INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

HUELISON KEMMERICH

Um estudo de caso usando BaaS em uma aplicação Mobile

Anubis Graciela de Moraes Rossetto

PASSO FUNDO
2017

HUELISON KEMMERICH

Um estudo de caso usando BaaS em uma aplicação Mobile

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientadora: Anubis Graciela de Moraes Rossetto

PASSO FUNDO

2017

HUELISON KEMMERICH

Um estudo de caso usando BaaS em uma aplicação Mobile

Trabalho de Conclusão de Curso aprovado em 01/12/207 como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Anubis Graciela de Moraes Rossetto

_____

Élder Francisco Fontana Bernardi

_____

José Antônio Oliveira de Figueiredo

_____

Rafael Marisco Bertei

PASSO FUNDO

2017

Aos meus pais e irmãos,
por estarem ao meu lado em todos os momentos.
Aos meus professores,
por todos os conselhos e ensinamentos.

“A dúvida é o princípio da sabedoria.”

Aristóteles

RESUMO

Devido à tendência na utilização da computação em nuvem, o BaaS (Backend as a Service) tem se tornado um modelo emergente para o desenvolvimento de aplicações. O objetivo do BaaS é prover serviços como armazenamento e gerenciamento de dados e arquivos, manutenção de usuários, juntamente com integração às redes sociais, abstraindo toda a complexidade que esses serviços possuem. Esse trabalho fez um levantamento sobre alguns dos principais serviços que o BaaS disponibiliza, elencando e analisando suas características. Como resultado do estudo, foi desenvolvido um estudo de caso empregando um dos serviços analisados: Firebase. Este estudo de caso engloba a solução para um aplicativo mobile de controle de rotas de leite, integrado com o ERP Administrador. Para essa integração, foi utilizado o serviço BaaS analisado, que provê os serviços de armazenamento e autenticação, atuando como o backend do projeto. Além disso, para o desenvolvimento foram utilizados o framework IONIC, juntamente com o Cordova e o Angular, para a app mobile, e o RAD Studio XE8, para a aplicação de integração com o sistema ERP.

Palavras-chave: BaaS; Firebase; Computação em Nuvem;

ABSTRACT

Due the trend in the use of cloud computing, the BaaS (Backend as a Service) has become an emerging model for application development. The purpose of BaaS is to provide services such as storage and management of data and files, maintenance of users, along with integration with social networks, abstracting all the complexity that these services have. This work surveyed some of the main services that BaaS provides listing and analyzing its features. As a result of the study, a case study was developed using one of the services analyzed: Firebase. This case study includes the solution for a mobile milk route control application, integrated with the ERP Administrador. For this integration, the analyzed BaaS service was used, which provide storage and authentication services, acting as the project backend. In addition, for the development were used the IONIC framework, along with Cordova and Angular, for the mobile app, and RAD Studio XE8, for the integration application with the ERP system

Keywords: BaaS; Firebase; Cloud Computing;

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Os últimos anos foram marcados pelo crescimento vertiginoso dos dispositivos móveis juntamente com aplicações para esta plataforma. Prova disso, é que atualmente a maior parte dos acessos à internet é feito por meio de dispositivos móveis. De acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE), 92,1% do acesso à rede passou a ser feito pelo dispositivo móvel (G1, 2016). Juntamente com esse crescimento surgiu a necessidade de uma infraestrutura compartilhada que desse suporte a serviços sob demanda e escalonáveis: a Computação em Nuvem. Essa estrutura se torna essencial, uma vez que o uso dos dispositivos móveis tem gerado uma dependência no sentido de que os serviços estejam disponíveis a qualquer hora e lugar e atendendo requisitos de alta disponibilidade e confiabilidade.

Uma prova da importância que a Computação em Nuvem vem tomando é que o uso da mesma vem crescendo nos últimos anos no Brasil e, como mostra uma pesquisa realizada pela EXAME (2015), existe uma expectativa que as empresas invistam cerca de 30% a mais entre 2013 e 2018 nessa área, o que demonstra a atenção que estas estão dando para essa nova modalidade de computação. Essa demanda esta relacionada, como já mencionado, ao aumento do número de usuários de dispositivos móveis e consequentemente ao maior uso de aplicativos. Nessa nova abordagem de acesso, o usuário almeja por informações em tempo real, de maneira fluida, indicando com isso a necessidade de um sistema que seja capaz de distribuir uma grande quantidade de dados, com segurança e estabilidade.

Muitas das aplicações que usamos pelos dispositivos móveis, mesmo sem percebermos, foram desenvolvidas e atuam por duas etapas distintas: o *frontend* e o *backend*. O *frontend* está relacionado com as ações que ocorrem entre o usuário e o dispositivo, por exemplo, a interface que contém botões, imagens, textos e animações. Já o *backend* é a infraestrutura que por sua vez diz respeito a servidores, armazenamento e rede. Nesse sentido, a Computação em Nuvem busca dar suporte para facilitar o trabalho no *backend*. Ela fornece ao usuário a capacidade de acessar, trabalhar, compartilhar e armazenar informações usando a Internet.

No *backend*, além da necessidade de instalação e configuração de servidor web, banco de dados, linguagens do lado servidor (*PHP, Python, Java,* ...) é

necessário também tratar de questões como performance, segurança, escalabilidade, bem como modelagem e construção de *APIs RESTful*.

Visando automatizar a programação/configuração do *backend*, surgiu o serviço denominado BaaS (*Backend as a Service*). Esse fica responsável por fornecer ao desenvolvedor os recursos que ele precisa para construir sua aplicação, incluindo servidores, banco de dados escaláveis e funcionalidades como notificações *push*, além de englobar as diversas características muitas vezes exigidas pelo usuário final, pois os serviços que disponibilizam esse modelo apresentam alta estabilidade, escalabilidade, otimização e segurança nos dados, além de possuírem diversos outros serviços integrados. Ou seja, serviços que antes eram realizados em locais mais remotos e usando papel para fazer o registro de informações por não se ter à disposição tecnologia móvel e acesso sem fio, hoje podem contar com recursos importantes para tornar viável seu desenvolvimento e uso efetivo por parte de um grande número de usuários. Como resultado, os usuários têm à disposição aplicativos que estão disponíveis em qualquer hora e lugar, fornecendo informações com grande agilidade.

Tendo essas premissas definidas, esse trabalho propôs fazer uma análise entre serviços que utilizam o modelo BaaS, apontando suas características para melhor definir uma ferramenta apropriada para desenvolvimento de um estudo de caso. Assim, como resultado, foi desenvolvido um sistema para Controle de Rotas de leite com integração ao ERP Administrador, utilizando-se do serviço escolhido.

1.1 OBJETIVOS

Este trabalho de conclusão tem como objetivo geral fazer uma análise entre serviços que utilizam o modelo BaaS, e como resultado disto desenvolver um sistema para controle de rotas de leite com integração ao ERP Administrador, utilizando-se do objeto de estudo.

A partir do objetivo geral, definiram-se os seguintes objetivos específicos:

- Fazer um levantamento de serviços que utilizam o modelo BaaS;
- Analisar características e funcionalidades de cada serviço abordado;
- Definir a estrutura dos dados para armazenamento no BaaS;

- Desenvolver um sistema mobile de controle de rotas de leite;
- Implementar a sincronização de dados com o ERP;
- Avaliar os ganhos obtidos com o uso deste modelo.

1.2 ORGANIZAÇÃO DO TEXTO

No Capítulo 2, Referencial Teórico, são tratados os conceitos básicos utilizados no projeto, como Computação em nuvem e BaaS, além de conter algumas informações sobre três serviços que utilizam deste modelo. No Capítulo 3, referente a modelagem do sistema, é apresentada uma descrição sobre o sistema definido para o estudo de caso, bem como, seus requisitos, como foi estruturado, diagrama de classes referente ao armazenamento no dispositivo e o diagrama de casos de uso com suas descrições. Além disso, o capítulo aborda como o sistema utiliza o BaaS, bem como quais ferramentas foram utilizadas para o seu desenvolvimento. No Capítulo 4, sobre a implementação do estudo de caso, é abordado como foi o processo de desenvolvimento, como cada parte integrante do sistema foi construída e o formato de funcionamento. Por fim, nas considerações finais são relatados os resultados obtidos ao se utilizar o BaaS, as principais dificuldades encontradas e quais são os trabalhos futuros a serem desenvolvidos.

2 REFERENCIAL TEÓRICO

Neste capítulo são abordados conceitos essenciais para o desenvolvimento do projeto, tais como Computação em Nuvem e BaaS, além de três serviços do modelo BaaS.

2.1 Computação em nuvem

O processo de evolução da tecnologia como um todo é inegável, novas funcionalidades, modelos, padrões e arquiteturas surgem constantemente. Necessidades como processamento e distribuição de dados se tornaram o cotidiano de várias empresas, e o custo para a manutenção dessas tarefas, além da dificuldade na escalabilidade dos sistemas que as executam, demandaram a criação de facilitadores para estas necessidades. Neste contexto, o termo de Computação em Nuvem começou a se tornar recorrente. Como cita Vandresen e Magalhães (2013):

> Na prática a Computação em Nuvem seria a transformação dos sistemas computacionais físicos de hoje em sistemas virtuais. Toda a infraestrutura necessária para o processamento, conectividade e armazenamento desses sistemas, aplicações e serviços, devem ser de responsabilidade de uma empresa que preste tal serviço. Isso torna possível uma grande economia quanto a suporte técnico e equipamento.

Com o surgimento da Computação em Nuvem, houve a necessidade da criação de modelos para padronizar esse serviço, dividindo cada um com base nas funcionalidades que possuem. Dentre os modelos, destaca-se:

- SaaS(Software as a Service): são aplicações implantadas sobre uma infraestrutura da nuvem, utilizada por consumidores. São acessadas a partir de interfaces, seja por um navegador ou por um dispositivo móvel. Neste modelo, o consumidor não gerencia a infraestrutura, nem os serviços que rodam sobre ela, e só são capazes de fazer configurações específicas sobre a aplicação que está utilizando. (Vieira, 2016)

- PaaS (Platform as a Service): neste modelo de serviço, o usuário não administra a infraestrutura que irá utilizar, ficando a cargo do provedor esta tarefa.

Porém, o usuário tem controle total sobre as aplicações implantadas, baseados na estrutura disponibilizada, como linguagens de programação e bancos de dados. (Vieira, 2016)

- IaaS (Infrasctruture as a Service): neste modelo, o provedor fornece toda a infraestrutura necessária para um serviço rodar, como servidores, rede e armazenamento. Com isso, o usuário não controla essa infraestrutura, porém, ele é capaz de controlar quais sistemas operacionais serão utilizados, como será o armazenamento e também quais os aplicativos que devem estar rodando. Este modelo de serviço é responsável por prover a infraestrutura para os outros modelos anteriormente citados, SaaS e PaaS. (Vieira, 2016)

- BaaS (*Backend as a Service*): Consiste basicamente no fornecimento de serviços como armazenamento de dados e arquivos e manutenção de usuários, além de ferramentas que facilitem o desenvolvimento, como notificações push, abstraindo toda a complexidade na implantação dos mesmos (RODRIGUEZ, 2015).

