Universidad
Rey Juan Carlos

TESIS DOCTORAL

Propuesta de Metodología de Diseño de Juegos Serios para la Enseñanza de Fundamentos de la Programación en Educación Secundaria

Autor:

Hernán Montes León

Directoras:

Diana Pérez Marín

Raquel Hijón Neira

Programa de Doctorado en Tecnologías de la Información y las Comunicaciones

Escuela Internacional de Doctorado

2021

Doctora Doña Diana Pérez Marín, Profesor Titular, Doctor del Departamento de Ciencias de la Computación, Arquitectura de Computadores, Lenguajes y Sistemas Informáticos, Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, como directora de la Tesis Doctoral "Propuesta de Metodología de Diseño de Juegos Serios para la Enseñanza de Fundamentos de la Programación en Educación Secundaria" realizada por el doctorando Montes León Hernán.

HACE CONSTAR

Que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles, a 28 de mayo de 2021

Doctora Doña Diana Pérez Marín

Doctora Doña Raquel Hijón Neira, Profesor Contratado, Doctor del Departamento de Ciencias de la Computación, Arquitectura de Computadores, Lenguajes y Sistemas Informáticos, Estadística e Investigación Operativa de la Universidad Rey Juan Carlos, como directora de la Tesis Doctoral "Propuesta de Metodología de Diseño de Juegos Serios para la Enseñanza de Fundamentos de la Programación en Educación Secundaria" realizada por el doctorando Montes León Hernán.

HACE CONSTAR

Que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles, a 28 de mayo de 2021

Doctora Doña Raquel Hijón Neira

A la memoria de mis padres:

María Sara León Toaquiza

Sergio Manuel Montes Changoluisa

Agradecimientos

Esta tesis ha sido el resultado de un gran esfuerzo y sacrificio, de muchas horas de trabajo y dedicación, sin embargo, no hubiese sido posible llegar a culminar con éxito, si no hubiese tenido el apoyo de diferentes personas que de una u otra manera me brindaron su apoyo durante este proceso.

Agradezco a mis directoras de tesis Diana Pérez y Raquel Hijón, quienes me han guiado y apoyado constantemente en el desarrollo de mi tesis.

A mi hermano Sergio, con quien he compartido muchas cosas. Por su compañía, apoyo incondicional, pensamientos y pautas para este trabajo y, sobre todo, por los momentos compartidos y los que nos quedan por vivir.

En general, a todas las personas que día a día aportaron un granito de arena para el desarrollo de esta investigación.

Índice General

Índice de Gráficos

Índice de Tablas

RESUMEN

Hoy en día los juegos son utilizados para la formación y se los conoce como Juegos Serios. Los juegos serios son utilizados en diferentes áreas como: educación, salud, publicidad, política, negocios, ejército, etc. Desde el punto de vista educativo, se utilizan para motivar a los estudiantes a que realicen actividades que les resultan tediosas o complicadas. Convertir una clase en un juego puede motivar a los alumnos a participar activamente en ella.

En este trabaja se propone la metodología MeDeJuSeEnPro (Metodología para el desarrollo de juegos serios para la enseñanza de programación) que ayudará a crear juegos serios que permita al estudiante adquirir conocimientos de programación informática. Está metodología contiene las siguientes fases: Preproducción, producción, implementación. La metodología propuesta fue validad con la creación de un juego serio "DFD-C" para la enseñanza de programación a estudiantes de educación secundaria. El juego serio DFD-C fue utilizado por 38 alumnos de K-10 de Educación Secundaria durante el curso 2019-2020, en donde se investigó si los estudiantes de educación secundaria mejoraban su aprendizaje de los fundamentos de la programación y si hay diferencias de género en la mejora. Además, se exploró la percepción de la experiencia de juego por parte de los alumnos al utilizar un juego serio como DFD-C, y si esta percepción es diferente en el caso de los chicos o las chicas. Como resultado del experimento del uso del juego serio DFD-C se puede indicar que el uso de juegos serios para enseñar a programar a los estudiantes de Educación Secundaria aumenta significativamente sus resultados de aprendizaje. Esta mejora es similar para alumnos y alumnas, ya que el género no parece ser un factor discriminatorio.

Por lo mencionado anteriormente se puede decir que, el uso de los juegos serios para enseñar a programar a los estudiantes de educación secundaria aumenta significativamente sus resultados de aprendizaje.

ABSTRACT

Nowadays games are used for training and are known as Serious Games. Serious games are used in different areas such as: education, health, advertising, politics, business, military, etc. From an educational point of view, they are used to motivate students to perform activities that they find tedious or complicated. Turning a class into a game can motivate students to actively participate in it.

In this paper it is proposed the methodology MeDeJuSeEnPro (Methodology for the development of serious games for teaching programming) that will help to create serious games that will allow the student to acquire knowledge of computer programming. This methodology contains the following phases: Pre-production, production, implementation. The proposed methodology was validated with the creation of a serious game "DFD-C" for teaching programming to high school students. The DFD-C serious game was used by 38 K-10 Secondary Education students during the academic year 2019-2020, where it was investigated whether High School students improved their learning of programming fundamentals and if there are gender differences in the improvement. In addition, the perception of the gaming experience by the students when using a serious game such as DFD-C was explored, and whether this perception is different for boys or girls. As a result of the experiment of using the DFD-C serious game, it can be indicated that the use of serious games to teach programming to High School students significantly increases their learning outcomes. This improvement is similar for male and female students, since gender does not seem to be a discriminating factor.

From the above it can be said that, the use of serious games to teach programming to Secondary Education students significantly increases their learning outcomes.

Capítulo 1. Introducción

1.1 Contexto

Con la aparición de los primeros ordenadores la única forma de programar era mediante el lenguaje de máquina; el lenguaje de máquina era un programa que se basaba en números binarios (0 y 1), con este lenguaje de programación era muy complejo realizar programas para los ordenadores de ese tiempo. Pero al ir evolucionando los ordenadores de igual manera fueron apareciendo diferentes lenguajes de programación, hasta tener lo que hoy en día conocemos como lenguajes de alto nivel. Los lenguajes de alto nivel son los que nos permiten escribir un código en lenguaje que se aproxima más al lenguaje natural humano.

Durante la última década los ordenadores se han vuelto una herramienta indispensable para las personas, es así que los ordenadores los encontramos en diferentes áreas (medicina, educación, negocios, ejercito, gobierno, etc.), facilitándonos el trabajo cotidiano con diferentes programas desarrollados para ejecutar una tarea específica. Con la finalidad de crear programas que ayuden a la gente, es muy importante adquirir conocimientos para aprender a programar. Así, en las universidades, colegios e incluso escuelas se ha optado por la enseñanza de la programación informática. Siendo su principal objetivo la formación y el desarrollo de habilidades por parte de los alumnos, que les permitan resolver problemas en el ámbito escolar, profesional o en la vida práctica, y teniendo en cuenta los recursos que proporcionan los diferentes lenguajes de programación (Díaz Tejera et al., 2018).

No obstante, los estudiantes a menudo abandonan la informática porque piensan que es confusa y difícil (Papadakis, 2020), debido a que aprender a realizar un programa para ordenadores no es como resolver un ejercicio matemático o una fórmula de física, o únicamente aprender

palabras reservadas de un lenguaje de programación determinado y luego escribir el programa. Por el contrario, aprender a programar es presentar una solución a un problema; cada problema se puede resolver de diferente manera y un programador puede resolver de diferente forma. ¿Es por ello que es difícil el aprendizaje de la programación informática? (Fuentes-Rosado y Moo-Medina, 2017).

Los profesores encargados de enseñar a sus estudiantes programación han utilizado diferentes metodologías para que logren aprender a entender un problema (abstraer, modelar, analizar), plantear soluciones efectivas (reflexionar sobre una abstracción, definir estrategias, seguir un proceso, aplicar una metodología, descomponer en problemas más simples), manejar lenguajes para expresar una solución (codificar, entender y respetar una sintaxis), utilizar herramientas que entiendan esos lenguajes (programar, compilar, ejecutar, depurar), probar que la solución sea válida (entender el concepto de corrección y de prueba), justificar las decisiones tomadas (medir, argumentar) (Zúñiga et al., 2014), entre los principales recursos que se está utilizando para mejorar el aprendizaje de la programación son juegos (puzles, LightBot, etc,), Scratch el cual permite resolver problemas, crear juegos mediante la programación por bloques, uso de robots educativos con los cuales se puede dar órdenes mediante la escritura líneas de código.

Uno de los enfoques para llegar a los estudiantes de forma atractiva es el aprendizaje basado en el juego aprovechando la pasión de los estudiantes por los videojuegos (Morrison y Preston, 2009).

Hoy en día los juegos no sólo son utilizados como entretenimiento de hecho los juegos han invadido áreas como la salud, la educación, los negocios, la política, etc. con el único objetivo de formar o entrenar en dichas áreas, a estos juegos se los conoce con el nombre de “Juegos Serios” (Girard et al., 2013). El juego serio es un concurso mental, jugado con un ordenador de acuerdo

con reglas específicas, que utiliza el entretenimiento para promover los objetivos de formación, educación, salud, política pública y comunicación estratégica del gobierno o de la empresa (Backlund y Hendrix, 2013).

Con la finalidad que el estudiante se sienta motivado en la asignatura de programación se está optando por la enseñanza de la programación mediante la utilización de los juegos con la finalidad de motivar y hacer que sea la materia más atractiva. Con respecto al diseño de los juegos serios para la enseñanza de la programación existe muy poca documentación que guie como desarrollarlos.

1.2 Motivación

Como profesor de programación en Educación Secundaria de K-10 (15 y 16 años) en el colegio "Hermano Miguel" en Ecuador, desde el año 2010 he visto la dificultad que tienen los estudiantes en el aprendizaje de la materia de programación. Muchos de ellos no han culminado sus estudios, pues se han cambiado de especialidad.

El principal inconveniente que han presentado los estudiantes es que no pueden realizar un diagrama de flujo, por ende, si el diagrama de flujo está mal desarrollado la codificación en lenguaje C estará mal. Sin embargo, si se entrega al estudiante el diagrama de flujo el estudiante pasa a lenguaje C sin ningún inconveniente.

En vista de estas dificultades presentadas por los estudiantes año tras año he buscado varios recursos para lograr que el estudiante mejore su rendimiento académico. Entre los principales fueron: en los períodos académicos 2013 -2014 y 2014-2015 utilicé el recurso "Programación con Scratch cuaderno de trabajo para estudiantes" (cuarta edición) (López García, 2011), y la "Guía para docentes sobre Algoritmos y Programación en la Educación Escolar" (López García, 2009),

con los estudiantes de décimo año K-9 (14 años) con la finalidad que se incentiven a seguir la especialidad de Informática y tengan las bases de lo que es la programación de ordenadores. En el período 2016-2017 utilicé la aplicación para dispositivos móviles Funjava (FunJava – Lite-Laboratorio de Tecnologías de la Información en la Educación, s/f), con un grupo de estudiantes de primer año y segundo año de bachillerato K-11 y K-10 (16 y 15 años), se logró que mejoraran en conocimientos de los conceptos teóricos con respecto a la materia (Montes-León et al., 2019). En el período 2017 -2018 con un grupo de estudiantes de primer año de bachillerato K-10 (15 años), utilicé los recursos educativos que consistían en: ejercicios de pensamiento computacional tomados de las competencias internacionales de Bebras (bebras.org, 2017), Ejercicios del Libro "100 problemas matemáticos" (Berabeu Soria, 2017), Ejercicios de las Pruebas de ingreso a la universidad "SER BACHILLER" (Formas - Pruebas ser bachiller, 2017), Juegos Mentales como Juegos de mesa: Damas chinas, tres en raya, juego del molino, Pitarra, esquinas chinas, alquerque, Zorro y cordero, Ajedrez de mesa y ajedrez gigante, Apps: 4 en raya, laberintos, dominó, damas chinas. Como resultado se pudo observar que influyó positivamente en la mejora del aprendizaje de la materia de Fundamentos de Programación en los estudiantes (Montes-León et al., 2020). Es así que en el período académico 2019-2020 me motivó a diseñar y desarrollar un juego en línea para el aprendizaje de fundamentos de programación, el cual fue evaluado con un grupo de estudiantes de primer año de bachillerato K-10 (15 años), los resultados de esta evaluación fueron que los estudiantes aumentaron sus resultados de aprendizaje (Montes et al., 2021).

En la búsqueda de recursos con la finalidad de que el estudiante logre mejorar su aprendizaje en la materia de programación he encontrado el que más se acopla a los estudiantes de Educación Secundaria son los juegos serios, como se ha demostrado en el período académico 2019-2020. Sin embargo, para realizar un diseño de un juego serio para la enseñanza de

fundamentos de programación en Educación Secundaria no existe una metodología, es por ello que en este estudio se propone una metodología de diseño de juegos serios para la enseñanza de fundamentos de la programación en Educación Secundaria.

1.3 Objetivos.

1.3.1 Objetivo General

Proponer una metodología de diseño de juegos serios para la enseñanza de fundamentos de la programación en Educación Secundaria, con la finalidad que cualquier profesor, investigador o educador que desee realizar un juego serio para la enseñanza de la programación tenga una guía para poder diseñar y desarrollar dicho juego.

1.3.2 Objetivos Específicos.

- Identificar los principales problemas que tienen los estudiantes para el aprendizaje de la asignatura de programación informática en Educación Secundaria.
- Definir una metodología a seguir para el diseño de juegos serios para la enseñanza de la asignatura de programación.
- Realizar una validación de la metodología propuesta mediante la creación de un juego serio.

1.4 Estructura.

En el **capítulo uno,** se proporciona la motivación por la cual se realiza esté trabajo, los objetivos a seguir, los cuales permitirán desarrollar la metodología propuesta.

En el **capítulo dos**, se realiza una descripción de los diferentes conceptos teóricos que involucra el desarrollo de la metodología propuesta, problemas de la enseñanza aprendizaje de la

programación, teorías de aprendizaje que pueden influir en el desarrollo cognitivo para la enseñanza de programación informática, juegos serios y su clasificación.

En el **capítulo tres**, se presenta las experiencias realizadas la cuales sirvieron de base para realizar la propuesta metodológica.

En el **capítulo cuatro**, se hace referencia a la propuesta de la metodología, la cual se compone de tres fases: Preproducción, Producción e Implementación. En la fase de preproducción se propone una matriz de contenidos para la enseñanza de la programación informática.

En el **capítulo cinco**, se realiza la validación la metodología propuesta elaborando el juego serio DFD-C y se aplica a estudiantes de primer año de bachillerato de la Unidad Educativa "Hermano Miguel" de Ecuador.

En el **capítulo seis**, se encuentran las conclusiones, además publicaciones que se realizaron para llegar al desarrollo de la tesis doctoral y trabajos futuros a desarrollar.

Capítulo 2. Estado del Arte

2.1. Pensamiento Computacional

La informática en la actualidad ha permitido muchos avances tecnológicos los cuales ha permitido al ser humano beneficiarse de dichos avances, además se encuentra influyendo en nuestra vida cotidiana, como, por ejemplo: el uso de aplicativos móviles para realizar transacciones bancarias, en la medicina para realizar diagnósticos e intervenciones quirúrgicas, en la aviación para control de tráfico, para realizar transferencias bancarias, etc. Es decir, nos encontramos en una sociedad influenciada por la informática. Debido a ello no es necesario que todos los futuros profesionales sean informáticos pero sí que puedan usar recursos de la informática para solucionar problemas de sus áreas como médicos, ingenieros, arquitectos o abogados, etc. con la finalidad de que comprendan cómo usar la computación para solucionar los problemas de sus profesiones y disciplinas (Adell-Segura et al., 2019), a esta enseñanza se le denomina pensamiento computacional.

El término pensamiento computacional apareció por primera vez en el artículo publicado por Wing (2006) en el que indica que el pensamiento computacional implica "la resolución de problemas, el diseño de sistemas y la comprensión del comportamiento humano, recurriendo a los conceptos fundamentales de la informática" (Wing, 2006), sin embargo el pensamiento computacional se remonta al trabajo construccionista realizado por Seymour Papert (1980, 1991), “Mindstorms” teorizó que el aprendizaje y la aplicación de la programación LOGO tendría un efecto en el aprendizaje y los conceptos de conocimiento de los estudiantes en múltiples disciplinas (Papert, 1980, 1996) .

Hasta el momento no existe un consenso para tener un concepto general sobre pensamiento computacional, por lo que a continuación se citarán algunas definiciones realizadas por varios autores.

Wing (2006). - Indica que "El pensamiento computacional implica la resolución de problemas, el diseño de sistemas y la comprensión del comportamiento humano, basándose en los conceptos fundamentales de la informática".

Wing (2008). - Dos años después aclara que "El pensamiento computacional incluye los procesos de pensamiento implicados en la formulación de problemas y de sus soluciones, de tal modo que éstos estén representados de una manera que pueda ser abordada efectivamente por un agente-procesador de información".

La Computer Science Teachers Association y la International Society for Technology in Education (CSTA & ISTE, 2011) desarrollaron una definición operativa en la que indican que "el pensamiento computacional es un proceso de resolución de problemas que incluye (pero no está limitado a) las siguientes características: Formular problemas de una manera que nos permita usar un ordenador y otras herramientas para ayudar a resolverlos; organizar y analizar datos de una manera lógica; representar datos a través de abstracciones tales como modelos y simulaciones; automatizar soluciones mediante el pensamiento algorítmico (una serie de pasos ordenados). identificar, analizar e implementar posibles soluciones con el objetivo de conseguir la combinación más eficaz de pasos y recursos; generalizar y transferir este proceso de resolución de problemas a una amplia variedad de problemas".

Aho (2012). – En su publicación "Computation and computational thinking" señala que el pensamiento computacional es "el proceso de pensamiento involucrado en la formulación de

problemas de tal manera que sus soluciones puedan ser representadas como pasos computacionales discretos y algoritmos".

Royal Society (2012).-Manifiesta que "El pensamiento computacional es el proceso de reconocimiento de los aspectos computables en el mundo que nos rodea, y de aplicar las herramientas y técnicas de las Ciencias de la Computación para comprender y razonar sobre sistemas y procesos, tanto naturales como artificiales".

Román González (2016).- Define al pensamiento computacional como "La capacidad de formular y solucionar problemas apoyándose en los conceptos fundamentales de la computación, y usando la lógica inherente a los lenguajes informáticos de programación: secuencias o direcciones básicas, bucles, condicionales, funciones, y variables".

Barefoot (2021).- Indica que el pensamiento computacional "consiste en aprender a resolver problemas, con o sin computadora. Estas habilidades de resolución de problemas no solo apoyan el plan de estudios de informática en la escuela primaria, sino que también juegan un papel en otras materias, desde matemáticas hasta educación física, ¡e incluso se pueden aplicar en la vida cotidiana!".

De acuerdo a las definiciones mencionadas anteriormente se puede decir que el pensamiento computacional es el proceso que se realiza para resolver un problema de cualquier índole siguiendo diferentes etapas como: el análisis (razonamiento lógico, matemático y abstracto), desarrollo de procedimientos y presentar la solución utilizando herramientas informáticas. (Brennan y Resnick, 2012) mencionan que el pensamiento computacional contiene 6 conceptos (lógica, algoritmos, descomposición, patrones, abstracción, evaluación) y 5 aproximaciones (experimentación, creación, depuración, perseverancia, colaboración)

2.1.1 Conceptos de Pensamiento Computacional

2.1.1.1 Lógica. La lógica nos permite analizar y comprobar los hechos a través de nuestro pensamiento de forma clara y precisa. Ejemplo: para obtener el valor de 4 podemos sumar 2 más 2, o 3 más 1.

2.1.1.2 Algoritmos. Un algoritmo es una serie de pasos de forma secuencial para realizar una tarea. Ejemplo: "Pasos para retirar dinero de un cajero".

2.1.1.3 Descomposición. Cuando existen problemas complejos la descomposición nos permite dividir en partes más pequeñas con la finalidad de poder entender y resolver sin ninguna dificultad. Ejemplo: "En una fábrica de calzado se desea saber cuánto de dinero se paga en sueldo de empleados al año. Para ello se debe calcular el sueldo mensual de cada empleado según su cargo, sumar todos los sueldos, el subtotal multiplicar por 12 que corresponde a los meses del año y de esta forma obtenemos el total de dinero que se paga por salarios al año".

2.1.1.4 Patrones. Patrones es el reconocimiento de situaciones anteriores similares, a partir de patrones se puede hacer predicciones, crear reglas y generalizar. Ejemplo:

"Encontrar el número que sigue en la siguiente serie:

5,10,30,120,480

La serie inicia con 5, luego se multiplica por 2, se obtiene 10, el 10 se multiplica por

3 se obtiene 120, y se va aumenta en 1 el número que multiplica al termino obtenido".

2.1.1.5 Abstracción. La abstracción permite identificar los aspectos más importantes e ir descartando aquellos aspectos irrelevantes. Ejemplo: "Juego Adivina la figura geométrica.

Se presenta a un grupo de estudiantes varias figuras geométricas de diferentes tipos tamaños y colores, un estudiante selecciona una figura y el resto debe adivinar la figura que escogió, para ello el estudiante debe dar pistas que permita a sus compañeros dar con la figura seleccionada. En el caso que haya seleccionado un cuadrado, las pistas relevantes o importantes sería: tiene cuatro lados, sus lados son opuestos y paralelos 2 a 2, todos los lados tienen la misma medida. Los datos irrelevantes en este caso podrían ser: es de color rojo, es pequeño. Sus líneas están trazadas con lápiz".

2.1.1.6 Evaluación. La evaluación es el proceso de verificar si la solución a la que se ha llegado es una solución adecuada o correcta. Ejemplo: "Se desea adquirir una computadora para recibir clases en línea. La computadora puede ser de escritorio o portátil, se puede adquirir de diferentes marcas, diferentes tecnologías, precios etc. La adquisición de la computadora dependería de lo que el usuario considere más importante".

2.1.2 Aproximaciones al Pensamiento Computacional

2.1.2.1 Experimentación. La experimentación es una forma de aprender, pues se puede cambiar el orden de los procesos, aumentar procedimientos o a su vez cuando se tiene algo nuevo se prueban todas las funciones para ver para ver qué realizan.

2.1.2.2 Creación. La creación es un proceso creativo el cual implica originalidad para la elaboración de un producto. Los productos creados a base del pensamiento computacional pueden ser programas o algún tipo de componente de hardware.

2.1.2.3 Depuración. Cuando desarrollamos un programa (escribimos código) siempre se va a generar errores, a la corrección de esos errores se los denomina depuración. La depuración

por lo general demanda más tiempo del consumido escribiendo el código pero, mediante la depuración, nos aseguramos de tener un programa cien por cien funcional y eficaz

2.1.2.4 Perseverancia. Aprender a programar siempre ha sido una tarea muy difícil, debido se debe tener una comprensión algorítmica para resolver el problema y luego desarrollar la solución(programa), esto se realiza escribiendo un código elegante y eficaz, por lo que se debe ser perseverante en las tareas difíciles.

2.1.2.5 Colaboración. Con la finalidad de obtener mejores resultados en el desarrollo de la solución se trabaja en equipo(parejas) como una forma particularmente eficaz de escribir código.

2.2 Recursos Sobre Pensamiento Computacional

Entre los principales recursos que se pueden encontrar son las actividades que se realizan sin utilizar el computador (Unplugged), actividades que se desarrollan mediante el uso de ordenador y kits de robótica. Dichos recursos se detallan a continuación.

2.2.1 Actividades Desconectadas

2.2.1.1 Proyecto CS Unplugged (Computer Science Education Research Group, 2017). "Este proyecto tiene como objetivo principal el de promocionar la Informática entre los jóvenes como una disciplina interesante, fascinante e intelectualmente estimulante; para ello se basa en una serie actividades de aprendizaje de forma gratuita, estas actividades son a base de juegos, puzles con la ayuda de cartas, cuerdas, lápices de colores y mucha actividad física".

Las actividades permiten explorar mapas, problemas de ordenación, problemas de patrones y criptografías, además permite que los estudiantes desarrollen la creatividad y se involucren de forma activa en la solución de problemas.

2.2.1.2 Bebras. Es una iniciativa internacional que tiene como objetivo promover la informática (informática o computación) y el pensamiento computacional entre los estudiantes de todas las edades (bebras.org, 2017).

2.2.2 Actividades Mediante el Uso del Computador

2.2.2.1 Lighbot (LightBot Inc, 2018), es un juego de rompecabezas basado en la programación; mientras el niño juega (Figura 1), secretamente le enseña lógica de programación (Secuencia, Sobrecarga, Procedimientos, Bucles recursivos).

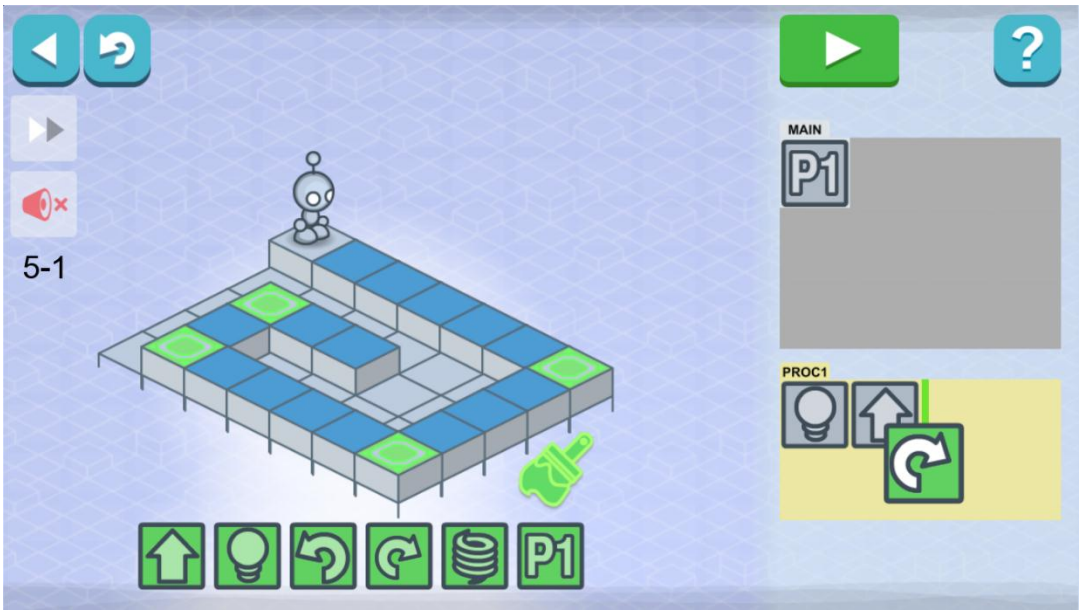

Figura 1: Lighbot - Juego uso de procedimientos
Autor: LightBot Inc

2.2.2.2 The Foss (codeSpark, 2017). Es un juego para los niños más pequeños, el cual introduce conceptos de programación (Figura 2). El juego se encuentra de forma gratuita para Android e iOS, aunque también puede accederse desde la web.

