Enseñando programación funcional en la Universidad

Franco Bulgarelli (flbulgarelli@gmail.com) [1 2]
Lucas Spigariol (lspigariol@gmail.com) [2 3 4]
Nahuel Palumbo (nahuel.palumbo@gmail.com) [2 4]

[1] Universidad Nacional de Quilmes
[2] Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
[3] Universidad Tecnológica Nacional - Facultad Regional Delta
[4] Universidad Nacional de San Martín

Resumen. Entre las tendencias actuales de desarrollo de software, una de las que se destaca es la que se podría categorizar como el redescubrimiento del estilo funcional de programación. Redescubirmiento, porque ciertamente el paradigma de programación funcional como tal ya tiene varias décadas, pero lo que para no pocos parecía una pieza de museo en los úttimos años ha recuperado vigencia. Estilo, porque se trata de un conjunto de conceptos que se ven no solo en lenguajes de programacion estricctamnte funcionales, sino en herramientas hibridas o incloso en lenguajes de otros paradigmas que incluyen ciertas caracteristicas funcionales. El riesgo es desaprovechar la potencia que las funciones tienen como abstracciones fundamentales en el paradigma y limitarlas a la concepción que hay de ellas desde la pesrpectiva imperativa. El desafio es utilizar al máximo las posibilidades del paradigma, lo que requiere de un especial esfuerzo en la formación de profesionales, que valore la especificidad de este modo de programar y encuentre los medios para hacerlo apropiadamento. En el presente trabajo se desarrolla una propuesta pedagógica en este sentido que incluye un itinerario formativo y un software educativo que lo acompaña y sostiene.

1 Introducción

En el ámbito de la programación se tiende a utilizar las computadoras como herramienta privilegiada en la dinámica de enseñanza, no sólo porque sea más practico para construir los programas, porque se trate de la forma habitual de realizar la tarea en el ejercicio profesional o porque en la actualidad son un recurso mucho más accesible que en otras épocas, sino especialmente porque didácticamente ofrecen notorias ventajas respecto al tradicional método del lápiz y papel.

El feedback que arroja la máquina durante la construcción de software es un elemento fundamental para retroalimentar el mismo proceso de desarrollo, tanto en la

medida en que avisa de errores o advierte de situaciones potencialmente problemáticas como también cuando el programa logra funcionar pero no obtiene los resultados esperados en su ejecución. Asumiendo el carácter iterativo e incremental del desarrollo de software, aun en la pequeña escala de programas realizados en situaciones educativas, la posibilidad de ir construyendo, probando y poniendo a punto progresivamente los diferentes componentes es impensable sin contar con la computadora.

Las objeciones a incluir tempranamente un determinado entorno de desarrollo profesional para realizar ejercicios de programación, especialmente en las materias iniciales del área, dan cuenta de las dificultades adicionales que esto puede traer aparejado: A los conceptos y herramientas propias de la programación en sí se suman otras situaciones, como el hecho de enfrentarse a los errores que arroja la computadora al intentar interpretar, compilar o ejecutar el código, la multiplicidad de herramientas y opciones que ofrecen generalmente los entornos integrados de desarrollo, la necesidad de construir casos de prueba para verificar la correctitud de la solución, e incluso la dificultad de comprensión de un idioma extranjero.

Sin desconocer estas dificultades, pero a la vez sin recorrer el camino tradicional de comenzar programando lejos de las computadoras y recién cuando ya se tienen comprendidos los principales conceptos pasar a ponerlos en práctica en la máquina, en el presente trabajo se analizan otras alternativas de software que permitan aprovechar al máximo el feedback que puede dar la máquina a la vez de minimizar los obstáculos que ésta pueda representar, de manera que la práctica sobre la computadora sea la forma de comprender los conceptos de la programación funcional. En particular, se aborda la utilización en el ambiente universitario de software creado específicamente con fines educativos.

2 Claves de análisis de software educativo

Como fundamento del análisis se presentan algunas afirmaciones basadas en una relectura de las corrientes pedagógicas clásicas desde el contexto actual.

2.1 Lectura crítica de las oportunidades actuales del contexto profesional

Resulta pertinente una lectura permanente del contexto profesional en el cual se desarrolla la actividad para así realizar una mirada crítica de la propuesta educativa universitaria, incluyendo el curriculum prescripto, la realidad de los estudiantes y la propia práctica docente. En este sentido, lejos de asumir una concepción mercantilista de la formación universitaria que la reduzca meramente a satisfacer las demandas presentes de la actividad privada, contar con una mirada amplia y crítica de la actualidad profesional, del proyecto de desarrollo productivo del país y sobre todo de las tendencias que se vislumbran, es sumamente importante para poder orientar acerca de las capacidades que se pretenden desarrollar en los estudiantes.

