MYTREAT: UM APLICATIVO PARA O AUXÍLIO E CONTROLE DE INGESTÃO DE MEDICAMENTOS

MyTreat: An Application to Manage and Control of Medication Intake

DIONI DA ROSA [1]*, MAURÍCIO SULZBACH [1]

[1]Universidade Regional do Alto Uruguai e da Missões (URI) – Frederico Westphalen – RS – Brasil.
*dioni@uri.edu.br

Resumo: A crescente evolução da tecnologia e a redução de custos dos dispositivos móveis tem permitido a criação, adaptação e execução de aplicações com vários propósitos, dentre elas, aplicações destinadas a área da saúde e bem-estar, como tratamentos pós-consultas médicas de usuários. Estas aplicações são conhecidas como mHealths, e auxiliam seus usuários em atividades como caminhadas, orientação de medicamentos, notificações e lembretes. Neste sentido, o presente trabalho tem como escopo abordar a criação de um aplicativo denominado MyTreat, com o intuito de auxiliar o usuário a ingerir seus medicamentos nos horários determinados, fazendo uso dos recursos disponibilizados nos dispositivos móveis para o agendamento e geração de notificações, evitando assim o esquecimento e auxiliando na saúde de seus usuários. O aplicativo obteve êxito em seus testes, sendo utilizado em emuladores disponível no kit de desenvolvimento de software (SDK) com a versões 4.1, 4.3, 5 e 6 do Android e um Smartphone com a versão 4.3.

Palavras-chave: mHealths, Saúde, MyTreat, agendamento e geração de notificações, Dispositivos Móveis.

Abstract: The growing evolution of technology and cost reduction of mobile devices has allowed the creation, adaptation and implementation of applications for various purposes, which may be applications for the health and well-being, such as medical post-consultation treatments. These applications are known as mHealths, and assist its members in activities such as hiking, guidance drugs, notifications and reminders. In this sense, this work has the scope the creation of an application in order to help the user consume their drugs called MyTreat, in certain times, making use of the resources on mobile devices, generating notifications to user, thus avoiding forgetfulness of it, to consume their drugs. The application has it was tested, being utilized in emulators available in the software development kit (SDK) with versions 4.1, 4.3, 5 and 6 and an Android Smartphone with version 4.3.

Keywords: mHealths, Health, MyTreat, Scheduling and Generating notifications, Mobile Devices.

1 INTRODUÇÃO

A utilização dos dispositivos móveis para diversas áreas cresce cada vez mais, podendo representar até a maioria dos sistemas computacionais utilizados no dia a dia das pessoas, pois são dispositivos que em grande parte possuem um baixo custo, são portáveis e de fácil manuseio. Entretanto criar uma aplicação para esses dispositivos traz vários desafios. Um deles é a otimização da utilização de recursos, pois em sua maioria, possuem pouco espaço de memória, processamento limitado e tela pequena, além de uma bateria com pouca duração. Outro aspecto a considerar é o contexto e o ambiente onde a aplicação será utilizada, pois a influência da claridade no ambiente pode dificultar a utilização de uma aplicação. Todos os aspectos para a criação de uma aplicação devem ser considerados, desde seu objetivo principal até o manuseio do futuro usuário, para que este possa ter uma boa experiência na utilização da aplicação e que esta aplicação contemple seu objetivo.

Para que o usuário se sinta confortável ao utilizar uma aplicação à mesma deve estar bem definida e ter um propósito, aplicações como *mHealth*(*Mobile Health*) também conhecidas como *eHealth*(*Electronic Health*), estão em crescimento e sendo muito utilizadas por usuários de diversos sistemas operacionais móveis. De acordo com TIZATTO (2014), pode-se enfrentar dificuldades no atendimento a pacientes, devido à grande demanda e a falta de profissionais no mercado. O autor também enfoca que a utilização dos dispositivos móveis para prover soluções na área da saúde pode auxiliar no controle de doenças infecciosas ou endêmicas, tais como a dengue e a gripe H1N1. Essas aplicações destinam-se a o auxílio da medicina e serviços de saúde através de dispositivos móveis. No Brasil a oferta e uso desses aplicativos cresce juntamente com a popularização dos Smartphones, sendo que a grande maioria das aplicações existentes para dispositivos os dispositivos utilizam o sistema operacional (SO) Android. De acordo com informações disponibilizadas pela Kantar WorldPanel Com Tech especialista mundial no comportamento dos consumidores, a parcela de utilizadores do Android no Brasil é de 92.4% em janeiro de 2016, assim os demais SO disponível como o iOS e Windows ocupam 3.3% e 4.1% e os outros que ficam com 0.1% da fatia do mercado (WORLDPANEL, 2016). Desta forma é notável que muitos desenvolvedores de aplicativos foquem suas aplicações para o sistema líder.

A estas considerações é possível pautar a relevância do desenvolvimento de uma aplicação que ofereça auxílio no tratamento pós consulta de um paciente, orientando o mesmo a fazer a ingestão dos seus medicamentos de forma correta e em seus horários estabelecidos, evitando assim, por muitas vezes, o esquecimento, o qual pode acarretar em um tratamento malsucedido. Também levando em consideração a grande porcentagem de brasileiros que utilizam o SO Android o mesmo foi escolhido para o desenvolvimento da aplicação, levando assim a mesma para

a maior parcela do mercado de consumidores presente no Brasil.

