

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - UERN  
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT  
DEPARTAMENTO DE INFORMÁTICA – DI  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GIOVANA LORENA COSTA DE ANDRADE

**DESENVOLVIMENTO EM NUVEM: UM ESTUDO DE CASO UTILIZANDO O  
FIREBASE COMO SERVIDOR BACKEND**

MOSSORÓ - RN

2018

GIOVANA LORENA COSTA DE ANDRADE

**DESENVOLVIMENTO EM NUVEM: UM ESTUDO DE CASO UTILIZANDO O  
FIREBASE COMO SERVIDOR BACKEND**

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Rommel Wladimir de Lima.

MOSSORÓ - RN

2018

**Catálogo da Publicação na Fonte.**  
**Universidade do Estado do Rio Grande do Norte.**

A553d Andrade, Giovana Lorena Costa de  
Desenvolvimento em nuvem: um estudo de caso utilizando o Firebase como servidor backend. / Giovana Lorena Costa de Andrade. - Mossoró, 2018. 51p.

Orientador(a): Prof. Dr. Rommel Wladimir de Lima.  
Monografia (Graduação em Ciência da Computação).  
Universidade do Estado do Rio Grande do Norte.

1. Computação em nuvem. 2. Backend as a Service. 3. Firebase. I. Lima, Rommel Wladimir de. II. Universidade do Estado do Rio Grande do Norte. III. Título.


Giovana Lorena Costa De Andrade

DESENVOLVIMENTO EM NUVEM: UM ESTUDO DE CASO UTILIZANDO O FIREBASE COMO SERVIDOR BACKEND

Monografia apresentada como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

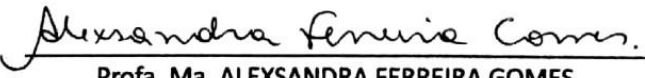
Aprovada em: 18/05/2018

Banca Examinadora




---

Prof. Dr. ROMMEL WLADIMIR DE LIMA  
Universidade do Estado do Rio Grande do Norte - UERN



---

Profa. Ma. ALEXSANDRA FERREIRA GOMES  
Universidade do Estado do Rio Grande do Norte - UERN



---

Profa. Dra. CÍCLIA RAQUEL MAIA LEITE  
Universidade do Estado do Rio Grande do Norte - UERN

*Aos meus pais.*

# Agradecimentos

Primeiramente, agradeço aos meus pais, por todo suporte, conselhos e confiança depositados em mim. E também a toda minha família que, a sua maneira, me apoia e alegra em todos os momentos.

Também gostaria de agradecer aos Professores do DI, em especial a André Pedro, Alysson Mendes, Ceres Germanna e Marcelino Pereira, por todo auxílio e conhecimento transmitido ao longo destes anos. Também as Professoras Alexandra Ferreira e Cícilia Maia por participarem da banca avaliadora e por todas as contribuições. Também agradeço ao meu orientador, o Professor Rommel Lima, por todo esforço dedicado a este trabalho.

Por último aos amigos que ganhei neste percurso, em especial aos do PETCC, que proporcionaram as melhores experiências.

*“Pode-se encontrar a felicidade mesmo nas  
horas mais sombrias, se a pessoa se lembrar  
de acender a luz.”*

(Harry Potter e o Prisioneiro de Azkaban)

# Resumo

Cada vez mais cresce a demanda de aplicações e sistemas *web*, com intuito de suprir as necessidades geradas pelas mais variadas áreas de aplicação. Similares a este crescimento de mercado estão as aplicações móveis, que constantemente apresentam-se indispensáveis. No entanto, também devido a esta demanda, estes sistemas acarretam na complexidade de desenvolvimento, além de ser necessário assegurar certos atributos que tornam-os agradáveis ao usuário. Frente a isto, a Computação em nuvem traz ferramentas para auxiliar na implementação, dentre elas está o *Backend as a Service*, tecnologia que permite ao desenvolvedor abstrair a complexidade do *backend* das aplicações. Este trabalho apresenta o *Backend as a Service* como opção para o desenvolvimento destas aplicações. Para isso, é realizado um estudo de caso expondo os serviços fornecidos pelo Firebase e demonstrando a utilização deste. Em seguida, é apresentada a aplicação resultante da implementação que utiliza o Firebase como recurso, demonstrando assim, a facilidade no desenvolvimento de aplicações utilizando um BaaS como provedor de serviços *backend*.

**Palavras-chave:** Computação em nuvem, *Backend as a Service*, Firebase.



# Abstract

Increasingly, the demand for web applications and systems grows, in order to meet the needs generated by the most varied application areas. Similar to this market growth there are the mobile applications, which are consistently indispensable. However, due to this demand, these systems also have the complexity of development, and it is necessary to ensure certain attributes that make them pleasant to the user. In this context, cloud computing brings tools to help with implementation, among them is Backend as a Service, a technology that allows the developer to abstract the complexity of the backend applications. This work presents the Backend as a Service as an option for the development of these applications. For this, a case study is performed exposing the services provided by Firebase and demonstrating its use. Then, the application resulting from the deployment using Firebase as a resource is presented, thus demonstrating the ease way in developing applications using a BaaS as a backend service provider.

**Keywords:** Cloud computing, Backend as a Service, Firebase.

# Lista de ilustrações

Figura 1 – Definição de Computação em nuvem segundo NIST. . . . .	19
Figura 2 – Estrutura de comunicação dos serviços Firebase. . . . .	31
Figura 3 – Diagrama de classe da aplicação. . . . .	32
Figura 4 – Tecnologias utilizadas para implementação. . . . .	33
Figura 5 – Painel de gerenciamento do <i>Hosting</i> . . . . .	41
Figura 6 – Página inicial da aplicação. . . . .	41
Figura 7 – Menu da aplicação móvel. . . . .	42
Figura 8 – Todas as notícias da aplicação. . . . .	42
Figura 9 – Notícia selecionada visão <i>web</i> . . . . .	43
Figura 10 – Notícia selecionada visão <i>mobile</i> . . . . .	43
Figura 11 – Publicação de notícia no sistema. . . . .	44
Figura 12 – Área de gerenciamento de avisos no sistema. . . . .	44
Figura 13 – Área de gerenciamento do perfil administrador. . . . .	45
Figura 14 – Registro de eventos do sistema. . . . .	46

# Lista de códigos

Código 1	– Código de inicialização do Back4App. . . . .	23
Código 2	– Exemplo de <i>snippet</i> de código. . . . .	25
Código 3	– Exemplificação da estrutura principal da aplicação. . . . .	34
Código 4	– Código de acesso a todas as notícias. . . . .	35
Código 5	– Código de acesso a notícia específica. . . . .	35
Código 6	– Método de login através de provedores. . . . .	37
Código 7	– Método de login através de <i>e-mail</i> e senha. . . . .	37
Código 8	– Referência do <i>Storage</i> . . . . .	39
Código 9	– Regras do <i>Storage</i> . . . . .	39

# Lista de tabelas

Tabela 1 – Recursos disponibilizados com pacotes gratuitos. . . . .	27
Tabela 2 – Referências do AngularFire. . . . .	36
Tabela 3 – Métodos do <i>\$firebaseAuth</i> . . . . .	38
Tabela 4 – Referências do AngularFire para <i>Storage</i> . . . . .	39

# Lista de abreviaturas e siglas

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
BaaS	Backend as a Service
CLI	Command Line Interface
CRUD	Create, Read, Update, Delete.
CSS	Cascading Style Sheets
DaaS	Database as a Service
DOM	Document Object Model
DSaaS	Data Storage as a Service
FCM	Firebase Cloud Messaging
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
IoT	Internet of Things
JSON	JavaScript Object Notation
MBaaS	Mobile Backend as a Service
NIST	National Institute of Standards and Technology
NOSQL	Not Only SQL
PaaS	Platform as a Service
SaaS	Software as a Service
SDK	Software Development Kit
SLA	Service Level Agreement
SPA	Single Page Application
SQL	Structured Query Language

SSL	Secure Socket layer
WWW	World Wide Web
XaaS	Anything as a Service
XML	eXtensible Markup Language

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>18</b>
<b>2.1</b>	<b>Computação em nuvem</b>	<b>18</b>
2.1.1	Backend as a Service	22
2.1.2	Provedores BaaS	23
2.1.2.1	Back4App	23
2.1.2.2	Backendless	23
2.1.2.3	Cloudboost	24
<b>2.2</b>	<b>Firebase</b>	<b>24</b>
<b>2.3</b>	<b>Tecnologias utilizadas</b>	<b>28</b>
2.3.1	NoSQL	28
2.3.2	JSON	28
2.3.3	Angular	29
2.3.4	Apache Cordova	29
2.3.5	Ionic	30
<b>3</b>	<b>ESTUDO DE CASO</b>	<b>31</b>
<b>3.1</b>	<b>Serviços BaaS utilizados</b>	<b>34</b>
3.1.1	Database	34
3.1.2	Authentication	37
3.1.3	Storage	38
3.1.4	Hosting	40
<b>3.2</b>	<b>Aplicação desenvolvida - BlogDI</b>	<b>40</b>
<b>4</b>	<b>CONSIDERAÇÕES FINAIS</b>	<b>47</b>
	<b>REFERÊNCIAS</b>	<b>49</b>

# 1 Introdução

No princípio da *World Wide Web* (WWW) os sites eram compostos por um simples conjunto de arquivos hipertexto linkados apresentando informações textuais e gráficos limitados. Ao longo do tempo, com a progressão das linguagens e surgimento de novas ferramentas, tornou-se possível fornecer capacidade computacional juntamente com as informações, para os assim definidos sistemas e aplicações *web*. Atualmente, estas aplicações evoluíram para sofisticadas ferramentas que, além de oferecer funções especializadas, também são integradas aos banco de dados corporativos e às aplicações de negócio (PRESSMAN; MAXIM, 2016).

Similar ao crescimento dos abundantes serviços e funcionalidades fornecidos pelas aplicações *web*, está a prolífera demanda dos serviços móveis. Segundo Luan et al. (2015), esses dispositivos oferecem facilidade de acesso à informação, independente da localização do usuário, fornecendo diversas aplicações de computação móvel.

Cada vez mais a computação móvel ganha mais espaço, aumentando a demanda das aplicações móveis (PANDEY; NEPAL, 2012). Com o notável progresso e avanço desta tecnologia, surge uma tendência de transferência da base de dados para centros de dados distribuídos, com o intuito de prover serviços através das aplicações móveis (COSTA, 2015). Mesmo com a maneira despreziosa de descoberta de conteúdo da *web*, a qual não necessita nenhuma instalação, ainda há uma preferência de público para com os aplicativos que gera, além de uma fidelização, uma experiência mais integrada à plataforma (LOPES, 2016).

Então, gradualmente, percebe-se a necessidade do desenvolvimento dessas aplicações, tanto *web* quanto *mobile*, para suprir a evidente demanda de sistemas que abrangem as mais diversas áreas. Essas tecnologias possuem como objetivo automatizar a vida das pessoas, facilitando cada vez mais os processos que as circundam. Porém, ao mesmo tempo, esse processo acarreta dificuldades de desenvolvimento de aplicações, geradas pela complexidade das mesmas.

Para seu completo desenvolvimento, os sistemas precisam garantir alguns serviços, que os caracterizam diferenciando-os dos sistemas de computador convencionais. Segundo Pressman e Lowe (2009), determinados atributos são encontrados na maioria dos sistemas, dentre eles têm-se:

- Intensidade da rede: a aplicação reside numa rede e deve atender às necessidades do cliente;
- Concorrência: um grande número de usuários pode acessar a aplicação de uma só



vez;

- Desempenho: razoável tempo de acesso para o processamento;
- Disponibilidade;
- Evolução contínua.