2.2 El uso de tecnología educativa en el contexto de una propuesta pedagógica en el que adquieren sentido

La fuerza del cambio tecnológico plantea en la actualidad la necesidad de reflexionar sobre su sentido y utilización. Juan Esteva afirma que la tecnología es un instrumento del hombre para lograr objetivos y que como tal, "su impacto no depende exclusivamente de su naturaleza y contenidos, sino también, y muy especialmente, de la forma de utilizarla. Esto significa, en otras palabras, que la tecnología no se impone por su propio peso, sino que necesita una intencionalidad en su utilización y su fuerza introductora" [1]. En el mismo sentido, Nicholas Burbules sostiene que "la tecnología no es solo la cosa, sino la cosa y las pautas de uso con las que se aplica, lo más nuevo, tal vez, no sea la tecnología, la cosa en sí, sino todos los otros cambios que la acompañan" [2]. Las herramientas no sólo ayudan a alcanzar los objetivos existentes, también pueden crear propósitos nuevos, nuevas metas, que jamás habían sido consideradas antes que dichas herramientas los tornasen posibles. De esta manera la tecnología provee funcionalidades que facilitan la distribución de la información y permiten el diseño de nuevos métodos de aprendizaje.

2.3 La necesaria y conflictiva articulación entre teoría y práctica

Si bien es un aspecto recuperado por la mayoría de las corrientes pedagógicas, la pedagogía crítica da especial importancia al ida y vuelta vital entre práctica y teoría, enriqueciéndose mutuamente. Paulo Freire es muy claro al respecto y son numerosas sus referencias a la práctica –a la que con un lenguaje más político, y profundizando su significación denomina "praxis"- y su relación siempre conflictiva con la teoría. Critica la dicotomía entre teoría y práctica y recomienda evitar todo tipo de propuesta "que menospreciase la teoría, negándole toda importancia y enfatizando exclusivamente la práctica como la única valedera o bien negase la práctica atendiendo exclusivamente a la teoría" [3]. También en los ambientes universitarios, generalmente tendientes a focalizar en conceptos teóricos, tiene sentido articular teoría y práctica. En esta dirección, un trabajo realizado por la Universidad Tecnológica Nacional propone "una nueva relación donde la práctica deja de ser la mera aplicación de la teoría para convertirse en fuente del conocimiento teórico" [4].

2.4 El docente como facilitador y mediador

De los múltiples rasgos o caracterizaciones que se le pueden dar al docente en ejercicio de su rol, sin excluir otros que también tienen su importancia, se destaca la de ser facilitador. Aún en una aparente ausencia, como en el caso del "Maestro Ignorante" que relata Jacques Ranciere [5], el docente es el que brinda los instrumentos de mediación para que el estudiante construya su conocimiento. La experiencia relatada -donde la utilización de un libro dado por el docente a sus estudiantes les permite aprender más allá del ambiente escolar- puede ayudar a redescubrir la importancia y variedad de mediaciones. En particular, ejemplifica la ruptura de la correlación lineal entre la explicación del profesor y la comprensión del alumno y el descubrimiento de

nuevas mediaciones, que en el caso mencionado es un libro y en el presente trabajo es un software.

2.5 La gradualidad y complejidad del proceso de enseñanza-aprendizaje

Cuando desde su mirada constructivista del aprendizaje Vigotsky plantea la "Zona de Desarrollo Próximo" como aquellos conceptos o habilidades que el estudiante no conoce pero que están a su alcance de ser aprendidos con las mediaciones adecuadas [6], está reconociendo el rol central del docente y a la vez está abriendo el juego a una variedad de recursos que facilitan el aprendizaje. En la actualidad, la posibilidad de utilizar herramientas tecnológicas con un fin de mediación dentro del rol docente va en clara sintonía con estas ideas. Asumiendo que el aprendizaje no es una acumulación de saberes sino un proceso de construcción, los errores no son vistos como un fracaso sino como una oportunidad de aprendizaje. Partiendo de ellos, entendiendo sus causas y consecuencias, es posible el reacomodamiento que supone el aprendizaje. La misma noción de gradualidad da sentido a la importancia del error como parte del proceso, en tanto que es un indicador de los pasos progresivos que se van dando y permite focalizar en los obstáculos parciales como estrategia de superación. De todas maneras, no es en sí el error mismo el que favorece el aprendizaje, sino la posibilidad de descubrirlo, asumirlo e interpretarlo dentro de una propuesta pedagógica. En otras palabras: si hay algo peligroso de los errores es no detectarlos.

2.6 La importancia de la motivación y del protagonismo del estudiante

Se reconoce el lugar central del estudiante en el proceso educativo y se entiende la educación como un acto de creación de cultura, en un proceso de recreación, producción y no solamente en una transmisión pasiva de conocimientos culturales. Un aporte fundamental sobre la dinámica de clase y el aprovechamiento de la presencia del docente en relación a otras mediaciones, es la que desarrollan Bergmann y Sams. Desde su experiencia docente personal, luego formalizada como "Flipped Classroom", afirman: "El momento en que los alumnos necesitan que esté físicamente presente con ellos es cuando se atascan en un tema y necesitan mi ayuda personal. No me necesitan en el aula con ellos para darles contenidos; los contenidos lo pueden recibir por su cuenta" [7].

