AGRUPAMENTO DE ESCOLAS DE
CISTER - ALCOBAÇA

Ano letivo
2016/2017

ESCOLA SECUNDÁRIA D. INÊS DE CASTRO

CURSO CIENTÍFICO-HUMANÍSTICO DE

CIÊNCIAS E TECNOLOGIAS

PROJETO DE APLICAÇÕES INFORMÁTICAS B

A Minha Primeira *App*

NOME DO ALUNO: Francisco Miguel Loiro Serralheiro **Nº:14**

ANO:12º **TURMA:** CT-A

PROFESSOR: Nuno José da Silva Trindade Duarte

DATA:26-02-2017

Índice

1. Resumo

Este relatório descreve o processo e o trabalho que esteve subjacente à criação de uma aplicação Android. Uma *app* elementar que teve como principal objetivo, além da introdução de conceitos essenciais para qualquer iniciante na área *App Developing*, mas também ajudar a compreender algumas noções mais específicas e concretas para facilitar o trabalho do desenvolvimento de uma próxima aplicação que vai ter como função auxiliar a vida escolar do aluno da ESDICA.

Começo por explicar a primeira grande decisão em qualquer começo no desenvolvimento de uma *app*– *Web App* vs. *Native App* - e continuo a fundamentar o percurso tomado, desde as opções do tipo de abordagem e *software* a escolher até à última linha de código. No que toca ao método e ao raciocínio o ciclo foi sempre o mesmo: programar, testar, se falhar pesquisar e repetir. No final, acabo o relatório com uma reflexão mais pessoal, na qual explicito duas grandes conclusões minhas: começar a programar não é difícil e mudar o mundo também não.

2. Introdução

O presente relatório é elaborado no âmbito da disciplina de Aplicações Informáticas B, com vista à conclusão de um novo projeto, neste caso, continuando a aprender novas áreas da Programação.

O projeto desenvolveu-se na Escola Secundária D. Inês de Castro, nas aulas da disciplina, em computadores da biblioteca escolar e no meu computador pessoal, durante o 2º Período (até à pausa do Carnaval). Foram abordadas várias linguagens de programação tais como Java, XML, HTML e JavaScript, com o apoio dos programas Visual Studio e Android Studio.

O relatório destina-se, não só a descrever os problemas surgidos ao longo do trabalho, como também a apresentar um enquadramento do projeto, o seu principal *maingoal*, de que maneira é que ajudou para o atingir, bem como todo o conhecimento adquirido sobre *mobile development*.

O principal objetivo que tracei para este período foi simples: fazer uma aplicação para telemóvel. O outro objetivo, igualmente importante, é que o projeto feito tem de ter impacto no utilizador/*consumer* e, consequentemente, tem de conseguir mudar o mundo, neste caso, o dos alunos da ESDICA. Juntando o útil ao agradável, um colega de turma tinha feito, como projeto para o 1º Período uma aplicação desktop que tinha como função indicar ao aluno onde iria ter aula de Educação Física. No entanto, sendo de desktop, não estava a atingir o máximo de utilidade que a ideia podia oferecer. Assim, e com a respetiva autorização do colega, ofereci-me para, sendo um tema que me suscitava bastante interesse, desenvolver a aplicação para telemóvel.

Com isto em mente, e como seria a minha primeira vez neste campo da programação, decidi, primeiramente, fazer uma aplicação simples, básica, explorando as várias maneiras e *approaches*. Com efeito, o nome escolhido foi "ModernHelloWorld" e teria como função mudar a cor de duas *textview's* - uma presente na *activity* e outra num *widget* associado à *app*.

3. Descrição do problema

O principal problema a ser tratado é a do desenvolvimento de uma aplicação para *smartphone* que indique ao aluno da ESDICA onde é que vai ser o local da próxima aula de educação física, o que pressupõe que já tenha alguns conhecimentos essenciais para a desenvolver. Por isso, este relatório destina-se descrever o desenvolvimento de uma aplicação mais simples, que me ajude a obter exatamente as aprendizagens fundamentais para desenvolver uma *app*.

