



**INGENIERÍA EN SISTEMAS AUDIOVISUALES Y
MULTIMEDIA**

Curso Académico 2016/2017

Trabajo Fin de Grado

**CREACIÓN DE LA PLATAFORMA
DR.PENCILCODE**

Autora: Sara Blázquez Calderay

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

CREACIÓN DE LA PLATAFORMA DR.PENCILCODE

Autora : Sara Blázquez Calderay

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2017

*Dedicado a mis hermanos,
Alejandro y Nuria, os quiero.*

Agradecimientos

No ha sido un camino fácil pero he de admitir que ésta ha sido una de las mejores y, a la vez, más difíciles etapas de mi vida. Creo que no podría haber elegido una carrera mejor con todo lo que ello conlleva. Durante todos estos años, he recibido el ánimo y el cariño de muchas personas a las que quiero agradecérselo infinitamente:

En primer lugar, quiero agradecer a mi familia, me siento afortunada por teneros a todos. El mayor gracias se lo doy a mis padres que, desde pequeña, me han inculcado los valores que me han convertido en quien soy hoy. Mamá, me has enseñado a luchar por lo que quiero, a ser perseverante y a que “todo ocurre por una razón”, de mayor quiero ser como tú. Papá, gracias por apoyarme tanto y haberme enseñado a tomarme la vida con un toque de humor Blázquez, no sé qué hubiera sido de mí sin eso. Sois las personas a las que más admiro del mundo, gracias por confiar en mi más que yo misma. Os quiero mucho.

A Carlos y a Marivi, que por diversas circunstancias, se han convertido en el segundo pilar fundamental de mi día a día. Gracias por saber escucharme y guiarme a lo largo de este recorrido. Os quiero.

A mis hermanos, Alex y Nuria, que siempre sacáis lo mejor de mí. Sois el regalo más grande que he podido tener y estar con vosotros me llena de felicidad. Os adoro.

Gracias a mis amigas de siempre, Cristina, Barbara y Cristina, que a pesar de seguir rumbos distintos y no vernos a menudo siempre han estado al pie del cañón, apoyándome cuando más lo necesitaba y soportando mis batallas.

También quiero agradecer a mis amigos de la universidad, sin los cuales, llegar hasta este punto no hubiera sido posible. Gracias, a Raquel y Alba, compañeras de estudio y de vida, por haberme levantado cuando ya no podía más y por haber luchado conmigo hasta el final. Estoy orgullosa de vosotras. A Luismo, Miguel, Diego y Diana por sacarme una sonrisa siempre que os veo. Sé que sois para siempre.

Gracias a mi segunda casa, Suecia, por haberme enseñado a ser independiente y a valorar aún más lo que se tiene, por darme la posibilidad de conocer a gente tan maravillosa que, en tan poco tiempo, me ha ayudado a crecer como persona y, por haberme enseñado grandes lecciones. Aunque estéis lejos siempre os llevo conmigo.

Por último, quiero dar las gracias a mi tutor, Gregorio Robles, por haber confiado en mí y haberme ofrecido la gran oportunidad de trabajar en este proyecto que, con el tiempo, espero seguir mejorando.

Resumen

En la actualidad, las nuevas tecnologías y la programación están ganando terreno en el área de la educación; la LOMCE añade una nueva competencia clave, la competencia digital, cuyo objetivo es ayudar a los alumnos a resolver de forma eficiente los problemas de la vida cotidiana mediante el uso de recursos tecnológicos y el uso de la programación.

Dr. Pencilcode es una aplicación web de software libre y código abierto - enfocada a la enseñanza - que se encarga de realizar análisis estáticos de proyectos desarrollados con Pencil Code con la intención de impulsar la capacidad tecnológica y la creatividad de los usuarios.

El objetivo principal de este proyecto es el de crear una plataforma inteligible y amigable con el fin de ayudar y dar soporte a usuarios que se están iniciando en el mundo de la programación, en especial a niños y profesores. Además, Dr. Pencilcode, trata de promover el desarrollo del pensamiento computacional y servir de motivación a los usuarios para fomentar su deseo de mejorar como programadores mediante el uso de puntuaciones y bonus.

Las tecnologías front-end que más se han utilizado para el desarrollo de esta aplicación han sido CSS y HTML apoyándose en tecnologías como Javascript, Bootstrap y jQuery para añadir dinamismo a la web y mejorar la interactividad y usabilidad de la herramienta. Por otro lado, la parte del servidor, se ha desarrollado principalmente con el framework Django además de otras tecnologías de gestión de bases de datos como MySQL.

Summary

Nowadays, new technologies and programming are spreading in the area of education, the LOMCE adds a new key competence - Digital Competence - whose objective is to help students to solve the problems of real life efficiently by using technological resources and programming.

Dr. Pencilcode is a free software and open source web application – focused on teaching – which is responsible for performing static analysis of projects developed with Pencil Code in order to boost the technological capacity and creativity of users.

The main objective of this project is to create an intelligible and friendly web platform in order to help and give feedback to users who are starting in the world of programming, specially kids and teachers. In addition, Dr. Pencilcode tries to promote computational thinking development and motivate users to encourage their desire to improve as programmers by using scores and bonuses.

Some of the most used front-end technologies for the development of this application have been HTML and CSS combined with technologies such as JavaScript, Bootstrap and jQuery in order to add dynamism and improve the interactivity and usability of the web page. On the other hand, the server side has been developed mainly with the Django framework and other database management technologies like MySQL.

Índice general

1. Introducción	1
1.1. Marco General	1
1.2. Marco de Referencia	2
1.3. Estructura de la memoria	4
2. Objetivos	5
2.1. Objetivo general	5
2.2. Objetivos específicos	6
2.3. Planificación temporal	7
3. Estado del arte	9
3.1. Modelo Cliente-Servidor	9
3.2. HTTP	10
3.3. Frontend	11
3.3.1. HTML	12
3.3.2. CSS	13
3.3.3. JavaScript	14
3.3.4. jQuery	15
3.3.5. Bootstrap	16
3.4. Backend	17
3.4.1. Django	17
3.4.2. Python	18
3.4.3. Servidor HTTP Apache	20
3.4.4. Mysql	20

3.4.5. CoffeeLint	22
3.5. Pencil Code	23
4. Diseño e implementación	27
4.1. Arquitectura general	27
4.1.1. Django	28
4.1.2. Bases de datos	29
4.2. Página principal	30
4.3. Rúbrica de Dr. Pencilcode	33
4.3.1. FASE I	34
4.3.2. FASE II	34
4.3.3. FASE III	35
4.4. Dashboard	37
4.4.1. FASE I	37
4.4.2. FASE II	39
4.4.3. FASE III	40
4.5. Análisis de proyectos Pencil Code	44
4.5.1. Coffee-Mastery	44
4.5.2. Análisis por URL	46
4.6. Páginas de ayuda	49
4.7. Formulario	51
4.8. Certificado	55
4.9. Diferencias entre Dr. Pencilcode y Dr. Scratch	57
5. Resultados	59
6. Conclusiones	65
6.1. Consecución de objetivos	65
6.2. Aplicación de lo aprendido	66
6.3. Lecciones aprendidas	67
6.4. Trabajos futuros	69
6.5. Valoración personal	70

A. Información Extra	71
A.1. E-mails con Yana	71
A.2. Formulario Dr. Pencilcode	73
Bibliografía	77

Acrónimos y abreviaturas

AJAX Asynchronous JavaScript and XML

CSRF Cross-Site Request Forgery

CSS Cascading Style Sheets

DOM Document Object Mode

FICOD Foro Internacional de Contenidos Digitales

GSYC Grupo de Sistemas y Comunicaciones

HTML HyperText Markup Language

HTTP The Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

JSON JavaScript Object Notation

LaTeX Lamport TeX

LOMCE Ley Orgánica para la mejora de la calidad educativa

MIME Multipurpose Internet Mail Extensions

MIT Massachusetts Institute of Technology

MVC Model-View-Controller

PDF Portable Document Format

TCP/IP Transmission Control Protocol/Internet Protocol

URJC Universidad Rey Juan Carlos

URL Uniform Resource Locator

WebGL Web Graphics Library

XHTML Extensible Hypertext Markup Language

Índice de figuras

3.1. Arquitectura Cliente-Servidor	10
3.2. Modelo mensajes HTTP	11
3.3. Tecnologías utilizadas en el front-end	12
3.4. Documento básico de HTML	13
3.5. Ejemplo de aplicación de estilo CSS	14
3.6. Document Object Model	15
3.7. Patrón de diseño MVC de Django	18
3.8. Ranking de sistemas de gestión de base de datos	21
3.9. Análisis con CoffeeLint	22
3.10. Logo de Pencil Code	23
3.11. Programa creado con Pencil Code.	24
3.12. Categorías de Pencil Code	25
4.1. Carpetas y ficheros del proyecto.	28
4.2. Arquitectura general de Dr. Pencilcode	29
4.3. Base de datos Dr. Pencilcode	29
4.4. Diseño página principal	30
4.5. Ventana modal URL	31
4.6. Sección Why	31
4.7. Sección How	32
4.8. Sección New Features	32
4.9. Sección Contact	33
4.10. Dashboard Fase I	37
4.11. Publicación en Twitter	38

4.12. Dashboard Fase II	39
4.13. Sección Details Fase II	40
4.14. Tour Dashboard	41
4.15. Dashboard Fase III	42
4.16. Página Final Dashboard	43
4.17. Detalles de los resultados	43
4.18. Salida de coffee-mastery.py	45
4.19. Errores al analizar un proyecto.	46
4.20. Nuevo error de código	46
4.21. Relación entre funciones.	47
4.22. Análisis por URL	49
4.23. Página de ayuda Move	50
4.24. Página de ayuda Points	51
4.25. Botón Help us	52
4.26. Página de formulario	53
4.27. Mensajes del formulario	54
4.28. Algoritmo de la función getFeedback.	55
4.29. Certificado Dr. Pencilcode	56
4.30. Esquema certificado	57
4.31. Número total de modificaciones	58
5.1. Programa Ejemplo 1	59
5.2. Análisis Ejemplo 1	60
5.3. Sección Details Ejemplo 1	60
5.4. Nivel 3 de Control	61
5.5. Programa Ejemplo 2	62
5.6. Análisis Ejemplo 2	62
5.7. Sección Details Ejemplo 2	63
5.8. Certificado Ejemplo 2	63
6.1. Taller de superprogramadores en FICOD	69
A.1. E-mail David Bau	71

A.2. Primer e-mail de Yana	72
A.3. E-mail sobre las categorías de la Fase I	72
A.4. E-mail sobre las páginas de ayuda	73
A.5. E-mail sobre el formulario	73

Índice de tablas

4.1. Clasificación de niveles fase I	34
4.2. Porcentajes fase II	35
4.3. Clasificación de niveles fase III	36

Capítulo 1

Introducción

1.1. Marco General

Vivimos dentro de una sociedad en constante cambio y esto se debe, en gran medida, al desarrollo tecnológico que está teniendo lugar en los últimos años. A consecuencia de la evolución, el ser humano debe adaptarse a lo que ésta demanda para poder seguir progresando y ofrecer, así, una buena educación.

Con la introducción de las nuevas tecnologías y con el fin de mejorar la calidad educativa, la LOMCE, incorpora en las aulas la competencia digital. “La competencia digital es aquella que implica el uso de las tecnologías de la información y la comunicación para alcanzar objetivos relacionados con el trabajo, la empleabilidad, el aprendizaje y la participación en la sociedad. Igualmente precisa del desarrollo de diversas destrezas relacionadas con el acceso a la información, el procesamiento, la creación de contenidos y la resolución de problemas, tanto en contextos formales como informales”. [3]

Cada vez más, hay que concienciar a los niños de la importancia de la programación. Actualmente, se ha demostrado que con la programación se ayuda al desarrollo del pensamiento computacional, lo que contribuye al alcance de los objetivos fijados por la LOMCE pues el pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la informática. [11]

Así mismo, en un futuro no muy lejano, se solicitará tener fundamentos de programación para la mayoría de los puestos de trabajo, por ello, poco a poco, se está introduciendo la programación en las aulas. Sin embargo, al tratarse de algo totalmente nuevo, no todos los docentes disponen de los conocimientos básicos de la programación; muchos de ellos deben hacer un triple esfuerzo, aprendiendo a programar, enseñando a su vez a los alumnos y evaluando los programas realizados por el alumnado.

En cualquier ámbito, no solo en el educativo, es indispensable disponer de los recursos adecuados para poder realizar bien tu trabajo. Tanto los alumnos como los docentes necesitan la incorporación de nuevas herramientas capaces de desarrollar la competencia tecnológica y el pensamiento computacional y plataformas capaces de facilitar a los maestros todo el proceso que este cambio conlleva; es en este punto cuando surge Dr. Pencilcode.

1.2. Marco de Referencia

Para poder entender correctamente este proyecto, antes de introducirlo, hay que mencionar las dos herramientas en las que está basado: Scratch y Dr.Scratch.

Scratch es en un lenguaje de programación formado por bloques y orientado a la enseñanza. Éste permite al usuario crear videojuegos, historias y animaciones sin necesidad de tener conocimientos - ni siquiera básicos - sobre programación, centrándose únicamente en la lógica del programa.

Hace un par de años, en el Grupo de Sistemas y Comunicaciones (Gsyc) de la Universidad Rey Juan Carlos (URJC), brotó la iniciativa de crear una herramienta - **Dr. Scratch** - destinada a analizar proyectos Scratch y a aportar realimentación y soporte sobre los diferentes aspectos del Pensamiento Computacional para potenciar, de esta forma, el aprendizaje de la programación. [4]

Tras el gran éxito que esto supuso, mi tutor en este proyecto, Gregorio Robles, tuvo la gran idea de extender este prototipo de herramienta educativa a otros lenguajes de programa-

ción. El primer candidato para implementar dicha idea fue **Pencil Code**, lenguaje que sigue la línea de Scratch en cuanto a programación con bloques. Pencil Code soporta CoffeeScript como lenguaje principal y, además, ofrece la posibilidad de escribir tu propio código. Al tratarse de un lenguaje de programación de propósito general, el análisis de los proyectos Pencil Code será más complicado que el de los proyectos Scratch.

