Lámpsakos
E-ISSN: 2145-4086
lampsakos@amigo.edu.co
Fundación Universitaria Luis Amigó
Colombia

Hernández Ramírez, Luisa Fernanda; Giraldo Giraldo, Fabián Andrés; Cárdenas Castellón, Jonathan Ray
GPLAD: PROGRAMACIÓN ESTRUCTURADA SOBRE DISPOSITIVOS ANDROID
Lámpsakos, núm. 8, julio-diciembre, 2012, pp. 23-31
Fundación Universitaria Luis Amigó
Medellín, Colombia

Lámpsakos | Nº. 8 | PP. 23 - 31 | julio-diciembre | 2012 | ISSN: 2145-4086 | Medellín - Colombia

GPLAD: PROGRAMACIÓN ESTRUCTURADA SOBRE DISPOSITIVOS ANDROID

GPLAD: STRUCTURE PROGRAMMING ON ANDROID DEVICES

Luisa Fernanda Hernández Ramírez
Fundación Universitaria San Martín. Bogotá, Colombia

Fabián Andrés Giraldo Giraldo
MSc (C) Fundación Universitaria San Martín. Bogotá, Colombia.

Jonathan Ray Cárdenas Castellón
Fundación Universitaria San Martín. Bogotá, Colombia.

(Recibido el 17/02/2012. Aprobado el 19/03/2012)

Resumen. La programación por medio de bloques es un enfoque visual que permite a las personas jóvenes adquirir interés en el desarrollo de software. Así lo han demostrado herramientas para computadores como Scratch, StarLogo y Alice. Hoy en día hay estudiantes e ingenieros programadores que desean implementar soluciones más estructuradas y hay pocas herramientas que, a través de programación gráfica ofrezcan la posibilidad de solucionar problemas que se manejen en primeros semestres universitarios. Por otro lado, los dispositivos móviles, en especial los que poseen sistema operativo Android, han tenido acogida en los últimos años, además, son utilizados para resolver problemas en tiempo real. El objetivo de este trabajo fue demostrar la utilidad de una aplicación que soporta la programación estructurada por medio de bloques sobre dispositivos Android, que facilita la creación y ejecución de código en el lenguaje Java sobre un servidor remoto. Se realizó un análisis para determinar el ambiente de programación, la creación de bloques con su representación intermedia y las validaciones para la generación de código, además, se realizaron pruebas a través de la formulación de problemas matemáticos que se enseñan durante los primeros cursos de Ingeniería de sistemas en la Fundación Universitaria San Martín. Finalmente, se obtuvo el código Java de la solución a cada problema planteado dentro de Gplad.

Palabras clave: Programación por bloques; Lenguajes; Traductor.

Abstract. Programming through blocks is a visual approach that allows young people to acquire interest in the development of software. This has been demonstrated by computer tools such as Scratch, StarLogo and Alice. Today students and engineers are programmers who want to implement structured solutions and there are few tools that, through graphical programming offer the possibility of solving problems that are handled in first universities semesters. Furthermore, mobile devices, especially those with Android operating system, have been well received in recent years, moreover, are used to solve problems in real time. The aim of this work was to demonstrate the utility of an application that supports structured programming through blocks on Android devices, facilitating the creation and execution of code in the Java language on a remote server. Analysis was performed to determine the programming environment, building blocks with intermediate representation and validations for code generation also were tested through the formulation of mathematical problems that are taught during the first courses in engineering systems in the San Martin University Foundation. Lastly, Java code was obtained from the solution to every problem presented within Gplad.

Keywords: Block programming; Languages; Translator.

1. INTRODUCCIÓN

Según la revista *BBC Mundo* [1]: "Del mismo modo que en su día el latín sirvió a los romanos para unificar culturas, comerciar y conectar a las gentes de su vasto imperio, la programación es hoy el lenguaje universal que nos permite conectar con el imperio de la tecnología", por lo tanto, se ha logrado despertar interés por parte de las personas que no poseen conocimiento en el ámbito de la programación.

