UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

GUILHERME MACIEL DA SILVA

AUTO AGENDA: SISTEMA DE CONTROLE ONLINE DE AGENDAS DE CONTATO

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

GUILHERME MACIEL DA SILVA

AUTO AGENDA: SISTEMA DE CONTROLE ONLINE DE AGENDAS DE CONTATO

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Especialista em Desenvolvimento para Dispositivos Móveis".

Orientadora: Prof.ª Dra. Maria Claudia Figueiredo Pereira Emer.

CURITIBA
2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

“Auto Agenda: Sistema de Controle Online de Agenda de Contato”

por

“Guilherme Maciel da Silva”

Este Trabalho de Conclusão de Curso foi apresentado às 18:53 do dia 08 de dezembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____ Profa. Maria Claudia Figueiredo Pereira Emer **(Presidente - UTFPR/Curitiba)**	_____ Prof. Robson Ribeiro Linhares **(Avaliador 1 – UTFPR/Curitiba)**
_____ Prof. João Alberto Fabro **(Avaliador 2 – UTFPR/Curitiba)**	

“A Ata de Aprovação assinada encontra-se na Coordenação do Curso.”

AGRADECIMENTOS

A Deus, pela vida, sabedoria e saúde.

A minha família, pelos ensinamentos e exemplos, que me transformaram na pessoa que eu me tornei.

Aos professores, que transmitiram o seu conhecimento, a mim e meus colegas de classe.

A professora Maria Claudia Emer, pela orientação, atenção e auxílio durante o desenvolvimento deste trabalho.

RESUMO

MACIEL DA SILVA, Guilherme. Auto Agenda: Sistema de controle online de agendas de contato. 2017. 59f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Os avanços tecnológicos e a popularização destas mesmas tecnologias, gera um aumento significativo nas formas com que as pessoas conseguem interagir, essas interações dependem fortemente da ferramenta ou tecnologia utilizada. Por vezes, é necessário saber de antemão a identificação do usuário com que se pretende interagir. Assim sendo, o desenvolvimento deste trabalho justifica-se por proporcionar ao usuário de aparelhos móveis uma forma de tratar as dificuldades de propagação de mudanças de seus contatos. Esse trabalho tem por objetivo desenvolver um aplicativo para centralizar a troca de informações de contato entre usuários de aparelhos móveis com sistema operacional Android. Para isto, o estudo realizado sobre as tecnologias utilizadas no desenvolvimento do aplicativo proposto é mostrado. E por fim, o aplicativo resultante desse trabalho é apresentado, assim como a explicação de como foi pensada a sua arquitetura.

Palavras-chave: Agenda de contatos, Comunicação, Interação, Android.

ABSTRACT

MACIEL DA SILVA, Guilherme. Auto Agenda: Contact list management system. 2017. 59f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017

The technological breakthroughs and its popularization lead to diversified ways for people to interact among themselves, often, those interactions strongly rely on the tools or technology used, where most of the time, an user needs beforehand, an identification of the user he intends to interact with. Due to that fact, this project is justified as it provides mobiles devices users a way to handle those difficulties in propagating their contacts information. This project focus on developing an Android mobiles app that will manage the information exchange among its users. It also shows all the studies made over the technologies used on this projects development as well as the resulting app developed from the studies and its architecture.

Keywords: Contacts, Communication, Interaction, Android.

LISTA DE FIGURAS

LISTA DE SIGLAS

APIs	Aplication Programming Interface (Interface de Programação de Aplicativos).
SDK	Software Development Kit (Kit de Desenvolvimento de Software).
SQL	Structured Query Language (Linguagem de Consulta Estruturada).
UML	Unified Modeling Language (Linguagem de Modelagem Unificada).

LISTA DE ACRÔNIMOS

ONU	Organização das Nações Unidas.
NoSQL	Not Only SQL (Não Somente SQL).
JSON	JavaScript Object Notation (Notação de Objeto JavaScript)

SUMÁRIO

1 Introdução

1.1 Contextualização

A Evolução das tecnologias vem provendo diversas mudanças em como a sociedade interage. A velocidade da troca de informações nunca foi tão rápida, na qual, dependendo da qualidade da infraestrutura, e a distância entre os pontos em contato, o tempo necessário para a transmissão da informação é quase nulo.

Empresas aproveitam a tecnologia para criar produtos e necessidades, que a algumas décadas, ninguém cogitaria que existissem. A Internet, que até pouco tempo atrás era um artigo de luxo, utilizado apenas por famílias de poder aquisitivo relativamente bom, hoje, já é considerada pela ONU, desde 2011, como um direito básico humano (CONSELHO DE DIREITOS HUMANOS DA ONU, 2011).

Uma pesquisa publicada pelo CENTRO REGIONAL DE ESTUDOS PARA O DESENVOLVIMENTO DA SOCIEDADE DA INFORMAÇÃO (2015) mostra a evolução do acesso à internet por partes das classes mais baixas. No ano de 2008, apenas 1% das residências das famílias de classe D e E possuíam acesso à internet, no ano de 2015, esse valor chegou a 16%. Um grande salto também foi visto com relação aos acessos das residências de famílias da classe C, no qual o salto foi de 16% a 49% no mesmo período.

Os resultados são muito mais expressivos que os obtidos pelas classes B (um aumento de 58% para 82%) e A (aumento de 91% para 97%) no mesmo período.

Ainda de acordo com a mesma pesquisa efetuada pelo CENTRO REGIONAL DE ESTUDOS PARA O DESENVOLVIMENTO DA SOCIEDADE DA INFORMAÇÃO (2015), essa mudança também é abrangente referente a área em que ela ocorre, pois também é possível ver o crescimento consistente tantos nas áreas urbanas (com 20% das residências com acesso à internet em 2008, e 56% com acesso em 2015) quanto nas rurais (com um aumento de 4% em 2008, para 22% em 2015).

Uma pesquisa realizada pela CISCO (2017), estima que no ano de 2021, o número de aparelhos com conexão à internet será 3.5 vezes maior que a população mundial, e que desses aparelhos, os aparelhos com conexão wireless e mobile, serão responsáveis por aproximadamente 63% do trafego de banda, um aumento de 12%, comparado com os 51% registrados em 2016.

O Avanço tecnológico também criou um fenômeno interessante, no qual o usuário final, deixou de ser apenas um consumidor de conteúdo, e passou a ser um dos principais geradores do mesmo, em redes sociais, blogs e fóruns, que são apenas repositórios, nos quais o próprio cliente "gera" o produto que vai consumir, como observa um artigo escrito em 2012 pelo Laboratório MídiaCom, PGC-TCC/Instituto de Computação Universidade Federal Fluminense (BRITO, 2012).

1.2 Motivação e justificativa

Apesar dos avanços tecnológicos, da grande troca de informações e geração de conteúdo, alguns casos ainda não parecem ter sido explorados abertamente pelo mercado. Enquanto redes sociais permitem postar informações constantemente, elas estão limitadas as informações ligadas ao propósito da rede, e também estão atreladas ao perfil do usuário na mesma.

Muitos dos dados pessoais geralmente não são utilizados nessas redes sociais, um e-mail ou telefone profissional, por exemplo, outros, usuários preferem não expor publicamente para qualquer pessoa, como um endereço ou número de telefone pessoal. Esse trabalho focará em uma maneira na qual, o usuário poderá compartilhar dados pré-selecionados com seus contatos de redes sociais, dados esses que normalmente não estão presentes nas mesmas redes sociais usadas pelo usuário.

1.3 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um aplicativo de gerenciamento de agenda de contatos, que atualize automaticamente os dados de contatos que se relacionam com o usuário em redes sociais pré-determinadas.

1.4 Objetivos específicos

Os objetivos específicos deste trabalho são os seguintes:

- Pesquisar alternativas de gerenciamento de agenda de contatos existentes no mercado.
- Criar uma ferramenta de integração dos dados salvos pelo usuário
- Configurar um serviço de push para o envio de notificações do aplicativo

- Desenvolver o aplicativo responsável por manter os dados da agenda de contato do usuário.
- Avaliar o aplicativo junto aos possíveis usuários.

1.5 Método

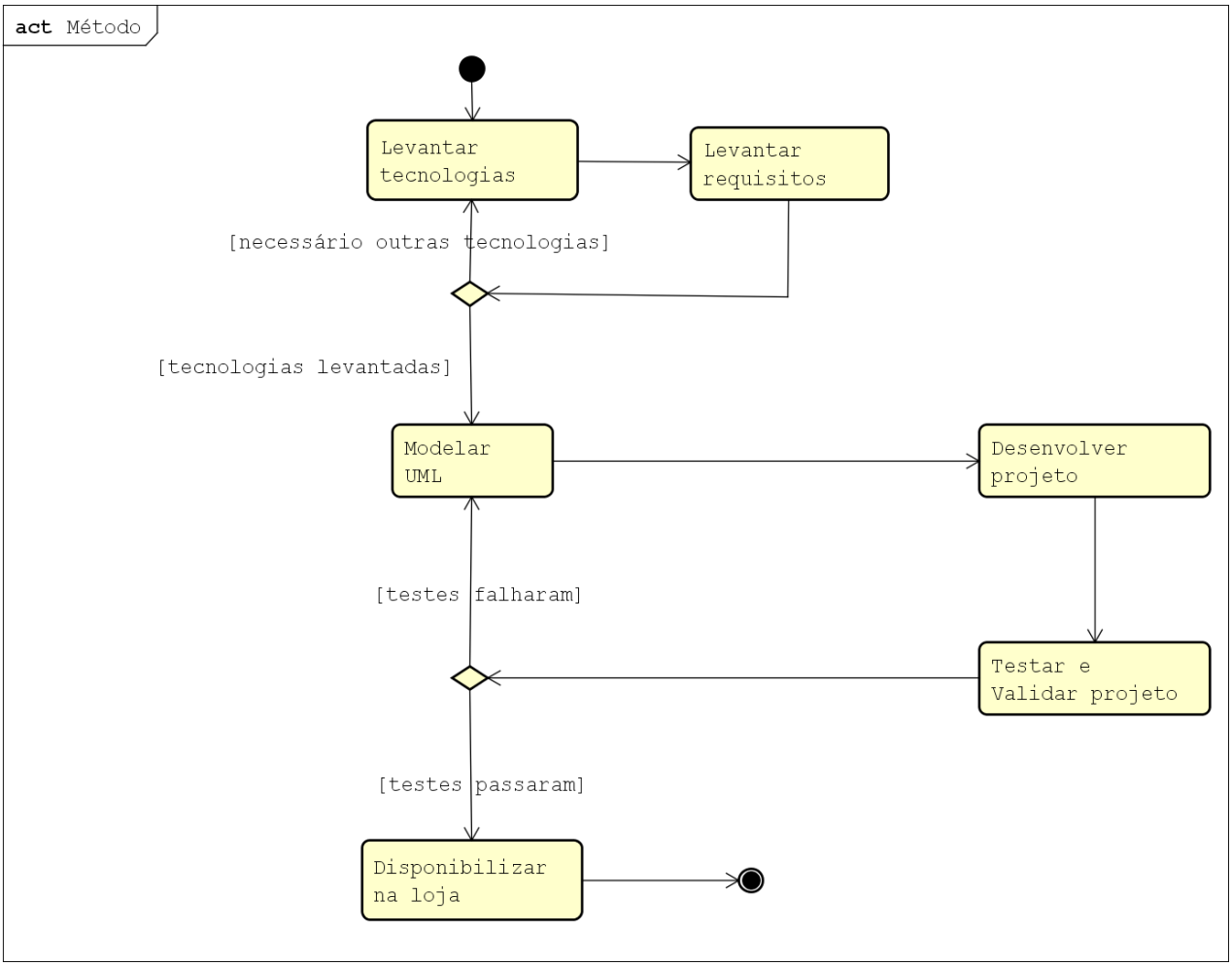
O trabalho foi desenvolvido de acordo com as etapas ilustradas na Figura 1. Essas etapas são descritas a seguir:

- Levantar tecnologias – As tecnologias utilizadas no desenvolvimento deste projeto foram escolhidas a partir de um levantamento de tecnologias e APIs, no qual as opções disponíveis no mercado foram analisadas considerando a relação de usabilidade e eficiência para atender as demandas que se apresentarem no projeto.
- Levantar requisitos - Fazer o levantamento aprofundado dos requisitos necessários para o desenvolvimento do sistema - A lista de requisitos funcionais e não funcionais foi obtida a partir do estudo de trabalhos relacionados, das funcionalidades desejadas pelo autor ou sugeridas por pessoas conhecidos.
- Modelar UML – Os casos de uso e o diagrama de casos de uso foram desenvolvidos a partir dos requisitos levantados para guiar o desenvolvimento de outros diagramas da UML e também do aplicativo. Os diagramas desenvolvidos foram: Diagrama de atividades, Diagrama de classes e diagrama de entidade-relacionamento.
- Desenvolver o projeto – A partir dos diagramas desenvolvidos, o desenvolvimento do Web Service e do aplicativo foi realizado. O WebService tem o objetivo de armazenar as informações proporcionadas pelo usuário na nuvem, e também disseminar as mesmas a sua rede de contatos. O aplicativo tem o objetivo de apresentar os dados da rede de contatos do usuário salvos no Web Service, assim como também apresentar meios do mesmo atualizar seus dados, a serem armazenados no Web Service.
- Testar e validar o projeto - Testes unitários de integração foram realizados durante o desenvolvimento do aplicativo e do Web Service. Após o desenvolvimento, o aplicativo foi apresentado a algumas pessoas para uma avaliação do mesmo. As três pessoas escolhidas não estavam sujeitas a seguir nenhum roteiro ou questionário, com o objetivo de validar se o propósito do aplicativo estava claro, assim como sua usabilidade. As informações foram colhidas a partir

da observação do uso do aplicativo, assim como o feedback verbal apresentados pelos avaliadores.