La decisión de desarrollar la aplicación para este lenguaje de programación surgió gracias a Gregorio Robles, tras ponerse en contacto con el creador de Pencil Code, David Bau, y proponerle la idea. Debido a su indisponibilidad, David Bau¹, tuvo la amabilidad de remitirnos a Yana Malysheva², responsable de Pencil Code en Boston durante el año académico 2016/2017, con quien hemos intercambiado diferentes puntos de vista acerca de la web y hemos mantenido contacto durante toda la realización del proyecto.

Dr. Pencilcode guarda el código de los programas en un JSON y analiza su contenido. La tarea de analizar toda la información del fichero JSON se atribuye al software *coffee-mastery.py*, plugin que tuvo que desarrollarse en paralelo junto con Dr. Pencilcode para poder realizar dicho análisis.

Por último, cabe destacar que la versión presentada en esta memoria no es más que la versión BETA de la aplicación; Dr. Pencilcode es un proyecto incipiente al que se le podría sacar mucho partido y que podría seguir creciendo exponencialmente con la inclusión de nuevas funcionalidades e ideas.

¹<http://davidbau.com/>

²<https://sites.google.com/site/yanamalportfolio/home/pencil-code>

1.3. Estructura de la memoria

Con el fin de facilitar la lectura de la memoria, se incluye esta sección donde se detalla la estructura de la misma y el contenido que se trata en cada uno de sus capítulos:

- **Introducción:** En este apartado, entre otras cosas, se enmarca y contextualiza el proyecto y se indica como surgió Dr. Pencilcode.
- **Objetivos:** Capítulo en el que se determinan los objetivos generales y específicos que fueron fijados durante el desarrollo del proyecto y, además, se explica la planificación temporal del mismo.
- **Estado del arte:** En esta sección se enumeran las tecnologías y la arquitectura utilizada en el desarrollo del proyecto realizando una breve descripción de cada una de ellas.
- **Diseño e implementación:** En este capítulo se explica como funciona la web; se habla de la arquitectura general de la herramienta y de su diseño e implementación. Así mismo, se indican las nuevas funcionalidades y modificaciones realizadas con respecto a Dr. Scratch.
- **Resultados:** Con el fin de aclarar lo que Dr. Pencilcode puede ofrecer y aportar al usuario, en este apartado, se realiza un análisis detallado de varios proyectos como ejemplo.
- **Conclusiones:** Aquí se habla sobre la consecución de los objetivos indicados con anterioridad en la memoria. Además, se comentan los conocimientos adquiridos durante el desarrollo de Dr. Pencilcode y qué funcionalidades podrían añadirse en el futuro para mejorar la plataforma. Finalmente, se incluye una valoración personal sobre el proyecto.

Capítulo 2

Objetivos

Uno de los aspectos más importantes a la hora de emprender un proyecto es la fijación de los objetivos. Para que un proyecto sea exitoso y cubra las necesidades deseadas hay que tener claro qué es lo que se quiere conseguir y cuales son nuestras metas. En este apartado se explican cuales han sido los objetivos iniciales de este proyecto y cual ha sido la planificación temporal del mismo.

2.1. Objetivo general

El objetivo principal de este proyecto es crear una aplicación web educativa capaz de incentivar a los alumnos y profesores a seguir mejorando sus habilidades como programadores a la par que desarrollan su pensamiento computacional.

Además, la finalidad de la plataforma es evaluar los programas creados con Pencil Code, mostrar los resultados obtenidos tras el análisis en un *dashboard* y aportar soporte y retroalimentación a los usuarios.

Con este proyecto se pretende transmitir ciertos valores como la perseverancia, el superarse a sí mismo, y el querer seguir aprendiendo. Es una forma más atractiva de aprender a programar y de transmitir que la programación puede llegar a ser muy divertida.

En definitiva, se pretende adaptar la página Dr. Scratch al entorno Pencil Code - teniendo

en cuenta que es un lenguaje de propósito general, mucho más complejo que Scratch - y hacer entender que se trata de una herramienta para aprender, no sólo evaluar.

2.2. Objetivos específicos

Como se ha mencionado anteriormente, Dr. Pencilcode sigue los pasos de Dr. Scratch. Al comienzo de este proyecto ya existía una versión avanzada de Dr.Scratch en la web; partiendo de esto, cabe destacar que los objetivos principales fueron:

1. **Adaptación de la implementación básica a Pencil Code.** El primer objetivo consistió en añadir y modificar código de Dr. Scratch para conseguir adaptar la página web a las necesidades de Pencil Code.
2. Para realizar esta adaptación fue necesaria la **creación de un Plugin** - Coffee-Mastery - capaz de analizar proyectos Pencil Code. Éste debería asignar una puntuación en función del grado de maestría de programación del usuario en las diferentes categorías disponibles: Move, Art, Text, Control, Sound y Operators.
3. Añadir **nuevas funcionalidades** y secciones a la página web - orientadas a Pencil Code - con el objetivo de seguir mejorando la plataforma.
4. **Aplicación en producción.** El siguiente paso, una vez conseguido el funcionamiento de la web, era tener la página en producción para que pudiera ser utilizada por cualquier usuario y estos pudieran dar retroalimentación y ayudar a mejorar la herramienta. Como Yana era la responsable de los talleres Pencil Code en Bostón, la idea principal era que sus alumnos fueran los que probaran primero nuestra aplicación.

Para lograr alcanzar estos objetivos fue necesaria la realización de un serie de pasos aún más específicos como:

- Diseñar una rúbrica para evaluar los proyectos creados con Pencil Code.
- Analizar proyectos Pencil Code mediante URL.
- Mejorar la página mostrada tras el análisis de un proyecto e incluir más dinamismo y elementos visuales con el objetivo de conseguir mayor inteligibilidad por parte del usuario.

- Añadir bonus en las puntuaciones - incluyendo estrellas en el dashboard - por complejidad, por diversidad, por bloques duplicados, por uso de todas las categorías.
- Incluir una sección donde se indiquen los bloques utilizados en cada una de las categorías de Pencil Code, así como una explicación de los bonus obtenidos.
- Incluir una sección donde dar consejos para que los usuarios puedan mejorar sus proyectos y puntuaciones.
- Crear un formulario para poder recibir retroalimentación - feedback - de los usuarios guardando sus respuestas en la base de datos.
- Eliminar la puntuación máxima con el fin de motivar a los programadores a seguir mejorando y evitar que piensen que hay un límite o que se conformen con la mayor puntuación.
- Diseñar seis páginas, una por cada categoría, incluyendo ejemplos de como mejorar el programa Pencil Code y como utilizar los bloques de la categoría seleccionada, dando soporte al usuario.
- Añadir tres nuevas páginas explicando los bloques que se pueden utilizar en cada categoría, la obtención de las puntuaciones y la adquisición de los bonus.
- Configurar un Servidor HTTP Apache donde poder subir la aplicación web.

2.3. Planificación temporal

Al igual que la mayoría de proyectos, el desarrollo de Pencil Code ha supuesto numerosas horas de trabajo. Se trata de un proyecto desarrollado únicamente por mí, junto con el soporte de mi tutor, Gregorio Robles, y la retroalimentación de Yana Malysheva. Este proyecto abarca una duración aproximada de dos años (cursos académicos) pudiéndo dividirse en dos fases principales.

FASE I o Fase de Aprendizaje: engloba el primer año del proyecto, desde noviembre de 2015 hasta mayo de 2016. Antes de comenzar a desarrollar cualquier proyecto, lo mejor es conocer su contexto, documentarse, ordenar las ideas y entender el propósito general del mismo.

Este periodo lo dediqué, exclusivamente, a comprender el funcionamiento de Dr. Scratch, lo que no fue tarea fácil pues una de las tecnologías pilares de Dr. Scratch es el framework web Django que, hasta la fecha, era un campo totalmente desconocido para mí. A raíz de esto, estuve una temporada documentándome e informándome sobre Django para poder entender el mecanismo de la web.

Así mismo, con el fin de involucrarme aún más, tuve la oportunidad de trabajar con MariLuz Aguado y Eva Hu, desarrolladoras de la herramienta Dr.Scratch, quienes me guiaron en mi comienzo con MySQL, Apache y Django. Junto a ellas, realicé varios talleres Scratch, los cuales me mostraron la importancia de introducir la programación a edades más tempranas y lo satisfactorio que resulta enseñar y ayudar en lo que a ti más te gusta. En esta etapa los objetivos que se fijaron fueron los siguientes:

- Comprender el código que conforma la web de Dr.Scratch.
- Aprender a utilizar nuevas tecnologías como Django, MySQL y Apache.
- Configuración de la máquina virtual para Dr. Pencilcode e instalación de Apache Server.
- Ponerme en situación contextualmente hablando.

FASE II o Fase de Desarrollo: este periodo abarca desde Octubre de 2016, como fecha de inicio, hasta Marzo de 2017, fecha en la que comencé a redactar esta memoria. Fue en esta fase donde conseguimos ponernos en contacto con Yana Malysheva con quien intercambiamos diferentes ideas y opiniones vía e-mail. A partir de aquí, comencé con el desarrollo completo de la web manteniéndola activa en el servidor. Los objetivos que se establecieron en esta fase fueron, prácticamente, todos los mencionados en el apartado de *Objetivos específicos*.

Capítulo 3

Estado del arte

En esta sección se realizará una breve explicación de las bases tecnológicas, arquitecturas y protocolos más significativos en las que está basado este proyecto.

3.1. Modelo Cliente-Servidor

El término cliente-servidor aparece en diferentes contextos dentro de la informática, uno de ellos es la arquitectura de red conocida como *Arquitectura Cliente-Servidor* [6] y utilizada en este proyecto. Se trata de un modelo para aplicaciones distribuidas - formadas por distintos elementos que se ejecutan en diferentes entornos - en el que las tareas se reparten entre clientes y servidores. Estos dos componentes interaccionan entre sí mediante peticiones y respuestas.

- **Cliente:** es la parte encargada de solicitar las peticiones al servidor, esperar y recibir sus respuestas. Mediante una interfaz gráfica de usuario, el cliente, interactúa de forma directa con los usuarios.
- **Servidor:** funciona como proveedor de servicios, compartiendo sus recursos con los clientes. Éste espera la llegada de las peticiones, las procesa y, posteriormente, envía una respuesta a los clientes.

El objetivo de la arquitectura cliente-servidor es proporcionar servicios que permitan compartir recursos a los usuarios de red.

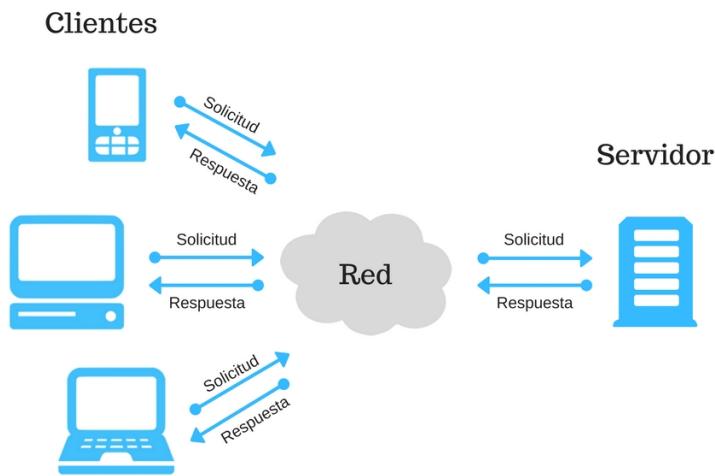


Figura 3.1: Arquitectura Cliente-Servidor

La arquitectura de software de una aplicación distribuida, o una aplicación cliente-servidor, se puede dividir en tres capas: capa de presentación (*front-end*), capa de lógica de aplicación y capa de servicio (*back-end*).

En los siguientes puntos de la memoria se explicarán los términos front-end y back-end y se hará hincapié en las tecnologías utilizadas en cada una de las capas.

3.2. HTTP

Hypertext Transfer Protocol, conocido por su abreviatura HTTP, es el protocolo de comunicación utilizado por los usuarios de la *Word Wide Web* para transferir documentos hypermedia y, por tanto, el protocolo usado en el modelo Cliente-Servidor para el intercambio de mensajes entre sus componentes.

Los mensajes HTTP son de **texto plano** y, generalmente, se estructuran en tres secciones [10]:

- **Línea inicial:** ésta es diferente para peticiones y respuestas. Su labor es la de describir el mensaje especificando el recurso que se solicita, indicando el método y la versión HTTP utilizada, etc.
- **Línea de cabecera:** Incluye información adicional -metadatos- sobre los mensajes de

petición y respuesta. Los datos sueles ser listas de atributos con forma *nombre:valor*. La cabecera acaba con una línea en blanco.

- **Cuerpo (opcional):** contiene cualquier tipo de datos. Los cuerpos de las peticiones llevan información al servidor, mientras que los de las respuestas llevan la información de vuelta al cliente. A diferencia de la línea inicial y la cabecera, el cuerpo puede incluir datos binarios (audio, imágenes, vídeo, etc) además de texto.

En las solicitudes HTTP se pueden incluir una serie de métodos ya predefinidos. Los métodos se encuentran en la línea de inicio e indican al servidor la acción que debe realizar sobre el recurso indicado. Entre los más utilizados en este proyecto se encuentran:

1. **GET:** solicita un objeto al servidor especificando su URL. Los datos son visibles por cualquier usuario.
2. **POST:** este método envía datos al servidor - normalmente los introducidos en un formulario, ocultos al usuario. Los datos van en el cuerpo del mensaje.

Los mensajes enviados por el servidor incluyen un código de respuesta. Éste informa al cliente sobre el estado de la solicitud, si ha tenido éxito o si se requiere alguna otra acción. Los más comunes son: **200 - OK** o **404 - Not Found** entre otros.