Por isso, a minha primeira *app* teria que, de alguma forma, ser desenvolvida *s*empre com o principal objetivo de esta estar interligada com o modo como vai ser feita a aplicação principal, isto é, tendo decidido que a aplicação fundamental iria ter a funcionalidade A, a aplicação a desenvolver neste projeto teria que, obrigatoriamente, ter a funcionalidade A, de forma a aprender como é que esta funcionalidade e a *app* comunicam e trabalham entre si.

No final, pretende-se atingir uma aplicação que tenha sido pensada e vista de diferentes modos, que cubra os problemas que a aplicação futura poderá oferecer, de modo a poupar tempo e trabalho no futuro e cujo desenrolar proporcione e conceda aprendizagens e *skills* básicas tangentes a qualquer *app developer*.

4. Recursos utilizados

Neste projeto utilizei variados recursos/ferramentas para o suporte e o desenvolvimento da *app*. No que toca ao ambiente de programação utilizei o Visual Studio Community 2015, na versão 14.0, com os plugins e *tools* "Project Ace", "Xamarin" e "Apache Cordova" e o "Android Studio". No que toca a emuladores de smartphones utilizei o "Visual Studio Emulator for Android", o "Xamarin Android Player" e os emuladores provenientes do "Android Studio". Para testar código HTML, CSS e JavaScript utilizei alguns *testers* online, tais como o *tester* do "w3schools.com"e o "JSFiddle".

Para manter o trabalho atualizado entre os diversos aparelhos usei uma pen USB e o Google Drive.

Foram-me disponibilizados equipamentos e ajuda informática pelo professor orientador e pela escola.

5. Desenvolvimento do projeto

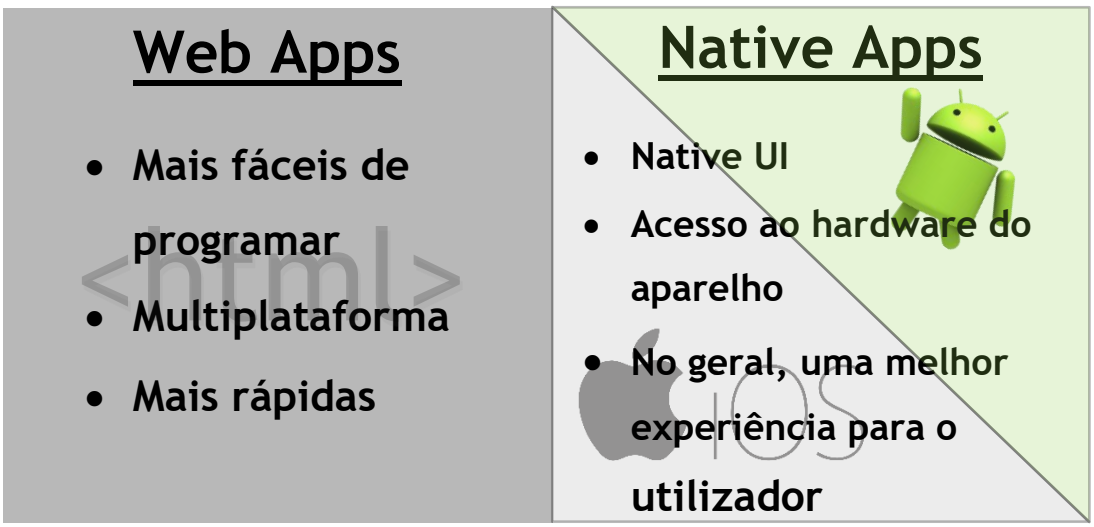
No decorrer do projeto fui-me encontrando com muitas dificuldades, tanto a nível de hardware como a nível de software, muitas delas forçaram-me mudar de visão e a *approach*, e por isso foi tudo menos um desenvolvimento direto e simples, o que, por outro lado, o tornou ainda mais enriquecedor e profícuo.

No início, ainda tentei desenvolver utilizando o Xamarin, mas este programa começou rapidamente a dar muitos problemas.