2.7 La potencialidad de crear las propias herramientas tecnológicas pedagógicas, entendiendo la Universidad como espacio de innovación

En un país que necesita de un sólido desarrollo tecnológico propio se reconoce el rol activo de la Universidad como innovadora en materia científico-tecnológica, y como ámbito no sólo donde se replica el conocimiento lanzando a la sociedad tandas de profesionales, sino donde también se investiga y se desarrolla tecnología. Con la mirada puesta sobre la importancia creciente del uso de software como recursos pedagógico, se identifica a la carreras de Sistema o Informática como el lugar donde la

necesidad y la capacidad confluyen y es posible desarrollar el software que se utilice con fines educativos en la misma carrera.

3 Desafíos de la programación Funcional

Dentro del ámbito profesional del desarrollo de software, en los últimos años, la programación funcional fue ganando espacio y despertando la atención tanto de desarrolladores novatos como de expertos en otras formas de programar.

Su elemento conceptual principal -la función- no es en sí mismo un elemento extraño para quien ya ha trabajado en programación, pero el mayor riesgo que generalmente se corre es que se quiera utilizar las funciones de manera imperativa, ignorando el potencial y la versatilidad que tiene el concepto de función desde la forma en que la concibe el paradigma funcional.

En algunos aspectos, el programador experimentado cuenta con conocimientos transversales a los diferentes paradigmas que sin duda son un punto a favor, como por ejemplo el manejo de parámetros, de variables o de tipos de datos. Sin embargo, hay muchos conceptos que parecen obvios de la programación, como la asignación destructiva o la iteración-, que en el paradigma funcional no tienen sentido o tienen una utilidad diferente y que le resultan más difíciles de comprender y poner en práctica adecuadamente que a un aprendiz que está dando sus primeros pasos en el mundo de sistemas.

El paradigma funcional, desde sus orígenes, se diferencia de la programación imperativa por múltiples razones, pero especialmente por su enfoque declarativo: En vez de plantear la solución a un problema como una secuencia ordenada de instrucciones, donde la principal preocupación es determinar detalladamente el flujo de ejecución que va a realizar la máquina y las modificaciones sucesivas de las celdas de memoria donde se guardan los datos, se la construye como un conjunto de definiciones de funciones que articuladamente describen la solución, basándose en que la maquina funciona como un motor de reducción con la suficiente inteligencia para evaluarlas adecuadamente.

Tanto es así, que en numerosas clasificaciones teóricas sobre los lenguajes de programación, sobre todo las de las primeras, se presentaba una primera y contundente diferenciación entre paradigmas de programación declarativos y no declarativos, dejando a la programación funcional -y también la programación lógica- entre los primeros y la programación procedural o estructurada en la segunda, también denominada imperativa. Con el surgimiento de la programación orientada a objetos el esquema no se modificó sustancialmente, sino que el nuevo paradigma, aun teniendo diferencias notorias con la programación procedural, clásicamente se lo ubicó entre los no declarativos, manteniendo cierta "pureza" declarativa para la programación funcional y lógica.

Es cierto también que en los últimos años esa división fue perdiendo rigurosidad. Esta flexibilización conceptual se puede atribuir a diferentes motivos que tienen que ver con la dinámica de creación, circulación, aceptación y utilización de nuevos lenguajes y herramientas de programación por parte de los profesionales de sistemas -donde las razones tecnológicas entran en conflicto con los intereses comerciales-, entre los que se pueden destacar al menos dos: Un proceso de hibridación, a partir de la aparición y crecimiento de lenguajes que combinan características de diferentes paradigmas y un fenómeno de extrapolación conceptual, donde el reconocimiento de la utilidad de ciertos conceptos y modos de pensar funcionales ha llevado a su aplicación en otros contextos de desarrollo. En particular, actualmente se tiende a utilizar el término de declaratividad desde una concepción más gradual que binaria para analizar transversalmente a los desarrollos hechos en todo tipo de lenguajes de programación, de manera que se pueden encontrar soluciones más declarativas que otras a los mismos problemas.

