CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE BOTUCATU

Programação para Dispositivos Móveis

Guia para criação de um projeto Android no Android Studio para uso do recurso de armazenamento de dados em banco de dados

Osvaldo Cesar Pinheiro de Almeida

(cesar@fatecbt.edu.br)

Botucatu - SP

Outubro - 2016

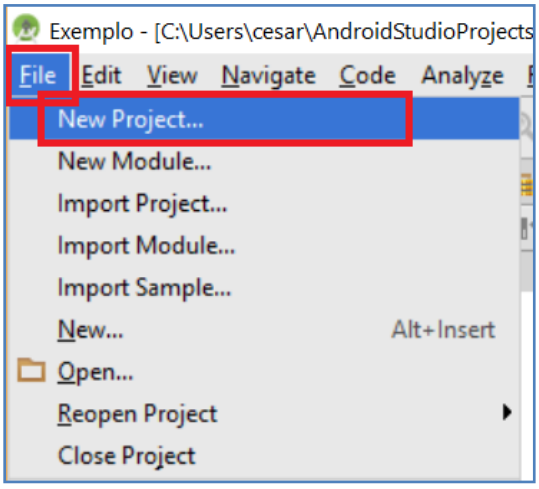
Inicialização

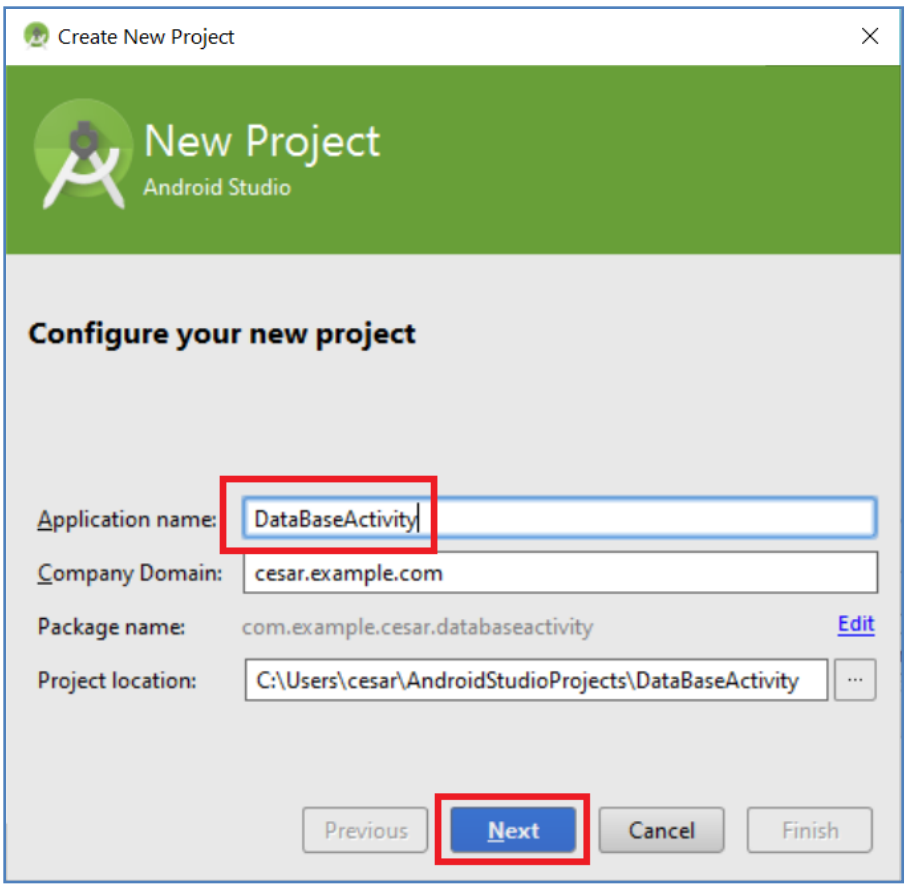
Requisitos iniciais:

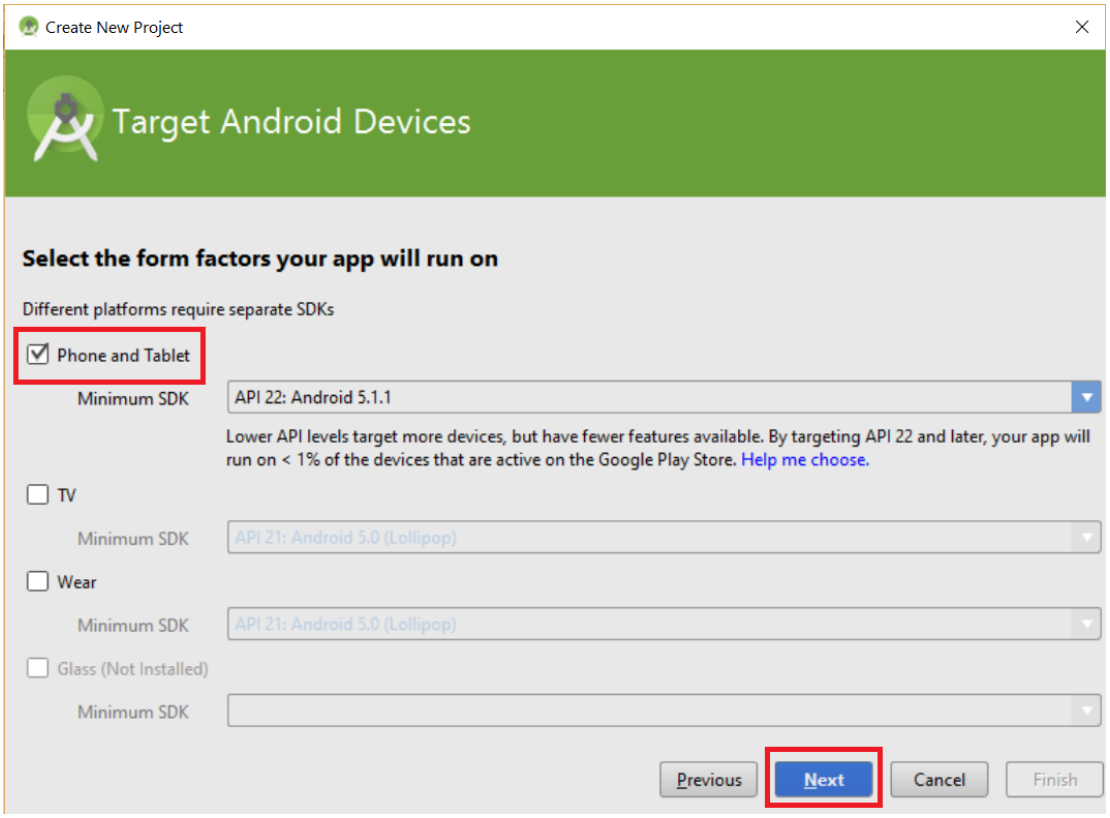
- Android Studio completo - Versão recomendada do Androis 1.1 ou maior;
- Observe as imagens com destaque em vermelho.

