FACULDADE DE TECNOLOGIA DE SÃO PAULO

AMANDA ANJOLIN RODRIGUES

DESENVOLVIMENTO DE APLICATIVO ANDROID UTILIZANDO OS SERVIÇOS DO FIREBASE

SÃO PAULO
2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

AMANDA ANJOLIN RODRIGUES

DESENVOLVIMENTO DE APLICATIVO ANDROID UTILIZANDO OS SERVIÇOS DO FIREBASE

Trabalho submetido como exigência parcial para obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas
Orientador: Profº Sergio Luiz Banin

SÃO PAULO
2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

AMANDA ANJOLIN RODRIGUES

DESENVOLVIMENTO DE APLICATIVO ANDROID UTILIZANDO OS SERVIÇOS DO FIREBASE

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador

_____
_____
_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinada por mim para avaliação do TCC.

Orientador: Profº Sergio Luiz Banin

SÃO PAULO, ____ de _____ de 2021.

Assinatura do Orientador

Assinatura do aluno

RESUMO

Conforme a informatização foi ganhando espaço no dia a dia das pessoas, houve um alto crescimento da demanda por novas aplicações que possam suprir as mais variadas necessidades da população. Grande parte dessa demanda refletiu sobre as aplicações *mobile*, uma vez que os *smartphones* vêm alcançando cada vez mais um lugar de destaque entre os dispositivos da atualidade.

Ao passo que o mercado de desenvolvimento *mobile* fica cada vez mais competitivo, é necessário ponderar as tecnologias utilizadas nessa jornada de desenvolvimento a fim de aprimorar as metodologias e tornar esse processo cada vez mais escalável. Nessa perspectiva, a utilização da Computação em Nuvem apresenta-se como alternativa para otimização de recursos computacionais, visto que oferece serviços acessíveis de maneira remota.

No âmbito das aplicações *mobile*, o serviço em nuvem de *backend*, chamado de *Backend as a Service*, surge como uma solução para as adversidades enfrentadas nesse meio. Este presente trabalho objetiva apresentar o conceito de *Backend as a Service*, bem como demonstrar sua implementação através do estudo e desenvolvimento de uma aplicação *mobile* real. Além disso, objetiva evidenciar esta implementação utilizando a ferramenta Firebase, desenvolvida pela Google, e seus principais serviços proveitosos a construção de um *backend*, visando corroborar com a concepção de simplicidade e agilidade no emprego de tais ferramentas.

Palavras-chave: *backend, Backend as a Service,* computação em nuvem, *mobile*, Firebase.

ABSTRACT

As computerization gained ground in people's daily life, a high growth in demand for new applications that could supply the most varied needs of the population was created. Large part of this demand reflected on mobile applications, since smartphones have been reaching more and more a prominent place among current devices.

While the mobile development market becomes increasingly competitive, it is necessary to consider the technologies used in this development journey in order to improve methodologies and make this process more and more scalable. From this point of view, cloud computing presents itself as an alternative to optimize computational resources, as it offers remotely accessible services.

In the context of mobile applications, the *backend* cloud service, called *Backend as a Service*, comes as a solution to the adversities faced in this environment. The purpose of this work is to present the concept of the "Backend as a Service", as well as demonstrate its implementation through the study and development of a real mobile application. In addition, it aims to evidence this implementation using the Firebase tool, developed by Google, and its main useful services to build a backend, aiming to corroborate with the concept of simplicity and agility in the use of such tools

Keywords: backend, Backend as a Service, Cloud Computing, mobile, Firebase.

LISTA DE ILUSTRAÇÕES

Figura 1 – Serviços oferecidos pelo Firebase 15
Figura 2 – Ferramentas do Google disponíveis pelo Firebase 15
Figura 3 – Diagrama de Classe 21
Figura 4 – Página inicial do Firebase 22
Figura 5 – Página de adição do projeto ao Firebase 23
Figura 6 – Página de download do google.services.json 23
Figura 7 – Estrutura de coleções do Cloud Firestore 26
Figura 8 – Primeira etapa para criação de link dinâmico 32
Figura 9 – Segunda etapa para criação de link dinâmico 33
Figura 10 – Modelos de exibição de mensagem 34
Figura 11 – Implementação do serviço InAppMessaging na aplicação 35
Figura 12 – Telas de login e cadastro 36
Figura 13 – Tela de conversas 37
Figura 14 – Telas de chat e contatos 38

LISTA DE CÓDIGOS

Código 1 – Regras de implementação do Firebase 24
Código 2 – Inserção do SDK do Firebase 24
Código 3 – Implementação do Cloud Firestore 27
Código 4 – Criação de uma coleção no Cloud Firestore 27
Código 5 – Atualização de mensagens de uma conversa..... 28
Código 6 – Criação de usuário 29
Código 7 – Implementação do envio de imagem 21

1. INTRODUÇÃO

Com a crescente disseminação do uso dos *smartphones* no dia a dia das pessoas, a busca pela implementação de serviços através de aplicações móveis vem aumentando de forma acelerada. Em 2019, a presença dos microcomputadores nos domicílios brasileiros apresentou uma queda quando comparada a anos anteriores. Por outro lado, os aparelhos celulares estão presentes em 94% dessas residências, número que vem aumentando gradativamente. Além disso, 98,6% das pessoas acessaram a internet através do aparelho celular, enquanto apenas 46,2% acessaram de um microcomputador (IBGE, 2019).

Pode-se então considerar que essa mudança na forma com a qual as pessoas acessam as informações e realizam suas tarefas e serviços diários impacta nas soluções tecnológicas que as empresas do mercado brasileiro oferecem a seus usuários, visando atender a uma demanda que só cresce, e tornando o desenvolvimento de aplicações móveis cada vez mais competitivo.

Surgem então diversas linguagens, *frameworks*, serviços e ferramentas de desenvolvimento que buscam evoluir cada dia mais os *softwares* da forma como conhecemos hoje e garantir uma infraestrutura adequada para a gestão das informações. Entre eles, a computação em nuvem surgiu como uma solução em termos de capacidade de armazenamento e processamento. Segundo a Microsoft (2021):

> "A computação em nuvem é o fornecimento de serviços de computação, incluindo servidores, armazenamento, bancos de dados, rede, software, análise e inteligência, pela Internet ("a nuvem") para oferecer inovações mais rápidas, recursos flexíveis e economias de escala."

Dentre algumas opções de modelo de serviço de computação em nuvem, o *Backend as a Service* (*BaaS*) oferece soluções para a construção do *backend* de uma aplicação. O *backend* é a parcela da aplicação responsável por garantir sua estabilidade e bom funcionamento, incluindo: banco de dados, armazenamento, troca de informações, processamento e segurança.

O *Backend as a Service* possibilita, através da utilização de interfaces de programação de aplicativos (API) e kits de desenvolvedores de software (SDK), a implementação de um *backend* com muito mais facilidade e de forma simplificada, fazendo a integração diretamente com o *frontend.*

A ferramenta Firebase apresentada neste documento trata-se de uma das opções disponíveis no mercado atual para implementação de um *BaaS* em uma aplicação *mobile.*

O presente trabalho objetiva demonstrar através do desenvolvimento de um aplicativo Android, os serviços disponíveis pelo Firebase, bem como suas características e aplicação em um projeto real. Além disso, visa explanar sobre os elementos que cercam o conceito de um *BaaS.*

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresentará os principais temas relevantes para a compreensão deste trabalho, a fim de esclarecer os conceitos básicos que amparam seu desenvolvimento.

2.1 Computação em nuvem

De acordo com o NIST (*National Institute of Standarts and Technology*), a definição de Computação em Nuvem pode ser dada como:

> *"Um modelo para habilitar o acesso por rede ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos de computação (como redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços."*

Por síntese, pode-se descrever a computação em nuvem como a disponibilidade através da internet de recursos computacionais, como armazenamento e a capacidade de processamento, onde o usuário paga apenas pelo que foi utilizado do serviço.

Sua criação surgiu na década de 1950, com uma proposta de *time-sharing*, ou seja, compartilhar o tempo ocioso das máquinas entre os usuários, onde cada um pagaria apenas pelo tempo de utilização. Porém, sua implementação como conhecemos hoje só foi possível a partir da década de 2000, ganhando popularidade conforme as empresas entendiam melhor seu funcionamento.

