Recursos pedagógicos tecnológicos para aprender a programar

Lucas Spigariol

Universidad Tecnológica Nacional, Facultad Regional Buenos Aires / Delta

lspigariol@gmail.com

Resumen. Frente a la dificultad que representa empezar a aprender a programar utilizando un lenguaje de alto nivel de uso profesional y valorando las ventajas que representa realizar prácticas de programación en computadora por sobre el uso exclusivo de papel, se presentan diferentes herramientas informáticas que pueden ser usadas como recursos pedagógicos para aprender a programar. Desde una fundamentación que abreva en el constructivismo y las pedagogías críticas latinoamericanas se proponen claves de análisis para las herramientas mencionadas y se busca ver en qué medida permiten un acercamiento sencillo y motivante a la programación para quien no tiene experiencia previa en la materia.

1. Introducción

En la enseñanza de la programación, como en tantas otras disciplinas, es fundamental combinar lo teórico con lo práctico, de manera que la práctica permita comprender mejor el sentido de los conceptos y a la vez que dichos conceptos permitan plasmar soluciones prácticas adecuadas a los problemas planteados. Tratándose de conceptos que fundamentan el desarrollo de software, su puesta en práctica implica escribir un determinado código de cierto lenguaje.

El uso de un lenguaje de programación de alto nivel para desarrollar soluciones informáticas, con su propio ambiente de trabajo, introduce una dimensión práctica y una aproximación profesional ineludible, a la vez que permite ir aprendiendo a construir soluciones, probándolas y validando los resultados con la máquina. Pero por otra parte, estas herramientas tienen una multiplicidad de opciones y requerimientos específicos que están pensados para agilizar la tarea de quien ya sabe programar, pero que para quien para quien está dando sus primeros pasos en la materia representa una dificultad adicional a la complejidad propia de la programación como tal. Afirmando que trabajar sobre la máquina tiene amplias ventajas, cabe reconocer una serie de

dificultades que también trae aparejado su uso en estudiantes sin experiencia previa. Por ejemplo, muchas veces se requiere del manejo de una amplia cantidad de conceptos y conocimientos previos para poder programar algo relativamente sencillo que funcione. El hecho de enfrentarse a la computadora y tener que estar atento a la complejidad de una sintaxis determinada, manipular entornos de desarrollo con múltiples opciones, configurar herramientas de soporte, poder interpretar y depurar errores, realizar seguimientos, validar resultados y todo lo que implica lograr que un programa funciones no es tarea sencilla. Puede ser desafiante para algunos estudiantes, pero probablemente frustrante para otros si no se realiza de la mejor manera. Así, al iniciar su formación en programación, entre otras dificultades, el estudiante se enfrenta a la rigurosidad del lenguaje de programación teniendo como herramienta su lenguaje natural cargado de sobreentendidos y ambigüedades, su acalorado entusiasmo -o desazón, cuando las cosas no salen- se choca con la frialdad inapelable de la computadora, partiendo de la multiplicidad de formas en que las personas interpretamos la información se encuentra con lo acotado de los tipos de datos que disponen los lenguajes de programación.

2. Objetivos

Una de las tensiones frecuentes en el ambiente docente de programación es si la propuesta pedagógica se limita a escribir el código en "papel" o si se decide implementarlo sobre una computadora u otro dispositivo afín donde realmente se ejecuten los programas. Descartando el camino que propone la primera opción, aparece no sólo la nueva decisión acerca de qué lenguaje de programación emplear, sino en qué momento y de qué manera hacerlo, y especialmente, qué recursos, herramientas y estrategias pedagógicas se ponen en práctica para superar las dificultades planteadas que tienen los lenguajes profesionales.

Frente a esta situación, se puede retrasar las instancias de práctica concreta con los lenguajes hasta tener una base conceptual más sólida –lo cual tiene el riesgo de demorar demasiado la práctica- o se puede explicar rápidamente muchos conceptos para poder acceder más rápidamente a una práctica concreta –lo que tiene el riesgo de apabullar a los estudiantes con numerosos conceptos sin el tiempo necesario para interiorizarlos adecuadamente.