No mesmo sentido é importante expor a escolha pelo SO Android para servir ao desenvolvimento da aplicação, considerando-se a popularidade que o mesmo possui no Brasil em utilização e consumo. O artigo está organizado em seis sessões. A segunda sessão apresenta os trabalhos relacionados. A terceira apresenta um breve apanhado sobre o assunto tratado e as tecnologias utilizadas. O quarto expõe a metodologia utilizada. A quinta demonstra os recursos e o aplicativo elaborado e a sexta expõe as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Tecnologias móveis estão em contínuo avanço em termos de disponibilidade, funcionalidade e custos, tornando sua utilização mais atraente. Essas tecnologias permitem o desenvolvimento de uma grande variedade de aplicações, sendo estas delineadas pelos recursos disponíveis nos dispositivos.

As aplicações podem ser desenvolvidas de forma nativa as quais servem para aparelhos ou sistema operacional específico, ou via aplicações web que podem ser acessadas de qualquer aparelho. Ambas se valem de interfaces adaptadas, que utilizam recursos disponíveis nos aparelhos como recurso de interface, GPS e WI-FI. Com a melhoria dos meios de comunicação e a facilidade na aquisição de dispositivos móveis, a utilização desses dispositivos cresceu rapidamente, devido comodidade e à mobilidade empregada neles.

Também melhorias na tecnologia para um melhor desempenho foram empregadas a esses dispositivos, dando oportunidade para o desenvolvedor inovar e gerando facilidade na utilização das aplicações elaboradas, além, da adaptação de processos que eram feitos apenas em desktops locais para serem utilizados em aparelhos móveis. Outro fator que está aumentando é a visibilidade dos desenvolvedores e empresas para esse mercado onde o investimento em tecnologias e a criação de software para o mercado eletrônico móvel começa a ser um investimento de diversas empresas, abrindo assim novas fronteiras e oportunidades de negócio e gerenciamento, os quais visam facilitar o acesso e uso dos usuários (HSIEH, 2007).

2.1 Sistema Operacional Android

O Android é o SO mais popular do mercado de Smartphones, sendo baseado em Linux, criado pela Google e direcionado a dispositivos móveis. Apesar de o sistema ser anunciado como um sistema da Google para dispositivos móveis, a responsabilidade do desenvolvimento do mesmo é da Open Handset Alliance, um conjunto de 47 empresas do ramo tecnológico e das telecomunicações. A plataforma é suportada por vários modelos de dispositivos móveis, como tablets e de smartphones. O Android suporta uma grande variedade de tecnologias de conectividade incluindo Bluetooth, EDGE, 3G e Wi-Fi. O Android é totalmente capaz de fazer uso de câmaras de vídeo, touchscreen, GPS, acelerômetros, e aceleração de gráficos 3D melhorando a experiência do usuário ao utilizá-lo. A sua versão atual é a 6.0 e chamada de marshmallow (ANDROID, 2016).

O SO Android utiliza como meio de armazenamento de dados o SQLite, sendo uma biblioteca em linguagem C, *open source*, que implementa um banco de dados SQL (*Structured Query Language*) embutido, não possuindo um processo rodando em um servidor separadamente, apenas faz leitura e escrita em um arquivo de banco de dados diretamente no disco. O Android utiliza o SQLite por ser uma biblioteca leve, fazendo uso entre 250kb e 500kb, sendo multiplataforma, onde se pode copiar livremente um banco de dados entre sistemas de 32bits e 64bits (MEDEIROS,2016).

2.2 Ferramentas de desenvolvimento

O Android Studio é a ferramenta oficial para o desenvolvimento de aplicativos para a plataforma Android. Esta ferramenta disponibiliza vários recursos para o desenvolvimento como opções de emuladores para o desenvolvimento e teste das aplicações. A mesma vem integrada com o SDK (*Software Development Kit*) atualizado com a versão mais recente do SO disponível no mercado sendo ela a versão 6.0 com codinome marshmallow, então dessa forma se fez necessário integração da versão 4.1 especialmente, pois será a versão de entrada para aplicação (ANDROID, 2016).

De acordo com desenvolvedor (Google), a ferramenta Android Studio necessita dos seguintes requisitos mínimos para sua instalação em SO Windows, SO Windows 7 ou superior, 2Gb de memória RAM, 4GB de espaço em disco para a instalação da ferramenta, uma tela com resolução mínima de 1280x800, JDK (*Java Development Kit*) 8 instalado e processador com suporte para virtualização (ANDROID, 2016).

Muitas áreas fazem a utilização da plataforma Android para desenvolvimento de aplicativos e disponibilização para usuários. Sendo que a saúde tem um conceito especifico, denominado mHealth o qual será abordado a seguir.

2.3 Mobile Health

De acordo com a OMS (Organização Mundial da Saúde) (2011) o *mHealth* é um componente da saúde que visa o auxílio através dos dispositivos móveis para as diversas áreas da saúde. Uma melhor definição foi estabelecida pelo Observatório Global para eHealth (GOE), que define *mHealth* ou aplicações móveis para saúde como prática médica e de saúde pública suportada por dispositivos móveis como telefones celulares, dispositivos de monitoramento, assistentes digitais pessoais (PDAs) e outros dispositivos sem fio. *mHealth* envolve o uso de recurso de um telefone/smartphone como o de voz e de serviço de mensagens curtas (SMS), bem como outras funcionalidades e aplicações mais complexas, incluindo *General Packet Radio Services* (GPRS), terceira e quarta geração de telecomunicações móveis (3G e 4G), sistema mundial de posicionamento (GPS) e tecnologia Bluetooth.