Hoje, a nuvem é utilizada para armazenamento de dados, emails, desenvolvimento, análise de *big data,* aplicativos *web, mobile* e outros. Suas características principais são, segundo o NIST:

- **Auto-atendimento sob demanda**: diz respeito a possibilidade de um consumidor utilizar os serviços da nuvem de maneira que seja possível aumentar ou diminuir sua capacidade computacional, bem como

armazenamento de rede e tempo de servidor, sem que seja necessária interação humana entre as partes.

- **Elasticidade rápida:** preocupação em provisionar recursos para atender momentos de pico ou de maior demanda de serviço, uma vez que seus serviços se adequam e provisionam apenas o que for necessário para atender as demandas

- ***Pool* de recursos:** os recursos de TI são projetados para atender a vários clientes, agrupados geograficamente, com recursos físicos e virtuais distribuídos dinamicamente. Os clientes não têm controle sobre a localização física dos recursos que estão usando, apenas informações mais amplas, como o país em que estão, estado ou datacenter.

- **Amplo acesso as redes:** significa dizer que a nuvem dispões de serviços que podem ser acessados de qualquer plataforma. Os mecanismos padrão usados são benéficos para o uso de plataformas heterogêneas. Portanto, os clientes podem acessar de seus telefones celulares, PC ou de qualquer outra plataforma.

- **Serviços mensuráveis:** os sistemas em nuvem são monitorados e otimizados automaticamente, medindo o uso, de maneira apropriada ao tipo de serviço, como memória utilizada, potência de processamento realizada ou contas de usuários ativos. Tem como finalidade que a utilização dos recursos possa ser monitorada e controlada de forma transparente para todas as partes.

A computação em nuvem possui diferentes tipos de modelos de implementação, que se diferem em acessibilidade e disponibilidade em um ambiente de *cloud computing*. A escolha de um modelo está diretamente ligada as questões supracitadas de acessibilidade e disponibilidade, de forma que para cada organização é definida os níveis de acesso aos recursos em nuvem, por exemplo, e consequentemente o tipo de modelo a ser utilizado. Os tipos definidos pelo NIST são:

- **Nuvem pública:** nesse modelo, as infraestruturas são construídas de forma a disponibilizar o acesso aos dados para todos os usuários, portanto, não é recomendada para organizações cujo dados são confidenciais.

- **Nuvem privada:** diferentemente da nuvem pública, este modelo é construído para ser um ambiente seguro e protegido, possuindo restrição de acesso à usuários específicos. Sua infraestrutura confere ao usuário, e sendo assim igualmente seu controle para implementação de aplicações.

- **Nuvem comunitária:** sua infraestrutura é baseada no compartilhamento por diversas organizações que possuem interesses em comum. Como exemplo podemos citar os requisitos de segurança e políticas. Nesse modelo de comunidade, o gerenciamento da nuvem pode ocorrer tanto pelas próprias organizações quanto por um terceiro agente externo.

- **Nuvem híbrida:** é composta por duas ou mais nuvens, independentemente de seus tipos individuais, que permanecem como entidade únicas, porém são compostas pela junção através de proprietárias de tecnologia padronizada que permite a portabilidade dos dados e aplicações. Esse modelo permite que os recursos de uma nuvem privada aumentem a partir de um *pool* de recursos em uma nuvem pública.

A computação em nuvem oferece diferentes tipos de serviços baseados no provisionamento e disponibilização de recursos. As principais categorias de serviços de nuvem são:

- ***Infrastructure as a Service – IaaS:*** nesse modelo, o provedor fornece ao cliente o provisionamento de recursos de hardware, processamento, armazenamento de dados, rede e recursos básicos de computação, além de sistemas operacionais, na forma de máquinas e dispositivos virtuais. O usuário não tem acesso a infraestrutura da nuvem em si, apenas o provedor.
- ***Platform as a Service – PaaS:*** o cliente que contrata esse serviço tem a disposição uma plataforma para desenvolvimento e teste de

aplicações em nuvem, ao invés de possuir esses serviços em máquinas internas. O provedor oferece então linguagens de programação, bibliotecas e ferramentas para desenvolvimento.

- ***Software as a Service – SaaS:*** o usuário executa uma aplicação que está hospedada na nuvem. O provedor irá fornecer a aplicação, enquanto o usuário pode acessá-la através da internet, por exemplo, e assim utiliza esse serviço. A aplicação pode ser executada num navegador *web* ou em uma interface de programa.

Essas três categorias são apresentadas pelo NIST e podem ser consideradas as principais e mais utilizadas, entretanto, existem outras modalidades de serviço em nuvem que cabe ressaltar, como: *Database as a Service, Hardware as a Service* e *Backend as a Service.* Sendo o *BaaS* objeto de estudo deste trabalho, seu conceito e definição estão contemplados na subseção a seguir.

2.1.1 Backend as a Service

O *Backend as a Service* é um serviço que fornece soluções que realizam a conexão de aplicativos ao armazenamento e processamento em nuvem do *backend* para desenvolvedores *mobile* e *web*. Além disso, também fornecem outros recursos pertinentes a um *backend* como: gerenciamento de usuários, notificações e integração de redes sociais (LANE, 2015).

> *"Um dos principais objetivos do BaaS é automatizar atividades repetitivas e evitar com que os desenvolvedores realizem tarefas de baixo valor agregado e foque nas tarefas complexas e de alto valor agregado e que não podem ser automatizadas. O custo total do desenvolvimento de um Backend pode ser reduzido em até 80% e o tempo de desenvolvimento do Backend também pode ser minimizado (BATSCHINSKI, 2016)."*

Ainda segundo Lane (2015), por mais que o *BaaS* tenha como foco principal o desenvolvimento móvel, sua abordagem possui elementos que podem facilmente ser utilizado em outras áreas do desenvolvimento, são elas:

- **Desenvolvimento *web:*** permite que o desenvolvimento *web* se torne mais flexível, uma vez que ele prove recursos como um *PaaS*, porém com menos restrições de interface.
- **Desenvolvimento *mobile*:** é o principal foco dos provedores, portanto possui grande concentração de otimizações de dados, utilizando diversos recursos como geolocalização.
- **Leitores:** refere-se à publicação no Kindle e em dispositivos cuja finalidade é oferecer suporte à publicação de livros e revistas.
- **Lançar API:** o *BaaS* fornece estruturas de API facilitadas para implantação e rápidas para lançamento.

Os serviços de um *BaaS* são ofertados através de provedores, que visam implementar soluções modulares para as funções mais comumente utilizadas. Porém, para além dessas funções, também possuem outros diferenciais competitivos entre si.

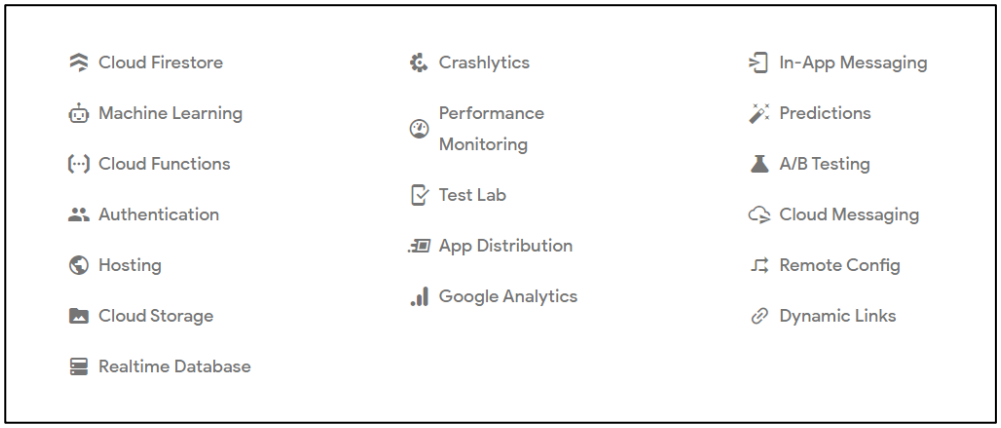
Algumas opções de provedores mais conhecidos são: Back4App, Kinvey, AWS Amplify e Firebase. Devido a sua ampla divulgação e alta confiabilidade em seus recursos, a opção escolhida para estudo de caso desse trabalho é o Firebase, que tem seus aspectos definidos na subseção abaixo.