2.2.2.3 Robot School (Robot School, 2017). Es un juego de programación adecuado para niños a partir de 7 años (Figura 3), en el cual aprenderá procedimientos, bucles e instrucciones condicionales, en la resolución creativa de problemas.

2.2.2.4 Scratch (Massachusetts Institute of Technology, 2017). Permite crear historias interactivas, juegos, y animaciones (Figura 4), así como también compartir las creaciones con otros miembros de la comunidad en línea.

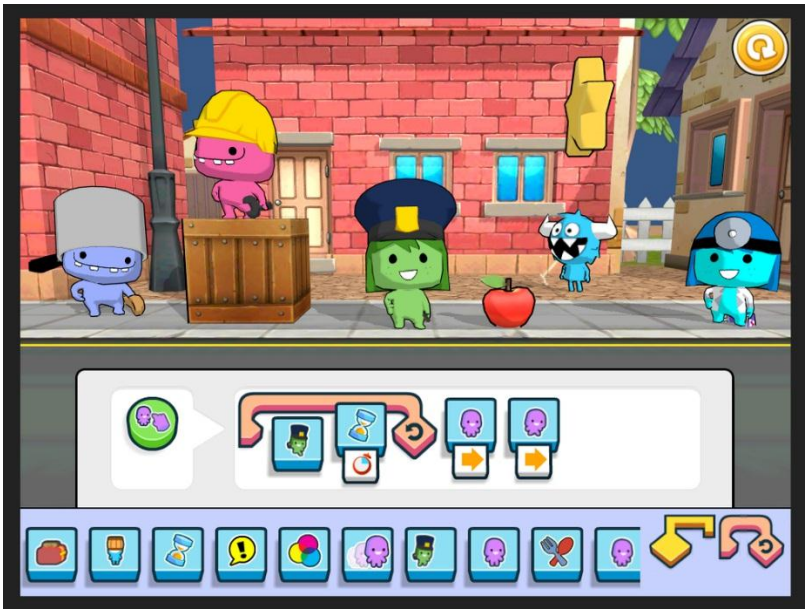

Figura 2: The Foss - Captura de pantalla a través de CodeSpark
Autor: TIM NEWCOMB

Figura 3: Robot School

Autor: EducationalAppStore

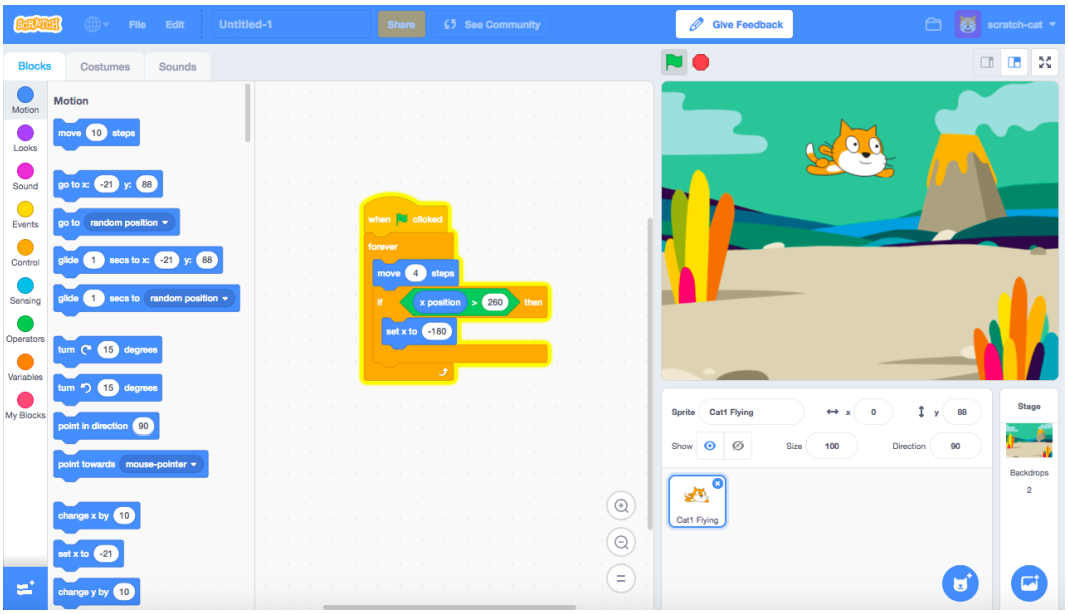

Figura 4: Scratch-Pantalla creación de un juego

Autor: Funside School Ltd.

2.2.2.5 The Hour of Code (Hour of Code, 2017). La Hora del Código es una introducción a las Ciencias de la Computación de una hora de duración, diseñada para mostrar que todo el mundo puede aprender a programar y así comprender los fundamentos básicos de la disciplina.

2.2.3 Kits de Robótica

Con la finalidad despertar la curiosidad de los alumnos y desarrollar su capacidad de resolución de problemas, su creatividad y su espíritu innovador (Bocconi et al., 2016), existen Kits de robótica como:

2.2.3.1 Lego Mindstorms: son robots programables basados bloques de construcción de Lego (Figura 5). Cada versión del sistema incluye un bloque Lego de computadora que controla el sistema, un conjunto de sensores y motores modulares, y piezas de Lego de la línea Technic para crear los sistemas mecánicos.

Figura 5: Kit Lego Mindstorms EV3 Education
Autor: Génération Robots

2.2.3.2 Bee-Bot: es una tortuga Logo simplificada diseñada para estudiantes de jardín de infantes a segundo grado (Figura. 6), Bee-Bot puede programarse para avanzar y retroceder y girar a la izquierda y a la derecha presionando las teclas de flecha correspondientes en su parte superior (Terrapin, 2021).

Figura 6: Niño observando el movimiento del bee bot.
Autor: EDUCACIÓN 3.0

2.2.3.3 Ratón Robot Code & Go: es un robot programable que permite una primera iniciación lúdica a la programación a partir de los 5 años (Figura 7), crear una línea de programación visual con la ayuda de 30 cartas "acción" y reproducir esta secuencia sobre el ratón presionando las teclas direccionales (Hop´Toys, 2021).

Figura 7: Set de actividades de ratón robot "Code & Go"

Autor: Amazon.com

2.3 Enseñanza de la Programación en las Escuelas

El uso de los computadores y las nuevas tecnologías están desempeñando un papel fundamental en nuestra sociedad, debido a que se encuentran presentes en diferentes áreas logrando una mayor eficacia y eficiencia mediante la optimización de recursos y el incremento de la productividad en dichas áreas. Debido a que la demanda de profesionales informáticos cualificados aumenta continuamente. La disciplina informática necesita, en consecuencia, atraer a estudiantes de calidad y educarlos para que sean profesionales capaces y responsables en el siglo XXI (Petri y von Wangenheim, 2017). Es por ello que el aprendizaje de la programación hoy en día ya no se está enseñando únicamente en las universidades, sino que también se está enseñando en escuelas y colegios.

A continuación, veremos los países que mediante el gobierno han tomado como iniciativa y han introducido la enseñanza de fundamentos de programación en la escuela:

2.3.1 América

2.3.1.1 Canadá. Debido a que existen diferentes tipos de gobierno teniendo en cuenta las realidades geográficas de Canadá, la diversidad de sus comunidades culturales y su doble patrimonio jurídico y lingüístico, los gobiernos provinciales y territoriales son los que tienen la competencia en materia de educación. Por ello que cada provincia determina el currículo escolar.

En el año 2016 el gobierno de Columbia Británica anunció que la programación sería obligatoria en sus escuelas. En agosto del 2017 Nueva Escocia, Nuevo Brunswick y Columbia Británica indicaron que la programación ya era parte obligatoria en su plan de estudios hasta sexto grado. Las provincias de Ontario y Saskatchewan han incluido la programación como una parte opcional del plan de estudios en diversos grados (Global News, 2021).

2.3.1.2 Argentina. En el año 2018 la resolución 343/18 del CFE estableció que debe enseñarse programación y robótica en todos los niveles de las escuelas argentinas de forma obligatoria. Algunas provincias ya las incorporaron al currículo, pero aún faltan muchas. Según la norma, tenían tiempo para hacerlo hasta 2020. La implementación de la enseñanza de programación ha obtenido resultados positivos sin embargo aún falta aplicarla en todo el país (TrendTIC, 2021).

2.3.1.3 Chile. En el balance de gestión integral año 2018 presentado y en cumplimiento de los compromisos de gobierno en materia de educación escolar, en 2018 la Subsecretaría de Educación presenta los avances y logros, entre ellos fue la inauguración del Centro de Innovación el cual fue creado con la finalidad de acompañar a los establecimientos educacionales en la

implementación de nuevas prácticas pedagógicas. Como es el Plan Nacional de Lenguas Digitales, que entró en vigor a partir del año 2020 (Ministerio de Educación, 2019). El Plan Nacional de Lenguajes Digitales es una iniciativa que desarrolla el Centro de Innovación, la cual es el resultado de una alianza público-privada en la que participan distintas organizaciones, realizando capacitaciones a docentes y/o profesionales de los establecimientos educacionales de distintas asignaturas para introducirlos en los conceptos del pensamiento computacional y la programación. Esta iniciativa es parte del programa de gobierno y de las medidas del Ministerio de Educación anunciadas en el plan "Chile Aprende Más" (Gobierno de Chile, 2021).

2.3.1.4 Costa Rica. En los años 80 se materializa la idea denominada Programa Nacional de Informática Educativa (PRONIE MEP-FOD), esta iniciativa fue liderada por el Ministerio de Educación Pública, en alianza con la Fundación Omar Dengo(Fundación Omar Dengo, 2021). Este programa se transformó en un pilar fundamental para el desarrollo del país, debido a que está inspirado en la visión de que las tecnologías pueden contribuir al desarrollo cognitivo de los estudiantes, en particular, que la programación puede ser un poderoso medio para desarrollar la capacidad de resolución de problemas y razonamiento lógico en los niños. Hoy la mayor parte de las escuelas primarias y secundarias urbanas de este país cuentan con un curso de informática educativa con clases semanales en el laboratorio de computación (Jara y Hepp, 2016).

2.3.2 Europa

2.3.2.1 Austria. En las planificaciones escolares de educación primaria no existe la materia de informática, pero los niños deben adquirir la alfabetización digital y aprender el manejo razonable de las nuevas tecnologías. Estos conceptos no se ofrecen explícitamente, sino se encuentran ocultos en varias asignaturas (Sabitzer et al., 2014).

2.3.2.2 Eslovaquia. La enseñanza de la informática se inició en 1985 en el segundo ciclo de secundaria, luego en el año 2005 se establece en el primer ciclo de secundaria y a partir del año 2008 se establece la informática como asignatura obligatoria para los alumnos de primaria (Kalas, 2015).

2.3.2.3 Finlandia. A partir del 2016 la programación es una actividad transversal obligatoria desde de primer año de escolarización. La programación se ha integrado en el plan de estudios básico nacional como parte tanto de la competencia transversal en TIC como de la asignatura de estudios de matemáticas y manualidades. El énfasis está en desarrollar habilidades de pensamiento computacional; los alumnos finlandeses aprenden programación en entornos adecuados para su edad. En los grados 1-2 realizan actividades de juegos desconectados, en los grados 3-6 realizan programación gráfica, en los grados 7-9 crean programas utilizando su pensamiento algorítmico con un lenguaje de programación textual (Code School Finland, 2021).

2.3.2.4 Francia. El 31 de marzo de 2015 el Ministro de Educación emitió un decreto sobre la educación obligatoria en el que constan 5 áreas formativas, en las que constan:

Lenguas para pensar y comunicarse: en esta área consta como objetivo el aprendizaje de los lenguajes de informáticos.

Los métodos y herramientas para el aprendizaje: en esta área consta como objetivo una enseñanza explícita de los medios de acceso a la información y documentación, las herramientas digitales.

Este decreto entró en vigor a partir del inicio del curso escolar 2016 (Le Ministère de l'Éducation Nationale, de la Jeunesse et des Sports, 2015).

2.3.2.5 Hungría. En 1995 se introdujo el plan de estudios nacionales el cual incluía la enseñanza formal de las Ciencias de la Computación/Informática con la finalidad de proporcionar al estudiante habilidades del pensamiento computacional y algorítmicas (Biró et al., 2015). El 29 de diciembre del 2015 el gobierno decidió emprender el programa de éxito digital el cual fue aprobado mediante una consulta popular, por lo que solicita al comisario del primer ministro encargado de la coordinación y ejecución de las tareas gubernamentales relacionadas con el programa de éxito digital que prepare y presente al Gobierno la estrategia de educación digital en cooperación con el Ministro de Capacidades Humanas. De acuerdo con esa petición, elaboraron y prepararon el sistema de educación y formación para llevar a cabo tareas de educación, instrucción y formación de acuerdo con las necesidades de la sociedad y la economía digitales en términos de infraestructura, tecnología, contenido, organización del trabajo y recursos humanos (Deutsch, 2016).

2.3.2.6 Italia. A partir del 2014-2015 se inició el proyecto "Programma il Futuro". Su objetivo fue difundir entre los profesores de las escuelas primarias y secundarias italianas una conciencia de la informática como base científica de las tecnologías digitales (Corradini et al., 2017).

2.3.2.7 Noruega. En febrero de 2016, la dirección de educación y formación de Noruega anunció el plan piloto de tres años para la programación en la Escuela Secundaria. El plan se inició en el mes de agosto del mismo año con el objetivo de: 1) aumentar la competencia y el interés de los alumnos por la tecnología, 2) crear experiencia con la enseñanza de la programación en los grados 8 a 10, y 3) aumentar la competencia sobre la programación entre los profesores (Corneliussen y Tveranger, 2018). Para el 2020 el gobierno preveía tener un nuevo plan de estudios

escolar y se centrará en las aplicaciones tecnológicas, la programación y el pensamiento algorítmico (SIMULA, 2018).

2.3.2.8 Polonia. A finales del 2008 fueron aprobadas asignaturas independientes de informática las cuales fueron introducidas en las escuelas primarias y secundarias. Las actividades informáticas debían estar totalmente integradas con otras actividades como la lectura, la escritura, el cálculo, el dibujo, el juego, etc. El nuevo plan de estudios de ciencias de la computación ha sido aceptado por el Ministerio de Educación Nacional y se incluye en el plan de estudios nacional, lo que necesitó la aprobación del Parlamento, en 2016 (Sysło y Kwiatkowska, 2015).

2.3.2.9 Portugal. En 2015 se elaboró la iniciativa "Iniciación a la Programación en Primaria" y fue implementado en el curso escolar 2015-2016 por el Ministerio de Educación y Ciencia, a través del Equipo ERTE-DGE. Esta iniciativa estuvo dirigida a los alumnos de 3º y 4º de primaria y su objetivo fue promover la enseñanza de la programación en Portugal y aumentar los niveles de alfabetización digital de los estudiantes en las escuelas públicas nacionales (Ramos et al., 2016). Hasta el año 2021 se continúa con este proyecto (Direção-Geral da Educação, s/f).

2.2.2.10 Inglaterra. En el 2014 el departamento de educación implementó el nuevo curriculo cuyo objetivo principal es equipar a los alumnos para usar el pensamiento computacional y la creatividad para entender y cambiar el mundo, así como utilizar la tecnología para crear programas, sistemas y contenidos y convertirse en alfabetizados digitales en un nivel adecuado para el mundo del futuro (Jara y Hepp, 2016).

2.3.3 África, Asia y Oceanía

2.3.3.1 Arabia Saudita. El Ministerio de Educación conjuntamente con Tatweer Company for Educational Services en el 2014, han convertido a Arabia Saudita en una de las primeras

naciones en hacer de la informática una asignatura básica de la Escuela Secundaria (Henne, 2015). Mediante esta iniciativa se enseñó a codificar a 225.000 estudiantes de los cuales la mitad de ellos eran mujeres.

2.3.3.2 Australia. En el 2015 el Ministro de Educación aseguraba que la programación informática del siglo XXI se enseñaría en las escuelas primarias a partir del quinto año, y la programación se enseñaría a partir del séptimo año. Existió un gran impulso para un mayor enfoque en la programación y materias de ciencia, tecnología, ingeniería y matemáticas (STEM) en las escuelas desde el nivel primario. Además el gobierno inyectó 12 millones de dólares en cuatro iniciativas STEM separadas: el desarrollo de un plan de estudios de matemáticas innovador y la introducción de la programación informática (SADLER, 2015).

2.3.3.3 China. El Consejo de Estado publicó una directriz en 2017 que requiere que las escuelas incorporen la programación en los cursos de computación. También alentó el desarrollo de interesantes herramientas de aprendizaje para codificar la educación. La Comisión de Educación Municipal de Chongqing emitió un aviso en 2018 que requiere que los alumnos de la escuela primaria tengan al menos 36 horas de instrucción en programación del tercer al sexto grado, con el mismo requisito para los estudiantes de la escuela intermedia del séptimo al noveno grado (The Straitstimes, 2020). En 2018 el ministerio de educación indicó que 18,63 millones de niños estaban matriculados en educación preescolar. Los estudiantes de Educación Infantil, Primaria y Secundaria son los usuarios principales del plan de estudios de educación de programación actual en la nación asiática digitalmente ambiciosa (Khan, 2020).

2.3.3.4 Corea del sur. A partir del 2017 el gobierno decidió que en las escuelas primarias la enseñanza de programación fuese obligatoria y que a partir del 2018 fuera como un curso optativo en las escuelas de secundaria. El Ministerio de Educación está considerando proporcionar

un período corto de capacitación en programación a los maestros que no se especializaron en ciencias de la computación y contratarlos como instructores de programación (Herald, 2014).

2.3.3.5 Israel. La informática ha sido una asignatura en los institutos israelíes desde mediados de la década de 1970. Sin embargo, en aquella época todavía no era una asignatura científica plenamente aceptada, pocos institutos ofrecían cursos de informática y las universidades no consideraban la informática tan importante como otras ciencias cursadas en el instituto. A finales de los años 80, el Ministerio de Educación reconoció la necesidad de rediseñar y reescribir el plan de estudios de informática de la escuela secundaria (Gal-Ezer y Stephenson, 2014), incorporando al currículum de secundaria la programación en LOGO y el uso de aplicaciones generales ya que anteriormente se programaba en BASIC (Jara y Hepp, 2016). En el 2011 el Ministerio de Educación cambió el currículo para centrarse más en las ciencias, incluida la informática, a partir del grado 7 a 9 (Gal-Ezer y Stephenson, 2014).

2.3.3.6 Japón. En abril de 2016, el primer ministro de Japón, Shinzo Abe, anunció en el Consejo de Competitividad Industrial que Japón haría obligatoria la educación en programación desde la escuela primaria y secundaria, en la que los estudiantes aprenden pensamiento lógico a través de la experiencia de programación en aritmética y ciencias. Por ejemplo, aprenden a crear comandos básicos para controlar las luces LED en la clase de ciencias. Sin embargo, la implementación se dio en abril del 2020. A partir de 2021, los estudiantes de las escuelas secundarias aprenderán dos tipos de programación, "Medición y control" y "Comunicación en red". A partir de 2022, las escuelas secundarias incluirán dos cursos selectivos de “Informática”, cada uno de los cuales requerirá 70 horas de clase (Elodie, 2020).

2.3.3.7 Malasia. La programación se ha introducido en las escuelas de Malasia desde 2016, comenzando con el sexto año de la escuela primaria. En 2017 introdujo dos materias llamadas

Ciencias de la Computación Básica y Diseño y Tecnología, en donde los estudiantes tienen la opción de aprender cualquiera de las materias. En Ciencias de la Computación Básica, los estudiantes aprenderán a codificar en diferentes lenguajes de programación como Scratch, HTML y Python en la resolución de problemas y proyectos (Salleh et al., 2021). El Ministerio de Educación declaró que a los alumnos se les enseñarán algoritmos mediante la asignatura de diseño y tecnología (RBT), incluyendo en dicha materia la inteligencia artificial (IA), la programación informática y la robótica. Esto lo realizarán a partir del 2020 en el cuarto grado de las escuelas (Sharon, 2019).

2.3.3.8 Singapur. En julio del 2019 el Ministro de Comunicaciones e Informática anunció que todos los alumnos de segundo ciclo de primaria deberán someterse a un programa obligatorio de 10 horas de programación. Este programa piloto inició ese mismo año, en el 2020 se amplió a todos los cursos de cuarto a sexto de primaria del país. El objetivo es ayudar a los estudiantes a desarrollar una apreciación del pensamiento computacional y de los conceptos de programación, a través de lecciones sencillas basadas en la programación visual (Tan, 2019).

Como se puede apreciar son muy pocos los países que han decido incorporar la enseñanza de programación en las escuelas mediante una política de estado. Sin embargo, se han creado proyectos en varios países los cuales incentivan a las escuelas a que incorporen una materia optativa para que puedan enseñar a los niños a programar.

2.3.4 Proyectos para la Enseñanza de Programación.

2.3.4.1 Code.org. En el 2013 se lanzó el proyecto Code.org. la cual es una organización sin fines de lucro dedicada a ampliar el acceso a la informática en las escuelas con la finalidad que cada estudiante en cada escuela tenga la oportunidad de aprender ciencias de la computación como

parte de su educación. Code.org, creó la campaña anual “Hour of Code” que ha involucrado a más del 15% de todos los estudiantes del mundo. Los cursos que ofrecen se encuentran disponibles en más de 67 idiomas y se utilizan en más de 180 países (Learn today, build a brighter tomorrow. | Code.org, s/f).

2.3.4.2 Computing at School (CAS). Se fundó en 2008 para establecer la informática como una asignatura fundamental que todo niño debería tener la oportunidad de aprender, desde la escuela primaria en adelante (The Royal Society, 2021). CAS se encuentra formada por profesionales apasionados por la informática que trabajan juntos para apoyar a los profesores y garantizar que todos los niños tengan una educación informática líder en el mundo (Computing At School 2021, s/f).

2.3.4.3 Codelearn. Es una escuela de programación, robótica y pensamiento computacional que prepara a los niños y adolescentes para el mundo que les tocará vivir. Codelearn desarrolló un método que convierte el aprendizaje de la programación en un juego y anima a los niños a ser creadores de tecnología, pues aprenden a programar mientras juegan dentro de una plataforma online, gracias a la cual se encuentran matriculados en los cursos estudiantes de todas las edades en más de 60 países y cuentan con más de 20 centros presenciales en España y otros países (Codelearn, 2021).

2.3.4.4 Logiscool. Inició sus actividades de enseñanza en 2014 en Budapest, Hungría, con la finalidad que todo niño conociera la programación porque todos los trabajos hoy en día se encuentran relacionados con la tecnología. Los estudiantes que se matriculan en los cursos de programación aprenden mientras juegan con una metodología experiencial en la que fomenta la creatividad: cuando programan, crean haciendo lo que realmente les gusta hacer. En los cursos

aprenden, además, el pensamiento lógico y mejoran sus habilidades para resolver problemas; con lo cual pueden mejorar su rendimiento en la escuela (Logiscool Ltd, 2020).

2.4 Metodologías

Hasta el momento no existe una metodología específica para la enseñanza de la programación sin embargo veremos las metodologías más utilizadas.

Amor (2016) en su trabajo de fin de máster menciona que, para la enseñanza de programación, se utiliza el aprendizaje basado en problemas o el aprendizaje orientado a proyectos; también indica que se pueden utilizar videojuegos.

2.4.1 Aprendizaje Basado en Problemas (ABP)

En este método se inicia con un problema o situación planteado por el profesor; luego se formarán grupos de trabajo donde los estudiantes deberán investigar y establecer de forma ordenada y coordinada, las fases que implican la resolución o desarrollo del trabajo en torno al problema o situación. Las principales ventajas al utilizar esta metodología son:

1) El desarrollo del pensamiento crítico y competencias creativas.
2) La mejora de las habilidades de resolución de problemas.
3) El aumento de la motivación del alumno.

La mejor capacidad de transferir conocimientos a nuevas situaciones (Realinfluencers, 2018). Mora en su investigación realizadas “Uso de la metodología del aprendizaje basado en problemas en la enseñanza de la programación” obtuvo como resultado que más del 75% de los estudiantes mejoraron favorablemente con la aplicación de esta metodología. Además, concluye que el objetivo del aprendizaje basado en problemas no se centra en resolver el problema, sino que éste se utiliza para identificar los objetivos de aprendizaje que realizará el alumno de manera

preferentemente grupal. Es decir, el problema sirve para motivar a los alumnos a cubrir los objetivos de aprendizaje (Mora et al., 2018).

2.4.2 Aprendizaje por Proyectos

Este método los estudiantes desarrollan un proyecto en un determinado tiempo en el cual deberán resolver un problema para lo cual deberán planificar, diseñar y realizar de una serie de actividades. Esto lo logran aplicando los conocimientos adquiridos y con el buen uso de los recursos, permite al estudiante desarrollar competencias complejas como el pensamiento crítico, la comunicación, la colaboración y la resolución de problemas (Fortea Bagán, 2019).

Aragón (2018) proporciona a los profesores herramientas y puntos claves para poder enfrentarse a este nuevo desafío. Además, presenta 3 ejemplos: a) impresión 3D que consiste en el diseño e impresión de un juego de lógica, b) robótica el cual consiste en el diseño y construcción de un sistema inteligente de reconocimiento y c) programación de un juego de Escape Room utilizando Scratch. En todos ellos deberá aplicar el aprendizaje basado en proyectos para fomentar la creatividad siguiendo las directrices propuesta.

2.4.3 Clase Invertida

Aguilera-Ruiz et al. (2017) mencionan que el aula invertida o flipped classroom es un método de enseñanza cuyo principal objetivo es que el alumno/a asuma un rol mucho más activo en su proceso de aprendizaje que el que venía asumiendo tradicionalmente" (Berenguer-Albaladejo, 2016). En definitiva, supone una inversión con el método anterior (Wasserman et al., 2017), donde los alumnos y alumnas estudiarán por sí mismos y mismas los conceptos teóricos que el docente les facilite y el tiempo de clase será aprovechado para resolver dudas, realizar prácticas e iniciar debates relevantes con el contenido (Aguilera-Ruiz et al., 2017).

Pattanaphanchai (2019) en su estudio obtuvo como resultado que la clase invertida es muy prometedora para mejorar el rendimiento del aprendizaje de los estudiantes en los cursos de programación, ya que los estudiantes estuvieron de acuerdo en que el aprendizaje en un aula invertida con prácticas en clase y estudio fuera de ella les ayudó a entender los conceptos de programación y a implementar programas a partir de esa comprensión.

2.4.4 Aprendizaje Basado en Juegos

Consiste en tomar algunos principios de los juegos y aplicar en entornos de la vida real con la finalidad de facilitar el "aprendizaje mediante el uso de un juego" (Whitton, 2012). El aprendizaje basado en el juego no se limita a crear juegos para que los alumnos jueguen, sino diseñar actividades de aprendizaje que pueden introducir conceptos de forma incremental y guiar a los usuarios hacia un objetivo final (Pho y Dinscore, 2015). Chang et al. (2020) implementaron un curso de aprendizaje basado en un juego llamado "Programmer Adventure Land", utilizando una estrategia basada en problemas. Se pidió a los estudiantes que jugaran al juego para mejorar sus conocimientos de programación. Los resultados obtenidos fueron que el enfoque de aprendizaje basado en problemas del juego puede mejorar la satisfacción, el disfrute, la motivación y la interfaz de usuario para el curso de aprendizaje de juegos basados en problemas.