Pero mantener el interés es tan solo un primer paso, pues su pérdida es común en el área de la Ingeniería de Sistemas, como lo demuestra un estudio realizado en la Universidad Monmouth de Estados Unidos, en el que, entre los años 2000 al 2005, un 70% de los estudiantes de Ciencias de la computación migraba hacia otras carreras [2]. Una de las principales razones para que esto ocurra está en lo complejo que puede ser la tarea de aprender a programar para personas de todas las edades. Los programadores novatos tienen que aprender a estructurar soluciones y entender cómo es ejecutado cada programa, además de una sintaxis rígida y comandos que parecen, inicialmente, como arbitrarios [3].

Para evitar que las personas novatas pierdan interés por causa de lo difícil que puede ser aprender a programar y el tiempo que esto toma, se sugiere la programación gráfica o visual que se da de dos maneras: lenguaje basado en nodos y lenguaje basado en bloques que permiten crear aplicaciones que producen una mínima cantidad de errores de sintaxis y compilación en comparación con la programación convencional, es decir, la programación basada en comandos de texto. Esta programación visual es útil para prácticas académicas con estudiantes desde tempranas edades puesto que los incentiva para programar de tal manera que se enfoquen en la lógica y a mejorar sus habilidades cognitivas y creativas [4].

La programación visual es un enfoque para la integración de varios bloques y la implementación de aplicaciones en un corto período de tiempo, sin las habilidades necesarias en la codificación textual, lo que genera una mínima cantidad de errores de sintaxis y compilación [5].

En un estudio realizado en el año 2006 por Wang et al., se afirma que el tiempo invertido por los estudiantes para programar, por medio de bloques, es menor que la manera tradicional puesto que se previenen errores que normalmente los frustraban y bajaban su interés en temas de desarrollo. Como resultado de esta investigación se concluye que las herramientas visuales evitan errores y proporcionan una programación aún más estructurada que la textual [6].

Pero, ¿qué sucede con las personas que desean programar desde un dispositivo móvil? Esta es una problemática que desde hace poco tiempo está siendo atacada, por ejemplo, en marzo de 2012 el equipo de Android innovó con una aplicación en el mercado llamada Android Integrated Development Environment –AIDE- () [7] que permite crear un programa por medio de comandos de textos en dispositivos Android y compilarlo directamente desde el celular. Además, se evitan los emuladores, que implican mayor tiempo.

Programar dentro de un dispositivo móvil por medio de comandos de textos puede ser un proceso incómodo por el tamaño de la pantalla, además, se está expuesto a cometer errores sintácticos, por ende, es más práctico realizarlo con lenguajes visuales previamente nombrados. Con este avance el usuario lograría minimizar la dependencia que se tiene con un computador para realizar las aplicaciones.

El objetivo de este artículo es mostrar la utilidad de una aplicación llamada Graphical Programming Language on Android Devices –Gplad- que permite generar código en lenguajes como Java y ejecutarlo sobre servidores remotos para obtener su respuesta a través de bloques desde dispositivos Android.

Una de las aplicaciones utilizadas para programar sobre dispositivos móviles es Catroid [8], una aplicación para Android que soporta la programación visual permitiendo elaborar animaciones y juegos que se compilan desde el mismo dispositivo. A diferencia de Gplad que soporta el paradigma de programación estructurada, además, es importante indicar que éste tiene una diferencia puesto que el enfoque gráfico propuesto garantiza que los programadores comprendan más fácilmente un algoritmo porque se visualiza el ámbito de cada una de las porciones de código modeladas.

La estructura de este artículo se describe de la siguiente manera: en la sección 2 se abarca el proceso de desarrollo de Gplad, su estructura y planteamiento de dos ejercicios dentro de dicha herramienta para exponer su nivel, alcance y competitividad frente a otras aplicaciones. En la sección 3 se muestran los resultados obtenidos a través de la aplicación. En la sección 4 se encuentra lo que se planea para el futuro con Gplad. Finalmente, en la sección 5, se exponen las conclusiones basadas en los logros alcanzados en las secciones previas.

