Revista Sinapsis. ISSN 1390 – 9770
Vol. 1, Nro. 23, diciembre de 2023

Thinking challenge: Pensamiento computacional para el programa de Tecnología en Desarrollo de Software

Thinking Challenge: Computational Thinking for the Software Development Technology program

Mary Luz Rubiano Acosta

Universidad de San Buenaventura, Colombia, Ingeniera de Sistemas, Email: ingmaryr@gmail.com, Código Orcid: https://orcid.org/0000-0002-0848-6653

Contacto: ingmaryr@gmail.com

Recibido: 05-07-2023 **Aprobado: 05-10-2023**

Resumen

El presente artículo expone los resultados de investigación obtenidos a partir del proyecto educativo Thinking Challenge, sobre Pensamiento Computacional, implementado para estudiantes del programa académico Tecnología en Desarrollo de Software de la Universidad San Buenaventura (Bogotá-Colombia). El propósito del proyecto consistió en aumentar el nivel de desarrollo de pensamiento computacional en los estudiantes de dicho programa académico de la universidad. El ambiente de aprendizaje que se llevó a cabo está sustentado en el desarrollo de un curso virtual en Pensamiento Computacional estructurado en cuatro unidades de aprendizaje. La investigación se realizó bajo un enfoque mixto, privilegiando la perspectiva interpretativa. Se aplicó el diseño metodológico de investigación evaluativa correspondiente al modelo CIPP. Los resultados mostraron que es posible determinar una mejora en las habilidades de pensamiento computacional de los participantes; así mismo, se motivó una alta expectativa con relación a la incorporación del pensamiento computacional en su formación académica para un futuro desempeño profesional; además, se identificaron las estrategias didácticas que fortalecieron este proceso.

Palabras Clave: Pensamiento computacional, educación, conceptos core, prácticas core, desarrollo de software.

Abstract

This article presents the research results obtained from the educational Thinking Challenge project, on Computational Thinking, applied for students of the academic program Technology in Software Development of the Universidad de San Buenaventura (Bogotá-Colombia). The aim of the project was to increase computational thinking development in the students from this university academic program. The learning environment applied on the project is supported by the creation of a course in computational thinking structured in four learning units. The research was carried out through a mixed approach, favoring the interpretive perspective. The methodological design of evaluative research corresponding to the CIPP model was applied. The results showed that it is possible to determine an improvement in the computational thinking skills

of the participants. Likewise, a high expectation was motivated in relation to the incorporation of computational thinking in their academic training for a future professional performance; furthermore, the didactic strategies that strengthened this process were identified.

Keywords: Computational thinking, education, core concepts, core practices, software development.

Introducción

Desempeñarse en el mundo de la informática exige, cada día, mayores competencias, habilidades y conocimientos, tanto para los creadores como los usuarios de los servicios o contenidos informáticos, en correspondencia a las permanentes actualizaciones o progresos tecnológicos en el mundo digital. La exigencia computacional constituye el desafío global donde se espera solución a los grandes problemas actuales, en todos los escenarios posibles (K — 12, 2016). Pese al reconocimiento de dichas exigencias, la informática enfrenta a sí misma bastantes retos en su desarrollo, algunos de ellos asociados a la necesidad por generar una cultura computacional que denote una importancia tan fundamental como la lectura, la escritura y las matemáticas. Muchos de los estudiantes de hoy usarán las herramientas informáticas en sus carreras futuras, no solo en campos de ciencias, tecnología, ingeniería y matemáticas (STEM, por sus siglas en inglés) sino también en campos que no son STEM (Change the Equation, 2015).