2.2 BaaS

O BaaS (*Backend as a Service*), termo que se refere a um modelo de Computação em Nuvem, tem se tornado um modelo emergente no desenvolvimento de aplicações. O fato de várias destas aplicações acessar e salvar dados na nuvem tornou-se um meio para a disseminação deste modelo de serviço. Com ele, tarefas como armazenamento de dados e arquivos, bem como manutenção de usuários e integração com redes sociais se torna abstrato, sem a necessidade de desenvolvimento do lado do servidor, tornando essa etapa ágil em relação ao modelo tradicional (no qual se utiliza uma linguagem de programação no lado do servidor, como o *PHP*, por exemplo). *“O objetivo claro do BaaS é fazer é fazer com que desenvolvedores mantenham o foco em gerar valor agregado ao seu aplicativo sem ter que se preocupar com o backend, economizando com infra-estrutura e administração de sistemas*.” (RODRIGUEZ, 2015)

Alexakis (2014) cita as funcionalidades essenciais para serviços que disponibilizam desta modelagem de computação:

- Armazenamento de dados baseado em nuvem: Considerado como uma das funcionalidades mais populares do BaaS, ele permite um armazenamento em nuvem

em que seja possível operações de escrita, alteração, leitura e exclusão de dados, sem a necessidade de um banco de dados local. Pode compreender também funcionalidades de direito de acesso aos dados salvos no banco.

- Gerenciamento de usuários: Possibilitar o cadastro e a manutenção de usuários nas aplicações que utilizam do serviço, fornecendo diversas formas de cadastro, que partem desde o tradicional cadastro com e-mail e senha, até vinculações com contas em redes sociais.

- Notificações via *push*: Possibilita o envio de notificações no formato *push*, possuindo também a possibilidade de o usuário personalizar quais notificações receber.

- *Push triggers*: É uma ferramenta que possibilita a automatização no envio das notificações *push*. Com ele, triggers são disparadas a partir de condições pré-estabelecidas pelo desenvolvedor, que irão enviar as notificações para o público alvo da mesma.

- Gerenciamento de arquivos: Esta funcionalidade compreende a possibilidade no armazenamento de mídias diversas na Nuvem, sem a necessidade de um armazenamento local das mesmas.

Segundo Lane (2013), o desenvolvimento de aplicações utilizando o BaaS apresenta diversas vantagens, onde destaca-se:

- Ganhos de eficiência;
- Agilidade na disponibilização para o mercado;
- Entrega dos aplicativos com um investimento menor em recursos;
- Otimização para smartphones e tablets;
- Infraestrutura segura e escalável;
- Possui diversos serviços comuns em aplicativos, reunidos em um único produto.

Atualmente existem diversos serviços que disponibilizam esta abordagem, tais como Firebase, Kinvey, Telerick Backend Services, Kumulos, back4app. Nas próximas sessões serão detalhados três destes serviços: Firebase, Kinvey e Kumulos.

2.2.1 Firebase

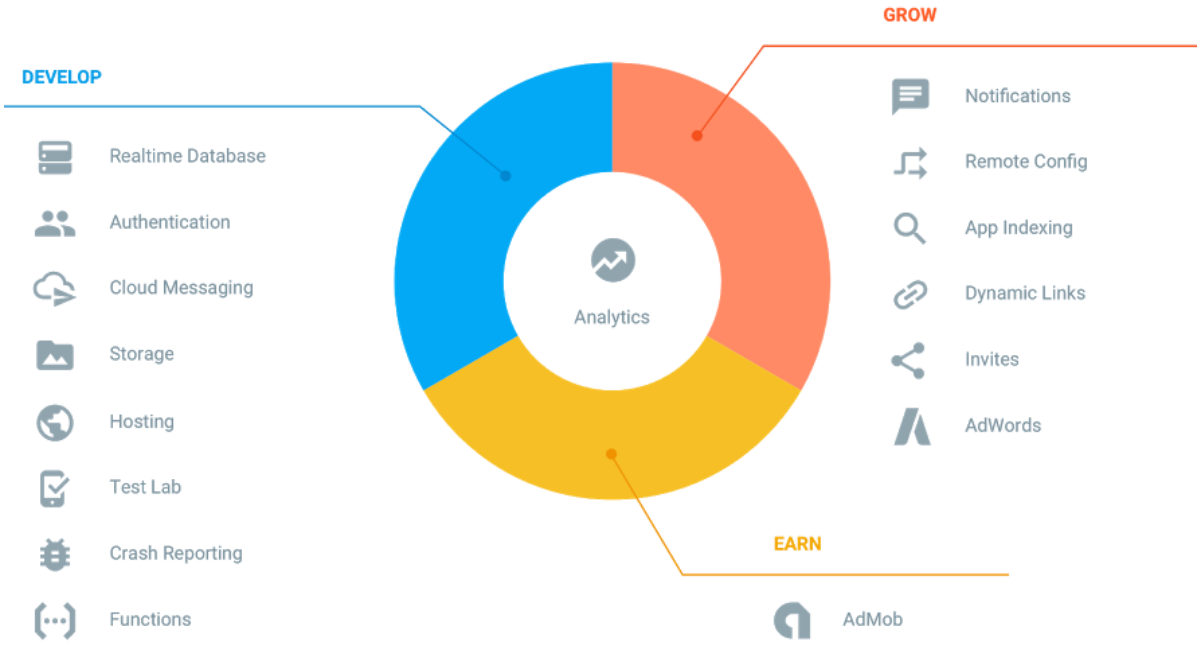
O Firebase é um serviço mantido atualmente pela Google e concebido com base no modelo BaaS, que provê diversas funcionalidades para o desenvolvedor.

> Firebase é um serviço criado em 2009 para auxiliar desenvolvedores de aplicações web e móveis a se preocuparem menos com a arquitetura da aplicação, sistemas de autenticação e hospedagem para dar ênfase ao produto final e gerar uma aplicação de qualidade rapidamente. (FIREBASE, 2009ª *apud* Lupchinski, 2015).

Dentre as principais vantagens na utilização do Firebase, destaca-se a diversidade nos serviços que disponibiliza, como por exemplo, armazenamento de dados, análise de uso e de erros, monetização no aplicativo (utilizando o *AdMob*), notificações *push* (que disponibiliza o envio destas notificações a partir de solicitações no formato *HTTPS*), autenticação e a alta disponibilidade dos serviços. Além disso, Moribe (2016) destaca: *"Outra grande vantagem do Firebase é que ele não se limita apenas para aplicativos Mobile (iOS e Android), você pode desenvolver aplicações Web (Javascript ou Node.js) ou mesmo integrar algumas de suas funcionalidades com seu Backend já existente (Node.js ou Java).*

Na Figura 1 são apresentados os serviços que o Firebase disponibiliza:

Figura 1: Serviços disponibilizados pelo Firebase.



Fonte: (FIREBASE, 2017)

Dentre estes serviços, destacam-se (FIREBASE, 2017):

Realtime Database: é um banco de dados não estruturado, em que os dados são armazenados em uma estrutura no formato JSON; e são sincronizados em tempo real;

Authentication: é um sistema de autenticação que suporta métodos de acesso no formato e-mail e senha, Facebook, GitHub, Twitter e Google Sign-In;

Cloud Messaging: utilizado para o envio de notificações *push* para os usuários da aplicação;

Storage: é um serviço que permite o armazenamento de arquivos gerados pelos usuários na nuvem;

Cloud Function: são códigos que podem ser armazenados na nuvem, e são disparados automaticamente por recursos do próprio Firebase, ou por solicitações em *HTTPs*.

Para a utilização do Firebase, três planos de contratação são disponibilizadas. Em todos eles, alguns serviços podem ser utilizados sem restrição, como *Cloud Messaging* e *Authentication*. Os três planos são:

Spark: Este plano gratuito basicamente limita a 100 conexões simultâneas, 1 GB de armazenamento e 10 GB de download mensal para o *Realtime Database*, além de permitir ao *Storage* o armazenamento de 5GB, além de 1GB de download, 20.000 operações de upload e 50.000 operações de download diários;

Flame: Este plano possui um custo de $25 por mês, permite um número ilimitado de conexões simultâneas, 2.5 GB de armazenamento e 20 GB de download mensal para o *Realtime Database*, além de permitir ao *Storage* o armazenamento de 50GB, além de 50GB de download, 100.000 operações de upload e 250.000 operações de downloads diários;

Blaze: Neste plano não possui limite de utilização, porém, o custo é calculado com base no que foi gasto no mês.

2.2.2 Kinvey

Fundado por Sravish Sridhar, o Kinvey entrou no mercado de serviços BaaS no ano de 2011 (KINVEY, 2017). Esse serviço possui as principais características deste modelo de Computação em Nuvem, como escalabilidade e segurança. Além

disso, contempla bibliotecas com um conjunto consistente de recursos, abrangendo as principais arquiteturas de desenvolvimento nativas, híbridas e web atuais (KINVEY, 2017). Dentre os serviços disponibilizados pelo Kinvey, destacam-se: manutenção de usuários, armazenamento de dados e arquivos, notificações *push*, envio de e-mail e sms, serviços de localização e encriptação de dados.

Para utilização do Kinvey são disponibilizadas quatro modalidades de contratação do serviço:

Developer: modalidade gratuita, onde é possível utilizar os recursos básicos do Kinvey, ter no máximo um administrador cadastrado na plataforma, além de ter um limite de 1 GB para armazenamento de dados e um limite mensal de 1.000 usuários ativos;

Startup: tem um custo mensal de $200,00, onde é possível utilizar os recursos básicos do Kinvey, ter no máximo três administradores cadastrados na plataforma, além de ter um limite de 10 GB para armazenamento de dados e um limite mensal de 100.000 usuários ativos;

Business: com um custo de $2000,00 mensais, porém pagos anualmente, esta modalidade permite a utilização dos recursos básicos do Kinvey, possibilita que sejam cadastrados um número ilimitado de administradores, possui um limite de 30 GB para armazenamento de dados e um limite mensal de 100.000 usuários ativos;

Enterprise: apresenta todas as características das modalidades anteriores, além de possibilitar um armazenamento de dados de 500 GB. Para obter o custo da mesma, é necessário entrar em contato com os representantes do Kinvey.

2.2.3 Kumulos

Kumulos é uma plataforma de desenvolvimento, semelhante ao Firebase e ao Kinvey, que pretende agilizar o processo de desenvolvimento de um aplicativo, disponibilizando para isto alguns serviços, como por exemplo, notificações push, relatórios e análise que possuem informações relevantes de acessos e popularidade, além de armazenamento de dados (KUMULOS, 2017). Dentre suas principais características, destacam-se:

- A possibilidade na importação de bancos de dados já existentes;

- Permite desenvolvimento de aplicativos em diversas plataformas, seja ela Desktop (C#), Web (PHP e JAVA), e mobile (Cordova, Android, Swift);

- Configuração e gerenciamento de aplicativos facilitados, segundo o próprio site do serviço, é possível configurar a infraestrutura com apenas 5 etapas.

O custo para a implementação deste serviço irá variar dependendo de quais módulos serão utilizados, pois o custo individual é para cada módulo, ao invés de ser igual aos outros serviços estudados, que dependem da quantidade utilizada.

3 MODELAGEM DO SISTEMA

Tendo como base o estudo realizado sobre o modelo BaaS, optou-se por selecionar um dos serviços estudados para o desenvolvimento de um estudo de caso e aprofundar o conhecimento sobre a forma de utilização do serviço. Dessa forma, foi selecionado o Firebase por ter um foco para aplicações mobile e possuir uma versão gratuita. Também considerou-se que o Firebase atualmente tem um grande apoio da comunidade, ou seja, existe bastante informações e experiências sobre o serviço por estar sendo utilizado amplamente.