2. METODOLOGÍA

2.1 Graphical Programming Language For Android Devices -Gplad-

Es una aplicación para dispositivos que tengan como sistema operativo Android 2.3 o superior; posee un Do-

main Specific Language -DSL- gráfico implementado en Java y utiliza una interfaz drag and drop, con cajas y botones que permite expresar un problema de programación dentro de un dispositivo móvil Android en un lenguaje simple, de tal manera que los usuarios programen sin pensar en los errores sintácticos y se enfoquen en la lógica de su aplicación.

La problemática planteada para la creación de Gplad consistió principalmente en crear una aplicación que permita a los ingenieros programadores o personas interesadas en el tema de la ingeniería de software, solucionar y trabajar sobre problemas o retos de programación en diferentes lugares que no sea un computador convencional, al que se le adiciona otro gran desafío que permita al usuario programar por medio de bloques desde un dispositivo móvil.

Fig. 1. Ambiente de programación de Gpald

Como puede verse en la Fig. 1, los proyectos o programas de Gplad son construidos con el uso de la técnica drag and drop de las instrucciones que se encuentran en la categoría de bloques y se ubican dentro del escenario. La aplicación permite programar de forma cómoda, en dispositivos móviles Android y soporta cualquier cantidad de bloques que encajen entre sí y permite el desplazamiento a través del visualizador en miniatura.

2.2 Etapa de construcción de GPLAD

Para diseñar el lenguaje específico de dominio (DSL) se hizo un análisis entre el gráfico y el textual, se tomaron las variables: el tamaño de la pantalla, la GUI, el tipo de respuesta que debían tener los eventos generados por la aplicación y las características del hardware, entre otros, de acuerdo con los estudios realizados en [9].

Para la creación del DSL se tomó como referencia e inspiración el lenguaje de programación por bloques StartLogo TNG [10], que permite saber si se está realizando bien la construcción del problema de manera que la forma de las figuras concuerden con otro único bloque. Para la producción de cada uno de los bloques existen dos categorías que determinan la forma de los bloques (Round, Sharp o Square) y cada una puede pertenecer a uno de los dos grupos (Command Connector o Command Slot).

Fig. 2. Categorías de formas de Command Slot

En la Fig. 2, se observan las maneras para crear espacios de encaje de cada bloque. A partir de las formas básicas se pueden componer nuevas y se proporciona a cada bloque un estilo particular que le permita encajar sólo con los que debe.

Fig. 3. Categorías de formas de Command Connector

Las categorías de Command Connector que se muestran en la Fig. 3, le permiten a cada bloque tener la forma de conexión apropiada para encajar con los bloques que tengan el espacio o command slot respectivo.

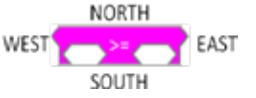


Fig. 4. Orientación de los bloques

Cada forma que se desee implementar a un bloque debe tener una coordenada de orientación basada en la Fig. 4, y se manipula con unos valores flotantes dentro del código.

Fig. 5. Bloques de Gplad

Para implementar el DSL escogido y aplicarlo como solución a los lenguajes de programación, se hizo un estudio de las principales estructuras de control sobre un lenguaje de programación para expresar programas básicos. Los resultados de los bloques se advierten en la Fig. 5. De acuerdo con el tipo de bloque se tiene una representación XML diferente. Dicha representación deja hacer una transformación intermedia del programa visual expresado por el usuario a un lenguaje textual basado en XML que posteriormente será interpretado para hacer la generación de código al lenguaje de propósito general Java.