Es así, que una serie de asociaciones y organizaciones a nivel internacional han emprendido la formulación y divulgación de un marco unificado que guíe el desarrollo de estándares y planes de estudio, desarrollen capacidades para la enseñanza de las ciencias de la computación y se implementen por vías al conocimiento de las ciencias de la computación; dicho marco de las ciencias de la computación ha sido denominado: K-12, una propuesta investigativa y educativa dirigida en atender los desafíos a una visión del siglo XXI, donde los estudiantes no solo son usuarios de computadoras, sino también creadores de las ciencias de la computación, logrando así una propuesta unificadora e integral para los planes de estudio en los diferentes niveles formativos.

La educación en ciencias de la computación implica una alfabetización informática, tecnología educativa, ciudadanía digital, tecnología de la información y las ciencias de la computación. Como la base de toda la informática, las ciencias de la computación son “el estudio de las computadoras y los procesos algorítmicos, incluidos sus principios, sus diseños de hardware y software, sus aplicaciones y su impacto en la sociedad” (Tucker et al., 2006, p. 2).

El marco de las ciencias de la computación K-12 promueve una visión en la que todos los estudiantes se involucran de manera directa en temas de las ciencias de la computación; abordan los problemas de manera innovadora; y desarrollan artefactos de la computación con una intención práctica, personal o social. Dicho marco establece una estrategia sustentada en la incorporación de *conceptos core* (es decir, lo que se debe saber) y *prácticas core* (es decir, lo que deben hacer los estudiantes). El marco de las ciencias de la computación K–12 organiza este conjunto de conocimientos en cinco conceptos core que representan áreas de contenido clave en ciencias de la computación y siete prácticas que representan acciones que los estudiantes utilizan para interactuar con los conceptos de manera significativa. Los conceptos core, del marco que

representan las áreas principales de contenido en el campo de las ciencias de la computación son: 1) Sistemas de Computación; 2) Redes y el Internet; 3) Datos y análisis; 4) Algoritmos y Programación; 5) Impactos de la Computación. Las prácticas core que representan los comportamientos que los estudiantes con conocimientos de computación y utilizan para involucrarse completamente con los conceptos core son: 1) Fomentar una cultura de las ciencias de la computación inclusiva; 2) Colaborar en torno a las ciencias de la computación; 3) Reconocer y definir problemas computacionales; 4) Desarrollar y usar abstracciones; 5) Crear artefactos computacionales; 6) Probar y refinar los artefactos computacionales; 7) Comunicar sobre la computación (K — 12, 2016).

La clave de efectividad o virtud en las prácticas core recae en la implementación y desarrollo del Pensamiento Computacional (PC), este proporciona un marco para el estudio de la computación de gran alcance, con aplicación amplia más allá de la computación en sí. El pensamiento computacional se refiere a los procesos de pensamiento implicados en la expresión de soluciones como pasos computacionales o algoritmos que pueden realizarse por computadora (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016). Esta definición se basa en la idea de formular problemas y soluciones en una forma que puede ser realizada por un agente de procesamiento de información (Cuny, Snyder y Wing, 2010) y la idea de que las soluciones deben tomar la forma específica de pasos computacionales y los algoritmos serán ejecutados por una computadora (Aho, 2011; Lee, 2016).

El pensamiento computacional resulta entonces como una herramienta indispensable en el desarrollo de habilidades tanto informáticas como humanas, que implica la forma de resolver problemas de manera inteligente e imaginativa (cualidades humanas que no poseen los ordenadores). Además, posee las características de combinar abstracción y pragmatismo, puesto que se fundamenta en las Matemáticas, un mundo de ideas, y se desarrolla a partir de proyectos de ingeniería que interactúan con el mundo real (Acevedo, 2018). Entre los rasgos que posibilitan un aprendizaje basado en pensamiento computacional, que menciona Wing (2006) se encuentran que: en el pensamiento computacional se conceptualiza, no se programa; son fundamentales las habilidades no memorísticas o no mecánicas; se complementa y se combina el pensamiento matemático con la ingeniería; y, lo importante son las ideas, no los artefactos.

