Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

Trabajo de Fin de Grado

Sistemas informáticos para la evaluación del entrenamiento del Pensamiento Computacional

Computer systems for the evaluation of Computational Thinking training

Darwin González Suárez

La Laguna, 1 de junio de 2018

Dra. **Coromoto León Hernández**, con N.I.F. 78.605.216-W profesor Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

Dr. **Carlos Segura González**, con N.I.F. 78.404.244-S profesor Investigador tipo C adscrito al Departamento de Ciencias de la Computación del Centro de Investigación Matemática (CIMAT) de México, como cotutor

I N F O R M A (N)

Que la presente memoria titulada:

"*Sistemas informáticos para la evaluación del entrenamiento del Pensamiento Computacional*"

ha sido realizada bajo su dirección por D. **Darwin González Suárez**, con N.I.F. 54.057.325-B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de junio de 2018

Agradecimientos

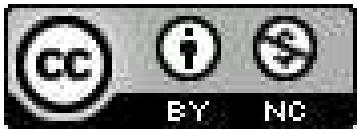
A mi tutora Coromoto, por darme la posibilidad de realizar este trabajo, siendo sin duda una tutora ejemplar. Agradecerle la gran implicación y el gran apoyo que me ha brindado.

A Carolina Giménez por el apoyo y la fuerza que me ha dado durante mi paso por el Grado en Ingeniería Informática, y como no, durante la realización de este trabajo.

A mis padres y a mi hermana por ser un apoyo incondicional en todo lo que me proponga.

A mis compañeros del grado, sobretodo a los componentes de la Delegación de Alumnnos, con los que he pasado algunas penas y muchas alegrías.

Licencia



Resumen

Se denomina Pensamiento Computacional (PC) al proceso que permite formular problemas de forma que sus soluciones pueden ser representadas como secuencias de instrucciones y algoritmos. Este proceso o manera de dar solución a problemas se puede manifestar de muchas formas diferentes tales como organizando una serie de datos y analizarlos detenidamente, representando datos mediante abstracciones como modelos o simulaciones, automatizando soluciones mediante pensamiento algorítmico,... entre otras muchas. Así pues, a modo general podemos determinar que el PC consiste en identificar, analizar e implementar soluciones con el objeto de encontrar la combinación de una serie de pasos de forma eficiente y efectiva.

La aplicación del PC conlleva una serie de consecuencias directas en relación con las habilidades implicadas en el mismo.Estas habilidades se apoyan y acrecientan mediante una serie de disposiciones o actitudes que son dimensiones esenciales del Pensamiento Computacional. Entre estas actitudes se encuentran la habilidad para lidiar con problemas no estructurados, tolerancia a la ambigüedad, capacidad de resolver problemas complejos partiendo de la descomposición del mismo para facilitar su resolución, etc.

Dicho lo anterior en este trabajo se realiza un estudio de la plataforma code.org, tratando aspectos muy interesantes que van desde su diseño e interfaz hasta su despliegue. Además se lleva a cabo el desarrollo de dos retos relacionados con el pensamiento computacional. En primer lugar es desarrollado un reto desde el rol de usuario, en la plataforma de code.org, mediante una herramienta denominada Game Lab. En segundo lugar se desarrolla un reto mediante la utilización de la librería Blockly, desde el rol de desarrollador. Ambos procesos de creación de los retos se documentan lo largo de este trabajo, así como las diferentes tecnologías implicadas en estos procesos y su configuración.

Cabe destacar además el estudio previo de los antecedentes y estado actual del tema que se realiza al principio de esta memoria, basado en una serie de artículos relacionados con el PC y que nos ayuda a comprender el contexto de este trabajo.

Palabras clave: pensamiento computacional, code.org, retos de programación.

Abstract

Computational Thinking (CT) is the process that allows you to formulate problems in such a way that your solutions can be represented as sequences of instructions and algorithms. This process or way of solving problems can be manifested in many different ways such as organizing a series of data and analyzing them carefully, representing data by means of abstractions such as models or simulations, automating solutions through algorithmic thinking,... among many others. So in a general way we can determine that the CT consists of identifying, analyzing and implementing solutions in order to find the combination of a series of steps efficiently and effectively.

The application of the CT involves a series of direct consequences in relation to the skills involved in it. These skills are supported and enhanced by a series of dispositions or attitudes that are essential dimensions of Computational Thinking. Among these attitudes are the ability to deal with unstructured problems, tolerance of ambiguity, ability to solve complex problems based on the decomposition of the same to facilitate its resolution, etc.

Said the previous thing in this work a study of the platform of code.org is made, treating very interesting aspects that go from its design and interface until its deployment. In addition, the development of two challenges related to computational thinking is carried out. In the first place, a challenge is developed from the user role, in the code.org platform, through a tool called Game Lab. In the second place, a challenge is developed by using the Blockly library, from the role of developer. Both processes of creation of the challenges are documented throughout this work, as well as the different technologies involved in these processes and their configuration.

It should also be noted the previous study of the background and current status of the topic, that is made at the beginning of this report, based on a series of articles related to the CT and that help us understand the context of this work.

Keywords: *computational thinking, code.org, programming challenges.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se introduce el concepto de pensamiento computacional, cuáles son los objetivos del trabajo y la metodología empleada.

1.1. ¿Qué es el pensamiento computacional?

A lo largo de los años se ha definido de diferentes formas el Pensamiento Computacional - (PC), siendo este un término que ha creado mucha controversia en cuanto a los diferentes campos en los que es aplicado. Dando una definición concreta del pensamiento computacional utilizaremos las palabras dadas por Jeannette Wing [28], las cuales definen el pensamiento computacional como una habilidad fundamental para todo el mundo, no sólo para las personas involucradas en las Ciencias de Computación, añadiendo que el pensamiento computacional debería considerarse como una habilidad analítica fundamental y básica, debiendo ser impartida desde las edades más tempranas tal y como lo son las actividades como leer o escribir. En ocasiones el término de pensamiento computacional puede llevar a confusión, pero tal y como dice J.Wing "Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers", eliminando el pensamiento erróneo relacionado con conseguir que los humanos piensen como los ordenadores. En relación a esto último J.Wing recoge también la siguiente reflexión: "Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting. Equipped with computing devices, we use our cleverness to tackle problems we would not dare take on before the age of computing and build systems with functionality limited only by our imaginations".

En relación con la definición del pensamiento computacional resulta interesante conocer la visión que aporta Miguel Zapata-Ros en su artículo [30] en el cual establece que "Una de las competencias que se muestran como más eficaces en la codificación es la forma de pensar y de afrontar el problema para su resolución, siendo esta útil no sólo en ese ámbito de actividades cognitivas,

las que se utilizan en el desarrollo y en la creación de programas y de sistemas informáticos". Refiriéndose al pensamiento computacional en el mismo artículo dice: "En definitiva sostienen que hay una forma específica de pensar, de organizar ideas y representaciones, que es propicia y que favorece las competencias computacionales. Se trata de una forma de pensar que propicia el análisis y la relación de ideas para la organización y la representación lógica de procedimientos". Afirma además que estas habilidades se ven favorecidas con cierta actividades o entornos de aprendizaje desde las primeras etapas.

Por otra parte, desde mi punto de vista y tras haber analizado la bibliografía existente el pensamiento computacional es una parte fundamental de la sociedad actual y de la sociedad del futuro. En base a lo investigado puedo definir el pensamiento computacional como un proceso mediante el cual un individuo afronta la resolución de un problema de manera analítica, estructurada y aplicando varios niveles de abstracción. Así pues, esta forma de resolución de problemas no queda restringida únicamente al entorno de las Ciencias de la Computación, sino que es aplicable a cualquier aspecto de la vida cotidiana, necesaria para comprender mejor los principales problemas del siglo en el que nos encontramos.

1.2. Herramientas para el desarrollo del pensamiento computacional

En la actualidad las plataformas orientadas al desarrollo del Pensamiento Computacional están ganando terreno en el campo de la educación, a la hora de enseñar Ciencias de la Computación y programación desde los niveles más bajos del sistema educativo. Así pues, están surgiendo diferentes formas de acercar la programación a los estudiantes a edades cada vez mas tempranas, un hecho que hace unos años podría parecer impensable debido a que con las herramientas tradicionales de programación es muy complicado enseñar a un estudiante conceptos como "bucle" o "iteración". Mediante estas nuevas plataformas y sus diferentes cursos, se está promoviendo la aplicación del Pensamiento Computacional en la resolución de problemas del día a día de los ciudadanos del siglo XXI, por lo que son primordiales para comprender el mundo en el que viven. Plataformas como code.org [8] y Make a World [9] están actualmente a la cabeza de lo relacionado con el Pensamiento Computacional, tal y como se tratará ampliamente en el Capítulo 2.

1.3. Objetivos

Los objetivos de este Trabajo Fin de Grado se pueden enmarcar en la cumplimentación de una serie de apartados expuestos a continuación:

- Realizar un estudio de la bibliografía relacionada, obteniendo así una serie de antecedentes e indicadores de la situación del Pensamiento Computacional.
- Seleccionar y estudiar una herramieta o plataforma relacionada con el Pensamiento Computacional.
- Llevar a cabo la codificación de retos de programación relacionados con el Pensamiento Computacional y con la plataforma que se ha elegido.
- Aportar la documentación necesaria en relación a cómo se ha llevado a cabo la codificación de los retos propuestos y a su funcionamiento.

1.4. Metodología

Se ha distribuido el trabajo y las diferentes tareas a realizar, de cara a la correcta realización del mismo. Además se han marcado las diferentes fechas de reuniones y seminarios así como los plazos de entrega de las tareas. Se fijaron las reuniones los miércoles a las 12:00, en las que se informaba de cómo iba la realización del trabajo y se hacía una valoración del progreso del mismo, tomando acciones correctivas en el caso de ser necesario. En la primera reunión realizada se estableció el siguiente plan de trabajo:

Tarea 1 - Revisión bibliográfica
Una vez realizada la planificación el siguiente paso fue obtener información de diferentes fuentes y hacer una selección de una serie de artículos relacionados con el tema del Pensamiento Computacional, para su posterior análisis y valoración de los antecedentes y estado actual del tema.

Tarea 2 - Selección de una plataforma de desarrollo del PC
En la búsqueda y análisis de artículos relacionados con el tema aparecen diferentes plataformas y herramientas relacionadas con el Pensamiento Computacional. Así pues en este apartado el objetivo era la selección de una de estas plataformas para someterla a estudio y trabajar con ella. Como se verá más adelante la elegida fue code.org debido a su gran calidad en los cursos y también un factor muy importante ha sido la disponibilidad de su código fuente disponible a través de la plataforma Github.

Tarea 3 - Estudio del diseño de la plataforma elegida
Una vez elegida la plataforma en cuestión se determinó un plazo de tiempo para indagar y comprender la estructura y funcionamiento de la plataforma en cuestión, en este caso, code.org.

Tarea 4 - Codificación de nuevos retos y validación
Así pues realizado ya el estudio y la familiarización con la plataforma en esta tarea se requiere la implementación de dos retos. Un reto orientado desde el perfil de desarrollador que quiere implementar un reto de programación para entrenar el pensamiento computacional de un usuario, en este caso mediante desplazamiento de bloques gráficos que generan un código fuente y producen una serie de acciones en un espacio de trabajo con animaciones. Por otra parte se propone realizar un reto desde la parte del usuario, es decir, codificar un juego u aplicación utilizando las herramientas que nos proporciona code.org a través de su laboratorio de código.