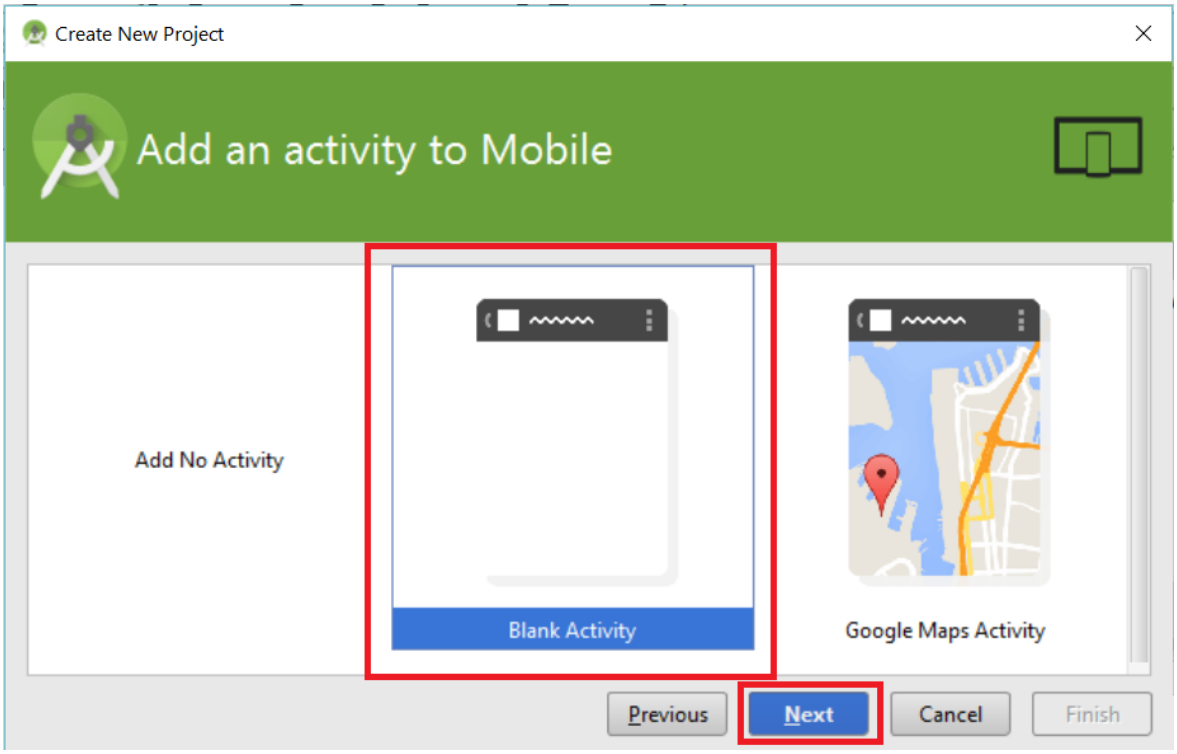
Criando Projeto

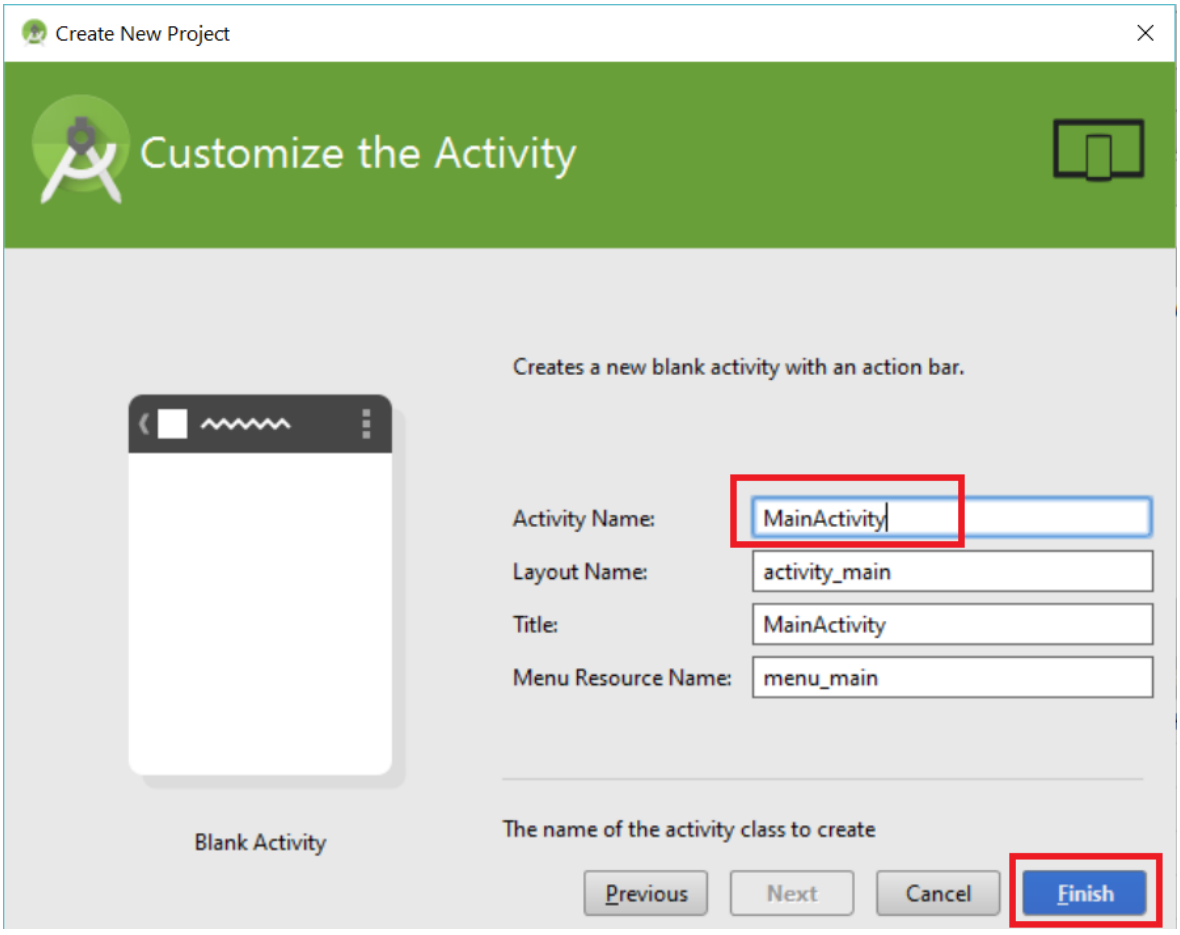
Usando o Android Studio iremos criar um novo projeto:

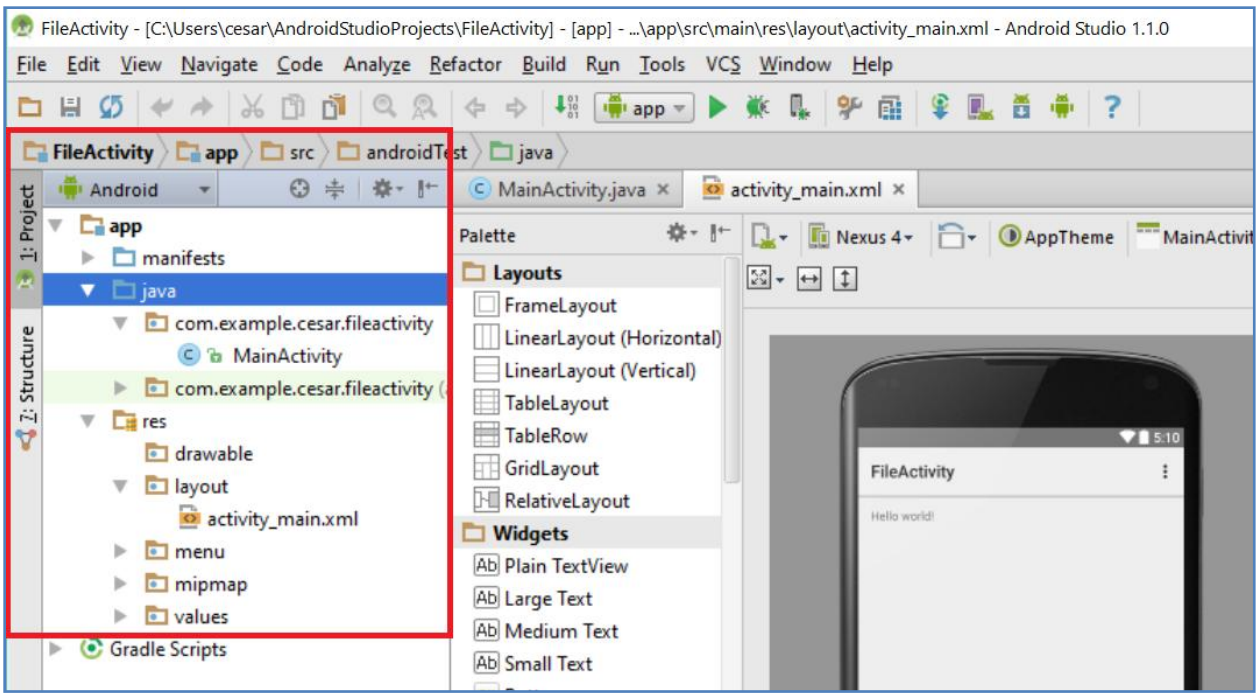



Create New Project
New Project
Android Studio
Configure your new project
Application name:
DataBaseActivity
Company Domain:
cesar.example.com
Package name:
com.example.cesar.databaseactivity
Edit
Project location:
C:\Users\cesar\AndroidStudioProjects\DataBaseActivity
Previous
Next
Cancel
Finish