3 ESTADO DA ARTE

De forma geral os trabalhos pesquisados abordam características e especificidades no tratamento de um paciente, como descrições e indicações de medicamentos.

Um desses aplicativos é o AdaFarma (SANTOS, 2012), sendo mesmo voltado para uma drogaria especifica, indica para o paciente os horários para a ingestão dos medicamentos, um leitor de código de barra para informações do remédio e também um representante da drogaria que pode ser encontrado, caso esteja chegando ao fim o remédio.

Outras aplicações mHealth verificadas são as que atuam em diferentes áreas da saúde, encaminhando dicas em forma de Short Message Service (SMS) para auxiliar na prevenção de doenças, como o Nike Training Club que auxilia em treinos tipo caminhadas e dispara mensagens de auxílio (NIKE, 2016) e o aplicativo Código de Ética Médica que disponibiliza todo o documento do Conselho Federal de Medicina contendo regras e princípios que o médico deve seguir no exercício da profissão (GORDILHO, 2016).

O aplicativo MyTreat tem em seu diferencial a liberdade do usuário de cadastrar e controlar seus medicamentos como desejar, não solicitando um cadastro completo permitindo o usuário identificar seu remédio como desejar, além de gerar notificações que não invadem a privacidade do usuário.

4 MyTreat

O propósito deste trabalho é apresentar uma ferramenta para o auxílio na ingestão de medicamentos, utilizando-se de recursos existentes para a criação de aplicações nativas, sendo que, o seu foco é voltado à plataforma Android. A mesma tem como objetivo efetuar notificações a seus usuários referente a seus remédios cadastrados e previamente agendados para serem ingeridos em determinados intervalos, a aplicação faz o controle dos intervalos de tempo e comunica o usuário através de notificação o momento em que o remédio deve ser ingerido.

Durante o trabalho foi desenvolvido um aplicativo nomeado de MyTreat (Meu tratamento), com o objetivo de unir recursos existentes na plataforma para o melhor gerenciamento e controle nos tratamentos médicos. Pode ser utilizada em vários dispositivos da plataforma Android, que possuem algumas diferenças, sendo elas tamanho de tela ou desempenho de hardware, determinou-se a versão 4.1 do SO como requisito mínimo para que a aplicação possa ser instalada, pois a mesma faz o uso de recursos de layout que em algumas versões não são disponíveis. Desta forma, os estudos desenvolvidos bem como, a criação do aplicativo, evidenciam a implementação de uma ferramenta de auxílio para o melhor gerenciamento dos medicamentos utilizados por pacientes.

4.1 Desenvolvimento

O aplicativo é composto por um banco de dados SQLite disponível nativamente nos dispositivos móveis, sendo que o mesmo armazena todas as informações geradas pelo usuário da aplicação, armazenando também no seu próprio dispositivo. A estrutura do banco de dados é formada por um arquivo onde são feitas as leituras e escritas dos dados com linguagem de consulta SQL, disponibilizado assim uma fonte de informações para a aplicação utilizar quando necessário.

A Figura 1 ilustra o diagrama de entidades e relacionamentos com as principais tabelas contidas na base, sendo, agendas, remédio e pessoa. A uma relação entre a tabela agendas e remédio, pois deve conter integridade na agenda com vinculação dos remédios que fazem parte da mesma. Já a tabela pessoa apenas guarda as informações pessoais do usuário e não necessita uma relação com as demais tabelas, pois o aplicativo MyTreat é de uso pessoal e não permite mais de um cadastro de usuário para um único dispositivo.

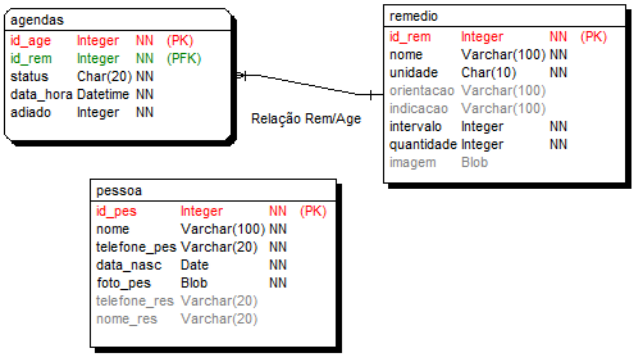

Fig. 1. Diagrama entidade relacionamento do Banco de Dados.

A interação do banco com as demais funcionalidades do aplicativo é feita através de um *Content Provider*, sendo ele um provedor de informações para que a aplicação faça uso das mesmas. A vantagem de utilizar um provedor de conteúdo é que o mesmo pode ser utilizado por várias aplicações.

O *Content Provider* possui uma estrutura pré-condicionada a sua criação, sendo o método query para recuperar dados, insert para inserir informações, *update* para atualizações, delete para apagar informações e getType utilizado para obter o MIME type de certa informação. Ambos os métodos possuem a URI (*Uniform Resourse Identifier*) em seus parâmetros internos além de vetores de valores, a URI é responsável por todas as operações de comunicação do *Provider* e da aplicação, pois na mesma tem a característica de um endereço onde todos os métodos fazem uso deste como parâmetro (ANDROID, 2016).