Posteriormente, reparei que ainda não tinha sequer tomado uma decisão fundamental em *App Developing*:

5.1. Web App vs. Native App

A primeira grande decisão que se tem de fazer quando se planeia uma aplicação para telemóvel é o tipo de aplicação que se vai desenvolver. Será uma *web app* desenvolvida com HTML, CSS e JavaScript ou será que a melhor solução seria uma *native app*, para o sistema operativo mais comum na escola? Ambas apresentam vantagens:



Por um lado, a *web app* é mais rápida, mais fácil de programar (o que seria vantajoso para alguém que está a começar) e seria muito útil para atingir um maior número de alunos, devido à capacidade que tem para ser instalada em qualquer OS móvel.

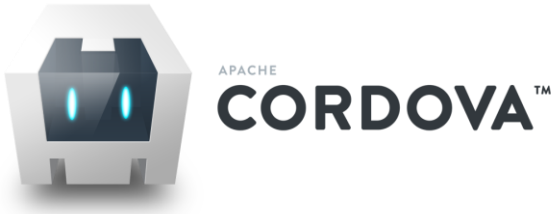
Por outro, e após deliberar e discutir com o professor orientador entre a opção de criar uma aplicação em *web*, numa página *web* em que se utilizaria uma base de dados para suportar e ajudar a gestão dos dados ou uma *web app* que utilizaria uma *webview* e uma aplicação móvel que seria diretamente instalada no aparelho do aluno, chegámos à conclusão que a melhor solução para o problema seria uma aplicação móvel com um *widget* associado em que é apresentado ao aluno o local das duas próximas aulas, porque ninguém pretende saber onde vai ser a aula no próximo mês (fora casos excecionais). Feita esta decisão fundamental, parti para a escolha das

diferentes abordagens e maneiras de desenvolver a ideia e os diferentes ambientes de programação em consideração.

5.2. Apache Cordovavs Android Studio

1. Apache CordovaTool no Visual Studio com o Project Ace

Com isto em mente, e como eu estava mesmo interessado na vantagem da multiplataforma que as *web apps* oferecem, procurei desenvolvê-la com um widget associado. Depois de uma pesquisa no Google encontrei o "Project Ace", que é um plugin para o Apache Cordova no Visual Studio. Foi assim que comecei a desenvolver o "ModernHelloWorld" com a abordagem da *webview*. A *app* seria muito simples. Com a ajuda de *testers* de *Web Development*(HTML, JavaScript e CSS) online consegui desenvolver uma aplicação cujo *click*num botão da *activity* iria gerar uma cor aleatória que, posteriormente iria ser atribuída para a *textview* da mesma *activity*. No entanto, e mesmo depois da experiência de desenvolvimento com o "Android Studio", adicionar um *widget* e programar é muito mais difícil e complexo comparativamente com o "Android Studio".