Para assegurar estes atributos é necessário uma infraestrutura com servidores, formas de armazenamento, estrutura de conectividade, equipes de gerenciamento e disponibilidade para ampliar recursos. Todos esses serviços compõe a infraestrutura necessária para o funcionamento do tratamento e processamento dos dados, o *backend*. Além disso, cabe ao desenvolvedor *backend* a implementação da aplicação e das estruturas necessárias à ela, como *Application Program Interface* (API).

Como auxílio a construção desses sistemas, o mercado de arquitetura para aplicações apresenta, constantemente, novos *frameworks* e padrões de desenvolvimento. Em destaque a Computação em nuvem que, segundo Chen et al. (2010 apud LI; LI; SHIH, 2014), “cria uma nova maneira de projetar, desenvolver, testar, implementar, executar e manter aplicações na Internet”. Apresentando-se como uma tecnologia que traz benefícios aos usuários otimizando recursos e reduzindo custos.

Na Computação em nuvem, muitos utilizam os serviços de *software* como base comercial. Os provedores destes serviços torna-os acessíveis ao usuário. A infraestrutura de computação necessária para hospedar os serviços são oferecidas “como um serviço” por fornecedores. Os modelos de serviços mais comumente apresentados são: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS) (VAQUERO et al., 2008).

Entretanto, mesmo sendo mais habitualmente citadas, estes modelos de serviços não são os únicos. Dentre eles está o serviço para *backend*, *Backend as a Service* (BaaS), que provê serviços para aplicação sem necessidade de programação do lado do servidor. Serviços esses que provêm armazenamento de dados e outras funcionalidades para sistemas, mostrando-se tendência nas aplicações atuais por serem mais flexíveis (SCHNEIDER, 2016). Esta tecnologia pode ser vista como uma ponte conectando o *backend* e *frontend*, auxiliando no desenvolvimento de aplicações *web* e *mobile*, acelerando seu processo e simplificando a criação de APIs. O BaaS tem-se mostrado promissor no mercado de desenvolvimento *web*, o qual apresentará uma taxa de crescimento de 80% ao ano, atingindo 28 bilhões de dólares em 2020 (MARKETSANDMARKETS, 2016).

O presente trabalho apresenta plataformas *Backend as a Service*, realçando seus serviços disponíveis para desenvolvedores, métodos de integração do BaaS com as aplicações e os planos de utilização existentes. Tendo como objetivo a utilização do Firebase, uma ferramenta BaaS, para o desenvolvimento alternativo de aplicações *web* e *mobile*,

exemplificando assim, a interoperabilidade entre sistemas que utilizam um BaaS como recurso. Além de apresentar, por meio desse, um estudo de caso que visa demonstrar a utilização dos recursos e serviços disponibilizados pelo BaaS em questão. O estudo de caso descrito, segue a seguinte estrutura de apresentação:

- I. Descrever o sistema desenvolvido;
- II. Exibir as tecnologias usadas para a implementação do sistema nas plataformas *web* e *mobile*;
- III. Demonstrar os serviços e recursos utilizados do Firebase.

A organização do presente documento dá-se da seguinte forma: no Capítulo 2, são apresentados os conceitos fundamentais e tecnologias empregadas para melhor compreensão deste trabalho. No Capítulo 3, é apresentado o estudo de caso descrevendo o desenvolvimento da aplicação, suas funcionalidades e demonstrando a utilização dos serviços utilizados do Firebase. E no Capítulo 4, são apresentadas as considerações finais.

## 2 Referencial teórico

### 2.1 Computação em nuvem

A nuvem é uma tecnologia ou grupo de serviços, que originalmente foi pensada para que houvesse uma combinação entre recursos e atividades. O que acontecia dentro da nuvem era desconhecido aos usuários, porém, dos mesmos, percebeu-se uma parcela que demonstrava interesse no que acontecia dentro da nuvem. Como todas as funcionalidades disponíveis, a nuvem e os serviços mudaram ao longo do tempo, para adaptar-se a necessidade dos clientes (ROUNTREE; CASTRILLO, 2013)

De acordo com Taurion (2009 apud PINELI; DUARTE, 2013):

O termo “nuvem” e sua representação gráfica já é utilizada a muito tempo em diagramas para representar a internet em uma determinada situação, conectando um servidor ou dispositivos dando a entender que o fluxo de dados trafega pela internet. Na computação em nuvem o desenho da nuvem muda de sentido e deixa de ser intangível e passa a representar poder de processamento computacional, local onde aplicações podem estar sendo executadas, tornando o centro de processamento e não apenas um caminho (TAURION, 2009 apud PINELI; DUARTE, 2013).

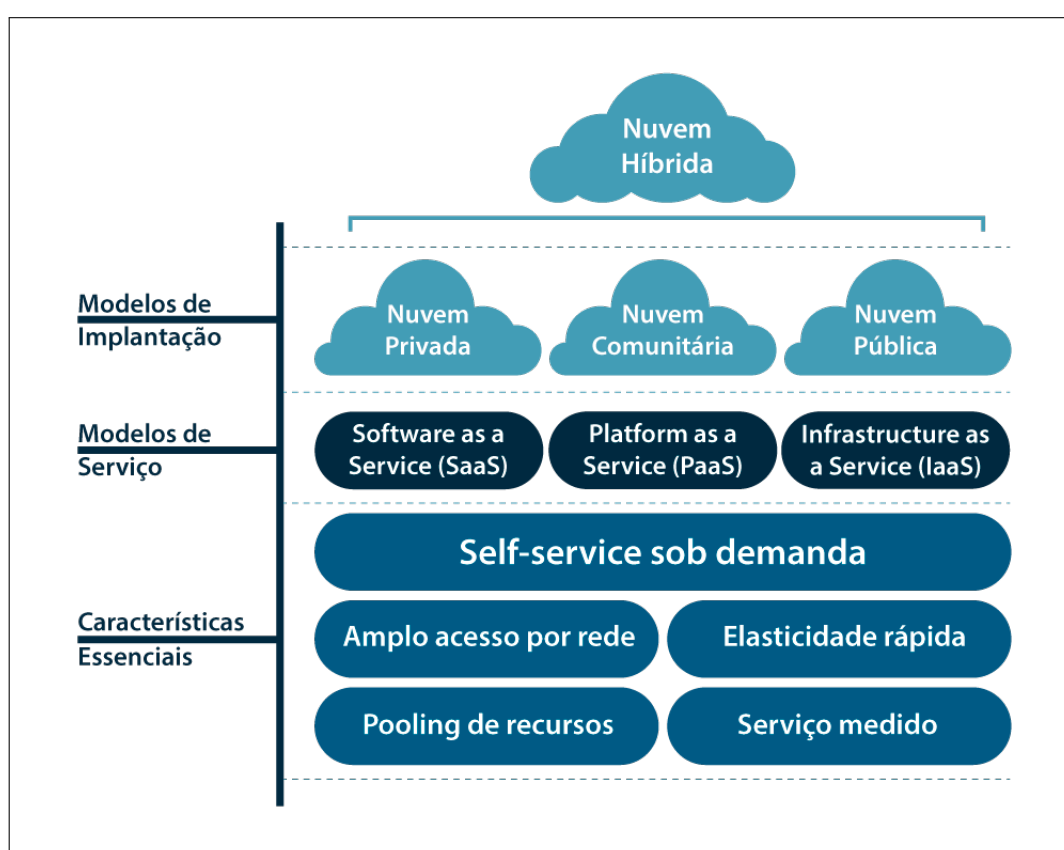
A computação em nuvem é um modelo utilizado para habilitar o acesso ubíquo, conveniente e sob-demanda de recursos computacionais compartilhados (redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente fornecido e liberado com o mínimo de esforço gerencial ou interação do provedor de serviço (MELL; GRANCE et al., 2011). Tem-se como um dos maiores desafios a definição de computação nas nuvens, como muitos outros termos na indústria tecnológica, todos têm sua própria definição. Dentre elas têm-se a de Veras (2012):

CLOUD COMPUTING é substituir ativos de TI que precisam ser gerenciados internamente por funcionalidades e serviços do tipo pague-conforme-crescer a preços de mercado. Estas funcionalidades e serviços são desenvolvidos utilizando novas tecnologias como a VIRTUALIZAÇÃO, arquiteturas de aplicação e infraestrutura orientadas a serviço e tecnologias e protocolos baseados na Internet como meio de reduzir os custos de hardware e software usados para processamento, armazenamento e rede (VERAS, 2012).

O *National Institute of Standards and Technology* (NIST), em português Instituto Nacional de Padrões e Tecnologia, publicou em Setembro de 2011 sua definição de computação em nuvem. De acordo com o NIST, o modelo de computação em nuvem é formado por cinco características essenciais, quatro modelos de implantação e três modelos de serviços. Como exemplificados na Figura 1, essas características são formadas por:

- *Self-service* sob demanda;
- *Pooling* de recursos;
- Amplo acesso por rede;
- Serviço medido;
- Elasticidade rápida.

Figura 1 – Definição de Computação em nuvem segundo NIST.



Fonte – Adaptado de (WILLIAMS, 2010).

“As características essenciais são as vantagens que as soluções de computação em nuvem vêm a oferecer. Em conjunto, algumas dessas características definem a computação em nuvem e faz a distinção com outros paradigmas” (RUSCHEL; ZANOTTO; MOTA, 2010). No *Self-service* sob demanda, os usuários podem acessar recursos computacionais sob demanda, ou seja, sem a necessidade de interação humana com cada provedor de serviço. O ambiente na nuvem pode ser automaticamente reconfigurado e suas mudanças são apresentadas de forma transparente ao usuário, podendo este, personalizar o mesmo. Os recursos computacionais do provedor servem múltiplos usuários, devido a sua organização

em *pool*, os quais são acessados de acordo a demanda. Para o usuário, não é necessário o conhecimento da localização física do recurso, somente especificar a mesma em um nível mais alto de abstração (SOUSA; MOREIRA; MACHADO, 2009).

Em relação ao amplo acesso à rede, os recursos devem estar disponíveis através da rede e acessados através de mecanismos padrões que permitam a utilização dos mesmos por plataformas heterogêneas. Os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos alavancando a capacidade de medição em um nível de abstração apropriado para o tipo de serviço. A utilização de recursos pode ser monitorada, controlada e reportada, proporcionando transparência tanto para o provedor como para o usuário do serviço utilizado (MELL; GRANCE et al., 2011). A computação em nuvem provê a ilusão de infinitos recursos disponíveis sob demanda. Portanto, usuários esperam que a nuvem forneça o recurso rapidamente em qualquer quantidade a qualquer momento. Sendo assim, Gong et al. (2010 apud COSTA, 2015) define a elasticidade como “a capacidade do ambiente computacional da nuvem aumentar ou diminuir de forma automática os recursos computacionais demandados e provisionados para cada usuário”.

A utilização de determinado modelo de implantação depende da necessidade das aplicações que serão implementadas, como acesso livre ou restrição do mesmo a um recurso em particular. Como exemplo, limitar o acesso de determinado usuário a uma funcionalidade ou informação específica. Segundo o NIST, os modelos de implantação da computação em nuvem podem ser divididos em: privado, público, comunidade e híbrido.

Às nuvens privadas são projetadas para uso exclusivo de uma única organização, podendo ser criada e gerenciada por provedores externos ou pela própria organização. Geralmente a última opção é adotada quando há a necessidade de alto grau de restrição de dados, não podendo confiar à terceiros. Ela oferece o maior grau de controle sobre desempenho, confiabilidade e segurança. No entanto, muitas vezes são criticadas quando comparadas a nuvem pública, pois necessitam de um investimento inicial (ZHANG; CHENG; BOUTABA, 2010).

Os serviços da nuvem pública são fornecidos sem premissas por provedores terceirizados para o público em geral e os recursos são compartilhados com outros clientes do provedor (WILLIAMS, 2010).

