APLICATIVO DE GESTÃO DE CASAMENTO

Alan Cristi Souza Resende[1]

RESUMO

Ao decorrer do desenvolvimento, foi notado que as linhas desenvolvidas poderiam ser utilizadas em outros projetos, pois as funcionalidades básicas chamadas de CRUD, podem ser readaptada em qualquer outro projeto, por isso o link desse projeto estará disponível no repositório no Github. Agregar novas funcionalidades em sua aplicação e a razão na qual os desenvolvedores enfrentam tal empreitada, criando certo afeto por suas linhas de código, e levando-as para o resto de sua vida, utilizando em diversas outras aplicações, o processo de desenvolvimento tem como característica, sempre deixar o mais limpo possível e o mais otimizado possível (Pode ocorrer que os exemplos citados no artigo não esteja conforme o código no github). Existem questões na qual ainda necessitam de estudos, que claramente apontam problemas, mas em sua maior parte foram resolvidos, e foram feitos da melhor forma possível na atualidade.
Palavras-chave: Android, Activity, APP, Android Studio

ABSTRACT

From the development, it was noted that as developed lines be used in other projects such as basic features called CRUD, it can be retrofitted in any other project, so the available project link does not repository in Github. Add new features in your application and in the area of risk management, creating right for your lines of code, and taking them for the rest of your life, using other applications, the development process has as a characteristic, always leave the most Possible and as optimized as possible (It may be that the aforementioned quoted in the article are not covered by the code in github). There is a question of quality that needs studies,

[1] Acadêmico em Análise e Desenvolvimento de Sistemas e técnico de Informáticas, ambos os cursos feitos nas Escolas e Faculdades QI, em Gravataí.

which clearly defines problems, but for the most part has been solved, and has been made the best possible way today.

1 INTRODUÇÃO

O tema deu origem a um diálogo pessoal com minha noiva, no qual ela estava se questionando do porquê de os aplicativos de noivas não cumprirem uma simples função, que seria a de listar tarefas de noivas. E complementou que esses aplicativos têm outras funções mais complexas, como contato direto com fornecedores, porém não oferecem uma lista das tarefas com o status de “aberto ou fechado”.

Como um quase recém formado, achei uma boa ideia desenvolver um app que cumpra esse tipo de expectativa até para criar um portfólio, a fim de apresentá-lo em entrevistas.

Obviamente o público alvo seriam mulheres de 18 a 30 anos que planeja se casar. As informações bases para esse projeto, na maior parte, foram absorvidas do site “casandosemgrana.com” que tem o foco de auxiliar noivas com conselhos de uma melhor organização de um casamento em formato de blog.

Basicamente o projeto chamado de “NoivaList” tem como proposta criar um aplicativo simples para gestão de tarefas de noivas.

Esse aplicativo vai simplificar as tarefas ?

As ferramentas e tecnologia utilizadas para criar o aplicativo são ferramentas gratuitas: a IDE Android Studio, o emulador Genymotion. Para versionar o código, utilizou-se o Github. A tecnologia usada é a linguagem Android, baseada em Java. Para persistência de dados foi usado SQLite.

2 TECNOLOGIAS UTILIZADAS

2.1 ANDROID

Desenvolvido especialmente para dispositivos móveis como aparelhos celulares, tablets, e mais atualmente Wearables (Dispositivo “Vestível” como um relógio.)