2.5 Tecnologías

Gracias a los avances en la tecnología hoy en día encontramos juegos, lenguajes de programación visuales, de texto y kits de robótica con los que pueden aprender a codificar. A continuación, veremos los principales recursos para la enseñanza y aprendizaje de la programación.

2.5.1 Juegos (Software)

2.5.1.1 LightBot. En el apartado 2.2.2.1 ya se explicó sobre este software.

2.5.1.2 The Foos. En el apartado 2.2.2.1 ya se explicó sobre este software. Entre los principales contenidos y habilidades que puede adquirir mientras juega es el pensamiento crítico, secuenciación, bucles, reconocimiento de problemas, condicionales, perseverancia, algoritmos, órdenes y parámetros (Gomes, 2016).

2.5.1.3 Kodable. Es un recurso educativo para niños de 5 años en adelante, en donde aprenden conceptos básicos de programación a través de juegos atractivos, cuenta con 50 niveles de juego con contenidos transversales que ayudan a practicar, de forma específica, secuencias, bucles, variables, condicionales, operaciones algorítmicas, resolución de problemas, habilidades comunicativas, pensamiento crítico, etc. Kodable es multiplataforma y se encuentra disponible Windows, Mac, Android e iOS además tiene una versión web. Los contenidos son gratuitos, aunque hay también una versión de pago (Kodable, 2021).

2.5.1.4 Robot Turtles. Es un juego de mesa para niños de 3 a 8 años. Los niños no lo sabrán, pero mientras juegan, aprenden los fundamentos de la programación (Robot Turtles, 2014).

2.5.2 Lenguaje de Programación (Software)

2.5.2.1 Tynker. Es una nueva plataforma de programación diseñada para que los más pequeños (8 – 12 años) desarrollen sus habilidades, se diviertan y a su vez desarrollan su creatividad mientras programan (Tynker, 2021). Los cursos de programación están diseñados para incentivar a los niños mediante un recorrido por varios niveles en donde crearan juegos, animaciones con dibujos y hasta app móviles (Polo, 2013).

2.5.2.2 Code.org. Sobre este proyecto ya se habló anteriormente en el apartado 2.3.4.1, sin embargo, es importante recalcar que esta iniciativa permite aprender a programar a niños desde los 4 años de edad mediante la utilización de lenguaje de bloques. Además, adquieren otras habilidades como las espaciales, matemáticas, artísticas y más. También ofrece diferentes cursos que enseñan las reglas de código y fórmulas para poder expresar ideas con el lenguaje de bloques. Su ayuda es mediante tutoriales, vídeos y programas gratis (Code.org, 2021).

2.5.2.3 Scratch. Es un lenguaje de programación desarrollado por un grupo de investigadores del Instituto de Tecnología de Massachusetts (MIT), diseñado para que niños entre una edad entre los 8 y los 16 años todo el mundo puedan iniciarse en el mundo de la programación, mediante la creación de historias interactivas, juegos y animaciones; además permite compartir los proyectos realizados con otras personas en su sitio web. Scratch se encuentra disponible de forma gratuita para descargar para los sistemas operativos: Windows, Ubuntu, Mac o también se le puede utilizar como una aplicación web ejecutable desde nuestro navegador (Massachusetts Institute of Technology, 2017).

2.5.2.4 Scratch Jr. Es una adaptación más sencilla del lenguaje de programación Scratch, para que los niños de entre 5 y 7 años lo utilicen de una manera divertida y lúdica. Está aplicación ayuda a los niños mediante la programación por bloques a crear su propias historias, cuentos o juegos interactivos en las que ellos son los protagonistas y de esta forma desarrollan la creatividad ya que diseñan proyectos muy originales en los cuales hacen volar su imaginación para resolver problemas del mundo que nos rodea mediante instrucciones y secuencias y de esta forma desarrollan su pensamiento computacional. Scrarch Jr es una aplicación gratuita, y lo encontramos disponible para Android, IOS (Ipad) y para Chromebook (ScratchJr, 2021).

2.5.2.5 Alice. Es un leguaje se programación basado en bloques que facilita la creación de animaciones, la creación de narrativas interactivas o la programación de juegos sencillos en 3D. Alice está diseñada para enseñar habilidades de pensamiento lógico y computacional, principios fundamentales de programación y para ser una primera exposición a la programación orientada a objetos. En Alice encontramos varios objetos tridimensionales (personas, muebles, vehículos, etc.) con los que se puede crear un mundo virtual y animar los objetos. La interfaz de usuario de Alice permite arrastrar y soltar objetos en el escenario para crear un programa donde las instrucciones corresponden a declaraciones estándar de un lenguaje orientado a objetos. El resultado se puede ver de forma inmediata, y de esta forma permite entender la relación entre el código y el comportamiento de los objetos (Alice 2.X, 2020). Alice es un proyecto de código abierto y se encuentra disponible para los sistemas operativos: Windows, MAC y Linux (Alice 2.X, 2020).

2.5.2.6 Kodu. Está diseñando para que los niños creen juegos mediante un lenguaje de programación visual. Kodu es utilizado para enseñar creatividad, resolución de problemas, narración de historias y programación. Kodu puede ser utilizado tanto por niños pequeños como por adultos sin habilidades de diseño o programación. Kodu se encuentra disponible para el sistema operativo Windows para descargar de forma gratuita (Kodu, 2021).

2.5.2.7 Blockly. Es un lenguaje de programación de bloques visual (Open Source con licencia Apache License 2.0, Blockly, 2021), permitiendo crear programas sencillos o complejos. Blockly viene con una "biblioteca JavaScript" que sirve, básicamente como una ubicación centralizada donde se puede encontrar y acceder fácilmente (Dodge, 2019) los lenguajes pre-escritos (como JavaScript, Python, PHP, Lua y Dark).

2.5.2.8 Python. Es un poderoso lenguaje de programación basado en texto (Python Software Foundation, 2021). Python es una excelente opción para introducir a los niños a la

programación, ya que permite usar buenos diseños de código. También alienta a los usuarios a utilizar el soporte en línea y las amplias bibliotecas de Python (Dodge, 2019). CoderZ (2021) menciona que, al utilizar Python, los niños tendrán muy pocas dificultades para su aprendizaje. Podrán aprender a construir ideas de programación dentro de sus cabezas y luego pasar esas ideas a instrucciones que la máquina pueda interpretar.

2.5.3 Programación de Robots (Hardware y Software)

Los robots educativos ayudan a que los alumnos de cualquier edad se familiaricen y profundicen en el estudio de la robótica y la programación, al tiempo que aprenden otras habilidades cognitivas básicas del pensamiento lógico matemático, que es el pensamiento computacional. Es decir, ayudan a desarrollar el proceso mental que se utiliza para resolver problemas de diversa índole mediante una secuencia de acciones ordenadas. Existen numerosos robots educativos; entre los principales se pueden citar: Makeblock mBot, Robo Wunderkind, OWI 535, Mindstorms EV3.

2.5.3.1 Makeblock mBot. Es un robot con ruedas diseñado para que los niños se inicien en la robótica, la programación y la electrónica. Es fácil de ensamblar y se controla con un software basado en Scratch diseñado para niños. Su compatibilidad con la plataforma Makeblock y sus piezas electrónicas basadas en el ecosistema de código abierto Arduino permiten a los usuarios con más conocimientos crear robots más complejos (Makeblock, 2021).

2.5.3.2 Robo Wunderkind. Es un conjunto de bloques de diferentes colores (cámara, micrófono, sensores de movimiento...) con el cual los niños pueden armar su propio robot, los niños pueden programarlo a través de una app para, entre otras funciones, reaccionar a determinados ruidos, esquivar obstáculos o reproducir música cuando alguien se aproxima. Robo

Wunderkind está diseñado para desarrollar las habilidades cognitivas de los niños y desbloquear su creatividad (Robo Wunderkind, 2021).

2.5.3.3 OWI 535. Es un brazo robótico para que puedan utilizar jóvenes a partir de 13 años. El brazo robótico tiene una gran variedad de movimientos y puede levantar objetos de hasta 100 gramos. Los movimientos pueden ser personalizados gracias a la programación (Moviltronics SAS, 2019).

2.5.3.4 LEGO Mindstorms EV3. Es un set de robótica que cuenta con más 600 elementos LEGO Technic, varios sensores y tres servomotores, con los cuales se pueden armar diferentes robots y programarlos para que puedan moverse, disparar, reptar, etc. (LEGO System A/S, 2021)**.** La programación de los robots se realiza por una interfaz sencilla e intuitiva. Cuenta con dos versiones: Home y Education. Este robot está recomendado para niños de más de 10 años (Iberdrola, 2021).

2.6 Problemática de la Enseñanza Aprendizaje de la Programación

El aprendizaje de la programación siempre ha sido un problema debido a las deserciones y fracasos en el curso de programación (Luxton-Reilly, 2016).

Con la finalidad de tener una mejor idea sobre los principales factores que influyen en la dificultad de la enseñanza aprendizaje de la programación, Cheah (2020) realiza una revisión de la literatura relacionada con los "factores que contribuyen a las dificultades en el aprendizaje de la programación informática" en la que se encontró con un gran número de trabajos y lo clasificó en cuatro grupos que se describen a continuación: las fases de la etapa de programación, las habilidades de resolución de problemas, la pedagogía ineficaz y los rasgos y la actitud personales.

2.6.1 Fases de la Etapa de Programación

La programación se divide en dos fases: resolución de problemas y ejecución (Dale y Weems, 2005). Es necesario completar cada fase para continuar con la siguiente teniendo en cuenta que, para superar cada una de estas fases, se requiere distintos conocimientos y habilidades.

2.6.1.1 Fase de Resolución de Problemas. En esta fase el estudiante tiene que aprender conceptos sobre programación y analizar un problema. Es aquí donde primero deben realizar el análisis y entender el problema para luego dar solución mediante un algoritmo, que consiste en una serie de pasos ordenados que permite realizar cálculos y obtener una solución. Después de implementar el algoritmo se debe verificar la solución mediante la prueba de escritorio con la finalidad de verificar y validar el algoritmo mediante la ejecución de las sentencias (entrada de datos, proceso, condiciones, ciclos repetitivos) para llegar al resultado (salida de datos). Por ello para desarrollar un buen algoritmo es muy importante el razonamiento y pensamiento lógico.

2.6.1.2 Fase de Aplicación. En esta fase se escribe el código (programa) en el lenguaje de programación que se haya seleccionado. Una vez desarrollado el programa se deben realizar pruebas de forma rigurosa utilizando diferentes tipos de datos de entrada y procesamientos repetitivos para garantizar la estabilidad del sistema. Las pruebas se realizan de forma rigurosa utilizando varios tipos de datos de entrada y procesamientos repetitivos con la finalidad de garantizar que el programa realice la tarea o resuelva el problema específico sin ningún error. En el caso de que el programa no realice o no resuelva el problema previsto, se llevará a cabo una fase iterativa de resolución de problemas para seguir afinando y corrigiendo la fase anterior de error (Dale y Weems, 2005). La falta de comprensión en la etapa inicial de escritura del programa, como las variables, las matrices, la recursividad y los bucles, hace que los programadores novatos no suelan escribir funciones y procedimientos eficaces (Luxton-Reilly, 2016; Savage y Piwek, 2019).

Por lo tanto, se podría decir que la falta de análisis en la resolución de problemas y de conocimientos de programación (sintaxis) durante la fase inicial no permitirá pasar a la siguiente fase de implementación y hará que se detenga y se repita la fase iterativa en la fase de resolución de problemas.

2.6.2 Habilidades para la Resolución de Problemas

Entre las principales necesidades de resolución de problemas están el fallo en el diseño del programa, la confusión en la selección de bucles y un modelo mental inexacto.

2.6.2.1 Fallo en el Diseño del Programa. En el trabajo de Tan et al. (2009) que investigan las razones de las dificultades en el aprendizaje de la programación informática se encontró que los alumnos tenían problemas para diseñar un programa que llevase a cabo una determinada tarea. Estos problemas se deben a que los estudiantes tienen dificultades para dividir las funciones en procedimientos individuales y carecen de habilidades para la resolución de problemas. Estos problemas existen desde hace mucho tiempo y se citan en investigaciones anteriores (Bosse y Gerosa, 2017; Luxton-Reilly, 2016; Piwek y Savage, 2020; Savage y Piwek, 2019; Spohrer y Soloway, 1986). Los programadores novatos tienden a fusionar diferentes procesos en un mismo trozo de código cuando deberían implementarse por separado. Una confusión similar ocurre cuando los novatos tienen el conocimiento inadecuado de la sintaxis (Bonar y Soloway, 1985; Robins, 2019).

Ismail et al. (2010) señala que los resultados muestran que la deficiencia en la capacidad de analizar un problema y la falta de habilidades para resolverlo han contribuido a que fracase el diseño de un programa. Estos problemas se deben a que carecen de habilidades, como la comprensión de la lógica en el curso de programación. Sin embargo, estos problemas de agravan

más cuando los estudiantes no son capaces de entender y no dominan bien la sintaxis y estructuras de programación (Qian y Lehman, 2017). Es decir, los estudiantes tienen dificultades para aprender la sintaxis del lenguaje y eliminar los errores del programa, lo que supone un proceso largo y tedioso (Qian y Lehman, 2017; Tan et al., 2009).

2.6.2.2 Confusión en la Selección de Bucle. Según la investigación realizada por Soloway et al. (1983), hay temas de programación que causan mucha confusión en relación con el uso de bucles. Los fallos y errores se encontraron comúnmente en las sentencias condicionales y los bucles en comparación con otros temas de programación (Robins, 2019; Spohrer y Soloway, 1986). Otra confusión que se produce similar a los cambios ocultos de la variable de forma automática es el valor del subíndice de un array (Robins, 2019; Savage y Piwek, 2019). Todos estos procesos requieren que el programador realice el procesamiento de los datos en su modelo mental (Du Boulay, 1986) . El conocimiento disponible, el lenguaje natural y las analogías también podrían ser factores que contribuyan a confundir a los programadores cuando se aplican a la tarea en la programación (Papadakis et al., 2014).

2.6.2.3 Modelo Mental Inexacto. Los resultados fueron concebidos además por investigadores anteriores que se refieren al modelo mental como "máquinas nocionales" (Cañas et al., 1994; Du Boulay, 1986; du Boulay et al., 1981). El objetivo de las "máquinas nocionales" es facilitar una comprensión básica del comportamiento de un programa en ejecución. El error más crítico que puede ocurrir es la proyección de un modelo mental falso, esto lleva a un diseño incorrecto y, de aquí, a un resultado erróneo en el programa final. Los errores pequeños pueden subsanarse sin que afecten el modelo del programa, pero los errores que requieren un ajuste exhaustivo suelen desembocar en cambios conceptuales importantes de gran calado.

2.6.3 Pedagogía Ineficaz

Aquí hay que tener en cuenta todos los factores que influyen en el entorno de aprendizaje como son: los tipos de material didáctico, las estrategias de enseñanza y el contenido del programa de estudios.

2.6.3.1 Material Didáctico. Los métodos tradicionales y los materiales estáticos como libros y diapositivas utilizados para la enseñanza, no parecen ser eficaces. Esto se debe a que la programación informática es una materia que está compuesta por conceptos de naturaleza dinámica. (Gomes y Mendes, 2007; Zhang et al., 2013) manifiestan que los estudiantes no comprenden la naturaleza dinámica del flujo de programas cuando se utilizan materiales estáticos para explicarles los conceptos. Los estudiantes tienen dificultades con las técnicas convencionales, como el pseudocódigo y los diagramas de flujo que se utilizan para explicar el método de resolución de problemas. Esto se debe a que las técnicas de presentación convencionales son adecuadas sólo para enseñar programación estructurada, pero son ineficaces para explicar la naturaleza dinámica de la programación informática y para la enseñanza de programación orientada a objetos (Gomes y Mendes, 2007). En la actualidad la mayor parte de lenguajes de programación se aprende con un enfoque orientado a objetos, lo cual requiere una mayor visualización para que el estudiante tenga una representación mental del problema (Robins, 2019). Esta cuestión se relaciona con las estrategias de enseñanza y los programas de estudios.

2.6.3.2 Estrategias de Enseñanza. En la actualidad los lenguajes de programación se basan en la programación orientada a objetos, por lo que las estrategias de enseñanza contemporáneas dejaron de ser pertinentes e ineficaces para el método de resolución de problemas y la programación. Para la enseñanza de la programación orientada a objetos es necesario usar un material que apoye el modelo mental; los materiales de enseñanza que apoyan el elemento espacial

y de visualización ayudan a los estudiantes a entender el proceso de control y flujo de datos (Savage y Piwek, 2019). Por ello continuar enseñando con el enfoque contemporáneo seguirá lastrando al estudiante en el proceso de aprendizaje y disminuirá aún más su interés por aprender a programar.

2.6.3.3 Programa de Estudios. En la investigación realizada por (Soloway y Spohrer, 2013) se menciona que la mayor parte del material didáctico de introducción a la programación se explica sobre la sintaxis y la semántica de la programación. Esto hace que el nivel de conocimientos que adquiere el estudiante sea bajo. Esta conclusión también es respaldada por los resultados obtenidos por investigadores anteriores (Bosse y Gerosa, 2017; McGill y Volet, 1997; Piwek y Savage, 2020; Savage y Piwek, 2019).

Además, para la enseñanza los lenguajes de programación elegidos para la enseñanza deben ser fáciles de entender para que facilite el aprendizaje en la etapa de introducción. A la hora de seleccionar no se debe atender a que sean populares a nivel empresarial (Gomes y Mendes, 2007).

2.6.4 Rasgos Personales y Actitud

Para que el estudiante tenga éxito en el proceso de aprendizaje debe estar motivado y tener interés por aprender una asignatura. En este sentido se han identificado tres áreas: la transferencia de conocimientos previos, la percepción negativa y la actitud.

2.6.4.1 Transferencia de Conocimientos Previos. Las pocas habilidades que poseen los estudiantes en la etapa inicial para la resolución de problemas hacen que tengan dificultades en entender el enunciado del problema de forma clara, lo cual lleva a que la solución obtenida sea incorrecta. Además, la falta de consolidación de las capacidades de conocimiento previo en las

habilidades de resolución de problemas y en el razonamiento lógico, empeora aún más los problemas durante la etapa inicial de aprendizaje (Bosse y Gerosa, 2017; Savage y Piwek, 2019).

2.6.4.2 Percepción Negativa. La mayoría de los estudiantes que han aprendido antes programación suelen realizar comentarios negativos a los nuevos estudiantes que inicia el aprendizaje de programación. Además, la idea errónea que se forman sobre las dificultades en el aprendizaje de la programación informática durante la etapa inicial disminuye la motivación y la confianza de los estudiantes para sobresalir en la asignatura (Qian y Lehman, 2017). La actitud juega un papel importante entre los estudiantes para motivar y crear una impresión positiva en las asignaturas de programación informática. Es imprescindible tener una actitud positiva, ya que las asignaturas de programación requieren persistencia y un aprendizaje continuo cada cierto tiempo (Robins, 2019). Los estudiantes suelen pedir soluciones inmediatas o rendirse cada vez que se enfrentan a obstáculos. La perspectiva que se inculca a los alumnos es vital, ya que determina su motivación intrínseca. El nivel de motivación se traduce en el esfuerzo de los estudiantes, ya sea para persistir en el estudio o para capitular en el aprendizaje de la programación informática (Qian y Lehman, 2017). Por lo tanto, es necesario garantizar que se cultive una mentalidad positiva en relación con los temas de programación informática desde la etapa inicial de cualquier curso de programación. Además de eso, se necesita un apoyo de aprendizaje continuo para seguir inculcando una impresión positiva. Esto mejorará y desarrollará la motivación intrínseca y la actitud positiva hacia futuras asignaturas de programación (Cheah, 2020; Jain y Sidhu, 2013; Qian y Lehman, 2017; Tai et al., 2003).

2.6.4.3 Actitud. Para que exista una motivación en los estudiantes debe existir una actitud positiva, ya que las asignaturas de programación requieren persistencia y un aprendizaje continuo cada cierto tiempo (Robins, 2019). El nivel de motivación del estudiante se ve reflejado en la

cantidad de esfuerzo que realiza para el aprendizaje de la asignatura de programación (Qian y Lehman, 2017). Por lo general cuando los estudiantes encuentran una dificultad tienen la tentación de rendirse o copiar la solución. Así pues, es necesario que se desarrollen una mentalidad positiva en los temas de programación informática desde la etapa inicial de un curso de programación.

En tal virtud de lo antes citado se puede decir que se debe comprender muy bien las etapas de la programación con el fin de reconocer las dificultades que se presentan en el proceso de la resolución de problemas. Los estudiantes que inicien con la asignatura de programación no se deben dejar llevar por los comentarios negativos de estudiantes que ya pasaron por el estudio de la asignatura de programación. Una vez que inicie la resolución de un problema, el estudiante debe tener en cuenta que debe superar cada etapa para pasar a la siguiente. Es aquí donde se necesita que tenga una actitud positiva y que su motivación sea lo suficientemente elevada como para no abandonar el desarrollo al encontrar alguna dificultad. Además, se debe cambiar el método tradicional de enseñanza basado en textos, libros y diapositivas, por un método de enseñanza-aprendizaje de conceptos de naturaleza dinámica, buscando un lenguaje de programación cuya sintaxis sea fácil de entender y no dejarse llevar por la publicidad o marketing de las empresas que desarrollaron el software.

2.7 Juegos Serios

Los juegos utilizados para la enseñanza son los llamados Serious Games (Girard et al., 2013). Un juego serio es aquel que tiene como finalidad la formación, la salud o la comunicación publicitaria, y no sólo el entretenimiento. Los juegos serios se desarrollan para diferentes áreas como: educación, salud, publicidad, política, negocios, ejército, etc. Desde el punto de vista educativo, se utilizan para motivar a los estudiantes a que realicen actividades que les resultan tediosas o complicadas (Montes-León et al., 2019). Convertir una clase en un juego puede motivar

a los alumnos a que participen. Hay estudios que concluyen que los alumnos que participan en la metodología gamificada ganan en atención, participación y proactividad (Furini, 2016).

2.7.1 Concepto

El término "Juegos Serios" fue utilizado originalmente por Viking Press en 1970 en su libro "Serious Games" refiriéndose a los juegos de mesa que pueden ser utilizados en la educación dentro o fuera del aula. Los juegos se basaban en computadores centrales o utilizaban papel y lápiz (Abt, 1987). En 1973 Jansiewicz publicó el libro "The New Alexandria simulation; a serious game of state and local politics" en el que describía un juego que inventó para enseñar los fundamentos de la política estadounidense (Jansiewicz, 1973). Varios años después, este concepto fue retomado por Mike Zyda, que en 2005 escribió un artículo titulado From Visual Simulation to Virtual Reality to Games, en el que actualizaba la definición de juego serio indicando que es un concurso mental, jugado con un ordenador según reglas específicas, que se utiliza para educar o entrenar objetivos corporativos, de educación, de salud, de políticas públicas y de comunicación estratégica (Zyda, 2005).

2.7.2 Clasificación

Existen trabajos en los que los autores han clasificado los juegos serios según diferentes criterios, tecnología y funciones que realizan. A continuación, se citan las principales clasificaciones.

Herz (1997) en su trabajo "Joystick nation: how videogames ate our quarters, won our hearts, and rewired our minds", describe la evolución de los videos desde que se inventaron los primeros juegos hasta los estudios donde se crean juegos tridimensionales y analiza cómo han

influido en nuestros patrones de pensamiento. Además, explica el por qué se han arraigado diferentes tipos de juegos.

Zyda (2005) en su investigación menciona que el desarrollo de los juegos serios es algo nuevo en el mundo de los videojuegos. Los juegos serios utilizan los principios del entretenimiento, la creatividad y la tecnología para construir juegos que persiguen un objetivo concreto y pueden desarrollarse en todos los ámbitos. Los clasifica en juegos para salud, política pública, comunicación estratégica, defensa, formación y educación.

Michael y Chen (2006) clasificaron a los juegos serios según los mercados: Juegos Militares, Juegos Gubernamentales, Juegos Educativos, Juegos Corporativos, Juegos Sanitarios, Juegos Políticos, Juegos Religiosos, Juegos Artísticos.

Alvarez et al. (2007) proponen cinco categorías: juegos de entrenamiento educativo (Edutainment), juegos publicitarios (Advergaming), juegos Edumarket (educación y marketing), juegos políticos y juegos de formación-simulación.

Elverdam y Aarseth (2007) proponen una clasificación de juegos mediante un modelo tipológico con diecisiete dimensiones: perspectiva virtual, posicionamiento virtual, dinámica del entorno, perspectiva física, posicionamiento físico, representación, teleología, prisa, sincronicidad, control de intervalos, composición, estabilidad, evaluación, desafío, objetivos, mutabilidad, salvabilidad. Estas dimensiones se agrupan en metacategorías como el tiempo y el espacio. Este modelo es abierto, lo que significa que las dimensiones individuales pueden modificarse, añadirse o rechazarse sin comprometer la integridad del modelo en su conjunto, por lo que en el caso de examinar únicamente juegos virtuales se puede prescindir de la metacategoría espacio físico. Este modelo puede utilizarse para clasificar la mecánica del juego de forma detallada.

Michaud y Alvarez (2008) presenta una selección de sectores sociales que emplean significativamente los juegos serios: defensa, enseñanza, y formación, publicidad, información y comunicaciones, salud, cultura, activismo.

Sawyer y Smith (2008) en su investigación "Serious games taxonomy" utilizan dos dimensiones:

- Finalidad: aquí se encuentran los juegos para la salud, advergames, juegos para la formación, juegos para la educación, juegos para la ciencia y la investigación, producción, juegos como trabajo.
- Sector: aquí encontramos: gobierno y ONG, defensa, sanidad, marketing y comunicación, educación, empresa, industria.

