**MINISTÉRIO DA DEFESA**

**EXÉRCITO BRASILEIRO**

**DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA**

**INSTITUTO MILITAR DE ENGENHARIA**

**Seção de Engenharia de Computação / SE8**

RAFAEL **ANASTÁCIO** ALVES

**ALAN** FERREIRA BORBA

WILSON CÂMARA **MARRIEL**

**DESENVOLVIMENTO DE SISTEMA DE ORGANIZAÇÃO PESSOAL BASEADO EM GTD PARA DISPOSITIVOS MÓVEIS**

**Rio de Janeiro**

**2014**

**INSTITUTO MILITAR DE ENGENHARIA**

RAFAEL **ANASTÁCIO** ALVES

**ALAN** FERREIRA BORBA

WILSON CAMARA **MARRIEL**

**DESENVOLVIMENTO DE SISTEMA DE ORGANIZAÇÃO PESSOAL BASEADO EM GTD PARA DISPOSITIVOS MÓVEIS**

Projeto de fim de curso apresentado ao Curso de Graduação de Engenharia de Computação como requisito parcial para a obtenção do título de Engenheiro.

Orientador: Prof. Ricardo Choren Noya

Rio de Janeiro

2014

c2014

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 – Praia Vermelha

Rio de Janeiro - RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

XXXX Silva, J.J.; Souza, A.M.; Couves, X.Z.

##TÍTULO DO PFC## / José João Silva; Ana Maria Souza; Carlos Couves. – Rio de Janeiro: Instituto Militar de Engenharia, 2014.

##NÚMERO DE PÁGINAS## p.: il

Projeto de Fim de Curso (Engenharia de Computação) – Instituto Militar de Engenharia, 2014.

1. ##1ª PALAVRA-CHAVE##. 2. ##2ª PALAVRA-CHAVE E ASSIM POR DIANTE##. I. Título. II. Instituto Militar de Engenharia.

CDD 005.1

**INSTITUTO MILITAR DE ENGENHARIA**

RAFAEL **ANASTÁCIO** ALVES

**ALAN** FERREIRA BORBA

WILSON CÂMARA **MARRIEL**

**DESENVOLVIMENTO DE SISTEMA DE ORGANIZAÇÃO PESSOAL BASEADO EM GTD PARA DISPOSITIVOS MÓVEIS**

Projeto de Fim de Curso apresentado ao Instituto Militar de Engenharia, como requisito para colação de grau no Curso de Engenharia de Computação.

Orientador: Prof.Ricardo Choren Noya

Aprovada em \_\_\_\_ de \_\_\_\_\_\_\_\_\_ de 2014 pela seguinte Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.Ricardo Choren Noya, D.C., do IME – Presidente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Maj. Anderson Fernandes P. dos Santos,  D.Sc. do IME

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Raquel Coelho Gomes Pinto, D.Sc. do IME

Rio de Janeiro

2014

SUMÁRIO

LISTA DE ILUSTRAÇÕES 5

LISTA DE SIGLAS 6

1 INTRODUÇÃO 9

1.1 MOTIVAÇÃO 9

1.2 OBJETIVO 9

1.3 METODOLOGIA 10

1.4 ORGANIZAÇÃO DA MONOGRAFIA 10

2 GTD 12

2.1 DESCRIÇÃO 12

2.2 USO DO GTD NA APLICAÇÃO 16

3 COMPONENTES UTILIZADOS 17

3.1 ANDROID 17

3.1.1 CAMADA DE APRESENTAÇÃO 18

3.1.2 CAMADA DE NEGÓCIOS 22

3.1.3 CAMADA DE DADOS 26

3.2 GOOGLE CALENDAR 28

4 FUNCIONAMENTO 31

4.1 ACESSAR LISTAS 33

4.2 ADICIONAR ITEM 35

4.3 REMOVER ITEM 37

4.4 REMANEJAR ITEM 38

4.5 COMPARTILHAR ITEM 39

5 CONCLUSÃO 40

6 BILIOGRAFIA 42

7 APÊNDICE 44

7.1 DIAGRAMA DE TELAS 44

7.2 DESCRIÇÃO DE CASOS DE USO 44

7.2.1 Inserir Item rápido 44

7.2.2 Inserir Item 45

7.2.3 Inserir Compromisso 46

7.2.4 Sincronizar com o Google Calendar 46

7.2.5 Cadastrar usuário do Google Calendar 47

7.2.6 Reordenar Itens 47

7.2.7 Marcar item como processado 48

7.2.8 Excluir Item 48

7.2.9 Criar nova lista 49

7.2.10 Deletar lista 49

7.2.11 Abrir lista 50

7.3 ESQUEMA DE BANCO DE DADOS 50

LISTA DE ILUSTRAÇÕES

[FIG. 2.1 - Diagrama do fluxo de trabalho do método GDT [1] 13](#_Toc388915930)

[FIG. 3.1 - Arquitetura do Android [2] 18](#_Toc388915931)

[FIG. 3.2 – Relacionamento entre as Views 19](#_Toc388915932)

[FIG. 3.3 – Hierarquia de classes para ListView. [3] 20](#_Toc388915933)

[FIG. 3.4 – Assinatura do método getView [3] 21](#_Toc388915934)

[FIG. 3.5 – Hierarquia de classes do TextView [3]. 21](#_Toc388915935)

[FIG. 3.6 – Hierarquia de classes do TextView [3]. 21](#_Toc388915936)

[FIG. 3.7 – Hierarquia de classes do ContentProvider [3] 22](#_Toc388915937)

[FIG. 3.8 – Hierarquia de classes do Application 23](#_Toc388915938)

[FIG. 3.9 – Hierarquia de classes do Context 24](#_Toc388915939)

[FIG. 3.10 – Componentes para funcionamento do Android [6] 25](#_Toc388915940)

[FIG. 3.11 – Hierarquia de classes do SQLiteDatabase 27](#_Toc388915941)

[FIG. 3.12 – Hierarquia de classes do SQLiteOpenHelper 27](#_Toc388915942)

[FIG. 3.13 – Hierarquia de classes do SQLiteCursor 28](#_Toc388915943)

[FIG. 3.14 – Calendar Provider data model [7]. 30](#_Toc388915944)

[FIG. 4.1 – Modelo da arquitetura da aplicação 32](#_Toc388915945)

[FIG. 4.2 – Tela inicial 33](#_Toc388915946)

[FIG. 4.3 – Uma lista aberta 34](#_Toc388915947)

[FIG. 4.4 – Outras Listas 35](#_Toc388915948)

[FIG. 4.5 – Adicionando item 36](#_Toc388915949)

[FIG. 4.6 – Criando evento utilizando aplicativo externo 37](#_Toc388915950)

[FIG. 4.7 – Menu de ferramentas para o item 38](#_Toc388915951)

[FIG. 4.8 – Remanejando um item 39](#_Toc388915952)

[FIG. 4.9 – Compartilhar um item 40](#_Toc388915953)

[FIG. 7.1 – DIAGRAMA DE NAVEGAÇÃO ENTRE TELAS 44](#_Toc388915954)

[FIG. 7.2 – Esquema de banco de dados 50](#_Toc388915955)

LISTA DE SIGLAS

API Application programming interface

MVC Model-View-Controller

GTD Getting Things Done

OS Operating System

SQL Structured Query Language

XML Extensible Markup Language

VM *Virtual Machine*

WYSIWYG *What You See Is What You Get*

ADT Android *Developer* *Tool*

SDK Software *Development* *Kit*

IDE *Integrated* Development Enviroment

URI *Unified Resource Identifier*

**RESUMO**

Este projeto tem por objetivo desenvolver, seguindo a metodologia GTD (*Getting Things Done*), um aplicativo de organização pessoal para dispositivos móveis (Android). Para isso, será desenvolvida uma *checklist* própria e será utilizada a API do Google Agenda, como componentes principais do aplicativo. No final do projeto, uma integração entre os componentes será implementada.

**ABSTRACT**

This project aims to develop, following the GTD (Getting Things Done) methodology, an application of personal organization for mobile devices (Android). To achieve this goal, it will be developed a check-list and it will be used the Google Calendar’s API, as main components of the application. At the end of the project, an integration between the components will be implemented.

# INTRODUÇÃO

Nos últimos 10 anos, é cada vez mais requisitado das pessoas que elas sejam organizadas, seja na vida pessoal, como na vida profissional. Vários métodos de organização pessoal foram propostos ao longo de décadas, dentre elas o GTD (*Getting Things Done*).

Apesar de escolher um método de organização pessoal ser importante no processo de busca por aumento de produtividade e de busca por diminuição de estresse decorrente da própria tentativa de aumentar a eficiência nas tarefas do dia a dia, isso, infelizmente, não é o suficiente. Para, de fato, se ganhar produtividade, é preciso que se escolha a ferramenta certa. Com a difusão das tecnologias digitais, a opção por *smartphones* e computadores tem ganhado mais adeptos, se comparado com a opção por tecnologias tradicionais como agenda, cadernos e pastas físicas. A causa mais óbvia para isso seria o fato de essa ferramenta ocupar menos espaço. Além disso, permite que se realize *backup* facilmente na nuvem.

## ****MOTIVAÇÃO****

Com a crescente popularidade da plataforma Android para dispositivos móveis, desenvolver um aplicativo de organização para esse sistema operacional constitui um meio oportuno para os alunos de graduação em Engenharia de Computação do IME aplicarem seus conhecimentos adquiridos ao longo do curso e, ao mesmo tempo, possibilitar o empreendedorismo.