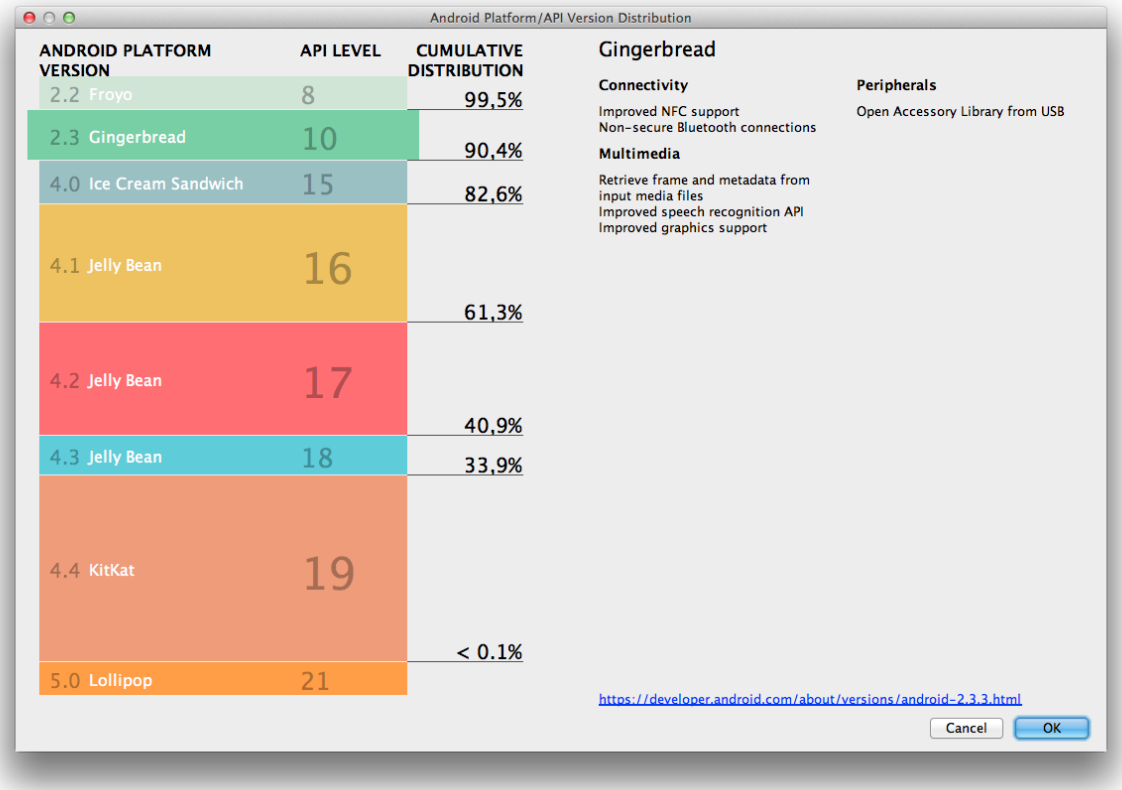
O android foi idealizado pela Google, depois passou a ser a Open HandSet Alliance, mas o Google que é responsável pela engenharia e direitos de uso do sistema operacional. em 2005 a Google comprou a Android inc, na qual tinha começado a trabalhar em um sistema operacional baseado em Linux (Sistema livre de licença).

Apenas em 2008 a Google lançou um device com android chamado HTC Dream, em 2010 saiu o primeiro tablet em android, lançado pela Motorola, chamado de Motorola Xoon, na versão 3.0 do sistema.

Hoje em dia o android é usado em diversas marcas de fabricantes de smartphones, como Sony, Motorola, Samsung, Lenovo. O motivo de estar em várias marcas e justamente por ele ser gratuito.

Em 2014 uma versão estável foi lançado o android 5.0 chamado de Lollipop, em maio de 2015 foi lançado a versão 6.0 chamada de Marshmallow que é a mais atual.

Em sua IDE de desenvolvimento o android mantém uma estatística de uso das versões do android, para que o desenvolvedor escolha a versão mais utilizada.



