Mnemosina
Una herramienta para Programación Android

Juan Félix Avila Herrera[a]

[a]Universidad Nacional de Costa Rica y Universidad de Costa Rica

RESUMEN

El uso de dispositivos móviles ha tenido un crecimiento realmente impresionante en la última década. Una de las opciones más importantes para el desarrollo de aplicaciones es Android Studio. En este documento se describe el uso de la herramienta denominada Mnemosina, desarrollada por el autor y que facilita el trabajar con Android Studio. Mnemosina es un *software* innovador, de uso libre, que permite programar aplicaciones Android mediante la herramienta Android Studio al mismo tiempo que también incrementa la productividad debido a que genera código para algunas situaciones comunes o rutinarias. Mnemosina también permite que los usuarios puedan incrementar la cantidad de recetas para Android.

Keywords: Dispositivos Móviles, Android Studio, Mnemosina, Generación de Código

1. INTRODUCCION

En el año 2015 la Escuela de Informática de la Universidad Nacional de Costa Rica[1] empezó a ofrecer el curso denominado " Diseño y programación de plataformas Móviles ", que coloquialmente llamábamos "Dispositivos Móviles" o más breve aún, "DM". Aunque inicialmente estaba pensado para abarcar las plataformas más conocidas, a saber, Android e IOs, debido a las limitaciones de equipo y capacitación, DM se concentró en programación Android.[2] En la primera edición de DM se utilizó la herramienta eclipse[3] pero tiempo después, cuando Android Studio[4] resultó una opción más estable, se optó por cambiar a este nuevo entorno de desarrollo. Si bien es cierto que Eclipse permitía el desarrollo de aplicaciones Android, se debía instalar un *Plugin* para Android. Además se hizo necesario buscar opciones para visualizar las aplicaciones desarrolladas pues el visualizador integrado con Eclipse tardaba demasiado. Se optó por utilizar la herramienta Genymotion[5] que permitía una licencia gratuita para uso individual no comercial.

La experiencia con Eclipse y lo limitado del laboratorio de cómputo asignado al curso resultó muy frustrante. Cada pequeño proceso requería de muchísimo tiempo y los estudiantes se frustraban pues varias veces durante un examen programado la computadora con la que trabajan se "congelaba" , obligándolos a empezar de nuevo. Es debido a esto y a las constantes quejas de los docentes que una de las primeras medidas que se tomaron fue cambiar totalmente el equipo del laboratorio utilizando equipo más moderno. Es importante mencionar que la migración de Eclipse a Android Studio fue realmente significativa puesto que requirió volver a preparar buena parte del material de las clases.

Una vez que el equipo del laboratorio y el ambiente de desarrollo dejaron ser la principal limitante, la principal queja se concentró en que el desarrollo de aplicaciones en Android Studio resultaba lento. Muchas tareas rutinarias como el uso de componentes visuales (tales como botones, cajas de texto, etc) tomaban buena parte del tiempo en el proceso de desarrollo de aplicaciones. Por otro lado, dado que este curso no estaba dirigido a estudiantes sin conocimientos de programación sino mas bien a estudiantes con fuertes bases en lenguajes tales como Java,[6] JavaScript[7] y C^{++},[8] se hacía necesario buscar una forma de mejorar el tiempo de desarrollo de las aplicaciones en Android Studio. Además de los elementos anteriormente mencionados, el uso de búsquedas en la web para realizar tareas tales como el uso de la cámara, del GPS, del audio, de Google Maps, etc, arrojaban muchos resultados que en algunos casos era desactualizados o incluso erróneos. Ciertamente el sitio Android Developer[9] es tal vez una de las mejores referencias para programación de aplicaciones Android, no obstante

Para mayor información: E-mail: delagarita@gmail.com

debido a que es un recurso altamente denso, muchas veces el tiempo de búsqueda puede alargarse más de lo deseado. Es debido a las razones mencionadas anteriormente que se decidió crear Mnemosina.