Um fator que motivou a escolha por desenvolver o aplicativo para dispositivos móveis, em detrimento do desenvolvimento de um serviço web, se deve ao fato de o fácil acesso a um *smartphone* ou *tablet*  facilitar a organização do usuário e ao fato de o aplicativo a ser desenvolvido permitir acesso *off-line*, não havendo necessidade de acesso à internet sempre que for utilizado.

Aliado a isso, o renome e a simplicidade do sistema de organização GTD fez com que se adotasse esse modelo para estabelecer as funcionalidades a serem incorporadas a esse aplicativo.

## OBJETIVO

O objetivo deste projeto é desenvolver um aplicativo de organização pessoal ,baseado nos preceitos do GTD (*Getting Things Done*), para *smartphones* que possuam Android.

## METODOLOGIA

Para alcançar o objetivo proposto, o projeto seguiu, iterativamente, as seguintes etapas:

* Estudo do método de organização GTD – no qual leu-se o livro e identificou-se os principais elementos que poderiam ser úteis para uma aplicação móvel (os detalhes podem ser obtidos no capítulo 2). Fez-se, assim, o levantamento de requisitos – na qual se identificou as características relevantes do sistema de organização do GTD.
* Análise de requisitos – na qual, dentre todas os requisitos funcionais e não funcionais requeridos do método GTD para uso num aplicativo móvel, definiu-se aqueles que poderiam ser incorporados ao software deste projeto e que poreriam satisfazer ao prazo de entrega, que é de aproximadamente um ano. Alocou-se, assim, tempo para estudo dos elementos da plataforma Android (que será abordado no capítulo 4), para o projeto do sistema, para codificação e para testes .
* Projeto do sistema – na qual se definiu os componentes e bibliotecas necessárias para o funcionamento do sistema.
* Codificação – na qual se implementou o que ficou especificado na análise de requisitos.
* Teste do sistema – na qual se realizou testes à medida que cada funcionalidade era implementada. Testes foram realizados ao fim da codificação de todas as funcionalidades.

## ORGANIZAÇÃO DA MONOGRAFIA

Este presente trabalho se encontra dividido em 3 capítulos:

Capitulo 2: que fará uma breve descrição do método GTD e como suas características influenciaram nas funcionalidades do sistema desenvilvido;

Capítulo 3: que fará uma descrição dos elementos utilizados pelo aplicativo e como eles serão utilizados no sistema.

Capítulo 4: que abordará, em detalhes visuais, o comportamento do sistema durante a utilização de suas funcionalidades.

# GTD

## DESCRIÇÃO

O GTD ( *Getting Things Done* ) é um método de gerenciamento pessoal que consiste em remover todas as preocupações do indivíduo de sua mente, registrá-los em algum lugar e agrupá-los – seja por meio de pastas, listas físicas ou digitais, dentre outros– por contexto.

Os principais contextos no GTD são “caixa de entrada”, “algum dia / talvez”, “referência”, “projetos”, “esperar”, “calendário”, “próximas ações” e “lixo” [1], que serão explicados em seguida. O autor do método afirma que um sistema de organização completo precisa levar em consideração todos esses contextos. Caso contrário, o indivíduo corre o risco de deixar algum preocupação em sua vida sem ser devidamente administrada.

Boa parte das preocupações do indivíduo podem ser registradas por meio da escrita em meios de anotação físico ou digital. Conseqïentemente, com exceção de “projetos”, “referência” e “lixo”, que requerem meios mais robustos de armazenamento, os demais grupos contextuais são melhor visualizados quando os itens neles contidos são listados. Dessa forma, o uso de listas é muito comum para visualização de seu conteúdo.

Cada item dessas listas equivale a alguma informação pode requerer ações do indivíduo ou armazenamento para uso futuro potencial em projetos em adamento. Após um item ser devidamente administrado, é recomendado marcá-lo como processado para que visualmente seja fácil distinguir os itens que foram administrados daqueles que ainda não.

De maneira geral, as preocupações do indivíduo, após serem inicialmente registradas, devem seguir o fluxo pelos contextos conforme ilustrado na FIG. 2.1 para uma completa organização.

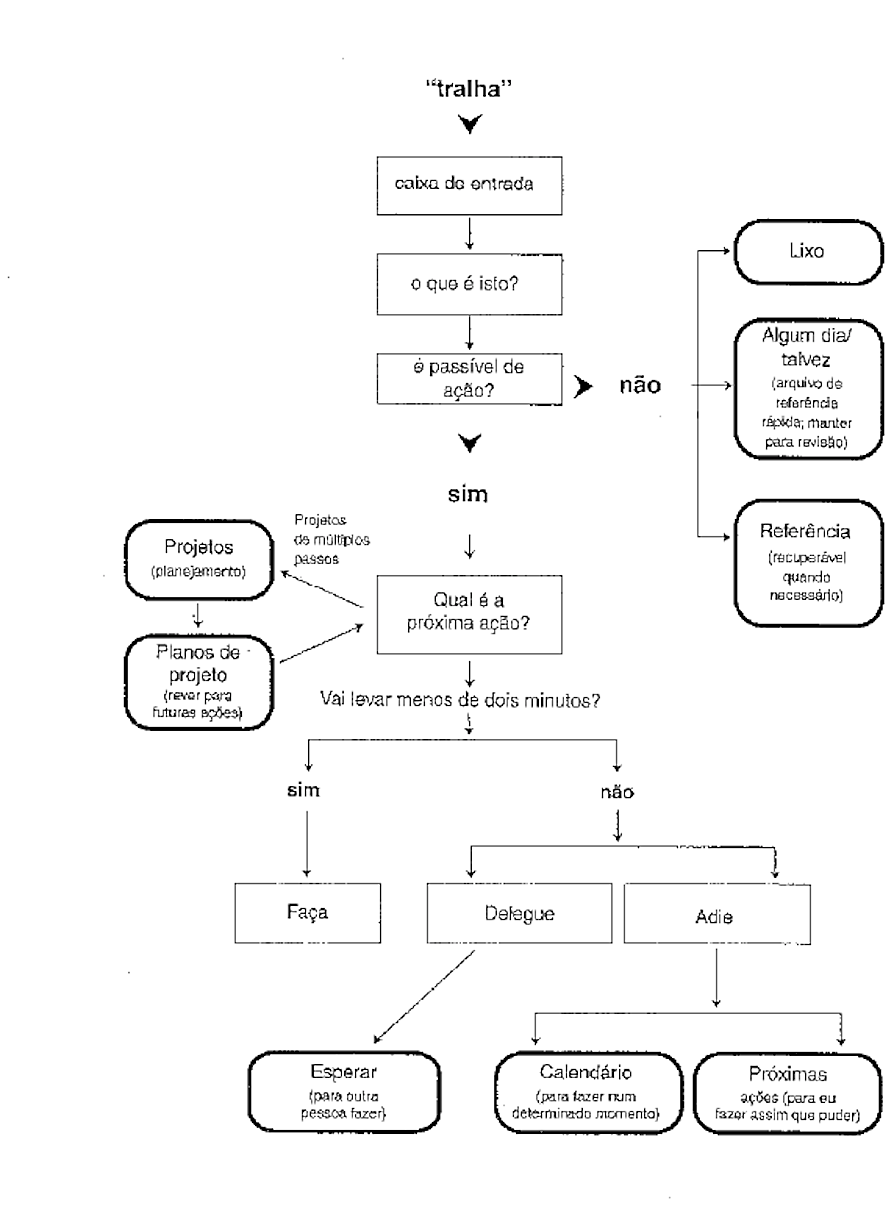


FIG. . - Diagrama do fluxo de trabalho do método GDT [1]

A seguir uma breve explanação sobre os contextos do GTD e suas funções .

CAIXA DE ENTRADA

Para administrar adequadamente o estoque de itens que exigem atenção, o indivíduo precisa armazená-los em “*containers”* que sirvam para guardar os itens pendentes até que ele tenha alguns minutos para decidir o que esses itens são e o que fazer com eles [1]. No método GTD, esses “*containers*” são chamados de caixa de entrada. Elas podem ser de diversos tipos:

* Caixa de Entrada física;
* Instrumentos de anotação de papel;
* Instrumentos de anotação eletrônicos;
* Instrumentos de gravação de voz;
* E-mail;

PRÓXIMAS AÇÕES

O contexto de próximas ações refere-se a ações que deverão ser executadas assim que possível. Elas não devem possuir data certa para acontecer, mas, sim, momentos certos.

São exemplos de ações desse contexto telefonemas a serem dados, compras a fazer, dentre outros. Dessa forma os telefonemas podem ser realizados quando houver um telefone disponível por perto. As compras podem ser feitas quando se estiver num shopping.

ESPERAR

Esse contexto contém informações acerca de ações que foram delegadas a outras pessoas, e que seria importante manter um registro delas para acomapanhamento do andamento.

CALENDÁRIO

Esse contexto se referem ações que possuem data e hora certa para acontecerem. Normalmente as informações desse contexto são armazenadas em agendas físicas ou digitais,

ALGUM DIA/TALVEZ

Esse contexto contém itens referentes a ações relacionadas a projetos futuros; seja porque o indivíduo possua muitos projetos no presente momento, seja porque essas ações dependem de recursos externos que não estão disponíveis no momento.

Por exemplo, se o indivíduo decide fazer uma viagem que só terá tempo de fazê-la nos próximos 2 meses. Ele poderá adicionar um item na lista “Algum dia/talvez” para que ele se lembre, durante suas revisoões semanais, de que ele pretendia fazer uma viagem . Dessa forma, ele lembrará periodicamente desse evento e poderá tomar decisões quando a data da viagem se aproximar.

INCUBAÇÃO

Um contexto relacionado a preocupações do indivíduo que requerem mais tempo para serem analisadas para que se possa tomar a ação correta a respeito.