Create New Project
Target Android Devices
Select the form factors your app will run on
Different platforms require separate SDKs
Phone and Tablet
Minimum SDK
API 22: Android 5.1.1
Lower API levels target more devices, but have fewer features available. By targeting API 22 and later, your app will run on < 1% of the devices that are active on the Google Play Store. Help me choose.
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
Glass (Not Installed)
Minimum SDK
Previous
Next
Cancel
Finish


Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Google Maps Activity
Previous
Next
Cancel
Finish


Create New Project
Customize the Activity
Creates a new blank activity with an action bar.
Activity Name:
MainActivity
Layout Name:
activity_main
Title:
MainActivity
Menu Resource Name:
menu_main
Blank Activity
The name of the activity class to create
Previous
Next
Cancel
Finish



Nesse projeto utilizaremos a Activity para permitir que o usuário possa trabalhar com armazenamento de informações em banco de dados. Os dados serão armazenados usando o banco de dados SQLite, banco de dados nativo do sistema Android. Para ilustração dos conceitos será utilizado armazenamento de informações sobre cliente.

Assim como no armazenamento de arquivos, o Android faz o armazenamento das aplicações no banco de dados usando o armazenamento interno. Por essa razão, os dados são privados para a aplicação que foi usada para manipular esses dados. Além disso, caso a aplicação seja removida do sistema, todos os dados vinculado à ela serão deletados.

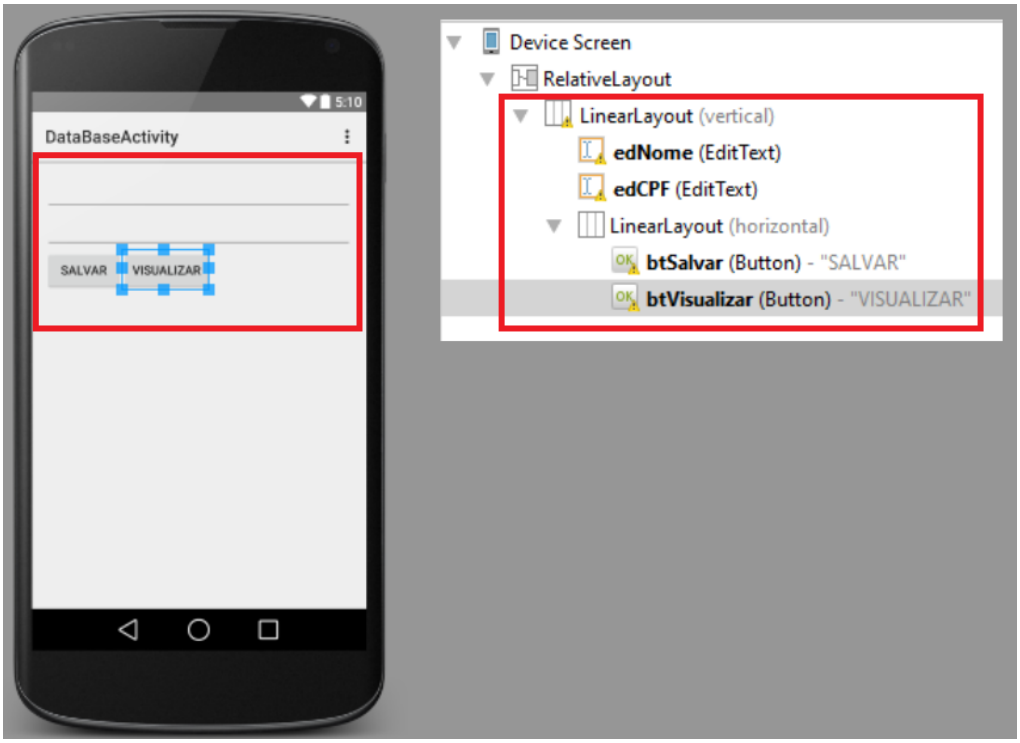
A aplicação de exemplo que iremos apresentar vai utilizar duas Activitys: uma para fazer o cadastro de nome e CPF de um cliente; outra para visualizar os dados de todos os clientes que já foram cadastrados. Dessa maneira, além da MainActivity (para cadastrar os dados de um cliente) teremos também a ViewActivity (para exibir os dados dos clientes).