2.2 Firebase

O Firebase é uma ferramenta que oferece serviços em nuvem para a construção completa de uma infraestrutura *backend* para aplicações *mobile* e *web*. Entre os serviços oferecidos, estão inclusos banco de dados, autenticação de usuários, armazenamento de arquivos, envio de notificações e diversos outros, inclusive alguns voltados para engajamento dos usuários, que podem ser visualizados na Figura 1. O Firebase possui suporte para as plataformas Android,

iOS, Web, C++ e Unity, e para as linguagens Swift, Objective-C, Java, Kotlin, JavaScript, C++ e Unity (FIREBASE, 2021).

Figura 1 – Serviços oferecidos pelo Firebase



Fonte: Firebase (2021).

Além disso, também possui integração com ferramentas do Google como *Google Ads* e *AdMob,* que são gerenciadores de anúncios e publicidades pagas. Essa integração pode trazer resultados positivos para tomada de decisão quando utilizada em conjunto aos serviços de análise de comportamento, também disponíveis pelo Firebase.

Figura 2 – Ferramentas do Google disponíveis pelo Firebase

Fonte Firebase (2021).

Nesta seção, serão abordados com foco principal exclusivamente os serviços do Firebase implementados na aplicação e suas características essenciais, com o propósito de gerar uma base de conhecimento para as implementações futuras.

Sendo assim, as principais ferramentas ofertadas e aplicadas neste trabalho são:

- ***Cloud Firestore:*** serviço de banco de dados NoSQL -tecnologia definida na seção 2.3.2, que permite o armazenamento de dados e a sincronização entre usuários e dispositivos em tempo real. No caso das aplicações *mobile*, os dados armazenados no banco de dados ficam disponíveis mesmo em estado *offline,* ou seja, sem conectividade com a internet. Além do mais, o *Cloud Firestore* tem grande potencial de escalabilidade, uma vez que suas consultas possuem desempenho proporcional ao seu conjunto de resultados.

- ***Authentication:*** serviço para autenticação de usuários baseado em métodos para autenticação de diversas formas, possibilitando um sistema completo de *login* e *signup*. É possível implementar a autenticação por meio das seguintes opções:
    - email e senha
    - provedores de identidade federada (Google, Facebook, Twitter, Github)
    - número de telefone
    - sistema personalizado

- ***Storage:*** serviço de armazenamento de arquivos em nuvem. Quando integrado aos aplicativos do Firebase, torna possível o acesso às práticas de segurança do Google e a proteção de *upload* e *download* nas aplicações. Através de seu SDK, também é possível gerenciar as mídias e acessá-las diretamente da conta.

- ***Dynamic Links:*** serviço de criação de *links* que encaminham o usuário diretamente para um conteúdo da aplicação. Esse serviço permite enviar usuários tanto da plataforma Android, quanto de iOS para um local da aplicação através do mesmo link. Funcionam também nas situações em que o usuário ainda não possui a aplicação instalada. Nessas situações, o *link* irá redirecioná-lo após a instalação, sem que ocorra a perda de dados.

- ***InAppMessaging:*** é o serviço de envio de mensagens customizadas para usuários da aplicação. Por meio desse serviço, os *pop-ups* enviados podem ser contextuais e personalizados para grupos específicos de usuários, o que

os torna mais relevantes para quem recebe. Possui opções de mensagens customizadas como cartões, *banners* e *modais.*

Para ter acesso a esses serviços oferecidos pelo Firebase, são ofertados dois planos diferentes: o Spark e o Blaze. O primeiro é o plano gratuito inicial que todas as aplicações recebem como padrão primeiramente. Esse plano, chamado de Spark, oferece praticamente todos os serviços do Firebase. Sua restrição está relacionada a quantidade de utilização dos serviços, ou seja, ele possui um limite de uso para a maior parte das funções.

Quando extrapolado esse limite, o Firebase oferece seu segundo plano: o Blaze. Este por sua vez oferece alguns serviços de forma gratuita, porém em menor quantidade que o primeiro, e sua cobrança é baseada na utilização dos serviços de forma unitária. No Quadro 1 são apresentados ambos os planos e suas comparações em relação aos serviços anteriormente descritos nessa seção e empregados na aplicação desenvolvida neste trabalho.

Quadro 1 – Planos do Firebase

Serviço	**Spark**	**Blaze**
Cloud Firestore	Grátis até 1*GB*	$0,108/*GB*
Authentication (email/senha)	Grátis	Grátis
Storage	Grátis até 5*GB*	$0,026/*GB*
InAppMessaging	Grátis	Grátis
Dynamic Links	Grátis	Grátis

Fonte: adaptado de Firebase (2021).

Com a finalidade de escolher uma ferramenta para estudo e desenvolvimento de um *backend* de uma aplicação *mobile*, foram analisados os provedores mais comentados e com maior relevância para esse objetivo.

2.3 Tecnologias Utilizadas

2.3.1 Android

O Android é um sistema operacional móvel de código aberto que possui uma plataforma baseada em Linux. O Android surgiu a partir da parceria da Google com a OHA, essa união configura os principais atores do cenário móvel mundial (SCHEMBERGER; FREITAS; VANI, 2009, p.2).

Além da OHA, a Google possui parceira com outras grandes empresas como fabricantes de celulares e operadoras móveis, algumas delas são: Samsung, ASUS e Acer (OHA, 2016).

O Android foi inicialmente criado para *smartphones* e *tablets,* porém com sua crescente expansão já pode ser encontrado em outros dispositivos como câmeras, relógios e TVs (ANDROID, 2017). Através de seu sistema, o Android proporciona navegação na internet, criação de documentos, gerenciamento de contatos e funções de GPS e localização (SIMÃO et. al., 2011, p.1).

O desenvolvimento para plataforma Android pode se dar através de duas linguagens: Java e Kotlin. Para a aplicação desenvolvida, usou-se a linguagem Java, descrita a seguir.

2.3.1.1 Java

A linguagem de programação Java é também uma plataforma computacional lançada em 1995 pela Sun Microsystems (JAVA, 2021). Essa linguagem possui orientação a objetos (POO) e por isso implementa características particulares como herança e encapsulamento (DEITEL, 2010). Além disso, é uma linguagem estaticamente tipada, o que significa dizer que cada variável deve ter seu tipo de dados declarado previamente

Sua utilização para o desenvolvimento Android está ligada ao fato de que a linguagem Java possibilita manter um sistema extensível e ainda com flexibilidade,

além de ser multiplataforma e possuir processamento múltiplo. Sua documentação é amplamente divulgada e, portanto, de fácil acesso a novos desenvolvedores.

2.3.2 NoSQL

O NoSQL (*Not Only SQL)* surgiu como uma alternativa ao banco de dados relacional. O NoSQL representa todos os bancos de dados que não seguem a estrutura antes estabelecida e relacional, ou seja, é um termo abrangente para categorizar diversos bancos existentes com abordagens arquitetônicas em comum. Esses bancos de dados não utilizam as já conhecidas tabelas e colunas, pois possuem um sistema distribuído, o que significa dizer que seus dados estão distribuídos por servidores (FOWLER, 2021).

A sua popularidade aumentou quando grandes empresas como Facebook e Google passaram a utilizá-los.

Alguns autores como Gonzalez et al. (2016), Farias et al. (2016), Corbellini et al. (2016) citam em suas obras que um banco de dados NoSQL pode ser uma alternativa quando se fala de aplicações que armazenam e processam dados em massa., uma vez que disponibilizam recursos eficazes para armazenamento de grandes quantidades de dados, além de escalabilidade e baixo custo.

3. DESENVOLVIMENTO DA APLICAÇÃO

A fim de demonstrar a aplicabilidade de um modelo *Backend as a Service* como uma opção ao desenvolvimento *backend* para plataformas *mobile,* bem como abordar os serviços da ferramenta Firebase implementados em uma aplicação real, este trabalho irá abordar o desenvolvimento de uma aplicação para plataforma Android, de forma a exemplificar a utilização desses serviços.

