Aplicaciones móviles y el diseño de sus interfaces

Bianca Sozzi[a], Mercedes Muñoz[a], Oscar León[a], Julio Monetti[a], Mariana Brachetta[a]

[a]Dpto. Ing. en Sistemas de Información, Facultad Regional Mendoza - UTN, Rodriguez 273
C.P.: 5000, Mendoza, Argentina
{oleon,jmonetti}@frm.utn.edu.ar, {mbrachetta,bsozzi,mercedesmp1305}@gmail.com

Resumen. Al diseñar una aplicación, uno de los aspectos principales es la interface de usuario, ya que un mal diseño puede hacer su utilización ineficiente. El tema no es nuevo y existen muchas referencias en el ámbito del desarrollo de software. Sin embargo, adquiere mayor importancia cuando se trata de aplicaciones móviles, donde las características de usabilidad se tornan críticas, dadas las restricciones de tamaño de la pantalla y de medios para efectuar la entrada y salida de datos. En el artículo, se describen las características de las alternativas de interfaces que se han estudiado para esta clase de aplicaciones, y se analizan los aspectos considerados para definir la mismaen una aplicación móvil orientada a soportar un ambiente de aprendizaje ubicuo, en el cual se integran diversas tecnologíasy servicios disponibles en "la nube".

Palabras Clave: computación móvil –interfaces – software – servicios

1 Introducción

El crecimiento en la utilización del teléfono móvil, hace que se presente como una herramienta potencialmente adecuada para proponer a los estudiantes actividades de enseñanza. El presente trabajo se enmarca en un proyecto de investigación, para el desarrollo de un entorno basado en tecnología de cloud-computing, a fin de crear un ambiente orientado al aprendizaje ubicuo, para estudiantes del ingreso a la universidad que realizan el curso en forma no-presencial. Para implementar este aspecto se requiere del uso de dispositivos móviles, los cuales van a operar como medios de input/output de datos, lo que hace que el diseño de la interface de usuario resulte un aspecto crítico.

El caso concreto que se presenta está vinculado al desarrollo una aplicación (para dispositivos Android®) que debe integrar diversos servicios de la "nube", además del acceso a recursos existentes en un servidor remoto, para:

- Identificar la geo-localización del alumno, a fin de activar objetos de aprendizaje.
- Proveer vínculos de ayuda activados mediante el uso de realidad aumentada.
- Visualizar material de enseñanza.
- Interactuar con la plataforma Moodle para la realización de actividades de evaluación.
- Mantener la trazabilidad de las actividades realizadas y sus resultados.

Esencialmente la interface debía ser simple, minimizando la cantidad de objetos visuales mostrados en pantalla, además de ofrecer un modo de realizar los inputs que permiten acceder y controlar la aplicación, como también un medio donde visualizar los outputs que genera la misma.

2 Antecedentes

Las características particulares de los dispositivos móviles, como tamaño, resolución de pantalla, teclado, entre otras, son más restrictivas respecto de las PCs y notebooks; por lo que el diseño de la interface alcanza suma importancia. Las características de esta última dependerán de si se basa en páginas web visualizadas en un navegador [1], o si interactúa directamente con la aplicación [2].

El diseño de la interface para una aplicación móvil se centra en analizar principalmente dos aspectos: la forma de interactuar con la aplicación y en cómo se pretende que sea la experiencia para el

usuario. Lo anterior abarca presentar una visualización agradable, facilitar la explotación de funcionalidades y efectividad de uso. En esto intervienen aspectos que van desde el diseño gráfico, pasando por la organización de los elementos (layout), hasta la programación de los componentes.

Uno de los aspectos principales es la definición del modelo de acceso a las funcionalidades de la aplicación, siendo el esquema de menú el más conocido, que se hereda del ámbito de las computadoras de escritorio, donde en una barra se enumeran múltiples opciones disponibles, cada una de la cuales a su vez puede desplegar otras. En algunos casos se permite visualizar el "recorrido efectuado" al desagregar paneles con ítems, hasta alcanzar aquel que muestra en pantalla lo que se busca, en otros casos al elegir un ítem la barra de opciones es ocultada por otra.

