COLÉGIO TÉCNICO DE CAMPINAS

SIMONE PIERINI FACINI ROCHA

ANDROID

CAMPINAS

2018

SIMONE PIERINI FACINI ROCHA

ANDROID

Apostila de Android elaborada para os cursos de Técnico em Informática e Técnico em Desenvolvimento de Sistemas do Colégio Técnico de Campinas, Departamento de Processamento de Dados

CAMPINAS

2017

SUMÁRIO

Activity

Uma **Activity** é um módulo único e independente que representa as telas de uma aplicação e suas interfaces e funcionalidades correspondentes. Por exemplo, quando um usuário pressiona um botão e um texto aparece na tela.

Classe R

A classe "R" é responsável por fazer a comunicação entre os arquivos ".xml" e os arquivos ".java". Não deve ser alterada manualmente, pois contém constantes que estão relacionadas a tudo o que acontece nos arquivos da pasta Res. Sem essa classe não seria possível buscar os valores de variáveis estáticas no arquivo "strings.xml", por exemplo.

Classe View

A classe View é responsável pela criação de qualquer componente gráfico, ela é a base para os componentes visuais, então todos os controles que falaremos a adiante herdam dessa classe. Representa o bloco de construção básico para os componentes da interface do usuário. A View ocupa uma área retangular na tela e é responsável pelo desenho e manipulação de eventos. A vista é a classe base para widgets, que são usados para criar componentes interativos de UI (botões, campos de texto, etc.). A subclasse ViewGroup é a classe base para layouts, que são contêineres invisíveis que possuem outras Views (ou outros Grupos de Views) e definem suas propriedades de layout

Projeto 1 – Usando as classes R e View

No menu do Android Strudio selecione File/New/New Project

A tela de configuração de um novo projeto será apresentada, onde será colocado o nome da aplicação e domínio. Além disso, configure o local onde o projeto será salvo. Nomeie o projeto como **Aula1**.

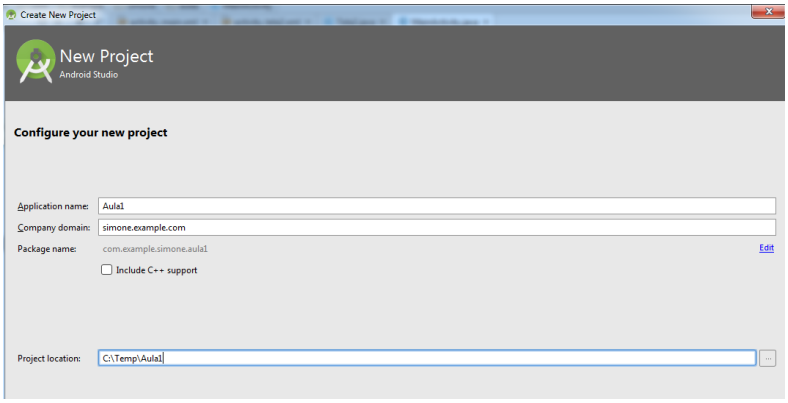


Figura 1 – Criando um novo projeto

Quando um novo projeto é criado, automaticamente a classe da Activity principal é criada junto com o método onCreate, que é responsável por realizar a inicialização para executar uma aplicação e onde são definidos os métodos de definição da interface e os objetos de elementos gráficos da tela.

Para cada Activity, existem os arquivos MainActivity.java, responsável pelas regras de negócio da tela e o arquivo activity_main.xml, onde é configurada interface de objetos da tela

Adicione na tela principal, arquivo activity_main.xml, modo Desing, dois TextView, um EditText e um Button.

Nomeie o primeiro TextView como txtNome, o EditText como edNome, o segundo TextView como txtResultado e o botão como btnProcessar.
Na janela projeto, selecione a pasta **app\res\values** e o arquivo **string.xml**. Edite esse arquivo como o exemplo abaixo.

```
<resources>
    <string name="app_name">Aula1</string>
    <string name="txt_nome">Digite seu nome:</string>
    <string name="btn_processar">Processar</string>
    <string name="txt_resultado">Seja bem vindo!!!!</string>
</resources>
```

No arquivo activity_main.xml faça a lógica de programação conforme o exemplo abaixo e teste o aplicativo no emulator do Android.

```
package com.example.simone.aula1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView txtNome;
    EditText edNome;
    TextView txtResultado;
    Button btnProcessar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtNome = (TextView) findViewById(R.id.txtNome);
        edNome = (EditText) findViewById(R.id.edNome);
        txtResultado = (TextView) findViewById(R.id.txtResultado);
        btnProcessar = (Button) findViewById(R.id.btnProcessar);
        btnProcessar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                txtResultado.setText("Seja bem vindo " + edNome.getText());
            }
        });
    }
}
```

Classe Intent

Representa a intenção da aplicação de realizar alguma tarefa. Permite interagir com componentes do mesmo aplicativo, e com componentes construídos por outras aplicações.

Projeto 2 - Usando a classe Intent

No projeto Aula1, acrescente o botão **Tela 2.** Nomeie o botão como btnTela2 e altere o arquivo string.xml para:

```
<resources>
    <string name="app_name">Aula1</string>
    <string name="txt_nome">Digite seu nome:</string>
    <string name="btn_processar">Processar</string>
    <string name="txt_resultado">Seja bem vindo!!!!</string>
```

```
    <string name="btn_Tela2">Tela2</string>
</resources>
```

Na janela Project, clique com o botão direito do mouse sobre a pasta **app do projeto Aula1,** criado acima, selecione a opção New / Activity / Empty Activity.

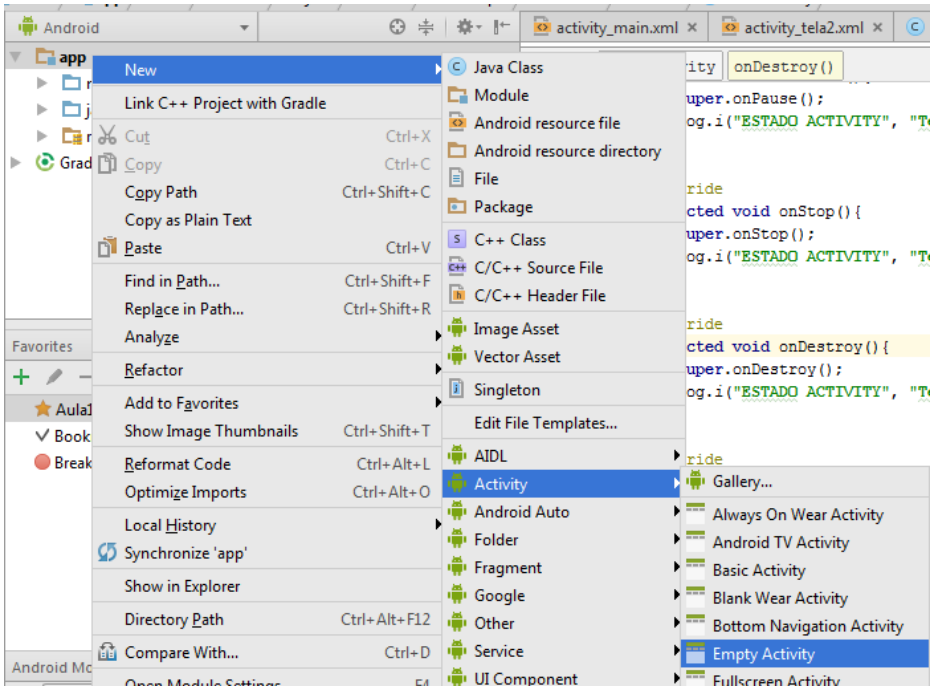


Figura 2 – Criando a segunda tela para testar a classe Intent

Nomeie a nova Activity como Tela2

No arquivo ActivityMain.java acrescente o código, como no exemplo abaixo.

```
public class MainActivity extends AppCompatActivity {

    TextView txtNome;
    EditText edNome;
    TextView txtResultado;
    Button btnProcessar;
    Button btnTela2;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtNome = (TextView) findViewById(R.id.txtNome);
    edNome = (EditText) findViewById(R.id.edNome);
    txtResultado = (TextView) findViewById(R.id.txtResultado);
    btnProcessar = (Button) findViewById(R.id.btnProcessar);
    btnProcessar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            txtResultado.setText("Seja bem vindo " + edNome.getText());
        }
    });

    btnTela2 = (Button) findViewById(R.id.btnTela2);
    btnTela2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,Tela2.class);
            startActivity(intent);
        }
    });

}
```

Execute o projeto e selecione o botão Tela 2 para verificar a chamada da segunda tela.

Ciclo de vida de uma Activity

As Activities são armazenadas na memória como se fosse uma pilha de telas, uma em cima da outra, ficando visível para o usuário a que está por cima de todas. Quando uma nova tela é apresentada, essa é colocada no topo da pilha, ficando a anterior logo abaixo.

Cada Activity possui um ciclo de vida específico.

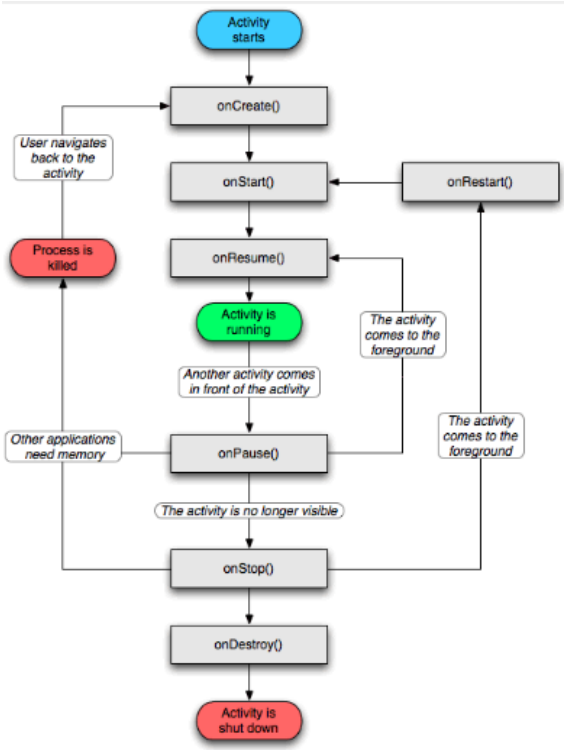


Figura 3 – Ciclo de vida de uma Activity

Resumo explicativo sobre os métodos do ciclo de vida de uma Activity

- onCreate(): Primeiro método a ser executado quando uma Activity é criada. Responsável por carregar os layouts XML e outras operações de inicialização. É executado apenas uma vez.

- onStart(): *Ch*amado imediatamente após a onCreate() , quando a Activity torna-se visível ao usuário. Chamado quando Activity que estava em background volta a ter foco.

- onResume() : Ocorre quando a Activity está no topo da pilha e o usuário já pode interagir com a tela

- onPause(): É o primeiro método a ser invocado quando a Activity perde o foco. Ocorre quando uma nova Activity é iniciada, ou quando está encerrando. Também, utilizado para gravar informações que ainda não foram salvas.

- onStop(): Método chamado quando a Activity fica completamente encoberta por outra Activity. Não está executando e saiu do topo da pilha de activities.

- onDestroy(): O último método a ser chamado. Quando a Activity é considerada finalizada. Pode ser por código, ou quando o sistema operacional precisa liberar recursos. Se o usuário voltar a requisitar essa Activity, um novo objeto será construído.

- onRestart() - Chamado imediatamente antes da onStart(), quando uma Activity volta a ter o foco depois de estar em background. Ocorre quando a Activity é pausada e chamada novamente.

Classe Log

A classe Log, juntamente com o método i(), localizados em library import android.util.Log, é utlizada para criar logs com níveis de informação.

Projeto 3 - Usando as classes Activity e Log

Para visualizar o funcionamento dos métodos citados acima e da classe Log, acrescente um novo objeto Button e nomeei como btnCiclo. Altere o arquivo string.xml para

```
<resources>
    <string name="app_name">Aula1</string>
    <string name="txt_nome">Digite seu nome:</string>
    <string name="btn_processar">Processar</string>
    <string name="txt_resultado">Seja bem vindo!!!!</string>
    <string name="btn_Tela2">Tela2</string>
```

```
    <string name="txt_tela2">Tela 2</string>
    <string name="btn_ciclo">Ciclo de vida de uma Activity</string>
</resources>
```

Na janela Project, do projeto **Aula1**, clique com o botão direito do mouse sobre a pasta **app** e selecione a opção New / Activity / Empty Activity.

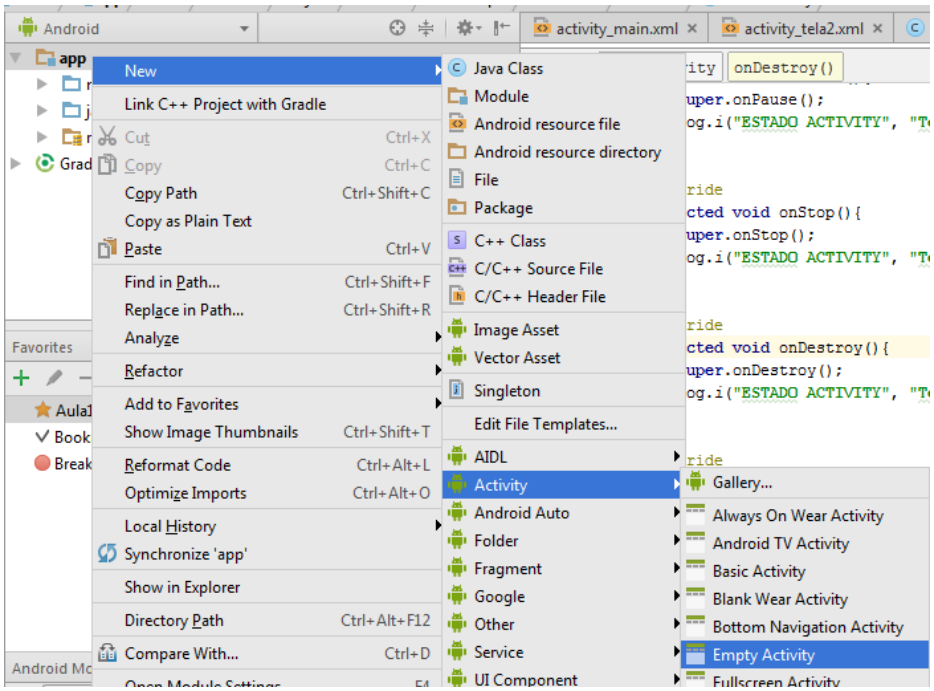


Figura 4 – Criando a segunda tela para testar o ciclo de vida de uma Activity

Nomeie a nova Activity como Tela3.

