



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

Trabajo Fin de Grado

Grado en Ingeniería Informática

Análisis de los resultados de los sistemas de entrenamiento del Pensamiento Computacional

*Analysis of the results of Computational Thinking
training systems*

Samuel Valcárcel Arce

La Laguna, 20 de junio de 2018

D^a. **Coromoto Antonia León Hernández**, con N.I.F. 78.605.216-W profesora Titular de Universidad adscrita al Departamento de Lenguajes y Sistemas Informáticos de la Universidad de La Laguna, como tutora

D. **Carlos Segura González**, con N.I.F. 78.704.244-S profesor Titular adscrito al Área de Ciencias de la Computación del Centro de Investigación Matemática en Guanajuato, México, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

“Análisis de los resultados de los sistemas de entrenamiento del Pensamiento Computacional”

ha sido realizada bajo su dirección por D. **Samuel Valcárcel Arce**, con N.I.F. 54.063.506-M.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 20 de junio de 2018

Agradecimientos

XXX

XXX

XXX

XXX

Licencia



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

Resumen

El objetivo de este trabajo ha sido bla, bla, bla bla, bla, bla bla, bla, bla

La competencia [E6], que figura en la guía docente, indica que en la memoria del trabajo se ha de incluir: antecedentes, problemática o estado del arte, objetivos, fases y desarrollo del proyecto, conclusiones, y líneas futuras.

Se ha incluido el apartado de 'Licencia' con todas las posibles licencias abiertas (Creative Commons). En el caso en que se decida hacer público el contenido de la memoria, habrá que elegir una de ellas (y borrar las demás). La decisión de hacer pública o no la memoria se indica en el momento de subir la memoria a la Sede Electrónica de la ULL, paso necesario en el proceso de presentación del TFG.

El documento de memoria debe tener un máximo de 50 páginas.

No se deben dejar páginas en blanco al comenzar un capítulo, ya que el documento no está pensado para ser impreso sino visionado con un lector de PDFs.

También es recomendable márgenes pequeños ya que, al firmar digitalmente por la Sede, se coloca un marco alrededor del texto original.

El tipo de letra base ha de ser de 14ptos.

Palabras clave: Palabra reservada1, Palabra reservada2, ...

Abstract

Here should be the abstract in a foreing language...

Keywords: *Keyword1, Keyword2, Keyword3, ...*

Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Metodología y plan de trabajo	2
2. Antecedentes y estado actual del tema	4
2.1. Estado del arte	4
2.2. Aplicaciones relacionadas	5
2.2.1. Code.org	6
2.2.2. Codecademy	6
2.2.3. Programamos	7
3. Tecnología utilizada	9
3.1. Ruby on Rails	9
3.2. Active Record	10
3.3. Bootstrap	10
3.4. Github	11
3.5. Chartkick	11
3.6. Pruebas con RSpec	12
4. Plataforma	13
4.1. Arquitectura de la aplicación	13
4.1.1. CodeCharts	13
4.1.2. Base de datos	14
4.1.3. Gemas utilizadas	14
4.2. Diseño de la aplicación	15
4.2.1. Página de inicio	15
4.2.2. Registro/Inicio de sesión	16
4.2.3. Cursos creados	16
4.2.4. Perfil del profesor	17
4.2.5. Detalles del curso	18
4.2.6. Detalles de la sección	19

5. Verificación y pruebas	22
5.1. Problemas encontrados	22
5.2. Testeo de la aplicación	22
6. Conclusiones y líneas futuras	25
7. Summary and conclusions	26
7.1. Sección Uno	26
8. Presupuesto	27
A. Título del Apéndice 1	28
A.1. Algoritmo XXX	28
A.2. Algoritmo YYY	28
B. Título del Apéndice 2	30
B.1. Otro apéndice: Sección 1	30
B.2. Otro apéndice: Sección 2	30
Bibliografía	30

Índice de figuras

2.1. Code.org	6
2.2. Codecademy	7
2.3. Programamos	7
3.1. Ruby on Rails	9
3.2. SQLite	10
3.3. Bootstrap	10
3.4. Github	11
3.5. Captura de ejemplo de la aplicación	11
4.1. Esquema de la Base de Datos	14
4.2. Registro e inicio de sesión	16
5.1. Pruebas realizadas al modelo de cursos	23
5.2. Pruebas realizadas al modelo de las secciones	23
5.3. Pruebas realizadas al modelo de las secciones	24

Índice de tablas

7.1. Tabla resumen de los Tipos	26
---	----

Capítulo 1

Introducción

En este apartado se detallan los diferentes motivos y objetivos a desarrollar, así como la metodología y el plan de trabajo llevado a cabo para la realización del Trabajo Fin de Grado.

1.1. Motivación

La tecnología con el paso de los años ha ido avanzando a pasos agigantados, de manera que cada día sea más accesible a todas y cada una de las personas que hacen uso de ella en mayor o menor medida. Esto ayuda a que sectores externos a la Informática, como la educación, avancen en materia de nuevos conocimientos, surgiendo así términos como el **Pensamiento Computacional-PC** [21].

Podemos entender como pensamiento computacional la capacidad del ser humano para solucionar problemas, crear sistemas y entender de qué manera se comporta el ser humano a través del ámbito de la Informática. Este tipo de “pensamiento” beneficia a los estudiantes de todas las edades, adquiriendo la capacidad de abstraer determinados problemas, que refuerzan y mejoran las habilidades intelectuales, permitiendo que se puedan aplicar en otros ámbitos.

En Internet existen diversas plataformas dedicadas a transmitir el concepto de pensamiento computacional en el ámbito educativo, como son Code.org [6], Codecademy [5], etc. Este trabajo se ha centrado en Code.org, una plataforma diseñada con la intención de permitir que este tipo de pensamiento sea divulgado tanto a alumnos de diferentes escuelas o institutos, en todas las partes del mundo, al igual que de un amplio rango de edades, de forma que desarrollen esta capacidad a través de las actividades que se disponen en la plataforma.

