RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

FLUTTER UM *FRAMEWORK* PARA DESENVOLVIMENTO *MOBILE*

FLUTTER A FRAMEWORK FOR MOBILE DEVELOPMENT

FLUTTER UN MARCO PARA EL DESARROLLO MÓVIL

Matheus Rissi[1], Felipe Diniz Dallilo[2]

e3112230
https://doi.org/10.47820/recima21.v3i11.2230

PUBLICADO: 11/2022

RESUMO

Atualmente sabemos que a tecnologia da informação (T.I) vem seguindo uma linha de crescimento constante onde perenemente precisa-se da criação de novas tecnologias ou novas atualizações das que já existem. Não é diferente no que se diz respeito ao mercado de desenvolvimento mobile: pode-se notar uma onda de usuários e até mesmo empresas optando pelo mercado mobile devido a sua enorme volatilidade em entregar um produto. Visto a necessidade de desenvolver esses *apps*, esse artigo tem por objetivo apresentar um *framework* capaz de agilizar odesenvolvimento para todos os ambientes, seja *mobile* (Android e iOS), *Web* ou até mesmo *desktop* (Windows, macOS e Linux), o Flutter. Dado essa demanda de multiplataformas, o Flutter vem com a proposta de resolver o problema do retrabalho no desenvolvimento, por diversas vezes é necessário realizar a criação de bases de códigos diferentes para a mesma funcionalidade devido a sua usabilidade (Mobile, Desktop, Web).

PALAVRAS-CHAVE: Flutter. Dart. Desenvolvimento de aplicativos.

ABSTRACT

We currently know that information technology (IT) has been following a line of constant growth where we continually witness the creation of new technologies or new updates to the existing ones. It is no different when it comes to the mobile development market: we can see a wave of users and even companies opting for the mobile market due to its enormous (volatility) in deliveringa product. Given the need to develop these apps, this article aims to present a framework capable of speeding up development for all environments, whether it is mobile (Android and iOS), Web or even desktop (Windows, macOS and Linux), the Flutter. Given this demand for multiplatforms, Flutter comes with the proposal to solve the problem of rework in development, since several timesit is necessary to create different code bases for the same functionality due to its usability (Mobile, Desktop, Web).

KEYWORDS: *Flutter. Dart. Mobile app development.*

RESUMEN

Actualmente sabemos que la tecnología de la información (TI) ha estado siguiendo una línea de crecimiento constante donde se necesita perennemente crear nuevas tecnologías o nuevas actualizaciones de las que ya existen. No es diferente cuando se trata del mercado de desarrollo móvil: uno puede notar un lugar donde los usuarios e incluso las empresas optan por el mercado móvil debido a su enorme volatilidad en la entrega de un producto. Dada la necesidad de desarrollar estas apps, este artículo pretende presentar un framework capaz de agilizar el desarrollo para todos los entornos, ya sean móviles (Android e iOS), Web o incluso de escritorio (Windows, macOS y Linux), Flutter. Ante esta demanda de multiplataforma, Flutter viene con la propuesta de solucionar el problema del retrabajo en desarrollo, varias veces es necesario llevar a cabo la creación de diferentes bases de código para una misma funcionalidad debido a su usabilidad (Mobile, Desktop, Web).

PALABRAS CLAVE: *Aleteo. Dardo. Desarrollo de aplicaciones.*

[1] Universidade de Araraquara - UNIARA
[2] Universidade de Araraquara - UNIARA

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

INTRODUÇÃO

Em janeiro de 2021 o relatório da ONU apontou que cerca de 5,22 bilhões de pessoas utilizam celulares, isso equivale a 66,6% da população. Paralelo a isso, houve uma pesquisa (GEEKHUNTER, 2021) que aponta que só no primeiro semestre de 2021 “houve um aumento de 600% nas buscas por desenvolvedores *mobile*, em relação ao mesmo período do ano de 2020”. Consequentemente, a procura por aplicativos de entretenimento e facilidade para o dia a dia teve uma grande procura. Junto com esse aumento de profissionais da área de desenvolvimento, procuram *frameworks* que facilitem o desenvolvimento da alta demanda de *softwares*, esses mesmos muitas vezes terão de ser criados em multiplataformas (Windows, Linux, Android, iOS).

Neste trabalho será examinado o Flutter e suas vantagens em um cenário de desenvolvimento de aplicações. O Flutter é um *framework* de código aberto do Google que se valeda linguagem de programação Dart, sendo usado para criar aplicativos moveis, *web*, *desktop* e incorporados a partir de uma única base de código (FLUTTER, 2022).

