

Workshop Boas Práticas em multiplos ecrãs

|  |  |
| --- | --- |
| **Curso(s):** | Mestrado em Computação Móvel |
| **Unidade(s) Curricular(es):** | Seminário I |
| **Ano Letivo:** | 2020/2021 |
| **Docente:** | Carlos Carreto |
| **Data:** | 30/01/2021 |

Índice

[Introdução 1](#_Toc62889539)

[Visão Geral 2](#_Toc62889540)

[Componentes 7](#_Toc62889541)

[Recursos 8](#_Toc62889542)

[Exemplos de Aplicações 9](#_Toc62889543)

[Exemplo Prático 11](#_Toc62889544)

[Conclusão 15](#_Toc62889545)

Índice de Figuras

[Figura 1 - Constraint Layout 3](#_Toc62832322)

[Figura 2 - Wrap Content e Match Parent 4](#_Toc62832323)

[Figura 3 - Layouts alternativos 4](#_Toc62832324)

[Figura 4 - Nine-patch bitmaps 5](#_Toc62832325)

[Figura 5 - dp e sp 5](#_Toc62832326)

[Figura 6 - várias versões do bitmap 6](#_Toc62832327)

[Figura 7 - Imagens vetoriais 6](#_Toc62832328)

Introdução

A evolução tecnológica na área de dispositivos móveis tem sido cada vez mais elevada, existem cada vez mais utilizadores que possuem dois a três dispositivos, cada um com as suas próprias propriedades, tamanhos de ecrãs, densidade de pixéis, *multi-window*, entre outros componentes distintos. Derivado destas características, os programadores têm de se adaptar quando desenvolvem aplicações, isto porque um *developer* não vai desenvolver uma *app* para cada tipo de dispositivo, mas sim tentar desenvolver uma aplicação que seja viável, prática, tanto aos diferentes tamanhos de ecrã como a própria densidade de pixéis sem perder nada na experiência de utilizador.

Existem boas práticas que podem ser adotadas na fase de desenvolvimento da *app* relativamente aos tipos de ecrãs dos dispositivos*,* tais como:

1. Layouts flexíveis ou alternativos;
2. Imagens extensíveis;
3. Independência da densidade de pixéis;
4. *Bitmaps* alternativos;
5. Imagens vetoriais;
6. Incompatibilidade de ecrãs

O conhecimento e aplicação destas boas práticas é necessária visto que a interface e a flexibilidade da aplicação entre dispositivos é um dos primeiros fatores que um utilizador irá avaliar ao usar a aplicação.

Neste workshop serão abordadas todas estas boas práticas, desenvolvendo e exemplificando cada uma delas, de seguida será feita uma demonstração numa aplicação android onde se irá corrigir diversos erros implementados com base nas boas práticas.

Fazer um apanhado das boas praticas que a google tem com a playstore (boas praticas da google; objetivo e motivação das boas praticas).

Para tal todo o material usado no workshop encontra-se disponível no link:

<https://github.com/manzarra/Seminario.git>

Visão Geral

A tabela x irá exibir as diferentes boas práticas juntamente com uma breve explicação sobre cada uma bem como um exemplo.