3.1 Escopo do Negócio

O estudo de caso tem como base a implementação de um aplicativo de coletas de leite. Todo o processo é basicamente realizado por duas pessoas. De um lado o funcionário da empresa, responsável pelo gerenciamento dos dados e dos recursos. Do outro, o motorista, responsável por realizar as coletas, baseados nas rotas informadas pelo funcionário da empresa.

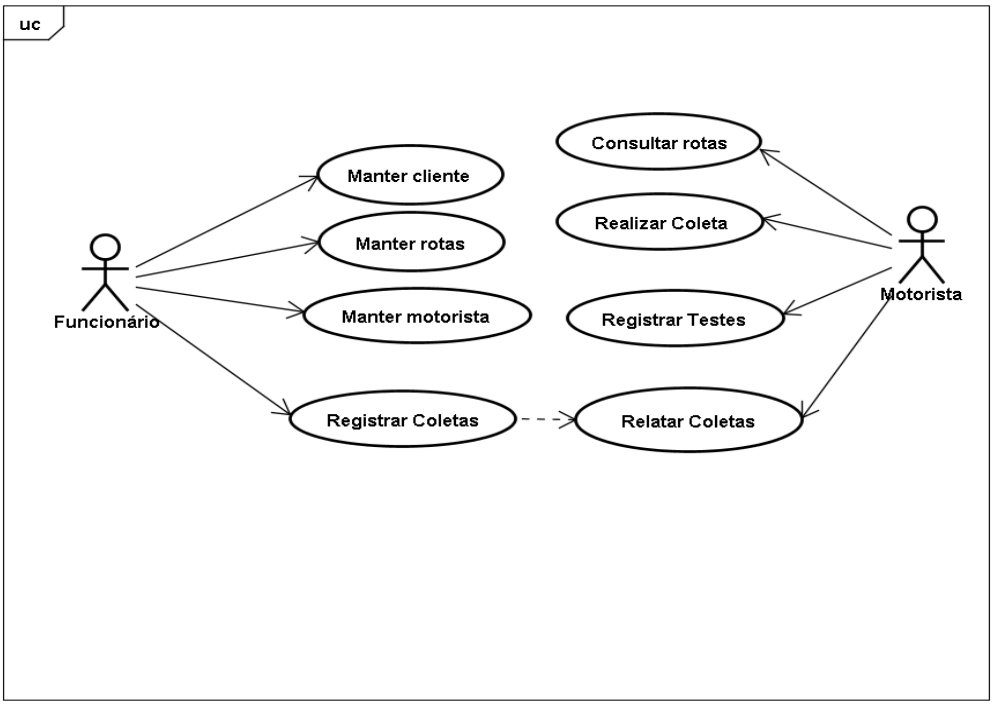
O funcionário fica responsável por cadastrar os clientes no sistema, informando a partir dos dados coletados, a qual rota cada cliente pertence. Após isso, é designado para cada motorista uma ou mais rotas, baseado principalmente na demanda da coleta e na localização de cada rota. Por fim, após o motorista retornar, o funcionário é responsável por lançar os dados referentes às coletas, informados pelo motorista, manualmente no sistema.

O motorista, em um primeiro momento, recebe a rota, e quais os clientes ele deve percorrer nela. Todo dia ele percorre a rota delimitada, realizando as coletas do leite nas propriedades previamente informadas. Durante a coleta, alguns testes são realizados e os resultados anotados junto às outras informações da coleta, a fim de testar a qualidade do produto. Os principais testes realizados são o de estabilidade ao Alizarol, para indicar a acidez do leite, e o de temperatura. Caso o produto não seja aprovado no teste, a coleta não é realizada, para evitar a contaminação da carga presente no caminhão. No final da rota, o motorista se dirige à empresa, onde descarrega o produto e informa os dados das coletas manualmente ao funcionário responsável.

Todos os dados dessas coletas são registrados no ERP. Esse foi desenvolvido com a ferramenta de desenvolvimento DELPHI (EMBARCADERO,2017), e utiliza-se para o armazenamento de dados, um banco de dados em Firebird, que roda localmente na empresa.

A Figura 2 apresenta o diagrama de Caso de Uso do Negócio que representa o fluxo atual do processo de coleta e suas interações com os atores.

Figura 2**:** Representação do Diagrama de Casos de Uso atual



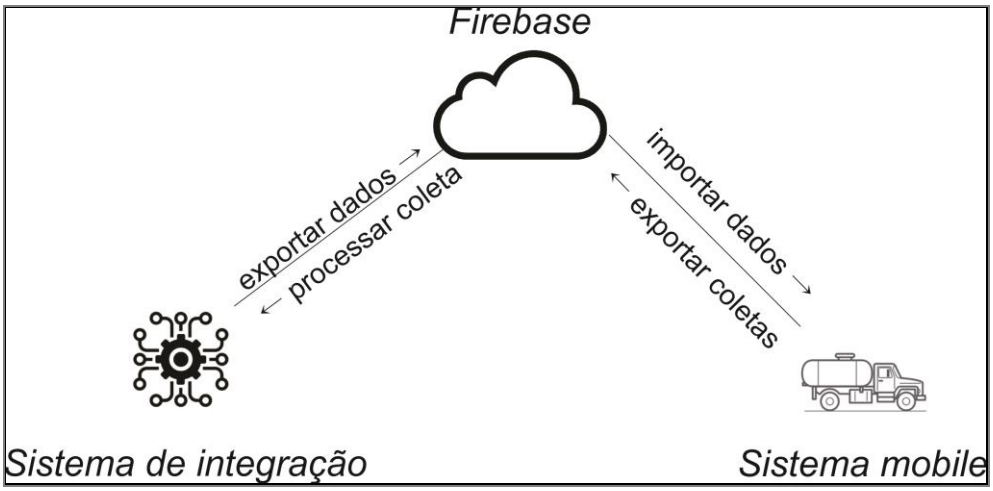
Fonte: (Autoria Própria, 2017)

3.2 Requisitos funcionais e não funcionais do sistema

Como citado anteriormente, o estudo de caso se baseia em um sistema de coletas de leite, em que o motorista é responsável basicamente por registrar suas coletas no aplicativo, e o funcionário da empresa é responsável por prover as informações para que a aplicação funcione, e processar os dados disponibilizados pelo aplicativo. Para a comunicação dos dados entre os dois extremos, ERP e motorista, é utilizado o Firebase, que provê os serviços básicos para o funcionamento.

A Figura 3 apresenta a arquitetura do sistema de coleta com a solução proposta e demonstra como os componentes do sistema estão integrados e como ocorre o fluxo de dados entre esses componentes.

Figura 3: Representação da estrutura das coletas



Fonte: (Autoria Própria, 2017)

A seguir são listados os requisitos funcionais da aplicação de integração:

- Exportar os dados do ERP;
- Processar as coletas.

Como requisitos funcionais para o aplicativo, pode-se citar como essenciais:

- Importar os dados da nuvem;
- Consultar rotas;
- Consultar coletas;
- Registrar coletas;
- Exportar coletas.

Para os requisitos não funcionais da solução proposta, destaca-se:

- O aplicativo deve ser multiplataforma;
- Os dados do aplicativo devem ser exportados para a nuvem;
- Permitir no aplicativo o armazenamento dos dados off-line;
- A aplicação de integração dever ser compatível com o banco de dados Firebird.

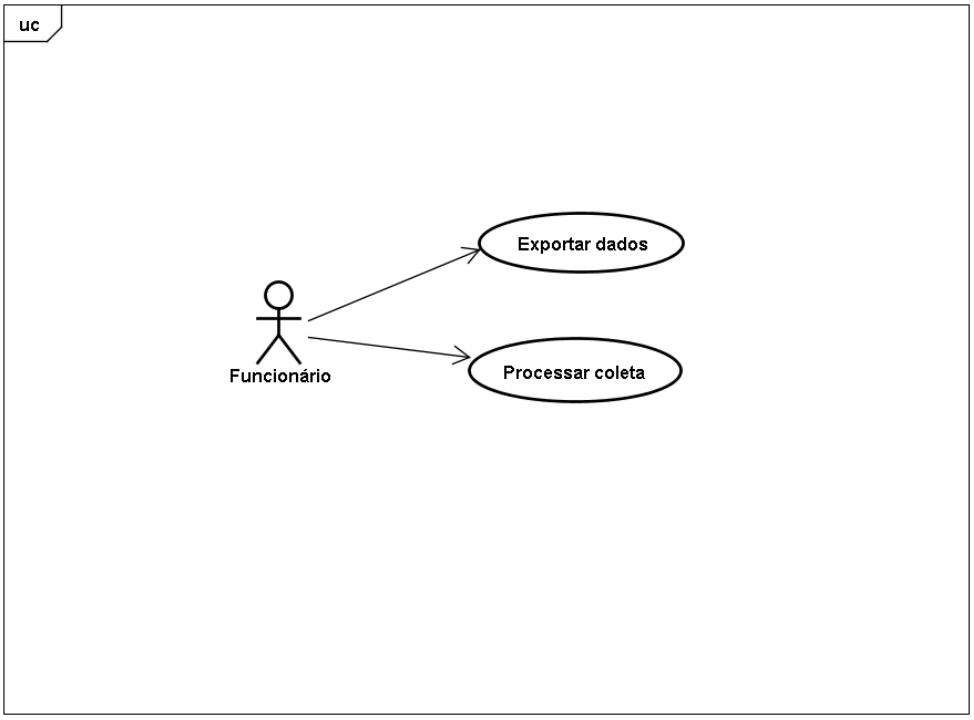
3.3 Aplicação de Integração

Esta seção apresenta a modelagem para o sistema de integração.

3.3.1 Diagrama de Casos de Uso da aplicação de integração

No diagrama de Casos de Uso da aplicação de integração é abordado o fluxo de integração entre o ERP Administrador e o sistema de armazenamento. Na Figura 4 está presente este diagrama:

Figura 4: Representação do Diagrama de Caso de Uso da aplicação de integração



Fonte: (Autoria Própria, 2017)

3.3.1.1 Descrição dos Casos de Uso da aplicação de integração

Caso de uso: Exportar dados

Nome do Caso de Uso	Exportar dados
Caso de Uso Geral	
Ator Principal	Funcionário
Atores Secundários	

Resumo	Neste caso de uso, o funcionário exporta para a nuvem os dados necessários para o aplicativo móvel. São eles: clientes, rotas e motoristas.
Pré-Condições	Conexão com a internet
Pós-Condições	Dados Exportados
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a exportação dos dados.	
	2. Selecionar os dados não sincronizados anteriormente.
	3. Realiza a exportação dos dados com o Firebase.
	4. Gerar um log das alterações realizadas no Firebase, para posterior sincronização com o aplicativo.

Caso de uso: Processar coleta

Nome do Caso de Uso	**Processar coleta**
Caso de Uso Geral	
Ator Principal	Funcionário
Atores Secundários	
Resumo	Neste caso de uso, o sistema sincroniza as últimas coletas realizadas e disponibilizadas pelo motorista, para que o funcionário possa importar definitivamente as mesmas.
Pré-Condições	Conexão com a internet
Pós-Condições	Coletas processadas
Fluxo Principal	

Ações do Ator	Ações do Sistema
1. Solicitar a sincronização das coletas pendentes.	
	2. Consulta no Firebase e sincroniza com o banco de dados local as coletas não sincronizadas anteriormente.

