UNIVERSIDAD TECNOLÓGICA NACIONAL

INSTITUTO NACIONAL SUPERIOR DEL PROFESORADO TÉCNICO

Prácticas pedagógicas innovadoras y proyectos de estudiantes

Desarrollos en la Formación de Técnicos Superiores en Informática Aplicada

María Gabriela Galli (Coordinadora)

Leandro E. Colombo Viña
Diego P. Corsi
Gabriela Crespo
María Gabriela Galli
Mario Eduardo Galvao
Matías E. García
Martín Jerman
Mónica Kuhn
Ricardo Rodolfo Leithner
Leonardo Miguel Michelli
Mariana Victoria Risé

edUTecNe

Editorial de la U.T.N.

UNIVERSIDAD TECNOLÓGICA NACIONAL
INSTITUTO NACIONAL SUPERIOR
DEL PROFESORADO TÉCNICO

Desarrollos en la Formación de Técnicos Superiores en Informática Aplicada

Prácticas pedagógicas innovadoras y proyectos de estudiantes

[Versión para web]

María Gabriela Galli (Coordinadora)

Leandro E. Colombo Viña
Diego P. Corsi
Gabriela Crespo
María Gabriela Galli
Mario Eduardo Galvao
Matías E. García
Martín Jerman
Mónica Kuhn
Ricardo Rodolfo Leithner
Leonardo Miguel Michelli
Mariana Victoria Risé

2017

Galli, Maria Gabriela
Desarrollos en la formación de técnicos superiores en informática aplicada : prácticas pedagógicas innovadoras y proyectos de estudiantes / Maria Gabriela Galli. - 1a ed .
Ciudad Autónoma de Buenos Aires : edUTecNe, 2017.

Libro digital, PDF
Archivo Digital: online

ISBN 978-987-1896-75-2

1. Enseñanza. 2. Capacitación Técnica. I. Título.
CDD 371.1

Versión para imprimir:

Diseño de tapa: Sector TIC UTN-INSPT

Revisión: Diego P. Corsi

Versión para la web:

Diseño de tapa: Sector TIC UTN-INSPT

Diseño y Revisión: edUTecNe

Universidad Tecnológica Nacional Instituto Nacional Superior del Profesorado Técnico

Av. Triunvirato 3174
(C1427AAR) Ciudad Autónoma de Buenos Aires - República Argentina
Teléfono: (+54 11) 4552 4176
www.inspt.utn.edu.ar

Editorial de la Universidad Tecnológica Nacional - edUTecNe
http://www.edutecne.utn.edu.ar
edutecne@utn.edu.ar

©[Copyright]
edUTecNe, la Editorial de la U.T.N., recuerda que las obras publicadas en su sitio web son de libre acceso para fines académicos y como un medio de difundir la producción cultural y el conocimiento generados por autores universitarios o auspiciados por las universidades, pero que estos y edUTecNe se reservan el derecho de autoría a todos los fines que correspondan.

Autoridades de UTN

Rector: Ing. Carlos Hector BROTTO

Vicerector: Ing. Pablo Andrés ROSSO

Autoridades de UTN-INSPT

Director

Arq. Luis De Marco

Secretario Académico

Ing. Gustavo Franco

Secretario de Planeamiento

Mg. Ing. Emilio Vetta

Secretario Administrativo

Cr. Fabián Pelagagge

Coordinadores Académicos

Lic. Jorge Arias / Lic. Carlos Gustavo Lovallo

Coordinador de Extensión y Relaciones Institucionales

Ing. Mario Roberto Gos

Coordinador de Laboratorios e Infraestructura

Ing. Roberto Barneda

Director de la Carrera Informática Aplicada

Mg. Delmiro M. Gil

Prefacio

En la actualidad, es difícil pensar la educación sin la mediación de la tecnología digital. Según la UNESCO, las Tecnologías de la Información y la Comunicación "se están imponiendo como elementos didácticos tanto en los recintos universitarios como en los sistemas de educación superior abiertos y a distancia", incorporando variadas herramientas para compartir información y crear conocimientos, donde la ubicuidad y el trabajo colaborativo se constituyen en algunos de sus pilares fundamentales.

En la Universidad Tecnológica Nacional – Instituto Nacional Superior del Profesorado Técnico (UTN-INSPT), desde la Tecnicatura Superior en Informática Aplicada, formamos a nuestros estudiantes con el propósito de que adquieran competencias que les posibiliten la participación en actividades relacionadas con el diseño e implementación de sistemas informáticos, centrando principalmente los procesos de enseñanza y de aprendizaje en el desarrollo de programas, y utilizando diversos lenguajes.

Para el logro de estos objetivos, durante la cursada se trabaja en el análisis de datos, la resolución de problemas y la potenciación del pensamiento crítico, poniendo en práctica la creatividad y la capacidad para diseñar algoritmos frente a cada situación compleja que se presenta como desafío.

La finalidad de esta publicación radica en presentar situaciones prácticas que describen diversas actividades llevadas a cabo durante el trayecto formativo de nuestros estudiantes, en las que se cristaliza el aprendizaje de contenidos signficativos, mediado por tecnología digital.

A partir de lo expuesto, la presente publicación pone a disposición de los lectores secuencias pedagógicas innovadoras llevadas a cabo por docentes de la carrera, las que apuntan al desarrollo de la creatividad, la construcción de nuevos conceptos a partir de saberes previos, la vinculación entre el binomio teoría-práctica y el trabajo colaborativo y cooperativo, instancias de interacción que deben formar parte de los procesos formativos.

En segundo lugar, se presentan trabajos de finalización de carrera, desarrollados por nuestros estudiantes en la asignatura Seminario, en los que se manifiestan las competencias puestas en juego para su elaboración, para que les sirvan como insumo a los actuales y futuros cursantes en el diseño y desarrollo de sus trabajos.

En tercer lugar, presentamos el proyecto desarrollado por una exalumna, quien, en su ejercicio profesional, aplica algunas de las competencias adquiridas durante su formación.

Agradecemos la colaboración de los autores, tanto docentes como egresados, y de las autoridades de UTN-INSPT por brindarnos el espacio para la difusión de este trabajo, esperando que las experiencias plasmadas en la presente publicación les sean de utilidad a los lectores.

María Gabriela Galli

Contenidos

Prácticas pedagógicas innovadoras llevadas a cabo por docentes de Informática Aplicada

Proyectos de fin de carrera desarrollados por estudiantes en la materia Seminario

Ir a:

Primer proyecto desarrollado por una egresada al iniciar su ejercicio profesional

Prácticas pedagógicas innovadoras llevadas a cabo por docentes de Informática Aplicada

Robot Rocola

Programación del robot Multiplo N6 para que se le pueda solicitar una canción y este la interprete

Diego P. Corsi
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
corsi@mail.com

***Resumen* — El aprendizaje de la *programación* representa un gran desafío para los estudiantes de la carrera de Informática Aplicada de UTN-INSPT y también para los profesores responsables de su enseñanza. Sin embargo, existen prácticas docentes innovadoras que pueden allanar el camino. La *robótica educativa* abre perspectivas prometedoras ya que permite aplicar los conceptos de la programación sobre objetos del mundo real y que los estudiantes interactúen con ellos. En este artículo se presenta una experiencia llevada a cabo con estudiantes del primer año de la carrera, quienes programaron el robot Multiplo N6 para que este, después de identificar un número, tocara la canción asociada a dicho número. A través de la observación de la experiencia se pudo constatar un aumento en la motivación de los estudiantes.**

Palabras clave: Robótica educativa; Introducción a la programación; Motivación.

I. INTRODUCCIÓN

Para muchos estudiantes de la carrera de Informática Aplicada de UTN-INSPT, aprender a programar es uno de los objetivos más difíciles de cumplir. Existe la percepción de que se requiere tener cierta *aptitud* para la programación, y que no poseerla conducirá irremediablemente al fracaso. Sin embargo, Scott y Ghinea [1] consideran que es precisamente la *creencia* de que se necesita una aptitud inherente para la programación lo que inhibe una práctica frecuente y deliberada de la escritura de código por parte de los estudiantes. Asimismo, Jenkins cuestiona la existencia de tal aptitud y afirma que "si un estudiante tiende a adoptar estilos de aprendizaje erróneos o su motivación es errónea, aprender a programar le resultará difícil" [2].

Entre las distintas alternativas pedagógicas que pueden ser útiles para aumentar la motivación de los estudiantes, últimamente una de ellas se viene destacado sobre las demás: la *robótica educativa*. Rastreando los orígenes de esta disciplina, Yousuf llega hasta el trabajo pionero de Seymour Papert, quien en los años 70 construyó una tortuga programable provista de sensores y más tarde postularía la teoría conocida como *construccionismo*, según la cual "los estudiantes aprenden muy efectivamente cuando están involucrados en la creación de un objeto externo que habita en el mundo real" [3]. Esta teoría, a su vez, tiene sus raíces en el *constructivismo* de Jean Piaget, para quien "el aprendizaje resulta de un proceso activo de construcción de conocimientos, los cuales pueden obtenerse a través de experiencias de la vida real y ser vinculados al conocimiento previo del aprendiz" [3].

Además del impacto positivo que el trabajo con robots puede tener sobre la motivación de los estudiantes, de acuerdo con Karatrantou y Panagiotakopoulos:

En los ambientes de robótica educativa, los estudiantes adquieren habilidades generales (por ejemplo, trabajo en equipo, pensamiento crítico, planificación y observación científica) y conocimientos científicos de campos como la ciencia experimental y la tecnología. La robótica educativa también presenta a los estudiantes conceptos avanzados de los campos de la simulación, la inteligencia artificial y la cognición. En estos ambientes de aprendizaje, los estudiantes descubren y utilizan diferentes nociones relacionadas con conceptos científicos, lenguajes de programación y tecnología, a través de un enfoque educativo interdisciplinario [4].

La utilización de robots para la enseñanza de lenguajes de programación de uso general es una práctica más común en los niveles secundario y superior. Por ejemplo, tanto Gomes y Abrantes [5] (en el nivel secundario) como Seung Han y Jae Wook [6] (en el nivel universitario) emplean robots para enseñar a programar en el lenguaje C.

En la República Argentina, la robótica educativa comenzó a popularizarse primero en las escuelas técnicas y las universidades, principalmente en aquellas con orientación en electrónica, en la última década del siglo XX, y gradualmente fue haciéndose presente en todo el sistema educativo. Las Olimpíadas Argentinas de Robótica se llevan a cabo desde el año 2000 y han sido declaradas de interés del Honorable Senado de la Nación [7]. Desde 2011, el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación viene entregando kits de robótica en escuelas, en el marco del programa *Robótica para educar* [8]. En 2015, el Ministerio de Educación de la Nación llevó adelante el festival *Liber.ar*, en el que más de 30 escuelas presentaron sus trabajos y experiencias en programación y robótica con software libre [9].

En UTN-INSPT, desde el año 2011 contamos con robots que se encuentran a disposición de los docentes de Informática Aplicada, para ser usados en las materias de la carrera. La experiencia que se narra en este artículo se llevó a cabo en un curso de Programación I en el primer cuatrimestre de 2016, trabajando en equipos con robots Multiplo N6 (Fig. 1).

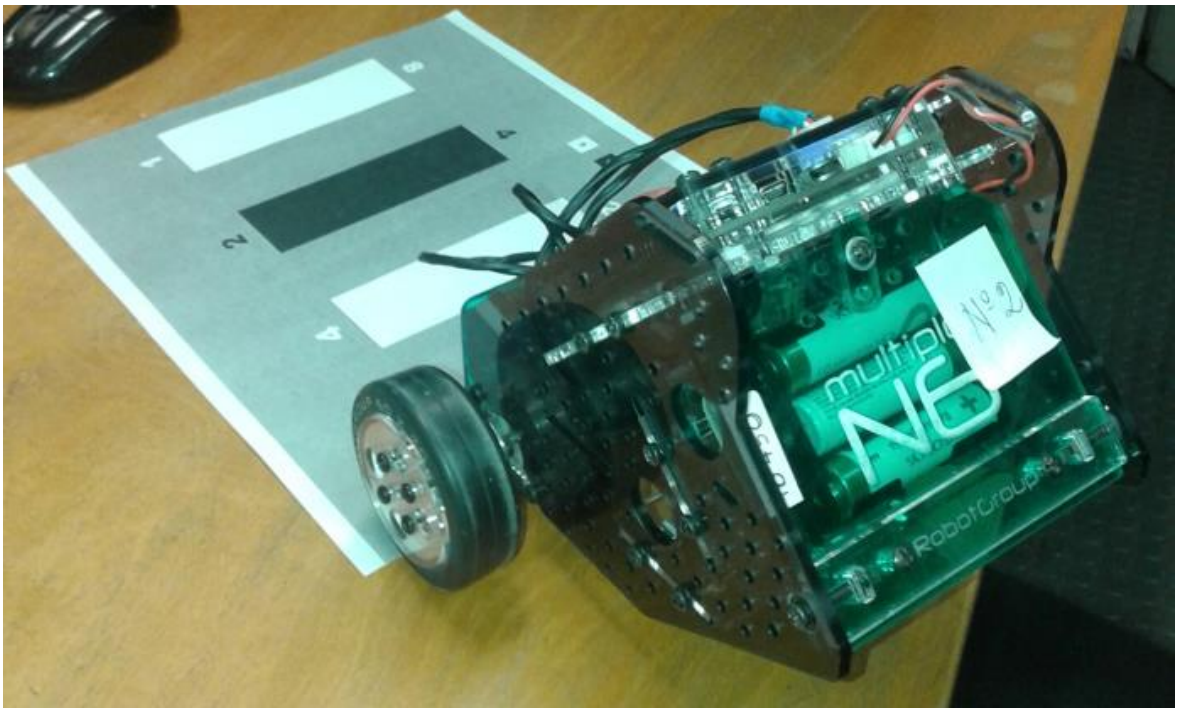


Fig. 1. Robot Multiplo N6

El contenido que se había dictado antes de realizar la experiencia era el manejo de arreglos en C/C++, por lo cual, en principio, se decidió utilizar los robots para aplicar este concepto, aunque la mayoría de los contenidos anteriores, como, por ejemplo, los tipos de datos primitivos, las estructuras de control y la definición de funciones, también se aplicarían al programar los robots.

Como objetivos generales de la experiencia, se esperaba que los estudiantes lograran:

- conocer los pasos a seguir para programar y poner en funcionamiento el robot Multiplo N6;
- entender la importancia de utilizar arreglos para almacenar una gran cantidad de datos homogéneos en memoria;
- comprender la equivalencia entre distintos sistemas de numeración.

También se esperaba, como objetivos específicos de la experiencia, que los estudiantes consiguieran:

- describir un algoritmo que permitiera obtener un número binario de cuatro dígitos, correspondiente a un número decimal del intervalo [0..15], a partir de la entrada continua de valores leídos de un sensor, durante el recorrido del robot sobre una hoja gris provista de cuatro franjas blancas o negras;
- buscar en la Web diferentes canciones expresadas en notación musical, y reescribirlas como secuencias de frecuencias y duraciones;
- desarrollar un programa para el robot, declarando arreglos de enteros y cargando en ellos las canciones encontradas, e implementando, además del algoritmo mencionado anteriormente, una función para recorrer uno de los arreglos tocando la canción que este contuviera, nota por nota, utilizando para ello el zumbador (también conocido como *buzzer*) del robot.

II. DESARROLLO

Para comenzar la experiencia, se mostró a los estudiantes el entorno de programación *Arduino* [10], el cual está inspirado en *Wiring* [11], que a su vez es un derivado de *Processing* [12]. Aunque en Arduino se utiliza la sintaxis de C/C++, los programas tienen una estructura especial, ya que existen dos funciones (`setup` y `loop`) dentro de las cuales el programador debe colocar, respectivamente, el código que se ejecutará una única vez (para llevar a cabo inicializaciones) o repetidas veces (para determinar el funcionamiento del robot hasta su apagado)

A continuación, se explicaron los pasos necesarios para compilar un programa y transferirlo al robot a través del cable USB. Se utilizó para ello uno de los varios ejemplos que trae Arduino entre sus archivos: `MotorSimpleLine`. En este programa se muestra cómo hacer girar las ruedas del robot a determinada velocidad, cómo esperar durante cierto tiempo y cómo frenar. Los estudiantes pudieron probar diversos valores de velocidad para que los robots avanzaran en línea recta o giraran en círculos.

Luego se pasó a estudiar otro ejemplo: `LineFollower`. En este caso, el robot utiliza sus sensores frontales (Fig. 2) para obtener valores relacionados con el color del piso sobre el que se mueve (los valores cercanos a 0 indican una superficie oscura y los colores cercanos a 1000 indican una superficie blanca) y así es capaz de recorrer cualquier circuito indicado por una línea negra lo suficientemente ancha como para que, en todo momento, ambos sensores puedan quedar ubicados sobre ella. Si, durante el recorrido, en alguno de los sensores (o incluso en ambos) se deja de obtener un valor perteneciente al intervalo que se considera como negro, significa que el robot se ha apartado de la línea, y el algoritmo ejecutado por el robot determina que una de las ruedas debe frenarse para corregir el rumbo, mientras que la otra debe mantener su velocidad actual, con lo cual eventualmente ambos sensores vuelven a quedar sobre la línea negra, y así el robot continúa su marcha recorriendo el circuito.

Fig. 2. Sensores frontales del Robot Multiplo N6

Una vez entendidos los conceptos aplicados en el ejemplo anterior, se les entregó a los estudiantes un conjunto de hojas grises con cuatro franjas blancas o negras en cada una. Haciendo que las franjas blancas representasen el dígito cero del sistema binario y las negras el uno, cada hoja contenía un valor decimal del intervalo [0..15], como puede observarse en la Tabla I.

TABLA I
CORRESPONDENCIA ENTRE LOS SISTEMAS DE NUMERACIÓN

BINARIO	DECIMAL
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	10
1 0 1 1	11
1 1 0 0	12
1 1 0 1	13
1 1 1 0	14
1 1 1 1	15

Los estudiantes tuvieron que describir, entonces, un algoritmo para que el robot, a medida que fuera avanzando sobre una de las hojas (Fig. 3), pudiera detectar que su sensor estaba entrando o saliendo de una franja y, basándose en el color de esta y en su posición (es decir, si era la primera, segunda, tercera o cuarta franja), ir calculando el valor en decimal del número binario representado. Después de describir el algoritmo, debieron codificarlo en C/C++, utilizando para ello el entorno de programación Arduino.

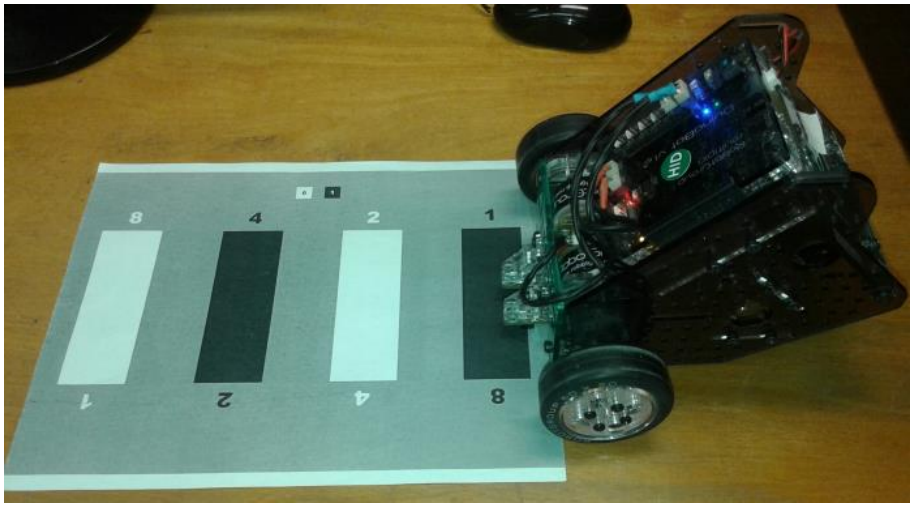


Fig. 3. Robot Multiplo N6 a punto de leer el número 5

Con el objetivo de verificar que el programa funcionara correctamente, en esta etapa de la experiencia los estudiantes hicieron que el robot, después de pasar sobre la hoja, emitiera un sonido la cantidad de veces correspondiente al número leído. Utilizaron para ello la función toneDelay, que recibe tres argumentos: el número de pin del zumbador (se puede usar directamente la constante SPEAKER), la frecuencia en Hz del sonido a tocar y su duración en milisegundos. Para hacer una pausa, se utilizó el valor de frecuencia cero. Para la duración, tanto de los sonidos como de las pausas entre ellos, se usó la constante 500 (es decir, medio segundo).

Para que los estudiantes pudieran programar las canciones que tocaría el robot, se les dio una introducción a la notación musical, empezando por los nombres de las notas musicales y su posición en el pentagrama (Fig. 4), usando la *clave de sol* (para notas agudas) y la *clave de fa* (para notas graves).

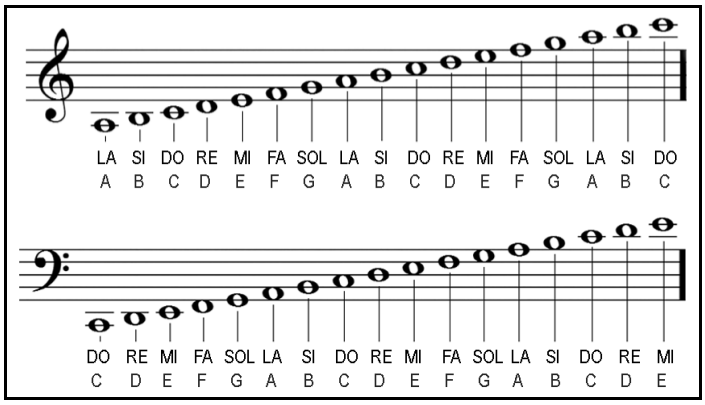


Fig. 4. Nombre de las notas musicales en clave de sol y en clave de fa

Se mencionó que en el mundo occidental contemporáneo se utiliza una *escala cromática o dodecafónica*, en la que una *octava* (el intervalo entre dos notas cuyas frecuencias tienen una relación de dos a uno, por ejemplo el intervalo entre un *do* y el siguiente *do* más agudo) se divide en doce *semitonos* mediante el agregado de cinco notas alteradas (Fig. 5). En sostenido (♯) y el bemol (♭) alteran las notas musicales naturales ubicadas inmediatamente a su derecha, subiéndolas o bajándolas un semitono, respectivamente, así como todas las notas del mismo nombre y altura que haya hasta la siguiente barra de compás. Su efecto, sin embargo, se cancela automaticamente cuando aparece un becuadro (♮).

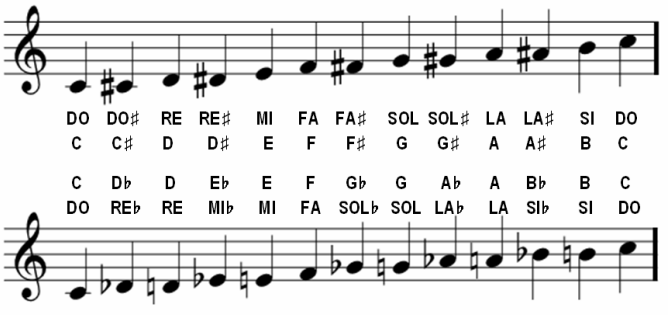


Fig. 5. Notas naturales y alteradas en una octava

También se aclaró que, en la práctica, las *alteraciones accidentales* como las anteriores se combinan con *alteraciones propias* indicadas en la *armadura de clave* (Fig. 6). El efecto de la armadura se extiende a lo largo de toda la canción, salvo que sea expresamente cancelado mediante una nueva armadura, un becuadro u otra alteración accidental, y también afecta las notas en las octavas superiores e inferiores.