A sua estrutura básica é content://<authority>/<parametro1>/ <parametro2>/…/<parametroN>, sendo que, *authority* é o "nome" do provedor de conteúdo, e os parâmetros são aqueles definidos pelo provedor.

A URI do provedor de SMS content://sms/conversations/10 apresenta o nome do provedor SMS como primeiro parâmetro que está solicitando uma conversa e o último parâmetro que é o id da conversa. Esse método além de evitar o acesso direto ao banco de dados traz assim mais segurança para as informações que serão transmitidas (LECHETA, 2013).

Para que o Content Provider do aplicativo seja utilizado efetuou-se o registro do mesmo no AndroidManifest como ilustra na Figura 2.

```
1  <provider
2      android:name=".c_banco"
3      android:authorities="com.medical.predator.mytreat.c_banco" />
```

Fig. 2. Registro do Content Provider

Na linha 2, define-se o nome da classe do provedor e na linha 3 a URI de autorização para a utilização do mesmo. Também foi programada a estrutura da classe com seus métodos, a Figura 3 ilustra um trecho de código da classe implementada.

```
1  static {
2      mMatcher = new UriMatcher(UriMatcher.NO_MATCH);
3      mMatcher.addURI(AUTHORITY, AGENDA_TABLE, TB_AGENDA);
4      mMatcher.addURI(AUTHORITY, REMEDIO_TABLE, TB_REMEDIO);
5      mMatcher.addURI(AUTHORITY, PESSOA_TABLE, TB_PESSOA);
6      mMatcher.addURI(AUTHORITY, VW_TABLE , TB_VW);
7  }
8  //            Métodos overrided de ContentProvider              //
9  @Override
10 public int delete(Uri uri, String selection, String[] selectionArgs) {...}
11 @Override
12 public String getType(Uri uri) {...}
13 @Override
14 public Uri insert(Uri uri, ContentValues values) {...}
15 @Override
16 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {...}
17 @Override
18 public boolean onCreate() {
19     mHelper = new DBHelper(getContext());;
20     return true;
21 }
22 @Override
23 public Cursor query(Uri uri, String[] projection, String selection,
24                     String[] selectionArgs, String sortOrder) {
25     Integer i = mMatcher.match(uri);
26     SQLiteQueryBuilder builder = new  SQLiteQueryBuilder();
27     SQLiteDatabase database = mHelper.getReadableDatabase();
28     Cursor cursor;
29     switch (mMatcher.match(uri)) {
30         case TB_AGENDA:
31             builder.setTables(AGENDA_TABLE);
32             builder.setProjectionMap(mProjectionA);
33             break;
34     }
35     cursor = builder.query(database, projection, selection, selectionArgs, null, null, sortOrder,"20");
36     cursor.setNotificationUri(getContext().getContentResolver(), uri);
37     return cursor;
38 }
```

Fig. 3. Trecho de código da classe o Content Provider

Na **Erro! Fonte de referência não encontrada.** pode-se se observar que em todos os métodos utilizados pelo *Content Provider* se encontra a URI e seus parâmetros restantes. Também neste mesmo trecho de código, entre as linhas 1 e 7, são definidas as autorizações para algumas tabelas e as mesas são inseridas em uma lista para terem as URIs acessadas pela aplicação, é necessário conter um endereço para cada uma que deseja utilizar os métodos do provier e devolver informações para a aplicação.

Para a o gerenciamento do banco faz se uso da classe *SQLiteOpenHelper* para criação e manutenção das tabelas, para fazer o uso da mesma é necessário implementar os seguintes métodos: *onCreate()* e *onUpdate()*. O primeiro para ser utilizado na abertura da aplicação, criando assim toda a estrutura de banco de dados. Já o segundo é utilizado para a alteração de versão do banco, como alteração de tabelas e adição das mesmas. A **Erro! Fonte de referência não encontrada.** ilustra a implementação de um trecho de código da classe.

```
1  static {
2      mMatcher = new UriMatcher(UriMatcher.NO_MATCH);
3      mMatcher.addURI(AUTHORITY, AGENDA_TABLE, TB_AGENDA);
4      mMatcher.addURI(AUTHORITY, REMEDIO_TABLE, TB_REMEDIO);
5      mMatcher.addURI(AUTHORITY, PESSOA_TABLE, TB_PESSOA);
6      mMatcher.addURI(AUTHORITY, VW_TABLE , TB_VW);
7  }
8  //            Métodos overrided de ContentProvider              //
9  @Override
10 public int delete(Uri uri, String selection, String[] selectionArgs) {...}
11 @Override
12 public String getType(Uri uri) {...}
13 @Override
14 public Uri insert(Uri uri, ContentValues values) {...}
15 @Override
16 public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {...}
17 @Override
18 public boolean onCreate() {
19     mHelper = new DBHelper(getContext());;
20     return true;
21 }
22 @Override
23 public Cursor query(Uri uri, String[] projection, String selection,
24                     String[] selectionArgs, String sortOrder) {
25     Integer i = mMatcher.match(uri);
26     SQLiteQueryBuilder builder = new  SQLiteQueryBuilder();
27     SQLiteDatabase database = mHelper.getReadableDatabase();
28     Cursor cursor;
29     switch (mMatcher.match(uri)) {
30         case TB_AGENDA:
31             builder.setTables(AGENDA_TABLE);
32             builder.setProjectionMap(mProjectionA);
33             break;
34     }
35     cursor = builder.query(database, projection, selection, selectionArgs, null, null, sortOrder,"20");
36     cursor.setNotificationUri(getContext().getContentResolver(), uri);
37     return cursor;
38 }
```

Fig. 4. Trecho de código da classe DBHelper.