Sosteniendo que en ambos extremos la disociación entre teoría y práctica que se genera es ciertamente contraproducente para el proceso de aprendizaje del estudiante, el objetivo del presente trabajo consiste en encontrar, describir y analizar herramientas de desarrollo de software que puedan ser utilizadas como herramientas pedagógicas para facilitar el acercamiento del estudiante sin experiencia previa a las herramientas profesionales de desarrollo de sistemas.

3. Convicciones iniciales y claves de análisis

Las convicciones que conforman el marco teórico desde el cual se desarrolla la investigación parten de entender a la enseñanza de la programación como una tarea en

la cual ni la impronta técnico profesional que se le imprime a la formación hace que pierda sentido el esfuerzo pedagógico de los docentes para llevar adelante su tarea educativa, sino todo lo contrario: la complejidad de los contenidos y lo cambiante del contexto profesional requiere una reflexión crítica permanente que permita establecer estrategias pedagógicas específicas.

Desde este posicionamiento, resulta fundamental prestar atención a una serie de aspectos pedagógicos desde los cuales tiene sentido analizar los recursos de tecnología educativa utilizados para enseñara a programar:

3.1 La necesaria, conflictiva y enriquecedora articulación entre teoría y práctica.

Un aporte importante de diversas corrientes pedagógicas es que permanentemente intentan plantear una interrelación fecunda entre teoría y práctica. Entre ellas, es enriquecedora la perspectiva crítica, que interpreta la relación práctica/teoría desde la acción/reflexión: "Si los hombres son seres del quehacer esto se debe a que su hacer es acción y reflexión. Es praxis. Es transformación del mundo. Y, por ello mismo, todo hacer del quehacer debe tener, necesariamente una teoría que la ilumine. El quehacer es teoría y práctica. Es reflexión y acción. No puede reducirse ni al verbalismo ni al activismo."[1] Es evidente que para la enseñanza de una disciplina instrumental como la programación es conveniente desarrollar una dinámica que combine lo teórico con lo práctico. Abstractamente, puede ser aceptado como un principio básico sin mayores discusiones, pero lo interesante es que al contextualizarlo en una práctica docente concreta surgen las inevitables –y bienvenidas- preguntas acerca de qué teoría se está hablando, qué se entiende por práctica y, por sobre todo, cómo se define o califica la “combinación”. Una primera aproximación es que la práctica permita comprender mejor el sentido de los conceptos teóricos y a la vez que dichos conceptos permitan plasmar soluciones prácticas adecuadas a problemas reales.

Se entiende a la actividad docente como un quehacer critico, creador y recreador, donde se asume la relación conflictiva y contradictoria entre teoría y práctica, no como una dicotomía, sino como interrelación fecunda que da lugar a nuevos aprendizajes.

3.2 La gradualidad y complejidad del proceso de aprendizaje

Los procesos de enseñanza y aprendizaje son complejos y requieren articulación y gradualidad. Lo generalmente abultado del elenco de contenidos que se prescriben en los programas de las asignaturas, las relaciones entre ellos y lo potencialmente inabarcable que se esconde detrás de cada término en el ámbito profesional actual, hace necesaria una cuidadosa estrategia de selección, secuenciación y articulación de su desarrollo. De esta manera, se trata de no apabullar al estudiante de entrada, de no plantear un ritmo demasiado intenso o una velocidad excesiva, como por otra parte tampoco subestimar su capacidad. En particular, Vigotsky, cuando desde su mirada

constructivista del aprendizaje se refiere a la zona de desarrollo próximo, está dando pistas acerca de cómo plantear gradualmente el desarrollo de los temas.

Como punto de partida, se asume la concepción constructivista de la enseñanza y del aprendizaje que explica el proceso educativo focalizando el análisis de la interactividad entre los participantes en colaboración y entendiendo a la educación en una dinámica constante de construcción y reconstrucción. Seymour Papert, discípulo de Jean Piaget y creador del lenguaje Logo, un emblema del software educativo orientado a la programación, tomó su concepción constructivista acerca del aprendizaje y a partir de ella expresó su propia filosofía educativa, denominada "construccionismo". Afirma que el aprendizaje se produce mediante la construcción de estructuras de conocimiento en diferentes circunstancias del aprendizaje y añade la idea de que esto ocurre más felizmente en un contexto donde la persona que aprende se involucra, participa en una actividad de construcción, con creatividad e iniciativa. Es más gráfico como él mismo lo explica: "El niño asimila los conocimientos de la misma forma que asimila el alimento; el aprendizaje, como proceso, se puede comparar a la transformación que sufre el alimento asimilado. El niño asimila conocimientos constantemente por medio de esquemas que se mantienen hasta el momento en que resulta necesario sustituirlos con otros esquemas nuevos, porque los anteriores se han vuelto insuficientes. Toda información nueva se compara con el esquema que el niño se ha creado y éste tiene validez hasta que aguanta la comparación; cuando ya no es así, se tiene un desequilibrio que implica la necesidad de volver a equilibrar, a reorganizar la estructura cognitiva"[2]. El fundamento de estas ideas son las categorías de asimilación y acomodación propias del constructivismo.