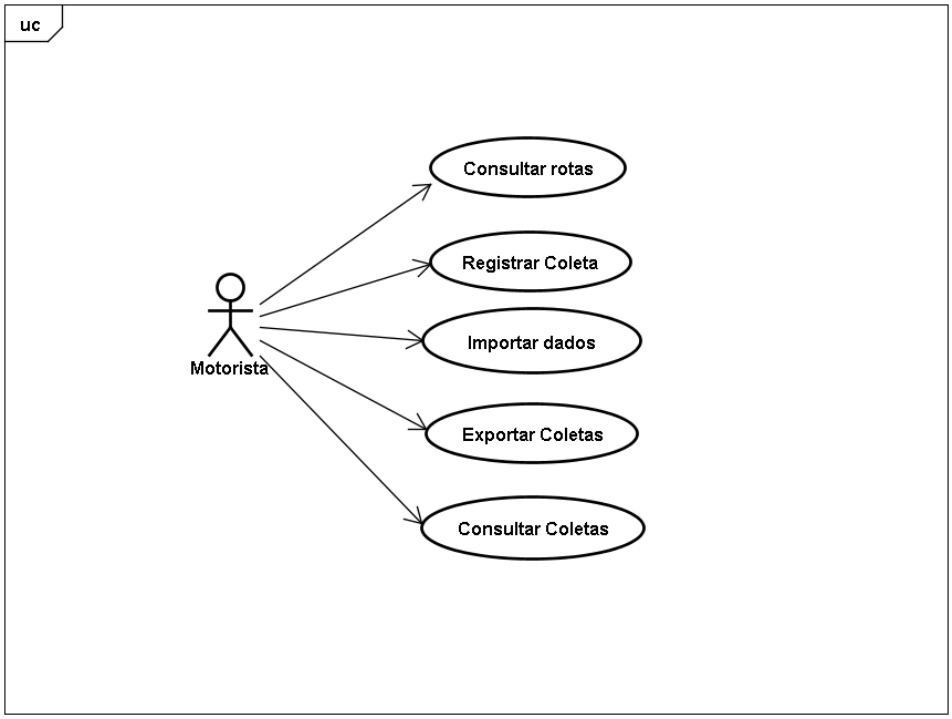
3.4 Aplicação Mobile

Esta seção apresenta a modelagem para o sistema mobile.

3.4.1 Diagrama de Casos de Uso da aplicação mobile

No diagrama de Casos de Uso da aplicação mobile é abordado o fluxo da coleta de leite com a utilização do sistema. Na Figura 5 está presente este diagrama:

Figura 5**:** Representação do Diagrama de Casos de Uso da aplicação mobile



Fonte: (Autoria Própria, 2017)

3.4.1.1 Descrição dos Casos de Uso da aplicação mobile

Caso de uso: Consultar rotas

Nome do Caso de Uso	Consultar rotas
Caso de Uso Geral	
Ator Principal	Motorista
Atores Secundários	
Resumo	Neste caso de uso, o motorista consulta as rotas e seus clientes vinculados, as quais está designado a realizar o processo.
Pré-Condições	Possuir rotas designadas ao motorista
Pós-Condições	Lista de clientes da rota
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a relação de rotas.	
	2. Exibir uma lista com as rotas designadas ao motorista.
3. Selecionar uma rota.	
	4. Exibir a lista de clientes associados a rota

Caso de uso: Registrar coleta

Nome do Caso de Uso	Registrar coleta
Caso de Uso Geral	
Ator Principal	Motorista
Atores Secundários	
Resumo	Neste caso de uso, o motorista escolhe uma rota, para então realizar as coletas da rota, informando

	algumas informações sobre cada coleta, como quantidade de leite e os resultados dos testes.
Pré-Condições	Possuir uma rota designada ao motorista
Pós-Condições	Coletas registradas
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a relação de rotas.	
	2. Exibir uma lista com as rotas designadas ao motorista.
3. Selecionar uma rota para iniciar a coleta.	
	4. Iniciar uma nova coleta, atribuindo os dados de data e caminhão.
	5. Exibir a lista de clientes na rota.
6. Selecionar um cliente.	
	7. Exibir tela para a informação dos dados da coleta para aquele cliente.
8. Informar os dados da coleta e solicitar a finalização da coleta do cliente selecionado.	
	9. Registrar os dados da coleta e exibir a lista dos clientes remanescentes na rota.

Caso de uso: Importar dados

Nome do Caso de Uso	Importar dados
Caso de Uso Geral	
Ator Principal	Motorista
Atores Secundários	
Resumo	Neste caso de uso, o motorista sincroniza os dados com a nuvem,

	importando os dados não sincronizados. São eles: clientes, rotas e motoristas.
Pré-Condições	Conexão com a internet
Pós-Condições	Dados Importados
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a importação dos dados.	
	2. Selecionar os dados não sincronizados anteriormente.
	3. Realiza a importação dos dados presentes no Firebase.

Caso de uso: Exportar coletas

Nome do Caso de Uso	**Exportar coletas**
Caso de Uso Geral	
Ator Principal	Motorista
Atores Secundários	
Resumo	Neste caso de uso, o motorista exporta as coletas realizadas para a nuvem, para que possam ser processadas posteriormente.
Pré-Condições	Conexão com a internet
Pós-Condições	Coletas exportadas
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a exportação das coletas.	
	2. Selecionar os dados não sincronizados anteriormente.
3. Selecionar as coletas.	
	4. Realiza o envio das coletas para a nuvem.

Caso de uso: Consultar coletas

Nome do Caso de Uso	Consultar coletas
Caso de Uso Geral	
Ator Principal	Motorista
Atores Secundários	
Resumo	Neste caso de uso, o motorista consulta as coletas previamente realizadas por ele.
Pré-Condições	Possuir coletas realizadas
Pós-Condições	Lista de coletas realizadas
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Solicitar a consulta das coletas.	
	2. Exibir uma lista com as coletas realizadas.
3. Selecionar uma coleta.	
	4. Exibir as informações de cada cliente coletado.

3.4.2 Diagrama de Classes

O diagrama de classes apresentado na Figura 6 representa como se dispõem os elementos da aplicação mobile, para que seja possível ver a ligação entre eles.

Com o diagrama de classes pode ser visualizado as relações entre cada elemento do sistema. Abaixo são detalhadas as classes pertencentes ao diagrama:

- Rota: Basicamente na classe rota é definido um nome para a rota, além de definir qual é o caminhão que atualmente realiza a rota;

- RotaCliente: na classe são definidos quais clientes estão presentes na rota, além de informar qual a ordem de coleta é a indicada;

- Caminhão: são definidos os dados essenciais do caminhão, como placa e id. É Importante afirmar que um caminhão pode percorrer mais de uma rota no mesmo dia;

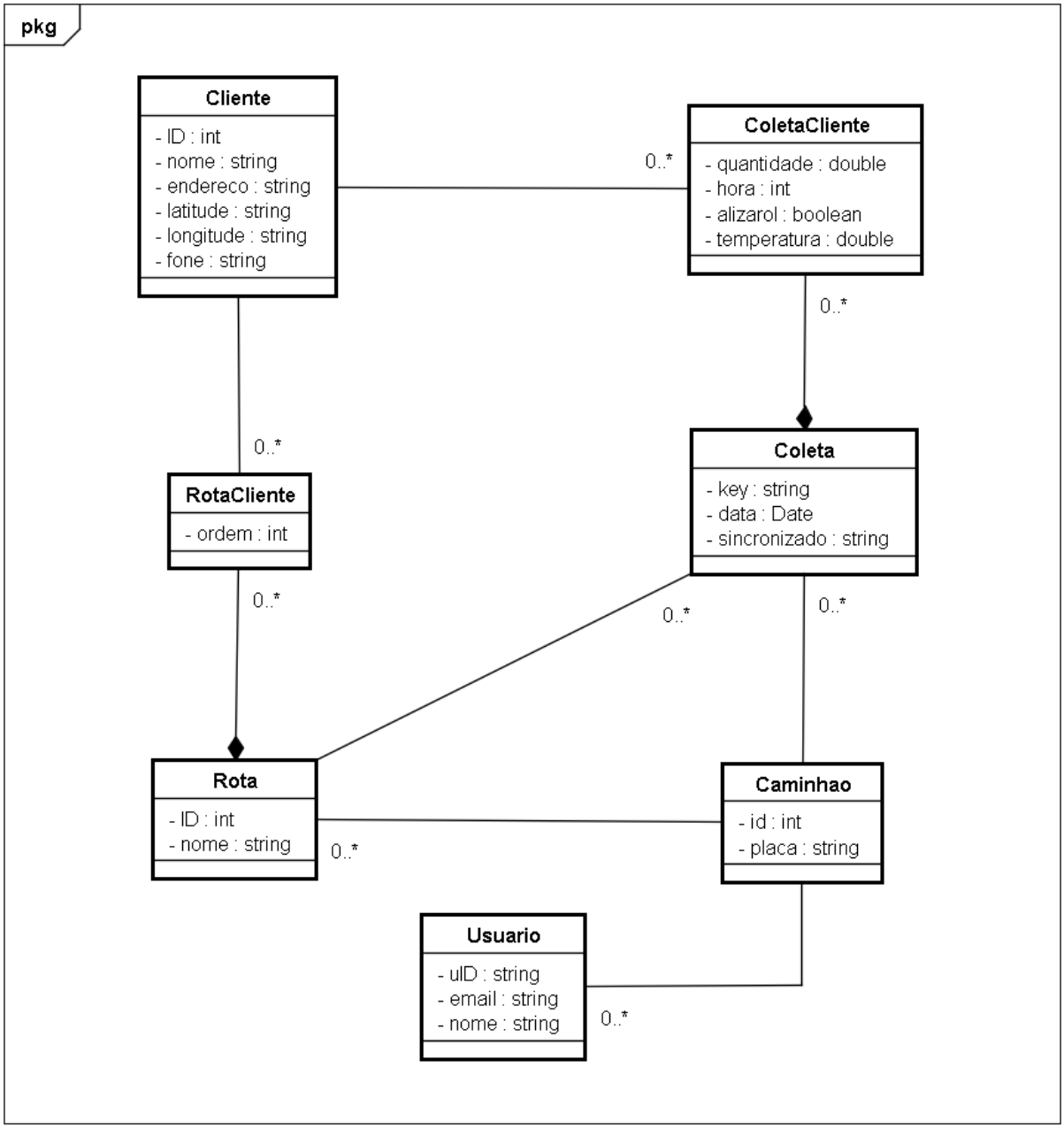
- Usuario: com essa classe é possível visualizar os dados básicos de cada usuário, que são email, nome, e uID, que é o código do mesmo. Também é definido qual o caminhão este usuário pertence;

- Cliente: define os dados de cada cliente, como nome, endereço e telefone, além da localização, através da latitude e longitude;

- Coleta: essa classe é a efetivação da rota planejada, ou seja, guarda as informações de uma coleta da rota em uma data específica. O campo sincronizado desta classe é responsável por determinar se a coleta foi enviada para o Firebase;

- ColetaCliente: são informados os dados referentes as coletas de cada cliente, com informações como horário da coleta, quantidade e testes realizados.

Figura 6: Representação do Diagrama de Classes da aplicação mobile



Fonte: (Autoria Própria, 2017)

3.5 Backend

O Realtime Database é o serviço de armazenamento de dados disponibilizado pelo Firebase. Esse possibilita que dados sejam atualizados de forma automática e instantânea em todos os dispositivos conectados ao banco quando ocorrerem atualizações neles. Como cita LUPCHINSKI(2015), este banco de dados se caracteriza por não ser estruturado, em que as manipulações realizadas nos dados

não seguem necessariamente um padrão, possibilitando que a estrutura de armazenamento dos dados seja diferente entre cada registro, dependendo da necessidade.

