Un Entorno de Desarrollo Integrado para Dispositivos Móviles

Nahuel Palumbo[1,2], Ivan Jawerbaum[1]

[1] Fundación Uqbar, Ciudad de Buenos Aires, Argentina
[2] Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, Lille, France
{nahuel.palumbo,ivojawerbaum}@gmail.com

Resumen Las aplicaciones para programar desde dispositivos móviles no ofrecen actualmente el mismo soporte que los entornos de desarrollo integrado (IDE) para computadoras. La falta de este tipo de herramientas vuelven a estos dispositivos inviables para poder construir programas de manera práctica.
En este trabajo presentamos Wollok Mobile, un IDE para programar desde dispositivos móviles con las funcionalidades clásicas de un IDE. Nuestra aplicación es un prototipo funcional que sirve para desarrollar programas en Wollok, un lenguaje de programación educativo orientado a objetos desarrollado en Argentina.
Este proyecto fue resultado de un trabajo en conjunto con docentes universitarios que decidieron involucrarse con el proyecto siguiendo la filosofía *open source*.

Keywords: IDE · dispositivos móviles · POO · OOP · smartphones · mobile.

1. Introducción

Para programar desde computadoras se usan entornos de desarrollo integrados (IDE) que ayudan a los programadores a interactuar con sus programas. En el mundo de los dispositivos móviles, las aplicaciones que se pueden encontrar en las tiendas son principalmente educativas [2,1,7]. Como mostramos en este trabajo, estas aplicaciones no ofrecen el mismo soporte que los IDEs actuales. La falta de este tipo de herramientas vuelven a estos dispositivos inviables para poder construir programas de manera práctica.

Las principales contribuciones de este trabajo son:

- Un análisis de las características de las aplicaciones para programar en dispositivos móviles en comparación con IDEs tradicionales. (Sección 2)
- Wollok Mobile: Un prototipo de IDE práctico para dispositivos móviles que sirve para desarrollar programas en Wollok. (Secciones 3 y 4)
- Una evaluación con docentes universitarios que expusieron tanto ventajas como limitaciones de nuestra propuesta. (Secciones 5 y 6)

2. ¿Qué herramientas usamos para programar?

Clasificamos algunas de las aplicaciones aún disponibles estudiadas por Camposagrado et al. [1] en tres grupos: Basadas en texto, en las cuales se programa textualmente similar a una computadora; Basadas en bloques, en donde el programa se arma encastrando bloques; e IDE mobile, donde se interactúa con el programa a través de una interfaz diseñada para dispositivos móviles.

	Edición	Visualización	Reporte de errores	Ejecución y debugging
Computadoras				
IDEs Tradicionales *(Eclipse, IntelliJ, VSCode)*	Textual con herramientas de reescritura, formateo y autocompletado	Código con *highlighting* y diferentes visualizaciones del AST	Previo a ejecutar, sobre el código y como listado de errores	Ejecución y depuración de programas y tests automatizados
Dispositivos móviles				
Basadas en texto *(Cxxdroid, Code Editor, JVdroid)*	Textual con plantillas de código predefinido y herramientas de formateo	Código con *highlighting*	Al momento de ejecutar (limitado)	Ejecución de programas
Basadas en bloques *(Pilas Bloques, Scratch, Code.org)*	Arrastrando bloques	Basado en bloques	Previo a ejecutar, sobre los bloques	Ejecución de programas animados, posibilidad de "paso a paso"
IDE mobile ***(Wollok Mobile**, Pocket Code, TouchDevelop)*	A través del AST con herramientas de autocompletado	Pantallas particulares para cada nodo del AST	Previo a ejecutar, sobre las vistas del AST y como listado de errores	Ejecución y depuración (limitada) de programas y tests automatizados

Cuadro 1. Cuadro comparativo entre las funcionalidades de un IDE tradicional para computadoras y aplicaciones para programar desde dispositivos móviles.

El Cuadro 1 compara las herramientas y funcionalidades que ofrecen los IDEs tradicionales respecto a los grupos identificados. La comparación se basa en cuatro áreas principales: Edición y Visualización de Programas, Reporte de errores y Ejecución y debugging.

Con este cuadro remarcamos que para poder programar de forma práctica desde dispositivos móviles es necesario ofrecer aplicaciones con las mismas herramientas que los IDEs tradicionales y con una interfaz diseñada para dichos dispositivos.

3. Wollok

Wollok[3] es un lenguaje de programación orientado a objetos diseñado con fines didácticos en Argentina por la Fundación Uqbar, al igual que el presente trabajo. Un ejemplo de su sintaxis puede verse en Listing 1.1.

```
class Golondrina {
  var energia = 100
  method comer(comida) {
    energia = comida.calorias()
  }
}
```

Listing 1.1. Definición de una clase *Golondrina* en Wollok. Define un atributo *energia* y un método *comer* donde se envía el mensaje *calorias* al parámetro *comida*.