3.3 El protagonismo del alumno

Resulta clave propiciar por todas las formas posibles que los alumnos sean protagonistas del proceso de aprendizaje, ya sea desde una apreciación del carácter instrumental de la motivación y participación como favorecedora de mejores resultados académicos, hasta una valoración ética del ser estudiante.

En relación a lo último, se trata de educar para la libertad y la esperanza como fundamento de la vida profesional, ciudadana y democrática, y contra cualquier tipo y forma de autoritarismo, sometimiento y opresión. Esta mirada representa para todos aquellos que asumen el desafío de la educación con un posicionamiento complejo y explícito: la opción por una educación crítica, comprometida y esperanzada, sin por ello perder calidad, todo lo contrario. "Educar sigue siendo un desafío que nos convoca éticamente, es una práctica ciudadana y política: cómo educar, por qué a quién, para qué sociedad, pensando en qué presente y en qué futuro, para qué país. Educar se sostiene también hoy – pese a que muchos intenten contradecirnos con argumentaciones falaces-, en los mismos valores democráticos de urbanidad, respeto, solidaridad, ética, justicia y responsabilidad." [3]

3.4 El rol docente en tanto 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. Lejos de de propiciar cualquier tipo de "facilismo" que caricaturice su tarea, se entiende el rol docente como mediador entre los contenidos y el estudiante y promotor de todo tipo de recursos que favorezcan el proceso de aprendizaje. En otras palabras, una de las convicciones que alienta este trabajo, es que lo que hace prestigiosa a una institución, no es lo alto de su vara ni su nivel de exigencia por sí mismo, sino los caminos que propone para que el estudiante aprenda significativamente.

En su emblemático libro "Cartas a quien pretende educar", Paulo Freire se dirige con profunda simpleza y sinceridad a todos aquellos que desde diversos ámbitos y realidades asumen el desafío de la tarea educativa. Entre otras apreciaciones, cuestiona la vocación y la profesionalidad, las actitudes autoritarias y espontaneístas y explicita las características fundamentales de todo educador progresista y transformador: Destaca la humildad de que nadie lo sabe todo ni nadie lo ignora todo, valorando la apertura a escuchar. También, recupera, junto a la valentía de amar, la valentía de luchar y una actitud de superar el miedo. Señala la importancia de la tolerancia necesaria para convivir con quien es diferente, con respeto y ética, con una tensión entre paciencia e impaciencia y fundamentalmente, alegría de vivir.

Vigotsky sostiene que sin ayuda no es posible el aprendizaje y le otorgan al docente la posibilidad de pensarse como un sujeto tan activo como sus estudiantes en el proceso educativo. "Ni el directivismo de un docente que todo lo sabe y todo lo explica sin dejar espacio a la construcción, ni la ausencia de un docente que espera que sus estudiantes construyan por sí mismos el conocimiento que a la humanidad le ha llevado siglos"[4].

3.5 Enseñar a programar

Complementariamente a las anteriores, que son de naturaleza netamente pedagógica y aplicables a múltiples áreas del conocimiento, por último se expresa una convicción que pone su foco en la concepción acerca de lo que significa la programación, o más precisamente, la enseñanza de la programación.