No arquivo MainActivity.java acrescente o código, como no exemplo abaixo.

```
public class MainActivity extends AppCompatActivity {

    TextView txtNome;
    EditText edNome;
    TextView txtResultado;
    Button btnProcessar;
    Button btnTela2;
    Button btnCiclo;
Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtNome = (TextView) findViewById(R.id.txtNome);
    edNome = (EditText) findViewById(R.id.edNome);
    txtResultado = (TextView) findViewById(R.id.txtResultado);

    btnProcessar = (Button) findViewById(R.id.btnProcessar);
    btnProcessar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            txtResultado.setText("Seja bem vindo " + edNome.getText());
```

```
            String msg = edNome.getText().toString();
            Log.i("msg", msg);

        }
    });

    btnTela2 = (Button) findViewById(R.id.btnTela2);
    btnTela2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,Tela3.class);
            startActivity(intent);
        }
    });


     btnCiclo = (Button) findViewById(R.id.btn);
    btnCiclo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this,Tela3.class);
            startActivity(intent);
        }
    });


    Log.i("ESTADO ACTIVITY", "Tela 1::onCreate");
}
@Override
protected void onStart(){
    super.onStart();
    Log.i("ESTADO ACTIVITY", "Tela 1::onStart");
}

@Override
protected void onResume(){
    super.onResume();
    Log.i("ESTADO ACTIVITY", "Tela 1::onResume");
}

@Override
protected void onPause(){
    super.onPause();
    Log.i("ESTADO ACTIVITY", "Tela 1::onPause");
}

@Override
protected void onStop(){
    super.onStop();
    Log.i("ESTADO ACTIVITY", "Tela 1::onStop");
}

@Override
protected void onDestroy(){
    super.onDestroy();
    Log.i("ESTADO ACTIVITY", "Tela 1::onDestroy");
}

@Override
protected void onRestart(){
    super.onRestart();
    Log.i("ESTADO ACTIVITY", "Tela 1::onRestart");
}


}
```

No arquivo Tela3.java codifique a lógica como o exemplo abaixo.

```
package com.example.simone.aula1;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class Tela3 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tela3);
        Log.i("ESTADO ACTIVITY", "Tela 3::onCreate");

    }

    @Override
    protected void onStart(){
        super.onStart();
        Log.i("ESTADO ACTIVITY", "Tela 3::onStart");
    }

    @Override
    protected void onResume(){
        super.onResume();
        Log.i("ESTADO ACTIVITY", "Tela 3::onResume");
    }

    @Override
    protected void onPause(){
        super.onPause();
        Log.i("ESTADO ACTIVITY", "Tela 3::onPause");
    }

    @Override
    protected void onStop(){
        super.onStop();
        Log.i("ESTADO ACTIVITY", "Tela 3::onStop");
    }

    @Override
    protected void onDestroy(){
        super.onDestroy();
        Log.i("ESTADO ACTIVITY", "Tela 3::onDestroy");
    }

    @Override
    protected void onRestart(){
        super.onRestart();
        Log.i("ESTADO ACTIVITY", "Tela 3::onRestart");
    }
}
```

Na parte inferior do Android Studio selecione a opção Android Monitor e janela logcat.

Execute o projeto Aula1 e veja a ordem dos métodos sendo executados, obedecendo ao ciclo de vida da Activity.

Passar Parâmetros entre Activities

Classe Bundle

A classe Bundle é um mapa de chave e valor, onde chave é uma String e valor tipos primitivos ou objetos empacotáveis (Parcelable) e/ou serializáveis. Funciona como um container para armazenar e transportar dados.

Métodos de Configuração de Propriedades da classe Bundle

putString, putBoolean, putChar, putByteArray, putShort, putInt, putLong, putFloat, putDouble

Métodos importantes na passagem de parâmetro.

putExtras

Recebe o objeto Bundle que será passado como parâmetro para a outra tela

getExtras

Retorna o objeto Bundle criado para enviar os parâmetros

getIntent

Recupera os parâmetros enviados para a segunda tela.

Projeto 4 – Usando a classe Bundle

Para testar a passagem de parâmetros de uma tela para outra, abrir novo projeto Aula2. Na classe MainActivity acrescentar dois TextView, dois EditText e um Button. Nomeie o primeiro TextView como txtNome e o segundo como txtIdade. O primeiro EditText como edNome e o segundo como edIdade. Nomeie o botão como btnTela2. Siga o exemplo abaixo para a codificação.

```
public class MainActivity extends AppCompatActivity {

    TextView txtNome;
    EditText edNome;
    EditText edIdade;
    Button btnTela2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtNome = (TextView) findViewById(R.id.txtNome);
        edNome = (EditText) findViewById(R.id.edNome);
        edIdade = (EditText) findViewById(R.id.edIdade);
        btnTela2 = (Button) findViewById(R.id.btnTela2);
        btnTela2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MainActivity.this,Tela2.class);
                Bundle params = new Bundle();
                String msgNome = edNome.getText().toString();
                String msgIdade = edIdade.getText().toString();

                params.putString("nome",msgNome);
                params.putString("idade",msgIdade);

                intent.putExtras(params);
                startActivity(intent);


            }
        });
    }
}
```

No projeto Aula2, acrescentar uma Empty Activity Tela2.java. Criar dois TextView, nomeados como txtNome e txtIdade. Siga o exemplo abaixo para a codificação da classe Tela2.

```
public class Tela2 extends AppCompatActivity {

    TextView txtNome;
    TextView txtIdade;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tela2);

        Intent intent = getIntent();
        Bundle params = intent.getExtras();
        String msgNome = params.getString("nome");
        String msgIdade = params.getString("idade");

        txtNome = (TextView) findViewById(R.id.txtNome);
        txtIdade = (TextView) findViewById(R.id.txtIdade);

        txtNome.setText(msgNome);
        txtIdade.setText(msgIdade);
        }
}
```

O arquivo string.xml deve ficar da seguinte forma:

```
<resources>
    <string name="app_name">Aula2</string>
    <string name="txt_nome">Digite seu nome:</string>
    <string name="txt_idade">Idade:</string>
    <string name="btn_tela2">Tela 2</string>
</resources>
```

Execute o projeto e selecione o botão Tela 2 para verificar a chamada da segunda tela e a passagem de parâmetros.

Classe CheckBox

Componente que permite selecionar uma ou mais opções de um conjunto.

Métodos:

isChecked(): informa se o checkbox está selecionado(true), ou não(false)

setChecked(): muda o estado selecionado

toggle(): alterna a seleção do checkbox

onCheckedChangeListener(): saber se o checkbox muda o estado do check para o estado inverso

onClickListener(): saber se o check foi selecionado

Classe RadioButton

Componente que permite uma opção de um conjunto, mutuamente exclusivos. Como são exclusivos devem ser agrupados dentro do componente RadioGroup, o que garante que apenas uma opção pode ser selecionada de cada vez.

Métodos:

check(): seleciona o radio especificado pelo id

clearCheck(): tira a seleção

getCheckedRadioButton():obtem o id do RadioButton selecionado no grupo, retornando -1 se não tiver opção selecionada.

onCheckedChangeListener(): saber se o radio muda o estado do check para o estado inverso

onClickListener(): saber se o radio foi selecionado

Projeto 5 – Usando as classe CheckBox e RadioButton

Criar o projeto ExemploCheckBoxRadioButton. Seguir o exemplo abaixo:

Arquivo activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="unicamp.cotuca.br.exemplocheckboxradiobutton.MainActivity">

    <CheckBox
        android:id="@+id/cbNetBeans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="62dp"
        android:text="@string/cb_netbeans"
        android:textSize="18sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:textColor="@android:color/holo_green_dark" />

    <CheckBox
        android:id="@+id/cbEclipse"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="15dp"
        android:text="@string/cb_eclipse"
        android:textColor="@android:color/holo_green_dark"
        android:textSize="18sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cbNetBeans" />

    <TextView
        android:id="@+id/txtFramework"
        android:layout_width="257dp"
        android:layout_height="36dp"
        android:layout_marginLeft="64dp"
        android:layout_marginTop="19dp"
        android:text="@string/txt_framework"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnCheckBox"
        tools:text="@string/txt_framework" />

    <Button
        android:id="@+id/btnRadioButton"
        android:layout_width="260dp"
        android:layout_height="37dp"
        android:layout_marginLeft="62dp"
        android:layout_marginTop="8dp"
        android:text="@string/btn_radiobutton"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/radioGroup" />

    <Button
```

```
        android:id="@+id/btnCheckBox"
        android:layout_width="261dp"
        android:layout_height="47dp"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="10dp"
        android:text="@string/btn_checkbox"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cbEclipse"
        tools:layout_editor_absoluteY="138dp" />

    <TextView
        android:id="@+id/txtLinguagem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/txt_linguagem"
        android:layout_marginTop="62dp"
        app:layout_constraintTop_toBottomOf="@+id/radioGroup"
        android:layout_marginLeft="60dp"
        app:layout_constraintLeft_toLeftOf="parent" />

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="170dp"
        android:layout_height="102dp"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="28dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtFramework">

        <RadioButton
            android:id="@+id/rbCSharp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/rb_csharp"
            android:textSize="18sp" />

        <RadioButton
            android:id="@+id/rbJava"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/rb_java"
            android:textSize="18sp" />
    </RadioGroup>

</android.support.constraint.ConstraintLayout>
```

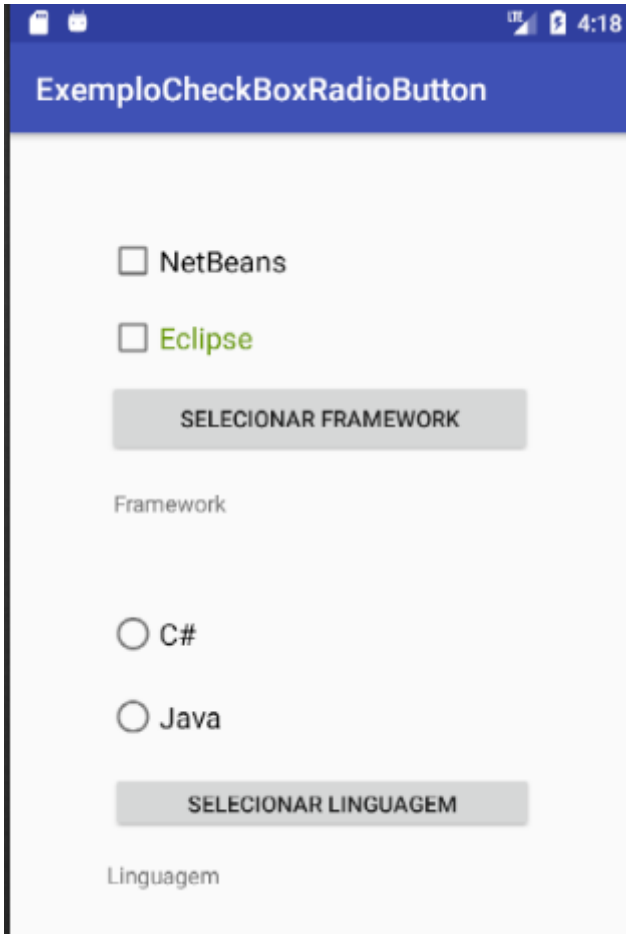
Arquivo string.xml

```
<resources>
    <string name="app_name">ExemploCheckBoxRadioButton</string>
    <string name="cb_netbeans">NetBeans</string>
    <string name="cb_eclipse">Eclipse</string>
    <string name="rb_java">Java</string>
    <string name="rb_csharp">C#</string>
    <string name="btn_checkbox">Selecionar FrameWork</string>
    <string name="btn_radiobutton">Selecionar Linguagem</string>
    <string name="txt_framework">Framework</string>
    <string name="txt_linguagem">Linguagem</string>
</resources>
```

Arquivo MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    CheckBox cbNetBeans;
    CheckBox cbEclipse;
    RadioButton rbJava;
    RadioButton rbCSharp;
    Button btnCheckBox;
    Button btnRadioButton;
    TextView txtFramework;
    TextView txtLinguagem;

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

        cbNetBeans = (CheckBox) findViewById(R.id.cbNetBeans);
        cbEclipse = (CheckBox) findViewById(R.id.cbEclipse);
        rbJava = (RadioButton) findViewById(R.id.rbJava);
        rbCSharp = (RadioButton) findViewById(R.id.rbCSharp);
        txtFramework = (TextView) findViewById(R.id.txtFramework);
        txtLinguagem = (TextView) findViewById(R.id.txtLinguagem);


        btnCheckBox = (Button) findViewById(R.id.btnCheckBox);
        btnCheckBox.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {

                 if (!(cbNetBeans.isChecked() || cbEclipse.isChecked()))
                     txtFramework.setText("Selecione uma opção");
                 else {

                     if (cbNetBeans.isChecked())
                         txtFramework.setText(cbNetBeans.getText());
                     if (cbEclipse.isChecked())
                         txtFramework.setText(cbEclipse.getText());

                     if (cbNetBeans.isChecked() && cbEclipse.isChecked())
                         txtFramework.setText(cbNetBeans.getText() + " e  " +
cbEclipse.getText());
                 }
             }
            }
        );

        btnRadioButton = (Button) findViewById(R.id.btnRadioButton);
        btnRadioButton.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {
                 if(rbJava.isChecked())
                  txtLinguagem.setText(rbJava.getText());
                 if(rbCSharp.isChecked())
                  txtLinguagem.setText(rbCSharp.getText());
             }
        }

        );
    }
}
```

Resultado Final:



Classe ArrayAdapter

Responsável em fazer uma ponte entre os dados a serem exibidos e uma ListView.

- Construtor público da classe:

  ArrayAdapter(Context context, int resource, List<T> objects)

- Argumentos do construtor:
  - context → O contexto atual.
  - resource → layout a ser escolhido para aparecer os itens da lista
  - objects → Os objetos à serem representados na ListView.

Método público da classe ArrayAdapter

getView()

Preenche cada item da lista, um a um.

Definição: View getView(int position, View convertView, ViewGroup parent)

Argumentos:

- int position: posição do item do ListView
- View convertView **:** antiga View para ser re-usada, se possível.
- ViewGroup parente: pai da View atual, que esta eventualmente poderá se associar.

Sobrescrevendo o método getView() para visualização de cada item.

Este método da classe **ArrayAdapter**, tem a responsabilidade de gerar um objeto **View** para cada item da lista de dados que lhe for passada. Para fazer isso ele será executado dentro de uma estrutura de repetição do **ListView** do Android, de modo que cada item da lista passará por ele. Essa estrutura de repetição é feita automaticamente pelo Android.

Dentro desse método, inflar (ação de pegar um layout XML e transforma-lo em um objeto **View** do Android) o layout recém-criado, lista_itens.xml, preenchendo os componentes de interface gráfica que estão nesse layout com as informações de cada carro.

Classe LayOutInflater

Classe abstrata que instancia um arquivo XML de layout em seus objetos View correspondentes.

Métodos Públicos da classe LayOutInflater

from

LayoutInflater from (Context context)

Obtém LayoutInflater a partir do contexto dado.

inflate

View inflate (int resource, ViewGroup root)

Inflar uma nova hierarquia de exibição do recurso xml especificado.

Classe ImageView

Classe pública que exibe recursos de imagem, por exemplo, Bitmap ou recursos Drawable.

Método Públicos da classe ImageView

setImageResource

setImageResource(int resId)

Define um desenho como o conteúdo deste ImageView.

Projeto 6 – Usando as classes ArrayAdapter, LayOutInflater e ImageView

Criar um novo projeto e nomeá-lo como ExemploListView.

Criar a main class como ListaCarrosActivity

Criar a classe Carro.java no mesmo pacote da classe principal.

```
public class Carro {
    private String modelo;
    private int imagem;

    public Carro() {
    }

    public Carro(String modelo, int imagem) {
        super();
        this.modelo = modelo;
        this.imagem = imagem;
    }

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public int getImagem() {
        return imagem;
    }

    public void setImagem(int imagem) {
```

```
        this.imagem = imagem;
    }

    @Override
    public String toString() {
        return modelo;
    }
}
```

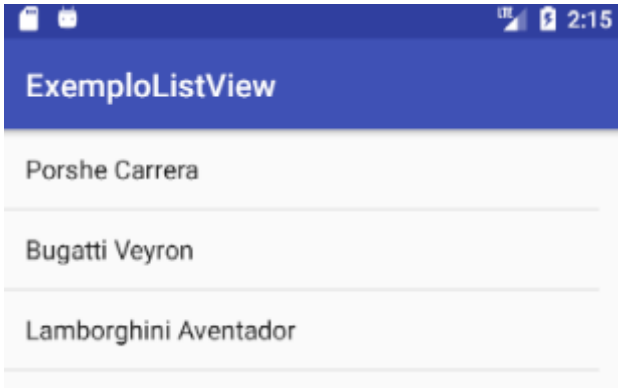
Copiar os arquivos das imagens para a pasta ListView\app\src\main\res**drawable** do projeto. Use ctrl+c e ctrl+v.

No XML activity_lista_carros.xml, excluir o TextView HelloWorld e acrescentar o componente ListView, na pasta Containers. Mudar a propriedade id para lvCarro.