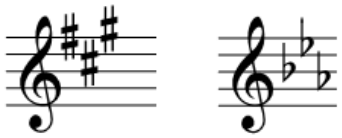


Fig. 6. Armaduras de clave

Luego se explicó que el sistema de afinación más utilizado actualmente es el *temperamento igual*, cuyas frecuencias conforman una progresión geométrica en la cual la razón es la raíz 12 de 2. Los estudiantes tuvieron que decidir cómo mantener guardadas en la memoria las frecuencias en hercios de las notas musicales que se muestran en la Tabla II. La opción elegida fue definir, para cada frecuencia, una *macro* con el nombre de cada nota y su octava, por ejemplo: D8 (nota *re* de la octava más aguda) o FS0 (nota *fa sostenido* de la octava más grave).

TABLA II
Frecuencia de las notas musicales (en HZ)

Nota	Octava								
	0	1	2	3	4	5	6	7	8
C	16	33	65	131	262	523	1047	2093	4186
C♯	17	35	69	139	278	554	1109	2218	4435
D	18	37	73	147	294	587	1175	2349	4699
D♯	20	39	78	156	311	622	1245	2489	4978
E	21	41	82	165	330	659	1319	2637	5274
F	22	44	87	175	349	699	1397	2794	5588
F♯	23	46	93	185	370	740	1475	2960	5920
G	25	49	98	196	392	784	1568	3136	6272
G♯	26	52	104	208	415	831	1661	3322	6645
A	28	55	110	220	440	880	1760	3520	7040
A♯	29	58	117	233	466	932	1865	3729	7459
B	31	62	124	247	494	988	1976	3951	7902

A continuación, se indicó que la duración de las notas y los silencios se expresa mediante las *figuras musicales* que se muestran en la Fig. 7. Aunque no se lo puede ver aquí, las *plicas* de las figuras musicales pueden aparecer hacia arriba o hacia abajo de la *cabeza* de la figura (exceptuando el caso de la *redonda*, que no tiene plica). Como indican las fracciones, dos *blancas* duran lo mismo que una *redonda*, dos *negras* lo mismo que una *blanca*, y así sucesivamente.

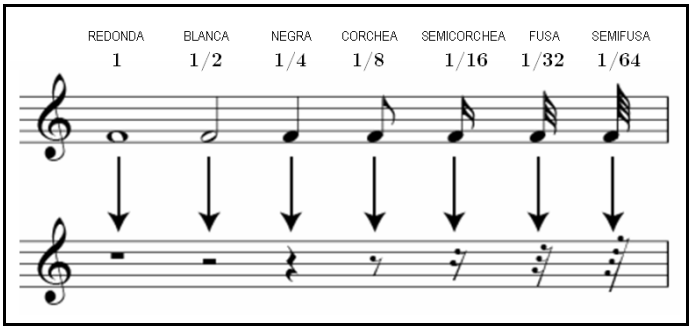


Fig. 7. Figuras musicales

Se explicó que otras duraciones son posibles mediante la utilización del *puntillo* (que aumenta la duración en la mitad de la figura original) o también de la *ligadura* (que une dos figuras que estén a la misma altura en el pentagrama y establece que deben tocarse como una única nota), como se ve en la Fig 8.

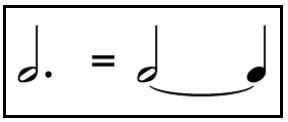


Fig. 8. Puntillo y ligadura

Además, se aclaró que, aunque dos o más corcheas, semicorcheas, fusas o semifusas, ubicadas una al lado de la otra dentro del mismo compás, aparezcan conectadas con líneas gruesas (Fig. 9), igualmente continúan siendo notas que deben ser tocadas por separado.

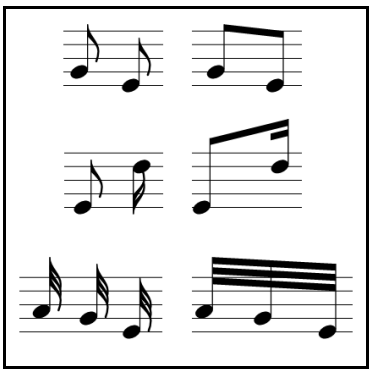


Fig. 9. Dos corcheas, una corchea y una semicorchea, tres fusas

Para finalizar la introducción a la notación musical, se mencionó que la duración de las figuras no es absoluta y depende del *tempo*, el cual hace referencia a la velocidad con que debe ejecutarse una pieza musical. Antiguamente, lo habitual era describir el *tempo* mediante una o más palabras en italiano (como *adagio*, *andante*, *allegro* y *presto*). Hoy en día, sin embargo, lo habitual es utilizar la *indicación metronómica*, que establece la velocidad de interpretación indicando cuántas figuras de un determinado valor deben tocarse en un minuto. En el ejemplo que se muestra en la Fig. 10, en un minuto deben caber 60 negras, por lo cual, en este caso, la duración de esta figura será de un segundo, las corcheas durarán 500 milisegundos, etc

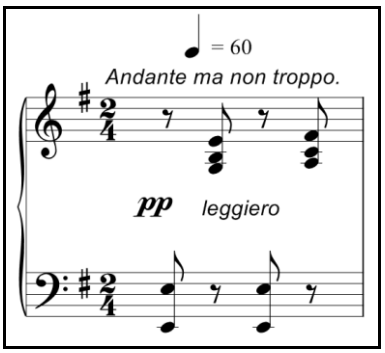


Fig. 10. Indicación del *tempo*

La información proporcionada, aunque no abarcó toda la notación musical completa, fue suficiente para que los estudiantes, a continuación, pudieran buscar en la Web algunas partituras y reescribir las canciones como secuencias de frecuencias y duraciones.

Por último, los estudiantes modificaron el programa que habían realizado antes, para que el robot, después de pasar sobre una hoja, tocara la canción correspondiente al número leído. Para ello, guardaron cada canción en un arreglo de enteros, expresándola como la indicación metronómica (negras por minuto), seguida de las frecuencias (en hercios) y las constantes de figuras (los denominadores de la Fig. 7) intercaladas. De esta forma, la duración de cada nota en milisegundos se podría calcular cuando correspondiera tocarla, aplicando la fórmula que se muestra en la Fig. 11. Además, escribieron una función para recorrer uno de los arreglos tocando la canción que este contuviera, nota por nota.

$$duración = \frac{240.000}{indicación\ metronómica \times constante\ de\ figura}$$

Fig. 11. Duración de una nota en milisegundos

Un aspecto destacable de la experiencia fue que los estudiantes formaron equipos (debido a que había menos robots disponibles que estudiantes) y, a medida que iban terminando de reescribir sus canciones, las intercambiaban con los demás a través de la red local, de modo que, al finalizar, todos los equipos disponían del mismo acervo de canciones en sus robots rocolas

Cabe mencionar, sin embargo, que en ciertos casos algunos estudiantes se desviaron de la tarea asignada y, de dos maneras distintas, fueron más allá de lo solicitado. En la mayoría de estos casos, los estudiantes buscaron en la Web canciones que ya estuvieran programadas en Arduino u otro sistema y, adaptándolas si era necesario, las incorporaron a los programas de sus robots. Pero hubo también una minoría que programó sus canciones sin usar una partitura ni buscarlas en la Web, escribiéndolas directamente como listas de frecuencias y duraciones. Evidentemente, estos últimos eran estudiantes que tenían conocimientos previos de música.

Algunas de las canciones disponibles en las versiones terminadas del *robot rocola* fueron:

- Super Mario Bros. Theme;
- Itchy & Scratchy Theme (Canción de *Tomy y Daly*);
- The Imperial March (*Darth Vader's Theme*);
- Thriller;
- Stairway to Heaven;
- Jingle Bells;
- Happy Birthday to You.

III. CONCLUSIÓN

La experiencia demostró que trabajar con tecnología digital, en particular con *robótica educativa*, es una alternativa muy interesante, no solo por su potencial como elemento motivador, evidenciado en la variedad de canciones que los estudiantes programaron, sino también por las competencias que consiguieron desarrollar, entre las cuales se destacan:

- *Capacidad creativa*, evidenciada al describir algoritmos;
- *Capacidad de aplicar conocimientos a la práctica*, evidenciada al escribir el programa para el robot;
- *Capacidad de relacionar diversas áreas de estudios*, evidenciada al aplicar conocimientos de progresiones geométricas para calcular las frecuencias;
- *Capacidad de investigación en diversas fuentes*, evidenciada al buscar partituras en la Web;
- *Capacidad de aprender*, evidenciada al reescribir la música de la partitura como una secuencia de frecuencias y duraciones;
- *Capacidad para plantear y resolver problemas*, evidenciada al deducir la fórmula de la duración de las notas;
- *Capacidad para el trabajo en equipo y autónomo*, evidenciada durante toda la experiencia.

El objetivo principal de la materia Programación I es enseñar los conceptos de la *programación estructurada*, y la práctica con los robots, haciendo uso del lenguaje C/ C++, se integra perfectamente en esta asignatura. Pero más allá de la aplicabilidad del lenguaje, es importante resaltar la apertura a nuevos modos de construcción de saberes, desde una perspectiva interdisciplinar que acompañará a los futuros Técnicos Superiores en Informática en su práctica profesional.

REFERENCIAS

[1] M. J. Scott y G. Ghinea, «Educating Programmers: A Reflection on Barriers to Deliberate Practice», presentado en 2nd Annual HEA STEM Conference, Birmingham, UK, 2013.

[2] T. Jenkins, «On the difficulty of learning to program», en *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002, vol. 4, pp. 53–58.

[3] M. A. Yousuf, «Robots in Education.», en *Encyclopedia of artificial intelligence*, Information Science Reference, 2009, pp. 1383-1388. [4] A. Karatrantou y C. Panagiotakopoulos, «Educational Robotics and Teaching Introductory Programming Within an Interdisciplinary Framework», en *Research on e-Learning and ICT in Education*, A. Jimoyiannis, Ed. New York: Springer, 2012, pp. 197-210.

[5] G. Gomes y P. Abrantes, «A robótica educativa no ensino da programação», presentado en II Congresso Internacional TIC e Educação, Lisboa, Portugal, 2012, pp. 2039-2055.

[6] K. Seung Han y J. Jae Wook, «Learning C Language Using Robots», presentado en ICCAS 2005, Corea del Sur, 2005, pp. 119–122.

[7] «Declarar de interés del Honorable Senado de la Nación la realización de las Olimpíadas Argentinas de Robótica, los dias 15 y 16 de octubre en la Ciudad Autónoma de Buenos Aires.» [En línea]. Disponible en: http://www.senado.gov.ar/parlamentario/comisiones/verExp/3598.10/S/PD. [Accedido: 11-jun-2016].

[8] «¿Qué es Robótica para Educar?» [En línea]. Disponible en: http://www.casarosada.gob.ar/pdf/robotica_para_educar.pdf. [Accedido: 11-jun-2016].

[9] «Festival de Robótica y Programación Liber.ar». [En línea]. Disponible en: http://www.educ.ar/sitios/educar/noticias/ver?id=128024. [Accedido: 11-jun-2016].

[10] «Arduino». [En línea]. Disponible en: http://www.arduino.cc [Accedido: 11-jun-2016].

[11] «Wiring». [En línea]. Disponible en: http://wiring.org.co. [Accedido: 11-jun-2016].

[12] «Processing». [En línea]. Disponible en: http://processing.org. [Accedido: 11-jun-2016].

CURRICULUM VITAE DEL AUTOR

Diego P. Corsi es Magíster en Ingeniería en Sistemas de Información, Licenciado en Tecnología Educativa y Profesor en Disciplinas Industriales, especialidad Informática Aplicada. Docente en UBA (Ingeniería en Informática y Licenciatura en Análisis de Sistemas) y en UTN-INSPT (Tecnicatura Superior en Informática Aplicada). Desde 2015 cursa su doctorado en la Universidad de Extremadura.

---

volver al Índice General

El Análisis Matemático y la Programación presentes en los Videojuegos

De la experimentación al desarrollo

María Gabriela Galli
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
gabriela.galli@inspt.utn.edu.ar

***Resumen* — En este artículo se presentan dos experiencias de inclusión de videojuegos con fines educativos llevadas a cabo durante el año 2014. En la primera experiencia, los participantes (estudiantes de la materia Análisis Matemático I) usaron un juego comercial y, en la segunda, un grupo de estudiantes de la asignatura Sistemas de Computación I programó un simulador de tiro oblicuo inspirado en ese mismo juego. A partir de ambas experiencias, fue posible identificar las competencias que se desarrollan al usar y programar videojuegos.**

Palabras clave: Tiro oblicuo; Ecuaciones de trayectorias; Videojuegos; Programación; Desarrollo de competencias

I. INTRODUCCIÓN

Con el objetivo de dinamizar la enseñanza, en el año 2014 los docentes de las asignaturas *Análisis Matemático I* y *Sistemas de Computación I* del turno mañana, hemos planificado y organizado una secuencia de actividades vinculantes para que los estudiantes apreciaran la importancia de la inclusión de entornos lúdicos como medio hacia el aprendizaje de contenidos significativos, aplicaran sus saberes previos durante la indagación y la selección de información, y sumado a ello, extrapolaran sus competencias en la programación de un objeto complejo como es un *videojuego*. Asimismo, se persiguió innovar con variadas metodologías y herramientas aplicadas a los procesos de construcción de conocimientos, desde una perspectiva de *trama*, poniendo en práctica habilidades de experimentación y de desarrollo.

Particularmente, consideramos que esto es importante para los Técnicos Superiores en Informática, de la Universidad Tecnológica Nacional – Instituto Nacional Superior del Profesorado Técnico, quienes focalizan sus saberes en el área de la programación con alto grado de empoderamiento tecnológico, donde el *programar* implica la puesta en práctica de la creatividad, la capacidad para diseñar algoritmos y la habilidad de aplicar diversos conceptos (como por ejemplo, de física, de electrónica, de matemática: ejes cartesianos en dos o tres dimensiones, trigonometría, vectores, matrices, derivadas, entre otros). A partir del trabajo con videojuegos, estos se ponen en práctica y favorecen el desarrollo de habilidades cognitivas en general, como también las relativas a las Tecnologías de la Información y Comunicación, entre las cuales podemos citar: los principios de relación causa-efecto, la resolución de problemas, el trabajo con el error y, ante todo, el *aprender jugando*.

En la actualidad, los videojuegos son considerados “un instrumento tecnológico que está plenamente integrado en la sociedad, [convirtiéndose en] un vehículo de cultura … por lo que debemos aprovechar ese impulso como algo positivo [para] empezar a construir nuevos sistemas de aprendizaje” [1], ya que nos permiten abordar distintos contenidos desde diferentes asignaturas, desarrollando y potenciando competencias.

El eje conductor de las actividades fue el videojuego comercial *Angry Birds Río* [2] el cual se basa en leyes físicas, como ser el movimiento parabólico (uno de cuyos casos más conocidos es el *tiro oblicuo*); la ley de elasticidad de Hooke[1] y las leyes de Newton[2], y donde se aplican conceptos matemáticos

El juego consiste en lanzar pájaros con una gomera o resortera con el propósito de liberar las especies exóticas que se encuentran atrapadas en las jaulas (Video 1).

Video 1: Demostración de Angry Birds Rio [3]

La secuencia didáctica estuvo integrada por dos momentos: la experimentación y aplicación de saberes, por medio del juego *Angry Birds Río* y, posteriormente, la producción final mediante el desarrollo de un simulador de tiro oblicuo, en el que se integraron contenidos de matemática, física y programación.

---

1 Ley de Hooke en el juego: si se estira mucho la gomera, la fuerza será mayor y el pájaro caerá más lejos. De manera análoga, si la fuerza es menor, también lo será la distancia alcanzada.

2 Leyes de Newton en el juego: *Principio de inercia*: el pájaro en la gomera permanecerá quieto a menos que se le aplique una fuerza para que comience el movimiento. *Principio de masa*: la fuerza que se le aplica a la gomera y la aceleración que esta adquiera, son magnitudes proporcionales y su constante es la masa del cuerpo; es decir, al aplicar una fuerza, el pájaro se pone en movimiento pudiendo variar su trayectoria, ya que a mayor fuerza, mayor movimiento. *Principio de acción y reacción*: al ejercer fuerza tirando de la gomera, esta ejerce una fuerza de igual magnitud y dirección, pero en sentido opuesto sobre la primera, es decir, si se arroja el pájaro hacia arriba, con una dirección casi paralela al soporte sostén, este caerá con mayor o igual fuerza, reacción de igual intensidad y sentido contrario.

II. DESARROLLO

La secuencia de actividades propuesta está en correlación con los contenidos curriculares que trabajan los estudiantes de la carrera en Informática Aplicada. Por un lado: ecuaciones, función derivada y trigonometría, correspondientes a la asignatura *Análisis Matemático I*, y por otro: el diseño y la programación con HTML5, CSS3 y JavaScript, para el desarrollo del simulador en la materia *Sistemas de Computación I*. Nuestra propuesta es comenzar a transformar la enseñanza tradicional en espacios de innovación con la inclusión de nuevas herramientas y, a la vez, fomentar la integración de espacios curriculares, tal que se contribuya a mejorar la comprensión de los contenidos.

Lo que nos ha motivado a abordar las temáticas descriptas es la observación de que muchos estudiantes no logran ver la aplicabilidad de los conceptos enseñados y, además, el deseo de innovar con el uso de videojuegos; para que, partiendo de la experimentación, se pongan en juego conocimientos, habilidades y actitudes al dar solución a las tareas planificadas por los docentes.

Primera Etapa: la experimentación y el cálculo

Específicamente en lo relacionado con matemática, es habitual abordar la enseñanza de los contenidos referidos a *recta tangente y normal* a partir de ecuaciones y gráficos, sin aplicaciones concretas. En esta oportunidad, en cambio, se propone el uso del videojuego *Angry Birds Río* como una herramienta que posibilita aproximarse a los contenidos citados anteriormente, pero de forma contextualizada.

En la experimentación se evidencia que la trayectoria que describe el ave es parabólica (Fig. 1)

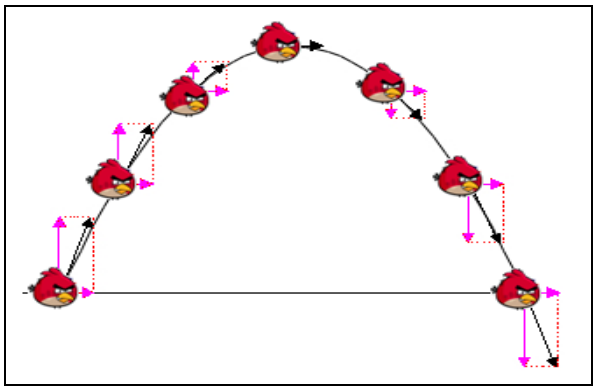


Fig. 1. La trayectoria en la descomposición de movimientos

Un movimiento de este tipo se conoce como *tiro oblicuo* y puede entenderse a partir de la composición de dos movimientos: un movimiento rectilíneo uniforme (horizontal) y otro rectilíneo uniformemente variado (vertical) (Fig. 2).

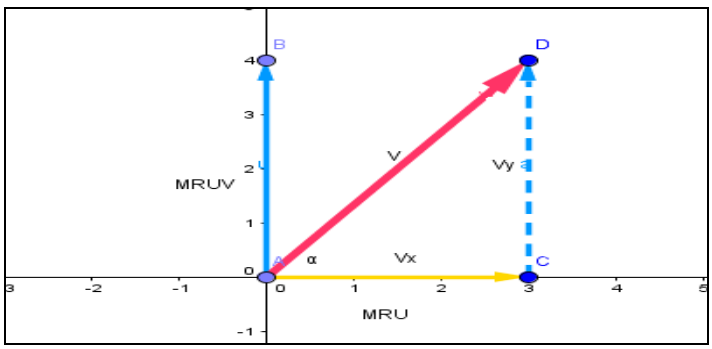


Fig. 2. Composición de Movimientos

En la Fig. 3 se muestra el ave (a la izquierda) antes de ser lanzada hacia las jaulas de las aves exóticas que deben ser rescatadas (a la derecha), y en la Fig. 4. puede observarse la trayectoria recorrida luego del lanzamiento.

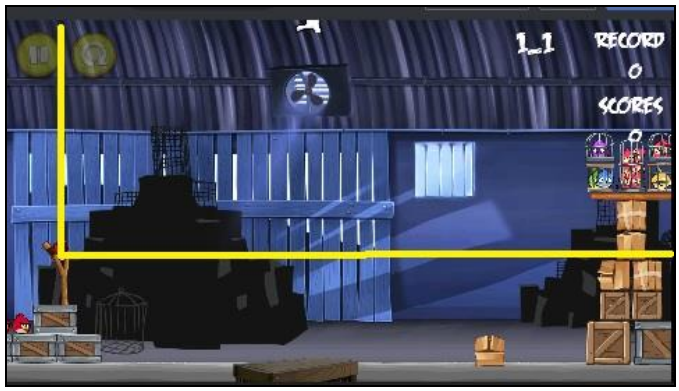


Fig. 3. Captura de pantalla anterior al lanzamiento del ave

Fig. 4. Captura de pantalla posterior al lanzamiento del ave

A partir de capturas de pantalla como las anteriores, se realizaron las siguientes actividades:

1. Reconstrucción de ecuaciones de trayectorias: $y = ax^2 + bx + c$

Ecuaciones de posición de las coordenadas horizontal y vertical:

(1) $x = v_0 \cdot \cos \alpha_0 \cdot t$

(2) $y = v_0 \cdot \operatorname{sen} \alpha_0 \cdot t - 0{,}5 \cdot g \cdot t^2$

Despejando t de (1) y reemplánzolo en (2) se obtiene que

$$y = \frac{v_0 sen\alpha_0}{v_0 cos\alpha_0} x - \frac{0{,}5g}{cos^2\alpha_0 v_0^2} x^2 \qquad y = \underbrace{tg\alpha_0}_{b} x - \underbrace{\frac{0{,}5g}{cos^2\alpha_0 \cdot v_0^2}}_{a} x^2$$

2. Ecuación horaria de la componente y de la velocidad e

Para ello, debemos recordar que cuando dos magnitudes están relacionadas funcionalmente, la derivada determina la tasa de cambio de una respecto de la otra (aplicado al movimiento rectilíneo uniformemente acelerado).st

Al ser lanzado el pájaro, el tiempo comienza a transcurrir, con lo cual t experimenta un incremento Δt. on lo cual t experimenta un incremento t.

Asimismo, el pájaro recorre cierta distancia; por ello, la posición y también varía en una cantidad Δy. Por ende, la razón de cambio indica la variación de la componente y del vector posición en el intervalo de tiempo Δt. $\frac{\Delta y}{\Delta t}$

Si calculamos $\lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t}$, obtenemos la componente en y de la velocidad instantánea alcanzada en cada punto de la trayectoria que recorre el pájaro. Por ello,

$$v_y = \lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t}$$

$$v_y = \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t}$$

Es decir, $v_y = y'(t)$

Específicamente, en nuestro caso, $v_y = v0 \cdot sen\ \alpha 0 - g \cdot t$

En conclusión, la velocidad es la derivada de la posición de una partícula, respecto al tiempo.

- v_x permanece constante y vale $v_x = v_0 \cdot \cos \alpha_0$
- El alcance máximo es $x_{máx} = v_0 \cdot \cos \alpha_0 \cdot t_{vuelo}$
- Cálculo de la v_y considerando $t = t_{vuelo}$ a partir de $v_y = v_0 \cdot \text{sen}\ \alpha_0 - g \cdot t$
- Cálculo de la $\overrightarrow{V_f} = \overrightarrow{V_x} + \overrightarrow{V_y}$

En general podríamos afirmar que la inclusión del videojuego colabora para visibilizar lo teórico en situaciones prácticas, comprender mejor los conceptos y consolidar saberes a partir de la experimentación y, ante todo, ver su aplicabilidad, tratando de responder *para qué* sirve tal o cual tema o para qué se incluye matemática en una carrera técnica. Consideramos, como escribió Roger Bacon en el siglo XIII, que "la matemática es la puerta y la llave de todas las ciencias" [4].