Debido a que en la aplicación mencionada anteriormente no se muestran los resultados de los retos realizados por los alumnos de manera gráfica, se ha desarrollado una plataforma que simule la representación de dichos datos de

los estudiantes. Esto permite que se manifieste visualmente en qué medida los alumnos están adquiriendo los conceptos que se intentan transmitir con los cursos o talleres propuestos en Code.org.

1.2. Objetivos

Puesto que Code.org ha sido la plataforma elegida para llevar a cabo el proyecto, el objetivo principal a conseguir es realizar una contribución a su servicio web a través de Github, plataforma donde tienen alojada Code.org, de forma que los profesores no sólo tengan los resultados de los alumnos en los cursos impartidos, sino también la posibilidad de observarlos de manera gráfica, permitiendo ver el progreso de sus alumnos.

En caso de no conseguir dicho hito, se realizará un servicio externo, con la intención de plasmar los datos de los alumnos en gráficas representativas. Dicho servicio simulará la plataforma de Code.org, donde un profesor registrará a los estudiantes en los diferentes cursos o talleres, y podrá observar gráficos filtrados por edades, sexo, niveles completados, etc, complementando así el servicio que ofrece la plataforma de Code.org.

1.3. Metodología y plan de trabajo

Para desarrollar el proyecto, se ha seguido un plan de trabajo que podría dividirse en diferentes etapas:

- Una primera fase, en la que se ha realizado un estudio de los beneficios que supone el aprendizaje del pensamiento computacional en el sector educativo, a través del movimiento global **Horas de Código** [10], unos talleres impartidos a alumnos en un amplio rango de edad, con la intención de fomentar el aprendizaje del pensamiento computacional.
- A continuación, se procedió a valorar las principales páginas (Code.org, Codecademy, Programamos [13]) dedicadas a plasmar este tipo de pensamiento en sus actividades. En última instancia, se decidió que Code.org era la más adecuada para realizar la contribución a los resultados.
- Debido a que se actualizaba constantemente el repositorio de Code.org en Github, con su consiguiente cambio en la estructura, se procedió a realizar una plataforma alternativa que intentara simular el funcionamiento de Code.org, con la particularidad añadida de la representación gráfica.

El resto de la memoria cuenta con un capítulo que aúna los antecedentes y estado actual del tema con respecto al pensamiento computacional, donde se abordará la explicación de algunas de las plataformas similares a la desarrollada para el proyecto.

En el capítulo 3 se explican las diferentes tecnologías utilizadas para realizar el proyecto final, complementando así al capítulo 4, en el que se expone tanto la arquitectura como el funcionamiento en detalle de la aplicación (CodeCharts).

A continuación, en el apartado 5, algunas de las pruebas y tests usadas para la verificación de la plataforma.

Capítulo 2

Antecedentes y estado actual del tema

Siguiendo el curso de la biblioteca de la ULL, haciendo uso de .Q [14], una herramienta de búsqueda de información, se indagó acerca de artículos relacionados con el Pensamiento Computacional, sobre todo de las editoriales ACM [1] (Association of Computing Machinery) y de IEEE [11] (Institute of Electrical and Electronics Engineers).

2.1. Estado del arte

Para explicar los antecedentes y el estado actual del tema en este proyecto se tendrán en cuenta las referencias citadas a continuación:

- En un artículo [22] publicado en 2017, “*Teacher Configurable Coding Challenges for Block Languages*”, se explica como una herramienta llamada **COPPER** (CustOmizable Puzzle Programming EnviRonment), desarrollada para crear puzzles de código en una cuadrícula usando lenguajes de programación basado en bloques, similar a los realizados en la plataforma Code.org “Hour of Code”, tiene el potencial de incrementar el interés y el compromiso con el pensamiento computacional.

De esta manera, se fomenta la forma de pensar computacionalmente a través de tareas que mejoren estas habilidades dentro un ambiente similar a un juego, normalmente fomentando que los estudiantes creen programas que dicten las acciones de un personaje a medida que avanza a través de un nivel de rompecabezas, similares a los de Code.org.

Este proyecto busca aumentar la utilidad de este tipo de puzzles, permitiendo a los profesores personalizar los elementos visuales y mecánicos

de los niveles que necesiten para impartir su contenido. Por ejemplo, un docente que tenga que dar una lección de historia sobre Cristóbal Colón, podría diseñar un nivel donde el estudiante escribiría el código necesario para controlar un barco, que debe navegar a través del océano y las olas del mar para llegar a tierra. Este nivel se podría complementar con diferentes preguntas, formuladas ocurridos ciertos eventos a lo largo del nivel, para comprobar si el alumno está comprendiendo la lección.

- En 2015 la revista llamada **ACM INROADS**, publicó un artículo [23] mencionando la asociación entre la plataforma Code.org y el NSF (National Science Foundation), una agencia federal independiente creada por el Congreso de los Estados Unidos en 1950 para promover el progreso de la ciencia, la salud nacional y muchos otros aspectos relevantes para el país, con lo que muchos de los alumnos en colegios sin recursos o estudiantes de color tuvieran acceso a una educación, tanto secundaria como primaria, digna en las Ciencias de la Computación.

El rol que adquiere Code.org en esta agrupación es mejorar las aptitudes adquiridas por los alumnos en el ámbito de las Ciencias de la Computación, mientras que NSF continuará apoyando la investigación de alta calidad en este sector.

- Por último, en un artículo [20] publicado en 2016, en el libro llamado “Proceedings of the 47th ACM Technical Symposium on Computer Science Education” se constata que las enseñanzas sobre la Ciencia de la Computación que se componen de actividades que usen la programación basada en bloques, como pueden ser con Scratch, Alice y las “Hour of Code” de Code.org, incentivan tanto a alumnos como profesores a indagar con más profundidad en el mundo del pensamiento computacional.