El *software* Mnemosina fue desarrollado en Delphi 2010 y por el momento corre solamente sobre Windows. Mnemosina pretende servir de repositorio de todas aquellas soluciones que el usuario considera importantes. Cada desarrollador está en capacidad de incrementar la base de datos de Mnemosina de acuerdo con sus intereses. De esta forma si un desarrollador encuentra cierto código que considera importante, puede guardarlo en Mnemosina, y cada vez que lo necesite, estará ahí disponible, con la certeza de que (probablemente) funcionará y no requiere Internet para obtener la información.

Una forma de utilizar Mnemosina es como una colección de recetas de programación y en este sentido, esto no es nuevo.[10,11] El buscador de Mnemosina permite buscar por una descripción dada por el propio usuario o bien por cualquier palabra que recuerde que esté contenida en su base de datos. Para efectos del curso de DM, a los estudiantes se les dio una versión de Mnemosina con lo necesario para abarcar los contenidos del curso.

Ciertamente Mnemosina puede convertirse en un recetario, lejos de ser un defecto, vale la pena mencionar que también en el arte culinario, en la perfumería, en farmacia, por mencionar solo algunos campos, los recetarios constituyen un recurso muy valioso que permite asegurar la calidad y consistencia de ciertos procesos. Tal vez la principal consigna con Mnemosina es recolectar, documentar y combinar procedimientos robustos para lograr hacer aplicaciones más interesantes y útiles en un menor tiempo. Lo que hoy resultó ser un desafío, mañana simplemente será un punto intermedio para lograr hacer un procedimiento más importante. Hay muchas cuestiones de programación que son afines a muchos lenguajes de programación tales como, ciclos, listas, vectores, matrices, bases de datos, entre otros. Sin embargo, los dispositivos móviles requieren código específico para administrar recursos particulares de un dispositivo móvil, tal es el caso de la cámara, del sensor de proximidad, de los mensajes de texto, llamadas telefónicas, GPS, acelerómetro, entre otros.[12] En este aspecto Mnemosina resulta particularmente útil.

2. PROGRAMACION ANDROID

Comentamos aquí algunas cuestiones básicas sobre programación Android. De ninguna manera se pretende cubrir en esta sección todo lo referente a este tema pues para ello se necesitaría uno o varios libros de texto.

En Android la programación se puede realizar mediante el lenguaje Java y el uso de XML[13] (también es posible utilizar el lenguaje Kotlin,[10] pero no hablaremos de esta opción en este documento). Podríamos decir que la parte "lógica" de la aplicación se hace con Java y la parte "visual", con XML. El XML es un lenguaje que nos permite incorporar componentes visuales a nuestra aplicación tales como por ejemplo botones o cajas de texto. Aquellos que hayan estudiado LaTeX[14] o HTML, encontraran cierta familiaridad con esta opción. Tanto Eclipse como Android Studio cuenta con opciones del tipo visual/texto que facilitan el proceso de agregar componentes visuales a nuestras aplicaciones.

Como es de esperar resulta importante poder referenciar los componentes visuales desde Java. A esta conexión le llamaremos "alambrado". El alambrado consiste entonces en crear, para cada componente visual, una correspondiente variable en Java. De esta manera es posible obtener y colocar datos en los componentes visuales mediante Java. Un vez hecho el alambrado, también es posible, dependiendo de su naturaleza, realizar la programación de eventos asociados a los diferentes componentes.

En Android, a lo que usualmente llamamos ventanas en Windows, le llamaremos actividades ("activities"). Una aplicación está compuesta de una o varias actividades. Cada actividad tendrá su propio XML asociado. Cada actividad puede existir de manera independiente (como si fuera una aplicación) y esto obliga a utilizar a elegir siempre una actividad para el arranque de la aplicación y el empleo de lo que se denominan "intents" para moverse de una actividad a otra. De particular importancia en este paradigma de programación resulta la forma de utilizar variables globales comunes a todas las actividades. Esto se puede lograr mediante una clase singleton o bien mediante una clase base de donde deriven todas las actividades. Android tambien permite enviar información mediante los "intents" utilizando un comando denominado putextra.

