



**Escuela Superior  
de Ingeniería y Tecnología**  
Universidad de La Laguna

# Trabajo de Fin de Grado

Ingeniería Informática

---

Sistemas software para monitorizar el  
entrenamiento personalizado del  
Pensamiento Computacional

*Software systems to monitor customized learning of  
Computational Thinking*

José Francisco Rodríguez Hernández

---

La Laguna, 09 de Septiembre de 2020

D. **Coromoto León Hernández**, con N.I.F. 78.605.216-W, profesora Catedrático de Universidad del área de Lenguajes y Sistemas Informáticos, adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

## **CERTIFICA (N)**

Que la presente memoria titulada:

*“Sistemas software para monitorizar el entrenamiento personalizado del Pensamiento Computacional”*

ha sido realizada bajo su dirección por D. **José Fco Rodríguez Hernández**, con N.I.F. 43.385.369-D.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 09 de septiembre de 2020

# Agradecimientos

Agradecer a mi madre Pili por todo el apoyo que siempre me ha dado que sin ella nada de esto podría haber sido posible.

A mi tío Ramón por ayudarme en los momentos más difíciles durante mis años de estudio.

A mis amigos por ser un apoyo incondicional y por todos los buenos momentos que hemos tenido.

A mi tutora Coromoto por motivarme a realizar este trabajo y ayudarme durante todo su desarrollo con su sabiduría y consejo.

# Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial 4.0 Internacional.

## Resumen

*El pensamiento computacional es la capacidad de resolver problemas haciendo uso de conceptos fundamentales de la informática, por lo que se ha decidido estudiar las plataformas que ofrecen enseñanzas en esta disciplina para saber si son aptas para entrenar a los usuarios que la utilizan.*

*Diversos estudios realizados han demostrado que los jóvenes entrenados en pensamiento computacional desarrollan habilidades analíticas, pensamiento crítico, mayor facilidad a la hora de afrontar problemas y les ayuda en sus estudios. por estos motivos junto a la tendencia incremental del uso de la tecnología en la vida diaria es interesante incluir alguna asignatura en el currículum educativo donde se entrene a los alumnos en pensamiento computacional.*

*El objetivo de este proyecto ha sido estudiar las plataformas disponibles actualmente para el desarrollo de pensamiento computacional y analizar la forma en la que realizan la trazabilidad de adquisición de conocimientos de los estudiantes. Además se ha realizado un caso de estudio de la implementación de un algoritmo de ordenación haciendo uso de estas herramientas.*

*No todas las plataformas estudiadas tienen un plan de estudios o curso de pensamiento computacional ni poseen un plan para el profesorado el cual lo guíe para realizar el seguimiento de sus alumnos a la hora de evaluar si los han adquirido tras realizar las actividades.*

**Palabras clave:** pensamiento computacional, plataformas, algoritmo de ordenación, programación por bloques, code.org, scratch, snap!, appinventor.

## **Abstract**

*Computational thinking is the solving problem ability making use of fundamental computing concepts. This project aims to study if those platforms are capable of teaching computational thinking to their users.*

*Several studies reveal that kids trained in computational thinking develop analytical skills, critical thinking, problem solving skills and how to face them and also it helps them in their studies. Because of all of this and being more and more common the use of technology everyday it would be great to include a subject that trains them in this discipline*

*The purpose of this project has been the study of the available platforms to study computational thinking and analyze how they trace users learning. Furthermore, a study case of searching algorithms has been developed with different platforms.*

*In conclusion not all the platforms studied have a curriculum or a learning plan neither do they have a teacher plan where they can track students learning so they could see if the students learnt the topic.*

**Keywords:** computational thinking, platforms, sorting algorithm, block coding, code.org, scratch, snap!, appinventor.

# Índice general

<b>Introducción</b>	<b>11</b>
Antecedentes y estado actual del tema	11
¿Qué es el Pensamiento Computacional?	12
Objetivos	14
Conceptos	15
<b>Plataformas</b>	<b>17</b>
Code.org	17
Alice	22
MIT App Inventor	26
Scratch	29
Snap!	31
Greenfoot	31
Agentsheets / Agentcubes	33
Pencil Code	34
Make World	35
<b>Casos de estudio: Algoritmos de ordenación.</b>	<b>36</b>
Algoritmos de ordenación	36
Definición	36
Ordenación por burbuja.	36
Casos de estudio	37
Code.org	38
Scratch	40
MIT App Inventor	42
Snap!	44
<b>Seguimiento Personalizado</b>	<b>45</b>
Seguimiento	45
Seguimiento personalizado	46
Code.org	46
Scratch	46
MIT App Inventor	46
Snap!	47

<b>Conclusiones y Líneas futuras</b>	<b>48</b>
Conclusiones	48
Líneas futuras	49
Implementación de un sistema software	49
<b>Summary and Conclusions</b>	<b>50</b>
<b>Presupuesto</b>	<b>51</b>
Presupuesto	51



# Índice de figuras

- 2.1 Cursos de pensamiento computacional en Code.org.
- 2.2 Cursos dentro del CS fundamentals.
- 2.3 Tabla con los tipos de tareas y cómo se evalúan.
- 2.4 Logo de Alice.
- 2.5 Logo de MIT App Inventor.
- 2.6 Logo de Scratch.
- 2.7 Logo de Snap!.
- 2.8 Logo de Greenfoot.
- 2.9 Greenfoot en modo Java.
- 2.10 Greenfoot en modo Stride.
- 2.11 Logo de Agents Sheets.
- 2.12 Logo de Pencil Code.
- 2.13 Logo de Make World.
  
- 3.1 Pseudocódigo del algoritmo de burbuja.
- 3.2 Traza de una iteración del algoritmo de burbuja.
- 3.3 Creación de un proyecto en Code.org.
- 3.4 Código implementado en Code.org.
- 3.5 Traza del algoritmo.
- 3.6 Barra de navegación principal de Scratch.
- 3.7 Código en Scratch.
- 3.8 Resultado de tras ejecutar el algoritmo en Scratch.
- 3.9 Barra de navegación de APP Inventor.
- 3.10 Código del algoritmo en App Inventor.
- 3.11 Resultado tras ejecutar el algoritmo en el Smartphone.
- 3.12 Algoritmo implementado en Snap!
- 3.13 Resultado tras ejecutar el algoritmo.

# Índice de tablas

1.1 Conceptos de Pensamiento Computacional.

7.1 Tabla de presupuestos para el proyecto.

# Capítulo 1 Introducción

El término Pensamiento Computacional es un tema candente en la actualidad, pero realmente ¿qué es el Pensamiento Computacional? Este concepto implica la capacidad de resolver problemas haciendo uso de conceptos fundamentales de la informática. Diversos estudios demuestran que las personas entrenadas en esta disciplina desarrollan distintas habilidades que le ayudan a resolver los problemas de su vida diaria, además de desarrollar pensamiento crítico y la capacidad de análisis.

A día de hoy existen diversas plataformas que ofrecen herramientas para entrenar este conjunto de habilidades mediante actividades de diferentes niveles de dificultad. El problema que surge con estas plataformas es que no disponen de un mecanismo que permita seguir el aprendizaje de cada estudiante de forma detallada. Por eso la pregunta que se va a abordar en este trabajo es: ¿Qué mecanismos debe tener un sistema software para permitir hacer una traza del seguimiento del aprendizaje de cada estudiante?

El objetivo de este trabajo es estudiar las distintas plataformas que se utilizan para la enseñanza del Pensamiento Computacional, reunir las características de cada una de ellas y analizar cómo realizan el seguimiento del aprendizaje de los estudiantes. A posteriori se realizará un caso de estudio de la implementación de un algoritmo haciendo uso de estas plataformas.

## 1.1 Antecedentes y estado actual del tema

En la Escuela Superior de Ingeniería y Tecnología, en los estudios del grado en Ingeniería Informática, se han realizado múltiples trabajos fin de grado relacionados con el pensamiento computacional. En los TFG [1][2][3][4] se realizan estudios sobre el estado actual de las herramientas de programación disponibles. En [1] se implementa una plataforma para medir la trazabilidad, errores, dificultades encontradas o cuáles fueron los conceptos más complejos de interpretar. En [2] y [3] se realizan estudios de las plataformas que se utilizan para la enseñanza del Pensamiento Computacional, las plataformas estudiadas han sido CODE.ORG, ALICE, SNAP!, GREENFOOT, APPINVENTOR y AGENTSHEETS. En [4] también se realiza un estudio de las distintas plataformas pero a diferencia de los TFG anteriores también se proponen dos casos de

estudio de algoritmos evolutivos con varias herramientas de programación visual (scratch y APPinventor). En [5] se ha desarrollado una plataforma de apoyo a través de Github para obtener estadísticas y resultados para que los profesores puedan analizar los resultados de los talleres impartidos. En [6] se utiliza Code.org para desarrollar un juego donde los alumnos tendrán que realizar un menú dietético equilibrado aplicando el Pensamiento Computacional. En [7] se realiza un estudio del estado actual del Pensamiento Computacional, como medirlo y un análisis de los diferentes métodos encontrados. Como segunda parte de este trabajo se desarrolla una plataforma que engloba un conjunto de herramientas necesarias para llevar a cabo el desarrollo y la medición del PC. En los trabajos [8][9][10] se ha desarrollado una plataforma web de programación donde los usuarios podrán desarrollar el comportamiento de un robot con sensores utilizando programación de bloques y posteriormente simular su comportamiento en el entorno desarrollado.

En relación con el Pensamiento Computacional, sus aplicaciones y beneficios si se enseña a tempranas edades se han escrito los siguientes artículos. En [11] se presenta un artículo que describe una nueva plataforma PENCIL CODE, que combina programación tradicional y programación de bloques para la enseñanza de PC, ayuda a los estudiantes mediante programación guiada pero también permite programación general mediante tutoriales. En [12] se realiza una revisión de la plataforma Make World, donde se enseña PC utilizando habilidades STEAM (Science, Technology, Engineering, Art and Math). En el artículo se muestran los resultados obtenidos y las dificultades encontradas por los alumnos al utilizar la plataforma. En el artículo [13] se presenta una plataforma accesible para niños ciegos tratando de introducir la programación visual de bloques y la enseñanza de PC a los niños con discapacidad visual. En el artículo [14] se hace una introducción interesante de una plataforma que utiliza la realidad virtual para la enseñanza del PC. Esta plataforma, CUBELY, está basada en el popular juego Minecraft, y presenta una propuesta inmersiva para aprender la programación orientada a bloques dentro de un entorno virtual 3D de una forma divertida y dinámica. Por último en [15] se presenta un proyecto con una propuesta colaborativa entre C++ y Code.org apoyando a los estudiantes en la creación de programas aumentando su confianza.

## **1.2 ¿Qué es el Pensamiento Computacional?**

El término Pensamiento Computacional (PC) es un tema candente en la actualidad, pero realmente ¿qué es el Pensamiento Computacional?. A lo largo del tiempo este ha recibido distintas definiciones generando discrepancia entre

los autores de estas definiciones contribuyendo a no llegar a un acuerdo con una definición.