Este tipo de nuvem reflete melhor os benefícios originais propostos pela computação em nuvem comparado às nuvens privadas. Diferente de uma nuvem privada, o investimento inicial de infraestrutura é zero, pois não é necessário adquirir servidores e outros equipamentos próprios. Nuvens públicas geralmente são oferecidas por grandes data centers que possuem capacidade computacional elevada ao de uma nuvem privada. A nuvem pública tem como vantagem delegar, para o provedor da nuvem, a responsabilidade pelo gerenciamento da infraestrutura e por manter o *Service Level Agreement* (SLA) (CAROLAN et al., 2009 apud COSTA, 2015).

A nuvem comunitária é fornecida para uso exclusivo por um grupo/comunidade de consumidores de organizações que compartilham interesses. Podendo ser própria, gerenciável e operada por uma ou mais organizações da comunidade, terceiros ou uma combinação destes (MELL; GRANCE et al., 2011). A infraestrutura da nuvem híbrida é composta pela combinação de duas ou mais nuvens, podendo ser privadas, comunidade ou públicas com o intuito de proporcionar o melhor dos modelos escolhidos, superando suas limitações. Mesmo com este arranjo de nuvens elas devem permanecer como entidades únicas, ligadas por uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações (SOUSA; MOREIRA; MACHADO, 2009).

Diferentes tipos de computação em nuvem são fornecidos como um serviço aos consumidores, recursos computacionais são alugados e nenhum recurso de *hardware* ou *software* é comprado diretamente pelo consumidor (WILLIAMS, 2010). Estes serviços se enquadram em uma ou mais das seguintes modalidades:

- *Software as a Service* (SaaS): É fornecido um conjunto de *software*, já hospedado, no qual não é necessário desenvolvimento ou programação somente a configuração. Assim, segundo Dillon, Wu e Chang (2010 apud COSTA, 2015), “aplicações podem ser liberadas em um ambiente de hospedagem para serem acessadas por diferentes clientes”.
- *Platform as a Service* (PaaS): Neste modelo é fornecido uma plataforma de software de alto nível, com dimensionamento transparente dos recursos, realizando uma abstração das *Virtual Machines* e seus componentes (COSTA, 2015).
- *Infrastructure as a Service* (IaaS): É uma evolução da tradicional hospedagem. É a entrega de *hardware* (servidor, armazenamento e rede) associado a um *software* como um serviço. Utiliza a virtualização para particionar a infraestrutura entre os usuários, os quais pagam pelos recursos que utilizarem de acordo com a capacidade das *Virtual Machines* alocadas (BHARDWAJ; JAIN; JAIN, 2010).

O NIST apresenta três modalidades básicas de serviços, no entanto, ainda existem alguns outros modelos de serviços, genericamente denominados de XaaS (*Anything as a Service*). A taxonomia das nuvens inclui diferentes participantes envolvidos, os quais, dependendo da necessidade, acoplam atributos e tecnologias para fornecer diferentes tipos de serviços “XaaS”, no qual o “X” é *software*, *hardware*, plataforma, infraestrutura, dados, negócios, etc (RIMAL; CHOI; LUMB, 2009).

Como exemplos destes outros tipos de serviços têm-se: *Database as a service* (DaaS), *Hardware as a Service* (HaaS), *Data Storage as a Service* (DSaaS) e *Backend as a Service* (BaaS). Também conhecido como *Mobile Backend as a Service* (MBaaS) pode ser definido, segundo COSTA (2015), como “uma categoria de computação em nuvem que é composta

por empresas que proporcionam facilidades aos desenvolvedores, tais como: instalação, utilização e operação de uma infraestrutura em nuvem”.

Na próxima seção será apresentado o conceito de BaaS, quanto a tecnologia, e também as características que abrangem este serviço.

### 2.1.1 Backend as a Service

*Backend as a Service* é um modelo de serviço de computação em nuvem que atua como um componente de *middleware*, permitindo, aos desenvolvedores, conectar seus aplicativos a serviços em nuvem através de *Software Development Kit* (SDK) e *Application Programming Interface* (API) (PAIVA; LEAL; QUEIRÓS, 2016). Ou seja, ele irá prover uma conexão a um *backend* fornecendo recursos como armazenamento e processamento, além de serviços de notificação e integração com rede social. O BaaS abstrai às complexidades de implantação e gerenciamento de sua própria infraestrutura servindo como ponte para oferecer recursos significativos e necessários para desenvolvedores, facilitando assim, a construção de aplicações.

Inicialmente, o BaaS era restrito para desenvolvimento de aplicativos móveis, porém o avanço desta tecnologia permitiu, também, que sua abordagem abrangesse facilmente várias áreas de desenvolvimento. E que, segundo Lane (2015), podem ser:

- Desenvolvimento *web* - Provê uma maior flexibilidade para o desenvolvimento de aplicações *web*. Fornecendo uma pilha significativa de recursos para o desenvolvimento variado de aplicações *web*.
- Aplicativos móveis - Seu foco consistia na otimização de dados para dispositivos móveis, reunindo recursos essenciais e elementos críticos de aplicativos móveis.
- *Readers* - Publicação para Kindle e outros dispositivos que são destinados a dar suporte a estas publicações características.
- *Launch APIs* - O BaaS fornece uma maneira rápida de lançar APIs e outras infraestruturas ao redor de dados e recursos tornando-o disponível para desenvolvedores *web* e *mobile*.

Para fornecer suporte a esta tecnologia, há empresas provedoras deste serviço, as quais constantemente atualizam e disponibilizam mais recursos e funcionalidades aos desenvolvedores, tais como Amazon e Google. Na próxima seção serão apresentados alguns destes provedores.

## 2.1.2 Provedores BaaS

Esta seção é destinada a apresentação de provedores *Backend as a Service* existentes no mercado. Será mostrado seus conceitos, serviços e descrição de como é realizado o gerenciamento do projeto desenvolvido por meio das plataformas.

### 2.1.2.1 Back4App

É uma plataforma BaaS que foi construída com base no código aberto Parse, o qual pode-se criar e hospedar APIs para aplicações *web*, *mobile* e *Internet of Things* (IoT). O Parse é uma suíte de código aberto que otimiza o desenvolvimento do *backend*, podendo trabalhar com uma velocidade incrementada em 80%. Por meio desta plataforma, é possibilitado ao desenvolvedor criar APIs, que habilitam a troca de dados entre duas aplicações, sem necessidade de infraestruturas complexas que acompanham o desenvolvimento de sistemas (BACK4APP, 2018).

A interação com a plataforma é realizada por meio de um *Command Line Interface* (CLI), em português Interface de Linha de Comando. Por meio desta, o cliente desenvolvedor faz uso de comandos para o programa na forma de sucessiva linhas de texto (linhas de comando). Sendo assim, através do CLI são executadas ações que vão desde iniciar novas aplicações a utilizar os serviços disponibilizados por este BaaS.

O Back4App permite a construção de aplicações *mobile*, com Android e IOS, e também *web* por meio do JavaScript. Após a instalação do SDK do Back4app e da inserção do código de inicialização, representado no Código 1, na aplicação são disponibilizados alguns serviços da plataforma aos desenvolvedores, como: *Database*, *User Registration*, *Files* e *Hosting*.

Código 1 – Código de inicialização do Back4App.

---

```
<script src="https://unpkg.com/parse/dist/parse.min.js"></script>      1
                                                                    2
<script>                                                            3
  Parse.initialize("APP_ID", "YOUR_JS_KEY");                        4
  Parse.serverURL = "https://parseapi.back4app.com/";              5
</script>                                                            6
```

---

Fonte – Adaptado do (BACK4APP, 2018).

### 2.1.2.2 Backendless

O Backendless é uma plataforma BaaS fornecedora de serviços API integrados e de uso geral, como também fornece suporte a implementação desses serviços e de funções personalizadas. É disponibilizado ao usuário tarefas e operações comuns a desenvolvedores *backend* dentre elas suporte para autenticação de usuário, persistência de dados e armaze-



namento de arquivos, permitindo a criação de aplicações *mobile* e *web* (BACKENDLESS, 2018).

Para configurar o ambiente de desenvolvimento do Backendless é realizado a inserção das dependências da ferramenta, biblioteca do Backendless, juntamente com as credenciais da aplicação, criada na área de gerenciamento disponibilizada pela plataforma, que são formadas pelo identificador da aplicação e a chave da API.

Mesmo sendo uma plataforma paga, o Backendless proporciona um pacote gratuito para desenvolvedores, estes, porém, encontram-se com algumas limitações não só de acesso a serviços, como também a alguns recursos como por exemplo: restrição ao espaço de armazenamento que é alocado para a aplicação, tabelas de banco de dados criadas e objetos armazenados nas tabelas.

### 2.1.2.3 Cloudboost

O Cloudboost é um provedor de BaaS híbrido que encarrega-se de toda a infraestrutura para que assim o cliente concentre-se apenas com a aplicação para fornecer melhores experiências do usuário. A ferramenta lida com todo o *backend*, provendo serviços de armazenamento, autenticação de usuário, pesquisa, entre outros (CLOUDBOOST, 2018).

A comunicação com a plataforma, ao contrário do Back4App, não utiliza um CLI e sim é disponível ao usuário uma área gerenciável. Sendo assim, não é necessário configuração. Mas para realizar a interação é preciso, primeiramente, fazer a chamada do SDK do Cloudboost no código e inserir as credenciais da aplicação.

O Cloudboost é uma plataforma paga, a qual possui pacotes para cada tipo de usuário, desde o *pro*, designado às *startups*, até o denominado *enterprise*, o qual fornece recursos avançados para grandes empresas.

As plataformas apresentadas aqui demonstram, basicamente, seu funcionamento e características, pois mesmo sendo todas BaaS ainda apresentam peculiaridades próprias dentre recursos, funcionamento e gerenciamento. No entanto, ainda existe uma plataforma que será exposta na seção 2.2, que é a ferramenta de estudo deste trabalho.

## 2.2 Firebase

O Firebase é uma plataforma *Backend as a Service* criada sobre a infraestrutura do Google com o objetivo de auxiliar os desenvolvedores a acelerar o desenvolvimento de aplicações, sem a necessidade de gerenciar a infraestrutura. Além de serviços geralmente disponibilizados por um BaaS como autenticação, banco de dados e *hosting*, a ferramenta também fornece produtos que operam em conjunto compartilhando dados e *insights* entre si. Além dos comumente mencionados aplicações *web* e *mobile*, o Firebase também provê

suporte para o desenvolvimento de aplicativos de jogos, por meio de SDKs tanto para C++ quanto para Unity (FIREBASE, 2018).

O gerenciamento dos projetos é realizado por meio do CLI do Firebase, o qual oferece uma série de ferramentas para visualização e implantação dos mesmos. Por meio dele ocorre a conexão da máquina local com a conta do Firebase e, conseqüentemente, aos projetos vinculados a mesma. A adição do Firebase a aplicação é realizada por meio de Firebase SDK e de um *snippet* de código, Código 2, de inicialização, o qual possui informações de identificação do projeto Firebase. A partir disso, pode-se então, acessar os serviços do FIREBASE (2018), são eles:

Código 2 – Exemplo de *snippet* de código.

---

```
<script> 1
// Initialize Firebase 2
var config = { 3
  apiKey: "AIzaSyDSzblbhpY8xb_kCmMcZggGa5yDDAqnB60", 4
  authDomain: "<nome-da-aplicacao>.firebaseapp.com", 5
  databaseURL: "https://<nome-da-aplicacao>.firebaseio.com", 6
  projectId: "<nome-da-aplicacao>", 7
  storageBucket: "<nome-da-aplicacao>.appspot.com", 8
  messagingSenderId: "617895230918" 9
}; 10
firebase.initializeApp(config); 11
</script> 12
```

---

Fonte – Autoria própria.