Otras alternativas básicamente ofrecen la misma funcionalidad, ya que se presentan pantallas que pueden ser activadas al seleccionar la pestaña correspondiente. La variante en "sándwich" permite mostrar más información para cada ítem, al organizarlos en un conjunto de capas apiladas, cada una de las cuales conduce a otras pantallas. Otro modelo es el denominado "Hub de Navegación", donde en una página principal se visualizan todas las opciones de navegación. Sin embargo, requiere regresar al "hub" para dirigirse a otra sección. Esta variante resulta adecuada cuando para cada opción no se debe mostrar mucha información, siendo innecesario avanzar en niveles crecientes de exploración.

La restricción viene dada por el tamaño de la pantalla, que es mucho más pequeña que la de una computadora o notebook. Entonces por ejemplo más de cinco ítems en la barra, hace que sean difíciles de elegir tocándolos con un dedo. Una solución a esto pueden ser los esquemas de "carrousell", utilizando un conjunto de íconos que se deslizan en una cinta, un tambor o un dial.

Independientemente que el menú se disponga en la parte superior o inferior de la pantalla; se debe tener en cuenta que se debe destinar una parte de la misma para mantener la lista de opciones visible, además del contenido asociado a cada ítem del menú.

Este modelo de navegación mantiene oculta parte de la navegación en los submenús; entonces si el usuario no tiene una idea general de la organización, puede que se le dificulte encontrar lo que está buscando, por estar oculto en las regiones más profundas de la estructura del menú. No es extraño descubrir características de una aplicación, luego de utilizarla mucho tiempo, por haber estado estas localizadas en regiones que no se exploraron.

Actualmente algunas aplicaciones incorporan técnicas de inteligencia artificial, para ayudar al usuario a navegar esquemas complejos de menús.

Definir un esquema intuitivo, no deja de ser algo subjetivo, aunque existen algunas recomendaciones a tener en cuenta [3] [4], respecto de dos tipos de medios.

En el proyecto, una de las principales era reducir al mínimo la cantidad de interacciones necesarias por parte del usuario para lograr su objetivo.

Texto:

- Evitar el requerir que se escriba texto, si se pueden ofrecer otras alternativas, dado que puede ser hasta tres veces más lento su introducción, que en una PC.
- Cuidar la calidad del contraste, teniendo en cuenta los cambios en las condiciones de iluminación que se pueden dar durante el uso de dispositivos móviles.
- En las salidas evitar que se deba hacer "scroll", y de no existir alternativa, evitar superar una extensión mayor a tres veces el alto de pantalla.
- Las fuentes "serif" o similares, resultan adecuadas en cualquier resolución y además, se debe ofrecer opción de cambiar el tamaño.

Multimedia:

- En los gráficos, mantener una relación razonable entre tamaño en pixeles de la imagen y tamaño de pantalla.
- Si se incluyen sonidos o videos, se deben agregar controles de reproducción.

Dependiendo de la aplicación a desarrollar, a lo anterior se pueden agregar otros elementos de interface con el usuario, como los entornos basados en realidad aumentada [5].

2.1 Elementos de la interface Android®

La interfaz de usuario de Android está formada por dos clases de objetos View y ViewGroup: View que hace referencia a cualquier objeto visible en la pantalla con el cual el usuario puede

interactuar y ViewGroup relacionada con otros componentes View o ViewGroup, donde la definición de un esquema de vistas sigue una jerarquía (Fig1).