Segunda Etapa: el diseño de un simulador

En esta instancia se les pidió a los estudiantes que desarrollaran un simulador de tiro oblicuo, similar a *Angry Birds*, partiendo de las preguntas *¿qué es un videojuego?* y *¿cómo se programa?*, para lo cual debieron llevar a cabo diversas actividades orientadas a la elaboración del producto final (Fig. 5).

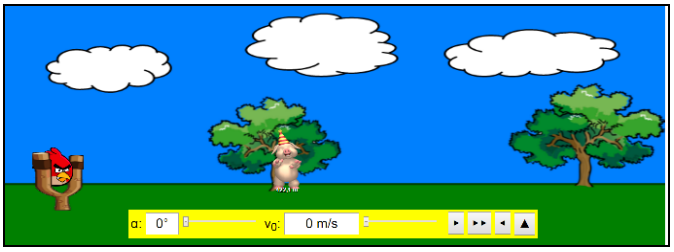


Fig. 5. Simulador de tiro oblicuo [5]

Esta propuesta está centrada en la aplicación de nociones previas de matemática y física manejadas por los estudiantes, ya que debieron aplicar las fórmulas

$$x = v0 \cdot t \cdot cos\ \alpha \qquad y = v0 \cdot t \cdot sen\ \alpha - 0{,}5 \cdot g \cdot t^2$$

Particularmente, en *Sistemas de Computación I*, para el desarrollo del juego se abordaron las siguientes temáticas: el sistema hexadecimal (al codificar los colores), las distintas resoluciones de la pantalla de video (al hacer que el juego se ajustara automáticamente), el reloj (al sincronizar las animaciones), compiladores e intérpretes (al utilizar JavaScript, ya que el código del juego es interpretado en lugar de ser compilado, como sería el caso si se hubiera optado por programarlo, por ejemplo, en C++).

III. CONCLUSIÓN

La experiencia desarrollada, ante todo, posibilitó que los estudiantes visualizaran la importancia de manipular diversas disciplinas para el desarrollo de un juego y, además, los motivó por saber que su propio producto iba a ser utilizado por otros, convirtiéndose en un desafío durante la elaboración.

El desarrollo del juego cumplió las funciones de:

- instrucción, como ampliación y refuerzo de contenidos;
- experimentación, al resolver problemas e investigar cosas nuevas;
- ser fuente de información, para los nuevos usuarios.

Asimismo, permitió abrir una discusión sobre la importancia del trabajo colaborativo y del error como *feedback* de los aprendizajes.

La experiencia nos ha demostrado que pueden incluirse juegos digitales en los espacios curriculares del nivel superior, herramientas que, como otras, implican una planificación previa y el establecimiento de acuerdos entre los participantes, a fin de efectivizar los procesos y realizar los ajustes necesarios durante su implementación

Desde la actividad profesional docente, nos permitió profundizar las investigaciones en torno a la potencialidad de los videojuegos y el posible desarrollo de competencias complejas dentro de espacios de formación tradicional. Por otro lado, sistematizamos información respecto de usos y prácticas videolúdicas de nuestros estudiantes mediante encuestas, y construimos categorías pedagógico-didácticas para el análisis de su evolución cognitiva.

Para evaluar la bondad de la experiencia como práctica educativa e identificar las competencias de aprendizaje que se han promovido, utilizamos como instrumentos el trabajo publicado por el grupo de investigación Nodo Educativo de la Universidad de Extremadura [6] y la clasificación en competencias genéricas y específicas planteada en el proyecto Alfa Tuning América Latina sobre los informáticos [7].

Como síntesis de la aplicación de ambos instrumentos destacamos que nuestros estudiantes incrementaron la capacidad de abstracción, análisis, síntesis y aplicación de conocimientos en la práctica. Se fortalecieron, además, las habilidades de organización y planificación de secuencias y el aprender a aprender. Particularmente, con el diseño del simulador y anexando competencias propias a las propuestas por Alfa Tuning América Latina, evidenciamos que se ha aplicado un enfoque sistémico en el análisis y resolución de problemas, la aplicación de conceptos matemáticos, el desarrollo de la creatividad, el dominio de conocimientos de programación, la manipulación de diversos lenguajes y la capacidad de abstracción.

De la evaluación que han llevado a cabo los estudiantes de Informática Aplicada acerca del uso de videojuegos en las prácticas áulicas surge, de la opinión general, que estos contribuyen a enriquecer el aprendizaje, pero que su uso debe ser acompañado por el docente y debe estar acorde con los contenidos que se estén desarrollando, no compartiendo el hecho de jugar por jugar, sino jugar con un fin determinado.

Esta experiencia abre en nosotros, docentes investigadores, el camino para continuar con nuevos desafíos en el área y en la exploración en torno al binomio programación/uso de videojuegos en el nivel superior, ámbito formal en cuyas prácticas pedagógicas comienzan, paulatinamente, a incluirse herramientas no tradicionales.

AGRADECIMIENTOS

Agradezco la colaboración del profesor Diego Corsi, docente de la materia Sistemas de Computación I de la Tecnicatura en Informática Aplicada, y las sugerencias de los profesores Marcelo Solimano, Martín Pérez y Carlos Dibarbora, de la Tecnicatura en Física Aplicada.

REFERENCIAS

[1] F. Revuelta Domínguez y J. Guerra Antequera, «¿Qué aprendo con videojuegos? Una perspectiva de meta-aprendizaje del videojugador.», en RED. Revista de Educación a Distancia, Número 33, 2012. [En línea]. Disponible en: http://www.um.es/ead/red/33/revuelta.pdf.

[2] «Angry Birds Rio», Rovio Entertainment. [En línea]. Disponible en: http://www.angrybirds.com/games/angry-birds-rio.

[3] «Demostración de Angry Birds Rio». [En línea]. Disponible en: http://youtu.be/ygd6S3T1c4k

[4] «Roger Bacon», en *Wikipedia, la enciclopedia libre*. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Roger_Bacon. [Accedido: 30-ago-2016].

[5] «Simulador de tiro oblicuo». [En línea]. Disponible en: http://inspt.diegocorsi.com.ar/AngryBirds

[6] J. Valverde Berrocoso (coord.), *Docentes e-competentes. Buenas prácticas educativas con TIC, Barcelona: Ed. Octaedro, 2011, pp.105-122*

[7] «Alfa Tuning América Latina: Innovación Educativa y Social (2011- 2013)». [En línea]. Disponible en: http://www.tuningal.org. [Accedido: 30-ago-2016].

CURRICULUM VITAE DE LA AUTORA

María Gabriela Galli es Especialista en Educación y TIC, Licenciada en Gestión Educativa, Técnica Superior en Informática y Profesora en Disciplinas Industriales con especialidad en Matemática. Jefa de laboratorio y docente en UTN-INSPT, y en escuela secundaria de CABA. Está doctorando en Política y Gestión de la Educación Superior.

volver al Índice General

La Matemática Aplicada y el Uso de Software en la Formación de Técnicos Informáticos

El cálculo diferencial presente en el diseño de *packaging*

María Gabriela Galli
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
gabriela.galli@inspt.utn.edu.ar

Gabriela Crespo
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
crespo.gabriela@hotmail.com

***Resumen* — El abordaje de conceptos matemáticos mediante la resolución de problemas permite contextualizar y comprender situaciones de la realidad. En este artículo se presenta una experiencia llevada a cabo con estudiantes de primer año de la Tecnicatura en Informática Aplicada, donde la importancia del hacer matemática, aplicando cálculo diferencial y uso de software, se cristaliza en el diseño del prototipo de una caja. Su resolución implicó revisar conceptos previos, elaborar estrategias, modelizar situaciones mediante representaciones gráficas, incrementar la comunicación, evaluar críticamente datos y constatar saberes.**

Palabras clave: Cálculo diferencial; GeoGebra; Problemas.

I. INTRODUCCIÓN

Muchas veces los estudiantes de la carrera de Informática Aplicada se preguntan: ¿Por qué se incluye en su estructura curricular tanta matemática? ¿Qué relación tiene con la programación? ¿Dónde se aplican los conceptos aprendidos?

Existen variadas razones que justifican su inclusión. En palabras de Devlin, diríamos que "el principal beneficio de aprender y utilizar matemáticas no son los contenidos específicos, sino el hecho de que se desarrolla la capacidad para razonar precisa y analíticamente acerca de estructuras abstractas definidas formalmente" [1].

De ahí que el uso de la matemática permita: desarrollar razonamientos lógicos, reconocer eventos, trasladar el conocimiento a otros ámbitos, traducir y ejemplificar hechos, clasificar y categorizar, inferir causas y consecuencias, constatar situaciones, enfrentarse con resolución de problemas complejos a partir de la descomposición de enunciados en partes más simples, aplicar teorías, reconocer significados, determinar patrones, formular hipótesis, verificar el valor de la evidencia, desarrollar soluciones creativas y la habilidad de sistematizar procesos. En consecuencia, la matemática se convierte en una herramienta necesaria para predecir y representar situaciones ya que, como afirma Cole, posee "el asombroso poder de explicar cómo funcionan las cosas [y] por qué son como son" [2].

Al mismo tiempo, Zalduendo manifiesta que "para hacer matemática (demostrar algo, resolver un problema) se necesitan muy pocos conceptos, pero bien definidos … [donde] será importante distinguir lo esencial de lo accesorio, buscar analogías, cambiar el punto de vista y captar relaciones escondidas" [3].

Por consiguiente, se sostiene que el *hacer matemática* se encuentra en la mayoría de las acciones que se ejecutan, y es una herramienta necesaria para abordar situaciones de la realidad, la cual nos interpela diariamente. Interpelación que estimula las habilidades de pensamiento en pos del logro de abstracción, moviliza a realizar búsquedas, conlleva el despliegue de estrategias, implica trabajar a partir de los errores cometidos, conduce a la generación de algoritmos, generaliza y optimiza procesos, y evalúa la validez de procedimientos, entre otras instancias donde se evidencian los puntos que la matemática tiene en contacto con la programación.

Dentro de las diversas temáticas que aglutina la matemática, en el presente trabajo nos centraremos en el Análisis Matemático, más específicamente en las *Aplicaciones del Cálculo Diferencial*.

La invención del Cálculo Diferencial se le atribuye a Newton (1642−1727) y a Leibniz (1646−1716) quienes desarrollaron métodos y algoritmos para abordar problemas de cálculo de áreas, volúmenes y determinación de rectas tangentes a las curvas. En el siglo XVIII, se consolidan los hallazgos y se extrapolan a otras áreas, como ser las ciencias naturales y la mecánica, de la mano de Euler (1707-1783) y Lagrange (1736-1813). Finalmente, a comienzos del siglo XIX, Bolzano (1781-1848) definió por primera vez la derivada a partir del concepto de límite y, poco tiempo después, Cauchy (1789-1857) sistematizó todos los conceptos anteriores en un cuerpo teórico, logrando su reconocimiento científico.

El Cálculo Diferencial tiene como principal objeto de estudio a las derivadas, y consiste mayoritariamente en el análisis de las variaciones de las funciones, según cambie el valor de la variable independiente. Puede utilizarse en el cálculo de las pendientes de rectas tangentes a curvas, en el análisis del comportamiento de funciones (valores máximos y mínimos, intervalos de crecimiento, puntos de inflexión, etc.), en la determinación de superficies y volúmenes, entre otras vastísimas aplicaciones de uso extensivo a diversas áreas del conocimiento.

En la actualidad, es posible encontrar informáticos trabajando colaborativamente con matemáticos y profesionales de distintas disciplinas, ya que el Cálculo Diferencial está presente en las más diversas situaciones, tales como: en la teoría electromagnética que se utiliza para diseñar los circuitos de los teléfonos inteligentes y las computadoras; en el procesamiento avanzado de imágenes digitales con el propósito de dotarlas de mejor nitidez mediante el operador gradiente; en la inteligencia artificial y las redes neuronales, cuyos algoritmos utilizan funciones derivadas; en las leyes del movimiento de Newton; en las aplicaciones de flujo de calor, en la resistencia de los materiales; en la optimización de costos, beneficios y productividad económicos; en el análisis delcrecimiento o decrecimiento poblacional; en medicina, en las resonancias magnéticas; en química, donde las ecuaciones diferenciales representan estados en transición, entre otras.

A partir de lo expuesto, proponemos que, dentro de las diversas estrategias didácticas que pueden implementar los docentes con sus estudiantes informáticos en el abordaje de la temática Cálculo Diferencial, se recurra a la resolución de problemas y la utilización de software como instancias para la contextualización de situaciones y la producción de significados, tratando de que se evidencie para qué sirve tal o cual tema, en pos de la creación de ámbitos de aprendizaje participativos e innovadores.

Particularmente la utilización de software matemático se convierte en una herramienta para la experimentación, a partir de la cual se pueden resolver múltiples problemas y desarrollar aplicaciones, ofreciendo la posibilidad de aprender y hacer matemática de forma más interactiva. Al respecto, Borba [4] señala que los ambientes de aprendizaje mediados por tecnología digital favorecen el abordaje experimental que motiva a los estudiantes a formular, verificar o rechazar y reformular hipótesis, anticipar resultados y trabajar con gráficos.

Las potencialidades del uso de estos últimos radican tanto en la visualización como en su operatividad algebraica, en la conjetura de conclusiones a partir de la instantaneidad con que se obtienen las soluciones y en la construcción de conocimientos a partir del aprender haciendo, situaciones que en la clase tradicional con pizarrón demandarían mucho tiempo de resolución.

La idea es que, cuando se trabaje con computadoras como herramienta cognitiva, se mejoren las capacidades a partir de la interacción [5].

Desde nuestra perspectiva, el éxito de una propuesta pedagógica no reside en el recurso empleado, sino en la actividad que lo sustenta, en analizar para qué y por qué se plantea la necesidad de su inclusión, tal que le permita al estudiante construir conocimientos significativos.

Sobre la base de lo expuesto, en la presente actividad introductoria se ha tenido como propósitos cristalizar la aplicabilidad del Cálculo Diferencial, contribuir con el desarrollo de habilidades de pensamiento y mejorar la comprensión, donde queda en manifiesto que la tecnología sirve como "herramienta de construcción del conocimiento, para que los estudiantes aprendan con ellas, no de ellas" [6].

I. DESARROLLO

La experiencia fue desarrollada en el año 2016, en UTN- INSPT, con un grupo de estudiantes pertenecientes a la carrera de Informática Aplicada, en el espacio curricular de Análisis Matemático I.

Al momento de iniciar la actividad, los estudiantes ya habían trabajado con software matemático en la realización de cálculos y gráficos de funciones, y también manejaban el concepto de función derivada, sus reglas de derivación y los procesos de determinación de las rectas tangentes a una curva.

Sobre la base de las competencias ya adquiridas, se dispuso como metodología la aplicación de saberes en la resolución de problemas mediados por tecnología digital, tal que se puedan efectivizar los procesos de enseñanza y aprendizaje, facilitando la praxis a partir de la simulación, con el propósito de internalizar y descubrir nuevos conceptos.

Se decidió centrar la actividad en el uso del software *GeoGebra*, a raíz de que ofrece representaciones diversas de los objetos a partir de sus posibles perspectivas: vistas gráficas, algebraicas, estadísticas, organización en tablas y planillas, y hojas de datos dinámicamente vinculadas;

- permite trabajar con objetos algebraicos y gráficos en dos y tres dimensiones;
- aborda aspectos de la matemática a través de la experimentación y la manipulación de distintos elementos, facilitando la realización de construcciones para deducir resultados y propiedades a partir de la observación directa;
- es gratuito y de código abierto (GNU GPL);
- está disponible en español, incluyendo un manual de ayuda;
- presenta foros en varios idiomas (entre ellos, el castellano).

A continuación se narra el abordaje didáctico de uno de los problemas trabajados con los estudiantes.

Primera etapa: Planteo del problema

A los estudiantes dispuestos en pequeños grupos, se les propuso la siguiente actividad disparadora: *Están trabajando en una empresa que diseña packaging y les solicitan el diseño de un prototipo que tendrá como materia prima un cartón rectangular de 25 cm x 20 cm, tal que, al ser doblado convenientemente, se pueda construir una caja sin tapa. ¿Qué posibles dimensiones tendría la caja? ¿Cuál será su volumen?*

¿Qué relación se puede establecer con el concepto de derivada tal que se pueda obtener un volumen máximo? ¿Cómo puede simularse la situación con el software GeoGebra?

Frente a los interrogantes planteados y el debate generado, se da comienzo a la etapa de comprensión del problema, identificando datos e incógnitas. Asimismo comienza a cristalizarse la importancia del uso de software como herramienta para simulación de situaciones y vehículo para la construcción de nuevos saberes.

Segunda Etapa: Momento de la resolución

Mediante la estrategia de ensayo y error, cada grupo elabora una posible figura que refleja la solución del problema (Fig. 1), experimentando con *GeoGebra* y proponiendo diferentes dimensiones que puede tener la caja (Tabla I).

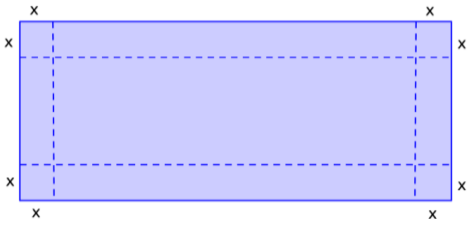


Fig. 1. Diseño del cartón que formará la caja

TABLA I

POSIBLES MEDIDAS DE LADOS PROPUESTAS POR LOS ESTUDIANTES

	Grupo 1	Grupo 2	Grupo 3
Largo:	18,4	20,5	20,6
Ancho:	3,4	5,5	5,6
Altura:	3,3	2,25	2,2

Con las soluciones propuestas, *¿el problema fue resuelto?*

¿Qué relación existe entre los datos? ¿Hay errores en los planteos? Estos interrogantes indujeron a la generación de conjeturas en virtud de los posibles valores que podría atribuirse a cada lado y a la resolución por analogía, concluyendo que es necesario extraer los extremos para que, al realizar los pliegues, se pueda conformar la caja.

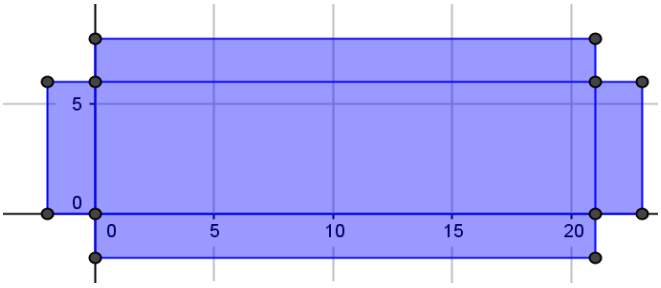


Fig. 2. Diseño de imagen en 2D sin extremos

Siguiendo cada equipo su propio método, al construir la caja en 3D y compartir entre sus pares las diversas representaciones gráficas obtenidas, se observa que tienen volúmenes distintos.

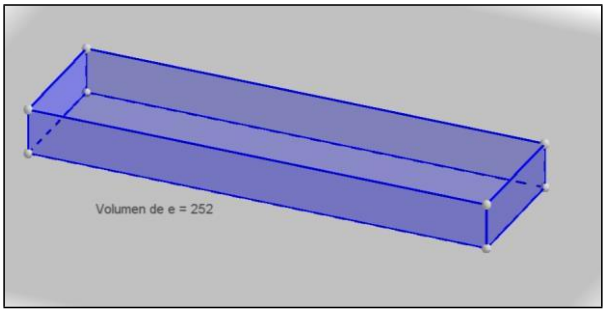


Fig. 3. Diseño final de la caja

De aquí se desprende la necesidad de utilizar un método general adecuado, vinculando las representaciones gráficas con las operaciones algebraicas, es decir, asociar cada propuesta con conceptos matemáticos. Para ello, se recordó la fórmula del volumen: $V = l \times a \times x$, siendo l el largo de la caja, a el ancho y x la altura. Específicamente en el problema planteado, el volumen, en función de la altura de la caja, es $V = (25 - 2x)(20 - 2x)\,x$ cm^3 y, en su forma generalizada, es $V = (l - 2x)(a - 2x)\,x$ cm^3.

A partir de la expresión, se obtuvieron los siguientes volúmenes en cada grupo, ratificando sus diferencias.

TABLA II

POSIBLES VOLÚMENES PROPUESTOS POR LOS ESTUDIANTES

Grupo 1	Grupo 2	Grupo 3
206.448 cm^3	253.68 cm^3	253.79 cm^3

Con esta simulación, *¿puede concluirse que no exista otra caja de mayor volumen? ¿Existe otra forma más precisa para llegar a la respuesta del problema?*

Tercera etapa: el uso de las derivadas

La utilización de derivadas para resolver el problema permite encontrar el resultado óptimo mediante el cálculo de máximos.

Los estudiantes derivaron $V = (25 - 2x)(20 - 2x)\,x$, obteniendo $V' = 12x^2 - 140x + 250$. Al igualar a cero esta expresión, se encuentran los puntos críticos de la misma, es decir, los posibles máximos y mínimos. Aquí surge otro interrogante: *¿Cuál de los valores obtenidos es el correcto?* Esta pregunta lleva a la inferencia de las condiciones que debe tener la altura.

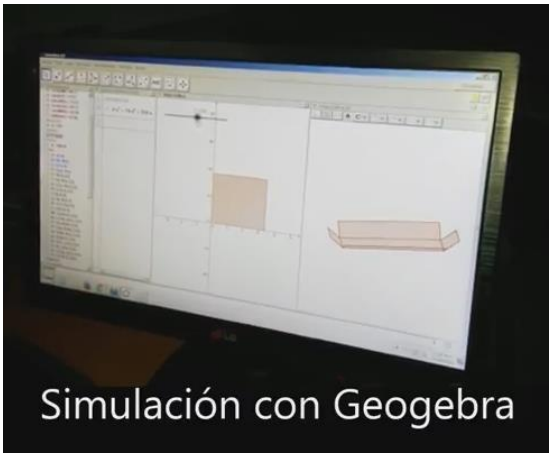
Entonces, a partir de lo trabajado, se concluye que los valores requeridos para resolver el problema son los que se muestran en la Tabla III.

TABLA III

Medidas de la caja para obtener volumen máximo

Extraer cuadrados de	Largo	Ancho
2,2 cm de lado	20,6 cm	15,6 cm

También con *GeoGebra*, los estudiantes pudieron realizar una simulación de cómo se construiría la caja, como puede verse a continuación (Video 1).



Video 1: Simulación de la construcción de la caja [7]

II. CONCLUSIONES

La actividad propuesta permite aplicar conceptos matemáticos en la resolución de problemas de la vida cotidiana, como ser, en este caso, el diseño del prototipo de una caja. Aquí se pone de manifiesto la importancia del uso de software como vehículo para la construcción de saberes a partir de la experimentación y visualización de situaciones de forma interactiva.

Las competencias y habilidades que hemos observado en los estudiantes se relacionan con la capacidad de recordar y revisar conceptos previos, elaborar estrategias de resolución y explorar posibles soluciones que necesitan de la validación. Estas instancias llevaron a aplicar técnicas, generar fórmulas, modelar situaciones mediante representaciones gráficas e incrementar la comunicación, generando argumentaciones y evaluando críticamente los datos matemáticos y procesos elaborados, a fin de efectivizar procesos y constatar saberes.

Consideramos que el abordaje de esta actividad, desde el trabajo en equipo, la mediación tecnológica y la resolución de problemas, visibiliza la importancia que el *hacer matemática* tiene para los técnicos informáticos en función de su actividad profesional futura, que es plausible de ser desarrollada en una amplia variedad de organizaciones, donde deberán dar respuesta a las demandas solicitadas poniendo en práctica la gama de saberes de los disponen.