Tarea 5 - Documentación y difusión de los resultados obtenidos
Finalmente una vez acabada está parte de implementación se debe documentar todo lo que se ha realizado desde un punto de vista técnico, y además, se debe dar una documentación asociada a el modo de uso de los retos y al funcionamiento de los mismo a modo de manual de usuario. Por otra parte cabe destacar que la documentación también debe contener un manual con los pasos que se ha seguido de cara a ayudar a futuros desarrolladores que quieran implementar uno de estos retos de programación.

1.5. Contenido de la memoria

Tal y como se puede observar en el índice, esta memoria está estructurada por capítulos. El resto de capítulos de esta memoria se distribuyen de la siguiente manera:

- Capítulo 2, se hace un estudio de antecedentes y estado actual del tema, explicando la metodología de búsqueda de los artículos relacionados y realizando un breve resumen de la bibliografía.
- Capítulo 3, se realiza un estudio de la plataforma code.org [8] y de Blockly [13]. Se trataran puntos como su funcionamiento, su despliegue, estructura, etc.
- Capítulo 4, se crean y se prueban una serie de retos, utilizando en primer lugar el punto de vista del usuario final y luego del desarrollador.
- Capítulo 5, se exponen las conclusiones y las líneas futuras del trabajo.

Finalmente en la última parte del documento se relacionan el presupuesto y la bibliografía utilizada en la realización de este trabajo.

Capítulo 2

Antecedentes y estado actual del tema

En este capítulo se explica la metodología de búsqueda de información contrastada y se analizan las referencias relacionadas.

2.1. Metodología de búsqueda de referencias bibliográficas

Este capítulo describe los antecedentes y el estado actual en el ámbito de las herramientas para el desarrollo del Pensamiento Computacional. Para ello, se ha realizado una búsqueda exhaustiva de documentación relacionada, en Bases de Datos de editoriales de prestigio más utilizadas en Ingeniería:

- Scopus
- Web of Science

PuntoQ es la herramienta de búsqueda de información de la Universidad de La Laguna, a través de ella los miembros de la comunidad universitaria acceden a las bases de datos, revistas y libros electrónicos que adquiere la Universidad. Más concretamente en lo relacionado a está última herramienta se ha buscado en el apartado de Bases de Datos y eligiendo las editoriales ACM, Digital Library y IEEE Xplore. También se han utilizado Google Scholar, Scopus y Web of Science que son Bases de Datos bibliográficas de resúmenes y citas de artículos de revistas científicas.

Los documentos obtenidos se han encontrado usando las palabras clave: "Pensamiento Computacional", "Programación en Bloques", "Code.org" y "Computational Thinking".

2.2. Listado de artículos seleccionados

En este apartado, después de haber completado la fase de búsqueda de referencias, se han seleccionado tres artículos relacionados con el Pensamiento Computacional. Así pues, nos disponemos en el mismo a listar cada uno de los artículos y a hacer un breve resumen de cada uno de ellos y del tema al que atañe. Los artículos son:

- Panel: Future Directions of Block-based Programming [4]
- Make World, A Collaborative Platform to Develop Computational Thinking and STEAM [9]
- Hour of code: Can it change students' attitudes toward programming? [8]

2.2.1. Panel: Future Directions of Block-based Programming

En el primer artículo [4] se trata el tema sobre la programación basada en bloques. Los autores de este artículo destacan que este tipo de programación se está convirtiendo en la manera en la que los aprendices son introducidos a la programación y a las Ciencias de la Computación. Esto se produce principalmente dirigido por herramientas como Scratch [23], Alice [26] y las horas de código de code.org [5] ya que en los entornos que utilizan se emplean técnicas y tecnologías relacionadas con este campo de la programación en bloques.

Así pues los autores creen que el gran desarrollo e interés que ha despertado este tipo de programación se debe a características como una interfaz amigable y sencilla de entender que es ayudada por los colores y formas de los bloques, capacidad de coger y dejar bloques con total libertad dentro de un espacio de trabajo, intuitividad del entorno, etc. Todo esto conforma un entorno de desarrollo donde los aprendices son capaces de escribir sus primeros programas de forma exitosa.

También se destaca que a la hora de crear un entorno basado en la programación en bloques hay que tener en cuenta diversos aspectos tanto de la interfaz como de las funcionalidades. Aspetos como la distribución de los elementos de la interfaz, interación entre los bloques, cuestiones del lenguaje de programación utilizado, entre otros muchos otros son claves para el buen funcionamiento de una plataforma de estas características.

Finalmente, en este artículo los autores exponen la posible visión de futuro de estos lenguajes de programación visual, determinando que en principio

tendrán un papel muy importante en lo relacionado con el entorno educacional tanto a corto como a largo plazo. Apuntan a que sólo con el paso del tiempo se podrá ver si finalmente estos lenguajes sólo ocuparán la función dedicada al entorno educacional o si por lo contrario podría llegar a tomar un primer plano, llegando a coexistir en entornos mixtos, o desplazando incluso en algunas ocasiones a la programación basada en texto tradicional que todos conocemos.

2.2.2. Make World, A Collaborative Platform to Develop Computational Thinking and STEAM

En el segundo artículo [9] los autores comienzan hablando sobre la gran demanda de profesionales de la programación informática formados en STEAM (acrónimo en inglés que sirve para designar las disciplinas académicas de Ciencia, Tecnología, Ingeniería y Matemáticas) que ha aumentado considerablemente en a última década. Iniciativas como "Hour of Code" y "CodeWeek" aprovechan las plataformas como code.org para llegar así a miles de estudiantes. A pesar del excelente diseño y gran calidad, estos cursos promovidos por code.org, los autores remarcan la existencia de plataformas como Make a World que provee a los estudiantes de un entorno donde ellos mismos pueden realizar retos de programación y compartirlos con otros usuarios.

Después de un año de uso de la plataforma por parte de profesores y estudiantes los autores determinan el modo en el que utilizan esta herramienta para proponer y resolver actividades que mejoran las capacidades relacionadas con el Pensamiento Computacional y con las competencias necesarias en STEAM.

Finalmente en este documento los autores describen una serie de características de la plataforma Make a World y es analizada basándose en datos obtenidos durante una fase piloto donde esta plataforma es usada por 500 estudiantes de primaria, y se valora el impacto que tiene esta plataforma en su aprendizaje, pudiendo observar en los datos recogidos que muchos de estos estudiantes valoraron positivamente el uso de esta plataforma y que además sus habilidades de aprendizaje se vieron mejoradas tras la utilización de Make a World.

Como conclusión los autores proponen que incrementar el interés en los campos de STEAM es primordial, así como mejorar el nivel y la calidad del aprendizaje del mismo, valorando que no se debe sólo al incremento de las ofertas de trabajo en estos campos, si no por la necesidad imperiosa de que los ciudadanos del siglo XXI puedan comprender el mundo en el que viven.

2.2.3. Hour of code: Can it change students' attitudes toward programming?

En el tercer artículo [8] los autores introducen la plataforma code.org, y más concretamente uno de sus cursos más famoso denominado "The Hour of Code", conocido en español como "La Hora de Código". Los autores explican que "The Hour of Code"es una hora de introducción a las Ciencias de Computación promovida por code.org, una plataforma sin ánimo de lucro dedicada a acercar a las personas aspectos relacionados con las Ciencias de Computación.

Así pues, en este documento se pueden ver los resultados de un estudio sobre el impacto de "The Hour of Code"en los estudiantes y sus capacidades de cara a las ciencias de computación y a sus conocimiento en programación. El estudio realizado involucra a diferentes alumnos de dos universidades que aún no han acabado sus estudios. Dichos alumnos son seleccionados para completar uno de los cursos de "The Hour of Code". Estos alumnos deben realizar un cuestionario online previo a la realización de la hora de código y uno posterior a ella para así poder valorar qué cambios se han producido en los alumnos de cara a sus habilidades de programación.

Los autores nos ofrecen finalmente una serie de resultados en forma de tablas y pequeñas gráficas con los resultados del estudio, pudiendo concluir que en definitiva "The Hour of Code"tiene en general una buena aceptación en los alumnos, y sobretodo un impacto positivo de cara a fomentar el interés y la actitud de los estudiantes hacia la programación. Sin embargo los autores hacen hincapié en que "The Hour of Code"no es una manera válida para incrementar las capacidades de los alumnos de cara a la programación, ya que no se puede concluir de forma fehaciente que estos estudiantes mejoraron sus habilidades en el campo de la programación.

Capítulo 3

La plataforma code.org y Blockly

En este capítulo se introducirá la plataforma code.org y el lenguaje de programación visual de Google denominado Blockly.

3.1. ¿Qué es code.org y code-dot.org?

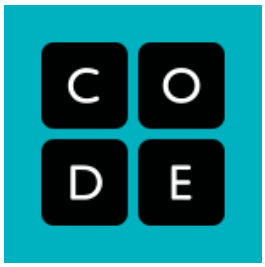


Figura 3.1: Logo code.org

Code.org [5] es una organización sin ánimo de lucro con un claro objetivo, difundir la programación como parte de la educación básica de los jóvenes, ofreciendo una plataforma que provee una serie de herramientas y cursos para impulsar el desarrollo del pensamiento computacional. Tal y como se dijo en una de las primeras tareas fijadas era necesario elegir una plataforma destinada al entrenamiento del pensamiento computacional, siendo esta la elegida, debido a que por una parte es lo suficientemente conocida como para encontrar documentación de calidad relacionada con la plataforma y a que sus cursos son de una calidad indiscutible. Además se encuentra disponible el código fuente de la plataforma en su totalidad en un repositorio público en la plataforma Github, siendo esta también una de las principales razones por la que se eligió esta plataforma.

Este repositorio recibe el nombre de code-dot-org [6] y en el se encuentra el código fuente de la plataforma code.org [5] así como la documentación necesaria para la comprensión de prácticamente todos los elementos que intervienen en esta plataforma. De la misma forma este repositorio funciona a modo de comunidad debido a que al ser de código libre o código abierto diferentes usuarios aportan funcionalidades, las cuales se pueden implementar en la plataforma principal, o no, dependiendo de los dueños de este repositorio.

3.2. ¿Cómo se despliega la plataforma en nuestra máquina?

A continuación en esta sección se explicará detalladamente los pasos necesarios para desplegar la plataforma que se encuentra disponible en un repositorio [6] público en Github. La plataforma en cuestión puede desplegarse utilizando diferentes sistemas operativos tales como OSX, Windows y Ubuntu. La configuración de Windows es bastante complicada debido a que es necesario la instalación de una maquina virtual con Ubuntu, pero es perfectamente factible. En este apartado, y debido a que este trabajo se ha desarrollado en un ordenador con OSX, se explicará como se ha llevado a cabo la configuración del entorno y el despliegue de la plataforma en este sistema operativo.