- Disponibilizar na loja - O aplicativo deve ser disponibilizado na loja de apps da Google, na qual as avaliações dos usuários serviram de norte para possíveis evoluções do aplicativo.

Figura 1- Diagrama de atividades referente ao método empregado no trabalho.



Fonte: O Autor (2017)

1.6 Estrutura do trabalho

O primeiro capítulo, a Introdução, apresentou a motivação, justificativa, objetivos e metodologia deste trabalho. O segundo capítulo, denominado Fundamentação teórica, mostra os aspectos teóricos que foram usados para o desenvolvimento do trabalho, analisando tecnologias

similares ou relacionadas, e quais as tecnologias serão utilizadas no desenvolvimento. O terceiro capítulo, denominado Desenvolvimento do projeto, descreve como o desenvolvimento foi realizado, com detalhamentos de protótipos, requisitos de sistema, especificações e implementação. O capítulo final, a Conclusão, expões as conclusões obtidas no desenvolvimento deste trabalho, pontos que ficaram pendentes e futuras implementações.

2 Fundamentação Teórica

Neste capítulo, são apresentadas as tecnologias estudadas para o desenvolvimento deste projeto e os trabalhos relacionados à categoria do aplicativo desenvolvido neste projeto.

2.1 Tecnologias

O estudo das necessidades que o aplicativo teria que sanar, alinhados com o conhecimento prévio das partes envolvidas, levaram a uma lista de tecnologias que serão utilizadas no desenvolvimento do aplicativo aqui proposto, elas são:

2.1.1 Android

Android é um sistema operacional para dispositivos móveis, que prove a capacidade de explorar as diversas funcionalidades que os smartphones atualmente oferecem .

Desenvolvido pela Open Handset Alliance, esta plataforma é focada em celulares e tablets, Assim como o principal concorrente, iOS, o sistema operacional Android é composto por camadas de software que proporciona uma gama de *frameworks* úteis no desenvolvimento de aplicativos móveis (SILBERSCHATZ, 2014).

A popularização do Android está ligada ao fato que líderes de mercado da tecnologia e telefonia estão envolvidos no seu desenvolvimento, cujos interesses estão na difusão de padrões abertos para aparelhos móveis (LECHETA, 2009).

2.1.2 Firebase

Firebase é uma plataforma de desenvolvimento desenvolvida pela empresa de mesmo nome, e adquirida pelo Google em 2014.

O propósito do Firebase é prover uma gama de serviços, como base de dados, autenticação, funcionalidades e armazenamento na nuvem, entre outras funcionalidades gratuitas e pagas. O Firebase trabalha com a ideia de descomplicar o desenvolvimento de aplicativos, permitindo o

desenvolvedor focar no desenvolvimento do aplicativo em si, enquanto o Firebase foca no resto (FARR, 2013).

O Firebase permite ao usuário trabalhar em diversas tecnologias, como Android, IOS, clientes JavaScript, C++ e Unity, como exemplos.

2.1.2.1 Real Time Database

A funcionalidade Real Time Database do Firebase, é uma base de dados NoSQL na nuvem relacionada a um projeto Firebase, A mesma apresenta algumas vantagens relacionadas ao padrão relacional de Base de dados (KUMAR, 2016), algumas são:

- A centralização dos dados permite uma maior facilidade de replicar os dados atualizados ou adicionados entre os dispositivos conectados na base de dados;
- Os dados são armazenados na nuvem, tornando sua acessibilidade e disponibilidades limitadas a uma simples conexão com a internet.
- Por ser uma API multiplataforma, várias versões do aplicativo, site ou sistema utilizam a base sem necessidades de adaptações impactantes.
- O Firebase armazena e gerencia as informações, significando que um grande fluxo de dados, não afeta o desempenho do sistema.

Real Time Database permite a criação de regras para a proteção das informações adicionadas (PLANGI, 2013), para prevenir ações indesejadas de update e delete. Regras para assegurar que o usuário adicionando, lendo ou removendo os dados, está devidamente autenticado, podem ser criadas para diferentes níveis dentro da árvore de dados armazenada na base. Regras de validação de dados também podem ser criadas para validar inserções ou atualizações na base de dados.

2.1.2.2 Firebase Authentication

A funcionalidade Firebase Authentication fornece serviços de *back-end*, SDKs e bibliotecas para realizar a autenticação do usuário no aplicativo que está utilizando a mesma, o desenvolvedor tem a opções de utilizar senhas, números de telefone e outras API de provedores de identidade, como Google, Facebook, Twitter, ente outros.

A Firebase Authentication, se integra com as outras funcionalidades do Firebase naturalmente.

2.1.2.3 Firebase Cloud Functions

A ferramenta Firebase Cloud Functions, permite a execução de códigos de *back-end* automaticamente, responsivos a eventos preestabelecidos envolvendo outras funcionalidades do Firebase ou solicitações HTTPS, pelo fato do código estar na nuvem, e ser executa em um ambiente gerenciado, não existe a necessidade de uso de servidores próprios do desenvolvedor para o processamento dessas funcionalidades.

Através do Firebase Cloud Functions, que tem o foco em eventos, o desenvolvedor amplia as possibilidades de comportamento do Firebase, por meio de inserção de códigos que respondem a ações preestabelecidas.

2.1.2.4 Firebase Cloud Messaging

O Firebase Cloud Messaging é uma ferramenta que possibilita o envio de mensagens entre plataformas diferentes, sem custo para o desenvolvedor.

Através do Firebase Cloud Messaging, é possível mandar mensagens *push* para todos os usuários de um aplicativo, mensagens especificas relacionadas a um tópico, para todos os usuários que estiverem escutando este tópico, assim como mensagens diretas a usuários específicos.

2.1.3 Facebook API

A API de desenvolvimento do Facebook permite integração com diversas tecnologias utilizadas no mercado, possibilitando a leitura e escrita de informações presentes na ferramenta Facebook. A disponibilização dessa API fortalece a disseminação do uso da rede social, na qual você atrela o seu uso a outros sistemas, que em teoria seriam independentes da rede. O uso desta tecnologia vem se mostrando vantajosa nos negócios *mobile*, pois ela se adequa devido a velocidade, volume de informações disponíveis e publicidade, pois os dispositivos moveis estão constantemente funcionando, e sempre ao alcance do usuário (SELVARATHY, 2014).

2.1.4 Web Service

Um web service é uma solução utilizada na integração de sistemas e comunicação entre aplicações diferentes, permitindo o envio e recebimento de dados, independentemente da plataforma empregada (MCLAUGHLIN, 2001). Para o desenvolvimento de um web service que sane as necessidades apresentadas neste projeto, foram levantadas as seguintes tecnologias.

2.1.4.1 Node.js

Node.js é um interpretador de código JavaScript que executa no do servidor. Sua principal vantagem é a possibilidade de criar aplicações de alta escalabilidade, manipulando milhares de conexões simultâneas, em um simples servidor.

Node.js usa a *Engine* V8, do Google, onde alterações foram executadas para permitir o funcionamento do javascript fora do contexto do *browser* (JUNIOR, 2012).

Voltado a eventos, o Node.js permite a criação de requisições assíncronas, nas operações de *input* e *output*.

O Node.js é um projeto de desenvolvimento distribuído desenvolvido pela Fundação Node.js, facilitado pela The Linux Foundation.

2.1.4.2 MongoDB

MongoDB é um banco de dados orientado a documentos de alta performance NoSQL. Devido ao fato do MongoDB ser baseado em documentos, sem envolver transações e *joins*, os resultados e as consultas são relativamente mais simples e ajustáveis. A ferramenta permite a divisão do processamento e armazenamento dos dados em diferentes maquinas, através de uma técnica chamada *sharding* (NASCIMENTO, 2010). Outra funcionalidade interessante da ferramenta é a possibilidade de guardar arquivos na base.

2.2 Trabalhos Relacionados

Em artigo publicado em 2011, Fatos e Seguí (2013) propuseram um algoritmo que trabalhasse na sincronização de uma agenda de contatos compartilhada, no qual eles identificavam o problema como um sistema distribuído, que envolvia várias conexões à mesma fonte de dados, no caso, uma agenda única, compartilhada e mantida por um número X de usuários. As principais dificuldades desse processo, eram como o conflito de informações seria atacado, no qual mais de um usuário alterasse o mesmo contato, e possíveis gargalos de processamento, caso essa lista de contatos crescesse exponencialmente.

A lista de aplicativos disponíveis na Google Play é muito extensa, cobrindo diversas categorias, a categoria Comunicação, na qual o projeto proposto se enquadra, não é diferente, tendo uma lista imensa de opções de aplicativos que trabalham com a lista de contato, dentre estes, alguns se destacam. Segue a seguir uma breve descrição deles e suas principais funcionalidades.

- Sync.ME, desenvolvido pela empresa de mesmo nome, tem como foco a identificação de quem está realizando a chamada ou mensagem recebida, dando a opção de bloquear ou não essas ligações e/ou mensagens, A ferramenta também busca informações das redes sociais dos contatos.
- Contacts+, desenvolvido pela Contacts Plus Team, tem como foco o registro dos contatos, ligações e mensagens, salvando essas informações nos seus servidores, como backup. O aplicativo também permite o bloqueio de ligações e mensagens de *SPAM* e contatos previamente selecionados.

- Simpler Contacts, desenvolvido pela SimplerApps, foca principalmente no gerenciamento dos contatos, evitando contatos duplicados na agenda, e criando grupos customizados para agrupar os contatos. A ferramenta também tem funções de Backup, que podem ser exportados para ferramentas de armazenamento na nuvem.

Analisando as funcionalidades desses aplicativos, percebe-se um foco em bloqueio de chamadas indesejadas, e no backup da lista de contatos do usuário. Em nenhum momento, se vê a necessidade do aplicativo em garantir a validade das informações armazenadas pelo mesmo, que é o foco principal desse projeto. Como as principais funcionalidades entre o projeto e os aplicativos citados anteriormente tem objetivos principais diferentes, é possível o uso desse com qualquer uma das opções acima (ou qualquer outra já utilizada pelo usuário) em paralelo, sem conflito de funcionalidades.

3 Desenvolvimento do projeto

Neste capítulo, são apresentadas as etapas do desenvolvimento, sendo elas, descrição da aplicação (seção 3.1), levantamento de requisitos (seção 3.2), casos de uso (seção 3.3), diagramas de classe (seção 3.4) e atividades (seção 3.5), implementação (seção 3.6), telas (seção 3.7) e avaliação (seção 3.8)

3.1 Descrição da aplicação.

A aplicação consiste em um aplicativo Android, executando no aparelho do usuário, se comunicando com funcionalidades presentes nos servidores na nuvem da Google, por meio da ferramenta Firebase. Uma API do Facebook também é utilizada na autenticação dos usuários.