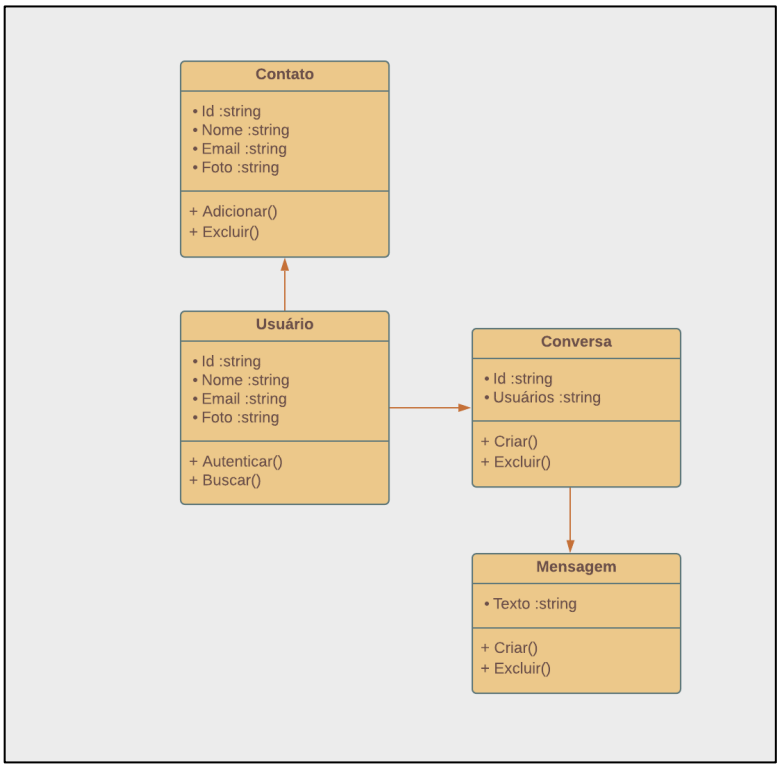
Baseado nos altos índices que indicam o serviço de troca de mensagens como principal utilidade dos *smartphones* para a população, a aplicação escolhida para desenvolvimento e estudo deste trabalho trata-se de um gerenciador pessoal de conversas e mensagens, ou seja, um *chat* entre usuários.

A aplicação foi desenvolvida para a plataforma Android, utilizando a linguagem de programação Java, e seu *BaaS* foi construído através dos serviços do Firebase. O emprego dessas ferramentas visa solidificar a possibilidade e capacidade de implementação de diversas funções a partir de um *BaaS* com o Firebase.

Em termos gerais, a aplicação permitirá adicionar outros usuários, chamados de contatos, em sua lista de contatos, e se comunicar através de mensagens escritas. Todas os dados são sincronizados em tempo real com todos os usuários, essa funcionalidade é possibilitada pelo *Cloud Firestore*, banco de dados do Firebase. Além dele, os serviços de *Storage, Authentication e Dynamic Links* também serão utilizados neste projeto.

Com objetivo de especificar os componentes do sistema e suas interligações graficamente, foi elaborado o diagrama de classes (Figura 3) no qual contém as especificações de classes e associações da aplicação.

Figura 3 – Diagrama de Classe



Fonte: autoria própria.

Conforme representado no diagrama, a classe Usuário corresponde a todos que possuírem cadastro, não havendo, pelo menos inicialmente, níveis diferentes de acesso à diferentes tipos de usuários. Os usuários podem autenticar no sistema e buscar por outros usuários. Seus atributos são para uso de identificação como *id*, nome, email e foto.

A classe Contato representa o vínculo entre usuários. Um usuário terá uma lista de contatos com os quais pode iniciar uma conversa. Essa classe possui os atributos comuns a classe Usuário, e as funções de manutenção adicionar e excluir.

A classe Conversa diz respeito a uma conversa iniciada entre dois usuários do sistema. Possui o atributo de identificação *id* e seus usuários participantes. Suas funções são criar e excluir, com finalidade de manter essas conversas.

A classe Mensagem se relaciona com a classe Conversa, e trata-se das mensagens enviadas entre os respectivos usuários de uma conversa. Assim como a

classe anteriormente citada, suas funções são criar e excluir, com finalidade de manter as mensagens.

Uma vez que definida a ideia principal do projeto como uma aplicação para troca de mensagens entre usuários, definiu-se a plataforma *mobile* Android como foco de estudo e desenvolvimento da aplicação, mais informações sobre o Android na seção 2.3.1.

No cenário atual, as linguagens Kotlin e Java estão disponíveis para o desenvolvimento de aplicações nativas Android. Neste trabalho, será utilizada a linguagem Java, descrita na seção 2.3.1.1, devido a sua ampla difusão e quantidade de informações divulgadas, tornando o desenvolvimento mais estável.

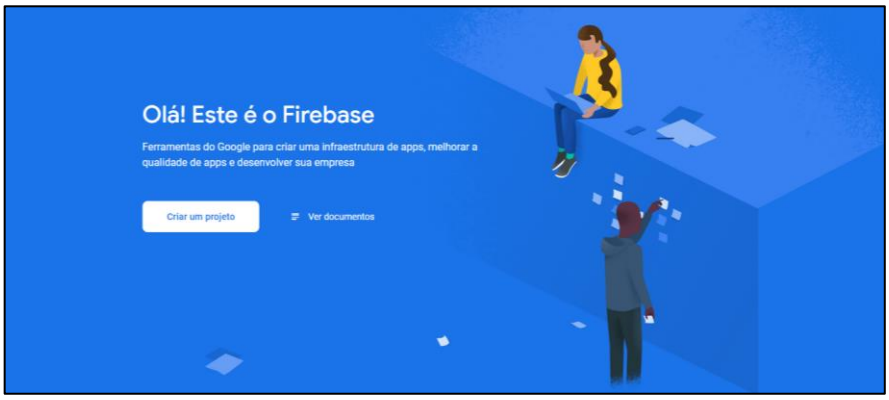
A integração entre a aplicação Android e o Firebase pode se dar de duas formas distintas.

1) Utilizando fluxo de trabalho de configuração do Console do Firebase.
2) Utilizando Firebase Assistente do Android Studio.

No presente trabalho, utilizou-se a opção de número 1, posto que esta é a forma recomendada pela própria ferramenta. Para realizar a integração, foram seguidos os passos descritos abaixo.

1) Criação do projeto no Firebase

Figura 4 – Página inicial do Firebase



Fonte: adaptado de Firebase (2021).

Primeiramente, é necessário criar um novo projeto em “Criar um projeto” conforme a Figura 4. Nessa etapa é preciso preencher algumas informações sobre o novo projeto e concluir a criação aceitando os Termos de Uso.

2) Registro do aplicativo com o Firebase

Figura 5 – Página de adição do projeto ao Firebase

Fonte: adaptado de Firebase (2021).

Nesta etapa, é necessário preencher o formulário de adição do projeto, conforme a Figura 5.

3) Adicionar arquivo de configuração

Figura 6 – Página de download do *google.services.json*

Fonte: adaptado de Firebase (2021).

O download do arquivo de configuração *google-services.json* deve ser realizado e após concluído, é necessário move-lo para o diretório de módulos do aplicativo.

Para ativar os serviços do Firebase no aplicativo, é preciso incluir algumas regras no build.gradle no nível raiz do projeto.

Código 1 – Regras de implementação do Firebase

```
buildscript {
  repositories {
    google()
  }
  dependencies {
    classpath 'com.google.gms:google-services:4.3.10'
  }
}
allprojects {
  repositories {
    google()
  }
}
```

Fonte: adaptado de Firebase (2021).

Em seguida, devem ser adicionadas no build.gradle no nível do aplicativo, novas regras e o *SDK* do Firebase. Conforme o seguinte código.