Na classe ListaCarrosActivity.java, programar o código abaixo:

```
public class ListaCarrosActivity extends AppCompatActivity {

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

        ListView listView = (ListView)findViewById(R.id.lvCarro);

        List<Carro> carros = gerarCarros();
        ArrayAdapter<Carro> carrosAdapter = new ArrayAdapter<Carro>(this,
android.R.layout.simple_list_item_1, carros);
        listView.setAdapter(carrosAdapter);

    }

    private List<Carro> gerarCarros() {
        List<Carro> carros = new ArrayList<Carro>();
        carros.add(criarCarro("Porshe Carrera", R.drawable.porsche));
        carros.add(criarCarro("Bugatti Veyron", R.drawable.bugatti));
        carros.add(criarCarro("Lamborghini Aventador", R.drawable.lamborghini));

        return carros;
    }

    private Carro criarCarro(String modelo, int image) {
        Carro carro = new Carro(modelo,image);
        return carro;
    }
}
```

Executar o código e aparecerá a seguinte tela:



Criando a própria classe ArrayAdapter

No exemplo acima, foi usado um *Adapter* da classe . Mas também, poderia ser utilizado um Adapter criado em tempo real.

Para melhorar a lista e acrescentar as imagens para os carros ir até a pasta app\src\main\res\layout e com o botão direito do mouse selecionar a opção Layout resource file. Nomear o arquivo XML em File Name: lista_itens.xml e Root elemento: LinearLayout. Copiar o código XML abaixo.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/black"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="5dp"
        android:background="@android:color/white"
        android:orientation="horizontal" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:orientation="vertical" >

            <ImageView
                android:id="@+id/image_view_carro"
                android:layout_width="120dp"
                android:layout_height="120dp"
                android:adjustViewBounds="true" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:orientation="vertical" >

            <TextView
                android:id="@+id/text_view_modelo_carro"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
                android:textSize="32sp" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

Criar a classe **ListaCarrosAdapter**, extendendo a classe ArrayAdapter do Android.

```
public class ListaCarrosAdapter extends ArrayAdapter<Carro> {

    //Declara os atributos da classe
    private Context context;
    private List<Carro> carros = null;


    public ListaCarrosAdapter(Context context,  List<Carro> carros) {
       /* invoca o constructor da superclasse, durante o constructor da subclasse, passando os
       parâmetros contexto, 0 e carros*/

        super(context,0, carros);
        this.carros = carros;
        this.context = context;
    }

    //Sobrescreve o método getView()
    @Override
    public View getView(int position, View view, ViewGroup parent) {

        //Posição do item em meio aos dados do Adapter
        Carro carro = carros.get(position);

       //Infla o xml lista_itens.xml
       if(view == null)
            view = LayoutInflater.from(context).inflate(R.layout.lista_itens, null);

        //Cria uma view de imagem
        ImageView imageViewCarro = (ImageView) view.findViewById(R.id.image_view_carro);
        imageViewCarro.setImageResource(carro.getImagem());

        //Cria uma view de texto
        TextView textViewModeloCarro = (TextView)
                                    view.findViewById(R.id.text_view_modelo_carro);
        textViewModeloCarro.setText(carro.getModelo());

        //Retorna o objeto view
        return view;
    }
}
```

Após criar a classe ListaCarrosAdapter do **Adapter** customizado, vincula-lo a classe ListaCarrosActivity,

Comentar a linha:

```
ArrayAdapter<Carro> carrosAdapter = new ArrayAdapter<Carro>(this, android.R.layout.simple_list_item_1, carros);
```

e acrescentar o código:

```
final ListaCarrosAdapter carrosAdapter = new ListaCarrosAdapter(this,carros);
```

substituindo a classe ArrayAdapter pela classe ListaCarrosAdapter, implementada

A classe ListaCarrosActivity ficará como o exemplo abaixo.

```
public class ListaCarrosActivity extends AppCompatActivity {


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


        //Cria um objeto do tipo ListView recebendo a view lvCarro já criada na Activity
        ListView listView = (ListView)findViewById(R.id.lvCarro);


        //Cria um objeto do tipo List e adiciona os valores através do método gerarCarros
        List<Carro> carros = gerarCarros();

        //ArrayAdapter<Carro> carrosAdapter = new ArrayAdapter<Carro>(this,
android.R.layout.simple list item 1, carros);

        final ListaCarrosAdapter carrosAdapter = new ListaCarrosAdapter(this,carros);

        listView.setAdapter(carrosAdapter);


    private List<Carro> gerarCarros() {
        List<Carro> carros = new ArrayList<Carro>();
        carros.add(criarCarro("Porsche Carrera", R.drawable.porsche));
        carros.add(criarCarro("Bugatti Veyron", R.drawable.bugatti));
        carros.add(criarCarro("Lamborghini Aventador", R.drawable.lamborghini));

        return carros;
    }

    //Cria o método criarCarro
    private Carro criarCarro(String modelo, int image) {
        Carro carro = new Carro(modelo,image);
        return carro;
    }
}
```

Ao executar novamente o código, a seguinte tela será exibida:


3:06
ExemploListView
Porshe
Carrera
Bugatti
Veyron
Lamborghini
Aventador

Implementar a seleção do item da lista

Criar três novas Activity Empty: PorshActivity, BugattiActivity e LamborghiniActivity.

Na PorshActivity acrescentar um TextView e colocar o texto Porsh, textSize 30sp. Fazer o mesmo para as outras Activities criadas.

O arquivo string.xml ficará com o seguinte conteúdo:

```
<resources>
    <string name="app_name">ExemploListView</string>
    <string name="txt_porsch">Porsch</string>
    <string name="txt_lamborghini">Lamborghini</string>
    <string name="txt_bugatti">Bugatti</string>
</resources>
```

No método onCreate do arquivo ListaCarrosActivity.java usar o método setOnItemClickListener anônimo para selecionar um item da lista. Criar um switch para chamar os métodos de cada carro.

Criar os métodos goPorsch(), goBugatti(), goLamborghini para chamar as respectivas Activities.

```
public class ListaCarrosActivity extends AppCompatActivity {


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


        ListView listView = (ListView)findViewById(R.id.lvCarro);
        List<Carro> carros = gerarCarros();

        //ArrayAdapter<Carro> carrosAdapter = new ArrayAdapter<Carro>(this,
android.R.layout.simple_list_item_1, carros);

        final ListaCarrosAdapter carrosAdapter = new
ListaCarrosAdapter(this,carros);
        listView.setAdapter(carrosAdapter);

        //Clicar na listView e executar uma ação
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i,
long l) {
                switch (i){
                    case 0: goPorsch();
                        break;
                    case 1: goBugatti();
                        break;
                    case 2: goLamborghini();
                        break;
                    case 3: finish();
                        break;
                }
```

```
                //Intent intent = new Intent(ListaCarrosActivity.this,PorschActivity.class);

                startActivity(intent);
            }

        });
    }

    private List<Carro> gerarCarros() {
        List<Carro> carros = new ArrayList<Carro>();
        carros.add(criarCarro("Porsche Carrera", R.drawable.porsche));
        carros.add(criarCarro("Bugatti Veyron", R.drawable.bugatti));
        carros.add(criarCarro("Lamborghini Aventador", R.drawable.lamborghini));

        return carros;
    }

    private Carro criarCarro(String modelo, int image) {
        Carro carro = new Carro(modelo,image);
        return carro;
    }


    private void goPorsch() {
        Intent intent = new Intent(ListaCarrosActivity.this,PorschActivity.class);
        startActivity(intent);
    }

    private void goBugatti() {
Intent intent = new Intent(ListaCarrosActivity.this,BugattiActivity.class);
        startActivity(intent);
    }

    private void goLamborghini() {
        Intent intent = new
Intent(ListaCarrosActivity.this,LamborghiniActivity.class);
        startActivity(intent);
    }
}
```

SQLite

O SQLite é um banco de dados simples e poderoso, formatado para ser executado na plataforma móvel. É formado por um conjunto de bibliotecas escritas em C, podendo ser integrado a programas escritos em diferentes linguagens com o intuito de possibilitar a manipulação de dados através de instruções SQL. A diferença para os outros bancos é que tudo isso pode ser feito sem que seja preciso acessar um SGBD (Sistema de Gerenciamento de Banco de Dados), ou seja, todas as instruções podem ser executadas diretamente via código fonte, como criação do banco e de tabelas, sendo por este motivo adotado como o banco de dados nativo da plataforma Android.

Na prática, o SQLite é um mini-SGBD, capaz de criar um arquivo, ler e escrever diretamente nele. O SQLite já está disponível na plataforma Android, não necessitando de instalação, configuração ou administração, suporta commit e rollback, além de ser um banco de dados gratuito.

Além disso, o Android oferece suporte completo ao banco, através de uma API com um rico conjunto de classes e métodos que abstraem as complexidades dos códigos SQL. Assim, não precisamos montar a cláusula SQL inteira para atualizar uma linha na tabela, ou ainda, para fazer uma pesquisa na mesma. O Android nos fornece um método, onde passando alguns parâmetros obtemos um apontador para os dados retornados, podendo navegar pelo resultado como se estivéssemos escolhendo uma folha em um arquivo.

Principais classes e métodos para usar o SQLite

SQLiteOpenHelper

Classe auxiliar para gerenciar a criação de banco de dados e o gerenciamento de versão. Como a classe SQLiteOpenHelper é uma classe abstrata, devemos herdá-la para poder usar todos os recursos que ela disponibiliza.

Para o construtor, devemos informar o Context, o nome do banco de dados e a versão.

Exemplo:

```
public class DbHelper extends SQLiteOpenHelper {
    public DbHelper(Context context) {
        super(context, "vendas.db", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        // Implementar a criação das tabelas
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Implementar a atualização do banco baseado nas versões (oldVersion e newVersion)
    }
}
```

A inicialização do DbHelper se resume a um simples new.

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        DbHelper dbHelper = new DbHelper(this);

    }

}
```

Meios de acesso

Existem duas formas de acessar nosso banco de dados: obter uma instância de leitura ou uma instância de escrita. Conseguimos isso através dos métodos getReadableDatabase() e getWritableDatabase() respectivamente.

Usando o getReadableDatabase(), obtemos uma instâncida de SQLiteDatabase para realizarmos operações de leitura. Já o getWritableDatabase(), obtemos uma instância de SQLiteDatabase para operações de escrita.

Implementando o método onCreate()

Ao obtermos uma instância de SQLiteDatabase, o SQLiteOpenHelper cuida de criar o banco caso este não exista. Assim o método onCreate() é executado para criar as tabelas do banco.

Sobrescrevendo o método onCreate(), recebemos uma instância de SQLiteDatabase por parâmetro para executarmos as ações desejadas. Lembrando que esta instância de SQLiteDatabase é uma instância que permite escrita.

No caso iremos usar apenas o execSQL(), que nos permite executar um comando SQL passado por parâmetro.

Como no nome do meu banco de dados deste exemplo é vendas.db, irei criar uma tabela chamada clientes.

```
@Override

public void onCreate(SQLiteDatabase db) {

    db.execSQL("CREATE TABLE clientes (" +

            "_ID INT PRIMARY KEY," +

            "nome TEXT NOT NULL," +

            "cpf TEXT NOT NULL);");

}
```

Então quando o banco de dados for criado pela primeira vez teremos nossa tabela clientes criada também.

Inserindo dados

Como já citado anteriormente, usamos o método getWritableDatabase() para executarmos operações de escrita no banco.

```
DbHelper dbHelper = new DbHelper(this);

SQLiteDatabase db = dbHelper.getWritableDatabase();
```

O método adequado para inserção de dados é o insert() da classe SQLiteDatabase.

Duas informações são necessárias para o método insert(): a tabela e os valores das colunas. A tabela é uma simples String e os valores são passados através de um ContentValues.

A classe ContentValues armazena os dados no formato chave-valor, que seria equivalente a coluna-valor.

```
DbHelper dbHelper = new DbHelper(this);

SQLiteDatabase db = dbHelper.getWritableDatabase();

ContentValues novoCliente = new ContentValues();

novoCliente.put("nome", "João Vitor");

novoCliente.put("cpf", "111.222.333-44");

db.insert("clientes", null, novoCliente);
```

O método insert() retorna um long que representa o id do novo registro, ou -1 caso ocorra um problema durante a inserção.

Atualizando dados

O método update() é bem parecido com o método insert(), a diferença é que podemos definir a condição de atualização, através de dois parâmetros: whereClause e whereArgs.

Para o whereClause definimos os campos que fazem parte da condição, como o whereClause é uma String e não um array, devemos definir os campos da seguinte maneira: “campo1 = ? AND campo2 = ?”. Exatamente como uma cláusula WHERE do SQL, porém ao invés de informar os valores, informamos interrogações.

Estas interrogações serão substituidas pelos valores do parâmetro whereArgs.

```
DbHelper dbHelper = new DbHelper(this);

SQLiteDatabase db = dbHelper.getWritableDatabase();

ContentValues novosValores = new ContentValues();
```

```
novoCliente.put("nome", "João");

db.update("clientes", novosValores, "cpf = ?", new String[]{"111.222.333-44"});
```

Esta técnica previne possíveis ataques de SQL Injection.

Removendo dados

Parecido com os métodos anteriores, no método delete() devemos informar apenas o nome da tabela, a condição e seus valores.

```
DbHelper dbHelper = new DbHelper(this);

SQLiteDatabase db = dbHelper.getWritableDatabase();

db.delete("clientes", "nome = ?", new String[]{"João"});
```

Consultando dados

O método query() é o mais indicado para recuperar dados do banco.

Com ele podemos informar a tabela, as colunas, a condição (WHERE), agrupamento (GROUP BY), condição pós-agrupamento (HAVING), ordenação (ORDER BY) e até definir limites (LIMIT).

Apesar de ter vários parâmetros, o mínimo necessário para o método query() funcionar, é a tabela e as colunas, as outras informações são opcionais e podemos informar null.

O retorno deste método é uma instância da classe Cursor. Esta classe representa o resultado de uma consulta.

```
DbHelper dbHelper = new DbHelper(this);

SQLiteDatabase db = dbHelper.getReadableDatabase();

String[] colunas = new String[]{"nome", "cpf"};

Cursor cursor = db.query("clientes", colunas, null, null, null, null, "nome")
```

A classe Cursor disponibiliza métodos de navegação como moveToFirst(), moveToLast(), moveToNext() e moveToPrevious(). Todos estes métodos retornam um Boolean, assim podemos saber se houve sucesso ou não.

Também podemos usar métodos auxiliares como getCount(), getPosition(), isAfterLast(), isBeforeFirst(), isFirst() e isLast().

Para acessar um dado da consulta, usamos o índice (iniciado em zero) da coluna. Um facilitador é o getColumnIndex() onde este método retorna o índice do nome da coluna passada por parâmetro.