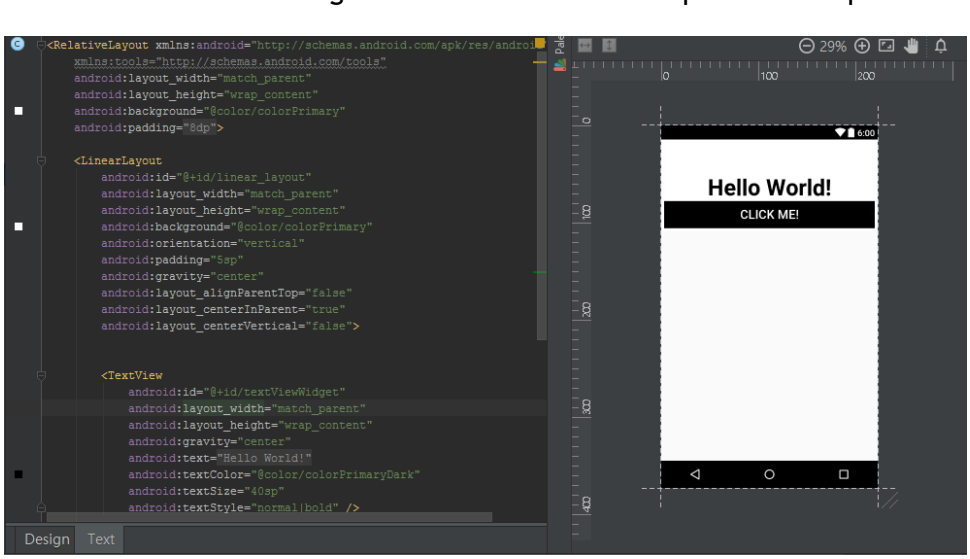


2. Android Studio

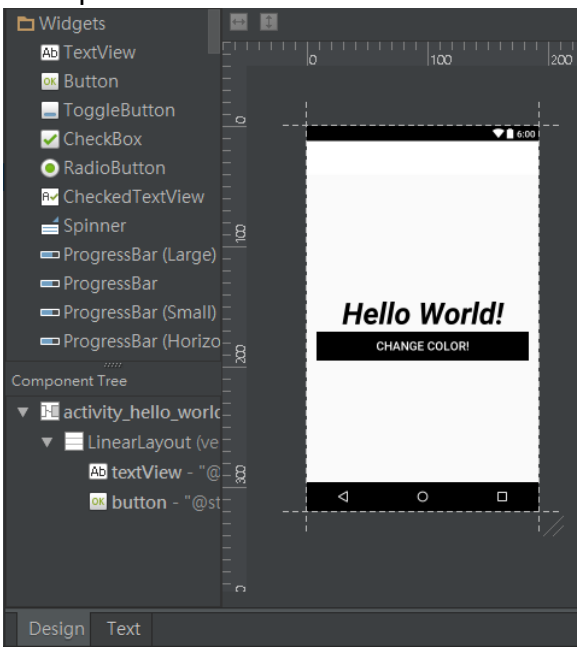
Visto que a melhor maneira de desenvolver um *widget* é através de software *native* ao sistema operativo do telemóvel e a tentativa de desenvolver um *widget* com o plugin "Project Ace" se revelou praticamente impossível, avançei para a programação *native* para Android. O motivo da escolha deste OS prende-se apenas no facto de ser o mais comum e o mais utilizado pelos alunos da escola. Além disso, o ambiente de programação fornecido pela Google no "Android Studio" é mais que razoável e suficiente para as necessidades deste projeto. Posto isto, segui para a programação do projeto utilizando o Android Studio como software de desenvolvimento.

5.3. Programar para Android

Primeiramente, criei o design da aplicação. O pretendido era uma *activity*, com um botão, uma *textview* com o texto "Hello World!" e um *widget* exatamente igual. Ao mergulhar no programa percebi que o design podia ser feito de duas maneiras: ou através de uma maneira mais visual ou através do código-source em XML. Acabei por usar um pouco dos dois para concebê-lo.



Layout do *Widget*



Layout da *Activity*

Posteriormente, passei para a lógica e o algoritmo da aplicação. A meta final desta aplicação era que ajudasse e tornasse muito mais fácil a conceção da aplicação escolar. Daí eu ter que criar uma aplicação que cubra, principalmente, a comunicação *widget-activity*, as propriedades de *textviews* e o simples evento de um botão.

Acabei por idealizar uma simples *app* que ao clicar num botão do *widget*, abria-se a *activity* e ao clicar no botão desta as *textviews*, tanto no *widget* como na activity mudavam para uma cor diferente. Com efeito, partindo de problema em problema:

- Problema do botão no *widget:* O objetivo era abrir uma *activity* com um *click* no botão num *widget*. Claro que uma simples pesquisa pela Internet me resolveu o problema, mas isso de nada servia se eu não percebesse o código fornecido.

```
// Create an Intent to launch Activity
Intent intent = new Intent(context, HelloWorld.class);
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);
//
// Get the layout for the App Widget and attach an on-click listener
// to the button
RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.hello_world_w);
views.setOnClickPendingIntent(R.id.button_widget, pendingIntent);

// Tell the AppWidgetManager to perform an update on the current app widget
appWidgetManager.updateAppWidget(appWidgetId, views);
```

Primeiramente, é declarado um "intent"

que é algo que liga duas componentes da aplicação em *runtime* e vai ser responsável por realizar uma ação, neste caso as duas componentes são o *widget*e a *activity* e a ação vai ser abrir essa *activity*.Depois, é declarado um "PendingIntent" que é aquilo que vai dar a ação ao *Intent*, daí o método "PendingIntent.getActivity()" que vai abrir a *activity* do "intent" declarado antes.