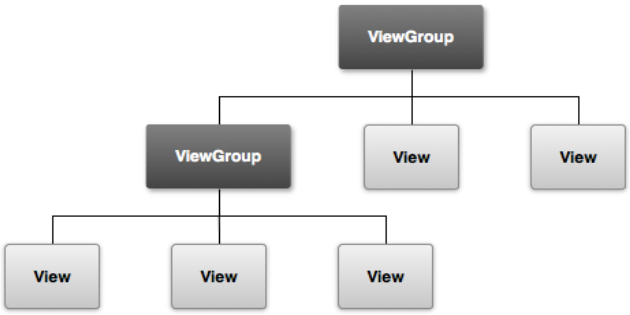

Fig1: jerarquía de vistas que define un diseño de UI**.**

Las View en Android® están formadas por dos tipos de elementos que definen las características de visualización en pantalla, algunos de los cuales fueron utilizados en el desarrollo del proyecto (Tabla 1). Como se puede observar presentan similitudes a las empleadas en las interfaces en aplicaciones de escritorio, salvando las restricciones en cuanto al tamaño de la pantalla.

Tabla 1. Elementos de visualización utilizados.

Organizativos	
LinearLayout	ubicación de elementos de forma horizontal o verticalmente
RelativeLayout	especificación de posición respecto de otros
ConstraintLayout	habilitación de un editor visual, al estilo Drag and Drop
SurfaceView	definición de área con características superiores a los objetos view
ScrollView	configuración del desplazamiento de scroll
TableLayout	agrupación de TableRows
TableRow	disposición horizontal de los descendientes de TableLayout
FrameLayout	distribución general de componentes en la pantalla
Funcionales	
TextView	definen cada elemento con el cual interactuará el usuario, para visualizar y realizar operaciones de input-output en la aplicación
Button	
ImageView	
ImageButton	
EditText	

Clase Activity:

La pantalla con la que los usuarios pueden interactuar se define a través de esta clase, creando una subclase que define la interface en particular, la cual puede escribirse utilizando XML, mediante la vinculación de ambos elementos (la clase y el archivo XML). Así por ejemplo sea una clase de nombre MainAcitivity, quedará vinculada a un archivo activity_main.xml.

Vistas:

Cada elemento de una vista debe instanciarse en el Activity correspondiente. Por ejemplo, si se quiere agregar funcionalidad a un botón se debe primero definir el elemento de tipo Button en el Activity y luego asignarle un onClickListener para poder especificar la funcionalidad deseada.

Atributos de Vista:

El **Id** identifica unívocamente una vista dentro de la jerarquía y permite instanciar dicho elemento desde el código de la Activity. Es obligatorio definir el ancho y alto del elemento dentro de la pantalla, usando los atributos Layout_width y Layout_height.Estos dos últimos elementos pueden tomar valores específicos o bien se puede aplicar wrap_content que “envuelve” el contenido; o match_parentque

hará que se expanda para ocupar tanto espacio como la vista padre permita. Esta característica de adaptabilidad del tamaño de los componentes a distintos tamaños de pantalla, mediante la gestión de tamaños relativos y no absolutos, es uno de los aspectos fundamentales para garantizar la adecuada visualización en una amplia gama de dispositivos móviles.
Algunas de las propiedades usadas para configurar la interface de la aplicación:

1. Extensión de un objeto respecto de un área.
   a. Padding: hace referencia a extender los límites del objeto dentro de ella.
   b. Margin: hace referencia a extender los límites del objeto fuera de aquella.
2. Valores que pueden tomar.
   a. Padding: define el padding sobre el elemento completo.
   b. Margin: define el margin sobre el elemento completo.
3. Para margin y padding pueden especificarse las siguientes características.
   a. Left: margin o padding hacia la izquierda.
   b. Top: margin o padding hacia arriba.
   c. Right: margin o padding hacia la derecha.
   d. Bottom: margin o padding hacia abajo.
4. Orientation (para LinearLayout): especifica si la orientación va a ser vertical u horizontal.
5. SP: Scale Independent pixels.
6. DP o DIP: Density independent pixels. Es una unidad abstracta de medida que se basa en la densidad de pixeles del dispositivo.
7. Text – textColor – fontFamily: permiten modificar las características de un elemento de texto, que también puede modificarse en tiempo de ejecución con el método setText.
8. Text-size: es el tamaño del texto, se utiliza la unidad de medida sp, que adapta el tamaño del texto al dispositivo.