```
DbHelper dbHelper = new DbHelper(this);
SQLiteDatabase db = dbHelper.getReadableDatabase();
String[] colunas = new String[]{"nome", "cpf"};
Cursor cursor = db.query("clientes", colunas, null, null, null, null, "nome");
if (cursor.moveToFirst()) {
    final int INDICE_NOME = 0;
    final int INDICE_CPF = 1;
    String nomeCliente;
    String cpfCliente;
    do {
        nomeCliente = cursor.getString(INDICE_NOME);
        cpfCliente = cursor.getString(INDICE_CPF);
         // Aqui você já em os valores desejados e pode executar
        // qualquer coisa que precisar com os valores
    } while (cursor.moveToNext());
}
```

Projeto 7 – CRUD no SQLite

No Android Studio criar o projeto CRUDSQLite no pacote br.unicamp.cotuca. Criar a MainActivbity.java e activity_main.xml normalmente.

No arquivo activity_main.xml criar a tela principal contendo dois TextView, dois PlainText, cinco botões e um ListView. A tela ficara da seguinte forma:

Text

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="br.unicamp.cotuca.crudsqlite.MainActivity">

    <TextView
        android:id="@+id/tvNome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
```

```
        android:layout_marginTop="36dp"
        android:text="Nome:"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvIdade"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="80dp"
        android:text="Idade:"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/edtNome"
        android:layout_width="195dp"
        android:layout_height="42dp"
        android:layout_marginStart="80dp"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:textColor="@android:color/black"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnAdicionar"
        android:layout_width="91dp"
        android:layout_height="36dp"
        android:layout_marginStart="40dp"
        android:layout_marginTop="152dp"
        android:text="Adicionar"
        android:textSize="14sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnEditar"
        android:layout_width="76dp"
        android:layout_height="38dp"
        android:layout_marginStart="152dp"
        android:layout_marginTop="152dp"
        android:text="Editar"
        android:textSize="14sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnRemoverPessoa"
        android:layout_width="86dp"
        android:layout_height="40dp"
        android:layout_marginStart="248dp"
        android:layout_marginTop="152dp"
        android:text="Remover"
        android:textSize="14sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <Button
        android:id="@+id/btnRemoverTabela"
        android:layout_width="152dp"
        android:layout_height="41dp"
        android:layout_marginEnd="20dp"
        android:layout_marginTop="204dp"
        android:text="Remover Tabela"
        android:textSize="14sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnVerRegistros"
        android:layout_width="167dp"
        android:layout_height="39dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="204dp"
        android:text="Ver Registros"
        android:textSize="14sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/edtIdade"
        android:layout_width="109dp"
        android:layout_height="44dp"
        android:layout_marginStart="64dp"
        android:layout_marginTop="64dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintStart_toStartOf="@+id/tvIdade"
        app:layout_constraintTop_toTopOf="parent" />

    <ListView
        android:id="@+id/lvPessoa"
        android:layout_width="336dp"
        android:layout_height="234dp"
        android:layout_marginEnd="8dp"
        android:layout_marginTop="260dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.75"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Design

Reservar a classe MainActivity.java para programação posterior.

No pacote br.unicamp.cotuca, criar o subpacote CRUD, onde serão criadas todas as classes do CRUD.

No pacote CRUD criar a classe Pessoa.java

```
package br.unicamp.cotuca.crudsqlite.CRUD;

/**
 * Created by simone on 30/08/2018.
 */
public class Pessoa {

    private String nome;
    private int idade;

    public Pessoa(){
    }

    public String getNome() {
        return nome;
    }

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

```
    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }
}
```

No pacote CRUD criar a classe Create.java que extenderá a classe SQLiteOpenHelper para criação do banco, versõa, tabela , path e contexto.

```
package br.unicamp.cotuca.crudsqlite.CRUD;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by simone on 30/08/2018.
 */

public class Create extends SQLiteOpenHelper {

    private static final String NOME_DB = "MEU_DB";
    private static final int VERSAO_DB = 1;
    private static final String TABELA_PESSOA = "TABELA_PESSOA";
    private static final String PATH_DB =
"/data/user/0/br.unicamp.cotuca.crudsqlite/databases/MEU_DB";
    private Context mContext;
    private SQLiteDatabase db; //instância de banco de dados

    //O null indica que usaremos um cursor padrão
    public Create(Context context) {
        super(context, NOME_DB, null,VERSAO_DB);
        this.mContext = context;
        db = getWritableDatabase(); //o método indica que será escrito no
db

    }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

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

    }

    public boolean createTable(){
        openDB();
        String createTable = "CREATE TABLE IF NOT EXISTS " + TABELA_PESSOA
+ "("
                                + "NOME TEXT, "
                                + "IDADE INTEGER)";
```

```
        try{
            db.execSQL(createTable);
            return true;
        } catch(Exception e){
            e.printStackTrace();
            return false;
        }
        finally {
            db.close(); //fecha o banco independente de ter conseguido
executar ou não
        }
    }

    private void openDB(){
        if(!db.isOpen()){

            db =
mContext.openOrCreateDatabase(PATH_DB,SQLiteDatabase.OPEN_READWRITE,null);
        }
    }
}
```

Criar a classe Update.java extendendo a classe SQLiteOpenHelper. A classe Updade será responsável por inserir e editar os registros inseridos na tabela Pessoa.

```
package br.unicamp.cotuca.crudsqlite.CRUD;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by simone on 30/08/2018.
 */

public class Update extends SQLiteOpenHelper {

    private static final String NOME_DB = "MEU_DB";
    private static final int VERSAO_DB = 1;
    private static final String TABELA_PESSOA = "TABELA_PESSOA";
    private static final String PATH_DB =
"/data/user/0/br.unicamp.cotuca.crudsqlite/databases/MEU_DB";
    private Context mContext;
    private SQLiteDatabase db; //instância de banco de dados


    //O null indica que usaremos um cursor padrão
    public Update(Context context) {
        super(context, NOME_DB, null,VERSAO_DB);
        this.mContext = context;
        db = getWritableDatabase(); //o método indica que será escrito no
db
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }
```

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

    public boolean insertPessoa(Pessoa p){
        openDB();
        try {
            ContentValues cv = new ContentValues();
            cv.put("NOME",p.getNome());
            cv.put("IDADE",p.getIdade());
            db.insert(TABELA_PESSOA,null,cv);
            return true;
        }
        catch (Exception e){
            e.printStackTrace();
            return false;
        }
        finally {
            db.close();
        }
    }

    public boolean updatePessoa(Pessoa p){
        openDB();
        try {
            String where = "NOME= '" + p.getNome() + "'";
            ContentValues cv = new ContentValues();
            cv.put("IDADE",p.getIdade());
            db.update(TABELA_PESSOA, cv, where, null);
            return true;
        }
        catch (Exception e){
            e.printStackTrace();
            return false;
        }
        finally {
            db.close();
        }
    }

    private void openDB(){
        if(!db.isOpen()){

            db =
mContext.openOrCreateDatabase(PATH_DB,SQLiteDatabase.OPEN_READWRITE,null);
        }
    }

}
```

Criar a classe Read.java extendendo a classe SQLiteOpenHelper. A classe Read será responsável pelas consultas ao banco.

```
package br.unicamp.cotuca.crudsqlite.CRUD;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
```

```
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

/**
 * Created by simone on 30/08/2018.
 */

public class Read extends SQLiteOpenHelper {

    private static final String NOME_DB = "MEU_DB";
    private static final int VERSAO_DB = 1;
    private static final String TABELA_PESSOA = "TABELA_PESSOA";
    private static final String PATH_DB =
"/data/user/0/br.unicamp.cotuca.crudsqlite/databases/MEU_DB";
    private Context mContext;
    private SQLiteDatabase db; //instância de banco de dados

    //O null indica que usaremos um cursor padrão
    public Read(Context context) {
        super(context, NOME_DB, null,VERSAO_DB);
        this.mContext = context;
        db = getReadableDatabase(); //o método indica que será lido do db
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

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

    public ArrayList<Pessoa> getPessoa() {
        openDB();
        ArrayList<Pessoa> pArray = new ArrayList<>();
        String getPessoas = "select * from " + TABELA_PESSOA;

        try{
            Cursor c = db.rawQuery(getPessoas,null);
            if(c.moveToFirst()){
                do{
                    Pessoa p = new Pessoa();
                    p.setNome(c.getString(0));
                    p.setIdade(c.getInt(1));
                    pArray.add(p);
                } while(c.moveToNext());
                c.close();
            }
        }
        catch (Exception e){
            e.printStackTrace();
            return null;
        }
        finally {
            db.close();
        }
        return pArray;
    }
```

```
    private void openDB(){
        if(!db.isOpen()){

            db =
mContext.openOrCreateDatabase(PATH_DB,SQLiteDatabase.OPEN_READWRITE,null);
        }
    }

}
```

Criar a classe Delete.java extendendo a classe SQLiteOpenHelper. A classe Dedlete será responsável exclusão de registros do banco e drop da tabela Pessoa.

```
package br.unicamp.cotuca.crudsqlite.CRUD;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by simone on 30/08/2018.
 */

public class Delete extends SQLiteOpenHelper {

    private static final String NOME_DB = "MEU_DB";
    private static final int VERSAO_DB = 1;
    private static final String TABELA_PESSOA = "TABELA_PESSOA";
    private static final String PATH_DB =
"/data/user/0/br.unicamp.cotuca.crudsqlite/databases/MEU_DB";
    private Context mContext;
    private SQLiteDatabase db; //instância de banco de dados

    //O null indica que usaremos um cursor padrão
    public Delete(Context context) {
        super(context, NOME_DB, null,VERSAO_DB);
        this.mContext = context;
        db = getWritableDatabase(); //o método indica que será escrito no
db
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

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


    public boolean deleteTable(){
        openDB();
        String delTable = "DROP TABLE IF EXISTS " + TABELA_PESSOA;
        try{
            db.execSQL(delTable);
            return true;

        }catch (Exception e){
            e.printStackTrace();
```

```
            return false;         }
        finally {
            db.close();
        }
    }

    public boolean deletePessoa(Pessoa p)
    {
        openDB();

        String where = "NOME= '" + p.getNome() + "'";

        try{
            db.delete(TABELA_PESSOA, where, null);
            return true;

        }catch (Exception e){
            e.printStackTrace();
            return false;         }
        finally {
            db.close();
        }
    }

    private void openDB(){
        if(!db.isOpen()){
            db =
mContext.openOrCreateDatabase(PATH_DB,SQLiteDatabase.OPEN_READWRITE,null);
        }
    }
}
```

Agora, com todas as classes do pacote CRUD prontas, vamos programar a classe MainActivity.java para implementar os botões e chamar os métodos necessários.

```
package br.unicamp.cotuca.crudsqlite;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Switch;
import android.widget.Toast;

import java.util.ArrayList;

import br.unicamp.cotuca.crudsqlite.CRUD.Create;
import br.unicamp.cotuca.crudsqlite.CRUD.Delete;
import br.unicamp.cotuca.crudsqlite.CRUD.Pessoa;
import br.unicamp.cotuca.crudsqlite.CRUD.Read;
import br.unicamp.cotuca.crudsqlite.CRUD.Update;

public class MainActivity extends AppCompatActivity {

Button btnAdicionar,btnEditar,btnRemover,btnVerRegistros,btnRemoverTabela;
```

```
EditText edtNome,edtIdade;
ListView lvPessoa;

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

        edtNome = (EditText) findViewById(R.id.edtNome);
        edtIdade = (EditText) findViewById(R.id.edtIdade);

        btnAdicionar = (Button)findViewById(R.id.btnAdicionar);
        btnEditar = (Button)findViewById(R.id.btnEditar);
        btnRemover = (Button)findViewById(R.id.btnRemoverPessoa);
        btnVerRegistros = (Button)findViewById(R.id.btnVerRegistros);
        btnRemoverTabela = (Button)findViewById(R.id.btnRemoverTabela);

        Create c = new Create(getApplicationContext());
        c.createTable();

        lvPessoa = (ListView)findViewById(R.id.lvPessoa);

        btnAdicionar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Pessoa p = new Pessoa();
                p.setNome(edtNome.getText().toString());

p.setIdade(Integer.parseInt(edtIdade.getText().toString()));

                Update u = new Update(getApplicationContext());
                if(u.insertPessoa(p)){
                    Toast.makeText(getBaseContext(), "Pessoa foi inserida
com sucesso!", Toast.LENGTH_SHORT).show();
                }
                else{
                    Toast.makeText(getBaseContext(), "Erro ao inserir
pessoa!", Toast.LENGTH_SHORT).show();
                }
            }
        });

        btnEditar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Pessoa p = new Pessoa();
                p.setNome(edtNome.getText().toString());

p.setIdade(Integer.parseInt(edtIdade.getText().toString()));

                Update u = new Update(getApplicationContext());
                if(u.updatePessoa(p)){
                    Toast.makeText(getBaseContext(), "Pessoa foi atualizada
com sucesso!", Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(getBaseContext(), "Erro ao atualizar
pessoa!", Toast.LENGTH_SHORT).show();
                }
            }
```

```
        });

        btnRemover.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Pessoa p = new Pessoa();
                p.setNome(edtNome.getText().toString());

                Delete d = new Delete(getApplicationContext());
                if(d.deletePessoa(p)){
                    Toast.makeText(getBaseContext(), "Pessoa foi removida
com sucesso!", Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(getBaseContext(), "Erro ao remover
pessoa!", Toast.LENGTH_SHORT).show();
                }
            }
        });

        btnVerRegistros.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Read r = new Read(getApplicationContext());
                ArrayList<Pessoa> pArray  = r.getPessoa();
                ArrayList list = new ArrayList();

                for (int i = 0; i < pArray.size(); i++) {
                    Pessoa p = pArray.get(i);
                    list.add("Nome: " + p.getNome() +  "\n" + "Idade: " +
p.getIdade());

                    System.out.println("NOME: " + p.getNome() + "IDADE: " +
p.getIdade());
                }
                ArrayAdapter<Pessoa> pessoaAdapter = new
ArrayAdapter<Pessoa>(getBaseContext(), android.R.layout.simple_list_item_1,
list);
                lvPessoa.setAdapter(pessoaAdapter);

            }
        });

        btnRemoverTabela.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Delete d = new Delete(getApplicationContext());
                if(d.deleteTable()){
                    Toast.makeText(getBaseContext(), "Tabela removida com
sucesso!", Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(getBaseContext(), "Erro ao remover
tabela!", Toast.LENGTH_SHORT).show();
                }
            }
        });

    }
}
```

Ao final da programa da classe MainActivity.java, testar o aplicativo no emulador do Android Studio.

Como funciona a Internet

Quando a **Internet** é acessada do computador e/ou celular, existe uma troca de dados entre os dispositivos. Um dado é solicitado e recebe uma resposta, utilizando a informação para alguma coisa.

Quando uma página é visitada, como o **google.com.br**, o navegador manda uma **requisição HTTP** para o endereço digitado, os computadores do **Google** processam a requisição e mandam uma resposta de volta.

Depois, o navegador interpreta a resposta e atualiza a tela mostrando a página do **Google** já conhecida.

O mesmo acontece quando se usa um celular, seja o aplicativo de navegação acessando uma página na **Internet** ou outro aplicativo utilizando uma **API**.

Os aplicativos podem coletar dados de uma fonte remota na Internet utilizando as **requisições HTTP**.

Entendendo as Requisições HTTP

As **requisições HTTP** de um **WebService** servem para incluir informações sobre os dados requisitados de um determinado computador na rede/internet.

Na requisição inclui-se um localizador de recurso, ou URL, que é o endereço dos dados.

A requisição é empacotada em um **“envelope”** e enviada do remetente (seu dispositivo, cliente) até o destinatário (o servidor) pela rede.

Para fazer o envio desse **“envelope”**, é necessário abrir uma conexão de rede entre o remetente e o destinatário utilizando o protocolo **HTTP**, por exemplo. Assim, podem-se trocar mensagens entre os computadores.

Os remetentes, ou clientes, enviam uma requisição para o servidor, quando o servidor recebe a requisição ele envia uma resposta podendo ser uma página HTML, JSON ou XML. Depois de receber a resposta, o cliente processa e exibe os dados.

O HyperText Transfer Protocol (HTTP)

O HTTP é a principal tecnologia por trás da troca de dados através da Internet. HTTP siginifica **Protocolo de Transferência de Hipertexto** e existe desde o começo da Internet quando os computadores transferiam apenas dados básicos.

O que é um protocolo?

Um protocolo são regras pré-definidas que permitem a interação e comunicação entre computadores para completar uma ação ou tarefa. Neste caso, a tarefa é trocar mensagens e dados através da web.

Essa comunicação funciona da seguinte forma, um computador cliente utilizando o HTTP, transmite uma mensagem que contêm métodos HTTP, ou verbos, que expressam intenções ou requisições para um servidor.

Veja alguns dos métodos HTTP ou verbos mais utilizados.

- **Get**: recupera dados do servidor.
- **Post**: cria um novo dado no servidor. Envio de dados.
- **Put**: atualiza dados existentes no servidor. Coloca os dados na URL especificada.
- **Delete**: exclui dados no servidor.

Threads

Quando uma aplicação é iniciada o sistema cria para aquela aplicação uma thread de execução chamada de thread principal ou thread “main”. Ela é muito importante, pois é encarregada de gerenciar os eventos de interface. Por ser a thread que interage com os componentes de interface também é conhecida como UI Thread ou Thread de Interface.

Todo processo possui apenas usam thread; todos os componentes que rodam em um mesmo processo são instanciados na mesma UI Thread e as chamadas que o sistema faz, a cada componente, são despachadas a partir dessa thread.

Quando a aplicação realiza um trabalho intensivo, em resposta à interação do usuário, um thread simples pode começar a responder lentamente. Especificamente, se tudo isso está rodando na UI Thread, realizar tarefas como acesso à rede, ou banco de dados, pode bloquear toda a interface de aplicação enquanto essa tarefa estiver executando. Quando a thread está bloqueada, nenhum evento pode ser iniciado. Do ponto de vista do usuário, o aplicativo parece travar. Se a aplicação demorar alguns segundos para responder, o sistema apresentará para o usuário uma caixa de diálogo com a mensagem “application not responding”.

Além disso, A UI thread é a único thread que pode modificar a interface gráfica e a principal thread de sua aplicação.
Se todo o fluxo de processamento for colocado no thread principal, em algum momento a aplicação vai se tornar lenta ou travar. O exemplo demonstrará como realizar um processamento mais oneroso (no caso um contador) para a aplicação em um thread. O resultado desse processamento será exibido na tela a cada ciclo, através da transferência de responsabilidade do thread que fez esse processamento, para a UI thread, a qual é o único thread que pode atualizar a interface gráfica.

Classe AsyncTask

Ao trabalhar com tarefas que demandam um maior tempo a serem executadas, como, por exemplo, acessar a rede, é importante que essa tarefa seja feita fora do thread principal.

A solução para esse problema é usar a classe AsyncTask que facilita o processamento em background e atualiza os dados da interface.

A AsyncTask encapsula todo esse processo de criação de Threads. Para utilizá-la deve ser criada e estendida à classe AsyncTask. E por ser genérica, ela é definida da seguinte forma:

AsyncTask<Parâmetros,Progresso,Resultado>

- Os parâmetros são o tipo de dados de qualquer valor que serão passados para os métodos da classe;
- Progresso é o tipo de dado (integer) que mostrará o progresso na tela, como a porcentagem de download;
- O Resultado é o retorno da Thread de processamento para a Thread principal que será trabalhado para ser exibido na interface do usuário.

Existem quatro métodos que podem ser sobrescritos na **AsyncTask**:

- O método onPreExecute é executado sempre antes da Thread ser iniciada, e é onde são colocadas as mensagens a serem exibidas na tela para o usuário. Este não é obrigatório e executa na mesma Thread que a interface gráfica;
- O método obrigatório doInBackground é o responsável por todo o processamento pesado, pois ele é executado em uma Thread separada. Enquanto é processada a informação, a tela do usuário não ficará travada. Seu retorno é passado por parâmetro para o método onPostExecute, ou onProgressUpdate;
- O método onPostExecute é o que recebe o retorno do doInBackground e é chamado utilizando um Handler. Sua execução se dá na mesma Thread que a interface gráfica;
- O método onProgressUpdate é o responsável por receber as informações retornas do método doOnBackGround. Também é executado na mesma Thread que a interface gráfica.

Estrutura AsyncTask