De particular importancia resulta el estudiar el ciclo de vida de una Actvidad.[9] Este ciclo muestra las distintas etapas por las que pasa un Actividad desde su nacimiento hasta que es destruida. Uno de esos estados

ISBN 978-84-09-02374-5

de la Actividad es el relacionado con el método OnCreate que siempre estará presente en el código java de cada Actividad. Dentro de OnCreate se deben programar o referenciar los principales en los que participa la Actividad. Es aquí en donde, por ejemplo, se puede realizar el alambrado de los componentes declarados en el XML asociado.

Los Fragmentos (Fragments) son también un recurso importante en Android. Un Fragmento es similar a una Actividad pero pueden utilizarse uno o varios de ellos dentro de una misma Actividad.[15] Los *Fragments* permiten además de aprovechar mejor las particularidades visuales de cada dispositivo Android, reducir el mantenimiento del código y propiciar la reutilización del mismo.

En cuanto a bases de datos, Android provee la opción Sqlite[16] que permite crear y administrar una base de datos local. También están las opciones de almacenar en archivos de texto, de archivos adjuntos y la de SharedPreferences (que permite guardar datos simples). La opción Firebase de Google permite también la posibilidad de crear una base remota de tiempo real y que puede ser administrada desde diferentes plataformas.

3. DESCRIPCION DE MNEMOSINA

Se describe aquí los principales aspectos de nuestro *software*. En la Fig. 1 se muestra la pantalla principal de Mnemosina. En el editor ubicado en la parte inferior izquierda se muestra el resultado de un registro buscado

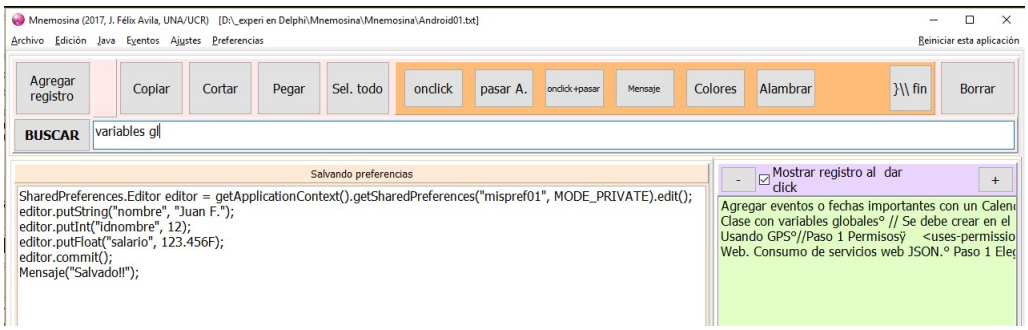

Figure 1. Pantalla principal de Mnemosina

y encontrado. Este mismo editor sirve para colocar una nueva contribución que se desea agregar a la base de datos. Para ello se utiliza el botón "Agregar Registro".

El botón "*onclick*" (Fig. 2) despliega un menú tipo "*PopUp*" que permite en primera instancia escoger si se desea programar un evento *onclick* para un componente en un Fragmento o en una Actividad. Seguidamente el usuario escoge el componente visual deseado de las opciones mostradas. Si la opción no se muestra, el usuario puede ingresar el nombre Java del componente y de igual forma Mnemosina generará el código correspondiente. En el menú de la Fig. 2 hay una opción denominada "prefijo" que permite asignar nombres mas representativos a los componentes visuales. Por otro lado hay varias opciones en cuanto al número de componentes que se desea agregar. La primera es "Onclick de un Button" (que cambiará si se elige otro componente que no sea un botón) en la que se genera el siguiente código que se coloca en el método OnCreate de una actividad:

```
// alambramos el Button
Button MiButton = (Button) findViewById(R.id.btnNombreControl);
//Programamos el evento onclick
 MiButton.setOnClickListener(new View.OnClickListener(){
@Override
      public void onClick(View arg0) {
        // escriba lo que desea hacer
      }
  });
```