Wollok es usado en distintas carreras de informática de varias universidades nacionales de Argentina en cursos de introducción a la programación orientada a objetos [5,8]. Los programas desarrollados con este lenguaje son de carácter educativo o lúdico, creados por docentes y estudiantes.

4. Wollok Mobile: Un IDE para dispositivos móviles

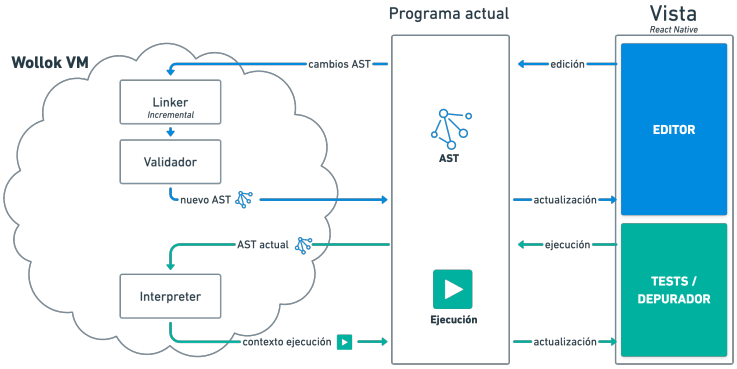

Figura 1. Arquitectura de Wollok Mobile. La vista actúa sobre el AST del programa actual y usa la VM de Wollok para validarlo y ejecutarlo.

Wollok Mobile es un entorno de desarrollo integrado para dispositivos móviles que sirve para desarrollar programas orientados a objetos en Wollok. El prototipo es una aplicación *open source*[4], desarrollada sobre el framework React Native, y

[3] https://www.wollok.org/
[4] https://github.com/uqbar-project/wollok-mobile

soportada por dispositivos Android y iOS. Una versión temprana pero funcional ya se encuentra disponible en Play Store de Android[5].

4.1. Creación y edición de programas

Nuestra propuesta se basa en manipular el *Abstract Syntax Tree* (AST) de los programas, sin necesidad de una representación textual [4]. Las entidades se crean y editan presionando botones y completando formularios, similar a las aplicaciones de uso cotidiano [3].

Ante cada cambio en el programa nuestro IDE ejecuta todas las validaciones del lenguaje automáticamente como se presenta en la Figura 1. Dado que no utilizamos sintaxis, todas las validaciones son semánticas sobre el AST del programa.

En las primeras dos pantallas de la Figura 2 se observan una clase, listando sus atributos y métodos, y el listado de tests, con sus resultados, respectivamente. Las siguientes dos pantallas muestran cómo se crean y editan las sentencias y expresiones (el "código") dentro de un test o método, a partir de un conjunto de referencias accesibles o bien una lista de posibles mensajes a enviar.

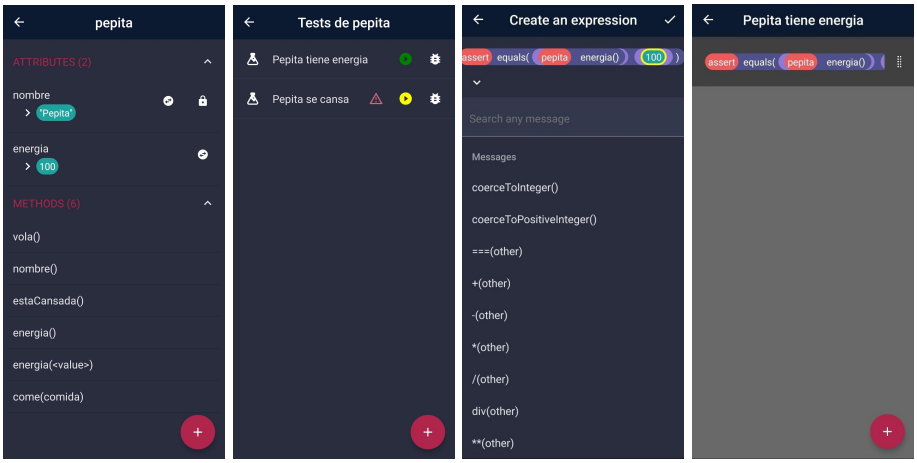

Figura 2. Pantallas de edición de una entidad, resultados de tests y creación de un test con envío de mensajes.

4.2. Ejecución y Depuración

Lo programas construidos con Wollok Mobile pueden ejecutarse localmente en el mismo dispositivo móvil, sin necesidad de conexión a internet. Para ello se usa una máquina virtual de Wollok construída en TypeScript (JavaScript)[6],

[5] `https://play.google.com/store/apps/details?id=com.wollokmobile`
[6] `https://github.com/uqbar-project/wollok-ts`

el mismo lenguaje en el que está construida nuestra aplicación, que se puede observar en la Figura 1.

Para ejecutar un programa existen los *Tests Automatizados*, ya incluidos en el lenguaje Wollok. Nuestra aplicación también posee algunas opciones básicas de *depuración* como ejecutar el nodo actual, continuar la ejecución y entrar o salir de la invocación de un método.