Existem vantagens e desvantagens na utilização do *framework*. Segundo Zammetti (2020)“O Flutter tem muito a oferecer, mas não é a panaceia”.

O Flutter é utilizado por desenvolvedores para facilitar o processo de criação em multiplataformas em uma única base de código.

Segundo Adjuste, empresa de análise que auxilia na tomada de decisões de profissionais de marketing, em todas as categorias de aplicativos há um aumento de 50% nas instalações em comparação com o ano de 2019 e 2020. É particularmente notável um crescimento de 74% a 76% quando comparado a 2019. A utilização de aplicativos é algo cada vez mais comum no cotidiano como é apontado no *e-book Mobile App Trends* 2021.

Os desenvolvedores tendem a buscar novas ferramentas para auxiliar na criação desses *apps*. Galante (2019), escritor do site USE, nos mostra que os melhores *frameworks* para desenvolvimento de aplicativos são: Ionic, React Native, Xamarin, Adobe, PhoneGap, Flutter, Corona SDK, JQuery Mobile, Native Scripts e Mobile Angular UI.

Zammetti (2020) “criar aplicativos moveis com aparência e funcionamento de aplicativos nativos e que também sejam multiplataforma é uma proposta complexa, até mesmo após anos como os desenvolvedores tentando atingir esse objetivo”.

Cada plataforma necessita da criação de uma base de código nativo, gerando retrabalho por parte do desenvolvedor, bem como uma insatisfação por parte do cliente final. Conforme Zammetti, “o Flutter é uma plataforma que fornece um meio de escrevermos uma única (mais ou menos) base de código que funcione igualmente bem no Android e no iOS e distribua ao mesmo tempo desempenho e recursos nativos”.

Será realizada uma revisão bibliográfica da criação do Flutter e das vantagens da utilização do *Framework* no desenvolvimento de aplicações multiplataformas, bem como a criação de um aplicativo.

O Aplicativo criado terá a função de demostrar a facilidade da utilização do Flutter no desenvolvimento de um aplicativo de *login* onde será possível realizar o cadastro de um novo usuário que será armazenado em um banco de dados do Firebase.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

1 REVISÃO BIBLIOGRÁFICA TÍTULO

Esta seção apresentará alguns conceitos importantes na utilização do Flutter, suas vantagens e desvantagens durante o desenvolvimento de projetos, com como suas camadas arquiteturais.

FLUTTER

Para melhor entendimento do Flutter, é relevante compreendermos seus quatro pilares: Dart, engine principal, interface e seus *widgets* (ZAMMETTI, 2020).

O Dart é uma linguagem utilizada no desenvolvimento de aplicativos *web*, código de servidor, aplicativos LoT (Internet of Things, Internet das coisas) e assim por diante. Ultimamente vem se tornando famoso devido ao uso no Flutter.

Diversos aspectos importantes podem ser considerados a respeito do Dart, como: ser totalmente orientado a objeto e possuir linguagem com coleta de lixo *(garbage-colleted)*. Com relação ao seu estilo possui linguagem baseada em C, que por sua vez é semelhante para diversos programadores. O sistema de tipagem é solido, sendo que, segundo Zammetti (2020), "[...]com flexibilidade na tipagem com o mesmo tipo, o que não o torna um estorvo, e sim uma verdadeira ajuda para os desenvolvedores".

Outro ponto favorável é o modo em que o Dart é executado, pois este antecipa acompilação para códigos nativos *(ahead-of-time compilation),* o que torna a linguagem muito bem performada, ou seja, proporcional ao Assembly. Além disso o Dart se adapta para os modos de compilação ARM, mais simples por ser baseado na RISC *(Reduced Instruction Set Computer),* e x86, que possui uma arquitetura mais complexa, ambas podem ser transpiladas para JavaScript (ZAMMETTI, 2020, p26).

O suporte Dart contém um grande acervo de pacotes com inúmeras funcionalidades além da linguagem básica que acabam por auxiliar o desenvolvedor. Também possui algumas IDEs popularmente utilizadas pelos programadores como Visual Studio Code e o Intelij IDEA, o que incentiva seu uso.

No núcleo do Flutter está sua Engine, uma codebase que é escrita principalmente em C++e suporta os requisitos primordiais das suas aplicações. O mesmo se responsabiliza por buscar um layout sempre que um novo frame for criado. Ele fornece a implementação de baixo nivel de API principal do Flutter, incluindo gráficos através do Skia (biblioteca gráfica *open source,* com um excelente desempenho em diversas plataformas), layout de texto I/O de arquivo e rede suporte a acessibilidade, arquitetura de plugin e o tempo de execução do Dart e compilar o Código (FLUTTER, 2022).