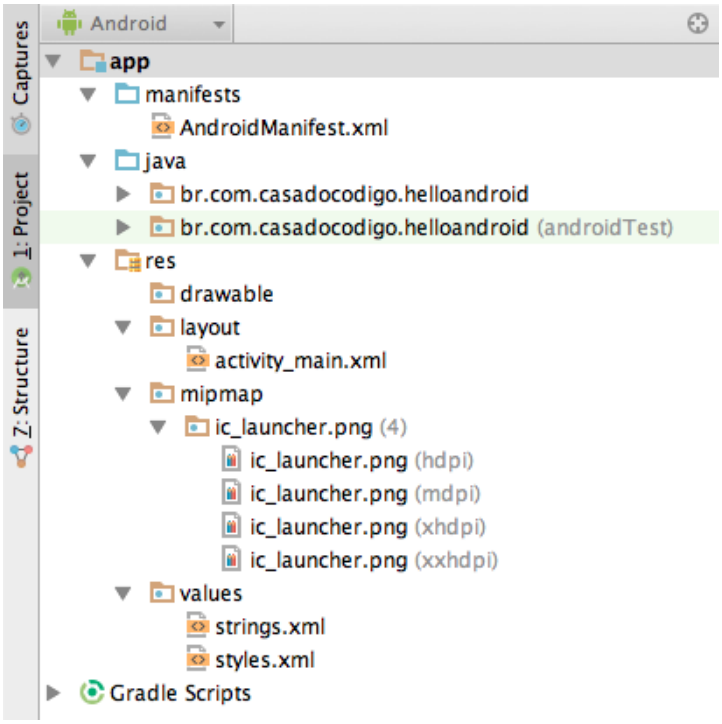
Desenvolvimento Android: O desenvolvimento de aplicativos, ocorrem através da IDE chamada Android Studio, essa ferramenta é responsável por compilar os trechos de códigos para a linguagem binária. O Android studio tem integrado um emulador da versão que deseja para ver o andamento do desenvolvimento, ou até mesmo testar outras versões em um único código, O emulador nativo se chama AVD (Android Virtual Device), mas por questões de otimização, vou utilizar um emulador externo chamado Genymotion.

Genymotion: A diferença de usar o Genymotion em vez do AVD nativo da IDE do android, e a execução leve e a variedade de versões de SO disponíveis, mas para rodar o Genymotion necessita instalar o VirtualBox.

SQLite: É uma banco de dados compacto, muito utilizado para desenvolvimento mobile, sua grande vantagem e por ser multiplataforma e de licença gratuita, e sua sintaxe é bem similar aos bancos de dados Mysql e SQL Server.

2.2 ESTRUTURA DO PROJETO:

A IDE Android Studio trabalha com o conceito básico de visões, agrupando arquivos em pastas, seguindo uma determinada lógica e com objetivo. A organização e feito pelos itens mais usuais no desenvolvimento de um aplicativo que são: Manifestos, código fonte e os recursos adicionais como arquivos de *layout*, ícones e imagens.



Manifesto: este diretório armazena o arquivo chamado AndroidManifest.xml, e um arquivo padrão para cada aplicação android, na qual você define o arquivo principal da sua aplicação, ou se o app precisa de acesso à internet.

Java: diretório criado para o código fonte, na qual fica toda a lógica de negócio, ou captura, ou envio de dados, basicamente toda a parte *backend*.

Res: diretório responsável por armazenar tudo em relação a *layout*, imagem ou animação. Também é responsável pelo XML na qual armazena *Strings* e *arrays* de valores, que são acesśiveis pela classe nativa R.

Também existem dois diretórios no qual armazenam itens específicos.

Assets: este diretório armazena diversos tipos de arquivos utilizados pela aplicação. São acessíveis apenas programaticamente.

Libs: diretório que armazena bibliotecas de terceiros utilizadas pela sua aplicação.

Gradle Scripts: *Gradle* é um sistema que automatiza os *builds* no AS. Toda vez que se cria uma projeto novo por padrão, é vinculado um *Gradle* da aplicação, na qual se define a API da versão do android e as dependências que serão utilizadas no app.

3 DESENVOLVIMENTO :

O projeto se inicia com dois arquivos relacionados, activity_main.xml e ActivityMain.java, os componentes do arquivo XML e um botão do estilo floatButton, e uma lista ListView. Dentro do arquivo java, o botão float é instanciado.

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
```

Depois da instância, é implementado o método OnClickListener.

```
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MainActivity.this, TaskActivity.class);
        startActivity(intent);
    }

});
```

Automaticamente a IDE implementa o onClick(), para ir a outra Activity. É necessário utilizar uma Intent dentro do onClick.

Criou-se uma intent da Activity Main, para a TaskActivity.