REFERÊNCIA

Esse contexto refere-se ao sistema de armazenamento de informação do indivíduo. O GTD recomenda que haja um sistema físico ou digital eficiente para armazenamento de informações que o indivíduo julgar pertinentes para si. Essas informações podem ser acessadas sempre que necessário.

Um exemplo, para fins de ilustração, seria uma pessoa guardar uma receita de bolo de uma revista na pasta pessoal (física ou digital) de receitas dela, porque ela achou que precisaria utilizá-la um dia.

PROJETO

Segundo GTD, projeto corresponde a uma ação de múltiplos passos. Simples ações do dia a dia como planejar uma festa, corresponderia a um projeto.

O GTD recomenda que o indivíduo tenha representado, físico ou digitalmente, todos os seus projetos ativos, de forma que cada ação do indivíduo possa estar associado a um projeto.

LIXO

Esse contexto representa o ato de descartar uma determinada preocupação, por não se enquadrar em nenhum outro contexto e, consequentemente, não ter utilidade alguma.

## USO DO GTD NA APLICAÇÃO

O aplicativo deste projeto, para atender os preceitos do GTD implementará, em forma de lista textual, os contextos “caixa de entrada”, “próximas ações”, “algum dia / talvez”, “calendário”, “esperar” e “incubação”. Os contextos de “projeto” e “referência” serão contemplados com listas, porém, o aplicativo fornecerá a funcionalidade de “compartilhar”, que proverá os meios para que o usuário possa remanejar os respectivos itens para seu sistema pessoal de “projeto” e ‘referência”, externo ao aplicativo. O contexto “lixo”, no aplicativo, corresponderá a opção de excluir ítens.

A fim de facilitar o fluxo de trabalho entre contextos prescrito pelo GTD (FIG. 2.1), o aplicativo oferecerá funcionalidade que permite que cada item possa ser remanejado, de uma lista para outra.

O aplicativo proverá o mesmo tratamento visual para as listas. O calendário, em especial, utilizará os serviços do Google Calendar para fornecer o conteúdo dessa lista.

A seguir, uma breve descrição das *checklists* fixas do GTD e como elas serão abordadas durante o decorrer do projeto.

Os requisitos estão formalmente documentados nas descrições de caso de uso, localizados no apêncide 7.2 . O resultado, bem como a sua descrição, são detalhados no capítulo 4.

# COMPONENTES UTILIZADOS

## ANDROID

DESCRIÇÃO

Android é uma pilha de software para dispositivos móveis que inclui um sistema operacional, middleware e aplicações chave.

Seu sistema operacional é baseado no kernel do Linux 2.6.

O Android SDK fornece as ferramentas e APIs necessárias para começar a desenvolver aplicações na plataforma Android usando a linguagem de programação Java.

A arquitetura do Android pode ser subdividia em 5, conforme FIG. 3.1:

* Aplicações: Camada formada pelos aplicativos utilizados pelo usuário. O aplicativ. Esses aplicativos utilizam o framework de aplicação poderoso e flexível para desenvolvimento. São escritos na linguagem Java.
* *Framework* da aplicação: Camada com a qual aplicativos interagem. Composto por serviços como Activity Manager, Content Provider, Telephony Manager, Location Manager, Resource Manager, dentre outros. Esses serviços são disponibilizados na forma de classes para que possam ser utilizados pelos programadores no desenvolvimento de seus aplicativos.
* Bibliotecas: Camada que permite que o dispositivo manipule diferentes tipos de informação. Elas são escritas em linguagem c ou c++ e são específicas para cada *hardware*.
* Android Runtime: Camada composta pela máquina virtual Dalvik – um tipo de máquina virtual Java desenvolvido e otimizado para Android – e pelas *core Java libraries*. Assim, a máquina virtual Dalvik pode utilizar recursos como gerenciamento de memória e de multi-*threading*, intrínseco à linguagem Java, bem como permite que cada aplicação funcionando tenha seu próprio processo, com sua própria instância da Dalvik *VM*.
* *Kernel* do Linux: Camada responsável por funcionalidades básicas de sistema operacional como gerenciamento de processos, de memória, de energia e de *drivers* de dispositivos (câmera, teclado, *display*).

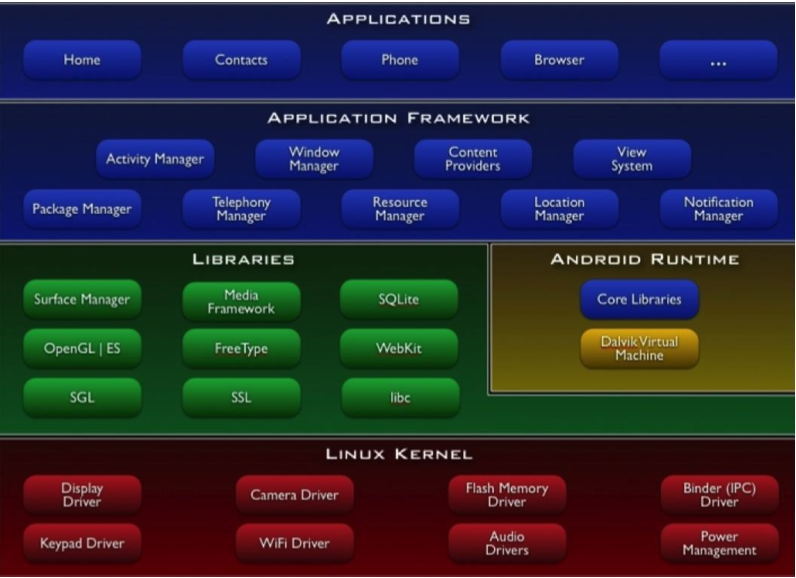


FIG. . - Arquitetura do Android [2]

A seguir, será detalhado o uso dos principais elementos do Android necessários para o desenvolvimento do aplicativo deste projeto, separado por camada.

### CAMADA DE APRESENTAÇÃO

**View System**

Representada pela classe View, representa o bloco básico para construir os componentes de interface de usuário. A classe View é a base para os *Widgets*, utilizados para a criação de componentes de interface do usuário interativos. [3]

Todos os objetos da classe View seguem o padrão WYSIWYG, e se estruturam em árvores. Uma GroupView, correspondente aos nós dessa árvore, são a base para os *layouts* (*containers* invisíveis que possuem referência às Views filhas), conforme esquematizado na FIG. 3.2:



FIG. . – Relacionamento entre as Views

As Views e suas estruturas podem ser declaradas tanto por meio de arquivo XML (normalmente localizadas no arquivo de nome “layout.xml” ) quanto por meio de código. Uma vez configuarada a estrutura de Views da aplicação, algumas operações são comumente feitas nas Views, como:

* Configurar propriedades: Estabelece configurações especefícas de algumas Views. Para uma TextView, por exemplo, é possível configurar o texto a ser exibido;
* Estabelecer foco: Uma View, por meio da chamada da função “requestFocus()“, é possível instruir o sistema para que ela receba os dados de entrada do usuário;
* Configurar *listeners*: Cada View pode ser configurada para responder aeventos do sistema como gestos de clique, pressionar por longo período, arrastar, dentre outros;
* Configurar visibilidade: Cada View pode estar visível ou não, conforme necessidade da aplicação cliente, por meio da chamada do método “setVisibility(int)”;

Dentre as subclasses mais utilizadas neste trabalho, destacam-se:

**ListView**

Corresponde a uma GroupView responsável exibir uma lista que desliza verticalmente. Os itens que constituirão a lista podem ser inseridos dinamicamente por meio de objeto da classe Adapter. Esse objeto, é responsável por adaptar uma determinada estrutura de dados, como um *array* ou uma *query* de banco de dados, de forma a associar cada elemento dela a uma View que constituirá a lista. As sublcasses de Adapter mais utilizadas são ArrayAdapter (quando os dados são provenientes de um *array*) e CursorAdapter (quando os dados provenientes de um *cursor*, normalmente relacionado a uma *query* a banco de dados).

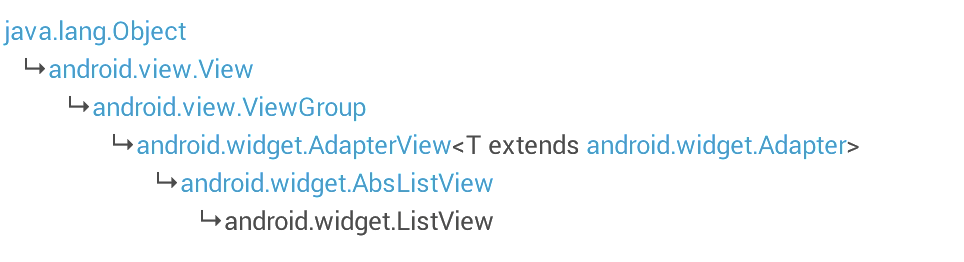
****

FIG. . – Hierarquia de classes para ListView. [3]

**ArrayAdatpter**

O ArrayAdapter, por padrão, espera que a ListView seja constituída de Views do tipo TextView, de forma que ela irá utilizar o método “toString()” para converter cada elemento do Array em texto.

Porém, para elaborar ListViews mais complexos, que possam conter outros elementos além de TextView, é possível sobrescrever o método “getView(int, View, ViewGroup)” da classe ArrayAdapter.

Conforme a assinatura detalhada na figura FIG. 3.4, utilizando *position* para referenciar cada elemento do Array pelo qual o ArrayAdapter é responsável, é possível instanciar outras Views, como CheckBox e Images. Essas Views são instanciadas através do uso dos objetos *convertView e parent* em colaborações de classes que não serão detalhados neste capítulo.