Sobre o banco de dados não estruturado, MELO (2016) explica:

> Bancos de dados não-relacionais tem como principal objetivo a alta escalabilidade e o alto desempenho alinhados com um baixo custo de manutenção e armazenamento de dados. Análogo à tabela em bancos de dados relacionais, no mundo não-relacional temos coleções. Essas coleções são compostas por objetos baseados no JavaScript Object Notation (JSON), onde possui uma estruturação simples de chave e valor.

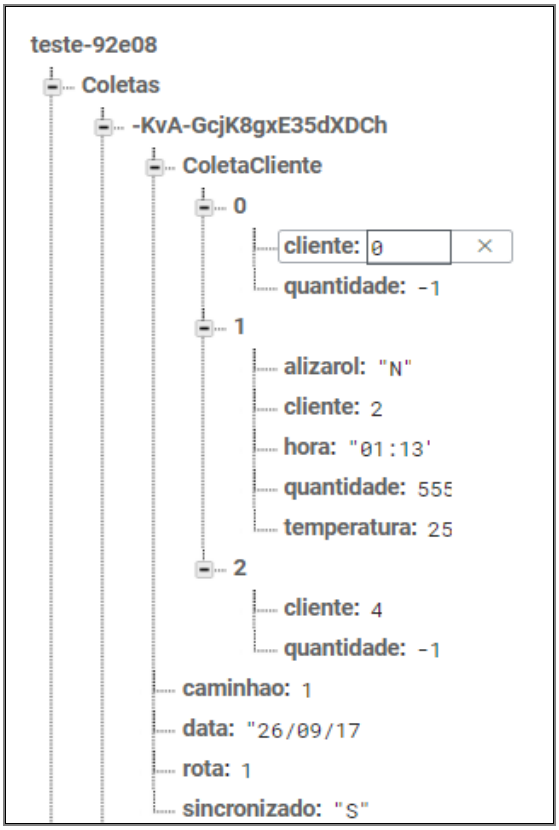
Para a modelagem do banco, LUPCHINSKI (2015) destaca que ela é baseada nas consultas que serão realizadas. Partindo destes conceitos, o modelo final para armazenar os dados do aplicativo é o seguinte, como mostra a Figura 7.

Figura 7: Representação final do modelo do banco de dados.

Fonte: (Autoria Própria, 2017)

Já na Figura 8 é demonstrada a estrutura de armazenamento dos dados referentes às coletas.

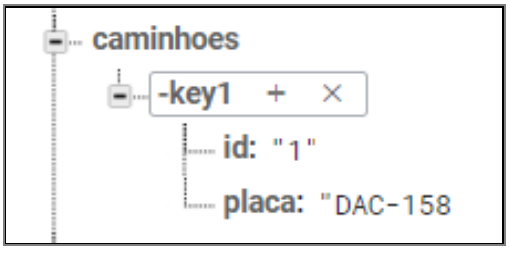
Figura 8: Representação da estrutura das coletas.



Fonte: (Autoria Própria, 2017)

A estruturação dos caminhões cadastrados é visualizada na Figura 9:

Figura 9**:** Representação da estrutura dos caminhões.



Fonte: (Autoria Própria, 2017)

Para o armazenamento dos usuários cadastrados no Firebase é utilizada a estrutura inclusa no nó *Users.* Esta estrutura é preenchida utilizando as *triggers* do

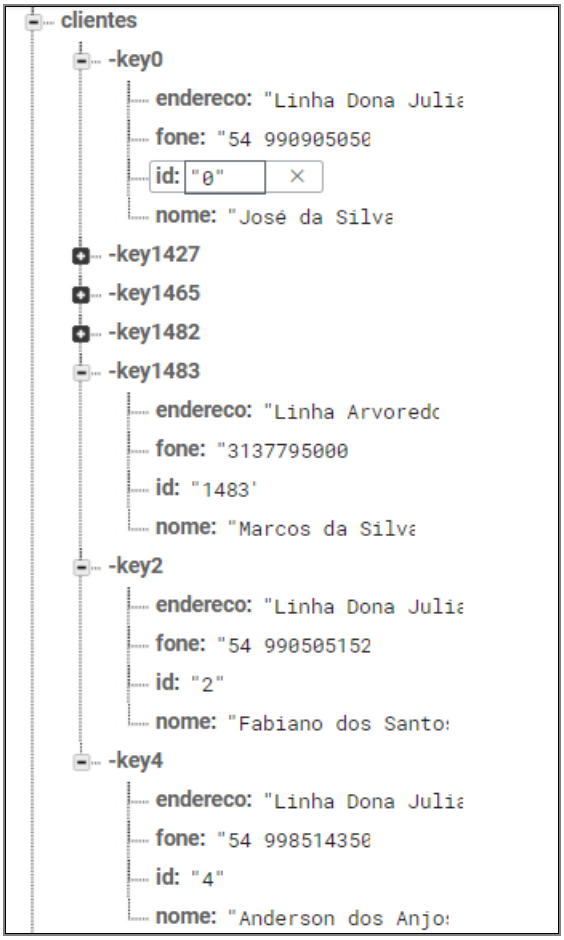
próprio Firebase, disparada no momento que um usuário é cadastrado. Na Figura 10 é demonstrada como é a estrutura final deste elemento:

Figura 10: Representação da estrutura de Users.

Fonte: (Autoria Própria, 2017)

Na Figura 11 está representada a estrutura a ser utilizada para armazenar os dados dos clientes:

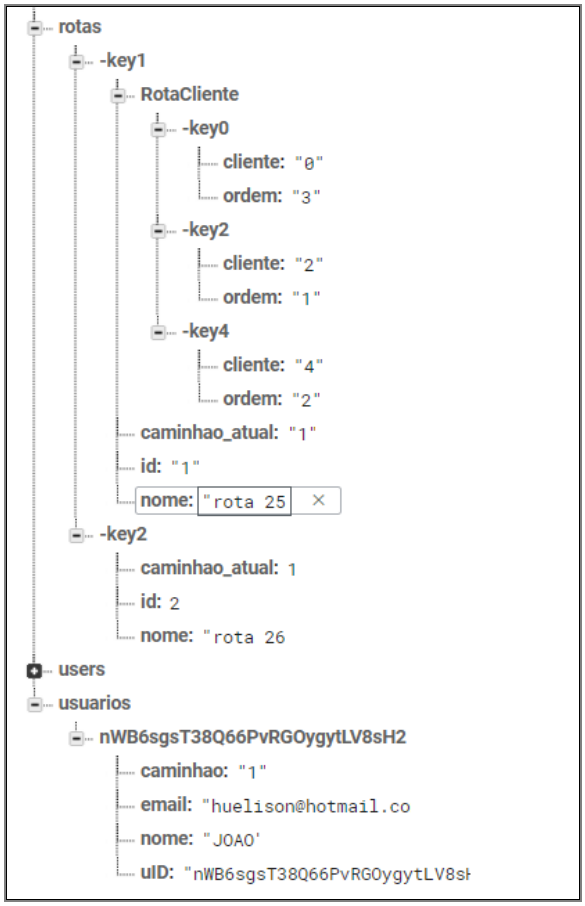
Figura 11: Representação da estrutura dos clientes.



Fonte: (Autoria Própria, 2017)

Por fim, na Figura 12, está a estrutura das rotas e dos usuários do banco de dados:

Figura 12: Representação da estrutura das rotas e dos usuários.



Fonte: (Autoria Própria, 2017)

Além disso, devido ao fato que o aplicativo poderá funcionar por um longo tempo sem conexão à Internet, os dados são armazenados em um banco de dados local, que é sincronizado com o banco de dados do Firebase quando necessário.

3.5.1 Autenticação de Usuário

Como mecanismo de segurança, o Firebase disponibiliza diferentes formas de autenticação ao seu serviço, como por exemplo, e-mail ou Facebook. Optou-se em utilizar o método padrão de usuário e senha disponibilizado pelo Firebase, em que o e-mail e a senha podem ser cadastrados e utilizados posteriormente para o *login*. Esse cadastro recebe um código que pode ser utilizado como referência ao usuário, e com base neste código gerado é possível determinar dentro do sistema a qual

caminhão este motorista está vinculado. Poderão ser verificados também quais usuários estão cadastrados no sistema com base na coleção usuários mostrada na Figura 12.

Como o dispositivo pode permanecer por um longo tempo sem conexão à internet, após o primeiro login, o usuário permanecerá conectado para fazer as operações necessárias, necessitando apenas fazer esse procedimento novamente no momento da sincronização dos dados ou da exportação das coletas, caso o acesso antigo tenha expirado no Firebase.

3.6 Ferramentas de Desenvolvimento

O aplicativo mobile foi desenvolvido utilizando um framework para desenvolvimento de aplicativos híbridos denominado IONIC (IONIC,2017). Ele foi criado em 2013 para o desenvolvimento de aplicativos híbridos e incorpora o AngulaJS (ANGULAR,2017), responsável pelo gerenciamento da arquitetura MVC, e o Apache Cordova (APACHE CORDOVA, 2017), que disponibiliza as bibliotecas para acesso a recursos nativos dos dispositivos móveis (SOBRAL, 2015).

O desenvolvimento da interface utilizando o IONIC é feita basicamente com a linguagem de marcação HTML, e pela linguagem de estilo CSS, além de se utilizar componentes disponibilizados pelo próprio framework, que são responsáveis pela padronização na visualização do aplicativo em diferentes sistemas operacionais. Em relação ao HTML e ao CSS, LUPCHINSKI (2015) explica a utilidade de cada um destes:

> HTML e CSS são as ferramentas básicas para gerar páginas Web estáticas. A linguagem de marcação HTML é reconhecida por todos os navegadores modernos utilizando o paradigma WYSIWYM, mostrando um resultado compilado e possivelmente diferente da forma como foi escrito. CSS entra no conjunto das ferramentas suportadas pelos navegadores modernos agindo como um conjunto de regras de estilo para certos componentes. Com tais regras é possível alterar totalmente a aparência de um documento, tornando a aplicação mais agradável para o usuário final.

Para suprir a necessidade do armazenamento de dados localmente no dispositivo mobile, foi utilizado um banco de dados em SQLite. Esse banco de dados funciona sem a necessidade de um SGBD para o gerenciamento dos dados, além

disso, ele está disponível na plataforma Android, retirando a necessidade da instalação do mesmo (SILVA, 2016).

Em relação ao aplicativo de integração, foi utilizada a ferramenta RAD Studio XE8 (EMBARCADERO,2017), desenvolvida pela Embarcadero, e que possibilita um desenvolvimento rápido de programas através da sua interface gráfica. Ele possui componentes que permitem a conexão de maneira fácil com o banco de dados Firebird, que é o tipo de banco utilizado pelo ERP Administrador, e de onde as informações serão utilizadas.

4 IMPLEMENTAÇÃO DO ESTUDO DE CASO

Nesta seção é detalhado o processo de desenvolvimento do estudo de caso, contemplando a aplicação de integração e a aplicação mobile.

4.1 Aplicação de integração

A aplicação de integração é responsável por fazer a integração entre a base de dados local do ERP e o Firebase. Para seu desenvolvimento foi utilizada a ferramenta Delphi XE8, juntamente com uma classe disponibilizada pela TDevRocks(2017), que foi alterada para atender as necessidades da aplicação, para realizar as requisições ao banco de dados e o cadastro de usuário no Firebase. Essa classe utiliza o protocolo HTTP para realizar a comunicação, onde, no momento da criação do objeto, é necessário informar a URL única que cada projeto do Firebase possui (essa URL pode ser obtida através do console que cada projeto possui no serviço). Para armazenar os dados, somente é necessário informar o local na árvore de dados da gravação, e o conteúdo em si, no formato json. Já para leitura de dados, mais de um método é disponibilizado, sendo que todos têm em comum o local na árvore de dados do objeto a ser lido. Para esta aplicação somente é necessário utilizar o método que retorna um objeto em json dos dados. Em relação à autenticação, a aplicação de integração fica responsável por cadastrar o usuário e vincular a este um motorista, para posterior sincronização.