O aplicativo Android consiste em três telas, responsáveis em exibir os dados do usuário, os contatos do usuário e seus detalhes. O aplicativo também tem um serviço que processa as notificações enviadas pelo Firebase.

O Projeto do Firebase consiste em três funções principais, a base de dados que armazena os dados salvos pelos usuários, funções que são disparadas quando eventos específicos nas bases de dados acontecem, e o serviço de autenticação, que é integrado com a API do Facebook.

Vale comentar que as funcionalidades do Firebase Cloud Functions alinhadas com o Firebase Real Time Database acabaram com a necessidade de desenvolvimento de um web service, devido ao fato das duas ferramentas suprirem todas as necessidades que seriam sanadas pelo web service, o armazenamento e propagação dos dados dos usuários.

3.2 Levantamento de Requisitos

A seguir são apresentados os requisitos funcionais e não funcionais identificados para a aplicação móvel que foi desenvolvida neste trabalho. Esses requisitos foram identificados a partir da descrição detalhada da aplicação e por meio da observação de aplicações móveis similares.

3.2.1 Requisitos Funcionais

RF001 – Cadastrar usuário pessoal no sistema
Descrição: Criar uma nova instância de contato pessoal, no primeiro login no sistema.
Prioridade: Essencial
Entrada e pré-condições: NA
Saídas e pós-condições: Usuário cadastrado e um novo contato registrado no sistema.

RF002 – Atualizar dados pessoais do contato pessoal
Descrição: Efetuar a atualização de qualquer dado do contato
Prioridade: Essencial
Entrada e pré-condições: Usuário já estar cadastrado.
Saídas e pós-condições: atualização do usuário pré-cadastrado no sistema.

RF003 – Receber lista de contato
Descrição: receber a lista de contatos cadastrados no sistema
Prioridade: Essencial
Entrada e pré-condições: Usuário autenticado com rede social.
Saídas e pós-condições: Lista de contato atualizada.

RF004 – Receber atualização de contato
Descrição: Receber a atualização de um contato já presente na sua agenda.
Prioridade: Essencial
Entrada e pré-condições: Contato que foi atualizado estar na lista de contatos do usuário.
Saídas e pós-condições: Contato atualizado na agenda do usuário

RF005 – Receber um novo contato.
Descrição: Receber dados de um contato ainda não presente na lista de contatos do usuário
Prioridade: Essencial
Entrada e pré-condições: Usuário autenticado com rede social

Saídas e pós-condições: Contato adicionado na agenda de contatos do usuário.

RF006 – Autenticar usuário já cadastrado.

Descrição: Autenticar um usuário já cadastrado no sistema. Recebendo os dados do web service.

Prioridade: Essencial

Entrada e pré-condições: Usuário já estar cadastrado no sistema.

Saídas e pós condições: Usuário é autenticado e os dados cadastrados são retornados pelo sistema.

3.2.2 Requisitos não funcionais

NF001 - Usabilidade.

A interface com o usuário é de vital importância para o sucesso do sistema, para isso o aplicativo Android deve conter um layout intuitivo e de fácil navegação.

Prioridade: Essencial

NF002 – Desempenho

Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um fator de qualidade de software. O aplicativo mobile deve poder ser executado com hardware com um gigabyte de memória RAM.

Prioridade: Importante

NF003 - Hardware e software

Avaliando a necessidade de conexão com a internet, para a transferência de dados do aplicativo, se faz necessário um aparelho que use conectividade 3/4G e wi-fi. O fato da ferramenta utilizar a maioria de suas funcionalidades na nuvem, uma conexão constante com a internet se faz necessária para o funcionamento do aplicativo. Porém, essa falta de conectividade não deve afetar os contatos já inclusos na lista de contatos do aparelho do usuário.

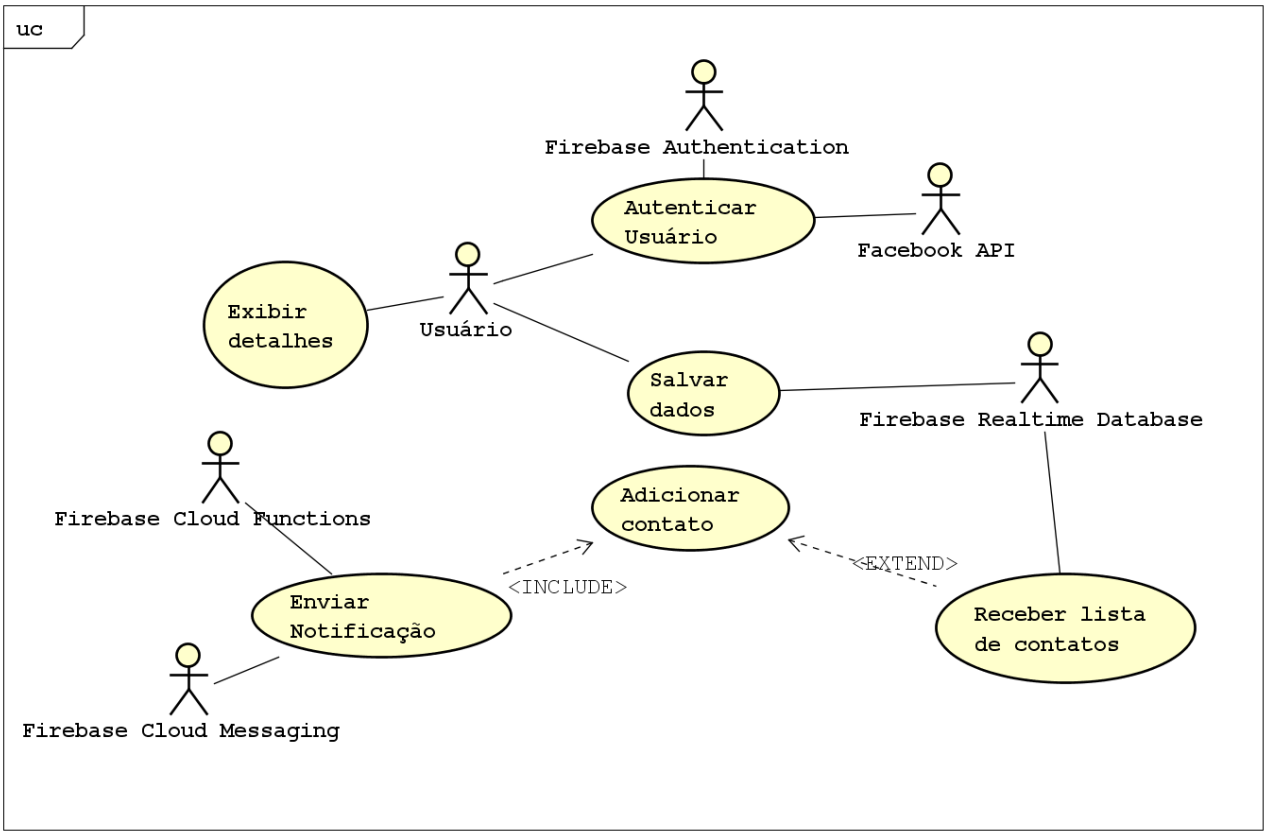
NF004 – Autenticação e fluxo de dados

A Autenticação do usuário e todo o registro de dados cadastrados e atualizados será feita pelas APIs do Facebook e Firebase, os dados dos contatos salvos na agenda do usuário, serão armazenados normalmente no sistema de contatos do sistema operacional Android.

3.3 Casos de uso

Como parte do desenvolvimento do sistema, alguns diagramas da UML foram desenvolvidos para servir como base para o desenvolvimento do aplicativo. A Figura 2 ilustra o diagrama de casos de uso. A descrição dos casos de uso é apresentada a seguir.

Figura 2- Diagrama de casos de uso



Fonte: O Autor (2017)

3.3.1 Descrição Detalhada

UC001

Nome do caso de uso: Autenticar usuário

Descrição:

Este caso de uso tem por objetivo permitir ao usuário se autenticar no aplicativo.

Eventos:

- Usuário solicita função de login
- API de login apresenta dela de login
- Usuário entra com dados para login
- API autentica usuário.
- Usuário cancela função de login.
- Aplicativo envia identificação recebida pela API para o Web service.

Atores:

- Usuário.
- Facebook API
- Firebase Authentication

Pré-Condições:

- O Aplicativo deve estar em execução na tela de início.
- O usuário não deve estar autenticado.
- O usuário tem uma conta na rede social usada para autenticação

Pós Condições:

1. Conclusão com sucesso:
    a. Usuário é autenticado.
2. Conclusão sem sucesso:
    a. Usuário não é autenticado

Fluxo básico:

1. Usuário clica no botão de login.
2. Aplicativo chama a tela de autenticação da API de login do Facebook.
3. Usuário entra com seus dados de login na tela de autenticação. A1, A2.
4. API do Facebook autentica usuário. A3
5. API do Facebook retorna dados da rede social.
6. Aplicativo envia dados da sessão para o Firebase Authentication.
7. Firebase autentica a sessão do usuário.
8. Fim do caso de uso

Fluxos alternativos:

A1: Usuário cancela processo de login.

1. Aplicativo retorna a tela inicial.
2. Fim do caso de uso.

A2: Aparelho já possui credenciais da rede social autenticadas.

1. Retorne ao passo 5 do Fluxo básico.
2. Fim do fluxo alternativo.

A3: Credenciais invalidas.

1. Retorne ao passo 3 do fluxo básico.
2. Fim do fluxo alternativo.

UC002

Nome do caso de uso: Adicionar contato.

Descrição:

Este caso de uso tem por objetivo adicionar um contato na agenda do usuário.

Eventos:

- Serviço do aplicativo recebe dados do Firebase Cloud Messaging.
- Serviço do aplicativo adiciona contato na agenda.

Atores:

Pré-Condições:

- O serviço do aplicativo recebeu os dados do contato.

Pós Condições:

1. Conclusão com sucesso:
    a. Contato adicionado à agenda.
2. Conclusão sem sucesso:
    a. Contato não adicionado à agenda.

Fluxo básico:

1. Serviço do aplicativo recebe os dados. A2
2. Serviço adiciona os dados do contato na agenda do aparelho. A1.
3. Fim do caso de uso.

Fluxos alternativos:

A1: Contato já existe na agenda do usuário.

1. Dados do contato existente são atualizados com as informações recém recebidas.
2. Fim do caso de uso.

A2: Contato recebido da base de dados

1. Aplicativo recebe os dados da base de dados.
2. Aplicativo adiciona os dados do contato na agenda do aparelho. A1
3. Fim do caso de uso.

UC003

Nome do caso de uso: Receber lista de contatos.

Descrição:

Este caso de uso tem por objetivo receber uma lista de contatos do web service.

Eventos:

- Aplicativo recebe os dados da Firebase Real Time Database.

Atores:

- API de notificação.

Pré-Condições:

- O aparelho deve estar conectado à internet
- O usuário deve estar autenticado.
- Esta é a primeira conexão ao sistema pelo usuário no aparelho.

Pós Condições:

1. Conclusão com sucesso:
    a. Contatos adicionado à agenda.
2. Conclusão sem sucesso:
    a. Contatos não adicionado à agenda.

Fluxo básico:

1. Aplicativo recebe lista de contatos do Facebook da API do Facebook.
2. Aplicativo valida quais contatos já estão na memória do aparelho.
3. Para cada item na lista de contatos, aplicativo baixa dados do contato da Firebase Real Time Database.
4. Chama UC002 para cada dado baixado.
5. Fim do caso de uso.

UC004

Nome do caso de uso: Salvar dados.

Descrição:

Este caso de uso tem por objetivo salvar dados do contato na Firebase Real Time Database.

Eventos:

- Aplicativo envia dados ao Firebase Real Time Database.
- Dados são salvos na base de dados.

Atores:

- Usuário.
- Aplicativo

Pré-Condições:

- O aparelho deve estar conectado à internet
- O usuário deve estar autenticado.
- O Usuário realizou alguma alteração nos seus dados pessoais.

Pós Condições:

1. Conclusão com sucesso:
    a. Dados do usuário são atualizados no web service
2. Conclusão sem sucesso:
    a. Dados não são atualizados.

Fluxo básico:

1. Usuário altera dados pessoais.
2. Aplicativo envia dados para a Firebase Real Time Database.
3. Dados são replicados na base de dados. A1
4. Chama UC004.
5. Fim do caso de uso.

