deseja manter seu planejamento financeiro de uma forma eficiente e prática. Visa também alcançar novos profissionais na área de desenvolvimento híbrido que pretendem-se iniciar na programação em Flutter ou em tecnologias similares. Dito isso, servirá tanto para pessoas que não sabem programação e querem o aplicativo para seu controle pessoal, quanto para desenvolvedores que querem aprender mais sobre essa ferramenta recente ou aprender mais sobre desenvolvimento híbrido.

⁵ <https://astah.net/> Acessado em 07 Abr. 2020.

⁶ <https://firebase.google.com/> Acessado em 07 Abr. 2020.

⁷ <https://code.visualstudio.com/> Acessado em 07 Abr. 2020.

1.7. ESTRUTURA DO TRABALHO

Este trabalho está dividido em capítulos. No primeiro capítulo é definido por introdução, objetivos, justificativas, motivação, perspectiva de contribuição, metodologia, público alvo e estrutura do trabalho.

O segundo capítulo explana sobre as tecnologias que foram utilizadas para o desenvolvimento desse trabalho.

No terceiro capítulo tem como ênfase explicar como funciona a arquitetura do Flutter e com uma profundidade de como funciona seu desenvolvimento.

O quarto capítulo é apresentando o desenvolvimento do projeto, onde é explicado como funciona os diagramas para se iniciar e comandar um projeto.

Desenvolvimento com Flutter é o quinto capítulo, onde é apresentado a facilidade em desenvolver com a ferramenta com um exemplo muito usando em inícios de aplicações.

No sexto capítulo é exibido o cronograma do projeto inteiro. Nele podemos ver o que foi executado como verde e o que não foi executado como azul.

Capítulo sete é apresentado as considerações finais do trabalho.

Por fim, o capítulo oitava consta as referências usadas nesse trabalho.

2. ARQUITETURA FLUTTER

Para o maior entendimento da tecnologia abordada nesse trabalho, nessa sessão será exibida a arquitetura do Flutter. A figura 4 abaixo mostra a visão geral do sistema.

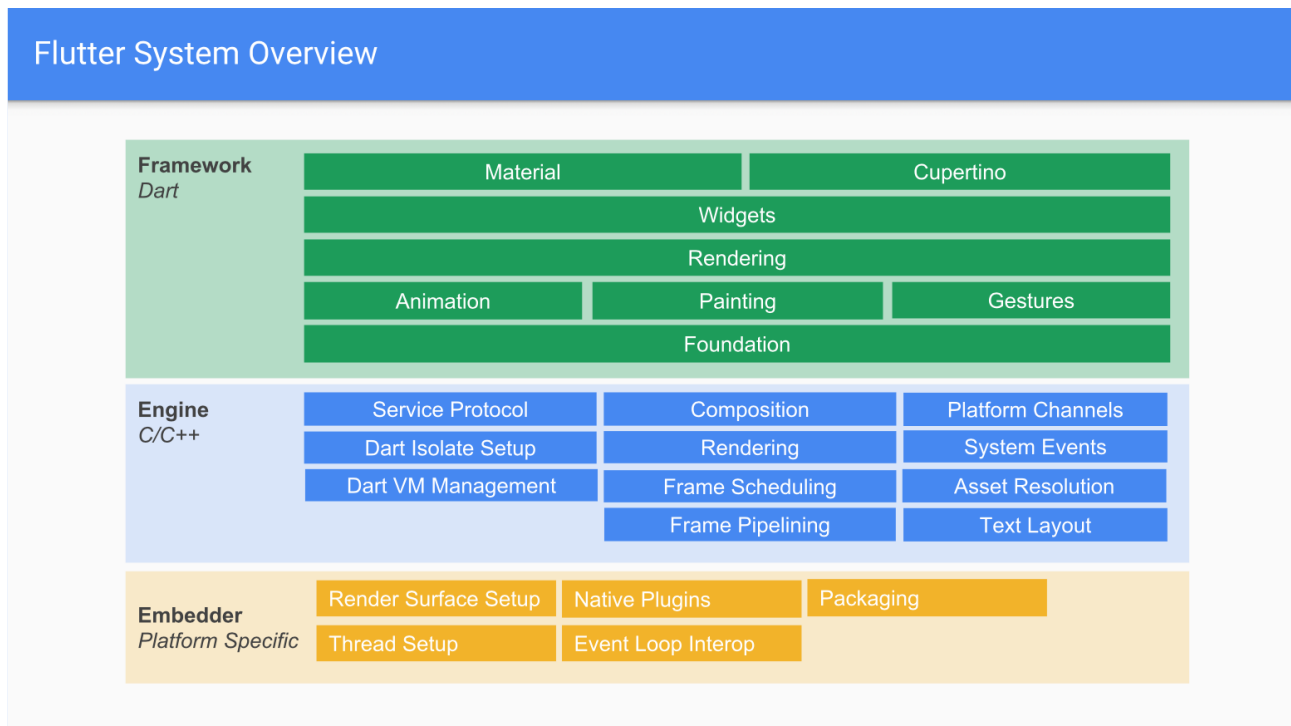


Figura 1: Diagrama da visão geral do sistema Flutter

Após a leitura da figura acima, tira-se a conclusão que o Flutter tem 3 camadas para seu funcionamento. A primeira camada em verde é toda a estrutura do Flutter escrita em Dart, onde será executado o trabalho do desenvolvedor, essa parte é muito importante que seja bem entendida para quem for desenvolver. A segunda camada em azul, é onde o desenvolvedor não precisa se preocupar, pois é o núcleo da ferramenta, onde fica o cérebro do mesmo. Nessa camada é onde fica designada para resolver toda parte gráfica para cada sistema, que é o ponto diferencial do Flutter comparada com outras tecnologias concorrentes. Na terceira camada em amarelo é onde fica uma ponte dinâmica entre a ferramenta e o sistema do dispositivo, sem a necessidade de ter uma ponte específica para cada sistema.

2.1. WIDGETS

Tudo em Flutter é lido como um Widget, como se fosse um recipiente dentro de outros recipientes. A finalidade dessa estratégia segundo Magalhães (2019) é para não haver herança e sim composições a fins de facilitar reaproveitamento de componentes já existentes.

Vamos imaginar que temos que criar um aplicativo que cada linha deve haver um texto, que seria “Texto 1”, “Texto 2” e “Texto 3”. De uma forma simples, o primeiro passo do aplicativo será iniciar com um *Container* com fundo vermelho, pois nele que seria armazenado as informações de texto. Obrigatoriamente precisa haver um *Widget* para iniciar o aplicativo. Na figura 5 será a simulação de forma didática de como ficaria esse primeiro processo.

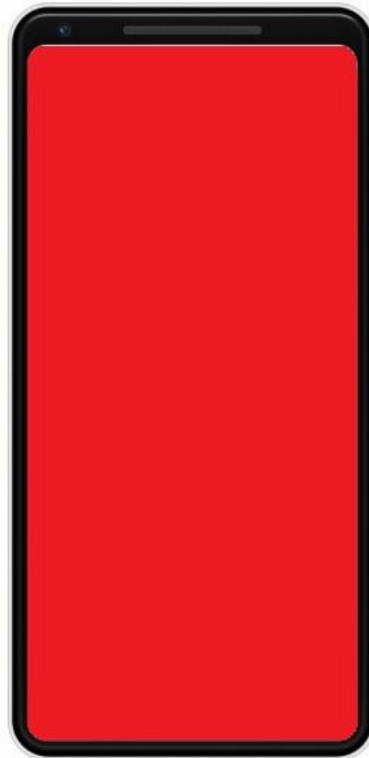


Figura 2: Simulação de aplicação (Container vermelho)

O *Container* pode ter qualquer altura, largura, imagem de fundo e cor, até mesmo gradiente de cores. Nele há uma propriedade que se chama *child*, onde é inserido outro *Widget* para fazer parte do aplicativo, que no nosso exemplo seria os textos, porém eu não posso inserir os 3 textos em *child*, pois essa propriedade somente aceita um *Widget* e para cada texto precisamos de um *Widget*. Para solucionar esse problema poderíamos inserir um *Widget*

chamado *Row* no *child* de *Container* onde nele é aceito mais de um *Widget*, pois nele há a propriedade chamada *children*, que resolveria nosso problema inicial, segue figura 6 exibindo esse exemplo:

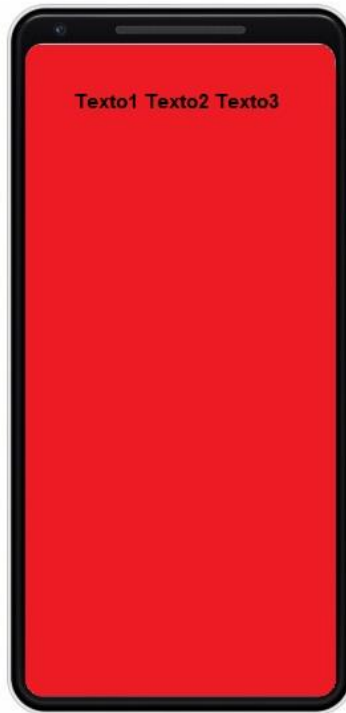


Figura 3: Simulação de aplicação (Container vermelho + Row)

Percebe-se pela explicação e imagem acima, que tudo que tiver *child* pode haver somente um *Widget* e o que tiver *children* pode ser inserir mais de um, porém no nosso exemplo os textos deveriam ficar um em cada linha e para isso, ao invés de chamar de primeiro momento o *Widget Row* que trabalha na horizontal, podemos chamar o *Widget Column* que trabalha na vertical e também tem a propriedade *children* e para cada *children* pode-se ser inserido uma *Row* e dentro de cada *Row* inserir um Texto. Segue figura 7 para entendimento onde laranja claro é o *Container*, azul claro é a *Column* e as *Row* são os verdes:

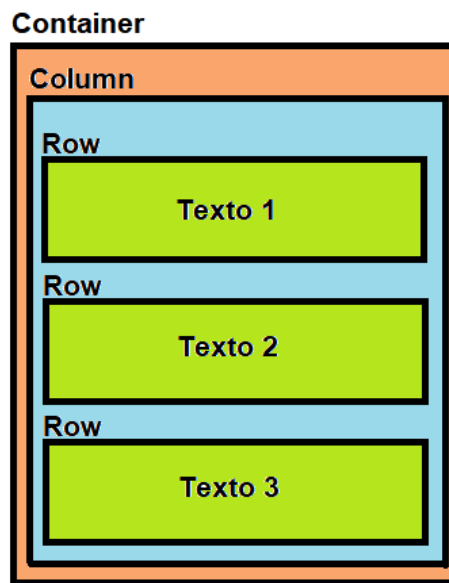


Figura 4: Ilustração do exemplo de Container, Column e Row

E com isso finalizaríamos nosso exemplo, e ficaria como a figura 8:

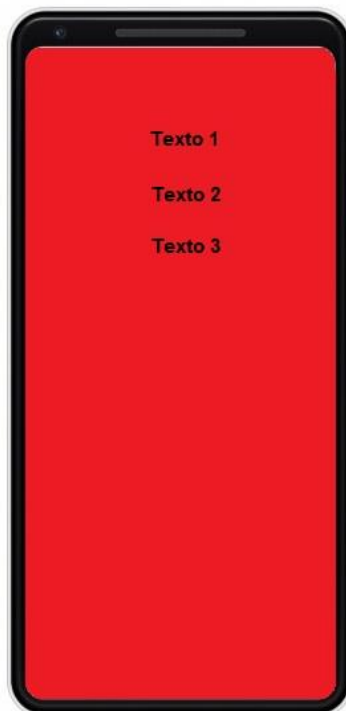


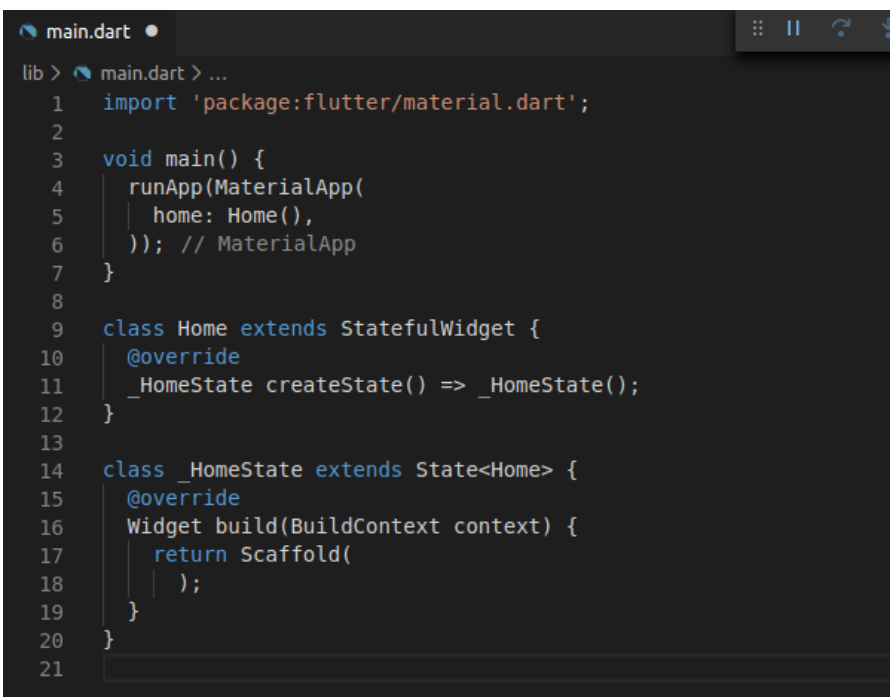
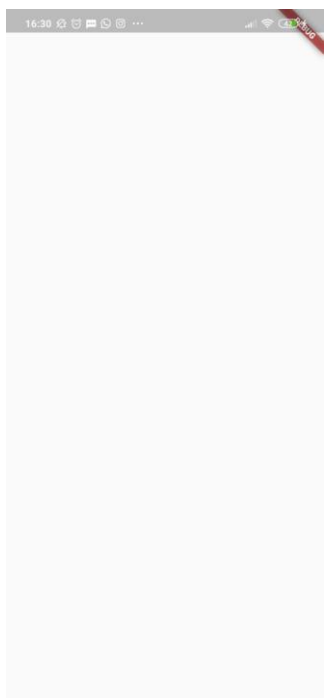
Figura 5: Simulação de aplicação (Container vermelho + Column + Row)

Conclui-se com esse exemplo que tudo que há na interface gráfica do aplicativo deve ser um Widget e para cada Widget há suas propriedades específicas, sendo assim, o desenvolvedor terá que analisar qual melhor Widget para o caso atual dele.

3. DESENVOLVIMENTO COM FLUTTER

Nessa sessão será apresentada facilidade de desenvolvimento utilizando Flutter, passo a passo como fazer uma interface gráfica de inserção de nome, sobrenome e idade com botão de adicionar, porém sem funcionalidades.

Do lado esquerdo ficará o código da aplicação e do lado direito a interface gráfica que é referente ao código lateral, isso será padrão para todas imagens abaixo dentro dessa sessão.