Para asegurar que el XML cumpla la estructura adecuada, se creó un DTD (Document Type Definition) que valida el XML que genera el aplicativo Gplad.

```
<!ELEMENT MAIN (VARIABLES?, LOGIC?)>
<!ATTLIST MAIN
name CDATA #REQUIRED>
<!ELEMENT VARIABLES (VARIABLE*)>
<!ELEMENT VARIABLE (#PCDATA)>
<!ATTLIST VARIABLE
name CDATA #REQUIRED
type CDATA #REQUIRED
value CDATA #REQUIRED>
<!ELEMENT LOGIC (CONDITIONAL | ASSIGN | LOOP | PRINT )*>
<!ELEMENT PRINT (#PCDATA)>
<!ATTLIST PRINT
type CDATA #REQUIRED
value CDATA #REQUIRED
variable CDATA #IMPLIED>
<!ELEMENT CONDITIONAL (RESTRICTIONS,THEN,ELSE?)>
<!ELEMENT RESTRICTIONS (LEFT,OPERATOR,RIGHT)>
<!ELEMENT LEFT (RESTRICTIONS?)>
<!ELEMENT RIGHT (RESTRICTIONS?)>
<!ELEMENT OPERATOR (#PCDATA)>
<!ATTLIST OPERATOR
type CDATA #REQUIRED>
<!ATTLIST CONDITIONAL
type CDATA #REQUIRED>
<!ELEMENT THEN (CONDITIONAL | ASSIGN | LOOP | PRINT )*>
<!ELEMENT ELSE (CONDITIONAL | ASSIGN | LOOP | PRINT )*>
<!ELEMENT ASSIGN (#PCDATA)>
<!ATTLIST ASSIGN
variable CDATA #REQUIRED
type CDATA #REQUIRED
value CDATA #REQUIRED>
<!ELEMENT LOOP (RESTRICTIONS?,THEN )>
<!ATTLIST LOOP
type CDATA #REQUIRED
block1 CDATA #IMPLIED
operator CDATA #IMPLIED
block2 CDATA #IMPLIED
from CDATA #IMPLIED
to CDATA #IMPLIED
step CDATA #IMPLIED>
```

Dentro de las sentencias que se muestran en el DTD, los valores #REQUIRED indica que el valor es requerido para que se realice, el valor #IMPLIED es un valor opcional. Cada programa sigue un dtdtree y se revisa que se cumpla cada uno de los parámetros, de tal manera que se genere correctamente la traducción y este árbol se encuentra expuesto a continuación:

```
MAIN (name)
|
|- VARIABLES?
| |
| +- VARIABLE* (name, type, value)
|
+- LOGIC?
  |
  |- CONDITIONAL (type)
  | |
  | |- RESTRICTIONS
  | | |
  | | |- LEFT
  | | | |
  | | | +- RESTRICTIONS?  **
  | | |
  | | |- OPERATOR (type)
  | | |
  | | +- RIGHT
  | |   |
  | |   +- RESTRICTIONS?  **
  | |
  | |- THEN
  | | |
  | | |- CONDITIONAL (type) **
  | | |
  | | |- ASSIGN (variable, type, value)
  | | |
  | | |- LOOP (type, block1?, operator?, block2?, from?, to?, step?)
  | | | |
  | | | |- RESTRICTIONS?  -->
  | | | |
  | | | +- THEN  **
  | | |
  | | +- PRINT (type, value, variable?)
  | |
  | +- ELSE?
  |   |
  |   |- CONDITIONAL (type) **
  |   |
  |   |- ASSIGN (variable, type, value)
  |   |
  |   |- LOOP (type, block1?, operator?, block2?, from?, to?, step?) -->
  |   |
  |   +- PRINT (type, value, variable?)
  |
  |- ASSIGN (variable, type, value)
  |
  |- LOOP (type, block1?, operator?, block2?, from?, to?, step?) -->
  |
        +- PRINT (type, value, variable?)
```

Con esta estructura y la lectura del XML creado por Gplad se produce un segundo árbol. Este árbol es generado y se respeta el orden que tienen las estructuras dentro del XML producido por Gplad con el objeto de evitar problemas en el momento de la traducción.

