1 – build my first app

Facil para criar, porem, não conseguimos emular um dispositivo real nas opções do desenvolvedor do aparelho(develeop options> settings).

* Para resolver fomos em numero de compilação (build number), localizado no smartphone e tocamos na tela 7 vezes.

Conseguimos rodar o real device(fazer o select deployment target)

Para criar uma interface não tivemos nenhum problema, pois ele usa os views que já estao todos pre definidos.

* Na hora em que fomos adicionar um EdiText o programa deu erro no @string/editmessage

Para corrigir o erro do EditText fomos em res>values>string.xml e adicionamos uma string para o editmessage

Em seguida colocamos um botao de enviar, depois disso, so fizemos dar um “run” no app e funcionou perfeitamente.

Colocamos um intent para criar uma nova atividade(activity)

Obs – apareceu alguns erros no mainActivity, mas para resolve-lo, importamos algumas classes para o mesmo.

Essa activity foi criada para um DisplayMessage(mostrar a mensagem)

2- tivemos dificuldade para achar a pasta das linguagens, mas para criar foi de boas, o interessante foi que quando criamos ele cria o arquivo xml com a bandeira do país.

Pra utilizar a string criada vc tem q referenciar ela dentro de qualquer arquivo xml ou do arquivo main do seu projeto q ele pegará a sua determinada string.

AO CRIAR um arquivo layout ele abre o aplicativo redimensionando a tela do aparelho de acordo com o tamanho da sua tela

Voce cria pastas para colocar imagens em diferentes resolucoes para varios tipos de tela

Tem q especificar os requisitos minimos para o app rodar em cada tipo de sdk.

Usar estilos de plataformas diferentes

Pra usar um estilo deve-se colocar o seguinte codigo <activity android:theme="@android:style/Theme.Dialog"> e para usar em uma application inteira usa-se o <application android:theme="@style/CustomTheme">

3 - Managing the activity lifecycle

Na parte do ciclo de vida aprendemos a utilizar as activitys , como gerar uma nova activity e como fazer a nova activity ser por exemplo a tela principal do nosso projeto, também sobre as funções oncreate,onresume e onstop, além de criar uma tela aprendemos a excluía la através do ondestroy, para podermos rodar um aplicativo e utilizar a câmera pegamos a função onpause e chamamos a câmera , assim que voltamos ao aplicativo utilizamos o onresume, aprendemos também a parar um aplicativo e depois reinicialo com as funções onstop e onstart ou o onresume e por final aprendemos que quando rodamos um aplicativo e damos um finish() , nos criaremos uma classe onSaveInstanceState() para que ele salve esta atividade sem que ela seja destruída seja perdida, já para reconstruir temos a função onRestoreInstanceState() e oncreate() fazendo com que a tela volte.

4- bulding a dynamics UI with Fragments

Já na parte de criar ui com fragmentos , ele começa a utilizer alguns recursos do ciclo de vida da atividade, esses fragmentos são como um parente que vão estar utilizando outro fragmento para criar telas de diferentes tamanhos, para se construir uma tela flexível criaremos outro fragmento agora na pasta layout não mais na layout-large , para que ele se adeque ao tamanho da do dispositivo que esta sendo utilizado , depois de colocar os fragmentos vamos chamo de dentro da atividade, podemos substituir um fragmento de uma tela por outro usando o método replace(), podemos também definir uma interface e entregar uma mensagem através destes fragmentos.

5 – saving data

Neste passo aprendemos o básico de como gravar e ler preferencias compartilhadas.

Nois pegamos dois métodos um getPreferences() para gravar arquivos de uma atividade sendo que este só precisa de um arquivo de preferencia compartilhada para a atividade, já o

[getSharedPreferences()](https://developer.android.com/reference/android/content/Context.html#getSharedPreferences(java.lang.String, int)) serve para métodos que pegam vários arquivos de preferencia compartilhada com nome,

Para obtermos acesso ao armazenamento externo tivemos que solicitar a permissão no arquivo de manifesto (WRITE\_EXTERNAL\_STORAGe)

Para salvar arquivos internamente podemos usar os métodos getFilesDir() representando um diretório interno e o getCacheDir() como o próprio nome já diz pega os arquivos temporários ou arquivos em cache.

Podemos também criar um arquivo novo instanciando um new file ou usando o método openfileoutput() para gravar um arquivo no diretório interno ou podemos também criar arquivos temporários com o createtempfiles(), para salvarmos um arquivo externamente usamos o [getExternalStorageState()](https://developer.android.com/reference/android/os/Environment.html#getExternalStorageState()), estes arquivos podem ser públicos ([getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html" \l "getExternalStoragePublicDirectory(java.lang.String)))  ou privados([getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String)) ).

Pode-se consultar arquivos vazios [getFreeSpace()](https://developer.android.com/reference/java/io/File.html#getFreeSpace()) ou [getTotalSpace()](https://developer.android.com/reference/java/io/File.html" \l "getTotalSpace()) e também excluir arquivos com o metode delete().

Também podemos criar um banco de dados utilizando o sqlliteopenhelper, além de utilizar algumas funções do banco como db.update, db.delete e db.insert.

6- [Interacting with Other Apps](https://developer.android.com/training/basics/intents/index.html)

Como foi feito e explicado na primeira parte nos temos as atividades , nas quais podemos chamar outras atividade ou aplicativos ou encaminhar para outros aplicativos.

Pode criar uma intenção implícita que vai executar uma ação seja um mapa ,calendário , pagina web e etc, temos que confirmar se há realmente um aplicativo para receber a intenção usando o comando  queryIntentActivities() para receber a lista de atividade que podem receber a intenção podemos também usar o isIntentSafe se o aplicativo respondera a intenção, podemos iniciar a atividade com intenção através do comando startactivity(intent).

Para obtermos um resultado da atividades teremos que usar o comando [startActivityForResult()](https://developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent, int)) para iniciar e depois devolver o método  [onActivityResult()](https://developer.android.com/reference/android/app/Activity.html" \l "onActivityResult(int, int, android.content.Intent)) para exibir o resultado da atividade

Para que outros aplicativos acessem o nosso temos que adicionar um intente filter que respondera as intenções, podemos responder esta intenção com o métodos oncreate e onstart

Fornecendo assim uma ação e para o resultado ser exibido so usaremos o método setresult().

7- [Working with System Permissions](https://developer.android.com/training/permissions/index.html)

As declarações de permissões para o app devem ser feitas dentro do arquivo manifest.

Temos as pesmisões por tempo de execução em que nelas poderemos checar as permissões com o método [ContextCompat.checkSelfPermission()](https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#checkSelfPermission(android.content.Context, java.lang.String)), utilizamos também o método

shouldShowRequestPermissionRationale() para vermos as permissões que foram feitas, utilizando o método requestpemissions() você poderá ver as permissões que será realmente necessário.

Na parte de permissões a única dificuldade foi entender como sera executado em tempo de execução.