3. Requerimientos de la interface

Para el proceso de creación de la interface, se siguieron pautas similares a las etapas de un proceso de desarrollo del software, como: relevar las características de los potenciales usuarios; identificar los requerimientos funcionales; definir el esquema de navegación; construir un prototipo para verificar el funcionamiento y la usabilidad. Para determinar el modelo de navegación se trabajó sobre variantes del modelo de "metáfora de descubrimiento", teniendo en cuenta principalmente dos aspectos:

- No abrumar al usuario con un cúmulo de opciones o de información.
- De fácil acceso y que ocupara poco espacio de pantalla.

Evidentemente lograr un equilibrio entre las consideraciones anteriores, depende del propósito de la aplicación, dado que, si bien no resultó nuestro caso, en otros la cantidad de elementos a mostrar (opciones de menú, vínculos a información, etc.) puede llegar a ser importante.
El objetivo es desarrollar un entorno ubicuo para enseñanza no presencial. La idea central es definir un "ambiente" donde el alumno acceda a diversos recursos, mediante herramientas de realidad aumentada (niveles 0, 1, 2). Para esto se combina el uso de posiciones geo-referenciadas, con objetos del mundo real, a fin de crear secuencias didácticas.
Se busca que el entorno ubicuo satisfaga características tales como accesibilidad, persistencia, disponibilidad, interactividad y localización. Para esto se definen puntos geográficos, donde se activan los planteos que los estudiantes deben investigar y solucionar. Por otra parte se crea un catálogo de objetos y marcadores de realidad aumentada, que ofrecen datos o información para abordar la resolución de las actividades propuestas. El guión de la secuencia didáctica involucra una red de puntos GPS, donde en cada punto se desarrolla una situación. El abordaje del desarrollo se basa en lo que se denomina programación extrema, dado que se está definiendo un prototipo donde se producen cambios de requisitos sobre la marcha.
La aplicación requiere considerar tres aspectos principales de diseño de la interface: los objetos 3D, la organización de la pantalla y el diseño del modo de interacción. Dado que una parte importante del input del usuario se produce mediante realidad aumentada; se optó por un modelo simple de menú en "sándwich", que conduce a pantallas que corresponden a dos servicios utilizados: (a) posición

geográfica del usuario [6] y (b) capturas con la cámara del dispositivo, para posterior reconocimiento de patrones [7] [8].

4. Definición de la interface

A fin de reducir la necesidad de tener que señalar con los dedos sobre la pantalla del dispositivo para seleccionar opciones, y considerando además que por las características delaprendizaje ubicuo (u-learning), la utilización tiene lugar en diferentes ámbitos (locales, vía pública, medios de transporte, etc.), se buscó proveer la interface para activar:

- Tareas a realizar: mediante geolocalización, cuando el usuario en su rutina diaria se encuentra ubicado en determinada localización, se activa el objeto de aprendizaje correspondiente.
- Ayudas para resolver tareas:mediante el uso de realidad aumentada, cuando el usuario enfoca con la cámara un determinado objeto vinculado a una tarea, se activa la ayuda para orientarlo en la solución.