ARQUITETURA DO FLUTTER

Projetado com um sistema extensível em camadas, o Flutter existe com uma série de bibliotecas independentes, que são atreladas a sua camada subjacente. Ressalva-se que cada camada de sua arquitetura é independente, ou seja, é projetada para ser opcional e substituível. (FLUTTER, 2022).

A arquitetura do Flutter é composta pelas seguintes camadas: *Framework, Engine e Embedder,* como apresentado a seguir.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

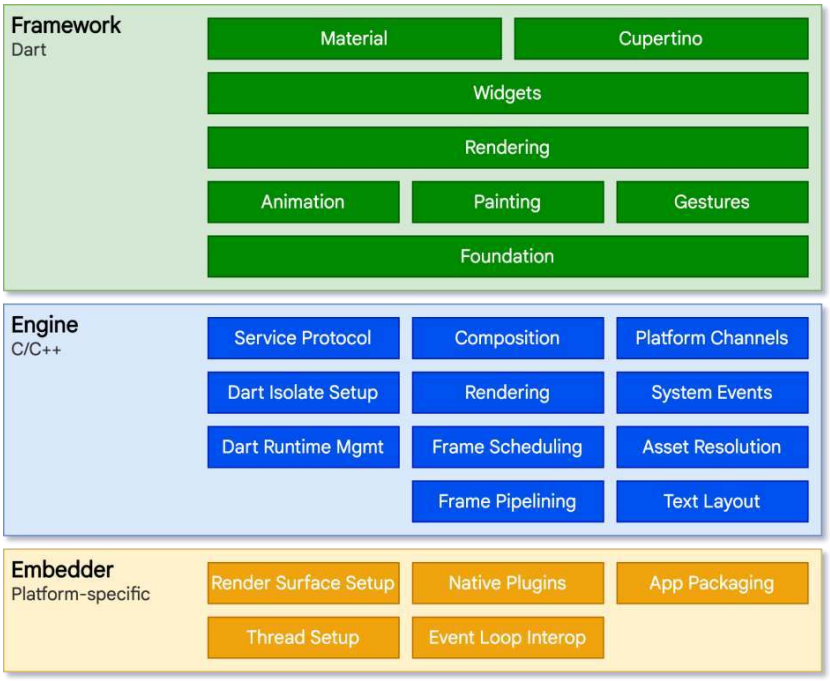


Figura 1 – Camadas arquitetônicas
Fonte: https://docs.flutter.dev/resources/architectural-overview

Referente a camada *Framework*, que possui por tradução direta estrutura, entende-se quesua função é unir um conjunto de técnicas e ferramentas usadas para resolver um problema, ou seja, um conjunto de códigos que se unificam para resolver uma situação. GAMMA (1995) define *framework:*

> Um conjunto de classes que cooperam entre si e compõem um projeto reutilizável para uma categoria específica de *software*. Um *framework* fornece direcionamento arquitetural do software, através do particionamento do projeto em classes abstratas e da definição de suas responsabilidades e colaborações. Um desenvolvedor customiza o *framework*, para uma aplicação particular, através da especialização e da composição de instâncias de classes do mesmo.

O *Framework* tem por objetivo facilitar o ambiente de desenvolvimento, fazendo assim com que os programadores foquem seus esforços na resolução do projeto, em vez de lidar com bibliotecas comuns e repetitivas.

A camada subsequente é dominada *Engine*, que tem a função de abstrair processos que teriam que ser feitos manualmente caso não houvesse a existência da mesma, realizando assim o agrupamento de todos os arquivos e bibliotecas que o Flutter irá precisar. Flutter (2022) traz a definição de *Engine* como um “aplicativo” portátil que roda em tempo de execução utilizado para hospedar aplicativos Flutter. É responsável por implementar as bibliotecas indispensáveis do Flutter, além de incluir animações, gráficos, entrada e saída de arquivos de rede, suporte a acessibilidade, *plugins* e diversas ferramentas de runtime para execução do *Dart*.

O Embedder, dentro da ciência da computação, é um código utilizado para a inserção de multimídia. No *Framework* Flutter, a camada de mais baixo nível, o *Embedder*, tem a função de interagir