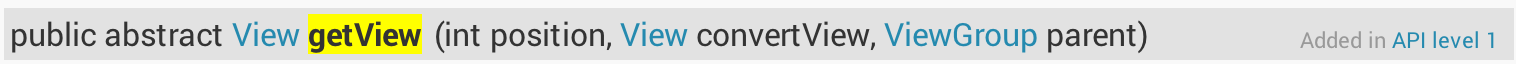


FIG. . – Assinatura do método getView [3]

**TextView**

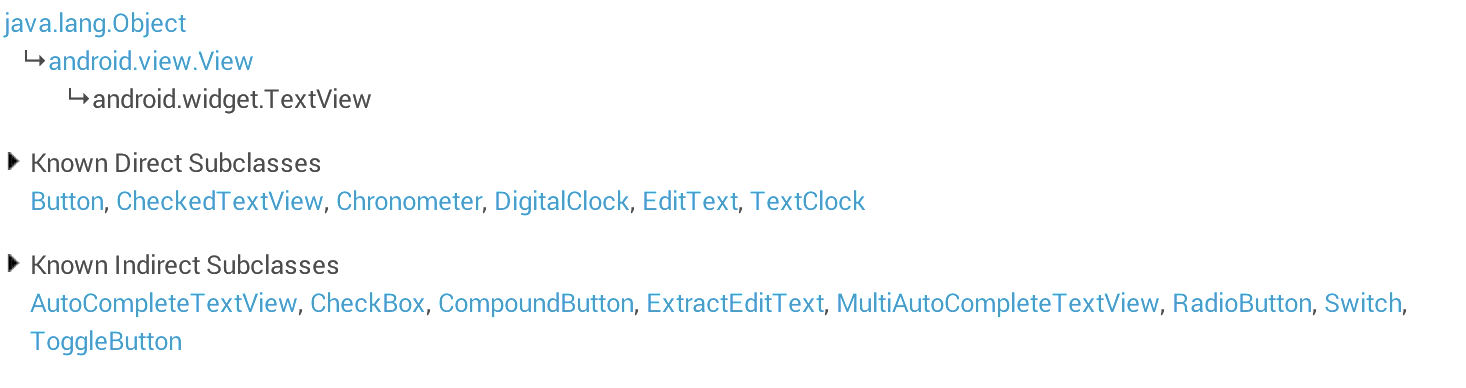
****

FIG. . – Hierarquia de classes do TextView [3].

Corresponde à subclasse de View responsável por exibir e, quando necessário, editar texto.

Para o configurar o texto de uma TextView, usa-se o método “setText(String)”. É possível também configurar sua aparência por arquivo XML ou por meio de código, utilizando-se métodos como “setTextAppearance(Context context, int resid)” .

**CheckBox**

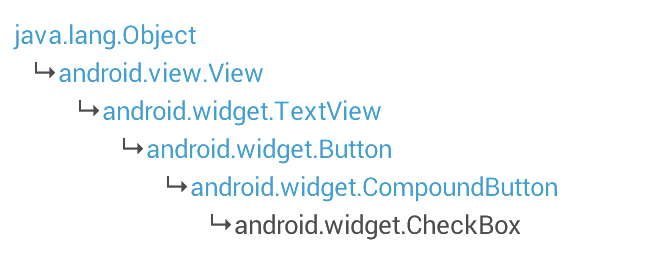
****

FIG. . – Hierarquia de classes do TextView [3].

Esta classe é a uma View correspondente a um botão de dois estados (*checked* e *unchecked*)*,* Os métodos mais utilizados são o “isChecked()” que retorna valor booleano informando se o CheckBox está no estado *checked* (retornando *true*) ou *unchecked* (retornando *false*).

**JUSTIFICATIVA**

Neste projeto, utilizaram-se subclasses de *Views* - ListView, TextView e CheckBox - para compor a interface gráfica equivalente a lista de tarefas com itens. Os itens recuperados da camada de dados referentes a uma lista serão armazenados em um *arraylist*. Uma instância de ArrayAdapter utilizará o *arraylist* para customizar a exibição final de cada item dessa lista de tarefas.

Foi utilizada uma subclasse de ListView de forma a gerar uma interface personalizada de lista, conhecida como *Swipelistview* [4]. Ela utiliza o padrão de ferramentas contextuais não modal [5], no qual se permite a exibição de ferramentas relativas a um item, o que será útil quando se desejar realizar ações sobre um item de uma lista.

Para a exibição das listas criadas pelo usuário, utilizou-se um objeto comum da classe ListView.

A aparência e uso final das listas e outros artefatos visuais será melhor esclarecido durante a descrição das funcionalidades do aplicativo, no capítulo 4.

### CAMADA DE NEGÓCIOS

**Content Provider**

****

FIG. . – Hierarquia de classes do ContentProvider [3]

Os *Content Provider*s são parte da arquitetura de um sistema Android e são responsáveis por prover às aplicações os dados necessários que elas precisam para funcionar. Eles gerenciam o acesso a um repositório central de dados e permite que aplicações diferentes compartilhem os mesmo dados. Com isso as aplicações conseguem facilmente, acessar os dados de um contato, da agenda ou ler mensagens recebidas.

Quando se quer acessar um dado em um *content* *provider*, a aplicação cliente deve utilizar um objeto *Content Resolver* juntamente com o objeto *Context* para se comunicar com o *provider.* O objeto do tipo *ContentResolver* se comunicará com a instância que implementa*ContentProvider.*

**Application**

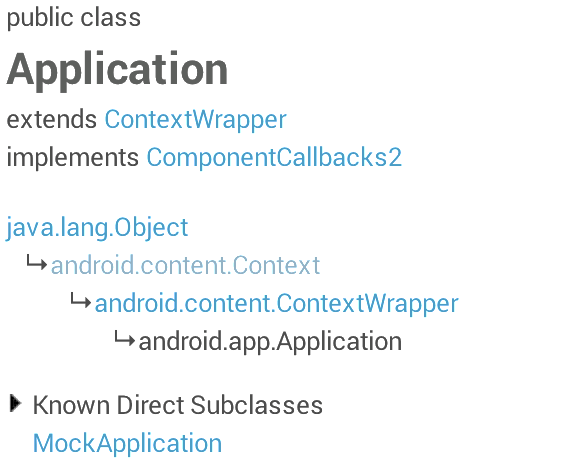


FIG. . – Hierarquia de classes do Application

A classe Application para os desenvolvedores que precisam manter o estado da aplicação.

Quando se decide por implementar a classe, é preciso declarar o nome da classe extendida no elemento XML <application> de AndroidManifest.xml. Dessa forma, uma instância dessa classe será criada pelo Android ao se iniciar o processo da aplicação. Para isso, a instância de Application pode ser recuperada por meio da chamada da função Context.getApplicationContext().

**Context**

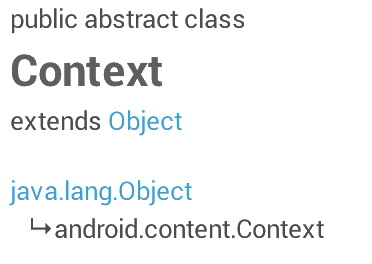


FIG. . – Hierarquia de classes do Context

O contexto é uma classe abstrata que serve de interface de acesso a informações globais, bem como a recursos e classes, pertencentes à aplicação. Parte da implementação é feita pelo Android.

Dentro do projeto da IDE, foram utilizadas as seguintes diretórios:

**/AndroidManifest.xml**

Esse arquivo contém configurações da aplicação que o Android obtém para oferecer as funcionalidades e permissões de que o aplicativo precisa para funcionar. A FIG. 3.10, mostra como o arquivo AndroidManifest.xml colabora com os demais componentes do Android.

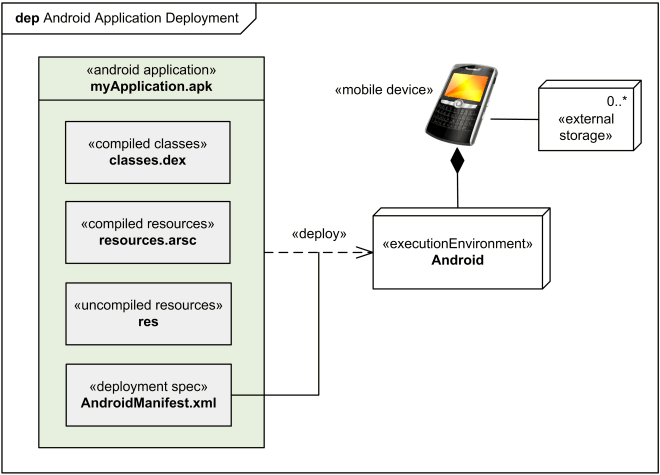


FIG. . – Componentes para funcionamento do Android [6]

Quando se utiliza o *framework* de desenvolvimento incluso no ADT-Bundle, alguns elementos XML do *AndroidManifest.xml* são previamente configurados. Os elementos e atributos XML que mais tiveram seus valores setados manualmente neste trabalho foram:

<activity> - utilizada para declarar uma *activity* que será utilizada na aplicação, bem como suas configurações. Declara-se também os recursos e permissões do Android necessários para o funcionamento do aplicativo.

android:parentActivityName – atributo do elemento <activity> que serve para citar o nome da *activity* pai da que está sendo configurada.