Código 2 – Inserção do SDK do Firebase

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
dependencies {
  implementation platform('com.google.firebase:firebase-bom:29.0.0')
  implementation 'com.google.firebase:firebase-analytics'
}
```

Fonte: adaptado de Firebase (2021).

3.1 Serviços do Firebase

Nessa seção serão apresentados alguns dos serviços ofertados pelo Firebase em seu plano inicial gratuito, através dos quais possibilitou o desenvolvimento de um *backend* completo para a aplicação.

As funcionalidades de cada um dos serviços utilizados serão demonstradas, assim como sua integração com a plataforma Android e linguagem Java, a fim de promover maior entendimento sobre a ferramenta principal desse estudo e consequentemente maior clareza acerca da aplicação desenvolvida.

Todos os serviços citados nas subseções a seguir devem ter sua dependência declarada no arquivo build.gradle no nível do *app* do módulo. As dependências de cada serviço são apresentadas no Quadro 2.

Quadro 2 – Dependências dos Serviços do Firebase

Serviço	**Dependência**
Cloud Firestore	`implementation 'com.google.firebase:firebase-firestore'`
Authentication	`implementation 'com.google.firebase:firebase-auth'`
Storage	`implementation 'com.google.firebase:firebase-storage'`
Dynamic Links	`implementation 'com.google.firebase:firebase-dynamic-links'`
InAppMessaging	`implementation 'com.google.firebase:firebase-inappmessaging-display'`

Fonte: adaptado de Firebase (2021).

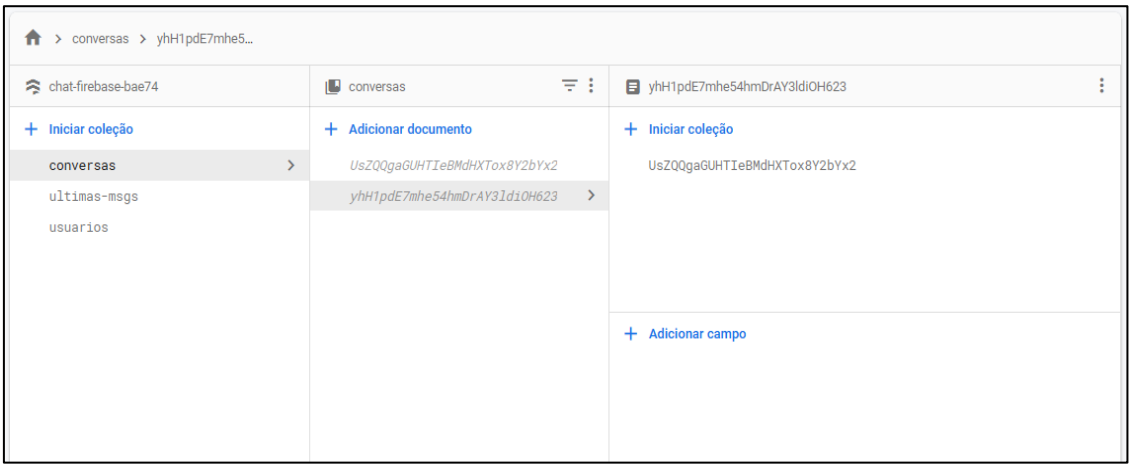
3.1.1 Cloud Firestore

O sistema de banco de dados da aplicação foi construído com o *Cloud Firestore*, serviço de banco de dados NoSQL do Firebase, que permite sincronizar dados para o desenvolvimento do lado do servidor e do cliente (FIREBASE, 2021).

O *Cloud Firestore,* descrito previamente na seção 2.2, permite a sincronia total dos dados em tempo real, além de oferecer suporte para uso off-line. Essas funções podem ser consideradas de grande importância na aplicação de *chat*, dado que a troca de mensagens tende a ocorrer de forma simultânea entre os usuários, requerendo então uma sincronia em tempo real. Ademais, a possibilidade de visualizar as mensagens de uma conversa, mesmo que na ausência de conectividade com a internet, ou seja, off-line, é igualmente de suma importância.

A estruturação desse banco de dados se dá por meio de coleções que possuem documentos. Este projeto segue a estrutura de coleções e documentos representadas na Figura 7.

Figura 7 – Estrutura de coleções do *Cloud Firestore*



Fonte: adaptado de Firebase (2021).

A coleção de usuários retrata todos os cadastros de usuários efetuados no sistema. Seus documentos são representados pelo *id* do usuário, e seus campos são nome, perfilUrl, tempo, ultimaMsg, uuid.

A coleção conversas corresponde as conversas iniciadas entre usuários. Nela, os documentos são identificados pelo *id,* tanto do usuário que iniciou a conversa, quanto do adicional usuário participante. As conversas são duplamente registradas, de maneira que integre sua existência para ambos os usuários.

A coleção ultimas-msgs foi criada com o objetivo de destinar as últimas mensagens enviadas em uma determinada conversa para a exibição na tela inicial do aplicativo, de forma mais acessível ao *Cloud Firestore*.

Para sua implementação é necessário inicializar uma instância do *Cloud Firestore*, conforme o Código 3.

Código 3 – Implementação do *Cloud Firestore*

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
```

Fonte: autoria própria.

A criação de coleções e documentos acontece de forma implícita ao passo que se faz a inserção de novos dados, sem necessidade de cria-los explicitamente. Como apresentado no Código 4, a criação da coleção “usuários” acontece no primeiro momento de inserção de dados.

Código 4 – Criação de uma coleção no *Cloud Firestore*

```
db.collection("usuarios")
        .document(uid)
        .set(usuario)
        .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                ...
            }
        });
```

Fonte: autoria própria.

No que se refere a uma das principais funções desse banco de dados, a sincronização em tempo real é implementada utilizando o método *onSnapshot()*, que

pode ser visualizado no código 5. Esse trecho de código é utilizado no método *fetchMessages()* com a finalidade de atualizar sincronicamente as mensagens de uma conversa.

Código 5 – Atualização de mensagens de uma conversa

```
db.collection("/conversas")
        .document(fromId)
        .collection(toId)
        .orderBy("tempo", Query.Direction.ASCENDING)
        .addSnapshotListener(new EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot value, @Nullable
FirebaseFirestoreException error) {
          List<DocumentChange> documentChanges =
value.getDocumentChanges();
          if(documentChanges != null){
            for (DocumentChange doc: documentChanges) {
            if (doc.getType() == DocumentChange.Type.ADDED){
                Log.i("teste","entrou aqui");
                Message message = doc.getDocument().toObject(Message.class);
                adapter.add(new MessageItem(message));
                teste.add(new MessageItem(message));
              }
            }
           }
          }
        });
```

Fonte: adaptado de Firebase (2021).

Esse serviço possui também outras funcionalidades para o gerenciamento dos dados do banco de dados, tornando cada vez mais completa a construção do *backend.*

3.1.2 Authentication

O serviço de reconhecimento de identidade de um usuário é ofertado pelo Authentication do Firebase. Conforme exposto na seção 2.2, o Authentication dispõe de algumas opções de autenticação. Nesse projeto, foi utilizada a autenticação através de email e senha como objeto de estudo do principal tipo de autenticação disponível.

3.1.2.1 Email e Senha

Nesta seção, serão apresentados respectivamente o método de criação de usuários e o método de *login* de usuários através de email e senha empregados na aplicação. O Código 6 demonstra o método desenvolvido para criação de usuário.

Código 6 – Criação de usuário

```
FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, senha)
.addOnCompleteListener(new OnCompleteListener<AuthResult>() {
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
              if(task.isSuccessful()) {
                      saveUser();
               }
      }
});
```

Fonte: autoria própria.

O método *createUserWithEmailAndPassword()*, quando completado com sucesso, chama a função *saveUser()*, que por sua vez implementa o cadastro dos dados do usuário no banco de dados *Cloud Firestore*. Essa inserção no banco de dados é descrita mais detalhadamente na seção 3.1.1.

No que tange o processo de *login* do usuário, foram empregados dois métodos pertinentes ao projeto, são eles:

1. *getCurrentUser():* verifica se existe algum usuário conectado;
2. *signInWithEmailAndPassword():* realiza a autenticação e o *login* de fato;

Contudo, o Firebase dispõe de diversos outros métodos no que se diz respeito ao gerenciamento de autenticação de usuários, cujo principais podem ser consultados no Quadro 3.