Nas próximas seções são apresentados detalhes sobre o desenvolvimento dos principais componentes da aplicação de integração.

4.1.1 Exportação de dados

A exportação de dados é responsável por armazenar os dados no Firebase. Para isso, como citado anteriormente, é utilizada uma classe para a gravação dos dados no Firebase. A estrutura de cada entidade do projeto é gravada com base na estrutura citada anteriormente. Basicamente, é criado um objeto json com a lista dos dados a serem gravados. Cada objeto da lista é gravado tendo como chave o *id* que cada um possui no sistema, precedido do texto *"-key"*, pois o Firebase, quando recebe uma lista de objetos em que a chave de todos seja um valor inteiro, ele converte as mesmas para valores sequenciais. Os dados referentes aos clientes, as

rotas, aos motoristas e aos veículos são exportados. Na Figura 13 pode-se visualizar o código responsável tanto pela instanciação do objeto FComunicacao (que realiza a comunicação com o Firebase), bem como a implementação da exportação de clientes:

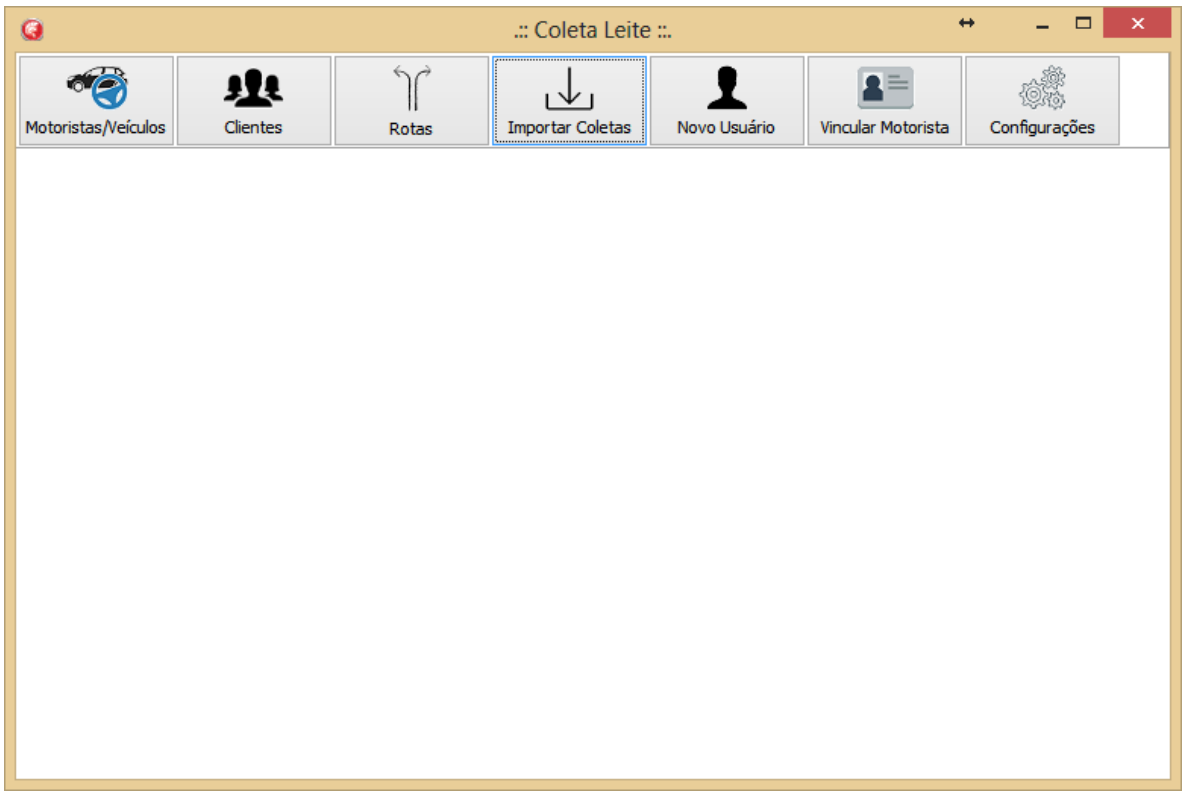
Figura 13: Código responsável pela instanciação do FComunicacao e pela exportação de clientes.

```
procedure TfrmPrincipal.FormCreate(Sender: TObject);
begin
  // na criação do formulário, o objeto TComunicação
  // é criado e recebe por parâmetro a url do projeto
  BaseURL := 'https://teste-92e08.firebaseio.com/';
  if not Assigned(FComunicacao) then
    FComunicacao := TComunicacao.Create(BaseURL);
end;

procedure TfrmPrincipal.gravarClientes;
var
  clientes, cliente_object: TJSONObject;
begin
  clientes := TJSONObject.Create; // a váriavel clientes irá armazenar
  dmPrincipal.qClientes.Close; // a lista de clientes que será gravada
  dmPrincipal.qClientes.open; // no Firebase
  dmPrincipal.qClientes.first;
  while not(dmPrincipal.qClientes.Eof) do
  begin
    // A variável cliente_object irá armazenar os dados de cada cliente,
    //ja no formato JSON
    cliente_object := TJSONObject.Create;
    cliente_object.AddPair('nome', dmPrincipal.qClientesNOME.AsString);
    if not(dmPrincipal.qClientesFONE.IsNull) then
      cliente_object.AddPair('fone', dmPrincipal.qClientesFONE.AsString);
    cliente_object.AddPair('endereco', dmPrincipal.qClientesENDERECO.AsString);
    cliente_object.AddPair('id', dmPrincipal.qClientesID.AsString);

    // Neste momento é criado um par de chave/valor em que a chave é composta por
    // -key + id do cliente
    clientes.AddPair(TJSONPair.Create('-key' + dmPrincipal.qClientesID.AsString,
                     cliente_object));

    dmPrincipal.qClientes.Next;
  end;

  // Com o evento GravaDados é possível salvar a lista no Firebase
  if (FComunicacao.GravaDados('clientes.json', clientes.ToString)) then
    ShowMessage('Sincronização de clientes realizada com sucesso!');
end;
```

Fonte: (Autoria Própria, 2017)

Cada entidade a ser exportada possui um botão na interface da aplicação de integração. Na Figura 14 pode ser visualizado o design final desta interface:

Figura 14: Representação da interface da aplicação de integração.



Fonte: (Autoria Própria, 2017)

4.1.2 Autenticação

Com a autenticação é possível definir quem pode acessar a aplicação mobile, bem como quais rotas estarão disponíveis para o usuário. A aplicação de integração é responsável por cadastrar o usuário no Firebase. Para tanto, faz uma requisição em HTTP, onde é enviado através do método POST um json com as seguintes chaves*: email, password* e *returnSecureToken.* Esta última chave, deve receber sempre o valor *"true"*, e retorna, em caso de sucesso, o identificador único do usuário no Firebase, além de algumas informações referentes ao login. Já em caso de erro, retorna o motivo do erro, como por exemplo, uma senha fraca. A requisição é feita ao endereço *"https://www.googleapis.com/identitytoolkit/v3/relyingparty /signupNewUser?key=vlr",* onde *"vlr"* deve ser substituído pelo *apiKey*, um identificador único por projeto fornecido pelo Firebase. Com o cadastro realizado, é possível vincular um determinado usuário do Firebase com um motorista cadastrado no ERP. No momento em que o cadastro é realizado, uma trigger no Firebase insere o *email* e o *uID* na base de dados, para ser utilizado no momento da vinculação.

4.1.3 Processar Coletas

Para que as coletas estejam disponíveis ao ERP é necessário processar elas dentro da aplicação de integração. Para isso, a aplicação faz uma consulta de todas as coletas não sincronizadas utilizando a classe de comunicação no elemento *Coletas* da árvore de dados do Firebase. Na url desta requisição são passados os parâmetros: *orderBy*, que é a propriedade para ordenar a consulta, neste caso a propriedade *sincronizada,* e o *equalTo*, que corresponde ao valor do filtro. Feito isso, a *string* de resposta é convertida em um objeto json, que é percorrido para inserir os dados na base de dados do ERP. Todas as coletas em que a quantidade coletada for maior que zero são importadas, e após isso, é alterado no Firebase, o valor da chave sincronizado de cada coleta para *"S",* para que não sejam reprocessados.

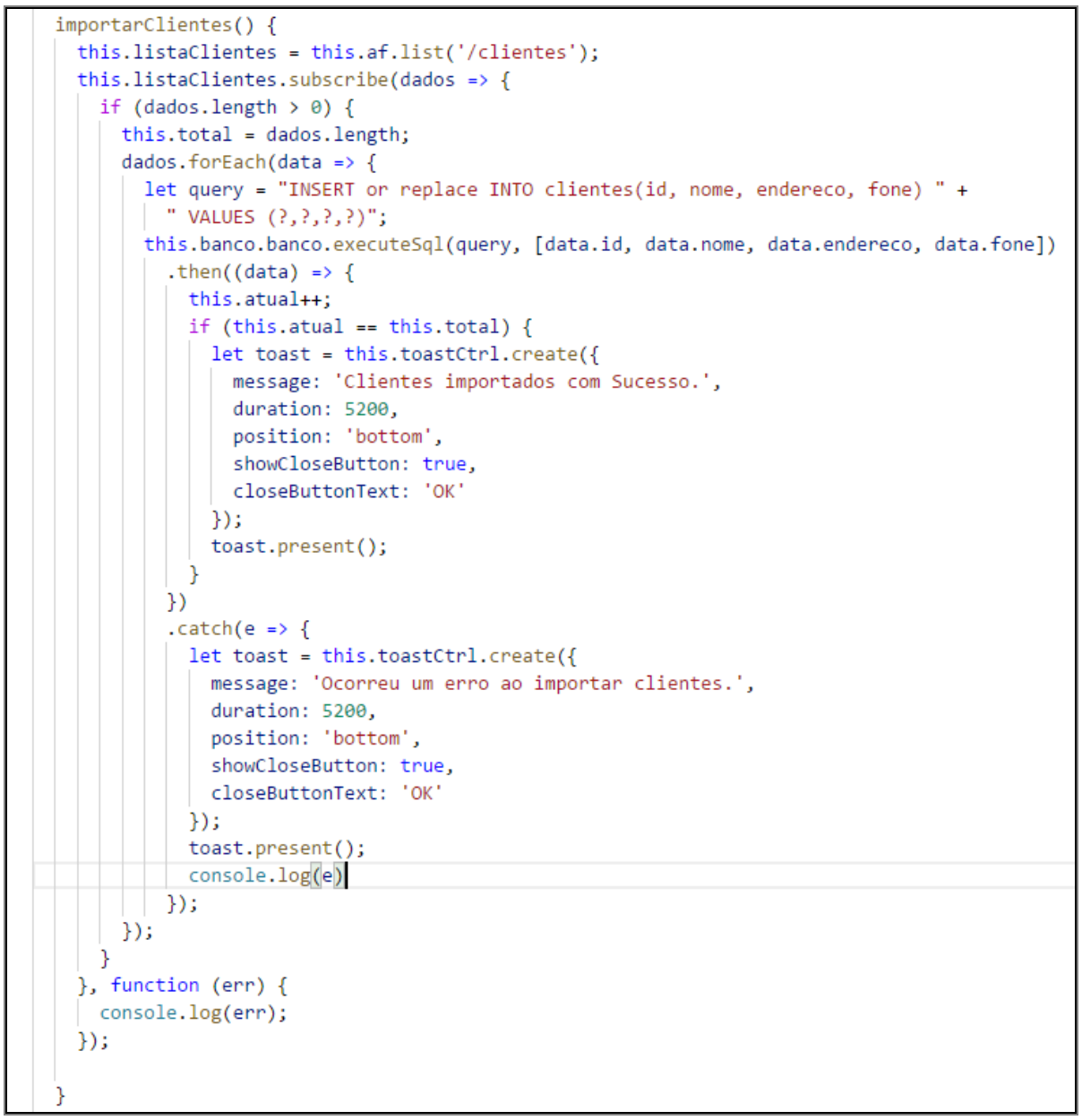
4.2 Aplicação Mobile

A aplicação mobile é responsável por possibilitar o acesso às informações das coletas para o registro dos dados, bem como listar outras diversas informações relevantes à aplicação. A aplicação foi desenvolvida utilizando-se o IONIC 3, e para as ações de acesso ao Firebase, foi utilizada a versão 4 da biblioteca AngularFire2, que abstrai toda a complexidade na implementação das ações, seja de acesso a banco ou autenticação. Como os dados devem estar disponíveis off-line, um banco de dados em SQLite foi utilizado. Este se baseia no diagrama de classes, e é utilizado para realizar todas as operações de consulta e armazenamento de dados dentro da aplicação.