Con afán de solventar esto los autores Jesús Moreno-León, Gregorio Robles, Marcos Román González y Juan David Rodríguez García han realizado una revisión bibliográfica [16] de las distintas definiciones que se han dado a lo largo del tiempo para intentar dar una definición que satisfaga a todos los autores. Tras la revisión llegaron a la siguiente proposición de definición: “CT would be the ability to formulate and represent problems to solve them by making use of tools, concepts and practices from the computer science discipline, such as abstraction, decomposition or the use of simulations. Such data-driven definition could be of interest for the educational community, since it clarifies the relationship between CT and programming (or robotics, for that matter), being the former a cognitive ability of the subject, and the latter just one of the means to develop it. “. En el artículo comentan que entre los términos más comunes de las definiciones hechas la palabra “coding” o programación no estaba presente, y se habla de que para enseñar PC no se debería necesitar aprender a programar si en un futuro se encuentra una manera para ello. No obstante esta definición deja clara la relación entre PC y programación siendo una habilidad cognitiva por la cual se puede desarrollar el PC pero no el único.

Tomando como referencia a Tasneem Raja [17] “If seeing the culinary potential in raw ingredients is like computational thinking, you might think of a software algorithm as a kind of recipe: a step-by-step guide on how to take a bunch of random ingredients and start layering them together in certain quantities, for certain amounts of time, until they produce the outcome you had in mind.” Aquí esta autora compara el PC con ver el potencial de platos que podemos preparar con alimentos que tengamos por casa. Salvando las distancias este es el principio en el que se basa la programación, se presenta un problema que a priori no tiene solución, compruebas que herramientas tienes a tu alcance para resolver dicho problema, divides el problema en subproblemas más pequeños los cuales haciendo uso de dichas herramientas podemos resolver de manera más sencilla, tras seguir unos pasos obtenemos una solución al problema inicial, ya sea un software o un plato de comida para saciar nuestro hambre.

Para dar una definición más concreta de lo que es Pensamiento Computacional utilizaremos la siguiente frase [17] “Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out

by an information-processing agent.” elaborada conjuntamente por Jeannete M. Wing, Jan Cuny y Larry Snyder. Esta definición da a entender al PC como un proceso mental por lo tanto es un atributo propio de los humanos y no de las máquinas tal y como comenta Jason Togyer, autor del artículo en cuestión. Este proceso mental conlleva la formulación de problemas y su resolución, no solo problemas matemáticos perfectamente definidos, también puede abarcar problemas cotidianos y la solución de los mismos es un sistema software muy complejo. Por lo que a partir de lo anterior podemos decir que el entrenamiento de esta disciplina puede ayudar a resolver problemas en nuestro día a día. Tal y como comenta Togyer cuando hacemos uso de PC intervienen otro tipos de pensamiento como el pensamiento lógico y el “systems thinking” que se centra en cómo funcionan los sistemas en el tiempo. Aparte de estos tipos de pensamientos intervienen muchos otros como el pensamiento paralelo y algorítmico que darán lugar a nuevos procesos mentales como búsqueda de patrones, pensamiento recursivo y procedural entre otros.

## 1.3 Objetivos

Como ya hemos comentado a día de hoy el Pensamiento Computacional es un tema de actualidad por lo que han surgido diversas plataformas dedicadas a la enseñanza de esta disciplina. Utilizan diversos métodos y técnicas para la enseñanza pero realmente cómo se comprueba que los estudiantes tras realizar un curso de una de estas plataformas hayan adquirido dichos conocimientos.

A partir de esta premisa surge la motivación de este TFG, es interesante saber cómo se miden la adquisición de los conocimientos para ver las similitudes y diferencias entre plataformas y cómo realizan la trazabilidad de la adquisición de los conocimientos

Los objetivos a conseguir en este TFG se describen a continuación:

- Realización de una revisión bibliográfica sobre el PC y las plataformas que ofrecen cursos basados en code.org.
- Estudio de la trazabilidad sobre la adquisición de conocimientos para cada plataforma.

- Implementación de un caso de estudio en diferentes plataformas y realizar una comparativa entre ellas.
- Documentación de la aplicación y elaboración de una memoria final donde se expondrán los resultados obtenidos.

## 1.4 Conceptos

En este apartado abordaremos los distintos conceptos a tratar dentro del PC aportando una definición para cada uno de ellos y su uso potencial o aplicación en la vida real.

Habilidad de PC	Definición
Abstracción	Abstracción es el proceso de hacer un objeto más entendible reduciendo las cosas innecesarias en él. La dificultad radica en la eliminación de las características correctas para que el problema se convierta en uno más simple sin perder nada que sea importante. Un aspecto clave en esta disciplina es elegir la representación correcta del sistema ya que diferentes representaciones hacen las cosas más fáciles.
Pensamiento Algorítmico	Pensamiento algorítmico consiste en llegar a una solución a través de un conjunto claro de pasos determinados en el orden establecido.
Automatización	La automatización es una tarea para ahorrar tiempo en la que un ordenador es entrenado para realizar una tarea repetitiva rápida y eficazmente comparado con el poder de procesamiento de un humano. En ese sentido las computadoras son "Automatizaciones de abstracciones"
Descomposición	Descomposición es una forma de pensar sobre los objetos en término de las partes que lo forman. Estas partes pueden ser entendidas, resueltas, desarrolladas y evaluadas de forma individual. Esto hace problemas complejos mucho más sencillos de resolver, situaciones novedosas entendibles fácilmente y grandes sistemas complejos más simples de diseñar.

Depuración	La depuración es la aplicación sistemática de análisis y evaluación haciendo uso de pruebas, seguimiento y pensamiento lógico para predecir y verificar los resultados.
Búsqueda de patrones, similitudes y conexiones	La identificación de patrones, similitudes y conexiones explotando esas funcionalidades. Es una forma rápida de resolver nuevos problemas basadas en soluciones de problemas anteriores aprovechando la experiencia previa. Haciendo preguntas como “¿Es este problema parecido a alguno que ya he resuelto?”. y “¿En qué se diferencia?” son importantes aquí, como es el proceso de reconocimiento de patrones tanto en los datos que se utilizan como en los procesos/estrategias que se usan. Los algoritmos que resuelven algunos problemas específicos pueden ser adaptados para resolver un tipo específico de problemas similares.
Programación	La programación es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un proyecto, preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados o elaborar programas para su empleo en computadoras.

Tabla 1.1. Conceptos de Pensamiento Computacional



# Capítulo 2 Plataformas

En este capítulo se hablará de las plataformas analizadas, si poseen algún tipo de curso, plan de estudio o tutoriales para enseñar pensamiento computacional, y de ser así cómo realizan el seguimiento del aprendizaje de los usuarios.

## 2.1 Code.org

Code.org es una plataforma online donde se pretende enseñar conceptos de Pensamiento Computacional a niños de distintas edades. Existen diferentes modalidades de cursos dependiendo del curso escolar de los estudiantes, pero todos ellos enfocados a estudios pre-universitarios.

Los cursos se dividen entre estudiantes de preescolar y primaria, teniendo una sección incluso para aquellos que aún no saben leer. Estudiantes de secundaria donde abordan distintos tópicos de PC y conceptos de informática en general. Por último existe un curso específico para los últimos años de secundaria y el Bachillerato. En la siguiente imagen se puede ver los distintos cursos que ofertan para los diferentes rangos de edad.

Elementary school						Middle school			High school			
K	1	2	3	4	5	6	7	8	9	10	11	12
									CS Principles			
						CS Discoveries						
CS Fundamentals												
Pre-reader Express		CS Fundamentals: Express										

Figura 2.1. Cursos de pensamiento computacional en Code.org

Los cursos que oferta Code.org se pueden dividir en dos grupos, un primer grupo serán los Express y un segundo grupo que llamaremos los de duración extendida.

Los cursos express son una versión condensada que abarca distintos tópicos de Pensamiento Computacional que sirven para dar una visión general a los estudiantes sobre el PC. Dentro de esta sección vemos dos cursos:

- Pre-reader Express: es un curso pensado para niños entre 4 y 8 años de edad que no necesariamente deben saber leer debido a que las actividades se basan principalmente en imágenes. No se necesitan conocimientos previos y tiene una duración entre 10 y 14 horas.

Los tópicos que se explican en este curso son:

- Secuenciación
  - Bucles
  - Eventos
- CS Fundamentals Express: este curso contiene todo los conceptos que se ven en el CS Fundamentals pero adaptado para niños de mayor edad. Esta modalidad está pensada para niños de entre 9 y 18 años, y tiene una duración estimada de 30 horas. Al igual que el resto de cursos este no necesita de conocimientos previos para participar en el.

Los tópicos que se explican en este curso son:

- Secuenciación
- Bucles
- Condicionales
- Funciones
- Variables
- Bucles for
- Sprites

Este curso cuenta con una actividad para introductoria y con proyecto final para que los alumnos pongan en práctica lo aprendido.

Por otra parte están los cursos de duración extendida, que son de larga duración pensados para realizarlos a largo plazo, debido a que se explican con más profundidad distintos tópicos de PC. Opcionalmente el profesorado que decida utilizar esta plataforma cuenta con una actividad introductoria donde se le explicarán los conceptos de PC que se van a tratar en la plataforma ya sea de forma intensiva durante 1 día de formación tanto presencial como online, además de tener soporte online en todo momento por las dudas que puedan surgir.

Los cursos que se encuentran disponibles actualmente son los siguientes:

- **CS Fundamentals** este curso está orientado a niños de entre 4 y 11 años de edad, corresponde a los cursos de primaria en España. Está compuesto por 6 cursos dependiendo del año escolar en el que se encuentren los estudiantes donde se abordan distintos tópicos los cuales tienen una duración de entre 10 y 25 horas cada uno.

Kindergarten	1 <sup>st</sup> Grade	2 <sup>nd</sup> Grade	3 <sup>rd</sup> Grade	4 <sup>th</sup> Grade	5 <sup>th</sup> Grade
Course A	Course B	Course C	Course D	Course E	Course F
Pre-Reader Express Course	Express Course				

Figura 2.2. Cursos dentro del CS fundamentals.

No es necesario tener conocimientos previos para curarlo y el profesorado dispone de soporte online y la posibilidad de realizar un curso intensivo de 1 día en el que se explican los contenidos del mismo.

Los tópicos que se explican en este curso son:

- Ciudadanía digital
  - Secuenciación
  - Bucles
  - Eventos
  - Impactos de la computación
  - Números binarios
  - Tratamiento de datos
  - Bucles anidados
  - Funciones
  - Variables
  - Sprites
  - Internet y su funcionamiento
- **CS Discoveries** es un curso introductorio donde introduce a estudiantes de la ESO distintos campos del PC haciendo uso de su propia creatividad, habilidad para la resolución de problemas y la comunicación. Este curso tiene una duración de entre 50 y 150 horas por lo que podría suponer una asignatura completa o adaptarse a un trimestre escolar.