3.2.1. Configuración del entorno de trabajo en OSX

Instalar Homebrew

Figura 3.2: Logo Homebrew

En primer lugar, se necesita instalar el Homebrew [12] en nuestra máquina. Homebrew es un sistema de gestión de paquetes que simplifica la instalación, actualización y eliminación de programas en los sistemas operativos Mac OS de Apple, instalando todo aquello que se necesita y que los dispositivos Apple no traen de serie. Así pues, se ejecutará el siguiente comando para la instalación:

```
$ ruby -e "$(curl -fsSL URL$
URL = https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Instalar Redis

Figura 3.3: Logo Redis

El siguiente paso será instalar Redis [24] que es un motor de Base de Datos en memoria, basado en el almacenamiento en tablas de hashes pero que opcionalmente puede ser usada como una Base de Datos durable o persistente. Debido a que previamente se ha instalado Homebrew [12], ejecutando el siguiente comando se instalaría Redis:

```
$ brew install redis
```

Ejecutaremos el siguiente comando:

```
brew install URL$ PARAMS$
URL = https://raw.github.com/quantiverge/homebrew-binary/pdftk/pdftk.rb
PARAMS = enscript gs mysql nvm imagemagick rbenv ruby-build coreutils sqlite phantomjs
```

En caso de no instalarse correctamente redis, se pueden dar dos situaciones:

- Por un lado, puede decirnos que no encuentra `Formula.sha1`, por lo que se debe eliminar:
  ‘‘`https://raw.github.com/quantiverge/homebrew-binary/pdftk/pdftk.rb`’’
  del comando anteriormente mencionado.
- Por otra parte, si se queja de una versión vieja del paquete, se debe ejecutar ‘‘`brew unlink <package>`’’ y de nuevo hacer ‘‘`brew install <package>`’’.

Configuración de MySQL

Figura 3.4: Logo MySQL

MySQL [21] es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la Base de Datos de código abierto más popular del mundo.

- Tener el comando ‘‘`launchd`’’ al arrancar mysql:
  ‘‘`ln -sfv /usr/local/opt/mysql/*.plist  /Library/LaunchAgents`’’
- Arrancar mysql ahora de la siguiente forma:
  ‘‘`launchctl load  /Library/LaunchAgents/homebrew.mxcl.mysql.plist`’’

Configurar rbenv

Figura 3.5: Logo rbenv

El rbenv es un manejador de versiones de Ruby [19] y utiliza una arquitectura basada en plugins para agregar más funcionalidad. Eso quiere decir que si se quiere instalar una nueva versión de Ruby, se debe hacer manualmente o instalar primero un plugin de `rbenv` llamado `ruby-build` (solo se debe instalar una única vez). Así pues, los pasos de configuración del mismo son:

- Ejecutar ‘‘`run rben init`’’
- Añadir ‘‘`eval "$(rbenv init -)"`’’ al archivo “`source ~/.bash_profile`”
- Hacer efectivos estos cambios con “`source ~/.bash_profile`”

Instalar Ruby 2.5.0

Figura 3.6: Logo Ruby

Ruby [19] es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Además, Ruby es un lenguaje de programación interpretado, reflexivo, orientado a objetos, multiparadigma y fuertemente tipado.

- Ejecutar ‘‘`rbenv install 2.5.0`’’
- Fijar la versión global de Ruby que se desee: ‘‘`rbenv global 2.5.0`’’

Configurar nvm

NVM es el acrónimo de “Node Version Manager” o “Gestor de Versiones de Node”, similar a RVM, que significa “Ruby Version Manager” o “Gestor de Versiones de Ruby”.

- Crear un directorio de trabajo de nvm si este no existe aún: ‘‘`mkdir /.nvm`’’
- Añadir el siguiente fragmento al archivo “`~/.bash_profile`”

```
# Load nvm function into the shell
export NVM_DIR=~/.nvm
source $(brew --prefix nvm)/nvm.sh
```

Figura 3.7: Fragmento de texto nvm

- Finalmente recogemos esos cambios con “`source ~/.bash_profile`”

Instalar Node, npm, y yarn

(a) Node (b) Npm (c) Yarn

Figura 3.8: Node, npm y yarn

Node.js [18] es una librería y entorno de ejecución de entrada/salida dirigida por eventos y, por lo tanto, asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google V8. Por otra parte, npm [29] es el manejador de paquetes por defecto para Node.js y, por último, yarn [25] es un nuevo tipo de instalador de paquetes JavaScript [22] y gestor de dependencias lanzado por Facebook en colaboración con Google entre otros donde introduce cambios en la gestión de dependencias, en la ejecución de tareas y algunas mejoras de rendimiento, también en el cambio de enfoque en la descarga e instalación de los paquetes y en su gestión de las dependencias. Así pues, una vez que se conoce que tecnología ejecuta cada acción, procedemos a instalarlas de la siguiente forma:

- Ejecutar `nvm install 6.9.0 && nvm alias default 6.9.0` que debería poner la versión indicada por defecto en el sistema. Una vez hecho saldrá por pantalla algo como:
  `Creating default alias: default ->6.9.0 (->v6.9.0)`
- Seguidamente se ha de instalar yarn en la versíon indicada:
  `curl -o- -L https://yarnpkg.com/install.sh | bash -s -- --version 0.23.2`
- Finalmente, para prevenir problemas futuros relacionados con la apertura de ficheros masiva, se pondrá un límite añadiendo al archivo `~/.bash_profile` la línea siguiente ‘‘`ulimit -n 8192`’’.

Instalar línea de comandos de Xcode

Figura 3.9: Logo Xcode

Xcode [1] es un entorno de desarrollo integrado para macOS que contiene un conjunto de herramientas creadas por Apple destinadas al desarrollo de software y, en este caso, es necesario instalar su línea de comandos para poder desplegar el proyecto. Se instalará ejecutando el siguiente comando:

```
$ xcode-select --install
```

3.2.2. Montaje de la plataforma

Una vez configurado el entorno con el que vamos a trabajar de la manera que se ha descrito debemos comprobar que contamos con las versiones determinadas para que todo funcione correctamente. Las versiones necesarias son las que aparecen a continuación:

- `ruby --version` debe ser `ruby 2.5.0`
- `node --version` debe ser `v6.9.0`
- `npm --version` debe ser `3.10.8`
- `yarn --version` debe ser `0.23.2`

Es importante que las versiones de las dependencias coincidan con las previamente mostradas, en caso contrario la plataforma no se podrá desplegar debido a problemas de incompatibilidad entre los elementos que componen la aplicación.

Descarga del contenido del repositorio

Una vez configurados los pasos anteriores, es el momento de traer el contenido del repositorio [6] a nuestra máquina. Esto se puede hacer de dos formas, por ejemplo si se utiliza `HTTPS` se ejecutará el siguiente comando:
‘‘`git clone https://github.com/code-dot-org/code-dot-org.git`’’.
Si, por el contrario, se emplea `SSH` haremos uso del siguiente enlace:
‘‘`git@github.com:code-dot-org/code-dot-org.git`’’

Seguidamente, se ejecutarán una serie de comandos para la instalación y configuración de las diferentes dependencias necesarias. Estos se ejecutarán en el orden en el que aparecen en la lista siguiente:

- `$ gem install bundler`
- `$ rbenv rehash`
- `$ cd code-dot-org`
- `$ bundle install`
- `$ rake install:hooks`
- `$ rake install`
- Finalmente para construir nuestra aplicación `$ rake build`

3.2.3. Despliegue de la plataforma

Llegados a este punto, nuestra plataforma estará construida a partir del código fuente que se ha descargado del repositorio [6], habiendo instalado todas las dependencias necesarias y montado la plataforma en sí misma. El siguiente paso a seguir es el despliegue de la misma, para así poder interactuar como si se tratara de la plataforma original de codeorg, solo con la diferencia de que esta plataforma estará desplegada en nuestra máquina y se tendrá acceso completo al código fuente de la misma (con todas las ventajas de cara al desarrollador que esto conlleva).

Arranque del servidor

Con vistas a arrancar el servidor de prueba o desarrollo, es necesario contar con una terminal abierta y ejecutar el siguiente comando: `$ bin/dashboard-server`.

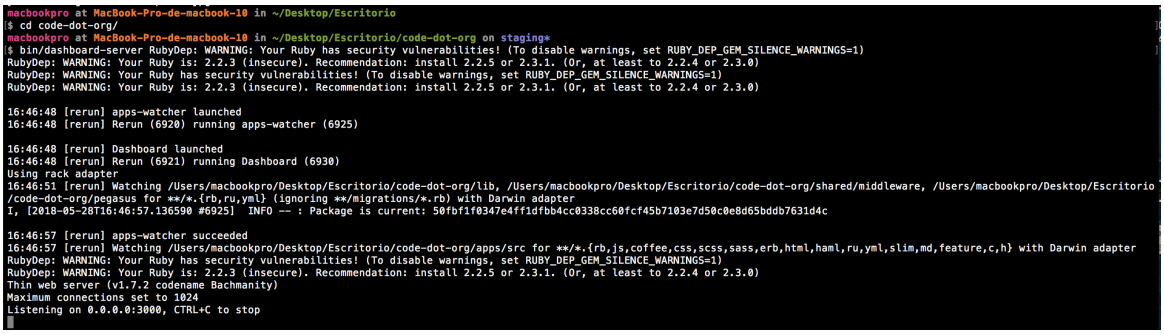


Figura 3.10: Servidor arrancado desde la terminal

Una vez realizado el paso anterior, el servidor se encuentra levantado (mientras no se cierre el proceso en la terminal). Finalmente, y para acceder a nuestra plataforma de prueba, es necesario abrir un navegador cualquiera e ir a la siguiente dirección:

http://localhost-studio.code.org:3000/

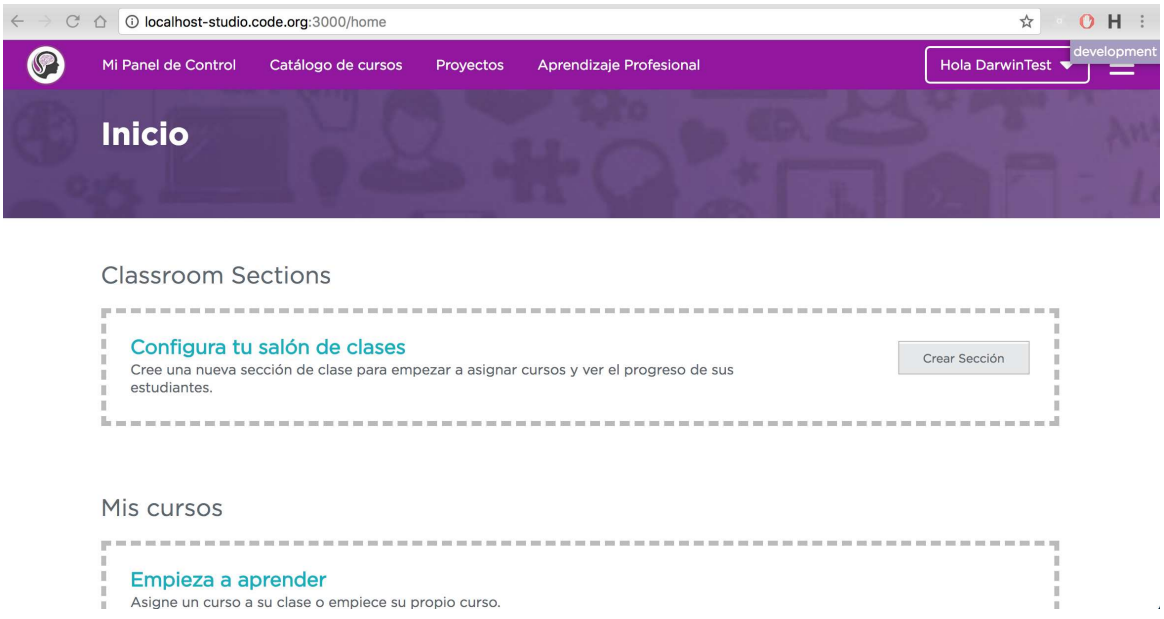


Figura 3.11: Captura de la plataforma de prueba desplegada

3.3. ¿Cómo está desarrollada la plataforma?

En está sección en primer lugar se procederá a explicar la estructura de directorios de la plataforma code.org, haciendo un breve resumen de los diferentes propósitos de cada una de estas carpetas. La plataforma code.org, tal y como se ha dicho, tiene disponible el código fuente en un repositorio en Github [6] distribuido de la siguiente manera:

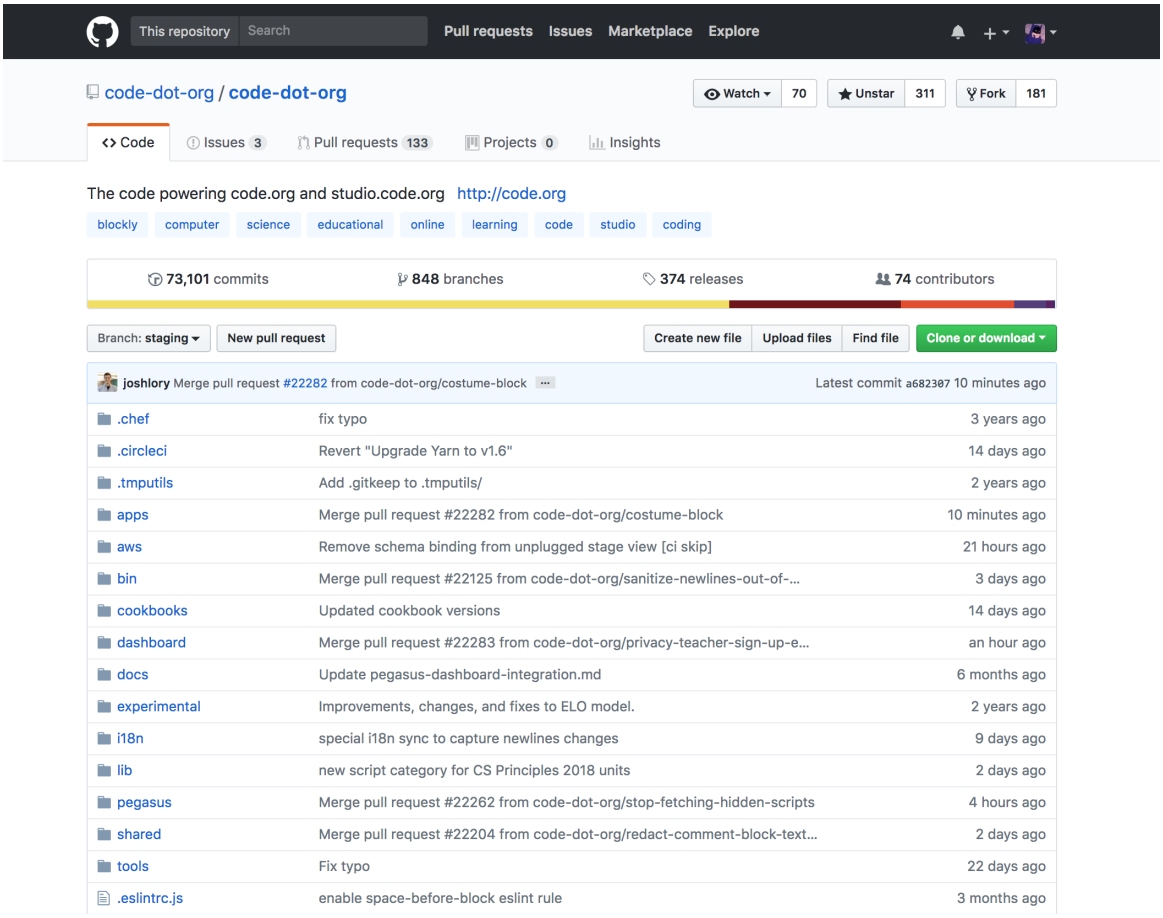


Figura 3.12: Estructura repositorio code.org

Como se puede ver hay una serie de carpetas y archivos que conforman la plataforma. Debido al gran volumen de información y archivos con los que se trabaja, nos centraremos en explicar los directorios más relevantes en relación al correcto funcionamiento de la plataforma code.org.

Directorio Dashboard

En este directorio esta contenido el servidor responsable de la plataforma de aprendizaje Code Studio, una aplicación desarrollada en Ruby on Rails [11] la cual es responsable:

- Configuración de los diferentes cursos, tutoriales y puzzles de la plataforma code.org.

- Manejo de las cuentas de usuario de la plataforma.
- Almacenamiento del progreso en cada uno de los diferentes cursos.
- Almacenamiento de los diferentes proyectos creados por el usuario (sea del rol que sea).
- Herramientas relacionadas con la creación o construcción de los diferentes niveles en los cursos o puzzles.

Directorio Pegasus

En este directorio se sitúa el servidor de la plataforma web Code.org, una aplicación desarrollada en Sinatra [10] responsable de aspectos como `code.org`, `hourofcode.com`, `csedweek.org` o `Teacher Dashboard`.

Directorio apps

En este directorio se encuentra todo el motor en JavaScript desarrollado utilizado en los diferentes tutoriales, puzzles y herramientas online. Este directorio tiene como objetivo englobar todo el código en JS para así poder obtener los beneficios de la utilización de utilidades relacionadas con el ECMAScript 6 [3], el nuevo estándar de JavaScript. Así pues, se encuentra el código fuente relacionado con los siguientes apartados:

- Tutoriales "Hora de código" o "Hour of code" como Classic Maze(Laberinto), Star Wars, Minecraft o Frozen.
- Herramientas como `Artist`, `Play Lab` y `App Lab`.
- Otro tipo de puzzles base como: Maze, Farmer, Bee, Bounce, Calc, Eval.
- Código desarrollado en JS necesario para los directorios previamente nombrados (Dashboard y Pegasus).

Además de estos directorios podemos encontrar una serie de directorios secundarios, pero totalmente necesarios para el funcionamiento de la plataforma como:

- **aws**: archivos de configuración y scripts que gestionan el despliegue de la plataforma.
- **bin**: diferentes herramientas y utilidades utilizadas por los desarrolladores.
- **cookbooks**: configuración de la administración de la plataforma a traves de Chef [17].

- **shared**: código y diferentes elementos utilizados en muchas de las partes de la aplicación.
- **tools**: información relacionada con los diferentes commits de Github, además de scripts para el manejo de los mismos.

Así pues, a modo de ejemplo , se podrá observar donde está contenido el código fuente principal necesario para la creación de uno de los cursos más famosos en la plataforma code.org denominado "Hour of Code".

3.4. Introducción Blockly y su modo de uso

Figura 3.13: Logo Blockly

Este apartado del Capítulo 3 se centrará en analizar Blockly. Blockly o también conocido como Google Blockly es una librería escrita en JavaScript, 100 % del lado del cliente y no requiere soporte del servidor (a menos que se desee usar alguna función de almacenamiento en la nube o escritura de datos en alguna base de datos). Este permite crear diferentes instancias de programación visual, permitiendo trabajar con programación basada en bloques, tema que se expuso en uno de los artículos [4] del capítulo 2. Así pues, Blockly crea un entorno de trabajo amigable y útil para aprender a programar de una forma intuitiva y simple, centrándose sobre todo en la lógica de programación que es uno de los puntos clave cuando se comienza a programar [13].

Blockly está diseñado para instalarse fácilmente en cualquier lugar, ya sea una aplicación web o en una aplicación móvil. Los usuarios arrastran bloques a un espacio de trabajo, Blockly genera un código y la aplicación hace algo con ese código generado. Desde el punto de vista de la aplicación, Blockly es solo un área de texto en la que el usuario escribe en JavaScript, Python, PHP, Lua, Dart.

Cabe destacar además que Blockly es una herramienta para crear lenguajes de programación visual, pero que no es un lenguaje de programación en si mismo, ya que para ejecutar un programa de Blockly debemos traducir el código

contenido en los bloques a un código que sí está escrito en un lenguaje de programación como lo son los previamente nombrados.

Una vez explicado qué es Blockly, se procederá a explicar los principales elementos que intervienen en su funcionamiento, y las diferentes características del mismo. En primer lugar al desarrollar una aplicación en Blockly lo primero que se puede ver es que la interfaz está dividida claramente en dos partes bien diferenciadas. Por una parte se encuentra el espacio donde estarán contenidos los bloques que se hayan creado, y por otra un espacio de trabajo donde se arrastrarán estos bloques y se entrelazarán a modo de piezas de puzzle. El lugar donde se encuentran los diferentes bloques se denomina "Toolbox" o "Caja de herramientas". Los bloques aquí contenidos son totalmente personalizables, ya sea mediante la creación de los mismos por codificación directa o a través de un editor de bloques que nos provee Google llamada "Blockly Developer Tool" [16] el cual se tratará a continuación. En la parte en la que se arrastran los bloques se pueden mover con total libertad cada uno de los bloques desde el "Toolbox" a este espacio, y entrelazarlos, pudiendo incluso desechar estos bloques mediante una papelera que se encuentra en el lado inferior izquierdo.

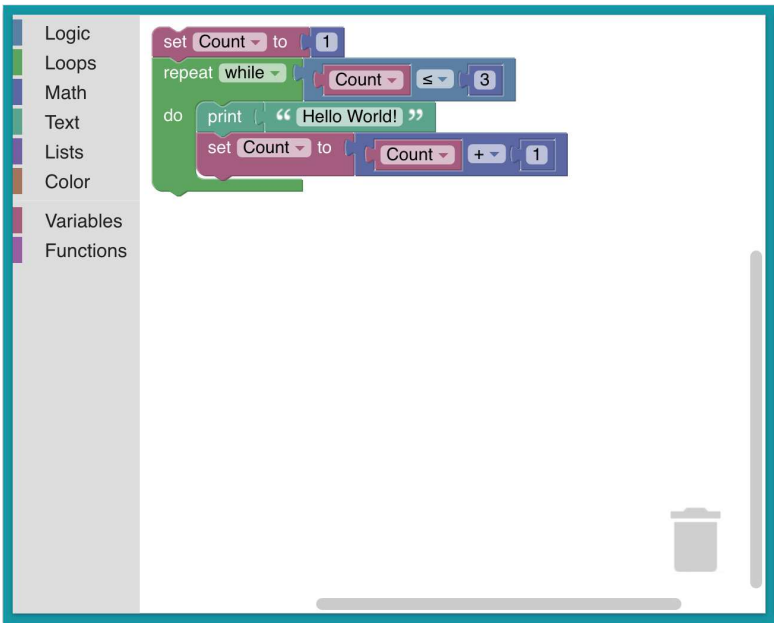


Figura 3.14: Interfaz Blockly