Quadro 3 – Métodos de gerenciamento de usuários

Método	**Descrição**
sendEmailVerification()	Envia email de verificação de autenticação
isEmailVerified()	Certifica se o email do usuário foi verificado
updateProfile()	Atualiza informações do perfil do usuário
updateEmail()	Atualiza o email do usuário
updatePassword()	Atualiza a senha do usuário
sendPasswordResetEmail()	Envia um email para redefinição de senha
delete()	Deleta um usuário

Fonte: autoria própria.

3.1.3 Storage

O *Cloud Storage* é um serviço de armazenamento de arquivos do Firebase, através do qual é possível armazenar em nuvem objetos de imagens, vídeos, áudios e outros conteúdos, fazendo a verificação de autorização e permissão de cada usuário para *download* e *upload* desses objetos, fornecendo maior segurança a aplicação de maneira integrada ao *Authentication*.

É apresentado no código X a implementação desse serviço na presente aplicação, que se deu a partir da funcionalidade de *upload* da imagem de perfil do usuário, que ocorre no momento de cadastro.

Código 7 – Implementação do envio de imagem

```
String filename = UUID.randomUUID().toString();
final StorageReference ref =
FirebaseStorage.getInstance().getReference("/images/"+filename);
ref.putFile(mUri)
      .addOnSuccessListener(new
      onSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            ref.getDownloadUrl().addOnSuccessListener(new
            OnSuccessListener<Uri>() {
                @Override
                public void onSuccess(Uri uri) {
                String perfilUrl = uri.toString();
         }
      });
```

Fonte: autoria própria.

O método *putFile()* implementa o *upload* de um arquivo local do dispositivo para o *Cloud Storage*, enquanto que o método *getDownloadUrl()* gera um URL para fazer o *download* do arquivo, usado posteriormente na aplicação para inserção no banco de dados do cadastro do usuário.

A aplicação não possui chamadas aos métodos de segurança disponíveis pelo Firebase, uma vez que todos os usuários podem realizar *upload* e download de arquivos e não há níveis diferentes de permissão neste projeto.

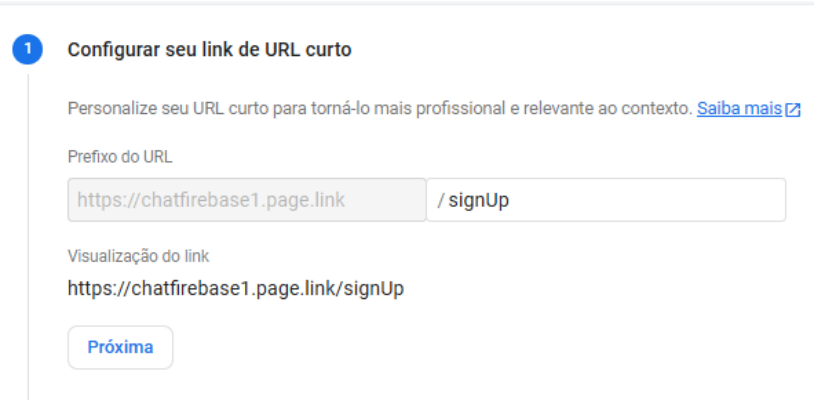
3.1.4 Dynamic Links

O Dynamic Links se trata de um dos serviços do Firebase enquadrados na categoria de Engajamento. Ele tem como função gerar links diretos para redirecionamento até sua aplicação *mobile*. É um serviço que visa solucionar alguns dos problemas enfrentados pelos desenvolvedores ao implementar um link direto, tendo como pontos principais as seguintes soluções:

- Suporte de redirecionamento para plataforma Android, iOS e Web através de um único *link.*
- Redirecionamento para URL alternativa nos casos em que o usuário ainda não possui a aplicação instalada.
- Preservação dos dados enviados pelo *link* durante o processo de instalação da aplicação.

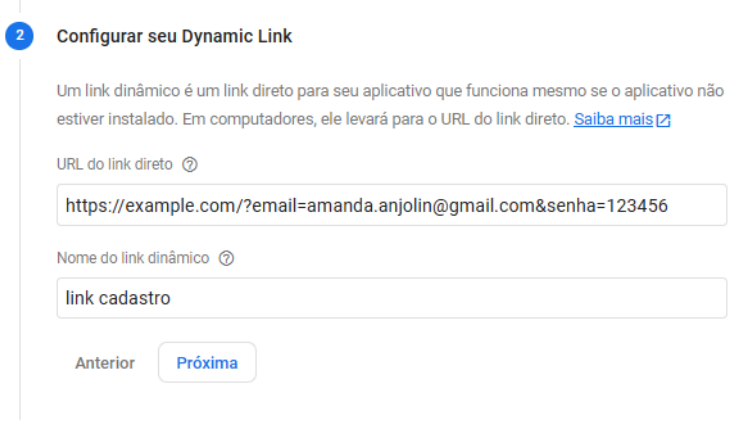
É um serviço facilmente implementado, pois requer apenas alguns passos para sua criação, dos quais a maior parte pode ser executada através do console do Firebase. A Figura 8 apresenta uma das etapas da criação do *link* que será utilizado para redirecionar um usuário para a tela de Cadastro da aplicação desenvolvida neste estudo. Enquanto a Figura 8 demonstra a criação da URL que será divulgada aos usuários, a Figura 9 apresenta a configuração do link que será processado pela aplicação.

Figura 8 – Primeira etapa para criação de link dinâmico



Fonte: Firebase (2021).

Figura 9 – Segunda etapa para criação de link dinâmico



Fonte: Firebase (2021).

O link de processamento usado nesta aplicação teve como finalidade apenas a demonstração da possibilidade de envio de parâmetros personalizados para a aplicação através do *Dynamic Links*. Na prática, o *link* criado desempenha o papel de divulgar o aplicativo na web, direcionando o usuário para a tela de cadastro, possibilitando quantificar o número de cadastros advindos dessa campanha.