Es a partir de la necesidad por estimular o fortalecer la experiencia formativa de los futuros profesionales en las áreas de la ingeniería de sistema o ingeniería de software que se formuló la propuesta Thinking Challenge, un proyecto educativo en pensamiento computacional para estudiantes de Tecnología en Desarrollo de Software de la Universidad San Buenaventura (Bogotá- Colombia) cuyo propósito estriba en aumentar el nivel de desarrollo de pensamiento computacional de los universitarios de dicha carrera y apuntar a una posible vinculación de esta propuesta educativa en el currículo del programa académico, e incluso sugerirlo en otros escenarios universitarios.

Población y muestra

El presente estudio se enmarcó en un enfoque mixto, el cual posibilita una lectura holística, inferencial, interpretativa, descriptiva y analítica respecto al fenómeno estudiado. En ese sentido,

Hernández, Baptista y Fernández (2014) establecen que los métodos mixtos representan un conjunto de procesos sistemáticos, empíricos y críticos de investigación e implican la recolección y análisis de datos cuantitativos y cualitativos, así como su integración y discusión conjunta, para realizar inferencias producto de toda la información recabada (meta inferencias) y lograr un mayor entendimiento del fenómeno bajo estudio. El alcance de esta investigación comprende así mismo un trabajo bajo el modelo de evaluación investigativa correspondiente a CIPP (por sus siglas en inglés) significando esto Contexto, Entrada, Proceso y Producto. A partir de estos criterios fue posible establecer, una vez que obtenidos los resultados de los análisis cuantitativos, cualitativos y mixtos, las inferencias, valoraciones, comentarios y conclusiones en la discusión.

Los participantes de la propuesta Thinking challenge corresponden a 33 estudiantes de primer semestre, de un total de 80 del programa académico de Tecnología en Desarrollo de software (en Colombia, además de la formación profesional también se oferta otros programas y niveles académicos previos a dicha titulación, siendo estos: técnico y tecnólogo). Se tomó en consideración esta población dado su reciente ingreso al programa, en donde aún no se han enfrentado a asignaturas que ofrezcan contenidos similares al pensamiento computacional.

La muestra, es decir, el número de participantes presentó una variación en lo corresponde al momento inicial frente al momento de finalización, en tanto, se empezó con un total de 33 estudiantes, pero debido a las condiciones de la emergencia sanitaria (Covid-19) una cantidad considerable de ellos no pudieron continuar en el proceso, siendo al final un total de 18 asistentes.

Participaron respondiendo los instrumentos de la fase de contexto y entrada 22 participantes, en proceso diligenciaron los instrumentos 9 y las entrevistas a profundidad fueron aplicadas a 6 estudiantes.

Instrumentos utilizados

En conformidad al enfoque seleccionado, se emplearon diferentes instrumentos para la recolección de datos los cuales permiten ser codificados como números y también analizados como texto o ser transformados de cuantitativos a cualitativos y viceversa (Hernández et al., 2014). Es así como los instrumentos diseñados y aplicados, que se mencionan a continuación, obedecieron a las fases del modelo CIPP, siendo estos:

- Fase de Contexto y Entrada: se realizó un cuestionario de autoconocimiento en pensamiento computacional; se aplicó una entrevista de caracterización a la población y una entrevista al coordinador del programa; así mismo, se recogió diagnóstico inicial, a través de un test, sobre conocimientos previos sobre pensamiento computacional.
- Fase de Proceso: durante el desarrollo del proyecto se implementó el curso de pensamiento computacional a través de Microsoft Teams, aprovechando la licencia que provee la universidad y todas las tecnologías asociadas de que dispone, como por ejemplo los canales a través de los cuales se realizaron encuentros grupales dando cabida al trabajo colaborativo y cooperativo, el curso se diseñó en cuatro unidades, cada una apunta a una de las prácticas core de pensamiento computacional del marco K12. Además, se dispuso

del correo institucional donde los estudiantes recibieron los contenidos antes de cada sesión.