Como se ha nombrado previamente y en relación con los bloques de Blockly, estos pueden ser creados de dos maneras diferentes. Tal y como se ha dicho con anterioridad, Blockly permite desarrollar un lenguaje de programación propio, pero para ello se deben generar o crear nuestros propios bloques. Cada bloque utilizado en Blockly, sea personalizado o por defecto en la librería de Blockly, consta de dos partes bien diferenciadas. En primer lugar un apartado

relacionado con la apariencia y la visualización del bloque, en la se determinan aspectos como su color, tamaño, textos, número de conexiones con otros bloques y el tipo de conexión en el caso que las tuviera. Por otra parte tenemos la parte del bloque encargada de generar el código necesario en el lenguaje de programación que se desee, siendo está parte siempre desarrollada en JavaScript independientemente del lenguaje al que se quiera exportar la información del bloque.

Figura 3.15: Ejemplo partes de un bloque de "selector de color"

Ya habiendo visto de forma clara las dos partes que componen un bloque en Blockly, se procederá a explicar la modificación o creación de bloques. Así pues, a la hora de la creación o modificación de bloques se puede hacer de manera manual manipulando o creando los ficheros JSON y JavaScript pertenecientes a cada bloque. En el caso que de elegir la vía manual podemos determinar que no es recomendable debido a que puede llegar a resultar complicado y complejo el trabajo con estos ficheros, pudiendo generar problemas de compatibilidad entre bloques. La plataforma de desarrolladores de Google Blockly se ha dado cuenta de esto y ha puesto a servicio público la herramienta "Blockly Developer Tool" [16]. Esta herramienta permite generar bloques de una manera personalizada mediante la utilización de una interfaz de fácil uso en la cual se pueden generar los bloques que se van a usar en el desarrollo utilizando la filosofía de "drag/-drop/connect", tal y como si se estuviera construyendo un puzzle.

En cuanto a la interfaz de esta herramienta podemos ver de nuevo dos partes diferenciadas, en la parte derecha podemos observar el espacio de trabajo con los diferentes modelos o esqueletos de bloques. Estos se interconectarán a gusto del creador determinado los diferentes parámetros, tipos de datos, conexiones, etc. Por otra parte en la parte izquierda tenemos el contenido de los ficheros

JSON y JS que definen el bloque que se está construyendo y un pequeño visualizador del aspecto que tendrá el bloque una vez generado.

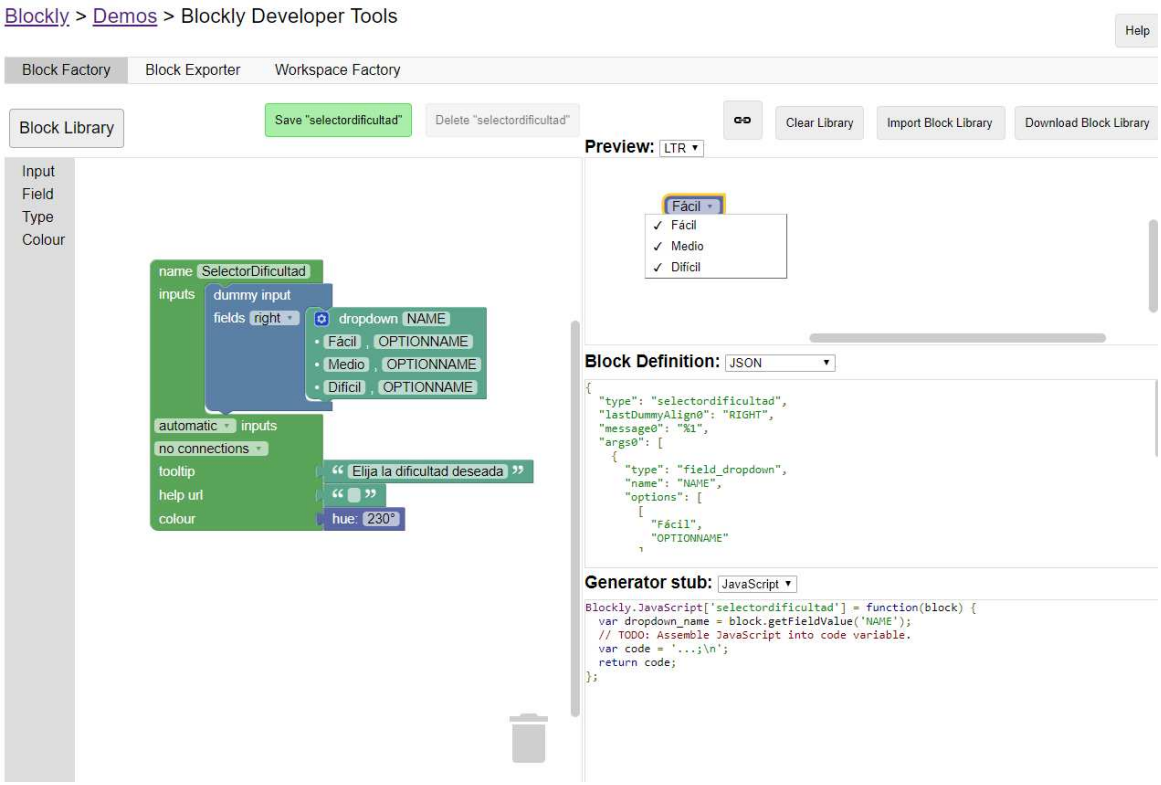


Figura 3.16: Ejemplo uso "Blockly Developer Tool"

Con la herramienta de generación de bloques lo único que se obtendrá es la parte de declaración o definición del bloque relacionada con su color, su aspecto, sus textos, conexiones, etc.La parte relacionada con la generación del bloque se tendrá que escribir de forma manual en la aplicación, una vez generado el bloque, en un fichero accesible desde nuestra aplicación.

Capítulo 4

Creando retos para el desarrollo del pensamiento computacional

En este capítulo se mostraran dos retos creados a partir de la plataforma code.org [8] y de Blockly [4], debiendo hacer una distinción entre ambos ya que en el primer caso se creará un reto desde el rol de un usuario con algunas nociones de programación que a través de las herramientas que le proporciona code.org es capaz de realizar un juego o aplicación. Por otra parte se mostrará cómo desde el rol de desarrollador se puede crear una "hora de código sencilla" utilizando Blockly, explicando como construirla y desplegarla para poder acceder a ella.

4.1. Reto creado por los usuarios

En esta sección del capítulo 4 se explicará detenidamente los pasos que se deben seguir para desarrollar un reto o aplicación en la plataforma code.org, en este caso, siendo un usuario de la plataforma. En primer lugar es totalmente necesario darse de alta o registrarse en la plataforma siguiendo los pasos que detenidamente se explican en la misma. Este registro se puede hacer de dos formas, por una parte con un rol al que llamaremos profesor, y por otro un rol que llamaremos alumno. El rol de profesor permite dar de alta a una serie de alumnos en una serie de cursos y a su vez puede monitorizar el progreso de los mismos, con acceso a una serie de estadísticas. El rol de usuario por el contrario puede realizar una serie de retos y cursos y además mediante la herramienta "Game Lab" será capaz de crear una serie de aplicaciones o juegos mediante la programación basada en bloques (pudiendo alternar con la visualización del código de manera tradicional si el usuario lo viera necesario).

Cabe destacar que para la realización de los cursos propuestos por code.org el usuario no necesita tener una serie de conocimientos de programación pre-

vios, pero en relación con el uso de Code Studio si es necesario al menos una familiarización o contacto anterior con la programación para el correcto uso de los bloques previamente creados y a disposición del usuario.

Así pues una vez dicho lo anterior procederemos a explicar la interfaz de Code Studio y a la codificación de un sencillo reto o juego, utilizando las herramientas que nos proveen. Finalmente se publicará el reto en la plataforma, ya que esta nos da la posibilidad de implementarlo y compartirlo con los usuarios de la misma, creando una plataforma con conocimiento compartido el cual puede ser consultado a la hora de crear futuros proyectos.

4.1.1. Herramienta Game Lab de code.org

Game Lab es una herramienta o entorno de programación basado en bloques de la plataforma code.org [5], donde cualquier usuario con nociones de programación (aunque no son completamente necesarias es recomendable ya que se tratan tópicos relacionados con la programación que de primeras pueden ser complicados para alguien que nunca ha visto nada relacionado con este ámbito) puede hacer animaciones simples o juegos con objetos que interactúe entre sí . Así pues es posible diseñar tu aplicación, codificarla mediante las funciones previamente dadas en forma de bloques o directamente escribiendo el código en JavaScript (pudiendo cambiar en todo momento entre estas dos opciones).

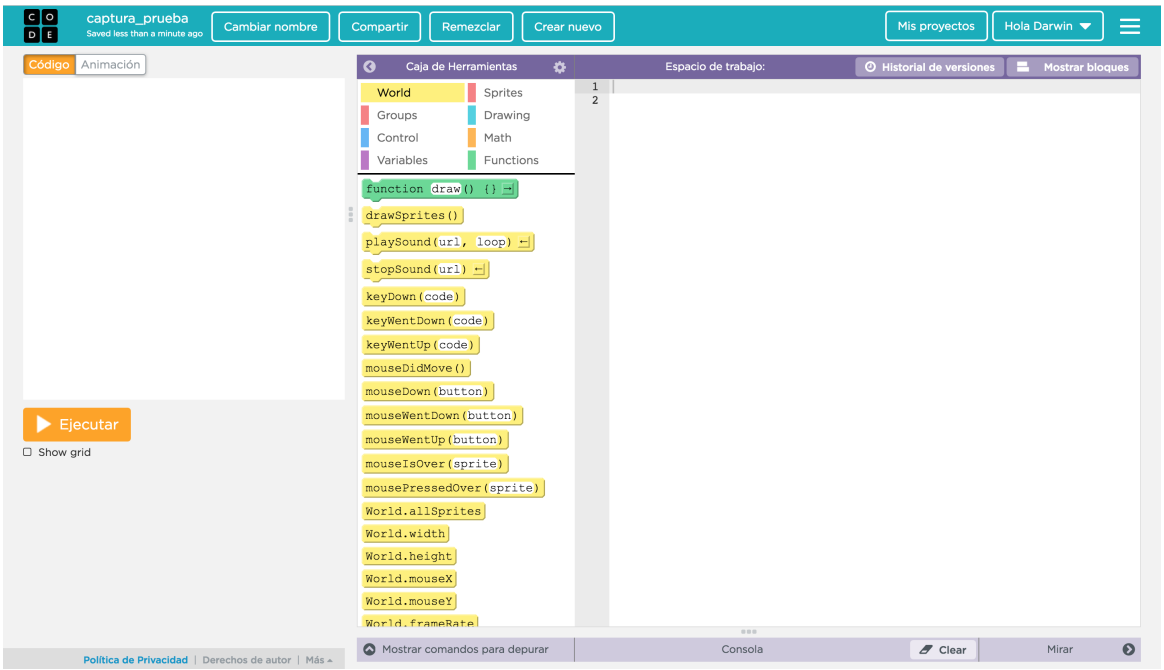


Figura 4.1: Interfaz GameLab

4.1.2. Análisis de la interfaz de Game Lab

Para analizar la interfaz de Game Lab podemos dividir el entorno de trabajo en dos partes claramente distinguidas. Por una parte en el lado izquierdo tenemos la parte donde se visualiza la ejecución del código, es decir, cuando le demos al botón de "Ejecutar". Además en la parte superior donde se muestra el tablero de juego podemos encontrar dos pestañas para cambiar entre el modo donde se muestra el código de la aplicación y la ventana de animaciones en las que nos llevará a un editor de imágenes donde podemos editar nuestros sprites y elementos del juego con posibilidad de cambiar su tamaño, su apariencia, su resolución y cualquier aspecto relacionado con la imagen que se mostrará al ejecutar la animación. Cabe destacar que a la hora de añadir una animación Game Lab aporta un catalogo extenso de elementos que podemos añadir en el entorno pero además de todo esto el usuario puede subir las imágenes personalizadas desde su ordenador y asignársela a cualquier objeto o elemento del juego.

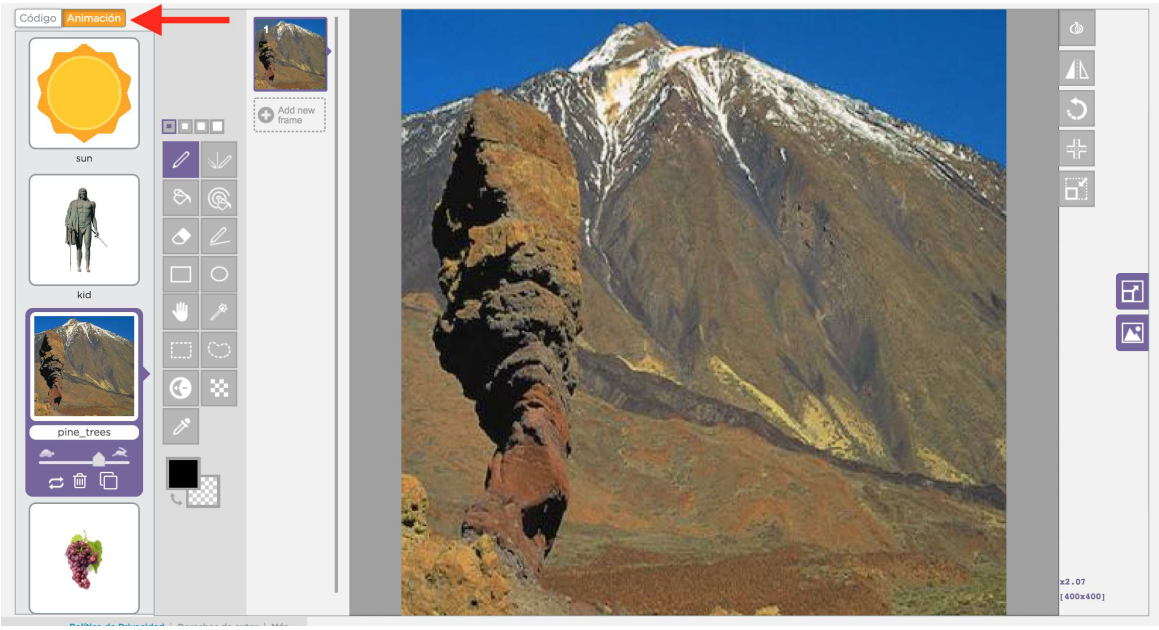


Figura 4.2: Animaciones reto usuarios

En la parte derecha se encuentra el espacio de trabajo donde se realiza la codificación del reto o juego. En este espacio se encuentra una librería con bloques de código que se pueden utilizar y arrastrar a el espacio de trabajo, pudiendo cambiar en todo momento a un modo tradicional en el que mediante una transición se pasa a modo texto para las personas que así lo prefieran. Además cada uno de los bloques tiene un enlace a una página en la que se encuentran diversos ejemplos de uso de cada bloque, con una documentación precisa y de calidad.

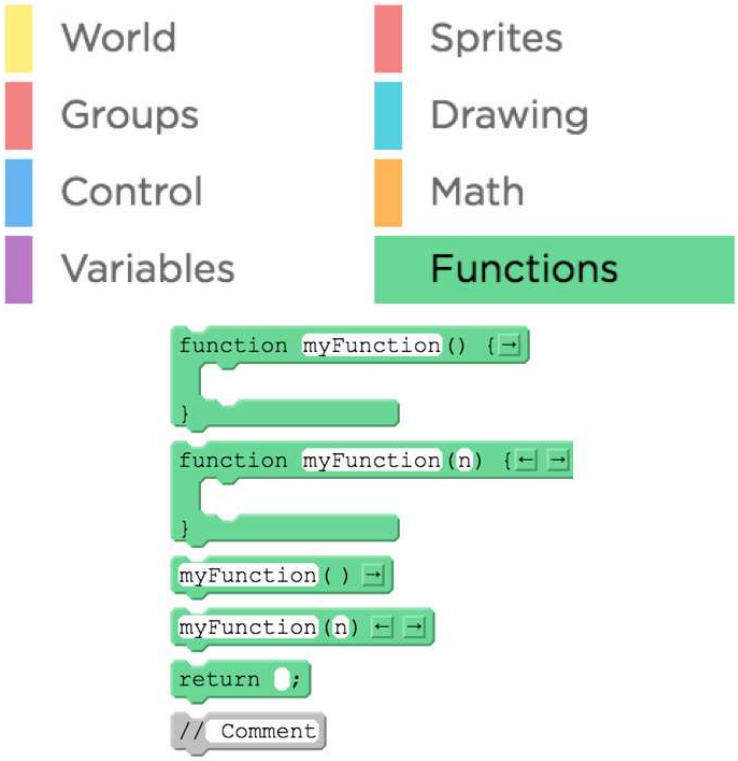


Figura 4.3: Bloques de GameLab

4.1.3. Implementación de reto en Game Lab

Una vez llegados a este punto se procederá a explicar el funcionamiento del reto. Se ha propuesto desarrollar un juego utilizando las herramientas de Game Lab, más concretamente el modo de programación visual basado en bloques. En primer lugar se han creado diversos sprites o animaciones para los elementos que componen el juego tales como el guanche, las uvas, el fondo, etc. Seguidamente, se ha añadido la funcionalidad del movimiento y la interacción con las uvas (para que desaparezcan al acercarse el guanche).

Así pues, el juego consiste en ayudar a el guanche a ir desde su punto inicial hasta el punto donde se genere el racimo de uvas, utilizando para ello las flechas de direcciones de nuestro teclado. A continuación, se puede ver una imagen del reto en Game Lab y el código fuente del mismo generado automáticamente por los bloques.

A continuación, se procederá a ver el código fuente del reto. Por una parte en en la forma tradicional (texto plano), y seguidamente utilizando los diferentes bloques que ofrece la herramienta Game Lab.

En este código se pueden observar las diferentes funciones que componen el juego. En la primera parte se puede observar las declaraciones de las animaciones que componen el entorno del juego y la escala o tamaño que tendrán