```
public class SuaClasse extends AsyncTask<String,Integer,Integer>{
        @Override
        protected void onPreExecute(){
                //Codigo
        }
        @Override
        protected Integer doInBackground(String... params) {
```

```
            //Codigo
        }
        @Override
        protected void onPostExecute(Integer numero){
            //Codigo
        }
        protected void onProgressUpdate(Integer… params){
            //Codigo
        }
}
```

O exemplo demonstrará como realizar um processamento mais oneroso (no caso um contador) para a aplicação em um thread, através da transferência de responsabilidade do thread que fez esse processamento,

O resultado desse processamento será exibido na tela a cada ciclo para a UI thread, a qual é a único thread que pode atualizar a interface gráfica.

Projeto 8 - Testando a classe AsyncTask

Nomear o projeto como TesteAsync. Pacote br.cotuca.unicamp.

No layout da Activity principal, acrescentar um Button e um TextView;

Com o projeto criado, algumas alterações serão realizada para o funcionamento da aplicação:

- Alteração do arquivo strings.xml no diretório res/values;
- Alterar o arquivo ActivityMain.xml no diretório res/layout;
- Alterar a classe responsável por manipular o arquivo ActivityMain.xml, o qual se encontra no package que foi criado no momento da criação do projeto;
- Criar a classe que vai fazer o processamento e depois chamar a UI thread.

Primeiro, alterar a codificação do arquivo strings.xml para ele ficar igual ao seguinte.

Exemplo de como o arquivo strings.xml vai ficar

```
<resources>
  <string name="app_name">TesteAsync</string>
```

```
    <string name="start">Processar..</string>
  </resources>
```

Exemplo de como o arquivo activity_main.xml vai ficar

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="unicamp.cotuca.br.testeasync.MainActivity">

    <Button
        android:id="@+id/bt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="53dp"
        android:layout_marginTop="78dp"
        android:text="@string/start"
        android:textSize="24sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="53dp"
        android:layout_marginTop="68dp"
        android:text="TextView"
        android:textSize="24sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/bt" />
</android.support.constraint.ConstraintLayout>
```

O código xml acima cuida da disposição dos elementos na tela, bem como o layout e os elementos que vão dentro do layout, que no caso são um botão e um TextView. O botão referencia a tag string com o name start através da propriedade android:text.

Exemplo de como o arquivo MainActivity.java deve estar inicialmente

```
package unicamp.cotuca.br.testeasync;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {

    private TextView text;
    private Button bt;

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

        text = (TextView) findViewById(R.id.text);

        bt = (Button) findViewById(R.id.bt);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                processamento();
            }
        });
    }

}
```

No caso acima, o processo acontece da seguinte forma: quando a activity é iniciada, o método **onCreate** é chamado, esse por sua vez carrega a tela através do método setContentView, o qual recebe uma referência ao arquivo activity_main.xml (no caso é o arquivo que responsável pelo elementos da tela). Depois o botão e a textview são recuperadas através de uma chamada a **findViewById** para que esses itens possam ser manipulados. O objetivo é usar essas referências no método processamento, o qual será invocado quando o botão for clicado, mas como este método ainda está vazio, isso será feito mais a frente.

Agora só falta criar a classe Num, ela vai ser criada no pacote onde foi criada a classe que comanda a Activity principal.

```
package unicamp.cotuca.br.testeasync;

import android.os.AsyncTask;
import android.widget.Button;
import android.widget.TextView;

/**
 * Created by simone on 14/09/2017.
 */


//AsyncTask<Parâmetros,Progresso,Resultado>
public class Num extends AsyncTask<Integer, Integer, Void> {
```

```
private TextView text;
private Button bt;


//Construtor
public Num(TextView text, Button bt) {
        this.text = text;
        this.bt = bt;
}

    //Processamento:
    //Este método é executado em uma thread a parte,
    //ou seja ele não pode atualizar a interface gráfica,
    //por isso ele chama o método onProgressUpdate,
    // o qual é executado pela
    //UI thread.

    @Override
    protected Void doInBackground(Integer... params) {
        int n = params[0];
        int i = 0;
        while(i < n){
            try{
                Thread.sleep(1000);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            //Notifica o Android de que ele precisa
            //fazer atualizações na
            //tela e chama o método onProgressUpdate
            //para fazer a atualização da interface gráfica
            //passando o valor do
            //contador como parâmetro.
            publishProgress(i);
            i++;
        }
        return null;
    }

    // É invocado para fazer uma atualização na
    // interface gráfica
```

```
        @Override
        protected void onProgressUpdate(Integer... values) {
            text.setText(String.valueOf(values[0]));
        }

    }
```

A classe AsyncTask tem como sua função prioritária executar código que demanda mais poder de processamento em um thread diferente da UI thread, e o melhor disso é que ele não põe o programador em contato direto com a complexidade da programação tradicional para a manipulação de threads. No caso acima, o comentário do código já adianta grande parte do trabalho, no entanto, é preciso dizer que para utilizar a classe Asynctask é necessário estender ela passando alguns parâmetros na forma de generics. Generics é um conceito não particular da programação android e serve para parametrizar os tipos de dados passados. É recomendado que você já conhecesse o funcionamento deles, pois eles são bastante importantes em algumas situações da programação android.
Já os métodos funcionam da seguinte forma: o método doInBackground executa fora da UI thread e simula o processamento que queremos fazer através de um loop, o qual cria um contador. A cada ciclo de processamento desse contador, o método **publishProgress(i)** é invocado e o valor de i é passado. A função desse método é invocar o método de baixo, no caso o **onProgressUpdate**, o qual executa da UI thread e faz a atualização da interface gráfica.
Ainda é necessário acrescentar algumas linhas de código a classe que comanda a activity principal para que o projeto execute com sucesso.

Exemplo de como a classe vai ficar

```
package unicamp.cotuca.br.testeasync;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private TextView text;
    private Button bt;
```

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

        text = (TextView) findViewById(R.id.text);

        bt = (Button) findViewById(R.id.bt);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                processamento();
            }
        });
    }

    private void processamento() {
        Num num = new Num(text, bt);
        // Executa o doInBackground e passa o valor 50 como parâmetro
        num.execute(50);
    }
}
```

Apenas foi colocado o código necessário para que seja utilizada a classe Num a partir do click do botão. Neste caso, ela é instanciada e depois usada o método execute(param), passando o valor do parâmetro para a contagem dos ciclos.

Terminado, executar o projeto no emulador android e ver como a aplicação se comporta.

Gerenciando uso de Rede

Se o aplicativo executa muitas operações de rede, deve-se fornecer configurações de usuário que permitam aos usuários controlar tráfego de dados da aplicação, O nível desse controle deve observar a freqüência do aplicativo sincronizar dados, seja para fazer upload ou downloads somente quando estiver no Wi-Fi, seja para usar dados durante a roaming, e assim por diante.

Antes de executar operações de rede, é uma boa prática verificar o estado da conectividade de rede.

Se uma conexão de rede não estiver disponível, a aplicação deve responder positivamente. Para verificar a conexão de rede, normalmente usa-se as seguintes classes:

- ConnectivityManager: responde consultas sobre o estado da conectividade de rede. Também notifica os aplicativos quando a conectividade de rede muda.

- NetworkInfo: descreve o status de uma interface de rede de um determinado tipo (atualmente, celular ou Wi-Fi).

Uma maneira mais concisa de verificar se uma interface de rede está disponível é a seguinte. O método getActiveNetworkInfo () retorna uma instância NetworkInfo que representa a primeira interface de rede conectada que pode encontrar ou nulo se nenhuma das interfaces estiver conectada (o que significa que uma conexão à internet não está disponível):

Verificar se o dispositivo está conectado a uma rede WI-FI:

```
ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

if(networkInfo.isConnected()

   Log.d(TAG, "Conectado no WI-FI");
```

Verificar se o dispositivo está conectado a uma rede mobile:

```
ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);

if(networkInfo.isConnected()

   Log.d(TAG, "Conectado na rede do celular");
```

Observe que não basea-se decisões sobre se uma rede está "disponível". Deve-se sempre verificar isConnected () antes de executar operações de rede, já que isConnected () lida com casos como redes móveis escamosas, modo de avião e dados de fundo restritos.

Projeto 9 – AsyncTask, Nodejs e Web Service

Criar o projeto AsyncTaskWS.

Arquivo activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="unicamp.cotuca.br.asynctaskws.MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="37dp"
        android:layout_marginTop="32dp"
        android:scrollbars="vertical"
        android:textSize="24sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="168dp"
        android:layout_marginTop="112dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
</android.support.constraint.ConstraintLayout>
```

Na pasta res, criar a pasta e o arquivo main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/limpar_tarefas"
        android:icon="@android:drawable/ic_delete"
        android:title="Item"
        app:showAsAction="always" />
    <item
        android:id="@+id/iniciar_tarefa"
        android:icon="@android:drawable/btn_star_big_on"
        android:title="Iniciar"
        app:showAsAction="always" />
</menu>
```

Arquivo MainActivity.java

```
package br.com.treinaweb.projetoapi;

import android.os.AsyncTask;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    TextView textViewTarefa;
    ProgressBar progressBar;

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

        textViewTarefa = (TextView) findViewById(R.id.textView);
        textViewTarefa.setMovementMethod(new ScrollingMovementMethod());

        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        progressBar.setVisibility(View.INVISIBLE);
    }

    protected void atualizarView(String messagem){
        textViewTarefa.append(messagem+"\n");
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.iniciar_tarefa){
            MyTask task = new MyTask();
            task.execute("task 1", "task 2", "task 3", "task 4");
        }

       if(item.getItemId() == R.id.limpar_tarefas){
            textViewTarefa.setText("");
            progressBar.setVisibility(View.INVISIBLE);
        }

        return super.onOptionsItemSelected(item);
    }

    private class MyTask extends AsyncTask<String, String, String>{
        @Override
        protected void onPreExecute() {
            progressBar.setVisibility(View.VISIBLE);
            atualizarView("Tarefa Iniciada");
        }

        @Override
        protected String doInBackground(String... params) {
            for (int i = 0; i<params.length; i++){
                try {
```

```
                    publishProgress("Trabalhando com: "+params[i]);
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            return "Tarefa Concluida";
        }

        @Override
        protected void onPostExecute(String s) {
            atualizarView(s);
            progressBar.setVisibility(View.INVISIBLE);
        }

        @Override
        protected void onProgressUpdate(String... values) {
            atualizarView(values[0]);
        }
    }
}
```

Após fazer toda a programação, executar o projeto e verificar o resultado.

JSON e XML

As **requisições HTTP** podem utilizar o **JSON** ou **XML** para transmitir dados entre o cliente e o servidor. O **JSON** e o **XML** são tipos de dados estruturados que representam as informações trocadas entre os dispositivos na rede.

JSON

O **JSON (JavaScript Object Notation)** é uma estrutura de dados para armazenamento e transmissão de informações no formato texto. É simples e utilizado por aplicações Web e Mobile devido a sua capacidade de estruturar informações de uma forma bem compacta e leve, tornando mais rápido a leitura dessas informações.

O JSON foi adotado pela maioria das empresas para fazer a troca de milhões de dados pela Internet.

Exemplo de JSON:

```
{
“Alunos”:
[
{“RA”: “16101”,
 “nome”: “Felipe”},
{“RA”: “16102”,
 “nome”: “Clara”},
{“RA”: “16103”,
 “nome”: “Daniel”}
]
}
```

A representação de informações do JSON é muito simples: para cada valor representado, atribui-se um nome que descreve o seu significado. Esta sintaxe é derivada da forma utilizada pelo **JavaScript** para representar informações.

XML

XML, do inglês **eXtensible Markup Language**, é uma linguagem de marcação recomendada pela **W3C** para a criação de documentos com dados organizados hierarquicamente, tais como textos e banco de dados. A linguagem **XML** é classificada como extensível porque permite definir os elementos de marcação.

Exemplo de XML:

```
<Alunos>
  <element>
    <RA>16101</RA>
    <nome>Felipe</nome>
  </element>
  <element>
    <RA>16102</RA>
    <nome>Clara</nome>
  </element>
  <element>
    <RA>16103</RA>
    <nome>Daniel</nome>
  </element>
</ Alunos >
```

O **XML** traz uma sintaxe básica que pode ser utilizada para compartilhar informações entre diferentes computadores e aplicações.

Android Manifest

Todo aplicativo tem que ter um arquivo AndroidManifest.xml (precisamente com esse nome) no diretório raiz. O arquivo de manifesto apresenta informações essenciais sobre o aplicativo ao sistema Android, necessárias para o sistema antes que ele possa executar o código do aplicativo.

Entre outras coisas, o arquivo do manifesto serve para:

- Nomear o pacote Java para o aplicativo. O nome do pacote serve como identificador exclusivo para o aplicativo.
- Descrever os componentes do aplicativo, que abrangem atividades, serviços, receptores de transmissão e provedores de conteúdo que compõem o aplicativo. Ele também nomeia a classe que implementa cada um dos componentes e publica suas capacidades, como as mensagens Intent que podem lidar. Essas declarações informam ao sistema Android os componentes e as condições em que eles podem ser inicializados.
- Determinar os processos que hospedam os componentes de aplicativo.
- Declarar as permissões que o aplicativo deve ter para acessar partes protegidas da API e interagir com outros aplicativos. Ele também declara as permissões que outros devem ter para interagir com os componentes do aplicativo.
- Listar as classes Instrumentation que fornecem geração de perfil e outras informações durante a execução do aplicativo. Essas declarações estão presentes no manifesto somente enquanto o aplicativo está em desenvolvimento e são removidas antes da publicação do aplicativo.
- Declarar o nível mínimo da Android API que o aplicativo exige.
- Listar as bibliotecas às quais o aplicativo deve se vincular.

Permissão de internet e status de rede no arquivo AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Na classe MainActivity criar o método isOnline