4.1.1 Interações com o usuário

A interação do usuário com a aplicação é realizada através das *Activities* da aplicação, sendo classes responsáveis por gerenciar as interfaces com o usuário construídas na linguagem Java. As interfaces são arquivos de layouts gerados em XML (*eXtensible Markup Language*) separados estruturalmente das classes. Nelas são encontradas todas as codificações de funcionalidades de uma determinada UI (User Interface) ou de várias em uma única classe (MONTEIRO, 2012). O aplicativo MyTreat conta com cinco classes principais de controle de Activitys que fazem a interação com o usuário. A Figura 5 ilustra o caso de uso das principais interações que o usuário tem na aplicação, já a Figura 6 ilustra as atividades e o fluxo das mesmas.

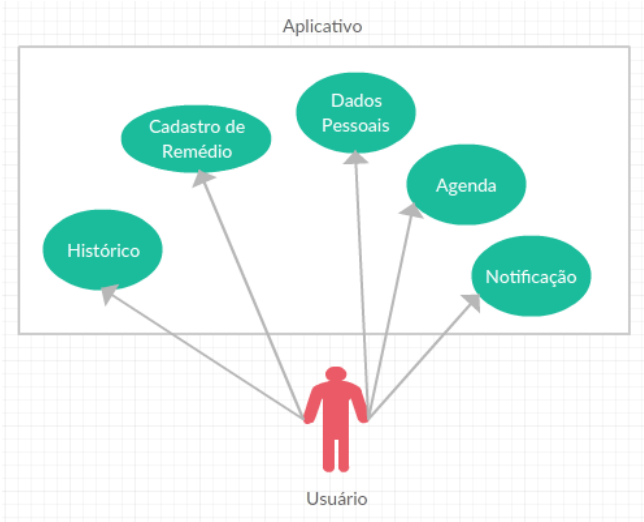

Fig. 5. Caso de Uso do Usuário.

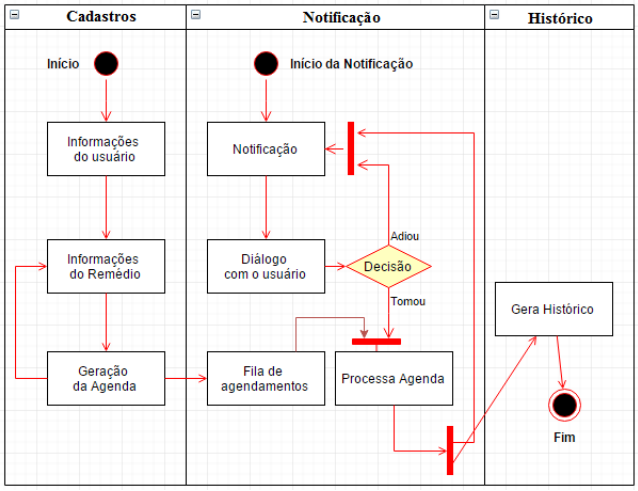

Fig. 6. Diagrama com o funcionamento do aplicativo *MyTreat.*

O usuário terá como atividades a inclusão de suas informações pessoais, o cadastro de seus remédios, visualização de notificações, a agenda e seu historio de remédios que já foram ingeridos. A *activity* agenda é a primeira interação com o usuário, sendo esta a principal, com o nome de *MainActivity.java*, porém se é o primeiro acesso à aplicação a mesma irá fazer uma chamada a *activity* dos dados pessoais (*DadosActivit.java*) solicitando o cadastro das informações para o usuário. Nela é possível inserir uma foto acessando o ícone da câmera posicionado

na parte superior da interface, que pode facilitar sua identificação. As interfaces dessas duas *activities* estão ilustradas na **Erro! Fonte de referência não encontrada.**.