Igualmente, durante esta fase se levantaron test de conocimientos sobre los aprendizajes obtenidos; se llevó a cabo un permanente diario de campo que visualiza las observaciones y valoraciones de las actividades; así como se diligenció un cuestionario, a manera de entrevista semiestructurada, que diera cuenta de las impresiones de los estudiantes respecto a las unidades y temáticas abordadas; también se tuvo en cuenta el cuestionario bloc de notas de clase.

- Fase de Producto, se llevó a cabo una prueba de conocimientos finales y la aplicación de una entrevista a profundidad donde los participantes pudieron evaluar el proyecto.

Análisis de resultados

Los resultados obtenidos, que a continuación se exponen, se encuentran organizados en conformidad a las etapas del modelo CIPP:

En la fase de contexto se identificaron las necesidades de los estudiantes frente al pensamiento computacional; la entrevista con el coordinador del programa de Tecnología en Desarrollo de Software evidenció que una de las necesidades urgentes corresponde al bajo nivel de desarrollo en habilidades de pensamiento computacional, que ha sido manifestada en forma recurrente por los docentes repercutiendo en el bajo rendimiento académico y deserción escolar. Dada la complejidad del programa académico, y en conformidad como lo manifestaron los estudiantes en los cuestionarios, su formación en básica secundaria les proporcionó básicos conocimientos, e inclusive mencionan un nivel nulo en algunos casos, en cuanto al pensamiento computacional se refiere.

De igual forma, el cuestionario de autoconocimiento en pensamiento computacional arrojó que tan solo el 35% de los participantes manifestaron estar en un nivel máster de competencia, así se corrobora el bajo nivel de desarrollo en competencias asociadas al pensamiento computacional. En este mismo instrumento los estudiantes calificaron en escala de 1 a 5 su desempeño o nociones referidas a las temáticas asociadas al pensamiento computacional; los resultados arrojaron que los conocimientos sobre programación en bloque tuvieron el menor promedio, seguido de algoritmos y lógica matemática.

Por cada una de las prácticas asociadas al pensamiento computacional del marco K-12 se establecieron criterios de desempeño con niveles junior, senior y máster obteniendo los siguientes resultados:

Frente al reconocimiento de habilidades de pensamiento computacional en la solución de problemas, se obtuvo que el 71% de los estudiantes se ubican en el nivel Junior, lo que significa que aún no resuelven problemas computacionalmente. En cuanto al desarrollo y uso de abstracciones (funcionalidades), se registró un 76% en este mismo nivel junior, por tanto, aún no cuentan con la capacidad analítica y conceptual para desarrollar sus propias abstracciones en los propios proyectos. Creación de artefactos computacionales tuvo un porcentaje de 52% de estudiantes en nivel Junior lo que traduce que aún no tienen en cuenta la planificación y análisis

a la hora de crear sus propios artefactos computacionales. Lo anterior permitió establecer que el nivel de manejo de pensamiento computacional de los estudiantes es bajo.

Dadas las temáticas que abordan los estudiantes en sus procesos académicos tanto en bachillerato como en las asignaturas de su formación se esperaría que no presentaran dificultades a la hora de asumir su carrera; sin embargo, el autodiagnóstico en pensamiento computacional realizado a los estudiantes involucrados en el proyecto, determinó que aspectos indispensables para un desarrollador de software no se abordaron con anterioridad, tal es el caso de programación en bloques que solo el 6% de los encuestados manifestó algún conocimiento al respecto; para el caso de algoritmos, que es una habilidad fundamental en los procesos de programación solo el 11% demuestran algún conocimiento; frente al pensamiento crítico el 17% de los encuestados reconoce su importancia y utilidad, estas cifras denotan un porcentaje bajo en competencias primordiales para esta disciplina.