ISBN 978-84-09-02374-5

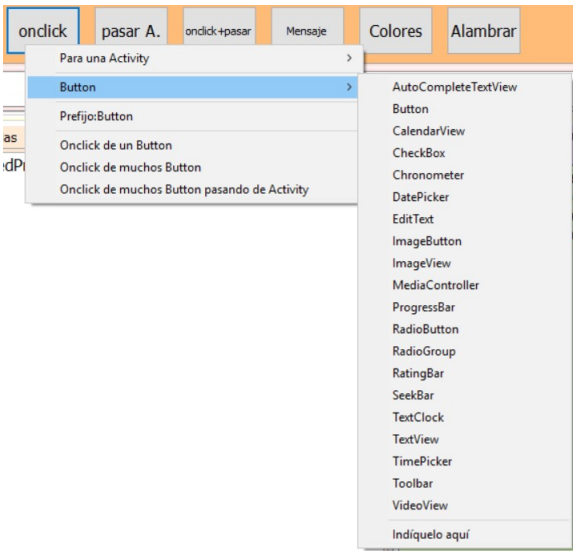

Figure 2. Evento *Onclick*

Si por el contrario el usuario desea agregar por ejemplo 4 botones, el código se genera en dos partes. La primera parte se coloca en el método OnCreate de la actividad y es como sigue"

```
OnclickDelButton(R.id.button1);
OnclickDelButton(R.id.button2);
OnclickDelButton(R.id.button3);
OnclickDelButton(R.id.button4);
```

La segunda parte sería un método de la clase Actividad y se puede colocar en cualquier parte antes del la llave final que cierra la clase:

```
public void OnclickDelButton(int ref) {
        // Ejemplo  OnclickDelButton(R.id.MiButton);
        // 1 Doy referencia al Button
        View view =findViewById(ref);
        Button miButton = (Button) view;
       //  final String msg = miButton.getText().toString();
       // 2.  Programar el evento onclick
        miButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                // if(msg.equals("Texto")){Mensaje("Texto en el bot\'on ");};
                switch (v.getId()) {
                    case R.id.button1:
                        Mensaje("Implementar Button1");
                        break;
                    case R.id.button2:
                        Mensaje("Implementar Button2");
                        break;
                    case R.id.button3:
                        Mensaje("Implementar Button3");
                        break;
                    case R.id.button4:
```


ISBN 978-84-09-02374-5

```
                    Mensaje("Implementar Button4");
                    break;
                default:break; }// fin de casos
            }// fin del onclick
        });
    }// fin de OnclickDelButton
```

Con referencia al botón denominado "onclick+pasar", este permite generar código para cambiar de actividad utilizando un componente

```
// alambramos el boton
Button MiBoton = (Button) findViewById(R.id.btnNombreControl);
//Programamos el evento onclick
 MiBoton.setOnClickListener(new View.OnClickListener(){
@Override
      public void onClick(View arg0) {
        Intent intento = new Intent(getApplicationContext(), SegundaActivity.class);
        startActivity(intento);
       // escriba lo que desea hacer
     }
  });
```

El botón "Mensaje" mostrado en la Fig. 3 brinda un menú para distintos tipos de mensajes, a saber mensajes usando *Toast*, usando el *Action Bar*, usando un *TextView*, usando diálogos mediante un botón *OK*. Está también el mensaje que puede desplegarse dentro de un *Fragment*.

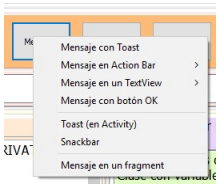

Figure 3. Evento Onclick

El botón "Alambrar" genera el código necesario para alambrar los componentes usuales. Ver Fig. 4. Se puede generar código tanto para el caso de una Actividad o bien de un Fragmento. Finalmente en el lado derecho de

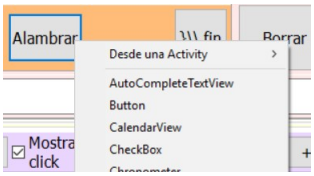

Figure 4. Alambrando componentes

la Fig. 1, se puede apreciar una sección en donde el usuario puede buscar temas o recetas para un determinado teman. Los temas mas complejos se dan mediante el uso de un estilo Paso-a-Paso como se muestra en el siguiente ejemplo.