 <pre>lib > main.dart > ... 1 import 'package:flutter/material.dart'; 2 3 void main() { 4 runApp(MaterialApp(5 home: Home(), 6)); // MaterialApp 7 } 8 9 class Home extends StatefulWidget { 10 @override 11 _HomeState createState() => _HomeState(); 12 } 13 14 class _HomeState extends State<Home> { 15 @override 16 Widget build(BuildContext context) { 17 return Scaffold(18); 19 } 20 } 21</pre>	
<p>Figura 6: Código inicial</p>	<p>Figura 7: Tela do aplicativo</p>

Como mostrada na figura 17, percebe-se que temos um método **main** na aplicação onde chama outro método **Home** que cria um estado para o método **_HomeState**. Esse método que é criado o estado retorna os *Widgets* que desenharam o aplicativo. Como não há nada retornando para desenhar no aplicativo, o próprio fica em branco.

```
class _HomeState extends State<Home> {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Test app"),  
      ), // AppBar  
    ); // Scaffold  
  }  
}
```

Figura 8: Código de inserção da AppBar

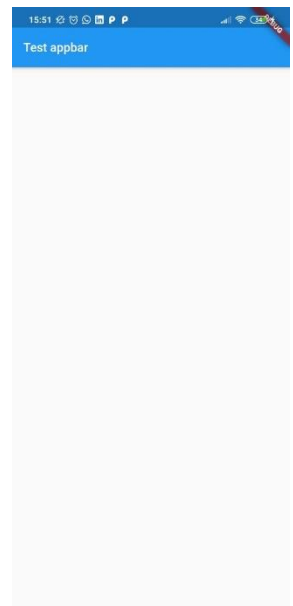


Figura 9: Tela da AppBar

Para criação de uma *AppBar*⁸ precisa obrigatoriamente ter como pai um *Scaffold*, que é um *Widget*, tudo em Flutter é feito com *Widget* dentro de outro *Widget*. Passamos como parâmetro dentro do *AppBar* um texto que será mostrado na barra. Para ficar claro o entendimento da velocidade do desenvolvimento, essa *AppBar* foi criada em 26 segundo, rodando instantaneamente no aplicativo em depuração.

⁸AppBar: Sua tradução é Barra de Aplicativos.

```
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Test appbar"),
        backgroundColor: Colors.green,
        leading: IconButton(
          onPressed: (){},
          icon: Icon(Icons.menu),
        ), // IconButton
      ), // AppBar
    ); // Scaffold
  }
}
```

Figura 10: Código de inserção do ícone menu

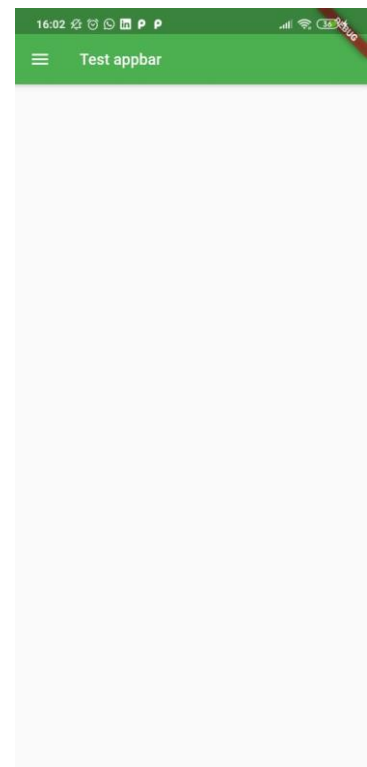


Figura 11: Tela da inserção do ícone

Para deixar um aplicativo mais atraente, foi inserido um ícone de menu na barra do aplicativo. Chamamos um *leading* dentro da própria *AppBar* e colocamos o ícone de botão com um método vazio e chamando a imagem do menu.


```

@override
_HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Test appbar"),
        backgroundColor: Colors.green,
        leading: IconButton(
          onPressed: (){},
          icon: Icon(Icons.menu),
          color: Colors.orange,
        ) // IconButton
      ), // AppBar
    ); // Scaffold
  }
}

```

Figura 12: Código de alteração de cor do ícone menu

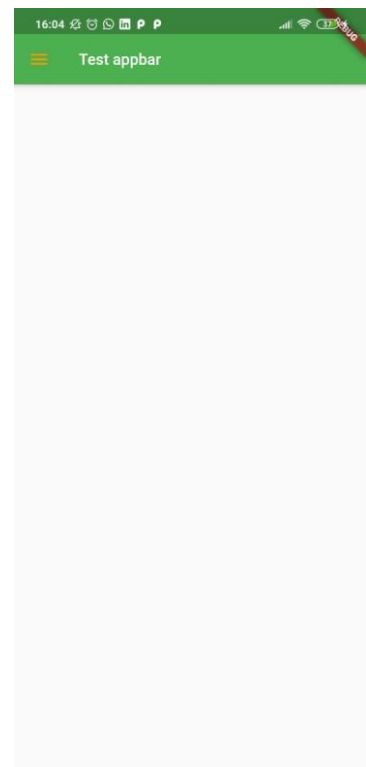


Figura 13: Tela da alteração de cor do ícone menu

Na figura 23 mostra-se o código para trocar a cor do ícone se achar necessário. No quesito cor, o Flutter contém várias cores definidas, porém se não achar a ideal, tem a opção de inserir o código “rgb” da cor requerida.

```
class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Test appbar"),
        backgroundColor: Colors.green,
        leading: IconButton(
          onPressed: (){},
          icon: Icon(Icons.save),
        ) // IconButton
      ), // AppBar
    ); // Scaffold
  }
}
```

Figura 14: Código de alteração do ícone

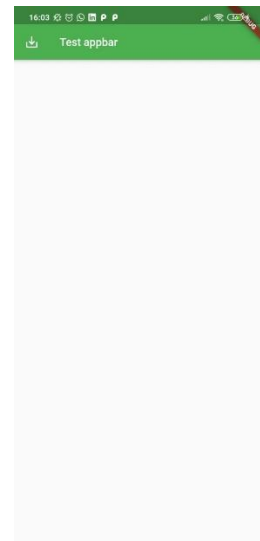


Figura 15: Tela da alteração do ícone

Na figura é exibida algumas das opções de ícones que começam com a letra s. No próprio site da ferramenta há todas as imagens dos ícones e seus nomes para poder solicitar.

```

appBar: AppBar(
  title: Text("Test appbar"),
  backgroundColor: Colors.black,
  leading: IconButton(
    onPressed: (){},
    icon: Icon(Icons.menu),
    color: Colors.white,
  ) // IconButton
), // AppBar
body: Container(
  padding: EdgeInsets.only(left: 40, right: 40, top: 10),
  child: Column(
    children: <Widget>[
      retornaTextField("Nome"),
    ], // <Widget>[]
  ), // Column
), // Container
); // Scaffold
}

```

Figura 16: Código da chamada de função de campo de texto

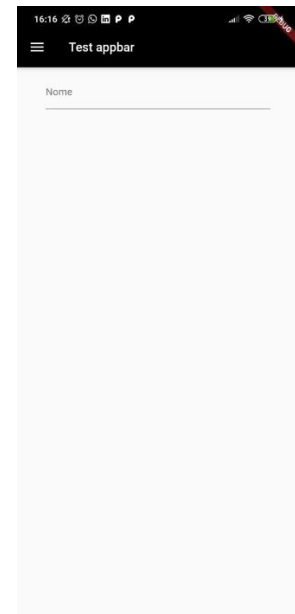


Figura 17: Tela da inserção do campo de texto

Para inserir uma caixa de texto, foi chamada uma função com o parâmetro do tipo texto e que retorna um *Widget TextField* que é o que foi mostrado na figura ao lado.

```

Widget retornaTextField(String descricao) {
  return TextField(
    decoration: InputDecoration(
      hintText: descricao,
    )); // InputDecoration // TextField
}

```

Figura 18: Código da função para inserir campo de texto

Na função **retornaTextField** é retornado um *Widget TextField*. Dentro desse *Widget* é inserido uma decoração de dica de texto, com finalidade de auxiliar o usuário a saber o que inserir no campo.

```
        retornaTextField("Nome"),
        SizedBox(height: 20.0,),
        retornaTextField("Sobrenome"),
        SizedBox(height: 20.0,),
        retornaTextField("Idade"),
        SizedBox(height: 20.0,),
      ], // <Widget>[]
    ), // Column
  ), // Container
  floatingActionButton: FloatingActionButton(
    child: Icon(Icons.add),
    onPressed: (){},
    backgroundColor: Colors.black,
  ), // FloatingActionButton
```