```
//Las animaciones deben estar declaradas en el orden en el que se quieran mostrar
//Declaración animación "teide" o fondo de pantalla del juego
var backgroundSprite = createSprite(200, 200);
backgroundSprite.setAnimation("teide");

//Declaración animación "guanche"
var kid = createSprite(300, 325);
kid.setAnimation("guanche");
kid.scale = 0.3;

//Declaración animación "sol"
var sun = createSprite(330, 60);
sun.setAnimation("sun");
sun.scale = 0.6;

//Declaración animación "uvas"
var uvas =  createSprite(randomNumber(100,300), randomNumber(100,300));
uvas.setAnimation("uvas");
uvas.scale = 0.8;

function draw() {
  //Dibuja las diferentes animaciones declaradas anteriormente en el espacio principal
  drawSprites();

  //Movimiento del personaje principal
  if(keyDown("right")){
    kid.x = kid.x + 1;
  }
  if(keyDown("left")){
    kid.x = kid.x - 1;
  }

  if(keyWentUp("up")){
    kid.y = kid.y - 15;
  }

  if(keyWentDown("down")){
    kid.y = kid.y + 10;
  }

  //LLamada a la función que detecta si las uvas están cerca del guanche
  cerca();

}

function cerca() {

  if(abs(kid.x -uvas.x) <=20 && abs(kid.y -uvas.y) <= 120 ) {
    playSound("sound://default.mp3", false);
    uvas.destroy();
    return true;
  }