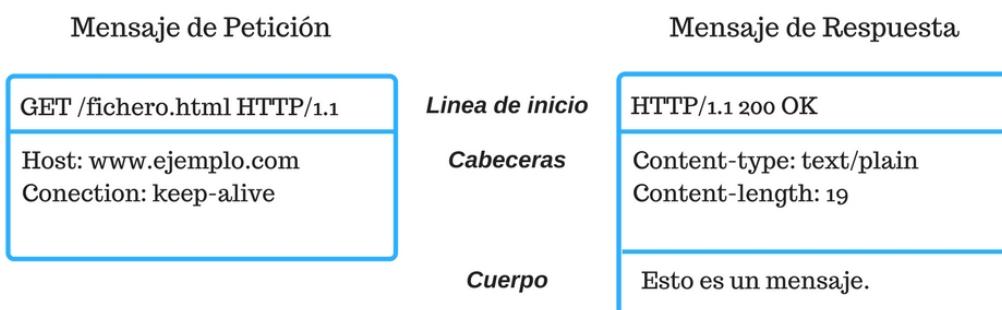


Figura 3.2: Modelo mensajes HTTP

3.3. Frontend

El front-end es la capa de presentación de cualquier página o aplicación web. Situada en el lado del cliente, ésta se centra en el diseño de la interfaz con el propósito de que el usuario sea capaz de interactuar con ella de una forma más sencilla, cómoda e intuitiva.

Las tecnologías más utilizadas para desarrollar el front-end de este proyecto han sido las enumeradas a continuación:



Figura 3.3: Tecnologías utilizadas en el front-end

3.3.1. HTML

HTML [9] - en inglés, HyperText Markup Language - es un lenguaje de etiquetas, también denominado lenguaje de marcado, con el que se pueden crear páginas y aplicaciones web. A través de las marcas o etiquetas, HTML permite dar estructura al texto dividiéndolo en secciones y párrafos, además de incorporar contenido multimedia en sitios web. Actualmente, se trata del estándar más utilizado por todos los navegadores.

En general, por cada elemento, es necesario añadir una etiqueta de apertura junto con su respectiva etiqueta de cierre, ésta última no siempre necesaria. El contenido mostrado en la página será el escrito entre dichas etiquetas. Las etiquetas de apertura pueden incluir atributos, que proporcionan información adicional sobre el contenido del elemento.

Las páginas HTML se suelen dividir en dos secciones principales: la cabecera y el cuerpo, ambas partes delimitadas por sus correspondientes etiquetas `<head></head>` y `<body></body>`. En la cabecera se incluye información relevante, no visible para el usuario, sobre la página web, como metadatos, enlaces a documentos externos o información acerca del estilo de la página,

mientras que en el cuerpo se incorpora todo el contenido que el usuario puede ver en pantalla como texto y multimedia.

A continuación se muestra cómo sería la estructura básica de un archivo HTML:

```
<!DOCTYPE html>
<html>
  <head>

    <title> Titulo de la página web </title>
    <meta name="author" content="URJC Libresoft FECyT">
    <link rel="stylesheet" href="static/app/content/style.css">

  </head>
  <body>

    <p> Información que queremos mostrar al usuario </p>

  </body>
</html>
```

Figura 3.4: Documento básico de HTML

3.3.2. CSS

Las Hojas de estilo en cascada [7] - cuyas siglas en inglés significan Cascading StyleSheets - describen como los elementos de un documento escrito en lenguaje de marcado, por ejemplo HTML, deberían ser mostrados o presentados en pantalla. Se utiliza principalmente para el diseño visual de la interfaz de usuario de páginas web, mejorando, así, su apariencia.

A pesar de que en los documentos HTML se puede definir el estilo de sus contenidos no es nada recomendable debido al costoso mantenimiento que esto acarrea. CSS tiene como objetivo marcar la separación del contenido del documento y la forma de presentación o aspecto de éste.

CSS consiste en un conjunto de reglas o normas. Para indicar qué norma de estilo debe seguir un elemento HTML, dicho elemento debe identificarse con un atributo de tipo id o class que también será indicado en la regla que se le quiera aplicar. Otra opción sería indicar la etiqueta deseada seguida de las reglas de estilo que se quieran incluir.

Algunas ventajas de utilizar CSS en nuestra web son:

- Compatibilidad con distintos dispositivos: mayor flexibilidad de adaptación a las diferentes plataformas, e.g, dispositivos móviles, impresoras, ordenadores, etc.
- Reducción de la complejidad del documento HTML, evitando la repetición de código fuente.
- Consistencia y ahorro de tiempo: Es aconsejable escribir el código CSS en un documento separado del código HTML. CSS ofrece la posibilidad de aplicar el mismo estilo, escrito en un documento externo .css, a diferentes archivos HTML y múltiples páginas web.

Independientemente de como se quieran aplicar las normas de estilo, es necesario incluir en el HTML la etiqueta <link> para hacer referencia al documento css que se va a utilizar.

```

<!DOCTYPE html>
<html>
  <head>
    <title> Aplicando un estilo </title>
    <link rel="stylesheet" href="estilo.css">
  </head>
  <body>
    <h1> Mi pagina web </h1>
    <p class='parrafo'> Dr Pencilcode </p>
  </body>
</html>
```

(a) html

```

h1 {
  font-family: Sans-serif;
  font-size: 16px;
  line-height: 28px;
  font-weight: 300;
  color: #313131;
  text-align: center;
}

.parrafo {
  font-family: Sans-serif;
  color: blue;
  font-size: 12px;
  font-style: italic;
  text-align: center;
}
```

(b) css

Figura 3.5: Ejemplo de aplicación de estilo CSS

3.3.3. JavaScript

JavaScript [8], conocido comúnmente como JS, es un lenguaje de programación interpretado orientado a objetos que se utiliza principalmente para añadir dinamismo a las páginas web. A pesar de su similitud con el nombre del lenguaje de programación Java, no debe confundirse con éste ya que ambos tienen finalidades muy diferentes.

Proporciona al usuario un mayor control sobre el navegador, además de añadir mejoras en la interfaz, dotando a la web de una mayor usabilidad, interactividad y navegabilidad.

Usado generalmente en el lado del cliente, JavaScript también es utilizado, junto con otras tecnologías o herramientas, para enviar y recibir información del servidor.

3.3.4. jQuery

jQuery es una biblioteca de JavaScript de código abierto y software libre que permite simplificar y facilitar el uso de este lenguaje en nuestra página web.

Entre las numerosas funcionalidades que jQuery ofrece, una de las más importantes es la manipulación del árbol DOM (Document Object Model). Mediante el uso de las funciones `jQuery()` o `$()` es posible modificar el contenido de la web sin tener que recargarla.

El DOM es una representación de todos los elementos que conforman una página web. Sigue una estructura en forma de árbol donde los elementos de XHTML, denominados nodos, están interconectados, especificando la relación que existe entre cada uno de ellos. jQuery simplifica la sintaxis para encontrar, seleccionar y manipular dichos elementos DOM.

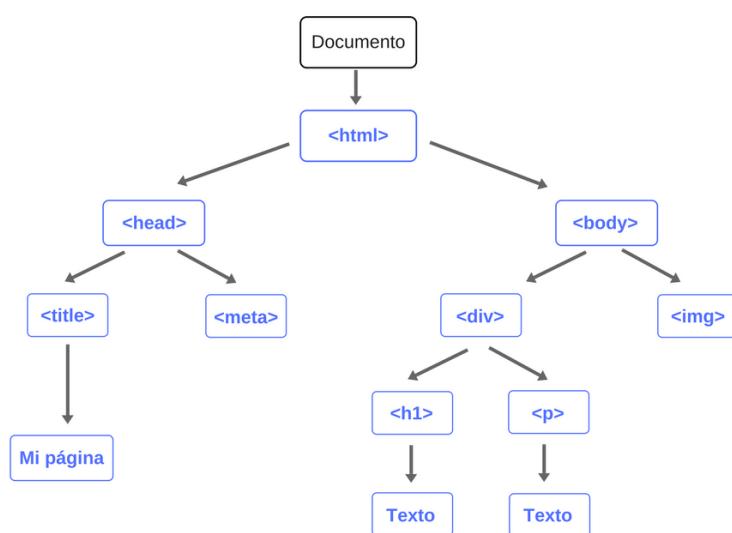


Figura 3.6: Document Object Model

Otras características de esta biblioteca multiplataforma son:

- Eventos HTML
- Manipulación de CSS.
- Animaciones y efectos.
- AJAX.

3.3.5. Bootstrap

Bootstrap¹ es uno de los frameworks más conocidos para la creación de páginas web. Desarrollado por Twitter, Bootstrap contiene plantillas basadas en HTML y CSS que facilitan la implementación del diseño web, así como extensiones adicionales de JavaScript. Su principal finalidad es convertir el desarrollo del front-end en una tarea más sencilla, rápida y eficaz.

Algunas de las características que hacen de Bootstrap una herramienta tan popular entre los desarrolladores web son las mencionadas a continuación:

- Se trata de una herramienta de código abierto, disponible en GitHub² de forma gratuita.
- Tiene compatibilidad con la mayoría de los navegadores web.
- Desde la versión 2.0, soporta diseño web responsive. Esta funcionalidad favorece la adaptación dinámica de la interfaz al tamaño del dispositivo (ya sean tablets, teléfonos móviles, pantallas de ordenador, etc) que se esté utilizando para acceder al sitio web.
- Incluye Grid system, útil para diseñar y maquetar por columnas. Este sistema permite añadir hasta un total de 12 columnas en la página.

¹<http://getbootstrap.com/>

²<https://github.com/twbs/bootstrap>

3.4. Backend

El back-end de una página web, también denominado capa de acceso a datos, se ejecuta en el lado del servidor. Éste es el encargado de analizar los datos de entrada del front-end y de realizar acciones tales como cálculos, procesamiento de la información, gestión de ficheros, interacciones de bases de datos, rendimiento, etc.

Las tecnologías elegidas en esta capa para desarrollo del proyecto han sido:

- **Python** como lenguaje de programación.
- **Django** como framework para la creación del sitio web.
- **MySQL** como base de datos.
- **CoffeLint** como analizador de código CoffeScript.
- **Servidor HTTP Apache**

3.4.1. Django

Escrito en el lenguaje de programación Python, **Django**, es un framework de software libre y código abierto enfocado al desarrollo web que tiene como objetivo ayudar al programador a crear aplicaciones web complejas de una manera más rápida y clara.

Sigue el patrón de arquitectura de software llamado **Modelo-Vista-Controlador** o MVC, éste divide la aplicación en tres partes interconectadas entre sí con el fin de separar la representación de información y la forma en la que la información es presentada y aceptada por el usuario (interacción con el usuario).

1. **Modelo:** es el componente principal del patrón. Gestiona el acceso y procesamiento de datos. Las peticiones para acceder a la base de datos se realizan a través del Controlador.
2. **Vista:** se encarga de presentar al usuario la información disponible en un formato adecuado (generalmente como interfaz de usuario, página HTML) para que éste pueda interactuar con ella. Dicha información se obtiene a través del Modelo.

3. **Controlador:** esta capa se encarga de gestionar y responder las solicitudes realizadas por el usuario apoyándose tanto en el Modelo como en la Vista.

En el caso de Django, el patrón de diseño sigue la estructura mostrada a continuación:

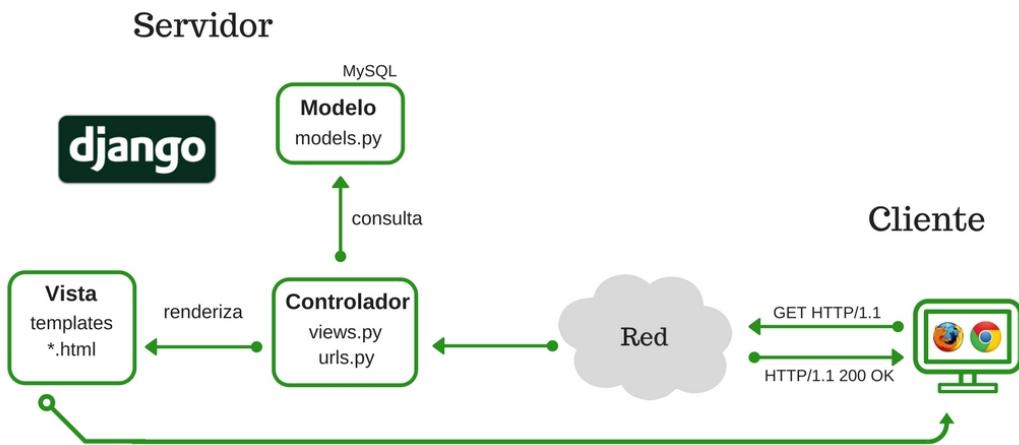


Figura 3.7: Patrón de diseño MVC de Django

Algunas otras características de Django son:

- **Seguro.** Django tiene muy en cuenta la seguridad y ayuda a los desarrolladores a evitar errores como cross-site scripting y cross-site request forgery (CSRF).
- **Rápido y eficaz.** Django fue diseñado para ayudar a los desarrolladores a crear sitios web lo más rápido posible.
- Sigue el principio de desarrollo de software **Don't Repeat Yourself** para evitar, así, la repetición de código.
- Soporta diferentes **bases de datos** entre las cuales están: PostgreSQL, MySQL, Oracle o SQLite.
- **Multiplataforma:** interopera en diversas plataformas informáticas.

3.4.2. Python

Desarrollado por Guido van Rossum a finales de los ochenta y con el objetivo de mejorar y ampliar las funcionalidades del *lenguaje de programación ABC*, **Python**, es un lenguaje de pro-

gramación de alto nivel, interpretado y orientado a objetos que se caracteriza por su concisión, legibilidad y transparencia. Debe su nombre a los humoristas británicos *Monty Python*.

Python ofrece un amplio abanico de posibilidades, desde la programación web tanto en el lado del cliente como del servidor (Django), hasta la programación con bases de datos. Es considerado como uno de los lenguajes más populares en lo que al mundo de Open Source se refiere.