Otra característica, no por cierto excluyente pero sin dudas típica del paradigma, es la variedad de alternativas que ofrecen sus lenguajes para resolver un mismo problema, dejando a criterio del desarrollador la elección de una u otra en cada situación. De todas maneras, que haya diferentes posibilidades no las hace a todas igualmente apropiadas o correctas. Desde un punto de vista pragmático y cortoplacista, podría considerarse que cualquier alternativa que funcione es correcta, pero desde el punto de vista de un profesional de sistemas, no es suficiente que funcione, sino que el código sea de calidad. Y en este contexto no por calidad se entiende un "lujo" con fines de autosatisfacción del experto que demuestra su saber o que busca el prestigio de superar estándares de evaluación externa, sino la calidad como robustez y versatilidad del código, lo que implica, entre otras bondades, las siguientes:

- Un programa que sea sencillo de leer e interpretar por otros desarrolladores, lo cual en una disciplina donde abunda el trabajo en equipo y a distancia y en un mundo globalizado es cada vez más importante.
- Un programa que sea fácil de modificar, lo que es central en una dinámica de desarrollo de software con un ritmo cada vez más vertiginoso y con requerimientos que cambian permanentemente.
- Un programa que se pueda escalar, donde su organización interna y las abstracciones planteadas permitan incorporar complejidad y crecer en volumen y alcance sin necesidad de comenzar nuevamente de cero.
- Un programa cuyos componentes se puedan reutilizar, en el que pensando no sólo en la finalidad del sistema que se está desarrollando sino en una dinámica de construcción de software con mirada a largo plazo, se puedan definir unidades de software lo suficientemente genéricos y desacoplados para que puedan potencialmente ser utilizados en otros contextos.

Al llevar estas afirmaciones sobre el desarrollo de software en el marco de la programación funcional a un ámbito educativo como la Universidad, al poner en tensión el curriculum prescripto institucionalmente con la experiencia profesional de los docentes, al pasar de una formulación de objetivos a una planificación concreta y al im-

plementar conceptos generales mediante un lenguaje de programación en particular surge un conjunto de cuestiones a resolver.

Un primer asunto es la elección y modo de utilización de un determinado software. Utilizar un lenguaje de programación en su ambiente de desarrollo provee herramientas que facilitan la escritura del código y ayuda a su puesta a punto mediante pruebas, pero sólo en la medida en que el usuario sepa manejar las dichas herramientas e interpretar los errores y advertencias que da la máquina. Los lenguajes de programación que son de uso profesional en la industria del software están pensados para optimizar la productividad del profesional que sabe programar y no para quien está empezando a dar sus primeros pasos. Para empezar, la misma instalación del lenguaje de programación tiene su complejidad que se añade a cuestiones iniciales de configuración de la herramienta. El problema central, se da en el *feedback* que arroja la máquina frente a los errores: muchas veces, el mensaje de error es incomprensible para el estudiante ya que se explica a partir de conceptos de programación que no maneja u otros elementos del lenguaje qué aún no se le explicaron. Un estudiante puede construir una solución y considerar que es correcta sin siquiera probarla o habiéndolo hecho con un juego de datos muy particular que no representa la variedad de situaciones a las que la solución debe dar respuesta adecuadamente. Cabe señalar que, sobre todo en los ejercicios más sencillos, cuando el estudiante se está iniciando en una nueva herramienta, el armado del caso de uso y todo el contexto en el cual la solución puede ser probada es tanto o más complejo que la solución en sí. Por otra parte, hay muchas veces donde el estudiante no logra que su solución funcione y por lo tanto no puede probarla, pero sin embargo, su planteo es correcto y con sólo ajustar cuestiones menores o detalles sintácticos se puede terminar de construir el programa correctamente.

Partiendo que se quiere trabajar sobre la computadora y no limitarse al "lápiz y papel", ¿usar un software profesional o uno educativo? ¿hay otras alternativas? Antes de abordar una posible respuesta, se agregan otras cuestiones pedagógicas que resolver, desde algunas más generales hasta otras específicas de la programación funcional:

Dentro de un esquema de introducción gradual de complejidad, ¿cómo organizar la secuencia didáctica de manera de no dejar lo más importante para último momento?

Desde una concepción de articular teoría y práctica, ¿cómo explicar el sentido y la utilidad de un nuevo concepto?

¿Hasta qué punto introducirse en cuestiones de bajo nivel, acerca de la implementación del lenguaje, su mecanismo de reducción y otras características?

¿Cuánta importancia darle dentro de la distribución de tiempo del programa de la materia y de la valoración en la evaluación al concepto de orden superior, que permite manipular funciones de diferentes maneras, considerándolas como un tipo de dato más?

¿En qué momento incorporar la composición de funciones, como forma típica de articular diferentes funciones?

4. Una propuesta de recorrido educativo, con herramientas de software creadas para tal fin

Como forma de dar respuesta a los problemas expuestos y desde las claves de interpretación presentadas, se aborda una propuesta de enseñanza de la programación funcional en la Universidad con un determinado itinerario teórico-práctico que es viabilizado mediante un software educativo denominado *Mumuki*, desarrollado por docentes universitarios con el fin de facilitar y organizar el proceso de aprendizaje de los estudiantes.