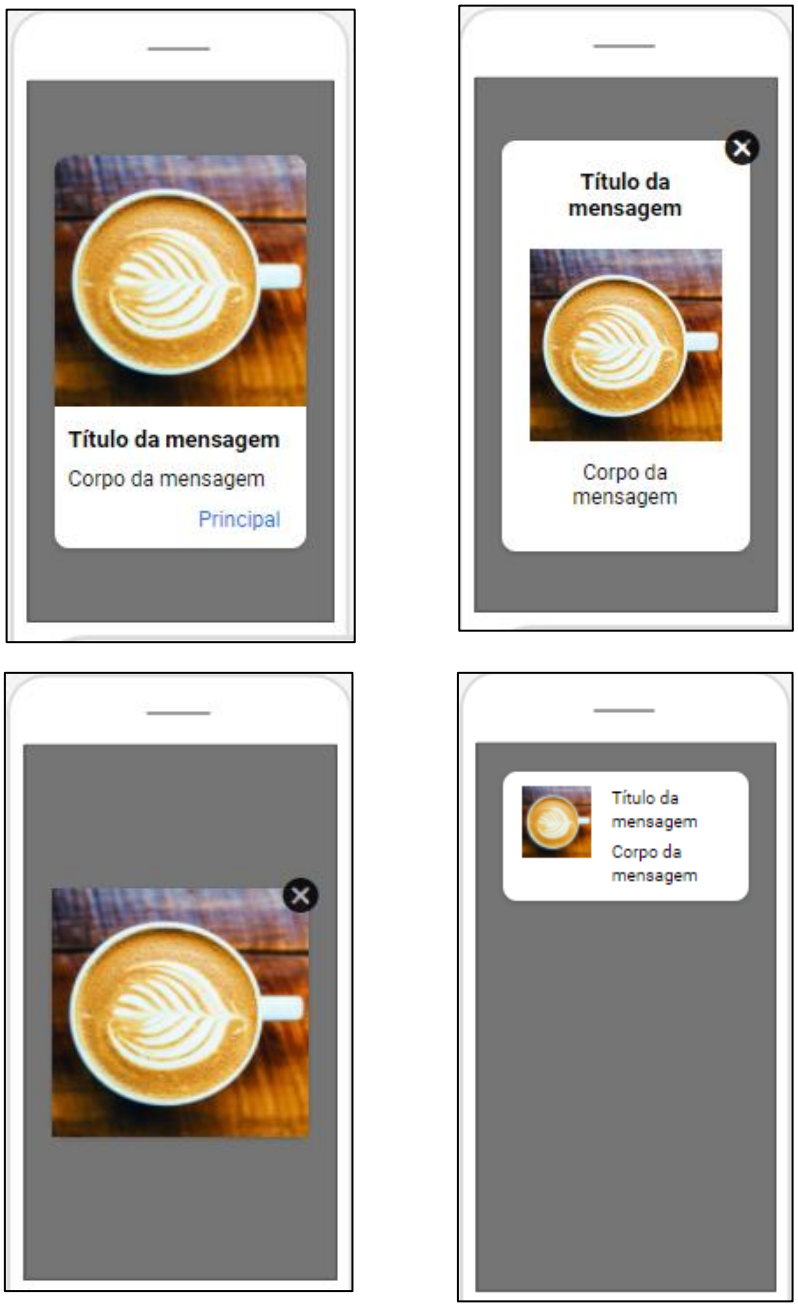
3.1.5 InAppMessaging

Além do Dynamic Links, outro serviço ofertado pelo Firebase com propósito de engajamento de usuários é o InAppMessaging. Esse serviço viabiliza a inserção de mensagens personalizadas na aplicação de forma customizada e direcionada, visto que oferece a possibilidade de criar campanhas diferentes para mensagens e grupos distintos de usuários. Sendo assim, o usuário receberá mensagens de campanhas mais coerentes ao contexto em que está sua experiência na aplicação.

Sob outra perspectiva, a ferramenta traz conjuntamente benefícios para o desenvolvedor, tornando a criação dessas campanhas mais simplificada e menos codificada, uma vez que é possível utilizar apenas o console do Firebase para tal ação.

Para mais, esse serviço dispõe de quatro modelos de exibição da mensagem: cartão, modal, imagem ou banner. Os modelos ofertados podem ser visualizados nas figuras abaixo.

Figura 10 – Modelos de exibição de mensagem



Fonte: adaptado de Firebase (2021).

Com intuito de demonstrar a implementação do serviço em um caso real, inseriu-se na aplicação em desenvolvimento, uma campanha que visa informar o usuário de uma nova atualização da versão do aplicativo, conforme Figura 11.

Figura 11 – Implementação do serviço InAppMessaging na aplicação

Fonte: autoria própria.

Essa mensagem foi enviada para todos os usuários, com a programação para exibição de no máximo uma vez ao dia. O botão de ação o encaminha para um link pré-definido, nesse caso foi utilizado um link genérico da PlayStore, que poderia ser alterado futuramente para um link direto para a página da aplicação específica na PlayStore.

Para essa implementação, não se fez necessário nenhum tipo de inserção de código na aplicação. Portanto, é viável afirmar que esse serviço pode ser utilizado de forma eficiente por pessoas que não possuem o conhecimento técnico de desenvolvimento e programação, tendo como exemplo a equipe de marketing de uma empresa.

3.2 Aplicação Desenvolvida

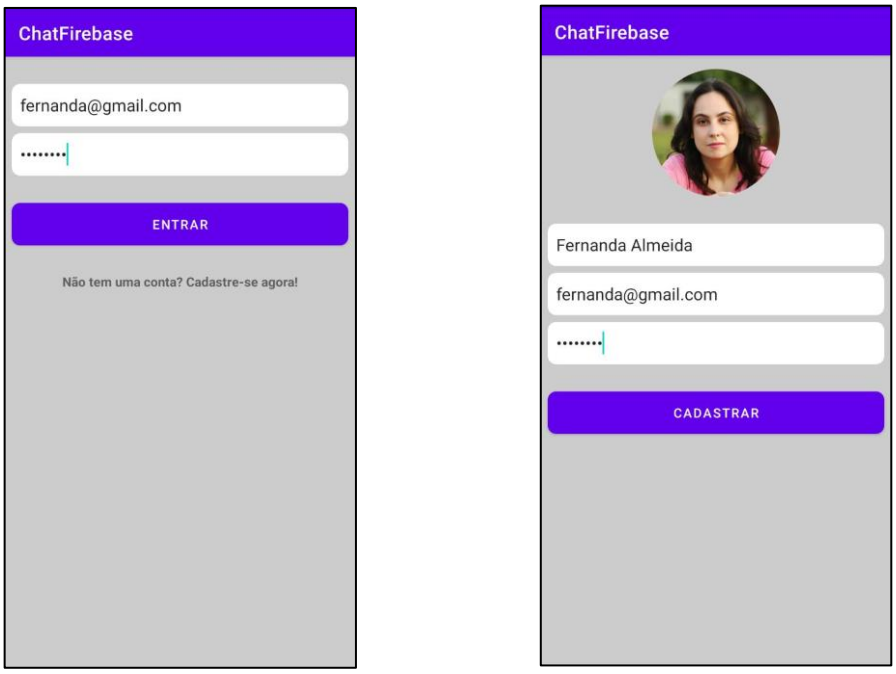
Como meio para aplicação dos serviços de um *BaaS* através da ferramenta do Firebase, foi desenvolvida uma aplicação *mobile* nativa para plataforma Android. A aplicação tem como finalidade a troca de mensagens entre os usuários do sistema através de um *chat* no qual duas pessoas podem participar.

A Figura 12a apresenta a tela inicial do aplicativo, que contém a área destinada ao *login* de usuários já cadastrados através do preenchimento dos campos de email e senha. Logo abaixo, é possível visualizar um botão que realiza o envio desses dados e realiza de fato a autenticação do usuário.

Ao fim da tela, há também outro botão, destinado aos usuários que ainda não possuem cadastro na aplicação. Seu clique redireciona a aplicação para a tela apresentada na Figura 12b.

Essa segunda tela por sua vez, representa a tela de cadastro de novos usuários na plataforma. Todos os seus campos (nome, email, senha e foto) são de preenchimento obrigatório para a correta criação de uma nova conta no aplicativo.

Figura 12 – Telas de *login* e cadastro

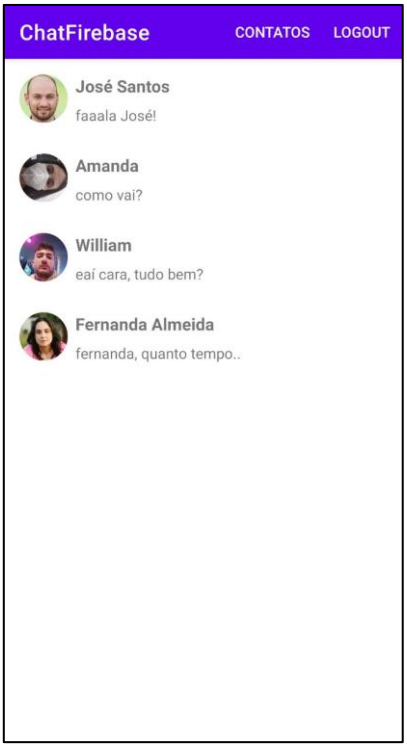


(a) (b)

Fonte: autoria própria.

Após a finalização do *login* ou do cadastro, o usuário é redirecionado para a tela demonstrada na Figura 13. Essa tela exibe as conversas recentes da qual o usuário autenticado faz parte. Para cada uma das conversas, são exibidos o nome, foto e última mensagem enviada, para melhor compreensão e reconhecimento dos *chats*.