```
private boolean isOnline(){
    ConnectivityManager cm =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo  networkInfo = cm.getActiveNetworkInfo();

    if(networkInfo != null  && networkInfo.isConnectedOrConnecting()){
        return true;
    } else {
        return false;
```

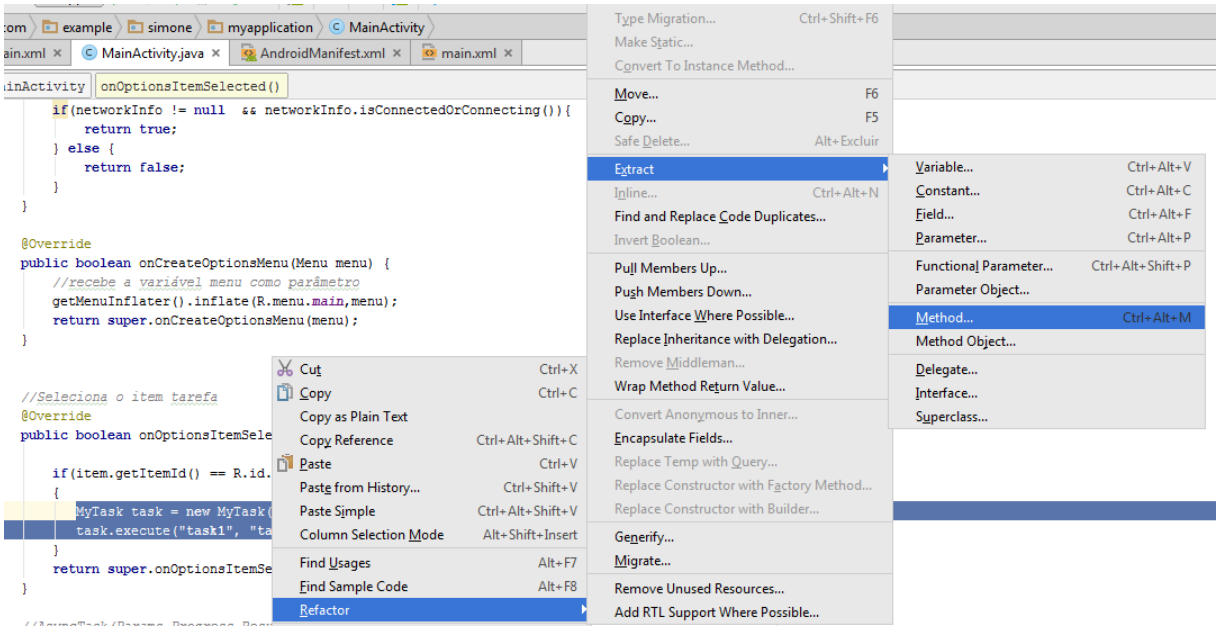
```
    }
}
```

Quando o usuário selecionar o item de menu Iniciar e executar a tarefa, verificar se o aplicativo está, ou não online.

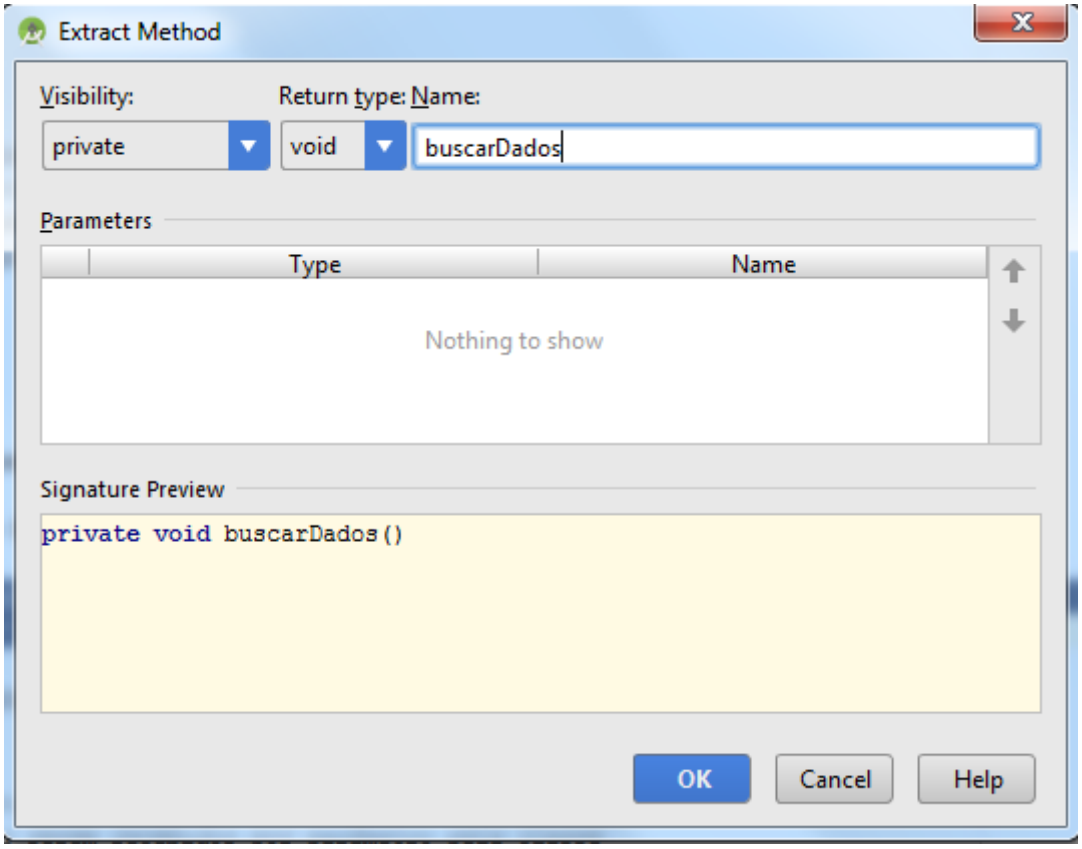
No método onOptionsItemSelected, extrair as linhas apontadas com a seta para um novo método.

```
//Seleciona o item tarefa
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if(item.getItemId() == R.id.iniciar_tarefa)
    {
    -> MyTask task = new MyTask();
    -> task.execute("task1", "task2", "task3", "task4");
    }
    return super.onOptionsItemSelected(item);
}
```

Selecione as duas linhas, clique com o botão direito do mouse, escolha as opções Refactor, Extract, Method



Nomear o novo método como buscarDados.



O método onOptionsItemSelected ficará como o exemplo:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if(item.getItemId() == R.id.iniciar_tarefa)
    {
        buscarDados();
    }
    return super.onOptionsItemSelected(item);
}
```

Para verificar se está on line, usar o método isOnLine()

```
//Seleciona o item tarefa
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if(item.getItemId() == R.id.iniciar_tarefa)
    {
        if(isOnline()) {
            buscarDados();
        } else{
            //Mostrar ao usuário
            Toast.makeText(this,"Rede não está
disponível",Toast.LENGTH_LONG).show();
        }
    }
```

```
    return super.onOptionsItemSelected(item);
}
```

A classe MainActivity ficará da seguinte forma:

```
package com.example.simone.myapplication;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    TextView textViewTarefa;


    protected void atualizarView(String mensagem){

        //o método append adiciona conteúdo ao TextView
        textViewTarefa.append(mensagem + "\n");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textViewTarefa = (TextView)findViewById(R.id.textView);
        //Setar o método de movimento do TextView
        textViewTarefa.setMovementMethod(new ScrollingMovementMethod());
    }

    private boolean isOnline(){
        ConnectivityManager cm =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo  networkInfo = cm.getActiveNetworkInfo();

        if(networkInfo != null  && networkInfo.isConnectedOrConnecting()){
            return true;
        } else {
            return false;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //recebe a variável menu como parâmetro
        getMenuInflater().inflate(R.menu.main,menu);
        return super.onCreateOptionsMenu(menu);
    }

    //Seleciona o item tarefa
    @Override
```

```
    public boolean onOptionsItemSelected(MenuItem item) {

        if(item.getItemId() == R.id.iniciar_tarefa)
        {
            if(isOnline()) {
                buscarDados();
            } else{
                //Mostrar ao usuário
                Toast.makeText(this,"Rede não está
disponível",Toast.LENGTH_LONG).show();
            }
        }

        if(item.getItemId() == R.id.limpar_tarefas){
           textViewTarefa.setText("");
           progressBar.setVisibility(View.INVISIBLE);
        }


        return super.onOptionsItemSelected(item);
    }

    private void buscarDados() {
        MyTask task = new MyTask();
        task.execute("task1", "task2", "task3", "task4");
    }


    //AsyncTask(Params,Progress,Result)
    //Params - tipos de dados a serem recebidos por parâmetro pela classe
    //Progress - quando a classe estiver executando, qual tipo de dados
será notificado no progresso
    //result - qual tipo de dado será retornado

    //Nessa classe será tudo String
    private class MyTask extends AsyncTask<String, String, String>{

        //Executa antes do doInBackGround
        @Override
        protected void onPreExecute(){

            atualizarView("Tarefa Iniciada");

        }
        //Código a ser executado em segundo plano
        @Override
        protected String doInBackground(String... params) {

            for(int i=0; i<params.length; i++){

                try {

                    //Aciona o método onProgressUpdate e monitora o que
está acontecendo no método doInBackGround
                    publishProgress("Trabalhando com:" + params[i]);

                    //Demora 1 segundo
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
            }
            return "Tarefa Concluída";
        }


        //o parâmetro s tem o retorno do método doInBackGround
        @Override
        protected void onPostExecute(String s){
            atualizarView(s);
        }

        //durante o processo da thread executando, pçode-se ver o progresso
        @Override
        protected void onProgressUpdate(String... values){
            atualizarView(values[0]);

        }
    }
}
```

Após fazer toda a programação, executar o projeto e verificar o resultado.

NodeJs

Na URL do browser digitar https://nodejs.org baixar a versão v6.11.0 LTS.

Instalar o nodejs seguinte as instruções da tela.

API

É um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla **API** refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos".

Copiar os arquivos package.json e api.js para a pasta do projeto

Arquivo package.json

```
{
  "name": "android-api",
  "version": "1.0.0",
  "description": "",
  "main": "api.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.17.1",
    "cors": "^2.8.3",
    "express": "^4.15.2",
    "js2xmlparser": "^3.0.0"
  }
}
```

Arquivo api.js

```
var express = require('express');
    var app = express();
    var cors = require('cors');
    var bodyParser = require('body-parser');
    var js2xmlparser = require("js2xmlparser");

    app.use(bodyParser.json());
    app.use(bodyParser.urlencoded({ extended: true }));
    app.use(cors());
```

```
app.get('/api/produtos/xml', function (req, res) {
	setTimeout(function(){
	   res.set('Content-Type', 'text/xml');
	   res.send(js2xmlparser.parse("produtos",
	            {"produto":[{
	               id: 1,
	               nome: 'Chocolate',
	               categoria: 'doce',
	               descricao: 'Chocolate ao leite'
	            },
	            {
	               id: 2,
	               nome: 'Coxinha',
	               categoria: 'salgado',
	               descricao: 'Massa de mandioca e recheio de frango desfiado'
	            },
	            {
	               id: 3,
	               nome: 'Suco',
	               categoria: 'bebida',
	               descricao: 'Suco de laranja natural'
	            }]}
	      ));
	   res.status(200).end();
	}, 4000);
});


app.get('/api/produtos/json', function (req, res) {
	setTimeout(function(){
	   res.header('Access-Control-Allow-Origin', '*')
	   .send(200,
	            [{
	               id: 1,
	               nome: 'Chocolate',
	               categoria: 'doce',
	               descricao: 'Chocolate ao leite'
	            },
	            {
	               id: 2,
	               nome: 'Coxinha',
	               categoria: 'salgado',
	               descricao: 'Massa de mandioca e rechei de franco desfiado'
	            },
	            {
	               id: 3,
	               nome: 'Suco',
	               categoria: 'bebida',
	               descricao: 'Suco de laranja natural'
	            }]
```

```
            )
        }, 4000);
});

app.listen(3000);
console.log('A API está no ar');
```

No Windows, abrir o aplicativo cmd. Acessar o local onde o arquivo api.

Instalar as dependências do arquivo package.json através da linha de comando:

>npm install

Para iniciar o arquivo api.js, escrever no prompt de comando:

>node api.js

A mensagem “A api está no ar” aparecerá na tela de comando.

Abrir o navegador e digitar a url:

localhost:3000/api/produtos/json

Os produtos serão retornado no formato json

No navegador digitar a url:

localhost:3000/api/produtos/xml

Os produtos serão retornados no formato xml

Com a api funcionando, os dados poderão ser acessados através de aplicação do Android via WebService.

Classe HttpURLConnection

Classe abstrata que contém os métodos utilizados para efetuar uma conexão HTTP

Método getInputStream

Retorna um fluxo de entrada que lê a partir desta conexão aberta.

Classe BufferedReader

Lê texto de um fluxo de entrada tipo caracter, bufferizando (invólucro) esses caracteres para fornecer uma leitura eficiente dos caracteres, arrays e linhas. Resumindo, é uma classe usada para fazer a leitura de uma InputStreamReader.

Classe InputStreamReader

Um InputStreamReader é uma ponte de fluxos de bytes para fluxos de caracteres: lê bytes e decodifica-os em caracteres usando um charset especificado.
Para uma maior eficiência, considere envolver um InputStreamReader dentro de um BufferedReader. Por exemplo:

BufferedReader em
= novo BufferedReader (novo InputStreamReader (System.in));

Classe StringBuilder

Permite criar e manipular dados de Strings dinamicamente, ou seja, podem criar variáveis de String modificáveis. Uma vantagem sobre a classe String é a concatenação de strings. Pois quando concatena strings com StringBuilder é invocado o método append. Esse método aloca novas strings concatenadas para o mesmo objeto, ou seja, cada vez que concatena strings não são criadas cópias dos objetos como é realizado pelo método **concat** da classe String, contribuindo para um melhor desempenho do sistema.

Classe URL

Representa um Localizador de Recursos Uniformes, um ponteiro para um "recurso" na World Wide Web.