Al cruzar las dos dimensiones se obtiene la Tabla 1:

Tabla 1: Taxonomía de Juegos serios (Sawyer y Smith, 2008)

	Juegos para la salud	**Advergames**	**Juegos para la formación**	**Juegos para la educación**	**Juegos para la ciencia y la investigación**	**Producción**	**Juegos como Trabajo**
Gobierno y ONG	Educación para la salud pública y respuesta a víctimas masivas	Juegos políticos	Formación de empleados	Informar al público	Recogida de datos/planificación	Planificación estratégica y política	Diplomacia pública, estudios de opinión
Defensa	Rehabilitación y bienestar	Reclutamiento y propaganda	Formación de soldados/apoyo	Educación escolar	Juegos de guerra / planificación	Planificación de la guerra e investigación de armas	Mando y control
Sanidad	Ciberterapia / Exergaming	Salud pública Política y Concienciación Sensibilización Campañas	Juegos de entrenamiento para profesionales de la salud	Juegos para pacientes Educación del Paciente y Enfermedad Gestión de enfermedades	Visualización y Epidemiología	Biotecnología fabricación y diseño	Salud pública Respuesta Planificación y Logística
Marketing y Comunicación	Tratamiento de la publicidad	Publicidad, marketing con juegos, colocación de productos	Uso del producto	Información sobre el producto	Investigación de opinión	Machinima	Investigación de opinión

Educación	Informar sobre enfermedades/ riesgos	Juegos de temática social	Formar a los profesores / Formar a la mano de obra habilidades	Aprendizaje	Informática y Contratación	Aprendizaje P2P Constructivismo ¿Documental?	Enseñanza a distancia Aprendizaje
Empresa	Información sobre la salud de los empleados y bienestar	Educación y concienciación de los clientes	Formación de los empleados	Educación continua y certificación	Publicidad / visualización	Planificación estratégica	Mando y Control
Industria	Seguridad laboral	Ventas y contratación	Formación de los empleados	Mano de obra Educación	Proceso Optimización Simulación	Nano/Biotecnología Diseño	Mando y Control

Djaouti et al., (2011) propone un modelo Gameplay / Purpose / Scope (G/P/S), este modelo clasifica los juegos en función de sus características relacionadas con “serio" y "juego". Se basa en tres aspectos:

- Jugabilidad: Este aspecto proporciona información sobre la estructura de juego del Serious Game.
- Finalidad: Este aspecto da cuenta las eventuales finalidades aparte del entretenimiento que pretende el diseñador del Serious Game.
- Alcance: Este aspecto sugiere el uso real relacionado con el Serious Game.

Estos tres aspectos utilizados en este modelo pueden utilizarse para construir criterios adecuados para la clasificación de cualquier videojuego.

Popescu y Bellotti (2012) realizan la clasificación siguiente: imaginación, compromiso con el aprendizaje y base de datos; basado en las siguientes dimensiones: descripción / clasificación de juegos serios, análisis de componentes de juegos serios, dominios de aplicación, pedagogía, despliegue y tecnologías, con la dimensión pedagógica dividida en dos partes: marcos teóricos y resultados pedagógicos.

Uskov y Sekar (2014) presentan una propuesta de clasificación basada en el modelo G/P/S de los juegos serios; sin embargo, la amplían en cuanto a las características orientadas a la finalidad de los juegos serios y las aplicaciones de los juegos serios:

- Características de los juegos serios y aplicaciones de los juegos serios orientados a los objetivos: contempla la toma de decisiones, simulación, compartir conocimientos, persuasión, recogida/intercambio/exploración de datos, motivación, formación.

- Características de alcance de los juegos serios y aplicaciones de los juegos serios: contempla ámbito y público.
- Características de los juegos serios y aplicaciones de los juegos serios orientadas a la jugabilidad: Contempla el tipo, objetivo y medios

De Lope y Medina-Medina (2017) presentan una taxonomía específica que facilita su comprensión y aplicación. Dividen los juegos serios en seis bloques agrupando criterios relacionados conceptualmente: Desarrollo (criterios relativos al proceso y la metodología utilizados para crear el juego), Plataforma (aspectos de la plataforma donde se ejecuta el juego), Diseño (aspectos a tener en cuenta durante la definición y el diseño del juego), Uso (atributos del comportamiento dinámico del juego), Usuarios (tipos de jugadores) y Modelo de negocio (formas de distribución).

2.7.3 Componentes

Realizar un juego serio demanda de mucho tiempo y talento ya que existen diferentes formatos y objetivos. Sin embargo la mayoría de los basados en el aprendizaje incluyendo cinco elementos (Gamelearn Team, 2021):

2.7.3.1 Una Historia. - La mayoría de los videojuegos tienen una historia principal, aunque no es imprescindible.

2.7.3.2 Gamificación. – La gamificación es la que anima y motiva a todos los jugadores. La dinámica de juego es un elemento fundamental aquí se incluyen los rankings, las recompensas y los sistemas de puntuación.

2.7.3.3 Feedback Inmediato e Individualizado. – En los juegos serios el jugador interactúa de manera directa con el juego al realizar una acción recibe inmediatamente una

recompensa o un castigo, que es lo que conoce como un feedback inmediato y personalizado. En los juegos más sofisticados los feedbacks indican a los jugadores en qué se han equivocado y con ello se espera a que lo realicen mejor la siguiente ocasión.

2.7.3.4 Simulación. – El juego serio reproduce situaciones de la vida real, esto lo realizan mediante la recreación de ambientes y personajes ficticios, por lo que el jugador se ve inmerso en ese mundo a través del juego.

2.7.3.5 El objetivo. – El elemento principal de un juego serio es de enseñar algo. Los juegos serios también deben tener una finalidad que no sea lúdica y que casi siempre estará relacionada con aspectos educativos o de capacitación.

2.7.4 Metodologías

Nadolski et al. (2008) proponen la Metodología EMERGO la cual permitirá desarrollar juegos serios basados en escenarios para la educación superior. Está metodología imita entornos de la vida real mediante simulaciones que proponen una serie de actividades de aprendizaje en la que se debe tomar decisiones, estrategias para la resolución de problemas y otras habilidades cognitivas. Está metodología comprende 5 casos: idea (análisis), escenario (diseño), desarrollo, entrega y evaluación ya que se basa en la metodología ADDIE (Análisis, Diseño, Desarrollo, Implementación y Evaluación).

Tran et al. (2010) proponen la metodología EDoS (Environment for the Design of Serious Games) que es un entorno de autoría interactivo que se basa en tres modelos:

- Modelo formal de dominio específico de los objetivos pedagógicos a los que apunta un juego serio.

- Un escenario pedagógico que se deriva del modelo IMS-LD (estándar pedagógico formal para diseñar unidades) para describir el contenido pedagógico del juego serio.
- El modelo de tarea CTT (Concurrent Task Trees) utilizado para formalizar el escenario HCI de las pantallas del CE después de organizar su representación de disposición.

La metodología EDos pretende ayudar a los diseñadores de juegos serios con un método para desarrollar de manera más rápida y eficiente un proceso bien organizado.

De Gloria et al. (2014) realizan un análisis de las diferentes metodologías pedagógicas utilizadas en el diseño de los juegos serios:

- Es así que algunos juegos utilizan la teoría constructivista del aprendizaje (Piaget, 1979), en donde el conocimiento se crea mediante la experiencia mientras explora el mundo y desarrolla sus actividades
- Otros juegos se basan en los conceptos de Csikszentmihalyi y Csikzentmihaly (1990) en el que se emplea para medir la participación en un juego educativo el GameFlow (Chen, 2007), que consta de ocho elementos: concentración, desafío, habilidades, control, objetivos claros, retroalimentación, inmersión e interacción social.

Barbosa et al., (2014) menciona que el aprendizaje real se genera mediante el uso de los juegos, pues los jugadores tienen que resolver para poder avanzar en el juego y de esta forma van obteniendo la experiencias y las herramientas necesarias en cada nivel, por lo que la pedagogía utilizada es la pedagogía del aprendizaje basada en problemas, en el que la estructura y la narrativa del juego proporcionan el propósito del aprendizaje y una motivación inmediata para perseguir los conocimientos necesarios. Además, debe incluir problemas significativos para resolver con el propósito de aprender los contenidos. Por lo que para el diseño y desarrollo de un juego serio proponen dos componentes principales: misiones y un conjunto de mecanismos de aprendizaje.

Plass et al. (2015) proponen la metodología del aprendizaje basado en juegos ya que una de las características distintivas de los juegos es la preocupación única de los diseñadores de juegos por la calidad de la experiencia de aprendizaje y, en parte debido a esta preocupación, por el hecho de que los juegos digitales son capaces de involucrar a los estudiantes en un nivel afectivo, conductual, cognitivo y sociocultural de maneras que pocos otros entornos de aprendizaje son capaces de hacer. El modelo consta de tres elementos clave: un reto, una respuesta y una retroalimentación.

Braad et al. (2016) proponen un modelo multicapa que distingue entre acciones y motivos fuera y dentro del juego, así como aspectos del contexto de uso, como la comunidad, las reglas y la cultura. Para el diseño de juegos serios, este modelo constituye una base reciente y completa para vincular las decisiones de diseño con los efectos previstos desde una perspectiva integrada.

2.7.5 Validación de un juego serio.

Los sistemas y servicios se están volviendo cada vez más ludificados y tener una experiencia de los juegos serios es cada vez más importante, por lo que es necesario medir la experiencia de los usuarios mientras utilizan un servicio, ya sea intencionalmente o no.

Eppmann et al. (2018) presentan un instrumento denominado GAMEX para captar las experiencias de los consumidores al involucrarse con aplicaciones gamificadas en contextos no lúdicos, es decir, la experiencia de juego. GAMEX valida seis factores: disfrute, absorción, pensamiento creativo, activación, ausencia de afecto negativo y dominio.

Högberg et al. (2019) diseñaron un instrumento denominado GAMEFULQUEST el cual se puede utilizar para modelar y medir la experiencia de juego de un usuario individual en sistemas

y servicios. GAMEFULQUEST contempla siete dimensiones: realización, desafío, concurso, guiado, inmersión, juguetón, experiencia social.

Capítulo 3. Experimentación

Como docente de la Unidad Educativa "Hermano Miguel" y con el propósito de buscar mejoras al proceso de enseñanza aprendizaje en la asignatura de programación informática en los estudiantes de bachillerato, he realizado experimentación con los juegos serios y actividades de pensamiento computacional con los estudiantes de Educación Secundaria. Estos experimentos han servido de base para la propuesta metodológica que se explica en el capítulo 4. Los experimentos se detallan a continuación:

3.1 Experiencia con Aplicaciones Móviles Basadas en Juegos para el Aprendizaje de la Programación.

3.1.1 Introducción

El uso y aumento de las aplicaciones móviles (Apps) para la enseñanza se ha generalizado en la actualidad, aunque esta actividad se realiza a través de juegos (Ortega et al., 2015). Los videojuegos representan un gran porcentaje del consumo audiovisual (Etxeberría, 2012). Por otro lado, los juegos utilizados para la enseñanza se denominan Serious Games (Girard et al., 2013), que utilizan la metodología de los juegos en diferentes ámbitos: se busca acercar a los usuarios haciéndoles partícipes de un contexto, fomentar la competición e incentivar el aprendizaje mediante la concesión de insignias o premios. Al igual que en los juegos, las empresas (y los ejércitos) utilizan incentivos para motivar a sus empleados (Robson et al., 2015).

Los Serious Games pueden utilizarse en muchos ámbitos, y su capacidad para fomentar la participación puede ser de gran ayuda, por ejemplo, para abordar problemas graves del siglo XXI, como la obesidad infantil, promoviendo hábitos saludables (Garcia-Iruela et al., 2017). Desde el punto de vista educativo, son útiles para motivar a los alumnos a realizar tareas que son tediosas y

complicadas (Hanus y Fox, 2015). Convertir una clase en un juego puede motivar a los alumnos a participar. Hay estudios que concluyen que los alumnos que participan en la metodología gamificada ganan en atención, participación y proactividad (Furini, 2016). Los alumnos tienden a abandonar cuando encuentran obstáculos en su formación, lo que les genera una ansiedad que los lleva a la frustración. Una forma de solucionarlo es mediante el uso de juegos serios, una metodología eficaz en el aprendizaje que hace que los alumnos se concentren y participen evitando la ansiedad (Ibañez et al., 2014) y desarrollando el pensamiento computacional (Burgos et al., 2016), teniendo en cuenta que la educación en los últimos 20 años está enfocada a enseñar a pensar (Nickerson y Smith, 1987), a desarrollar el pensamiento, las habilidades, las destrezas, etc., y no sólo a memorizar (Claro, 2010).

Montes-León et al. (2019) presentan una experiencia de uso de una aplicación para el aprendizaje de habilidades de programación a través de la gamificación, llamada FunJava, desarrollada para jóvenes estudiantes a los que les gusta utilizar los teléfonos inteligentes en su vida cotidiana y probablemente en el aula. En esta experiencia participaron 144 estudiantes de dos cursos de 1º y 2º de bachillerato, y cada curso se dividió aleatoriamente en grupo experimental y control, formando 2 grupos experimental y 2 grupos control de 1º y 2º de bachillerato. Así, del análisis cuantitativo se extrajeron interesantes conclusiones: una medida de la eficacia educativa del uso de la App Serious Games, y una medida de la eficacia de la metodología tradicional.

3.1.2 Objetivo

Es utilizar la aplicación FUNJAVA, una App para smartphones, para el aprendizaje de habilidades de programación a través de un profesor virtual que presenta los contenidos nivelados de un curso de introducción a la programación en Java y ofrece juegos para ponerse a prueba mientras juegan con otros compañeros.

3.1.3 Participantes y Contexto

La experiencia se llevó a cabo en el Colegio 'Hermano Miguel' de la ciudad de Latacunga (Ecuador), del 5 al 16 de junio de 2017, con un total de 144 alumnos de dos cursos de bachillerato técnico con especialidad en administración de sistemas: el primer curso de bachillerato (66 alumnos divididos en dos grupos, con el mismo profesor), y el segundo curso de bachillerato (78 alumnos divididos en dos grupos, con el mismo profesor, material y recursos)

3.1.4 Material Experimental

Los materiales utilizados en la experiencia fueron la aplicación FUNJAVA sobre smartphones o tabletas.

FUNJAVA es una aplicación para aprender Java en cualquier smartphone o tableta con el sistema operativo Android. Cubre una introducción básica al curso de programación, a saber: I. Elementos básicos, II. Instrucciones estructuradas, III. Subprogramación, IV. Recursividad, V. Arrays, VI. Archivos.

En la Figura 8 un profesor/avatar explica los contenidos en voz alta en el pizarrón (arriba); luego los estudiantes juegan un juego trivial con otras personas, donde ponen a prueba su aprendizaje y también aprenden (abajo). Los participantes tiran un dado y luego se le hacen preguntas al azar de cualquiera de las 6 lecciones.

Esta aplicación tiene dos secciones de contenido diferentes, por un lado, la clase teórica (Figura 8 arriba) y por otro lado la sección de concurso (Figura 8 abajo). En "la clase" un agente profesor (llamado "Enrique") explica en voz alta los contenidos de la programación Java, apoyándose en la explicación interactiva de "su pizarra". En la sección de concurso la gamificación

se consigue a través de una dinámica similar a la del juego "Trivial Pursuit": uno o más jugadores deben responder a preguntas relacionadas con la programación en Java. Al lanzar un dado, el usuario recibe preguntas aleatorias de uno de los 6 temas en los que se agrupan las preguntas. El ganador es el que responde correctamente tres preguntas de cada tema.

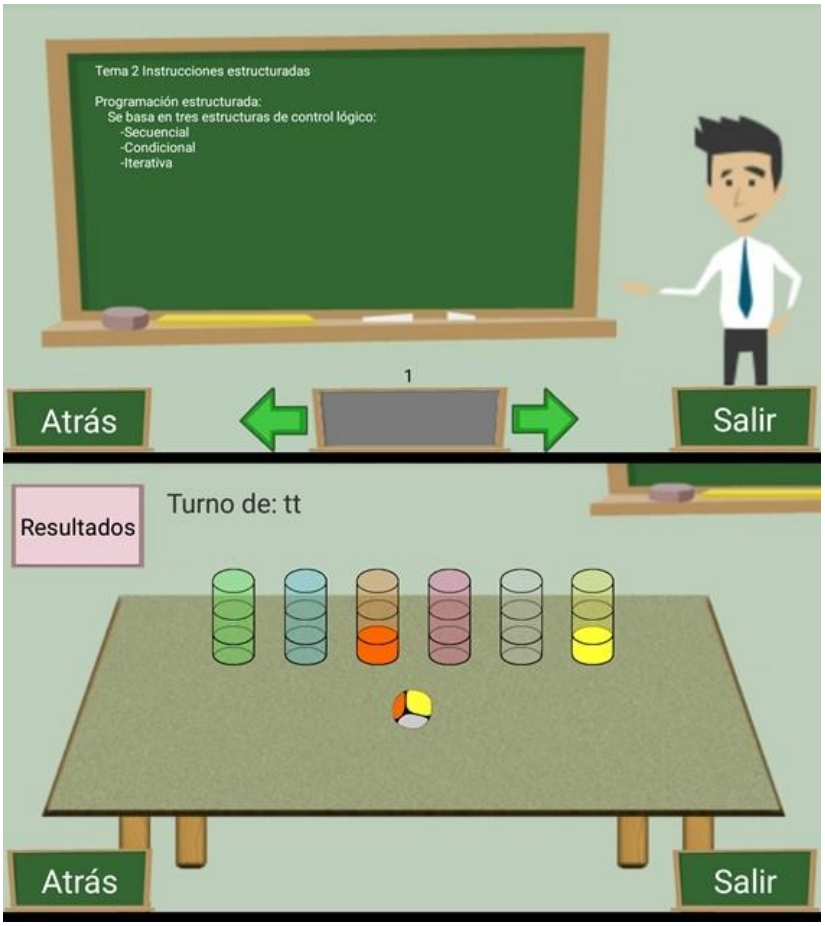

Figura 8: Aplicación FunJava (arriba el agente docente explicando en voz alta con la pizarra, abajo en modo concurso).

3.1.5 Procedimiento

En el primer año de Educación Secundaria, los 66 estudiantes fueron separados aleatoriamente en dos subgrupos, el grupo experimental con 33 estudiantes y el grupo control con

33 estudiantes. De la misma manera, los 78 estudiantes del segundo año de Educación Secundaria fueron separados aleatoriamente en dos subgrupos: un grupo experimental (39 estudiantes) y un grupo control (39 estudiantes). El experimento se llevó a cabo durante dos semanas de junio, con 2 sesiones semanales de 90 minutos cada una, un total de 4 sesiones (360 minutos de entrenamiento). El grupo experimental utilizó la aplicación FUNJAVA, tanto en la revisión de contenidos (1. - Elementos básicos de Java, 2. - Instrucciones estructuradas), como en juegos (modo concurso) y evaluación. El grupo control siguió la metodología tradicional de la clase: construyendo diagramas de flujo y codificando los problemas propuestos, y revisando la documentación teórica.

Durante las dos semanas, los estudiantes del grupo experimental, tanto de primer año como de segundo año, utilizaron la aplicación; la primera semana de las dos sesiones revisaron la primera lección (Elementos básicos de Java), la segunda semana de la primera sesión revisaron el tema 2 (Instrucciones estructuradas), y en la segunda sesión utilizaron la opción del modo Concurso, que consiste en un juego en el que dos jugadores compiten y miden sus conocimientos.

Mientras tanto, en estas dos semanas los alumnos del grupo control, tanto de primer como de segundo curso, procedieron a realizar la clase con normalidad, como es habitual: realizando los DFDs y su respectiva codificación, además de leer la documentación teórica aportada.

Tanto el grupo experimental como el grupo control realizaron la misma prueba de conocimientos de programación antes de comenzar la experimentación (pretest) y al final (posttest).

El grupo control trabajó sin problemas siguiendo una metodología similar. No mostraron un interés particular; sólo escucharon al maestro e hicieron la tarea. Se hizo evidente que los

estudiantes de este grupo generalmente no realizaban tareas voluntarias, sino sólo las requeridas por el profesor.

El grupo experimental, que sólo usaba la aplicación, parecía involucrarse un poco más porque no tenían que escribir nada en absoluto, sino leer o escuchar lo que la aplicación explicaba.

Cuando entraron en el Modo Concurso, los estudiantes se encontraron con preguntas sobre temas que no habían revisado (subprogramas, introducción a la recursividad, matrices y archivos); entonces, se dieron cuenta de que, para participar en el juego y, sobre todo, para tener éxito y ganar, era necesario revisar esos temas. Era un buen incentivo, porque los estudiantes investigaban por su cuenta, sin necesidad de que el profesor los obligara.

3.1.6 Resultados

Como el objetivo de este estudio era evaluar si el uso de la aplicación FUNJAVA mejora o no el aprendizaje, los análisis de los resultados se orientaron a realizar un análisis descriptivo de los datos presentados y, posteriormente, a deducir si existió o no una mejora significativa entre el pre-test y el post-test de la variable evaluada en los dos cursos de bachillerato.

3.1.6.1 Estudiantes de 1ro de Bachillerato (K-10)

Las variables "objeto de estudio" son el PreTest y el PostTest, que reflejan la puntuación total -relativa a 10- obtenida por el alumno en una prueba diseñada por el profesor para evaluar la comprensión del alumno antes y después de utilizar el programa FUNJAVA. En primer lugar, se realizó un análisis descriptivo de los resultados. La Figura 9 muestra un diagrama de cajas para el pre y el post test. Las cajas están delimitadas por los valores Q1 (primer cuartil) y Q3 (tercer cuartil). Cada caja contiene el 50% de los casos correspondientes, destacando la mediana. Los

valores mínimos y máximos en los extremos del diagrama corresponden a valores no inferiores a Q1-1,5 - (Q3-Q1) y no superiores a Q3 + 1,5 (Q3-Q1).

Por lo tanto, se puede observar que los datos del grupo control mantienen una tendencia central en torno al mismo valor de la mediana, que ha sufrido una ligera variación en la fase previa y posterior a la prueba. Sin embargo, la dispersión de los datos es muy grande, aunque disminuye en el postest respecto a la fase anterior. Existe una ligera asimetría en ambos casos y algunos valores atípicos en el caso de la preprueba. Sin embargo, en el caso del grupo experimental, se aprecia un notable aumento de los valores centrales del conjunto de datos. Los valores exactos se discutirán más adelante con referencia a la Tabla 2. La variabilidad de los datos también aumentó, pero fue menor que en el grupo control, con una ligera asimetría en ambos casos. La Tabla 2 muestra los valores de la mediana, la media, la desviación estándar y el coeficiente de asimetría de los dos primeros grupos de muestras. La inclusión de la mediana como valor característico tiene sentido para poblaciones no asimétricas, como era el caso, ya que su valor es más representativo que la media.

Estos valores corroboran las afirmaciones anteriores referidas a la Figura 9, ya que en el grupo control la media presenta una ligera mejora (de 7,65 a 8,07) que es casi imperceptible en el caso de la media. La desviación estándar, baja ligeramente, respecto a su asimetría se puede decir que la mayoría de las valoraciones están por debajo de la media. Por el contrario, el grupo experimental mostró una evidente mejora (de 6,56 a 7,24 en el caso medio), con datos ligeramente dispersos, como se deduce de la ligera disminución de la desviación típica; respecto a su asimetría podemos decir que hay un porcentaje medio de notas por debajo de la media.

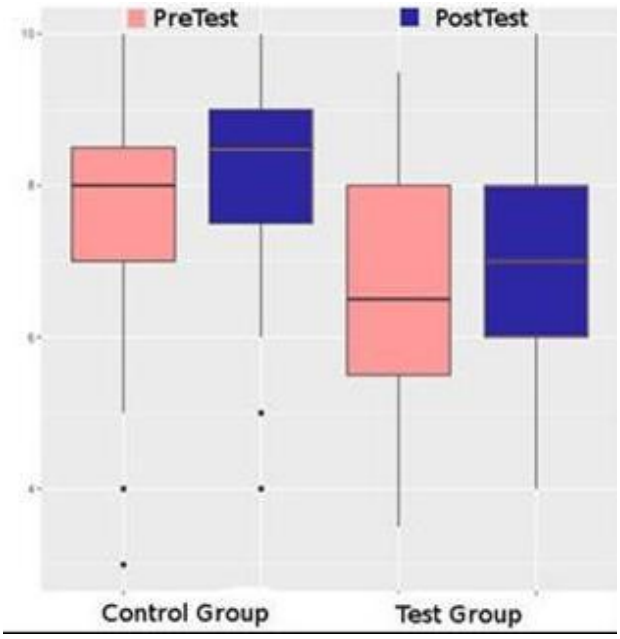

Figura 9: Boxplot 1er curso para el grupo control y experimental fases pre y post test

Tabla 2: Resultados del pre y post test en el grupo control y experimental en el primer año (K-10)

Grupo	Mediana	Promedio	Desviación estándar	Coeficiente de asimetría
Control Pre-test	8	7.65	1.53	-1.12
Control Post-test	8.5	8.07	1.27	-1.23
Test Pre-test	6.5	6.56	1.74	-0.12
Test Post-test	7	7.24	1.36	-0.173

El análisis descriptivo da una primera idea de la distribución de la muestra. Ahora, ofrecemos un estudio más detallado que compara los resultados de las fases pre y post test en ambos grupos para deducir si hubo alguna mejora significativa.

En primer lugar, estudiamos la fase previa a la prueba para determinar si había diferencias entre los grupos experimental y control. Debido al tamaño relativamente grande de la muestra, utilizamos la prueba T-Student para muestras independientes y concluimos que hay diferencias significativas entre ellos, con un significado $p<0,05$. Por lo tanto, podemos suponer que ambos

grupos no son homogéneos en la fase previa a la prueba con respecto a las variables objeto de estudio.

Sin embargo, observamos diferencias significativas entre ambos grupos en la fase post-test, con una significación $p<0,05$. Esto indica que ha habido una mejora significativa; sin embargo, los grupos siguen sin ser homogéneos. Para obtener información adicional sobre la magnitud del cambio en el grupo experimental, se calculó el tamaño del efecto mediante el estadístico d de Cohen. El valor obtenido ($d=0,43$) corresponde a un efecto moderado. Los resultados del primer año de bachillerato indican que el uso de FUNJAVA influyó positivamente en el aprendizaje de los alumnos. Se encontraron diferencias significativas en el grado de valoración de los conocimientos antes y después de utilizar FUNJAVA en el grupo experimental, pero no en el grupo control. Estimamos que la magnitud de esta mejora es destacable, contando con un efecto moderado por su uso en la muestra estudiada, con una mejora en su media.

3.1.6.2 Alumnos de 2do de Bachillerato (K-11)

En el caso de los estudiantes de segundo año se siguió el mismo procedimiento que en el caso de los estudiantes de primer año de secundaria. La Figura 3 muestra un diagrama de cajas para el pre y el postest.

Según la Figura 10, puede decirse que el grupo control mantiene los valores en una tendencia central en relación con el valor medio. La dispersión de los datos es baja, pero disminuye un poco más en el postest, con una ligera asimetría en ambos casos. Los valores exactos pueden verse en la Tabla 3. La variabilidad de los datos también aumentó, pero fue menor que en el grupo control, con una ligera asimetría en ambos casos. La Tabla 3 muestra los valores de la mediana, la

media, la desviación típica y el coeficiente de asimetría para el segundo curso de Educación Secundaria de los dos grupos de la muestra.

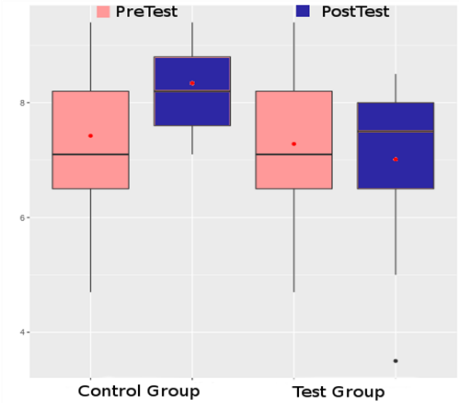

Figura 10: Boxplot 2do curso para el grupo control y la experimental en las fases pre y post test.