REFERENCIAS

[1] K. Devlin, «Why universities require computer science students to take math», en *Communications of ACM*, vol. 46, nº 9, pp. 37-39, 2003.

[2] K. Cole, *El universo y la tasa de té. Las matemáticas de la verdad y la belleza*, Ediciones B, p.11, 1999

[3] I. Zalduendo, «Por qué aprender matemática», *La Nación*, 17 de Mayo de 2011.

[4] M. Borba, «O uso de calculadoras gráficas no ensino de funções na sala de aula», en *Anais da Semana de Estudos em Psicologia da Educação Matemática*, pp. 67-72, 1995.

[5] D. Jonassen, *Learning from, learning about, and learning with computing: a rationale for mindtools*, New Jersey: Merrill Prentice- Hall, 1996, pp. 3-22.

[6] D. Jonassen, C. Carr y H. Ping Yueh, «Computers as Mindtools for Engaging Learners in Critical Thinking,», en *TechTrends*, vol. 43, nº 2, pp. 24-32, 1988.

[7] «Simulación de la construcción de la caja». [En línea]. Disponible en:

http://youtu.be/8W_VRRbOgTE

CURRICULUM VITAE DE LAS AUTORAS

María Gabriela Galli es Especialista en Educación y TIC, Licenciada en Gestión Educativa, Técnica Superior en Informática y Profesora en Disciplinas Industriales con especialidad en Matemática. Jefa de laboratorio y docente en UTN-INSPT, y en escuela secundaria de CABA. Está doctorando en Política y Gestión de la Educación Superior.

Gabriela Crespo es Licenciada en Matemática Aplicada y Profesora en Disciplinas Industriales con especialidad en Matemática, docente en la Universidad Nacional de La Matanza y UTN-INSPT. Se desempeña también como docente en nivel secundario de CABA

.

volver al Índice General

Software Libre en Educación

Experiencias sobre la implementación de Software Libre en la materia
Seminario de la Tecnicatura en Informática Aplicada en UTN - INSPT

Matías E. García

Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
matias@profmatiasgarcia.com.ar

***Resumen* — Aunque la presencia del Software Libre en las instituciones educativas de todos los niveles va ganando progresivamente terreno en todo el mundo, la realidad es que su adopción no ha sido tan amplia como ha ocurrido en otros sectores. Las características principales del Software Libre, como los principios de libertad, compartición y cooperación, son determinantes para la producción colectiva del saber que es una de las metas que persigue la educación. Con el afán de promover su utilización, en este artículo se comentarán las experiencias realizadas con los estudiantes desde el año 2011 a la actualidad, las desventajas o inconvenientes que se fueron descubriendo con los diferentes cursos y los logros que se han obtenido.**

Palabras clave: Software Libre; GNU Linux; Educación.

I. INTRODUCCIÓN

El *Software Libre* es aquel que, una vez obtenido, permite ser usado, copiado, estudiado, modificado y redistribuido libremente. Está disponible gratuitamente desde los repositorios de las diferentes distribuciones de GNU Linux o en Internet, o a un precio razonablemente económico, conservando su carácter de libre [1]. Todo programa informático está compuesto, en parte, por secuencias de instrucciones, denominadas código fuente, que el equipo computacional (PC, *smartphone*, sistema embebido...) puede procesar. Es casi imposible realizar cambios en el programa sin acceso al código fuente ni observar cómo ha sido diseñado o qué realiza realmente: solo puede ejecutarse. Se entiende que el software es libre si garantiza las siguientes cuatro libertades: libertad de ejecutar el programa con cualquier propósito (privado, educativo, público, comercial, etc.); libertad de estudiar y modificar el programa (para lo cual es necesario poder acceder al código fuente); libertad de copiar y distribuir el programa; y libertad de mejorar dicho programa y hacer públicas las mejoras, de forma que se beneficie toda la comunidad [1].

Por otro lado, el *software privativo* hace referencia al que es distribuido bajo una licencia restrictiva, que no garantiza estas cuatro libertades citadas, pudiendo ser pago o gratuito, muchas veces con limitaciones de utilización. La normativa de propiedad intelectual reserva la mayoría de los derechos de modificación, duplicación y redistribución para el titular de los derechos de propiedad intelectual, mientras que, como se ha mencionado, el dispuesto bajo una licencia de Software Libre elimina específicamente la mayoría de estos derechos reservados.

Lo que diferencia al Software Libre del privativo no son cuestiones técnicas ni económicas, sino éticas, sociales y políticas. El uso de software privativo crea una dependencia que a la corta o a la larga genera un problema social.

Es un sometimiento. El Software Libre promueve el desarrollo de la sociedad porque se comparte entre todos. Por otro lado, el privativo es un ataque a la solidaridad social porque no permite compartir. El software privativo exige una fe ciega, porque al no poder ver el código se debe confiar en el programa, que puede tener errores, intencionales o no, y uno no los puede verificar ni corregir [2].

En esta *Sociedad del Conocimiento* [3], el software es el gran intermediario entre la información y la inteligencia humana, como explica Mas i Hernàndez [4]. De la misma manera que nos preocupa la libertad para poder acceder a la información y si existe censura, nos debe preocupar con igual intensidad quién controla este intermediario y qué garantías tenemos de su transparencia y fiabilidad.

Las instituciones educativas cumplen una función fundamental para la sociedad. Esto es un motivo determinante para respetar e impulsar los estándares abiertos: No hacerlo supone favorecer a determinado fabricante y a sus clientes. Además, las instituciones deben garantizar la privacidad de los datos de los particulares. Sin acceso al código fuente, es imposible saber qué hace una aplicación con estos datos y cómo son tratados. El Software Libre ofrece una transparencia total y permite realizar auditorías de seguridad sobre el software.

En cambio, los formatos propietarios son creados por las empresas y estas, generalmente, no los hacen públicos, de manera que solo dichas empresas saben cómo tratarlos. El problema surge cuando el uso de uno de estos formatos propietarios se extiende mucho y acaba convirtiéndose en un estándar de facto. Esto perjudica a la perennidad de los datos y afecta a la libertad del ciudadano para escoger su software. Es indispensable que la utilización y el mantenimiento del software no dependan de la buena voluntad de los fabricantes ni de las condiciones monopolísticas impuestas por estos. Utilizar software propietario deja al usuario absolutamente en las manos de las empresas que los fabrican [5].

Por otra parte, si las instituciones educativas no utilizan estándares y formatos abiertos en el dictado de sus cursos, los estudiantes se pueden ver obligados a comprar productos de software a empresas que se verían beneficiadas de una situación de monopolio. Si un alumno no quiere o no puede pagar este software, no podría realizar las actividades solicitadas por el docente, lo cual constituye una discriminación flagrante. Peor aún, si el alumno consiguiera el software de una forma ilegal o el docente le ofreciera una copia del mismo, se incurriría en un delito. No se puede forzar a los estudiantes a adquirir software de determinada empresa para estudiar o realizar las actividades, existiendo alternativas libres.

Es más, hay que ser consciente de que la utilización de software privativo en el ámbito educativo fuerza a los miembros de la comunidad educativa, sobre todo a los estudiantes, a utilizar el mismo tipo de software en su casa y/o en el ámbito laboral.

En definitiva, como indica Amatriain [6], el aprendizaje se promueve a partir de un acceso libre a la información. Por esta razón, en la mayoría de países existen escuelas y bibliotecas públicas. Hoy en día, el software no solo es información en sí mismo, sino que es el principal canal de acceso a la información de cualquier tipo. Además, el software también es un entorno de aprendizaje, en el que los estudiantes desarrollan las competencias propias de los estudios que realizan.

Además, los valores que una institución educativa debería promover están muy relacionados con los correspondientes al movimiento del Software Libre: libertad de pensamiento y expresión, igualdad de oportunidades, esfuerzo, cooperación y beneficio colectivo. De hecho, la libertad quizás sea el valor más importante relacionado con la educación. La educación sin libertad se convierte en mero adoctrinamiento [6].

Según Richard Stallman [7], los valores morales que sustentan la ética del Software Libre son: solidaridad, colectivismo, honestidad, cooperativismo, responsabilidad social, sensibilidad humana y altruismo. Así, el Software Libre garantiza la soberanía tecnológica de los países, fundamentalmente de países subdesarrollados, favoreciendo el proceso de identidad nacional (idiomas nacionales, culturas autóctonas), a diferencia del software privativo que profundiza y fortalece la transculturación globalizante y lleva a la pérdida de autonomía.

En el Primer Encuentro *Hacia un Movimiento Pedagógico Latinoamericano* [8] realizado en Bogotá en 2011 se declaró que debe revisarse el papel que las nuevas tecnologías de la información y la comunicación desempeñan en el sistema educativo y en la sociedad. También que los nuevos lenguajes y herramientas potencian cambios sociales acelerados, sobre los cuales es preciso reflexionar y actuar. Al mismo tiempo, se afirmó como imprescindible contar con herramientas que permitan compartir, estudiar, manipular y desarrollar en libertad, y la única herramienta que garantiza estas libertades es el Software Libre.

Hace muchos años que empleo parte de mi tiempo en promover el uso de Software Libre, tanto para el uso personal como el empresarial y, sobre todo, en educación. Entre algunas de las actividades que llevo a cabo con este propósito están la redacción de mi *blog*, intervenciones en varias redes sociales, la participación en congresos como disertante y la organización de *Festival Latinoamericano de Instalación de Software Libre* (*FLISoL*) [9] en CABA.

Creo que el Software Libre ayudaría a mejorar la sociedad, a tener una relación superadora con esta nueva realidad digital con la que el hombre convive y es parte de su vida en muchos aspectos, brindaría ética, legal y tecnológicamente los medios para poder acercarse y compartir el conocimiento y abarataría los costos de implementar las nuevas tecnologías de la información y la comunicación en un mundo en continuo desarrollo y avance.

Desde 2011 me desempeño como profesor de la materia Seminario de la Tecnicatura en Informática Aplicada, en el turno noche, y me propuse dictar el contenido de la materia utilizando Software Libre con los siguientes objetivos:

- que los estudiantes tengan su primera aproximación al uso de *GNU Linux* [10] (Fig. 1), si es que aún no la habían tenido por su cuenta o por parte de otro docente y brindarles el soporte y la ayuda necesarios para tener una buena experiencia con este sistema operativo;
- comprendan la importancia de las licencias de software [11], entendiendo la diferencia entre el Software Libre y el privativo y cómo utilizarlas en sus futuros proyectos de desarrollo;
- puedan usar un entorno de aprendizaje libre, que puedan adquirir de forma legal, pudiendo compartir lo aprendido y lo descubierto entre sus pares y con otros;
- que los estudiantes utilicen los conocimientos en el uso y desarrollo con Software Libre en su vida personal y laboral.

II. DESARROLLO

Seminario es una materia del tercer año de la carrera de Técnico en Informática Aplicada de UTN-INSPT. El temario de la misma está dividido en dos partes principales. En la primera, se explica y se realizan actividades en *LISP* [12], un lenguaje funcional, y en la segunda se llevan a cabo actividades en algún lenguaje que los estudiantes no hayan visto en otras materias o por el cual haya algún interés particular, por estar de moda, ser nuevo, entre otros factores.

Cuando en 2011 comencé a dictar las clases de Seminario en el turno noche, los equipos de los laboratorios de informática de UTN-INSPT solo contaban con *MS Windows* como sistema operativo para utilizarlos. Instalados en este, había software privativo y Software Libre. Vale aclarar que los laboratorios son utilizados por docentes de diferentes carreras, con diferentes requerimientos de software para sus clases. Estos requerimientos son solicitados antes de comenzar las clases para que el personal de soporte IT pueda instalarlos.

Al comenzar el ciclo lectivo en Seminario se realiza un cuestionario, oral, a los estudiantes para saber su conocimiento sobre las diferencias entre Software Libre y privativo, licencias y otros conceptos sobre paradigmas y lenguajes de programación que luego son reforzados con un trabajo de investigación que deben entregar. Gratamente se puede confirmar que los resultados de esta encuesta fueron mostrando un aumento del conocimiento de los estudiantes sobre Software Libre, partiendo de un 20% en 2011, hasta alcanzar un admirable 70% en 2016.

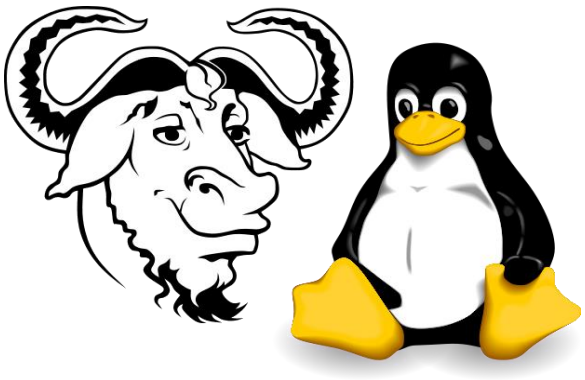


Fig. 1. Logotipos de GNU y Linux

Para reforzar conceptos, los estudiantes realizan un trabajo de investigación, como primer trabajo práctico de la materia, que los lleva a poder comprender mejor qué es el Software Libre, las diferencias con el privativo y la importancia de las licencias de software para el futuro profesional que tendrán. Así toman conciencia del valor real que conlleva la ingeniería de software, el trabajo que deben realizar los programadores y las empresas de desarrollo, la importancia de la obtención legal de programas y de defender la propiedad intelectual y, por el otro lado, las bondades de trabajar en el desarrollo de software comunitariamente o cómo obtener beneficios económicos desarrollando Software Libre, entre otros aspectos.

Durante ese primer año se trabajó con los estudiantes con el sistema operativo privativo ya instalado y se comenzó utilizando los intérpretes de LISP que ya eran usados en la materia en los otros turnos: *CLISP* [13] y *XLISP-PLUS* [14]. El primero es Software Libre y el segundo es un *freeware* de obtención y uso gratuito pero no Software Libre. En el segundo cuatrimestre se realizaron actividades en *JAVA* utilizando como IDE el software *Eclipse* [15] que es *open source*.

Simplemente observando, se podía apreciar que los estudiantes utilizaban el Software Libre y el privativo indistintamente y sin inconvenientes, consultándolos no podían asegurar de qué tipo era cada uno, exceptuando el sistema operativo y los programas de ofimática. De hecho el primer trabajo práctico fue entregado por casi la totalidad de los estudiantes en un formato digital privativo (*.doc*) y sin la licencia de documento, lo que llevó al primer cambio en la forma de entrega de los trabajos para la materia.

Se instauró el formato estándar abierto internacional *PDF* (*Portable Document Format*) para todos los trabajos entregados de forma digital y la licencia de documento *Creative Commons* [16] para los permisos de los mismos. Los estudiantes no tuvieron inconvenientes en generar los documentos en PDF, pero algunos sí tuvieron problemas en seleccionar correctamente la licencia. Una vez solucionado, los trabajos restantes fueron entregados correctamente y los estudiantes pudieron comprender la importancia del uso de licencias de documentos.

Para el segundo año, 2012, se decidió dejar de utilizar el sistema operativo privativo y utilizar *LiveCD*s de *Ubuntu Linux* [17] para arrancar los equipos. Además se descartó el intérprete XLISP-PLUS y se reemplazó por el editor de texto *EMACS* [18] junto al intérprete *SLIME* [19], *plug-in* del editor. En la segunda parte de la materia se decidió enseñar el lenguaje *C#* para el cual, lamentablemente, no se pudo encontrar un *IDE* que funcionara correctamente con los LiveCDs y se optó por utilizar uno privativo en el sistema operativo privativo.

Fig. 2. Laboratorio D en UTN-INSPT con Ubuntu Linux

La encuesta de comienzo lectivo dio mejores resultados, inclusive algunos estudiantes ya habían realizado pruebas en sus equipos particulares con alguna distribución de Linux.

Ya desde la solicitud del trabajo de investigación se dieron las pautas de formato de entrega y licencia a adjuntar, continua actualmente de la misma forma.

La primera impresión que tuvieron los estudiantes para arrancar desde los LiveCDs no fue la mejor, algunos equipos tenían problemas con las lectoras de CD/DVD y otros equipos no estaban configurados para aceptar arranque de esta forma. Una vez solucionados estos primeros inconvenientes se pudo continuar.

La utilización de LiveCDs trajo como beneficio que los estudiantes pudieran utilizar el mismo software que usaban en UTN-INSPT en sus equipos personales de forma legal, incluso algunos se animaron a instalar alguna distribución de GNU Linux en sus equipos al ver que todo funcionaba correctamente.

Trabajando toda la primera parte del ciclo lectivo en Ubuntu Linux, los estudiantes pudieron experimentar las diferencias con el sistema operativo privativo, ventajas y problemas que podían surgir, consultando al docente ante dudas o recomendaciones de utilización. Para la mayoría, estas experiencias fueron enriquecedoras.

Demostrado que se podía dar las clases de la materia con GNU Linux, para el ciclo lectivo 2013 se solicitó la configuración del laboratorio con *Dual-Boot* (Fig. 2), lo que mejoró la performance del uso de los equipos. Se mantuvo el uso de EMACS y SLIME como intérprete y se trabajó con el *Proyecto Mono* [20] para poder llevar a cabo la programación con lenguaje C#.

Algunos estudiantes comenzaron a traer sus propios equipos con alguna distribución GNU Linux instalada para utilizar en las clases, otros solicitaban concejo para poder hacer una correcta instalación en sus casas.

Monodevelop [21], el IDE elegido para desarrollar en C#, fue todo un desafío. Algunos estudiantes ya tenían experiencia en trabajar en otros IDEs privativos y Monodevelop aún es un proyecto, con algunas falencias e incompatibilidades que llevaron, más de una vez, a investigar en blogs en Internet para poder solucionar problemas. Igualmente, se pudieron realizar los trabajos y los estudiantes pudieron experimentar la comunidad que existe entre los desarrolladores de Software Libre para ayudar ante las dudas o inconvenientes.

Este fue el primer año en que algunos estudiantes indicaron que realizarían sus proyectos finales en Software Libre. La materia Seminario solicita que los estudiantes realicen un desarrollo completo de una aplicación como final de la materia. Este puede ser comercial o no, pero debe realizarse utilizando alguna tecnología no vista durante los años de cursado de los estudiantes.

Para el año 2014, los estudiantes ya ingresaban sabiendo que la materia se daba exclusivamente con Software Libre y planteaban sus dudas para instalarlo en sus equipos.

En la edición del FLISoL-CABA de ese año, cinco exalumnos de la carrera en Informática Aplicada de UTN-INSPT presentaron sus desarrollos de Software Libre, y otros tantos, de todos los años de la carrera, fueron al evento y a presenciar diferentes charlas (Fig. 3).

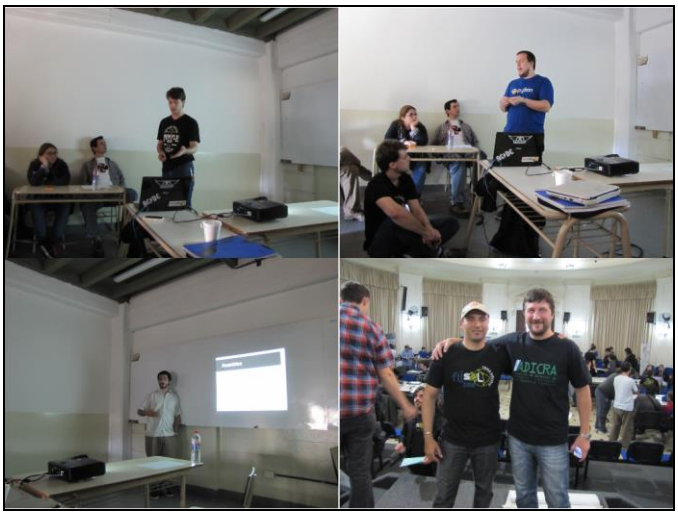


Fig. 3. Exalumnos de UTN-INSPT en FLISoL CABA 2014

Para la segunda parte del año se decidió trabajar con *Android Studio* [22] como IDE para desarrollo de aplicaciones *mobile*. La instalación del mismo generó varios contratiempos, sobre todo por el requerimiento de hardware que tiene y la cantidad de espacio de disco necesario. Los estudiantes lograron desarrollar algunas aplicaciones básicas y en su mayoría informaron que utilizarían esta tecnología en el desarrollo de sus aplicaciones finales.

En 2015 se mantuvo el mismo software, se pulieron los problemas con la instalación de Android Studio, y varios estudiantes concurrieron con sus equipos portátiles, con alguna distribución GNU Linux, para realizar los trabajos.

Ese año UTN-INSPT fue sede de FLISoL-CABA con una gran convocatoria que llenó los pasillos del instituto de entusiastas por el Software Libre. Algunos docentes y estudiantes de la institución dieron charlas y una gran cantidad de público pudo asistir a un evento internacional dedicado a la promoción de la cultura libre (Fig. 4).

Fig. 4. FLISoL CABA 2015 en UTN-INSPT

III. CONCLUSIÓN

La experiencia demuestra que es factible utilizar Software Libre para dictar clases y que es beneficioso en varios aspectos para los estudiantes y futuros Técnicos en Informática Aplicada, como ser:

- poder utilizar el mismo software que usan en UTN-INSPT en sus equipos personales de forma legal;
- contar con la experiencia de utilizar GNU Linux y Software Libre en las clases y la ayuda del docente, para luego hacer sus propias pruebas;
- comprender mejor que es el Software Libre, las diferencias con el privativo y la importancia de las licencias de software para su futuro profesional;
- comprender la importancia del uso de licencias de documentos y los formatos abiertos para sus futuras producciones;
- e interiorizarse sobre la comunidad que existe entre los usuarios y desarrolladores de Software Libre para ayudarse ante las dudas o inconvenientes.

REFERENCIAS

[1] «¿Qué es el software libre? - Proyecto GNU - Free Software Foundation». [En línea]. Disponible en: http://www.gnu.org/philosophy/free-sw.es.html. [Accedido: 01-jul-2016].

[2] «Software libre y educación - Proyecto GNU - Free Software Foundation». [En línea]. Disponible en: http://www.gnu.org/education/education.es.html. [Accedido: 01-jul-2016].

[3] «Sociedad del conocimiento», Wikipedia, la enciclopedia libre. 17-jun-2016.

[4] J. Mas i Hernàndez, «Software Libre: técnicamente viable, económicamente sostenible y socialmente justo». Zero Factory S.L., 2005.

[5] A. M. D. García y R. O. Cuello, «La promoción del uso del software libre por parte de las universidades», Rev. Educ. Distancia, n.o 17, 2007.

[6] X. Amatriain, «El software libre en la educación: guía para su justificación e implementación», presentado en III Jornadas de Software Libre, Escuela Politécnica Superior de Ingeniería, Universitat Politécnica de Catalunya, 2004.

[7] «Richard Stallman», Wikipedia, la enciclopedia libre. 27-may-2016.

[8] «Hacia un movimiento pedagógico latinoamericano», presentado en 1er Encuentro Hacia un movimiento pedagógico latinoamericano, Bogotá, Colombia, 2011.

[9] «Festival Latinoamericano de Instalación de Software Libre», Wikipedia, la enciclopedia libre. 03-may-2016.

[10] «El Sistema Operativo GNU - Proyecto GNU - Free Software Foundation». [En línea]. Disponible en: http://www.gnu.org/gnu/gnu.es.html. [Accedido: 01-jul-2016].

[11] «Licencias - Proyecto GNU - Free Software Foundation». [En línea]. Disponible en: http://www.gnu.org/licenses/licenses.es.html. [Accedido: 01-jul-2016].

[12] «Lisp», Wikipedia, la enciclopedia libre. 09-jun-2016. [13] «CLISP», Wikipedia, la enciclopedia libre. 20-jul-2014. [14] «XLISP-PLUS Page». [En línea]. Disponible en: http://www.almy.us/xlisp.html. [Accedido: 01-jul-2016].