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

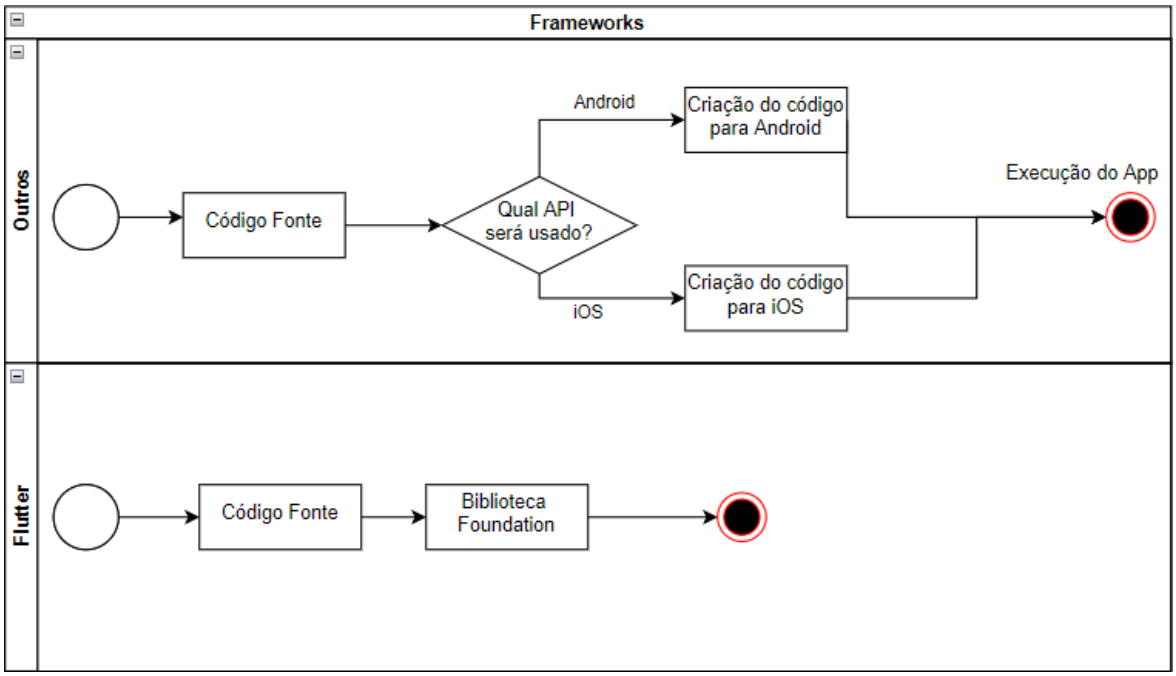
com os sistemas operacionais (S.Os), como por exemplo: Android, iOS, Windows , Linux e macOS. Essa camada é escrita na linguagem específica do sistema se apropriando dependendoda plataforma: Java e C++ para Android, Objective-C/Objective-C++ para iOS e macOS e C++ para Windows e Linux. Além disso, ao utilizar incorporador, é possível integrar o código Flutter como um modulo em um aplicativo já existente.

INTERFACE

Pela definição de *Oxforf Languages interface* é “um elemento que proporciona uma ligação física ou lógica entre dois sistemas ou parte de um sistema que não poderiam ser conectadas diferentemente”, ou seja, por onde o usuário interage com aplicativo, página *Web* e até mesmo o sistema operacional.

A interface do Flutter nos fornece a biblioteca *foundation*, essa tem o objetivo de remover as diferenças entre as APIs das plataformas nativas. Em outras palavras não precisa se perguntarque API deve usar em uma plataforma ou na outra, só precisa saber que a chamada da API do Flutter deve ser executada para iniciar o aplicativo, ou seja, funciona em ambas as plataformas (ZAMMETTI, 2020).

Figura 2 – Diferença entre *frameworks*



Fonte: Elaborada pelo autor (2022)

A utilização da figura acima retrata o fluxo de desenvolvimento de uma aplicação entre um *framework* “qualquer” e o Flutter. Nela é possível observar a utilização da Biblioteca *Foundation* (existente na *interface* do Flutter) que exclui a etapa de decisão e criação das API nativas.

O último pilar é o *Widget*, um componente visual para definir a *interface* de um aplicativo. Analogicamente falando, cada *widget* é como uma pequena peça de um quebra cabeça, que ao final do conjunto representará uma *interface* completa.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Pinheiro definiu que para esse comportamento, o Flutter divide seus *widgets* em duas categorias: *layout* e *interface*. O *layout*, organiza o posicionamento dos widgets, delimitando o espaço de cada *widget*. A *interface* cria componentes visuais que são exibidos ao usuário, eles são usados na escolha do componente vinculado a *interface* principal do *app*.

Segundo Zammetti, os *widgets* podem ser óbvios, como por exemplos: botões, listas, imagens, campos de textos de formulários, etc. Ressaltando ainda que diversos itens que normalmente em linguagens de desenvolvimento não são considerados *widgets*, no Flutter são, um exemplo disso seria o preenchimento *(padding)* ao redor de uma imagem, o texto exibido na tela e até mesmo seus estados do campo de texto de um formulário.