Depois é necessário "plantar" um "on-click listener" que é algo que vai ficar em *standby* e vai ser acionado aquando o *click*. Para isso é primeiro necessário declarar a "RemoteView" do *widget*, que é apenas o nome dado ao "layout" do *widget* no código. Finalmente, após plantar o "on-click listener" no botão do *widget*, vai ser preciso enviar um "update" ao *widget*, que é a função do "appWidgetManager".

- Problema da cor das textviews: O primeiro passo seria descobrir como programar um botão no "Android Studio". Apesar de haver mais que uma maneira (como a que foi utilizada no widget), acabei por ir pela a que me pareceu, na minha opinião, a mais fácil. Todas a *views* têm uma propriedade, que pode ser declarada no documento XML da *view*, "onClick" que declara o nome do método chamado aquando o "click" na *view*. Neste caso, acabei por criar um método chamado "ChangeTextColor".
  Sabendo isto, bastava-me criar um método que continha o código a ser executado. O código teria que, primeiro, obter a *textview* e depois atribuir-lhe uma cor.

```
public void ChangeTextColor(View view) {
    //get textview
    TextView tv = (TextView) findViewById(textView);
    //set random color to textivew
    tv.setTextColor(generateRandomColor());
```

- Atribuição de uma cor aleatória à *textview* da *activity*

```
public static int generateRandomColor() {
    int randomColor;
    Random rand = new Random();
    //generates random color
    int r = rand.nextInt();
    int g = rand.nextInt();
    int b = rand.nextInt();
    randomColor = Color.rgb(r, g, b);
    return randomColor;

}
```

- Método que vai gerar a cor aleatória

A pergunta óbvia a fazer ao analisar o código é "Porquê o método a mais? Porque não gerar a cor aleatória no mesmo método?" O que nos leva ao próximo passo do problema, atribuir uma cor, não necessariamente a mesma (o que interessa é o que o *click* do botão na *activity* provoca no *widget*), à *textview* do widget. Este foi o passo que, para mim, foi o mais difícil. Conseguir programar uma maneira (e entendê-la) de mudar a cor da *textview*do *widget* aquando o *click* do botão na *activity*. Sabendo que seria aquando o *click* do botão teria que ser no método "ChangeTextColor()". Ao pesquisar sobre a comunicação *widget-activity* deparei-

me com o método “OnReceive()” que é uma componente de qualquer *widget* que vai ter um papel-chave na solução do problema. Este método funciona como uma espécie de “antena”, que vai estar sempre preparada para receber *intents* que vai realizar uma ação no *widget*.

Sempre que é preciso realizar uma certa ação é primeiro necessário criar, como vimos anteriormente, um “intent”. Neste caso, como a ação que lhe vai ser dada é algo que ainda vai ser programado, que não está predefinida no “Android Studio” (como a “.getActivity”) vai ter que ser criada uma “ação”. Essa ação, para que a aplicação a conheça, vai ter que ser declarada no “Android Manifest”. O “Android Manifest” funciona como um cartão de identificação para o sistema. Quando a aplicação é instalada o sistema lê o “manifest” para obter informações essenciais, tais como as *activities*que tem, o ícone e o nome do “package”.

```
<receiver android:name=".HelloWorldW">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
        <action android:name="change_color" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/modern_hello_world_w_info" />
</receiver>
```