```
Intent intent = new Intent(MainActivity.this, TaskActivity.class);
        startActivity(intent);
```

Primeiramente foi instanciada e colocada na variável intent.

Como parâmetro colocou-se a *Activity* atual (MainActivity), por isso é utilizado o .this, o segundo parâmetro e a *Activity* na qual deseja ir, ao clicar no botão, que no caso seria a TaskActivity.

Depois de configurar os parâmetros, coloca-se o comando *startActivity* e a variável da intent.

Na TaskAcitivity, há um cadastro de tarefas. Em sua interface existe apenas um EditText e um Button, ambos instanciados. A diferença está no onClick() do Button.

```
final EditText inputTask = (EditText)findViewById(R.id.inputTask);
Button btn = (Button)findViewById(R.id.btn_task);
```

Dentro do onClick() do botão de ID btn_task, instanciou-se uma classe chamada BancoController, na qual a programação será informada mais à frente. Logo em seguida é feito getText().toString() do EditText.

```
@Override
public void onClick(View view) {
  inputTask = (EditText)findViewById(R.id.inputTask);
  BancoController controller = new BancoController(getBaseContext());
  System.out.println(inputTask);
  controller.addItem(inputTask.getText().toString());
  finish();
}
```

O método addItem grava os dados no SQLite e pega a ID no elemento da interface. A instância da classe BancoController permite acessar o método addItem para pegar o texto digitado no EditText (getText) e em seguida transformá-lo em *string* (.toString);

O método finish() serve para finalizar a *activity*. Depois que gravar os dados, assim ele retorna-os diretamente para a *activity* principal, na qual se encontra o Listview.

Classe BancoController obtém os métodos de CRUD, que é instanciado em todas as outras, na qual precisa ser listado ou inserido contéudo do banco.

Método de inserir: na classe ele se chama addItem, nele existe uma variável do tipo *string* chamada de sql, que recebe a query de select.

```
String sql = "INSERT INTO task (nome) VALUES ('"+nome+"')";
```

O resto do método contém o SQLiteDatabase e trata-se de um método exportado nativamente para gerenciar operações comuns no Sqlite.

O ContentValue também é uma importação nativa, que serve para armazenar um conjunto de valores para o ContetResolver processar.

Depois que se instanciou o ContetValue, deve-se usar o *put*, um operador do ContenValue, que envia as informações para ele.

Por fim utiliza-se o método *insert* do SQLiteDatabase. Deve-se informar como parâmetros o nome da tabela do SQLite, um nullColumnHack que, no caso, é null, e o *values*, que, no caso, é a variável da instância do ContentValues.

Para gravar os dados no banco, usa-se a função execSQL();

O android executa uma única instrução, na qual não devem retornar dados. O único parâmetro a ser usado é um do tipo *String*, que contém a instrução de query, ou seja, a variável sql. Por último o método fecha o banco de dados db.close();

```
public void addItem(String nome){
  String sql = "INSERT INTO task (nome) VALUES ('"+nome+"')";
  SQLiteDatabase db = acessoBD.getWritableDatabase();
  ContentValues values = new ContentValues();
  values.put("", nome);
  db.insert("task ", null, values);
  db.execSQL(sql);
  db.close();
}
```

O getWritableDatabase, e usado apenas na escrita, ele serve para colocar na instância do SQLiteDatabase para o método saber que ele deve apenas escrever.

Método getAllItems(): que é o meu método de coleta.

Ele utiliza alguns recursos do método de inserção(addItem), como o SQLiteDatabase e a variável sql na qual recebe a query(uma query diferente).

Os métodos de coleta trabalham com cursor e arrayList, isso deixa a tarefa um pouco mais complicada, O método rawQuery tem a função de receber os dados da query e passar para um cursor.