Como se resume en los artículos anteriores, poco a poco se intenta inculcar a los niños desde edades tempranas que profundicen acerca del pensamiento computacional, debido a sus beneficios de cara a afrontar nuevos retos a lo largo de su carrera estudiantil. Gracias tanto al sector de la enseñanza como a empresas externas que fomentan este tipo de aprendizaje, se conseguirá que en un futuro este tipo de conocimiento no sea algo novedoso para todos, sino algo cotidiano que los alumnos puedan aprovechar el resto de sus vidas.

2.2. Aplicaciones relacionadas

Existen actualmente diversas plataformas dedicadas a fomentar el aprendizaje del pensamiento computacional, algunas de ellas mencionadas en puntos anteriores.

2.2.1. Code.org

Code.org es una organización no gubernamental creada en 2012 fundada por Hadi y Ali Partovi, que tiene como objetivo principal motivar a la gente, sobre todo a estudiantes de diferentes rangos de edades en colegios o institutos, a aprender sobre las Ciencias Computacionales. Es el organizador principal del movimiento “Hora del Código”, en el que se incentiva a todas las personas que lo realicen a que aprendan sobre el pensamiento computacional.

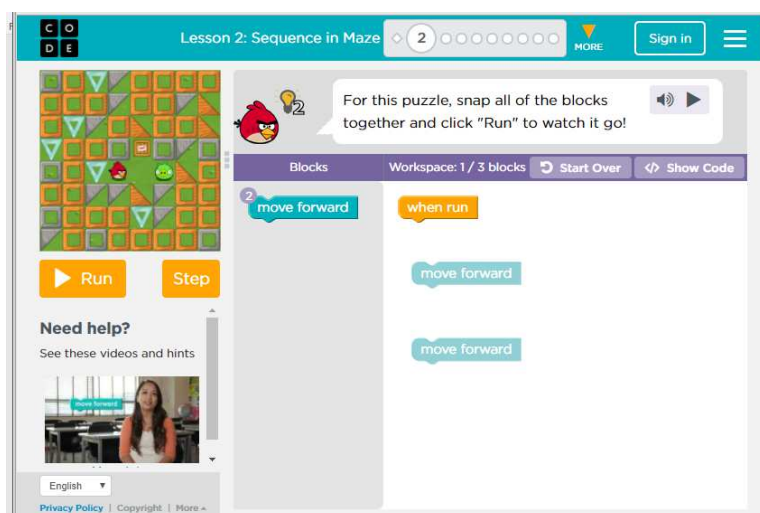


Figura 2.1: Code.org

En su web se encuentran una serie de cursos y actividades destinados a los alumnos que quieran aprender, organizados por temas y edades. Los profesores, a su vez, pueden aprender a crear talleres para enseñar a sus alumnos en la propia plataforma.

Como se explica en puntos anteriores, fue la plataforma elegida para desarrollar el Trabajo Fin de Grado, debido a su amplia variedad de cursos a la hora de fomentar el aprendizaje del pensamiento computacional. Su página está destinada a que profesores enseñen a sus alumnos a través de actividades, mientras que las otras plataformas están enfocadas a aprender de manera autodidacta.

2.2.2. Codecademy

Codecademy es una empresa enfocada principalmente a la educación y, al igual que Code.org, es una de las plataformas online más relevantes dentro del ámbito de la enseñanza de la programación en la red. Ofrece múltiples cursos gratuitos, de manera que cualquier persona pueda aprender a programar en diversos lenguajes de programación.

Los cursos incluidos en la plataforma están estructurados de manera sencilla



Figura 2.2: Codecademy

por temas (sintaxis, funciones, etc.). En caso de que el usuario tenga alguna duda dentro de cada tema, se disponen numerosas clases para dominar los diferentes apartados.

2.2.3. Programamos

Otra asociación sin ánimo de lucro se llama Programamos, con el objetivo de desarrollar el pensamiento computacional de personas de diferentes edades (desde infantil hasta formación profesional), a través de la programación (en aplicaciones móviles o videojuegos).



Figura 2.3: Programamos

Algunos de los objetivos que persigue conseguir la plataforma son:

- Cambiar el modo de interactuar entre el usuario y la tecnología, para que pase de ser solo consumidor a creador.
- Desarrollar capacidades como la creatividad, imaginación, etc.
- Permitir al profesorado formarse, de manera que posteriormente pueda enseñar a programar a los alumnos.

Al igual que se mencionaba anteriormente acerca de que en el sector estudiantil se intenta incentivar a los alumnos acerca de este tipo de conocimiento, existen empresas y organizaciones dedicadas únicamente a este ámbito. De esta manera, tanto los profesores como los alumnos se pueden beneficiar de estos proyectos.

Code.org está diseñado para que los profesores aprendan previamente acerca de como impartir este tipo de cursos o talleres sobre el pensamiento computacional y, una vez hayan adquirido el conocimiento necesario, transmitirlo a

sus alumnos. La plataforma Codecademy, en cambio, está pensado para que el usuario sea autodidacta, completando las diferentes clases que se le proponen para obtener los conocimientos básicos en un lenguaje de programación.

Capítulo 3

Tecnología utilizada

Después de realizar una selección entre las diversas tecnologías que se utilizan para el desarrollo web, en este capítulo se describen con detalle las usadas para realizar este proyecto.

3.1. Ruby on Rails

El “*Framework*” en un proyecto se puede entender como el entorno de trabajo usado para desarrollar la aplicación web, es decir, una estructura dividido en capas que indica qué programas deben o pueden ser construidos y de qué manera se relacionan.