4.2.1 Importação de Dados

Para que os dados que estão disponibilizados no Firebase possam ser utilizados pela aplicação mobile, é necessário fazer a importação dos dados disponíveis no serviço. Para isso, foi realizada a consulta dos dados utilizando-se do *AngularFireDatabase*. Devido ao fato de que as comunicações utilizando-se dessa biblioteca serem assíncronas, a gravação dos dados na base local é realizada dentro da resposta de sucesso da consulta. Na Figura 15 pode ser visualizada a implementação da importação dos clientes para o sistema:

Figura 15: Código responsável pela importação dos clientes.



```
importarClientes() {
  this.listaClientes = this.af.list('/clientes');
  this.listaClientes.subscribe(dados => {
    if (dados.length > 0) {
      this.total = dados.length;
      dados.forEach(data => {
        let query = "INSERT or replace INTO clientes(id, nome, endereco, fone) " +
          " VALUES (?,?,?,?)";
        this.banco.banco.executeSql(query, [data.id, data.nome, data.endereco, data.fone])
          .then((data) => {
            this.atual++;
            if (this.atual == this.total) {
              let toast = this.toastCtrl.create({
                message: 'Clientes importados com Sucesso.',
                duration: 5200,
                position: 'bottom',
                showCloseButton: true,
                closeButtonText: 'OK'
              });
              toast.present();
            }
          })
          .catch(e => {
            let toast = this.toastCtrl.create({
              message: 'Ocorreu um erro ao importar clientes.',
              duration: 5200,
              position: 'bottom',
              showCloseButton: true,
              closeButtonText: 'OK'
            });
            toast.present();
            console.log(e)
          });
      });
    }
  }, function (err) {
    console.log(err);
  });

}
```

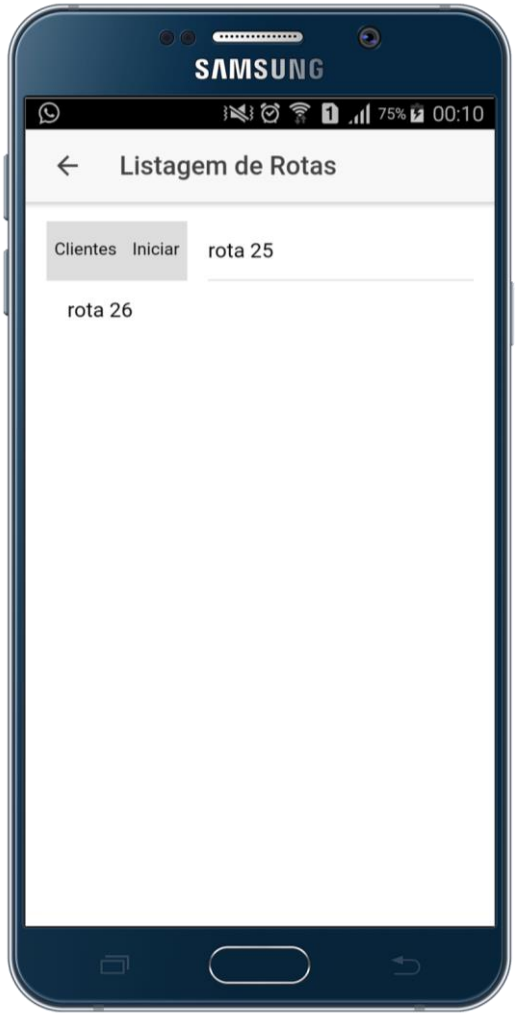
Fonte: (Autoria Própria, 2017)

4.2.2 Listagem de dados

A listagem dos dados presentes no aplicativo possibilita que o motorista realize diversas tarefas, bem como visualizar informações relevantes. A listagem de rotas é exibida em dois locais do aplicativo: no primeiro, é exibida em uma tela própria, que pode ser acessada através do menu em qualquer momento, já o segundo, é exibida na tela inicial, quando não existe nenhuma coleta pendente. Com essa listagem, é possível iniciar uma coleta baseada na rota selecionada, bem como

listar todos os clientes da mesma. Na Figura 16 pode ser visualizada a tela dessa listagem:

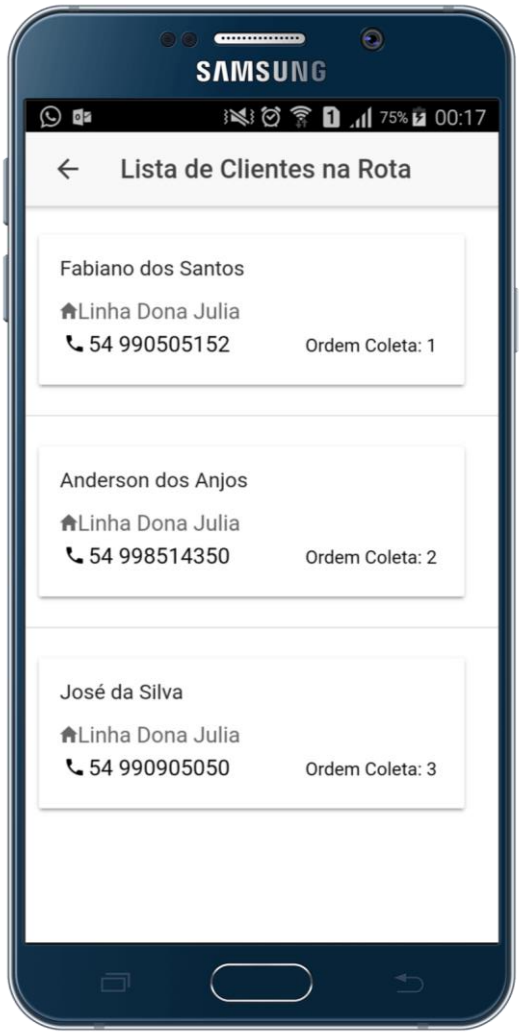
Figura 16: Tela de listagem de Rotas.



Fonte: (Autoria Própria, 2017)

A listagem de clientes é responsável por exibir os dados básicos do cliente, bem como a informação da sua ordem de coleta, baseado na rota. Esta listagem pode ser acessada através da listagem de rotas, na opção exibir clientes que cada item da listagem possui. Na Figura 17 pode ser visualizada a listagem de clientes:

Figura 17: Tela de listagem de Clientes.



Fonte: (Autoria Própria, 2017)

Por fim, a listagem de coletas é responsável por exibir todas as coletas realizadas por esse dispositivo, desde o último login. Ela é acessada através do menu correspondente, onde ao abrir a tela, são listados os dias da coleta, e a rota responsável. Para que sejam exibidas as informações de cada cliente coletado, é necessário acessar a opção exibir coleta, localizada no menu deslizante do item da listagem de coletas. Nesta tela, dados como horário e quantidade coletada são exibidos. Na Figura 18.A pode ser visualizada a primeira tela das coletas e na Figura 18.B pode ser visualizada as informações detalhadas das coletas.

Figura 18: Tela de listagem e de detalhamento de Coletas.

A B

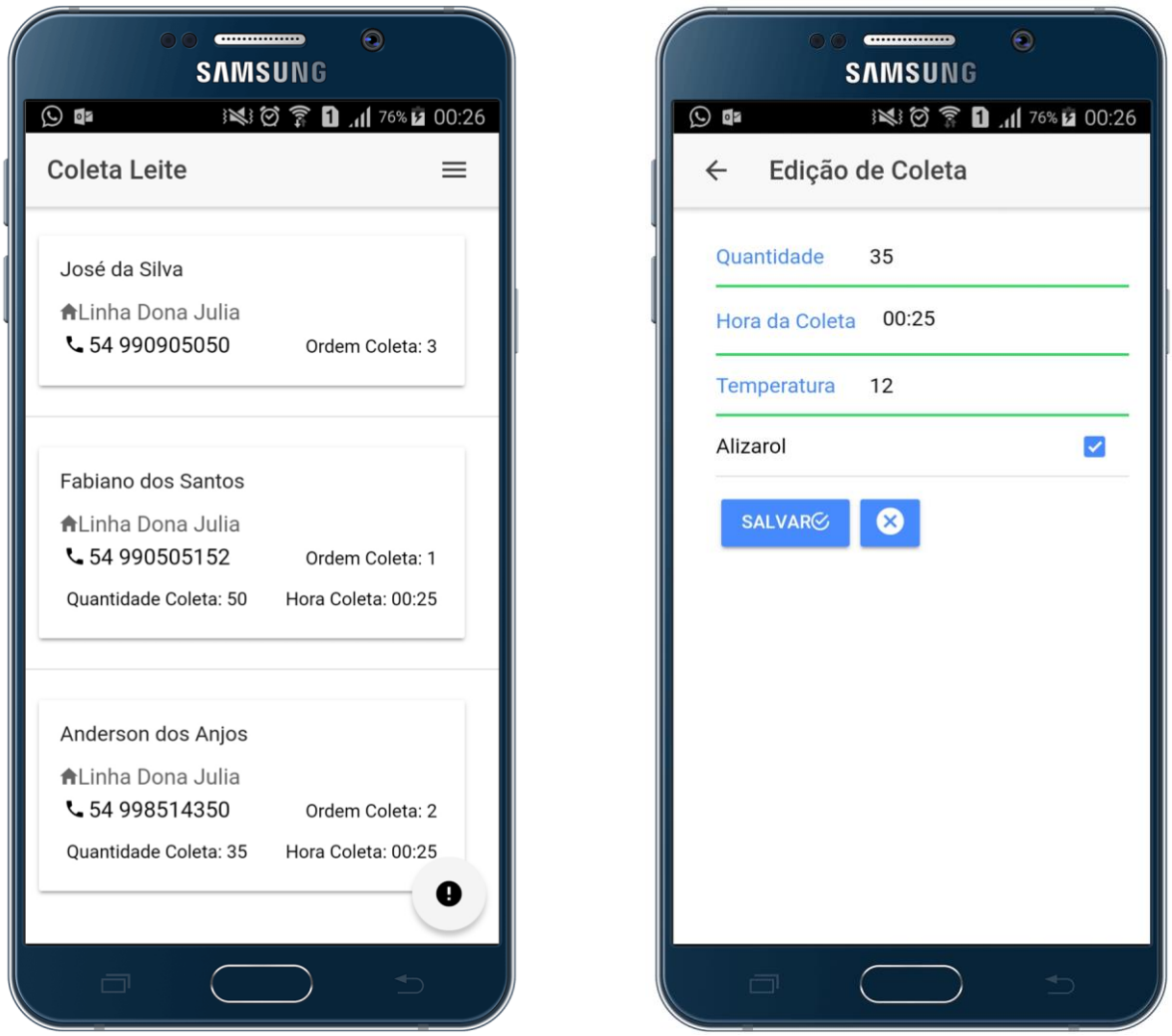
Fonte: (Autoria Própria, 2017)