Acessando dados Json e XML via HttpURLConnection

No Projeto 8 - AsyncTaskWS criar a classe HttpManager

```
package unicamp.cotuca.br.asynctaskws;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

/**
 * Created by simone on 28/09/2017.
 */

public class HttpManager {
```

```
    public static String getDados(String uri){
        BufferedReader reader = null;

        try{

            URL url = new URL(uri);
            HttpURLConnection con =
(HttpURLConnection)url.openConnection();

            StringBuilder stringBuilder = new StringBuilder();
            reader = new BufferedReader(new
InputStreamReader(con.getInputStream()));

            String line;
            while ((line = reader.readLine())!= null){
                stringBuilder.append(line + "\n");
            }

            return stringBuilder.toString();
        }
        catch(Exception e){
            e.printStackTrace();
            return null;
        }
        finally{
            if(reader!=null){
                try{
                    reader.close();
                }
                catch (IOException e){
                    e.printStackTrace();
                    return null;
                }
            }
        }
    }
}
```

Na classe MainActivity, fazer as seguintes alterações:

Método onOptionsItemSelected(MenuItem item)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if (item.getItemId() == R.id.iniciar_tarefa){

        if (isOnLine())
            buscarDados("http://177.220.18.43:3000/api/produtos/xml");
        else
            Toast.makeText(this,"Rede não está
disponível",Toast.LENGTH_LONG).show();
    }

    if(item.getItemId() == R.id.limpar_tarefas){
        textViewTarefa.setText("");
        progressBar.setVisibility(View.INVISIBLE);
    }
```

```
    return super.onOptionsItemSelected(item);
}
```

Método buscarDados()

```
private void buscarDados(String uri) {
    MyTask task = new MyTask();
    progressBar.setVisibility(View.VISIBLE);
    //task.execute("task1","task2","task3","task4");
    task.execute(uri);
}
```

Classe MyTask. Método doInBackground()

```
@Override
protected String doInBackground(String... params) {

   String conteudo = HttpManager.getDados(params[0]);
    return conteudo;

 /*
    for (int i=0; i<params.length;i++){
        try {
            Thread.sleep(1000);
            publishProgress(("Trabalhando com " + params[i]));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    return "Tarefa Concluída";
    */
}
```

Finalmente, a classe MainActivity ficará como o modelo abaixo:

```
package unicamp.cotuca.br.asynctaskws;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    TextView textViewTarefa;
    ProgressBar progressBar;

    protected  void atualizarView(String mensagem){

        textViewTarefa.append(mensagem + "\n");
```

```
    }

    private boolean isOnLine(){

        ConnectivityManager cm =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo networkInfo = cm.getActiveNetworkInfo();

        if(networkInfo != null && networkInfo.isConnectedOrConnecting()){
            return true;
        }
        else{
            return false;
        }



    }

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

        textViewTarefa = (TextView)findViewById(R.id.textView);
        textViewTarefa.setMovementMethod(new ScrollingMovementMethod());

        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        progressBar.setVisibility(View.INVISIBLE);



    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //recebe a variável menu como parâmetro
        getMenuInflater().inflate(R.menu.main,menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        if (item.getItemId() == R.id.iniciar_tarefa){

            if (isOnLine())
                buscarDados("http://177.220.18.43:3000/api/produtos/xml");
            else
                Toast.makeText(this,"Rede não está
disponível",Toast.LENGTH_LONG).show();
        }

        if(item.getItemId() == R.id.limpar_tarefas){
            textViewTarefa.setText("");
            progressBar.setVisibility(View.INVISIBLE);
        }

        return super.onOptionsItemSelected(item);
```

```
    }

    private void buscarDados(String uri) {
        MyTask task = new MyTask();
        progressBar.setVisibility(View.VISIBLE);
        //task.execute("task1","task2","task3","task4");
        task.execute(uri);
    }

    //AsyncTask(Params,Progress,Result)
    //Params - tipos de dados a serem recebidos por parâmetro pela classe
    //Progress - quando a classe estiver executando, qual tipo de dado será
notificado no progresso
    //result- - qual tipo de dado será retornado
    private class MyTask extends AsyncTask<String,String,String>{

        @Override
        protected void onPreExecute() {
            atualizarView("Tarefa Iniciada");
        }

        @Override
        protected String doInBackground(String... params) {

           String conteudo = HttpManager.getDados(params[0]);
            return conteudo;

         /*
            for (int i=0; i<params.length;i++){
                try {
                    Thread.sleep(1000);
                    publishProgress(("Trabalhando com " + params[i]));
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            return "Tarefa Concluída";
            */
        }




        // o parâmetro s recebe o retorno do método doInBackGround
    @Override
    protected void onPostExecute(String s) {
            atualizarView(s);
            progressBar.setVisibility(View.INVISIBLE);

        }


    @Override
    protected void onProgressUpdate(String... values) {
        atualizarView(values[0]);
    }
}
}
```

Após fazer toda a programação, executar o projeto e verificar o resultado.

Trabalhando com Dados XML e Json

No Projeto 8 - AsyncTaskWS criar a classe Produto

```
package unicamp.cotuca.br.asynctaskws;

public class Produto {
    private int id;
    private String nome;
    private String categoria;
    private String descricao;

    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

    public String getCategoria() {
        return categoria;
    }

    public void setCategoria(String categoria) {
        this.categoria = categoria;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
}
```

Criar a classe ProdutoXMLParser

```
package unicamp.cotuca.br.asynctaskws;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

public class ProdutoXMLParser {
```

```
    public static List<Produto> parseDados(String conteudo) {

        try {

            boolean dadosNaTag = false;
            String tagAtual = "";
            Produto produto = null;
            List<Produto> produtoList = new ArrayList<>();

            XmlPullParserFactory factory =
XmlPullParserFactory.newInstance();
            XmlPullParser parser = factory.newPullParser();
            parser.setInput(new StringReader(conteudo));

            int eventType = parser.getEventType();

            while (eventType != XmlPullParser.END_DOCUMENT) {

                switch (eventType) {
                    case XmlPullParser.START_TAG:
                        tagAtual = parser.getName();
                        if (tagAtual.equals("produto")) {
                            dadosNaTag = true;
                            produto = new Produto();
                            produtoList.add(produto);
                        }
                        break;

                    case XmlPullParser.END_TAG:
                        if (parser.getName().equals("produto")) {
                            dadosNaTag = false;
                        }
                        tagAtual = "";
                        break;

                    case XmlPullParser.TEXT:
                        if (dadosNaTag && produto != null) {
                            switch (tagAtual) {
                                case "id":

produto.setId(Integer.parseInt(parser.getText()));
                                    break;
                                case "nome":
                                   produto.setNome(parser.getText());
                                   break;
                                case "categoria":
                                    produto.setCategoria(parser.getText());
                                    break;
                                case "descricao":
                                    produto.setDescricao(parser.getText());
                                    break;
                                default:
                                    break;
                            }
                        }
                        break;
                }

               eventType = parser.next();

            }
```

```
            return produtoList;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Classe JSONArray

Representa uma matriz JSON (uma sequência ordenada de valores).
Método lenght(): retorna o número de valores da matriz

Classe JSONObject

Representa um objeto JSON
Método getJsonObject: retorna o valor JSONObject associado a uma chave.

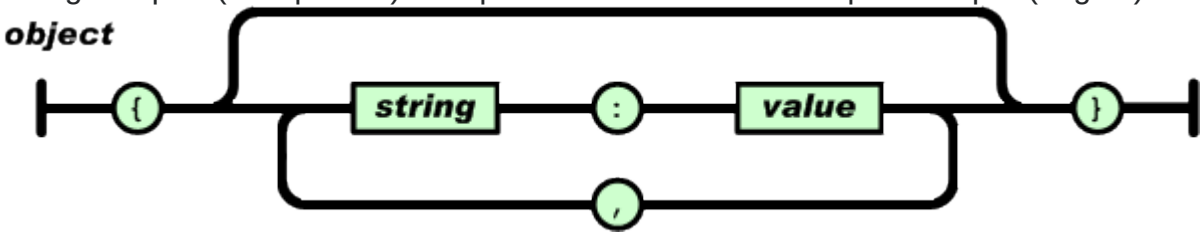
JSONObject:

- Contém valores nomeados (chave-> pares de valores)
  - como {ID : 1}
- A ordem dos elementos não é importante
  - um JSONObject de {id: 1, name: 'B'} é igual a {name: 'B', id: 1} .

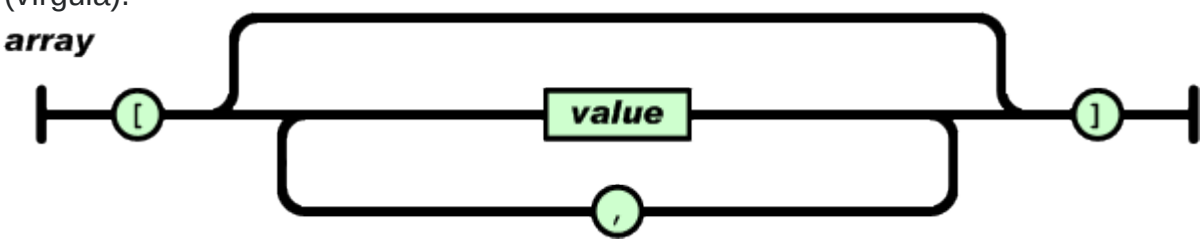
JSONArray:

- Contém apenas valores de série
  - como [1, 'value']
- A ordem dos valores é importante
  - array of [1,'value'] não é o mesmo que ['value',1]

objeto: um objeto é um conjunto não ordenado de pares nome / valor. Um objeto começa com {(braçadeira esquerda) e termina com} (braçadeira direita). Cada nome é seguido por: (dois pontos) e os pares nome / valor são separados por (vírgula).



matriz: uma matriz é uma coleção de valores ordenada. Uma matriz começa com [(suporte esquerdo) e termina com] (suporte direito). Os valores são separados por (vírgula).



Criar a classe ProdutoJsonParser

```
package unicamp.cotuca.br.asynctaskws;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

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

public class ProdutoJSONParser {
    public static List<Produto> parseDados(String content) {
        try {

            JSONArray jsonArray = new JSONArray(content);
            List<Produto> produtoList = new ArrayList<>();

            for (int i = 0; i< jsonArray.length(); i++){

                JSONObject jsonObject = jsonArray.getJSONObject(i);
                Produto produto = new Produto();

                produto.setId(jsonObject.getInt("id"));
                produto.setNome(jsonObject.getString("nome"));
                produto.setCategoria(jsonObject.getString("categoria"));
                produto.setDescricao(jsonObject.getString("descricao"));

                produtoList.add(produto);
            }

            return produtoList;
        } catch (JSONException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Na classe MainActivity fazer as seguintes alterações:

Criar uma lista de produtos

```
List<Produto> produtoList;
```

No método onCreate iniciliazar a lista de produtos:

```
produtoList = new ArrayList<>();
```

No método onPostExecute comentar a linha:

// atualizarView(s);

E acrescentar as linhas:

```
produtoList = ProdutoJSONParser.parseDados(s);
atualizarView();
```

O método onPostExecute ficará da seguinte forma:

```
@Override
protected void onPostExecute(String s) {

    //atualizarView(s);

    //produtoList = ProdutoXMLParser.parseDados(s);
    produtoList = ProdutoJSONParser.parseDados(s);
    atualizarView();
    progressBar.setVisibility(View.INVISIBLE);

}
```

Fazer uma nova assinatura para o atualizarView

Remover o parâmetro String mensagem

```
protected void atualizarView(){
    if (produtoList != null){
        for (Produto produto: produtoList) {
            textViewTarefa.append(produto.getNome() + "\n");
        }
    }
}
```

Finalmente, a classe MainActivity ficará da seguinte forma:

```
package unicamp.cotuca.br.asynctaskws;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

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

public class MainActivity extends AppCompatActivity {
    TextView textViewTarefa;
    ProgressBar progressBar;

    List<Produto> produtoList;

    @Override
```

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

        textViewTarefa = (TextView) findViewById(R.id.textView);
        textViewTarefa.setMovementMethod(new ScrollingMovementMethod());

        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        progressBar.setVisibility(View.INVISIBLE);

        produtoList = new ArrayList<>();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.iniciar_tarefa){
            if (isOnline()){
                //buscarDados();
                buscarDados("http://177.220.18.43:3000/api/produtos/json");
                            }else {
                Toast.makeText(this, "Rede não está disponível",
Toast.LENGTH_LONG).show();
            }

        }


       if(item.getItemId() == R.id.limpar_tarefas){
               textViewTarefa.setText("");
               progressBar.setVisibility(View.INVISIBLE);
        }

        return super.onOptionsItemSelected(item);
    }



    /*private void buscarDados(){
        MyTask task = new MyTask();
        task.execute("task 1", "task 2", "task 3", "task 4");
    }*/

    private void buscarDados(String uri) {
        MyTask task = new MyTask();
        task.execute(uri);
    }

    protected void atualizarView(String message){
        textViewTarefa.append(message + "\n");
    }

    protected void atualizarView(){
        if (produtoList != null){
            for (Produto produto: produtoList) {
                textViewTarefa.append(produto.getNome() + "\n");
```

```
            }
        }
    }

    protected boolean isOnline(){
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnectedOrConnecting()){
            return true;
        }else{
            return false;
        }
    }

    private class MyTask extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            atualizarView("Tarefa Iniciada");
            progressBar.setVisibility(View.VISIBLE);
        }

        @Override
        protected String doInBackground(String... params) {
            /*for (int i = 0; i<params.length; i++){
                publishProgress("Trabalhando com: "+params[i]);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }*/

            String conteudo = HttpManager.getData(params[0]);
            return conteudo;
        }

        @Override
        protected void onPostExecute(String s) {

            //atualizarView(s);

            //produtoList = ProdutoXMLParser.parseDados(s);
            produtoList = ProdutoJSONParcer.parseDados(s);
            atualizarView();
            progressBar.setVisibility(View.INVISIBLE);

        }

        @Override
        protected void onProgressUpdate(String... values) {
            //atualizarView(values[0]);
        }
    }
}
```

Após fazer toda a programação, executar o projeto e verificar o resultado. O Projeto 8 está finalizado.

Adroid Bluetooth

A tecnologia Bluetooth expande os limites das aplicações Android fazendo com que elas interajam diretamente com outras aplicações no formato de redes ponto a ponto. Sua utilização pode ir desde a sincronização de dados do dispositivo móvel com aplicações desktop (como agendas e listas de afazeres), até a criação de jogos multiplayer. A tecnologia Bluetooth permite a criação de pequenas redes sem fio, também chamadas de WPANs (Wireless Personal Area Network), onde seus participantes podem trocar informações a curta distância (normalmente até 10 metros). A partir da versão 2.0 (Donut), a plataforma Android disponibilizou o acesso à API de Bluetooth (pacote **android.bluetooth**) de modo que os desenvolvedores pudessem utilizar esse recurso em suas aplicações. Através dela é possível ter acesso a toda a pilha de protocolos Bluetooth, permitindo realizar a busca por dispositivos disponíveis em sua área de alcance, estabelecer a conexão, e por fim trocar informações entre eles. (DEVMEDIA).

O recurso de Bluetooth é ativado através da API Bluetooth, disponível no pacote android.bluetooth.

Principais Classes:

BluetoothAdapter

> Representa o adaptador Bluetooth local (radio Bluetooth). O BluetoothAdapter é o ponto de entrada para todas as interações Bluetooth. Usando ele, você pode descobrir outros dispositivos Bluetooth, consultar uma lista de dispositivos emparelhados, instanciar um BluetoothDevice usando um endereço MAC conhecido, e criar um BluetoothServerSocket para escutar por comunicações de outros dispositivos.

BluetoothDevice

> Representa um dispositivo remoto Bluetooth. Use isso para requerer uma conexão com um dispositivo remoto através de um BluetoothSocket ou consultar informações sobre o dispositivo como seu nome, endereço, e estado da ligação.

A conexão é estabelecida através das classes:

BluetoothServerSocket e **BluetoothSocket**.

Elas funcionam de forma semelhante ao **java.net.ServerSocket** e **java.net.Socket**.

BluetoothSocket

Representa a interface para um socket Bluetooth (similar a um Socket TCP). Ele é um ponto de conexão que permite que uma aplicação troque dados com outro dispositivo Bluetooth através do InputStream e OutputStream.

BluetoothServerSocket

Representa um socket de servidor que escuta por requisições que chegam ao dispositivo onde a aplicação roda (similar a um ServerSocket TCP). De forma a conectar dois dispositivos Android, um dos dispositivos precosa abrir um socket servidor com essa classe. Quando um dispositivo Bluetooth faz uma requisição de conexão a esse dispositivo, o BluetoothServerSocket retornará um BluetoothSocket conectado quando a conexão é aceita.

Para utilizar os recursos de Bluetooth do aparelho, a aplicação precisa ter as permissões:
android.permission.BLUETOOTH e *android.permission.BLUETOOTH_ADMIN*.

Essas permissões devem estar declaradas no arquivo *AndroidManifest.xml*. A primeira é necessária para utilizar qualquer recurso Bluetooth, enquanto a segunda é requerida para permitir ativar o Bluetooth do aparelho, caso esteja desligado, e para realizar a busca por dispositivos.

Permissão para utilização de Bluetooth no arquivo AndroidManifest.xml

```
<uses-permission
android:name="android.permission.BLUETOOTH"/>
<uses-permission
android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Para utilizar qualquer recurso Bluetooth em um aplicação, precisa-se da instância da classe **BluetoothAdapter**, que pode ser obtida através do método estático **getDefaultAdapter.** Quando esse método retorna **null**, quer dizer que o aparelho não tem suporte a Bluetooth.

Verificando o suporte à API.