**Pasta de *resources:*** Conforme recomendado por Android, qualquer recurso da aplicação deve ser externalizado em arquivos específicos de nomes convencionados. Por meio dessa prática, é possível prover recursos alternativos para cada dispositivo com diferentes configurações e tornar o próprio sistema operacional responsável por decidir quando utilizá-los Dentre os prícipais diretórios de recursos utilizados neste projeto, encontram-se:

**/res/values** – Pasta que contém arquivos XML relacionados a valores simples como *strings*, inteiros, e cores, Os arquivos mais comuns são:

colors.xml – Utlilizado para conter todas as cores personalizadas pelo desenvolvedor.

dimension.xml – Utilizado para armazenar dimensões personalizadas pelo desenvolvedor.

string.xml – Utilizado para armazenar strings que serão comumente utilizadas pelo usuário.

style.xml – Utilizado para armazenar configurações de aparência e formato para a interface de usuário.

**/res/layout**  - Contém arquivos XML responsáveis por armazenar configurações de *layout* da interface de usuário.

**JUSTIFICATIVA**

Durante a utilização do aplicativo, o usuário irá transitar entre várias telas, tais como a de menu principal, a da lista aberta pelo usuário, dentre outras. O uso de hierarquias ente elas, no Android, facilita a navegabilidade para o usuário, por meio dos butões *up* e *back*. A fim de prover isso, utilizou-se, no arquivo “AndroidManifest.xml“, dentro de alguns elementos <activity>, o atributo “android:parentActivityName” para que a atividade com esse atributo possa levar o usuário à *acitivity* pai em questão e, assim, facilitar a navegação entre as listas.

Uma instância da subclasse da classe *Application* servirá como controlador do aplicativo, intermediando parte da navegação entre as telas e coordenando a comunicação entre a camada de dados e a camada de apresentação. Cada subclasse de *Activity* se comunicará com o controlador informando o contexto da aplicação, encapsulado em um objeto *Context.*

### CAMADA DE DADOS

Dentre as bibliotecas disponibilizadas pelo Android, o projeto fará uso de:

**SQLite**

No Android, é possível utilizar o pacote “android.database.sqlite”. Nesse pacote se encontram as classes que uma aplicação precisa para administrar sua *database* privada.

As classes mais utilizadas do pacote são:

**SQLiteDatabase**



FIG. . – Hierarquia de classes do SQLiteDatabase

O SQLliteDatabase é responsável por expor os métodos que gerenciam um banco de dados do SQLite. Dentre os métodos, se encontram aqueles de criar, excluir, executar comandos SQL e realizar outras tarefas comuns de banco de dados.

**SQLiteOpenHelper**

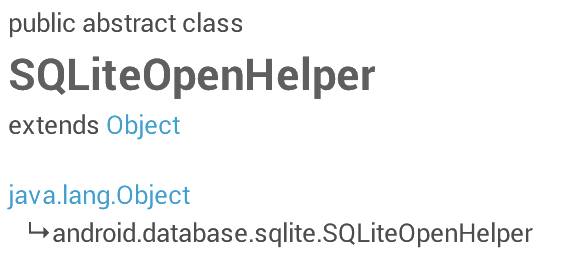


FIG. . – Hierarquia de classes do SQLiteOpenHelper

O SQLiteOpenHelper é responsável pela criação, bem como pelo versionamento, de banco de dados. Ela também é responsável por abrir um banco de dados, criar um se não existe, e atualizar sua versão se necessário

Para que essas funcionalidades existam, o programador deve implementar os métodos onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) e, opcionalmente, onOpen(SQLiteDatabase).

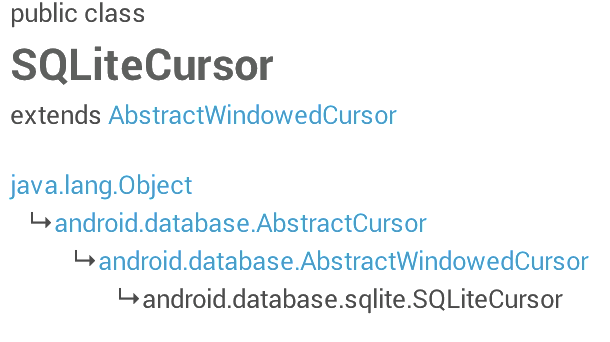


FIG. . – Hierarquia de classes do SQLiteCursor

O SQLiteCursor é responsável por expor os resultados de uma *query* sobre um *SQLiteDatabase*.

JUSTIFICATIVA

Utilizaram-se as classes Cursor, SQLiteDatabase e SQLiteOpenHelper para que colaborem entre si para gerenciamento interno de banco de dados. Dessa forma, a aplicação poderá persistir as informações contidas nos ítens da lista do usuário.

## GOOGLE CALENDAR

A seguir, uma breve descrição dos componentes do Android necessários para o funcionamento dos casos de uso envolvendo o Google Calendar.

CONTENT PROVIDER

*Content Providers* geram acesso a conjuntos de dados, encapsulando-os e fornecendo mecanismos para garantir sua segurança [3].

Para requisitar algum dado, uma aplicação cliente deve usar um objeto da classe *ContentResolver* como interface para a requisição a determinada instância de *Content Provider.*

CALENDAR PROVIDER

O *Calendar Provider* permite às aplicações terem acesso aos eventos do calendário como realizar uma inserção, uma remoção, uma atualização ou simplesmente uma busca sobre os eventos do dia.

Uma aplicação pode realizar uma escrita ou leitura em um calendário desde que tenha as permissões de aplicação necessárias. Essas operações de escrita e leitura são feitas utilizando um conjunto de rotinas fornecidas pelo *Calendar Provider*.

Qualquer dado nele armazenado possui uma URI pública, única e encaspulada em um objeto *Uri*. No esquema de dados, elas referenciam tabelas e colunas. Todos as URIs começam com “content://”, que servem para identificar qual entidade de *Content Provider* a está controlando; e possui o formato <*table>.CONTENT\_URI*.

A figura FIG. 3.14 exibe uma representação gráfica de exemplo do modelo de dados do *Calendar Provider*.

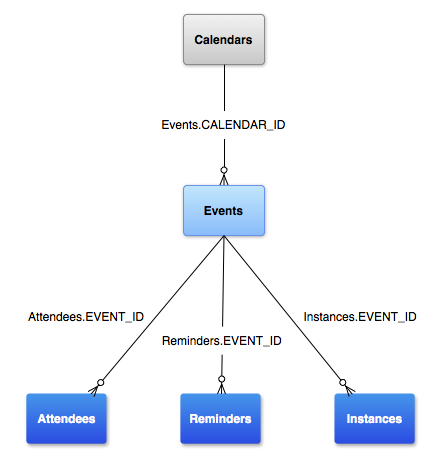


FIG. . – Calendar Provider data model [7].

Uma aplicação pode ter acesso a mais de um calendário, Isso é possível devido ao fato de que cada Objeto *Calendar*s criado recebe uma identificação única de sua base de dados.

JUSTIFICATIVA

O uso do Calendar Provider será necessário para que o sistema desenvolvido possa obter todos os compromissos do usuário. Ele terá a responsabilidade de se comunicar com o sistema do Google Calendar para gerenciar não só as informações de *login*, mas também os compromissos do usuário. Assim, o sistema desenvolvido não terá responsabilidade alguma sobre o gerenciamento de dados referentes ao Google Calendar do usuário. O aplicativo ficará responsável apenas intermediar solicitações do usuário como deletar ou criar novo compromisso.

Apesar de não ser uma necessidade para o projeto,

# FUNCIONAMENTO

Para satisfazer às especificações do aplicativo, utilizou-se os elementos mencionados no capítulo 3, organizados conforme a FIG. 4.1:

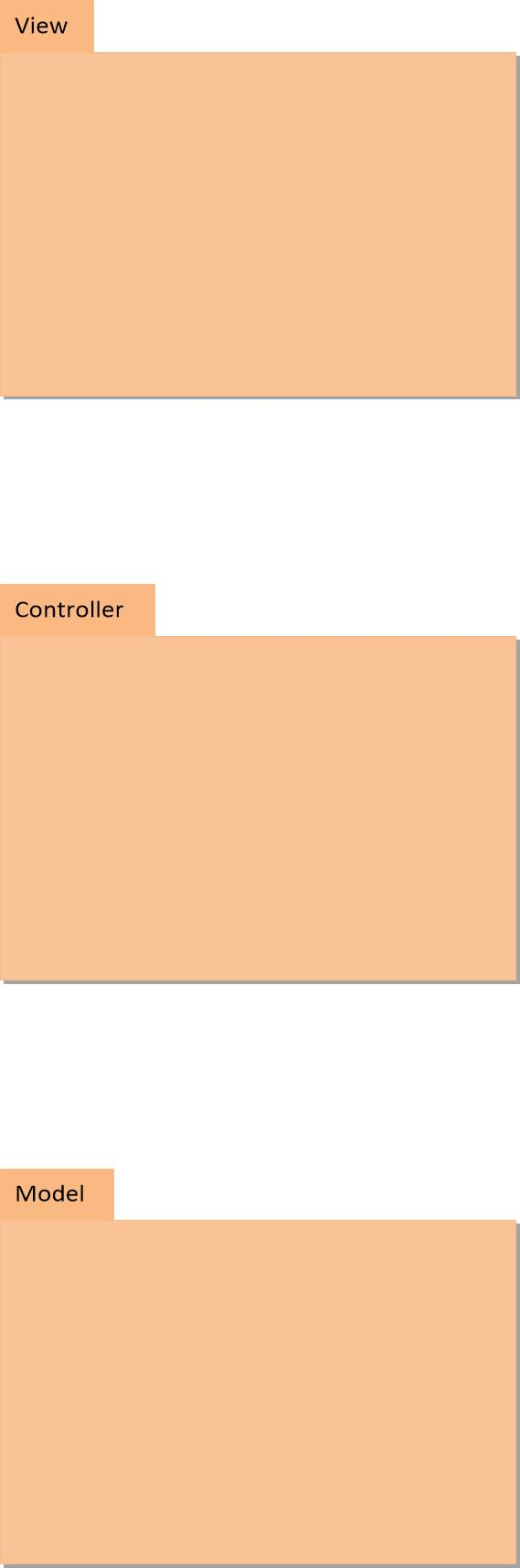


FIG. . – Modelo da arquitetura da aplicação

As classes de projeto do aplicativo colaborarão entre si de forma a prover as seguintes funcionalidades:

* Acessar listas;
* Adicionar item;
* Remover item;
* Remanejar item;
* Compartilhar item;

As colaborações de cada funcionalidades serão detalhadas a seguir.