En coincidencia con la iniciativa oficial de Program.ar, que tiende a acercar a los jóvenes al aprendizaje de la programación y las ciencias del a computación en general y a concientizar a la sociedad en general sobre su importancia, se presentan algunos criterios que desmitifican ciertas visiones construidas social y educativamente sobre la enseñanza de la computación:

Frente a la tendencia a que enseñar computación consiste en enseñar a usar paquetes de software para diferentes tareas cotidianas, educativas, creativas, comunicativas, etc. se plantea la enseñanza de "algoritmos, modelos, formas de representación de la información, lenguajes, programación, etc".[5]

Un prejuicio con el que muchas veces se aborda el tema sugiere que las Ciencias de la Computación, en tanto algoritmos, modelos, representaciones, abstracciones, etc. son un contenido muy difícil de aprender. En cambio, se sostiene que "la enseñanza de los conceptos y competencias centrales de computación se puede hacer en forma espiralada"[6], como por ejemplo el algoritmo de la suma, en matemáticas, que se aprende desde temprana edad.

Desde una mirada defensiva hacia las innovaciones que engloba bajo una misma etiqueta de "tecnología" una gran variedad de fenómenos, se suele aludir a que necesitamos volver a lo básico, a la lectura, a la escritura y a las operaciones matemáticas, diagnosticando que aprender las tecnologías modernas empeora el rendimiento de las habilidades realmente básicas. En contrapartida, con un conocimiento más preciso de lo que implica aprender sobre Ciencias de la Computación, se afirma que "desarrolla el pensamiento computacional, lo que también puede mejorar el rendimiento en otras áreas, especialmente en el pensamiento matemático que es en donde tiene su origen"[7].

4. Antecedentes históricos

El antecedente más significativo acerca del uso didáctico de herramientas informáticas lo componen los desarrollos teóricos de Seymour Papert y su implementación del lenguaje de programación Logo. Se trata de un lenguaje de programación, creado con fines educativos, que tuvo gran repercusión a nivel mundial por su objetivo de acercar de una manera sencilla a los estudiantes una disciplina por entonces compleja y aun muy distante de la población en general. Junto con sus colegas del laboratorio de inteligencia artificial del Instituto Tecnológico de Massachussets, tradujeron su concepción de la enseñanza de la programación a una herramienta de aplicación concreta para la educación.

Logo consta de instrucciones que sirven para generar otras instrucciones que, a su vez, se pueden ensamblar en un programa. Se convierten ellas mismas en lenguaje y se pueden utilizar para otras órdenes. En este sentido, Logo es un lenguaje de procedimiento. Los programas se crean reuniendo las órdenes, las funciones primitivas, están muy cerca del lenguaje natural y es un lenguaje que se puede desarrollar.

Una de las características más populares de Logo es la realización de gráficos a partir de utilizar la "tortuga" - que se transformó en el ícono del software- como cursor que se va desplazando por la pantalla a medida que se ejecutan las instrucciones dejando su rastro a modo de dibujo.

Papert considera que en el uso conservador del ordenador es éste quien controla al niño, mientras que en el ambiente Logo es el niño quien toma las riendas y "programa", controlando al ordenador. Esto que parece tan simple conlleva un cambio substancial en el concepto de la educación. El fundamento está en que se toma consciencia de la forma en que construyen las nuevas ideas y su aprendizaje. Por ejemplo, para enseñarle a la tortuga a realizar un movimiento el niño lo tiene que hacer él primero, al menos mentalmente y transferir esa reflexión al programa, con lo cual ya se está llevando a cabo una reflexión sobre las propios acciones y los propios pensamientos.

En este lenguaje, el proceso de crear el propio producto es más importante y educativo que el producto terminado en sí.

5. Análisis de herramientas pedagógicas tecnológicas para aprender a programar

Reconociendo que "el software educativo durante los últimos años, ha tenido un creciente desarrollo y gran parte del mismo ha sido realizado en forma desorganizada y poco documentada"[8] hay experiencias que merecen destacarse.

Existen diferentes instituciones educativas que manteniendo la preocupación pedagógica acerca del aprendizaje de la programación, hacen una relectura del panorama general del desarrollo de software y a partir de sus conclusiones utilizan herramientas específicas con fines educativos. Más aún, el de sistemas es un ámbito donde la necesidad y la capacidad confluyen y es posible desarrollar el software que se utilice con fines educativos en el mismo lugar. Que los docentes de programación puedan utilizar como recurso pedagógico software creado por ellos mismos, es una singularidad que permite un sinnúmero de posibilidades.