Se conocen tres versiones, 1.X, 2.X y 3.X, la primera de ellas obsoleta, cuyos últimos lanzamientos han sido las versiones 1.6, 2.7 y 3.4 respectivamente. Desde la versión 2.1, Python posee una licencia compatible con *GNU General Public License*, denominada **Python Software Foundation License**, además, desde entonces, Python Software Foundation es la fundación sin ánimo de lucro responsable del código del lenguaje y de procesos como la administración de los derechos de autor y la obtención de fondos para la comunidad Python.

Las características más significativas de este lenguaje son:

- **Orientado a objetos:** todo son objetos.
- **Multiplataforma:** al igual que Django, se puede implementar en diferentes plataformas informáticas.
- **Open Source.**
- **Interpretado:** el código se ejecuta directamente mediante un interprete, siendo innecesaria la compilación previa del programa a código máquina.
- **Multiparadigma:** soporta el uso de dos o más paradigmas de programación. Python es orientado a objetos, imperativo, reflexivo y funcional.
- **Fuertemente tipado:** si los tipos no son exactamente iguales, no es posible pasar una variable como parámetro de procedimiento. Se debe realizar una conversión con anterioridad para que coincidan.
- **Librerías:** Hay cantidad de librerías disponibles para ampliar la funcionalidad de Python.
- **Case sensitive:** sensible a mayúsculas y minúsculas.

Python está ganando terreno. Fue situado en tercera posición por el IEEE en el *ranking* de los lenguajes de programación más populares del 2016³.

3.4.3. Servidor HTTP Apache

El Servidor HTTP Apache⁴ es el servidor web más utilizado del mundo. Fue desarrollado por una comunidad de usuarios bajo supervisión de la *Apache Software Foundation* dentro del proyecto HTTP Server (httpd). El objetivo es ofrecer un servidor fiable, eficiente y flexible que ofrezca servicios HTTP.

Como cualquier otro servidor, la función principal de Apache es almacenar, procesar y entregar páginas web, estáticas y dinámicas, a los clientes.

Apache soporta el protocolo HTTP/1.1 y su arquitectura es modular, se compone de un núcleo en el que se encuentra el código fuente y una serie de módulos que amplían la funcionalidad básica. Cualquier usuario podría escribir un módulo para realizar una función específica. Agregar y eliminar módulos es una gran ventaja ya que, con ello, se puede compilar un servidor Apache diseñado específicamente bajo las exigencias del usuario para el desarrollo de su sitio Web [5].

Una gran característica del servidor HTTP Apache es que es una tecnología de software libre y de código abierto, además de multiplataforma.

3.4.4. Mysql

En la actualidad, el uso de bases de datos es esencial para que los sitios web puedan recopilar información de una forma más eficiente, por ello, es necesario un sistema gestor que se encargue del almacenamiento, recuperación y administración de datos.

MySQL es un sistema de gestión de base de datos relacional desarrollado por *Oracle Corporation*. Se trata, junto con Oracle y Microsoft SQL Server, de uno de los sistemas de gestión de bases de datos más populares y más utilizados en el ámbito mundial, como se puede observar en la *Figura 3.8*⁵.

³<http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>

⁴<https://httpd.apache.org/es>

⁵<https://db-engines.com/en/ranking>

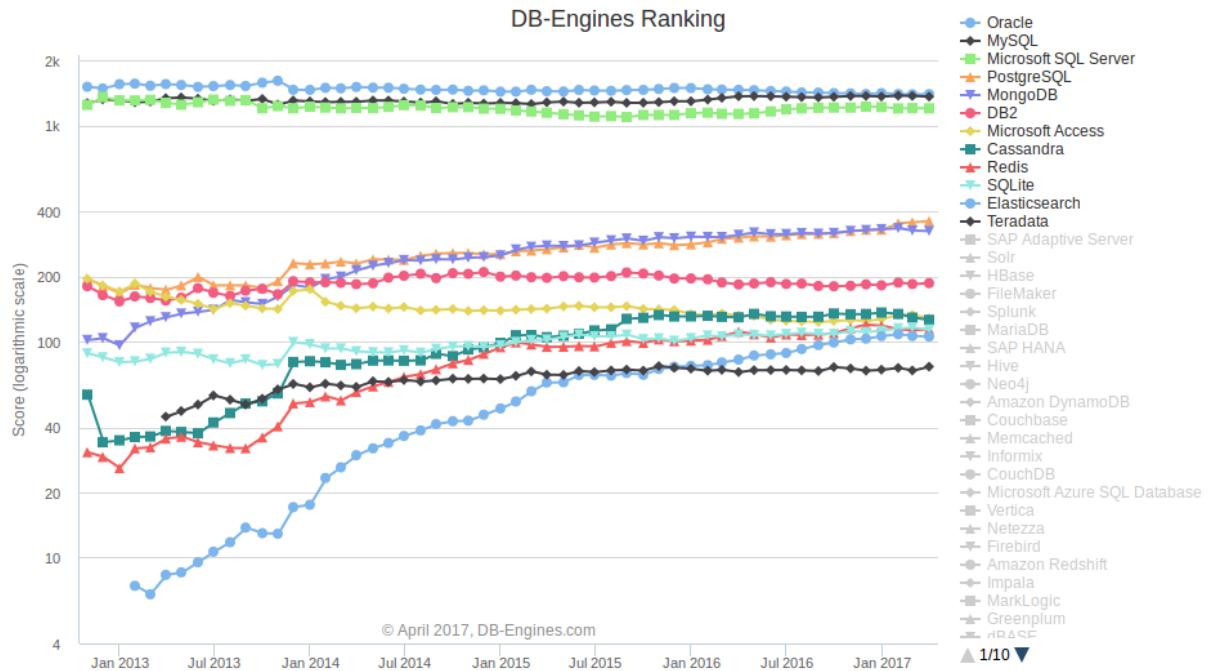


Figura 3.8: Ranking de sistemas de gestión de base de datos

Algunas características de MySQL son:

- Diseñado para ser **multihilos**, capaz ejecutar múltiples hilos de ejecución, usando hilos kernel.
- Soporta gran cantidad de datos, hasta 50 millones de registros.
- Es **rápido y fácil** de usar.
- Los clientes usan sockets TCP/IP para conectarse al servidor MySQL.
- Es **fiable**, ofrece fuerte protección de datos mediante sistemas de contraseñas y privilegios.
- Es **multiplataforma**, se puede implementar en distintas plataformas informáticas⁶.

⁶<https://www.mysql.com/support/supportedplatforms/database.html>

3.4.5. CoffeeLint

CoffeLint es un analizador estático de programas escritos en CoffeScript, se trata de un software de código abierto⁷ bajo la licencia MIT (Massachusetts Institute of Technology) que puede instalarse vía línea de comandos. Éste se encarga de comprobar que el programa sea consistente, limpio y esté correctamente escrito.

```

2 speed 100
3 rt 90 # ejemplo comentario
4 ht()
5 for color in [red, gold, green, blue]
6   jump -40, -160
7   for sides in [3...6]
8     pen path
9     for x in [1..sides]
10       fd 100 / sides
11       lt 360 / sides
12       fill color
13       fd 40

```

sara@vaio:~\$ coffeelint ejemplo.coffee
 ✓ ejemplo.coffee
 ✓ OK! » 0 errors and 0 warnings in 1 file
 sara@vaio:~\$

(a) Programa Coffeescript
(b) Análisis

Figura 3.9: Análisis con CoffeeLint

Por defecto, CoffeLint asegura que el programa esté escribiendo en CoffeScript idiomático, sin embargo, todas las reglas son opcionales y configurables, por lo que se pueden modificar para adaptarse a las necesidades o estilo de programación del usuario.

⁷<https://github.com/clutchski/coffeelint>

3.5. Pencil Code

Pencil Code⁸ es una aplicación web desarrollada, en su mayoría, por David Bau, miembro del equipo educativo de Google. Orientada en su totalidad a la enseñanza, Pencil Code trata de introducir a sus usuarios al mundo de la programación de una manera más fácil y sencilla, fomentando así la creatividad y el razonamiento.



Figura 3.10: Logo de Pencil Code

Se trata de un proyecto open source que soporta CoffeeScript como lenguaje principal además de otros lenguajes como Javascript, HTML y CSS. Fue diseñado con el objetivo de ser lo suficientemente flexible y simple para ser usado tanto por principiantes como por profesionales.

Con ayuda de un editor, Pencil Code permite dibujar, tocar música, crear juegos e historias así como experimentar con funciones matemáticas, geometría, algoritmos, simulaciones e incluso páginas web.

La pantalla de la web está dividida en dos partes, en la mitad izquierda se encuentra el código fuente mientras que en la derecha se muestra la salida del programa. Pencil Code ofrece la posibilidad de programar de dos formas diferentes, usando bloques o escribiendo código. Mientras que los programadores avanzados optan por escribir en los lenguajes estándar Javascript, CoffeeScript y HTML, los principiantes, pueden editar esos mismos lenguajes haciendo uso de bloques. Para poder programar a tu gusto, Pencil Code tiene habilitados dos botones, uno de ellos para seleccionar el lenguaje en el que quieras trabajar y el otro para pasar de la programación con bloques a texto y viceversa.

⁸<https://pencilcode.net/>

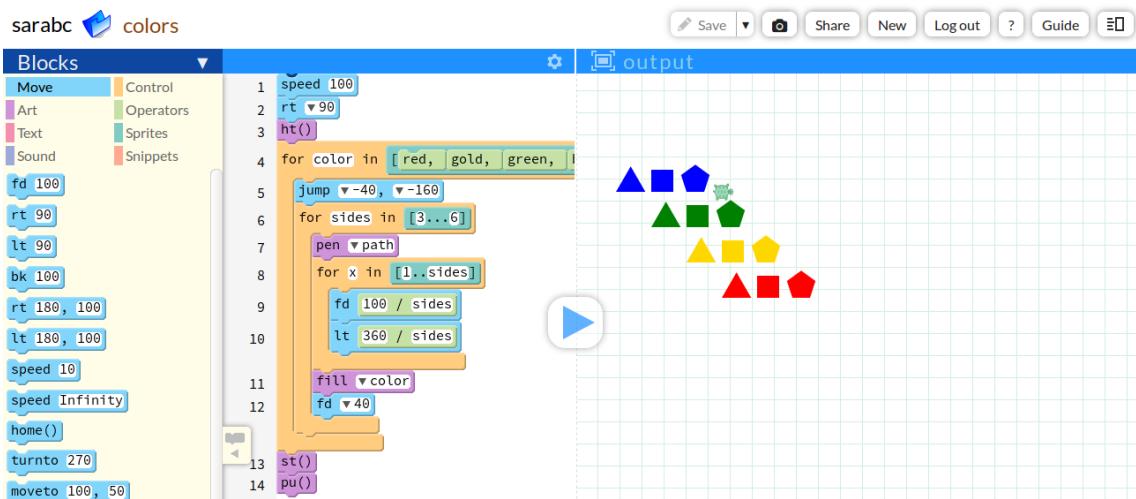


Figura 3.11: Programa creado con Pencil Code.

Pencil Code consta de ocho categorías - identificadas con diferentes colores - donde se encuentran los bloques que se pueden utilizar para crear nuestro programa. Arrastrando y uniendo las piezas - como si fuera un puzzle - se va dando forma y añadiendo instrucciones a éste. Entre las ocho categorías, mencionadas anteriormente, se encuentran: Move, Art, Text, Sound, Control, Operators, Stripes y Snippets.

1. **Move:** Encargada del movimiento, rotación, velocidad y visibilidad de la tortuga que aparece por defecto al empezar un programa.
2. **Art:** Con los bloques de esta categoría - encargada de la parte visual - se puede dibujar usando diferentes técnicas e incluso incluir imágenes en el programa que, posteriormente, serán mostradas por pantalla.
3. **Text:** Esta categoría añade texto a nuestro programa y campos de entrada donde el usuario puede introducir/escribir información.
4. **Sound:** Permite reproducir tonos y notas musicales además de archivos de audio.
5. **Control:** Las estructuras o sentencias de control se encargan de controlar el flujo del programa. Éstas modifican el orden de ejecución del programa, tomando decisiones o repitiendo un proceso varias veces. La categoría *Control* también contiene bloques que permiten añadir interactividad con el usuario.

6. **Operators:** En esta categoría se encuentran los bloques relacionados con aritmética, trigonometría, expresiones lógicas, funciones, etc.
7. **Stripes y Snippets:** Muestran fragmentos de bloques ya creados - como referencia - para que se puedan añadir al programa.

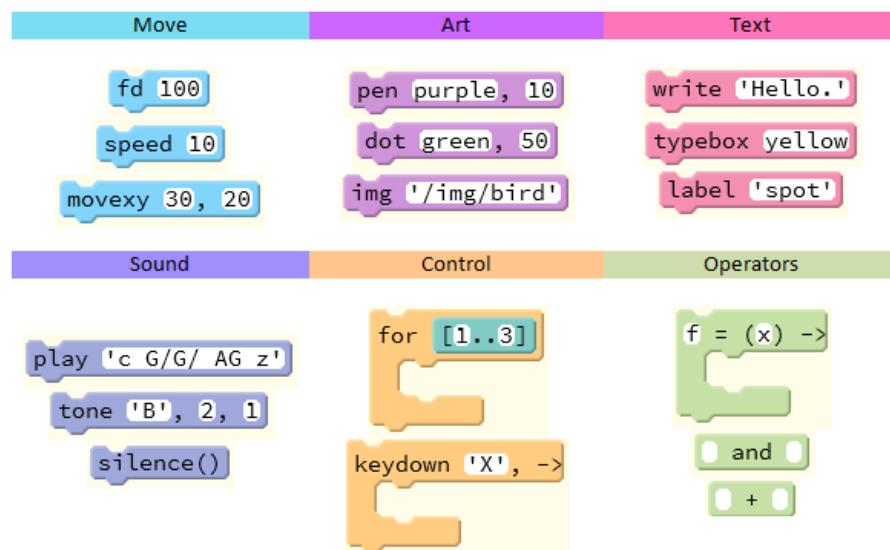


Figura 3.12: Categorías de Pencil Code

Cualquier programa creado con Pencil Code es público por lo que puede ser visto, compartido y copiado por cualquier usuario. De esta forma, es posible aprender a programar apoyándose en proyectos realizados por otros usuarios además de colaborar y aportar tus ideas a los mismos.