Figura 19: Código da inserção do botão flutuante

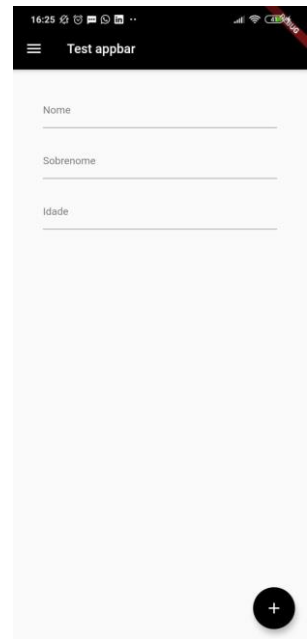


Figura 20: Tela da inserção do botão flutuante

Com o intuito de deixar um botão padronizado na tela, foi criado um botão flutuante que independentemente do tamanho da tela e de informações que nela contém ele sempre ficará fixo. O botão poderá ficar no lado direito, esquerdo ou centro, com qualquer ícone de qualquer cor.

```

        retornaTextField("Nome"),
        SizedBox(height: 20.0,),
        retornaTextField("Sobrenome"),
        SizedBox(height: 20.0,),
        retornaTextField("Idade"),
        SizedBox(height: 20.0,),
      ], // <Widget>[]
    ), // Column
  ), // Container
  floatingActionButton: FloatingActionButton(
    child: Icon(Icons.add),
    onPressed: (){},
    backgroundColor: Colors.black,
  ), // FloatingActionButton
  bottomNavigationBar: BottomAppBar(
    color: Colors.black,
    child: Container(
      height: 50.0,
    ), // Container
  ), // BottomAppBar
]; // Scaffold
}
}

```

Figura 21: Código da inserção da barra inferior de navegação

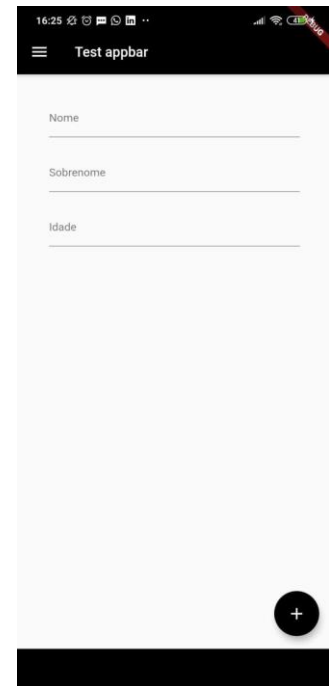


Figura 22: Tela da inserção da barra inferior de navegação

Para deixar o aplicativo de exibição mais elegante, foi inserido como mostra na figura 32 uma barra de navegação inferior, nela poderá também existir ícones e funcionalidades, porém com a inserção dessa barra o botão acabou subindo e ficando um pouco deslocado, porém isso tem resolução e será exibido na próxima imagem.

```

        return TextField( Sobrenome //
        SizedBox(height: 20.0),
        retornaTextField("Idade"),
        SizedBox(height: 20.0),
    ], // <Widget>[]
    ), // Column
  ), // Container
  floatingActionButton: FloatingActionButton(
    child: Icon(Icons.add),
    onPressed: () {},
    backgroundColor: Colors.black,
  ), // FloatingActionButton
  bottomNavigationBar: BottomAppBar(
    color: Colors.blueGrey,
    child: Container(
      height: 50.0,
    ), // Container
  ), // BottomAppBar
  floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
); // Scaffold
}
}

```

Figura 23: Código da ancoragem do botão com a barra de navegação



Figura 24: Tela da ancoragem do botão com a barra de navegação

Para finalizar o aplicativo de exibição, foi resolvido o problema da barra inferior com o botão flutuante fazendo a chamada do *FloatingActionByttonLocation* e chamando o **centerDocked** para se misturar com a barra inferior.

O desenvolvimento dessa simples tela foi feita em 5 minutos e 49 segundos. Percebe-se a alta velocidade de desenvolvimento da ferramenta Flutter para uma tela simples de inserção de dados. Acredita-se que com pouco aprendizado qualquer pessoa que tem o mínimo de conceito poderá fazer o aplicativo sem nenhum problema.

4. CONCEITOS E TECNOLOGIAS COMPLEMENTARES UTILIZADAS NO TRABALHO

Nessa sessão será explanada sobre as tecnologias que foram utilizadas para desenvolver esse trabalho por completo.

A principal tecnologia estudada é o Flutter, que contém a linguagem de programação Dart, pois trata-se de uma linguagem de compilação eficaz. Firebase terá como responsabilidade ser o banco de dados do aplicativo, pois é um banco em nuvem que promete ser rápido. Por fim, para entendimento melhor da aplicação, será utilizado a linguagem UML feita no programa Astah Community.

Para a compreensão mais aprofundada das tecnologias citadas anteriormente, será explicada detalhadamente cada uma abaixo:

4.1. UML

Durante o trabalho de modelagem da aplicação serão desenvolvidos os diagramas conforme a linguagem UML. Para Portalgsi (2011?) UML é uma linguagem de modelagem que é utilizada para fazer as modelagens de objetos do mundo real. Essa linguagem é para auxiliar no desenvolvimento de todos tipos de sistemas para facilitar o entendimento do mesmo em forma de “desenhos”.

4.2. ASTAH COMMUNITY

Por fim, para auxiliar na produção de diagramas da linguagem UML será utilizada a ferramenta Astah Community. E conforme Lima (2016):

Astah Community é um software para modelagem UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada) com suporte a UML 2, desenvolvido pela Change Vision, Inc e disponível para sistemas operacionais Windows 64 bits. Anteriormente conhecido por JUDE, um acrônimo de Java and UML Developers Environment (Ambiente para Desenvolvedores UML e Java).

4.3. DART

Será necessário uma linguagem de programação para o desenvolvimento do aplicativo, e será utilizada a linguagem Dart que segundo Guedes (2019), é uma linguagem de programação que foi criada pela empresa Google que é fortemente tipada, que significa que todos as características de um objeto deve ser informado em sua declaração. O objetivo principal dessa linguagem foi para substituir o JavaScript em criações de aplicações web. Mas, sua evolução foi mais do que esperado pelos desenvolvedores, fez com que ela se tornasse uma linguagem multi-paradigma.

A linguagem não obteve sucesso em seu objetivo inicial que era para substituir o JavaScript, mas com a grande evolução do Flutter, o Dart voltou a ganhar o público novamente.

Essa tecnologia foi inspirada na linguagem C, sendo assim, facilitando migrações de desenvolvedores que já trabalham com Java, C# e outras linguagens similares.

4.4. FIREBASE

Será utilizado Firebase⁹ para fazer o armazenamento do banco de dados NoSQL. Conforme Viana (2017) Firebase é uma plataforma web de desenvolvimento que foi adquirida pela Google e tem seu foco no back-end¹⁰ de fácil manuseio e de uma enorme facilidade e usabilidade. Existem diversos recursos que essa ferramenta nos auxilia no desenvolvimento e gerenciamento de aplicações, como um banco de dados em tempo real e autenticações através de contas da própria Google.

Além disso, o que chama a atenção nesse banco de dados é que se caso o dispositivo não tiver conexão com a internet, automaticamente é salvo no aplicativo uma cópia dos dados que foram alterados durante o uso, e quando estabelecer uma conexão os dados serão atualizados na nuvem.

Como trata-se de uma tecnologia NoSQL, o desenvolvedor do aplicativo que estiver usando o Firebase como banco, segundo Maes (2015?) terá um ganho significativo em tempo, pois não será necessário criar estruturas gigantes de implementação relacional e para a implementação da mesma. Sendo assim, sua implementação será rápida e fácil, focando o tempo em problemas mais importantes.

⁹ <https://firebase.google.com/>

¹⁰ Etapa de desenvolvimento responsável em implementar as regras de negócios da aplicação.

A figura 1 explana um exemplo do poder do Firebase, onde um ou mais dispositivos conversam com a ferramenta que faz o serviço de juntar os dados para que todos consigam ver as mesmas informações em tempo real. O dispositivo manda informação para o Firebase que é encarregado de popular a lista com essa informação e depois retorna para os demais dispositivos.