Las 6 unidades que componen este curso son las siguientes:

1. Informática y resolución de problemas.
    - Técnicas de resolución de problemas aplicadas a puzzles, retos y escenarios de la vida real.
    - Cómo realizan los ordenadores operaciones de entrada, salida y almacenamiento de la información.
  2. Desarrollo web.
    - Estructura de una página web.
    - Lenguaje HTML y CSS.
    - Programación y depuración.
  3. Juegos e interacciones animadas.
    - Programación haciendo uso de sprites con GameLab.
    - Desarrollo de un programa interactivo.
  4. El proceso de diseño.
    - Impacto de la informática en la sociedad.
    - Identificación de requisitos para el desarrollo de aplicaciones.
    - Prototipado de aplicaciones tanto a mano como en digital.
    - Pruebas con usuarios finales.
  5. La sociedad y los datos.
    - Importancia de los datos en la resolución de problemas.
    - Sistemas para la representación de información.
    - Uso de colecciones de datos para resolver problemas.
  6. Informática física y hardware.
    - Introducción a diferentes tipos de hardware (sensores, ratones teclados, monitores ...).
    - Desarrollo de una aplicación que hace uso de dispositivos de E/S.
- **CS Principles** este curso cubre la mayoría de tópicos principales de PC además de abordar temas como el Internet, Big Data y privacidad, así como la programación y la algoritmia. Este curso tiene una duración de entre 100 y 180 horas y se debería de enseñar como una asignatura anual completa, CS Principles puede ser enseñado como un curso de Ubicación Avanzada o no, el cual se puede continuar durante los estudios universitarios posteriores accediendo a la opción de obtener títulos de las materias al respecto.

Al igual que el resto de cursos este no necesita de ningún conocimiento previo para poder participar en él. El temario que contiene es el siguiente:

1. Información Digital.
  - Representación de la información en medios digitales.
  - Cómo se representan los números, textos, imágenes y sonidos en formato de texto.
  - Compresión de datos.

- Impacto de la digitalización de la información.
- 2. El Internet.
  - Protocolos de red.
  - Funcionamiento general de internet.
  - Impacto en la sociedad.
- 3. Introducción al diseño de aplicaciones.
  - Introducción a la programación y diseño de una aplicación.
  - Depuración, programación con un compañero y test de usuario.
  - Diseño de interfaces y desarrollo de aplicaciones basadas en eventos.
- 4. Variables, condicionales y funciones.
  - Programación de aplicaciones complejas haciendo uso de variables, condicionales y funciones.
- 5. Listas, bucles y recorrido de árboles.
  - Desarrollo de aplicaciones con gran cantidad de datos.
- 6. Algoritmos.
  - Diseño de algoritmos para resolver problemas.
  - Análisis de la velocidad y eficiencia de los algoritmos.
- 7. Parámetros, librerías y retorno de funciones.
  - Desarrollo de aplicaciones haciendo uso de estos nuevos conceptos.
- 8. Crear una tarea de rendimiento.
  - Lecciones para aprender y crear un plan para llevarlo a cabo.
- 9. Tratamiento de datos.
  - Conversión de datos a información útil.
  - Búsqueda de patrones y visualización de los datos.
  - Machine learning.
- 10. Ciberseguridad e Impacto Global.
  - Impacto de la informática en la sociedad.
  - Privacidad y seguridad de nuestros datos.
  - Encriptación de la información.

La forma en la que Code.org realiza el seguimiento de sus estudiantes se rige mediante la siguiente tabla:

Level Type	Level Details			Level Status				
				Not started	In progress	Completed (too many blocks)	Completed (perfect)	Assessments / Surveys
Concept	Text	Video	Map			N/A		N/A
Activity	Unplugged Lesson Extras	Online Assessment	Question Choice level					

Tabla 2.3. Tabla con los tipos de tareas y cómo se evalúan.

En primer lugar debemos distinguir entre dos grupos las tareas de conceptos y las actividades. Este primer grupo lo forman todas aquellas tareas en las que se explica un concepto del temario que se esté cursando, este puede estar explicado mediante un texto, un vídeo o un mapa conceptual. La forma en las que se clasifican estas tareas es no comenzada, en progreso o terminada.

El segundo grupo lo forman las actividades donde se diferencian entre actividades desconectadas u online, preguntas, lecciones extras, evaluación o nivel de elección. Las actividades desconectadas proporcionan el material necesario para llevar a cabo la actividad en un aula. Para el resto de actividades es necesario el uso de un dispositivo con acceso a internet preferiblemente un ordenador. El seguimiento de estas actividades es el siguiente: no comenzada, en progreso, completada aquí hace distinción en si la solución que ha proporcionado el alumno es la más óptima o si por el contrario proporcionó una solución con pasos extras o innecesarios, por último están las lecciones de evaluación y encuestas en las que el profesor que esté impartiendo el curso deberá comprobar manualmente la respuesta proporcionada por los alumnos.

## 2.2 Alice



Figura 2.4. Logo de Alice.

Alice es otra plataforma que ofrece cursos sobre tópicos de Pensamiento Computacional haciendo uso de su propio entorno de desarrollo y lenguaje de programación por bloques.

En cuanto al entorno de desarrollo ofrece dos versiones Alice 2 y Alice 3, la diferencia principal es que Alice 2 provee un entorno con herramientas más simples y sencillas para el usuario final, esta versión es mejor para los estudiantes de primaria y secundaria, no obstante Alice 3 a pesar de ser más compleja provee más funcionalidades por lo que al final es una herramienta más potente que será más interesante a aquellos estudiantes que ya tengan conocimientos de PC o de programación.

Los recursos que ofrecen en la plataforma de Alice están divididos en las siguientes secciones:

- **How To 's** esta sección contiene un conjunto de recursos que explican las distintas secciones del entorno de desarrollo.

**Alice2:** en esta sección se ofrece un tutorial que te introduce a la aplicación y sus distintas partes, además existe otro curso introductorio para el editor de escenas del entorno de desarrollo.

**Alice3:** por otra parte esta versión de Alice cuenta con más tutoriales con respecto a la anterior. Dichos tutoriales cubren aspectos como el editor de escenas, el editor de código, animaciones para los personajes de las escenas, interactividad con la que se pueden añadir vistas en primera persona o un contador de puntos, audio para añadirle sonidos a la historia que se esté creando, programación en Realidad Virtual junto con la importación de modelos ya creados por otros usuarios, el reproductor de Alice para poder visualizar los proyectos creados y por último un apartado para saber como compartir los desarrollos creados en la plataforma.

- **Lessons** en este apartado se encuentran las lecciones sobre distintos tópicos de Pensamiento Computacional.

**Alice2:** esta versión cuenta con un único curso con una lección de una hora introductoria al mundo de Alice para que los alumnos tengan una primera toma de contacto con el entorno de desarrollo y realicen una historia animada.

**Alice3:** las lecciones que componen este apartado en su versión más actual son las siguientes:

- Getting started apartado que contiene lecciones básicas de cómo utilizar el entorno de desarrollo de Alice además de un hour of code introductorio.
- Methods, Procedures, Functions and Parameters donde se explica el uso de funciones, métodos y parámetros.
- Control Structures donde se da una introducción a las estructuras de control, además de explicar los condicionales, los bucles y las expresiones relacionales (AND, OR, ==, != >...)
- Events, Listeners, and Handlers en este apartado explican el funcionamiento de los eventos y cómo utilizarlos en distintas situaciones.
- Comments, Data Types, Arrays, and More esta sección introduce el uso de tipos de datos y cuales son, el uso de Arrays para almacenar

datos además de el uso de comentarios para ayudar a entender el código desarrollado. En último lugar tienen unas lecciones sobre expresiones aritméticas y relacionales.

- Design Process en este último apartado se explican los procesos de diseño y desarrollo de distintos tipos de proyectos como: historias interactivas, videojuegos o Realidad Virtual.

Además de las lecciones que proporcionan los creadores de Alice, la propia comunidad ha creado un repositorio[18] con distintas materias escolares, las cuales están pensadas para niños desde preescolar hasta bachillerato. Las actividades propuestas describen el ejercicio a realizar junto con el material necesario para llevarlo a cabo, existen actividades para ambas versiones.

Entre estas materias se encuentran:

- Matemáticas
  - Ciencia
  - Lenguas extranjeras
  - Arte
  - Historia/Estudios sociales
  - Inglés como primera lengua.
  - Inglés como segunda lengua.
  - Teatro
  - Negocios
  - Aplicaciones computacionales
- **Exercises & Projects** este apartado incluye los ejercicios propuestos tanto por los creadores de Alice como por la comunidad de profesores en las distintas convenciones que se han celebrado.

**Alice2** cuenta con un ejercicio introductorio donde se explican mediante videos o de manera escrita la puesta a punto del entorno para llevar a cabo los ejercicios que proponen posteriormente. Estos ejercicios tratan del desarrollo de una historia animada pensada para estudiantes de primaria donde deberán narrar las aventuras de Garfield.

**Alice3** por el contrario cuenta con numerosos proyectos distintos. En primer lugar tiene una sección con distintos tutoriales donde deben realizar un ejercicio práctico en la plataforma para lograr un objetivo, además cuenta con otras secciones donde se proponen ejercicios para la creación de escenas, la animación de personajes y los procesos de diseño.

Los recursos que nos proporcionan los creadores de Alice no son tan



extensos como en otras plataformas, sin embargo cuenta con una gran cantidad de recursos creados por la comunidad de profesores que han utilizado Alice a la hora de enseñar PC, además no se ofrecen directrices o un patrón a la hora de llevar a cabo una evaluación de las tareas, debido a que son creaciones de historias o aventuras dejan lugar a la creatividad del alumno en hacer sus propias propuestas.

- **Audio Library** se trata de una librería externa que se puede añadir como una extensión en el entorno de desarrollo de Alice que permite utilizar nuevas funciones para darle sonido a los proyectos que se creen. La extensión trae un conjunto de sonidos que se pueden utilizar para añadir realismo a las historias creadas además de numerosos sonidos de pequeñas animaciones comunes. Un aspecto más interesante de esta librería es que permite importar a la plataforma sonidos creados por el usuario por lo que da la opción a los alumnos de crear y editar sonidos propios únicos para cada uno de sus proyectos.
- **Curriculums** esta sección se encuentra vacía para la versión de alice2, sin embargo para la versión de alice3 la plataforma ofrece una tabla donde se explica cómo crear un curriculum para utilizando alice se enseñe cómo programar es la plataforma además de enseñar diferentes tópicos de Pensamiento Computacional. Esta tabla explica el orden de las lecciones, de qué tipo son, cuál es el objetivo de la lección, cuáles son los requisitos para poder cursarla...

En cuanto a tópicos de PC que se proponen en el modelo para crear un curriculum encontramos:

- Uso de comentarios en el código.
  - Creación de métodos procedurales, abstracción, jerarquía y herencia.
  - Parámetros y variables.
  - Uso de funciones.
  - Expresiones aritméticas.
  - Estructuras de control, e introducción a conceptos como secuenciación, concurrencia, condicionales y repetitivas.
  - Estructuras de control repetitivas y tipos de bucles.
  - Programación funcional.
  - Introducción a la programación orientada a eventos, explicación de los listeners y handlers. Tipos de eventos (proximidad y colisión).
  - Procesos de diseño.
- **Textbooks** en este apartado se recomiendan distintos libros y lecturas que consideran de utilidad a la hora de trabajar con el entorno de Alice.

Estos libros explican conceptos de programación y el uso del lenguaje de alicé con otros como por ejemplo Java.