Para usar de maneira organizada o banco de dados no Android, serão criadas as classes Cliente (para modelar os dados de cliente), ClienteContract (para registrar a estrutura da base de dados usada no projeto) e ClienteDbHelper (para implementar as operações de gravação e consulta dos dados no projeto).

Configurando o Layout MainActivity

Para a criação do layout de exemplo da MainActivity serão usados 5 componentes gráficos, sendo 2 caixas de entrada de texto, 2 botões e 1 lista de visualização. Além disso, serão usados elementos "LinearLayout", tanto vertical e horizontal para ajustar a posição dos componentes visuais.

Após remover o texto "hello world!", será adicionado o componente de posicionamento LinearLayout (vertical). Depois iremos inserir, na seqüência: um LinearLayout (horizontal) com um EditText (para a entrada do nome do cliente a ser armazenado), mais um EditText (para a entrada do CPF do cliente) e um LinearLayout (vertical) contendo um botão para salvar as informações no banco de dados e um botão para visualizar os dados de clientes já armazenados. Esse último botão deverá fazer a requisição de uma nova Activity. O layout finalizado deve ficar como a figura abaixo.

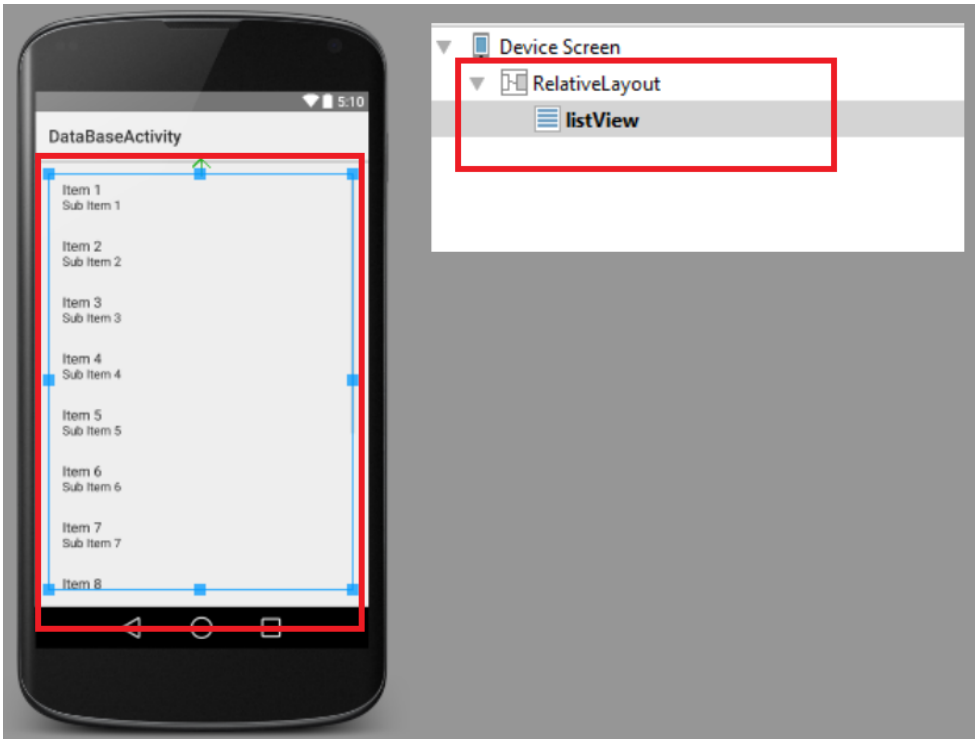


Como pode ser observado é importante mudar o "id" dos componentes para que possamos manipular os componentes na codificação. Assim, atente-se aos campos de EditText com os nomes edNome (para a entrada do nome) e edCPF (para a entrada do CPF). Além dos botões btSalvar, que será usado para salvar as informações das caixas de entrada no banco de dados, e btVisualizar, que será usado para fazer a chamada da ViewActivity, que irá exibir os dados dos clientes cadastrados no banco.