```
cursor = db.rawQuery(sql, null);
```

Depois é criado um if para ver se o cursor está vazio, se aprovado eles entram em um do while que tem o método add. Por fim o método moveToNext() verifica se ele percorreu toda a extensão do cursor. No fim do método retorna o array itens.

Classe AcessoBanco define a estrutura do banco, literalmente o *create* Table do Mysql, porém este são os atributos concatenados com variáveis. A classe AcessoBanco estende da SQLiteOpenHelper e são criados dois métodos nativos dessa classe, o **public void onCreate**() no qual se utilizam para definir as variáveis a serem utilizadas para criar a estrutura da tabela.

As variáveis, para criar o banco, devem ser privadas, estáticas e finais, criando um atributo imutável, inalterável, conseguido a partir da adição do modificador final.

```
private static final String DB_NAME = "Banco";
```

O método **public void onUpgrade** é utilizado para dar *upgrade* na estrutura do banco. Inicialmente esse método não será utilizado.

Na String TABLE_ITENS tem a query de criar a tabela que se chama task com os atributos ID e NOME. No onCreate e passado por parâmetro o SQLiteDatabase é executado o execSQL com a variável TABLE_ITENS.

```
public class AcessoBD extends SQLiteOpenHelper {

  private static final String NOME_BANCO = "noivalist";
  private static final int VERSAO = 1;

  private static String TABLE_ITENS =
        "CREATE TABLE task("+
              "id INTEGER PRIMARY KEY AUTOINCREMENT,"+
              "nome TEXT"+
              ");";


  public AcessoBD(Context context){
```

```
        super(context, NOME_BANCO, null, VERSAO);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(TABLE_ITENS);

    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Chamando os métodos na Activity principal: Como a Listview vai ficar na Activity principal, o método getAllItens deve ser chamado lá. No começo do código e feito a instância do arrayList de Strings que se chama itens. Também será instanciado a classe BancoController, o trecho de código fica como o abaixo:

```
final ArrayList<String> itens;
BancoController controller = new BancoController(this);

itens = controller.getAllItens();
```

O acesso ao getAllItens() se da a instancia da classe BancoController(); e passo tudo para o arrayList itens, depois deve ser criado um arrayAdapter para a Listview.

A criação do arrayAdapter na verdade é uma instância de uma classe nativa do android, basta passar o tipo de dado que irá ser inserido no Listview, cria uma variável de acesso define os parâmetros como o simple_list_item_1, é a variável do array que vai alimentar essa lista. Depois deve ser pego o ID do elemento da interface para servir de referência ao deposito das informações, e por fim ocorre o “setamento” do adapter, o código deve ficar parecido com o abaixo:

```
final ArrayList<String> itens;
BancoController controller = new BancoController(this);

itens = controller.getAllItens();
final ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, simple_list_item_1, itens);
final ListView listview = (ListView) findViewById(R.id.lista);
listview.setAdapter((ListAdapter) adapter);
```

Problemas encontrados: Ao criar o método de alimentação da Listview notei que a lista não era atualizada automaticamente depois de uma inserção, apenas quando o app era reiniciado. Como solução tive que optar em usar o método onRestart(); para que isso ocorresse foi necessário replicar o método criado no onCreate da activity principal. Chamado de LoadList().

Depois de re criado, foi necessário criar um método com @Override chamado onRestart(), por definição da documentação do android, o onRestart() o método toda vez que o estado da activity está como onStart(). Ou seja irá executar o método de getAllItens() toda vez que voltar para activity principal, resolvendo o problema da atualização da lista.

```
@Override
protected void onRestart(){
  super.onRestart();
  LoadList();
}
```

3.1 DESENVOLVIMENTO, PARTE II

Na segunda parte do desenvolvimento do APP, foi implementada a classe de modelo, pois, sem ela, dificulta as referências nos métodos de exclusão de dados. O nome da classe modelo se chama *task*, que implementa um *serializable*, com apenas dois atributos um id do tipo inteiro, e nome do tipo String. O resto do script dessa classe são métodos construtores.