Particularmente, se analiza una plataforma virtual on-line denominada *Mumuki*, desarrollada por un grupo de docentes e investigadores universitarios. Es un software educativo para aprender a programar a partir de explicaciones y resolución de problemas; propone aprender conceptos de programación en un recorrido conformado por guías de ejercicios donde se combina teoría y práctica Esta herramienta se presenta al estudiante como una aplicación web interactiva, en la que se articulan explicaciones y ejemplos con el foco puesto en que cada uno realice su solución y sea probada y corregida instantáneamente, orientando acerca de los aciertos y errores.

Está diseñado para ser utilizado tanto a distancia como en forma presencial. Como recurso complementario al aula y con el ritmo propio que cada estudiante le imprime a su forma de practicar, como en espacios educativos formales de laboratorios o talleres, donde cada alumno -o grupo- trabaja con su máquina, con la presencia orientadora de los docentes.

4.1 Los ejercicios

Frente a las dificultades con la tecnología en los primeros pasos del proceso de aprendizaje de la programación funcional y ratificando la importancia de una práctica constante por parte de los estudiantes articulación con la teoría, surge el desafío de fomentar una mayor ejercitación y realizar una devolución adecuada por parte de los docentes y que redunde en un aprendizaje más significativo.

La plataforma presenta básicamente una gran cantidad de problemas concretos, organizados y articulados entre sí, que el estudiante debe resolver mediante el desarrollo de un programa en el lenguaje *Haskell*, uno de los más emblemáticos de la programación Funcional. Junto con los ejercicios, se presentan orientaciones, ejemplos y conclusiones.

Al elegir un ejercicio, el alumno se encuentra con una explicación donde se enuncia el problema a resolver, con ejemplos y orientaciones iniciales, y se habilita un panel donde puede escribir el código de la solución.

Según como se haya diseñado el problema, puede que disponga de ayuda adicional -que al solicitarla se muestra de inmediato-, que haya alguna parte del problema ya resuelta y deba concentrase en lo faltante u otras variantes. Cuando el estudiante termina de plantear su solución, con sólo presionar un botón la "envía" y la plataforma la ejecuta y evalúa, informando el resultado.

Hay tres situaciones posibles. La primera, es que la solución sea correcta, es decir, que pase todas las pruebas predefinidas. En este caso se presenta una conclusión, donde se puede subrayar algún aspecto de la solución a destacar, se puede reforzar cierto concepto teórico o incluso explicitar un nuevo concepto que surge de la solución efectuada. También es la posibilidad de felicitar al estudiante por el resultado obtenido y alentarlo a que continúe.

Cuando la solución no satisface las pruebas, se indica cuáles son los errores y puede deberse a dos motivos: Que el programa tenga errores básicos que directamente no permiten que se ejecute o que sí logre hacerlo, pero no da los resultados esperados. Ante el primero de los problemas, en muchos casos, la plataforma es lo suficientemente inteligente para darse cuenta de errores típicos y en vez de mostrar el mensaje que arroja un compilador -orientado a programadores experimentados- lo explica en un lenguaje natural orientado a un estudiante, con precisiones y orientaciones propias de la forma de presentar la materia por parte de los docentes. En los casos en que no logra detectar el motivo del error, no hay más alternativa que mostrar directamente el error del compilador. Superado el primer problema, cuando el programa propuesto por el estudiante logra ejecutar, lo que hace *Mumuki* es "correr" una serie de *tests* que cubren el abanico de casos a los que el programa debería dar una repuesta adecuada. Cuando detecta una diferencia informa diciendo cuál era el resultado esperado ante cierto lote de datos de entrada y cuál es el que obtuvo con la solución planteada por el estudiante.

Una tercera situación, que pretende ofrecer una diferencia sustantiva respecto de lo que el estudiante obtiene como *feedback* utilizando el entorno habitual del lenguaje de programación, es cuando la solución realizada funciona, obtiene los resultados esperados ante todos los casos planteados, pero desde un punto de vista conceptual podría mejorarse. Entendiendo que el objetivo es que el estudiante comprenda y sepa aplicar los conceptos y herramientas teóricas de la programación funcional, no basta con que la función "funcione", sino que lo haga utilizando adecuadamente los principios del paradigma. Llevándolo al caso de cualquier ejercicio, si su objetivo era mostrar cómo determinado concepto facilita la resolución de cierto tipo de problema, y el estudiante recurrió a una solución diferente, probablemente más precaria o innecesariamente compleja, y esquivó la utilización de dicho concepto, no puede considerarse correcta la solución. En estos casos, la plataforma reconoce que los resultados son los esperados, advierte sobre cuáles son los conceptos u herramientas no utilizados e invita a que el alumno mejore su solución. Ciertamente, es enorme la cantidad de soluciones diferentes a un problema que se pueden hacer y por más que haya casos típicos, no deja de sorprender que haya estudiantes que logran hacer soluciones alternativas no previstas por los docentes, ya sea por lo buena que resultan o todo lo contrario. Que se puedan reconocer ciertos errores conceptuales y se brinden orientaciones inmediata y automáticamente ayuda, pero de ninguna manera sustituye la mirada atenta del docente, que a su tiempo y personalmente puede hacer una devolución más adecuada. Por otra parte, técnicamente hablando, no es nada sencillo automatizar la detección de errores de este estilo. Más difícil resulta aun cuando se reconoce que la programación, más allá de su fama de disciplina exacta, tiene una componente subjetiva importante y cada docente tiene su propia apreciación respecto de lo que considera una buena solu-