La fase de entrada permitió identificar la disponibilidad de recursos disponible para el desarrollo de la propuesta, y a partir de estos planear las acciones a seguir. Inicialmente, el proyecto fue planteado de modo presencial, pero debido a la emergencia sanitaria se optó por trasladar la implementación a la modalidad virtual haciendo uso de Microsoft Teams, por esta razón los estudiantes tuvieron que disponer de sus propios equipos de cómputo y conexión a Internet, que al final resultaron suficientes para la consecución del proyecto.

En la fase de proceso, a pesar de que en forma recurrente los estudiantes manifiestan que todas las actividades les resultan necesarias, se pudo establecer que las actividades del proyecto Thinking Challenge que tuvieron mayor participación e interés por parte de los estudiantes son las asociadas a: análisis de problemas, diseño e implementación de patrones, abstracción, programación orientada a objetos y usabilidad. Es interesante apreciar la relación entre las temáticas de interés y las fases del ciclo de vida de desarrollo de software donde resultan aportando significativamente a este proceso.

En la fase de producto se identificaron las mejoras en cuanto a nivel de pensamiento computacional, se trianguló la información suministrada a través de cuestionarios, diarios de campo, bloc de notas de clase y entrevistas semiestructuradas, luego se abstrajeron los logros de los estudiantes frente a: problemas, algoritmos, abstracciones, patrones, planeación, prototipos, usabilidad, en la categoría LogroPC.

Para ello, en cada sesión, se dispuso del cuestionario bloc de notas de clase, con la estructura que se aprecia en la ilustración 1, allí se registró el antes y el después de cada encuentro, permitiendo identificar puntualmente las temáticas en las que los estudiantes presentaron avances más significativos.

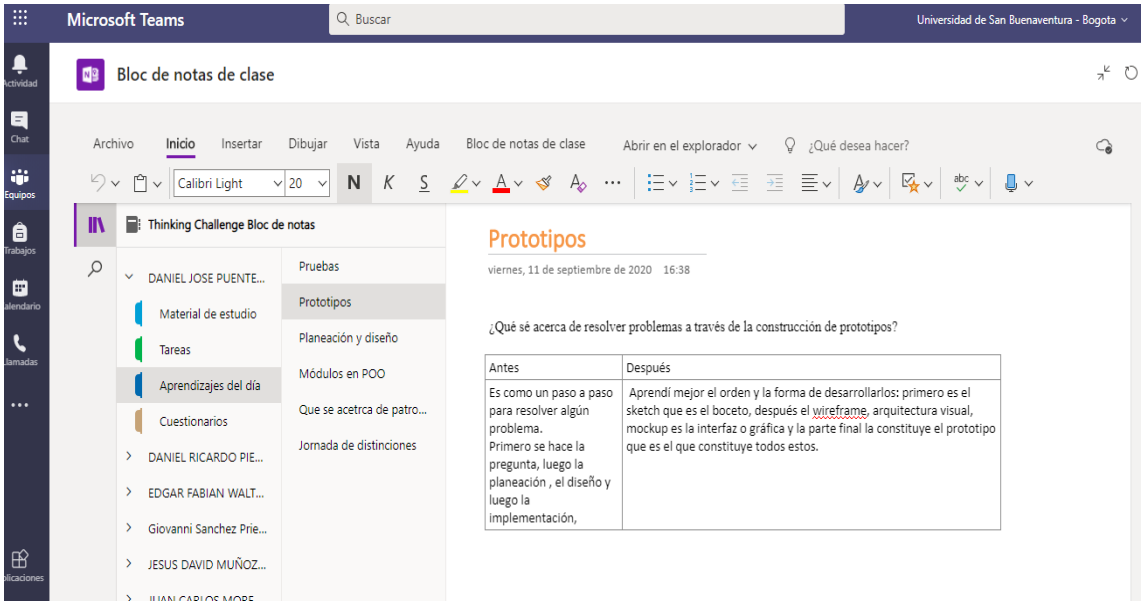

Figura 1. Captura de pantalla bloc de notas de clase, aprendizajes del día

Fuente: Plataforma institucional Universidad de San Buenaventura (2020)