-Declaração do *Widget* no *Android Manifest* com respetivas “actions”

```
public void ChangeTextColor(View view) {
    //get textview
    TextView tv = (TextView) findViewById(textView);
    //set random color to textivew
    tv.setTextColor(generateRandomColor());
    //tell widget to change color too
    Intent intent = new Intent();
    intent.setAction("change_color");
    sendBroadcast(intent);

}
```

-O *Intent com a ação criada é enviada para* o *widget*, que vai ser recebida pelo método “onReceive()”

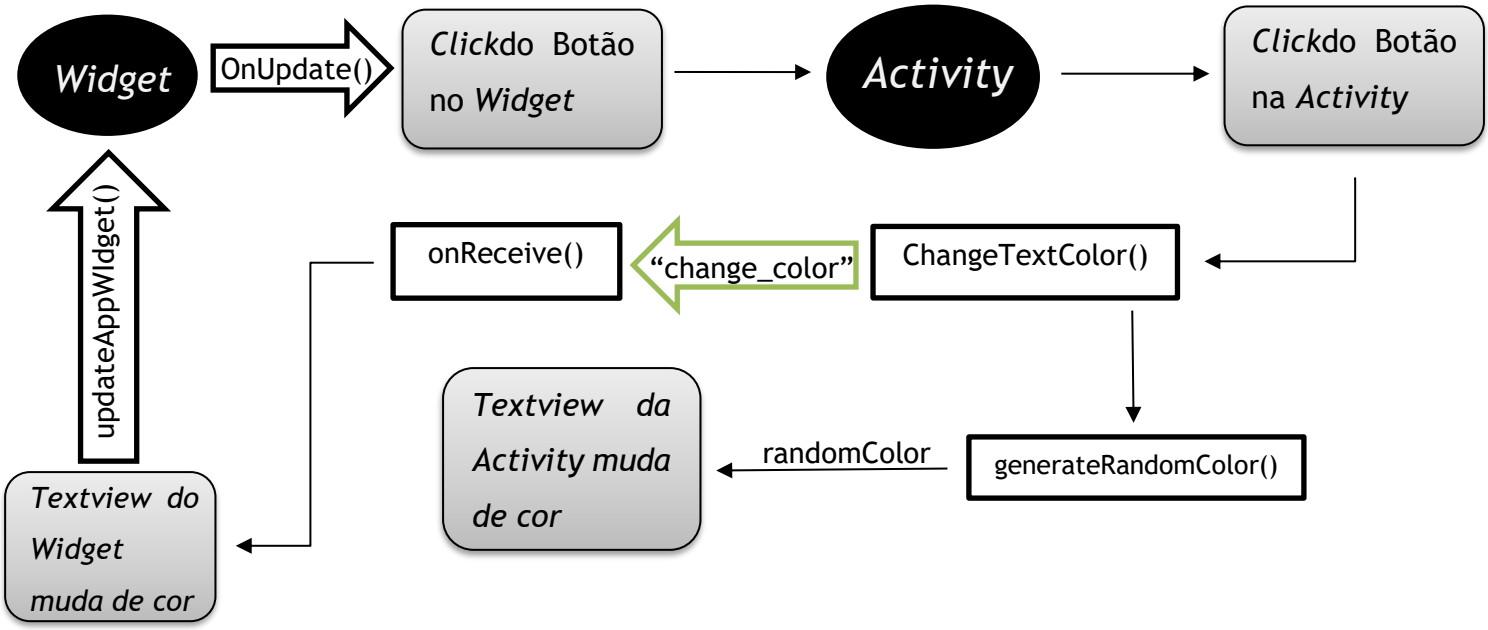
Após declarar a ação, basta programar o que é que ela vai desencadear no *widget*:

```
public void onReceive(Context context, Intent intent) {
    super.onReceive(context, intent);

    String action = intent.getAction();
    if (action.equals("change_color")) {

        //change text color
        RemoteViews views2 = new RemoteViews(context.getPackageName(),
                R.layout.hello_world_w);
        views2.setTextColor(textViewWidget, generateRandomColor());
        //update widget
        AppWidgetManager mManager = AppWidgetManager.getInstance(context);
        ComponentName cn = new ComponentName(context, HelloWorldW.class);
        mManager.updateAppWidget(cn, views2);
    }
}
```

Desta forma, é mais fácil programar qualquer ação externa que seja preciso fornecer ao *widget*. Se houvesse mais ações, com condições de controlo de fluxo é simples direcionar o código para a ação enviada. Neste caso, criei uma ação chamada "change_color". Primeiramente, atribuir uma cor aleatória para a *textview*do *widget*. Daí eu ter criado o método "generateRandomColor()". Depois foi só preciso atualizar o *widget*.