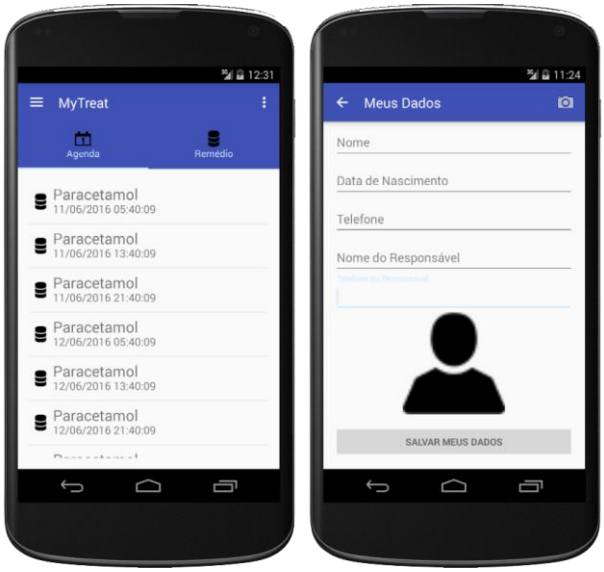

Fig. 7. Diagrama com o funcionamento do aplicativo *MyTreat.*

O usuário pode optar por não cadastrar seus dados apenas pressionando o ícone para retroceder à interface principal, podendo assim fazer esse cadastro quando achar conveniente. Nota-se na imagem que a interface principal conta com duas abas e duas opções de menu, na primeira aba dada como Agenda pode-se visualizar a lista de remédios com data e hora para que possam ser ingeridos a outra aba dada como Remédio lista todos os remédios cadastrados podendo fazer a edição do mesmo bastando acessar o de escolha, a primeira opção de menu à esquerda lista todas as opções que interface que o usuário tem com a aplicação, já a outra opção é um atalhos que se chegar aos dados pessoais, essas duas opções de menu são ilustrada na **Erro! Fonte de referência não encontrada.**.

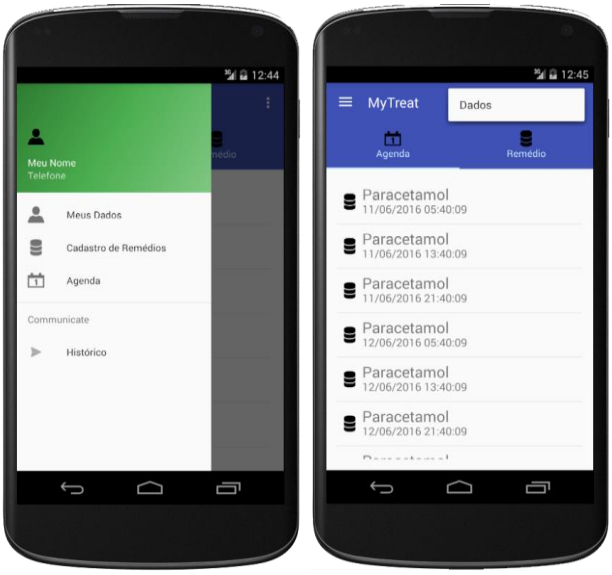

Fig. 8. Menus.

No menu principal tem as opções de interação como acessar os dados pessoais novamente, cadastro de remédios, visualização da agenda e visualização do histórico de remédios já ingeridos. Ao clicar em Meus dados a aplicação remete o usuário à interface dos seus dados pessoais que está ilustrado na f na mesma ao clicar no botão de cadastro a *activity* faz a utilização de alguns eventos e métodos para fazer a inserção da informação no banco de dados o trecho de código das principais funções utilizadas e dos métodos para utilização do *Content Provider* estão expostos na **Erro! Fonte de referência não encontrada.**.

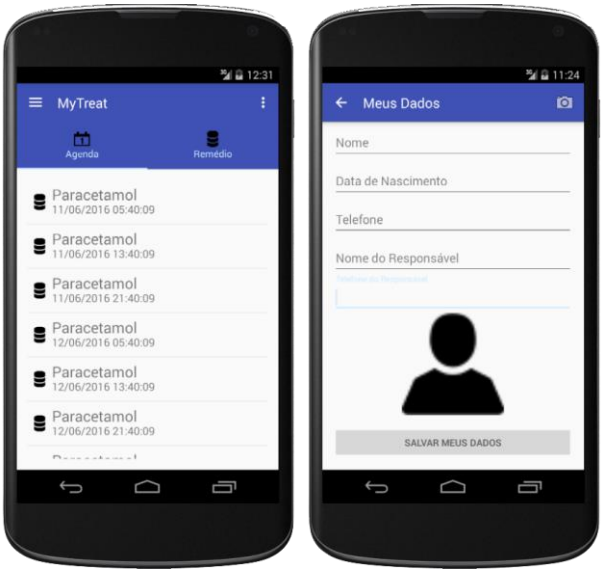

Fig. 9. Utilização dos métodos do *Content Provider*

Na linha 1 elenca-se um cursor para fazer a manipulação de um dos métodos do provider, o método query serve para selecionar ou executar comando em uma determinada tabela, no trecho de código apresentado na Figura 9, o método está buscando todas as informações que estão na tabela Pessoa, pois o único parâmetro adicionado é o endereço da tabela que se pretende selecionar, com o cursor alimentado é adicionado o valor da coluna _id_pes para a variável idPes para posteriores manipulações, caso cursor retorne sem valores/null, o mesmo é encerrado na linha 8.

Na linha 10 é feito o teste com o idPes, assim pode-se saber se o mesmo está sendo editado ou não, caso o id seja maior que zero, indica que os dados estão em estado de edição, desta forma é utilizado o método *update* do *Content Provider*, onde se atribui o endereço da tabela através da URI, os valores a serem inseridos, as condições e os valores paras as condições contidos em um vetor de *string*. O método *update* na linha 17 recebe todos esses parâmetros e executa a operação na base o mesmo ocorre para o método *insert* na linha 23, porem com uma peque diferença nele só é necessário informar a URI e os valores que serão inseridos.

4.1.2 Interação com os recursos do Android

Em todas as interfaces que possuem algum tipo de manipulação com dados que proveem do *Content Provider*, fazem a utilização desses três métodos mudando apenas os parâmetros.