Estos valores corroboran las afirmaciones anteriores referidas en la Figura 10, ya que en el grupo de control la media muestra una notable mejora (de 7,42 a 8,43). La desviación estándar del 47% de su asimetría puede decirse que se acerca a la media. Por otro lado, el grupo experimental mostró un ligero descenso (de 7,28 a 7,01 en el caso de la media), con datos ligeramente dispersos, como se deduce del ligero descenso de la desviación típica, respecto a su asimetría. Se puede decir que hay un porcentaje medio de puntuaciones por encima de la media.

Tabla 3: Resultados pre-test y post-test en los grupos control y experimental en el ^{2do} año (K-11)

Grupo	Mediana	Promedio	Desviación estándar	Coeficiente de asimetría
Control Pre-test	7.1	7.42	1.13	-0.13
Control Post-test	8.2	8.43	0.62	0.10
Test Pre-test	7.1	7.28	1.16	-0.02
Test Post-test	7.5	7.01	1.15	-1.14

Se realizó un análisis descriptivo comparando los resultados de las fases pre y post test en los dos grupos para determinar si había o no una mejora significativa.

En primer lugar, estudiamos la fase previa a la prueba para determinar si había diferencias entre los grupos experimental y de control. Debido al gran tamaño de la muestra, utilizamos la prueba T-Student para muestras independientes y concluimos que hay diferencias significativas entre ellos, con una significación $p>0,05$. Por lo tanto, podemos suponer que ambos grupos son homogéneos en la fase previa a la prueba con respecto a las variables objeto de estudio.

Sin embargo, observamos diferencias significativas entre los dos grupos en la fase posterior a la prueba, con una significación de $p <0,05$, lo que indica que ha habido una mejora significativa, y que ahora tenemos grupos no homogéneos.

Para obtener información adicional sobre la magnitud del cambio en el grupo experimental, se calculó el tamaño del efecto mediante el estadístico d de Cohen. El valor obtenido ($d=0,23$) corresponde a un efecto débil.

Los resultados obtenidos para los alumnos de 2º de Bachillerato de este estudio indican que el uso de FUNJAVA no influyó positivamente en el aprendizaje de los alumnos. Sin embargo, los estudiantes se encontraban motivados por el uso de la app.

3.2 Experimentación con Mejora del Pensamiento Computacional en Estudiantes de Secundaria con Tareas Unplugged.

3.2.1 Introducción

Hoy en día la enseñanza de la programación tanto en la Educación Primaria, Secundaria y Universidades es muy importante, en especial en las carreras de Ingeniería ligadas a las tecnologías

(Blake, 2011). Por tal motivo es muy importante que los estudiantes logren desarrollar habilidad de escribir programas correctamente, eficientes, bien organizados y adecuadamente documentados para ser un buen desarrollador (Compañ-Rosique et al., 2015). El aprendizaje de la programación no es como aprender un procedimiento o fórmula, la cual se aplica en un cálculo diferencial, tampoco es memorizarse una lista de fechas importantes y luego repetirlas. Aprender a programar no es solo aprender palabras reservadas de un determinado lenguaje y luego aplicarlo. Aprender a programar es dar solución a un problema, cada problema se soluciona de manera diferente y un programador lo resuelva de diferente forma, es aquí donde se encuentra la dificultad de aprender a programar (Fuentes-Rosado y Moo-Medina, 2017). Estas habilidades se pueden desarrollar con un pensamiento específico, es decir un pensamiento computacional (Zapata-Ros, 2015). El pensamiento computacional y la programación empieza a formar parte del currículo oficial en los sistemas educativos formales (Berrocoso et al., 2015). Al igual que se inicia con proyectos de capacitación para incorporar el pensamiento computacional en las escuelas (Dapozo et al., 2016). En un estudio previo (Montes-León et al., 2019) realizó una experiencia de uso y aplicación para el aprendizaje de habilidades de programación a través de la gamificación, llamada FUNJAVA, desarrollada para los jóvenes estudiantes que disfrutan el uso de teléfonos inteligentes en el aula. La experiencia se realizó con 144 estudiantes de primero y segundo año de Educación Secundaria y cada curso fue dividido al azar en grupos experimental y control, formando 2 grupos de experimentación y 2 grupos de control de primer y segundo año de Educación Secundaria. Así pues, del análisis cuantitativo se extrajeron interesantes conclusiones: una medida de la eficacia educativa del uso de la aplicación de juegos serios y una medida de la eficacia de la metodología tradicional, es decir introducción a la programación donde se resuelven unos pocos problemas matemáticos y luego se procede al desarrollo de diagramas de flujo.

En el presente estudio son 80 estudiantes de primer año de bachillerato divididos en dos grupos (paralelo A y B). El grupo experimental con 40 estudiantes que corresponde al paralelo "A" y el grupo control que corresponde al paralelo "B" con la finalidad de incorporar actividades de pensamiento computacional para la enseñanza- aprendizaje en fundamentos de programación. Entre las principales actividades que se realizaron pueden citarse: ejercicios tomados del concurso internacional de Bebras, ejercicios matemáticos, ejercicios de razonamiento lógico matemático tomado de las pruebas "Ser Bachiller 2017", juegos mentales. Es así que después de un análisis, se logró obtener conclusiones interesantes con respecto a la propuesta de que el pensamiento computacional influye en la mejora del aprendizaje de programación en estudiantes de bachillerato (K-10).

3.2.2 Hipótesis:

En la experimentación se incorporaron actividades de pensamiento computacional para mejorar la enseñanza-aprendizaje de la asignatura de fundamentos de programación de primer año de bachillerato. Se probaron dos hipótesis:

H1: La metodología propuesta desarrolla el pensamiento computacional en estudiantes de bachillerato (K-10).

H2: La metodología propuesta de pensamiento computacional influye en la mejora del aprendizaje de fundamentos de programación en estudiantes de bachillerato (K-10).

3.2.3 Participantes y Contexto:

La experiencia se llevó a cabo en el Colegio Hermano Miguel de la ciudad de Latacunga (Ecuador), del 6 de noviembre del 2017 al 8 de abril del 2018, con un total de 80 alumnos de dos cursos de primer año de bachillerato técnico con especialización en Informática divididos en dos

grupos (A y B) con el mismo profesor. En el grupo A se encuentran 40 alumnos, de ellos 21 son hombres (52.5%) y 19 son mujeres (47.5%). En el grupo B se encuentran otros 40 alumnos, de los cuales 22 son hombres (55%) y 18 son mujeres (45%).

3.2.4 Material Experimental

- Test de pensamiento computacional (Román-González et al., 2015).
- Ejercicios propuestos de la página de Bebras, que son desafíos que promueven habilidades de resolución de problemas y conceptos informáticos, incluida la capacidad de dividir tareas complejas en componentes más simples, diseño de algoritmos, reconocimiento de patrones, generalización de patrones y abstracción (bebras.org, 2017)
- Ejercicios del libro "100 problemas matemáticos" (Berabeu Soria, 2017).
- Ejercicios de analogías, relación entre figuras, secuencias horizontales gráficas, armar cubos de las pruebas de ingreso a la universidad "Ser Bachiller" (Formas - Pruebas ser bachiller, 2017).
- Juegos mentales y juegos de mesa: damas chinas, tres en raya, juego del molino, pitarra, esquinas chinas, alquerque, zorro y cordero, ajedrez de mesa y ajedrez gigante, Apps: 4 en raya, laberintos, dominó, damas chinas

3.2.5 Procedimiento

Está experiencia fue un estudio experimental que se aplicó un grupo de estudiantes que se asignaron de forma aleatoria al grupo experimental y al grupo control. El grupo experimental con 40 estudiantes que corresponde al grupo "A" y el grupo control también con 40 estudiantes, que corresponde al grupo "B". En esta investigación se pretende evaluar, con la metodología propuesta, la ganancia de pensamiento computacional y la mejora del aprendizaje de fundamentos de

programación. El experimento se llevó a cabo durante 5 meses, con 2 sesiones semanales de 90 minutos cada una, un total de 40 sesiones.

La primera semana de 6 al 10 de noviembre el grupo experimental y grupo control completaron el test de pensamiento computacional (Román-González et al., 2015).

Con el grupo experimental se realizaron las actividades como se ve en la Tabla 4 desde el 13 de noviembre del 2017 hasta el 23 de febrero del 2018. Las actividades fueron seleccionadas según los conceptos de pensamiento algorítmico, descomposición, patrones, abstracción y evaluación ya que permiten que los estudiantes tengan creatividad y se involucren de forma activa en la solución de problemas.

Tabla 4: Actividades realizadas con el grupo experimental (K-10)

Actividades	**Descripción**
Ejercicios concurso Internacional de Bebras	Castores en el río, castores y bisontes, falso operación grúa, buscando el tesoro, registro de castores, dibujo de estrellas, fabricación de tazones, familia súper power, etc.
Ejercicios del Libro "100 problemas matemáticos" (Berabeu Soria, 2017)	La habitación de mi casa, blancas y negras, escaleras, el gato y el ratón, el banco, cuadrados mágicos I y II.
Ejercicios de las Pruebas de ingreso a la universidad "SER BACHILLER" (ejercicio3.pdf, ejercicio4.pdf) (Formas - Pruebas ser bachiller, 2017)	Ejercicios analogías, relación entre figuras, secuencias horizontales gráficas, armar cubos
Juegos Mentales	Juegos de mesa: damas chinas, tres en raya, juego del molino, pitarra, esquinas chinas, alquerque, zorro y cordero, ajedrez de mesa y ajedrez gigante. Apps: 4 en raya, laberintos, dominó, damas chinas

Con el grupo control se trabajó desde el 13 de noviembre del 2017 hasta el 23 de febrero del 2018 desarrollando los ejercicios del documento digital de Razonamiento: Lógico, Matemático, Inductivo, Deductivo, Abstracto de Orlando Ayala (Ayala, 2017) de la siguiente manera:

- Razonamiento matemático (suma, resta, multiplicación, regla de tres simple, inversa y compuesta, tanto por ciento) hasta el 19 de enero del 2018.
- A continuación, se trabajó con los estudiantes (Figura 11) con juegos mentales hasta el 23 de febrero del 2018, los juegos de mesa que se realizaron fueron: damas chinas, tres en raya, juego del molino, pitarra, esquinas chinas, alquerque, zorro y cordero, ajedrez de mesa y ajedrez gigante. Además, se trabajó con apps: 4 en raya, laberintos, dominó y damas chinas.

Figura 11: Estudiantes realizando actividades mentales en juegos de mesa en la institución "Hermano Miguel"

Las semanas del 26 al 2 de marzo se aplicó el test de pensamiento computacional tanto al grupo experimental como al grupo control.

Del 5 al 30 de marzo del 2018, con el fin de comprobar si estos resultados obtenidos influyen positivamente en el aprendizaje de la asignatura de fundamentos de programación, se procedió a impartir clase a los dos grupos (control y experimental) con la planificación establecida para ese año académico, con el mismo profesor, los principales temas fueron:

Conceptos básicos, metodología para la solución de problemas por medio de computadora, tipos de datos, expresiones, operadores y operandos, técnicas para la formulación de algoritmos: diagrama de flujo, pseudocódigo, estructuras algorítmicas secuenciales y ejercicios.

Una vez finalizados estos temas, en la semana de 2 al 6 de abril se elaboró y pasó un test sobre la materia de fundamentos de programación: conceptos básicos, metodología para la solución de problemas, operadores lógicos, aritméticos y relacionales. Como práctica se les pidió realizar las siguientes tareas: diseño de un diagrama de flujo de estructura secuencial, prueba de escritorio y corrección de errores de una codificación en lenguaje C.

El test se aplicó tanto al grupo control como al grupo experimental, con la finalidad de evaluar si la metodología explicada previamente influyó positivamente en el aprendizaje de la materia de fundamentos de programación en el grupo experimental.

3.2.6 Resultados

Dado que el objetivo de este estudio era desarrollar el pensamiento computacional en el estudiante de bachillerato (K-10) y evaluar si el empleo de actividades de pensamiento computacional ayuda o no a mejorar el aprendizaje o no de fundamentos de programación, los

análisis de los resultados se orientaron a realizar un análisis descriptivo de los datos presentados y, a continuación, a deducir si había habido o no una mejora significativa de la variable evaluada en primero de bachillerato entre el grupo control y el grupo experimental.

3.2.6.1 Resultados Mejorar en Pensamiento Computacional

En la Figura 12 se muestra un boxplot para el pretest y postest de pensamiento computacional aplicado tanto al grupo control como al grupo experimental. Se puede observar que los resultados del pre-test no tienen valores similares entre el grupo experimental y control, esto posiblemente se debe a que el momento que se aplicó el test a los dos grupos se les indicó que no tiene una calificación, en el grupo control se pudo observar un desinterés muy grande al contestar el pre test debido a que no tenía ninguna calificación, sin embargo, en el grupo experimental se observó, por alguna razón desconocida, una mayor motivación.

Las cajas están delimitadas por los valores Q1 (primer cuartil) y Q3 (tercer cuartil). Cada cuadro contiene el 50% de los casos correspondientes, destacando la mediana. Los valores mínimo y máximo en los extremos del diagrama corresponden a valores que no son inferiores a Q1-1.5 - (Q3-Q1) y no superiores a Q3 + 1.5 (Q3-Q1).

Puede observarse que los datos del grupo control mantienen una tendencia central en torno al mismo valor mediano, que ha sufrido una ligera variación en la fase previa y posterior a la prueba. Sin embargo, la dispersión de los datos es muy grande, aunque desaparece en la fase posterior a la prueba en comparación con la fase anterior. Hay una ligera asimetría en ambos casos. Sin embargo, para el grupo experimental, se observa de igual manera una tendencia central en torno al mismo valor mediano, que ha sufrido una gran variación en la fase previa y posterior a la prueba debido a que en la fase de prueba se agrupan la mayoría de los datos sobre el valor de la media. Sin

embargo, existe una dispersión pequeña de los datos en la fase posterior a la prueba. En la Tabla 5 se muestran los valores de la mediana, la media, la desviación típica y el coeficiente de asimetría de los grupos control y experimental.

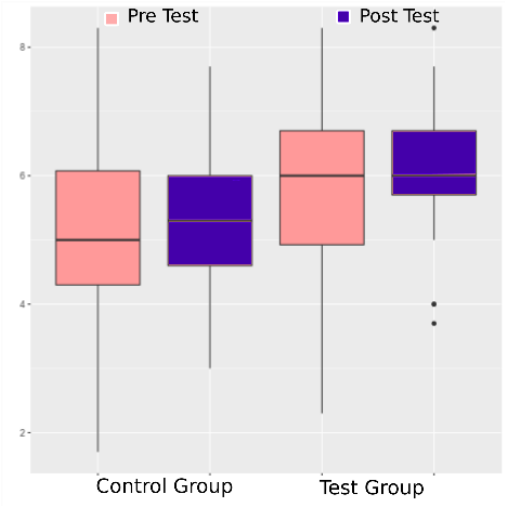

Figura 12: Boxplot para el grupo control y experimental en las fases pre y post de la prueba de Pensamiento Computacional

Tabla 5: Resultados pre-test y post-test del grupo control y experimental en el primer año (K-10)

Grupo	Mediana	Promedio	Desviación estándar	Coeficiente de asimetría
Control Pre-test	5	5,15	1,44	-0,13
Control Post-test	5,3	5,33	1,12	0,15
Test Pre-test	6	5,76	1,48	-0,38
Test Post-test	6	6,06	0,96	-0,28

Estos valores corroboran las afirmaciones anteriores referidas a la Figura 12, ya que en el grupo control la media presenta una ligera mejora (de 5,15 a 5,3) que es casi imperceptible en el caso de la mediana. La desviación estándar, ligeramente baja, con respecto a su asimetría podemos

decir que la mayoría de las calificaciones están cercanas a la media. Por el contrario, el grupo experimental mostró una evidente mejora (de 5,76 a 6,06 en el caso promedio), con datos ligeramente dispersos; con respecto a la desviación típica se nota una considerable baja; respecto de su asimetría podemos decir que hay un porcentaje promedio de notas por encima de la media.

El análisis descriptivo da una primera idea de la distribución de la muestra. Ahora, ofrecemos un estudio más detallado que compara los resultados de las fases previas y posteriores a la prueba en ambos grupos para deducir si hubo alguna mejora significativa.

En primer lugar, estudiamos la fase previa al test para determinar si había diferencias entre el grupo experimental y el control. Debido al tamaño relativamente grande de la muestra, utilizamos la prueba T-Student para muestras independientes y llegamos a la conclusión de que hay diferencias significativas entre ellos, con un significado $p<0,05$. Por lo tanto, podemos asumir que ambos grupos no son homogéneos en la fase de pre-test con respecto a las variables en estudio, sin embargo, observamos diferencias significativas entre ambos grupos en la fase posterior a la prueba, con una significación $p<0,05$.

Esto indica que ha habido una mejora significativa en el grupo experimental; sin embargo, los grupos siguen siendo no homogéneos.

3.2.6.2 Resultados del Test para la mejora Fundamentos de programación

En la Figura 13 se muestra un boxplot para el pretest y el postest de fundamentos de programación. Las cajas están delimitadas por los valores Q1 (primer cuartil) y Q3 (tercer cuartil). Cada cuadro contiene el 50% de los casos correspondientes, destacando la mediana. Los valores mínimo y máximo en los extremos del diagrama corresponden a valores que no son inferiores a Q1-1.5 - (Q3-Q1) y no superiores a Q3 + 1.5 (Q3-Q1).

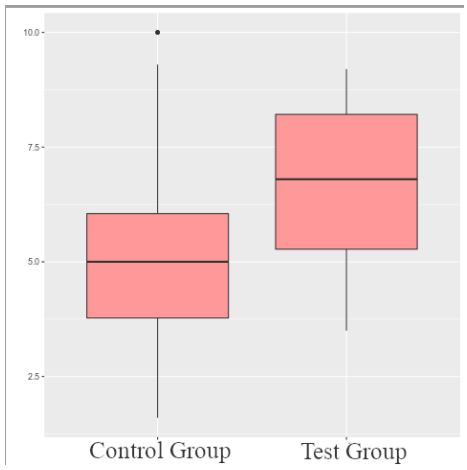

Figura 13: Boxplot para el grupo control y experimental del post test de fundamentos de programación.

Se puede observar que los datos del grupo control mantienen una tendencia central en torno al valor mediano, agrupándose la mayoría de los datos bajo el valor mediano, además existe una dispersión de datos muy grande. Sin embargo, para el grupo experimental se puede observar que el valor mediano es superior al grupo control, los datos mantienen una tendencia central entorno al valor mediano. En la Tabla 6 se muestran los valores de la mediana, la media, la desviación típica y el coeficiente de asimetría de los grupos control y experimental de muestra.

Tabla 6: Resultados de la prueba de fundamentos de programación en el grupo control y experimental en el 1er año (K-10)

Grupo	**Mediana**	**Promedio**	**Desviación Estándar**	**Coeficiente de Asimetría**
Control	5	5,17	2,19	0,68
Test	6,8	6,74	1,69	-0,27

El análisis descriptivo da una primera idea de la distribución de la muestra. Ahora, ofrecemos un estudio más detallado que compara los resultados de la prueba en ambos grupos para deducir si hubo alguna mejora significativa.

En primer lugar, estudiamos la fase previa al test para determinar si había diferencias entre el grupo experimental y el grupo control. Debido al tamaño relativamente grande de la muestra, utilizamos la prueba T-Student para muestras independientes y llegamos a la conclusión de que hay diferencias significativas entre ellos, con un significado $p<0,05$ siendo $p = 0,0007$. Esto indica que ha habido una mejora significativa con el grupo experimental; sin embargo, siguen siendo no homogéneos.

Como puede observarse, el grupo experimental ha obtenido un mejor aprendizaje con respecto al grupo control en la materia de Fundamentos de Programación, previo a realizar las actividades de pensamiento computacional detalladas anteriormente en el apartado procedimiento. Es decir, en el grupo experimental en base a un enunciado (problema) que se les indique a los estudiantes, estos pueden realizar el diagrama de flujo, prueba de escritorio y su respectiva codificación sin mayores complicaciones, encontrando mayores dificultades, en general, en el grupo control.

Capítulo 4. Propuesta Metodológica

4.1 Introducción

Para diseñar un juego serio es necesario conocer cómo se da el proceso de aprendizaje, metodología a seguir para el desarrollo del juego serio, para que va a ser utilizado, a que usuarios va dirigido y la mecánica del juego. En este capítulo se presenta una propuesta de metodología de diseño de juegos serios para la enseñanza de fundamentos de la programación en Educación Secundaria, a fin de aportar una metodología que permita crear juegos serios en el área de programación informática.

El uso de una metodología para el diseño de un juego serio para la enseñanza de fundamentos de programación en Educación Segundaria es importante ya que en la actualidad no existe una metodología enfocada a esta área del conocimiento las metodologías vistas en el capítulo anterior son generales.

Con el desarrollo de esta la metodología para el diseño de un juego serio para la enseñanza de fundamentos de programación en Educación Segundaria, se pretende disminuir las principales dificultades en el aprendizaje de la asignatura de fundamentos de programación como son las fases de la etapa de programación, las habilidades de resolución de problemas, la pedagogía ineficaz y los rasgos y la actitud personales.

4.2 Fases de la Metodología

El desarrollo de un juego serio, debe considerarse como un proceso de ingeniería de software en el que deben establecerse los papeles que juega cada integrante del equipo de desarrollo, las actividades que desarrollan y el producto que se va a obtener (Juego Serio).

La metodología que se propone se llama MeDeJuSeEnPro (Metodología para el desarrollo de juegos serios para la enseñanza de programación). Está metodología incluye las siguientes fases: preproducción, producción, implementación.

4.2.1 Preproducción. Elaboración de un Guion Instruccional

La preproducción es clave para obtener un juego serio de calidad y tiene como objetivo pasar la idea del diseñador mediante un guion instruccional; este documento dará la pauta a la planificación y producción del juego serio.

El guion instruccional es una lista detallada, pantalla por pantalla y procedimientos escritos que indican como se debe desarrollar el juego serio. Además, contempla la tecnología, pedagogía y el contenido que se desea que el estudiante adquiera.

El guion debe detallar el funcionamiento de cada uno de los contenidos en cada una de las pantallas del proyecto, es decir, al diseñar las pantallas se hace funcional el guion.

El guion instruccional permitirá orientar la creación del concepto gráfico y la narrativa de los contenidos de cada una de las pantallas o escenas, así como las sugerencias respecto a imagen fija y en movimiento, audio y video. “Cada pantalla se convierte en un problema de diseño que hay que resolver” (Suarez, 2017).

Para elaborar el guion instruccional debe seguir los siguientes pasos:

4.2.1.1 Identificación del Usuario. Se debe identificar a qué tipo de usuario va dirigido el juego serio. Por ejemplo: “El juego serio está dirigido a estudiantes de Educación Secundaria (15-16 años)”.

4.2.1.2 Tema: Determinar qué es lo que desea que el estudiante aprenda (Tema). Ejemplo: “Tema: Fundamentos de programación”.

4.2.1.3 Objetivos. Establecer los objetivos que tendrá el juego serio. Ejemplo:

Objetivo General

Brindar a los estudiantes el conocimiento teórico y práctico sobre las técnicas básicas de programación estructurada mediante la presentación de conceptos claros y técnicas que permitirá entenderlas fácilmente para que sean capaces de resolver de manera eficaz distintos problemas de carácter general, con independencia del lenguaje de programación utilizado.

Objetivos Específicos

- Aplicar una metodología para la solución de problemas usando conceptos de algoritmia y programación.
- Desarrollar algoritmos que utilicen estructuras simples y de decisión para ser incorporadas como métodos en la solución de problemas.
- Desarrollar algoritmos para la solución de problemas cubriendo las diferentes fases de su elaboración: análisis, diseño, codificación y prueba.
- Usar un lenguaje de programación para implementar los algoritmos planteados en la solución de problemas.

4.2.1.4 Contexto de Aprendizaje. Aquí es necesario identificar y analizar los recursos y las posibilidades con los que cuenta el estudiante, además las limitaciones del medio donde será utilizado el juego serio. Ejemplo: uso de ordenadores, celulares, tablets, internet.

4.2.1.5 Conocimientos Previos. Se deben identificar los conocimientos previos que posee el estudiante, lo cual nos ayudará para la construcción del nuevo conocimiento. Ejemplo:

- El estudiante debe resolver problemas (basados en la vida real) en los que impliquen estrategias cálculos (suma, resta, multiplicación, división, regla de tres, porcentajes).
- El estudiante debe decodificar, comprender y elaborar críticas y propuestas pertinentes en relación al contenido de los textos.

4.2.1.6 Motivación. Se debe crear un juego serio altamente motivacional, es por ello que se debe considerar tanto la motivación interna como la motivación externa.

- Motivación interna es aquella que nos motiva por el simple gusto de hacer las cosas; la recompensa es la ejecución de la propia actividad. Por ejemplo: revisar contenidos.
- Motivación externa es aquella que mueve cuando existe incentivos. Por ejemplo: cómo recibir una recompensa a cambio de haber realizado correctamente una actividad o un castigo por realizarla mal.

El juego debe ser capaz de motivar a los alumnos para que permanezcan inmersos por largos periodos gracias a una serie de características del juego que son de naturaleza motivacional. Estas características incluyen estructuras de incentivos, como estrellas, puntos, tablas de clasificación, insignias y trofeos, así como mecánicas de juego y actividades que los alumnos disfrutan o encuentran interesantes.

4.2.1.7 Elaborar el Temario y Subtemas. Para ello elaboraremos una matriz que se muestra en la Tabla 7.

4.2.1.8 Elaborar Resumen. El temario y subtemas permitirá elaborar el resumen, ejercicios y la evaluación de cada uno de los temas, es decir el contenido (conocimiento) que el estudiante adquirirá al utilizar el juego serio.

Tabla 7: Matriz de contenidos

Objetivos del tema	Temario	Instrumentos de evaluación	Evaluación
Conocer los conceptos básicos de fundamentos de programación	CONCEPTOS BÁSICOS - Definición de lenguaje - Definición de algoritmo - Algoritmos cotidianos - Definición de lenguajes algorítmicos	Prueba de opción múltiple, emparejamiento, verdadero o falso	Solicitar al estudiante que conteste el cuestionario sobre los conceptos básicos
Identificar los pasos a seguir para resolver un problema utilizando un ordenador	METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DE ORDENADORES. - Definición del problema - Análisis del problema - Diseño del algoritmo - Codificación - Prueba y depuración - Documentación - Mantenimiento	Prueba de opción múltiple, emparejamiento, verdadero o falso	Solicitar al estudiante que conteste el cuestionario sobre Metodología para la solución de problemas por medio de computador
Reconocer los tipos de datos, expresiones y operadores que se utilizan en el desarrollo de un algoritmo.	DESARROLLO DE ALGORITMOS - Tipos de datos - Expresiones - Operadores y operandos	Prueba de opción múltiple, emparejamiento, verdadero o falso	Solicitar al estudiante que conteste el cuestionario sobre los conceptos básicos
Conocer los símbolos utilizados para diseñar un diagrama de flujo.	TÉCNICAS PARA LA FORMULACIÓN DE ALGORITMOS - Diagrama de flujo - Recomendaciones para el diseño de Diagramas de Flujo	Prueba de emparejamiento	Solicitar al estudiante que identifique cada elemento utilizado para el diseño de un diagrama de flujo.
Diseñar diagramas de flujo que permitan dar solución a problemas de la vida cotidiana	ESTRUCTURAS ALGORITMICAS - Secuenciales - Condicionales	Prueba de resolución de problemas.	Solicitar al estudiante que realice un diagrama de flujo, prueba de escritorio y codificación en lenguaje C.