FERRAMENTAS UTILIZADAS PARA DESENVOLVIMENTO DO APLICATIVO

Essa seção é responsável pela apresentação das ferramentas que serão utilizadas para o desenvolvimento do aplicativo.

VISUAL STUDIO CODE

O editor de código utilizado no projeto será o Visual Studio Code por ser um recurso com diversas ferramentas e extensões para as linguagens de C++, C#, Java, Python, PHP, Go e incluso a extensão do Dart e Flutter, essenciais para criação do aplicativo. Outra característica para a escolha do editor é sua eficiência que entre as opções (Android Studio e inteliJ IDA) para compilação se torna mais viável.

ANDROID STUDIO

O Android Studio também é um editor de código, em específico para o sistema operacional Android, como nome já sugere, possibilita a criação de aplicações completas e disponibiliza um acervo enorme de customizações. Apesar de fornecer esses recursos o mesmo acaba por ter uma performance inferior as outras IDEs.

Contudo, necessitamos de sua utilização, pois é necessário fazer o uso do Android SDK para o desenvolvimento do aplicativo, além de fornecer um emulador Android para testar a execução do código.

FIREBASE

Segundo Orlandi, escritor da *rocketseat*, o Firebase é um Baas *(Backend as a Service)* para aplicações *Web* e *Mobile* do Google, que atualmente para alguns projetos é uma das melhores opções devido a imensa quantidade de serviços e a facilidade de sua implementação. facilidade de implementação.

Um detalhe importante a ser considerado no desenvolvimento de um aplicativo é como serão armazenados os dados. Nesse trabalho será utilizado o Firebase devido a sua facilidade de integração com o Flutter, e pela imensa quantidade de serviços oferecidos por ele. O Android Studio também é um editor de código, em específico para o sistema operacional Android.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

2 DESENVOLVIMENTO

O desenvolvimento deste trabalho apresenta as ferramentas que serão utilizadas para a construção do aplicativo, bem como os estudos realizados para a sua execução, seguindo os conceitos da revisão bibliográfica.

CASO DE USO (USE CASE)

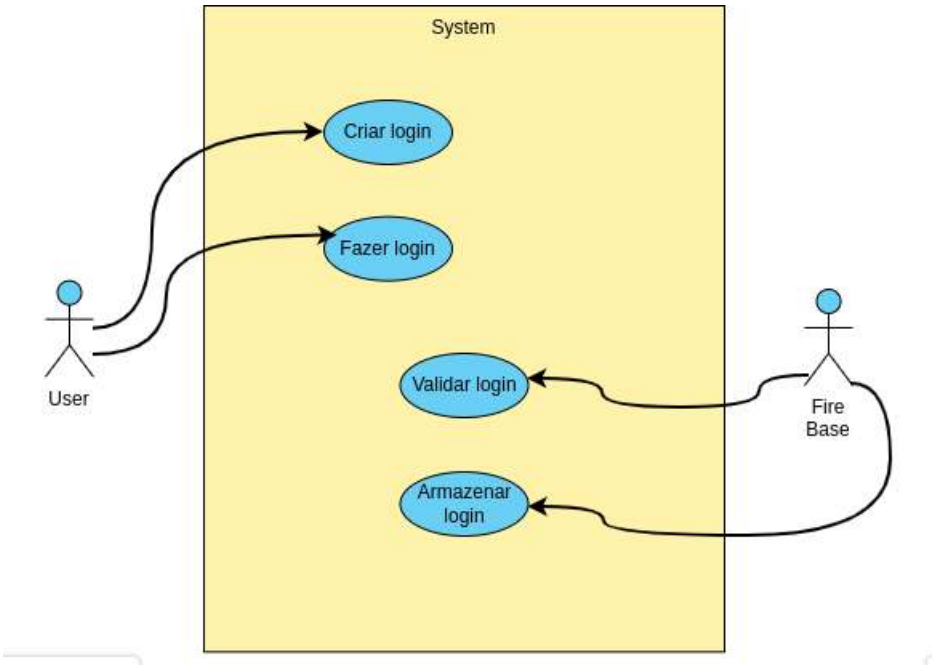
Para Melo (2010), o use case é responsável por descrever uma sequência lógica de ações representando um cenário principais (representatividade da vida real) e alternativos, que seria a representatividade do comportamento do *software* ou uma parte específica do mesmo.

Assim conseguindo auxiliar entre a comunicação entre analista e cliente, além de forneceruma visão gráfica simplificada dos processos do sistema.

Nesse diagrama poderão ser notados atores (representados por “bonecos” com seu nome abaixo), podendo ser humanos ou simplesmente outro sistema computacional; caso de uso que é representado por uma elipse definindo uma função do sistema; relacionamentos que demonstra a associação entre um ator e o caso de uso, podendo ser usado também entre os atores (representado com uma linha).