E assim, consegui criar uma aplicação simples e básica, que englobou conceitos simples mas essenciais para desenvolver mais aplicações no futuro, nomeadamente a aplicação escolar.

Em síntese, e de maneira a resumir e globalizar o software programado:



6. Conclusões

Neste projeto introduzi-me ao mundo do desenvolvimento de aplicações móveis. Tudo começou numa ideia, cujo potencial não era aproveitado se não fosse traduzida para uma *app*. Mas para se contruir uma casa não se pode começar pelo teto, sem saber muito bem o que se está a fazer, e o mesmo se aplica para o *app developing*. Se eu ia programar uma aplicação móvel, com o intuito de ser lançada no meio escolar e com o grande objetivo de melhorar a vida de cada aluno da ESDICA, teria que, em primeiro lugar, programar uma aplicação simples e básica.

No final, consegui uma aplicação funcional, descomplicada e cujo desenvolvimento abrangeu as questões inicias e essenciais para o desenvolvimento de qualquer aplicação. Posso, portanto, afirmar com certeza que, quando começar a desenvolver a aplicação-alvo já sei que abordagem vou utilizar, como vou programar o design, onde e como a testar e muito mais bagagem que me vai, como era o pretendido, poupar tempo e trabalho.

7. Reflexão Final

Este projeto fez-me chegar à conclusão que, pelo menos no que toca à área de programação, com vontade e persistência podemos fazer praticamente tudo. Comecei este ano letivo sem qualquer *background* em programação e, em poucos meses, já ajudei a desenvolver um simulador físico, razoavelmente bom, e desenvolvi uma Android *app*. O método é sempre o mesmo. Programar, testar, se falhar pesquisar e repetir. E a parte mais interessante e importante deste ciclo é que qualquer pessoa o consegue fazer, basta, lá está, ter vontade, ambição e sobretudo nunca desistir.

Daí, e é sobretudo a principal mensagem que quero transmitir com este projeto, se a pessoa quiser, e não é nada do outro mundo como muitas vezes possa parecer, é sim possível mudar o mundo, que seja, no mínimo, o nosso mundo.

8. Referências

1. https://www.xamarin.com/ - Consultado no início, aquando a tentativa de desenvolver com o Xamarin.
2. https://cordova.apache.org/
3. https://www.w3schools.com/
4. http://microsoft.github.io/ace/

2, 3 e 4 Consultados aquando o desenvolvimento através do Apache Cordova com o plugin Project Ace.

5. https://jsfiddle.net/
6. https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

5 e 6 Consultados para testar HTML, CSS e JavaScript na Web.

7. https://developer.android.com/studio/index.html - Consultado para esclarecer dúvidas sobre o Android Studio e como programar para Android.
8. http://stackoverflow.com/ - Consultado ao longo do projeto para esclarecer as mais variadas dúvidas sobre programação.

9. Anexos

9.1. Código-fonte

Código do layout da *activity*:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_hello_world"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.example.v_user.modernhelloworld.HelloWorld">

<LinearLayout
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
tools:ignore="UselessParent">

<TextView
android:text="@string/appwidget_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:id="@+id/textView"
android:textSize="50sp"
android:textStyle="normal|bold|italic"
android:gravity="center"
android:textColor="@color/colorPrimaryDark" />