[15] «Eclipse Foundation». [En línea]. Disponible en: http://eclipse.org/org. [Accedido: 01-jul-2016].

[16] «Creative Commons». [En línea]. Disponible en: http://creativecommons.org. [Accedido: 01-jul-2016].

[17] «Ubuntu PC operating system | Ubuntu». [En línea]. Disponible en: http://www.ubuntu.com/desktop. [Accedido: 01-jul-2016].

[18] «Emacs», Wikipedia, la enciclopedia libre. [Accedido: 21-jun-2016].

[19] «SLIME», Wikipedia, the free encyclopedia. [Accedido: 24-abr-2016].

[20] «Mono Project». [En línea]. Disponible en: http://www.mono-project.com. [Accedido: 01-jul-2016].

[21] «MonoDevelop». [En línea]. Disponible en: http://www.monodevelop.com. [Accedido: 01-jul-2016].

[22] «Android Studio». [En línea]. Disponible en: http://developer.android.com/studio/index.html. [Accedido: 01-jul-2016].

CURRICULUM VITAE DEL AUTOR

Matías García es Profesor y Técnico en Informática Aplicada, egresado del Instituto Nacional Superior de Profesorado Técnico de UTN. Actualmente se desempeña como docente de nivel superior especializado en Lenguajes de Programación, Bases de Datos, IA, Software Libre y TIC. Miembro de Ubuntu-ar y CaFeLUG, promotor del uso de Software Libre en educación. Consultor en migración de empresas a plataformas libres.

volver al Índice General

Calculadora

Programación en Java de una calculadora con interfaz gráfica para PC

Mónica Kuhn
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
monicakuhn2013@gmail.com

Resumen — **Este trabajo centra su atención en la enseñanza del manejo de eventos de interfaz gráfica, a través de la escritura de un programa en Java consistente en una calculadora, para estudiantes del segundo año de la carrera de Informática Aplicada de UTN-INSPT. A partir de los conocimientos previos adquiridos por los estudiantes durante el primer año de la carrera, la docente como orientadora de los procesos de enseñanza y de aprendizaje, brinda las herramientas necesarias para la construcción de nuevos saberes, como ser los principios de diseño de las interfaces gráficas de usuario (GUI) y el manejo de los eventos.**

Palabras clave: definición de clases; interfaz gráfica de usuario (GUI); manejo de eventos.

I. INTRODUCCIÓN

Una estrategia pedagógica para aumentar la motivación de los estudiantes en programación, es pedirles que escriban programas que tengan apariencia visual y que puedan aplicar a la vida real [1]. Asimismo, con el propósito de ayudar a los estudiantes a implementar y probar sus algoritmos de manera sencilla e inmediata, es posible utilizar un entorno de desarrollo de aplicaciones (IDE) llamado *NetBeans* [2].

El objetivo, al momento del diseño de la propuesta, fue que los estudiantes lograran:

- crear interfaces gráficas de usuario;
- manejar los eventos generados por la interacción de los usuarios con la GUI durante la corrida del programa;
- establecer la secuencia de ejecución del programa;
- aprender acerca de las clases e interfaces manejadoras de eventos;
- crear y manipular botones, campos de texto y paneles;
- aprender a utilizar los administradores de esquemas (*layout managers*) para ordenar los componentes de la GUI;
- entender el manejo de los eventos del ratón;
- describir un algoritmo que permitiera obtener el valor de una expresión aritmética escrita en notación infija aplicando las prioridades de los operadores aritméticos aprendidos en Matemática.

II. DESARROLLO

En el inicio de la actividad, los estudiantes crean un proyecto básico, con el entorno de desarrollo de aplicaciones. Luego agregan una clase que hereda de `JFrame`. Se les explica el código que deben escribir para agregar a la ventana un administrador de esquemas que permita colocar los componentes de la ventana en determinado orden.

Luego escriben el código para agregar un botón a la ventana. A continuación se compila y ejecuta la aplicación, viendo como resultado la ventana con el botón agregado (Fig. 1). Paso seguido, se escribe el código para que cuando el usuario haga clic en el botón se produzca algún efecto, como, por ejemplo, cambiar el color del fondo de la ventana. Se presenta el concepto de manejador de eventos, que es el código que realiza una tarea en respuesta a un evento. Se crea entonces, para el manejo de eventos, una clase interna privada que implemente la interfaz `ActionListener` y que defina el código del método `actionPerformed`, que es el código que se va a ejecutar cada vez que se haga clic en el botón. Los estudiantes investigan en la Web qué código tienen que agregar en el método para cambiar el color de la ventana e insertan dicho código en el método. Finalmente, escriben las instrucciones para indicarle a cada componente cuál es su manejador de eventos.

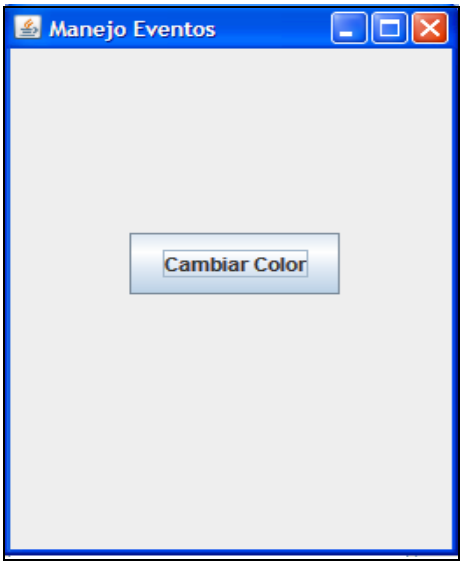


Fig. 1. Proyecto con un botón

Después de compilar y ejecutar la aplicación, se les pide que agreguen a la ventana un campo de texto y el manejador del evento `Enter` para el componente. Una vez que funciona todo bien, se les solicita que diseñen la interfaz de una calculadora con los manejadores de eventos correspondientes, donde las teclas de los números y las operaciones aritméticas de la calculadora estén representadas por botones y, para mostrar la expresión aritmética ingresada y el resultado de la expresión, se utilizarán dos campos de texto.

Fig. 2. Diseño final de la Calculadora

Una vez diseñada la ventana con los botones y sus manejadores de eventos (Fig. 2), los estudiantes agregan al proyecto otra clase que es la encargada, a través de la definición del método `valor`, de calcular el valor de una expresión aritmética en notación infija, que incluye también operadores relacionales y operadores booleanos cuyas prioridades son las mismas que las que se usan en el lenguaje de programación C. La docente explica, en el pizarrón, el algoritmo para calcular el valor de distintas expresiones, utilizando para ello dos pilas para acumular los operadores y los operandos que se van encontrando al recorrer la expresión de izquierda a derecha, y operando de acuerdo a las prioridades de los operadores que haya en la pila de operadores con respecto al operador que aparezca en la expresión. Luego, los estudiantes, con la guía de la profesora, tratan de pasar a código el algoritmo explicado, utilizando para ello conocimientos de programación y de estructuras de datos El objetivo es que sean los estudiantes quienes descubran por sí mismos la implementación de sus algoritmos en la computadora, siendo capaces de realizar el proceso de depuración y probar con distintos datos su aplicación. En las primeras corridas del programa, aparecen errores que, gracias a los mensajes de error que emite el IDE, se van corrigiendo con la orientación de la profesora.

Cabe mencionar que algunos estudiantes fueron más allá de lo solicitado y superaron los requerimientos en relación a las funciones que debían estar presentes en la calculadora, y agregaron funciones adicionales como, por ejemplo, un botón que calcula el factorial de un número.

III. CONCLUSIÓN

Como síntesis, destacamos que nuestros estudiantes desarrollaron y fortalecieron las siguientes competencias:

- Capacidad creativa, evidenciada al crear la interfaz gráfica de usuario;
- Capacidad de relacionar diversas áreas de estudios, evidenciada al aplicar conocimientos de prioridades de los operadores aritméticos definidas en Matemática;
- Capacidad de aplicar conocimientos previos para escribir el algoritmo que evalúa una expresión aritmética en notación infija en formato `String` y el código de las funciones manejadoras de eventos;
- Capacidad para el trabajo en equipo y autónomo durante todo el desarrollo de la aplicación.

Por último, debe mencionarse que el manejo de eventos es indispensable para toda aplicación actual con interfaz gráfica, y dominarlo les proporciona a nuestros estudiantes de la Tecnicatura Superior en Informática Aplicada de UTN-INSPT grandes oportunidades para insertarse laboralmente, además de poder extrapolar lo aprendido a otros ámbitos.

REFERENCIAS

[1] S. Papert y I. Harel, «Situating constructionism», Constructionism,

vol. 36, pp. 1–11, 1991 [2] http://netbeans.org

CURRICULUM VITAE DE LA AUTORA

Mónica Kuhn es Licenciada en Análisis de Sistemas de FIUBA y profesora de Matemática y Cosmografía del INSP "J. V. González". Ejerce como profesora Adjunta en FIUBA y profesora Titular en la Tecnicatura Superior de Informática Aplicada en UTN-INSPT.

---

volver al Índice General

Desarrollo de Aplicaciones Nativas para Dispositivos Móviles con Sistema Operativo Android

Programación de una agenda de contactos

Mónica Kuhn

Universidad Tecnológica Nacional Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
monicakuhn2013@gmail.com

Matías E. García

Universidad Tecnológica Nacional Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
matias@profmatiasgarcia.com.ar

Resumen — **Este trabajo centra su atención en la enseñanza de la programación de *aplicaciones* para dispositivos móviles con el sistema operativo *Android*, a través de la escritura de la aplicación *Agenda de Contactos* para dicha plataforma. La propuesta está dirigida a los estudiantes del tercer año de la carrera de Informática Aplicada de UTN-INSPT. A partir de los conocimientos previos de programación orientada a objetos y manejo de Bases de Datos, adquiridos por los estudiantes durante el segundo año de la carrera, los docentes como orientadores de los procesos de enseñanza y de aprendizaje brindan las herramientas necesarias para la construcción de nuevos saberes, como ser los principios de diseño de interfaces gráficas de usuario (GUI), el manejo de eventos y el acceso a Bases de Datos para la plataforma Android.**

Palabras clave: interfaz gráfica de usuario (GUI); manejo de rventos; bases de datos.

I. INTRODUCCIÓN

Hoy en día, la vida cotidiana de los seres humanos está relacionada inevitablemente con los dispositivos móviles inteligentes. Actualmente, ya existen aplicaciones móviles para facilitar la vida en casi todos los ámbitos. De hecho, han inspirado una nueva clase de emprendedores con impacto real en el empleo, ya que el desarrollador de aplicaciones está entre los perfiles digitales más demandados en los próximos años, lo que genera muchas expectativas de futuro para nuestros estudiantes de Informática de UTN-INSPT.

Para ayudar a los estudiantes a implementar y probar sus algoritmos de manera sencilla e inmediata, se utilizó un entorno de desarrollo de aplicaciones para Android (IDE) llamado *Android Studio* [1].

El objetivo de esta propuesta, diseñada para la plataforma Android, fue que los estudiantes lograran:

- crear interfaces gráficas de usuario;
- manejar los eventos generados por la interacción de los usuarios con la GUI durante la corrida del programa;
- establecer la secuencia de ejecución de la aplicación;
- aprender acerca de las clases e interfaces manejadoras de eventos;
- crear y manipular *widgets* y *layouts* para ordenar los componentes de la GUI;
- acceder y manejar la base de datos del dispositivo móvil.

II. DESARROLLO

En el inicio de la actividad se les mostró a los estudiantes el entorno de desarrollo para aplicaciones Android llamado *Android Studio*, (el IDE oficial para Android) que incluye un emulador de dispositivos móviles que permite probar las aplicaciones sin necesidad de tener un móvil conectado a la computadora.

Luego se les indican los enlaces de descarga del software requerido para desarrollar dichas aplicaciones, así como también los requisitos para su instalación.

A continuación, se crea una aplicación básica "Hello World" (`Empty Activity`) cuya plantilla viene con el IDE, creándose automáticamente una serie de directorios y archivos básicos que aparecen en una ventana lateral y que muestran la estructura de un proyecto para dispositivos Android (Fig. 1).

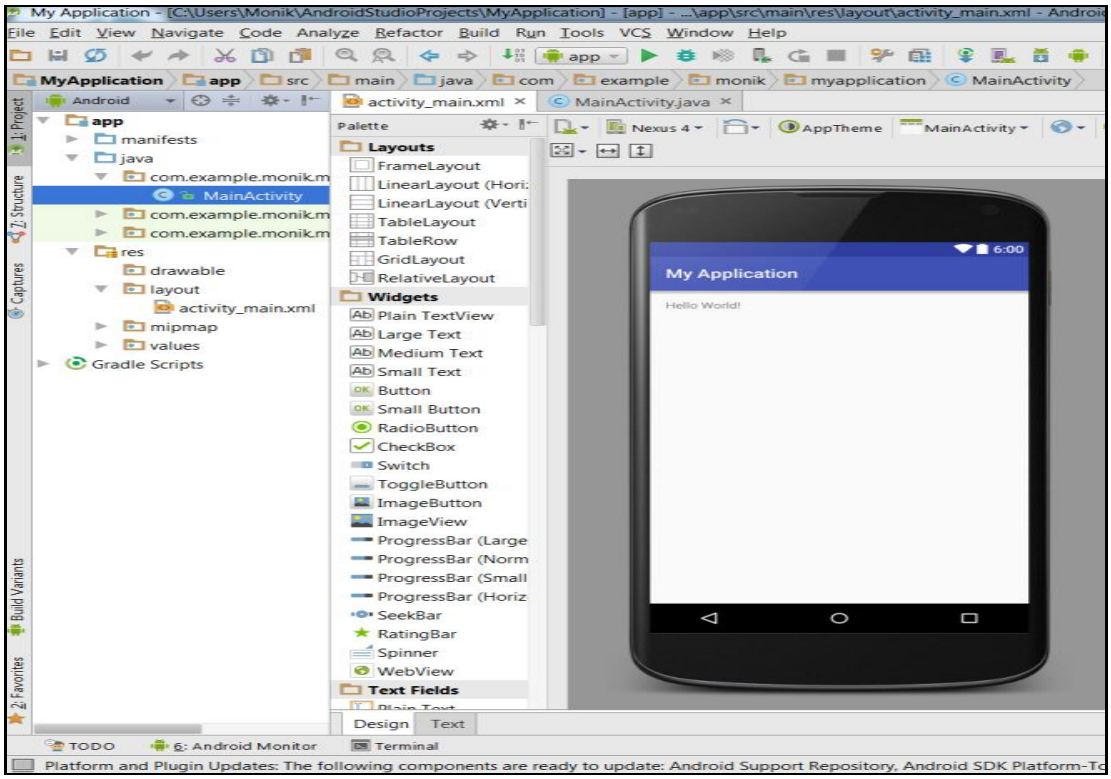


Fig. 1. Aplicación básica "Hello World"

Se les explica a los estudiantes cómo correr la aplicación seleccionando el emulador por defecto y, si no aparece ninguno, cuáles son los pasos para crear uno.

A continuación, se les pide que modifiquen la aplicación básica y le agreguen un nuevo botón, arrastrando este componente desde la paleta que aparece cuando se edita el archivo `activity_main.xml` en modo diseño y soltándolo en ese archivo. Se vuelve a correr la aplicación para ver sus efectos.

Paso seguido se escribe el código para que, cuando el usuario haga clic en el botón, se produzca algún efecto como, por ejemplo, mostrar una ventana contextual que muestre un saludo.

Aunque el lenguaje de programación que se utiliza para aplicaciones Android es Java, la estructura del programa es diferente a la de las aplicaciones de escritorio [2], dado que el código de inicialización de los atributos se escribe dentro del método `onCreate()` de la clase `MainActivity.java`, a continuación de la llamada a `setContentView()`.

Para realizar el diseño, se utiliza la técnica de arrastrar y soltar los elementos de la paleta (*layouts* y *widgets*) a la vista de diseño o bien al árbol de componentes. Para conseguir el diseño deseado, utilizamos un *layout* principal de tipo `LinearLayout`.

Luego se definen las referencias a los diferentes controles como atributos de la clase. Para obtener dichas referencias, se utiliza el método `findViewById()`, indicando el ID de cada control, definido en la clase `R`. Los estudiantes agregan un manejador de eventos del botón y escriben, dentro del método correspondiente al evento *click* del botón, el código para mostrar una ventana *flash* con el saludo.

Corren la aplicación y ven los efectos en el emulador. Luego pasan el archivo compilado a un dispositivo móvil para ver cómo funciona.

Esta primera aplicación sencilla produce en los estudiantes una gran satisfacción y la alegría de constatar que no es difícil desarrollar una aplicación para celulares, y los estimula a seguir aprendiendo más de esta tecnología.

A continuación, borramos el botón anterior y agregamos a la vista (`ActivityMain`) cuatro botones (Fig. 2) para listar, agregar, modificar y borrar contactos de una base de datos que vamos a utilizar para guardar la información de nuestros contactos.

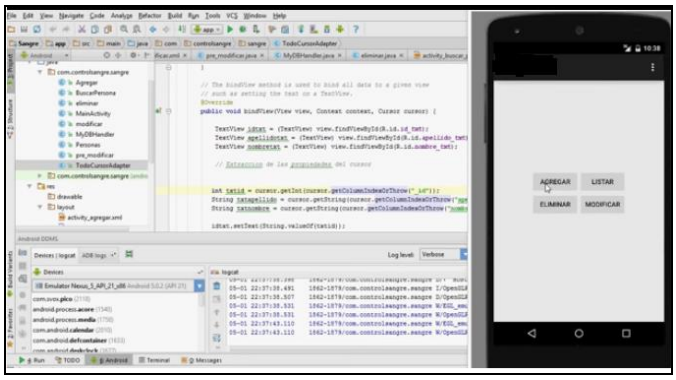


Fig. 2. Menú de 4 opciones

Luego creamos la vista `ActivityListar` (Fig. 3), que se va a mostrar si se toca el botón correspondiente, y le insertamos un `ListView` para mostrar los datos almacenados en la base de datos (Fig. 4).

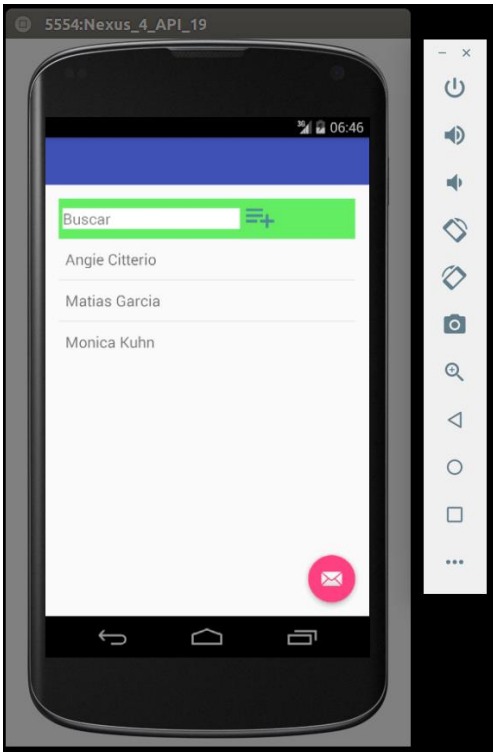


Fig. 3. *Activity* para buscar un Contacto

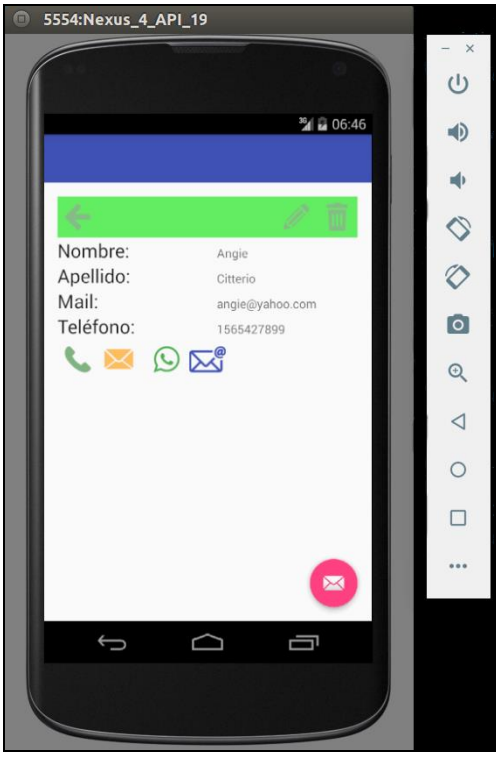


Fig. 4. *Activity* que muestra los datos de un Contacto

Para ello agregamos al proyecto una clase manejadora de datos para *SQLite* [3] con los métodos ABM (altas, bajas y modificaciones) correspondientes.

Paso seguido, agregamos la vista `ActivityAgregar` que servirá para dar de alta a los contactos (Fig. 5). Dicha vista contiene los *widgets* `EditText` y `TextView` para que el usuario, en tiempo de ejecución, complete los datos del nuevo contacto que se agregará a la BD, así como también un botón para efectivizar el alta.

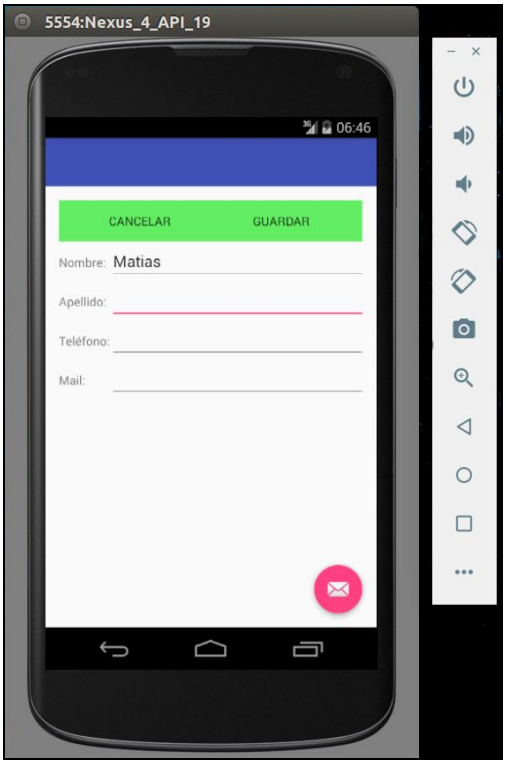


Fig. 5. *Activity* que permite agregar un Contacto nuevo

Una vez terminada esta aplicación básica, se les pide a los estudiantes -como trabajo práctico a realizar en sus casas- una aplicación *Agenda de Contactos* que funcione como la que viene en el celular y que permita hacer llamadas y enviarles a los contactos mensajes por e-mail, SMS y *WhatsApp*.

El objetivo es que sea el estudiante quien descubra por sí mismo el camino para lograr con éxito el desarrollo de la aplicación en la computadora, realizando el proceso de depuración y prueba con distintos datos. En las primeras corridas del programa aparecen errores que, gracias a los mensajes que emite el IDE, pueden ser corregidos con la orientación del profesor.

Vale la pena mencionar que algunos estudiantes superaron las expectativas en cuanto a las funciones que debían estar presentes en la *Agenda de Contactos*, y agregaron otras funciones como, por ejemplo, la posibilidad de agregar una foto para cada usuario de la lista de contactos.

III. CONCLUSIÓN

Como síntesis, destacamos que nuestros estudiantes desarrollaron y fortalecieron las siguientes competencias:

- Capacidad creativa, evidenciada al crear la interfaz gráfica de usuario utilizando distintos *layouts* y *widgets*;
- Capacidad de relacionar diversas áreas de estudios, evidenciada al aplicar conocimientos de manejo de bases de datos relacionales y programación orientada a objetos;
- Capacidad de aplicar conocimientos previos de la programación para escribir el código de las funciones manejadoras de eventos y el código en *SQL* de las consultas a la base de datos *SQLite*;
- Capacidad para el trabajo en equipo y autónomo durante todo el desarrollo de la aplicación.