## 2.3 MIT App Inventor



Figura 2.5. Logo de MIT App Inventor.

Esta plataforma provee un entorno de desarrollo con un lenguaje intuitivo, de programación visual, que permite a todo el mundo incluso a los niños a construir aplicaciones funcionales para los smartphones y tabletas. Este lenguaje está basado en programación orientada en bloques que simplifica el proceso de desarrollo consiguiendo resultados de manera sencilla y más rápida que con un lenguaje de programación convencional. El objetivo principal de esta plataforma es incentivar a la gente joven a ser creadores de aplicación en vez de ser meros consumidores.

En el apartado de profesores podemos ver los distintos cursos que ofrecen esta plataforma. En primer lugar hay un tutorial de como preparar el entorno de clase para los alumnos, aquí se explica como instalar App Inventor y configurarlo, los dos tipos de visualización que tiene la aplicación: diseño y bloques. En el modo de diseño se puede implementar la interfaz gráfica que se verá en el dispositivo móvil y por otra parte en el modo bloques se accede a las distintas sentencias que forman el lenguaje para crear la lógica de la aplicación. Los alumnos podrán ir probando las aplicaciones que creen mediante el emulador que trae integrado el entorno de desarrollo o por el contrario si poseen un dispositivo android se pueden conectar con el ordenador y probarla directamente en el móvil.

Esta plataforma contiene dos cursos principales:

- **Curriculum de PC:** este currículum está compuesto de 10 unidades de una duración aproximada de 50 horas, donde los estudiantes aprenderán a pensar computacionalmente mientras desarrollan aplicaciones para

dispositivos móviles. El curso está pensado para ser desarrollado en un mismo año y se puede modificar para que dure 6 semanas, 12 semanas o todo un trimestre para alumnos de la ESO o superiores. Los tópicos de pensamiento computacional que se imparten son:

- Descomposición de problemas (divide y vencerás).
- Abstracción y modularización.
- Reutilización y refactorización del código desarrollado y las soluciones implementadas.
- Testing, probar las aplicaciones desarrolladas y comprobar si la solución alcanzada es la deseada.

A lo largo del temario se le enseña al alumnado a desarrollar distintos tipos de aplicaciones como reproductores de música, distintos tipos de juegos interactivos, la creación de un piano y añadir los sonidos a cada una de las teclas entre otros; se le dan al alumnos distintas ideas para luego en un proyecto final desarrollen ellos una idea propia poniendo en práctica los conocimientos adquiridos.

- **Mobile CSP:** Este otro plan de estudios se trata de una introducción a los principios de la informática y el desarrollo profesional basado en la informática móvil. Este programa está basado en proyectos, donde los estudiantes pueden colaborar y hacer uso de su creatividad para llevarlos a cabo.

Al igual que el currículum de PC este curso está compuesto por 10 temas los cuales son:

- Dos cursos introductorios tanto al proyecto en sí, como a las aplicaciones móviles y la programación conjunta.
- Creación de imágenes y unidades gráficas paso a paso.
- Animación, simulación y modelado.
- Algoritmos y abstracción.
- Comunicación a través de internet, protocolos de red.
- Tratamiento y análisis de datos e información.

Además de los cursos explicativos incluyen exámenes que sirven como método de seguimiento para comprobar que los estudiantes han asimilado los conocimientos, en cuanto a las propias tareas como se hace apego a la creatividad no tienen una guía de evaluación simplemente comprobar que los alumnos realizan las actividades.

A parte de los cursos principales podemos encontrar en la web de la plataforma los siguientes recursos disponibles que ayudarán a los profesores a llevar a cabo los cursos y las creaciones que deseen:

- Hora de código, ofrecen un programa de una hora de duración estimada donde se le introduce al alumno al entorno de desarrollo para que lleve a cabo un ejercicio que luego podrán probar en un dispositivo móvil o tablet.
- Tutoriales, esta sección explican en detalle cómo realizar distintas tareas respecto a distintos tópicos, haciendo uso de bases de datos, NFC, juegos, etc... Estos tutoriales están graduados por dificultad para que los profesores o los estudiantes puedan guiarse sencillamente.
- Inteligencia Artificial: en este apartado se introducirá al alumno al mundo de la inteligencia artificial de una manera sencilla para despertar la curiosidad en esta disciplina en auge. Los cursos que componen este apartados son varios:
  - Curriculum introductorio a la IA, pensado para estudiantes de la ESO o superior donde aprenderán las nociones básicas del Machine Learning en concreto crearán una aplicación para la clasificación de imágenes.
  - Clasificador de imágenes personalizado: este curso es la siguiente versión del anterior, aquí los estudiantes crearán un modelo personalizado donde puede identificar y clasificar imágenes. Deberán probar el modelo que han creado y por último lo utilizarán para crear un juego.
  - Tutoriales sobre la creación de un bot terapéutico el cual responderá a los textos que introduzcas. Además del tutorial anterior hay otro en el cual los estudiantes deberán crear un modelo que sea capaz de aprender a jugar a piedra, papel o tijeras.

En general el seguimiento de las actividades es bastante libre, debido a que la temática de los ejercicios es una idea y se deja en el alumnado la decisión de cómo quieren implementarlo haciendo uso de la creatividad y los conceptos, no obstante, en el currículum de Mobile CSP está incluido en el temario una serie de exámenes con los que se pretende evaluar si el alumnado ha adquirido los conocimientos que se explican en los temas propuestos.

## 2.4 Scratch



Figura 2.6. Logo de Scratch.

Scratch es una plataforma que ofrece un entorno de desarrollo haciendo uso de lenguaje basado en bloques donde los estudiantes podrán crear aplicaciones interactivas, crear historias animando personajes o desarrollar juegos entre otras cosas. Los recursos que ofrece la plataforma para los profesores son los siguientes:

- **Guía para educadores:** esta guía contiene un conjunto de ejercicios de Hora de código. Estos ejercicios tienen una guía previa para el profesor por lo que no necesariamente tiene que ser un experto para realizarlo con su clase, además se añade una guía para los estudiantes paso a paso de como realizar el ejercicio con ejemplos gráficos. Se recomienda al profesor realizar un ejemplo antes de empezar para que los alumnos tengan una referencia a seguir e incentivarlos a completar la tarea.
- **Plan de estudios de informática creativa:** este currículum ha sido creado por el departamento de computación de la universidad de Harvard, se trata de una introducción a los alumnos a distintos tópicos de pensamiento computacional. El temario del curso se puede dividir en lecciones únicas de un tema en concreto o se puede enseñar como un curso completo. Las unidades son las siguientes:
  - Unidad 0. Introducción. En esta unidad se introduce tanto al profesor como a los alumnos al entorno de scratch, la creación de cuentas de usuarios y de proyectos.
  - Unidad 1. Explorando. En este tema se hace una introducción a los alumnos al tópico de secuenciación, donde deberán identificar y especificar una secuencia ordenada de instrucciones. Además se deberán implementar dichos ejercicios haciendo uso del editor de

código de scratch.

- Unidad 2. Animaciones. Una vez que los estudiantes han aprendido sobre el tópico de secuenciación se le introducen los conceptos de bucles, eventos y paralelismo además del uso de videos para animar sus creaciones.
- Unidad 3. Historias. En la creación de historias los estudiantes deberán trabajar en conjunto para poner en práctica los conocimientos adquiridos en los temas anteriores, haciendo especial hincapié en los eventos y el paralelismo. Además podrán poner en práctica las iteraciones de código haciendo uso de la refactorización y combinación de los scripts creados anteriormente.
- Unidad 4. Juegos. Los estudiantes en esta unidad tomarán el rol de diseñadores de videojuegos para crear su propio proyecto. Se introducirá al alumnado a las mecánicas de los videojuegos y el desarrollo donde se les explicarán conceptos como condicionales, operadores, procesamiento de datos y el uso de variables además de abstracción y modularización.
- Unidad 5. A fondo. Al contrario que en las unidades anteriores en esta se pretende revisar y reflexionar sobre el trabajo realizado. Las actividades que la componen son flexibles y su objetivo es volver a realizar los retos que han hecho anteriormente, aprender nuevas habilidades o refinar metodologías aprendidas.
- Unidad 6. Hackathon[19]. Esta es la unidad final del plan de estudios y los estudiantes tienen la posibilidad de crear un proyecto a su elección. Utilizan el concepto de hackathon donde los alumnos en un entorno colaborativo aprenderán a resolver problemas y realizar un desarrollo iterativo donde pueden planificar, implementar y compartir sus resultados.

Analizando el seguimiento de las actividades que se plantean en scratch realmente no proponen ningún método de evaluación, esto se debe a que las tareas proponen un objetivo y el alumno debe de hacer uso de su creatividad para alcanzar dicho objetivo, por lo que será tarea del profesor llevar a cabo el seguimiento de las tareas y como debe de realizarlas.

## 2.5 Snap!



Figura 2.7. Logo de Snap!.

Snap! Es una reimplementación de scratch donde el programador puede construir sus propios bloques además de otras funcionalidades extras, ofrece un entorno de desarrollo en un navegador web, donde los usuarios podrán registrarse y de manera sencilla compartir sus proyectos con el resto del mundo. En la página principal se pueden visitar las creaciones de otro usuarios, ir al entorno de desarrollo o visitar el manual de referencia.

En esta plataforma no ofrece ningún curso, tutorial o plan de estudios simplemente ofrece un extenso manual de referencia donde se explican distintos conceptos tanto de pensamiento computacional como de programación. Se explican todos los recursos disponibles en la plataforma y cómo utilizarlos.