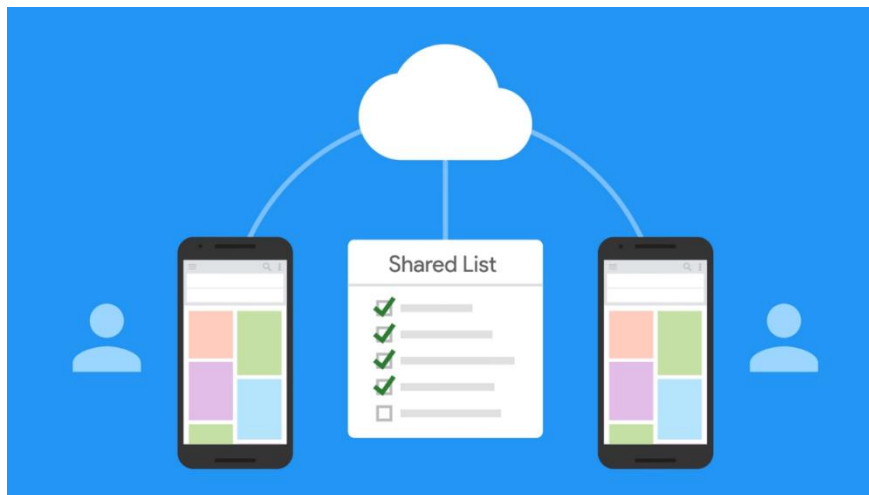


Figura 25: Imagem ilustrativa Firebase com uma lista compartilhada

Já a figura 2 tem um exemplo diferente, porém bem parecido. Os dispositivos mandam informações para o Firebase e o mesmo retorna esses dados para todos os dispositivos conectados.

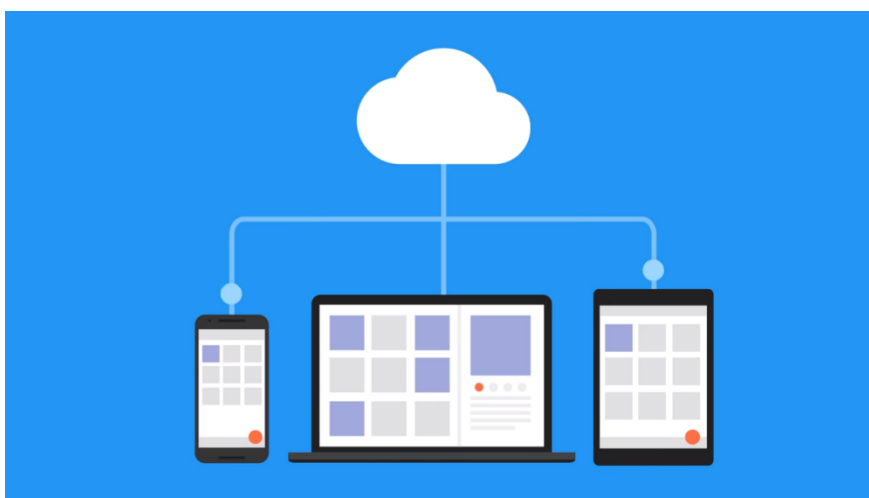


Figura 26: Imagem ilustrativa Firebase com vários dispositivos

Na figura 3 já é a junção dos exemplos acima, mostrando que o sistema pode ser complexo e que a ferramenta tem total suporte. Podemos concluir que o Firebase tem como foco colher informações e compartilhar com todos dispositivos envolvidos.

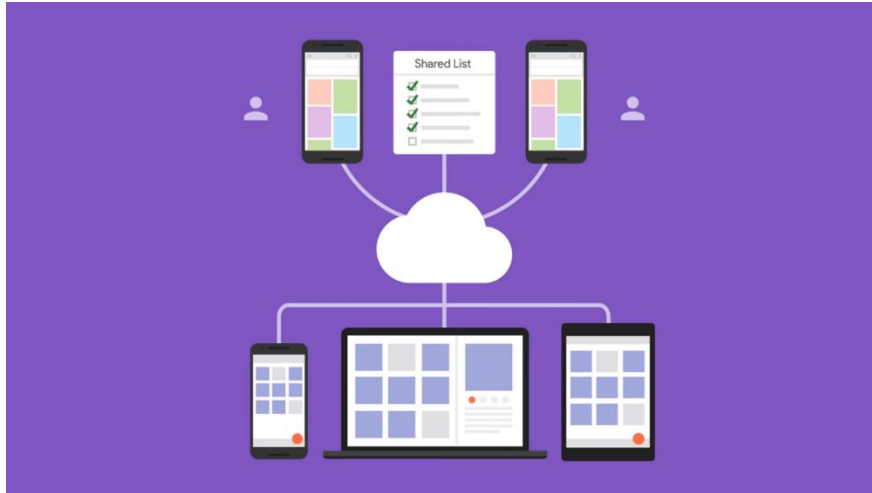


Figura 27: Imagem ilustrativa Firebase com uma lista compartilhada e vários dispositivos

4.5. VISUAL STUDIO CODE

Para codificar o projeto é essencial usar um editor de texto para auxiliar no desenvolvimento do mesmo, e nesse trabalho será utilizado o Visual Studio Code, que conforme Dionisio (2016) é um editor leve, gratuito e de multi-plataforma que foi desenvolvido pela empresa Microsoft. Sua ideia inicial era para desenvolvimento web, porém como se tornou uma ferramenta de código aberto, a comunidade acabou estudando e criando novas funcionalidades para o editor.

Para esse projeto em Flutter o Visual Studio Code auxiliou muito, pois nele é possível baixar plug-ins que facilita o desenvolvimento de código, como por exemplo a funcionalidade de comunicar com o celular e rodar a aplicação em tempo real e auto completar do código.

5. DESENVOLVIMENTO DO PROJETO

Pensando na facilitação do desenvolvimento do aplicativo, nessa sessão será exibida ferramentas que também fazem parte da disciplina de engenharia de software que auxiliou no entendimento geral do aplicativo. Para todos exemplos abaixo, foi utilizado o Astah Community que foi citado anteriormente.

5.1. DIAGRAMA DE CASO DE USO

O diagrama de casos de uso normalmente é concluído no início do projeto, na parte de análise e os levantamentos de requisitos da aplicação. Isso não quer dizer que será somente usado no início, mas sim no desenvolvimento inteiro do aplicativo. A finalidade desse diagrama é mostrar de uma forma simples para o desenvolvedor, usuário e até mesmo leitor desse trabalho o que o aplicativo irá fazer, mesmo a pessoa não sabendo nada da área de desenvolvimento, porque tem uma leitura de fácil compreensão.

A Figura 9 apresenta o Diagrama de Caso de Uso geral.