En este proyecto, se ha usado Ruby on Rails [16], un entorno Open Source (de código abierto) para el diseño de toda la plataforma. Rails fue lanzado en 2003 por David Heinemeier Hansson y desde ese momento se ha ido extendiendo entre la comunidad de programadores, consiguiendo así una expansión y una gran comunidad.



Figura 3.1: Ruby on Rails

Alguna de las características que definen a este *framework* son:

- Usa el estilo de arquitectura Modelo-Vista-Controlador (MVC), que separa los datos, la lógica y la interfaz de la aplicación.
- Evita la repetición de código.

- *Framework* con posibilidad de validación de formularios y manejo de sesiones.

3.2. Active Record

Para gestionar la base de datos de la aplicación, se ha optado por Active Record [2], incluido en Ruby on Rails. Active Record se podría definir como la “M” en la estructura “Modelo, Vista, Controlador”, ya que es la capa responsable de representar los datos y la lógica de la aplicación.

Rails viene integrado por defecto con SQLite para el manejo de las bases de datos, debido a su sencillez y su uso sin necesidad de un servidor externo. Se utiliza tanto en el entorno de desarrollo (development) como el de pruebas (test) del proyecto. En el caso de que el proyecto se use en el entorno de producción, no se recomienda el uso de SQLite, sino una aplicación más potente para el manejo de los datos.



Figura 3.2: SQLite

3.3. Bootstrap

Para el diseño *front-end* de la plataforma, se usó Bootstrap [3], un *framework* de código abierto que hace uso de HTML, CSS y JavaScript, de manera que el usuario pueda usarlo como base para el diseño *responsive* de la aplicación, es decir, con la intención de adecuar su correcta visualización en diferentes dispositivos. Bootstrap dispone de una gran variedad de elementos para modificar el aspecto visual de la página.



Figura 3.3: Bootstrap

3.4. Github

Github [9] es una plataforma, también de código abierto como las anteriores, de desarrollo en la que cualquier usuario puede alojar su proyecto de manera cómoda y sencilla. Debido a su compatibilidad con muchas de las aplicaciones usadas actualmente por desarrolladores, lo hacen ser la plataforma puntera en este ámbito.

GitHub

Figura 3.4: Github

Algunas de las características que lo definen son:

- Colaboración entre los desarrolladores de un proyecto dentro de un mismo repositorio.
- Control de versiones, manejo de ramas.
- Permite la posibilidad de usarlo tanto por línea de comandos, como por su interfaz gráfica de usuario.

3.5. Chartkick

El objetivo principal del proyecto fue la representación gráfica de los resultados de los alumnos una vez que realizaban ciertos cursos o talleres. Al valorar diferentes posibilidades, se decidió que Chartkick [4] era la más asequible, debido a su compatibilidad con Ruby (y otros lenguajes como Python, JavaScript, etc.) y su variedad de representaciones.

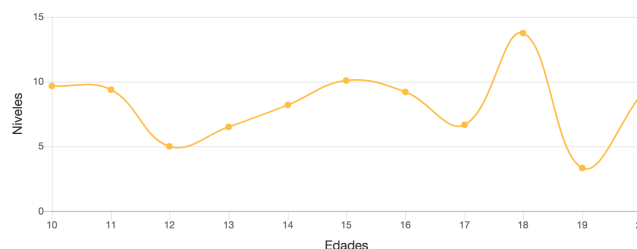


Figura 3.5: Captura de ejemplo de la aplicación

Esta herramienta permite al usuario, a partir de unos datos dados, mostrar en la aplicación diferentes gráficos con solo una línea de código.

3.6. Pruebas con RSpec

Existen diferentes herramientas para llevar a cabo la “verificación” de una aplicación, de manera que se realicen ciertas pruebas, por ejemplo en controladores o modelos, con la intención de que funcionen como se espera. En esta aplicación se usó RSpec [15], una de las utilizadas con más frecuencia en el entorno de producción.

Capítulo 4

Plataforma

La plataforma desarrollada para el Trabajo Fin de Grado, llamada “CodeCharts”, está destinada a mostrar los resultados de alumnos que han realizado cursos o talleres de Pensamiento Computacional, de manera gráfica. Permite al profesorado crear cursos e insertar todos los datos referentes a sus alumnos en ellos.



En las siguientes secciones se describen en detalle la arquitectura y el diseño de la aplicación.

4.1. Arquitectura de la aplicación

4.1.1. CodeCharts

CodeCharts fue desarrollado en Ruby on Rails, como se menciona en el punto 3.3.1, siguiendo la estructura en este tipo de proyectos “MVC” (Modelo, Vista, Controlador), así como el resto de herramientas que se describen en dicho capítulo.

4.1.2. Base de datos

Desde un principio se usó una base de datos relacional, que nos proporcionaba Rails a través de SQLite y ActiveRecord.

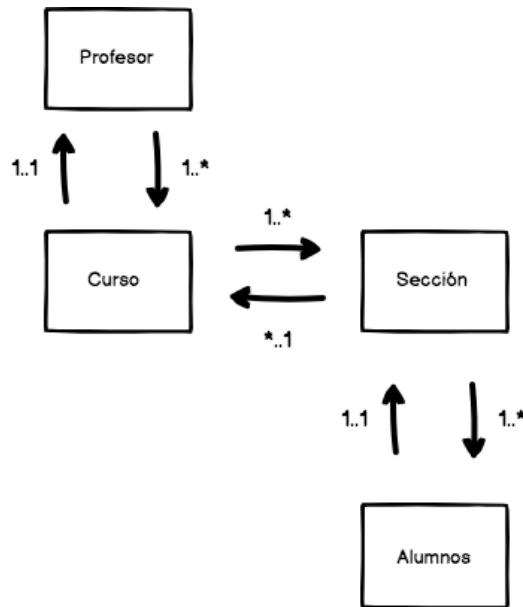


Figura 4.1: Esquema de la Base de Datos

Como se muestra en la figura 4.1, la estructura de la base de datos se divide en cuatro tablas:

- Profesor: donde se almacena toda la información referente a los profesores que se registren en la plataforma.
- Curso: creados por el profesorado, se guarda la información detallada del curso (título, descripción, etc.).
- Sección: se podría definir como el lugar de realización del curso, como por ejemplo “Arona”, que tiene registrados a los alumnos con sus resultados en el mismo.
- Alumno: tabla en la que se almacenan los datos, así como los resultados, de los estudiantes que realizan los cursos o talleres de pensamiento computacional.