Capítulo 4

Diseño e implementación

Como ya se ha mencionado en varias ocasiones, Dr. Pencilcode se basó en Dr. Scratch. Para poder realizar este proyecto, primero tuve que entender como funcionaba la aplicación de Dr. Scratch y, para ello, también tuve que aprender a utilizar Django.

Dr. Pencilcode ha ido evolucionando con el paso del tiempo gracias a la colaboración de mi tutor, Gregorio Robles, y Yana Malysheva, responsable de Pencil Code en Boston.

En este capítulo se explicará la arquitectura general de la plataforma y las distintas fases por las que ha pasado el proyecto, centrándome únicamente en la última versión para detallar las funcionalidades de la aplicación.

4.1. Arquitectura general

Todo lo que se explica en los siguientes apartados se encuentra disponible en la web gracias a la máquina virtual que me proporcionó la universidad Rey Juan Carlos; en ella tuve que instalar Apache, lo que me llevó un tiempo, para poder gestionar la aplicación. Actualmente, la web se puede encontrar con el dominio <http://drpencilcode.libresoft.info/> o con la IP <http://193.147.79.206/>.

4.1.1. Django

Como ya se ha explicado en el capítulo 3 de la memoria, Dr. Pencilcode es una plataforma desarrollada con Django [1]. En la *Figura 4.1*, se puede observar el árbol de ficheros y directorios que componen la aplicación.

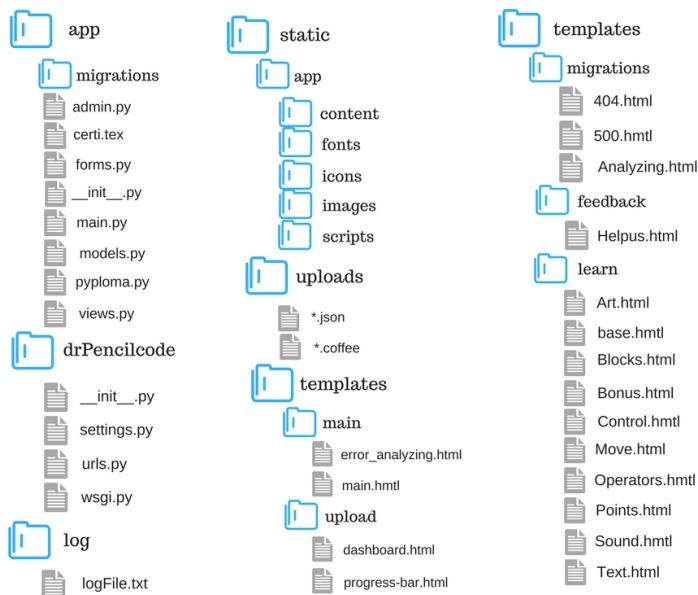


Figura 4.1: Carpetas y ficheros del proyecto.

Apache tiene como objetivo enviar las peticiones HTTP a Django. El fichero `urls.py` es el encargado de gestionar dichas peticiones y de comprobar si el recurso solicitado por el cliente es servido o no por nuestra aplicación. `urls.py` no es más que una lista en la que se especifican los recursos que son servidos por Dr. Pencilcode.

En el caso de que el recurso solicitado - la URL - no coincida con ninguno de los indicados en `urls.py`, se redirigirá al usuario a la página principal; en caso contrario, Django, pasará la petición a la función asociada de `views.py`.

Una vez encontrada la función asociada a dicha URL, ejecutado el código y procesado toda la información, dicha función deberá devolver una página de respuesta HTML - template - renderizada con los datos que se quieren mostrar en el HTML. La arquitectura general se muestra en la *Figura 4.2*.

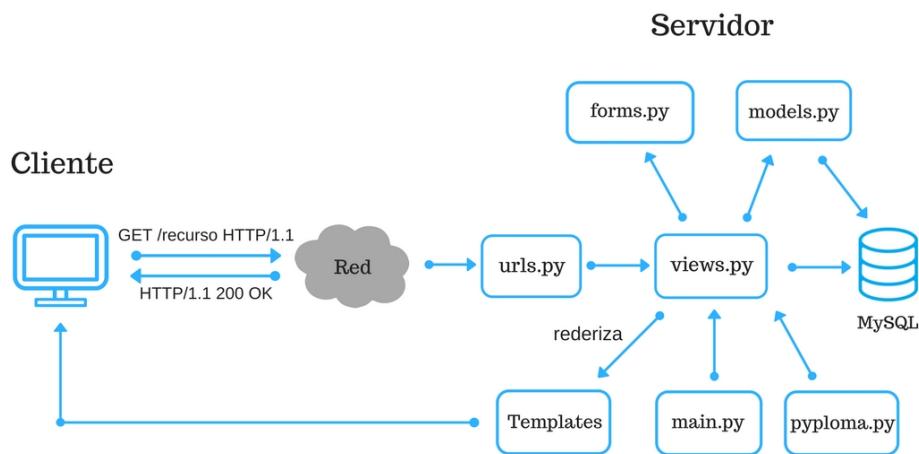


Figura 4.2: Arquitectura general de Dr. Pencilcode

4.1.2. Bases de datos

Para poder almacenar y gestionar toda la información, Dr. Pencilcode, dispone de una base de datos. SQLite es la base de datos que se usa en Django por defecto y es la que se utilizó en local durante el desarrollo del proyecto. Sin embargo, a la hora de subir la aplicación al servidor, decidí cambiar a MySQL - versión 5.5.53 - ya que es un sistema de gestión mucho más potente, pudiendo soportar mayor cantidad de datos.

Tables_in_drpencilcode
app_activity
app_dashboard
app_file
app_survey
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session

Figura 4.3: Base de datos Dr. Pencilcode

Como se puede observar en la *Figura 4.3*, existen algunas tablas cuyos nombres comienzan por app, auth y django; estas dos últimas, son tablas que se definen automáticamente por Django

y que se encargan de funcionalidades como las sesiones, las migraciones, la autenticación, etc. Las tablas que comienzan por app son las que realmente se crean en el fichero `models.py`.

4.2. Página principal

Lo primero que se observa nada más acceder al sitio web es el diseño mostrado en la *Figura 4.4*. En esta sección se hace una breve presentación de la herramienta y se muestra el cuadro en el que se introduce la URL para dar paso al análisis del proyecto.

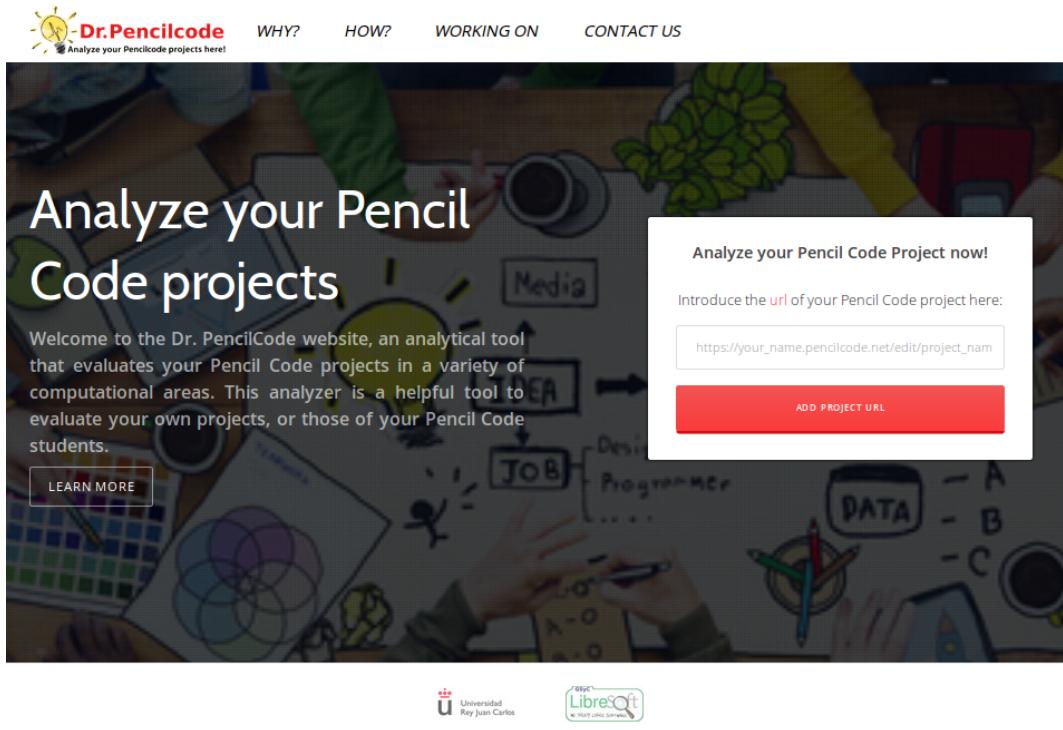


Figura 4.4: Diseño página principal

La mayoría de los niños podrían no saber lo que es una URL, para solucionar esta cuestión, se desarrolló una ventana modal en la que se incluía información extra de cómo conseguir la URL de los proyectos Pencil Code. La ventana modal se muestra en la *Figura 4.5* y ésta aparece cuando el usuario pincha sobre la palabra `url` que está marcada en rojo.

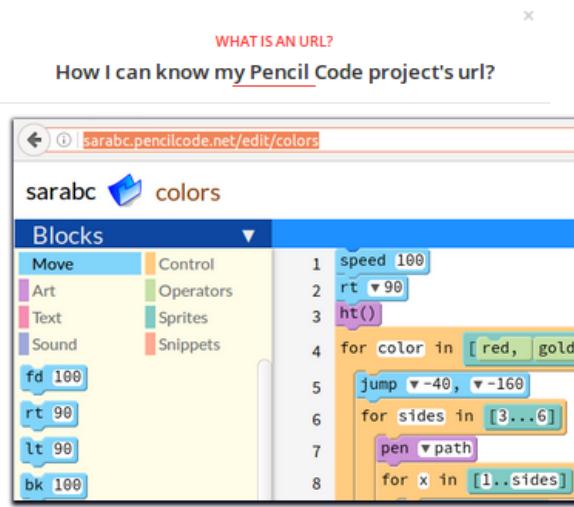


Figura 4.5: Ventana modal URL

A parte de esta visión, la página principal incluye 4 secciones más donde se habla de diferentes aspectos relacionados con Dr. Pencilcode:

1. **Why?**: explica los puntos fuertes de la aplicación y los motivos por los que su uso es tan positivo. Además, se hace hincapié en el continuo deseo de mejorar la plataforma teniendo en cuenta la opinión del usuario. Sección mostrada en la *Figura 4.6*.

The image shows the "Why" section of the Dr. Pencilcode website. The title is "Why you'll love Dr. Pencilcode" with a subtitle "A powerful tool to analyze your Pencil Code projects". There are three cards:

- Simple & Fun**: Your students can easily learn how to improve their programming skills in a fun way.
- Improve your skills**: We provide feedback on several aspects which are related to Computational Thinking.
- Always improving**: We're continually enhancing the features of Dr. Pencilcode, and we'd love to hear your thoughts.

Figura 4.6: Sección Why

2. **How?**: guía a los usuarios principiantes en su primer análisis indicándoles los pasos a seguir. El diseño de esta sección se puede ver en la *Figura 4.7*.

The screenshot shows a section titled "How it works" with a red "HOW?" header. Below the title is a sub-instruction: "In order to analyze a Pencil Code project, you just have to provide the project url and Dr. Pencilcode will do the rest." A text input field labeled "URL provided by Pencilcode" is shown. Below the input field is a descriptive sentence: "By analyzing your projects with Dr. Pencilcode, you can easily check your Computational Thinking Score." At the bottom of the section, there are four bullet points with icons:

- 💡 It is extremely easy to use, both for beginners and expert programmers.
- 💡 You receive feedback to improve your coding skills.
- 💡 You can keep on improving your programming skills by yourself even from home.
- 💡 You can track your progress through charts and stats.

Figura 4.7: Sección How

3. **Coming soon:** en esta sección se explican las nuevas funcionalidades que se quieren incluir a la aplicación en un futuro cercano, como puede ser la traducción de la página a distintos idiomas, la introducción de cuentas de usuarios o la creación de un plug-in en Firefox y Chrome para tener Dr. Pencilcode siempre a la vista. Por el momento, ninguna de ellas ha sido implementada, aunque espero continuar con este proyecto y realizarlas próximamente. El aspecto de esta sección se muestra en la *Figura 4.8*.

The screenshot shows a section titled "Coming Soon" with a red "NEW FEATURES" header. Below the title is a sub-instruction: "We are working on new features to improve Dr. Pencilcode". The section contains five items, each with an icon and a brief description:

- 💡 Multilingual Website: We know the importance of receiving feedback on your own language to understand every little detail or hint, so we'll translate Dr. Pencilcode to facilitate coding learning to as many people as possible.
- 💻 Teacher Accounts: Teachers will be able to group (and follow) their students to keep track of their progress in a fast and simple way.
- ☁️ Cloud Platform: We are in the process of migrating our servers to the Cloud with the aim of providing a better performance.
- ✖️ Plug-ins: With Dr. Pencilcode plug-ins for Chrome and Firefox, students will analyze their project WHILE programming on the Pencil Code website.
- ⭐️ Gamification: We'd like to take learners curiosity one step further. Do you have cool ideas to gamify our platform?
- 💬 Social Network: The new social network functionalities will allow users to communicate with each other to exchange ideas and challenges.

Figura 4.8: Sección New Features

4. **Contact:** muestra todas las vías (Twitter, email y GitHub) por las que el usuario puede ponerse en contacto con el equipo de Dr. Pencilcode. Esto nos permitirá conocer diferentes puntos de vista y opiniones acerca de la aplicación y si está cumpliendo, o no, su propósito. Su diseño se puede ver en la *Figura 4.9*.