Figura 13 – Tela de conversas

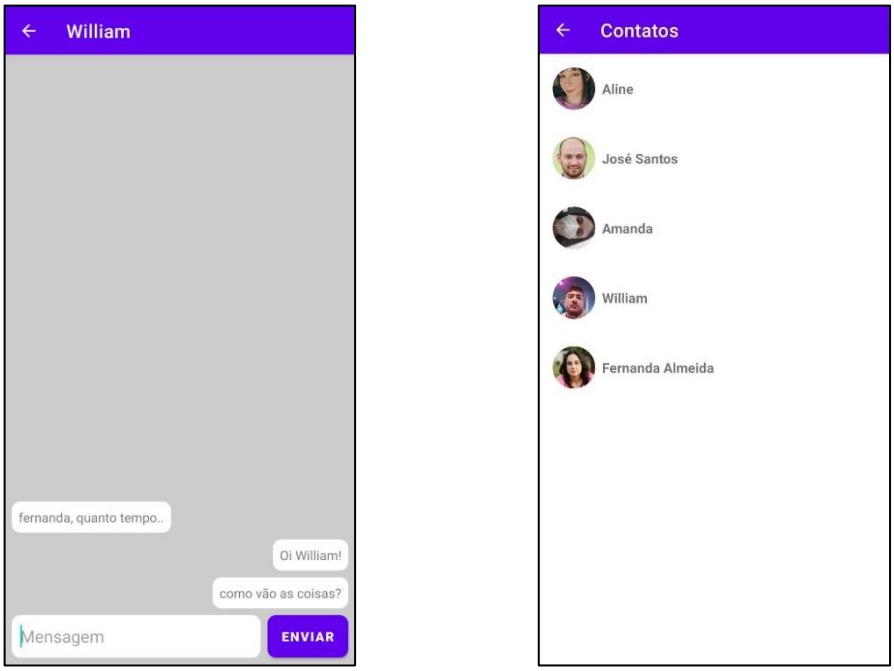


Fonte: autoria própria.

A partir da tela de conversa é possível seguir para três caminhos diferentes: tela de um *chat*, *logout* e tela de contatos. O botão de *logout* presente na tela de conversa está diretamente ligado ao Authentication do Firebase, assim como o *login*. Por meio dele, o usuário pode desconectar sua conta do aplicativo, e é redirecionado novamente para a tela de *login.*

Ao clicar em uma conversa, abre-se a tela de *chat*, demonstrada na Figura 14a, exibindo todas as mensagens trocadas entre os usuários e um campo para o envio de uma nova mensagem. Inicialmente, não será permitido a exclusão e edição de mensagens.

Figura 14 – Telas de *chat* e contatos



(a) (b)

Fonte: autoria própria.

A tela de contatos demonstrada na Figura 14b é visualizada quando o usuário toca no botão de Contatos da tela de conversa. Essa tela exibe uma lista com todos os contatos disponíveis para iniciar uma conversa que redirecionam para a tela de *chat* anteriormente citada.

4. Considerações Finais

O universo da tecnologia tem crescido gradual e rapidamente ao passo que a sociedade adota cada vez mais em seu cotidiano soluções que visam facilitar tarefas corriqueiras e até mesmo as mais complexas. Nesse contexto, há de um lado grande requerimento da população em geral por essas soluções, e de outro empresas e organizações buscando atender a essas demandas. Cria-se então, um cenário competitivo de constantes lançamentos tecnológicos, onde a velocidade e agilidade de desenvolvimento podem fazer plena diferença entre o que será uma aplicação de sucesso ou não.

Com base nas referências teóricas abordadas durante este trabalho, pode-se dizer então que a Computação em nuvem fornece ferramentas e serviços que visam solucionar alguns desses problemas de desenvolvimento. No âmbito do desenvolvimento *mobile*, o *Backend as a Service* dispõe de serviços que resultam em maior simplicidade e agilidade na criação de um *backend* para uma aplicação, visto que essa tarefa tende a ser trabalhosa e muitas vezes custosa. Com base nesse modelo de serviço, surgem ferramentas no mercado com o propósito de implementar tais tarefas.

Diante disso, o presente trabalho objetivou demonstrar as tecnologias empregadas nesse processo de desenvolvimento e apresentar um estudo de aplicação real desenvolvida através do serviço de *BaaS* chamado Firebase.

Por meio dos serviços ofertados pelo Firebase foi possível desenvolver uma aplicação *mobile* Android nativa com diversas funcionalidades através da nuvem. A praticidade e facilidade no desenvolvimento se fez notória ao término do sistema, o qual possui banco de dados, autenticação de usuários, armazenamento de arquivos, envio de mensagens dentro do *app* e links de redirecionamento. Todas as funcionalidades citadas puderam ser implementadas através do Firebase, utilizando os respectivos serviços: *Cloud Firestore, Authentication, Storage, InAppMessaging* e *Dynamic Links.*

As vantagens percebidas durante esse processo são, primordialmente, em relação a quantidade de tempo e recursos tecnológicos reduzidos. Dessa forma,

dispensa-se o uso de grandes estruturações de banco de dados e emprega-se mais tempo em outras funcionalidades da aplicação, como sua usabilidade por exemplo.

Propõe-se como trabalhos futuros a implementação de novas funcionalidades a aplicação. A utilização do *Cloud Messaging* em conjunto com o *Cloud Functions* para envio de notificações unitárias, e a possibilidade de criar conversas em grupos com mais de duas pessoas.

REFERÊNCIAS

IBGE, Uso de internet, televisão e celular no Brasil. Disponível em: < https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html > (Acessado em 20 de novembro de 2021).

NIST. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology. Special Publication 800-145. 2011. Disponível em: < http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf > (Acessado em 18 de novembro de 2021).

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications p.7–18, 2010.

PAIVA, J. C.; LEAL, J. P.; QUEIRÓS, R. A. P. de. Design and implementation of an ide for learning programming languages using a gamification service. Gamification-Based E-Learning Strategies for Computer Programming Education, IGI Global, p. 295–308, 2016.

LANE, K. Rise of Mobile Backend as a Service (MBaaS) API Stacks. 2012.

CAROLAN, J.; GAEDE, S. Introduction to cloud computing architecture. Microsystems Inc., 2009.

COSTA, I. d. O. Modelos par análise de disponibilidade em uma plataforma de mobile backend as a service. Universidade Federal de Pernambuco, 2015.

BATSCHINSKI, George. Backend as a Service: Prós e Contras. 2016.

BATSCHINSKI, George. O que é um Backend as a Service?. Disponível em: < https://blog.back4app.com/pt/o-que-e-backend-as-a-service/ > (Acessado em 24 de novembro de 2021).

SCHEMBERGER, Elder; FREITAS, Ivonei; VANI, Ramiro. Plataforma Android.UNIOESTE- Universidade Estadual do Oeste do Paraná – CascavelPR. UNIVEL- União Educacional de Cascavel – Cascavel- PR. 2009.

SIMÃO, A. M. de L. et. al. Aquisição de Evidências Digitais em Smartphones Android. In: Procedings of the Sixth International Conference on Forensic Computer Science Investigation ICoFCS 2011 (2011). Disponivel em: <

https://docplayer.com.br/2390320-Aquisicao-de-evidencias-digitais-em-smartphones-android.html > (Acessado em 28 de novembro de 2021).

ROUSE, M. Mobile Backend as a Services. 2015.

JAVA, O que é a tecnologia Java e porque preciso dela? < https://www.java.com/pt-BR/download/help/whatis_java.html > (Acessado em 24 de novembro de 2021).

DEITEL, Harvey. M; DEITEL, Paul J. Java: Como programar. 8. Ed. São Paulo: Pearson Prentice Hall, 2010.

Rodrigues, Wagner Braz. NoSQL: a análise da modelagem e consistência dos dados na era do Big Data. 2017. 70 f. Dissertação (Mestrado em Tecnologia da Inteligência e Design Digital) - Programa de Estudos Pós-Graduados em Tecnologia da Inteligência e Design Digital, Pontifícia Universidade Católica de São Paulo, São Paulo, 2017. Disponível em: < https://repositorio.pucsp.br/jspui/handle/handle/20590 > (Acessado em 20 de novembro de 2021).

Fowler, Adam. NoSQL Misconceptions. Disponível em: < https://www.dummies.com/programming/big-data/10-nosql-misconceptions/ > (Acessado em 20 de novembro de 2021).

Gonzalez-Aparicio, M. T., Younas, M., Tuya, J., and Casado, R. (2016). A new model for testing crud operations. In Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications.

Farias, V. A. E., Sousa, F. R. C., Maia, J. G. R., Gomes, J. P. P., and Machado, J. C. (2016). Machine learning approach for cloud nosql databases performance modeling. In Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Networking and Parallel/Distributed Computing (SNPD), Federal University of Ceara.

Corbellini, A., Mateos, C., Zunino, A., Godoy, D., and Schiaffino, S. Persisting big-data: The nosql landscape. Information Systems. 2016.