|  |  |  |
| --- | --- | --- |
| Boa prática | Explicação | |
| Usar *Constraint Layout* ou RecyclerView | | Este layout permite especificar a posição e tamanho de cada *view* baseado nas relações espaciais com outras *views* no mesmo *layout* |
| Usar os valores *wrap*\_*content* e *match*\_*parent* | | *Wrap\_content* diz a *view* para definir o tamanho para o que achar necessário para o conteúdo automaticamente. *Match\_Parent* faz a *view* expandir o máximo possível dentro da *view* principal |
| Usar layout alternativos | | Uma app deverá providenciar layout alternativos otimizados para certos tamanhos de ecrã, usar largura qualificada como mais pequena ou largura disponível, usar qualificadores de modo *portrait* e *landscape* |
| Usar *nine-patch bitmaps* | | Trata-se de uma imagem PNG standard, mas com uma margem de mais 1 pixel em toda a volta que indica quais pixéis da imagem podem ser esticados |
| Testar a app em todos os ecrãs | | Certificar que a app escala bem nos diferentes tamanhos |
| Declarar tamanhos de ecrã específicos no suporte da app | | Maneira de outras apps (googleplay) poderem saber se podem recomendar a app ao dispositivo de onde está a ser acedido |
| Usar dp e sp | | Dp trata-se de uma unidade virtual de pixel, dizer 1dp é diferente de dizer 1px e o impacto que pode ter numa app pode ser tremendo. Sp é o mesmo que dp só que é virado para textos |
| Providenciar múltiplas versões do mesmo bitmap na app | | Ter várias versões de ícones disponíveis para a app escolher |
| Usar imagens vetoriais | | Mais indicado para icons, não tanto fotografias |
| Testar a app em todas as densidades de pixéis | | Certificar que a app escala bem nas diferentes densidades |
| Definir variáveis globais | | Evitar variáveis locais visto que em eventuais correções, leva muito mais tempo e não são tão precisas |

De seguida será exibido um exemplo para cada uma das boas práticas referidas na tabela:

* Usar *Constraint* *Layout* ou *Recycler View*

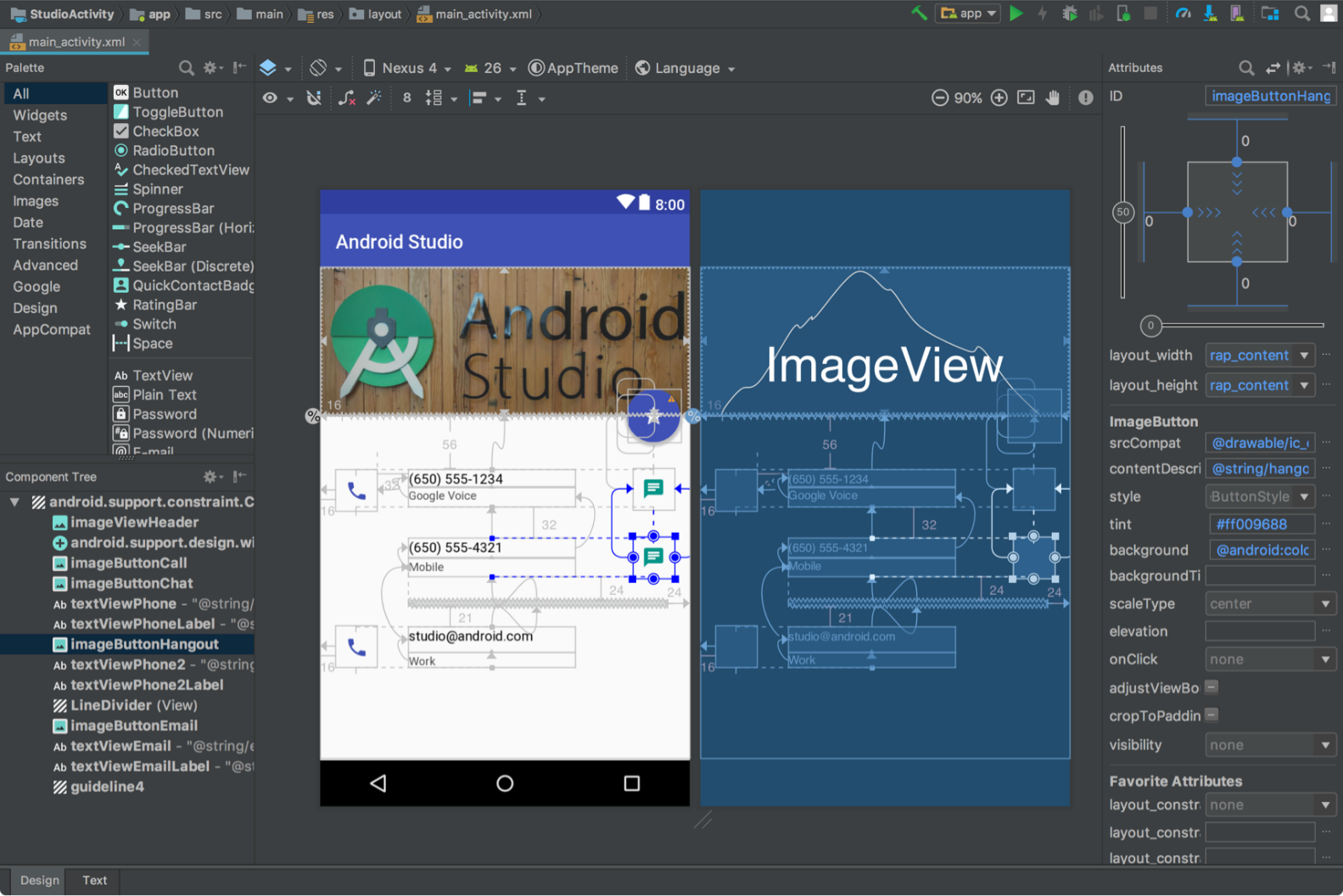


Figura 1 - Constraint Layout

* Usar os valores *wrap*\_*content* e *match*-*parente*



Figura 2 - Wrap Content e Match Parent

* Usar *Layouts* alternativos

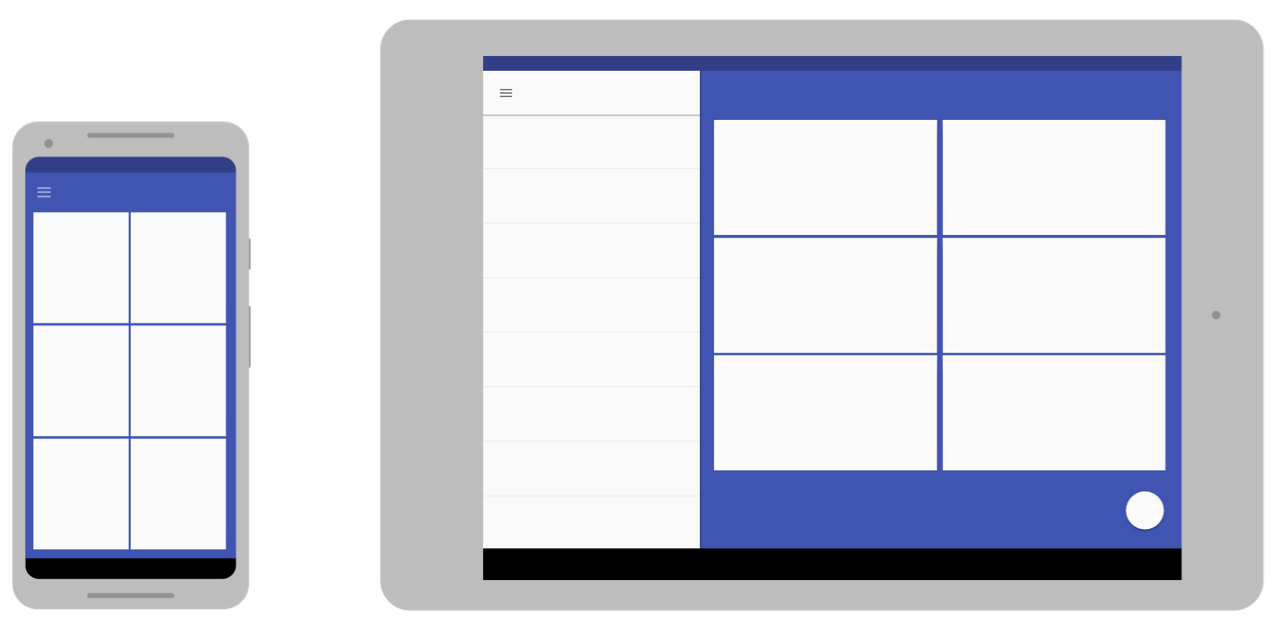


Figura 3 - Layouts alternativos

* Usar *nine*-*patch* bitmaps

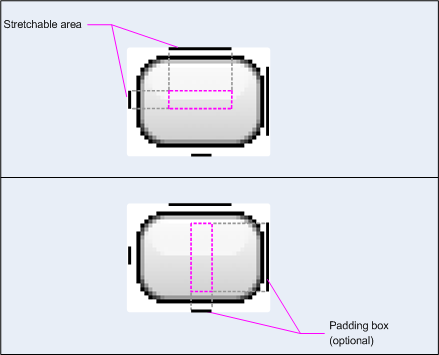


Figura 4 - Nine-patch bitmaps

* Usar dp e sp

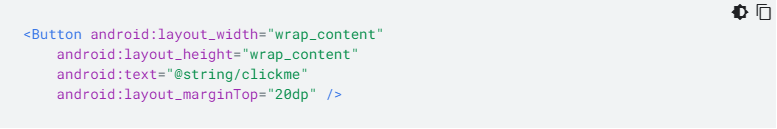




Figura 5 - dp e sp

* Providenciar múltiplas versões do mesmo bitmap

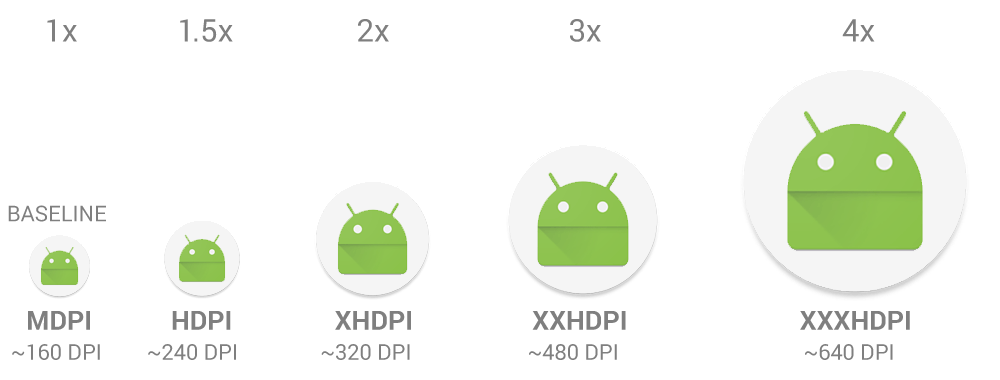


Figura 6 - várias versões do bitmap

* Usar imagens vetoriais

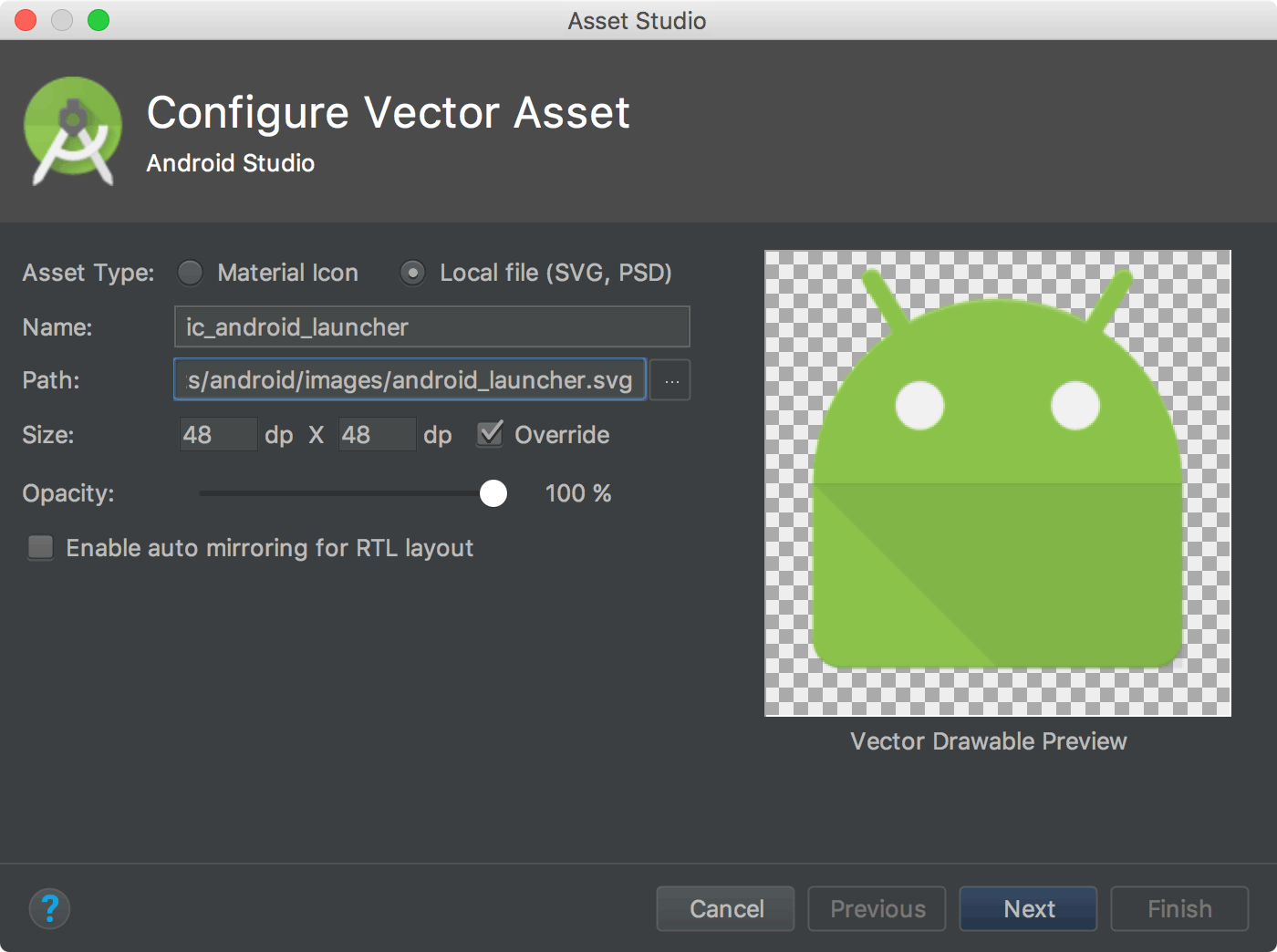