Cabe destacar que el desarrollo de aplicaciones para dispositivos móviles les permite a nuestros estudiantes de la Tecnicatura Superior en Informática Aplicada de UTN- INSPT aumentar su espectro de conocimientos sobre distintas plataformas, además de extrapolar lo aprendido a otros ámbitos.

REFERENCIAS

[1] http://developer.android.com/studio/index.html.

[2] Deitel, Paul J., Harvey M. Deitel y Abbey Deitel. *Android for programmers: an app-driven approach. Pearson Education, 2012.*

http://www.sqlite.org

CURRICULUM VITAE DE LOS AUTORES

Mónica Kuhn es Licenciada en Análisis de Sistemas de FIUBA y profesora de Matemática y Cosmografía del INSP "J. V. González". Ejerce como profesora Adjunta en FIUBA y profesora Titular en la Tecnicatura Superior de Informática Aplicada en UTN-INSPT.

Matías García es Profesor y Técnico en Informática Aplicada, egresado del Instituto Nacional Superior de Profesorado Técnico de UTN. Actualmente se desempeña como docente de nivel superior especializado en Lenguajes de Programación, Bases de Datos, IA, Software Libre y TIC. Miembro de Ubuntu-ar y CaFeLUG, promotor del uso de Software Libre en educación. Consultor en migración de empresas a plataformas libres.

volver al Índice General

Proyectos de fin de carrera desarrollados por estudiantes en la materia Seminario

SEACAT

Decisiones que te llevan al final de un principio...

Leandro E. Colombo Viña

Egresado de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional -
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
colomboleandro@bitson.com.ar

Resumen — **A través de este relato quiero contarles mi experiencia con el trabajo final de la materia Seminario, correspondiente al tercer año de la carrera de Técnico Superior en Informática Aplicada de UTN-INSPT, y cómo las decisiones que tomé para aquel proyecto adquirieron una importancia suprema en mi carrera profesional actual.**

Palabras clave: decisiones; Python; tecnologías.

I. INTRODUCCIÓN

En el año 2013, durante mi cursada del tercer año de la tecnicatura en Informática Aplicada en UTN-INSPT, realicé el trabajo final para la materia Seminario, siguiendo las consignas de la profesora Lic. Mónica Kuhn y de su ayudante, el Prof. Matías García. El objetivo del trabajo era realizar un desarrollo de software desde el principio, para aplicar los conceptos aprendidos a lo largo de la carrera, utilizando preferentemente tecnologías no vistas en las materias cursadas.

En aquel entonces, me encontraba trabajando en el soporte técnico del Colegio Pío IX, escuela en la que había estudiado y donde trabajaba desde hacía casi 10 años. El nuevo director había decido que era el momento de actualizar algunos procesos administrativos de la escuela y decidió aprovechar la oportunidad que le presentaba mi necesidad de llevar a cabo un trabajo final para poder recibirme. Fue así como, en conjunto, decidimos encarar el desarrollo de un nuevo sistema de inscripción *online* para los ingresantes a la escuela, el cual debía ajustarse a las necesidades propuestas por la secretaría del colegio. Esto posibilitaría que los ingresantes estén dentro del nuevo sistema informático y permitiría así la evolución a un nuevo sistema de registro escolar para todo el colegio.

Habiéndose presentado la necesidad, decidí encararla y me aventuré en el desarrollo.

II. DESARROLLO

Lo primero que tenía que hacer era decidir qué hacer. Tenía casi todo el primer cuatrimestre para elegir el trabajo, así durante el desarrollo del segundo cuatrimestre podría empezar a trabajar en la entrega final. Estaba seguro de que tenía que terminarlo lo antes posible, por dos razones sencillas: tenía que recibirme y el colegio comenzaría el proceso de ingreso luego del receso invernal. Con el director habíamos planificado entonces que, durante el período de ingreso de ese año, se iba a realizar el análisis, para que el sistema estuviera terminado al año siguiente y las inscripciones del ingreso 2015 pudieran realizarse con el sistema nuevo.

Según lo que veníamos trabajando en la materia Programación III con la profesora Lic. Patricia Miadonna, lo primero que debía hacer al comenzar el proyecto era un análisis preliminar que me permitiera tomar la decisión sobre qué herramientas utilizar para el desarrollo del software. También aproveché mucho la experiencia y el conocimiento del profesor Mg. Diego Corsi, quien fue de gran ayuda con respecto a las tecnologías web y protocolos de comunicación.

Inicialmente fue un período de gran incertidumbre, ya que el análisis lo fui realizando mientras pasaban las clases y aprendía, junto a mis compañeros, nuevos aspectos de esta temática. Lo más interesante fue que, entre todos, pudimos aprovechar el tiempo de clase con "ejemplos de la vida misma". Si bien estos ejemplos tenían ciertas falencias porque eran nuestros primeros análisis, nos permitieron discutir y aprender diferentes técnicas y metodologías para llevar a cabo nuestro trabajo.

Luego de tener un poco más en claro cuáles debían ser los alcances de mi proyecto, decidí buscar las herramientas con las que iba a realizar el sistema de inscripción. En primer lugar, tomé la decisión de utilizar herramientas de software libre, no solo porque tuve la posibilidad de elegir libremente, desde la escuela así como también desde la asignatura, sino porque estoy totalmente convencido de que el software libre es mejor. Ya hacía casi 4 años que había dado mis primeros pasos con el software libre y había migrado completamente, tanto en el uso personal como en el profesional (en mi trabajo administraba servidores GNU/Linux). Por otra parte, el desarrollo web está muy ligado al software libre, y el tema de que no viniera atado a costos de licenciamiento era una premisa interesante para la escuela. Finalmente, el hecho de utilizar software libre permitiría que la escuela se quedara con un producto que luego podría extender y utilizar para aprender.

Por otro lado, debía elegir el lenguaje. Al tratarse de un desarrollo para la web, tenía tres candidatos: PHP, Python y Ruby. Para entonces, PHP venía perdiendo terreno. Las nuevas tendencias marcaban que Python y Ruby eran las tecnologías web del futuro. Tenía que decidirme por alguno de ellos. Cuando empecé a investigar la sintaxis de Python, me pareció simple, bella y fácil. Además, encontré que Python tiene una gran comunidad de desarrolladores en la Argentina, por lo que Ruby prácticamente no tuvo chances y fue descartado en mi proyecto.

Python cumplía con algunas características que me resultaron de mucho interés. Es un lenguaje interpretado, el cual no habíamos utilizado en la carrera, debido a que hasta ese momento solo habíamos usado lenguajes compilados. Python es multiplataforma, por lo que me brindaba la posibilidad de ejecutar el sistema en Linux, Windows o Mac. También es multiparadigma y, sobre todo, de muy fácil aprendizaje. Ya estaba decidido, el proyecto lo iba a realizar con Python.

Posteriormente, llegó el momento de elegir el sistema de gestión de bases de datos. También tenía tres candidatos: MySQL, PostgreSQL y MariaDB.

El primero había sido recientemente adquirido por Sun Microsystems, empresa que luego sería comprada por Oracle, y se rumoreaba que esta última tenía intenciones de quitarle el soporte en breve y, por ende, el panorama que se presentaba no era muy alentador para un nuevo proyecto. MariaDB era un *fork* de MySQL, realizado por el mismo que le había vendido este SGBD a Sun, por lo cual no me inspiraba demasiada confianza. PostgreSQL, por el contrario, es un desarrollo 100% de software libre, con más de 20 años en la industria, muy robusto y que soporta grandes proyectos. En conclusión, los datos iban a estar resguardados en un servidor con PostgreSQL.

Finalmente, había escuchado que muchos desarrollos se realizan usando un *framework*. Me parecía interesante aprender a trabajar con estas herramientas ya que, con frecuencia, se las veía en las ofertas laborales, y nosotros no habíamos tenido la posibilidad de utilizarlas durante la carrera. En este caso solo tenía dos candidatos: Django y Web2Py. El primero me resultó muy complejo en comparación con el segundo. Además, el segundo había sido creado con fines educativos, por lo que terminó de convencerme. Consecuentemente, encararía mi desarrollo con Web2Py.

Por último, en una de las tantas búsquedas que realicé en su momento, me di cuenta de que, para el desarrollo de software, era necesario utilizar una herramienta que se conoce como *Sistema de Control de Versiones*. La misma permite ir llevando el registro de los avances y sirve para volver atrás la historia de un proyecto, en caso de “meter la pata”. Durante la carrera ningún profesor nos había contado sobre eso, pero aparentemente era algo que todos los desarrolladores de software utilizaban. Entre todos los sistemas de control de versiones que encontré, dos me llamaron la atención: Mercurial y Git.

Casualmente, ambos fueron creados en el mismo momento para resolver la misma problemática. El primero estaba escrito en Python, por lo que parecía la opción natural. Pero luego de probar ambos, Git me resultó más sencillo y opté por él como mi SCV.

Las herramientas ya estaban seleccionadas. Ahora tenía que aprender a usarlas. El director de la escuela tenía intenciones de que todo saliera de la mejor manera posible y me facilitó la posibilidad de recibir capacitación con el Lic. Mariano Reingart, quien, además de tener su propio emprendimiento, es docente en un terciario y utiliza las herramientas que yo había elegido.

La capacitación recibida me allanó bastante el camino. Mientras aprendía cómo usar las herramientas, dedicaba parte de mi horario laboral a realizar las entrevistas necesarias para obtener los requerimientos del sistema.

El segundo cuatrimestre transcurrió sin sobresaltos, de manera normal y tranquila, aunque mis compañeros y yo teníamos mucha incertidumbre. Realizar un desarrollo desde cero, sin tener del todo claro los objetivos, sin siquiera saber utilizar las herramientas y sin tener la experiencia de haber realizado un análisis fue, muchas veces, como caminar a oscuras. Por suerte, a lo largo de todo el cuatrimestre pudimos compartir las experiencias con los docentes.

Durante la cursada de Programación III con la profesora Lic. Patricia Miadonna, aprovechamos para realizar algunos de los análisis que iban surgiendo de los proyectos. Eso me ayudó a poder centrarme en lo que tenía que hacer para cumplir con lo esperado.

Me dediqué entonces a aprender a utilizar las herramientas mientras iba realizando diferentes análisis de las partes de la aplicación. Al finalizar el segundo cuatrimestre, me dediqué a prepararme para rendir exámenes finales, mientras iba programando muy lentamente el sistema de inscripción. Durante las vacaciones me aboqué en un 100% a terminar esta etapa necesaria para poder recibirme.

A mediados de febrero, me puse en contacto con los docentes para comunicarles que ya tenía algo para mostrarles y, a principios de marzo, ya estaba nuevamente en el instituto para presentar mi proyecto y poder recibirme. Finalmente, así fue. Luego de tres intensos años de cursada y unas vacaciones “codeando”, alcancé mi objetivo.

Preparé una presentación [1] para mostrar la aplicación y la compartí en un repositorio público [2]. Mis docentes estaban tan contentos como yo, y hasta me invitaron a participar del FLISOL [3] que se iba a realizar en UTN-INSPT ese año.

III. CONCLUSIÓN

Ya han pasado un par de años y al mirar hacia atrás puedo reconocer que fue una gran experiencia de aprendizaje. Fue un trabajo muy intenso, con mucha investigación y, por sobre todo, integrador. Aunque el producto obtenido no era de lo mejor y estaba por debajo de los estándares, considero que el resultado fue excelente, teniendo en cuenta que fue una aventura hacia lo desconocido.

Sin lugar a dudas, las capacidades técnicas que desarrollé a lo largo de la carrera se pusieron en juego. Los conceptos que parecían abstractos y, a veces, sin mucho sentido, cobraron valor. Por suerte, había elegido herramientas que estaban sustentadas en el poder comunitario. Sin lugar a dudas, eso me motivó a seguir trabajando. Encontrar que había gente que compartía sus aprendizajes con el afán de que les sirvieran a otros fue un excelente descubrimiento.

Además, adquirí la confianza necesaria para poder encarar nuevos proyectos y nuevos desafíos. Sin duda, fue la mejor experiencia de aprendizaje que tuve en la carrera. No fue para nada sencilla, pero al final resultó muy gratificante. Redondeó una experiencia de tres años muy intensos de aprendizaje y redescubrimiento personal.

Lamentablemente, por cuestiones institucionales, el sistema nunca llegó a ver la luz del día. A pesar de ello, la experiencia vivida me dio la posibilidad de animarme a seguir mi camino. Actualmente, trabajo en una empresa cooperativa que fundamos junto a mis socios y amigos [4]. Estoy convencido de que, si el resultado hubiese sido otro, no habría tenido la confianza necesaria para embarcarme en este proyecto.

En la cooperativa, constantemente me encuentro encarando nuevos proyectos y desarrollos. Muchas de las tareas que corresponden a mis responsabilidades tienen que ver con lo que hice por primera vez en el proyecto para Seminario. Aquel primer ejercicio me dio la posibilidad de pensar qué era lo mejor que podía utilizar para resolver los problemas de otros, buscando la mejor tecnología para trabajar lo más eficientemente posible.

Muchas de las elecciones que tomé al encarar el proyecto final son las que hoy me sirven en mi desarrollo profesional. De hecho, en la empresa, usamos para nuestras aplicaciones como motor de base de datos PostgreSQL. Nuestro lenguaje de *backend* preferido es Python, aunque, si la velocidad o el diseño lo requieren, volvemos por un rato al viejo y querido C. Para versionar nuestro código usamos Git y, siempre que podemos, optamos por usar Software Libre en nuestras aplicaciones. No sigo usando Web2Py, ya que encontramos otro framework que se ajusta mejor a las necesidades que nos presentan nuestros clientes.

Por otro lado, el proyecto me acercó a la comunidad de Python en Argentina. Actualmente, estamos en el proceso de formación de la *Asociación Civil Python Argentina* para fortalecer a la comunidad y llevar adelante acciones que nos sirvan para difundir el lenguaje. Si no me hubiesen sugerido que usara un nuevo lenguaje para mi proyecto, tal vez nunca habría tenido la oportunidad de conocer Python.

Nuestra empresa está federada dentro de FACTTIC (Federación Argentina de Cooperativas de Trabajo de Tecnología, Innovación y Conocimiento) y, en 2016, junto a otras empresas de la federación, trabaja colaborativamente y participa de la organización de la Conferencia de Python Argentina ("PyConAr"), el evento de difusión de Python más grande del país, realizado este año en Bahía Blanca.

Después de terminar la carrera de Técnico Superior, y dada mi vinculación con el sector educativo, decidí hacer también el profesorado en el Instituto, pero esta vez en la modalidad a distancia, ya que tenía que dedicarle tiempo a hacer crecer nuestra empresa. Luego de tres cuatrimestres, obtuve el título de profesor, y hoy me encuentro trabajando como tal en un instituto terciario de la Ciudad Autónoma de Buenos Aires, en el cual dicto cinco materias de la carrera de Técnico Superior en Análisis de Sistemas. Tres de esas materias corresponden al primer año de la carrera y tienen contenidos de programación, que abordo utilizando el lenguaje Python.

Cuando empecé a trabajar en ese instituto terciario, los contenidos de programación se dictaban utilizando el lenguaje C. Sin embargo, luego de un par de años, logré convencer al rector sobre la conveniencia de dictar los mismos contenidos pero con Python. Actualmente, seguimos utilizando este lenguaje, cada vez con mejores resultados, ya que los estudiantes pueden realizar interesantes desarrollos con relativamente pocas líneas de código.

Las decisiones que tomé en mi trabajo final de Seminario fueron determinantes para perfilar mi situación laboral actual. Por eso creo que esas elecciones que me ayudaron a terminar la carrera, son las que hoy me ayudan a continuar con mi desarrollo profesional.

REFERENCIAS

[1] http://prezi.com/eez5inbum9ba/seacat

[2] http://code.google.com/archive/p/seacat

[3] http://www.bitson.com.ar/flisol/seacat

[4] http://www.bitson.com.ar

CURRICULUM VITAE DEL AUTOR

Leandro E. Colombo Viña es Técnico Superior en Informática Aplicada y Profesor en Disciplinas Industriales, especialidad Informática Aplicada, egresado de UTN-INSPT. Socio fundador de *Cooperativa de Trabajo BITSON Ltda.*, allí realiza tareas de *Project Manager*, *Software Architect* y *Backend Developer*, y se desempeña como Tesorero. Profesor de informática de nivel Terciario para el GCBA, dicta clases de Programación, Análisis de Sistemas y Arquitectura de la PC. Es miembro y socio fundador de *Asociación Civil Python Argentina*.

volver al Índice General

NinjaBoom

Desarrollo en Unity Game Engine con C# de una versión en 3D del videojuego Bomberman

Mario Eduardo Galvao

Egresado de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
mario.eduardo.galvao@gmail.com

Resumen — **NinjaBoom es una versión en 3D para plataformas Windows del videojuego Bomberman. Está escrito en el lenguaje de programación C# utilizando el motor para la creación de videojuegos *Unity Game Engine*. Este artículo explora los motivos que propiciaron el desarrollo del proyecto, describe diversos aspectos de los procesos de diseño e implementación y concluye con consideraciones sobre la programación de videojuegos en general.**

Palabras clave: Videojuegos 3D; C#; Unity Game Engine.

I. INTRODUCCIÓN

Hacia finales del año 2013, en ocasión de la cursada de la asignatura anual Seminario, dictada por la Lic. Mónica Kuhn y correspondiente al último año del trayecto formativo, el alumnado se aprestaba a presentar una propuesta como trabajo final.

Si bien se trataba de la instancia final de aprobación de la materia, por envergadura, también oficiaba de trabajo final de la carrera, pues consistía en el desarrollo de un programa informático que reuniera la mayor cantidad de contenidos posibles adquiridos durante la Tecnicatura.

Desde mi perspectiva, el programa debía resolver un problema lo suficientemente atractivo, a fin de cuentas, el trabajo por delante sería arduo. Así nació la idea de desarrollar un videojuego. No tenía la intención de “inventar” uno, sino más bien adaptar un clásico a los tiempos que corren, respetando su mecánica de juego original e incorporando algunas ideas propias.

La elección de un clásico no fue caprichosa. El objetivo era encontrar un modelo ampliamente conocido con una mecánica de juego de extensión moderada, después de todo, se trataría de un proyecto individual que requeriría más que programar.

Desde un punto de vista artístico y luego de haber realizado cursos de modelado y animación 3D, la decisión adoptada fue que, para que el proyecto “cobrara vida”, la mejor opción sería presentarlo en un mundo tridimensional, hoy por hoy un estándar en la industria de los videojuegos. Si bien el apartado artístico no era una exigencia del proyecto pues excedía los conocimientos construidos en la carrera, desde mi óptica era sumamente importante y un factor más a considerar, ya que entre mis objetivos también se situaba que el resultado logrado se asemejara visualmente lo más posible a un juego comercial.

Por otra parte, de los distintos paradigmas de la programación abordados a lo largo de los tres años de formación, la programación orientada a objetos (POO), por sus características, era la que mejor se adecuaba para desarrollar un proyecto de esta naturaleza.

Fue entonces que este trabajo final se transformó en la “excusa perfecta” para experimentar, en primera persona, todo lo que implica pasar por las etapas de especificación y análisis de requisitos, codificación, depuración y documentación para el desarrollo de un videojuego en 3D. Solo restaba considerar dos cuestiones: de qué videojuego crearía mi propia versión y qué herramientas utilizaría para ello.

La primera respuesta no fue fácil. Se barajaron distintas opciones: Pong, Arkanoid, PacMan, Space Invaders, pero finalmente se optó por Bomberman, un videojuego de los años 80 que desde su aparición se ha convertido en una franquicia con más de 60 videojuegos lanzados hasta el día de hoy.

Si elegir la fuente de inspiración no fue tarea fácil, encontrar un entorno de desarrollo que me permitiera crear el videojuego en cuestión resultó un desafío aún mayor: debía ser gratuito, admitir POO y ofrecer la posibilidad de importar modelos creados y animados en 3D.

Finalmente, y luego de investigar distintas alternativas que se ajustaran a estos tres requisitos, la elección recayó en *Unity Game Engine*, un motor multiplataforma para el desarrollo de videojuegos creado por Unity Technologies. El enfoque de la compañía es “democratizar el desarrollo de juegos” y hacer la producción de contenidos interactivos en 2D y 3D lo más accesible posible para todo el mundo.

De a poco, el proyecto tomaba forma. Se implementaría una versión en 3D de Bomberman, manteniendo la dinámica de juego clásica pero con algunos “toques” personales. El nombre del juego sería NinjaBoom (Fig. 1), correría en plataformas Windows y emplearía C# como lenguaje de programación bajo el motor Unity Game Engine.

Fig. 1. Pantalla de inicio de NinjaBoom

II. DESARROLLO

El siguiente paso en la secuencia de trabajo consistió en la planificación de las características concretas que incorporaría el proyecto. Como producto de este análisis, surgieron las especificaciones que se detallan a continuación.

El jugador tendrá el control de un ninja que, en base a colocar bombas para destruir cajas, se abrirá paso a lo largo de cinco niveles con ambientaciones diferentes (Fig. 2). ¿Qué sentido tiene destruir las cajas? Debajo de alguna de ellas, al azar, se esconde una puerta que le permitirá al jugador pasar de un nivel a otro.

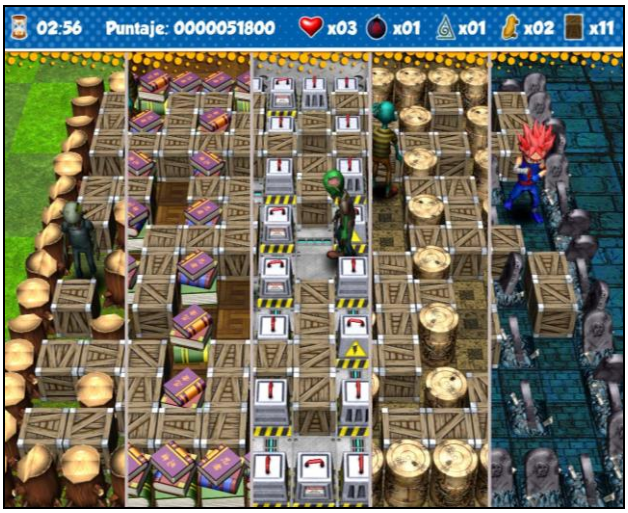


Fig. 2. Las cinco ambientaciones de NinjaBoom

Pero esto no le será nada fácil al jugador. Cada nivel está poblado con cuatro enemigos y ofrece un tiempo límite de tres minutos para descubrir la puerta. Los enemigos en cuestión son zombies con una inteligencia artificial formada por distintos estados a los efectos de exhibir comportamientos variados durante esos tres minutos. Obviamente, el jugador no solo se abrirá paso entre los niveles explotando cajas, también podrá explotar a los enemigos. Además, algunas cajas ocultan distintos ítems que una vez recogidos le permitirán al jugador “evolucionar”: caminar más rápido, conseguir vidas adicionales, colocar más bombas o que las explosiones tengan un mayor alcance.

Para que las cosas resulten un poco más complicadas e interesantes, una vez agotados los tres minutos iniciales, el jugador aún tendrá la oportunidad de pasar de nivel descubriendo la puerta oculta debajo de alguna caja, pero sus enemigos ya no serán los zombies (porque, en caso de quedar alguno, desaparecerá) sino seis fantasmas cuya inteligencia artificial será mucho más agresiva.

Se obtendrán puntos por explotar cajas, eliminar zombies y/o fantasmas y completar desafíos opcionales, los cuales otorgarán una cantidad adicional de puntos. Superado un nivel, se calculará y mostrará el puntaje total en base a los puntos acumulados y desafíos completados. Ya sea que se complete el juego o se pierda la partida, los puntos acumulados competirán contra el puntaje máximo previamente registrado para transformarse en el nuevo récord.