- *Authentication*: serviço para autenticar usuários na aplicação. Este serviço fornece suporte para autenticação por meio de senhas, números de telefone e provedores de identidades federadas, como: Google, Facebook, Twitter entre outros. Este serviço apresenta um meio otimizado para realização de *logins* e cadastro na aplicação. O funcionamento dá-se pela utilização de credenciais, que são fornecidas no momento do *login* do usuário. Em seguida, às credenciais são transmitidas ao SDK do Firebase *Authentication*, então os serviços de *backend* verificam e enviam uma resposta ao cliente.
- *Storage*: é um serviço para armazenamento de objetos, criado para veicular conteúdo gerado pelo usuário. Os arquivos são armazenados em um repositório do Google *Cloud Storage* e são acessados através do SDK do Firebase que permite realizar *upload* e *download* dos arquivos, utilizando a segurança do Google.
- *Database*: é um banco de dados de tempo real hospedado na nuvem. Cada aplicação criada que utiliza os SDKs, independente da plataforma, recebe automaticamente

atualizações com os dados mais recentes, já que as aplicações compartilham uma instância do banco de dados. Conta com recursos como:

- Tempo real;
  - *Off-line* (Somente aplicativos móveis);
  - Acessível em dispositivos clientes;
  - Escalonamento entre vários bancos de dados.
- *Hosting*: serviço disponibilizado para hospedar arquivos HTML, CSS e JavaScript do *site*. O Firebase *Hosting* conta com infraestrutura, recursos e ferramentas adaptadas a implantação e o gerenciamento de *websites* estáticos. Possui como principais recursos:
    - Serviço por uma conexão segura;
    - Entrega rápida de conteúdo;
    - Implantação rápida;
    - *Rollbacks* de um clique.

O Firebase ainda dispõe de bibliotecas para auxiliar no desenvolvimento de aplicações. Essas tecnologias variam em sua forma de codificação e, também, em sua linguagem foco, pois cada uma foi originada para trabalhar colaborativamente com determinada linguagem e conseqüentemente determinado tipo de aplicação. São elas:

- Suporte *web*: são bibliotecas que contribuem tanto para *frameworks* quanto para outras bibliotecas JavaScript.
  - AngularFire;
  - ReactFire;
  - EmberFire;
  - BackboneFire;
  - VueFire;
  - PolymerFire.
- Suporte *mobile*.
  - GeoFire;
  - FirebaseUI;
  - FirebaseJobDispatcher;
  - Firebase-util.

Dentre as citadas acima, destaca-se a biblioteca AngularFire, a qual conecta o Firebase e o *framework* JavaScript Angular, apresentado na subseção 2.3.3. Essa conexão permite associar às referências do Firebase com o Angular, para que sejam transparentes e mantidos em sincronia com o Firebase *Database* e com todos os clientes que utilizam a aplicação. Seu foco é abstrair ao máximo o processo de conexão entre os dois, além de facilitar a utilização dos serviços fornecidos pelo BaaS (ANGULARFIRE, 2018).

Ainda inclui pacotes que podem ser escolhidos de acordo com a aplicação desenvolvida como:

- *Spark*: Plano gratuito designado aos desenvolvedores amadores. Possui alguns limites na habilitação de alguns recursos, indicado para aplicações de pequeno porte.
- *Flame*: Plano pago, porém seu limite é maior do que o Spark, indicado para aplicativos de médio porte em expansão.
- *Blaze*: Plano pago e seu preço é calculado por escala, também chamado de “*pay as you go*” pagamento por utilização.

Tabela 1 – Recursos disponibilizados com pacotes gratuitos.

PLATAFORMA	DATABASE	STORAGE	DISPONIBILIDADE
Firestore	1 GB	5 GB	ilimitado
Backendless	5 tabelas com 1000 objetos/-tabela	1 GB	ilimitado
Back4App	0,5 GB	5 GB	ilimitado
Cloudboost	50,000 registros	1 GB	30 dias

Fonte – Autoria própria.

A Tabela 1 representa uma comparação dos serviços e recursos disponibilizados, no plano gratuito, pelas plataformas apresentadas na subseção 2.1.2 e o Firebase. Com relação ao serviço de banco de dados (*Database*), o Firebase disponibiliza o dobro de armazenamento que o Back4App, enquanto que o Backendless e o Cloudboost limitam tanto a quantidade de tabelas inseridas na aplicação quanto os registros e objetivos contidos nas mesmas. O que pode ser um pouco prejudicial dependendo da estrutura idealizada do banco de dados. Já o *Storage* o único que equipara-se com o Firebase é o Back4App, os demais concedem apenas 1 Gigabyte em comparação aos 5 fornecidos pelo Firebase. A disponibilidade de acesso a esses recursos, a partir da conta de usuário, são permitidos de forma ilimitada com exceção do Cloudboost que fornece apenas por 30 dias.

Os serviços, produtos e dados, apresentados nessa tabela, serviram como critério para seleção, às quais exibiam o Firebase como notória escolha pois demonstrava maior recursos e facilidade ao desenvolvedor apresentando maiores e melhores configurações de suporte a aplicação, tanto ao plano *Spark* quanto ao plano *Blaze*.

Um dos fatores que também deu suporte a escolha do Firebase como ferramenta para o estudo foi possuir uma biblioteca, que auxilie no desenvolvimento de aplicações de forma a otimizar recursos e serviços das tecnologias utilizadas. Além dos recursos bases providos por um BaaS, como banco de dados e autenticação, o Firebase provê produtos que podem trabalhar em conjunto com a aplicação, *Predictions*, *App Indexing* e *Firestore Cloud Messaging* (FCM), o que oferece maior oportunidade e facilidade de expansão da aplicação.

## 2.3 Tecnologias utilizadas

Nesta seção serão apresentadas as tecnologias utilizadas para a implementação do sistema descrito no estudo de caso no Capítulo 3.

### 2.3.1 NoSQL

A tecnologia *Not Only SQL* (NoSQL) é utilizada junto aos novos modelos de banco de dados, compondo a próxima geração que comportam-se como não relacionais, distribuídos e horizontalmente escaláveis. A intenção original, em seu desenvolvimento, foi a criação de um banco de dados moderno e escalável para a *web*, para aplicações que necessitem de alta disponibilidade e rápido processamento. Para isso, cada vez mais características se aplicam, como: fácil replicação, livre de esquema, API simples e eventualmente consistente (NOSQL, 2018).

Para a manipulação dos dados em um banco, é necessário a utilização de uma linguagem de consulta. Geralmente, a linguagem padrão é a *Structured Query Language* (SQL), a qual não é usada pelo NoSQL, pois a maioria de seus bancos fornecem sua própria linguagem de consulta. Devido ausência, completa ou parcial, do esquema que define a estrutura na modelação dos dados, o NoSQL é caracterizado como mais rápido, simplificando também a escalabilidade e aumentando a disponibilidade (SILVA, 2017).

### 2.3.2 JSON

O *JavaScript Object Notation* (JSON) é uma formatação leve de texto para troca de dados baseado em um subconjunto da linguagem JavaScript, sendo completamente independente de linguagem e estruturado em duas estruturas: pares nome/valor e uma lista ordenada de valores. Esta tecnologia representa quatro tipos primitivos (booleanos,

cadeia de caracteres, *null* e números) e dois tipos estruturados (matrizes e objetos) (BRAY, 2017).

Este formato pode ser usado para transferência de dados em aplicações *web*, e acabou sendo introduzido como substituto ao formato *eXtensible Markup Language* (XML), que também é uma linguagem de marcação capaz de descrever diversos tipos de dados. Isso ocorreu devido a complexa consulta XML, pois, além de serem realizada em dois níveis, servidor e cliente, as mensagens costumam ser pesadas e detalhadas, ocupando muita largura de banda. Já o JSON não precisa de nenhuma especificidade de implementações, que levou-o a tornar-se o formato preferencial para o intercâmbio de dados da Internet (SRIPARASA, 2013).

### 2.3.3 Angular

O Angular é um *framework* JavaScript de fácil entendimento, projetado, mantido e atualizado pela Google que simplifica o desenvolvimento de aplicações *web* robustas e dinâmicas. O mesmo ainda é um *meta-framework* para *Single Page Application* (SPA), que são aplicações desenvolvidas em JavaScript que rodam quase inteiramente no lado do cliente (*browser*). O Google foi o primeiro nesta tecnologia com o Gmail, e atualmente há diversas aplicações que utilizam esse padrão (GREEN, 2014).

De acordo com Mikowski e Powell (2013) páginas escritas em SPA minimizam o tempo de resposta para o usuário. Já que o *browser* apenas renderiza trechos de códigos para o cliente, diminuindo assim, o tráfego na rede e ganhando em performance para o usuário. O Angular surgiu para atender as necessidades do desenvolvedor, acelerando no processo de desenvolvimento, implementado conceitos da engenharia de *software* e deixando as aplicações interativas.

Essa é uma tecnologia estrutural para aplicativos da *web* dinâmicos. Permitindo a utilização do HTML como linguagem modelo, estendendo sua sintaxe para expressar os componentes da aplicação desenvolvida de maneira clara e sucinta. O Angular lida com todo *Document Object Model* (DOM) e *Asynchronous Javascript and XML* (AJAX), colocando-os numa estrutura bem definida, que o torna opinativo sobre como um aplicativo CRUD (*Create, Read, Update, Delete*) deve ser construído. Possui como características: vinculação de dados, diretivas de modelo básico, validação de formulários, roteamento, vinculação profunda, componentes reutilizáveis e *dependency injection* (ANGULAR, 2018).

### 2.3.4 Apache Cordova

O Apache Cordova é uma plataforma de código aberto desenvolvido pela *Apache Software Foundation* para a construção de aplicações multiplataformas utilizando HTML5. Seu principal objetivo é tornar fácil o desenvolvimento de aplicações *mobile* híbridas com

a combinação de tecnologias nativas e de aplicações *web*. Além disso, o Cordova também implementa um conjunto de APIs que estendem recursos nativos do dispositivo para uma aplicação *web* rodando no *container* nativo (WARGO, 2013).

Aplicações híbridas são um tipo de aplicação *web* que estendem suas funcionalidades por meio de APIs de plataformas nativas de determinado dispositivo (GOK; KHANNA, 2013). A escolha de determinada tecnologia foi, principalmente, devido às facilidades providas por tal, que além de permitir a construção de uma aplicação apenas utilizando uma única linguagem de programação também é aplicável para diferentes plataformas *mobile* Android, iOS e Windows Phone.

### 2.3.5 Ionic

O Ionic é um *framework front-end* SDK para o desenvolvimento de aplicações *mobile* híbridas fornecendo componentes HTML, CSS e JavaScript otimizados para dispositivos móveis, como também ferramentas para a criação de aplicações interativas. Além disso, as possibilidades se tornam ilimitadas com a inserção do Angular dentro de um *framework* como o Ionic (RAVULAVARU, 2015).

O Ionic utiliza o Apache Cordova para construção da aplicação de um *Webview*, e por meio desta, é possível apresentar conteúdos da *web* através de aplicativos. Devido integração com o Angular, o Ionic herdou várias de suas características, dentre elas a viabilidade de estender o HTML de modo a fornecer reuso, componentização e lógica para a marcação (WAHLBRINCK; BONIATI, 2017).

Toda a criação e desenvolvimento de aplicações que utilizam o Ionic acontece, principalmente, por meio do utilitário de linha de comando o CLI. E somente através do Apache Cordova pode-se construir/implantar a aplicação, até então elaborada por meio de tecnologias *web*, como um aplicativo nativo de determinada plataforma.

No próximo capítulo será apresentado o Estudo de caso que utilizou dos conceitos, tecnologias e ferramentas descritas aqui para sua completa estruturação, como também expõe a forma em que as tecnologias foram utilizadas e exhibe a aplicação desenvolvida por meio deste.