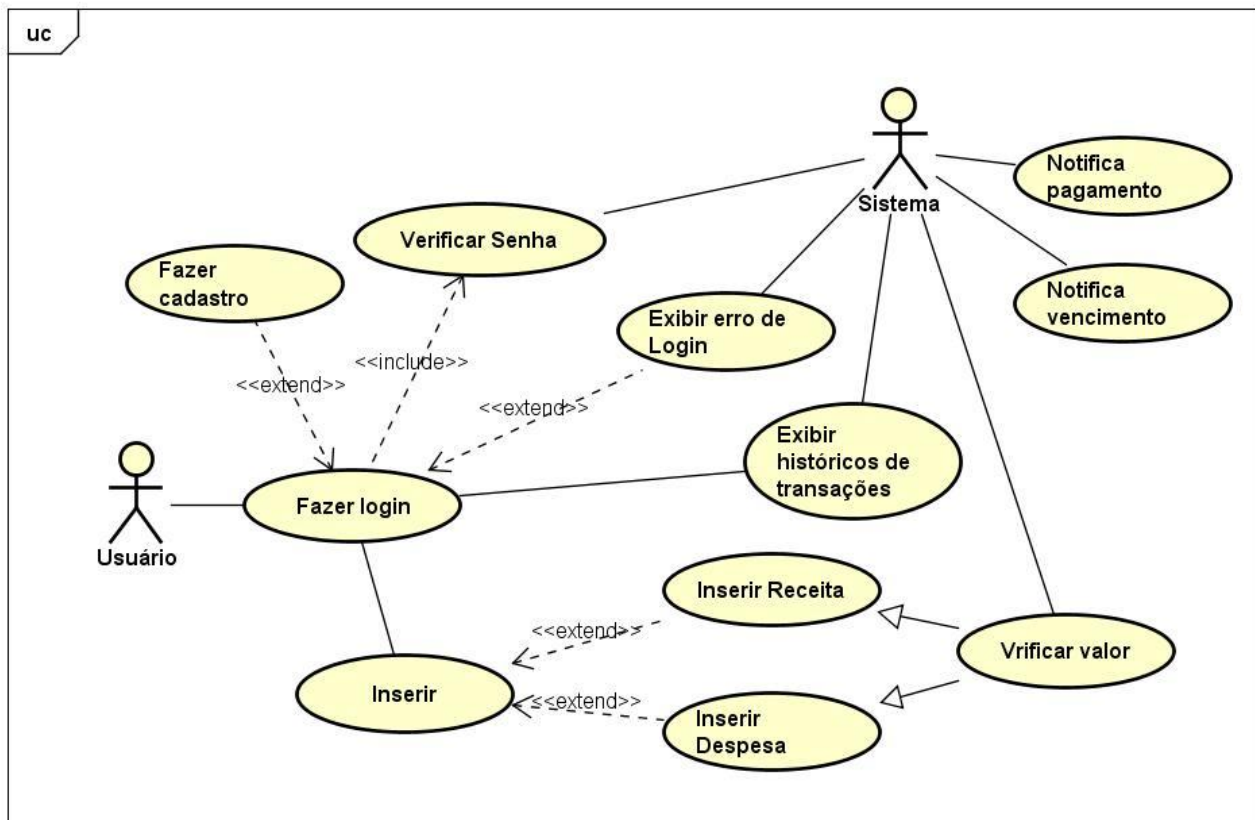


Figura 28: Diagrama de Caso de Uso – Geral

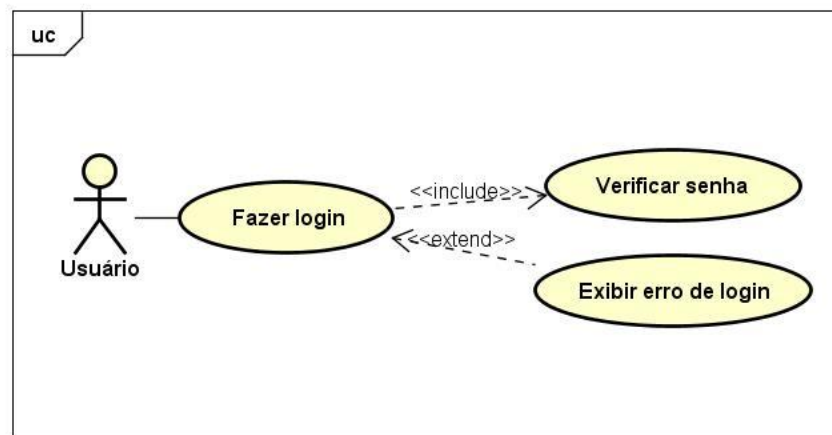


Figura 29: Diagrama de Caso de Uso – Fazer login

1. **Finalidade / Objetivo:** O usuário realiza o acesso com o aplicativo.
2. **Ator:** Usuário.
3. **Evento inicial:** O ator insere as suas informações de acesso para entrar no aplicativo.
4. **Fluxo principal:**
 - a. O sistema oferece interface gráfica para o usuário inserir suas informações de acesso;
 - b. O usuário informa seus dados para acessar;
 - c. O usuário confirma suas informações e seleciona a opção Logar (A1).

5. **Fluxo Alternativo:**

A1 – O usuário não tem cadastro:

- a. O sistema exibe que não existe cadastro para as informações inseridas;
- b. O usuário clica em cadastrar;
- c. O sistema exibe uma interface gráfica para o usuário inserir suas informações;
- d. O usuário insere suas informações de usuário e senha;
- e. O usuário clica em cadastra;
- f. O sistema salva os dados no banco;
- g. O sistema exibe a interface gráfica de login.

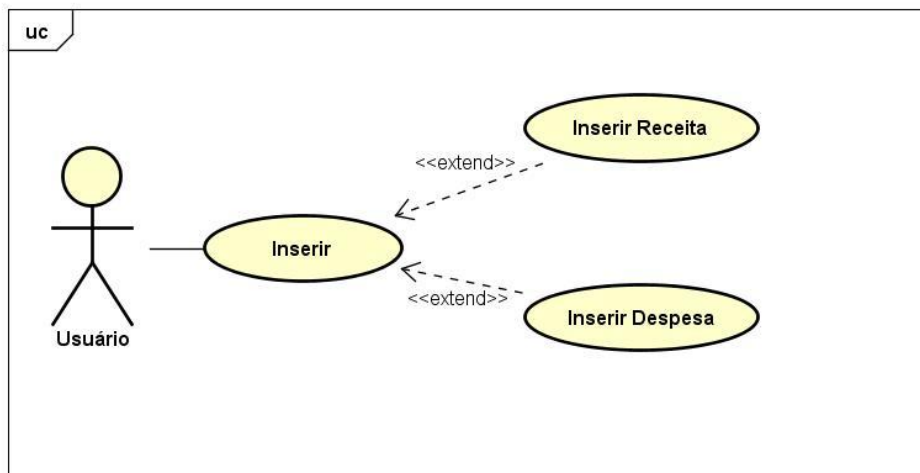


Figura 30: Diagrama de Caso de Uso – Inserir receita ou despesa

1. **Finalidade / Objetivo:** O usuário insere despesa ou receita.
2. **Ator:** Usuário.
3. **Evento inicial:** O usuário começa o caso de uso clicando em inserir *despesa/receita*.
4. **Fluxo principal:**
 - a. O sistema oferece uma interface gráfica para iniciar a inserção (A1);
 - b. O usuário seleciona se é *despesa* ou *receita*;
 - c. O usuário insere o valor;
 - d. O usuário seleciona a *categoria* da despesa ou receita;
 - e. O usuário insere uma *descrição* da despesa ou receita;
 - f. O usuário tem a opção de ativar *receita* ou *despesa mensal*;
 - g. O usuário insere a data de pagamento ou cobrança;
 - h. O usuário clica no botão de *salvar*;
 - i. O sistema valida o valor (A2);
 - j. O sistema salva no banco as informações;
 - k. O sistema exibe a interface gráfica principal do aplicativo.
5. **Fluxo alternativo:**

A1 – O usuário cancela a operação:

 - a. O usuário cancela a inserção;
 - b. O sistema exibe a tela de início.

A2 – O usuário digita informações inválidas:

- a. O sistema informa para o usuário por meio de uma interface gráfica as informações inválidas.

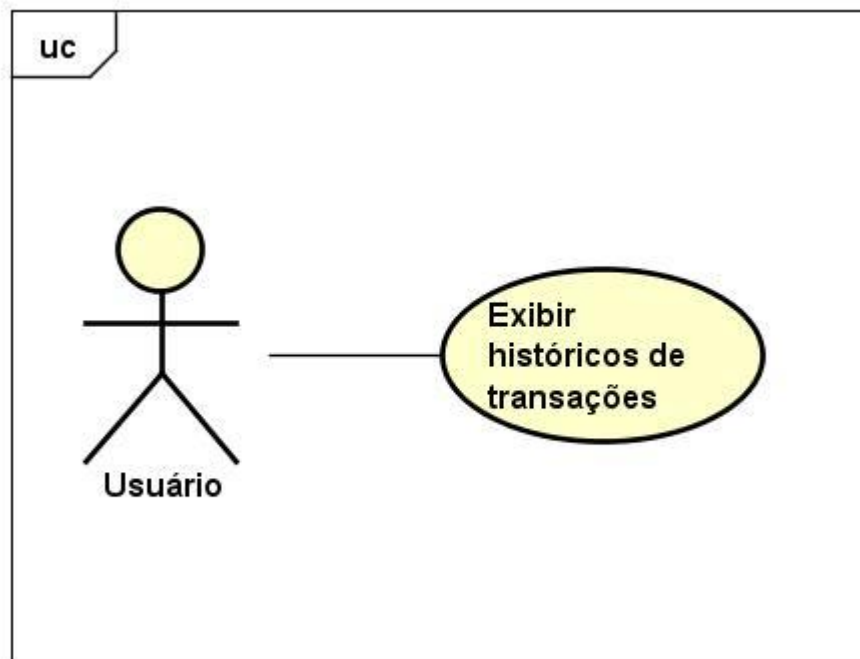


Figura 31: Diagrama de Caso de Uso – Exibir transações

1. **Finalidade / Objetivo:** O usuário consegue ver todas as transações feitas.
2. **Ator:** Usuário.
3. **Evento inicial:** O ator clica no saldo atual que está na tela inicial do aplicativo.
4. **Fluxo principal:**
 - a. O sistema oferece uma interface gráfica de todas as transações do usuário;
 - b. O usuário informa o período que deseja para analisar;
 - c. O sistema busca no banco as informações;
 - d. O sistema exibe todas as transações desejadas;