Fluxos alternativos:

A1: Dados não são salvos na base de dados.

1. Fim do caso de uso.

UC005

Nome do caso de uso: Enviar notificação.

Descrição:

Este caso de uso tem por objetivo uma enviar notificação dos dados do contato alterado no tópico especifico.

Eventos:

- Dados são alterados na Firebase Real Time Database.
- Firebase Cloud Messaging envia uma notificação push com os dados do contato alterado.

Atores:

- Firebase Cloud Messaging.
- Firebase Real Time Database.
- Firebase Functions

Pré-Condições:

- Dados foram alterados na base de dados

Pós Condições:

1. Conclusão com sucesso:
    a. Push notification é enviada. Iniciando UC002
2. Conclusão sem sucesso:
    a. Mensagem não é enviada.

Fluxo básico:

1. Firebase Functions ativa função para alteração de dados na base de dados. A1
2. Função envia os dados do contato alterado por notificação push para o tópico do id do contato alterado, iniciando UC002.
3. Fim do caso de uso.

Fluxos alternativos:

A1: Contato foi inserido, não atualizado.

1. Firebase Functions ativa função para inserção de dados na base de dados.
2. Função envia os dados do contato alterado por notificação push para os tópicos da lista de contatos do Facebook do contato.
3. Fim do caso de uso.

UC006

Nome do caso de uso: Exibir detalhes.

Descrição:

Este caso de uso tem por objetivo exibir os dados de um contato da lista do usuário.

Eventos:

- Usuário escolhe contato.
- Aplicativo exibe dados do contato selecionado.

Atores:

- Usuário.
- Aplicativo

Pré-Condições:

- Usuário estar autenticado.

- Usuário possuir uma lista de contatos já baixada.

Pós Condições:

1. Conclusão com sucesso:
    a. Aplicativo exibe os dados do contato.
2. Conclusão sem sucesso:
    a. Dados do contato não são exibidos.

Fluxo básico:

1. Usuário escolhe a opção lista de contatos no menu do aplicativo.A1, A2, A3.
2. Aplicativo exibe a lista de contatos.
3. Usuário escolhe um contato. A1, A2, A3.
4. Aplicativo exibe os dados do contato escolhido.
5. Fim do caso de uso.

Fluxos alternativos:

A1: Usuário escolhe a opção Dados Pessoais.

1. Aplicativo exibe a tela de Dados Pessoais.
2. Fim do caso de uso.

A2: Usuário escolhe a opção Logout.

1. Aplicativo encerra a autenticação do usuário com a API do Facebook e Firebase.
2. Aplicativo exibe a tela de login.
3. Fim do caso de uso.

A3: Usuário escolhe a opção Sair.

1. Aplicativo encerra o funcionamento.
2. Fim do caso de uso.

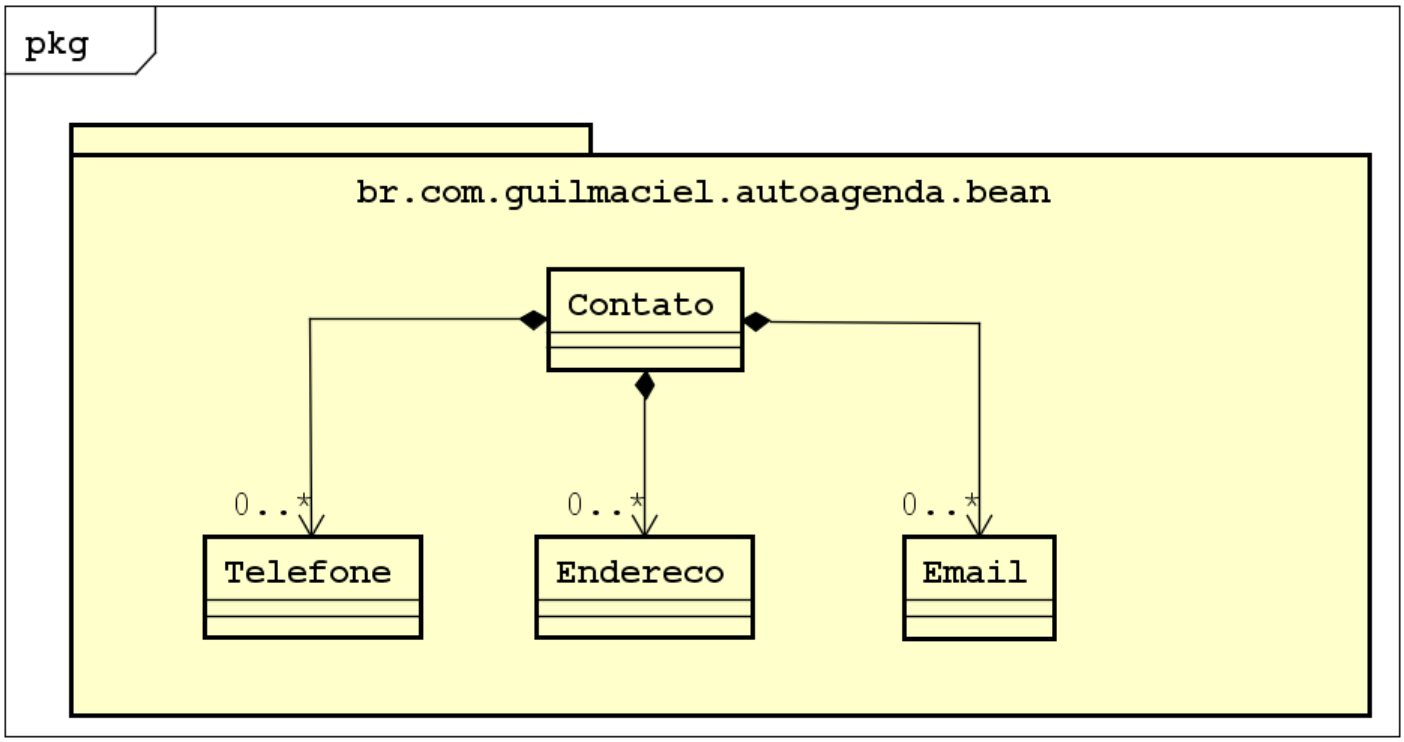
3.4 Diagrama de Classes

A representação do diagrama de classes, por questões de legibilidade, foi dividida em 3 partes, que são exibidas a seguir:

Model:

A Figura 3 apresenta a representação do simples modelo de dados que é utilizado, uma classe Contato, com 3 classes relacionadas por agregação com as classes Telefone, Endereco e Email.

Figura 3- Diagrama de classes, referente ao pacote bean do projeto Android.

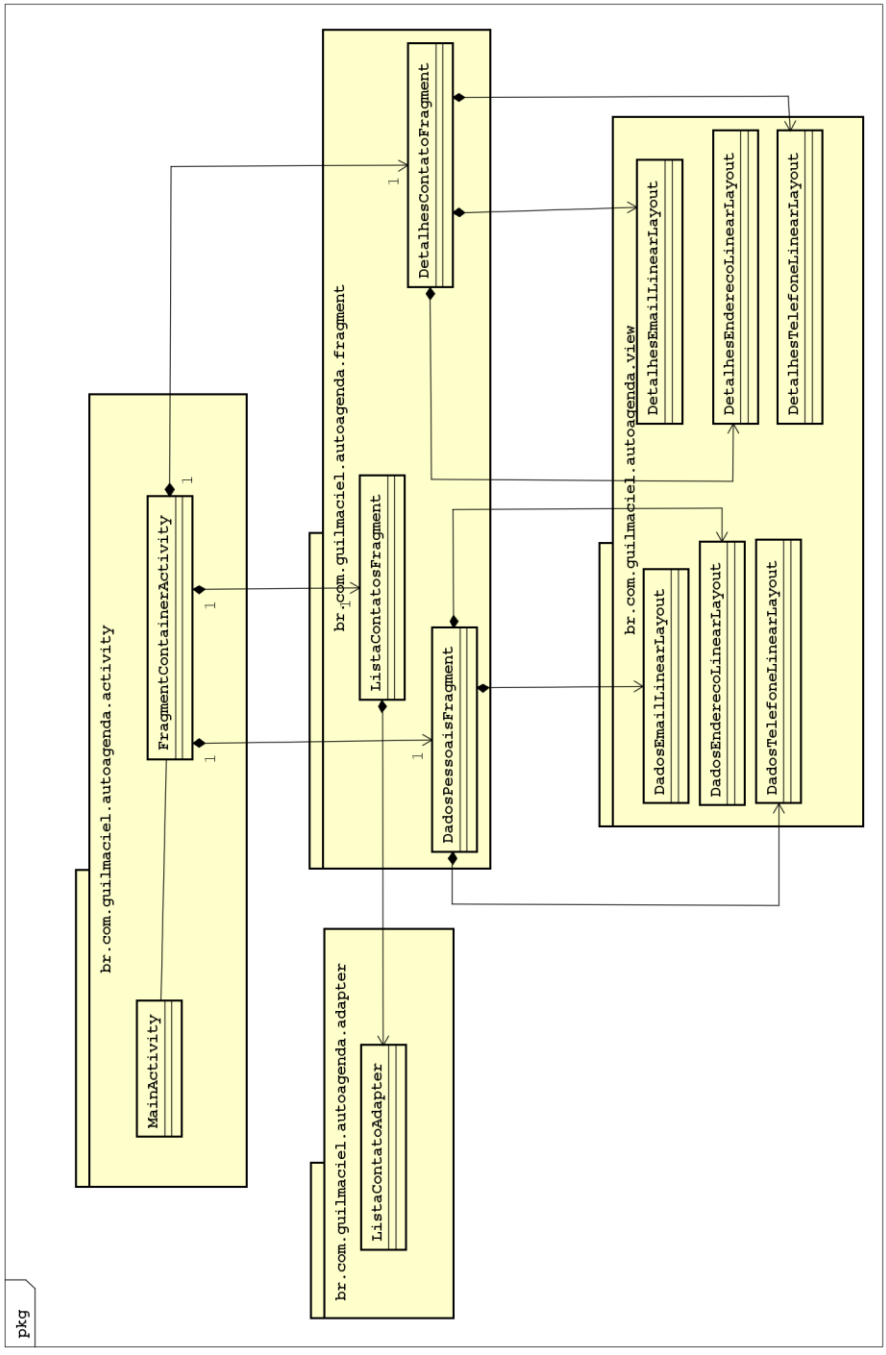


Fonte: O Autor (2017)

View:

A Figura 4 apresenta a representação das classes envolvidas na interface gráfica, mostrando as Agregações entre as classes de Activities com as de Fragments, as quais tem uma relação de agregação com classes de *layout* e *adapters*.