Sempre que desejar o usuário pode voltar a tela do menu principal, pressionando o botão “up” localizado no canto superior esquerdo de qualquer tela.

## ACESSAR LISTAS

Uma vez no menu inicial, terá a sua disposição o acesso às 5 listas principais do GTD – Nomeadas de Caixa de Entrada, Próximas Ações, Agenda, Delegados, Talvez e Incubadora. Adicionalmente, é exibida a opção de exibir as demais listas que, por ventura, o usuário decida criar, conforme exibido na FIG. 4.2



FIG. . – Tela inicial

Caso o usuário escolha uma das principais listas, uma nova tela é exibida, exibindo os itens nela contidos, conforme a FIG. 4.3.

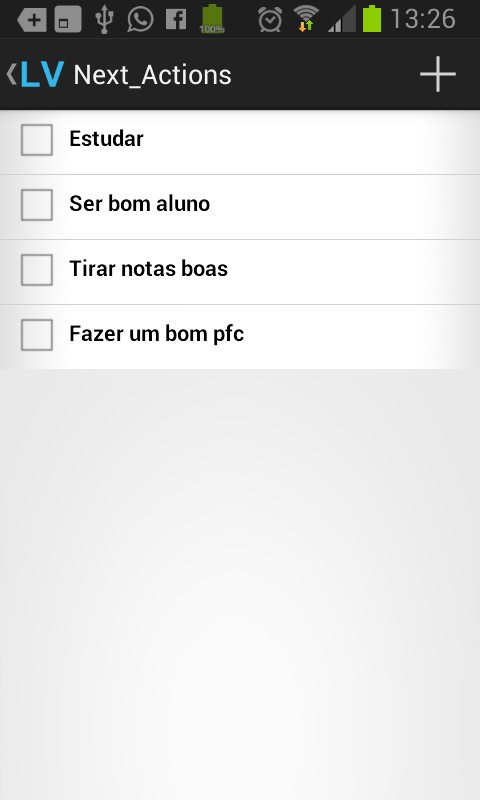


FIG. . – Uma lista aberta

Caso o usuário opte por exibir as demais listas, uma tela exibindo as listas criadas pelo usuário é exibida, conforme FIG. 4.4.

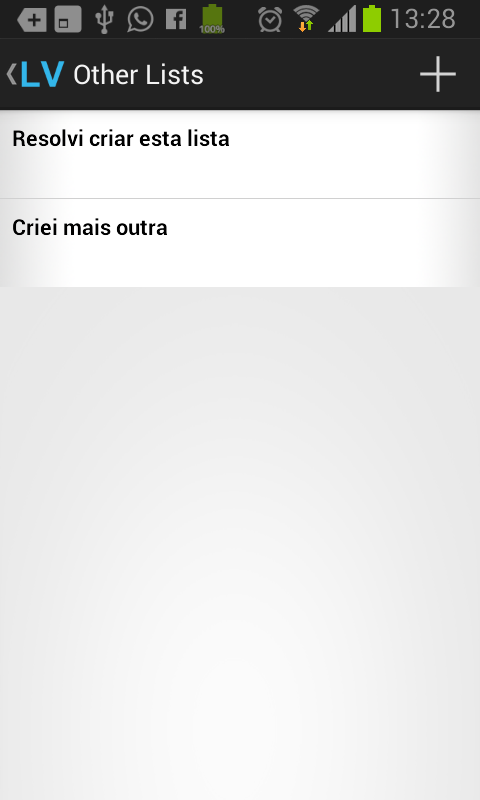


FIG. . – Outras Listas

O usuário ao escolher a lista, é levado à tela que exibe os itens dessa lista.

## ADICIONAR ITEM

Uma vez que se encontre em alguma das listas abertas, o usuário pode adicionar um novo item pressionando o botão com símbolo “+” no canto superior direito da tela, conforme FIG. 4.3.

Uma vez optado por adicionar o item, uma caixa de diálogo é exibida oferecendo um campo para que o usuário digite o texto do novo item que será criado, conforme a FIG. 4.5.

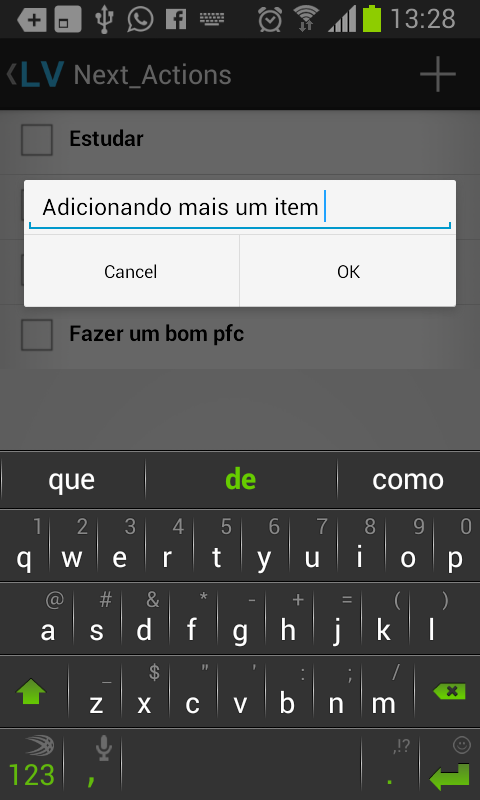


FIG. . – Adicionando item

Caso o usuário se encontre na lista Agenda, ao optar por adicionar novo item, o aplicativo delegará a um aplicativo de calendário, a ser escolhido pelo usuário (FIG FIG. 4.6), para que o novo item – no caso, um compromisso - seja criado.

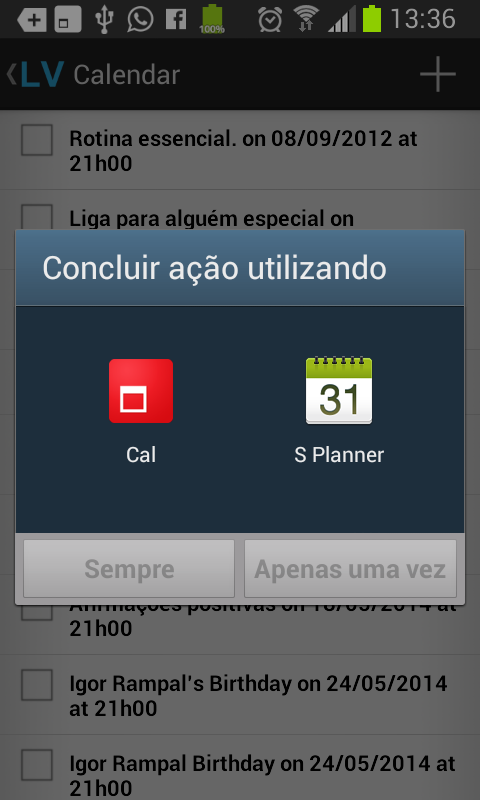


FIG. . – Criando evento utilizando aplicativo externo

## REMOVER ITEM

Em qualquer lista que o usuário se encontre, é possível que ele opte por excluir determinado item de sua lista. Para isso, basta que o usuário exiba menu de ferramentas mantendo pressionado por longo tempo o item desejado. Após ter aberto o menu, demonstrado na figura FIG. 4.7 basta que o usuário clique em excluir – representado por um ícone em forma de lixeira.

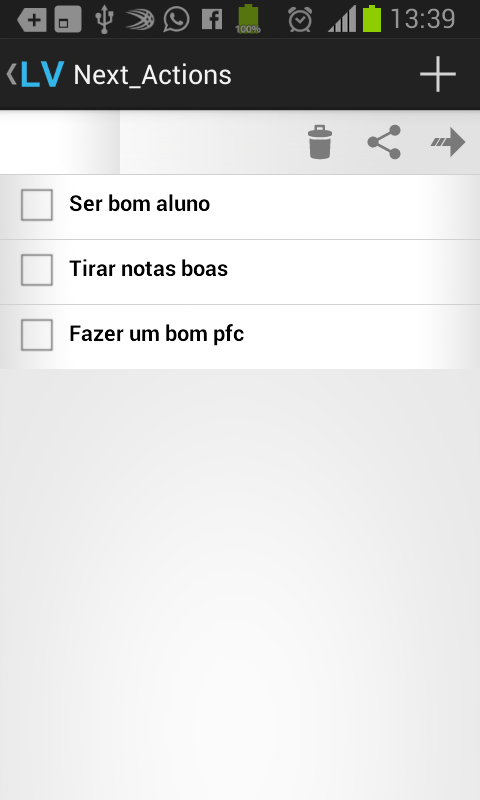


FIG. . – Menu de ferramentas para o item

## REMANEJAR ITEM

Seguindo os preceitos do GTD, no qual um item deve ser remanejado entre diversos contextos conforme a necessidade – representado, no aplicativo, por listas -, o aplicativo permitirá o remanejamento de itens entre listas. O remanejamento envolverá apenas o envio de uma cópia do item para outra lista, uma vez que o usuário provavelmente queira realizar outras ações sobre o mesmo item e ele só deverá ser descartado quando ele finalmente utilizar a funcionalidade de excluir o item.