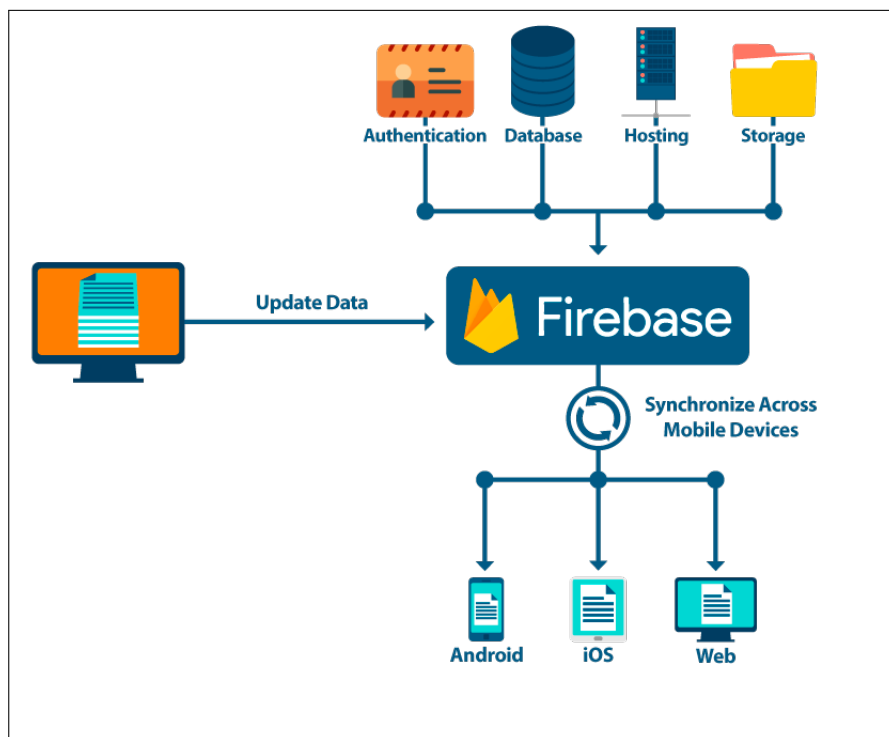
### 3 Estudo de caso

Com o propósito de abordar novas técnicas de programação, alternativas às demais formas convencionais, com serviços *backend*, será apresentado aqui um estudo de caso levando em conta as informações e teorias levantadas neste estudo sobre BaaS. O estudo será exposto de forma a defender e exemplificar a utilização de *backend* como serviço em um cenário real.

Para representar a utilização de serviços de um BaaS, foi desenvolvida uma aplicação *web* e uma *mobile*, demonstrando assim, além de sua interoperabilidade entre estas aplicações de diferentes plataformas, os serviços concedidos pelo serviço de *backend*. Foi então implementado um sistema de gerenciamento de notícias e informes, o qual permite, dependendo do nível do usuário, publicação e visualização.

Como pode ser visto na Figura 2, a aplicação foi desenvolvida utilizando o BaaS Firebase, o qual possui serviços disponibilizados a desenvolvedores. Foram usados os serviços de: *Database*, *Authentication*, *Storage* e *Hosting*.

Figura 2 – Estrutura de comunicação dos serviços Firebase.



Fonte – Adaptado de (FIREBASE, 2018).

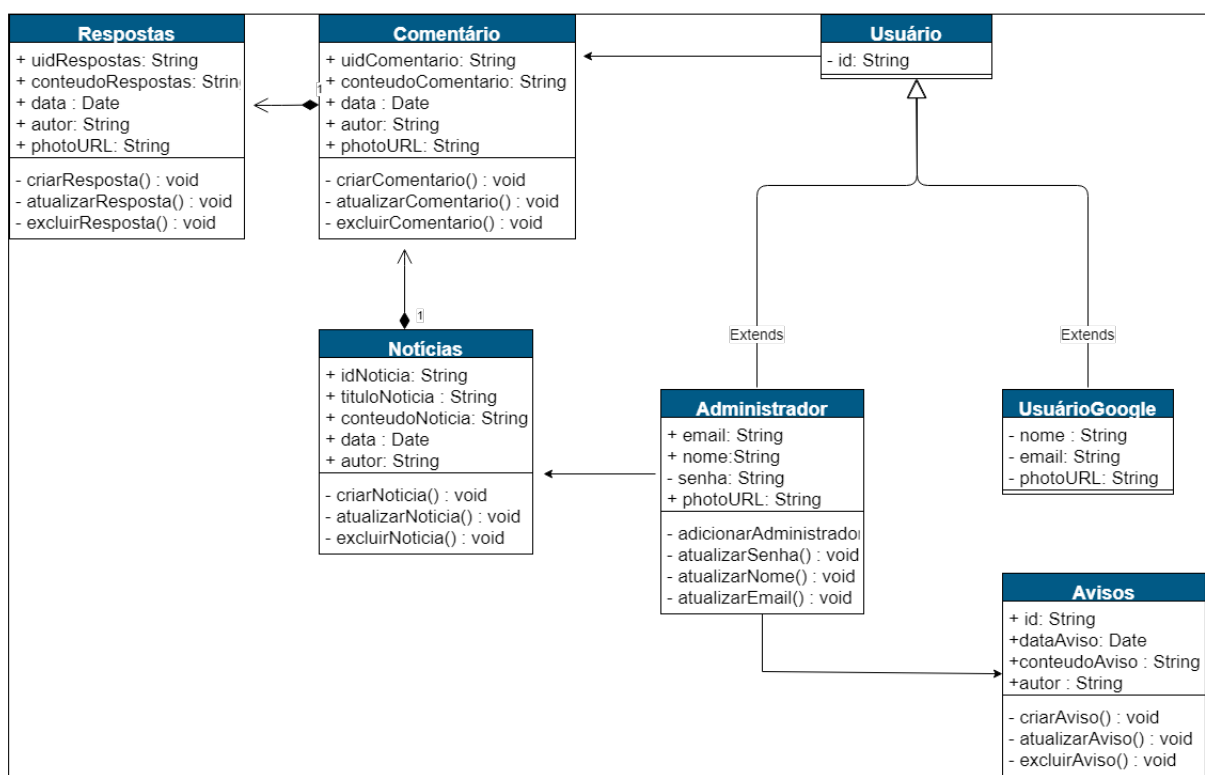
Os serviços descritos são acessíveis a todos os dispositivos conectados ao projeto,



independente da sua plataforma. Quando o projeto é inicializado na aplicação, o qual necessita de um código denominado de *snippet*, portará uma instância do Firebase *Database* que fornecerá, a cada dispositivo conectado, uma sincronização automática dos dados com todos os dispositivos.

Toda a estrutura do sistema desenvolvido pode ser descrita em um diagrama de classe e por meio dele são expostas as informações contidas na estrutura e na comunicação de suas classes, apresentando, assim, as associações existentes nelas. Na Figura 3, pode-se identificar as classes e seus respectivos métodos e atributos utilizados na implementação.

Figura 3 – Diagrama de classe da aplicação.



Fonte – Autoria própria.

Tem-se, no sistema, a classe *Usuario*, representando uma generalização das classes *Administrador* e *UsuarioGoogle*. A diferença entre estas classes, que representam uma extensão da classe *Usuario*, corresponde ao nível de acesso aos dados e funcionalidades da aplicação. Qualquer usuário pode acessar e utilizar os recursos das classes *Comentario* e *Resposta*, no entanto somente os administradores podem se comunicar com as classes *Aviso* e *Noticia*.

A classe *Administrador* possui informações necessárias para navegação, permissão e acesso como atributos. O administrador é capaz de cadastrar outros administradores, como também, atualizar todas as suas informações de cadastro. Além disso, é o único

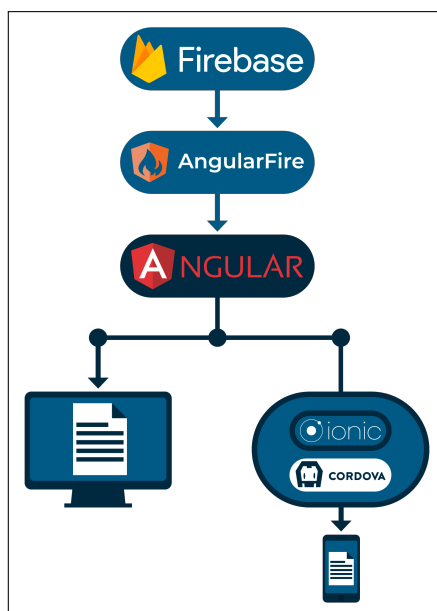
usuário com permissão de acesso às classes Noticia e Aviso.

Estas duas últimas classes mencionadas, Noticia e Aviso, são classes similares, porém apresentam funções distintas dentro do sistema. No passo que Noticia apresenta informações novas de interesse aos usuários em geral, tem-se Aviso que possui maior relevância, em especial, aos alunos da graduação. Ambas possuem métodos para auxiliar no seu gerenciamento, como criar, atualizar e excluir.

A classe Comentario está diretamente vinculada a Noticia que, uma vez publicada, possibilita a incorporação de comentários ao seu escopo. Por sua vez, Comentario também pode conter respostas de forma semelhante às classes mencionadas anteriormente, com a classe Resposta adicionada a Comentario. Estas classes também dispõem de funções que colaboram com sua manutenção.

Para a elaboração das suas funcionalidades, o desenvolvimento da aplicação ocorreu por meio da utilização de recursos e serviços do Firebase. Mesmo dispondo de abundantes, foram empregados apenas serviços estritamente necessários para o funcionamento desse sistema. Ainda que seja proposta sua utilização em diferentes plataformas, o Firebase possui alguns recursos em exclusividade para dispositivos móveis. Logo, para este sistema, foram empregados serviços em destaque para desenvolvedores, os quais abrangem as duas plataformas escolhidas: sistema *web* e aplicativo *mobile*.

Figura 4 – Tecnologias utilizadas para implementação.



Fonte – Autoria própria.

Para o sistema, foram empregados algumas tecnologias, apresentadas na Figura 4, que contribuiriam para o desenvolvimento do mesmo. O AngularFire, biblioteca do Firebase

apresentado na seção 2.2, atua como intermediário entre o Firebase e o *Framework* Angular, que dispõe de flexibilidade e agilidade no processo de desenvolvimento, para a implementação tanto *web* quanto *mobile*.

O sistema *web* foi desenvolvido utilizando recursos HTML e CSS, em conjunto com o *framework* para *front-end* Angular, que pode ser vista na subseção 2.3.3. Já o aplicativo *mobile* é híbrido, o qual também utiliza linguagens de programação *web* como JavaScript, HTML e CSS, para seu desenvolvimento. Foi então utilizado o Apache Cordova, que permite a utilização de recursos nativos do *smartphone*, com o auxílio do *framework* Ionic, mais informações na subseção 2.3.5, o qual foi idealizado com base no Angular.

Através da cooperação com o Apache Cordova, subseção 2.3.4, o Ionic supre as necessidades para a criação de interfaces *mobile*, possibilitando assim a criação de aplicações híbridas fornecendo suporte multiplataforma: Android e IOS.

## 3.1 Serviços BaaS utilizados

Nas próximas seções serão apresentados os serviços fornecidos pelo Firebase e que foram utilizados no desenvolvimento do sistema. O método de implementação fez uso do AngularFire, a biblioteca oficial para utilização do *Framework* Angular e do BaaS Firebase juntos.

Esta biblioteca possui alguns métodos, já existentes, que servem como intermediários na utilização dos serviços disponibilizados pelo Firebase. Os métodos empregados serão apresentados, assim como, o vínculo entre às funcionalidades da aplicação e os respectivos serviços BaaS utilizados.