Figura 7 - Imagens vetoriais

## Componentes

Das boas práticas referidas, aquelas que são mais importantes e que devem ser aplicadas em todas as aplicações são:

* Boa prática de usar *constraint* layout e a *recycler* *view*, visto que usar outros tipos de *view* pode exigir mais recursos e processamento por parte do dispositivo, ou seja, mais gasto de memória e de bateria.
* Boa prática de usar o *wrap\_contente* e o *match*\_*parent,* devido ao facto que a boa resolução em vários ecrãs é completamente impossível sem usar estas duas váriaveis.
* Boa prática de testar a aplicação em diversos dispositivos, importante em todos os tipos de contexto

## Recursos

Suporte a diferentes tipos de ecrã –

<https://developer.android.com/training/multiscreen/screensizes#TaskUse9Patch>

Visão geral dos recursos das aplicações - <https://developer.android.com/guide/topics/resources/providing-resources#java>

Suporte a diferentes densidades de pixéis –

<https://developer.android.com/training/multiscreen/screendensities#java>

Suporte de múltiplas janelas - <https://developer.android.com/guide/topics/ui/multi-window>

Restrições de ecrã - <https://developer.android.com/guide/practices/screens-distribution>

Visão geral de compatibilidade de ecrã - <https://developer.android.com/guide/practices/screens_support.html>

*Smartwatch interface design* -<https://uxdesign.cc/your-guide-to-smartwatch-interface-design-designing-for-all-1a588a6a1181>