La primera temática abordada con los estudiantes corresponde a resolución de problemas haciendo uso de pensamiento computacional; dicha temática, además de ser una de las más interesantes para ellos, permitió que los estudiantes lograran trasladar a otros contextos lo aprendido más allá del ámbito académico que compete al desarrollo de software, además consiguieron ser capaces de analizar un problema dado, definir los requisitos funcionales y no funcionales de este y por último plantear una solución. Cabe anotar que al inicio del proceso la mayoría desconocía la temática, de ahí la importancia de estos resultados.

En cuanto a algoritmia, al analizar las respuestas consignadas en el cuestionario de bloc de notas se obtuvo que, el 46% de los participantes manifestaron no tener conocimientos acerca de resolver problemas a través de algoritmos o tener conocimientos básicos. Al finalizar en las encuestas y cuestionarios manifestaron en su mayoría que no conocían Scratch y era la primera vez que trabajaban con él, luego reconocieron las características que este tipo de aplicaciones aporta en su proceso formativo.

Una de las dificultades que se presentaron al momento de resolver algoritmos estuvo determinada por el uso de variables; sin embargo, gracias a la facilidad de Scratch lo lograron comprender, tal como lo manifestaron los participantes en las entrevistas y cuestionarios.

La abstracción es una de las habilidades de mayor relevancia para el pensamiento computacional, se pudo extraer del diario de campo que al inicio del proceso los estudiantes presentaron dificultad para reconocer, identificar e implementar patrones y al finalizar la sesión construyeron sus propios patrones, además de resolver problemas que implicaron su uso e implementación. Además, lograron definir patrones correctamente.

En cuanto a la temática correspondiente a planeación, ninguno de los grupos había implementado para su proyecto integrador una herramienta que les permitiera hacer el seguimiento de las tareas, tampoco tenían conocimiento sobre ellas, al finalizar la sesión lograron hacer uso de la herramienta Planner en donde plasmaron e hicieron seguimiento de las tareas a realizar para su proyecto. En este mismo aspecto los estudiantes conocieron nuevas herramientas que permiten hacer seguimiento a las tareas que requiere un proyecto de desarrollo de software.

En lo referente a prototipos la mayoría de los estudiantes manifestaron desconocimiento total acerca del tema, durante la sesión se familiarizaron con el proceso de construcción de un prototipo llegando al punto de realizar los propios ejemplares haciendo uso de una metodología adecuada, que consiste en primero diseñar a mano alzada por medio de sketch, luego realizar el wireframe, pasar al mockup y finalizar con el prototipo; además los estudiantes socializaron los prototipos y retroalimentaron a sus compañeros.

En cuanto a usabilidad al iniciar la sesión correspondiente a esta temática, se evidencia en el diario de campo, que solo uno de los estudiantes tenía conocimientos previos al respecto, los demás participantes manifestaron que para ellos era un tema nuevo. Al finalizar la sesión, los participantes logran reconocer la importancia de la usabilidad para el desarrollo de productos software de calidad.

En esta fase se evidencia una mejora significativa en cada uno de los contenidos asociados a pensamiento computacional por parte de los estudiantes, cabe anotar que a pesar de ser un curso que no representaba ningún tipo de obligatoriedad en su proceso formativo, siempre los estudiantes tuvieron una actitud proactiva y participativa, a pesar de las dificultades que conlleva pasar de una formación presencial a virtual.