Más que plantear un exhaustivo elenco de experiencias de utilización y desarrollo de herramientas existentes se proponen algunos casos significativos que presentan características distintivas, a partir de las cuales se pueden definir categorías de sistematización de otras herramientas. Cada una de ellas presenta un camino diferente de abordaje de la problemática planteada.

5.1 Lenguajes de programación con fines educativos

Siguiendo el camino abierto por *Logo*, un recurso frecuente es comenzar el proceso de aprendizaje con un lenguaje de programación diseñado con fines educativos.

Dentro del abanico de opciones, se destaca una herramienta didáctica denominada Scratch, desarrollado –al igual que Logo- por el Instituto Tecnológico de Massachusset. Entre otros, Karen Brennan y Mitchel Resnick son quienes llevan adelante la reflexión pedagógica del proyecto que enfatiza el hecho de diseñar, en tanto crear y no solamente interactuar con cosas ya hechas, y promover un proceso de aprendizaje a partir de opciones personales relevantes y significativas. A su vez, plantea un modelo de aprendizaje colaborativo, trabajando con otros en las creaciones, y reflexivo, revisando permanentemente las propias prácticas creativas. Los mismos creadores afirman que "aprendiendo las lecciones de la experiencias de Papert con Logo, hemos diseñado Scratch para ir más allá de Logo, en tres dimensiones, por lo que la programación más manipulable, más significativo, y más social."[9].

El lenguaje Scratch permite a estudiantes con ninguna experiencia en programación realizar cosas sin tener que preocuparse por la sintaxis, dado que provee un lenguaje manipulable con bloques que se arrastran y se combinan entre sí. Uno de los principales objetivos es motivar a los estudiantes a crear proyectos de programación en donde realizan juegos o animaciones interactivas. Desde el punto de vista técnico,

es multiplataforma y fácil de instalar, lo que lo convierte en una alternativa accesible a la hora de implementar actividades en las aulas con esta herramienta, y cuenta con una comunidad numerosa y un foro donde se comparte con otros usuarios los proyectos realizados en todas partes del mundo. Existe mucho material educativo, libros sobre enseñanza inicial de la programación y páginas con tutoriales.

Teniendo en cuenta el rol del estudiante, ciertamente el lenguaje propicia su protagonismo y apela fuertemente a su creatividad. Es útil para trabajar en equipos y permite una participación activa de todos. La variedad de gráficos disponibles, la facilidad para incluir movimientos y sonidos hace que sea relativamente sencillo hacer aplicaciones interactivas y con fuerte impacto visual. Esta característica, que es fundamental para la motivación en un estudiante de menor edad, corre el riesgo de transformarse en un arma de doble fijo en un estudiantado universitario, ya que en vez de motivar, su aspecto puede ser considerado infantil y despertar rechazo. De todas maneras, tiene un abanico de herramientas y desde la propuesta docente puede dejarse de lado la interfaz gráfica y concentrarse en los conceptos, herramientas y formas de programar el núcleo de una aplicación y hacer un uso progresivo de herramientas de mayor abstracción.

La posibilidad de articular teoría y práctica es evidente para los conceptos básicos de programación, aunque hay otros conceptos más avanzados que no están contemplados en su diseño. Dentro de los múltiples enfoques posibles de la programación y de las diferentes perspectivas o acentuaciones que cada docente o institución educativa hacen de los conceptos que quieren que sus estudiantes aprendan, es muy relativo a qué se denomina un concepto básico y cuál se considera avanzado. Por ejemplo, Scratch permite manejar muy sencilla e intuitivamente las estructuras de control tales como decisiones o iteraciones, pero la forma en que administra las llamadas a funciones y el paso de parámetros está acotada de manera tal que resulta muy diferente a la forma en que se trabaja en lenguajes de uso profesionales.

Una característica central, que lo hace especialmente apropiado para dar los primeros pasos en programación, es que en vez de escribir el código, se lo construye moviendo bloques, a modo de rompecabezas, eligiendo entre opciones preestablecidas y con un mínimo de escritura que se limita a nombrar elementos o definir ciertos valores. A su vez, esta característica es la que lo hace diferente a los lenguajes de programación de uso industrial, lo que implica una transición más difícil en cuanto al estilo de trabajo. En función de esto, una de las claves para su uso está en la manera en que el docente lo inserta en su propuesta pedagógica, lo articula con otras herramientas, y gradúa su aplicación, por ejemplo, utilizándolo inicialmente para introducir y afianzar ciertos conceptos y luego progresivamente dejándolo de lado y pasando a utilizar otras herramientas didácticas más parecidas a los lenguajes de algo nivel. En otras palabras, puede cumplir una función de "andamiaje" pedagógico, fundamental en cierto momento para sostener el proceso y luego totalmente prescindible.