### 3.1.1 Database

Todas as informações necessárias à aplicação são armazenadas no banco de dados de tempo real do Firebase. O *Realtime Database* é um banco de dados NoSQL, modelo visto na subseção 2.3.1, original do Firebase, o qual armazena e sincroniza dados em tempo real para todos os usuários conectados a aplicação. O *Database* é estruturado no formato JSON, apresentado na seção subseção 2.3.2, o qual permite maior flexibilidade na construção no banco da aplicação, proporcionando diferentes estruturas com diferentes tipos de dados. Devido a isso, este serviço fornece uma maneira simples de armazenar e acessar os dados.

Código 3 – Exemplificação da estrutura principal da aplicação.

```
{ 1
  "noticia": { 2
    "uid": { 3
      "noticia1": {} 4
```

```

    },
    "uid2": {
        "noticia2": {}
    }
},
"aviso":{
    "uid": {
        "aviso1": {}
    },
    "uid2": {
        "aviso2": {}
    }
},
"usuario":{
    "uid": {
        "usuario1": {}
    },
    "uid2": {
        "usuario2": {}
    }
}
}

```

Fonte – Autoria própria.

O Código 3 é uma representação dos principais nós da aplicação. Todos os subnós criados na árvore, contêm elementos relacionados aos nós raiz, e ainda são separados e identificados por um “uid” que são gerados automaticamente pelo *Database*. Em virtude disso, é demonstrado a facilidade e otimização no processo de criação e busca no Firebase pois, dessa forma, é habilitado ao desenvolvedor a permissão de escrita e leitura a nós específicos da árvore, por meio do “uid”. Sendo assim, não há desperdício de processamento carregando todos os subnós de um objeto, se houver interesse em apenas um nó em especial.

#### Código 4 – Código de acesso a todas as notícias.

```

getAllNoticias = function () {
    return $firebaseArray($firebaseRef.noticia);
};

```

Fonte – Autoria própria.

O acesso aos dados para leitura e escrita no *Database* pode ser realizado por meio de funções exemplificadas nos Códigos 4 e 5. O Código 4 retorna as referências de todas as notícias armazenadas no *Database*. Já o Código 5 retorna a referência de uma notícia específica, sem a necessidade do carregamento das demais, informando apenas o “uid” de determinado subnó.

## Código 5 – Código de acesso a notícia específica.

---

```

getNoticia = function (uid) {
    const ref = firebase.database().ref('noticia').child(uid);
    return $firebaseObject(ref);
};

```

---

Fonte – Autoria própria.

As funções *\$firebaseArray()*, *\$firebaseObject()* e *\$firebaseRef()* expostas nos Códigos 4 e 5, são serviços nativos da biblioteca AngularFire. Esses serviços são utilizados para realizar a comunicação e gerência dos dados presentes no Firebase *Database*.

Tabela 2 – Referências do AngularFire.

SERVIÇOS	MÉTODOS	DESCRIÇÃO
<i>\$firebaseObject()</i>	<i>\$remove()</i>	Utilizado para remoção do objeto tanto localmente quanto do database. Retorna uma promessa que, com a confirmação da remoção, será cumprida.
	<i>\$save()</i>	Envia alterações realizadas para o servidor. Retorna uma promessa que será cumprida com a confirmação da gravação.
	<i>\$loaded()</i>	Seu retorno também é uma promessa, que no final é o próprio objeto. Ela é resolvida de forma assíncrona quando os dados são carregados do Firebase.
	<i>\$ref()</i>	Retorna a referência do Firebase para a criação/utilização de determinado objeto.
<i>\$firebaseArray()</i>	<i>\$add(newData)</i>	Cria um novo registro no database.
	<i>\$remove(Index)</i>	Remove um registro no database. É enviado, como parâmetro, o índice ou um \$ref a um item existente.
	<i>\$save(Index)</i>	Atualiza e salva registros individuais no database. É enviado, como parâmetro, o índice ou um \$ref a um item existente.
	<i>\$getRecord(key)</i>	Retorna o registro do array para a chave fornecida como parâmetro.

Fonte – Autoria própria.

A Tabela 2 apresenta alguns métodos usados em conjunto com os serviços mencionados e suas funcionalidades. Esses métodos foram utilizados para realizar o gerenciamento dos dados armazenados no *Database*.

### 3.1.2 Authentication

O Firebase dispõe do serviço de autenticação (*Authentication*) para o reconhecimento da identidade do usuário. Ele fornece diferentes métodos de *login*, dentre estes, os utilizados no sistema foram: autenticação baseada em *e-mail* e senha e integração do provedor de identidade federada.

O AngularFire também inclui suporte para autenticação e gerenciamento de usuários por meio do serviço *\$firebaseAuth()*. Para o método de *login* de integração com provedor, o qual é permitido pelo Firebase a integração da aplicação com provedores de identidade federadas, é necessário a escolha de determinado provedor para o *login*. Os provedores válidos são: Github, Facebook, Twitter e Google.

O método de login com provedor Google é utilizado na aplicação para que usuários, visitantes, acessem recursos, mas com limite de permissão. Representando assim, a classe *UsuarioGoogle* apresentada na Figura 3. O Código 6 exemplifica o acesso ao recurso utilizando o serviço *\$firebaseAuth()* juntamente com o método de *login* com o provedor: *\$signInWithPopup()*.

Código 6 – Método de login através de provedores.

---

```
_signInGoogle = function () { 1
  const auth = $firebaseAuth(); 2
  auth.$signInWithPopup("google"); 3
}; 4
```

---

Fonte – Autoria própria.

A autenticação baseada em *e-mail* e senha é realizada por meio de dados, *e-mail* e senha, próprios, os quais não dependem de integração com provedor de identidade federada, ao contrário do *UsuarioGoogle*, para realizar a ação. Por este modelo fornecer assistência através de métodos para criar e gerenciar usuários, foi então destinado aos usuários com nível de administradores. Para isto, usou-se o método *\$createUserWithEmailAndPassword()*, exemplificado no Código 7, para a criação dos usuários administradores, os quais enviam unicamente a senha e *e-mail* para realização do cadastro no Firebase. Logo, para o seu login na aplicação é utilizado o método *\$signInWithEmailAndPassword(e-mail, senha)*, também demonstrado no Código 7.

Código 7 – Método de login através de *e-mail* e senha.

---

```
const auth = $firebaseAuth(); 1
auth.$createUserWithEmailAndPassword(user.email, user.senha); 2
3
auth.$signInWithEmailAndPassword(login.email, login.senha) 4
  .then(function (firebaseUser) { 5
    console.log("Login realizado com sucesso!"); 6
  }).catch(function (error) { 7
```

---

---

```

        console.log(error);
    });

```

---

Fonte – Autoria própria.

O gerenciamento disponibilizado aos usuários com permissão de administradores dá-se por meio de algumas funções, também disponíveis pelo serviço *\$firebaseAuth()*. Esses métodos e suas funcionalidades podem ser visualizadas na Tabela 3.

Tabela 3 – Métodos do *\$firebaseAuth*

MÉTODOS	DESCRIÇÃO
<i>\$createUserWithEmailAndPassword(email, password)</i>	Cria uma nova conta de usuário com a combinação de email e senha.
<i>\$updatePassword(password)</i>	Altera a senha do usuário conectado.
<i>\$save(Index)</i>	Modifica o email do usuário conectado.
<i>\$deleteUser()</i>	Exclui o usuário autenticado.
<i>\$sendPasswordResetEmail(email)</i>	Envia uma email de redefinição de senha para o proprietário da conta.

Fonte – Autoria própria.

### 3.1.3 Storage

O *Cloud Storage* para o Firebase é um serviço de armazenamento de objetos fornecido ao desenvolvedor para armazenar e veicular conteúdo gerado pelo usuário, como, imagens, áudios e vídeos. O *download* e *upload* dos arquivos é realizado por meio dos SDKs do Firebase.

Este serviço foi utilizado para habilitar a funcionalidade de armazenamento de arquivos multimídias na aplicação, em especial fotos dos usuários e imagens relacionadas às notícias. O acesso ao *Storage*, também por meio do AngularFire, é feito através do serviço *\$firebaseStorage()*. Para sua implementação é necessário os seguintes passos:

- I. Integrar os SDKs do Firebase para o Cloud Storage;
- II. Criar uma referência;
- III. Realizar *upload* ou *download*;
- IV. Proteção dos arquivos.

A integração do SDK é feita com a inserção das configuração do projeto Firebase para a aplicação. Já a referência é criada com as informações presentes no Código 8 .

Código 8 – Referência do *Storage*.

---

```
const storageRef = firebase.storage().ref();
```

---

Fonte – Autoria própria.

Sendo assim, com esses passos concluídos, qualquer ação pode ser realizada a partir do serviço *\$firebaseStorage()*, dentre elas o *upload* e *download* dos arquivos. Para isto, o AngularFire ainda fornece métodos para a realização de determinadas funcionalidades, e a demonstração de *download* e *upload* pode ser vista na Tabela 4.

Tabela 4 – Referências do AngularFire para *Storage*.

MÉTODO	EXEMPLO	DESCRIÇÃO
<i>\$put(file, metadata)</i>	var fileRef = storage- Ref.child(id);	Método utilizado para <i>upload</i> de arquivos no <i>Storage</i>
<i>\$getDownloadURL()</i>	urlImagem = snapshot.downloadURL;	Método utilizado para <i>download</i> de arquivos no <i>Storage</i>

Fonte – Autoria própria.

Para a proteção dos arquivos é fornecido ao desenvolvedor regras de segurança do Firebase para o *Cloud Storage*. Com isto, é permitido baixar e salvar os arquivos dependendo da sua permissão, no caso da aplicação proposta, é permitido somente aos usuários autenticados realizarem o *upload* de arquivos. Para isso, o serviço de autenticação é vinculado ao *Cloud Storage*.

Código 9 – Regras do *Storage*.

---

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

---

Fonte – Autoria própria.

O exemplo de utilização da rules no Storage pode ser visto no Código 9. Este código informa que às regras especificadas correspondem (*match*) a determinado caminho do arquivo. No primeiro momento acontece o *match* com o nome do repositório do objeto, o *bucket*. Logo após, é informado o caminho de determinados arquivos, e então às regras descritas serão aplicadas para os arquivos que compõem o caminho especificado.



### 3.1.4 Hosting

É um serviço de hospedagem de conteúdo *web* para desenvolvedores. Ele oferece conexão segura por meio do protocolo *Secure Socket Layer* (SSL), que é incorporado ao Firebase, não havendo necessidade de configuração. Ainda fornece, gerenciamento das versões hospedadas com auxílio de *rollbacks*.

Para sua implementação é necessário o CLI do Firebase, o qual fornece uma série de ferramentas de gerenciamento, visualização e implantação para projetos, e dentre eles, há o *Hosting*. Para sua aplicação tem-se os seguintes passos:

- I. Firebase CLI;
- II. Configuração do diretório do projeto;
- III. Implantação do site;
- IV. Gerenciar e reverter aplicações.