}

```

Figura 4.4: Reto formato tradicional

con respecto al tablero. En la segunda parte se dibujan las animaciones y se controla el movimiento del personaje principal, añadiendo además la función que comprueba si el personaje principal se encuentra cerca de su objetivo, en este caso, las uvas.

Figura 4.5: Formato bloques de GameLab

Finalmente se puede observar el código generado al arrastar los bloques al espacio de trabajo de GameLab, representando con diferentes colores dependiendo de la función que desempeñen en el programa. Cabe destacar que la conversión del formato tradicional al formato en bloques es totalmente automática, teniendo que pulsar simplemente el botón "Mostrar Bloques / Mostrar Código".

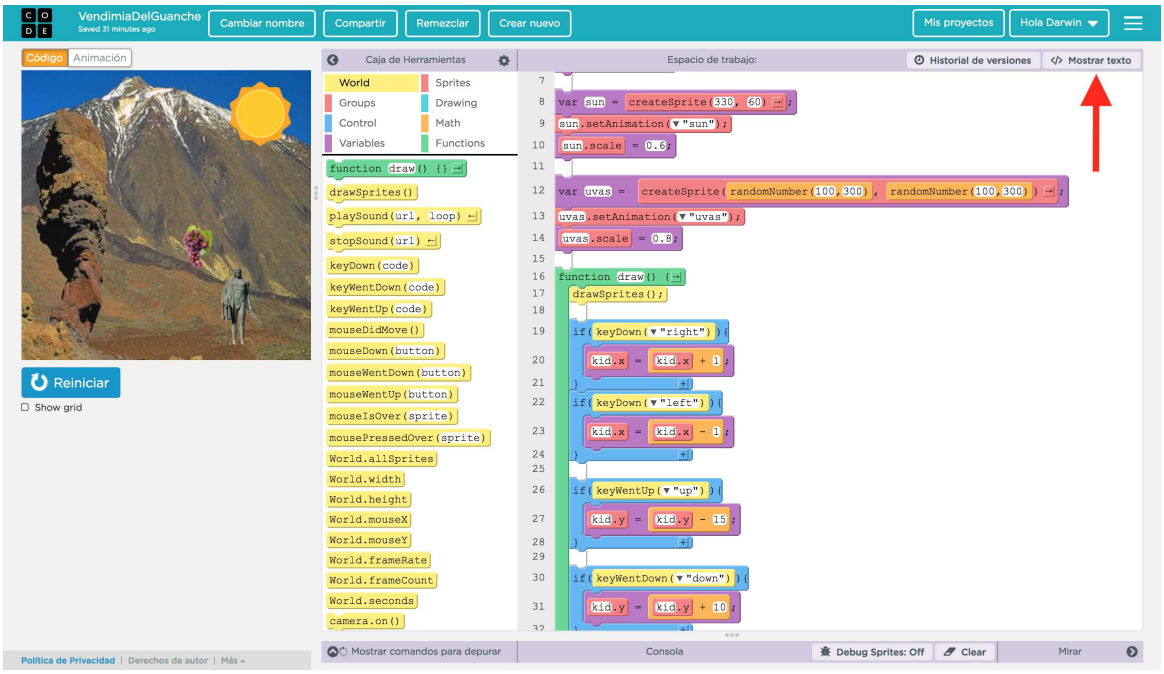


Figura 4.6: Modo vistas GameLab

4.2. Reto creado con Blockly

En esta otra sección del capítulo 4 se explicarán los diferentes pasos que se han realizado para la codificación y puesta en marcha de un reto. Este ha sido implementado utilizando una librería escrita en JavaScript de código abierto denominada Blockly [13], vista anteriormente en el Capítulo 3 donde se explicó con detalle cada una de sus características y modo de empleo. Así pues, se tiene como objetivo la realización de un reto utilizando la programación visual o programación basada en bloques.

4.2.1. Propuesta del Reto

En cuánto al reto propuesto se ha decidido la realización de una aplicación web en la cual tenemos dos principales elementos, por una parte un niño y por otra una fruta animada. El objetivo del reto no es otro que mediante la unión de una serie de bloques proporcionados, en un orden fijado previamente, conseguir que el niño llegue desde su punto de partida hasta la fruta. Así pues, podemos

determinar que el niño es el elemento móvil cuyos movimientos vendrán dados por el orden en el cual se unan los bloques proporcionados y la fruta es el objetivo o meta que se debe conseguir. Una vez aclarado el concepto del juego, se ha determinado que este este distribuido en diez niveles de dificultad, yendo desde más fácil a más difícil a medida que el usuario avanza por los diferentes niveles.

Para la implementación del mismo se ha tomado como modelo de referencia uno de los juegos desarrollados por Google en su plataforma Blockly Games [14] en la cual hay una serie de juegos desarrollados mediante Blockly totalmente gratuitos y de código abierto, permitiendo trabajar sobre ellos. El código de estos juegos puede ser descargado ya que se encuentra en un repositorio [15] en la plataforma Github, además en el se encuentran una serie de manuales de uso y instrucciones de despliegue.

4.2.2. Codificación e implementación del reto

Para comenzar a explicar la implementación de este reto se comenzará con una breve descripción de los diferentes elementos y archivos que componen el mismo. Así pues, pese a que se mostrará el conjunto de directorios y archivos que componen el reto, la explicación de este apartado se centrará en la descripción de dos archivos en concreto donde se recogen las principales funcionalidades de nuestro reto, y las que de verdad interesa explicar. Estos archivos de denominan `maze.js` y `blocks.js`.

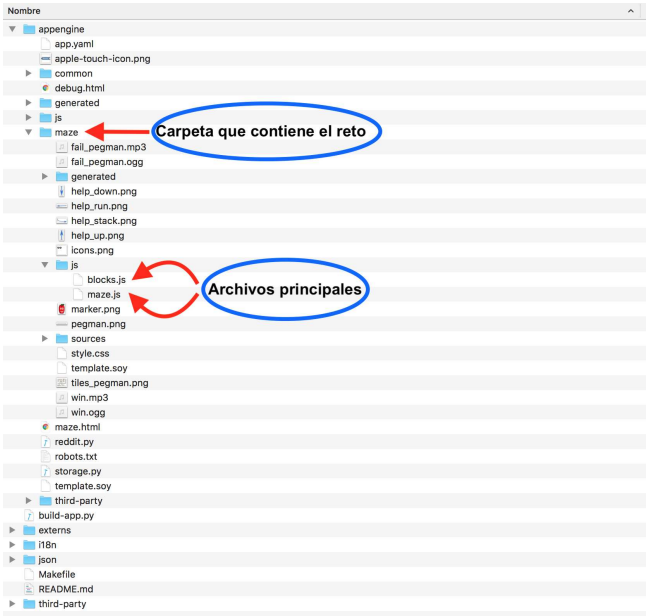


Figura 4.7: Estructura de directorio reto desarrollador

En primer lugar comenzaremos explicando el contenido del archivo `blocks.js`. En él se definen los diferentes bloques que estarán disponibles en el espacio denominado "Caja de herramientas" o "Toolbox", por ejemplo el bloque `IF, MOVEFORWARD, TURN, IFELSE, FOREVER`. Estos bloques estarán formados por dos partes tal y como se explicó en el punto 3.4 Capítulo 3, su parte de inicialización o declaración y su parte generadora de código. A continuación se mostrará uno de los bloques contenidos en el archivo `blocks.js`, más concretamente el bloque `IF`.

```
Blockly.Blocks['maze_if'] = {
  /**
   * Block for 'if' conditional if there is a path.
   * @this Blockly.Block
   */
  init: function() {
    var DIRECTIONS =
        [[BlocklyGames.getMsg('Maze_pathAhead'), 'isPathForward'],
         [BlocklyGames.getMsg('Maze_pathLeft'), 'isPathLeft'],
         [BlocklyGames.getMsg('Maze_pathRight'), 'isPathRight']];
    // Append arrows to direction messages.
    DIRECTIONS[1][0] += Maze.Blocks.LEFT_TURN;
    DIRECTIONS[2][0] += Maze.Blocks.RIGHT_TURN;
    this.setColour(Maze.Blocks.LOGIC_HUE);
    this.appendDummyInput()
        .appendField(new Blockly.FieldDropdown(DIRECTIONS), 'DIR');
    this.appendStatementInput('DO')
        .appendField(BlocklyGames.getMsg('Maze_doCode'));
    this.setTooltip(BlocklyGames.getMsg('Maze_ifTooltip'));
    this.setPreviousStatement(true);
    this.setNextStatement(true);
  }
};

Blockly.JavaScript['maze_if'] = function(block) {
  // Generate JavaScript for 'if' conditional if there is a path.
  var argument = block.getFieldValue('DIR') +
      '(\'block_id_' + block.id + '\')';
  var branch = Blockly.JavaScript.statementToCode(block, 'DO');
  var code = 'if (' + argument + ') {\n' + branch + '}\n';
  return code;
};
```

Figura 4.8: Inicialización y generador bloque en archivo blocks.js

Por otra parte tenemos el archivo `maze.js` el cual se podría considerar como núcleo de la apliación o del reto en cuestión. En él se construyen e instancian todos los elementos que aparecerán en la pantalla principal del reto, en este caso, serán mostrados a través del archivo llamado `maze.hmtl`. Así pues, este archivo es el encargado entre otras cosas de:

- Crear e implementar las diferentes interacciones y elementos dentro de la ventana principal del reto.

- Instanciar e inicializar una serie de niveles predefinidos.
- Implementación de funciones de control encargadas del movimiento del personaje y de los eventos que se produzcan.
- Control de la interfaz y de la interacción con el usuario a tráves de cuadros de diálogo.
- Control de manejo de bloques y comprobación de la resolución de cada nivel.

Construción previa del reto

Para utilizar el siguiente código es necesario seguir una serie de pasos que se exponen a continuación. En primer lugar es necesario obtener las dependencias y construir las mismas. Para ellos debemos posicionarnos dentro de la carpeta de el reto:

- Desplazarse a la carpeta del reto `$ cd TFG_dev`
- Ejecutar el comando: `$ make deps`

Una vez hecho esto se debe construir el juego en sí, en este caso se construirá el reto del tipo "Laberinto clásico", denominado en ingles como "maze" y se construirá de la siguiente forma:

- `$ make maze-en`

Por último solo queda ver el resultado de la construcción del reto, para ello debemos abrir en el navegador la siguiente ruta `/TFG_dev/appengine/maze.html`. Finalmente cabe destacar que si se realiza algún tipo de modificación en el código fuente del proyecto habrá que reconstruir con el make el código para que los cambios sean efectivos.

Interfaz del reto de desarrollador

Una vez se ha dado una visión general de cómo está distribuido el código fuente del reto y cuáles son sus principales elementos se procederá a explicar la interfaz y funcionalidad de este reto. Para comenzar se puede apreciar dos partes bien diferenciadas en la interfaz del mismo, por una parte en el lado izquierdo de la imagen se sitúa el tablero en el cúal podemos visualizar tanto el niño como su objetivo, la fruta animada. Además en este mismo sector de la interfaz, en la parte inferior, contamos con un botón que servirá para ejecutar el programa cuando se crea conveniente probar la solución dada mediante el

desplazamiento de los bloques proporcionados.

Por otra parte en el lado derecho de la imagen se encuentra el espacio denominado 'Toolbox" con una serie de bloques (estos varían dependiendo del nivel en el que se encuentre y de la dificultad del mismo, tal y como se le indica en el archivo `maze.js`) y el espacio de trabajo donde se deberán arrastrar los bloques, en el que además se encuentra una pequeña papelera para eliminar los bloques que se desee.

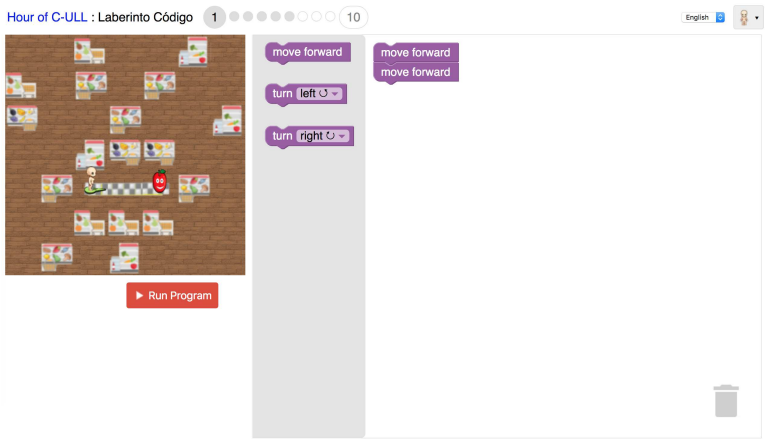


Figura 4.9: Interfaz reto desarrollador

Tal y como se puede ver en la interfaz se encuentran todos los elementos del reto, pero estos son creados previamente mediante la compilación del mismo, es decir, es necesario construir de forma previa nuestro reto con una serie de comandos que se verán a continuación. El archivo `maze.html` no contiene todos estos apartados desde un principio, sino que mediante el archivo `maze.js` se van creando y se van insertando en el archivo con la interfaz final. Dicho esto se procederá a ver el contenido del archivo `maze.html` y los diferentes comandos necesarios de cara a compilar y construir nuestro reto.

maze.html ×

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <meta name="google" value="notranslate">
6      <meta name="viewport" content="target-densitydpi=device-dpi, width=device-width, initial-scale=1.0, user-scalable=no">
7      <title>Blockly Games : Laberinto de código</title>
8      <link rel="stylesheet" href="common/common.css">
9      <link rel="stylesheet" href="maze/style.css">
10     <script src="common/boot.js"></script>
11     <script src="/common/storage.js"></script>
12   </head>
13   <body>
14   </body>
15 </html>
16
```

Figura 4.10: Contenido estático del archivo maze.html

Distribución de los niveles y dificultad asociada

En este apartado se procederá a explicar cómo se realiza la construcción de niveles y la inicialización de los mismos con los diferentes elementos que componen el tablero del reto. La inicialización y creación de los diferentes niveles que componen el reto de hace de forma integra en el archivo previamente nombrado `maze.js`. En el en primer lugar se crean una serie de matrices con una serie de números que representan cada uno a un elemento, por ejemplo:

- El `[0]` representa el entorno por el que el usuario no puede desplazarse(textura de madera y estanterías supermercado).
- El `[1]` representa el camino por el que el usuario puede desplazarse.
- El `[2]` representa el elemento móvil, el niño que irá a por la fruta animada.
- El `[3]` representa el objetivo a conseguir, punto final o meta. Es representado por una manzana.

```
Maze.SquareType = {
  WALL: 0,
  OPEN: 1,
  START: 2,
  FINISH: 3
};

// The maze square constants defined above are inlined here
// for ease of reading and writing the static mazes.
Maze.map = [
// Level 0.
 undefined,
// Level 1.
 [[0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0],
  [0, 0, 2, 1, 3, 0, 0],
  [0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0]],
// Level 2.
 [[0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 1, 3, 0, 0, 0],
  [0, 0, 2, 1, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0, 0]],
// Level 3.
```

Figura 4.11: Inicialización de la distribución inicial niveles

Capítulo 5

Conclusiones y líneas futuras

En este capítulo se expondrán las conclusiones a las que se ha llegado después de la realización del trabajo y las posibles líneas de actuación futuras.