*Watch apps interface guidelines* - <https://usabilitygeek.com/7-user-interface-guidelines-for-designing-watch-apps/>

*Mobile* *vs* *Tablet* *Apps* - <https://ugem.design/blog/mobile-and-tablet-apps-ui-design>

*Design* *for* *smarttv* - <https://marvelapp.com/blog/designing-for-television/>

*Design for cars* - <https://developers.google.com/cars/design>

Exemplos de Aplicações

Nos dias atuais, não é só para ecrãs de tablet ou de telemóvel que as aplicações são desenvolvidas, atualmente, outros tipos de dispositivos movéis são também muito usados e requerem aplicações novas e funcionais para trabalhar, exemplos desse tipo de dispositivos são os *smartwatch*, *smarttv* e os ecrãs inseridos nos carros.

Começando pelos *smartwatch*, uma das características que tem de ter é uma navegação baseada em gestos, ou seja, como o ecrã é tão pequeno, um simples clique com o dedo tem de ter algum tipo de interação, isto tem de ser levado em conta para a interface da aplicação. Por exemplo, o deslizar para o lado direito terminar a aplicação em vez de o layout ter um botão para voltar atrás e assim ficando com espaço disponível para outro contexto da app. Outra implementação que tem de existir é um método fácil de alternar entre aplicações, permitindo uma navegação fácil pela interface. É necessário ainda limitar as ações que podem ser executadas no ecrã de maneira a que os utilizadores possam fazer a ação certa no momento certo. Outras implementações necessárias para *smartwatch* passam por evitar funções complexas e hierarquias complexas, permitir o recuperar de ações não pretendidas, respeitar margens e o preenchimento das aplicações bem como os botões e cores usados. Por fim uma implementação adicional passa por criar aplicações que não sejam exaustivas ao utilizador, por exemplo, não ter vibrações por cada Acão que é realizada na aplicação.

Por sua vez os *tablets* e o *smartphone*, apesar de muito parecidos são usados de maneiras distintas o que faz com que as aplicações para cada tipo tenham de ser específicas também. Isto não quer dizer que uma aplicação de *smartphone* não funcione no *tablet* da mesma maneira, mas tudo depende do motivo para o qual a aplicação é desenvolvida. Uma aplicação para *tablet*, pode ser desenvolvida tanto baseando-se em aplicações de telemóvel, ou baseada em *desktop web aplications.* Outras características que marcam a diferença entre as aplicações é o tempo de uso dispensado entre cada dispositivo, as funcionalidades que cada dispositivo tem e até a própria maneira como o utilizador interage entre cada um, que é completamente diferente. Por fim, uma interface de tablet não deve copiar a interface de uma aplicação de telemóvel, mas também não deve ser completamente diferente ao ponto de não ter as mesmas funcionalidades que a aplicação do *smartphone* e tem de ser sempre levada em atenção a diferença de ecrã que no caso dos *tablets* por ser maior, leva a que possa ser exibida muito mais informação. Um exemplo de uma aplicação que não está otimizada para os tablets é o *Instagram*.

A *smarttv* ainda é algo novo e em desenvolvimento, não existindo um *design* definido como base para este dispositivo, os únicos aspetos a considerar são que uma *smarttv* geralmente terá algum tipo de controlo para aceder, a limitação de cor que geralmente é sempre inferior à de um computador e que toda a informação é geralmente exibida em listas.

Para terminar, as *apps* para carros, apesar de muito parecidas em termos de tamanho de ecrã com os tablets, tem um conjunto de características muito parecidas com o *smartwatch*, visto que um utilizador deste dispositivo está em movimento e a atenção que pode disponibilizar ao aparelho é muito reduzida. Assim, algumas características que têm de existir são que o conteúdo exibido tem de ser bem legível e vislumbrável, com botões grandes e fáceis de identificar. O texto nestes dispositivos tem de ser limitado de modo a não exigir muita atenção ao utilizador, botões separados de modo a evitar o toque em múltiplas ações, os elementos da interface têm de ser consistentes. Por fim, em condução as ações ativas e ações de background têm de ser claras e definidas.