Em qualquer lista que o usuário se encontre, é possível que ele opte por remanejar determinado item para uma outra lista. Para isso, com o menu de ferramentas do item aberto, basta que o usuário pressione o botão de remanejar – representado por uma seta, conforme exibido na FIG. 4.7.

Em seguida, será exibido para o usuário uma lista contendo todas as listas para que ele possa escolher o destino do item a ser remanejado, conforme ilustra a FIG. 4.8.

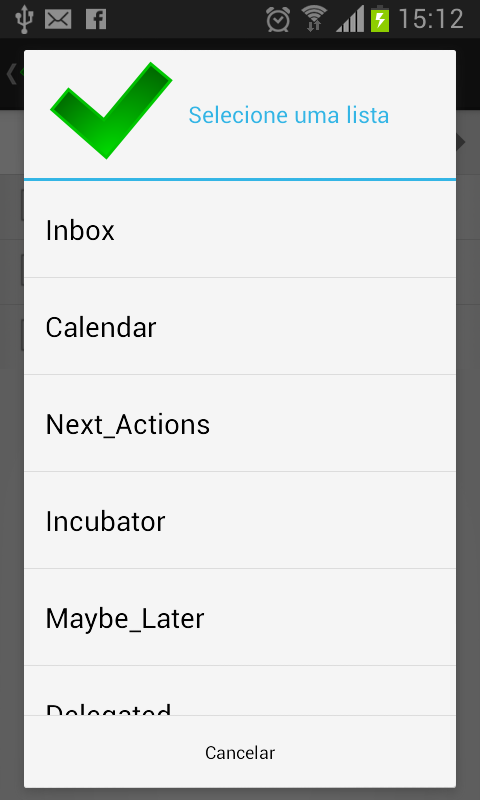


FIG. . – Remanejando um item

Caso a lista de destino seja a Agenda, o usuário deverá escolher um aplicativo externo de calendário para criar um evento. O evento conterá em seu campo de preenchimento de evento do aplicativo, o texto associado ao item escolhido para ser remanejado.

Após feita qualquer escolha de lista, no final, o aplicativo retorna para a última lista aberta.

## COMPARTILHAR ITEM

Em qualquer lista que o usuário se encontre, é possível que ele opte por compartilhar o item por meio de outros aplicativos. Para isso, com o menu de ferramentas do ítem aberto, basta que o usuário pressione o botão compartilhar, conforme FIG. 4.9.

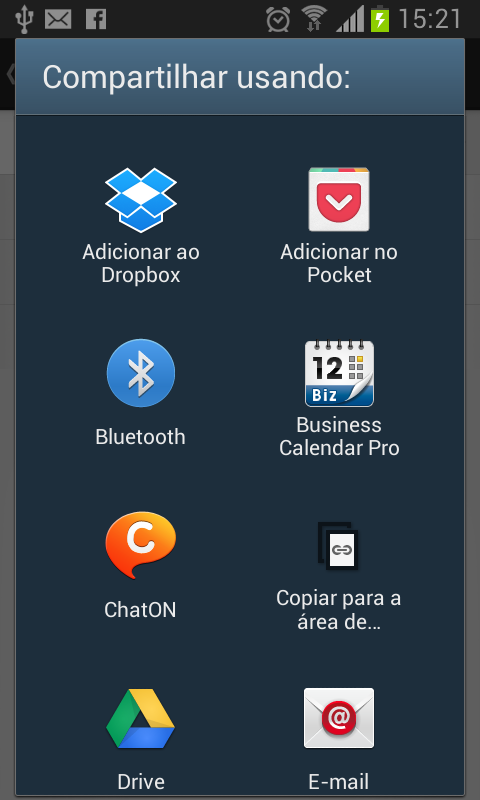


FIG. . – Compartilhar um item

Em seguida o usuário deverá escolher um aplicativo para que possa ser levado ao mesmo e seguir outros passos, sob responsabilidade do aplicativo escolhido.

# CONCLUSÃO

O presente projeto teve como objetivo desenvolver um aplicativo de organização para Android que pudesse utilizar preceitos do GTD em suas funcionalidades.

A primeira etapa do trabalho – levantamento de requisitos – foi cumprida por meio do estudo do método GTD e por meio da explicação dos preceitos no capítulo 2.

A segunda etapa do trabalho – análise de requisitos – foi realizada com sucesso e resultou no capítulo 3 e nos diagramas de caso de uso, localizados no apêndice 7.2.

A terceira etapa do trabalho – projeto do sistema – foi cumprida e resultou no capítulo 3, onde os componentes essenciais para o funcionamento do sistema foram detalhados.

A quarta etapa do trabalho – a codificação – foi realizada com sucesso e resultou no capítulo 4 e no apêndice XXXXXXX . Juntos eles explicam, detalhadamente, o comportamento do sistema e exibem sua estrutura interna após realizada a toda a codificação. A cada funcionalidade codificada, testes foram realizados. Ao fim, todas as funcionalidades, integradas, foram testadas.

Desta forma, o grupo deste projeto conseguiu – por meio de especificação, projeção, codificação e teste – desenvolver um aplicativo capaz de utilizar os conceitos de GTD e ainda prover integração com o Google Calendar.

Como sugestão para trabalhos futuros, indica-se a evolução desse sistema, buscando acrescentar funcionalidades como um assistente que permita o usuário planejar suas ações selecionando os que ele pretenda executar de sua agenda e das demais listas. Outras funcionalidades possíveis seriam indentar (provendo hierarquia), reordenar e editar itens.

# BILIOGRAFIA

x

|  |  |
| --- | --- |
| 1. | ALLEN, D. **A arte de fazer acontecer**. 2a. ed. Rio de Janeiro: Campus, 2005. 300 p. ISBN 8535219080. |
| 2. | ANDROID. Android Developers. **System Architecture**, 2014. Disponivel em: <http://developer.android.com/images/system-architecture.jpg>. Acesso em: 11 Março 2014. |
| 3. | ANDROID''''''''. Android Developers. **Developers**, 2014. Disponivel em: <http://developer.android.com/index.html>. Acesso em: 10 Março 2014. |
| 4. | ISHAK, A. Tute Central. **Android Swipe ListView**, 2014. Disponivel em: <http://www.tutecentral.com/android-swipe-listview/>. Acesso em: 10 Março 2014. |
| 5. | NEIL, T. **Padrões de Design para Aplicativos Móveis**. Tradução de Eduardo Kraszczuk. 1a. ed. São Paulo: Novatec, 2012. 208 p. ISBN 9788575223192. |
| 6. | UML DIAGRAMS. **Android UML Examples**, 2013. Disponivel em: <http://www.uml-diagrams.org/examples/android-uml-examples.html>. Acesso em: 11 Março 2014. |
| 7. | ANDROID. Android Developers. **Calendar Provider**, 2014. Disponivel em: <http://developer.android.com/guide/topics/providers/calendar-provider.html>. Acesso em: 20 Março 2014. |
| 8. | VOGEL, L. Vogella. **Using lists in Android (ListView) - Tutorial**, 2014. Disponivel em: <http://www.vogella.com/tutorials/AndroidListView/article.html>. Acesso em: 10 Março 2014. |
| 9. | SMITH, S. Tuts+. **Android SDK:** Implementing Drag-and-Drop Functionality, 2014. Disponivel em: <http://code.tutsplus.com/tutorials/android-sdk-implementing-drag-and-drop-functionality--mobile-14402>. Acesso em: 10 Março 2014. |
| 10. | GAMMA, E. et al. **Padrões de Projeto - Soluções reutilizáveis de software orientado a objetos**. Tradução de Luiz A. Meirelles Salgado. [S.l.]: Bookman, 1994. 366 p. ISBN 978-85-7307-610-3. |
| 11. | DUCKETT, C. TechRepublic. **Making a sortable ListView in Android**, 2012. Disponivel em: <http://www.techrepublic.com/blog/australian-technology/making-a-sortable-listview-in-android/>. Acesso em: 10 Março 2014. |
| 12. | DALISAY, M. Java Code Geeks. **Android ViewHolder Pattern Example**, 2013. Disponivel em: <http://www.javacodegeeks.com/2013/09/android-viewholder-pattern-example.html>. Acesso em: 10 Março 2014. |
| 13. | CAFEÍNA T.I. Cafeína T.I. **Android – Resolvendo o problema com ListView e CheckBox**, 2014. Disponivel em: <http://cafeinati.wordpress.com/2011/09/05/android-resolvendo-problema-com-listview-e-checkbox/>. Acesso em: 10 Março 2014. |
| 14. | TAMADA, R. Android Hive. **Android SQLite Database with Multiple Tables**, 2013. Disponivel em: <http://www.androidhive.info/2013/09/android-sqlite-database-with-multiple-tables/ >. Acesso em: 12 Março 2014. |
| 15. | DEITEL, P. et al. **Android for Programmers, An App-Driven Approach**. [S.l.]: Prentice Hall, 2012. |