5.2 Frameworks y bibliotecas

Dentro de la amplia variedad de lenguajes de programación en la actualidad, hay varios que siendo multipropósito, con una inserción real en la industria del software

tienen una sintaxis y formulación relativamente simple, que los hace más accesibles que otros a ambientes educativos. Aprovechando esta facilidad, pero sin detenerse allí, se han creado diferentes bibliotecas, frameworks y entornos de desarrollo que proveen de una variedad de opciones más sencillas de manejar, que esconden cierta complejidad, y que detrás tienen ejecutando al lenguaje real de programación.

Uno de ellos, basado en el lenguaje *Python*, se denomina "Pilas-engine" y es una plataforma que está orientada a programar haciendo juegos de manera simple. Es un proyecto iniciado en Argentina por un estudiante de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional, y que luego fue creciendo y sumando participantes y respaldo. En particular, tiene una versión que sobre el sistema operativo *Huayra* y viene incluido en las computadoras del programa oficial de "Conectar Igualdad", que se distribuyen a los estudiantes de los colegios de todo el país. Así se lo presenta desde dicho programa: "*Pilas-engine* es una herramienta que permite a estudiantes y docentes construir sus propios videojuegos, usando un enfoque constructivista que les permite aprender mientras realizan un proyecto de software. A través de la creación de un videojuego, los alumnos y docentes tienen la oportunidad de descubrir el funcionamiento de las computadoras, aprender a darle órdenes usando un lenguaje de programación y llevar adelante proyectos creativos, de forma colaborativa y significativa"[10].

Se trata de un software multiplataforma y como la mayoría de las herramientas que apuntan a lograr una difusión sin un espíritu netamente comercial, es software libre, por lo que se puede copiar, modificar y distribuir el motor libremente. Las características que lo hacen propicio para facilitar el aprendizaje de la programación, es el uso de sesiones interactivas, donde apenas se escribe el código se ve su efecto visual, sin mediar los procesos engorrosos de compilación o tener que completar la aplicación para poder probarla. Tiene la facilidad de tener la documentación completamente en español e incluir varios ejemplos y tutoriales para ir lentamente paso a paso haciendo cosas modestas, lo que se combina con tener a disposición todo lo que el lenguaje Python provee, permitiendo también plantear programas de suma complejidad. Cuenta con una serie de objetos prediseñados y tareas frecuentes resueltas, lo que hace sencillo los primeros pasos en el desarrollo de un programa. Dentro de la variedad de posibles aplicaciones, el enfoque y el estilo de los gráficos elegidos refleja que está dirigido a programar videojuegos, lo cual es además un factor motivante no sólo para niños o adolescentes, sino para toda persona que entiende que estudiar y aprender no tiene porqué se algo aburrido. En la presentación de la herramienta, Hugo Ruscitti, su creador, sostiene que "pilas es una gran oportunidad de acercar el desarrollo de videojuegos a todas las personas, principalmente jóvenes con interés en aprender a programar computadoras y darle vida a sus ideas"[11] y reconoce que está profundamente inspirada en las ideas de Seymour Papert y el lenguaje *Logo*.

Más allá de la interfaz gráfica, la forma de escribir los programas con estas herramientas se rige por las mismas normas que el lenguaje de programación real, lo cual tiene la ventaja de que quien aprendió a usar correctamente la herramienta y es capaz con ella de de crear programas de cierta complejidad, está en condiciones de seguir utilizando el lenguaje *Python* sin *Pilas* y tener una transición mucho más suave hacia otras herramientas profesionales de desarrollo.

Desde el rol docente como mediador, es posible ir graduando la complejidad de los problemas que se propone resolver y la incorporación progresiva de conceptos. Con las mismos personajes que propone la plataforma, se pueden hacer tanto cosas muy básicas como otras más complejas, y el hecho de tener como marco el lenguaje de programación real le abre las puertas al estudiante de curiosear e investigar más allá de lo evidente y le permite al docente profundizar en conceptos de programación más avanzados que crea conveniente, articulándolos con una práctica concreta.