Para la implementación del videojuego, las múltiples elecciones de diseño realizadas implicaron, entre otras, la resolución de cuatro problemas fundamentales: la construcción del tablero de juego, la detección de colisiones, la creación de una caché de objetos y, por último, la inteligencia artificial (IA) de los enemigos.

Tablero de juego

El tablero de juego (Fig. 3) de cada uno de los cinco niveles que componen el proyecto se encuentra lógicamente dividido en un cuadrado de tamaño fijo de 15x15 casilleros.

Fig. 3. Tablero de juego de NinjaBoom

Cada uno de esos casilleros está mapeado a una posición de una matriz. Cada posición de la matriz almacena una referencia a una pila (estructura LIFO) que contiene, a su vez, en cada posición, una referencia a uno de los objetos situados en un mismo casillero del tablero de juego. Por ejemplo, el primer casillero del tablero se corresponde con la primera posición de la matriz. En dicho casillero se podrían ubicar el piso del tablero, un ítem y una caja. Cada uno de estos objetos es referido desde una posición de la pila siendo la cima o tope de la misma accesible desde la primera posición de la matriz. Continuando con el ejemplo, el objeto situado en la cima de la pila sería una caja. Cuando se destruya la caja, se actualizará el tope de la pila con el ítem que se encontraba oculto debajo de ella. Al recogerse el ítem, el nuevo tope será el piso. Y esto ocurrirá con cada una de los casilleros que componen el tablero de juego para cada posición de la matriz de pilas de objetos.

Un controlador de nivel se ocupa de inicializar esta matriz y las pilas correspondientes al crear el tablero de juego y, luego, se encarga de actualizar estas estructuras mientras el nivel se desarrolla. Por ejemplo, se actualiza al colocarse una bomba, al crearse las llamas de una explosión, al destruirse una caja, al recogerse o destruirse un ítem, etc.

No se registra en esta matriz la posición del jugador ni de los enemigos, pero sí es consultada por estos, a través del controlador de nivel, para determinar hacia dónde es posible moverse en el tablero de juego.

Por último, el tablero de juego es generado dinámicamente, es decir, los objetos con que se va a poblar cada uno de los casilleros que componen el tablero son determinados de manera aleatoria al generarse el nivel para lograr así mayor variedad dentro del mismo.

Detección de colisiones

El sistema de detección de colisiones se divide en dos partes: colisiones por posición y colisiones por contacto.

Las colisiones por posición se implementaron para detectar colisiones con los ítems, con la puerta para pasar de nivel y con las llamas que provocan las bombas al explotar. Esto se debe a que dichos elementos aparecen en posiciones específicas del tablero de juego. Estas posiciones *discretas* serán el centro de cada uno de los casilleros. Mediante la matriz descripta en el apartado anterior es posible acceder a cada objeto del tablero y detectar tales colisiones. Los ítems y la puerta para pasar de nivel afectan solo al jugador, mientras que las llamas afectan al jugador y a los enemigos.

Las colisiones por contacto detectan las colisiones del jugador con los enemigos y se basan en la utilización de volúmenes de colisión. Forman parte de *Unity Game Engine* y esencialmente consisten en dotar a cada objeto de un objeto miembro que delimite un volumen apropiado. En este caso, una cápsula para el jugador y para cada enemigo. Cuando tales volúmenes colisionan entre sí, se reporta un evento a cada objeto que participa en la colisión para que actúe. No es posible aquí la detección de colisiones por posición porque el jugador puede colisionar con un enemigo en la transición de un casillero a otro. Por lo tanto, se requiere una detección *continua* de colisiones.

Caché de objetos

La recurrente instanciación y destrucción de objetos que tiene lugar mientras se está jugando un nivel conduce a invertir tiempo de procesamiento en la ejecución del recolector de basura, que tarde o temprano entra en acción. El recolector de basura es un concepto implementado por algunos lenguajes de programación, entre ellos C#, para reciclar la memoria ocupada por objetos destruidos durante la ejecución de un programa.

Dependiendo del hardware del usuario, la ejecución del recolector provocará una súbita caída en la cantidad de cuadros por segundo (referida como FPS, *Frames Per Second*), lo cual afectará la experiencia de juego (coloquialmente, se dice que el juego se "traba" o "congela" durante algunos instantes). Para contrarrestar este efecto y no depender del hardware de quien está jugando, una técnica posible consiste en la implementación de cachés de objetos y el empleo de "pantallas de carga".

En lugar de instanciar objetos y, cada vez que sea necesario, destruirlos (o sea, quitarles todas las referencias), se crea una cantidad de objetos de antemano y se los desactiva hasta que sea necesario utilizarlos. Los objetos ocuparán memoria pero no tiempo del procesador. Todos estos objetos serán accesibles mediante una *caché de objetos* que funcionará como un repositorio y se encargará, entre otras cosas, de activar y desactivar tales objetos. Entonces, cuando se requiera uno nuevo, en lugar de instanciarlo, se utilizará alguno de los disponibles en la caché y, cuando ya no sea necesario, se lo devolverá a la caché para un próximo uso. Si se requiere un objeto de cuyo tipo no hay ninguno disponible, la caché podrá instanciar uno nuevo que se pondrá en uso inmediatamente y se añadirá al repositorio cuando sea devuelto, incrementándose de esta manera el tamaño de la caché. En NinjaBoom no fue necesario implementar una caché de tamaño variable porque se pudo determinar la cantidad máxima de objetos de cada tipo que podrían existir en cualquier instante. Por eso, la estructura empleada como repositorio de la caché es un vector por cada tipo de objeto, y el conjunto de los vectores es manipulado a través de un diccionario.

Ahora bien, para crear todos los objetos de un nivel, incluyendo la caché, se dispone de un nivel especial, distinto de los niveles de juego: la "pantalla de carga" (Fig. 4). Esta se ocupa de instanciar todos los objetos de un nivel de juego en segundo plano mientras brinda un feedback visual del progreso del proceso al usuario. Una vez que se instancian e inicializan todos los objetos del nivel de juego, se lo presenta al jugador.

Fig. 4. Pantalla de carga de NinjaBoom

Cabe destacar que la “pantalla de carga” es un tiempo muerto para el usuario. Por eso, los videojuegos de última generación implementan estrategias mucho más sofisticadas tendientes a “disimular” los tiempos de carga, como si todo el juego fuera un gran nivel. En NinjaBoom, ese tiempo es marginal pero suficiente como para que una vez iniciado un nivel, el jugador no experimente fluctuaciones en la tasa de FPS como consecuencia de la ejecución inoportuna y recurrente del recolector de basura a causa de la instanciación y destrucción continua de objetos (colocación y explosión de bombas, cajas destruidas, las partículas que generan la cajas al ser destruidas, las llamas que se crean al explotar una bomba, etc.).

Inteligencia artificial

A diferencia de lo que ocurre con el jugador, cuyos movimientos son controlados por el usuario, los movimientos de los enemigos son definidos por ellos mismos. A tal fin, cada enemigo determina qué casilleros adyacentes son abordables (no hay bombas, objetos indestructibles ni cajas) y los almacena en un vector. Si un enemigo se encuentra en “modo libre”, elegirá al azar qué casillero abordar, pero si se encuentra en “modo persecución”, ocupará el casillero más próximo al jugador, pues ese será el objetivo a alcanzar con la elección de cada movimiento.

La decisión de qué casillero abordar es tomada por cada enemigo únicamente en el instante en que ocupa el centro de un casillero del tablero de juego. Después de decidir qué casillero abordar, el enemigo no tomará una nueva decisión hasta ocupar el centro del casillero destino. Una vez alcanzado el casillero destino, y antes de decidir su próximo movimiento, el enemigo determinará si ha colisionado con algún objeto empleando el sistema de colisiones por posición: las llamas de una explosión lo eliminarán, y los ítems y la puerta para pasar de nivel no lo afectarán. Evaluadas las colisiones por posición, de no haber sido eliminado, el enemigo decidirá un nuevo movimiento.

Los enemigos nunca retroceden, a menos que hacerlo sea la única opción. Esto implica que la oferta de casilleros abordables por ellos consistirá de tres posibilidades como máximo: continuar con la misma dirección y sentido en que se venía desplazando, girar a la derecha o girar a la izquierda.

Además de los modos libre y persecución, los zombies tienen un tercer estado del que carecen los fantasmas: el "modo confusión". Este estado se presenta cuando a los zombies no les queda otra opción más que retroceder. En este caso, se calculará una pequeña probabilidad para que, si su valor queda comprendido dentro de cierto rango, el zombie se frene en su posición durante algunos segundos. Transcurrido ese tiempo, el zombie iniciará el retroceso. Esos instantes se convierten en una posibilidad para que el jugador "encierre" al zombie con una bomba, garantizando su eliminación una vez que la misma explote.

En cuanto a los fantasmas, además de no tener un modo confusión pueden, marcando otra diferencia con los zombies, flotar por encima de las cajas y las bombas que coloca el jugador, lo que amplía su oferta de casilleros abordables en la elección de cada movimiento. A eso hay que sumarle que son más veloces y que pasan más tiempo en el modo persecución que en el modo libre. Todas estas características configuran una inteligencia artificial más agresiva cuando se la compara con la de los zombies.

III. CONCLUSIÓN

¿Será posible aplicar los conocimientos construidos en el área de programación durante el recorrido del trayecto formativo, al desarrollo de videojuegos? Esta es la pregunta rectora que pretendió responder positivamente el proyecto realizado, así como dar testimonio de que el desarrollo de un videojuego trasciende la programación. Consiste en una actividad interdisciplinaria que en parte se asemeja al proceso de producción de cualquier otro software pero difiere en la gran cantidad de contribuciones creativas requeridas y aportadas por otras disciplinas: niveles, historia, música, efectos de sonido, personajes, animaciones, texturas, etc.

El propio acto de programar es "moldeado" en varios aspectos por los requerimientos específicos que tienen los videojuegos y que no son tenidos en cuenta para otros tipos de software. Una caché de objetos es un buen ejemplo de esto. Otro podrían ser los FPS, que se tornan prácticamente una obsesión a la hora de las optimizaciones.

Frente a la naturaleza interdisciplinaria que revelan los videojuegos, herramientas como Unity demuestran su valía al brindar un entorno de desarrollo que permite integrar animaciones, sonidos, programación, motor de física, detección de colisiones, administración de memoria, motor gráfico, generación de partículas, entre otros. No cabe duda que utilizar un motor para la creación de juegos electrónicos es, en sí mismo, un obstáculo adicional a superar por el programador en términos de aprendizaje, pero bien vale la pena, considerando que su uso facilita y agiliza la creación de videojuegos. No por nada se han transformado en un estándar de la industria.

CURRICULUM VITAE DEL AUTOR

Mario Eduardo Galvao egresó en 2015 como "Técnico Superior en Informática Aplicada" y "Profesor en Disciplinas Industriales" de UTN-INSPT. Desde entonces se desempeña como profesor de las asignaturas "Tecnología de la Información" y "Educación Tecnológica" en el Colegio Jesús María de Buenos Aires.

volver al Índice General

SGA - Sistema de Gestión de Asistencia

Desarrollo de un proyecto final para la asignatura Seminario, desde el diseño hasta su presentación

Martín Jerman
Egresado de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional - Instituto Nacional Superior del Profesorado Técnico Ciudad Autónoma de Buenos Aires, Argentina
martin.jerman@gmail.com

Resumen — **En este artículo se describe cómo fue desarrollado SGA (Sistema de Gestión de Asistencia), un proyecto final para la materia Seminario de la Tecnicatura Superior en Informática Aplicada de UTN-INSPT.**

Palabras clave: gestión; asistencia de alumnos; proyecto final.

I. INTRODUCCIÓN

En muchas carreras hay que desarrollar un trabajo final aplicando los conocimientos adquiridos al cursar sus diferentes materias. La carrera de Técnico Superior en Informática Aplicada de UTN-INSPT no es una excepción, ya que la asignatura Seminario tiene esta exigencia.

En este marco, opté por trabajar con tecnologías que no había utilizado antes, pero aplicando conceptos vistos durante mi formación. Es así que diseñé, programé y probé una aplicación web para la gestión de asistencia de alumnos (SGA). Dicha aplicación fue programada en PHP con jQuery, empleando *responsive design*, el motor de base de datos MySQL y respetando el patrón MVC (Modelo-Vista-Controlador) en todo el sistema.

Aquí convergieron varios conceptos del diseño de sistemas, bases de datos y programación e instalación de servidores; así como también dos ideas que son tendencia en el mercado: la usabilidad web y el diseño adaptativo para dispositivos móviles.

El sistema cuenta con tres perfiles de usuarios (director, secretario, profesor) quienes tienen distintos roles. El director puede ver las estadísticas de asistencia, el secretario arma los cursos con alumnos y les asigna profesores y, finalmente, el profesor toma asistencia en cada curso que le es asignado. Todas las funcionalidades están disponibles a través de la web, lo cual, gracias al diseño adaptativo, permite que, por ejemplo. un profesor tome asistencia desde su teléfono celular u otro dispositivo móvil.

II. DESARROLLO

Para el desarrollo del Sistema de Gestión de Asistencia (SGA) de alumnos, elaboré un cronograma de actividades secuenciadas que fueron el eje conductor de mi proyecto de trabajo final para la materia Seminario.

En todo proyecto, el diseño es una cuestión muy importante. Eso es algo que se enfatiza particularmente en la asignatura Programación III, la cual, al igual que Seminario, corresponde al tercer año de la carrera. Por ello, empecé por dedicar buena parte del tiempo (resultó un 20% del total del desarrollo) a diseñar el sistema: definir los roles de usuarios, el alcance del sistema, las clases necesarias, las tablas de la base de datos, etc. Con el diseño casi definido (porque fue modificado levemente, por cuestiones técnicas, según se fue desarrollando), me di cuenta de que había conocimientos técnicos que me faltaban.

Invertí tiempo (12% del total del proyecto, aproximadamente) en investigar las tecnologías que pensaba utilizar, como ser la programación orientada a objetos en PHP, el uso de jQuery con CSS para el diseño adaptativo, las conexiones a la base de datos y la instalación de servicios/servidores. Por este motivo, realicé pequeños desarrollos independientes para poner a prueba las funciones específicas de cada tecnología. Desarrollé sitios web auxiliares para probar resoluciones de nombres (DNS) y diversos diseños adaptativos, así como también pequeñas aplicaciones tipo *bots* para testear las conexiones a la base de datos, entre otros.

Con el diseño definido y el conocimiento técnico adquirido, inicié el desarrollo del sistema. Empecé por instalar y configurar un servidor web y un DNS en mi red doméstica para poder desarrollar el sistema. Utilicé una máquina virtual para emular un servidor y mi *laptop* para el desarrollo. De esta manera, podía utilizar mi celular en la red local para llevar a cabo pruebas reales, dado que el sitio web estaba disponible dentro de mi red doméstica.

Una vez implementada la infraestructura básica necesaria, seguí trabajando con el modelo de diseño adaptativo de prueba que había desarrollado en la etapa anterior, hasta completarlo con el formato final del proyecto, de manera que la estructura visual quedara definida. Luego implementé el patrón MVC que había estudiado en la materia Programación II, pero adaptado a un entorno web. Desarrollé una pantalla de autenticación (*login*) básica a la cual le añadí la validación contra una base de datos. Una vez probado el funcionamiento, añadí la capa visual del sistema que había implementado anteriormente en HTML. En este punto, solamente tenía una aplicación que aceptaba un *login* del usuario. Parece poca cosa, pero ya tenía la infraestructura implementada, siguiendo el patrón MVC, y con el componente estético (*frontend*) independiente del funcional (*backend*). Solo restaba seguir desarrollando funcionalidades, lo cual resultó muy sencillo porque siempre respeté el patrón MVC. Por cada funcionalidad que desarrollaba, llevaba a cabo una serie de pruebas para asegurarme de que todo funcionase correctamente y, así, poder dar por cerrada la tarea. A veces, a pesar de haber sido probada y dada por terminada, cierta funcionalidad luego tuvo que ser modificada o mejorada para poder funcionar correctamente junto con otras. Pero siempre la metodología era la misma: desarrollar la funcionalidad, probarla, evaluar si el resultado era el correcto (y corregir si fuera necesario) y pasar a la siguiente iteración (o al desarrollo de la siguiente funcionalidad).

En cuanto a la base de datos, el sistema solo requería guardar y leer información, así que MySQL resultó ser más que suficiente, y el desarrollo solamente se restringió al manejo de este motor de bases de datos.

Completar el desarrollo del sistema, lo cual fue llevado a cabo en pequeñas etapas definidas como *funcionalidades* (una idea aprendida en la materia Programación III, en la que diseñamos sistemas de información), me tomó el 40% del tiempo total.

Una vez terminado el desarrollo, probé el sistema en su totalidad. Para ello, armé *sets* de pruebas paso a paso y los ejecuté para ver si el resultado era el que esperaba. Gracias a un buen diseño y a un uso disciplinado de la metodología de desarrollo elegida, encontrar errores y solucionarlos fue realmente una tarea muy sencilla. Esta etapa correspondió apenas al 10% del tiempo total del proyecto.

III. CONCLUSIÓN

Habiendo empezado el proyecto en noviembre de 2013, logré terminarlo para mediados de febrero de 2014. Durante todo ese tiempo, me dediqué casi exclusivamente a este trabajo. Realmente me interesaba aplicar, en un desarrollo hecho por mí, muchas de las ideas y conceptos que había visto durante la carrera. Muchos dirán que es un desarrollo simple, y eso es cierto: la aplicación es muy sencilla. Pero para un estudiante que nunca había programado más allá de ejercicios puramente académicos, este desarrollo con tanta carga teórica y práctica representó todo un desafío desde todos los puntos de vista: (auto)capacitación, diseño, desarrollo y presentación (en una exposición oral ante las autoridades).

Además, yo quería que el producto terminado fuera útil para alguien. Considero que en el ámbito educativo hay mucho por hacer. Muchas veces, la gestión de asistencia aún se hace en papel. Esta necesidad fue la que me motivó a desarrollar un sistema que realmente podría ayudar a las instituciones a ahorrar tiempo.

CURRICULUM VITAE DEL AUTOR

Martín Jerman es Técnico Superior en Informática Aplicada y Profesor en Disciplinas Industriales, especialidad Informática Aplicada, egresado de UTN-INSPT. Previamente fue estudiante de Ingeniería en Sistemas en UTN-FRBA. Es políglota (habla español, inglés, esloveno y portugués) y se desempeña en el área de soporte técnico en empresas multinacionales.

---

volver al Índice General

Memotest

Programando el clásico juego de encontrar coincidencias

Ricardo Rodolfo Leithner
Egresado de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
rrleithner@yahoo.com.ar / Twitter: @RicardoLeithner

Resumen — **Descripción del proyecto "Memotest" y su proceso de realización.**

Palabras clave: HTML5; Juego; Informática educativa.

I. INTRODUCCIÓN

El proyecto formó parte del trabajo final de la materia Seminario de la Tecnicatura en Informática Aplicada del Instituto Nacional Superior del Profesorado Técnico-UTN. Los docentes a cargo de la cátedra eran, en ese momento, la Lic. Mónica Kuhn y el Prof. Matías García. Si bien la cursada de la materia tuvo lugar en 2008, el proyecto vio la luz recién en 2014.

Durante las tareas desarrolladas para el proyecto se investigó acerca del lenguaje HTML5, sus posibilidades gráficas y de programación, su normativa y su situación de compatibilidad, empleando para ello diversos navegadores (Internet Explorer, Chrome, Mozilla Firefox, Opera, etc).

El objetivo fue diseñar un juego educativo orientado al Nivel Inicial y al Primer Ciclo del Nivel Primario y que tuviese la virtud de ser configurable. Se pretendió así lograr una herramienta educativa que permitiese una gran flexibilidad a la hora de modificar los contenidos trabajados por parte del docente.

El proyecto "Memotest" (Fig. 1) permitió, mediante el desarrollo de un programa de juegos, la investigación del HTML5, un lenguaje destinado a transformarse en el estándar de la web. Su elección se basó en su sencillez y en sus recursos gráficos combinados con la programación en JavaScript. El tipo, la complejidad y las prestaciones del juego fueron recibiendo a lo largo del proyecto ajustes y correcciones en sus especificaciones, en la medida de las posibilidades del lenguaje utilizado.

Se eligió como estructura para el juego educativo la de un clásico "juego de memoria", ya que este tipo de juego permitiría su exitosa aplicación en el contexto escolar elegido (Nivel Inicial y Primer Ciclo del Nivel Primario).

Además, se consideró importante la flexibilidad de este juego, ya que permite independizar totalmente la lógica de su funcionamiento con respecto al contenido de las figuras utilizadas. Esa misma flexibilidad es la que permite una funcionalidad adicional: la personalización de los temas por parte de quien lo descargue para su uso fuera de línea.

II. DESARROLLO

El punto de partida para el proyecto estuvo en los conocimientos elementales y previos acerca de las antiguas versiones de HTML. Fue necesario entonces experimentar y actualizar, utilizando diversas fuentes de consulta [1] [2], el conocimiento sobre las posibilidades que ahora propone el estándar HTML5 y, asimismo, corregir los usos que pasaron a ser desestimados. Por dar un ejemplo, era imprescindible abandonar el viejo maquetado por tablas para adoptar la estructuración por secciones y el correcto empleo de la etiqueta .

Luego de procedió a desarrollar modularmente la estética y las funciones que el diseño del juego requería. Allí se comenzó a poner en juego el equilibrio entre las prestaciones deseadas para el juego y las posibilidades reales dentro del contexto del lenguaje elegido. Más tarde, se procedió a llevar a cabo la integración de los módulos y sus naturales correcciones y ajustes.

Fig. 1. Pantalla de inicio de Memotest

Internamente, se organizaron los archivos como en un sitio web, tanto para el modo en línea como para el local (Fig. 2). El archivo `index.html` sirve como página de bienvenida y menú. En la carpeta `ImgWeb` se guardan las imágenes utilizadas en las páginas, y las imágenes que forman parte del juego se almacenan en las de *Temas* correspondientes. La carpeta `HTMLScript` contiene los archivos correspondientes a la página de descarga, la hoja de estilos CSS, la página que aloja el juego y el código en JavaScript que maneja su lógica.

Fig. 2. Organización de los archivos de Memotest

Finalmente, se sometió al código a todos los ensayos y validaciones propuestos por el W3C [3]. La validación por el W3C (*World Wide Web Consortium*) es una prueba necesaria para asegurar que las páginas de un sitio web no poseen errores de sintaxis y que no utiliza *tags* obsoletos. El sitio pasó las pruebas exitosamente.

Para el sitio se eligió una licencia de software libre, específicamente la GNU Affero General Public License 3. Esta licencia está diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red. Así es que el código de esta página puede ser redistribuido y/o modificarlo bajo los términos de la GNU AGPLv3 [4].

Durante el desarrollo del proyecto se utilizaron las siguientes herramientas:

- Sistema operativo: GNU/Linux Huayra
- Editor HTML: Bluefish (GPL)
- Navegadores: Chromium (BSD) - Iceweasel (GPL) -Opera (Freeware)
- Editores gráficos: InkScape (GPL) - GIMP (GPL)

El diseño

El juego consiste en descubrir los pares de imágenes relacionadas (Fig. 3). En su versión original, hay disponibles cuatro temas: “Números y Letras”, “Opuestos”, “Cantidades” e “Iniciales”. En cada tema, el juego presenta tres niveles de dificultad en los que se incrementa sucesivamente el número de cartas. Se contabilizan el tiempo empleado y los aciertos.