x

# APÊNDICE

## DIAGRAMA DE TELAS

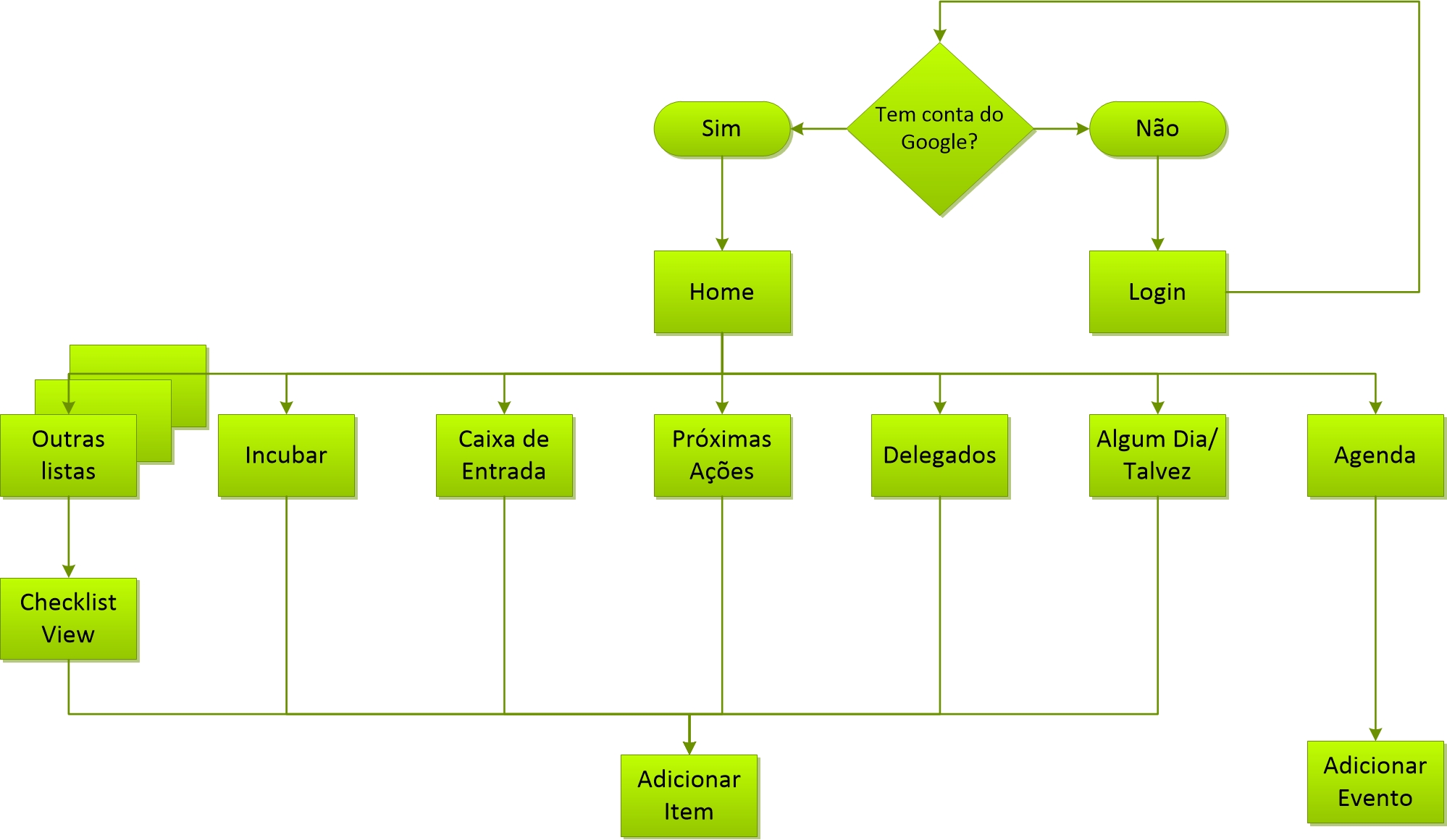


FIG. . – DIAGRAMA DE NAVEGAÇÃO ENTRE TELAS

## DESCRIÇÃO DE CASOS DE USO

### Inserir Item rápido

Ator Principal: Usuário

Precondição: Nenhuma das listas se encontra aberta

Fluxo Básico de eventos:

1. Esse caso de uso começa quando o usuário opta por inserir novo item rápido;
2. O sistema fornece o espaço para que o usuário insira as informações;
3. O usuário insere as informações e confirma;
4. O sistema armazena a informação na lista “caixa de entrada” e o caso de uso termina;

Pós-Condição: (Sucesso) O sistema adicionou inseriu o novo item na caixa de entrada;

Outras Informações: 1- O sistema, por padrão, colocará o novo item na última posição da lista “caixa de entrada”.

2 – A maneira de realizar o passo 1 ainda não está definida.

### Inserir Item

Ator Principal: Usuário

Precondição : Que alguma lista, com exceção da agenda, esteja sendo utilizada.

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por inserir um novo item na lista em que o usuário esteja utilizando
2. Um espaço é gerado na lista pelo sistema e o sistema solicita que o usuário insira as informações.
3. O usuário digita as informações no espaço gerado;
4. O sistema armazena as informações geradas e o caso de uso termina;

Pós-Condição: (Sucesso) O sistema insere o novo item criado pelo usuário;

Outras Informações: 1 – Ao optar inserir novo item, o usuário pode escolher, por meio de gesto ou outro recurso a ser definido posteriormente, em que posição na lista ele quer inserir o novo item.

### Inserir Compromisso

Precondição: O sistema deve possuir uma conta de usuário do Google Calendar cadastrada.

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por inserir um novo compromisso;
2. O usuário fornece informações como nome do compromisso, data de início e data de término;
3. O sistema armazena as informações, confirmando visualmente a inserção do novo compromisso e o caso de uso termina;

Pós Condição: - O sistema armazenou o novo compromisso;

Outras Informações: 1 - Um compromisso equivale a um item.

### Sincronizar com o Google Calendar

Precondição: O sistema deve possuir um usuário de Google Calendar cadastrado

Ator Principal: Usuário

Ator Secundário: Sistema Google Calendar

Fluxo Básico de Eventos:

1 – Esse caso de uso começa quando o usuário deseja sincronizar sua agenda com a do sistema Google Calendar;

2 – O sistema sincroniza a agenda local com a do Google Calendar e o caso de uso termina;

Fluxo Alternativo:

(Falha de Sincronização)

1 – No passo 2 do Fluxo Básico de Eventos, caso haja falha de sincronização, o sistema deve retornar mensagem de falha de sincronização e o caso de uso termina;

Pós Condição: (Sucesso)O sistema sincronizou com o Google Calendar;

(Falha de Sincronização)O sistema não realiza nenhuma sincronização;

### Cadastrar usuário do Google Calendar

Pré-condição: O sistema não possui nenhuma informação de login no Google Calendar

Ator principal: Usuário

Ator Secundário: Sistema do Google Calendar

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por cadastrar suas informações de *login* do Google Calendar;
2. O sistema solicita o *login* e a senha;
3. O usuário preenche as informações de *login* e senha;
4. O sistema valida as informações de *login* no Google Calendar
5. O Google Calendar valida positivamente.
6. O sistema armazena as informações de *login* e o caso de uso termina;

Fluxo Alternativo:

(Falha de Login)

1 – No passo 5, caso as informações de Login sejam inválidas, o sistema exibe mensagem de informações inválidas e retorna para o passo do fluxo básico de eventos.

Pós-condição: (Sucesso) O sistema armazenou com sucesso as informações de login de usuário do Google Calendar;

### Reordenar Itens

Precondição: Uma lista esteja sendo utilizada pelo usuário

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por reordenar os itens da lista;
2. O sistema solicita que o usuário reordene os itens;
3. O usuário reordena e confirma;
4. O sistema armazena as mudanças e o caso de uso termina;

Pós Condição: (Sucesso) O sistema reordena os itens da lista;

Outras informações: Este caso de uso não será considerado obrigatório para a concretização do projeto.

### Marcar item como processado

Precondição: Nenhuma;

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta marcar um ou mais itens como processado;
2. O sistema alterou o status dos itens marcados pelo usuário para processado e o caso de uso termina;

Pós Condição: (Sucesso) O sistema alterou o status dos itens para processado;

Outras informações: 1 – O status de processado deve possuir um aspecto visual diferente do que não possui, como aparência esmaecida;

### Excluir Item

Precondição: O usuário está usando uma lista e ter itens marcados como processados ou ter aberto um menu de opções para o item;

Ator Principal: Usuário;

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por excluir itens;
2. O sistema exclui os itens selecionados e o caso de uso termina;

Pós Condição: O sistema deletou os itens selecionados pelo usuário;

### Criar nova lista

Pré-condição: Nenhuma

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por criar uma nova lista;
2. O sistema solicita o nome da lista;
3. O usuário fornece o nome da lista que deseja e confirma;
4. O sistema exibe a nova lista e o caso de uso termina;

Pós-condição: O sistema criou uma nova lista;

### Deletar lista

Pré-condição: Nenhuma

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por deletar lista;
2. O sistema confirma solicitação;
3. O usuário confirma;
4. O sistema deleta a lista e o caso de uso termina;

Pós-condição: O sistema excluiu a lista escolhida pelo usuário;

### Abrir lista

Pré-condição: O sistema se encontra exibindo o menu principal

Ator Principal: Usuário

Fluxo Básico de Eventos:

1. Esse caso de uso começa quando o usuário opta por abrir lista;
2. O sistema exibe a lista selecionada e o caso de uso termina.

Pós-condição: O sistema exibiu a lista selecionada pelo usuário.

## ESQUEMA DE BANCO DE DADOS

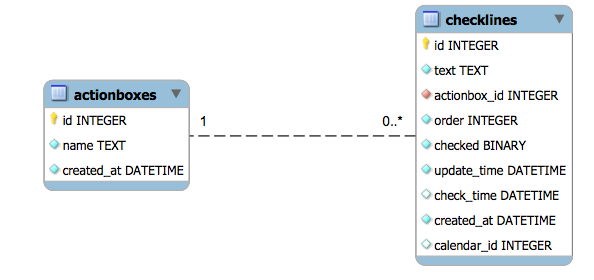


FIG. . – Esquema de banco de dados