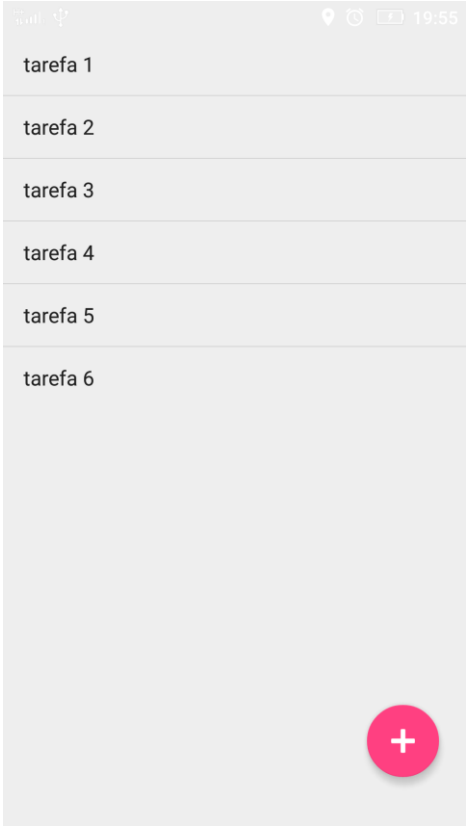
Correção de Bug: na exclusão de um item da lista, o app estava dando erro de cast, por isso a implementação de classe modelo foi necessária. O método onItemLongClick ganhou novas funcionalidades como alertas e confirmação de exclusão.

```
        listview.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
        @Override
        public boolean onItemLongClick(AdapterView<?> parent, View view, int
position, long id) {
            final Context ctx = view.getContext();
            final Task task = (Task)listview.getItemAtPosition(position);

            AlertDialog.Builder builder = new AlertDialog.Builder(ctx);
            builder.setTitle("Confirmação")
                    .setMessage("Tem certeza que deseja excluir este cliente?")
                    .setPositiveButton("Excluir", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            new BancoController(getBaseContext()).deleteItem(task.id);
                            LoadList();
                            Toast.makeText(ctx, "Cliente excluído com sucesso!",
Toast.LENGTH_LONG).show();
                        }
                    })
                    .setNegativeButton("Cancelar", null)
                    .create()
                    .show();
            return false;
        }
        });
```

Parte visual:

activity_main.xml é a activity inicial, nela contém elementos de listview e float button. Ela é responsável por captar o que foi cadastrado no banco, dando a possibilidade de o usuário deletar, pressionando a linha em questão. Também existe o botão de cadastro que, ao clicar nele, é encaminhado para outra *activity*.