En contrapartida, esta fuerte dependencia con un lenguaje de alto nivel, provoca que para empezar a construir algo, sea necesario previamente algunos conceptos y cierto conocimiento de los comandos específicos y reglas sintácticas del lenguaje, lo cual si bien no constituye un impedimento, requiere que el docente lo tenga presente y sea criterioso a la hora de incluir esta herramienta en su propuesta curricular, por ejemplo previendo cierto tiempo de preparación, de explicaciones iniciales y desarrollando casos sencillos o problemas ya resueltos que sirvan de guía y faciliten los primeros pasos con *Pilas*.

5.3 Diagramas y pseudocódigos

Una estrategia frecuente en espacios educativos es el uso de pseudocódigos, es decir, lenguajes simples, reducidos en su vocabulario pero flexibles a incorporar expresiones no predefinidas y con una sintaxis menos estricta que un lenguaje profesional. Son lenguajes inexistentes en la vida real, que no pueden ser interpretados por la computadora, pero que son pensados con fines didácticos para plasmar los principales conceptos, para construir algoritmos simples y progresivamente ir incorporando complejidad y facilitar el acercamiento a un lenguaje de programación de algo nivel. Para el caso de los países donde no se habla habitualmente inglés –el idioma en el que están basados prácticamente todos los lenguajes de programación- el pseudolenguaje tiene la ventaja de usar términos conocidos de la propia lengua.

Otro recurso que se usa en numerosos espacios educativos elaborar una serie de diagramas como paso previo o en paralelo a la escritura de código propiamente dicho. Los más típicos son los diagramas de flujo, que con sus variantes en cuanto a la simbología, convenciones y alcances, coinciden en tener como objetivo representar un determinado algoritmo y facilitar el seguimiento del flujo de ejecución. En otras palabras, el diagrama ofrece una versión simplificada del código del programa, en el que por ejemplo, de un golpe de vista es posible detectar las estructuras básicas de control.

Uniendo estas dos ideas, junto a la necesidad de generar un mayor acercamiento a los lenguajes de uso profesional, existe una herramienta, llamada “PSeInt”, que es útil para el aprendizaje de los primeros pasos en programación estructurada, que combina un simple pseudolenguaje en español con un editor de diagramas de flujo, y tiene la posibilidad de generar código en un lenguaje de programación de alto nivel. Su desarrollo comenzó en el año 2003, como proyecto final de una materia en carrera de informática del a Universidad Nacional de Lanús. Permite centrar la atención del estudiante en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos.

PSeInt es un intérprete de pseudocódigo, que sirve para enseñar los conceptos, la lógica básica, la abstracción necesaria para empezar a programar, sin tener que preocuparse por un lenguaje y sus particularidades. Consta de un editor con coloreado de sintaxis, autocompletado, ayuda rápida en pantalla y la posibilidad de visualizar el algoritmo como diagrama de flujo y de convertirlo luego a código del lenguaje C++. Lo más importante que la computadora interpreta y ejecuta el algoritmo en caso de ser correcto y señala los errores en caso contrario, lo cual es mucho más difícil e interesante que lo primero. Pablo Novaro, el creador de la herramienta, describe la motivación inicial en clave autobiográfica: "En la universidad, antes de hacernos programar en un lenguaje real, nos trataban de enseñar los conceptos básicos, como características de un algoritmo, constantes, expresiones, estructuras de control y arreglos, utilizando un pseudolenguaje. El pseudocódigo parecía un arma de doble filo: si bien era mucho más simple que un lenguaje real, no había posibilidad de correrlo y la única forma que había para saber si el programa estaba bien, era la consulta al profesor. Con más de 200 ingresantes, apenas 3 profesores, y mil formas distintas pero igualmente válidas de resolver los ejercicios, la duda era eterna. Cuando al final del año nos dijeron que debíamos presentar un proyecto final para aprobar el curso, no lo dudé: iba a escribir un intérprete de pseudocódigo, para que cualquiera pueda probar su algoritmo y ver si daba lo que tenía que dar"[12].