```
1. BluetoothAdapter adaptador =

     BluetoothAdapter.getDefaultAdapter();
```

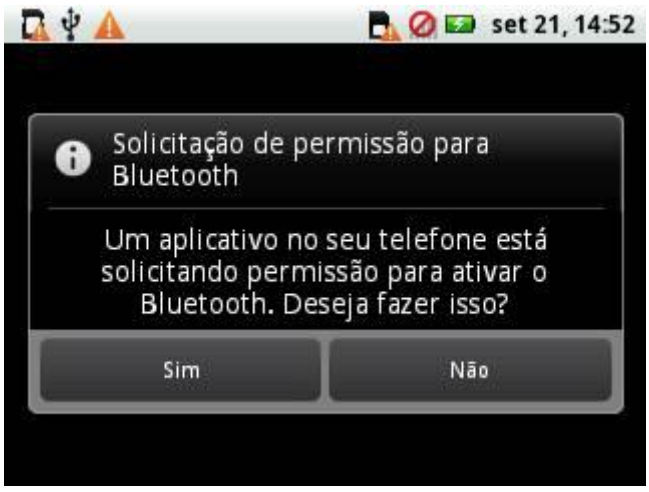
```
2. if (adaptador == null) {

3.   // Aparelho não suporta Bluetooth

4. }
```

Depois de obter a instância do adaptador, é necessário checar se o aparelho está com o Bluetooth ativado. Deixar o Bluetooth do aparelho ligado consome bateria, sendo assim, é interessante manter os recursos não utilizados desligados, e ativá-los apenas quando necessário. No código abaixo mostra como a API permite checar se o Bluetooth está ativo, e caso não esteja, solicitar ao usuário que o ative.

Deixando o aparelho visível.

```
1. if (!adaptador.isEnabled()) {

2.   Intent enableBtIntent =

       new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);

3.   startActivityForResult(enableBtIntent, BT_ATIVAR);

4. }
```

A solicitação para que o usuário ative o Bluetooth é feita através de uma **Intent**, que tem a ação *ACTION_REQUEST_ENABLE* definida na classe **BluetoothAdapter**. Quando essa intenção é disparada, o Android exibe uma mensagem:



A requisição é feita através do método **startActivityForResult()**, que recebe como argumento, além do objeto **Intent**, um inteiro que representa o código de requisição.

Projeto 8 – Android com Arduino Bluetooth

Criar o projeto LEDBluetoothRA. Usar a API 22, Android 5.1 Lollipop.

Package: br.unicamp.cotuca.ledbluetooth

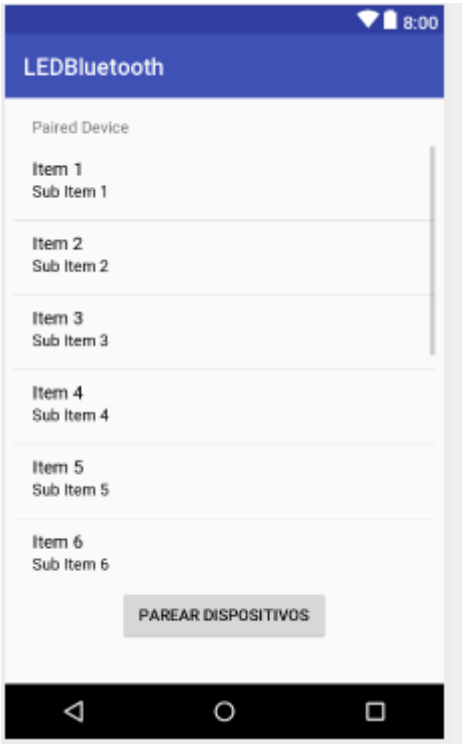
Nomear a classe da MainActivity como DeviceList

Criar o XML da classe DeviceList

```
<?xml version=”1.0” encoding=”utf-8”?>
<android.support.constraint.ConstraintLayout
xmlns:android=”http://schemas.android.com/apk/res/android”
    xmlns:app=”http://schemas.android.com/apk/res-auto”
    xmlns:tools=”http://schemas.android.com/tools”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    tools:context=”br.unicamp.cotuca.ledbluetooth.DeviceList”>

    <TextView
        android:id=”@+id/textView”
        android:layout_width=”wrap_content”
        android:layout_height=”wrap_content”
        android:layout_marginStart=”24dp”
        android:layout_marginTop=”16dp”
        android:text=”Paired Device”
        app:layout_constraintStart_toStartOf=”parent”
        app:layout_constraintTop_toTopOf=”parent” />
```

```
    <Button
        android:id=”@+id/btnPaired”
        android:layout_width=”183dp”
        android:layout_height=”wrap_content”
        android:layout_marginBottom=”36dp”
        android:layout_marginStart=”100dp”
        android:layout_marginTop=”8dp”
        android:text=”Parear dispositivos”
        app:layout_constraintBottom_toBottomOf=”parent”
        app:layout_constraintStart_toStartOf=”parent”
        app:layout_constraintTop_toBottomOf=”@+id/deviceList”
        app:layout_constraintVertical_bias=”1.0” />

    <ListView
        android:id=”@+id/deviceList”
        android:layout_width=”368dp”
        android:layout_height=”376dp”
        android:layout_marginStart=”8dp”
        android:layout_marginTop=”8dp”
        app:layout_constraintStart_toStartOf=”parent”
        app:layout_constraintTop_toBottomOf=”@+id/textView” />

</android.support.constraint.ConstraintLayout>
```

A tela ficará da seguinte forma:




```
package br.unicamp.cotuca.ledbluetooth;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;

public class DeviceList extends AppCompatActivity {

    Button btnPaired;
    ListView deviceList;


    //Criar as variáveis para controlar o Bluetooth:
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> pairedDevices;
```

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

        //Iniciar a variáveis
        btnPaired = (Button) findViewById(R.id.btnPaired);
        deviceList = (ListView) findViewById(R.id.deviceList);


        //Verificar o Bluetooth do dispositivo
        myBluetooth = BluetoothAdapter.getDefaultAdapter();
        if (myBluetooth == null) {
            //Não achou o Bluetooth do dispositivo
            Toast.makeText(getApplicationContext(), "Bluetooth Device Not
Available", Toast.LENGTH_LONG).show();
            //Finaliza a aplicação
            finish();
        }
        else {
            //Checa se o aparelho está com o Bluetooth ativado
            if (myBluetooth.isEnabled())
            {}
            else {
                /*
                 A solicitação para que o usuário ative o Bluetooth
                 é feita através de uma Intent, que tem a ação
                 ACTION_REQUEST_ENABLE definida na classe BluetoothAdapter.
                */
                Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);

                /*
                O método startActivityForResult() recebe como argumento o
                Intent turnBTon e um inteiro que representa o código de
                requisição para ativar o Bluetooth
                */
                startActivityForResult(turnBTon, 1);
            }

        }

        //Selecionar botão para parear dispositivos
        btnPaired.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                pairedDevicesList();
            }
        });

    }

    //Método para parear dispositivos
    private void pairedDevicesList() {
        /*
        Para obter a lista de aparelhos conhecidos, deve-se utilizar o
        método getBoundedDevices() do objeto BluetoothAdapter
        */
        pairedDevices = myBluetooth.getBondedDevices();
        ArrayList list = new ArrayList();
        //Adiciona os aparelhos pareados em uma lista
```

```
        if (pairedDevices.size() > 0) {
            for (BluetoothDevice bt : pairedDevices) {
                /Pega o nome e o endereço do dispositivo Bluetooth
                 list.add(bt.getName() + "\n" + bt.getAddress());}
        }
       else {
            Toast.makeText(getApplicationContext(), "Não foi encontrado
dispositivo pareado.", Toast.LENGTH_LONG).show();
        }
        final ArrayAdapter adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, list);
        deviceList.setAdapter(adapter);

        //Seleciona um item da lista e chama o método myListClickListener
        deviceList.setOnItemClickListener(myListClickListener);

    }//Fecha o método pairedDevicesList()

    //Seleciona o dispositivo bluetooth
    private AdapterView.OnItemClickListener myListClickListener = new
AdapterView.OnItemClickListener() {
        public void onItemClick (AdapterView<?> av, View v, int arg2, long
arg3) {
            //Pega o endereço MAC do dispistivo Bluetooth conectado na placa
            Arduino
             String info = ((TextView) v).getText().toString();

             //Pega o nome lógico do Bluetooth - HC05 e número Mac:
             20:16:10:17:29:24
             System.out.println("info:" + info);

             //Separa só o número Mac: 20:16:10:17:29:24
             String address = info.substring(info.length() - 17);
             System.out.println("address:" + address);

             // Se prepara para chamar a activity ledControl
             Intent intent = new Intent(DeviceList.this, ledControl.class);

            //Guarda a variável de sessão para o endereço ser passado para
             activity ledControl
             Bundle param = new Bundle();
             param.putString("endereco",address);
             intent.putExtras(param);


             //Chama a ledControl
             startActivity(intent);
        }
    };

}
```

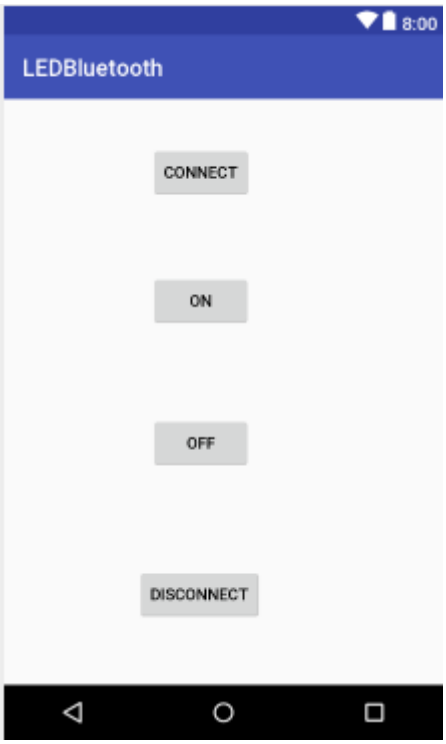
Criar a classe LedControl.java

Package: br.unicamp.cotuca.ledbluetooth

Criar o XML da classe LedControl

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="br.unicamp.cotuca.ledbluetooth.ledControl">

    <Button
        android:id="@+id/btnOn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="128dp"
        android:layout_marginTop="152dp"
        android:text="On"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnOff"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="128dp"
        android:layout_marginTop="76dp"
        android:text="Off"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnOn" />

    <Button
        android:id="@+id/btnDis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="116dp"
        android:layout_marginTop="84dp"
        android:text="Disconnect"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnOff" />

    <Button
        android:id="@+id/btnConn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Connect"
        tools:layout_editor_absoluteX="128dp"
        tools:layout_editor_absoluteY="40dp" />

</android.support.constraint.ConstraintLayout>
```

A tela XML ficará no seguinte format:



Classe LedControl.java

```
package br.unicamp.cotuca.ledbluetooth;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.util.UUID;

public class ledControl extends AppCompatActivity {

//Declara os componentes a serem usados
    Button btnOn, btnOff, btnDis, btnConn;
    String address = null;
    BluetoothAdapter myBluetooth = null;
    BluetoothDevice dispositivo = null;
    BluetoothSocket btSocket = null;
    private boolean isBtConnected = false;
```

```
    //Atribui um identificador exclusivo.  Universally Unique IDentifier
    static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");

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

        //Atribui os componentes XML às variáveis correspondentes
        btnOn = (Button) findViewById(R.id.btnOn);
        btnOff = (Button) findViewById(R.id.btnOff);
        btnDis = (Button) findViewById(R.id.btnDis);
        btnConn = (Button) findViewById(R.id.btnConn);

        //Recebe o endereço do Bluetooth passado por parâmetro da Activity
DeviceList
        Intent intent = getIntent();
        Bundle params = intent.getExtras();
        address = params.getString("endereco");

        //Método para o botão Connect
        btnConn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Connect();
            }
        });

        //Método para o botão Disconnect
        btnDis.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Disconnect();
            }
        });

        //Método para o botão ON
        btnOn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnLed();
            }
        });

        //Método para o botão Off
        btnOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOffLed();
            }
        });
    }

    private void Connect() {

        //Instancia a subclasse AsyncTask ConnectBT
        ConnectBT bt = new ConnectBT();
        System.out.println("Endereço no LedControl:"+ address);
```

```
        //Passa para a Classe ConnectBT (AsyncTask) o endereço MAC
        bt.execute(address);
    }

    private void Disconnect() {
            if (btSocket!=null) //Se o btSocket estiver ocupado
            {
                try {
                    btSocket.close(); //fecha a conexão
                }
                catch (IOException e) {
                    msg("Erro");
                }
            } finish(); //returna
        }

    private void turnOnLed() {

        if (btSocket!=null) {
            try {

btSocket.getOutputStream().write("H".toString().getBytes());
            }
            catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOffLed() {
            if (btSocket!=null) {
                try {

btSocket.getOutputStream().write("L".toString().getBytes());
                }
                catch (IOException e) {
                    msg("Error");
                }
            }
        }

    private void msg(String s)
    {
        Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();
    }

   // AsyncTask<Parâmetros,Progresso,Resultado>
   /
    private class ConnectBT extends AsyncTask<String, Void, String> {
        //Declara uma variável para o retorno do AsyncTask
        private String ConnectSuccess = "OK";

        @Override
        protected String doInBackground(String... devices) {
            try {
                if (btSocket == null || !isBtConnected) {

                    //Classe BluetoothAdapter verifica se aparelho tem
suporte ao Bluetooth
                    myBluetooth = BluetoothAdapter.getDefaultAdapter();
```

```
                    //Variável address recebe o número MAC
                    String address = devices[0];
                    //Classe BluetoothDevice obtem o endereço Mac do
Bluetooth Arduino
                    dispositivo = myBluetooth.getRemoteDevice(address);
                    /*
                    Classe BluetoothSocket estabele a conexão para
OutputStream
                    O objeto BluetoothSocket, através do método
createRfcommSocketToServiceRecord(), recebe o identificador exclusivo do
serviço como argumento.
                    */
                    btSocket =
dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
                    //Cancela o dispositivo
                    BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                    //Conecta com o dispositivo selecionado
                    btSocket.connect();

                }
            } catch (IOException e) {
                ConnectSuccess = "NOK";
            }
            return ConnectSuccess;
        }

        @Override
        protected void onPostExecute(String resultado) {

            if (resultado=="NOK") {
                msg("Falhou a conexão.Tente novamente.");
                finish();
            } else {
                msg("Conectado.");
                isBtConnected = true;
            }
        }
    }
}
```

Referências Bibliográficas

LECHETA, Ricardo R. Google Android. 3ª edição. São Paulo: Editora Novatec, 2013

http://www.devmedia.com.br/principais-controles-no-android-conheca-sua-base/21200. Data de acesso: 14/08/2017

https://developer.android.com/reference/android/view/View.html. Data de acesso: 14/08/2017

http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922. Data de acesso: 15/08/2017

http://www.devmedia.com.br/exibindo-mensagens-no-android-com-a-classe-toast/26668. Data de acesso: 20/08/2017

https://androiddevbr.wordpress.com/2013/12/09/tudo-sobre-listview/. Data de acesso: 20/08/2017

http://www.devmedia.com.br/asynctask-trabalhando-com-tarefas-assincronas/29823 Data de acesso: 12/09/2017

https://developer.android.com/reference/java/net/HttpURLConnection.html. Data de acesso: 20/09/2017

https://www.klebermota.eti.br/2012/08/26/usando-o-bluetooth-no-android-traducao-da-documentacao-oficial/. Data de acesso: 06/06/2018

https://www.devmedia.com.br/utilizando-sqlite-em-aplicativos-android/32117 Data de acesso: 07/08/2018

https://www.youtube.com/watch?v=gNlTeSz33BI.
Data de acesso: 08/08/2018

http://joaoretamero.com.br/persistencia-android-sqlite
Data de acesso: 09/08/2018