4.2.1.9 Creación del Guion Instruccional. Una vez que se tiene listo el resumen de los contenidos se procede a elaborar el guion instruccional. En el diseño del guion instruccional debe constar lo siguiente:

- **Pantalla**: Diseño de la interfaz gráfica que se debe hacer en el orden en que van a ir apareciendo. Aquí se incluirán las imágenes que formarán el escenario (fotos, ilustraciones, pictogramas, íconos, infografías, etc.), se insertará él o los personajes que interactuarán en el juego serio y el texto que desea que se visualice en el escenario; el contenido textual debe ser redactado en primera persona.
- **Instrucciones para el diseño**: Aquí se deberá escribir detalladamente las instrucciones, como debe ir presentándose la escena (pantalla), es decir se debe describir el contenido que se pretende enseñar, el tipo de imágenes (fotos, ilustraciones, pictogramas, iconos, infografías, etc.), animación que tendrán los elementos gráficos, si el personaje debe moverse o debe hablar, qué texto deberá decir el personaje, los sonidos o fondos musicales que tendrá la escena.

Para la elaboración del guion instruccional podemos utilizar hojas de papel con el formato de la Figura 14 o se puede utilizar un editor de texto, hoja de cálculo o un software para realizar presentaciones. Aquí debe ir diseñando pantalla por pantalla como desea que sea las interfaces del juego serio como se observa en la Figura 15. Además, hay que ir escribiendo detalladamente como se debe desarrollar el juego serio (Figura 16).

En el transcurso del proyecto se van a encontrar situaciones como, por ejemplo, que el guion instruccional no es funcional porque no puede llevarse a la pantalla. Entonces habrá que hacer las correcciones pertinentes. También aparecerá la necesidad de recopilar nuevo material o rehacer el que se tiene.

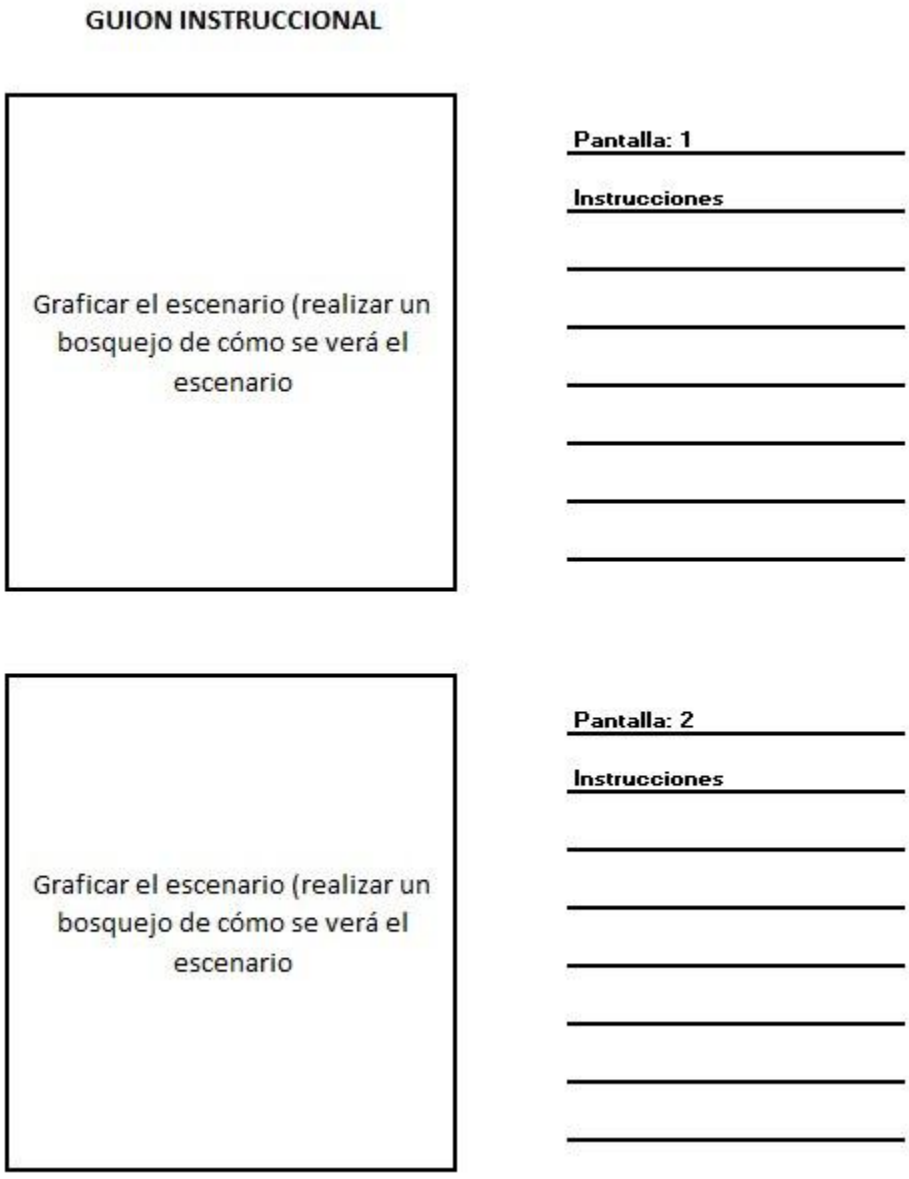

Figura 14: Formato del guion instruccional

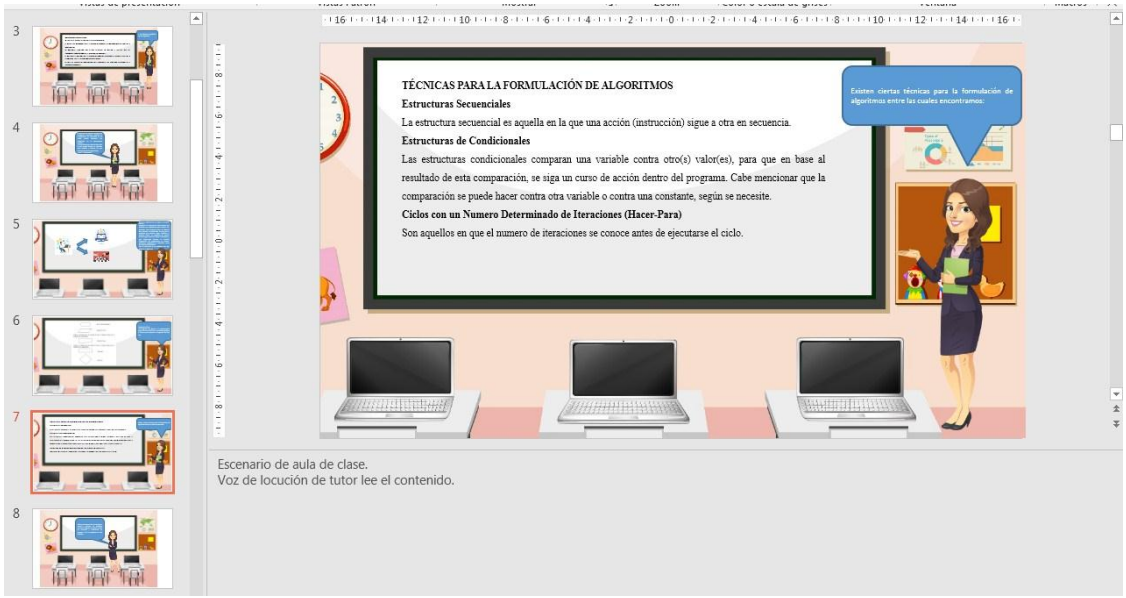

Figura 15: Diseño del guion instruccional – Pantalla con contenidos

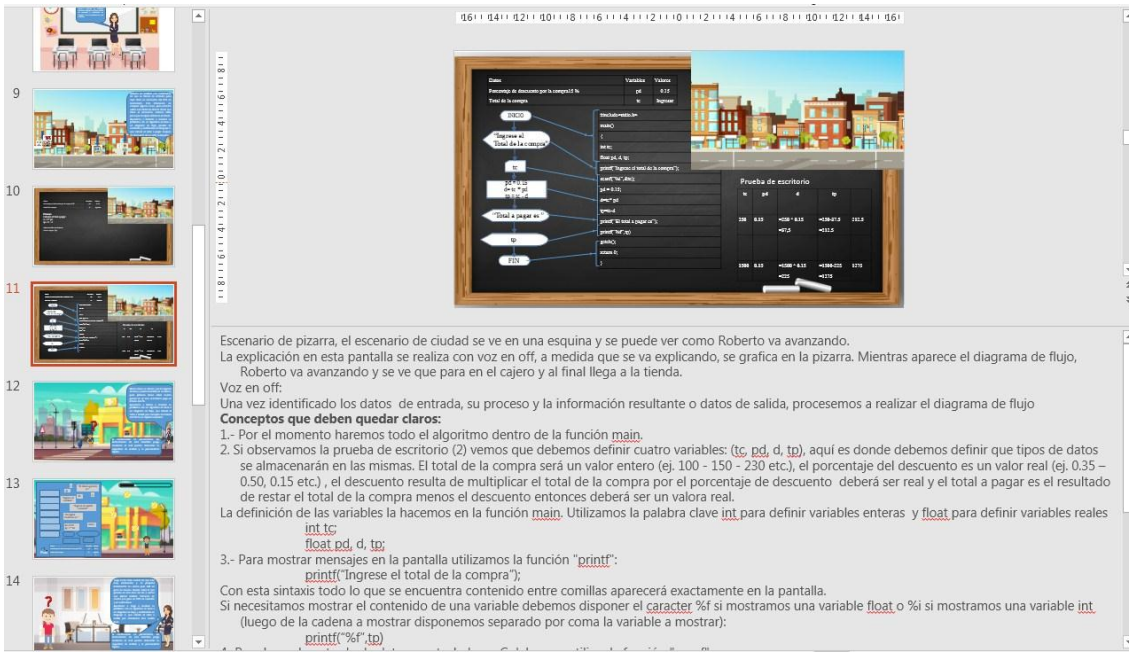

Figura 16: Diseño del guion instruccional – Pantalla resolución de ejercicios y notas que detalla la explicación del desarrollo del ejercicio.

4.2.2 Producción

4.2.2.1 Selección de Recursos. Para la selección del recurso se debe utilizar las instrucciones para el diseño que se encuentra en el guion instruccional, donde están detalladas las imágenes, personaje, texto y sonidos que se requieren en cada escena.

Imágenes. Se puede solicitar a un diseñador gráfico que diseñe todas las imágenes (fotos, ilustraciones, pictogramas, íconos, infografías, personajes etc.) que se requieren en cada una de las escenas. En el caso de no poder contar con un diseñador gráfico se pueden buscar imágenes en internet y descargarlas para ser utilizadas.

Texto. Los textos que expresará el personaje se deben grabar en formato de audio con la ayuda del software pertinente.

Las imágenes, personajes, audios y demás recursos que utilice deben ser de su autoría o deben tener una licencia Creative Commons.

4.2.2.2 Selección del Software. Existe un elevado número de software que permite crear juegos, sin embargo, la elección del software depende del diseñador del juego y de la experiencia que él tenga en el desarrollo de juegos.

4.2.2.3 Diseño del Juego Serio. El diseñador del juego toma el guion instruccional y va definiendo los elementos fundamentales que se requieren para el diseño de juego serio. Los elementos que se utilizaran en el diseño del juego serio son los que propone Plass et al., (2015). Los elementos son: mecánica del juego, diseño estético visual, diseño narrativo, sistema de incentivos, partitura musical, los contenidos y competencias relacionados que abarcará el juego.

Mecánica del Juego. La mecánica de juego es la que incorpora el comportamiento esencial en este caso vinculado a la actividad de aprendizaje o de evaluación, establecen las acciones que

se ejecutarán cuando el jugador haga clic en una determinada zona. En el guion instruccional se encuentran indicadas las actividades a desarrollar, además una idea de cómo deberá presentarse las actividades para que el alumno las desarrolle e ir midiendo el nivel de conocimiento que va adquiriendo. Por ejemplo: En el juego "De camino al Colegio" (Figura 17), el jugador debe ir respondiendo las preguntas. Si la respuesta es correcta, puede avanzar; de lo contrario no puede hacerlo hasta que responda correctamente la pregunta.

Diseño Estético Visual. El diseño visual es el que determina como se verá el juego tanto en el aspecto general como de los personajes que intervienen en el juego. El diseño visual determina cómo se visualizan las herramientas y las funciones de la mecánica del juego, es decir permite tener una función cognitiva y otra estética. El diseñador del juego tomará como basé las pantallas que se encuentran en el guion instruccional, y los recursos seleccionados (imágenes, personajes). Por ejemplo: En el juego DFD-C la pantalla que se presenta con los elementos del diagrama de flujo es un tablero verde, los elementos del diagrama de flujo son objetos transparentes, los obstáculos son calabazas que se mueven para impedir que los objetos pasen a la zona blanca de la izquierda, que es la zona para montar el DFD como se muestra en la Figura 18.

Diseño Narrativo. La narrativa de un juego son las líneas argumentales que se avanzan a través de elementos como las escenas, las acciones del juego, los diálogos y las voces en off. El diseñador del juego tomará como basé las pantallas y las instrucciones que se encuentran en el guion instruccional ya que allí indica las secuencias de las pantallas, los diálogos que los personajes realizarán, todos los audios los encontrará en los recursos seleccionados. Por ejemplo: En el juego DFD-C existe narración en todo el contenido, en cada actividad hay una descripción detallada de lo que el jugador debe hacer escrita en forma de diálogos como se muestra en la Figura 19.

Figura 17: Juego "De camino al Colegio" El jugador debe armar el DFD para el algoritmo secuencial. Al arrastrar la pieza del DFD (derecha) y soltarla en el tablero (izquierda), se produce una colisión con la calabaza.

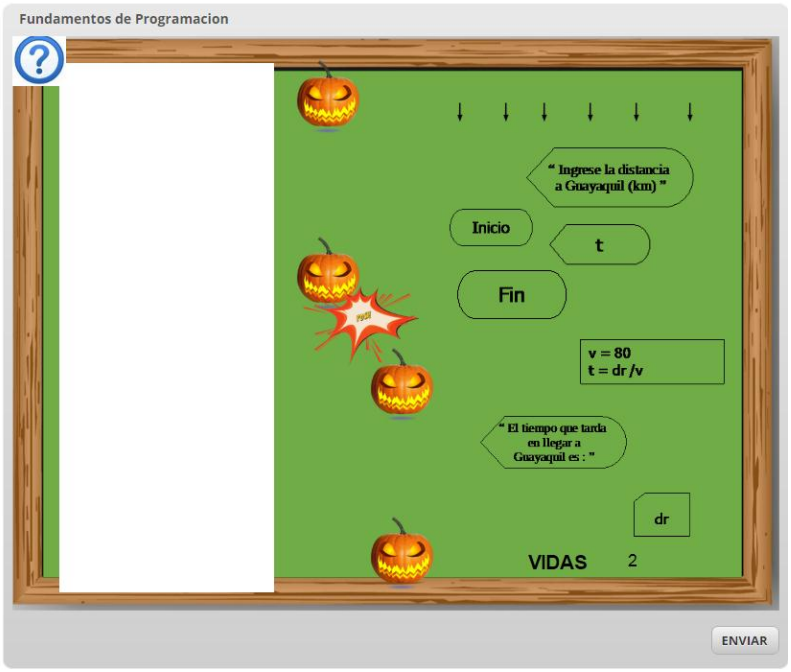

Figura 18: Juego "Evita las calabazas" el jugador debe armar el DFD para el algoritmo secuencial. Al arrastrar la pieza del DFD (derecha) y soltarla en el tablero (izquierda), se produce una colisión con la calabaza

Figura 19: Narración de una actividad en el brazo el juego DFD para un algoritmo condicional.

Sistema de Incentivos. El sistema de incentivos de un juego muestra todos los elementos de motivación que pretenden animar a los jugadores a continuar con sus esfuerzos y la retroalimentación trata de modificar adecuadamente su comportamiento. El diseñador del juego encontrará en el guion instruccional sección motivación el sistema de incentivo que utilizará el juego (estrellas, puntos, tablas de clasificación, insignias y trofeos). Por ejemplo: En el juego DFD-C "De camino al Colegio" (Figura 17), los alumnos deben responder a las preguntas, mientras avanzan hacia la escuela. Si la respuesta no es correcta se les da otra oportunidad (tratando siempre de animarlos a seguir revisando). En el juego "Ronda de penalizaciones", el jugador debe marcar un gol respondiendo a la pregunta. Si la respuesta es correcta, marca un gol y acumula puntos como se muestra en la Figura 20. Por cada 30 puntos, los alumnos obtienen una vida extra que pueden utilizar en cualquiera de los juegos DFD-C.

Figura 20: Juego "Ronda de penalizaciones". Cuando el alumno responde correctamente suma puntos que le pueden dar una vida extra

Partitura Musical. La partitura musical de un juego incluye los sonidos de fondo que suelen utilizarse para dirigir la atención del jugador en momentos importantes o específicos del juego. Los sonidos de fondo que se utilizará en el juego, el diseñador del juego los puede encontrar en los recursos seleccionados basados en el guion instruccional. En el juego DFD-C, hay diferentes sonidos de fondo en cada actividad. Por ejemplo, en el juego "ronda de penaltis" hay varios sonidos de fondo para diferentes momentos, como: mientras se prepara para patear el balón, marcar el gol, fallar el gol, ganar o perder la ronda de penaltis.

Contenidos y Competencias. Son los contenidos de la asignatura y las competencias que el juego pretende enseñar. El contenido de un juego serio tiene un profundo impacto en todos los elementos principales del juego y su diseño. Es por ello que el diseñador del juego debe tener siempre a mano el guion instruccional, ya que allí se detalla todo el contenido de aprendizaje y como se desea que se presente. El juego DFD-C tuvo en cuenta los siguientes: conceptos básicos

de metodología de programación, C un lenguaje estructurado, inicio de la programación, estructuras estáticas, sentencias secuenciales y condicionales. Estos contenidos se presentan antes de los juegos y actividades como se muestra en la Figura 21.

Figura 21: Ejemplo de pantalla con contenidos, en este caso "Tipos de operadores."

4.2.3 Implementación

Una vez comprobadas las funcionalidades del juego serio, se procederá a instalar la aplicación en los recursos seleccionados en la fase análisis en el punto contexto de aprendizaje. Por ejemplo: el juego serio DFD-C se puso a disposición de los estudiantes en línea (Montes León, 2020) como se observa en la Figura 22.

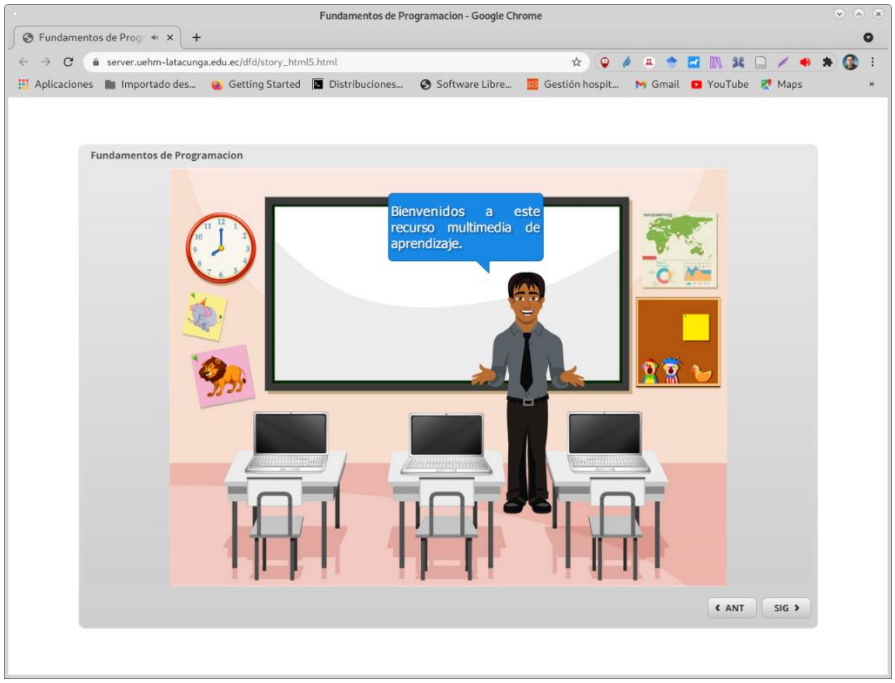

Figura 22: Implementación del Juego Serio DFD-C en línea.

El juego se encuentra disponible en línea en la dirección web: https://server.uehm-latacunga.edu.ec/dfd/story_html5.html

Capítulo 5. Validación de la Metodología

Con la finalidad de validar si la metodología propuesta MeDeJuSeEnPro (Metodología para el desarrollo de juegos serios para la enseñanza de programación) ayuda a crear juegos serios que permita al estudiante adquirir conocimientos de programación informática, se procedió a desarrollar un juego serio "DFD-C" para la enseñanza de programación a estudiantes de secundaria.

5.1 Objetivo

- Determinar si los estudiantes mejoraron significativamente su comprensión de los fundamentos de la programación al utilizar el juego serio DFD-C desarrollado con la metodología MeDeJuSeEnPro
- Evaluar si el uso de los juegos serios fue una experiencia de juego positiva.

5.2 Participantes y Contexto

El experimento se llevó a cabo en el Colegio "Unidad Educativa Hermano Miguel" de la ciudad de Latacunga (Ecuador), desde el 27 de abril de 2020 hasta el 9 de junio de 2020, con un total de 38 estudiantes de K-10 (15 y 16 años) del primer año del bachillerato técnico en la especialidad de informática. El curso se dividió aleatoriamente en un grupo experimental de 19 estudiantes (68% hombres 32% mujeres) que utilizaron el juego serio DFD-C y un grupo control de 19 estudiantes (47% hombres y 53% mujeres) que utilizaron el enfoque convencional como en cursos anteriores. Un instructor con 10 años de experiencia en la enseñanza del curso "Fundamentos de la programación" impartió clases a los 38 estudiantes. Las herramientas de investigación de este experimento fueron un test de Fundamentos de la Programación y un cuestionario para evaluar los 7 aspectos de la experiencia de juego: logro, reto, competición, orientación, inmersión, juego y experiencia social de los estudiantes.

5.3 Instrumentos

Una vez formados los grupos, los profesores procedieron a trabajar de la siguiente manera:

Ocho especialistas en informática (5 profesores de secundaria y 3 de universidad), todos ellos ingenieros informáticos, diseñaron y validaron la prueba de Fundamentos de la Programación (que se utilizaría como pretest y postest) basándose en el programa del curso. Las preguntas del test se seleccionaron en función de las competencias que los alumnos deben adquirir una vez finalizada cada unidad del curso; la estructura del test se explica en la Tabla 8.

Tabla 8: Pre-test y post-test de fundamentos de programación

Parte	Pregunta/Concepto/Problema	Tipo	Puntuación (pts.)
Concepto	Definición de ordenador	Sí o no	0.5
	Análisis de problemas, codificación, documentación, mantenimiento, pruebas y depuración	A juego con las definiciones	0.5
	Metodología para la resolución de problemas por ordenador	Pedir	0.5
	Prioridad de los operadores	Opción múltiple	0.5
Práctica	Dado un número de porcentajes para la nota final de un alumno en un curso, definir su nota final pidiendo entradas, y luego mostrar las salidas	Diseñar un diagrama de flujo de la estructura secuencial (DFD)	3
	Dado un diagrama de flujo sobre el cálculo de los importes totales en función de los valores dados	Realice la prueba de escritorio	3
	Dado un código escrito en C con entradas, operaciones de datos y salidas	Corregir errores de sintaxis	2

Los conocimientos previos del alumno se evaluaron con el pre-test (que fue el mismo que el post-test) con una escala de puntuación de 1 a 10 puntos. La prueba se viene utilizando como

instrumento de evaluación en la Unidad Educativa "Hermano Miguel" de Ecuador desde el curso 2016-2017.

La experiencia lúdica se evaluó mediante el cuestionario validado "Gamefulquest" (Högberg et al., 2019), para valorar el DFD-C SG, con una escala Likert de 7 puntos que se detalla en la Tabla 9.

Tabla 9: Cuestionario de experiencia de juego (gamefulquest)

Dimensión	Número de items	Escala
Realización	8	7 puntos Likert
Desafío	7	
Concurso	7	
Guiado	7	
Inmersión	9	
Juguetón	9	
Experiencia social	8	

Se obtuvo un valor alfa de Crombatch de 0,71 para la prueba utilizada como pre-test y post-test. Para el cuestionario Gamefulquest con una escala Likert de 7 puntos, que incluye 7 dimensiones, se obtuvieron los valores alfa de Crombatch para sus dimensiones de logro (0,98), reto (0,81), competición (0,89), orientación (0,98), inmersión (0,91), juego (0,91) y experiencia social respectivamente (0,96).

5.4 Hipótesis

- H1: Los estudiantes que utilizan el juego serio DFD-C, ¿mejoran significativamente su aprendizaje de los fundamentos de la programación en comparación con los que no lo utilizan?

- H2: ¿El género no influye en la mejora de los estudiantes que utilizan el juego serio DFD-C?

5.5 Procedimiento

El procedimiento del estudio experimental se ilustra en la Figura 23. Incluye 5 semanas de clases presenciales de 2 horas semanales sobre el aprendizaje de conocimientos básicos de programación, que forman parte del curso de Fundamentos de Programación. Debido a la aparición de la pandemia mundial causada por el COVID19, las clases presenciales se suspendieron en Ecuador el 16 de marzo de 2020, lo que obligó a las instituciones educativas a buscar medios y recursos tecnológicos que ayuden a continuar la preparación académica de los estudiantes. Así, en la Unidad Educativa Hermano Miguel se implementó un aula virtual para la institución, que entró en funcionamiento el 30 de marzo. El aula virtual fue utilizada por los profesores para subir recursos sobre los temas a tratar en las clases en línea de una hora semanal. También se estableció que el aula virtual fuera el medio para que las autoridades y los docentes dieran las indicaciones necesarias para poder avanzar con los planes establecidos para el año académico. Así, en la última semana de abril, se autorizó a los profesores a dar clases por videoconferencia según un horario que fue entregado por el Vicerrector, para que la experimentación pudiera continuar. La intervención comenzó con un pre-test de 1h 30' que realizaron todos los alumnos de primer curso de la especialidad de informática en el aula virtual. A continuación, se iniciaron las clases por videoconferencia de una hora a la semana que duraron 6 semanas. Al grupo experimental se le entregó en línea el juego serio DFD-C, indicándoles que "pueden entrar tantas veces como deseen y en cualquier momento" y que "las actividades y la puntuación obtenida no influyen en la calificación de la asignatura de fundamentos de programación". Paralelamente, al grupo control se le entregó un conjunto de ejercicios resueltos que cubrían los mismos temas que el grupo

experimental, y que podían trabajar también por su cuenta, como usualmente se ha hecho en esta asignatura. Cada grupo dispuso de 6 semanas. Al final de este periodo, los estudiantes volvieron a realizar el post-test de 1h 30', para evaluar y controlar cualquier cambio en los Fundamentos de Programación. Además, el grupo experimental completó el cuestionario Gameful Experience de 20 minutos para evaluar su percepción hacia el uso del juego serio DFD-C.