## 2.6 Greenfoot



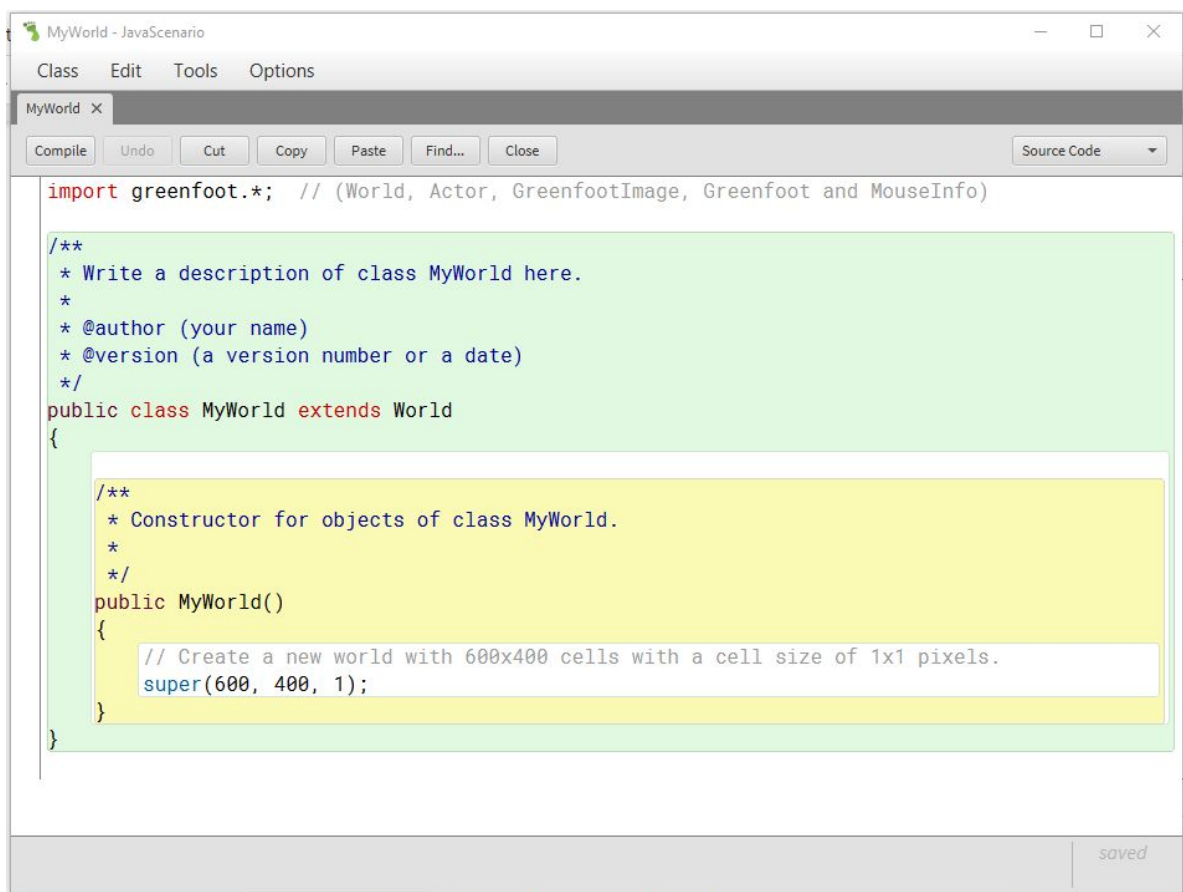
Figura 2.8. Logo de Greenfoot.

GreenFoot es un software que ofrece un entorno de desarrollo basado en Java donde se pretende enseñar programación orientada a objetos haciendo

uso de un entorno visual donde los estudiantes podrán relacionar distintos objetos y ver la interacción entre ellos.

En esta plataforma no se ofertan ningún curso, o plan de estudio pero sí poseen distintos tutoriales en los que iniciarnos en el uso de la aplicación, como instalarla, como crear distintos escenarios y actores, implementar comportamientos para los actores...

Greenfoot ofrece dos modalidades de desarrollo o tipos de escenarios: Stride y Java. La principal diferencia entre ambos es en el editor de código, en el modo Java se puede ver que es un editor de código convencional donde se pueden editar las clases que se creen, por el contrario en el modo Stride se pueden ver las distintas secciones pretendiendo ser más sencillo e intuitivo desarrollar en este modo.



```
MyWorld - JavaScenario
Class Edit Tools Options
MyWorld x
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class MyWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MyWorld extends World
{
    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public MyWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

saved

Figura 2.9. Greenfoot en modo Java.



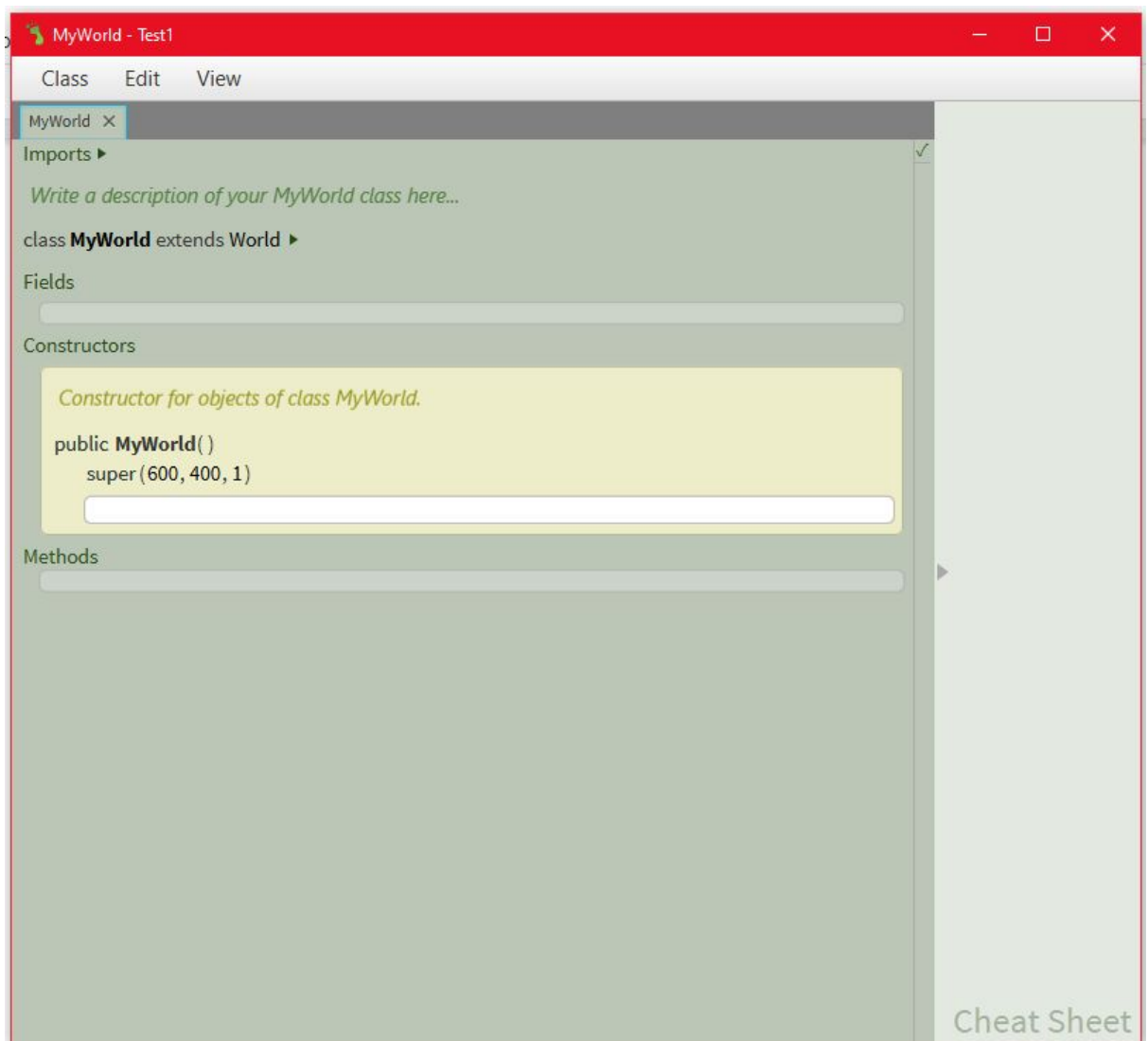


Figura 2.10. Greenfoot en modo Stride.

## 2.7 Agentsheets / Agentcubes



Figura 2.11. Logo de Agents Sheets.

AgentsCubes es un entorno de desarrollo 3D basado en un lenguaje de programación por bloques donde los estudiantes pueden programar el comportamiento de los agentes en un escenario para crear simulaciones, juegos e historias entre otros.

En esta plataforma no se oferta ningún curso de PC ni ningún plan de estudios, sin embargo, los tópicos principales que se ponen en práctica en la plataforma son: abstracción, los estudiantes deben dividir los problemas en patrones de PC, automatización, implementar los comportamientos de los distintos agentes y análisis, por último los alumnos pueden comprobar si lo que han desarrollado cumple con las ideas iniciales. Por otra parte se ofertan distintos tutoriales para aprender a utilizar la plataforma, entre esos tutoriales hay un Hora de código inicial que te enseña paso a paso a desarrollar un ejercicio práctico.

## 2.8 Pencil Code



Figura 2.11. Logo de Pencil Code.

Pencil Code es una plataforma de desarrollo colaborativa para realizar creaciones artísticas, reproducir música y crear videojuegos, a parte de todo esto se puede experimentar con funciones matemáticas, geometría, gráficos, páginas web, simulaciones y algoritmos. Todos los programas desarrollados en la plataforma son de acceso libre y al alcance de todos.

El lenguaje principal que se utiliza en esta plataforma es Coffeescript [<https://coffeescript.org/>], que es un lenguaje de programación que se compila a JavaScript, que pretende simplificar la experiencia de programación. Además se puede utilizar en la plataforma programación por bloques. A su vez esta plataforma se puede utilizar para aprender Javascript, HTML y CSS, ya que puede ajustarse el editor de texto para utilizar dichos lenguajes.

Pencil Code no oferta ningún plan de estudios o cursos explícitos de Pensamiento Computacional, sin embargo como hemos visto en las otras plataformas ofrecen distintos tutoriales donde se explican paso a paso y en detalle cómo realizar distintas tareas, además se puede visitar el trabajo de otras personas y ver cómo la han desarrollado y tomar ideas para tus propias implementaciones.

## 2.9 Make World



Figura 2.12. Logo de Make World.

Make World es una plataforma que ofrece un entorno de desarrollo que principalmente proporciona herramientas y metodologías para el aprendizaje de Ciencia, Tecnología, Ingeniería y Matemáticas, además de enseñar distintos tópicos de Pensamiento Computacional como secuenciación, abstracción y análisis entre otros. Los alumnos podrán ver los trabajos de los otros usuarios además de exponer los suyos propios en la plataforma.

Make World no posee ningún curso ni un plan de estudio donde se expliquen distintos conceptos de PC pensados para un curso académico. Existe un manual al alcance de los profesores en los que se pueden ver ejemplos de ejercicios para llevar a cabo en una clase donde se explican en detalle cómo realizarlos, las competencias que se ejercitan, los conocimientos necesarios y el tiempo que se tarda en realizar.

# Capítulo 3 Casos de estudio: Algoritmos de ordenación.

En este punto se nombrará el concepto de algoritmos de ordenación y su utilidad. Posteriormente se realizará un caso de estudio de uno de estos algoritmos en varias plataformas seleccionadas.

## 3.1 Algoritmos de ordenación

### 3.1.1 Definición

En [computación](#) y [matemáticas](#) un **algoritmo de ordenamiento** es un [algoritmo](#) que pone elementos de una [lista](#) o un [vector](#) en una secuencia dada por una [relación de orden](#), es decir, el resultado de salida ha de ser una [permutación](#) —o reordenamiento— de la entrada que satisfaga la relación de orden dada. Las relaciones de orden más usadas son el orden numérico y el orden lexicográfico[20].

## 3.2 Ordenación por burbuja.

El algoritmo de ordenación por burbuja es dentro de esta categoría uno de los algoritmos más sencillos. Se trata de un algoritmo iterativo que va revisando pares de elementos de la lista y comprobando si están correctamente colocados, en caso erróneo intercambian sus posiciones. Es necesario revisar la lista varias veces hasta que no se necesiten más intercambios lo cual significa que la lista está ordenada.

Existen varias implementaciones de este algoritmo con ciertas mejoras en su rendimiento, como la colocación de un centinela o un condicional que si no se realiza ninguna modificación termina el algoritmo sin necesidad de realizar todas las iteraciones, pero para este caso en concreto se ha decidido la implementación más sencilla que podemos ver en la siguiente imagen:

```

procedimiento DeLaBurbuja ( $a_0, a_1, a_2, \dots, a_{n-1}$ )
  para  $i \leftarrow 1$  hasta  $n - 1$  hacer
    para  $j \leftarrow 0$  hasta  $n - i - 1$  hacer
      si  $a_{(j)} > a_{(j+1)}$  entonces
         $aux \leftarrow a_{(j)}$ 
         $a_{(j)} \leftarrow a_{(j+1)}$ 
         $a_{(j+1)} \leftarrow aux$ 
      fin si
    fin para
  fin para
fin procedimiento

```

Figura 3.1. Pseudocódigo del algoritmo de burbuja.

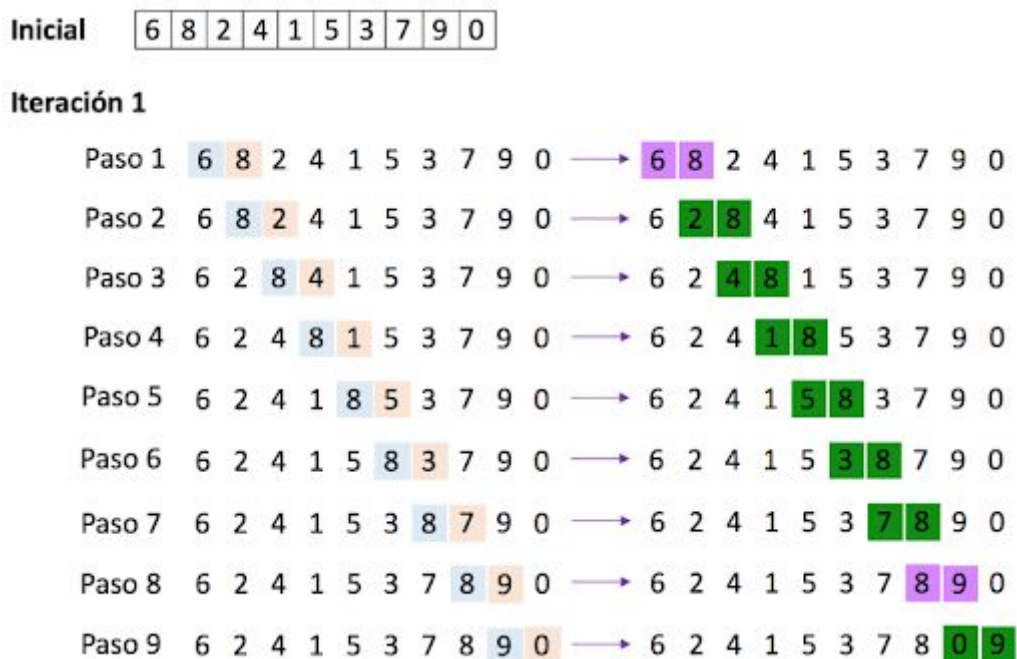


Figura 3.2. Traza de una iteración del algoritmo de burbuja.

### 3.3 Casos de estudio

Las plataformas elegidas para realizar la implementación del algoritmo han sido las siguientes:

- Code.org
- Scratch

- MIT App Inventor
- Snap!

El motivo por el que las otras plataformas no han sido elegidas es debido a que principalmente son entornos de programación orientados a desarrollar historias visuales, comportamientos de actores, o realizar gráficos y no están pensadas para implementar un algoritmo de este tipo.

Se ha decidido utilizar en todos los casos de estudio la misma lista a ordenar, se trata de un array de 10 elementos desordenados, que es la siguiente:

***shortList = [ 50, 22, 3, 11, 17, 70, 99, 1, 30, 85]***

### 3.3.1 Code.org

Para implementar el algoritmo en la plataforma de code.org fue muy sencillo, simplemente basto con ir a la esquina superior derecha y crear un nuevo “AppLab”, este tipo de proyecto sirve para simular la pantalla de un teléfono móvil donde se pueden controlar distintos elemento de la interfaz del móvil mediante componentes propios de HTML.

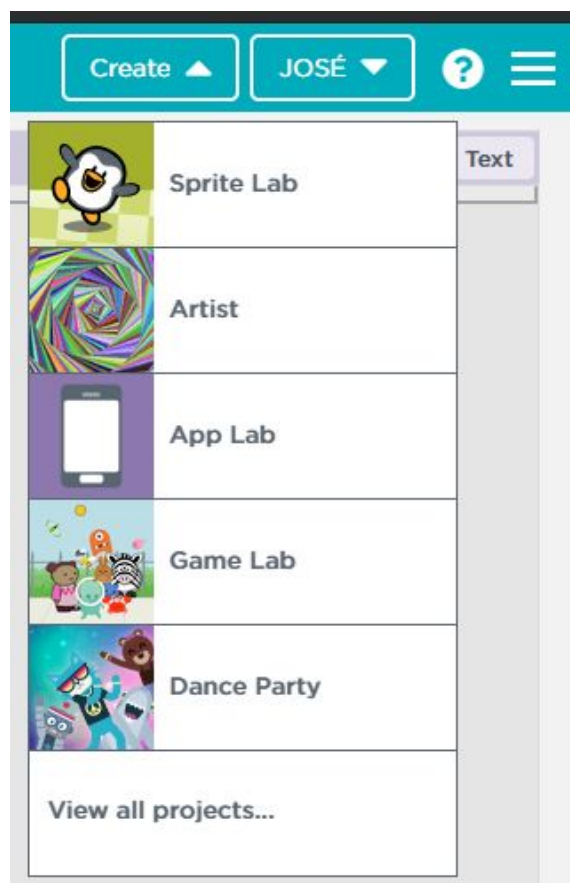


Figura 3.3. Creación de un proyecto en Code.org.

La implementación del algoritmo fue bastante sencilla, la interfaz de desarrollo es muy intuitiva y tiene todos los recursos a mano, en la siguiente imagen podemos ver el código hecho en la plataforma.

```
1 var list = [50, 22, 3, 11, 17, 70, 99, 1, 30, 85];
2 write("Inicial -> " + list);
3 var n = list.length;
4 for (var i = 1; i <= n-1; i++) {
5   for (var j = 0; j <= (n - (i - 1)); j++) {
6     if (list[j] > list[j+1]) {
7       var aux = list[j];
8       list[j] = list[j+1];
9       list[j+1] = aux;
10    }
11  }
12  write(i + " -> " + list);
13 }
14 write("Final -> "+list);
15
```

Figura 3.4. Código implementado en Code.org.

En esta plataforma no tuve ningún problema para implementar el algoritmo, de hecho fue realmente rápido y sencillo codificarlo y ver los resultados. El entorno de desarrollo poseía todos los elementos necesarios para llevarlos a cabo.