Para crear esta representación intermedia se obtiene con base en la serialización de cada bloque que se haya creado dentro del escenario.

Fig. 6. Arquitectura de Gplad

Gplad requiere del uso de dos componentes de hardware elementales para el proceso de traducción y compilación; la parte gráfica se realiza desde el dispositivo Android y el servidor contendrá el compilador y traductor.

Como puede observarse en la Fig. 6, para la comunicación por medio del dispositivo Android y el servidor se empleó Jade Android [11], una plataforma que permite, por medio de protocolos de comunicación de agentes, específicamente Fipa Request, establecer un enlace entre el dispositivo móvil y el equipo para realizar envío de información a través de diferentes actos de comunicación entre agentes.

Para la conexión entre el agente y el equipo remoto, es necesario designar propiedades que necesita identificar el agente para saber con quién se va a conectar.

Estas propiedades son la IP y el puerto:

```
Properties props = new Properties();
  props.setProperty(Profile.MAIN_HOST,ip);
  props.setProperty(Profile.MAIN_PORT, port);
```

Estos valores se pasan por parámetros globales de tipo Application, obtenidos en la interfaz de configuración, en el que el usuario ingresa la dirección IP y el puerto donde está el agente del equipo remoto que espera un mensaje. Al realizar las propiedades de manera correcta se invoca el método *JadeGateway.connect* donde se realiza la conexión.

La creación del código XML generado se almacena en una variable que pasa por el agente que se encarga de recibir el código intermedio, que, de acuerdo con el lenguaje objetivo (Java), es enviado a un agente que se encuentra sobre el equipo remoto que recibe la información del dispositivo móvil Android, verifica el tipo de lenguaje y envía el comando al compilador, posteriomente, el agente localizado en el computador envía un mensaje de recibido y la respuesta de la traducción y la ejecución al agente en el dispositivo Android.

Para la implementación del traductor de código se hizo uso de 2 librerías llamadas Matra y JDOM, el primero se usa para leer la definición del lenguaje que está en un archivo Document Type Definition. –DTD-. JDOM es el encargado de leer el XML generado por Gplad y de leer el plugin del lenguaje con el que se va a trabajar.

Para la creación del código, fue necesario definir un mecanismo de extensión con plugin (lenguajes especificados en XML). Cada plugin tiene un esquema de traducción entre el lenguaje origen (bloques) y el lenguaje destino.

Por último, se produce la integración entre el agente y el traductor, es decir, el agente traductor envía al agente de la plataforma móvil la especificación del lenguaje (Java) y los resultados de la ejecución, en caso de que el programa estuviera correctamente especificado, o en su defecto, informa de los errores de compilación de acuerdo con las falencias sintácticas existentes.

2.3 Pruebas de desarrollo en Gplad

Gplad permite implementar programación estructurada a través de bloques sobre dispositivos móviles, lo que puede significar que esté al nivel de herramientas de desarrollo visual como Scratch y Alice, puesto que estas dos últimas herramientas son orientadas para gente joven (8 a 16 años) [12], y empleadas para historias animadas y juegos, mientras que Gplad, además de diferenciarse por ser una herramienta móvil, es capaz de soportar soluciones estructuradas a problemas básicos que se presentarían en los primeros cursos de Ingeniería de Sistemas.

La determinación de las pruebas de Gplad se estableció según ejercicios de cursos iniciales de la Facultad de Ingeniería de Sistemas de la Fundación Universitaria San Martín. Con el fin de demostrar uno de los ejercicios que se pueden implementar en esta aplicación se ha seleccionado una prueba para hacer operaciones matemáticas y que precise de lógica por parte del usuario.