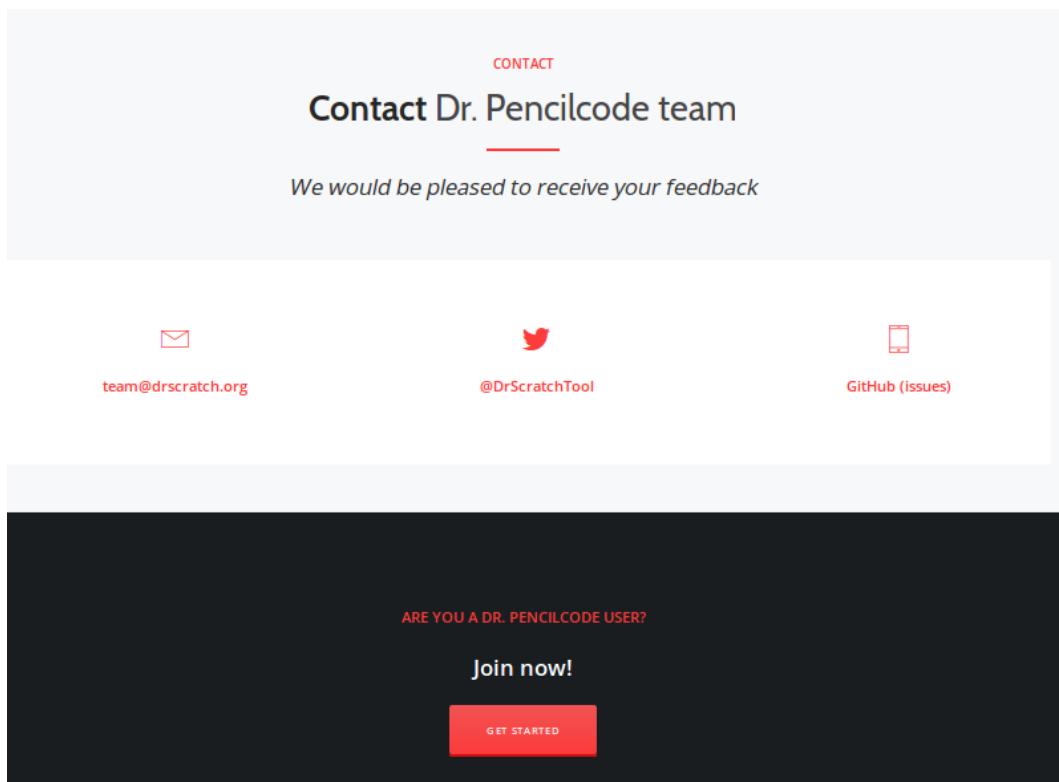


Figura 4.9: Sección Contact

En la actualidad, el equipo de Dr. Pencilcode no dispone de cuentas de e-mail y Twitter propias por lo que se remite a los usuarios a las cuentas de Dr. Scratch.

4.3. Rúbrica de Dr. Pencilcode

Durante el diseño de la página web fue necesario detenerme a pensar cómo quería que se analizaran los proyectos de Pencil Code. Quizá, ésta fue una de las partes que más tiempo me llevó ya que al tratarse de un lenguaje de programación de propósito general y por tanto, un lenguaje más complejo, no tenía claro como podía enfocar dicha evaluación.

La rúbrica inicial es muy diferente a la que está disponible actualmente en la web. A lo largo de mi año de trabajo, se han ido realizando diferentes modificaciones hasta llegar al método de evaluación que estábamos buscando. Se pueden distinguir tres fases principales en el desarrollo de la rúbrica:

4.3.1. FASE I

Al comenzar el proyecto se decidió realizar el análisis de la misma forma que en Dr. Scratch. El objetivo era evaluar los programas calificando aspectos del pensamiento computacional - *pensamiento lógico, control de flujo, abstracción, paralelismo, interactividad con el usuario, representación de la información y sincronización* - en función de los bloques utilizados. En cada uno de los 7 aspectos se podía obtener un total de 3 puntos, lo que llevaba una puntuación máxima de 21 puntos. Dependiendo de la puntuación obtenida, el usuario podía tener un nivel *basic, developing o proficiency*, tal y como se muestra en la *Tabla 4.1*

Nivel	Puntos
Basic	Puntos < 7
Developing	7 < Puntos < 15
Proficiency	15 < Puntos < 21

Tabla 4.1: Clasificación de niveles fase I

Tras varios correos con Yana y a medida que se probaba y se desarrollaba la rúbrica, nos dimos cuenta de que algunos de los aspectos de Pensamiento Computacional, como la sincronización y el paralelismo, no eran aplicables a Pencil Code ya que eran dimensiones más específicas de Scratch. A partir de ese momento, se decidió rediseñar el método de evaluación dando lugar a las Fase II.

4.3.2. FASE II

Visto que lo comentado en el apartado anterior era inviable, decidimos evaluar los proyectos centrándonos en las dimensiones más significativas de Pencil Code: *Move, Text, Sound, Art, Control y Operators*.

En esta fase, se quitaron las puntuaciones y los niveles, las categorías se evaluaban en función del número de bloques utilizados en cada una de ellas y, además, se cambiaron las fracciones por porcentajes.

Para medir el porcentaje de bloques usados, se contó el número de bloques visibles que aparecían por defecto en las categorías de Pencil Code - denominado *maxblocks*- y, a continuación, se aplicaron las reglas mostradas en la *Tabla 4.2*. El número de bloques en cada categoría era: Maxblocks = {’Move’: 12, ’Art’: 12, ’Text’: 8, ’Sound’: 6, ’Control’: 15, ’Operators’: 18}

Bloques utilizados	Condición
100 %	$3/4 \cdot \text{maxblocks} < \text{nº bloques} < \text{maxblocks}$
75 %	$1/2 \cdot \text{maxblocks} < \text{nº bloques} < 3/4 \cdot \text{maxblocks}$
50 %	$1/4 \cdot \text{maxblocks} < \text{nº bloques} < 1/2 \cdot \text{maxblocks}$
25 %	$0 < \text{nº bloques} < 1/4 \cdot \text{maxblocks}$
0 %	$\text{nº bloques} = 0$

Tabla 4.2: Porcentajes fase II

Una vez más, tras probar la rúbrica, decidimos volver a cambiar la forma de evaluar principalmente por dos motivos:

1. Debido a la capacidad de Pencil Code de cambiar el modo de programación de bloques a código, el número total de bloques por categoría no era posible indicarlo con determinación ya que el usuario podía escribir tanto código como quisiera, incluyendo bloques no visibles en la interfaz de cada categoría.
2. Este método, realmente, no evaluaba los proyectos, si no que mostraba el porcentaje de bloques que se habían utilizado. Desde mi punto de vista, esta rúbrica no impulsaba tanto el deseo de mejorar y aprender a programar como lo podía hacer la rúbrica de la fase I (con niveles y puntuaciones).

4.3.3. FASE III

Para tener en cuenta todas las posibilidades, se buscó información sobre todos los bloques que se podían utilizar en cada categoría, incluyendo tanto los que aparecen por defecto en Pen-

cil Code como los que no se muestran en la interfaz. Se introdujeron de nuevo los niveles y las puntuaciones y se pasó de porcentajes a fracciones.

La rúbrica final de Dr. Pencilcode y la que se encuentra disponible actualmente en la web es la que se explica a continuación:

- En cada categoría se puede obtener un máximo de 3 puntos, dependiendo del tipo de bloque utilizado en dichas categorías, se podrán obtener 1, 2 o 3 puntos.
- Sumando los puntos obtenidos en cada categoría se puede llegar a obtener un total de 18 puntos.
- La puntuación máxima se consigue usando bloques de nivel 3 en todas las categorías y obteniendo todos los bonus.
- En función de los puntos obtenidos, el nivel de programación del usuario puede ser: Básico, intermedio, avanzado o experto, mostrado en la *Tabla 4.3*.

Nivel	Puntos
Básico	Puntos < 6
Intermedio	6 < Puntos < 12
Avanzado	12 < Puntos < 18
Experto	Puntos > 18

Tabla 4.3: Clasificación de niveles fase III

Finalmente, para motivar aún más a los niños, se decidió añadir Bonus. Por el momento hay 4 tipos de bonus disponibles:

1. **Bonus por diversidad:** se obtiene cuando el usuario utiliza bloques pertenecientes a los 3 niveles de una misma categoría.
2. **Bonus por complejidad:** se consigue cuando el usuario utiliza más bloques de nivel 3 que bloques de niveles inferiores en la misma categoría.

3. **Bonus por todas las categorías:** Como ya sabemos, Pencil Code está formado por 6 categorías principales, este bonus se obtiene si el programa analizado contiene bloques de las 6 categorías.
4. **Bonus por repetición de bloques:** se obtiene cuando se utiliza el mismo bloque más de una vez.

4.4. Dashboard

A medida que se iba modificando la rúbrica de Dr. Pencilcode, también lo iba haciendo el diseño de la página de análisis. En esta sección se explicará y se mostrará la evolución - modificaciones y nuevas funcionalidades - del dashboard durante las fases mencionadas anteriormente.

4.4.1. FASE I

Al tratarse de la primera rúbrica, la apariencia del dashboard era muy similar al de Dr.Scratch, véase *Figura 4.10*. Esta versión de la página constaba de cuatro secciones principales:

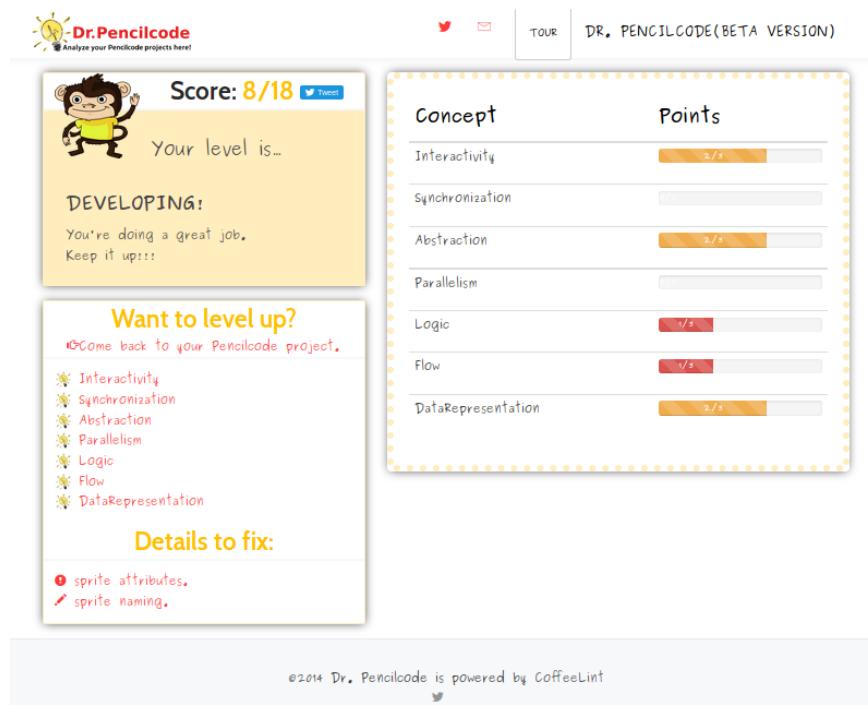


Figura 4.10: Dashboard Fase I

1. **Cabecera superior:** esta sección incluía el logo de Dr. Pencilcode, presentaba iconos con enlaces a algunas redes sociales de nuestro equipo como e-mail y Twitter y, además, contenía un botón de ayuda - *Tour*¹ - encargado de explicar cada una de las secciones del dashboard.
2. **Puntos y nivel:** mostraba la puntuación total adquirida sobre 18 junto con el nivel obtenido (basic, developing o proficiency). En esta sección también se ofrecía la posibilidad de publicar la puntuación en Twitter, tal y como se muestra en la *Figura 4.11*.

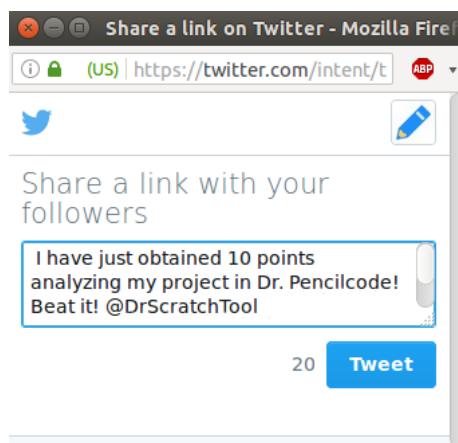


Figura 4.11: Publicación en Twitter

3. **Tabla de puntuaciones:** aquí se mostraban las puntuaciones parciales de cada uno de los aspectos del pensamiento computacional. Como se puede observar en la imagen, las habilidades de *Sincronización* y *Paralelismo* aparecían vacías debido a que no estaban definidas para Pencil Code. Además, incluía un enlace para poder volver al proyecto Pencil Code y seguir con la programación.
4. **Want to level up?:** ofrecía enlaces a páginas de ayuda, donde los usuarios podían encontrar ejemplos para subir su puntuación final. En ese momento las páginas de ayuda todavía no estaban adaptadas a Pencil Code.

¹<http://bootstraptour.com/>

4.4.2. FASE II

El cambio de rúbrica pasando del análisis de los aspectos del pensamiento computacional al análisis de las principales categorías de Pencil Code - Move, Art, Sound, Text, Control, Operators - conllevó la realización de varias modificaciones en el dashboard, como se puede observar en la *Figura 4.12*.