Configurando o Layout ViewActivity

Para a criação do layout de exemplo da ViewActivity será usado apenas o componente ListView, para listar os clientes cadastrados no banco de dados.

Devemos então criar no projeto a ViewActivity e editar o seu layout. Primiero removemos o texto "hello world!". Depois será adicionado o componente ListView. O layout finalizado deve ficar como a figura abaixo.



Criando as classes

O projeto se baseia no uso de três classes que devem criadas:

- Cliente.java

A classe Cliente.java deve ser criada de maneira que contenha os dados id, nome e cpf, com seus respectivos métodos "get" e "set", dois métodos construtores, sendo um com parâmetros para o nome e o CPF e outro com parâmetros referentes a todos os atributos da classe, além da sobrescrita do método "toString". A classe com a sua implementação deve ficar como a figura abaixo.

```java
public class Cliente {

    private long id;
    private String nome;
    private String cpf;

    public Cliente(long id, String nome, String cpf) {
        this.id = id;
        this.nome = nome;
        this.cpf = cpf;
    }

    public Cliente(String nome, String cpf) {
        this.nome = nome;
        this.cpf = cpf;
    }

    public long getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    @Override
    public String toString() {
        return id + " - " + nome + " - " + cpf;
    }
}
```

- ClienteContract.java

A classe ClienteContract.java deve ser criada de maneira que contenha as definições básicas da base de dados qu será criada. No exemplo será definida uma tabela para cliente. Caso fosse necessário criar novas tabelas, suas estruturas seriam definidas nessa classe. A codificação da classe deve ficar como ilustrado na figura abaixo.

```java
public final class ClienteContract {

    public ClienteContract() {
    }

    public static abstract class ClienteDb implements BaseColumns {
        public static final String TABLE_NAME = "cliente";
        public static final String COLUMN_NOME = "nome";
        public static final String COLUMN_CPF = "cpf";
    }

}
```

- ClienteDbHelper.java

A classe ClienteDbHelper.java deve ser criada para realizar a manipulação da tabela no banco de dados. Ela será responsável por fazer as operações de acesso do banco. Para isso, a classe deve estender a classe SQLiteOpenHelper. A codificação da classe deve ficar como na figura abaixo.

```java
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import com.example.cesar.databaseactivity.ClienteContract.ClienteDb;

import java.util.ArrayList;

public class ClienteDbHelper extends SQLiteOpenHelper{
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "Cliente.db";

    private static final String CREATE = "create table " + ClienteDb.TABLE_NAME + "( "
            + ClienteDb._ID + " integer primary key autoincrement, "
            + ClienteDb.COLUMN_NOME + " text, "
            + ClienteDb.COLUMN_CPF + " text)";
    private static final String DELETE = "drop table if exists " + ClienteDb.TABLE_NAME ;


    public ClienteDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE);
    }
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(DELETE);
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }

    public boolean salvarCliente(Cliente cliente){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(ClienteDb.COLUMN_NOME, cliente.getNome());
        contentValues.put(ClienteDb.COLUMN_CPF, cliente.getCpf());
        long id = db.insert(ClienteDb.TABLE_NAME, null, contentValues);
        cliente.setId(id);
        return true;
    }

    public ArrayList consultarClientes(){
        ArrayList lista = new ArrayList();
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery("select * from " + ClienteDb.TABLE_NAME, null);
        while(cursor.moveToNext()){
            lista.add(new Cliente(cursor.getLong(cursor.getColumnIndex(ClienteDb._ID)),
                    cursor.getString(cursor.getColumnIndex(ClienteDb.COLUMN_NOME)),
                    cursor.getString(cursor.getColumnIndex(ClienteDb.COLUMN_CPF))));
        }
        return lista;
    }
}
```

Programando os processos na classe MainActivity

A programação dos processos deve ser feita na classe MainActivity. No exemplo serão utilizados dois processos, um vinculado ao botão "btSalvar" e outro vinculado ao botão "btVisualizar". A programação da classe constará das seguintes implementações:

1) Definição de atributos que irão representar as duas caixas de entrada e o objeto da classe ClienteDbHelper, fazendo suas inicializações no método "onCreate";

```
public class MainActivity extends Activity {

    private ClienteDbHelper base;
    private EditText nome, cpf;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        base = new ClienteDbHelper(getApplicationContext());
        nome = (EditText) findViewById(R.id.edNome);
        cpf = (EditText) findViewById(R.id.edCPF);
    }
```