Paso 1. Crear carpeta anim res → *Click* derecho → New → Android Resource directory
En resouce type escoja anim
Paso 2. anim → *Click* derecho → New → Animation resouce file →

ISBN 978-84-09-02374-5

Poner nombre de acuerdo a la animación deseada (minúscula-inglés)
Ejemplo
mianimacion.xml
Paso 3. Usar la animación xml sobre un componente (digamos un ImageView) mediante java.
ETC.

4. VALORACIÓN DE LOS ESTUDIANTES SOBRE MNEMOSINA

En cada uno de los tres grupos de DM, en los que se utilizó Mnemosina, se pasó una encuesta que mide los alcances del curso y el desempeño docente. Esta encuesta se pasa obligatoriamente en todos los cursos de la Universidad Nacional de Costa Rica. Tanto en esta encuesta como en manifestaciones directas, los estudiantes reconocen en Mnemosina una herramienta que facilita el desarrollo rápido de aplicaciones y propicia el aprendizaje de Java para Android.

5. CONCLUSIONES

Mnemosina es un *software* orientado principalmente a incrementar la productividad cuando se utiliza el entorno Android Studio. Es además una herramienta didáctica pues permite aprender de otros y reutilizar cualquier proceso o código que el usuario encuentra importante. Mnemosina es capaz de reducir sustancialmente el tiempo de desarrollo a la vez que reduce los errores debido al uso de código que ha sido probado con anterioridad. Los usuarios pueden concentrarse en tareas más interesantes dejando de lado los procesos rutinarios.

Mnemosina además permite ser una herramienta colaborativa. En su última versión aprovechando el entusiasmo de los estudiantes del curso de DM, se lograron incorporar procesos que no estaban contemplados aún en esta herramientas. Esto hace que para las futuras ediciones del curso, los nuevos estudiantes puedan realizar aplicaciones más complejas.

RECONOCIMIENTOS

Se agradece el apoyo dado por la Escuela de Informática de la Universidad Nacional de Costa Rica.

REFERENCIAS

[1] UNA, "Escuela de informática." http://www.escinf.una.ac.cr. (Accedido: 28-10-2018).
[2] Lequerica, J. R., [*de livre: Desarrollo de aplicaciones para Android*], Anaya Multimedia (2017).
[3] E, "Eclipse." http://www.eclipse.org. (Accedido: 28-10-2018).
[4] Smyth, N., [*Android Studio 2 Development Essentials*], EBookFrenzy (2016).
[5] Rodríguez, T., "Genymotion, el emulador más rápido de android," (2014).
[6] Deitel, P. and Deitel, H., [*Java How to program*], Prentice Hall Press (2011).
[7] Zakas, N. C., [*Javascript Para Desarrolladores Web/javascript for Web Development (Anaya Multimedia)*], Anaya Publishers (2006).
[8] Deitel, H. M. and Deitel, P. J., "C++, how to program. how to program series," *Prentice Hall* **3**, 42 (2014).
[9] AD, "Android developer." https://developer.android.com. (Accedido: 28-10-2018).
[10] Roy, A. S. and Karanpuria, R., "Kotlin programming cookbook: Explore more than 100 recipes that show how to build robust mobile and web applications with kotlin, spring boot, and android," (2018).
[11] Steele, J. and To, N., [*The Android developer's cookbook: building applications with the Android SDK*], Pearson Education (2010).
[12] Gironés, J. T., [*El gran libro de Android*], Marcombo (2012).
[13] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F., "Extensible markup language (xml).," *World Wide Web Journal* **2**(4), 27–66 (1997).
[14] Lamport, L., [*LaTeX: A Document Preparation System*], Addison-Wesley, Reading, Mass. (1994).
[15] MacLean, D. and Komatineni, S., "Fragments fundamentals," in [*Android Fragments*], 1–34, Springer (2014).
[16] Revelo, J., "Tutorial de bases de datos sqlite en aplicaciones android," (2014).

 ISBN 978-84-09-02374-5