Portanto, para auxiliar na modelagem dos processos e relações que o projeto terá, foi desenvolvido um diagrama de Use Case (Figura 3), onde podemos ver a representação dos relacionamentos entre o administrador e o usuário final, além de demonstrar as ações que cada um poderá executar.

Figura 3 – Diagrama de uso, relação entre atores e casos de uso



Fonte: Elaborada pelo autor (2022)

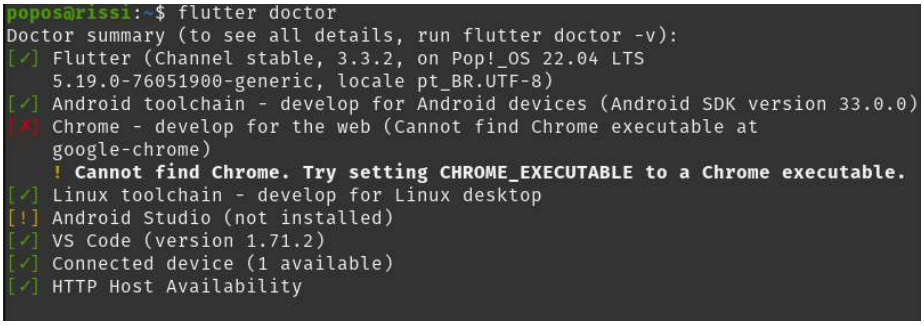
RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

INSTALAÇÃO E CONFIGURAÇÃO

A instalação do *flutter* é simples e pode ser executada de duas maneiras: a primeira é realizando o *Download* de um ZIP e descompactando e a segunda é clonando o projeto pelo *Git*. O próprio *site* do *framework* já nos fornecesse um passo a passo prático para instalação e execução do *flutter* nos ambientes do Windows, Linux, macOS e ChromeOS a partir do *link*: https://docs.flutter.dev/get-started/install.

Após a instalação é necessário rodar o comando *"flutter doctor*" no terminal para analisar se as dependências se encontram de acordo com o necessário para a execução do *flutter* (Figura 5).

Figura 4 – *flutter doctor*



Fonte: Elaborada pelo autor (2022)

DESENVOLVIMENTO DO APLICATIVO

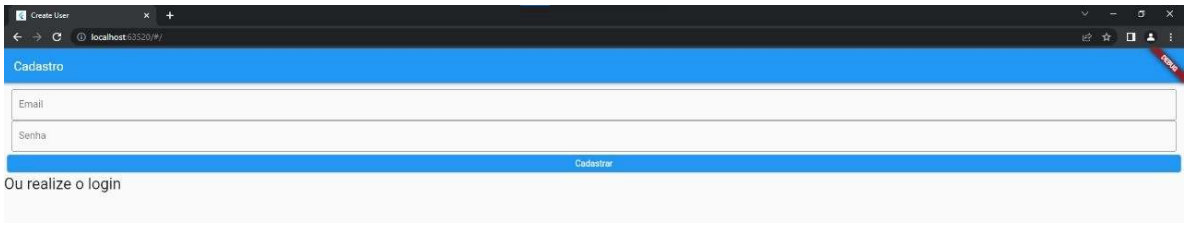
Após realizar os ajustes necessários de configuração, foi desenvolvido a *interface* do usuário onde o mesmo consegue realizar o *login* ou caso seja seu primeiro acesso, possa criar um login novo.

INTERFACE DO USUÁRIO

A *interface* do usuário ou *front-end* é considerada uma das programações de mais alto nível por se tratar de um contexto onde o usuário final realiza suas interações com a aplicação. Com isso, foi desenvolvido uma *interface* de fácil entendimento para que o usuário realize a criação de sua conta e efetue seu *login* (Figura 5), cujo código fonte será apresentado abaixo (Figura 6).

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Figura 5 – *Interface* do Usuário