```
Inicial -> 50,22,3,11,17,70,99,1,30,85
1 -> 22,3,11,17,50,70,1,30,85,99
2 -> 3,11,17,22,50,1,30,70,85,99
3 -> 3,11,17,22,1,30,50,70,85,99
4 -> 3,11,17,1,22,30,50,70,85,99
5 -> 3,11,17,22,30,50,70,85,99
6 -> 3,11,17,22,30,50,70,85,99
7 -> 1,3,11,17,22,30,50,70,85,99
8 -> 1,3,11,17,22,30,50,70,85,99
9 -> 1,3,11,17,22,30,50,70,85,99
Final -> 1,3,11,17,22,30,50,70,85,99
```

Figura 3.5. Traza del algoritmo.

### 3.3.2 Scratch

En cuanto a la implementación del algoritmo en scratch se ofrecen dos modalidades para utilizar el entorno de desarrollo, la primera se realiza online desde un navegador web donde podremos acceder al mismo directamente, y una segunda modalidad desde nuestro ordenador en local si se descarga la aplicación disponible en la web para ello, la ventaja de esta segunda modalidad es q se puede utilizar sin necesidad de conexión a internet. En ambos casos el entorno de desarrollo es el mismo y da las mismas posibilidades.

Para crear un nuevo proyecto en Scratch basta con pulsar el botón en la barra principal de create, y nos redirigirá al entorno de desarrollo vacío.



Figura 3.6. Barra de navegación principal de Scratch.

A diferencia de code.org la implementación del algoritmo en esta plataforma no fue tan sencillo y esto se debe a que no existe una estructura de control que realice un bucle for, por lo que se tuvo que simular su comportamiento haciendo uso de las estructuras de repetición disponibles en la plataforma, en la siguiente imagen se puede ver el código implementado:



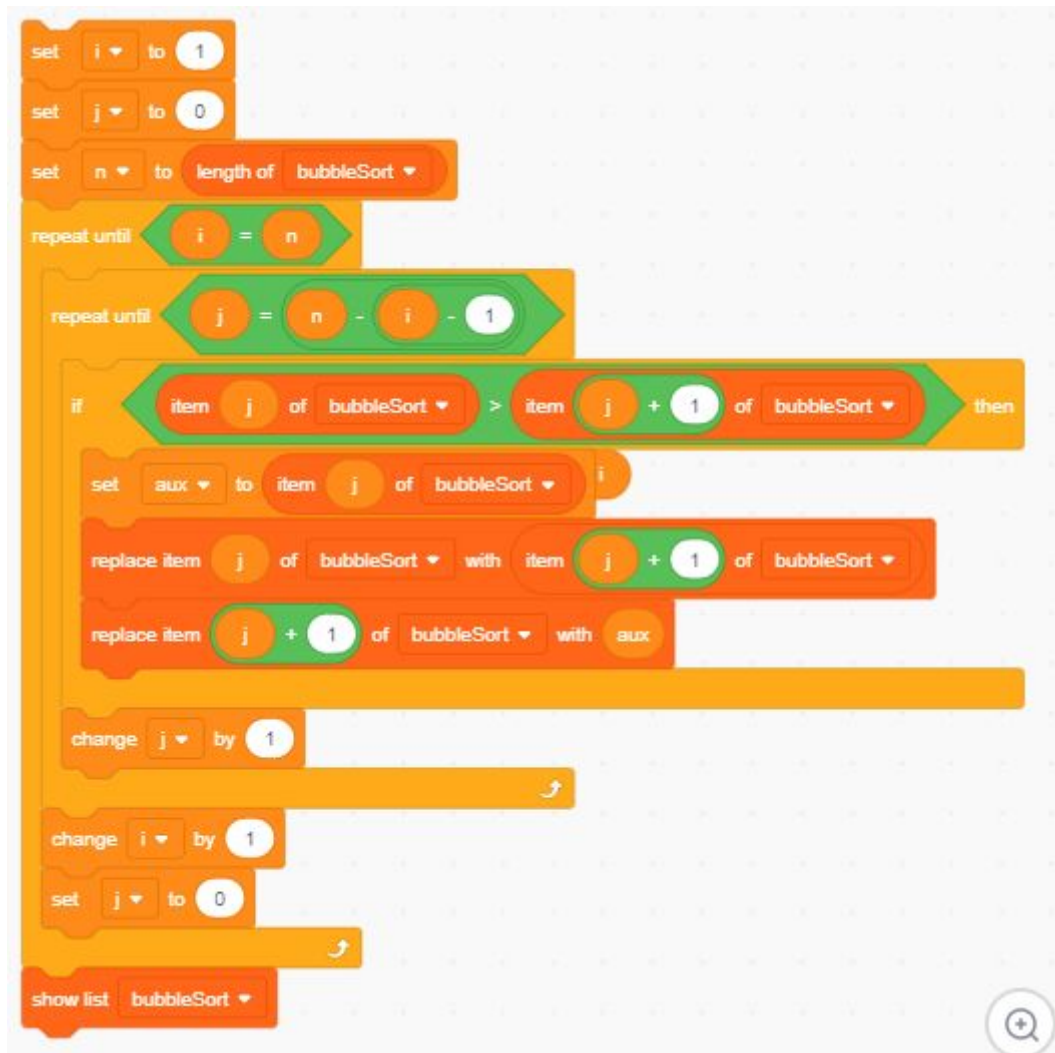


Figura 3.7. Código en Scratch.

En esta plataforma las variables se declaran en el menú de la izquierda en vez de directamente en el código. En esta plataforma fue más complejo por el hecho de que no existe un bucle for como tal y se tuvo que simular su comportamiento con los **repeat until** y haciendo uso del bloque **change [variable] by x (amount)**, este último bloque se utiliza para incrementar las variables *i* y *j*. A pesar de no existir un bloque explícito para realizar el bucle for se pudo implementar el algoritmo.

En la siguiente imagen podemos ver el resultado tras ejecutar el algoritmo sobre la lista inicial:

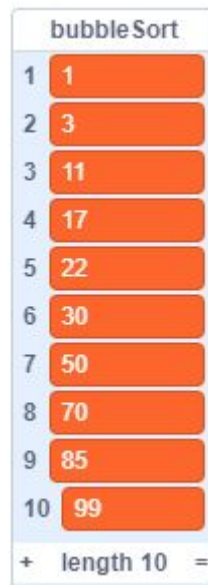


Figura 3.8. Resultado de tras ejecutar el algoritmo en Scratch.

### 3.3.3 MIT App Inventor

Esta plataforma es un tanto especial debido a que está pensada para desarrollar aplicaciones para dispositivos móviles o tablets. Para crear un proyecto nuevo simplemente se debe acceder a la página principal y hacer click sobre el botón de Create Apps!.



Figura 3.9. Barra de navegación de APP Inventor.

Al igual que en Code.org fue bastante sencillo implementar el algoritmo debido a que posee todas los bloques necesarios para ello y además pude compilar la aplicación generada y probarla en mi propio smartphone, esta es una peculiaridad interesante que hablaré luego sobre ella.

En la siguiente imagen podemos ver el código del algoritmo en esta plataforma, donde se ve que la declaración de las variables se realiza directamente en el código. Debido a que todos los desarrollos en esta plataforma van asociados a un evento una función se decidió que al pulsar un botón se ejecutara el algoritmo y mostrase en la pantalla el resultado.

Es interesante la manera en la que esta plataforma obtiene el contenido de las distintas posiciones de las listas, el bloque sigue una estructura que recuerda a una consulta SQL.

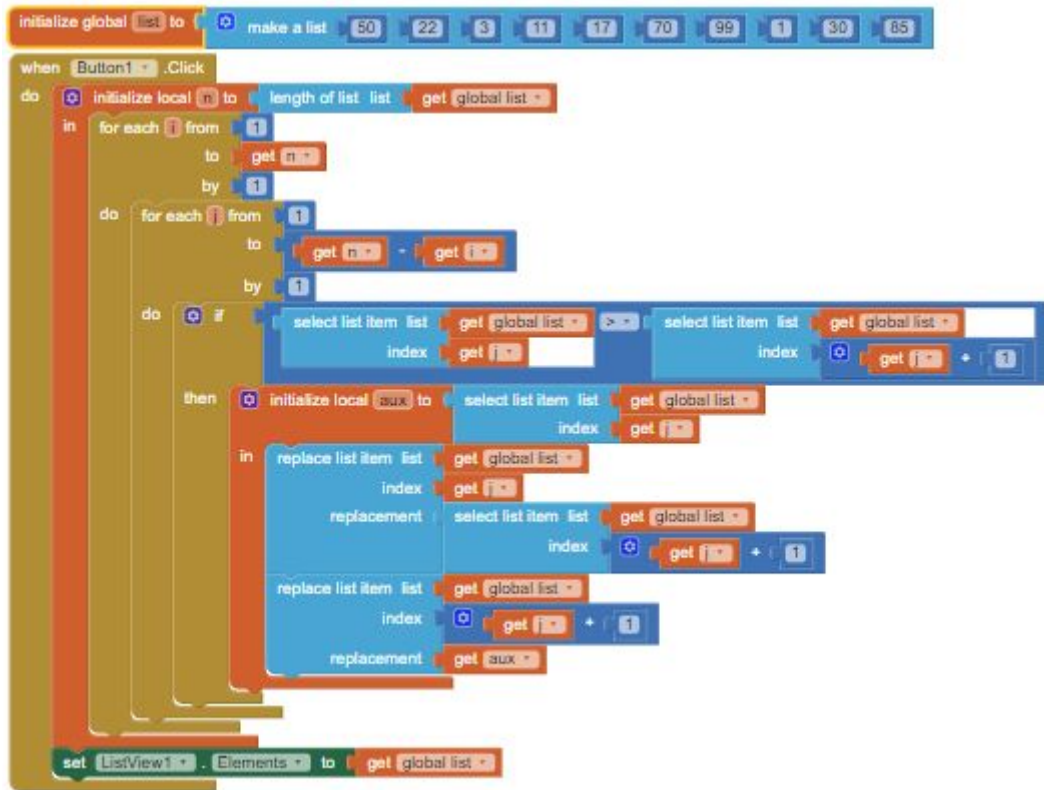


Figura 3.10. Código del algoritmo en App Inventor.

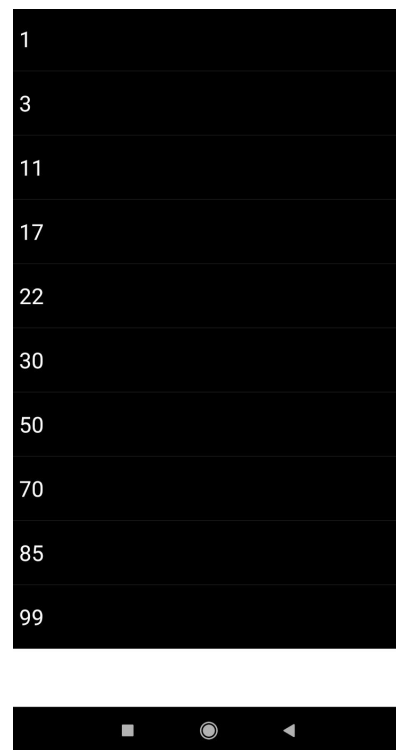


Figura 3.11. Resultado tras ejecutar el algoritmo en el Smartphone.

### 3.3.4 Snap!

Esta es la última plataforma seleccionada para el estudio y como se nombró con anterioridad, es una extensión de Scratch que incluye librerías más extensas y nuevos bloques con numerosas funcionalidades.

Para crear un nuevo proyecto en esta plataforma se debe hacer click en el botón de run Snap! en la barra de navegación principal y el usuario será redirigido al entorno de desarrollo.

Al igual que en scratch en esta plataforma se declaran en el menú de la izquierda, como peculiaridad no existe un tipo de variable lista como tal sino que debemos asignarle a una variable convencional el valor de una lista. A la hora de implementar el algoritmo fue bastante sencillo ya que la plataforma posee todas las estructuras necesarias para ello.

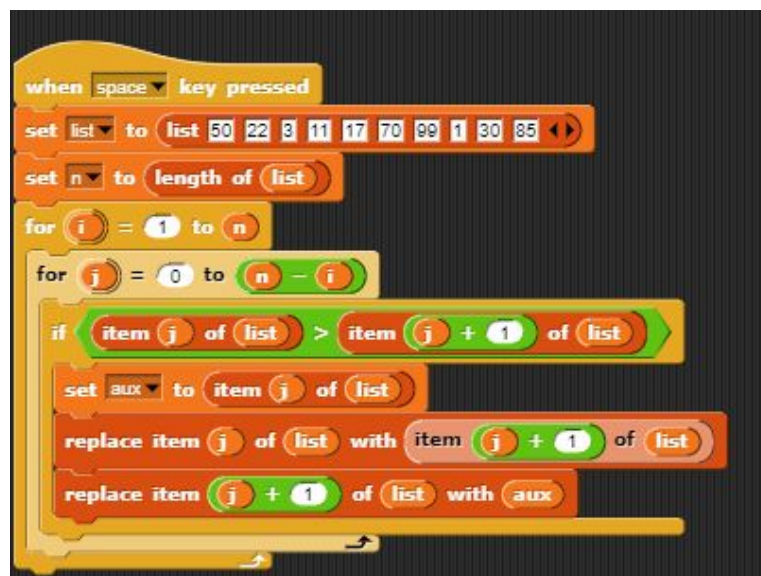


Figura 3.12. Algoritmo implementado en Snap!



Figura 3.13 . Resultado tras ejecutar el algoritmo.

# Capítulo 4 Seguimiento Personalizado

En este apartado se describe cómo se debería realizar el seguimiento personalizado del caso de estudio desarrollado para cada plataforma.

## 4.1 Seguimiento

En primer lugar, debido a que la tarea consiste en implementar un algoritmo en concreto, se debe comprobar si la salida para una supuesta entrada predeterminada es la misma, así partimos de que el algoritmo está bien implementado. A posteriori se debe comprobar el código implementado para ver si la implementación que ha realizado el alumno es la óptima siguiendo el pseudocódigo que se la ha proporcionado o no.

En segundo lugar se debería controlar el tiempo que ha tardado en realizar la tarea así se comprobaría si el alumno ha tenido dificultades a la hora de llevarla a cabo.

Como se ha comentado sobre el modelo incremental, como parte de la tarea se le ofrece el reto de realizar una mejora en el algoritmo, para ello se proporciona un nuevo pseudocódigo con la mejora. Para este caso en concreto se puede añadir una condición en el bucle principal de que si en una iteración no se realizó ningún cambio resultaría que la lista está ordenada y el algoritmo termina.

En tercer lugar como tarea siguiente o evaluativa se le mandaría al alumno la implementación, por seguir con la temática, de otro algoritmo de ordenación (algoritmo de inserción) y así comprobar que han adquirido los conocimientos.

## **4.2 Seguimiento personalizado**

### **4.2.1 Code.org**

En cuanto a la plataforma Code.org hemos utilizado el entorno de desarrollo de apps para móviles y esta permite implementar una interfaz para una pantalla móvil, es por eso que se pueden incluir tareas para mejorar la salida por pantalla del algoritmo, como por ejemplo añadir un botón para que se ejecute el algoritmo al pulsarlo, poder introducir una lista de elementos de cualquier tamaño o incluso utilizar distintos elementos gráficos que permite la plataforma.

Otra de las tareas que se pueden realizar sería la utilización del debugger para la depuración del código implementado, esto ayudaría a los alumnos en sus futuros desarrollos además de aprender a realizar una traza de, en este caso, los algoritmos implementados desde la propia plataforma.

### **4.2.2 Scratch**

La plataforma de Scratch no está pensada para la implementación de algoritmos sino que está orientada a la animación de sprites, ya sea añadiendo movimiento o sonido al sprite seleccionado. Es por esto que se podría realizar una tarea en base a la salida del algoritmo, es decir, añadir distintos tipos de animaciones o sonidos los sprites. Este tipo de tarea tocaría, a parte de la temática de la implementación del algoritmo, animación de sprites, condicionales y distintas estructuras de control.

### **4.2.3 MIT App Inventor**

App Inventor se centra en el desarrollo de aplicaciones móviles, es por esto que se puede aprovechar para introducir al alumno en distintos aspectos de la programación de aplicaciones para móviles y tablets. La actividad consistiría en implementar distintas pantallas desde las cuales el usuario pueda navegar con distintos los distintos algoritmos implementados, además que el usuario pueda introducir desde la pantalla los números que formarán la lista. Se puede hacer uso del propio dispositivo para que utilizando alguno de los sensores del teléfono se realice una acción dependiendo de la salida del algoritmo.

#### **4.2.4 Snap!**

Esta plataforma al igual que Scratch está centrada en el desarrollo o animación de un sprite, por lo que se puede aprovechar la salida de los algoritmos implementados para realizar alguna acción de animación sobre el sprite, sin embargo, esta plataforma ofrece más funcionalidades que Scratch, por lo que se puede realizar una tarea donde se le plantee al alumno hacer uso de los bloques para pintar en el canvas, de tal manera que los alumnos deban implementar el dibujo de cada número de la lista final.

# Capítulo 5 Conclusiones y Líneas futuras

## 5.1 Conclusiones

El pensamiento computacional es un término que en los últimos años ha ganado popularidad y ha demostrado ser beneficioso para aquellas personas que han sido entrenadas en esta disciplina, en concreto se ha centrado en el beneficio en los niños ayudando en sus estudios y en el desarrollo de sus vidas diarias. Debido a este motivo me parece interesante que las plataformas que promueven la enseñanza de esta disciplina creen cursos o planes de estudios pensados para que los profesores puedan adaptarlos en sus clases y educar a los alumnos en nociones de pensamiento computacional.