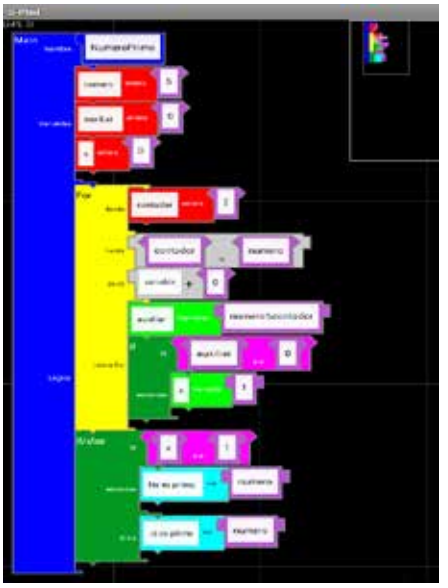
Para las pruebas se hizo uso de una tableta Samsung con sistema operativo Android 4.0 a la que se le instaló Gplad. Además de un computador portátil Lenovo con sistema operativo Windows 7, el cual, de manera provisional, contiene el traductor y funciona como medio remoto. Es de manera provisional porque, como trabajo futuro, se establecerá como servicio web para ser desplegada sobre computadoras personales.

Ejercicio uno: número primo

El programa debe recibir un número entero y determinar si es un número primo o no lo es. Para el caso base se debe probar con el número 5. Para solucionar este problema hay que tener claro que un número primo es un entero n si n>1 y si los únicos divisores posibles de n son 1 y n. Si n>1 y n no es primo, entonces se llama compuesto [13].

Con el objetivo de solucionar el problema de números primos dentro de Gplad se inició la implementación de los bloques correspondientes en el escenario, según la lógica planteada para su solución.

Como se puede observar en la Fig. 7, se hizo uso de bloques de control, operadores, variables, asignaciones e impresiones que previamente fueron expuestos en la Fig. 5.



Una vez se implementaron los bloques con los respectivos nombres y lógica, se configuró la ip y el puerto de acuerdo con la conexión de internet a la que se encuentre conectado el dispositivo móvil y el computador encargado de la traducción. Después, se configuró el agente y se indicó el lenguaje en el que se desea obtener el código fuente (Java). Se adquiere, de manera inmediata, su código fuente en dicho lenguaje. Cuando se realizó el proceso de traducción se observó no sólo el código en el lenguaje objetivo (Java), sino, también, su representación intermedia en XML.

Fig. 7. Implementación del ejercicio uno en Gplad

```
<MAIN name="NumeroPrimo">
<VARIABLES>
<VARIABLE name="numero" type="integer" value="5"/>
<VARIABLE name="auxiliar" type="integer" value="0"/>
<VARIABLE name="x" type="integer" value="0"/>
</VARIABLES>
<LOGIC>
<LOOP type="for"><FROM><VARIABLE name="contador" type="integer" value="2"/>
</FROM> <TO><RESTRICTIONS><LEFT>contador</LEFT> <OPERATOR type="&lt;">
</OPERATOR><RIGHT>numero</RIGHT></RESTRICTIONS></TO> <STEP variable="contador" operator="+"
value="1"></STEP>
<THEN><ASSIGN variable="auxiliar" type="asignaExp" value="numero%contador"/>
<CONDITIONAL type="if"><RESTRICTIONS><LEFT>auxiliar</LEFT> <OPERATOR type="==">
</OPERATOR><RIGHT>0</RIGHT></RESTRICTIONS>
<THEN>
<ASSIGN variable="x" type="asigna" value="1"/>
</THEN>
</CONDITIONAL>
</THEN></LOOP>
<CONDITIONAL type="if_else"><RESTRICTIONS><LEFT>x</LEFT> <OPERATOR type="==">
</OPERATOR><RIGHT>1</RIGHT></RESTRICTIONS>
<THEN>
<PRINT type="textAndVar" value="NO es primo " variable="numero"/>
</THEN>
<ELSE>
<PRINT type="textAndVar" value="SI es primo " variable="numero"/>
</ELSE>
</CONDITIONAL>
</LOGIC>
</MAIN>
```

Obsérvese la representación XML estructurada de la unión de cada uno de los bloques implementados en el escenario de Gplad. Por ejemplo, en la primer línea se encuentra el inicio del método cuyo nombre fue asignado en el bloque con </MAIN name="NumeroPrimo"> y finaliza con el cierre del bloque, como se ve en la última línea con la sentencia </MAIN>. Así mismo funcionan los bloques implementados dentro del escenario, de acuerdo con su estructura y representación en XML.