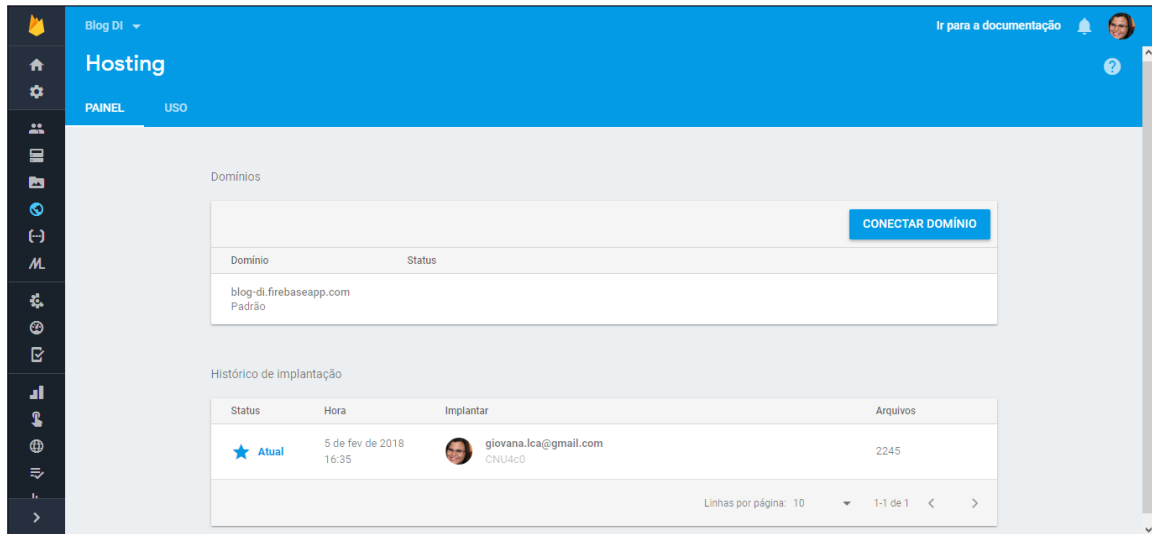
O Firebase CLI é configurado no início do projeto, o qual vincula a aplicação desenvolvida com o projeto Firebase auxiliando tanto o desenvolvimento quanto a implantação do *website*. Para inicializar a aplicação determina-se um diretório como raiz pública (*public*), contendo um arquivo “index.html”.

Para a implantação executa-se o comando “*firebase deploy*” e a aplicação é desenvolvida no domínio <nome-da-aplicação>.firebaseapp.com. O gerenciamento é fornecido por meio do painel *hosting*, apresentado na Figura 5, do Firebase console, o qual é possível ver o histórico das implantações, conectar um domínio já existente e reverter o estado do *website* por meio das implantações já realizadas.

## 3.2 Aplicação desenvolvida - BlogDI

A aplicação desenvolvida foi o BlogDI que objetiva, a priori, reunir e centralizar as principais informações geradas pela graduação e suas áreas afins. Contribuindo com a troca de informações, assim como tentar manter os discentes atualizados e informados sobre o andamento das atividades e acontecimentos do semestre.

O BlogDI é uma plataforma que permite visualização e inclusão de notícias e avisos, configurando-se como meio de comunicação aberto entre discentes e o corpo docente do curso, transformando-o em um canal focado nas necessidades do graduando, assim como em uma vitrine, expondo as contribuições geradas pelo programa de graduação a comunidade acadêmica e a sociedade como um todo criando dessa forma, uma identidade digital para o curso.

Figura 5 – Painel de gerenciamento do *Hosting*.

Fonte – Autoria própria.

Figura 6 – Página inicial da aplicação.



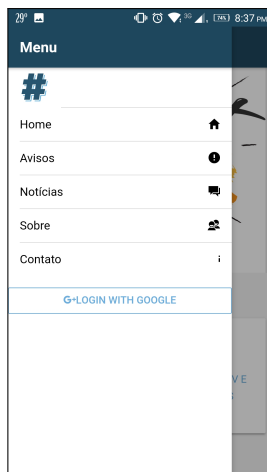
Fonte – Autoria própria.

A Figura 6 apresenta a página *home* do sistema, disponível tanto para administradores como visitantes. É possível visualizar, na Figura 6a, a área de *login* do visitante, disposta no lado superior esquerdo, que ao simples clique, inicia o processo de autenticação com o provedor Google. Além disso, apresenta também, as demais informações base do sistema, notícias e avisos, os quais sofrem uma restrição de visibilidade permitindo que apenas as últimas apareçam. Os avisos são mostrados de forma alternada e automática ao usuário, já para as notícias é necessário a utilização das setas para sua movimentação.

Diferente da mencionada, a Figura 6b apresenta somente as últimas informações

adicionadas, a funcionalidade de *login* é disposta no menu da aplicação, apresentado na Figura 7. No menu ainda são expostas informações de navegação para o usuário.

Figura 7 – Menu da aplicação móvel.



Fonte – Autoria própria.

Na área de Notícias, Figura 8, é apresentado ao usuário todas as notícias já criadas e que residem no banco de dados da aplicação. A exposição é constituída de uma imagem da notícia juntamente com o título, data da criação e um pequeno texto introdutório.

Figura 8 – Todas as notícias da aplicação.



(a)

(b)

Fonte – Autoria própria.

Ao clicar em determinada notícia o usuário será encaminhado a página da notícia específica, na qual irá deparar-se com as informações completas da respectiva notícia, além do título, data de criação e o autor, como exemplificado na Figura 9.

Além das respectivas informações disposta na página da notícia, ainda é permitido ao usuário criar comentários, Figura 10b, a respeito da mesma, o qual, após realizado

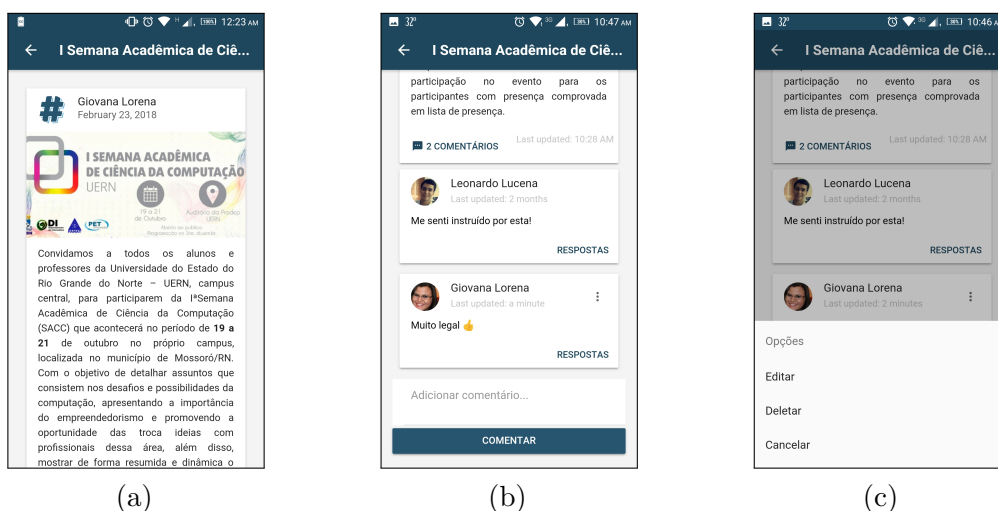
Figura 9 – Notícia selecionada visão *web*.



Fonte – Autoria própria.

*login* na aplicação, é habilitado. Uma vez criados, também é permitido ao usuário realizar alteração e até mesmo exclusão do comentário, por meio de um menu, apresentado na Figura 10c, fornecido somente ao criador de determinado comentário e aos usuários com nível de permissão de administradores.

Figura 10 – Notícia selecionada visão *mobile*

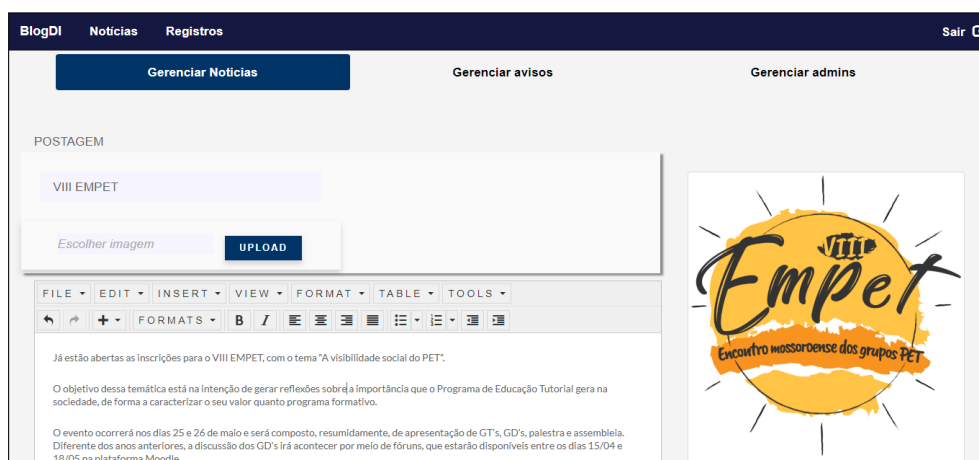


Fonte – Autoria própria.

Todas as informações apresentadas no BlogDI são geradas a partir de uma área no *site* destinada aos administradores, a qual é somente disponível na plataforma *web*. Ao realizar *login* no sistema o administrador é encaminhado para a área inicial do sistema gerenciador, Figura 11.

Neste momento, é mostrado ao usuário a área destinada a publicação de notícias da

Figura 11 – Publicação de notícia no sistema.



Fonte – Autoria própria.

aplicação, na qual é possível inserir o título da notícia, realizar *upload* de uma imagem e, com auxílio de um editor de texto, adicionar informações pertinentes a mesma. É fornecido ao administrador um menu que permite-o navegar entre gerenciar notícia, aviso e outros administradores.

Figura 12 – Área de gerenciamento de avisos no sistema.

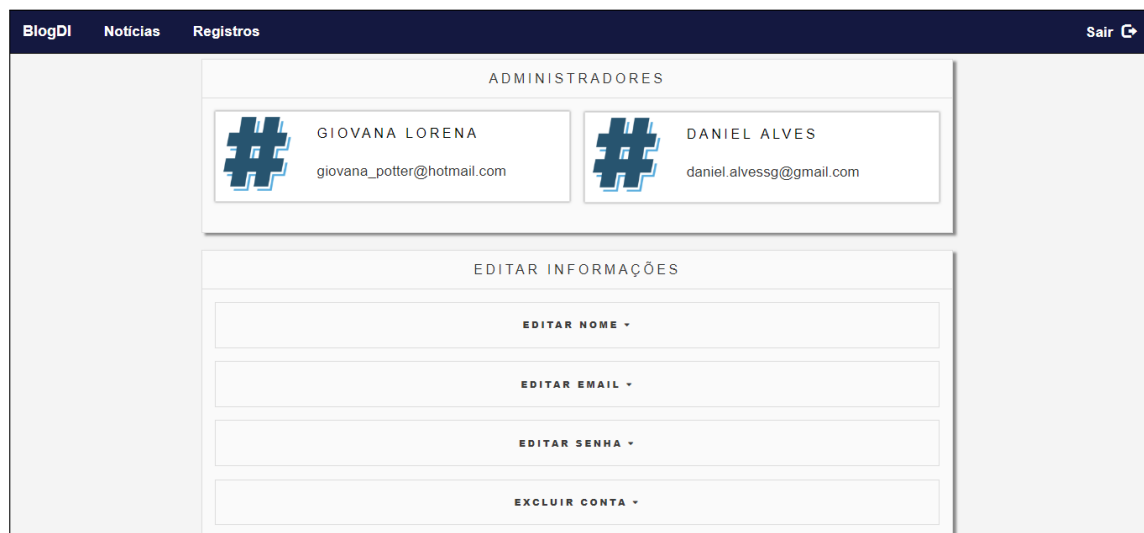


Fonte – Autoria própria.

Em gerenciar avisos, além de permitir a publicação, o usuário tem acesso, por meio de uma tabela, a todos os avisos já criados, podendo também alterar e excluir. Os avisos, Figura 12, são apresentados, conteúdo ao lado, informações como: data e hora da criação e botões para editar e excluir o aviso.

Na área de gerenciar administradores, Figura 13, é fornecido uma lista com todos

Figura 13 – Área de gerenciamento do perfil administrador.