4.2.3 Manter Coletas

Quando uma coleta é iniciada na listagem de rotas, é inserido um registro de cada cliente da coleta com todos os valores nulos, com exceção da quantidade que recebe um valor *"-1"*. Com isso, ao acessar a tela inicial do aplicativo, é listada a ultima coleta não finalizada no aplicativo, priorizando no topo as coletas não realizadas e seguindo a ordem de coleta estipulada. Ao clicar em uma das coletas, é exibida uma tela para informar os dados de cada uma, como quantidade coletada e temperatura. Após o motorista concluir a rota em questão, ele deve finalizar a coleta,

no botão localizado no rodapé direito da tela inicial. Na Figura 19.A é exibida a tela inicial com algumas coletas já realizadas, e na Figura 19.B é exibido o formulário onde são preenchidas as informações de cada coleta.

Figura 19: Tela inicial com uma coletas pendentes e tela de manutenção de coleta.



A B

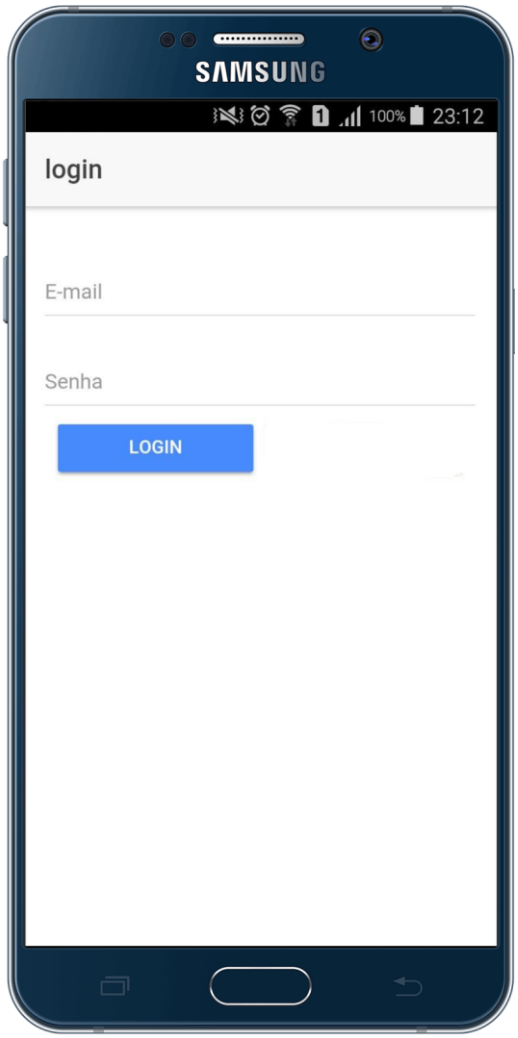
Fonte: (Autoria Própria, 2017)

4.2.4 Autenticação

O processo de autenticação dentro do aplicativo é realizado utilizando o sistema de usuário e senha disponibilizado pelo Firebase, e pode ser divido em duas partes. A primeira delas é realizada no primeiro login do usuário dentro do aplicativo, ou após ele escolher desconectar o usuário. Já a segunda é realizada toda vez que

o usuário acessar a tela de sincronização de dados, caso o login anterior esteja inválido. O principal objetivo da autenticação no aplicativo é identificar qual motorista está usando ele, para então poder selecionar qual rota será disponibilizada, além de garantir que não ocorram acessos indevidos à aplicação. Quando ocorre uma autenticação do usuário no aplicativo, alguns dados básicos são gravados localmente, para que não seja solicitado esse login toda vez que o usuário acessar o sistema. Assim, no momento que a aplicação identifica que um usuário se autenticou anteriormente, ela não apresenta a tela de login. Pelo aplicativo, o usuário só consegue realizar a autenticação, pois o processo de cadastro é realizado na aplicação de integração. Na Figura 20 é apresentada a tela de login do aplicativo:

Figura 20: Tela de login do aplicativo.



Fonte: (Autoria Própria, 2017)

4.2.5 Exportar coletas

A exportação de coletas é responsável por enviar ao Firebase todas as coletas finalizadas que ainda não foram sincronizadas. Para realizar esse processo, o aplicativo monta um json com cada coleta não sincronizada, baseando-se na estrutura informada anteriormente. Após cada coleta, então é executado o método *push* do *angularFire*, que por sua vez gera uma chave única para o registro e o salva na base de dados do Firebase. Caso esta operação não retorne um erro, é alterado o status de sincronizado do registro para verdadeiro, impedindo um reprocessamento do mesmo.

5 CONSIDERAÇÕES FINAIS

A utilização de um serviço BaaS se mostrou extremamente útil, principalmente pela facilidade que provê na implementação de serviços vitais para o desenvolvimento de uma aplicação. A agilidade com que uma aplicação pode ser disponibilizada no mercado, além do custo-benefício da sua implantação são os pontos fortes na sua utilização em nível de mercado.

No modelo tradicional é necessário desenvolver as duas camadas básicas, *frontend* e *backend*, sendo necessário um tempo maior para desenvolvimento da camada do *backend*, além de despender recursos para implementação de testes, correção de bugs e custeio da hospedagem. Já os serviços que se utilizam do BaaS necessitam basicamente apenas de recursos para pagar a sua utilização, e um tempo extremamente reduzido para a implantação dos projetos.

Com o desenvolvimento do estudo de caso, que se utilizou dos diversos serviços disponibilizados por um BaaS, pode-se constatar na prática a agilidade e o custo na implantação. Porém, a principal dificuldade em se utilizar esses serviços é o armazenamento de dados, pois, a maioria daqueles que provêm um serviço no modelo BaaS, disponibilizam como forma de armazenamento de dados um banco não estruturado, o que muitas vezes acaba por dificultar a sua modelagem, uma vez que não seguem um padrão, como comumente pode ser visto em um banco de dados estruturado.

Para trabalhos futuros pode-se citar a adição de notificações push quando atualizações de dados estejam disponíveis, bem como sincronização automática de dados na aplicação de integração.

6 REFERÊNCIAS

ALEXAKIS, Brian. *6 Essential BaaS Features Every Mobile App Needs.* 2014. Disponível em: <https://www.programmableweb.com/news/6-essential-baas-features-every-mobile-app-needs/sponsored-content/2014/10/21>. Acesso em: 30 abril 2017.

Angular. *One framework. Mobile & desktop.* Disponível em: <https://angular.io/>. Acesso em: 28 junho 2017.

Apache Cordova. Disponível em: <https://cordova.apache.org/>. Acesso em: 28 junho 2017.

Embarcadero. *Fast Cross-Platform App Development Software* . Disponível em: <https://www.embarcadero.com/>. Acesso em: 28 junho 2017.

EXAME. *8 motivos para aderir à computação em nuvem.* 2015. Disponível em: <http://exame.abril.com.br/tecnologia/8-motivos-para-aderir-a-computacao-em-nuvem/>. Acesso em: 30 março 2017.

G1. *Celular se consolida como principal meio de acesso à internet no Brasil, aponta IBGE.* 2016. Disponível em: <http://g1.globo.com/economia/noticia/cai-pela-1-vez-no-brasil-o-acesso-a-internet-por-meio-de-computador-diz-ibge.ghtml>. Acesso em: 30 março 2017.

Ionic. *Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular.* Dísponivel em: < http://ionicframework.com/ >. Acesso em: 28 junho 2017.

Kinvey. *Mobile Backend as a Service (mBaaS) for the Enterprise.* Disponível em: < https://www.kinvey.com/ >. Acesso em: 14 maio 2017.

Kumulos. The Mobile App Management Platform for Mobile App Agencies. Disponível em: < https://www.kumulos.com/ >. Acesso em: 20 maio 2017.

LANE, Kin. *Overview Of The Backend as a Service (BaaS) Space.* 2013. Disponível em: <http://kinlane-productions.s3.amazonaws.com/whitepapers/API+Evangelist+-+Overview+of+the+Backend+as+a+Service+Space.pdf>. Acesso em: 30 abril 2017.

LUPCHINSKI, Raphael de Leon Ferreira. *Desenvolvimento de uma Aplicação de Página-Única e Banco de Dados Não-Relacional para Organização e Controle de Eventos Esportivos*. 2015. 70f. Monografia (Graduação em Ciência da Computação) - Universidade Federal Do Rio Grande Do Sul, Porto Alegre, 2015. Acesso em: 2 abril 2017.

MORIBE, Fernando. *Firebase — Vantagens de um BaaS para sua Startup.* 2016. Disponível em: < https://medium.com/@fgmoribe/firebase-vantagens-de-um-baas-para-sua-startup-38fd3891329a>. Acesso em: 25 abril 2017.

RODRIGUEZ, William S. . *Azure Mobile Services, entenda o que é, suas vantagens e como começar.* 2015. Disponível em: <http://williamsrz.azurewebsites.net/2015/11/24/azure-mobile-services-entenda-o-que-e-suas-vantagens-e-como-comecar/>. Acesso em: 21 março 2017.

SILVA, Raphael Lira da. *Aplicativo para facilitar a coleta de dados em pesquisas origem/destino.* 2016. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/5771/1/CM_COINT_2016_1_01.pdf>. Acesso em: 03 junho 2017.

SOBRAL, Felipe Braga. *E-SENHA: aplicativo para notificação de senhas de atendimento em smartphones.* 2015. Disponível em: <http://fatecsjc.azurewebsites.net/trabalhos-de-graduacao/wp-content/uploads/2016/04/Aplicativo-para-Notifica%C3%A7%C3%B5es-de-Senhas-de-Atendimento-em-Smartphones-Felipe-Braga-Sobral1.pdf>. Acesso em 19 março 2017.

TDEVROCKS. *TDevRocks.* 2017. Disponível em: <https://github.com/tdevrocks/classes-uteis>. Acesso em: 9 setembro 2017.

VANDRESEN, Rogério Schueroff; MAGALHÃES, Willian Barbosa. *Conceitos e Aplicações da Computação em Nuvem.* 2013. Disponível em: <http://ftp.unipar.br/~seinpar/2013/artigos/Rogerio%20Schueroff%20Vandresen.pdf>. Acesso em: 5 maio 2017.

VIEIRA, Cristiano Costa Argemon. *Um Modelo de Escalonamento de Requisições de Máquinas Virtuais em Provedores de IaaS Considerando Diferentes Requisitos dos Usuários.* 2016. Disponível em: < http://repositorio.unicamp.br/bitstream/REPOSIP/321704/1/Vieira%2c%20Cristiano%20Costa%20Argemon_D.pdf >. Acesso em: 01 junho 2017.