4.1.3. Gemas utilizadas

Una de las funcionalidades y características que tiene Ruby, el lenguaje usado para desarrollar el proyecto, es la incorporación de gemas, al igual que ocurre con el entorno NodeJS y sus paquetes “npm”.

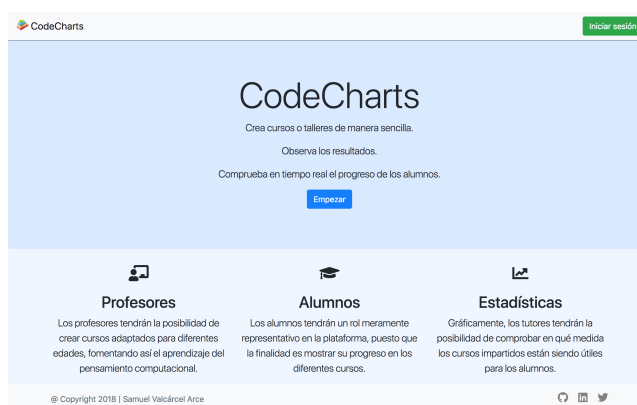
CodeCharts posee algunas relevantes como:

- **Rails**, la gema que incluye Ruby on Rails por defecto para la utilización del framework.
- **SQLite** [17], para el manejo de las bases de datos, así como ActiveRecord.
- **Devise** [7], que ofrece una solución flexible para la autenticación de los usuarios en la aplicación.
- **Bootstrap** [8], gema que permite el diseño del front-end de la plataforma con Bootstrap.
- **Will-paginate** [19], de manera que si hay una gran cantidad de usuarios dentro de una sección, permita agruparlos en páginas.
- **Wicked-pdf** [18], con lo que el profesor puede descargar en formato .PDF los resultados de los alumnos.
- **Chartkick**, herramienta que muestra en forma gráfica los resultados obtenidos en los test de los estudiantes.
- **Jquery-rails** [12], que incluye JQuery en la aplicación.

4.2. Diseño de la aplicación

4.2.1. Página de inicio

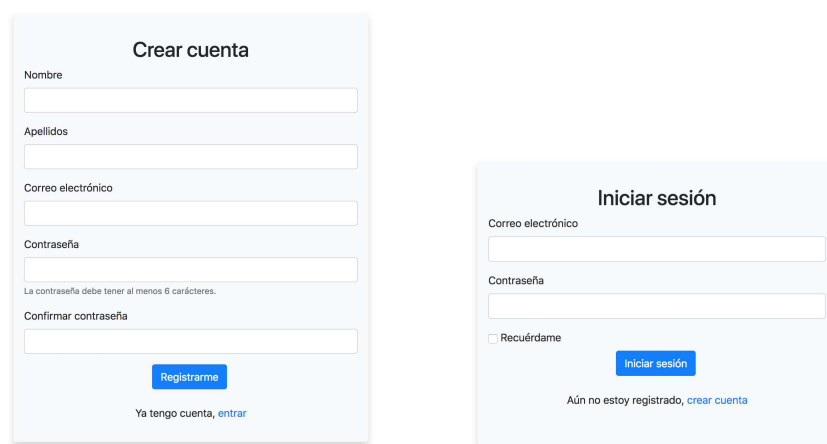
La primera vez que se accede a la aplicación, el profesor, que será el usuario a registrarse en la plataforma, verá una descripción de en qué consiste la aplicación y lo que ofrece al profesorado en los cursos impartidos.



Dentro de esta página, el profesor se podrá registrar por primera vez, o por el contrario, si ya dispone de una cuenta creada previamente, puede entrar con la misma.

4.2.2. Registro/Inicio de sesión

En las capturas que se observan a continuación, el profesor puede tanto crearse una cuenta, como iniciar con una ya creada, si ya se ha registrado previamente.



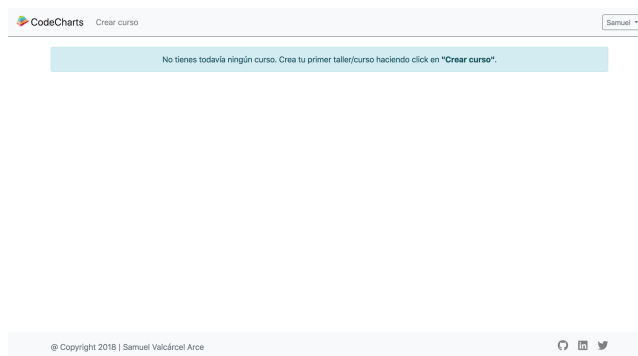
La imagen muestra dos formularios de interfaz de usuario. El formulario de la izquierda, titulado 'Crear cuenta', contiene campos para 'Nombre', 'Apellidos', 'Correo electrónico', 'Contraseña' (con una advertencia de longitud mínima de 6 caracteres) y 'Confirmar contraseña'. Incluye un botón 'Registrarme' y un enlace 'Ya tengo cuenta, entrar'. El formulario de la derecha, titulado 'Iniciar sesión', contiene campos para 'Correo electrónico' y 'Contraseña', una opción 'Recuérdame' y un botón 'Iniciar sesión'. Incluye un enlace 'Aún no estoy registrado, crear cuenta'.