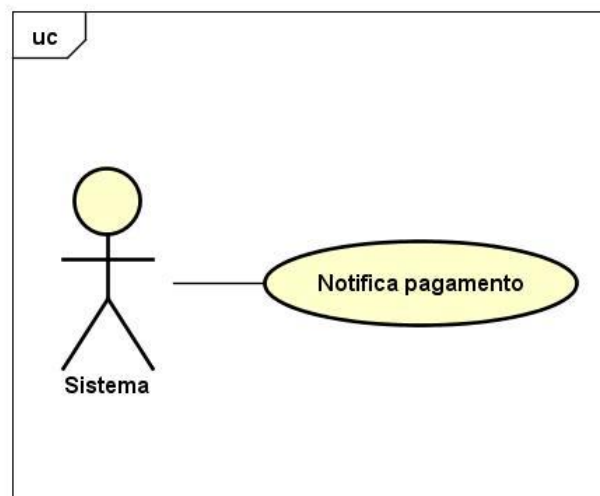


Figura 32: Diagrama de Caso de Uso – Notifica pagamento

1. **Finalidade / Objetivo:** Exibir notificação de pagamento a receber.
2. **Ator:** Sistema.
3. **Evento inicial:** O sistema faz uma varredura de contas a receber do dia atual;
4. **Fluxo principal:**
 - a. O sistema colhe informações das contas a serem recebidas;
 - b. O sistema exibe na barra de notificação do celular do usuário que o mesmo tem um pagamento a receber.

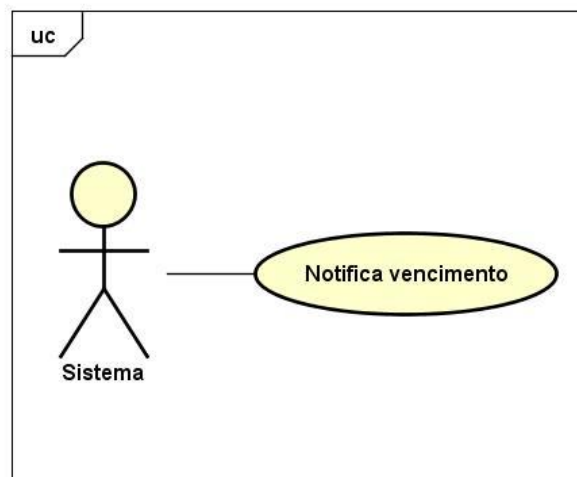


Figura 33: Diagrama de Caso de Uso – Notifica vencimento

1. **Finalidade / Objetivo:** Exibir notificação de contas a pagar.
2. **Ator:** Sistema.
3. **Evento inicial:** O sistema faz uma varredura de contas a pagar do dia atual;
4. **Fluxo principal:**
 - a. O sistema colhe informações das contas a serem pagas;
 - b. O sistema exibe na barra de notificação do celular do usuário que o mesmo tem uma conta a receber.

5.2. DIAGRAMA DE CLASSES

O diagrama de classe é um diagrama que tem como público alvo pessoas que estão desenvolvendo o projeto, mais específico para o desenvolvedor, pois de acordo com Tybel (2016), o Diagrama de Classe é exibição de uma estrutura do banco de dados, ou seja, estrutura das classes interligadas que são como se fossem modelos para a aplicação. Dito isso, facilita para o programador a estruturação do banco para o próximo passo, que seria o desenvolvimento do aplicativo. Na imagem a seguir, será exibido o diagrama de classe que contém os principais atributos e métodos que será necessário em cada objeto para a aplicação móvel.

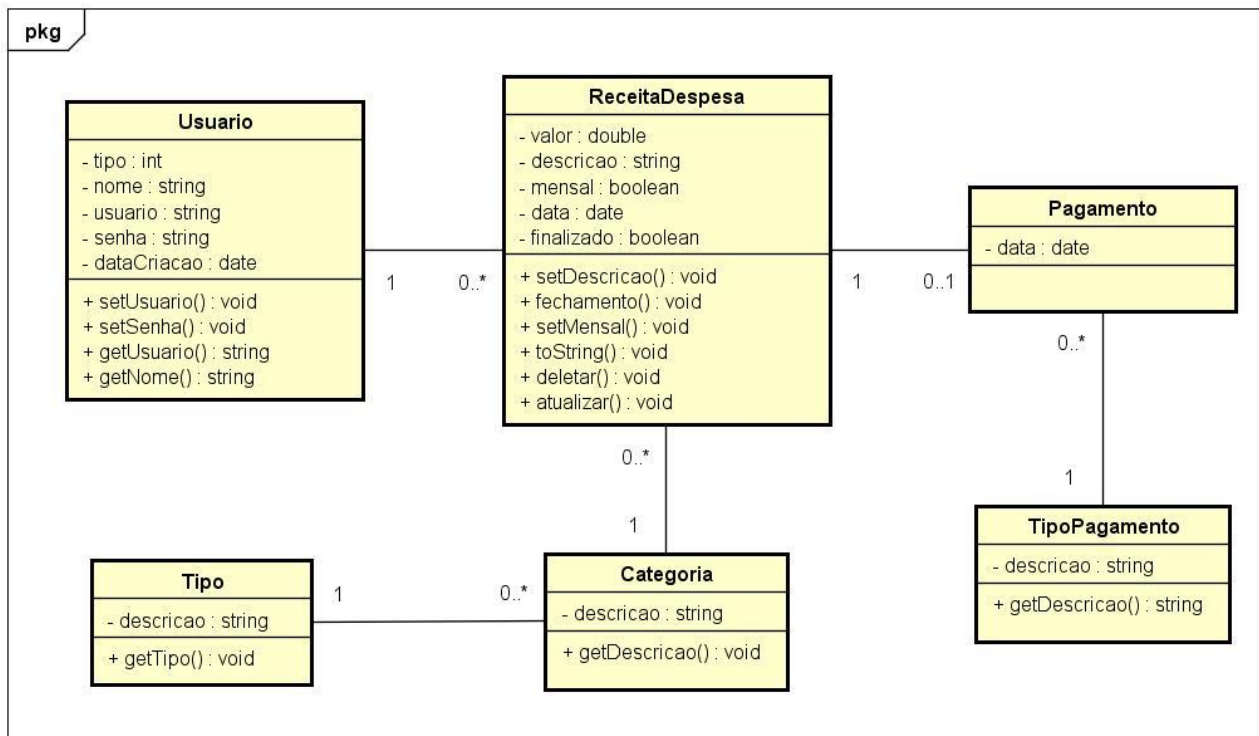


Figura 34: Diagrama de Classes da aplicação

A leitura da figura exibida acima é da seguinte forma:

- Usuário pode ter zero ou muitas receitas ou despesas;
- Receita ou despesa obrigatoriamente precisa ter um usuário;
- Receita ou despesa pode ter zero ou um pagamento;
- Receita ou despesa deve ter exatamente 1 categoria;
- Pagamento deve ter pelo menos um tipo de pagamento;
- Pagamento tem que ter pelo menos um tipo de receita ou despesa;
- Tipo de Pagamento pode ter zero ou vários pagamentos;
- Categoria pode ter zero ou muitas receitas ou despesas; e,
- Tipo pode existir zero ou várias categorias.