Na interface de cadastro de remédios, ilustrada na **Erro! Fonte de referência não encontrada.**, além da utilização dos métodos tratados acima, são realizados interação com algumas classes disponíveis no Android, pois após a realização do cadastro do medicamento é gerado o

agendamento do mesmo, salvando as datas e intervalos no banco de dados e criando alarmes para notificar o usuário, as classes utilizadas são: *AlarmManager*, *Notification* e o *BroadcastReceiver*.

Fig. 10. Cadastro de remédios e listagem da agenda.

A classe *AlarmManager* fornece acesso aos serviços de alarme do sistema, permitindo a programação de execuções em tempos programáveis. Quando o alarme é disparado, se executa uma *Intent* registrada para o alarme, iniciando automaticamente a aplicação de destino, se já não estiver em execução.

O alarme pode ser ativado mesmo com o dispositivo em estado de bloqueio, enquanto o método do receptor de alarme *OnReceive ()* está em execução, garantido que o dispositivo não entre em estado de *sleep* enquanto o broadcast não terminar a operação. Para que se garanta a execução do alarme é necessário programar a classe *BroadcastReceiver* ou Service garantindo que o dispositivo continue funcionando até que o serviço executado seja encerrado. A **Erro! Fonte de referência não encontrada.** ilustra o trecho de código, onde a classe é instanciada a classe *AlarmManager*, nas linhas 1 e 2 é definido qual será a intenção da aplicação, nessa linha é definido que será iniciado uma ação de notificação, nas linhas 3 e 4 é definido o destino que será enviado essa execução gerando uma identificação para que o *BroadcastReceiver* possa interpretar, nas linhas 5 e 6 é instanciado o *AlarmManager* e nas linhas 7 e 8 é configurado que o alarme será do tipo de repetição e adicionado os parâmetros de execução, no primeiro parâmetro definisse que o alarme terá início mesmo com o dispositivo em estado de bloqueio ou *sleep*, no próximo a data de início da execução em milissegundos, posteriormente o intervalo entre execuções e por fim o identificador.

```
1  Intent intent = new Intent(getBaseContext(),
2      a_notificacao.class);
3  PendingIntent pendingIntent = PendingIntent.getBroadcast(
4      getBaseContext(), ID, intent, 0);
5  AlarmManager alarmManager = (AlarmManager)getSystemService(
6      Context.ALARM_SERVICE);
7  alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,
8      c.getTimeInMillis(), intervalo*60*1000,pendingIntent);
```

Fig. 11. Código de implementação do *AlarmManager*

O BroadcastReceiver garante que as execuções de tarefas agendadas sejam feitas mesmo com o aplicativo fechado. Desta forma a notificação é executa na interface do dispositivo no horário agendado para exibir a notificação é utilizado a_notificacao.class especificada nas linhas 1 e 2 da Figura 11, a classe é estendida a um BroadcastReceiver, que fica aguardando a sua chamada.

Na classe a_notificacao é utilizada a classe *Notification* para criar a estrutura de notificação que irá aparecer para o usuário. A Figura 12 ilustra a estrutura de criação de uma notificação, na mesma é definido um *builder (*Pacote) para que seja apresentada na notificação, que contém título, descrição, descrição de acessibilidade, o tempo de exibição, a ação executada, cancelamento automático e o ícone da notificação, após efetuar a construção é enviado para serviço de notificação do sistema e notificado ao usuário.

```
1  myNotification = new NotificationCompat.Builder(context)
2          .setContentTitle("Agenda de Remédios!")
3          .setContentText("Verifique sua agenda")
4          .setTicker("Notificação!")
5          .setWhen(System.currentTimeMillis())
6          .setContentIntent(pendingIntent)
7          .setDefaults(Notification.DEFAULT_SOUND)
8          .setAutoCancel(true)
9          .setSmallIcon(R.drawable.ic_calendar)
10         .build();
11 notificationManager =  (NotificationManager)context.
12         getSystemService(Context.NOTIFICATION_SERVICE);
13 notificationManager.notify(NOTIFICATION_ID, myNotification);
14
```

Fig. 12. Utilização da Classe *Notification*

A **Erro! Fonte de referência não encontrada.** ilustra a notificação apresentada ao usuário, ao acessar o usuário é redirecionado para a interface principal da aplicação onde pode verificar o remédio que deve ser ingerido, clicando no mesmo irá surgir uma caixa de diálogo solicitando o que o usuário deseja fazer, as opções são tomar o remédio ou voltar.

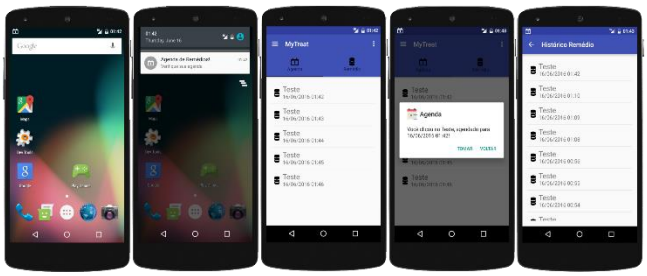

Fig. 12. Etapas da notificação

Ao clicar em tomar, o alarme com a identificação do agendamento do remédio é cancelamento, também é feito o desconto da unidade ingerida na quantidade total cadastrada do remédio. Após esses procedimentos o usuário é redirecionado a interface de histórico, que irá mostra todos os remédios já ingeridos pelo usuário e também avisará se o remédio ingerido está chegando ao seu fim.