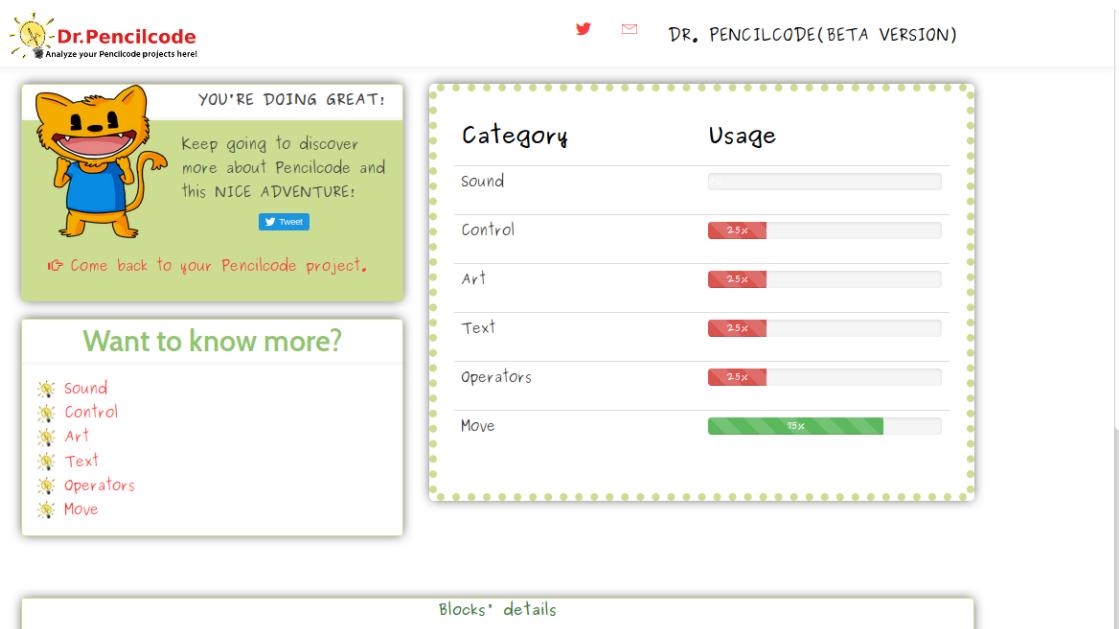


Figura 4.12: Dashboard Fase II

En cuanto a las 4 secciones principales:

- 1. Cabecera superior:** se mantuvo el mismo aspecto que en la fase I, exceptuando la omisión del botón *Tour*.
- 2. Puntos y nivel:** Se eliminaron las puntuaciones y los niveles sustituyéndose por un mensaje que motivara a los usuarios a mantener su curiosidad y seguir programando. El botón de Twitter y el enlace al proyecto de Pencil Code se trasladaron a esta sección.
- 3. Tabla de puntuaciones:** Se modificó por completo la tabla cambiando los nombres de los aspectos de pensamiento computacional al de las categorías de Pencil Code y cambiando las barras de progreso de fracciones a porcentajes, pudiendo obtenerse un 25 %, 50 %, 75 % o 100 % en cada categoría.

4. **Want to know more?**: al desaparecer las puntuaciones se cambió la sección *want to level up?* a *want to know more?* donde se añadieron enlaces a las páginas de ayuda de las distintas categorías.

Además, para que los usuarios pudieran obtener más información acerca de su programa, se añadió una **nueva sección** *Blocks Details* en la que se mostraban las categorías y los bloques utilizados en cada una de ellas. En el caso de que en alguna de las categorías no se hubiera utilizado ningún bloque se indicaba con un **Not used** asociado a dicha categoría. Para poder ver esta sección los usuarios tenían que hacer scroll en la pantalla; su diseño se puede observar en la *Figura 4.13*.

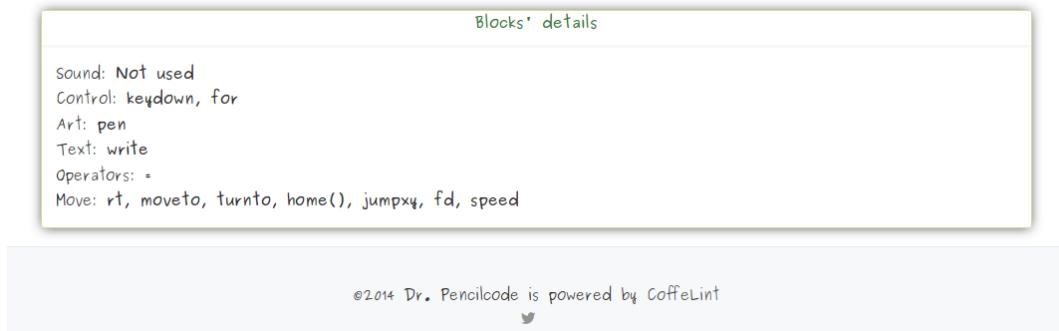


Figura 4.13: Sección Details Fase II

4.4.3. FASE III

Con la nueva y última modificación del método de evaluación se volvió a cambiar el diseño del dashboard. Además de los cambios en la secciones, se añadieron nuevas implementaciones y un mayor número de elementos visuales para dotar a la web de mayor inteligibilidad, véase *Figura 4.15*.

En cuanto a las secciones previamente definidas:

1. **Cabecera superior**: Se habilitó de nuevo el botón *Help* añadiendo 3 pasos adicionales al Tour, uno por cada nueva sección: *Details: Used Blocks*, *Download Certificate* y *Help us!*. El aspecto del tour es el que se muestra en *Figura 4.14*.

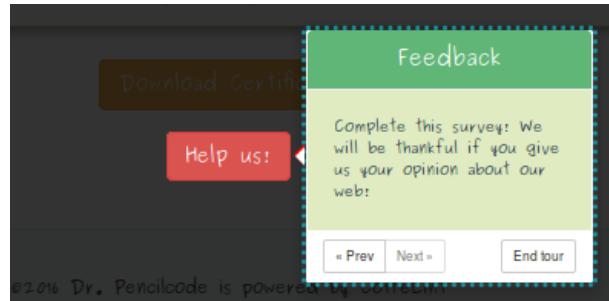


Figura 4.14: Tour Dashboard

2. **Puntos y nivel:** Se volvieron a incluir puntuación y niveles (*basic, intermediate, master* y *expert*), pero no puntuación máxima.
3. **Tabla de puntuaciones:** Con la introducción de los Bonus, se añadió una nueva fila a la tabla de puntuaciones en la que estos quedaban representados con estrellas; se rellenan tantas estrellas como bonus se consigan. Además, el hecho de tener una sección dedicada únicamente a los enlaces de las páginas de ayuda quitaba demasiado espacio, por ello, se decidió habilitar los enlaces en la tabla de puntuaciones y aprovechar la sección, ahora llamada, *What can you improve?* para introducir consejos sobre como mejorar el programa.
4. **What can you improve?:** se introdujeron mensajes donde se aconsejaba al usuario en cual de las categorías debía trabajar más para subir su nivel, así como sugerencias para completar todos los bonus. Estos mensajes se centran en las categorías en las que se ha conseguido el nivel más bajo.
5. **Details: Used Blocks:** para conocer los bonus conseguidos de una forma más específica, se añadió a la sección *Details* un desplegable *Bonus* donde se explicaba porqué motivo se habían obtenido. También se añadieron elementos visuales.

A esta altura del proyecto, surgió la idea de crear un formulario para recopilar diferentes opiniones sobre la web y añadir un diploma que pudiera descargarse. Por ello, se decidió habilitar dos nuevos botones, uno de ellos para redirigir al formulario y otro para descargar el certificado. En este momento, el aspecto de la página era el mostrado en la *Figura 4.15*.

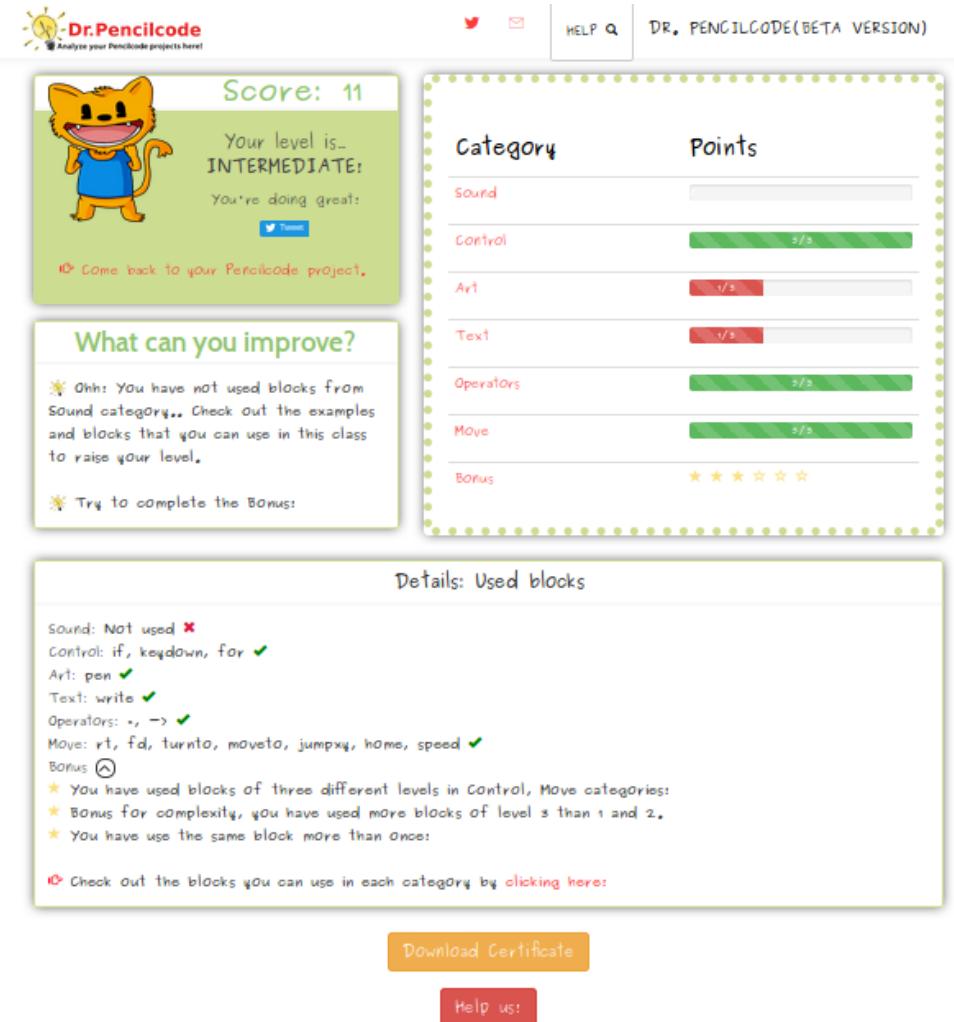


Figura 4.15: Dashboard Fase III

Finalmente, se llevó a cabo una última modificación que dio lugar al diseño actual de la plataforma, disponible en la web <http://drpencilcode.libresoft.info/>.

Por lo general, los niños no suelen hacer scroll en la pantalla, esto podía desencadenar que la sección de *Details* y los nuevos botones añadidos en la parte inferior fueran ignorados por el usuario, por ello, se decidió agrupar las categorías de tal forma que todas fueran visibles al cargar la página `dashboard.html`.

Al tratarse de información adicional, la sección *Details*, se hizo desplegable; así, si el usuario quiere conocer más sobre su proyecto, lo único que tiene que hacer es pinchar en la flecha de despliegue. La última versión del dashboard es la que se muestran en las *Figuras 4.16 y 4.17*.

The screenshot shows the final dashboard of the Dr. Pencilcode project. At the top, there's a logo with a pencil and a lightbulb, followed by the text "Dr. Pencilcode" and "Analyze your Pencilcode projects here!". To the right are links for "HELP" and "DR. PENCILCODE(BETA VERSION)". Below this, a large green box displays a cartoon cat character, a score of 10, and the message "Your level is... INTERMEDIATE! You're doing great!" with a "Tweet" button. A call-to-action "Come back to your Pencilcode project." is also present. To the right is a chart titled "Category Points" with the following data:

Category	Points
Sound	0/5
Control	2/5
Art	2/5
Text	0/5
Operators	0/3
Move	3/3
Bonus	★★★☆

Below the chart is a "Details" section with a dropdown arrow. Underneath are buttons for "Download Certificate" (orange) and "Help us!" (red). A "Details" section at the bottom contains a list of categories and their usage status:

- Sound: Not used ✗
- Control: keydown, for ✓
- Art: pen, os ✓
- Text: Not used ✗
- Operators: -, → ✓
- Move: rt, fd ✓
- Bonus

A note states: ★ You have used the same block more than once! and a link: Check out the blocks you can use in each category by clicking here!

Figura 4.16: Página Final Dashboard

This screenshot shows the "Details" section of the dashboard, which provides a breakdown of the usage of various Scratch blocks. The categories and their status are as follows:

- Sound: Not used ✗
- Control: keydown, for ✓
- Art: pen, os ✓
- Text: Not used ✗
- Operators: -, → ✓
- Move: rt, fd ✓
- Bonus

A note at the bottom says: ★ You have used the same block more than once! and a link: Check out the blocks you can use in each category by clicking here!

Figura 4.17: Detalles de los resultados

4.5. Análisis de proyectos Pencil Code

A pesar de la gran evolución que ha tenido el proyecto, a partir de ahora, me centraré únicamente en la Fase III ya que es la que está implementada actualmente y es así como funciona realmente la aplicación.

Al igual que Dr.Scratch realiza el análisis apoyándose en plugins de Hairball, era necesario crear un nuevo plugin, `coffee-mastery.py`, para llevar a cabo la implementación de la rúbrica de Dr.Pencilcode. En esta sección se explica cómo se analizan los proyectos, desde que se introduce la URL hasta que se muestran los resultados en el dashboard.

El análisis de los proyectos sigue una estructura similar a la de Dr. Scratch, sin embargo, se han tenido que hacer modificaciones en las funciones `selector`, `processStringUrl`, `sendRequestgetJSON`, `handler_upload` y `analyzeProject` que son las principales encargadas de realizar todo el proceso de análisis.

4.5.1. Coffee-Mastery

Como se ha comentado, con el objetivo de implementar la rúbrica de Dr. Pencilcode se desarrolló `coffee-mastery.py`; en él se definen seis funciones - una por categoría - donde se encuentran los conjuntos² (básico, intermedio y avanzado) en los que se determinan a qué niveles pertenecen los bloques de cada categoría. Coffee-mastery es el que realmente se encarga de analizar los proyectos de Pencil Code; su finalidad es devolver información del proyecto analizado.

Para analizar un proyecto con Coffee-mastery se debe introducir el comando `python coffee-mastery.py colors.coffee`, especificando como parámetro el nombre del archivo CoffeeScript. La salida del programa es la que se muestra en la *Figura 4.21*.