Figura 4 - Diagrama de classes, referente aos pacotes que compõe as classes utilizadas na interface gráfica do projeto Android

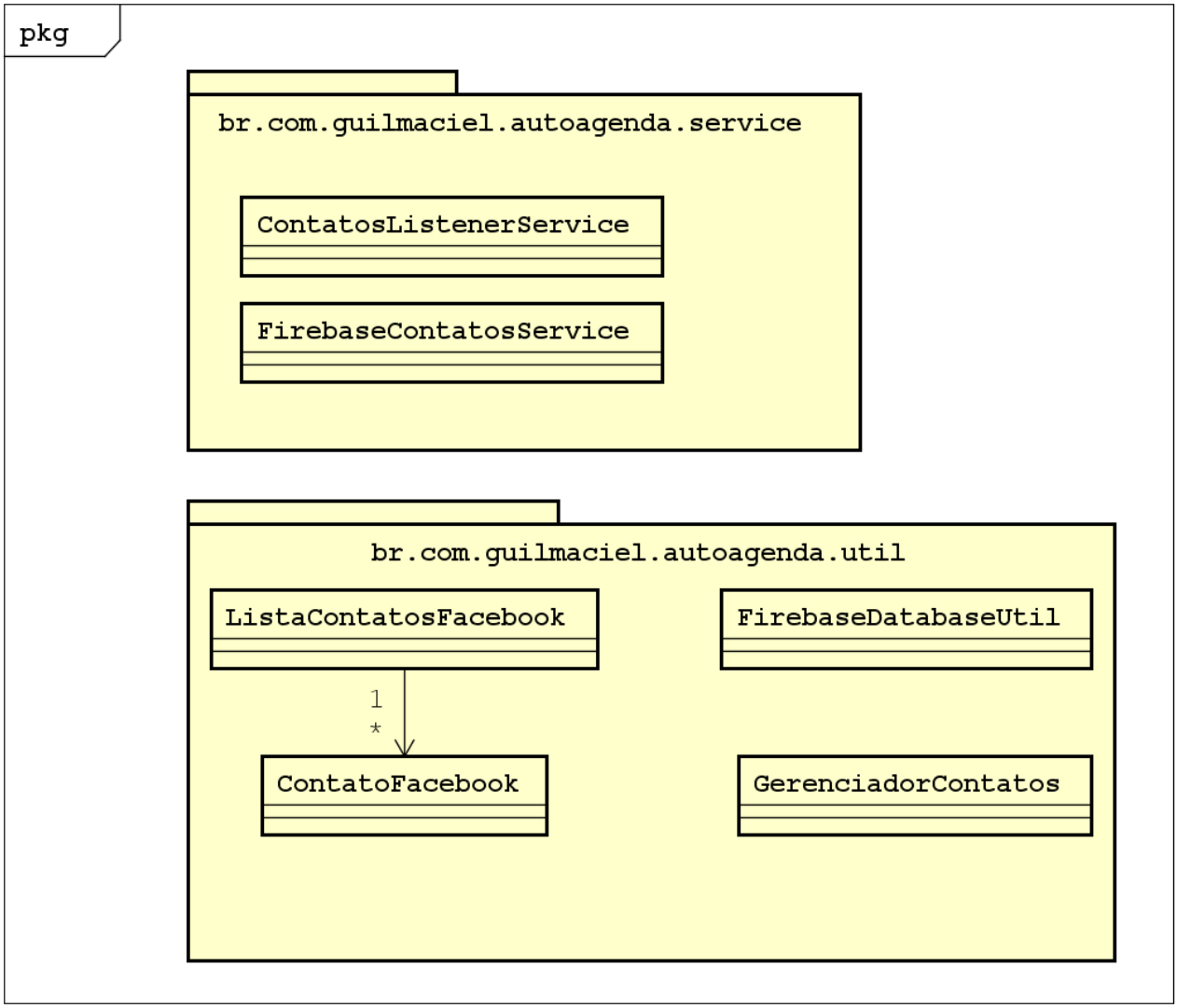


Fonte: O Autor (2017)

Util e Service:

O diagrama da Figura 5 exibe as classes que não possuem interações com as outras classes.

Figura 5 - Diagrama de classes, referente aos pacotes service e util.



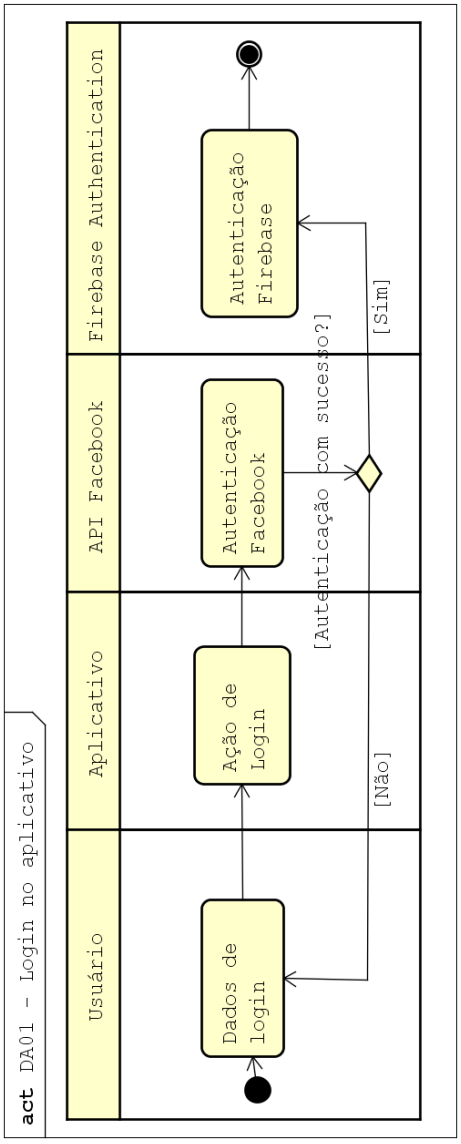
Fonte: O Autor (2017)

3.5 Diagrama de atividades

Foram criados 5 diagramas de atividade para mostrar o comportamento do sistema, assim como o usuário interage com o mesmo.

O diagrama de atividades representado na Figura 6 mostra o fluxo de login do aplicativo. Desde o usuário entrando com os dados de login, até a autenticação finalizada.

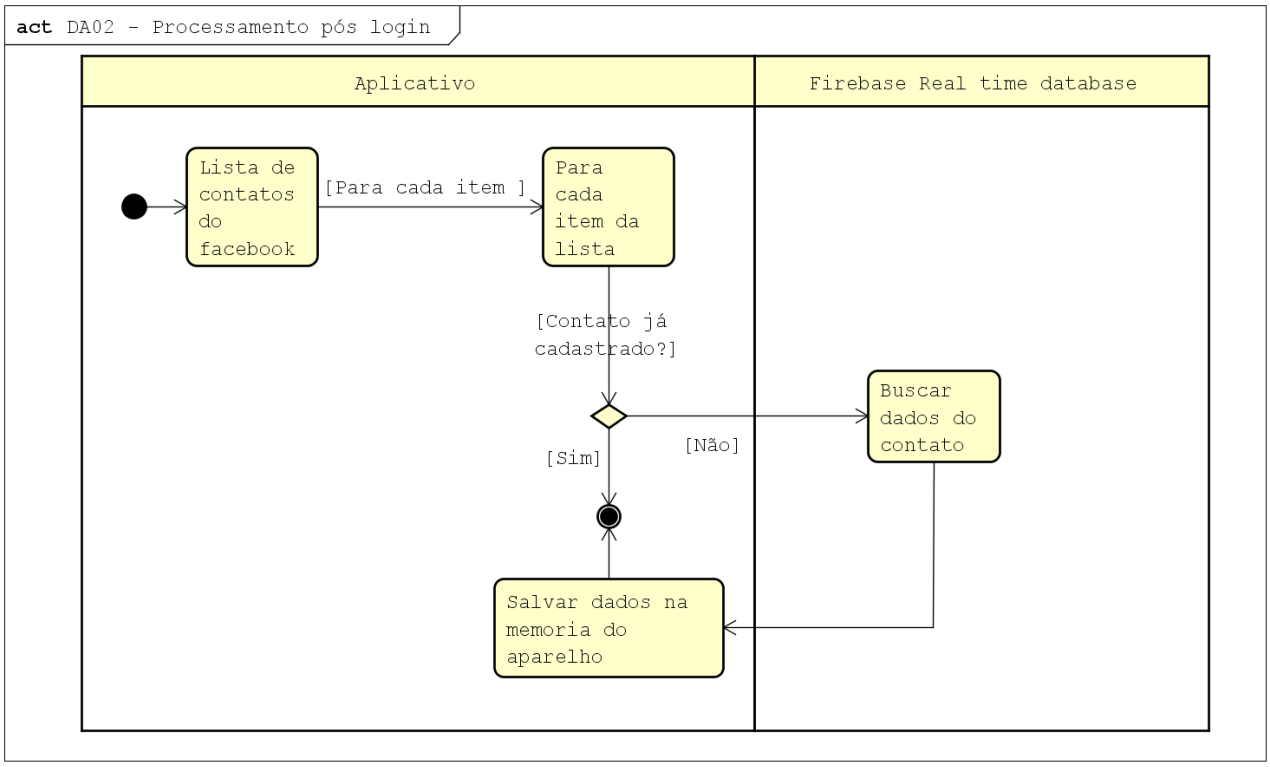
Figura 6 - Diagrama de atividades, referente ao fluxo de autenticação.



Fonte: O Autor (2017)

O diagrama de atividades representado na Figura 7 mostra o fluxo após o login no aplicativo. As atividades são executadas sem intervenção do usuário, com a sua conclusão exibindo a próxima tela ao usuário.

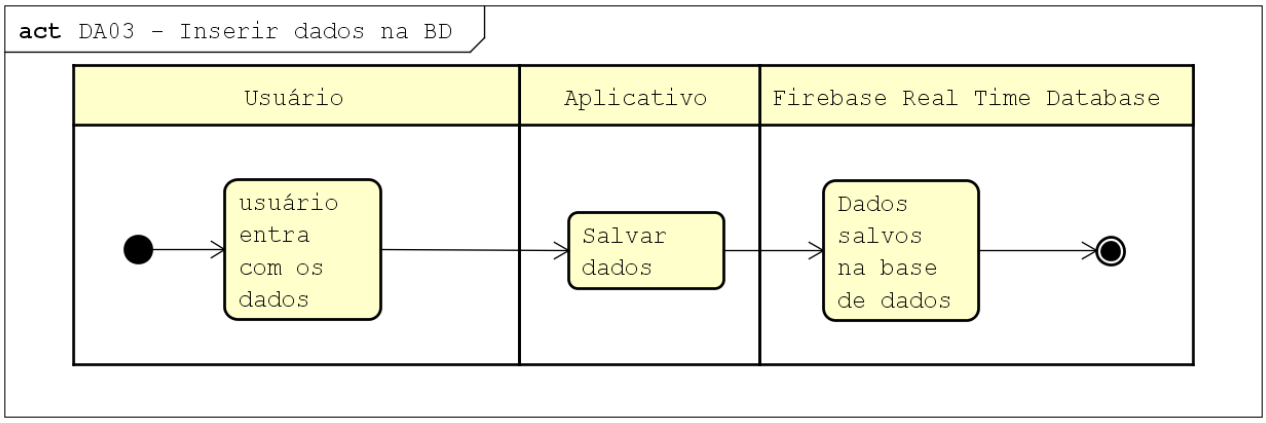
Figura 7 - Diagrama de atividades, referente ao fluxo pós autenticação.



Fonte: O Autor (2017)

O diagrama de atividades representado na Figura 8 mostra o fluxo de inserção de dados na base do Firebase Real Time Database.

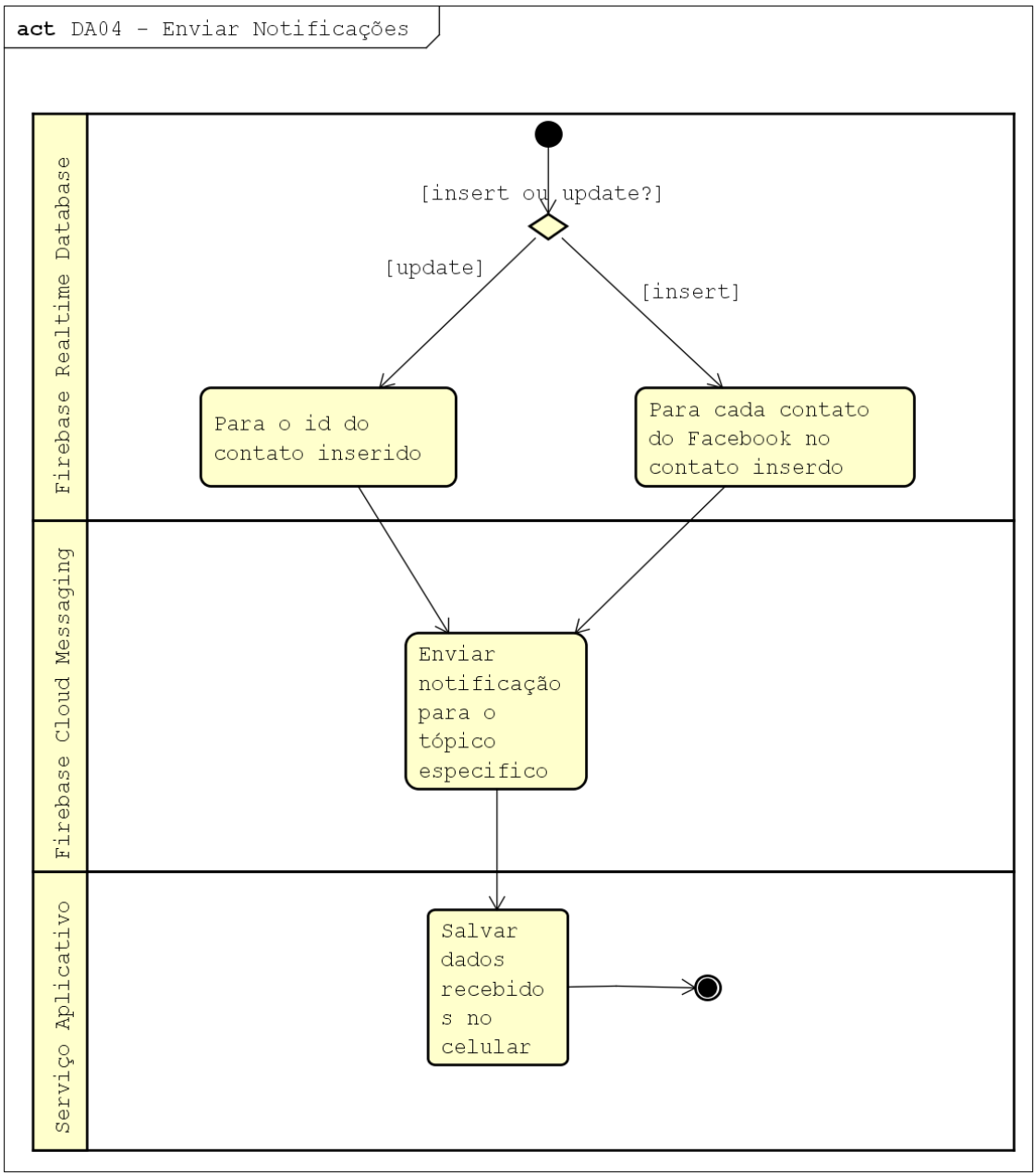
Figura 8- Diagrama de atividades referente ao fluxo de inserção de dados.