Conclusiones

1. Los estudiantes que inician su proceso formativo en Tecnología en desarrollo de software presentan pocos o nulos conocimientos informáticos y más específicamente en pensamiento computacional a diferencia de otros países del mundo donde inclusive han contemplado incluir contenidos asociados al pensamiento computacional dentro de los currículos. Conforme con Acevedo (2018) a lo largo de todo el mundo son diversas las investigaciones que se han realizado en el campo del pensamiento computacional donde es unánime la conclusión de que el aprendizaje de la programación de computadores es una de las herramientas más eficaces que se poseen en la actualidad para este propósito. Es así como esta temática resulta un gran aporte para todos los programas de formación, pero con mayor relevancia en los que tienen que ver con desarrollo de software.
2. Es posible mejorar las habilidades de pensamiento computacional en estudiantes de tecnología en desarrollo de software mediante proyectos como Thinking Challenge o similares, que presenten una estructura pedagógica y didáctica que se ajuste a diferentes niveles de formación. Es en esa línea se proyecta la implementación de las ciencias de la computación que plantea K-12 (2016), en tanto, es posible comenzar desde los primeros grados y continuando hasta el máximo nivel de formación, donde los estudiantes desarrollarán una base de conocimiento de las ciencias de la computación y aprenderán

nuevos enfoques para la resolución de problemas aprovechando el poder del pensamiento computacional para convertirse en usuarios y creadores de la tecnología de las ciencias de la computación.

3. Resulta un reto mantener la participación de los estudiantes en la modalidad virtual, debido a los distractores como redes sociales y entornos no apropiados (tal como lo señalaron los mismos participantes) para estar dispuestos en las sesiones, de ahí la importancia de planear metodologías y herramientas adecuadas. Conforme a Henao (2002) el diseño de materiales de instrucción para utilizar en la red debe basarse en principios o leyes de la percepción. No obstante, esta norma se viola frecuentemente en el diseño de muchos contenidos en línea, —incluidos los cursos virtuales —, donde se sugiere crear material audiovisual adecuado para evitar la aparición de elementos distractores o que generen dificultad en su comprensión.
4. Es aconsejable que los programas que están vinculados con el desarrollo de software implementen estrategias que propendan al mejoramiento del pensamiento computacional, en los ciclos técnico, tecnólogo y profesional. Esto coincide con lo planteado por Acevedo (2018), en tanto, en diferentes países se observa cómo se vinculan a diferentes proyectos para el desarrollo del pensamiento computacional las universidades, gobiernos y algunas entidades privadas, quienes aportan sus recursos con el fin de estimular a los jóvenes al estudio de la programación y de esta forma desarrollar en ellos las habilidades propias del pensamiento computacional.

Referencias Bibliográficas

1. Acevedo, N. (2018). *Desarrollo del pensamiento computacional mediante Scratch en estudiantes de educación media del municipio de Pamplona* (tesis de pregrado). Universidad de Pamplona, Colombia.
2. Aho, A.V. (2011, January) Computation and Computational Thinking. ACM Ubiquity, 1, 1-8.
3. Change the Equation. (2015, December 7). The hidden half [Blog post]. Recuperado de http://changetheequation.org/blog/hidden-half
4. Cuny, J., Snyder, L., & Wing, J.M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress. Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf
5. Henao, O. (2002). La enseñanza virtual en la educación superior. Recuperado de https://www.fumc.edu.co/wp-content/uploads/resoluciones/arc_914.pdf
6. Hernández, R., Baptista, P., y Fernández, C. (2014). Metodología de la investigación. México: McGraw-Hill Interamericana.
7. K – 12 Marco de las Ciencias de la Computación. (2016). Recuperado de http://www.k12cs.org.
8. Lee, I. (2016). Reclaiming the roots of CT. CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12(1), 3–4. Retrieved from http://www.csteachers.org/resource/resmgr/Voice/csta_voice_03_2016.pdf
9. Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee (2nd ed.). New York, NY: Association for Computing Machinery.

10. Valverde, J., Fernández, M., & Garrido, M. (2015). El pensamiento computacional y las nuevas ecologías del aprendizaje. Revista de Educación a Distancia (RED), (46). https://doi.org/10.6018/red/46/3
11. Wing, J. M. (2006). Wing06-ct, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215