Exemplo Prático

Link do github : <https://github.com/manzarra/Seminario.git>

Uso do weightsum no linear layout (Resolvido)

<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:weightSum="10">  
  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_height="0dp"  
 android:layout\_width="match\_parent"  
 android:layout\_weight="3"  
 android:text="Aqui será testado o Linear Layout" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_height="0dp"  
 android:layout\_width="match\_parent"  
 android:layout\_weight="3"  
 android:text="O linear layout também tem maneiras praticas de se conseguir manter o layout otimizado para vários ecrãs" />  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_height="0dp"  
 android:layout\_width="match\_parent"  
 android:layout\_weight="3"  
 android:text="A este metodo chama-se weightSum, e basicamente dividimos o layout em partes( neste caso 10) onde damos depois um peso a cada componente do layout, isto funciona tanto em horientação vertical como horizontal" />  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_height="0dp"  
 android:layout\_width="match\_parent"  
 android:layout\_weight="1\*"  
 tools:srcCompat="@tools:sample/avatars" />  
</LinearLayout>

Boa prática de variáveis globais (tentativa 1)

<TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Aqui é onde vamos criar as variaveis globais"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:layout\_editor\_absoluteX="67dp" />  
  
<TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Exercicio - Alterar os valores a seguir para +1 e -1 sequencialmente"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="12"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView2"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="13dp" />  
  
<TextView  
 android:id="@+id/textView4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="7"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView3"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView5"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="25"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView4"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView6"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="7890"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView5"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView7"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="75"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView6"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView8"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="90"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView7"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView9"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="500"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView8"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView10"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="10000"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView9"  
 tools:layout\_editor\_absoluteX="-16dp" />

Boa prática de variáveis globais (tentativa 2)

<TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Aqui é onde vamos criar as variaveis globais"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:layout\_editor\_absoluteX="67dp" />  
  
<TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Exercicio - Alterar os valores a seguir para +1 e -1 sequencialmente"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global1"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView2"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="13dp" />  
  
<TextView  
 android:id="@+id/textView4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global2"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView3"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView5"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:text="@string/Global3"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView4"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView6"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global4"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView5"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView7"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global5"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView6"  
 tools:layout\_editor\_absoluteX="0dp" />  
  
<TextView  
 android:id="@+id/textView8"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global6"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView7"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView9"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global7"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView8"  
 tools:layout\_editor\_absoluteX="-16dp" />  
  
<TextView  
 android:id="@+id/textView10"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/Global8"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView9"  
 tools:layout\_editor\_absoluteX="-16dp" />

Como inserir uma imagem 9.png. Este vídeo faz um exemplo muito bom e rápido para se entender como funciona

Link: <https://www.youtube.com/watch?v=FZEPg-8tJgo&t=222s&ab_channel=CodinginFlow>

Conclusão

As boas práticas referidas ao longo do workshop são ideias ou metodologias aconselhadas aos *developers* de modo a proporcionar uma melhor experiência de desenvolvimento, bem como um melhor desempenho para os utilizadores que irão usar a aplicação.

Estas boas práticas também mostram que desenvolver uma aplicação não passa só por ter atenção no backend, considerando até que um utilizador ao usar a aplicação só vai interagir com o frontend (layout) é de extrema importância tentar seguir estas metodologias e ideias já estudadas ou desenvolvidas para melhorar a *user* *experience*.

Outra consideração, uma pessoa ter múltiplos dispositivos móveis é uma realidade do dia a dia, o que leva a que saber programar ou desenvolver para vários ecrãs é cada vez uma realidade maior para *developers* de dispositivos móveis. Mais ainda porque os dispositivos atuais suportam atualmente a opção de *multi-window* o que faz que até mesmo num pequeno ecrã, possam existir 2 ou 3, até mesmo mais aplicações abertas em simultâneo.

Para terminar, podemos concluir que a utilização das boas práticas proporciona um melhor desempenho quando aplicadas.