Ejercicio dos: número invertido

Se debe generar una aplicación que reciba un número de tres cifras y de debe obtener el número de manera invertida. Por ejemplo, si se recibe 284, se debe generar como solución el número 482. Al obtener ese resultado se evidencia el intercambio del orden de los números.

Para solucionar este problema dentro de la aplicación Gplad, se hizo uso de bloques de variables enteras, ciclos, expresiones e impresiones.

Fig. 8. Implementación del ejercicio dos en Gplad

De la misma forma que ocurrió con el ejercicio uno, se genera una representación intermedia en XML correspondiente al problema implementado en Gplad y ocurre el mismo procedimiento con los agentes para la traducción y compilación del código generado en la Fig. 8, para obtener el número invertido del entero 513.

3. RESULTADOS

Ejercicio uno: número primo

Fue satisfactoria la implementación del problema para determinar si un número es primo o no lo es.

En la Fig. 9, el código fue recibido sin ninguna notificación de error, el nombre y la estructura se corresponden con lo que se esperaba obtener. Al ejecutarlo desde el computador el resultado obtenido arrojó una respuesta válida ante el problema planteado.

```
import java.io.BufferedReader;
 import java.io.InputStream;
 import java.io.InputStreamReader;
 public class NumeroPrimo {
public static void main(String args[]) {
int numero = 5 ;
int auxiliar = 0 ;
int x = 0 ;

for ( int contador = 2 ;
 (  contador  <  numero  );contador = contador + 1  ){
auxiliar = numero%contador ;
if (  auxiliar  ==  0  ){
x = 1 ;
}}if (  x  ==  1  ){
System.out.print(" NO es primo  " + numero );
}else { System.out.print(" SI es primo  " + numero );
}}}
```

Fig. 9. Código generado en Java del ejercicio uno por Gplad

La variable ingresada se había determinado que fuera el número 5, y como se observa en la Fig. 10, y el resultado obtenido dentro del dispositivo móvil coincidió con el obtenido al ejecutar el código Java generado dentro de la herramienta Eclipse. El resultado fue que el número 5 sí es un número primo y es un resultado correcto que satisface la solución del problema. Para probar que el código generado funciona con cualquier cantidad de números y valores se realizó una prueba con los números 1, 3, 7, 11, 84, 99, 100 y funcionó correctamente con cada una de las pruebas.

Ejercicio dos: número invertido

Después de implementar los bloques en Gplad e indicar que se desea la traducción en Java, se recibe su código y resultado dentro del dispositivo móvil.

El código generado por la aplicación en el lenguaje objetivo (Java), ilustrado en la Fig. 10, no presenta ningún error de sintaxis o semántica, lo que permite afirmar que el resultado fue satisfactorio. La variable del número para invertir fue el valor 513, como se indica en el bloque llamado número de la Fig. 8. El resultado obtenido dentro del dispositivo móvil Android coincidió con el creado al ejecutar el código Java que se observa en Fig. 10

```
import java.io.BufferedReader;
 public class NumeroInvertido {
public static void main(String args[]) {
int numero = 513 ;
int invertido = 0 ;
int restante = 0 ;
int respaldo = numero ;

while (  numero  >  0  ){
restante = numero%10 ;
numero = numero/10 ;
invertido = invertido*10+restante ;
}System.out.print(" \n numero ingresado  " + respaldo );
System.out.print(" \n numero invertido  " + invertido );
}}
```

Fig. 10. Código generado en Java del ejercicio dos por Gplad

Para validar de manera reiterativa se hicieron pruebas con números como: 2, 12, 406, 111, 202, 798 y no presentaron ningún inconveniente, lo que permite afirmar que la solución planteada fue válida.