ción: Que tal función ande o no ande no presenta mucha discusión, en cambio, que por ejemplo en su definición sea mejor utilizar composición y aplicación parcial que una expresión lambda, no es tan claro. Frente a estas problemáticas, en algunos ejercicios *Mumuki* asume un criterio prudente de no analizar ciertos conceptos, como por ejemplo la expresividad o la declaratividad de la solución, y en otros lo hace contextualizando y explicitando con mayor nivel de detalle lo que se espera. Continuando con el ejemplo señalado, si el ejercicio se encuentra en una guía que trata sobre composición de funciones, se señala como error conceptual la utilización de una expresión lambda no porque en sí sea incorrecto, sino porque el objetivo del ejercicio es aprender otro concepto.

En todos los casos, se da la posibilidad que el alumno corrija su solución y la vuelva a enviar, tantas veces como sea necesario. También se despliega una consola, donde el mismo alumno puede ir haciendo las propias pruebas sobre las funciones definidas y viendo cómo va funcionando su programa.

4.2 Forma de organización del contenido

Los ejercicios se organizan en guías, en función de los temas que se abordan. Mantienen un hilo conductor y están pensadas para una resolución progresiva. A medida que se avanza en la resolución de los ejercicios se van poniendo en evidencia los elementos conceptuales relacionados que dan sustento a la práctica y a la vez permiten pasar a nuevos ejercicios de mayor complejidad.

El recorrido de guías combina algunas más orientadas al aprendizaje, que presentan temas nuevos mediante ejercicios donde se la especial atención a los conceptos teóricos necesarios para poder resolverlo, con guías llamadas de "desafío" donde con mejor cantidad de explicaciones se prioriza la cantidad y variedad de situaciones sobre temas ya explicados. El esquema básico intercala cada guía de aprendizaje con al menos una de desafíos.

La secuencia presentada y la cantidad de ejercicios por cada tema no es arbitraria, sino que responde a la forma en que el docente ordena y pondera los diferentes temas. De todas maneras, es el estudiante quien decide la forma y la intensidad de uso, pudiendo detenerse más en cierta guía que en otra, adelantarse o volver hacia atrás de la forma que considere necesario. Precisamente, con el fin de permitir recorridos diferentes y por otra parte minimizar la frustración de un ejercicio que no puede ser finalizado, la plataforma no requiere haber terminado correctamente un ejercicio para pasar al siguiente. Esta flexibilidad permite también que diferentes docentes planteen sus propios recorridos conceptuales teniendo como base el mismo conjunto de guías.

4.3 Recorrido funcional

Si bien es una herramienta en constante actualización y permite que cada docente establezca variantes, el recorrido estándar plantado por *Mumuki* en la actualidad, consta de unas 20 guías con un promedio de 15 ejercicios cada una. Sin entrar en de-

masiado detalle que disperse el foco de atención, algunas de las características presentadas son:

Énfasis en pensar en funciones como valores de primera clase. Desde el convencimiento de que el principal aporte que hace el paradigma funcional al campo profesional de la programación es el manejo de funciones como valores, los diferentes temas que abonan esta idea, están destacados en el itinerario y algunos aspectos se presentan ya desde el inicio del recorrido, como para utilizarlos a lo largo de todo el proceso de aprendizaje. Por ejemplo, apenas habiendo presentado un par de funciones con datos simples, que transforma por ejemplo un número en otro mediante alguna operación matemática, ya se plantea la idea de composición de funciones como una abstracción que opera sobre funciones. Más adelante, antes de plantear el manejo de listas y otras estructuras, se presenta la idea de aplicación parcial, enfatizando en cómo obtener una función a partir de otra.

"Usar" antes que "definir". Apelando al criterio de encontrar la utilidad de una herramienta antes que su modo de construcción, no sólo como estrategia motivacional sino para ganar significatividad en el aprendizaje, a lo largo de los ejercicios generalmente se ve primero a las funciones como una caja negra donde a partir de cierta entrada se obtiene cierta salida, postergando la comprensión acerca de su definición interna.