Es innegable decir que la sociedad cada día se dirige hacia un futuro donde la tecnología formará parte de la vida diaria de todos los ciudadanos, y cada vez se utilicen dispositivos inteligentes, por lo que si desde temprana edad se enseña su correcto uso, como funcionan desde bajo nivel puede ayudar a los usuarios a utilizarlos de manera más sencilla y correcta. Sin embargo, no hay que olvidar que se puede enseñar pensamiento computacional sin hacer uso de ordenadores o smartphones, y creo que es una opción interesante a la hora de reducir las horas de pantalla en niños y adolescentes.

Por otra parte, en cuanto a las plataformas debo decir que todas son muy interesantes a su manera, y que cada una, aunque la finalidad pueda ser la misma, la forma en la que interactúa con sus usuarios, su interfaz gráfica y las posibilidades que ofrece pueden marcar la diferencia. Debido a esto en mi opinión de todas las plataformas analizadas Code.org es la que mejor elaborada está, esto es debido a que posee diferentes entornos de desarrollo (aplicaciones móviles, sprites, juegos...), a su vez ofrece cursos y planes de estudios pensados para todas las edades y cualquier nivel de conocimiento, por lo que no deja a nadie fuera. Cabe destacar la forma en la que están pensadas las tareas, las cuales van aumentando su dificultad paulatinamente donde los alumnos pueden ir poniendo en práctica poco a poco sus conocimientos, prácticamente



sin darse cuenta. Por otra parte, no todas las actividades que se ofertan son haciendo uso de un ordenador o smartphone, tienen actividades “desconectadas” que pueden realizarse en un aula utilizando lápiz y papel ayudando a lo se comentó anteriormente sobre las horas de pantalla.

En cuanto al seguimiento de los estudiantes y la forma en la que comprueban si estos han adquirido los conocimientos expuestos en las lecciones, a excepción de Code.org, Alice y MIT App Inventor, el resto deberían plantear una manera en la que evaluar a los estudiantes si han adquirido los conocimientos que se enseñan en los cursos.

## **5.2 Líneas futuras**

### **5.2.1 Implementación de un sistema software**

La línea futura principal para este proyecto sería la implementación de un sistema software que permita realizar el seguimiento de los estudiantes en cualquier plataforma que ofrezca cursos de pensamiento computacional. Este sistema software deberá ser un recurso web de fácil acceso para todo el mundo.

Dicha plataforma debe tener tanto cursos, como planes de estudios que abarquen todos los tópicos de pensamiento computacional, sería ideal además que dichas tareas educativas no hiciera falta el uso de un ordenador o un smartphone, es decir, que se consiga enseñar pensamiento computacional sin hacer uso de la programación. Esta sería la situación ideal debido a que se podría enseñar PC sin hacer uso de ninguna plataforma específica.

En cuanto a la manera de evaluar a los alumnos sobre si han adquirido los conocimientos o no, en mi opinión, la mejor manera sería la proposición de un reto donde ellos deberán poner en práctica lo que han aprendido para llevarlo a cabo. A su vez creo que los conceptos deberían enseñarse de manera incremental, y al final de la lección se debería lanzar dicho reto para comprobar si los alumnos han adquirido los conocimientos o no.

# Capítulo 6 Summary and Conclusions

Computational Thinking is a term that has gained popularity in the last few years, several studies have been made and learning skills from this discipline is actually beneficial for the people, specially if you start learning it since you are young. Kids trained in CT develop critical thinking, analysis capacity, the ability to identify problems and solve them, in other words, being trained in this discipline will help the students in their everyday activities.

The platforms that offer CT training have some room to improve the courses they already have, and more importantly they should improve the way they teach CT to students. From my experience learning plans that make use of incremental cycles introducing a new concept on each one are better, because the student doesn't feel overwhelmed with a lot of new concepts. Throwing a challenge at the end of each plan is a great way to track if each student learnt about the topic that was taught in that lesson. Code.org is the one platform that does this kind of learning plan the others should try to make it that way as well.

The use of technology is more and more common for everybody's daily lives and it's here to stay, with that being said it's really important that since the early stages of our lives we know how this technology works. Training students in CT apart from helping them in their studies will also help them in creating new ways to use this technology.

# Capítulo 7 Presupuesto

En este capítulo se hará un presupuesto según el tiempo empleado para realizar las distintas tareas.

## 7.1 Presupuesto

<b>Tareas</b>	<b>Horas</b>	<b>Presupuesto</b>
Revisión Bibliográfica	50 horas	5€/hora
Escritura memoria	60 horas	7€/hora
Estudio de las plataformas	80 horas	10€/hora
Pruebas realizadas	30 horas	15€/hora
Total	220 horas	1920€

Tabla 7.1: Tabla de presupuestos para el proyecto.

Para llevar a cabo este proyecto hicieron falta 220 horas que tienen un precio final de 1920€ totales.

# Bibliografía

- [1] Carla Ramos Alonso. Pensamiento Computacional: Trazabilidad de los retos de entrenamiento. <http://riull.ull.es/xmlui/handle/915/14514>
- [2] Darwin González Suárez. Sistemas informáticos para la evaluación del entrenamiento del pensamiento computacional. <http://riull.ull.es/xmlui/handle/915/8523>
- [3] Borja Barrera Villagrasa. Programación en estudios pre-universitarios. Oportunidades, retos y caso de estudio. <http://riull.ull.es/xmlui/handle/915/3059>
- [4] Eliana Abdel Majid Hassan. Pensamiento computacional. Herramientas y aplicaciones.
- [5] Samuel Valcarcel Arce. Análisis de los resultados de los sistemas de entrenamiento del Pensamiento Computacional. <http://riull.ull.es/xmlui/handle/915/9400>
- [6] Rafael Herrero Álvarez. Una aproximación al pensamiento computacional a través de la nutrición. <http://riull.ull.es/xmlui/handle/915/5123>
- [7] Alexis Daniel Fuentes Pérez. Desarrollo y evaluación del pensamiento computacional: una propuesta metodológica y una herramienta de apoyo. <http://riull.ull.es/xmlui/handle/915/5122>
- [8] Cristian Manuel Angel Diaz. Desarrollo de Sistemas Software para robótica educativa: Robótica educativa y pensamiento computacional. <http://riull.ull.es/xmlui/handle/915/15460>
- [9] Eduardo de la Paz González. Simulador de robots para fomentar el pensamiento computacional a través de lenguajes de programación visual. <http://riull.ull.es/xmlui/handle/915/9409>
- [10] Andrea Rodríguez Rivarés. Robótica educativa mediante programación visual. <http://riull.ull.es/xmlui/handle/915/10306>
- [11] David Bau y D. Anthony Bau. A Preview of Pencil Code: A Tool for Developing Mastery of Programming. <https://doi.org/10.1145/2688471.2688481>
- [12] Guenaga, M., Mentxaka, I., Garaizar, P., Eguiluz, A., Villagrasa, S., Navarro, I. Make world, a collaborative platform to develop computational thinking and STEAM.

[https://www.scopus.com/inward/record.uri?eid=2-s2.0-85025138667&doi=10.1007%2f978-3-319-58515-4\\_5&partnerID=40&md5=b7664aaf6b5ae30e5fea26719c390a9b](https://www.scopus.com/inward/record.uri?eid=2-s2.0-85025138667&doi=10.1007%2f978-3-319-58515-4_5&partnerID=40&md5=b7664aaf6b5ae30e5fea26719c390a9b).

[13] Milne, L.R., Baker, C.M., Ladner, R.E. Blocks4All demonstration: A blocks-based programming environment for blind children.  
<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041400222&doi=10.1145%2f3132525.3134774&partnerID=40&md5=c4d3c94311f97baa5ac45976f1957e6b>

[14] Vincur, J., Konopka, M., Tvarozek, J., Hoang, M., Navrat, P. Cubely: Virtual reality block-based programming environment.  
<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85038596612&doi=10.1145%2f3139131.3141785&partnerID=40&md5=f116469b01c41891dda5df1034ee7f1c>

[15] Deshmukh, S., Raisinghani, V. ABC: Application based collaborative approach to improve student engagement for a programming course.  
<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85061339282&doi=10.1109%2fT4E.2018.00012&partnerID=40&md5=7ed5514fdf7ea953dc1d465f922d1037>

[16] Moreno-León, J., Robles, G., Román-González, M., & Rodríguez García, J. D. (2019). Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming.

[17] <https://www.motherjones.com/media/2014/06/computer-science-programming-code-diversity-sexism-education/>

[18] <https://www2.cs.duke.edu/csed/alice/aliceInSchools/>

[19] <https://es.wikipedia.org/wiki/Hackathon>

[20] [https://es.wikipedia.org/wiki/Algoritmo\\_de\\_ordenamiento](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento)

[21] <https://code.org/>

[22] <https://scratch.mit.edu/>

[23] <https://snap.berkeley.edu/>

[24] <https://appinventor.mit.edu/>

[25] <https://www.greenfoot.org/>

[26] <https://pencilcode.net/>

[27] <https://makeworld.eu/>

[28] <https://www.alice.org/>