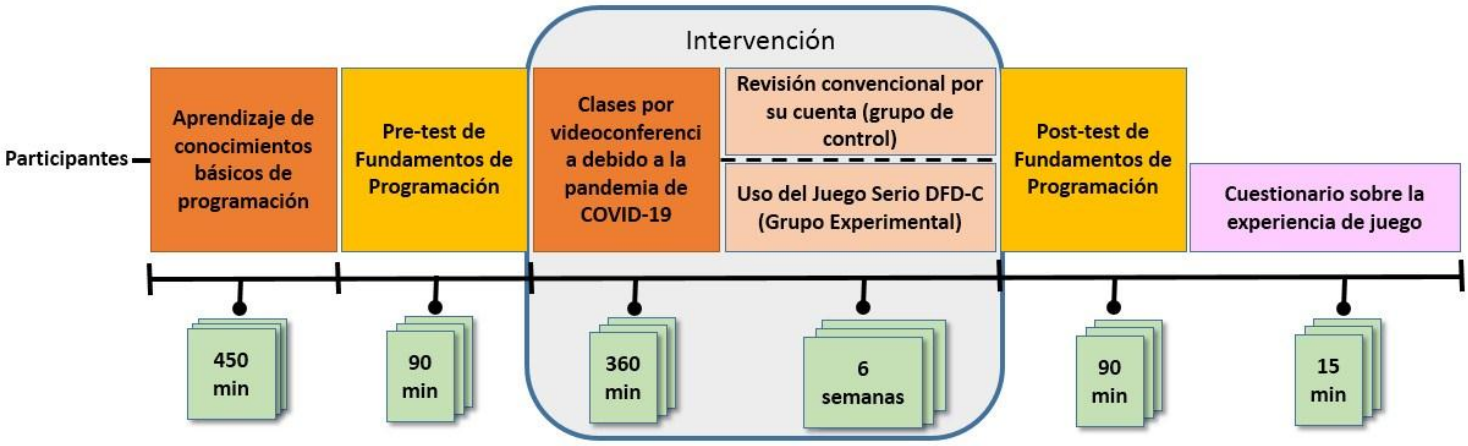

Figura 23: Diagrama del diseño experimental.

5.6 Resultados

5.6.1 Resultados Aprendizaje de los Fundamentos de la Programación

Como uno de los objetivos principales de este estudio, se investigó la eficacia del sistema propuesto considerando su papel en la mejora del rendimiento escolar de los alumnos. Los análisis de resultados se orientaron a realizar un análisis descriptivo de los datos presentados y luego a

deducir si había o no una mejora significativa entre el grupo control y el grupo experimental de la variable evaluada en el primer año de bachillerato.

La Figura. 24 muestra un boxplot para el pre-test y el post-test aplicado al grupo control y al grupo experimental sobre los conocimientos de los fundamentos de la programación.

Las cajas están delimitadas por los valores Q1 (primer cuartil) y Q3 (tercer cuartil). Cada caja contiene el 50% de los casos correspondientes, destacando la mediana. Los valores mínimos y máximos en los extremos del diagrama corresponden a valores que no son inferiores a Q1-1,5 - (Q3-Q1) ni superiores a Q3 + 1,5 (Q3-Q1).

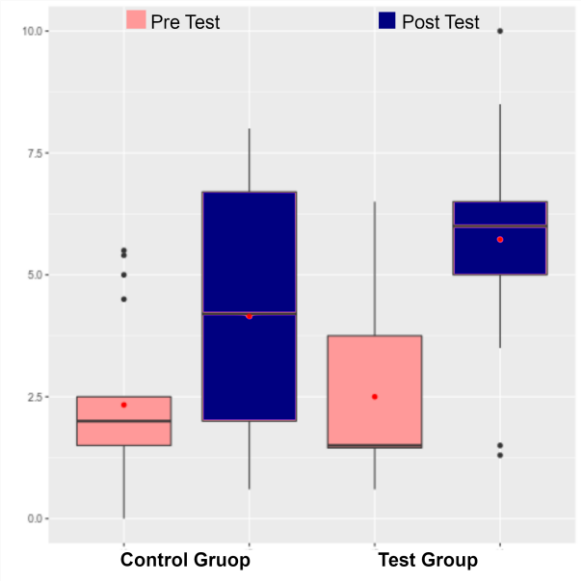

Figura 24: Boxplot para el grupo control y experimental en las fases pre y postest a la prueba de conocimientos de fundamentos de programación.

Se observa que los resultados del pretest, tanto del grupo control como del grupo experimental, mantienen una tendencia central en torno al mismo valor mediano. En cuanto a los resultados del pos-test, se observa un pequeño aumento del valor de la mediana en el grupo control, mientras que en los resultados del grupo experimental hay un aumento considerable del valor de

la mediana, y también hay una pequeña dispersión de los datos. La Tabla 10 muestra los valores de la mediana, la media, la desviación estándar y el coeficiente de asimetría del grupo control y del grupo experimental.

Tabla 10: Resultados de la prueba pre y pos-test del grupo control y experimental sobre los conocimientos de programación

Grupo	**Mediana**	**Promedio**	**Desviación estándar**	**Coeficiente de asimetría**
Control Pre-test	2	2.33	1.58	0.72
Control Post-test	4,2	4.15	2.40	0.19
Test Pre-test	1,5	2.5	1.75	0.92
Test Post-test	6	5.72	2.10	-0.26

Estos valores corroboran las afirmaciones anteriores referidas a la Figura 24, ya que en el grupo control la media muestra una ligera mejora (de 2,33 a 4,15) que es casi imperceptible en el caso de la mediana. La desviación estándar, ligeramente baja, con respecto a su asimetría indica que la mayoría de las valoraciones están cerca de la media. Por el contrario, el grupo experimental mostró una mejora evidente (de 2,5 a 5,72), con datos ligeramente dispersos; con respecto a la desviación estándar es considerablemente baja; por su parte la asimetría indica que hay un porcentaje medio de calificaciones por debajo de la media.

El análisis descriptivo da una primera idea de la distribución de la muestra. A continuación, se realiza un estudio más detallado en el que se comparan los resultados de la fase previa y posterior a la prueba en ambos grupos para deducir si hubo alguna mejora significativa.

En primer lugar, se estudió el pre y el pos-test para determinar si había diferencias entre los grupos experimental y control. Se encontró normalidad y homocedasticidad en las muestras para realizar la prueba T.

Para este estudio se utilizó la prueba T de Student para muestras independientes, concluyendo que había diferencias significativas en el valor que evalúa el aprendizaje, con una significación de $p<0,05$, donde $p = 0,04$ para el grupo experimental.

Esto indica que ha habido una mejora significativa en el grupo experimental que utilizó el juego serio DFD-C en su curso.

El análisis se amplió para evaluar las diferencias de aprendizaje con respecto al género, como se muestra en la Figura 25.

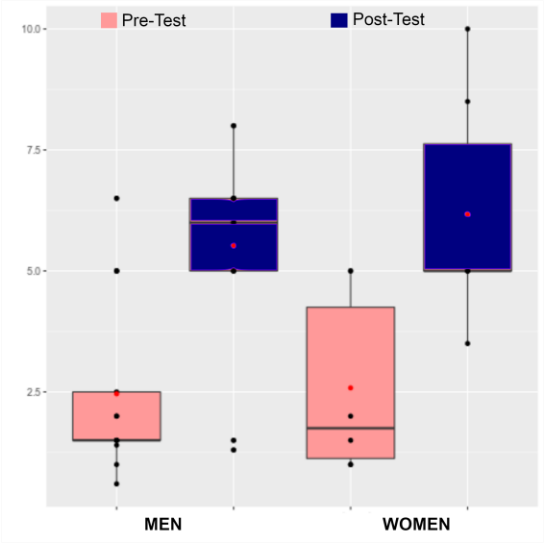

Figura 25: Boxplot para el grupo de hombres y mujeres en las fases de pre y post test de los conocimientos de fundamentos de programación del grupo experimental

Se puede observar en el post-test que tanto los hombres como las mujeres tuvieron una mejora significativa, pero, considerando el valor de la mediana, parece que las mujeres mejoran algo más que los hombres. La Tabla 11 muestra los valores de la mediana, la media, la desviación estándar y el coeficiente de asimetría del grupo experimental.

Estos valores corroboran las afirmaciones anteriores referidas en la Figura 25, ya que los hombres muestran una mejora (de 2,46 a 5,52), con la desviación típica, ligeramente baja; respecto a su asimetría se puede decir que la mayoría de las valoraciones se acercan a la media. Por otro lado, las mujeres muestran una mejora evidente (de 2,58 a 6,17), con datos ligeramente dispersos; con respecto a la desviación estándar es considerablemente baja. Por su parte, la asimetría indica que hay un porcentaje medio de calificaciones por encima de la media.

Tabla 11: Resultados de la prueba previa y posterior del grupo de hombres y mujeres sobre conocimientos básicos de programación.

Grupo	**Mediana**	**Promedio**	**Desviación estándar**	**Coeficiente de asimetría**
Control Pre-test	1.5	2.46	1,76	1.06
Control Post-test	6	5.52	1.99	-0.92
Test Pre-test	1.75	2,58	1.74	0.45
Test Post-test	5	6.17	2.29	0.48

El análisis descriptivo da una primera idea de la distribución de la muestra. Ahora se ofrece un estudio más detallado en el que se comparan los resultados de las fases previa y posterior a la prueba en ambos grupos para deducir si existe alguna diferencia significativa en la mejora entre hombres y mujeres.

En primer lugar, se estudió la fase previa a la prueba para determinar si había diferencias entre hombres y mujeres.

Para este estudio se utilizó la prueba T de Student para muestras independientes. Se concluyó que no había diferencias significativas entre hombres y mujeres, con una significación de $p<0,05$, donde $p = 0,59$.

Esto indica que no ha habido diferencias significativas en la mejora entre hombres y mujeres.

5.6.2 Resultados Cuantitativos en la Experiencia de Juego

Otra preocupación importante de este estudio era determinar si los estudiantes habían tenido una actitud positiva hacia el uso del juego serio DFD-C. Por lo tanto, se evaluó la experiencia lúdica del uso del Juego Serio por parte de los estudiantes del grupo experimental mediante el Cuestionario de Experiencia Lúdica (Högberg et al., 2019) "Gamefulquest" realizado para medir la experiencia lúdica del DFD-C donde se consideraron siete dimensiones o aspectos. La Figura 26 muestra los resultados globales por dimensión y también divididos por hombres y mujeres.

Se puede observar que todas las dimensiones se valoran de forma bastante positiva en una escala de 1 a 7. Las dimensiones más valoradas fueron la orientación y el juego, seguidas (en este orden) por la experiencia social y la realización. La competición, el reto y la inmersión también fueron muy bien valorados por los alumnos.

5.6.2.1 Realización - Los participantes afirmaron comúnmente estar de acuerdo en un 31,6% y en un 31,6% estar muy de acuerdo con tener un sentido de realización. También se mostraron de acuerdo en un 31,6% y muy de acuerdo en un 36,8% con que parecía ser parte de un impulso de progreso y una voluntad de mejorar siempre.

Figura 26: Gráfico de barras de los resultados del Cuestionario de Experiencia de Juego en los 7 aspectos evaluados en una escala Likert de 1 a 7 (para mujeres y hombres).

5.6.2.2 Reto - El 47,4% de los participantes indicó estar totalmente de acuerdo, y el 26,3% estar de acuerdo en que se necesita mucho esfuerzo para tener éxito en las actividades, lo que a su vez les motiva para hacer cosas muy exigentes y para mejorar continuamente.

5.6.2.3 Competición - El 31,6% de los participantes mencionó que se siente totalmente de acuerdo en que trabajan como si se tratara de una competición y el 31,6 % está de acuerdo en que el reto les hace querer estar en primer lugar.

5.6.2.4 Guiado - El 52,6% de los participantes estuvo totalmente de acuerdo en que se sintió guiado y el 47,4% estuvo totalmente de acuerdo en que sintió que tenía un instructor, además de recibir ayuda y comentarios guiados.

5.6.2.5 Inmersión - El 26,3% de los participantes reseñaron que estaban algo de acuerdo y el 26,3% que estaban de acuerdo en que había un cambio en sus percepciones del mundo real,

como el rápido paso del tiempo o un comportamiento dirigido que se hizo menos difícil porque el juego serio actuó como una distracción y captó la atención de cada jugador.

5.6.2.6 Lúdica - El 78,9% de los participantes refirieron que estaban totalmente de acuerdo en que era una experiencia lúdica, el 42,1% indicaron que les daba la sensación de explorar las cosas y querer saber lo que viene después, haciéndoles sentir que estaban descubriendo cosas.

5.6.2.7 Experiencia social - El 42,1% de los participantes indicaron que estaban totalmente de acuerdo en que se sentía como una experiencia social, el 31,6% de los participantes indicaron que les dio la sensación de no estar solos; se sintieron como si estuvieran socialmente involucrados, y de estar conectados con otros, así como tener la sensación de ser apreciados por lo que habían logrado.

En cuanto a la diferencia de percepciones entre hombres y mujeres, los valores son muy similares, pero destaca que las mujeres han tenido una actitud ligeramente más positiva en todas las dimensiones excepto en la de realización, en la que los valores están casi igualados. Por lo tanto, las mujeres muestran una actitud ligeramente más positiva hacia el uso del juego serio DFD-C que los hombres.

5.7. Conclusiones

El uso de juegos serios para enseñar a programar a los estudiantes de Educación Secundaria aumenta significativamente sus resultados de aprendizaje. Esta mejora es similar para alumnos y alumnas, ya que el género no parece ser un factor discriminatorio.

Un experimento en el que 38 estudiantes de K-10 se dividieron en el grupo control (sin uso de DFD-C) y el grupo experimental (uso de DFD-C) se llevó a cabo durante el año académico

2019-2020. Los estudiantes del grupo experimental mejoraron significativamente sus puntuaciones en una prueba pre-post de conocimientos, tanto para hombres como para mujeres.

También se puede concluir que los estudiantes de Educación Secundaria, de ambos sexos, perciben la experiencia lúdica como positiva, con una percepción ligeramente más positiva por parte de las mujeres. A los estudiantes del grupo experimental (uso del DFD-C) se les pidió que completaran el cuestionario validado Gamefulquest para explorar sus experiencias de juego en relación con el logro, el desafío, la competencia, la orientación (dimensión guiada), la inmersión, la lucidez y la experiencia social. La dimensión lúdica fue la más valorada, seguida de la orientación y la sociabilidad. Las alumnas valoraron la experiencia de juego ligeramente por encima de los alumnos. Parece que las mujeres no sólo rindieron tan bien como los hombres en las pruebas, sino que también percibieron un poco mejor la experiencia lúdica.

El uso de los juegos serios para enseñar a programar a los estudiantes de Educación Secundaria aumenta significativamente sus resultados de aprendizaje. El juego serio DFD-C fue creado utilizando la metodología MeDeJuSeEnPro (Metodología para el desarrollo de juegos serios para la enseñanza de programación). Por todo ello, se puede afirmar que la metodología propuesta es válida.

Capítulo 6. Conclusiones

6.1 Objetivos Cumplidos

En el proceso de desarrollo de la propuesta se han investigado conceptos teóricos del proceso de enseñanza-aprendizaje de programación informática. Logrando identificar los principales problemas que tienen los estudiantes para el aprendizaje de esta asignatura. Esto ha sido posible gracias a los trabajos realizado y explicados anteriormente en el capítulo 3. Con la base teórica del proceso y la evidencia de las dificultades de la enseñanza-aprendizaje de programación informática y la experiencia de más de 10 años como docente de la asignatura de programación informática, se ha podido identificar con facilidad en el aula los principales problemas que tiene el proceso de enseñanza-aprendizaje de programación informática. Se han identificado los siguientes problemas:

- En la fase de etapa de programación, uno de los principales problemas es la dificultad de analizar y entender el problema para luego dar solución mediante un algoritmo.
- Con respecto a las habilidades de resolución de problemas, estas están directamente relacionadas con la fase anterior, ya que si el estudiante tiene dificultades de analizar y entender el problema por carecer de habilidades como la comprensión de la lógica serán incapaces de proseguir. Además, no dominan bien la sintaxis y las estructuras de programación, lo que resulta en un diseño del programa de mala calidad y defectuoso.

Gracias a las experiencias “Aplicaciones móviles basadas en juegos para el aprendizaje de la programación” que se explican en el capítulo 3, se ha podido concluir que el uso de una app de juegos serios para estudiantes de Secundaria como recurso didáctico en el aula ayuda a los estudiantes a aumentar sus conocimientos en los fundamentos de la programación.

La aplicación de la experiencia “Mejoras del pensamiento computacional en estudiantes de secundaria con tareas Unplugged” que se explica en el capítulo 3, ayudan a los estudiantes a mejorar el pensamiento computacional. Estas actividades, a su vez influyen positivamente en sus conocimientos de fundamentos de la programación porque mejora su pensamiento lógico-matemático en el proceso de enseñanza y aprendizaje.

Los juegos serios pueden ser integrarse en todas las áreas del proceso educativo tales como las matemáticas, la física, los estudios sociales, etc. debido a que existe una variedad de ellos. Sin embargo, metodología disponible a seguir para el desarrollo de juegos serios en cada área es muy limitada. La metodología difiere debido a los distintos contextos educativos, objetivos y contenidos ya que cada uno de los elementos del juego serio marca realmente su adecuación al contexto de aprendizaje.

Se ha propuesto la metodología MeDeJuSeEnPro para diseñar juegos serios enfocados a la enseñanza de la asignatura de programación. La metodología tiene tres fases: Preproducción, Producción e Implementación. En la fase de preproducción se requiere la colaboración de los docentes que imparten la asignatura de programación informática ya que son ellos quienes manejan y conocen los contenidos. Además, conocen las posibilidades con que cuenta el estudiante y las limitaciones del medio donde se utilizará el juego serio. Antes incluso de la fase de producción debe cuidarse al máximo la buena comunicación entre docentes participantes a fin de elaborar un guion instruccional eficiente y entendible para que, ya en la fase de producción el diseñador del juego no tenga dificultad en interpretarlo y realizar el juego serio. La fase de implementación tiene una estrecha relación con la fase de preproducción ya que en esa fase es donde de define el contexto de aprendizaje.

La metodología MeDeJuSeEnPro que se ha presentado se validó mediante el diseño e implementación del juego serio DFD-C en la Unidad Educativa “Hermano Miguel” de la ciudad de Latacunga (Ecuador), donde se aplicó a los estudiantes de primer año de bachillerato (15-16 años). Una vez implementado el juego serio DFD-C fue evaluado atendiendo a si los estudiantes mejoraban el rendimiento escolar utilizando este recurso. El resultado fue que quienes utilizaron el juego DFD-C mejoraron significativamente en el aprendizaje de programación informática con relación a los estudiantes que no utilizaron el juego.

También se pidió a los estudiantes que utilizaron el juego serio DFD-C que completasen el cuestionario validado "Gamefulquest" (Högberg et al., 2019). Se pretende con ello explorar sus experiencias de juego en relación con varios aspectos, en concreto: el logro, el desafío, la competencia, la orientación (dimensión guiada), la inmersión, la lucidez y la experiencia social. El resultado es que la dimensión lúdica fue la mejor valorada, seguida de la orientación y la sociabilidad. De acuerdo con los resultados positivos obtenidos en el uso del juego serio DFD-C, se concluye que la metodología propuesta es válida para crear juegos serios utilizables con éxito en la enseñanza de programación informática a estudiantes de Educación Secundaria.

6.2 Contribuciones

Se contribuye al área de los juegos serios y a la enseñanza de la programación con las siguientes publicaciones:

Montes-León, H., Hijón-Neira, R., Pérez-Marín, D., y León, S. R. M. (2019). Improving Programming Learning on High School Students through Educative Apps. 2019 International Symposium on Computers in Education (SIIE), 1–6. https://doi.org/10.1109/SIIE48397.2019.8970112.

Montes-León, H., Montes-León, S., Pérez-Marín, D., y Hijón-Neira, R. (2020). Mejora del Pensamiento Computacional en Estudiantes de Secundaria con Tareas Unplugged. https://doi.org/10.14201/eks.23002. El artículo es un SJR Q3.

Montes, H., Hijón-Neira, R., Pérez-Marín, D., y Montes, S. (2021). Using an Online Serious Game to Teach Basic Programming Concepts and Facilitate Gameful Experiences for High School Students. IEEE Access, 9, 12567–12578. El artículo es un JCR Q1.

Además, en el desarrollo del juego serio DFD-C se propone un contenido de programación informática (conceptos, metodología de solución de problemas, desarrollo de algoritmos, técnicas de formulación de algoritmos, estructuras algorítmicas secuenciales y condicionales) para utilizarse con estudiantes de Educación Secundaria.

6.3 Limitaciones

Como en toda investigación experimental, este estudio también presenta algunas limitaciones que hay que tener en cuenta.

Los resultados preliminares de la validación son limitados hasta el momento ya que se ha realizado en un grupo de estudiantes. Sin embargo, indican claramente que la metodología MeDeJuSeEnPro ayudará a los desarrolladores a producir juegos serios más eficientes para la enseñanza de programación informática.

La metodología MeDeJuSeEnPro facilita el diseño de los juegos serios de una manera clara y sencilla. Sin embargo, sigue siendo necesario tener una amplia gama de conocimientos en el uso de software para desarrollar juegos serios.

6.4 Trabajo futuro

Como trabajo futuro se pretende buscar expertos en el desarrollo de juegos serios y conjuntamente con ellos ampliar los conceptos de programación informática a introducir en el juego y que de esta manera lo utilicen más instituciones de Educación Secundaria y Universidades.

Como el juego serio DFD-C, desarrollado con la metodología MeDeJuSeEnPro y validado en la Unidad Educativa "Hermano Miguel", se tuvieron resultados positivos en la enseñanza-aprendizaje de programación informática. Desde estas líneas se propone a las autoridades la de la Institución que lo incorporen desde el primer año de Educación Básica hasta el Tercer Año de Bachillerato, como un plan piloto con respecto a los colegios de la ciudad de Latacunga (Ecuador).

Referencias

Abt, C. C. (1987). *Serious Games*. University Press of America. https://books.google.com.ec/books?id=axUs9HA-hF8C

Adell-Segura, J., Llopis Nebot, M. Á., Esteve-Mon, F. M., & Valdeolivas Novella, M. G. (2019). *El debate sobre el pensamiento computacional en educación*. http://repositori.uji.es/xmlui/handle/10234/189983

Aguilera-Ruiz, C., Manzano-León, A., Martínez-Moreno, I., del Carmen Lozano-Segura, M., & Yanicelli, C. C. (2017). El modelo flipped classroom. *International Journal of Developmental and Educational Psychology*, *4*(1), 261–266. https://www.redalyc.org/pdf/3498/349853537027.pdf

Aho, A. V. (2012). Computation and computational thinking. *The computer journal*, *55*(7), 832–835. https://ieeexplore.ieee.org/abstract/document/8130248/

Alice 2.X. (2020). *Alice – Tell Stories. Build Games. Learn to Program.* http://www.alice.org/

Alvarez, J., Rampnoux, O., Jessel, J.-P., & Methel, G. (2007). Serious Game: Just a question of posture. *Artificial & Ambient Intelligence, AISB*, *7*, 420–423. http://ja.games.free.fr/These/_perso%20Ecris/Articles%20propose%CC%81s/DIgra/DIGRA07_AlvarezRampnoux.doc

Amor, E. P. (2016). *Máster Universitario en Software y Sistemas Trabajo de Fin de Máster*. http://oa.upm.es/id/eprint/47134/contents

Aragón, M. T. (2018). *Aplicación del aprendizaje basado en proyectos para fomentar la creatividad en la asignatura de Tecnología*. Ice. http://oa.upm.es/53155/1/TFM_MERCEDES_TERRONES_ARAGON.pdf

Ayala, O. (2017). *Razonamiento: Lógico, Matemático, Inductivo, Deductivo, Abstracto*. https://drive.google.com/open?id=1EPlL974OSwMlMXyRInbaCFP2f5NNXZGM

Backlund, P., & Hendrix, M. (2013). Educational games-are they worth the effort? A literature survey of the effectiveness of serious games. *2013 5th international conference on games and virtual worlds for serious applications (VS-GAMES)*, 1–8.

Barbosa, A. F., Pereira, P. N., Dias, J. A., & Silva, F. G. (2014). A new methodology of design and development of serious games. *International Journal of Computer Games Technology*, *2014*. https://www.hindawi.com/journals/ijcgt/2014/817167/abs/

Barefoot. (2021). *Home Learning*. Barefoot. https://www.barefootcomputing.org/homelearning

bebras.org. (2017, octubre). *What is Bebras | www.bebras.org*. https://www.bebras.org/

Berabeu Soria, G. (2017). *(PDF) RECURSOS PARA EL AULA 100 problemas matemáticos*. https://www.academia.edu/28036859/RECURSOS_PARA_EL_AULA_100_problemas_matem%C3%A1ticos

Berenguer-Albaladejo, C. (2016). *Acerca de la utilidad del aula invertida o flipped classroom*. https://rua.ua.es/dspace/bitstream/10045/59358/1/XIV-Jornadas-Redes-ICE_108.pdf

Berrocoso, J. V., Sánchez, M. R. F., & Arroyo, M. del C. G. (2015). El pensamiento computacional y las nuevas ecologías del aprendizaje. *Revista de Educación a Distancia*, *46*.

Biró, P., Csernoch, M., Máth, J., & Abari, K. (2015). Measuring the level of algorithmic skills at the end of secondary education in Hungary. *Procedia-Social and Behavioral Sciences*, *176*, 876–883. https://www.sciencedirect.com/science/article/pii/S187704281500590X

Blake, J. D. (2011). Language considerations in the first-year CS curriculum. *Journal of Computer Science*, *26*(6), 124–129.