2) Criação de um método chamado "salvarCliente", que irá implementar o processo para realizar o armazenamento dos dados de um novo cliente no banco de dados. Este método deve ser vinculado ao botão "btSalvar";

```
public void salvarCliente(View view){
    Cliente cliente = new Cliente(nome.getText().toString(),cpf.getText().toString());
    base.salvarCliente(cliente);
    nome.setText("");
    cpf.setText("");
}
```

3) Criação de um método chamado "visualizarClientes", que irá implementar o processo de chamada da ViewActivity, para a exibição dos dados dos clientes que já foram cadastrados. Este método deve ser vinculado ao atributo "onClick" do botão "btVisualizar";

```
public void visualizarClientes(View view){
    Intent intent = new Intent(getApplicationContext(),ViewActivity.class);
    startActivity(intent);
}
```

Não esqueça de vincular o método "salvarCliente" ao atributo "onClick" do botão "btSalvar" e o método "visualizarClientes" ao atributo "onClick" do botão "btVisualizar".

Programando os processos na classe ViewActivity

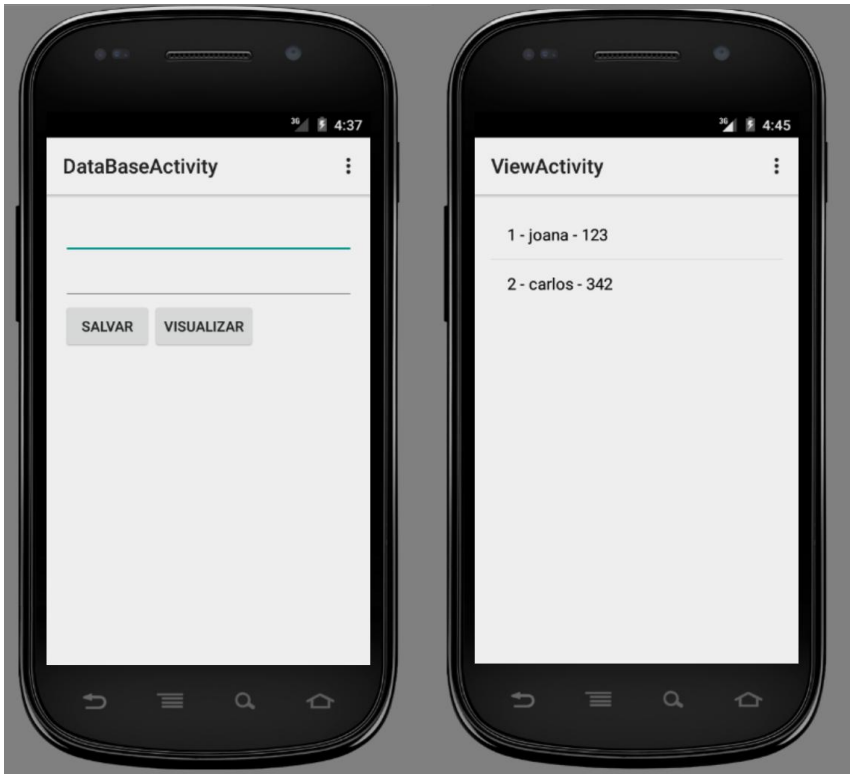
A programação na classe ViewActivity deve implementar o processo de chamada da consulta ao banco de dados para selecionar todos os clientes cadastrados, exibindo a lista de clientes na tela (usando o componente ListView). Essa implementação deve ser realizada no método "onCreate".

```
public class ViewActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view);

        ListView lista = (ListView) findViewById(R.id.listView);
        ClienteDbHelper base = new ClienteDbHelper(getApplicationContext());
        ArrayAdapter<Cliente> arrayAd = new ArrayAdapter<Cliente>(getApplicationContext(),
                android.R.layout.simple_list_item_1,base.consultarClientes());
        lista.setAdapter(arrayAd);
    }
```

Se tudo estiver correto, basta executar o programa.



Bom trabalho!!!