Una vez realizado el análisis general acerca de cómo estructurar y configurar interfaces en Android®, se definieron las tres áreas principales de la misma: visualización del mapa, cámara para realidad aumentada y reconocimiento de texto para realidad aumentada. Cada una de estas funcionalidades se accede desde la pantalla principal que está definida a partir del siguiente fragmento de código XML.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayaout
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:layaout_width="match_parent"
android:layaout_height="match_parent"
tools:context="com.mercedes.pid_demo2018.MainActivity"
android:orientation="vertical">
<Button
android:id="@+id/Button_mapa"
android:layaout_width="match_parent"
android:layaout_height="wrap_content"
android:text="@string/button_mapa_text"
android:background="@drawable/button_style"
android:layaout_marginLeft="16dp"
android:layaout_marginRight="16dp"
android:layaout_marginTop="16dp"
android:layaout_marginBottom="8dp"
android:backgroundTint="@color/colorPrimaryDark"
android:textColor="@android:color/background_light"/>
```

Dada la disposición que tendrían los elementos, se optó por implementar un LinearLayout con orientación vertical y 3 elementos Button dentro de ella. El estilo de los botones se definió a través de un archivo externo XML que se importó como atributo Background sobre el layout actual.

Cada una de las pantallas a las que se accede a través de esos botones, se definieron en base a tres tipos de componentes cuyo propósito es minimizar la necesidad de input/output por parte del usuario.

- Visualización del mapa

La etiqueta xml<fragment> permite definir el espacio sobre el cual se cargará, en tiempo de ejecución, el mapa que se recibe de la API de Google Maps, que se genera a partir del siguiente código XML.

```
<fragmentxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:map="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/map" android:name = "com.google.android.
gms.maps.SupportMapFragment"
android:layaout_width="match_parent"
android:layaout_height="match_parent"
tools:context="com.mercedes.pid_demo2018.MainActivity"/>
```

- Cámara para Realidad Aumentada

Existen algunos navegadores para realidad aumentada, como Wikitude, Layar, Augment, que ofrecen diversas funciones de acuerdo al propósito que se tenga. En nuestro caso no se trabajó con ellos, sino

con Unity3D®, el cual es un motor de videojuegos multiplataforma; combinado con Vuforia®, una plataforma de software para desarrollar aplicaciones de RA, que ofrece facilidades para gestionar reconocimiento de patrones y búsqueda visual, mediante la identificación de puntos característicos usados para definir un objeto en particular dentro de una imagen. La tarea principal es la obtención de los puntos del objeto que permitan identificarlo nuevamente a partir de ellos. Para esto se configuraron los objetos a visualizar en la pantalla cuando el objetivo es detectado. Por ejemplo si se debe mostrar un texto cuando se enfoca una esfera, el mismo está configurado por código para cambiar acorde a si hay un objetivo detectado (la esfera) o no frente a la cámara.

- Reconocimiento de Texto

Para la sección de reconocimiento de texto se trabajó con la etiqueta XML <SurfaceView>, sobre la cual se define la cámara. Asíen tiempo de ejecución, la aplicación mostrará la cámara y el usuario podrá capturar elementos con ella. A continuaciónse muestra un fragmento del código XML utilizado.

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"/>
xmlns:map="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layaout_width="match_parent"
android:layaout_height="match_parent"
tools:context=".TextRecognition.TextCaptureActivity"
<SurfaceView
android:id="@+id/surfave_view"
android:layaout_width="match_parent"
android:layaout_height="match_parent"
android:layaout_centerVertical="true />
<Button
android:id="@+id/button_gotoLink"
android:layaout_width="match_parent"
android:layaout_height="wrap_content"
android:layaout_alignBottom="@+id/surface_view"
android:text="Leer sobre el nuevo tema"
android:background="@drawable/button_style"
android:backgroundTint="@color/colorPrimaryDark"
android:textColor="@android:color/background_light" />
android:layaout_margin="16dp" />
</RelativeLayaout>
```

Particularmente en esta sección, se configuró la aplicación para que sólo muestre el botón en las situaciones indicadas en el código, a fin de no sobrecargar el área de visualización en la pantalla. Lo que se hizo fue definir que cuando detectara el texto deseado (en este caso se seleccionó la palabra “Determinante”), se visualizara el botón permitiendo redirigirse al navegador de internet con información acerca del tema detectado.