Fonte – Autoria própria.

os administradores cadastrados no sistema, como também uma área destinada aos mesmos para alteração em determinadas informações, são elas:

- Editar nome;
- Editar *e-mail*;
- Alterar senha;
- Excluir conta.

Além de todas as funcionalidades mencionadas, o sistema ainda possui uma área destinada para observação e monitoramento das ações executadas no mesmo, o registro de eventos apresentado na Figura 14.

Para este meio, a aplicação apresenta informações pertinentes, relacionadas as ações que ocorreram no sistema. Informações estas que são mostradas, por meio de uma tabela, com alguns atributos como o tipo de ação realizada, data da ação, o autor e função (administrador ou visitante). O registro é estruturado em quatro campos:

- Últimas modificações: apresenta de forma geral e sem restrições, a sequencia de ações ocorridas de forma decrescente;
- Creat logs: informações sobre a utilização de determinada funcionalidade que permita a criação/adição de notícias, avisos, administradores e comentários;

- Update logs: apresenta todas as informações que foram atualizadas no sistema;
- Delete logs: registro de todas as ações de exclusão executada na aplicação.

Figura 14 – Registro de eventos do sistema.

Ação realizada	Data	Autor da ação	Função	
"Publicação de notícia"	📅 03/05/2018 🕒 18:51:30	Giovana Lorena	Administrador	...
"Novo administrador adicionado"	📅 11/04/2018 🕒 12:42:06	Giovana Lorena	Administrador	...
"Exclusão de comentário"	📅 19/03/2018 🕒 15:32:59	Giovana Lorena	Usuário Google	...
"Exclusão de comentário"	📅 19/03/2018	Giovana Lorena	Usuário Google	...

Fonte – Autoria própria.

## 4 Considerações finais

O desenvolvimento de aplicações está crescendo a um patamar em que, cada vez mais, é necessário atribuições de recursos tecnológicos como auxílio aos programadores. Isto para suprir a crescente demanda de atributos e funcionalidades para complexos sistemas, alavancando na dificuldade e, conseqüentemente, demora de sua implementação. Como mecanismo de abstração parcial, do desenvolvimento de sistemas, está a contribuição da Computação em nuvem, que por meio de seus serviços, promove suporte, auxílio e simplicidade para elaboração de aplicações.

Através destes serviços da Computação em nuvem, são fornecidos como plataformas de gerenciamento e utilização de recursos o *Backend as a service*, o qual provê serviços aos programadores para o desenvolvimento de sistemas. Havendo assim, uma otimização de tempo e recursos na implementação. O BaaS Firebase, desenvolvido pela Google, é apresentado como meio alternativo a programação, fornecendo serviços e produtos como recursos as aplicações.

Este trabalho apresentou tecnologias para desenvolvimento, tanto *web* quanto *mobile* híbrido, de aplicações que desempenham e possuem atributos distintos e requisitados das mais diversas formas. Para isso propôs-se uma plataforma fornecedora de uma tecnologia que oferecesse o suporte necessário para o desenvolvimento dessas aplicações, o *BaaS*. Expondo suas características, ferramentas e serviços propostos para esse meio.

Com isso, foi demonstrado, por meio de um estudo de caso, o desenvolvimento de um sistema utilizando o Firebase. Através deste, foi exemplificado as facilidades e a otimização na forma de se programar sistemas. Para isso, foi apresentado alguns dos serviço disponibilizados pelo Firebase e a forma que foram utilizados para implementar as funcionalidades e requisitos do sistema exposto. Os serviços foram: *Database, Authentication, Storage, Hosting*.

A utilização deste serviço trouxe como vantagem a redução das tecnologias usadas e implementadas, pois, sem isso, haveria a necessidade de confecção de um banco de dados além da definição do esquema como coleções e tabelas. Como também, a criação de uma API para facilitar a usabilidade da aplicação, gerenciamento de dados e comunicação, implementada de uma forma em que respeite os padrões de desenvolvimento para fornecer segurança. E por último, a configuração de um servidor criando toda a estrutura para comunicação de todas as camadas anteriores, banco de dados e API.

E com os serviços fornecidos pelo BaaS em questão, foi possível desenvolver todo o sistema com auxílio de uma biblioteca, também disponibilizada pelo Firebase, como intermediário conectando os serviços do Firebase com o *frontend* da aplicação, o AngularFire.



Essa conexão é estabelecida por meio da invocação de métodos do AngularFire que estão vinculados aos serviços do Firebase.

Como trabalhos futuros procura-se adaptar a aplicação para serem utilizados mais dos serviços oferecidos pelo Firebase, dentre eles encontram-se o envio e recebimento de mensagens e notificações o *Firebase Cloud Messaging*, *Cloud Functions* para criação de funções autônomas e o *Cloud Firestore* banco de dados orientado a documentos.

Desta forma, o sistema será remodelado para a inserção de mais funcionalidades sendo transformado em: Um sistema de gerenciamento de notícias e informes com notificação seletiva de interesse. O FCM ficará a cargo das notificações seletivas, já a *Cloud Functions* será responsável pelas funções que serão acionadas por modificações no banco de dados e *Cloud Firestore* como novo banco de dados, pois, levando em consideração as alterações e inserções de funcionalidades no sistema, torna-se mais indicado, devido a complexidade na estrutura do banco de dados. Além disso, também será proposto a utilização, além dos demais serviços, de produtos para expansão da aplicação como *Google Analytics* para análise e *Invites* para gerenciar o compartilhamento de conteúdo da aplicação, proporcionando assim crescimento, otimização e agilidade do sistema.

Propõe-se também a realização de um estudo comparativo entre diferentes tipos de provedores BaaS. Para que, através deste, possa-se avaliar os requisitos classificando e diferenciando-os dos demais, tendo em vista desempenho, serviços e produtos fornecidos.

# Referências

- ANGULAR. *Angular*. 2018. Documentação do Framework Angular. Acessado em 2018. Disponível em: <<https://docs.angularjs.org/guide/introduction>>.
- ANGULARFIRE. *AngularFire*. 2018. Documentação do AngularFire. Acessado em 2018. Disponível em: <<https://www.firebase.com/docs/web/libraries/angular/api.html>>.
- BACK4APP. *Back4App*. 2018. Documentação do Back4app. Acessado em 2018. Disponível em: <<https://www.back4app.com/docs/parse-documentation>>.
- BACKENDLESS. *Backendless*. 2018. Documentação do Backendless. Acessado em 2018. Disponível em: <<https://backendless.com/products/documentation/>>.
- BHARDWAJ, S.; JAIN, L.; JAIN, S. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, v. 2, n. 1, p. 60–63, 2010.
- BRAY, T. The javascript object notation (json) data interchange format. 2017.
- CAROLAN, J. et al. Introduction to cloud computing architecture. *White Paper, 1st edn. Sun Micro Systems Inc*, 2009.
- CHEN, X. et al. Primary exploration of mobile learning mode under a cloud computing environment. In: IEEE. *E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on*. [S.l.], 2010. v. 2, p. 484–487.
- CLOUDBOOST. *Cloudboost*. 2018. Documentação do Cloudboost. Acessado em 2018. Disponível em: <<https://tutorials.cloudboost.io>>.
- COSTA, I. d. O. Modelos par análise de disponibilidade em uma plataforma de mobile backend as a service. Universidade Federal de Pernambuco, 2015.
- DILLON, T.; WU, C.; CHANG, E. Cloud computing: issues and challenges. In: IEEE. *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. [S.l.], 2010. p. 27–33.
- FIREBASE. *Firebase*. 2018. Documentação do Firebase. Acessado em 2018. Disponível em: <<https://firebase.google.com/docs/>>.
- GOK, N.; KHANNA, N. *Building Hybrid Android Apps with Java and JavaScript: Applying Native Device APIs*. [S.l.]: "O'Reilly Media, Inc.", 2013.
- GONG, C. et al. The characteristics of cloud computing. In: IEEE. *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*. [S.l.], 2010. p. 275–279.
- GREEN, S. B. *Desenvolvendo Com Angularjs: Aumento de Produtividade Com Aplicações Web Estruturadas*. [S.l.]: Novatec Editora, 2014.
- LANE, K. Overview of the backend as a service (baas) space. *API Evangelist*, 2015.

- LI, K.-C.; LI, Q.; SHIH, T. K. *Cloud computing and digital media: fundamentals, techniques, and applications*. [S.l.]: CRC Press, 2014.
- LOPES, S. *Aplicações mobile híbridas com Cordova e PhoneGap*. [S.l.]: Editora Casa do Código, 2016.
- LUAN, T. H. et al. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.
- MARKETSANDMARKETS. *Cloud Backend-as-a-service (BaaS) Market - Global Advancements, Business Models, Technology Roadmap, Forecasts Analysis (2012-2017)*. 2016. Acessado em 2018. Disponível em: <<https://www.marketsandmarkets.com/Market-Reports/mobile-backend-as-a-service-mbaas-market-813.html>>.
- MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.
- MIKOWSKI, M. S.; POWELL, J. C. Single page web applications. *B and W*, 2013.
- NOSQL. *NoSQL*. 2018. NoSQL Databases. Acessado em 2018. Disponível em: <<http://nosql-database.org>>.
- 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.
- PANDEY, S.; NEPAL, S. Modeling availability in clouds for mobile computing. In: IEEE. *Mobile Services (MS), 2012 IEEE First International Conference on*. [S.l.], 2012. p. 80–87.
- PINELI, J. C.; DUARTE, M. Ambiente de desenvolvimento de software em nuvem. *Revista ef@ tec*, v. 3, n. 1, 2013.
- PRESSMAN, R.; LOWE, D. *Web Engineering: A Practitioner's Approach*. [S.l.]: McGraw-Hill, Inc., 2009.
- PRESSMAN, R.; MAXIM, B. *Engenharia de Software-8ª Edição*. [S.l.]: McGraw Hill Brasil, 2016.
- RAVULAVARU, A. *Learning Ionic*. [S.l.]: Packt Publishing Ltd, 2015.
- RIMAL, B. P.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. *NCM*, v. 9, p. 44–51, 2009.
- ROUNTREE, D.; CASTRILLO, I. *The basics of cloud computing: Understanding the fundamentals of cloud computing in theory and practice*. [S.l.]: Newnes, 2013.
- RUSCHEL, H.; ZANOTTO, M. S.; MOTA, W. d. Computação em nuvem. *Pontifícia Universidade Católica do Paraná, Curitiba, Brazil*, 2010.
- SCHNEIDER, A. H. Desenvolvimento web com client side rendering: combinando single page application e serviços de backend. 2016.

- SILVA, C. B. D. Um modelo computacional para integração de problemas de otimização utilizando banco de dados orientados a grafos. 2017.
- SOUSA, F. R.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)*, p. 150–175, 2009.
- SRIPARASA, S. S. *JavaScript and JSON essentials*. [S.l.]: Packt Publishing Ltd, 2013.
- TAURION, C. Cloud computing: computação em nuvem: transformando o mundo da tecnologia da informação. *Rio de Janeiro: Brasport*, v. 2, n. 2, p. 2–2, 2009.
- VAQUERO, L. M. et al. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, ACM, v. 39, n. 1, p. 50–55, 2008.
- VERAS, M. *Cloud Computing: nova arquitetura da TI*. [S.l.]: Brasport, 2012.
- WAHLBRINCK, K. A.; BONIATI, B. B. Aplicações mobile híbridas: Um estudo de caso do framework ionic para construção de um diário de classe. 2017.
- WARGO, J. M. *Apache Cordova 3 Programming*. [S.l.]: Pearson Education, 2013.
- WILLIAMS, M. I. *A quick start guide to cloud computing: moving your business into the cloud*. [S.l.]: Kogan Page Publishers, 2010.
- ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, Springer, v. 1, n. 1, p. 7–18, 2010.