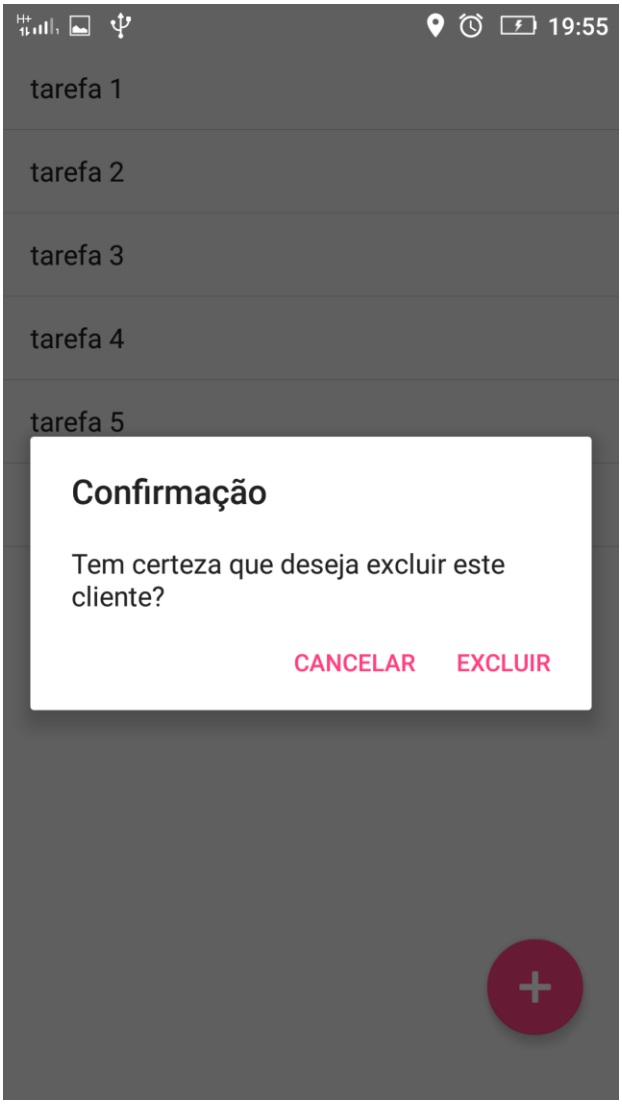


Activity_task.xml:

Essa *activity* é chamada quando o usuário clica no botão de cadastro na activity_main. Basicamente tem um editText e um botão, para o que usuário preencha com a tarefa que deseja listar, e em seguida clica em salvar. O mesmo é salvo e redirecionado para a activity_main para ver a listagem.

Activity_main deletar:

Para deletar uma tarefa cumprida, basta pressionar o dedo em cima da tarefa, que logo virá uma confirmação. Basta clicar em confirmar para deletar da lista, ou clicar em cancelar para desfazer o cancelamento.



Updates futuros: o projeto deve ser atualizado com novas *features*, tal como a possibilidade de fornecedores se cadastrarem e listarem seus próprios produtos, Se o cliente final se interessar, poderá negociar ou saber mais informações do fornecedor pelo app.

Com relação à implementação do método de conclusão de tarefas, provavelmente será criada uma nova estrutura de banco, para replicar os itens deletados caso o cliente queira restaurar determinada tarefa.

API do google maps deve ser implementada, pois o fornecedor deve informar sua localização, para que o cliente possa ir até o estabelecimento ver os produtos, ou até mesmo negociá-los pessoalmente.

Também se deve fazer o trabalho de imagem, repaginar o app, dando uma aparência melhor, um novo logo, ícones e cores. Ainda é importante desenvolver o método de alteração, assim como novos atributos para a classe modelo task, como descrição, preço estimado, local, ou até mesmo uma categorização, e futuramente a lista poderá ser filtrada por categoria.

Categorias personalizadas, a ideia de APP é deixar o mais maleável possível para o cliente final, pois é ele quem entende mais a sua necessidade. Nesse caso não adiantaria deixar categorias definidas, pois quando se trata de casamento, ele é sempre uma constante de mudanças, em questões de estilos e tendências.

Feitas essas alterações, o app deve ir para a loja, de forma gratuita, é o código deve ser disponibilizado no github para que qualquer pessoa tenha acesso.

Implementação do logo e do ícone do APP.

Updates futuros II: remoção de query, trocar para os comandos nativos do SQLiteOpenHellper; aplicação do firebase nas movimentações de dados.

4 CONCLUSÃO

Desenvolver esta aplicação demonstrou ser um desafio complicado relacionando à vida pessoal e profissional, problemas com referências técnicas e resoluções de problemas mostrou-se melhor tratando com a ajuda de comunidades de profissionais ou até mesmo conversas pessoais com professores e alunos de faculdades do que propriamente com livros. Por ser uma área nova e sempre atualizada, os livros de desenvolvimento se mostraram muito desatualizados para o que estávamos procurando. Provavelmente alguns recursos utilizados no desenvolvimento deste singelo APP já vão estar em defasados no final deste artigo.

Mas acreditamos que esta aplicação irá ajudar muitos novatos no desenvolvimento Android a perceber como funcionam os métodos básicos de um APP. Em especial este aplicativo foi criado com a ajuda da comunidade brasileira de desenvolvimento android (Androiddevbr).

REFERÊNCIAS

DUARTE, Luiz. *Criando APPS Para Empresas Com Android* . Porto Alegre: Amazon, 2016.
___. *Meu Primeiro APP Android*. Porto Alegre: Editora, 2016.

Bosco Monteiro, João. Google Android Crie Aplicações Para Celular e Tablets. São Paulo: Casa do Código, 2015