Figura 4.2: Registro e inicio de sesión

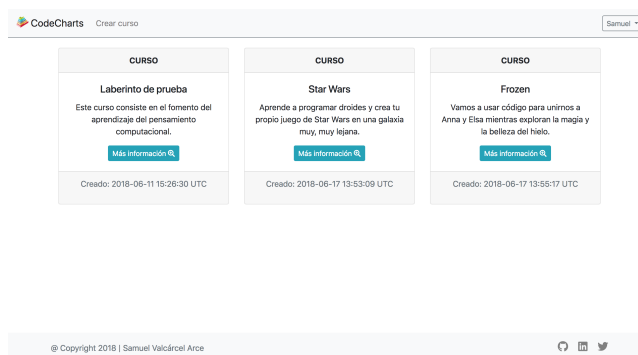
El formulario de *creación de usuario* para la plataforma es muy sencillo, dado que consta de los datos básicos que se solicitan en la gran mayoría de páginas, como su nombre y apellidos, su correo electrónico y una contraseña.

4.2.3. Cursos creados

Una vez que el profesor ha accedido a la plataforma por primera vez, se mostrará una página como la que se observa en la figura. En ella se puede observar que se le informa al usuario que no tiene creado ningún curso/taller para sus alumnos, con lo que puede crearlos a través del botón de la parte superior izquierda.



Los cursos creados se irán mostrando al profesor en forma de cuadrícula, de manera que el tutor sepa los cursos que tiene creados, con su información asociada (título, descripción y fecha de creación).



4.2.4. Perfil del profesor

Al profesor que haya iniciado sesión en ese momento, se le permitirá la opción de editar su perfil, a la vez que borrar su cuenta en caso de que ya no vaya a usar su cuenta en la plataforma. Para que cualquier cambio, como se informa en el formulario, es necesaria la contraseña actual.

Editar perfil

Nombre

Samuel

Apellidos

Valcarcel

Correo electrónico

samuvalcarcel@gmail.com

Nueva contraseña (si no hay cambios, dejar en blanco)

La contraseña debe tener al menos 6 caracteres.

Confirmar nueva contraseña

Contraseña actual (*)

Actualizar

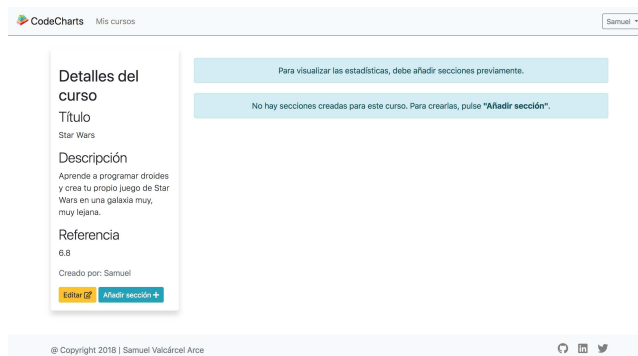
Volver

¿Desea eliminar su cuenta?

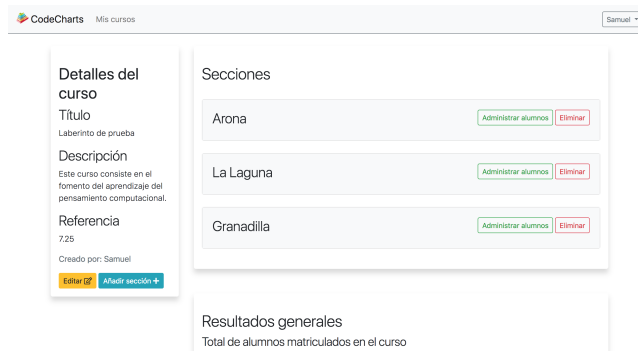
Borrar mi cuenta

4.2.5. Detalles del curso

En el momento que el docente cree el curso o taller, podrá acceder para observar los detalles del mismo, así como crear secciones dentro de la actividad.



Al igual que con la página de las actividades, se le comunica al profesor que para observar las estadísticas de ese curso, tiene que añadir el lugar (Sección) donde se celebró el taller. Uno de los detalles a destacar en el curso sería el de “Valor de referencia”, que se podría definir como un número representativo que indica el nivel que tienen los alumnos de media que realizan esa actividad.



En la captura anterior se puede observar un curso creado de ejemplo, junto con las secciones donde se realizó el curso. Tiene tanto la posibilidad de administrar a los alumnos, como se muestra en el botón derecha, al igual que eliminarlo.

4.2.6. Detalles de la sección

El objetivo final de la plataforma es que el profesor, cuando inserte los resultados de los alumnos obtenidos de la plataforma Code.org, observe con gráficos las calificaciones de cada uno.

Se pueden registrar alumnos en la plataforma tanto de manera manual (de uno en uno), como a través de un fichero .json con la información requerida, entre la que se encuentra el nombre, apellidos, edad, etc. Entre los datos más relevantes, los niveles completados en ese reto, así como las líneas de código utilizadas en total.

Al finalizar los retos por parte de los alumnos, el profesor obtendrá una tabla con toda la información de cada uno paginados, de manera que la tabla no se vuelva muy extensa al matricular a muchos alumnos, como se muestra a continuación.