4.4 Lenguaje cercano

Los ejercicios, explicaciones y los mensajes de error, sugerencias y ayudas se encuentran íntegramente en español, con una redacción informal, orientada al diálogo y la pregunta, usando recursos gráficos como *emoticones* e imágenes, y basada en el *voseo*, característico de Argentina. Esto busca establecer una mayor cercanía con la forma de hablar del estudiante actual, a la vez que va introduciendo lenguaje técnico. En los casos en que se debe informar acerca de un error de parte del compilador, se pasa de una mera traducción a una explicación detallada del problema, no orientada a relatar el error evidente, sino al origen conceptual más probable del mismo. Por ejemplo, en lugar de indicar “uso de identificador no declarado”, la plataforma reportará “Estás usando un predicado, pero no lo declaraste antes. Fijate si no te olvidaste de declararlo o si escribiste mal su nombre”.

4.5 Seguimiento de los estudiantes

En forma complementaria, hay una serie de opciones que están pensadas para los docentes. Entre ellas, cabe destacar la herramienta para el seguimiento de los estudiantes de un grupo, *Mumuki Classroom*, con la que el docente puede ver el avance de cada alumno en la resolución de ejercicios propuestos. Permite ver al docente de manera organizada y clara la información sobre el progreso de sus propios estudiantes, llegando al detalle de no solo ver el código correcto de la solución que fue validado por *Mumuki*, sino también todos los intentos intermedios que fue probando, de forma tal que sea un indicador más a la hora de planificar la materia. En esencia, busca dar

respuesta a preguntas sobre el grado de avance del curso -¿qué porcentaje de estudiantes resolvió tal guía? ¿qué ejercicios obtuvieron mayor cantidad de soluciones incorrectas?- así como también sobre seguimiento individual de los estudiantes -¿en qué se está equivocando tal alumna? ¿demuestra una mejora en sus soluciones sucesivas a partir del *feedback* de la plataforma?

4.6 Gestión del contenido

Una herramienta clave que incluye *Mumuki* es la que permite editar contenido, lo que le permite al docente redactar los enunciados de los ejercicios, presentar las ayudas típicas y explicitar las expectativas de aplicación de conceptos en la resolución del problema. Esta característica es fundamental para la utilización de la plataforma en diferentes ámbitos educativos e independizar la tarea de creación de contenido de la que significa el mantenimiento de la plataforma como tal.

4.7 Aporte al desarrollo científico

La plataforma de ejercitación y corrección es en su totalidad código libre y gratuito, con el objetivo de facilitar la colaboración entre docentes de diferentes universidades y entusiastas en general. La libertad de las herramientas y materiales de estudio es fundamental para la democratización del conocimiento y entendiendo en desarrollo de software como una industria, constituye un modesto aporte al desarrollo productivo nacional. Frente al software privativo y a diferenciándose de ciertos enfoques mercantilistas acerca de la educación y la circulación del conocimiento, la apuesta es contar con un software vivo, en constante evolución, con aportes de la comunidad de desarrolladores y permitiendo el uso de la herramienta en otros contextos y por parte de otros docentes.

4.8 Utilización en Universidades

La utilización de la plataforma se da en diferentes universidades. Uno de los casos es en la asignatura Paradigmas de Programación, materia obligatoria del segundo nivel de la carrera de Ingeniería en Sistemas de Información, en el ámbito de la Universidad Tecnológica Nacional, tanto en la Facultad Regional Buenos Aires como en la Facultad Regional Delta. *Mumuki* también se usó en la materia Paradigmas de Programación, pero de la Universidad Nacional de San Martín, que se ubica dentro del sexto y último cuatrimestre de la Tecnicatura Universitario en Programación Informática.

4.9 La mirada de los docentes

Los docentes consultados de las diferentes universidades explican que las guías de aprendizaje fueron recomendadas para aquellos estudiantes que no pudieron asistir a alguna clase, mientras que las guías de desafío fueron recomendadas para todos, para

practicar fuera del aula y más allá de los trabajos prácticos requeridos para la aprobación de la materia, con carácter opcional. Además, en algunos cursos fue utilizado en tiempo de clase, en el espacio de laboratorio de sistemas, y en otros cursos algunos de los ejercicios fueron requeridos puntualmente como trabajo práctico de entrega obligatoria.

Otra realidad que se percibe analizando la complejidad de los ejercicios y la cantidad de soluciones, y teniendo en cuenta las observaciones de los docentes de los diferentes cursos, es que la mayoría de los estudiantes en algún momento de la cursada de la materia se sienten lo suficientemente seguros como para dejar de usar *Mumuki* y pasar a utilizar el entorno profesional del lenguaje de programación, o por cierto tiempo los utilizan simultáneamente.