Blockly. (2021). *Google Developers*. https://developers.google.com/blockly/

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., & others. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice*. Joint Research Centre (Seville site). http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf

Bonar, J., & Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human–Computer Interaction*, *1*(2), 133–161. https://doi.org/10.1207/s15327051hci0102_3

Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes*, *41*(6), 1–6. https://www.researchgate.net/profile/Marco_Aurelio_Gerosa/publication/312142448_Why_is_programming_so_difficult_to_learn_Patterns_of_Difficulties_Related_to_Programming_Learning_Mid-Stage/links/5a39aecba6fdcc34776a38c4/Why-is-programming-so-difficult-to-learn-Patterns-of-Difficulties-Related-to-Programming-Learning-Mid-Stage.pdf

Braad, E., Žavcer, G., & Sandovar, A. (2016). Processes and models for serious game design and development. En *Entertainment computing and serious games* (pp. 92–118). Springer. https://doi.org/10.1007/978-3-319-46152-6_5

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American*

educational research association, Vancouver, Canada, *1*, 25. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Burgoa, J. A. B., Salvador, M. R. A., & Narváez, H. O. P. (2016). Complex thinking to computational thinking: Contemporary challenges in education. *Educación*, *21*(1), 143–159. https://doi.org/10.17163/soph.n21.2016.06

Cañas, J. J., Bajo, M. T., & Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies*, *40*(5), 795–811. https://doi.org/10.1006/ijhc.1994.1038

Chang, C.-S., Chung, C.-H., & Chang, J. A. (2020). Influence of problem-based learning games on effective computer programming learning in higher education. *Educational Technology Research and Development*, *68*(5), 2615–2634. https://link.springer.com/article/10.1007/s11423-020-09784-3

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology*, *12*(2), ep272.

Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, *50*(4), 31–34. http://www.ccis.northeastern.edu/home/lieber/courses/csu670/f08/materials/p31-chen-flow-in-games.pdf

Claro, M. (2010). *Impacto de las TIC en los aprendizajes de los estudiantes: Estado del arte*. https://repositorio.cepal.org/bitstream/handle/11362/3781/lcw339.pdf?sequence=1&isAllowe

Code School Finland. (2021). *Coding in Finnish curriculum • Code School Finland*. https://www.codeschool.fi/2019/04/finnish-curriculum/

Codelearn. (2021). *Codelearn.es | Escuela de programación, robótica y pensamiento computacional*. https://codelearn.es/

Code.org. (2021). *Code.org—Learn Computer Science*. https://studio.code.org/courses

CoderZ. (2021). *5 Best Programming Languages for Kids—CoderZ*. https://gocoderz.com/all-news/5-best-programming-languages-kids/

CodeSpark. (2017, octubre). *Coding App for Kids | codeSpark Academy*. https://codespark.com/

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., & Molina-Carmona, R. (2015). Enseñando a programar: Un camino directo para desarrollar el pensamiento computacional. *Revista de Educación a Distancia*, *46*.

Computer Science Education Research Group. (2017, octubre). *CS Unplugged*. Informática sin un ordenador. https://csunplugged.org/

Computing At School 2021. (s/f). *Computing at School*. Recuperado el 18 de febrero de 2021, de https://www.computingatschool.org.uk/

Corneliussen, H. G., & Tveranger, F. (2018). Programming in secondary schools in Norway: A wasted opportunity for inclusion. *Proceedings of the 4th Conference on Gender & IT*, 175–182. https://dl.acm.org/doi/pdf/10.1145/3196839.3196867

Corradini, I., Lodi, M., & Nardelli, E. (2017). Computational Thinking in Italian Schools: Quantitative Data and Teachers' Sentiment Analysis after Two Years of" Programma il Futuro". *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 224–229.

Csikszentmihalyi, M., & Csikzentmihaly, M. (1990). *Flow: The psychology of optimal experience* (Vol. 1990). Harper & Row New York.

https://mktgsensei.com/AMAE/Consumer%20Behavior/flow_the_psychology_of_optimal_experience.pdf

CSTA & ISTE. (2011). *Computational Thinking: Leadership toolkit*. https://cdn.iste.org/www-root/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4

Dale, N. B., & Weems, C. (2005). *Programming and problem solving with C++*. Jones & Bartlett Learning.

Dapozo, G. N., Petris, R. H., Greiner, C. L., Espíndola, M. C., López, M., & others. (2016). *Capacitación en programación para incorporar el pensamiento computacional en las escuelas*. XI Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET 2016).

De Gloria, A., Bellotti, F., & Berta, R. (2014). Serious Games for education and training. *International Journal of Serious Games*, *1*(1). https://doi.org/10.17083/ijsg.v1i1.11

De Lope, R. P., & Medina-Medina, N. (2017). A comprehensive taxonomy for serious games. *Journal of Educational Computing Research*, *55*(5), 629–672.

Deutsch, T. (2016). *DIGITAL EDUCATION STRATEGY OF HUNGARY Annex to the Government's Proposal*. https://digitalisjoletprogram.hu/files/0a/6b/0a6bfcd72ccbf12c909b329149ae2537.pdf

Díaz Tejera, K. I., Fierro Martín, E., & Muñoz Pentón, M. A. (2018). La enseñanza de la programación: Una experiencia en la formación de profesores de informática. *Educación*, *27*(53), 73–91.

Direção-Geral da Educação. (s/f). *Iniciação à Programação no 1.º Ciclo do Ensino Básico | ERTE*. Recuperado el 27 de febrero de 2021, de https://www.erte.dge.mec.pt/iniciacao-programacao-no-1o-ciclo-do-ensino-basico

Djaouti, D., Alvarez, J., & Jessel, J.-P. (2011). Classifying serious games: The G/P/S model. En *Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches* (pp. 118–136). IGI Global. https://www.igi-global.com/chapter/classifying-serious-games/52492

Dodge, D. (2019, diciembre 8). *Top 7 Kids Coding Languages of 2020 (everything you need to know)*. https://codakid.com/top-7-kids-coding-languages-of-2018/

Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*(1), 57–73. https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: Presenting computing concepts to novices. *International Journal of man-machine studies*, *14*(3), 237–249. https://doi.org/10.1016/S0020-7373(81)80056-9

Elodie. (2020, junio 26). *Programming and Robotics in Japanese schools: Learning the skills for tomorrow*. https://medium.com/@elodie_52864/programming-and-robotics-in-japanese-schools-learning-the-skills-for-tomorrow-53ac73072fb9

Elverdam, C., & Aarseth, E. (2007). Game classification and game design: Construction through critical analysis. *Games and Culture*, *2*(1), 3–22. https://journals.sagepub.com/doi/abs/10.1177/1555412006286892

Eppmann, R., Bekk, M., & Klein, K. (2018). Gameful experience in gamification: Construction and validation of a gameful experience scale [GAMEX]. *Journal of Interactive Marketing*, *43*, 98–115.

Etxeberría, F. (2012). Videojuegos: Riesgos y oportunidades en educación. *Actas I Congreso Internacional de Videojuegos y Educación*, 22–31.

Formas - Pruebas ser bachiller. (2017). *Razonamiento abstracto | Formas—Pruebas ser bachiller*. https://drive.google.com/open?id=1ymz_nVuZdBjv_Wo408exDYdzRSohYZCO

Fortea Bagán, M. Á. (2019). *Metodologías didácticas para la enseñanza/aprendizaje de competencias*. http://repositori.uji.es/xmlui/bitstream/handle/10234/182369/MDU1.pdf

Fuentes-Rosado, J. I., & Moo-Medina, M. (2017). Dificultades de aprender a programar. *Revista Educación en Ingeniería*, *12*(24), 76-82-76–82.

Fundación Omar Dengo. (2021, enero). *¿Quiénes somos? | FOD*. https://fod.ac.cr/quienes-somos/

FunJava – Lite- Laboratorio de Tecnologías de la Información en la Educación. (s/f). Recuperado el 16 de agosto de 2020, de http://lite.etsii.urjc.es/tools/funjava/

Furini, M. (2016). On Gamifying the Transcription of Digital Video Lectures. *Entertainment Computing*, *14*. https://doi.org/10.1016/j.entcom.2015.08.002

Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education (TOCE)*, *14*(2), 1–18. https://dl.acm.org/doi/abs/10.1145/2602483

Gamelearn Team. (2021). *Serious games examples that explain all you need to know*. https://www.game-learn.com/all-you-need-to-know-serious-games-game-based-learning-examples/

Garcia-Iruela, M., Hijón-Neira, R., & others. (2017). *Experiencia de juegos serios en el aula de formación profesional*. https://riull.ull.es/xmlui/bitstream/handle/915/6682/CIVE17_paper_17.pdf?sequence=1

Girard, C., Ecalle, J., & Magnan, A. (2013). *Serious games as new educational tools: How effective are they? A meta - analysis of recent studies. Journal of Computer Assisted Learning, 29(3)*. https://doi.org/10.1111/j.1365- 2729.2012.00489.x

Global News. (2021). *Enseñanza de la codificación en las escuelas canadienses: ¿Cómo se comparan las provincias? | Globalnews.ca*. https://globalnews.ca/news/3693932/teaching-coding-in-canadian-schools-how-do-the-provinces-measure-up/

Gobierno de Chile. (2021, enero). *¿Qué es el Plan Nacional de Lenguajes Digitales?* http://sitios.mineduc.cl/lenguajesdigitales/que-es-el-plan.html

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. *International Conference on Engineering Education–ICEE*, *7*. https://www.academia.edu/download/40987591/IC26_Learning_to_program_-_difficulties_and_solutions.pdf

Gomez, A. (2016, enero 22). *El aprendizaje de fundamentos de la programación es un juego de niños con The Foos*. https://codigo21.educacion.navarra.es/2016/01/22/el-aprendizaje-de-fundamentos-de-la-programacion-es-un-juego-de-ninos-con-the-foos/

Hanus, M. D., & Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & education*, *80*, 152–161. https://doi.org/10.1016/j.compedu.2014.08.019

Henne, G. (2015, febrero 8). *How We Taught Computer Science to 225,000 Students in Saudi Arabia*. https://www.edsurge.com/news/2015-02-08-how-we-taught-computer-science-to-225-000-students-in-saudi-arabia

Herald, K. (2014, julio 27). *Coding in schools*. http://www.koreaherald.com/view.php?ud=20140727000121

Herz, J. (1997). *Joystick nation: How videogames ate our quarters, won our hearts, and rewired our minds*. https://b-ok.lat/dl/2530674/9f33a7

Högberg, J., Hamari, J., & Wästlund, E. (2019). Gameful Experience Questionnaire (GAMEFULQUEST): An instrument for measuring the perceived gamefulness of system use. *User modeling and user-adapted interaction*, *29*(3), 619–660.

Hop´Toys. (2021). *Ratón robot code & go—HOPTOYS*. https://www.hoptoys.es/orientacion-espacio-temporal/raton-robot-code-go-p-12873.html

Hour of Code. (2017, octubre). *Hour of Code: Join the Movement*. Code.org. https://hourofcode.com/

Ibañez, M.-B., Di-Serio, A., & Delgado-Kloos, C. (2014). Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies*, *7*(3), 291–301. https://doi.org/10.1109/TLT.2014.2329293

Iberdrola. (2021). *Robotica Educativa: Definición, Ventajas y Ejemplos—Iberdrola*. https://www.iberdrola.com/innovacion/robots-educativos

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: A need assessment analyses. *TOJET: The Turkish Online Journal of Educational Technology*, *9*(2). http://tojet.net/articles/v9i2/9214.pdf

Jain, Y., & Sidhu, G. K. (2013). Relationship between anxiety, attitude and motivation of tertiary students in learning English as a second language. *Procedia-Social and Behavioral Sciences*, *90*, 114–123. https://doi.org/10.1016/j.sbspro.2013.07.072

Jansiewicz, D. R. (1973). *The New Alexandria simulation; a serious game of state and local politics*. Canfield Press; /z-wcorg/. https://archive.org/details/newalexandriasim00jans

Jara, I., & Hepp, P. (2016). *Enseñar Ciencias de la Computación: Creando oportunidades para los jóvenes de América Latina*. http://repositorio.minedu.gob.pe/handle/20.500.12799/5936

Kalas, I. (2015). Programming at pre-primary and primary levels: The pipeline can start that early. *KEYCIT 2014: Key competencies in informatics and ICT*, *7*, 29. https://books.google.com/books?hl=en&lr=&id=tr_FCgAAQBAJ&oi=fnd&pg=PA29&dq=Kalas,+I.+(2015).+Programming+at+pre-primary+and+primary+levels:+the+pipeline+can+start+that+early.+KEYCIT+2014:+Key+competencies+in+informatics+and+ICT,+7,+29&ots=CmAx2z6KlP&sig=d02RsqplKTSKRqEYisild-9-DfA

Khan, Q. (2020, febrero 11). *Overview of the Chinese Children's Programming Education Industry in 2019*. https://equalocean.com/analysis/2020021113558

Kodable. (2021). *Programming for Kids | Kodable*. https://www.kodable.com/

Kodu. (2021). *Kodu Game Lab | KoduGameLab*. http://www.kodugamelab.com/

Le Ministère de l'Éducation Nationale, de la jeunesse et des sports. (2015). *Socle commun de connaissances, de compétences et de culture*. Ministère de l'Education Nationale de la Jeunesse et des Sports. https://www.education.gouv.fr/bo/15/Hebdo17/MENE1506516D.htm

Learn today, build a brighter tomorrow. | *Code.org*. (s/f). Recuperado el 18 de febrero de 2021, de https://code.org/

LEGO System A/S. (2021). *LEGO® MINDSTORMS® EV3 31313 | MINDSTORMS® | Buy online at the Official LEGO® Shop US*. https://www.lego.com/en-us/product/lego-mindstorms-ev3-31313

LightBot Inc. (2018, febrero). *LightBot*. https://lightbot.com/

Logiscool Ltd. (2020). *Logiscool Belgique*. https://www.logiscool.com/be-fr/

López García, J. C. (2009). *ALGORITMOS Y PROGRAMACIÓN GUÍA PARA DOCENTES* (Segunda Edición). http://eduteka.icesi.edu.co/pdfdir/AlgoritmosProgramacion.pdf

López García, J. C. (2011). *PROGRAMACIÓN CON SCRATCH CUADERNO DE TRABAJO PARA ESTUDIANTES DE GRADOS 3° a 6°* (Cuarta Edición). http://eduteka.icesi.edu.co/pdfdir/AlgoritmosProgramacionCuaderno1.pdf

Luxton-Reilly, A. (2016). Learning to program is easy. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 284–289. https://doi.org/10.1145/2899415.2899432

Makeblock. (2021). *MBot | Makeblock*. https://www.makeblock.com/mbot

Massachusetts Institute of Technology. (2017, octubre). *Scratch—Imagine, Program, Share*. https://scratch.mit.edu/

McGill, T. J., & Volet, S. E. (1997). A conceptual framework for analyzing students' knowledge of programming. *Journal of research on Computing in Education*, *29*(3), 276–297. https://doi.org/10.1080/08886504.1997.10782199

Michael, D., & Chen, S. (2006). *Serious Games: Games That Educate, Train, and Inform*.

Michaud, L., & Alvarez, J. (2008). Serious games. *Advergaming, edugaming, training… IDATE Consulting & Research*. http://www.ludoscience.com/files/ressources/EtudeIDATE08_UK(1).pdf

Ministerio de Educación. (2019). *BALANCE DE GESTIÓN INTEGRAL AÑO 2018*. https://www.dipres.gob.cl/597/articles-188310_doc_pdf.pdf

Montes, H., Hijón-Neira, R., Pérez-Marìn, D., & Montes, S. (2021). Using an Online Serious Game to Teach Basic Programming Concepts and Facilitate Gameful Experiences for High School Students. *IEEE Access*, *9*, 12567–12578. https://doi.org/10.1109/ACCESS.2021.3049690

Montes León, H. (2020). *Fundamentos de Programación*. https://server.uehm-latacunga.edu.ec/dfd/story_html5.html

Montes-León, H., Hijón-Neira, R., Pérez-Marín, D., & León, S. R. M. (2019). Improving Programming Learning on High School Students through Educative Apps. *2019 International Symposium on Computers in Education (SIIE)*, 1–6. https://doi.org/10.1109/SIIE48397.2019.8970112

Montes-León, H., Montes-León, S., Pérez-Marín, D., & Hijón-Neira, R. (2020). *Mejora del Pensamiento Computacional en Estudiantes de Secundaria con Tareas Unplugged*. https://doi.org/10.14201/eks.23002.

Mora, D. E. L., Coloma, M. A. V., & Pino, Á. M. B. (2018). Uso de la metodología del aprendizaje basado en problemas en la enseñanza de la programación. *Pro Sciences: Revista de Producción, Ciencias e Investigación*, *2*(12), 12–16. http://www.journalprosciences.com/index.php/ps/article/view/69

Morrison, B. B., & Preston, J. A. (2009). Engagement: Gaming throughout the curriculum. *ACM SIGCSE Bulletin*, *41*(1), 342–346.

Moviltronics SAS. (2019). *Brazo Robotico OWI-535—Moviltronics*. https://moviltronics.com/tienda/brazo-robotico-owi-535/

Nadolski, R. J., Hummel, H. G., Van Den Brink, H. J., Hoefakker, R. E., Slootmaker, A., Kurvers, H. J., & Storm, J. (2008). EMERGO: A methodology and toolkit for developing serious games in higher education. *Simulation & Gaming*, *39*(3), 338–352. https://doi.org/10.1177/1046878108319278

Nickerson, R. S., & Smith, E. E. (1987). *Enseñar a pensar*. Ediciones Paidós Barcelona. https://cursos.aiu.edu/Desarrollo%20de%20Habilidades%20del%20Pensamiento/PDF/Tema%201.pdf

Ortega, M. V., Lozano, J. J. M., & Tristancho, S. L. Z. (2015). APPS en el rendimiento académico y autoconcepto de estudiantes de ingeniería. *Revista Logos Ciencia & Tecnología*, *6*(2), 198–208. https://doi.org/10.22335/rlct.v6i2.21

Papadakis, S. (2020). Evaluating a game-development approach to teach introductory programming concepts in secondary education. *International Journal of Technology Enhanced Learning*, *12*(2), 127–145.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2014). Novice programming environments. Scratch & app inventor: A first comparison. *Proceedings of the 2014 workshop on interaction design in educational environments*, 1–7. https://doi.org/10.1145/2643604.2643613

Papert, S. (1980). *Children, computers and powerful ideas*. http://www.medientheorie.com/doc/papert_mindstorms.pdf

Papert, S. (1996). An exploration in the space of mathematics educations. *Int. J. Comput. Math. Learn.*, *1*(1), 95–123. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.571.4630&rep=rep1&type=pdf

Pattanaphanchai, J. (2019). An Investigation of Students' Learning Achievement and Perception using Flipped Classroom in an Introductory Programming Course: A Case Study of Thailand Higher Education. *Journal of University Teaching and Learning Practice*, *16*(5), 4. https://files.eric.ed.gov/fulltext/EJ1237873.pdf

Petri, G., & von Wangenheim, C. G. (2017). How games for computing education are evaluated? A systematic literature review. *Computers & education*, *107*, 68–90. https://www.sciencedirect.com/science/article/pii/S0360131517300040

Pho, A., & Dinscore, A. (2015). *Game-Based Learning*. https://acrl.ala.org/IS/wp-content/uploads/2014/05/spring2015.pdf

Piaget, J. (1979). *El mecanismo del desarrollo mental*. Editora Nacional.

Piwek, P., & Savage, S. (2020). Challenges with learning to program and problem solve: An analysis of student online discussions. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 494–499. https://doi.org/10.1145/3328778.3366838

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of game-based learning. *Educational Psychologist*, *50*(4), 258–283.

Polo, R. (2013, agosto 16). *Tynker, introducción a la programación para los pequeños*. https://wwwhatsnew.com/2013/08/16/tynker-introduccion-a-la-programacion-para-los-pequenos/

Popescu, M.-M., & Bellotti, F. (2012). APPROACHES ON METRICS AND TAXONOMY IN SERIOUS GAMES. *eLearning & Software for Education*, *2*.

Python Software Foundation. (2021). *Welcome to Python.org*. https://www.python.org/

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 1–24. https://doi.org/10.1145/3077618

Ramos, E. & others. (2016). Iniciação à Programação no 1º Ciclo do Ensino Básico. Estudos de Avaliação. En *Estudos de Avaliação. Ministério da Educação-Direção Geral da Educação. 205pp. ISBN eletrónico*.

Realinfluencers. (2018). *8 metodologías que todo profesor del siglo XXI debería conocer*.

Robins, A. V. (2019). Novice programmers and introductory programming. *The Cambridge handbook of computing education research*, *1*, 327–376. https://www.otago.ac.nz/computer-science/otago706761.pdf

Robo Wunderkind. (2021). *Robo Wunderkind | Coding and Robotics for Kids*. https://www.robowunderkind.com/

Robot School. (2017). Robot School. Programming For Kids. *App Store*. files/11/id943154220.html

Robot Turtles. (2014). *Robot Turtles | The Board Game that Teaches Programming to Kids*. http://www.robotturtles.com/

Robson, K., Plangger, K., Kietzmann, J. H., McCarthy, I., & Pitt, L. (2015). Is it all a game? Understanding the principles of gamification. *Business horizons*, *58*(4), 411–420. https://doi.org/10.1016/j.bushor.2015.03.006

Román González, M. (2016). *Codigoalfabetización y pensamiento computacional en educación primaria y secundaria: Validación de un instrumento y evaluación de programas*. http://e-spacio.uned.es/fez/eserv/tesisuned:Educacion-Mroman/ROMAN_GONZALEZ_Marcos_Tesis.pdf

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2015). Test de Pensamiento Computacional: Diseño y psicometría general. *III Congreso Internacional sobre Aprendizaje, Innovación y Competitividad (CINAIC 2015)*.

Royal Society. (2012). *Shut down or restart?: The way forward for computing in UK schools*. Royal Society. https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Sabitzer, B., Antonitsch, P. K., & Pasterk, S. (2014). Informatics concepts for primary education: Preparing children for computational thinking. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 108–111. https://dl.acm.org/doi/abs/10.1145/2670757.2670778

SADLER, D. (2015, septiembre 20). *Coding to be taught in Australian schools from primary age*. https://mashable.com/2015/09/21/coding-schools-australia/

Salleh, F. H. M., Dewi, D. A., & Liyana, N. A. (2021). Issues and Challenges for Teaching Successful Programming Courses at National Secondary Schools of Malaysia.

Computational Science and Technology, *724*, 501. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7978907/

Savage, S., & Piwek, P. (2019). *Full report on challenges with learning to program and problem solve: An analysis of first year undergraduate Open University distance learning students' online discussions*. http://oro.open.ac.uk/68073/

Sawyer, B., & Smith, P. (2008). Serious games taxonomy. *Paper presented at the serious games summit at the game developers conference, San Francisco, USA*, 23–27.

ScratchJr. (2021). *ScratchJr—Home*. https://www.scratchjr.org/

Sharon, A. (2019, mayo 31). *Malaysia's Ministry of Education to introduce AI, robotics and computer programming in school*. https://opengovasia.com/malaysias-ministry-of-education-to-introduce-ai-robotics-and-computer-programming-in-school/

SIMULA. (2018, julio 2). *Simula educates teachers in programming*. https://www.simula.no/news/simula-educates-teachers-programming

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM*, *26*(11), 853–860. https://doi.org/10.1145/182.358436

Soloway, E., & Spohrer, J. C. (2013). *Studying the novice programmer*. Psychology Press. https://doi.org/10.4324/9781315808321

Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM*, *29*(7), 624–632. https://doi.org/10.1145/6138.6145

Suarez, A. M. G. (2017). La importancia del guion instruccional en el diseño de ambientes virtuales de aprendizaje. *Revista Academia y Virtualidad*, *10*(2), 47–60. https://doi.org/10.18359/ravi.2868

Sysło, M. M., & Kwiatkowska, A. B. (2015). Introducing a new computer science curriculum for all school levels in Poland. *International conference on informatics in Schools: Situation, evolution, and perspectives*, 141–154.

Tai, D. W., Yu, C.-H., Lai, L.-C., & Lin, S.-J. (2003). A study on the effects of spatial ability in promoting the logical thinking abilities of students with regard to programming language. *World Transactions on Engineering and Technology Education*, *2*(2), 251–254. http://www.wiete.com.au/journals/WTE&TE/Pages/Vol.2,%20No.2%20(2003)/Tai12.pdf

Tan, P.-H., Ting, C.-Y., & Ling, S.-W. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. *2009 International Conference on Computer Technology and Development*, *1*, 42–46. https://doi.org/10.1109/ICCTD.2009.188

Tan, Z. (2019, julio 11). *Singapore makes coding classes mandatory for primary school students, starting 2020*. https://kr-asia.com/singapore-makes-coding-classes-mandatory-for-primary-school-students-starting-2020

Terrapin. (2021). *Programming Journey Robots—Productos*. https://www.terrapinlogo.com/products/robots.html

The Royal Society. (2021, enero 19). *Computing at school*. https://royalsociety.org/blog/2021/01/computing-at-school/

The Straitstimes. (2020, diciembre 16). *Coding to be included in China's primary and secondary school curricula*. https://www.straitstimes.com/asia/east-asia/coding-to-be-included-in-chinas-primary-and-secondary-school-curricula

Tran, C., George, S., & Marfisi-Schottman, I. (2010). EDoS: An authoring environment for serious games. Design based on three models. *4th European Conference on Game-Based Learning*, 393–402. https://www.researchgate.net/profile/Sebastien_George/publication/268379008_EDoS_An_authoring_environment_for_serious_games_design_based_on_three_models/links/54cf406b0cf29ca810fdc888.pdf

TrendTIC. (2021). *Argentina:por ley, todas las escuelas deben enseñar programación en 2020—TrendTICTrendTIC*. https://www.trendtic.cl/2020/01/argentinapor-ley-todas-las-escuelas-deben-ensenar-programacion-en-2020/

Tynker. (2021). *Coding For Kids, Kids Programming Classes & Games | Tynker*. https://www.tynker.com/

Uskov, A., & Sekar, B. (2014). Serious games, gamification and game engines to support framework activities in engineering: Case studies, analysis, classifications and outcomes. *IEEE international conference on electro/information technology*, 618–623. https://doi.org/10.1109/EIT.2014.6871836

Wasserman, N. H., Quint, C., Norris, S. A., & Carr, T. (2017). Exploring flipped classroom instruction in Calculus III. *International Journal of Science and Mathematics Education*, *15*(3), 545–568. https://link.springer.com/article/10.1007/s10763-015-9704-8

Whitton, N. (2012). Games-Based Learning. En N. M. Seel (Ed.), *Encyclopedia of the Sciences of Learning* (pp. 1337–1340). Springer US. https://doi.org/10.1007/978-1-4419-1428-6_437

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2008.0118

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *Revista de Educación a Distancia*, *46*. https://revistas.um.es/red/article/view/240321

Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2013). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education*, *24*(2), 147–155. http://jise.org/Volume24/n2/JISEv24n2p147.pdf

Zúñiga, M. E., Rosas, M. V., Fernández, J., & Guerrero, R. A. (2014). El desarrollo del pensamiento computacional para la resolución de problemas en la enseñanza inicial de la programación. *XVI Workshop de Investigadores en Ciencias de la Computación*.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, *38*(9), 25–32.