Lista de alumnos Subir fichero Estadísticas

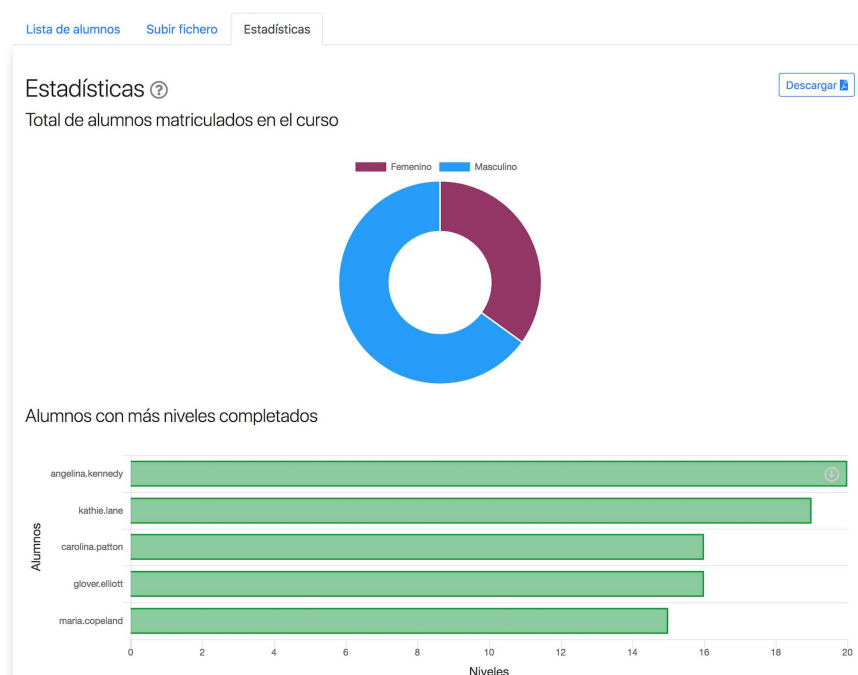
La Laguna

Alumnos matriculados ? [Añadir alumno +](#)

← 1 2 →

Nombre de usuario	Edad	Sexo	Valor de referencia del alumno	Total de líneas	Niveles completados (0-20)	
carolina.patton	11	Femenino	4.1559	32	16	Eliminar
ola.medina	12	Masculino	9.5855	94	13	Eliminar
bernadette.butler	15	Masculino	8.3674	23	6	Eliminar
roxanne.mcintyre	11	Masculino	5.4832	32	3	Eliminar
glover.elliott	11	Masculino	4.6431	97	16	Eliminar
fowler.bartlett	15	Masculino	3.5508	83	9	Eliminar
evelyn.brock	12	Masculino	8.8911	89	1	Eliminar
ellison.welch	20	Femenino	9.379	34	10	Eliminar
paul.wynn	16	Femenino	9.3226	99	0	Eliminar
sellers.wall	14	Masculino	7.1138	55	2	Eliminar

Por otro lado, en el apartado de estadísticas, se mostrarán gráficos desarrollados con la intención de ver en qué nivel se fomenta el aprendizaje en los alumnos.



En la imagen anterior se pueden observar algunos de los resultados obtenidos en forma de gráficos por los alumnos en ese curso.

Capítulo 5

Verificación y pruebas

En este capítulo se describen algunas de las pruebas realizadas para probar ciertos aspectos de la plataforma durante el desarrollo del proyecto, así como los problemas que se presentaron a la hora de realizar la aplicación.

5.1. Problemas encontrados

Durante el proceso de desarrollo del Trabajo Fin de Grado, surgieron diversos problemas para llevar a cabo la integración de la herramienta para la representación gráfica de los resultados de los alumnos.

En primer lugar, se decidió que la plataforma “Code.org” era la más completa al disponer de retos suficientes para los profesores, de manera que fomenten el aprendizaje del pensamiento computacional. A través de su repositorio en Github, donde se aloja toda la página, se realizó un estudio para comprobar de qué manera estaba estructurada la página y posteriormente se procedió a intentar realizar la contribución.

Al comprobar que su estructura se actualizaba constantemente en el repositorio y era imposible integrar la herramienta, se diseñó un servicio web externo. Este servicio, como se explica en puntos anteriores, funciona de manera similar a Code.org, con la particularidad de que se añade la información de los alumnos una vez hayan finalizado el curso en el lugar (sección) que se esté realizando.

5.2. Testeo de la aplicación

Como se describió en el apartado de “Tecnología utilizada”, las pruebas se realizaron con RSpec. Las pruebas se ejecutaron sobre alguno de los modelos de la aplicación:

- **Curso:** se validan que los atributos que se le pasan al curso de ejemplo son correctos. A su vez, si un curso tiene una o más secciones, así como insertar cursos en la aplicación.

```
RSpec.describe Course, :type => :model do
  subject { described_class.new }

  it "No se valida sin valor de referencia" do
    subject.value_reference = "Prueba de valor de referencia"
    expect(subject).to_not be_valid
  end

  it "Tiene uno o más usuarios en las secciones de ese curso" do
    should have_many(:users).through(:sections)
  end

  it "Se pueden añadir cursos" do
    @new_course = Course.new(title: "Titulo", body: "Descripcion", value_reference: "10")
    @new_course.save
    expect(Course.find_by(title: "Titulo")).to eq(@new_course)
  end
end
```

Figura 5.1: Pruebas realizadas al modelo de cursos

- **Sección:** para este modelo se comprueba que, tanto las secciones como los usuarios en las mismas, se crean correctamente, de igual manera que se pueden eliminar los usuarios.

```
RSpec.describe Section, :type => :model do
  subject { described_class.new }

  it "Se pueden añadir nuevos usuarios desde fichero" do
    @new_user = User.new(first_name: "Samuel", last_name: "Valcarcel", gender: "Chico", age: "23",
                        username: "cosaca", total_lines: "40", completed_levels: "20", section_id: "1",
                        user_value_reference: "7.5")
    @new_user.save
    expect(User.find_by(first_name: "Samuel")).to eq(@new_user)
  end

  it "Se pueden eliminar usuarios" do
    @new_user = User.new(first_name: "Samuel", last_name: "Valcarcel", gender: "Chico", age: "23",
                        username: "cosaca", total_lines: "40", completed_levels: "20", section_id: "1",
                        user_value_reference: "7.5")
    @new_user.destroy
  end

  it "Se pueden añadir secciones" do
    @new_section = Section.new(name: "Nombre")
    @new_section.save
    expect(Section.find_by(name: "Nombre")).to eq(@new_section)
  end
end
```

Figura 5.2: Pruebas realizadas al modelo de las secciones

- **Usuarios:** al igual que con la sección, se valida que se pueden introducir manualmente con éxito.