4.10 Producción de los estudiantes

Analizando estadísticas de la base de datos de soluciones subidas por los estudiantes, surge que fue usado de maneras variadas. Al momento, hay registradas más de unas 120000 soluciones y más de 500 usuarios, lo cual arroja un promedio de 200 soluciones por cada uno, de las cuales aproximadamente el 30% fue correcto, lo que sugiere que la solución correcta corresponde en promedio al tercer intento. Ciertamente, estos promedios son indicadores relativos, ya que hay ejercicios que la amplia mayoría lo resuelve correctamente al primer intento, y en el otro extremo, hay algunos ejercicios donde las soluciones correctas son muy pocas. Haciendo un seguimiento más detallado, hay estudiantes que han hecho un uso muy intensivo de la plataforma y casos con muy pocos ejercicios resueltos, lo que muestra que simplemente ingresaron a la plataforma para conocerla y por algún motivo no la continuaron utilizando.

5 Conclusiones

El aspecto central a destacar es la confluencia de una estrategia pedagógica con opciones y criterios definidos y una herramienta de software. Poniendo foco en el desafío inicial de abordar el carácter mediador de *Mumuki* como herramienta pedagógica para facilitar el proceso de aprendizaje, teniendo en cuenta la descripción de la plataforma y considerando sus primeras experiencias de utilización, es una herramienta que promete buenos resultados. Tal vez sea prematuro afirmar con grandilocuencia sus logros -que se verán con el tiempo-, pero la percepción inicial es que va dando pasos firmes. Hay dos indicadores evidentes de esta realidad: que en tan poco tiempo se haya adoptado por diferentes Universidades, por propio convencimiento de los docentes, y la cantidad de soluciones subidas por los estudiantes, que refleja un alto nivel de apropiación de la herramienta.

La diferenciación entre la plataforma en sí y el contenido de ejercitación -que para el estudiante es transparente ya que se presenta como un todo- permite la identificación de roles que confirman el carácter dinámico del proyecto y posibilitan su crecimiento. Los desarrolladores del software de la plataforma, más allá que también sean docentes, pueden seguir trabajando en la mejora continua del producto de manera

autónoma. A su vez, todo docente que quiera utilizar la plataforma para sus clases no necesita involucrarse con el desarrollo sino concentrase en el contenido, ya sea tomando el recorrido sugerido, adaptándolo a sus necesidades, o generando el propio según sus propias opciones pedagógicas.

Para equipos docentes que marcan la diferencia entre teoría y práctica, incluso separando las clases unas de otras, *Mumuki* puede ocupar parcialmente el lugar de soporte hacia la práctica. Pero en propuestas más integradoras como la que se sugiere se le puede sacar mayor provecho. De todas maneras, a partir de la experiencia de uso, el planteo de poder practicar los conceptos teóricos explicados en clase, generalmente mediante guías de desafío, es el más consolidado. La utilización de la plataforma en tiempo de clase como recurso para descubrir los conceptos a partir de la práctica es una apuesta de los desarrolladores que va calando en los equipos docentes, pero aún necesita maduración. Ciertamente un elemento clave para aprovechar mejor esta estrategia didáctica pasa más por la elaboración de contenido apropiado o por la reflexión que los mismos docentes hacen sobre su práctica que por cambios en la infraestructura de la plataforma.

Desde otra perspectiva, un aspecto a destacar es lo fecundo de la confluencia entre la vocación docente, la iniciativa para el desarrollo de software y la actitud investigadora. Que los docentes de programación puedan utilizar como recurso pedagógico un software creado por ellos mismos, es una singularidad que permite un sinnúmero de posibilidades, como aporte e incentivo a la investigación y por la potencialidad de ir adaptando permanentemente la herramienta a las nuevas necesidades que descubren y opciones que toman.

6 Trabajo futuro

En cuanto a la plataforma en general, es posible hacerle mejoras y seguir agregando opciones que tiendan a un mejor seguimiento, como así también seguir generando nuevos ejercicios y revisando los actuales para proponer recorridos más significativos. Pero la principal línea de trabajo pendiente es incluir más inteligencia a la hora de analizar la correcta utilización de los conceptos de la programación funcional.

Referencias

1. Esteva, Juan. *La dimensión tecnológica en la formación universitaria*. Ed. por Plaza y Valdés Editores. Mexico, 1997.
2. Burbules, Nicholas *Educación: riesgos y promesas de las nuevas tecnologías de la información.* Ed. por Granica. Buenos Aires, 2001.
3. Freire, Paulo, *Carta a quien pretende enseñar,* Siglo Veintiuno, Buenos Aires, 1994.
4 Secretaría Académica y de Planeamiento (UTN), *Didáctica en la Universidad,* Universidad Tecnológica Nacional, Buenos Aires, 2007
5. Ranciere, Jacques, *El maestro ignorante*, Laertes, Barcelona, 2002.
6. Vigotsky, Lev. *Pensamiento y lenguaje,* Visor, Madrid, 1993
7. Bergmann, Jonathan; Sams, Aaron, *Dale la vuelta a tu clase*, Innovación educativa, España