Fonte: O Autor (2017)

O diagrama de atividades representado na Figura 9 mostra o fluxo de envio de notificações pelo Firebase Cloud Messaging.

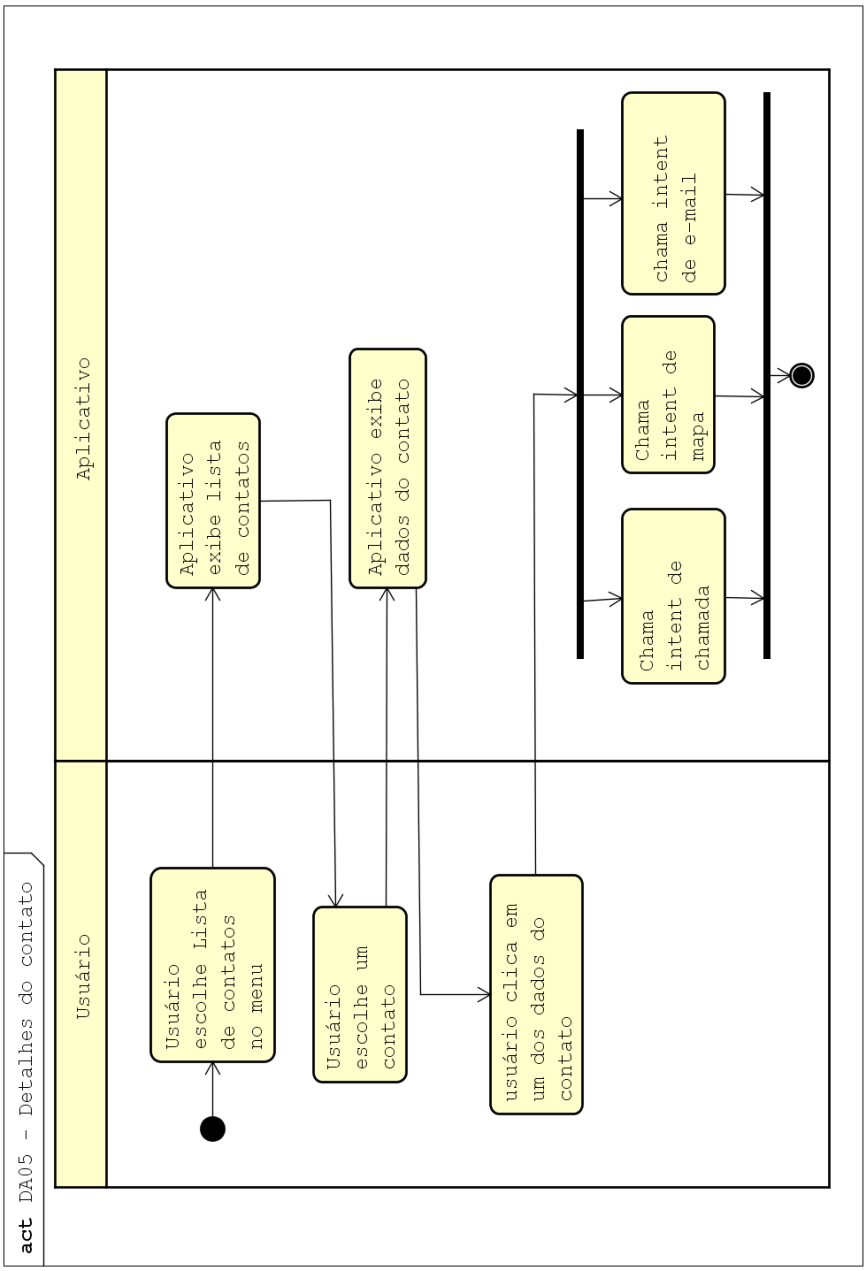
Figura 9 - Diagrama de atividades, referente ao envio de notificações.



Fonte: O Autor (2017)

O diagrama de atividades representado na Figura 10 mostra o fluxo de navegação de um usuário abrindo a tela de Detalhes do contato.

Figura 10 - Diagrama de atividades, referente ao fluxo de navegação.



Fonte: O Autor (2017)

3.6 Implementação

O Desenvolvimento deste aplicativo envolveu o uso de diversas tecnologias. O uso destas tecnologias se deu devido a pesquisas que mostraram que empregar as mesmas, facilitariam e acelerariam o processo de desenvolvimento deste projeto, também agregando conhecimento sobre as mesmas.

Nesta seção, é mostrado como a implementação destas tecnologias foi realizada.

3.6.1 Firebase Real Time Database

A implementação das funcionalidades de base de dados do Firebase no projeto Android se dá por meio da biblioteca “com.google.firebase:firebase-database:11.0.4”, que é adicionada ao arquivo Gradle (Como mostra a Figura 11).

Figura 11 – Dependências Firebase do arquivo build.gradle

```
dependencies {
    implementation 'com.google.firebase:firebase-database:11.0.4'
    implementation 'com.google.firebase:firebase-messaging:11.0.4'
    implementation 'com.google.firebase:firebase-auth:11.0.4'
    compile fileTree(dir: 'libs', include: ['*.jar'])
```

Fonte: O Autor (2017)

Com o uso da biblioteca, é possível o uso classes e funções que simplificam a manipulação dos dados presentes na base de dados na nuvem, a inserção de um dado na base de dados é possível com apenas 2 linhas, sendo necessário apenas uma instância da classe DatabaseReference, e o objeto a ser adicionado.

Como mostra a Figura 12, o processo de inserção é muito simples, no qual um novo item, com ID “id-contato”, é inserido na árvore JSON no nó chamado “contatos”. Se tanto o nó, quando o id não existirem, eles são criados no momento da inserção, se a combinação de nó e ID já existirem na base, os valores são atualizados pelo novo objeto informado.

Figura 12 - Exemplo de inserção de dados.

```
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( s: "contatos");
reference.child("id-contato").setValue(new Contato());
```

Fonte: O Autor (2017)

A ferramenta aceita objetos compostos e listas como parâmetro para inclusão na base de dados. Esperando um construtor vazio, e outro com todos os atributos a serem adicionados na base. A Figura 13 mostra um trecho da classe Contato, com seus atributos (três deles sendo listas de outros objetos), construtores e métodos.

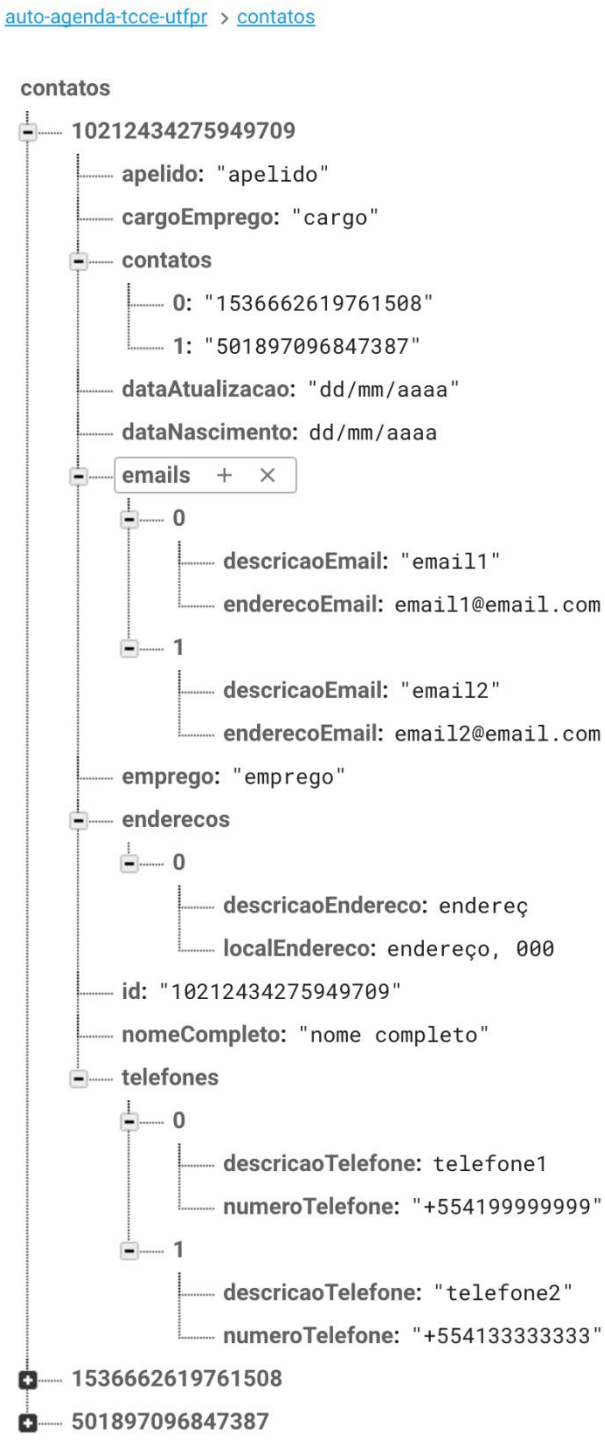
Figura 13 - Trecho da classe Contato.

```
package br.com.guilmaciel.autoagenda.bean;

import java.util.ArrayList;
import java.util.List;

public class Contato {

    private String id, nomeCompleto, apelido, cargoEmprego, dataAtualizacao, dataNascimento, emprego;
    private List<Endereco> enderecos;
    private List<Email> emails;
    private List<Telefone> telefones;
    private List<String> contatos;

    public Contato(){...}

    public void setId(String id) { this.id = id; }

    public void setNomeCompleto(String nomeCompleto) { this.nomeCompleto = nomeCompleto; }

    public void setApelido(String apelido) { this.apelido = apelido; }

    public void setCargoEmprego(String cargoEmprego) { this.cargoEmprego = cargoEmprego; }

    public void setDataAtualizacao(String dataAtualizacao) {...}

    public void setDataNascimento(String dataNascimento) { this.dataNascimento = dataNascimento; }

    public void setEmprego(String emprego) { this.emprego = emprego; }

    public void setEnderecos(List<Endereco> enderecos) { this.enderecos = enderecos; }

    public void setEmails(List<Email> emails) { this.emails = emails; }

    public void setTelefones(List<Telefone> telefones) { this.telefones = telefones; }
```

Fonte: O Autor (2017)

A Figura 14 mostra a estrutura da arvore JSON salva na base de dados, de uma instancia da classe Contato.

Figura 14 - Estrutura de dados salva na base de dados



Fonte: O Autor (2017)

A Leitura dos dados também se dá de forma simples, através de *event listeners* implementados em ids adicionados à base de dados, é possível esperar recuperar a última versão do dado na base de

dados, também como é possível criar um evento que é acionado sempre que o id especificado for modificado na base de dados.