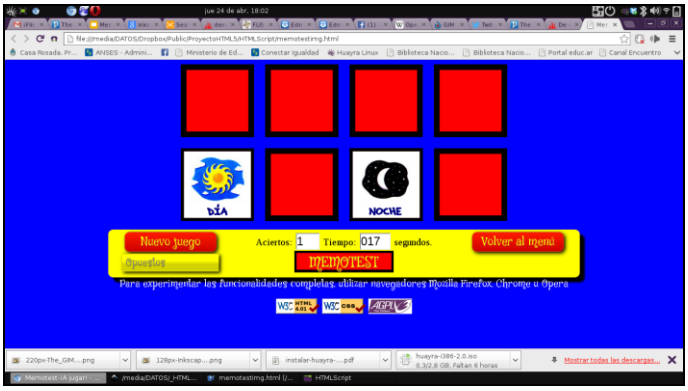


Fig. 3. Una partida de Memotest

Actualmente, el juego está disponible para utilizarse en línea, en el contexto de un blog destinado al software educativo aplicado en clase [5].

Fig. 4. Memotest en línea

También es posible descargar los archivos del juego y descomprimirlos localmente sin requerir instalación.

Además, los temas del juego pueden ser personalizados con diseños propios. En la misma página de descarga se detalla, paso a paso, cómo llevar a cabo este proceso.

El lenguaje utilizado

El HTML5 es un lenguaje de marcado regulado por el Consorcio W3C. Aún en etapa experimental, está destinado a ser el futuro estándar en la web. Incorpora elementos orientados a la web semántica. El elemento `canvas` permite gran versatilidad en el manejo y animación de gráficos e imágenes mediante *scripting*.

A la hora de su elección se consideraron varios aspectos: tiene un gran potencial en su aplicación educativa como introducción a lenguajes de programación, se trata de un excelente punto de partida ya que muchos alumnos tienen algún conocimiento básico sobre HTML, no requiere de complejos entornos de trabajo y, además, es posible encontrar numerosas y sencillas aplicaciones.

III. CONCLUSIÓN

En cumplimiento con sus propósitos, el proyecto permitió experimentar la programación aplicando el lenguaje HTML5. Al mismo tiempo, produjo una interesante herramienta de software educativo que fue muy valorada por docentes de diversas áreas, tanto por su aplicación directa como por su potencial flexibilidad a la hora de trabajar diversos contenidos.

Para obtener más informaciones, se recomienda visitar la presentación en línea del desarrollo del proyecto [6].

REFERENCIAS

[1] http://www.w3schools.com/html/html5_intro.asp

[2] http://w3c.github.io/html

[3] https://www.w3.org

[4] http://www.gnu.org/licenses

[5] http://alsinaprimerciclo.blogspot.com.ar/2015/10/memotest-el-clasico-juego-de-encontrar.html

[6] http://docs.google.com/presentation/d/19UldoBA-cXSJ-8I_CrvmKRVskml-6FCdNWUsVRdD8XA/present

[7] http://lasnetbooksnomuerden.blogspot.com.ar

CURRICULUM VITAE DEL AUTOR

Ricardo Rodolfo Leithner es Técnico Superior en Informática Aplicada y Profesor en Disciplinas Industriales, especialidad Informática Aplicada, egresado de UTN-INSPT. Se desempeña como profesor de Informática en Nivel Inicial, Primario y Medio en escuelas públicas y de gestión privada de la CABA. Promotor del uso de las TIC y el Software Libre en las prácticas educativas, su blog [7] es consultado frecuentemente por docentes del país y del mundo. Es vicepresidente de ADICRA.

volver al Índice General

DW Task

Gestioná tus tareas, simplificá tu vida

Leonardo Miguel Michelli

Egresado de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
michelli.leonardo@gmail.com

Resumen — **Las siguientes líneas describen la aplicación para Android que he desarrollado como trabajo final de la materia Seminario de la Tecnicatura Superior en Informática Aplicada que se dicta en UTN-INSPT. La asignatura fue cursada en 2012 y el trabajo final fue presentado en marzo de 2013.**

Palabras clave: Android; administración de tareas.

I. INTRODUCCIÓN

La aplicación DW Task brinda al usuario la posibilidad de organizar sus actividades más importantes, incorporando alarmas y envíos de *e-mails* de forma automática, a través de una interfaz agradable y llamativa.

Las principales tecnologías que fueron utilizadas en este trabajo son Java, XML y SQLite. Podemos mencionar algunos de los conceptos más importantes aplicados al desarrollar esta aplicación: interfaces gráficas con y sin XML, manipulación de alarmas, bases de datos (SQLite), *threads*, uso de sonidos/música y articulación con Gmail.

Al comenzar este proyecto, buscaba incursionar en algo diferente de todo lo visto durante el transcurso de la carrera de Informática, que fuese una experiencia innovadora, pero también divertida. Finalmente me decidí a hacer una aplicación para Android porque adentrarme en esta plataforma móvil representaría un desafío motivador e interesante.

II. DESARROLLO

Para realizar este trabajo, en diciembre del año 2012 elaboré un cronograma detallado de acciones y tiempos estimados y comencé a desarrollar la aplicación para ser presentada en las fechas de finales de febrero/marzo. A mediados de ese mismo año ya había buscado información y había tratado de asimilar los fundamentos del sistema operativo Android, pero debido a la evolución constante del mismo, algunos de los conceptos cambiaban constantemente. No obstante, esa recopilación de información me sirvió para que pudiera estar preparado y con la capacidad de lograr predecir los tiempos que demandaría cada etapa del proyecto. Esto fue clave para terminar la aplicación en tiempo y forma.

En general, puede decirse que el 50% del éxito de una aplicación depende de que sea visualmente ordenada y atractiva. Es por eso que le dediqué tiempo a esto, buscando imágenes y adaptándolas para la *app* con programas de edición de gráficos.

Al proyecto podríamos separarlo en dos ejes: acceso a bases de datos y entorno gráfico.

Acceso a base de datos

La pantalla inicial solicita un nombre de usuario y una contraseña (Fig. 1), haciendo posible trabajar con múltiples usuarios en, por ejemplo, una misma *tablet*.

El manejo de usuarios y contraseñas se logra guardando dichos datos en una base de datos empleando SQLite, un sistema de gestión de bases de datos que está compuesto por fragmentos de código que conforman lo que llamamos una "biblioteca", la cual nos permite guardar, recuperar y eliminar datos mientras un usuario interactúa con la aplicación.

Una vez ingresados un nombre y una contraseña, la *app* permite crear días. Por ejemplo, en la Fig. 2, se pueden observar las fechas específicas que contendrán las tareas a realizar. Cada solapa representa un día y contiene opciones que afectan a todas las tareas que alberga.

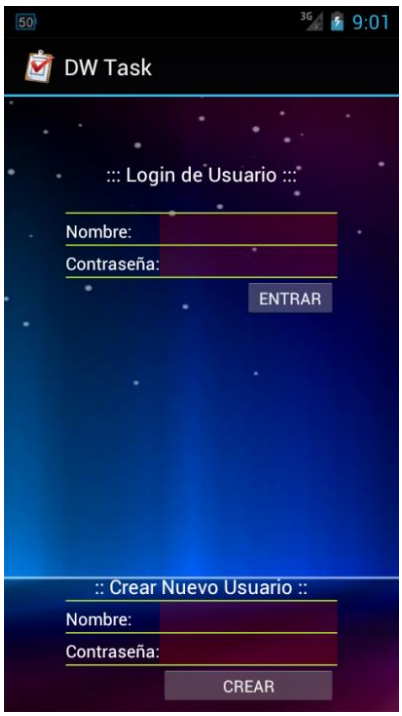


Fig. 1. Pantalla de inicio de DW Task

Cada acción realizada de forma táctil implica acceder a la base de datos (empleando SQLite) para modificar el estado del control tocado. Borrar una tarea o agregar una nueva, modificar el nombre de una tarea o una alarma, etc., se traducen en un acceso de la *app* a la base de datos para asentar esta acción.

Fig. 2. Tareas a realizar en fechas específicas

Entorno gráfico

El diseño es visualmente texturizado y apila las tareas, una tras otra, con la opción de deslizarse a través de ellas, por lo que la cantidad de eventos a mostrar no es un inconveniente. La aplicación permite borrar las tareas de forma individual, parcial o total de manera ágil, gracias a los accesos directos.

La programación de una aplicación para Android se puede realizar en distintos entornos. En su momento, utilicé el software oficial de Google, que era un IDE Eclipse modificado para realizar *apps* para Android. Para acomodar los elementos e insertar componentes, Eclipse ofrece una vista llamada “graphical layout” o su contraparte de diseño “xml”. Las dos fueron de ayuda, pero, a medida que la aplicación evoluciona, la vista de “xml” va tomando mayor prioridad para acomodar mejor los elementos. Actualmente, Google ofrece otro IDE llamado *Android Studio*, con características similares.

Características generales de la *app*

En el proyecto se implementó el patrón Modelo-Vista- Controlador (MVC) [2] estudiado en la materia Programación II. Este patrón es de gran ayuda para organizar el sistema, lo que implica ahorrar tiempo de programación ante una posterior actualización, volver el código más eficiente, disminuir posibles errores inesperados, etc.

La Fig. 3 muestra las opciones que aparecen cuando se intenta crear una nueva tarea. Se aprecia que una tarea puede ser de “inicio/fin” o “esporádica”. Se define una tarea esporádica como aquella que no tiene una alarma asociada y su objetivo es simplemente que, en algún momento del día, el usuario la lleve a cabo (ej: recordar hacer las compras). También se programó que una tarea pudiese tener dos alarmas asociadas. Una que indicara el inicio de la misma y otra su fin. Para terminar, se codificó que una tarea pudiese enviar un *e-mail* de forma automática si se activa dicha opción.

Fig. 3. Creación de una nueva tarea

La Fig. 4 muestra las configuraciones generales de la aplicación, en donde podemos destacar la opción de borrar nuestra cuenta (con todos los datos que incluya), o simplemente cambiar la contraseña. También podemos especificar una dirección de *e-mail* y la contraseña para el acceso al correo (opciones importantes para enviar *e-mails* de forma automática). Y, por último, existe la posibilidad de borrar todas las tareas, en caso de presentarse un escenario en donde borrarlas desde la otra vista demandaría mucho tiempo.

Fig. 4. Configuraciones generales

Para cada cambio efectuado táctilmente por el usuario, es importante que exista un código programado que se encargue de traducir esas acciones en datos que luego Android pueda recuperar (por ejemplo, debió programarse que, luego de que un usuario ingrese una contraseña para la cuenta y presione sobre el botón aplicar, la misma sea guardada en la base de datos, pues luego, cuando se requiere ingresar en la aplicación, el programa debe comparar entre la contraseña ingresada y la guardada, y permitir el acceso solo si ambas son iguales).

Para probar el correcto funcionamiento de la aplicación existen dos caminos: hacerlo desde un celular o emular el sistema operativo Android en una computadora. En mi caso opté por la segunda opción.

También usé programas de terceros [3] para manipular la base de datos y ver reflejados más rápidamente los datos colocados de forma visual.

En internet existe abundante documentación que sirve para resolver los problemas que surgen en el transcurso del desarrollo de un proyecto [4] [5] [6] [7].

Algunas utilidades de la aplicación:

La aplicación puede ser interesante para llevar adelante rutinas de ejercicios físicos. El usuario puede establecer tiempos para correr durante 30 minutos, luego flexiones de brazos durante 15 minutos y, finalmente, 15 minutos de abdominales o intercalar distintos tipos de rutinas como desee.

Otra utilidad podría ser enviar una invitación. Se redacta un *e-mail* para ser enviado en un momento determinado. Esta funcionalidad está recomendada para aquellos que saben que pueden olvidarse de enviar dicha invitación si no la escriben en el momento.

También puede usarse simplemente para recordar llamar a alguien, hacer las compras, recordar pagar un impuesto, etc.

III. CONCLUSIÓN

Actualmente, la programación para Android ocupa un espacio importante dentro de lo que son las ofertas laborales. Desarrollar esta aplicación me abrió un gran abanico de posibilidades.

Fueron de mucha ayuda los conceptos aprendidos en las materias Programación II y Estructura y Base de Datos. En general, todas las asignaturas que componen el currículo de la carrera aportan sus ingredientes para una formación técnica en informática. Si las materias se cursan con dedicación, van generando en el estudiante una suerte de instinto que ayuda a resolver las problemáticas que van surgiendo. Efectivamente, si bien fue importante recopilar información y practicar durante algunos días, otra tarea distinta fue enfrentar problemas reales y resolverlos sin trucos y de forma eficiente. No sabía mucho de los problemas que podían ocurrir antes de que aparecieran. Fui aprendiendo a medida que desarrollaba la aplicación, y fueron dos meses intensos respirando Android.

Antes de comenzar a realizar el proyecto, noté que si quería terminarlo dentro de los tiempos estipulados, iba a requerir de una computadora potente, capaz de virtualizar un sistema operativo dentro de otro sin que se presentaran conflictos de ningún tipo. Por tal motivo, armé una nueva computadora. Sin ella, considero que quizás hubiera tardado hasta tres veces más en terminar el proyecto, debido a que el emulador de Android era pesado y tardaba mucho en iniciarse en mi computadora anterior.

A medida que escribía el código de la aplicación, iba probando su correcta operación. Se podría enumerar una serie de obstáculos que tuve que enfrentar al desarrollar mi proyecto, pero de eso se trata la resolución de problemas, tomarlos como un desafío, proponer soluciones y tratar de resolverlos.

La aplicación es mejorable. Siempre lo será, pues no existe una aplicación perfecta y porque el público es variado, tiene gustos diversos, y las aplicaciones siempre pueden interactuar con otras herramientas. De hecho, quisiera aclarar que la *app* DW Task, en un principio, estaba pensada para el público estudiantil. Inicialmente, tenía como objetivo gestionar las actividades de un estudiante, mostrar sus fechas de exámenes, establecer recordatorios y conectarse con Google Calendar, calcular promedios y dar una ayuda orientativa sobre las materias en donde se necesitaba más atención y refuerzo. Definitivamente, lo que tenía en mente era muy atractivo, pero también era demasiado extenso como para poder terminarlo a tiempo.

REFERENCIAS

[1] http://www.sqlite.org

[2] http://es.wikipedia.org/wiki/Modelo-vista-controlador

[3] http://www.sqliteexpert.com

[4] http://developer.android.com/guide/index.html

[5] http://stackoverflow.com

[6] http://cloudrail.com/best-resources-to-learn-android-programming [7] http://androideity.com

CURRICULUM VITAE DEL AUTOR

Leonardo Miguel Michelli es Técnico Superior en Informática Aplicada y Profesor en Disciplinas Industriales, especialidad Informática Aplicada, egresado de UTN-INSPT. Se desempeña como ayudante en la asignatura Programación II en UTN-INSPT y como Técnico en Electrónica en una empresa de servicios industriales.

---

volver al Índice General

Primer proyecto
desarrollado por una egresada
al iniciar su ejercicio profesional

Sistema BIS

Sistema de control de *downtimes* en líneas de producción de una fábrica

Mariana Victoria Risé
Egresada de la Carrera Técnico Superior en Informática Aplicada
Universidad Tecnológica Nacional
Instituto Nacional Superior del Profesorado Técnico
Ciudad Autónoma de Buenos Aires, Argentina
marianarise@hotmail.com

Resumen — **Este es un proyecto que tuve que hacer para mi primer trabajo en sistemas. El sistema controla y registra en tiempo real las paradas de las líneas de producción, que implican una merma en la productividad.**

Palabras clave: Líneas de producción; control de downtimes.

I. INTRODUCCIÓN

Este proyecto surge en la empresa donde tuve mi primera experiencia laboral en sistemas, durante el año 2014, una vez terminada la Tecnicatura Superior en Informática Aplicada en UTN-INSPT. El cliente era una de las dos empresas más importantes ensambladoras de celulares, tablets y notebooks, situada Río Grande, Provincia de Tierra del Fuego.

La gran problemática que afrontaba la empresa -y que le generaba consecuencias económicas graves- era que, por diversos motivos, las líneas de producción no cumplían con el promedio estipulado de entrega de productos, por lo cual no se alcanzaba la productividad esperada.

Los involucrados (supervisores y trabajadores del área de IT) se culpaban entre sí y, en ocasiones, incluso a los mismos trabajadores de la línea.

Las líneas de producción se detienen por paradas programadas o no programadas. Por ejemplo, las causas de una detención pueden ser problemas técnicos repentinos, falta de material, paradas por almuerzo, etc.

Si estas paradas no se controlan mediante un sistema, donde los involucrados puedan dejar registro en tiempo real de cuándo comienzan los *downtimes*, cuándo intervienen los técnicos o áreas responsables de dar solución al problema, y cuándo queda solucionado, suceden estas irregularidades y no hay cómo demostrar qué sucede.

La primera etapa del proyecto consistió en el relevamiento de los factores que motivaban las fallas. Para ello, un grupo de compañeros de trabajo, con el rol de “Funcionales”, viajaron a Río Grande a definir cuál era el problema que existía, y comenzaron a documentar lo que serían los requerimientos del sistema.

En esta etapa tuvimos mucha interacción con el cliente, tanto con los ejecutivos como con los empleados, llevándose a cabo reuniones, dejando registro de lo actuado y relevando documentación. De esta manera se delimitaron el alcance y el objetivo del sistema a desarrollar.

Luego se estimó, en horas, el tiempo que demandaría el desarrollo del sistema. Nos juntamos el equipo de desarrolladores junto al líder técnico y ponderamos las tareas, para definir entregas intermedias y cuando estaría estipulada la entrega final.

El desarrollo del sistema se llevó a cabo desde cero, ya que no existía un sistema previo que se pudiera mejorar, porque toda la información hasta ese momento se manejaba en papel. El objetivo de BIS era informatizar eventos que sucedían, y era importantísimo que fueran registrados con fidelidad.

Luego del desarrollo del sistema, en la etapa de implementación, viajamos varios miembros del equipo a ponerlo en marcha, participamos en la instalación de televisores con las pantallas de las líneas de producción, semáforos por línea, *tablets* para los supervisores, entre otras tareas. También realizamos reuniones donde capacitamos al personal y relevamos algunos cambios menores.

II. DESARROLLO

Para el desarrollo de BIS se utilizó la metodología ágil *Scrum*. Nos sirvió para organizar por etapas (*sprints*) entregas intermedias. Las *dailies* nos sirveron para estar sincronizados como equipo, en cuanto al avance de tareas, y para poder destrabar cualquier problema que tuviéramos.

Utilizamos *Trello* para organizarnos con las tareas. Teníamos un *backlog* con todas las tareas del sistema, y de allí íbamos tomando las mismas, que tenían cuatro estados: *to do* que eran aquellas que entraban en el *sprint*, *doing*, *to test* y *done*.

Cada tarea tenía sus horas estimadas y, una vez asignada la tarea, es decir cuando alguien la tomaba y la pasaba a *doing*, esas horas debían actualizarse al día siguiente, mostrando así el avance. Cualquier tarea que se pasara de las horas estimadas, significaba un desvío y debía ser absorbida por el equipo ajustando las horas de otra tarea.

En cuanto a la parte técnica, el sistema debe, al iniciar un *downtime*, actualizar dos pantallas que reportan los estados de las líneas. Es decir, dos páginas web deben monitorear el estado de las líneas.

La opción más obvia sería enviar consultas constantes y repetidas a la BD, pero esto generaría mucha carga en la red, en el servidor de base de datos y demoras de refresco. Para evitar esto, existen diversas tecnologías de transporte que proporcionan una mejor solución. El inconveniente es que no todas las plataformas -ni todos los navegadores- las soportan.

SignalR [1] es un framework de Microsoft que nos soluciona este inconveniente, integrando varias tecnologías en unas sola API, dejando que el mismo sistema evalúe la plataforma y el navegador y, entonces, ejecute la tecnología más conveniente.

Las tecnologías que utiliza, en orden de prioridad, son:

- *WebSockets*: es la ideal, pero requiere IE8 o posterior. *SignalR* intentará utilizar esta tecnología y, si no lo logra, pasará a la siguiente.
- *ServerSentEvents* (*Eventsource*): es soportada por todos los navegadores, menos IE.
- *ForeverFrame*: funciona solo para IE.
- *LongPooling*: basado en AJAX, es soportada por la mayoría de los navegadores y plataformas. Es la menos recomendada pero la más compatible.

Usando *SignalR*, logramos que el cliente y el servidor mantengan un canal de comunicación constante. Para pasarse mensajes, el cliente utiliza JavaScript y el servidor una clase de tipo HUB que está al mismo nivel de los controladores de la capa de negocios.

MVC

A nivel de arquitectura, estamos usando esta separación en tres capas, separadas por su responsabilidad, conocida como MVC o bien Modelo-Vista-Controlador. Una contendrá las vistas "UI", otra la lógica de negocios "Business" y, por último, la capa que interactúa con la base de datos "Entidad".

Las capas se comunican entre sí por medio de referencias y en el orden en que están, es decir, la capa "UI" referencia a la de negocios "Business" y se comunica con esta. A su vez, la capa de negocios "Business" referencia a la base de datos "Entidad" y se comunica con esta. La capa "UI" no se comunica directamente con la base de datos "Entidad".

Este software es robusto, con un ciclo de vida adecuado, ya que se potencian la facilidad de mantenimiento, la reutilización del código y la separación de conceptos.

ActiveX

El control del semáforo desde la página web es uno de los requerimientos centrales de la aplicación. Para ello, debemos lograr que la página se vincule con el hardware específico que está conectado por un puerto USB, es decir, que lo reconozca y pueda enviarle directivas de encendido y apagado. Desde un *browser* no es posible acceder directamente a la PC, por lo tanto debemos incluir un software que sí lo permitirá.

La aplicación ActiveX es un programa de escritorio que permite acceder a la PC del cliente, que puede ser ejecutada desde el explorador. Tiene una limitación, ya que solamente puede ejecutarse desde IE.

Los principales métodos para controlar el semáforo son:

- `Open`: inicia un nuevo hilo y enciende el semáforo llamando al método `Encender();`
- `Encender`: Accede al puerto y enciende el semáforo. Le envía continuamente una señal al dispositivo hasta que se ejecute la función `Apagar();`
- `Apagar`: corta el envío de señal con lo cual el semáforo se apaga.

III. CONCLUSIÓN

El proyecto me representó un gran desafío desde muchos aspectos.

En cuanto a lo técnico, lógicamente, ya que con poca experiencia fui afrontando tareas que me aportaron muchísimos conocimientos.

Desarrollé las competencias necesarias para trabajar en equipo, aportar ideas, debatirlas, trabajar con el código de otro y brindar un código lo más legible, prolijo y estandarizado posible.

Aprendí acerca de los roles que existen en un grupo de trabajo: *funcionales*, *desarrolladores*, *líder del proyecto*, *líder técnico*, *QA* o *testing*, *clientes*. Cada persona, desde su rol, garantiza distintos aspectos necesarios para llevar adelante el proyecto, y muchas veces una misma persona tiene más de un rol.

Con el producto entregado, el cliente estuvo conforme, y lo mismo vale para mí. El sistema cumplía con los requerimientos y también estaba cubierto algo que es muy importante: la escalabilidad del sistema. Al hacer la implementación, surgieron algunos cambios o nuevos requerimientos, que pudieron afrontarse con facilidad ya que la arquitectura era desacoplada y modular. Entonces, también aprendí que hay que pensar siempre en que el sistema puede crecer, y soportar ese crecimiento.

REFERENCIAS

[1] http://www.asp.net/signalr

CURRICULUM VITAE DE LA AUTORA

Mariana Victoria Risé es Técnica Superior en Informática Aplicada. Después de egresar de UTN-INSPT, trabajó como analista desarrolladora .NET y Sharepoint, funcional. Actualmente es analista desarrolladora PHP, funcional, en el Ministerio de Justicia de la Nación.

volver al Índice General