5.3. DIAGRAMA DE ATIVIDADE

Para se comunicar dentro de uma organização sobre um projeto e mostrar de uma forma mais clara de como ele funciona, usam-se diagrama de atividades que segundo Lucidchart (2018?) possui como finalidade principal unir desenvolvedores e pessoas da área de negócio a entender um determinado processo, ou seja, uma atividade.

Nesse diagrama mostra-se as lógicas do algoritmo, etapas das atividades, processo de negócio entre usuários e sistema e também o esclarecimento do projeto. Na figura 16 é exibido o diagrama de atividade do projeto, nele consegue-se entender como funciona uma inclusão de despesa ou receita no aplicativo.

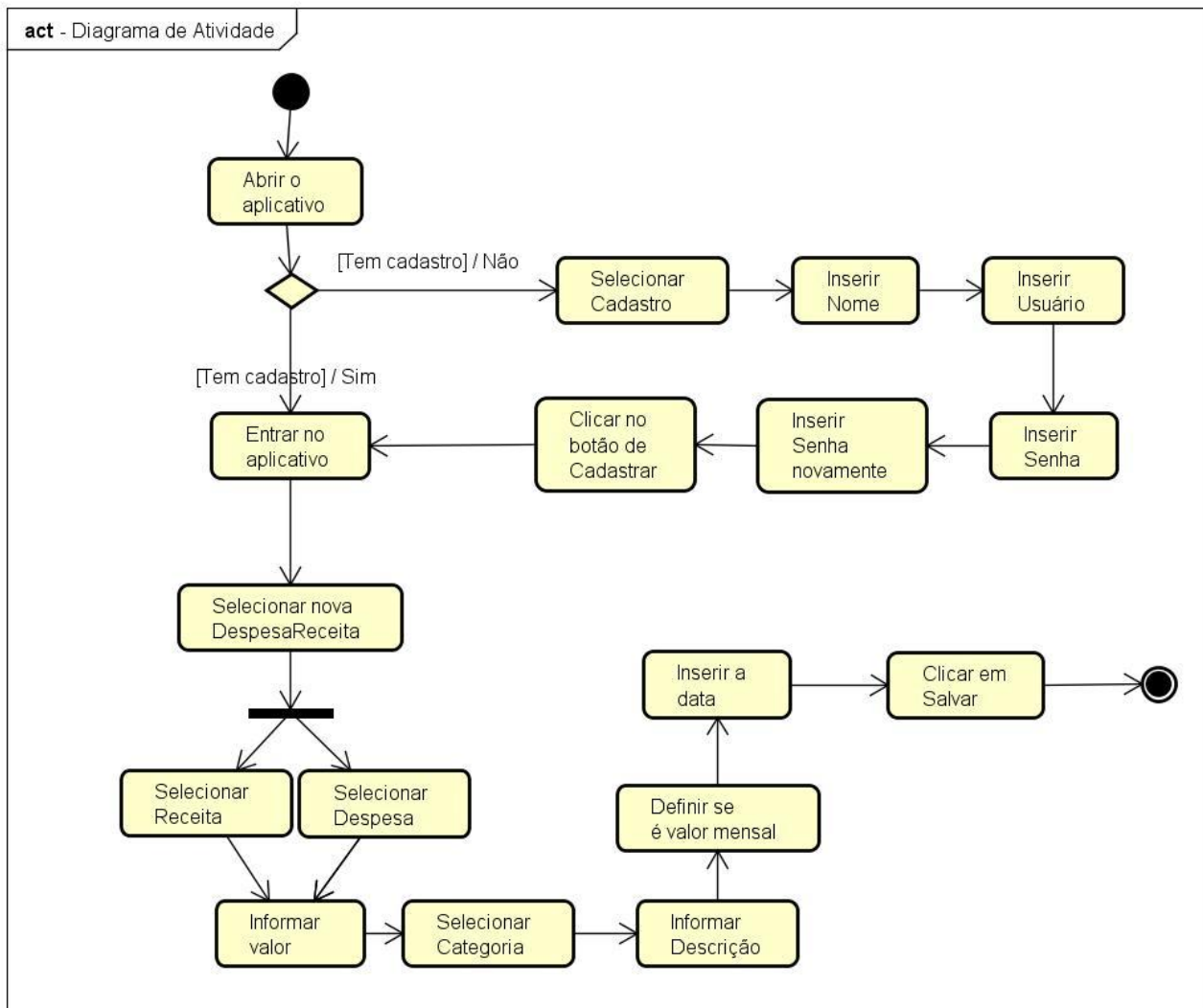


Figura 35: Diagrama de Atividade para inserir despesa ou receita

7. CONSIDERAÇÕES FINAIS

Houve diversas adversidades no processo de finalização desse trabalho, pois trata-se de uma tecnologia nova no mercado e consequentemente não obtendo muitos exemplos em livros e artigos que pudessem agregar na construção do mesmo, entretanto, a maioria dos objetivos foram alcançados graças a pesquisas em inglês, proporcionou-se o sucesso desse projeto e com uma excelente experiência em aprendizado. A maior dificuldade mesmo que tive foi achar *Widgets* personalizados, como um botão que troca de posição dependendo da opção selecionada com a finalidade de deixar a aplicação mais atraente ao usuário. Obtive outras dificuldades também como conectar meu aplicativo com o banco do Firebase, pois alguns tutoriais ou sites não estavam dando certo para minha versão, somente uma vídeo aula que me ajudou a ter sucesso na conexão.

Vale enfatizar a facilidade e rapidez do desenvolvimento do aplicativo quando se estuda mais a fundo a ferramenta e suas funcionalidades.

Os projeto inteiro ficará disponível no GitHub, com isso, auxiliará novos desenvolvedores, pessoas interessadas nessa tecnologia maravilhosa e principalmente nos meus trabalhos futuros como desenvolvedor. E com base na finalização desse trabalho, chegou-se à conclusão que a aplicação mencionada terá um valor enorme para comunidade.

8. REFERÊNCIAS

DIONISIO, Edson. **Introdução ao Visual Studio Code**. Disponível em <<https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418>>. Acesso em: 02 mar. 2020.

GUEDES, Marylene. **O que é Dart?**. Disponível em <<https://www.treinaweb.com.br/blog/o-que-e-dart/>>. Acesso em: 11 fev. 2020.

LIMA, Davi. **Modelos softwares com Astah Community**. Disponível em <<https://www.techtudo.com.br/tudo-sobre/astah-commmunity.html>>. Acesso em: 19 nov. 2019.

LUCIDCHART. **O que é um diagrama de atividades**. Disponível em <<https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-atividades-uml><https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418>>. Acesso em: 02 mar. 2020.

MAES, Jefferson. **Firebase o que é e para que serve?**. Disponível em <<http://digitalprimews.com/google-firebase/>>. Acesso em: 25 fev. 2020.

MAGALHÃES, Túlio. **Flutter: tudo sobre o queridinho do google**. Disponível em <<https://www.zup.com.br/blog/flutter>>. Acesso em: 21 out. 2019.

PAULA, Welington. **Rumo do Desenvolvimento Mobile**. Disponível em <<https://www.devmedia.com.br/rumo-do-desenvolvimento-mobile/24129>>. Acesso em: 21 nov. 2019.

PORTAGSI. **O que é UML?**. Disponível em <<https://www.portalgsti.com.br/uml/sobre/>>. Acesso em: 21 nov. 2019.

TYBEL, Douglas. **Orientações básicas na elaboração de um diagrama de classes**. Disponível em <<https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224><https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418>>. Acesso em: 02 mar. 2020.

VIANA, Daniel. **Firebase: descubra no que esta plataforma pode te ajudar**. Disponível em <<https://www.treinaweb.com.br/blog/firebase-descubra-no-que-esta-plataforma-pode-te-ajudar>>. Acesso em: 03 out. 2019.