A Figura 15 exibe o trecho de código da função para leitura única dos dados da base de dados, de uma entrada da base de dados.

Figura 15 – Event Listener de dados na base de dados.

```
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( s: "contatos");
reference.child("id-contato").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

        Contato contato = dataSnapshot.getValue(Contato.class);

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});
```

Fonte: O Autor (2017)

A Figura 16 exibe o trecho de código que cria um *listener* que escuta mudanças feitas em um item específico da base de dados.

Figura 16 - Event Listener de mudanças nos dados da base de dados.

```
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( s: "contatos");
reference.child("id-contato").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        Contato contato = dataSnapshot.getValue(Contato.class);
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});
```

Fonte: O Autor (2017)

3.6.2 Firebase Functions

A funcionalidade de Functions do Firebase não se integra diretamente com o Android, mas é voltada para executar tarefas pré-determinadas que são acionadas conforme gatilhos ocorrem no Projeto Firebase.

A funcionalidade específica utilizada neste projeto é a de envio de mensagens na nuvem, que tem como gatilho quando dados são incluídos ou alterados no nó “contatos” da base de dados.

As Figuras 17 e 18 mostram a lógica que controla essas funções.

A Figura 17 mostra a função ativada quando um dado novo é inserido na base de dados “contatos”.

- A linha 8 cria uma função de envio de notificações, para qualquer modificação efetuada no módulo “/contatos/usuário” na base de dados.
- A linha 9 armazena os dados do usuário alterado em uma variável.
- A linha 10 armazena o valor de identificação do usuário alterado em uma variável.
- Das linhas 13 a 18, é criada uma variável com o corpo da mensagem.
- Das linhas 20 a 31, é criada uma variável com as propriedades da notificação.
- Na linha 26, a lista de contatos é armazenada em uma variável
- Das linhas 29 a 33, a notificação é enviada usando as variáveis criadas nos passos anteriores, para cada item na lista de contatos do contato adicionado.

Figura 17 – Função acionada quando dados são adicionados à base de dados.

```
1  const functions = require('firebase-functions');
2
3  const admin = require('firebase-admin');
4  admin.initializeApp(functions.config().firebase);
5
6
7
8  exports.sendNotificationInsert = functions.database.ref('/contatos/{usuario}').onCreate(snapshot => {
9      const data = snapshot.data;
10     const id = snapshot.params.usuario;
11     console.log('Mensagem recebida.');
12
13     const payLoad = {
14         data:{
15             title: id,
16             body: JSON.stringify(data.val())
17         }
18     };
19
20     const options = {
21         priority: "high",
22         timeToLive: 60*60*2
23     };
24
25
26     const contatos = data.child('contatos').val();
27
28
29     for (let contato in contatos){
30         let topic = contatos[contato] + "-insert";
31         console.log(contatos[contato] + "-insert");
32         admin.messaging().sendToTopic(topic, payLoad, options);
33     }
34     return true;
35
36 });
```

Fonte: O autor (2017)

A Figura 18 mostra a função ativada quando um dado já existente é alterado na base de dados “contatos”.

- A linha 8 cria uma função de envio de notificações, para qualquer modificação efetuada no módulo “/contatos/usuário” na base de dados.
- A linha 9 armazena os dados do usuário alterado em uma variável.
- A linha 10 armazena o valor de identificação do usuário alterado em uma variável.
- Das linhas 11 a 16, é criada uma variável com o corpo da mensagem.
- Das linhas 18 a 21, é criada uma variável com as propriedades da notificação.
- Na linha 23, a notificação é enviada usando as variáveis criadas nos passos anteriores.

Figura 18 – Função acionada quando dados são alterados na base de dados

```
1   const functions = require('firebase-functions');
2
3   const admin = require('firebase-admin');
4   admin.initializeApp(functions.config().firebase);
5
6
7
8    exports.sendNotification = functions.database.ref('/contatos/{usuario}').onUpdate(event => {
9      const data = event.data;
10     const id = event.params.usuario;
11     const payLoad = {
12         data:{
13             title: id,
14             body: JSON.stringify(data.val())
15         }
16     };
17
18     const options = {
19         priority: "high",
20         timeToLive: 60*60*2
21     };
22
23     return admin.messaging().sendToTopic(id, payLoad, options);
24 });
```

Fonte: O Autor (2017)

3.7 Telas.

O Sistema é composto por 4 telas e um menu, as mesmas estão mostradas a seguir, com uma breve descrição em sequência.

Tela inicial Login:

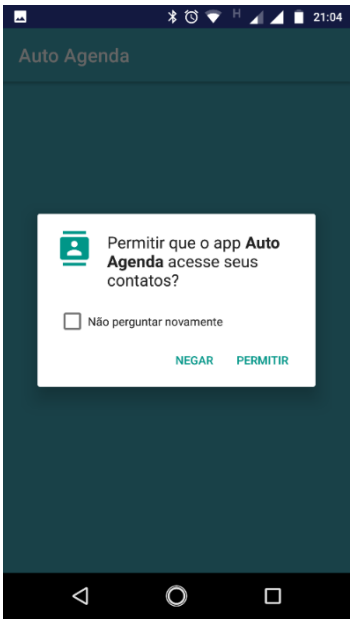
Nesta tela, o usuário encontrará o botão necessário para a autenticação com sua conta do Facebook (Figuras 19 e 20), nessa tela também é realizada a verificação se o usuário concedeu

permissões de acesso necessárias ao aplicativo (Figuras 21 e 22) (Leitura e escrita dos contatos, e acesso à internet).

Caso o usuário já tenha uma conta do Facebook registrada na opção de contas do aparelho, e as permissões já tiveram sido concedidas previamente, o sistema autenticará o usuário automaticamente, (Não acontecendo as situações das figuras 19, 20 e 21) não sendo necessária nenhuma interação do usuário com a tela.

A Figura 19 exibe a tela de login, solicitando as permissões necessárias ao funcionamento do aplicativo ao usuário.

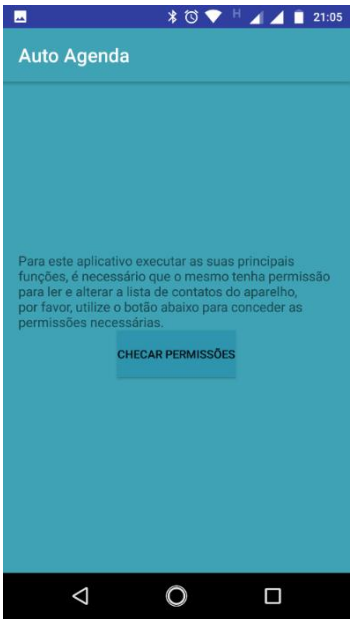
Figura 19 – Tela inicial, em processo de requisição de permissões necessárias para o funcionamento do aplicativo



Fonte: O Autor (2017)

A Figura 20 exibe a tela de login, mostrando uma mensagem explicando a necessidade das permissões para o funcionamento do aplicativo, assim como um botão que reexibe a tela solicitando as permissões.

Figura 20 – Tela inicial com a mensagem a ser exibida caso o usuário não tenha concedido as permissões necessárias.

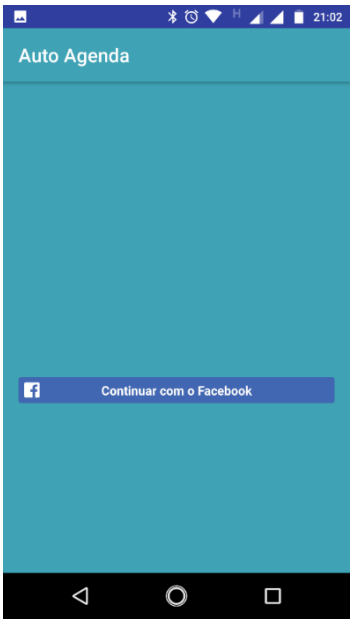


Fonte: O Autor (2017)

‘

A Figura 21 exibe a tela de login, após as permissões terem sido concedidas, mostrando o botão de login.

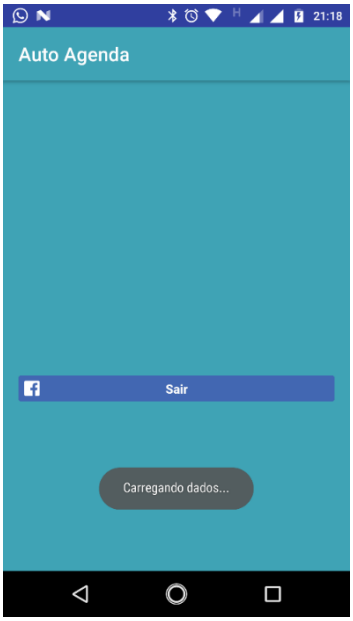
Figura 21 – Tela inicial, após as permissões necessárias serem garantidas, antes de iniciado o processo de login.



Fonte: O Autor (2017)

A Figura 22 exibe a tela de login, durante o processo de autenticação.

Figura 22 – Tela inicial, após as permissões necessárias serem garantidas e iniciado o processo de login.



Fonte: O Autor (2017

Dados Pessoais:

Esta tela é exibida logo após o login ser efetuado, ou quando o usuário escolhe a opção de mesmo nome no menu. No caso de um primeiro acesso de um usuário, essa tela será exibida com os campos em branco, caso contrário, os dados salvos anteriormente serão exibidos nos respectivos campos. É possível adicionar novos campos para o preenchimento de telefones, e-mails e endereços (por meio dos botões “Adicionar ...”), também é possível remover campos vazios, ou já preenchidos, através do botão de exclusão, presente em cada linha.

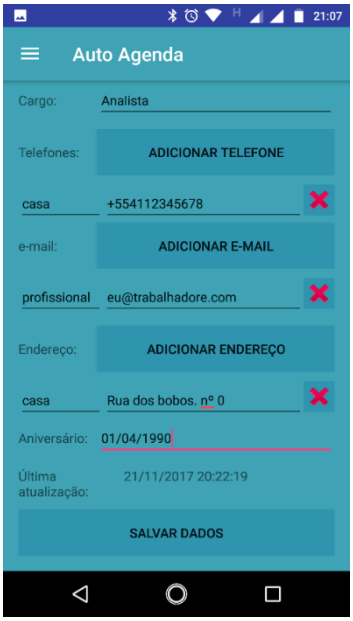
Qualquer mudança só é replicada para a base de dados, e consequentemente, para os outros usuários, quando o botão “Salvar Dados” for acionado, e a confirmação das alterações for confirmada pelo usuário (confirmação representada na Figura 23).

Figura 23 – Tela de dados do usuário.

Fonte: O Autor (2017)

A Figura 24 exibe o final da tela de dados do usuário, onde é possível encontrar o botão para salvar os dados alterados.

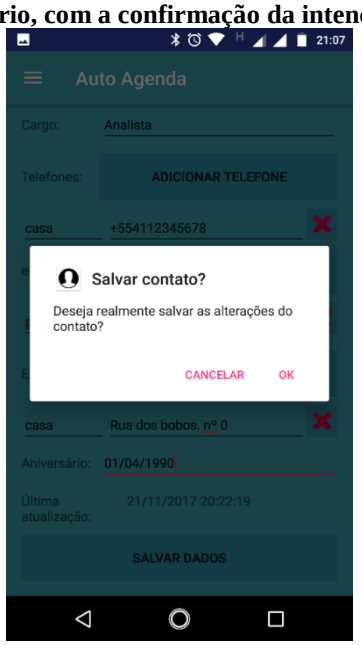
Figura 24 – Tela de dados do usuário.



Fonte: O Autor (2017)

A Figura 25 mostra a tela de Dados do usuário após os dados serem salvos.

Figura 25 – Tela de dados do usuário, com a confirmação da intenção de alterar os dados do mesmo.