4. TRABAJOS FUTUROS

Gplad es generadora de código Java, lenguaje sustentado como uno netamente objetual, por lo que será posible adaptar esta aplicación para que permita la POO en un futuro cercano. Adicionalmente, se está trabajando en complementar la aplicación, de tal manera que soporte vectores, estructuras matemáticas y trigonométricas, además del paradigma de la Programación Orientada a Objetos -POO-.

5. CONCLUSIONES

Tras haber realizado las pruebas de programación dentro de Gplad, se pudo demostrar que la aplicación permite a los usuarios solucionar problemas de software por medio de bloques en tiempo real sobre dispositivos móviles Android, lo que la convierte en una aplicación innovadora frente a herramientas de programación visual como Scratch, Alice, StarLogo TNG que están hechas para trabajar desde computadores.

Los usuarios de esta herramienta deben tener conocimiento en temas de programación básica o lógica de resolución de problemas a diferencia de los que utilicen Scratch o Alice que están orientados para jóvenes sin conocimiento alguno en estos temas; esto no implica que para hacer uso de Gplad deban conocer la sintaxis de un lenguaje como Java, puesto que la aplicación a través de sus figuras encajan en únicos bloques, lo cual les facilita la abolición de errores y evita que se invierta tiempo en la corrección de errores sintácticos y semánticos.

REFERENCIAS

[1] Aradas, A. La programación de computadoras es "el latín del siglo XXI". Disponible en: <http://www.bbc.co.uk/mundo/noticias/2012/03/120307_tecnologia_ programacion_aa.shtml/>. Acceso: 13 Abr. 2012.

[2] Johnsgard, K. & McDonald, J. (2008). Using Alice in overview courses to improve success rates in programming I. 8.

[3] Kelleher, C; Pausch, R. Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers, tech. report CMU-CS-03-137, School of Comp. Science, Carnegie Mellon, 2003.

[4] Xiajian, C.; DANLI, W.; HONGAN, W. Design and Implementation of a Graphical Programming Tool for Children. IEEE International Conference on Computer Science and Automation Engineering - CSAE, 2011, p. 572-576.

[5] Mohamad, H.; Patel, A.; Latih, R.; Qassim, Q.; Na, L. Y. Tew. Block-based Programming Approach: Challenges and Benefits, Conference: International Conference on Electrical Engineering and Informatics – ICEEI, pp. 1-5, 2011.

[6] Wang, K.; McCaffrey, C.; Wendel, D.; Klopfer, E. 3D Game Desing with Programming Blocks in StarLogo TNG. In ICLS: Proceedings of the International Conference on Learning Sciences. International Society of the Learning Sciences. 2006. p.1008-1009.

[7] APPFOUR GMBH. (2012). Recuperado el 6 de agosto de 2012, de AIDE - Android Java IDE: https://play.google.com/store/apps/details?id=com.aide.ui&hl=es

[8] Catroid. Recuperado el 24 de febrero de 2013, Obtenido de http://www.catroid.org/catroid/index/1

[9] Cárdenas, J. R. Dsl Gráfico para la solución de problemas de programación sobre dispositivos móviles Android. 2011. 151 p. Monografía (Ingeniería de Sistemas). Fundación Universitaria San Martín, Bogotá, 2011.

[10] MIT. StarLogo TNG. Recuperado el 19 de octubre de 2012, de http://education.mit.edu/projects/starlogo-tng

[11] JADE. Java Agent Development Framework. Disponible en: < http://jade.tilab.com/>, Acceso 25 de Agosto de 2012.

[12] Maloney, J. et al. The Scratch Programming Language and Environment. ACM Transactions on Computing Education, New York, vol. 10, n. 4, p. 1-15, Nov. 2010.

[13] Apóstol, T. (2012). Introducción a la teoría analítica de números. Editorial Reverté. p. 19.