Por último, como se observa en la imagen, una vez que se ejecuta el comando “rspec -fd”, nos muestra el título de cada una de las pruebas y en la parte inferior la cantidad de pruebas que se realizaron y, en caso de errores, nos lo indicaría. En este caso, todas las pruebas pasan correctamente.

```
[samuel@MacBook-Pro-de-Samuel:~/Desktop/Memoria/TFG/code$ rspec -fd

Course
  Se comprueba con atributos válidos
  No se valida sin título
  No se valida sin descripción
  No se valida sin valor de referencia
  Tiene uno o más usuarios en las secciones de ese curso
  Se pueden añadir cursos

Section
  Se comprueba con atributos válidos
  No se valida sin título
  Pertenece a un curso
  Debe tener uno o más usuarios
  Se pueden añadir nuevos usuarios desde fichero
  Se pueden eliminar usuarios
  No se validan los usuarios sin todos los atributos
  La edad debe ser en números
  Las líneas de código deben ser en números
  Los niveles completados deben ser en números
  Se pueden añadir secciones

User
  Se pueden añadir nuevos usuarios manualmente

Finished in 0.0781 seconds (files took 1.18 seconds to load)
18 examples, 0 failures
```

Figura 5.3: Pruebas realizadas al modelo de las secciones

Capítulo 6

Conclusiones y líneas futuras

La plataforma desarrollada, llamada CodeCharts, pretende facilitar al profesorado la visualización del progreso de sus alumnos, de manera que una vez que los estudiantes hayan realizado los retos, puedan comprobar en qué medida los cursos o talleres están ayudando en el fomento del pensamiento computacional a la hora de afrontar las actividades.

Una de las peculiaridades de esta plataforma es que está pensada para que solo la use el personal de la docencia, por ello se ha desarrollado para su uso de la manera más sencilla y cómoda. En el momento que el profesor disponga de todos los datos de los alumnos, sólo debe proceder a introducirlos en la web, y ya podrá disfrutar de la funcionalidad que lo define, las estadísticas.

Capítulo 7

Summary and conclusions

Este capítulo es obligatorio. Toda memoria de Trabajo de Fin de Grado debe incluir un presupuesto.

7.1. Sección Uno

Tipos	Descripcion
AAAA	BBBB
CCCC	DDDD
EEEE	FFFF
GGGG	HHHH

Tabla 7.1: Tabla resumen de los Tipos

Capítulo 8

Presupuesto

Apéndice A

Título del Apéndice 1

A.1. Algoritmo XXX

```
*****
*
* Fichero .h
*
*****
* AUTORES
*
*
* FECHA
*
*
* DESCRIPCION
*
*
*****/
```

A.2. Algoritmo YYY

```
/*****
*
* Fichero .h
*
*****
* AUTORES
*
* FECHA
*
* DESCRIPCION
*
*****/
```

*

*****/

Apéndice B

Título del Apéndice 2

B.1. Otro apéndice: Sección 1

Texto

B.2. Otro apéndice: Sección 2

Texto

Bibliografía

- [1] ACM. <https://www.acm.org/publications/magazines>. Accedido en abril de 2018.
- [2] Active Record Basics. http://guides.rubyonrails.org/active_record_basics.html. Accedido en mayo de 2018.
- [3] Bootstrap. <https://getbootstrap.com/>. Accedido en mayo de 2018.
- [4] Chartkick. <https://www.chartkick.com/>. Accedido en mayo de 2018.
- [5] Codecademy. <https://www.codecademy.com/es>. Accedido en mayo de 2018.
- [6] Code.org. <https://code.org/>. Accedido en mayo de 2018.
- [7] Devise. <https://rubygems.org/gems/devise>. Accedido en mayo de 2018.
- [8] Gem Bootstrap. <https://rubygems.org/gems/bootstrap>. Accedido en mayo de 2018.
- [9] Github. <https://github.com/>. Accedido en mayo de 2018.
- [10] Hour of Code. <https://hourofcode.com/>. Accedido en mayo de 2018.
- [11] IEEE. <https://www.ieee.org/publications/periodicals.html>. Accedido en mayo de 2018.
- [12] JQuery-Rails. <https://rubygems.org/gems/jquery-rails>. Accedido en mayo de 2018.
- [13] Programamos. <https://programamos.es/>. Accedido en mayo de 2018.
- [14] PuntoQ. http://www.bbtck.ull.es/view/institucional/bbtck/Biblioteca_Digital/es. Accedido en abril de 2018.
- [15] RSpec. <http://rspec.info/>. Accedido en mayo de 2018.
- [16] Ruby on Rails. <https://rubyonrails.org/>. Accedido en mayo de 2018.

- [17] SQLite3. <https://rubygems.org/gems/sqlite3/versions/1.3.11?locale=es>. Accedido en mayo de 2018.
- [18] Wicked-PDF. https://rubygems.org/gems/wicked_pdf. Accedido en mayo de 2018.
- [19] Will-paginate. https://rubygems.org/gems/will_paginate. Accedido en mayo de 2018.
- [20] Neil C.C. Brown, Jens Möning, 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.
- [21] Jeannete M. Wing. Computational Thinking. *COMMUNICATIONS OF THE ACM March*, 49(3), 2006.
- [22] Nath Tumlin. Teacher configurable coding challenges for block languages. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 783–784, New York, NY, USA, 2017. ACM.
- [23] Cameron Wilson. Hour of code: Bringing research to scale. *ACM Inroads*, 6(2):18–18, May 2015.