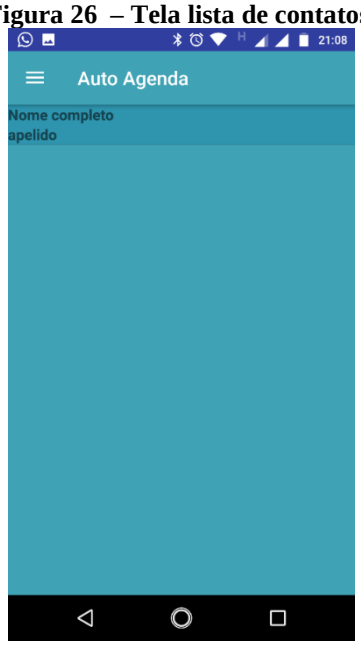


Fonte: O Autor (2017)

Lista de contatos:

Esta é a tela mais simples do aplicativo, na qual se escolhe um contato da lista, para serem exibidos os seus dados. Nesta lista são exibidos somente os contatos recebidos pelo Aplicativo e Serviço desenvolvidos neste projeto, os contatos adicionados por outros modos no aparelho, não são exibidos nesta tela.

Figura 26 – Tela lista de contatos.



Fonte: O Autor (2017)

Detalhes do contato:

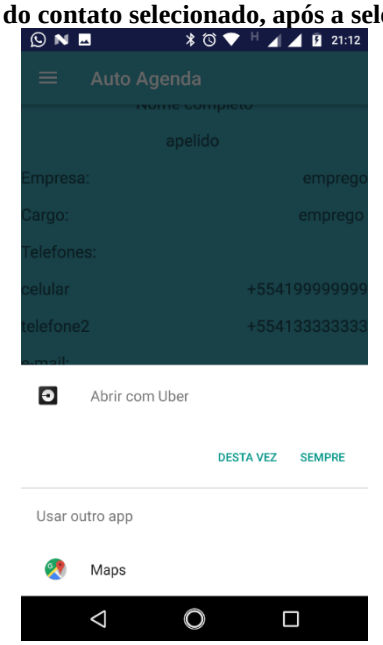
Nesta tela, estão presentes os dados preenchidos por outros usuários, que fazem parte da lista de contatos do usuário na rede social Facebook, clicando nos telefones, e-mails ou endereços listados, o aplicativo exibirá uma lista de aplicativos previamente instalados, relacionados a opção escolhida, para uso do usuário.

Figura 27 – Tela de detalhes do contato selecionado.

Fonte: O Autor (2017)

No exemplo de tela da Figura 28, foram exibidas as opções de aplicativos disponíveis relacionados a geolocalização.

Figura 28 – Tela de detalhes do contato selecionado, após a seleção de um endereço da lista.



Fonte: O Autor (2017)

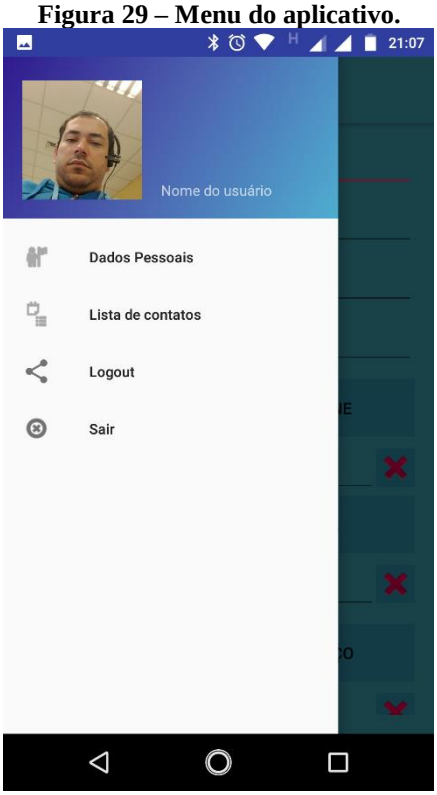
Menu:

Como o botão de voltar foi desativado nas telas, o menu é a única forma de navegação no aplicativo. Através dele, é possível navegar entre as telas Dados pessoais e Lista de contatos (a tela Detalhes do contato só é acessível através da tela Lista de contatos).

Por meio do menu, também é possível fazer logout da conta do Facebook, assim como fechar o aplicativo.

O Menu também exibe a atual foto do perfil do Facebook do usuário, assim como o nome que o mesmo salvou no aplicativo.

Figura 29 – Menu do aplicativo.



Fonte: O Autor (2017)

3.8 Avaliação

Ao final da elaboração deste projeto, o desenvolvimento está na fase de “Testes e Validações” descrita no método, no Capítulo 1.5. As validações ocorreram com usuários previamente selecionados, com relativo conhecimento em tecnologia móvel.

O modo de avaliação utilizado foi o de deixar o usuário utilizar o aplicativo, sem instruções prévias, analisando as ações tomadas, e, após este uso, explicar as funcionalidades e ouvir opiniões, sugestões. O objetivo de não explicar como o aplicativo funciona ajuda a prevenir possíveis “vícios” que um usuário com conhecimento prévio possa ter, afetando no resultado dos testes.

O resultado destas avaliações mostra que alguns ajustes ainda se fazem necessários, entre eles estão:

- Melhoras na navegação do aplicativo.

Como o botão de voltar foi desabilitado no aplicativo, os usuários instintivamente usando esse botão, acreditam que o aplicativo apresenta erros. Faz-se necessário, ou uma mensagem avisando da indisponibilidade do botão, ou reativá-lo.

- Melhoras no tempo de carregamento dos dados.

Como os dados são carregados da nuvem, a velocidade de conexão do usuário será um fator determinante na velocidade que os mesmos são carregados, porém, uma análise mais precisa pode mitigar essa lentidão, através de requisições de dados mais eficientes.

- Melhoras no *layout* do aplicativo.

Algumas telas podem ser melhor modeladas, para melhor aproveitar espaços, tamanhos de botões e textos também podem ser revistos.

- Funcionalidades com comportamento fora do esperado.

Algumas funcionalidades precisam ser revisadas, para que apresentem o comportamento esperado, garantindo a coesão do aplicativo.

O aplicativo também agradou os usuários com algumas de suas funcionalidades, o fato de usuário poder usar sua conta do Facebook, sem precisar criar cadastros novos, é um ponto considerado positivo pelos mesmos. Também foi visto positivamente o fato do aplicativo listar os contatos obtidos diretamente por ele, também adicionando esses mesmos contatos na lista de contatos padrão do celular.

Porém o ponto de maior aceitação, foi o foco principal do projeto, a troca automática de informações entre os contatos, em tempo real, de informações pessoais.

4 Conclusão

Apesar de ainda não finalizado, o Aplicativo já cumpre o objetivo proposto de controlar como a transferência de contatos do usuário pode ser feita, na qual alterações feitas por um usuário são repassadas automaticamente à sua lista de contatos.

O modelo da solução desenvolvida baseia-se na integração dos projetos Android e Firebase, criando um produto único que atenda às necessidades que o projeto se propôs a resolver.

A fase de desenvolvimento mostrou algumas surpresas, pontos que seriam considerados relativamente fáceis de serem desenvolvidos, como a manutenção dos contatos na agenda do celular, se mostraram um dos itens de maior dificuldade no desenvolvimento. Um ponto que se mostrava trabalhoso seria o desenvolvimento do web service, que seria responsável pela integração de dados entre usuários, se mostrou simples, quando um estudo mais aprofundado foi feito com relação as ferramentas disponibilizadas pela plataforma Firebase. As tecnologias de funções e base de dados, que foram utilizadas no projeto, mostraram que o desenvolvimento do web service poderia ser feito utilizando as mesmas.

Outro ponto de dificuldade, foi encontrar artigos, pesquisas na literatura que abordassem o tema proposto, essa dificuldade de encontrar estes textos de apoio, limitou a maturação e desenvolvimento da solução à experiência do autor, e possíveis *feedbacks* de seus conhecidos.

Esses *feedbacks* não afetaram a versão final do aplicativo, mas indicam correções que não afetam o propósito do projeto, propósito este, que foi fortemente elogiado nas avaliações

4.1.1 Trabalhos Futuros

Alguns passos se mostram necessários, outros inevitáveis de serem tomados para o avanço e futuro do aplicativo, entre eles:

- Correção das sugestões encontradas na fase de Testes e Validações no desenvolvimento do aplicativo.
- Disponibilizar o aplicativo na loja Google Play
- Trabalhar em modos de monetizar o aplicativo, muito provavelmente com propagandas, e possivelmente uma versão paga, sem os anúncios.
- Desenvolvimento da versão IOS do aplicativo.

- Criação de uma página Web, na qual é possível ver os dados da lista de contatos do usuário, sem a necessidade do aplicativo no celular.
- Método de transferência de dados off-line.

Além dos itens citados, também se mostra necessária a evolução dos dados disponíveis no aplicativo, pois com o possível surgimento de “novos dados”, estes dados poderão ser adicionados na lista de opções do aplicativo; assim como a consideração das avaliações feitas pelo usuário final nas lojas de aplicativos poderão levar a alterações e terão papel determinante no futuro do aplicativo.

5 Referências

ARUN K, SELVARATHY T. **A Review on Building Social Networking APIs**. Disponível em http://www.ijiset.com/v1s3/IJISET_V1_I3_20.pdf Acesso em 17 out. 2017.

BRITO, GABRIEL M. DE; VELLOSO, PEDRO B.; MORAES, IGOR M. **Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet**. Disponível em < https://inf.ufes.br/~magnos/IF/if_files/MC2012.pdf > Acesso em 10 set. 2017.

CENTRO REGIONAL DE ESTUDOS PARA O DESENVOLVIMENTO DA SOCIEDADE DA INFORMAÇÃO. **Apresentação dos principais resultados TIC Domicílios 2015**. Disponível em: < http://cetic.br/media/analises/tic_domicilios_2015_coletiva_de_imprensa.pdf > Acesso em 10 set. 2017.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology, 2016-2021** Disponivel em < https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf > Acesso em 10 set. 2017.

CONSELHO DE DIREITOS HUMANOS DA ONU. **Promotion and protection of all human rights, civil, political, economic, social and cultural rights, including the right to development, 2011.** Disponível em: < http://www2.ohchr.org/english/bodies/hrcouncil/docs/17session/A.HRC.17.27_en.pdf >. Acesso em 10 set. 2017.

FARR, CHRISTINA. **Firebase's scalable backend makes it '10 times easier' to build apps**. Disponível em < https://venturebeat.com/2013/02/13/firebases-backend-makes-it-ten-times-easier-to-build-apps/ > Acesso em 21 set. 2017.

JUNIOR, FRANCISCO DE ASSIS RIBEIRO. **Programação Orientada a Eventos no lado do servidor utilizando Node.js**. Disponível em <http://infobrasil.inf.br/userfiles/16-S3-3-97136-Programa%C3%A7%C3%A3o Orientada___.pdf> Acesso em 21 set. 2017.

KUMAR, K.N. MANOJ; AKH, KAILASA; GUNTI, SAI KUMAR; REDDY, M. SAI PRATHAP. **Implementing smart home using Firebase**. Disponivel em http://euroasiapub.org/wp-content/uploads/2016/11/16EASOct-4186-1.pdf Acesso em 15 out 2017.

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com Android SDK**. São Paulo: Novatec Editora, 2009.

MCLAUGHLIN, BRETT. **Java & XML, 2nd Edition**. Sebastopol, O'Reilly, 2001.

NASCIMENTO, JEAN. **3 razões para usar MongoDB**. Disponível em <https://imasters.com.br/artigo/18334/mongodb/3-razoes-para-usar-mongodb/?trace=1519021197&source=admin> Acesso em 21 set. 2017.

PASCUAL, VICENT SEGUÍ, XHAFA, FATOS. **Evaluation of contact synchronization algorithms for the Android platform**. Disponível em <http://ac.els-cdn.com/S0895717711008132/1-s2.0-S0895717711008132-main.pdf?_tid=3717bf50-968b-11e7-b993-00000aab0f6c&acdnat=1505091212_254795d9ad87bb4b53d5252722beaf47> Acesso em 10 set. 2017.

PLANGI, SIIM. **Overview of Backend as a Service platforms**. Disponível em <https://pdfs.semanticscholar.org/80d2/5c42b1d9f86b03804fc2aac6e0c16041631b.pdf> Acesso em 15 out. 2017.

SILBERSCHATZ, ABRAHAM; GALVIN, PETER BAER, GAGNE, GREG. **Operating System Concepts Essentials**. Disponível em < http://lib.sgu.edu.vn:84/dspace/bitstream/TTHLDHSG/2808/1/Operating_System_Concepts_Essentials.pdf> Acesso em 21 set. 2017.