<Button
android:text="@string/button_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/button"
android:textSize="20sp"
android:onClick="ChangeTextColor"
android:background="@color/colorPrimaryDark"
android:textColor="@color/colorPrimary" />
</LinearLayout>
</RelativeLayout>
```

Código do layout do *widget*:

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/colorPrimary"
android:padding="@dimen/widget_margin">
<LinearLayout
android:id="@+id/linear_layout"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:background="@color/colorPrimary"
android:orientation="vertical"
android:padding="5sp"
android:gravity="center"
android:layout_alignParentTop="false"
android:layout_centerInParent="true"
android:layout_centerVertical="false">

<TextView
android:id="@+id/textViewWidget"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="center"
android:text="@string/appwidget_text"
android:textColor="@color/colorPrimaryDark"
android:textSize="40sp"
android:textStyle="normal|bold" />

<Button
android:id="@+id/button_widget"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/colorPrimaryDark"
android:text="@string/widget_text"
android:textColor="@color/colorPrimary"
android:textSize="22sp" />
</LinearLayout>

</RelativeLayout>
```

Código Java da *Activty*:

```
package com.example.v_user.modernhelloworld;


import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


import java.util.Random;
import static com.example.v_user.modernhelloworld.R.id.textView;


public class HelloWorld extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_hello_world);


   }


   public static intgenerateRandomColor() {
intrandomColor;
       Random rand = new Random();
       //generates random color
int r = rand.nextInt();
int g = rand.nextInt();
int b = rand.nextInt();
randomColor = Color.rgb(r, g, b);
```

```
        return randomColor;


    }
    public void ChangeTextColor(View view) {
        //get textview
TextViewtv = (TextView) findViewById(textView);
        //set random color to textivew
tv.setTextColor(generateRandomColor());
        //tell widget to change color too
        Intent intent = new Intent();
intent.setAction("change_color");
sendBroadcast(intent);
    }
}
```

Código Java do *Widget:*

```
package com.example.v_user.modernhelloworld;


import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.widget.RemoteViews;


import static com.example.v_user.modernhelloworld.HelloWorld.generateRandomColor;
import static com.example.v_user.modernhelloworld.R.id.textViewWidget;


/**
```

```
 * Implementation of App Widget functionality.
 */
public class HelloWorldW extends AppWidgetProvider {

   static void updateAppWidget(Context context, AppWidgetManagerappWidgetManager,
intappWidgetId) {

       // Construct the RemoteViews object
RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.hello_world_w);
       // Instruct the widget manager to update the widget
appWidgetManager.updateAppWidget(appWidgetId, views);
   }
   @Override
   public void onUpdate(Context context, AppWidgetManagerappWidgetManager, int[]
appWidgetIds) {
       // There may be multiple widgets active, so update all of them
       for (intappWidgetId : appWidgetIds) {
updateAppWidget(context, appWidgetManager, appWidgetId);

           // Create an Intent to launch Activity
           Intent intent = new Intent(context, HelloWorld.class);
PendingIntentpendingIntent = PendingIntent.getActivity(context, 0, intent, 0);
           // Get the layout for the App Widget and attach an on-click listener to the button
RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.hello_world_w);
views.setOnClickPendingIntent(R.id.button_widget, pendingIntent);
           // Tell the AppWidgetManager to perform an update on the current app widget
appWidgetManager.updateAppWidget(appWidgetId, views);
//
      }
```

```
    }
    @Override
    public void onEnabled(Context context) {
        // Enter relevant functionality for when the first widget is created
    }

    @Override
    public void onDisabled(Context context) {
        // Enter relevant functionality for when the last widget is disabled
}
    @Override
    public void onReceive(Context context, Intent intent) {
super.onReceive(context, intent);
        String action = intent.getAction();
        if (action.equals("change_color")) {

            //change text color
RemoteViews views2 = new RemoteViews(context.getPackageName(),
R.layout.hello_world_w);
            views2.setTextColor(textViewWidget, generateRandomColor());
            //update widget
AppWidgetManagermManager = AppWidgetManager.getInstance(context);
ComponentNamecn = new ComponentName(context, HelloWorldW.class);
mManager.updateAppWidget(cn, views2);
        }
    }
}
```