5.1. Conclusiones

El papel del pensamiento computacional es primordial en la sociedad actual, y lo será aún más en el futuro. La aplicación de esta técnica de resolución de problemas hace que sea necesario enseñarla desde edades tempranas, de la misma forma que se enseña a leer o a escribir.

La manera más sencilla de introducir el pensamiento computacional a edades tempranas es a través de la programación visual basada en bloques tal y como se ha visto a lo largo del desarrollo de este trabajo, aunque cabe destacar que no es la única forma. Así pues, es aquí donde plataformas como code.org están haciendo un trabajo excelente ofertando los diferentes cursos para hacer la programación algo entretenido y divertido a la par que didáctico.

Desde el punto de vista técnico, la plataforma code.org es un servicio web. Cuenta con el servidor de Code Studio desarrollado en Ruby on Rails el cual es el responsable de todo lo relacionado con la información de los usuarios y del almacenamiento del progreso en cada uno de sus cursos o actividades. Por otra parte, Sinatra es el encargado del servidor de la plataforma web en sí, el cual alberga desde la página principal hasta el panel de control de los usuarios que pertenecen a el rol de profesor. Finalmente y no por ello menos importante JavaScript es el principal motor en el que se basan las funcionalidades de todos los puzzles, cursos y tutoriales que en está plataforma se utilizan, eso sí, cabe destacar que se hace uso de la tecnología Babel [20], un transcompilador que permite la utilización de las funcionalidades disponibles en el estándar de

ECMAScript 6. En relación con las tecnologías web que se utilizan para los estilos o creación de elementos en la plataforma podemos encontrar REACT [27], HTML [2] y CSS [7] como las principales responsables de esta tarea.

5.2. Líneas futuras

Una vez realizadas las diferentes tareas que han sido propuestas a lo largo del desarrollo de este trabajo se enumeran a continuación una serie de actividades relacionadas directamente con su mejora:

- Integración de reto en la plataforma code.org: sería interesante la posibilidad de integrar un reto de desarrollador en la plataforma code.org. Al igual que existe la posibilidad de crear un reto siendo un usuario de la plataforma code.org podría ser interesante la posibilidad de integrar retos y "Horas de Código" personalizadas desarrolladas desde el rol de desarrollador.

- Personalización niveles: desarrollar una herramienta mediante la cual el usuario pueda arrastrar elementos a el mapa y así crear sus niveles personalizados. Esta herramienta se podría denominar como modo "creativo" y en él el usuario (sin ser un requisito previo tener conocimientos en programación) tendría total libertad de personalización del mapa y sus elementos, pudiendo resolver el reto una vez haya realizado la distribución de los elementos en el mapa.

- Despliegue de reto: debido a que en gran parte este reto está desarrollado en tecnologías principalmente relacionadas con la web no debería resultar difícil el despliegue de este reto de desarrollador en un servidor. El despliegue del mismo permitiría un seguimiento de la actividad del mismo, lo que lleva a plantearnos el siguiente punto.

- Obtención de estadísticas, datos y resultados: una vez se tuviera el servidor desplegado con el reto se podrían llevar a cabo una serie de recogida de datos como por ejemplo el número de errores que tiene un usuario de media en la resolución de cada nivel, porcentajes relacionados con el progreso del usuario, reporte de posibles problemas detectados en la aplicación, media de tiempo que el usuario tarda en resolver cada nivel, etc.

- Desarrollo de plataforma PC: por último se anima a desarrollar una plataforma sencilla, de las mismas características de code.org. Esta plataforma debería ofrecer la posibilidad de agregar retos relacionados con el pensamiento computacional y podría ser desplegada de manera similar al campus virtual de la Universidad de La Laguna, pudiendo ser ofertada a

colegios y demás centros que la demandarán. Así se obtendría datos de gran valor y además se llevaría a cabo la principal tarea a tratar en este trabajo, la evaluación del entrenamiento del PC a través de una plataforma, en este caso propia.

Capítulo 6

Summary and Conclusions

In this chapter will be the conclusions that have been determined throughout the accomplishment of the work.

6.1. First Section

The role of computational thinking is paramount in today's society, and will be even more so in the future. The application of this technique problem solving makes it necessary to teach it from an early age, in the same way that is taught to read or write.

The simplest way to introduce computer thinking at an early age is through programming block-based visual as seen throughout the development of this work, although it should be noted that it is not the only way. So, this is where platforms like code.org are doing an excellent job offering the different courses to make the programming something entertaining and fun as well as didactic.

From the technical point of view, the code.org platform is a web service. It has the Code Studio server developed in Ruby on Rails which is responsible for everything related to user information and the storage of progress in each of its courses or activities. On the other hand, Sinatra is in charge of the server of the web platform itself, which hosts from the main page to the control panel of users who belong to the role of teacher. Last but not least, JavaScript is the main engine on which the functionalities of all the puzzles, courses and tutorials that are used in this platform are based. However, it is worth mentioning that Babel technology is used, a transcompiler that allows the use of the functionalities available in the ECMAScript 6 standard. In relation to the web technologies that can be used to create elements in the platform or to manage the style of the platform, REACT, HTML and CSS are used as the main responsible for this task.

Capítulo 7

Presupuesto

Este capítulo recoge un modelo de presupuesto para el tiempo empleado en este trabajo de fin de grado.

7.1. Desglose presupuesto

Tareas	**Horas**	**Presupuesto**
Revisión bibliográfica	35h	7 €/h
Selección plataforma de desarrollo del PC	10h	5 €/h
Estudio diseño de la plataforma elegida	65h	15 €/h
Documentación sobre tecnologías implicadas	30h	10 €/h
Codificación de retos y validación	90h	25 €/h
Documentación y difusión de los resultados	10h	20 €/h
TOTAL	**240h**	**4020€**

Tabla 7.1: Tabla modelo de presupuesto

En la tabla anterior se recoge una relación entre las horas invertidas en cada una de las tareas realizadas en este trabajo y el precio estipulado de cada hora, en relación a la actividad realizada. Así pues, por un total de 240 horas invertidas la cantidad necesaria asciende a un total de 4020.00 euros.

Bibliografía

[1] Apple. Xcode. `https://developer.apple.com/xcode/`, 2003. Acceso: 2018-05-20.

[2] Tim Berners-Lee. Html. `https://www.w3.org/html/`, 1993. Acceso: 2018-05-31.

[3] Ecma International Brendan Eich. Ecmascript 6. `http://es6-features.org/#Constants`, 2015. Acceso: 2018-05-28.

[4] Neil C.C. Brown, Jens Mönig, Anthony Bau, and David Weintrop. Panel: Future directions of block-based programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 315–316, New York, NY, USA, 2016. ACM.

[5] Code.org. Code ong. `https://code.org/`, 2012. Acceso: 2018-05-09.

[6] Code.org. Repositorio github plataforma code-dot-org. `https://github.com/code-dot-org/code-dot-org`, 2014. Acceso: 2018-05-09.

[7] World Wide Web Consortium. Css. `https://www.w3.org/Style/CSS/`, 1996. Acceso: 2018-05-31.

[8] J. Du, H. Wimmer, and R. Rada. "hour of code": Can it change students' attitudes toward programming? *Journal of Information Technology Education: Innovations in Practice*, 15(1):53–73, 2016. cited By 3.

[9] Mariluz Guenaga, Iratxe Mentxaka, Pablo Garaizar, Andoni Eguiluz, Sergi Villagrasa, and Isidro Navarro. Make world, a collaborative platform to develop computational thinking and steam. In Panayiotis Zaphiris and Andri Ioannou, editors, *Learning and Collaboration Technologies. Technology in Education*, pages 50–59, Cham, 2017. Springer International Publishing.

[10] Blake Mizerany Ryan Tomayko Simon Rozet Konstantin Haase. Sinatra. `http://sinatrarb.com/`, 2011. Acceso: 2018-05-28.

[11] David Heinemeier Hansson. Ruby on rails. `https://rubyonrails.org/`, 2005. Acceso: 2018-05-28.

[12] Max Howell. Homebrew. `https://brew.sh/index_es`, 2009. Acceso: 2018-05-20.

[13] Google INC. Blockly games development description. `https://developers.google.com/blockly/`, 2011. Acceso: 2018-05-09.

[14] Google INC. Página web blockly games. `https://blockly-games.appspot.com/`, 2011. Acceso: 2018-05-28.

[15] Google INC. Repositorio github blockly games. `https://github.com/google/blockly-games`, 2011. Acceso: 2018-05-28.

[16] Google INC. Blockly developer tools. `https://blockly-demo.appspot.com/static/demos/blockfactory/index.html`, 2016. Acceso: 2018-05-20.

[17] Adam Jacob. Chef. `https://www.chef.io/`, 2011. Acceso: 2018-05-28.

[18] Node.js Developers Joyent. Nodejs. `https://nodejs.org/es/`, 2009. Acceso: 2018-05-20.

[19] Yukihiro Matsumoto. Ruby. `https://www.ruby-lang.org/es/`, 1995. Acceso: 2018-05-20.

[20] Sebastian McKenzie. Babel. `https://babeljs.io/`, 2014. Acceso: 2018-05-31.

[21] Sun Microsystems y Oracle Corporation MySQL AB. Mysql. `https://www.mysql.com/`, 1995. Acceso: 2018-05-20.

[22] Mozilla Fundation Netscape Communications Corp. Javascript. `https://www.javascript.com/`, 1995. Acceso: 2018-05-20.

[23] Mitchel Resnick. Scratc. `https://scratch.mit.edu/`, 2002. Acceso: 2018-05-29.

[24] Salvatore Sanfilippo. Redis. `https://redis.io/`, 2009. Acceso: 2018-05-20.

[25] Yarn software. Yarn. `https://yarnpkg.com/es-ES/`, 2016. Acceso: 2018-05-20.

[26] Carnegie Mellon University. Alice. `http://www.alice.org/`, 1998. Acceso: 2018-05-29.

[27] Jordan Walke. Reactjs. `https://reactjs.org/`, 2013. Acceso: 2018-05-31.

[28] Jeannette M. Wing. Computational thinking. *Commun. ACM*, 49(3):33–35, March 2006.

[29] Collin Winter. npm. `https://www.npmjs.com/`, 2014. Acceso: 2018-05-20.

[30] Miguel Zapata-Ros. Pensamiento computacional: Una nueva alfabetización digital. 2015-10-23.