Desde el punto de vista de la experiencia como estudiante, en relación a las estrategias que se basan en papel, la posibilidad de poder trabajar sobre la máquina, probar que las soluciones funcionen realmente, son un estímulo motivacional fundamental y una forma más interactiva de participar creativamente en el desarrollo de una solución informática. Por mencionar un ejemplo, la posibilidad de realizar un programa, poder ver su código o pseudocódigo y el diagrama, y comprobar cómo al hacer pequeños cambios en el código se actualiza el diagrama –o viceversa- y se obtienen nuevos resultados, es realmente útil para un estudiante que se inicia en la materia.

6. Conclusiones

Los diferentes recursos presentados, como así también las tipologías en la que se los encuadra, con sus matices y diferencias, coinciden en dar una respuesta acorde a la problemática plantada.

Retomando el punto de partida de analizar herramientas que permitan aprender a programar mediante la construcción de soluciones que se puedan ejecutar realmente sobre la computadora, se constata en todos los casos presentados que se trata de una característica central. A su manera, ya sea por ejemplo con mayor o menor preponderancia de lo gráfico, todas permiten que el estudiante pueda ver su creación funcionando, brindándole resultados concretos y palpables a los problemas planteados. Esta posibilidad no sólo es gratificante y motivante para que el estudiante siga aprendiendo, sino que es un feedback crucial para corregir, modificar y extender los programas realizados.

Un elemento fundamental en común es que permiten poder practicar con la computadora desde el primer momento y mucho más sencillo que otros lenguajes, es decir, con un menor bagaje de conocimientos previos. En otras palabras, tener código fun-

cionando mucho más rápido que utilizando directamente lenguajes de uso profesional, y evita que los estudiantes antes de poder correr su primer programa tengan que comprender muchos conceptos.

El carácter introductorio y facilitador del aprendizaje de estas herramientas se puede sintetizar con la metáfora de ser un "primer escalón" para entender bien lo básico, y poder avanzar a conceptos más importantes con fundamentos sólidos. A su vez, propician un vínculo más fluido con la máquina, que deja de ser un obstáculo a superar o un enigma a develar para transformarse en la herramienta privilegiada con la cual desplegar el potencial creador y creativo típico de la programación.

Una línea de investigación pendiente para continuar con la temática, consiste en hacer un trabajo de campo que dé cuenta de la utilización de estos recursos por parte de grupos de estudiantes en el marco de enmarcados de propuestas pedagógicas que las contextualicen. Desde esta perspectiva, se podrían establecer conclusiones más profundas que esta primera aproximación, y que aunque no pretendan su generalización, permitan interpretar mejor la significatividad de la utilización pedagógica de estos recursos en sus respectivos contextos.

Referencias

1. FREIRE, Paulo. (1970) Pedagogía del Oprimido. Ed. Tierra Nueva. Montevideo. p 157
2. PAPERT, Seymour; HAREL, Idit (2002). *Situar el Construccionismo*. Centro Latinoamericano para la Competitividad y el Desarrollo Sostenible, Alajuela. Traducido de PAPERT, Seymour e HAREL, Idit (1991) *Constructionism.* Ablex Publishing Corporation.
3. BIXIO, Cecilia (2010) *Maestros del siglo XXI. El oficio de educar. Homenaje a Paulo Freire*. Ediciones Homosapiens. Rosario. p 23.
4. Ibíd., p 83.
5. PROGRAM.AR (2004) *Por qué todos tienen que aprender a programar.* Ministerio de Ciencia Tecnología e Innovación Productiva, Ministerio de Educación, Fundación Sadosky. (Consultado en program.ar, abril 2015)
6. Ibíd.
7. Ibíd.
8. CATALDI, Zulma (2000). *Metodología de diseño, desarrollo y evaluación de software educativo.* Tesis de Magister en informática. UNLP.
9. RESNICK, Mitchel. "Reviving Papert's Dream" en *Educational Technology*. Vol 52. N° 4. Julio-Agosto 2012. p 42).
10. CONECTAR IGUALDAD. (Consultado en huayra.conectarigualdad.gob.ar/noticias, abril 2015)
11. RUSCITTI, Hugo (2014) Sitio oficial de Pilas-Engine (pilas.readthedocs.org, consultada en abril 2015)
12. NOVARO, Pablo (2012) "Quien es quien: hoy PSeInt" Sitio web personal. (Consultado en cucarachasracing.blogspot.com.ar, en abril 2015)