La configuraciónse realizó de la siguiente forma:

En el método onCreate del activity que referencia el layout deseado se define la línea de código goToLink.setVisibility(View.GONE); donde goToLink es la referencia al objeto del tipo Button, y el método setVisibility permite visualizarlo o no en la pantalla. Dicho método tiene tres posibles valores:

- View.GONE: la vista está oculta y no ocupa lugar en el layout.
- View.INVISIBLE: la vista está oculta pero ocupa lugar en el layout.
- View.VISIBLE: la vista está visible.

Luego, se crea un método que habilita el botón cuando el texto detectado es el deseado:

```
prívate void enableButtom() {
      gotoLink.setVisibility(View.VISIBLE);
      gotoLink.setText (“ver el nuevo tema”);
}
```

5. Conclusiones

El esquema de acceso a las prestaciones de una aplicación móvil, hace que la misma sea exitosa o no. Con el tiempo se han definido diferentes patrones de navegación para intentar resolver el problema, presentando ventajas y desventajas en cada caso. A lo largo de este documento se han presentado algunas tecnologías que fueron investigadas, para el desarrollo de la interface de una aplicación prototipo, que minimice la necesidad de interacciones usuario-dispositivo, y que así resulte sencilla de

utilizar, en este caso, en un ambiente de enseñanza basado en u-learning, donde se supone que el estudiante estará en movimiento al usar el dispositivo y bajo diferentes condiciones ambientales.

Por las características de los ambientes en que se usan las aplicaciones móviles (luz, movimiento, etc.), se debe tener en cuenta la simplicidad de diseño de la interface. Por ejemplo para evitar tener que considerar aspectos "nativos" de los dispositivos, el uso de una interface web resulta una alternativa, sin embargo, se debe tener en cuenta que una página web compleja, probablemente no podrá ser visualizada correctamente; que no es viable ofrecer interfaces "ricas" en controles, como íconos sofisticados (que no pueden ser apreciados visualmente) y facilidades de "ayuda" (que sobrecargan en pantalla), dado que todo esto probablemente degrade el rendimiento de la aplicación. La interface debería ser simple e intuitiva, con el mínimo necesario de interacciones con el dispositivo.

Agradecimientos

El artículo es resultado del proyecto PID UTN4741: "Desarrollo de un entorno basado en Cloud Computing para Aprendizaje Ubicuo" el cual es financiado por la FRM - UTN, Mendoza- Argentina.

Referencias

[1] Tobar, L. e. (2009). WebA Mobile (Web Analysis Mobile): Assistance Tool for the Design and Evaluation of Websites for Mobile Devices. New Trends on Human–Computer Interaction, 127-139.

[2] Hussain, Z. e. (2007). User Interface Design for a Content-aware Mobile Multimedia Application: An Iterative Approach. MoMM 2007 &iiWAS 2007 Workshops Proceedings, (pp. 115-120).

[3] Gong, J. &. (2010). Guidelines for Handheld Mobile Device Interface Design. Retrieved febrero 20, 2018, from https://pdfs.semanticscholar.org/4350/ ed959d6a638168ba24058892338e8ee37338.pdf

[4] Georgiev, T. G. (2009). Investigation of the Text Entry Speed and Accuracy in Mobile Devices. Proceedings of the CompSysTech. Rousse, Bulgaria.

[5] Liao, T. (2012). A framework for debating augmented futures: Classifying the visions, promises and ideographs advanced about augmented reality. Mixed and Augmented Reality (ISMAR-AMH), 2012 IEEE International Symposium on (pp. 10.1109/ISMAR-AMH.2012.6483982). IEEE.

[6] Google Maps, A. (2017). Sitiooficial Google. Retrieved from https://developers.google.com/maps/documentation/android-api/current-place-tutorial

[7] Vuforia. (2017). Sitiooficial. Retrieved from https://library.vuforia.com/

Weiser, M. (1993). Ubiquitous Computing. Communications of the ACM (pp. 137-143). Nikkei Electronics.

[8] Unity. (2017). Sitiooficial Unity. Retrieved from https://unity3d.com/es