5 VALIDAÇÃO

Os testes efetuados na aplicação foram realizados em um dispositivo móvel Xperia SP, com SO Android 4.3 (Jelly Bean), processado Dual-core 1.7 GHz Krait, uma GPU (Graphics Processing Unit) Adreno 320, 1 GB RAM e tela de 4.6 polegadas (GSMARENA, 2016). Também foram

efetuados testes em emuladores disponíveis no kit de desenvolvimento para o Android (Nexus 4, Nexus 5, Tablet Nexus 7, e Nexus S) e em todos os testes a aplicação se comportou bem, efetuando todas as suas operações e funções.

6 CONCLUSÕES

A utilização de vários dispositivos com formatos e tamanhos distintos em uma mesma plataforma exige-se aplicações planejadas para interagir com seus usuários em suas várias demandas, fazendo uso dos recursos de hardware diferenciados entre dispositivos.

Paralelo a tais demandas, o avanço tecnológico dos dispositivos móveis (smartphones e tablets) tem permitido a execução de recursos para o desenvolvimento de aplicativos cada vez mais complexos, que apresentam interfaces cada vez mais ricas e com funcionalidades que antes só eram possíveis de se visualizar em ambientes com grande poder de processamento no caso os desktops. Os recursos disponíveis para o desenvolvimento de aplicativos para a plataforma Android são diversificados, pois existem muitas bibliotecas que são compartilhadas da linguagem Java e utilizadas nas aplicações móveis, além de uma gama de bibliotecas próprias para o Android que fazem o meio de comunicação entre hardware e software, também vale reiterar que a documentação por parte do fabricante é bem detalhada, facilitando o entendimento dos recursos que são utilizados em uma aplicação. Porém vale enfatizar, que apesar da quantidade de recursos disponíveis para serem utilizados, o desenvolvedor fica amarrado a uma plataforma perdendo a portabilidade de uma aplicação.

Nesse sentido este trabalho apresentou algumas formas de utilização dos recursos dos dispositivos móveis que utilizam a plataforma Android para o desenvolvimento de aplicações cujo intuito é o auxílio à vida e a saúde das pessoas. Utilizando o seu Smartphones para o controle de tratamentos médicos, controlando os horários de ingestão de seus medicamentos. A aplicação contempla uma das principais etapas de um tratamento com medicamentos que é o agendamento dos horários para se ingerir os remédios, com descrição de horários, datas e Histórico, auxiliando assim o gerenciamento dos horários dos usuários. Os testes efetuados com o aplicativo MyTreat contemplaram as suas funcionalidades, notificando o usuário dos horários de ingestão de seus medicamentos.

REFERÊNCIAS

ANDROID. (2016), Marshmallow; Disponível em: <https://www.android.com/intl/pt-BR_br/versions/marshmallow-6-0/>; Acessado em: 26/05/2016.

GORDILHO, Raphael. (2016), O mundo móvel e o momento brasileiro para mHealth. Disponível em: <http://www.aplicativosdesaude.com.br/codigo-de-etica-medica-aplicativos-medicos/>; Acessado em: 31/05/2016.

GSMARENA. (2016), Smartphone Sony Ericsson Xperia SP. Disponível em: < http://www.gsmarena.com/sony_xperia_sp-5364.php>; Acessado em: 11/06/2016.

LECHETA, Ricardo Rodrigues. (2013), Google Android: Aprenda a criar a aplicações para dispositivos móveis com o Android SDK. 3ªEd. 824 pg. São Paulo, Editora Novatec.

MEDEIROS. (2012), Higor; Projetando e criando Aplicativos para Dispositivos Móveis; Disponível em: < http://www.devmedia.com.br/projetando-e-criando-aplicativos-para-dispositivos-moveis/30671>; Acessado em: 26/05/2016.

MONTEIRO, João B. (2012), Google Android: Crie aplicações para celulares e tablets. São Paulo, Editora Casa do Código.

NIKE, Inc. (2016), NIKE+ TRAINING CLUB. Disponível em: <https://play.google.com/store/apps/details?id=com.nike.ntc&hl=pt_BR>; Acessado em: 31/05/2016.

OMS. (2011), mHealth: New horizons for health through mobile technologies. Disponível em: <http://apps.who.int/iris/bitstream/10665/44607/1/9789241564250_eng.pdf>; Acessado em: 31/05/2016.

SANTOS, J.; WAGNER, P. K.; Navarro, B. R; BAKLISKY, M.; ARAUJO, L. V. (2012), Adafarma: Aplicativo para Auxílio na Fase de Aderência ao Tratamento. Anais do XIII Congresso Brasileiro de Informática em Saúde. Curitiba, 2012.

TIZATTO, Luiz. (2014), O mundo móvel e o momento brasileiro para mHealth. Disponível em: < http://saudebusiness.com/noticias/mundo-movel-momento-brasileiro-mhealth/>; Acessado em: 31/05/2016.

WORLDPANE. (2016), Kantar; Smartphone OS sales market share; Disponível em: <http://www.kantarworldpanel.com/global/smartphone-os-market-share>; Acessado em: 26/05/2016.