Fonte: Elaborada pelo autor (2022) Figura 6 – Código fonte da *interface*

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Login'),
    ),
    body: Padding(
      padding: const EdgeInsets.only(top: 10),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 12),
            child: TextFormField(
              keyboardType: TextInputType.text,
              controller: mailController,
              decoration: const InputDecoration(
                  hintText: "Email", border: OutlineInputBorder()),
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 12),
            child: TextFormField(
              keyboardType: TextInputType.text,
              controller: passwordController,
              decoration: const InputDecoration(
                  hintText: "Senha", border: OutlineInputBorder()),
              obscureText: true,
              enableSuggestions: false,
              autocorrect: false,
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 5, vertical: 5),
            child: ElevatedButton(
              onPressed: () => authUser(),
              child: const Text("Login"),
            ),
          ),
        ],
      ),
    ),
  );
}
```

Fonte: Elaborada pelo autor(2022)

Na figura 6 é mostrado o conjunto de *widgets* que são necessários para construção da interface do usuário. Utilizando-se do conjunto de *widgets* que o *Flutter* nos fornece, para acriação da interface alguns dos *widgets* utilizados são: *Padding* (determina o distanciamento entre os elementos), *ObscureText* (não permite que a senha seja visualizada enquanto o usuário digita amesma), *HintText* (texto mostrado nos campos de digitação como autoexplicação no caso do e- mail e senha).

BACK-END

Como já citado, o *back-end* do projeto foi baseado na ferramenta do Google, Firebase, devido a sua volumosa quantidade de funções. Uma das ferramentas é o autentificador de *login*

(Autentification) utilizado no código, esse é responsável pela validação da existência de um usuário dentro do banco de dados, sendo assim, o autentificador tem a função de retornar se o acesso foi permitido ou negado dependendo do e-mail e senha digitados. O *Autentification* é configurado no ambiente do Firebase, para que seja possível o controle de cadastros realizados dentro do *app*, armazenando o cadastro do e-mail que será utilizado para validação (Figura 7). A partir daí foi gerada a função para que essa validação fosse possível

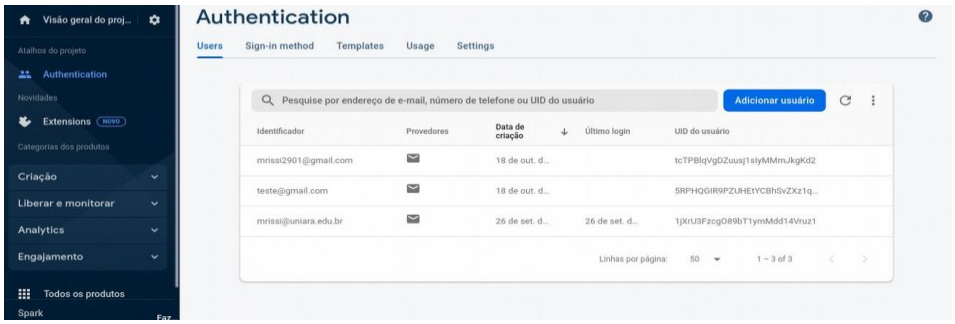


Figura 7 – *Autentification*
Fonte: Elaborada pelo autor

3 RESULTADOS E DISCUSSÕES

O resultado obtido com o desenvolvimento desse artigo foi a obtenção de conhecimento sobre o *framework* Flutter, com isso, foi gerado um senso crítico sobre seus benéficos e malefícios.

Com a fácil instalação e configuração do *framework*, foi possível realizar a construção do aplicativo mostrando a sua capacidade de atuar em multiplataformas sem dificuldade ou erro a partir de um único código fonte, juntamente a algumas funcionalidades que outros *frameworks* não possuem nativamente como o caso do Hot Reload que auxilia o desenvolvedor. Vale ressaltar queo flutter também traz uma variedade de *widgets* próprios e de terceiros que auxiliam na criação doprograma. Em contra partida, essa qualidade se torna um malefício, pois é necessário a utilização de diversos *widgets* para criação de “algo simples” e paralelo a isso o tamanho do *software* também é desvantagens uma vez que se têm de incluir o engine básico do Flutter, as bibliotecas de suporte e outros recursos.

A figura a seguir (figura 8) compara a diferença na criação de campos de textos que recebem valor *(input texts)* entre o *framework* Flutter e o React Native. Esse foi escolhido para nível de comparação devido a sua “competitividade” com o Flutter, como é demonstrado pela autora Duggal, escritora do *site SimpliLearning*, que lista os melhores frameworks de desenvolvimento mobile de 2022, sendo o *React Native* em primeiro lugar, seguido do Flutter. Além disso, ambas possuem a possibilidade de desenvolvimento em multiplataformas, são recentes no mercado e foram bem aceitos pela comunidade.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Figura 8 – Comparação entre códigos, Flutter e Reactive Native

Fonte: Elaborada pelo autor (2022)