5. Evaluación

Docentes que usan Wollok estuvieron probando las distintas versiones preliminares de la aplicación. La validación de nuestra propuesta con otros docentes durante el proceso iterativo de desarrollo permitió prevenir problemas técnicos, corroborar la experiencia de usuario y recibir sugerencias de mejoras o implementación de próximas funcionalidades.

5.1. Pruebas de usuario

Se realizaron dos pruebas de usuarios en donde participaron nueve docentes en total. Una consistió en resolver un ejercicio real de una de las cursadas y la otra en explorar la aplicación sin ninguna consigna específica.

A pesar de eventuales desaciertos para encontrar determinadas opciones, todos pudieron pudieron descargar Wollok Mobile y programar algo. Solamente un usuario pudo terminar el ejercicio propuesto debido a que el resto tuvieron errores lanzados por la aplicación después de un uso prolongado (30 minutos).

Se destaco positivamente la facilidad de adaptar una herramienta como Wollok Mobile. Un participante dijo: *"Trabajar con un IDE basado en el AST me resultó más sencillo de lo que esperaba"*. La aplicación presentó problemas de performance con pausas de más de 1 segundo después de editar un método.

Además, los docentes entrevistados se sintieron interesados en el proyecto por más de no sentirse seguros sobre cómo impactará en las cursadas.

5.2. Cambios después de las pruebas

A partir de los resultados de las evaluaciones se implementaron los siguientes cambios:

- El AST se reconstruye de modo asincrónico no bloqueante para evitar largas pausas al editar el programa. De esta forma se logró mejorar la experiencia a pesar de las limitaciones de hardware.
- El creador de expresiones se rediseñó para simplificar la creción de subexpresiones. Ahora toda la sentencia es visible mientras se crea, como se muestra en la Figura 2.
- Se desarrollaron las funcionalidades que faltaban de borrado, edición o mismo reordenamiento de partes del programa.

6. Conclusiones

En este trabajo hemos presentado Wollok Mobile, un IDE para dispositivos móviles que ofrece herramientas similares a los IDEs tradicionales. Su interfaz de usuario está adaptada a los diseños estándares que se ven en las aplicaciones comerciales. Las pruebas de un prototipo funcional realizadas con usuarios expertos validaron algunas de las ventajas de nuestra propuesta y también demostraron algunas limitaciones.

7. Trabajos relacionados

Nuestras ideas se basaron en las propuestas de Pocket Code [7] y TouchDevelop [9]. Nosotros trasladamos la propuesta a un lenguaje orientado a objetos e incorporamos herramientas de desarrollo como un debugger, soporte de autocompletado y validaciones.

Otras propuestas relacionadas son las adaptaciones para móviles de lenguajes previamente basados en bloques como LEGO MINDSTORMS o Scratch, al igual que las accesibles desde sitios web *responsive* como PilasBloques [6] de producción argentina.

Referencias

1. Camposagrado, C.G.P., et al.: Assessment of elearning tools utilized by it faculty of programming course in the new normal. Turkish Journal of Computer and Mathematics Education (TURCOMAT) **12**(13), 511–516 (2021)
2. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. ACM Trans. Comput. Educ. **10**(4) (nov 2010). https://doi.org/10.1145/1868358.1868363, `https://doi.org/10.1145/1868358.1868363`
3. Mew, K.: Learning Material Design. Packt Publishing Ltd (2015)
4. Olivero, F., Lanza, M., Lungu, M.: Gaucho: From integrated development environments to direct manipulation environments. Proceedings of FlexiTools **2010** (2010)
5. Passerini, N., Lombardi, C., Fernandes, J., Tesone, P., Dodino, F.: Wollok: Language + ide for a gentle and industry-aware introduction to oop. In: 2017 Twelfth Latin American Conference on Learning Technologies (LACLO). pp. 1–4 (2017). https://doi.org/10.1109/LACLO.2017.8120933
6. Sanzo, A., Schapachnik, F., Factorovich, P., O'Connor, F.S.: Pilas bloques: A scenario-based children learning platform. In: 2017 Twelfth Latin American Conference on Learning Technologies (LACLO). pp. 1–6 (2017). https://doi.org/10.1109/LACLO.2017.8120926
7. Slany, W.: Tinkering with pocket code, a scratch-like programming app for your smartphone (2014), `http://www.ist.tugraz.at/catrobat/pub/Main/ThesesAndPublicationsAboutCatrobat/Catrobat_submission_20140427_hq.pdf`
8. Spigariol, L., Palumbo, N., Passerini, N.: Competencias en programación orientada a objetos. Electronic Journal of SADIO (EJS) **20**(1), 77–101 (2021)
9. Tillmann, N., Moskal, M., Halleux, J., Fähndrich, M.: Touchdevelop–programming cloud-connected mobile devices via touchscreen (09 2011). https://doi.org/10.1145/2048237.2048245