Interface Gráfica (Parte 1)

- User Interface Classes

Referências

- DEITEL, Abbey, DEITEL, Paul, DEITEL, Harvey. Android para programadores: uma abordagem baseada em aplicativos. 2 ed. Porto Alegre: Artmed. 2015.
- Programming Handheld Systems – Prof. Adam Porter of University of Maryland (https://www.coursera.org/course/android)

- http://developer.android.com
- https://developers.google.com/university/courses/mobile
- http://oreilly.com/training/androidapps

View

- Bloco chave para construção de interface com usuário
- Ocupa um espaço retangular na tela
- Responsável por sua elaboração (desenho) e manipulação de eventos

Views pré-definidas

- Button
- ToggleButton
- CheckBox
- RatingBar
- AutoCompleteTextView

Operações comuns de *Views*

- `Set Visibility`: exibe ou esconde a *view*
- `Set Checked State`
- `Set Listeners`: código que deve ser executado quando um determinado evento ocorre
- `Set Properties`: opacidade, fundo e rotação
- `Manage Input Focus`: permite a *view* receber ou requisitar o foco

Origem dos eventos de *Views*

- Interação com usuário
  - Touch
  - Keyboard / Trackball / D-pad
- Controle do sistema
  - Mudanças no ciclo de vida da aplicação

Tratamento de eventos de *Views*

- Normalmente é feito com *Listeners*
- Existem várias interfaces definidas na classe *View*

View Listener Interfaces

- `OnClickListener.OnClick()`
  - *View* foi clicada
- `OnLongClickListener.OnLongClick()`
  - *View* foi pressionada e segura

View Listener Interfaces

- `OnFocusChangeListener.OnFocusChange()`
  - *View* recebeu ou perdeu o foco
- `OnKeyListener.OnKey()`
  - *View* recebe o pressionamento de uma tecla

Exibição de *Views*

- *Views* são organizadas em uma árvore
- A exibição possui múltiplos passos:
  - *Measure*: obtém as dimensões de cada *View*
  - *Layout*: posicionamento de cada *View*
  - *Draw*: desenha cada *View*

Manipulando os eventos de *Views*

- `onMeasure()`
  - Determina o tamanho da *View* e dos seus filhos
- `onLayout()`
  - A *View* deve atribuir um tamanho e uma posição para todos os seus filhos
- `onDraw()`
  - A *View* deve “renderizar” seu conteúdo

Manipulando os eventos de *Views*

- `onFocusChanged()`
  - O estado do foco da *View* foi alterado
- `onKeyUp(), onKeyDown()`
  - Uma tecla do dispositivo foi pressionada
- `onWindowVisibilityChanged()`
  - O conteúdo que contém a *View* teve a seu status de visibilidade

ViewGroup

- É uma *View* invisível que contém outras *Views*
- Utilizado para agrupar e organizar um conjunto de *Views*
- Classe para comportar o conteúdo e o layout de *Views*

ViewGroups pré-definidos

- RadioGroup
- TimePicker
- DatePicker
- WebView
- MapView

Adapters & AdapterViews

- *AdapterViews* são *Views* com filhos gerenciados por um *Adapter*
- *Adapter* gerencia os dados e provê os dados da *View* para o *AdapterView*
- *AdapterView* exibe os dados das *Views*

ListView

- *AdapterView* exibe uma lista de rolagem com itens que podem ser selecionados
- Esses itens são gerenciados pelo *ListAdapter*
- Um *ListView* pode filtrar a lista de itens com base em um parâmetro de entrada

Spinner

- Um *AdapterView* que também provê uma lista de rolagem de itens
- O usuário pode selecionar um item da lista
- Os itens são gerenciados pelo *SpinnerAdapter*

Gallery

- Um *ViewGroup* que exibe um lista de rolagem de itens horizontalmente
- Os itens são gerenciados pelo *SpinnerAdapter*

Layouts

- Um *ViewGroup* genérico que define uma estrutura para conter outras *Views*

Tipos de *Layouts (1/2)*

- *LinearLayout*
  - *Views* filhas são organizadas em uma única linha horizontal ou vertical
- *RelativeLayout*
  - *Views* filhas ficam em uma posição relativa a outras *Views* ou a *View* pai

Tipos de *Layouts (2/2)*

- *TableLayout*
  - *Views* são organizadas em linhas e colunas
- *GridView*
  - *Views* filhas são organizadas em uma *grid* bidimensional que permite rolagem

Menus & ActionBar

- As *Activities* suportam menus
- Essas *Activities* podem
  - Adicionar itens ao menu
  - Manipular a seleção de itens do menu

Tipos de *Menus*

- *Options*
  - É exibido quando o usuário pressiona
- *Context*
  - Menu de uma *View* específica
  - Aparece quando o usuário toca e segura a *View*
- *Submenu*
  - Um menu ativado quando o usuário toca um item do menu

Criação de *Menus*

- Definir um recurso menu em um arquivo XML
  - Armazenado em `res/menu/filename.xml`
- Inflando o recurso menu utilizando o Menu Inflater nos métodos `onCreate...Menu()`
- Manipulando a seleção de item em um método `on...ItemSelected()`

Menus

- Outras características suportadas
  - Agrupamento de itens do menu
  - Atribuição de *shortcuts* para itens do menu
  - Atribuição de *intents* para itens do menu

ActionBar

- Similar às *Application Bars* de muitas aplicações convencionais
- Habilita um acesso rápido para operações comuns

FragmentDynamicLayout WithActionBar

- Exibe itens chave e uma área adicional de informação para o item selecionado
- Provê ações para a *ActionBar*
- Três objetos principais
  - `QuoteViewerActivity`
  - `TitleFragment`
  - `QuoteFragment`

ActionBar.Tab

- A tela é dividida em abas e áreas de conteúdo
- Permite múltiplos *Fragments* compartilhar uma única área de conteúdo

ActionBar.Tab

- Cada aba é associada com um *Fragment*
- Exatamente uma única aba é selecionada em um dado momento
- O *Fragment* correspondente a aba fica visível na área de conteúdo

- Vide `UITabLayout`

Dialogs

- É utilizado pelas *Activities* para promover a comunicação com o usuário
- *Dialog subclasses*
  - `AlertDialog`
  - `ProgressDialog`
  - `DatePickerDialog`
  - `TimePickerDialog`

TP1 (Parte 2/2)
Entrega em 26/09/2017 via SGA

1. Inserir uma nova atividade no TP que ofereça ao usuário a possibilidade de publicar o seu evento na sua conta *Facebook* e no *Linkedin*.

TP1 (Parte 2/2)
Entrega em 26/09/2017 via SGA

2. A app deve exibir mensagens de erro, quando for o caso (Dica: pacote `AlertDialog`)
3. A app deve seguir as melhores práticas de desenvolvimento de apps móveis