A partir da comparação foi possível notar uma maior simplicidade na sintaxe do React Native que entrega um entendimento mais facilitado da leitura. Porém, é indicado a utilização do Flutter caso o projeto a ser trabalhado seja mais robusto e necessite de programação nativa.

CONSIDERAÇÕES FINAIS

O Flutter se mostrou um *framework* exímio ao que propõem, um desenvolvimento singular para multiplataformas que se adapta ao ambiente conforme o uso do usuário final e que possui uma fácil integração com diversas ferramentas, como é o caso do Firebase aqui utilizado.

Todavia, no período de desenvolvimento do *app*, foi possível notar um desconforto principalmentepor conta da sintaxe do código. Coisas simples que em alguns *frameworks* precisam de apenas duas ou três linhas, com o flutter é necessário montar uma árvore de *widgets* que são essenciais para que o resultado final seja o esperado.

Apesar disso, o benefício de não ter de ficar criando diversos códigos para a mesma aplicação em diversos ambientes se sobrepõem as dificuldades iniciais de sintaxe, sendo assimse valida o uso do *framework* como positivo para os aplicativos nele desenvolvido.

Como projeto futuro, pode-se dar continuidade na implementação do aplicativo, como: incrementação de novas funcionalidades, validação de 2 fatores utilizando o Firebase e um melhor desenvolvimento da *interface* gráfica para utilização do *app*.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

REFERÊNCIAS

ADJUST. **Mobile app Trends 2021**: Um studio global sobre a performance de aplicativos.(Online). [S. l.]: ADJUST, 2021. Disponivel em: https://www.adjust.com/pt/resources/ebooks/mobile-app-trends2021/?utm_source=PressRelease. Acesso em:10abr. 2022

AMMETTI, F. **Flutter na prática**. Tradução: Aldir Coelho Corrêa da Silva. São Paulo: Novatec, 2020

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML, Guia do Usuário**. 2 ed. Rio de Janeiro: Campus, 2005

CHISHOLM, Kevin J. **What's new in Flutter 3**. [S. l.]: Medium, 2022. Disponível em: https://medium.com/flutter/whats-new-in- flutter-3-8c74a5bc32d0. Acesso em: 03 jun. 2022.

DUGGAL, N. **Top 10 Mobile App Development Frameworks in 202**. [*S. l.: s. n.*], 2022. Disponível em: https://www.simplilearn.com/top-mobile-development-applications-article. Acesso em: 23 out. 2022.

FLUTTER. **Crie aplicativos para qualquer tela**. [S. l.]: Flutter, 2022. Disponivel em: https://flutter.dev. Acesso em: 10 abr. 2022.

FLUTTER. **Flutter architectural overview (Online).** [S. l.]: Flutter, 2022. Disponível em: https://docs.flutter.dev/resources/architectural-overview. Acesso em: 05jun. 2022

GABRIEL, N. M. B. Flutter – Vale a pena investir nesse framework?. **Blog iteris**, 2022. Disponível em: https://blog.iteris.com.br/flutter-vale-a-pena-investir-nesse-framework/. Acesso em 03 jun. 2022.

GALANTE, V. R. **Melhores Frameworks para o desenvolvimento de aplicativos**. [S. l.]: Usemobile, 2019. Disponível em: https://usemobile.com.br/framework-desenvolvimento-aplicativos-2019/. Acesso em: 10 abr. 2022.

GAMMA, E. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Tradução: Luiz A. Meireles Salgado. Porto Alegre: Bookman, 2000.

GEEKHUNTER, Flutter: por que aprender o framework da Google é uma boa ideia em um mercado mobile crescente. **Blog Geekhunter**, 2021. Disponível em: https://blog.geekhunter.com.br/flutter/. Acesso em: 07jun. 2022.

ISTO É DINHEIRO. Número de usuários de Internet no mundo chega aos 4,66 bilhões. **ISTO É DINHEIRO**, 2022.Disponível em: https://www.istoedinheiro.com.br/numero-de-usuarios-de-internet-no-mundo-chega-aos-466-bilhoes/. Acesso em: 07jun. 2022.

MELO, Ana Cristina, **Desenvolvendo aplicações com UML 2.2**: do conceitual à implementação. 3. ed. Rio de Janeiro: Brasport, 2010.

ORLANDI, C. Firebase: serviços, vantagens, quando utilizar e integrações. **Blog Rocketseat**, 2022. Disponível em: https://blog.rocketseat.com.br/firebase/. Acesso em: 24 set. 2022.

OXFORD. **English Dictionary.** Oxford: Oxford University Press, 2017.

PINHEIRO, F. Flutter: O que são widgets e qual sua importância. **Blog Flutter**, 2022. Disponível em: https://www.treinaweb.com.br/blog/flutter-o-que-sao-widgets-e-qual-sua-importancia. Acesso em: 05 jun. 2022.

ZAMMETTI, Frank. **Flutter na Prática**: Melhore seu Desenvolvimento Mobile com o SDK Open Source Mais Recente do Google. São Paulo: Novatec, 2020.