²NOTA: Un conjunto es una lista desordenada y sin elementos repetidos. Los conjuntos soportan operaciones matemáticas como la unión, la intersección y la diferencia. La función `set()` puede utilizarse para crear conjuntos [2].

```

Move: {'1': ['rt', 'lt', 'fd'], '3': ['jump'], '2': ['speed']}
Art: {'1': ['pen', 'fill'], '3': 'false', '2': ['ht', 'st']}
Text: false
Sound: false
Control: {'1': ['in'], '3': 'false', '2': ['for']}
Operators: {'1': ['-', '/'], '3': 'false', '2': 'false'}
Duplicados: {'for': 3, 'color': 2, '-': 2, '/': 2, 'fd': 2, 'in': 3, 'sides': 3}
Levels: {'Sound': 0, 'Control': 2, 'Art': 2, 'Text': 0, 'Operators': 1, 'Move': 3}

```

Figura 4.18: Salida de coffee-mastery.py

Como se puede observar, coffee-mastery ofrece información sobre los bloques que se han utilizado en cada una de las categorías, indicándose también el nivel al que pertenece cada bloque; los bloques duplicados y el número de veces que se ha repetido cada uno de ellos y, por último, el número total de puntos que se ha obtenido en cada categoría.

El proceso que sigue Coffee-mastery hasta conseguir toda esta información es el siguiente:

1. Lo primero que hace el plugin es leer el contenido del fichero .coffee
2. A continuación, elimina los comentarios del programa, así como los caracteres numéricos, los paréntesis y todo texto que vaya entre comillas.
3. Una vez que el código está “limpio”, se crea una lista de tokens - tokenList - con lo que queda del código del programa.
4. Se llama a la función *duplicates()*, pasando como parámetro tokenList, que devuelve un diccionario con los bloques repetidos.
5. Se llama seis veces a la función *check()*, una vez por cada categoría. Ésta se encarga de realizar la intersección entre el set() del código del programa - tokenSet - y los Sets() básico, intermedio y avanzado de esa categoría. Si en alguna categoría no se encuentran elementos comunes con el código del programa, la función le asignará un *False*; en caso contrario, se devolverá un diccionario con los niveles y los bloques utilizados en el nivel correspondiente.
6. Por último, se llama a la función *niveles()* y ésta, a su vez, a la función *checking()*. *Checking()* comprueba mediante intersección de Sets qué niveles se han utilizado en el programa y devuelve el nivel más alto. *Niveles()* recoge esta información y crea un diccionario con las categorías y sus puntuaciones.

4.5.2. Análisis por URL

A la hora de introducir la URL en Dr.Scratch, si ésta no es correcta, pueden aparecer 3 tipos de errores: si la URL está mal escrita, si la URL/proyecto no existe en los servidores de Scratch o si en la URL no se ha introducido ningún valor.

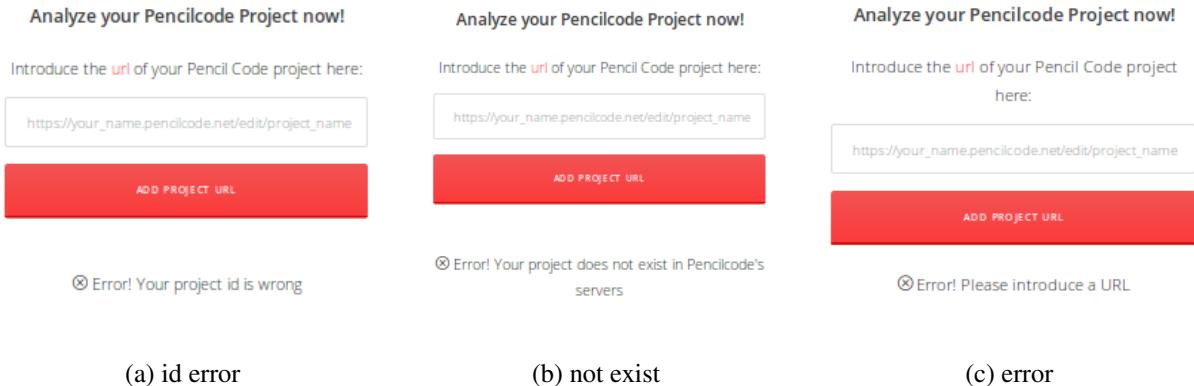


Figura 4.19: Errores al analizar un proyecto.

En Dr.Pencilcode, además de esos tres, se incluye un nuevo error, **code_error**, el cual se invoca cuando el código del programa Pencil Code no está correctamente escrito. Para poder lanzar este error se hace uso de la herramienta CoffeeLint que, como se ha explicado en el capítulo anterior, es un analizador estático de código CoffeeScript.

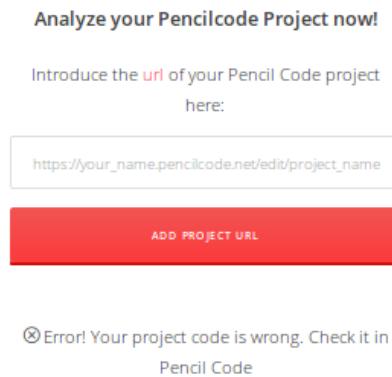


Figura 4.20: Nuevo error de código

A continuación explicaré, a grandes rangos, los pasos que se siguen para el análisis de los proyectos y, posteriormente, mostraré el diagrama de flujo para tener una idea más clara de las funciones implicadas y su propósito.

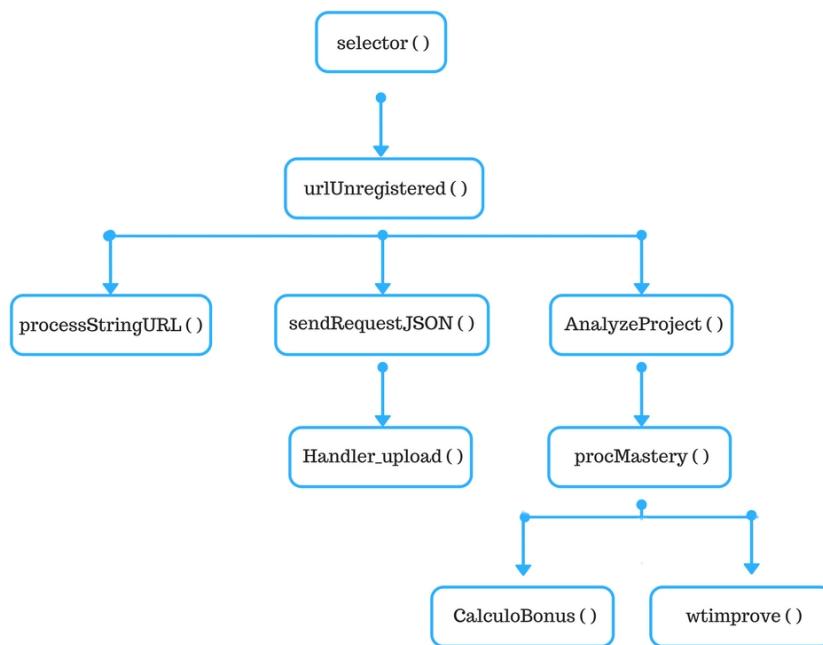


Figura 4.21: Relación entre funciones.

1. Una vez introducida la URL, se llama a la función `selector()`; ésta se encarga de devolver la página `dashboard.html` (renderizada con un diccionario que contiene toda la información del análisis de ese proyecto) o la página `main.html` (con alguno de los 4 errores comentados anteriormente) en función de si ha habido o no problemas a la hora de ejecutar el resto de funciones.
2. `urlUnregistered()` comprueba si el formulario es válido y llama a las funciones `ProcessStringUrl()`, `sendRequestgetJSON()` y `analyzeProject()`. Al final, devuelve un diccionario a `selector()` con información del proyecto o información de error (`code_error`, `no_exists` y `error analyzing`) dependiendo de si las funciones a las que llama se ejecutan correctamente o no.
3. `processStringUrl()` comprueba si la URL introducida por el usuario está correctamente escrita; si es así, devolverá la URL con el siguiente formato: `http://sarabc.pencilcode.net/load/first`, en caso contrario, devolverá un error de ID (`id_error`).

4. *sendRequestgetJSON()* Guarda información del proyecto en la base de datos y en log-File.txt. Además, realiza la petición a Pencil Code, abre la URL <http://sarabc.pencilcode.net/load/first>, lee los datos que contiene (en este caso el código del programa analizado) y escribe todo en un fichero .json.
5. *handler_upload()* se encarga de comprobar si el fichero *.json existe ya en la carpeta de */uploads*. En caso de que exista, el documento recibirá un nombre con formato ***(num).json** donde num es el número de veces que el documento original está repetido.
6. *AnalyzeProject()* es la encargada de analizar los proyectos haciendo uso del plugin coffee-mastery y la función *procMastery()*. Comprueba si el código del programa está bien escrito usando CoffeeLint. Finalmente, devuelve el resultado del análisis en un diccionario, que es el que llega a *selector()*.
7. *ProcMastery()* se encarga de montar el diccionario y guardar todo en la base de datos; suma los puntos, comprueba los bonus llamando a *CalculoBonus()* y comprueba los mensajes de ayuda de la sección *What can you improve?* llamando a *wtimprove()*.

En la *Figura 4.22* se puede observar el diagrama de flujo del análisis de los proyectos por URL.

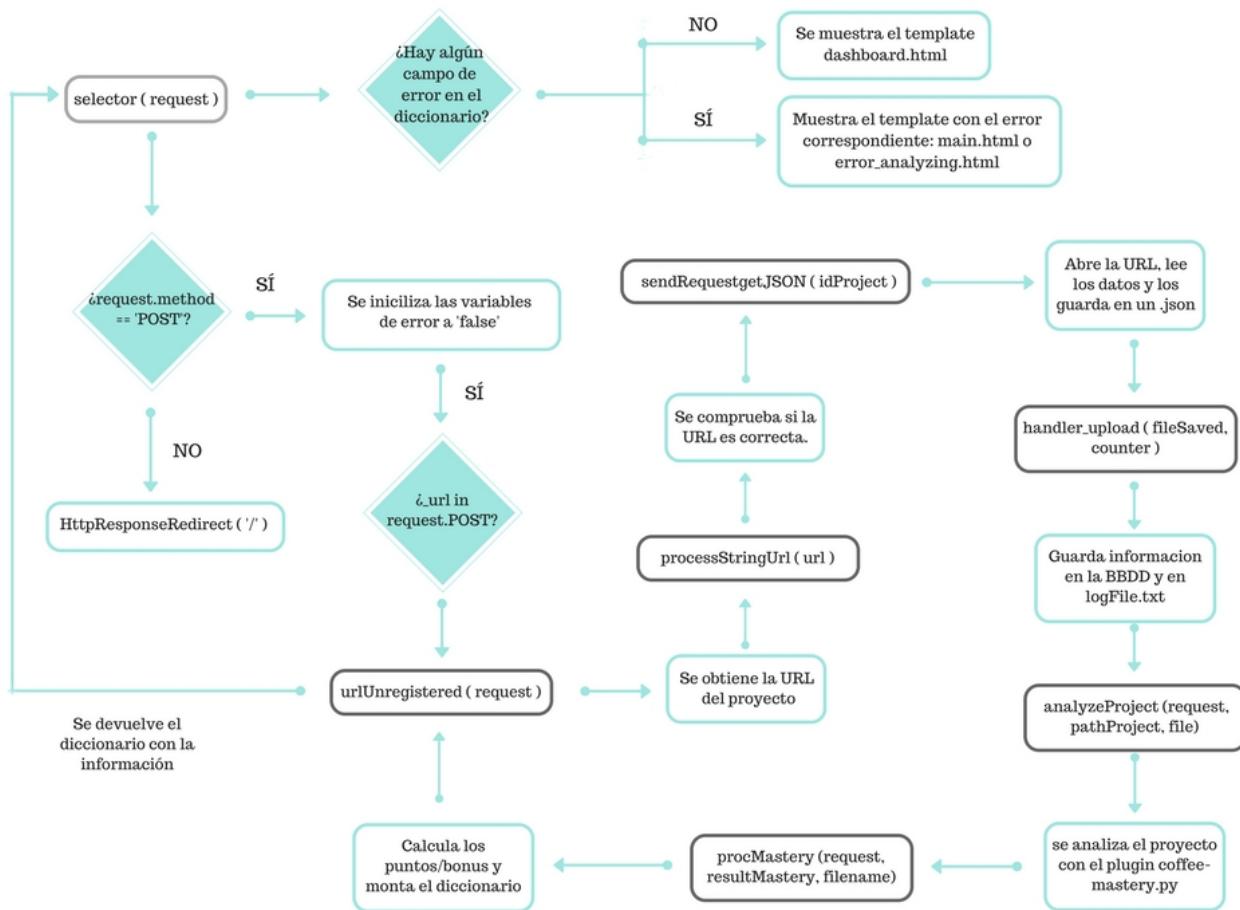


Figura 4.22: Análisis por URL

4.6. Páginas de ayuda

Al tratarse de una herramienta educativa, en Dr. Pencilcode, se incluyen nueve nuevas páginas de ayuda que dan soporte a los usuarios. Estas páginas son extensiones del documento `base.html` que es el que contiene el diseño básico de todo el conjunto. A través de `base.html` o `dashboard.html`, podemos acceder a cada una de las páginas de ayuda vía enlace.

Seis de ellas (`Move.html`, `Art.html`, `Text.html`, `Sound.html`, `Control.html`, `Operators.html`) contienen información sobre las diferentes categorías de Pencil Code. Estas páginas son muy útiles cuando el usuario quiere subir su puntuación; en ellas se incluyen varios ejemplos, con las respectivas explicaciones del programa, de cómo utilizar los bloques de cada nivel. Además, al final de cada página, se añaden enlaces a páginas externas donde hay

gran cantidad de programas con los que pueden aprender.

Para acceder a las páginas de ayuda se debe pinchar en los enlaces (con el nombre de las categorías) que aparecen tras analizar un proyecto. El diseño de estas seis páginas es similar y es el que se observa en la *Figura 4.23*

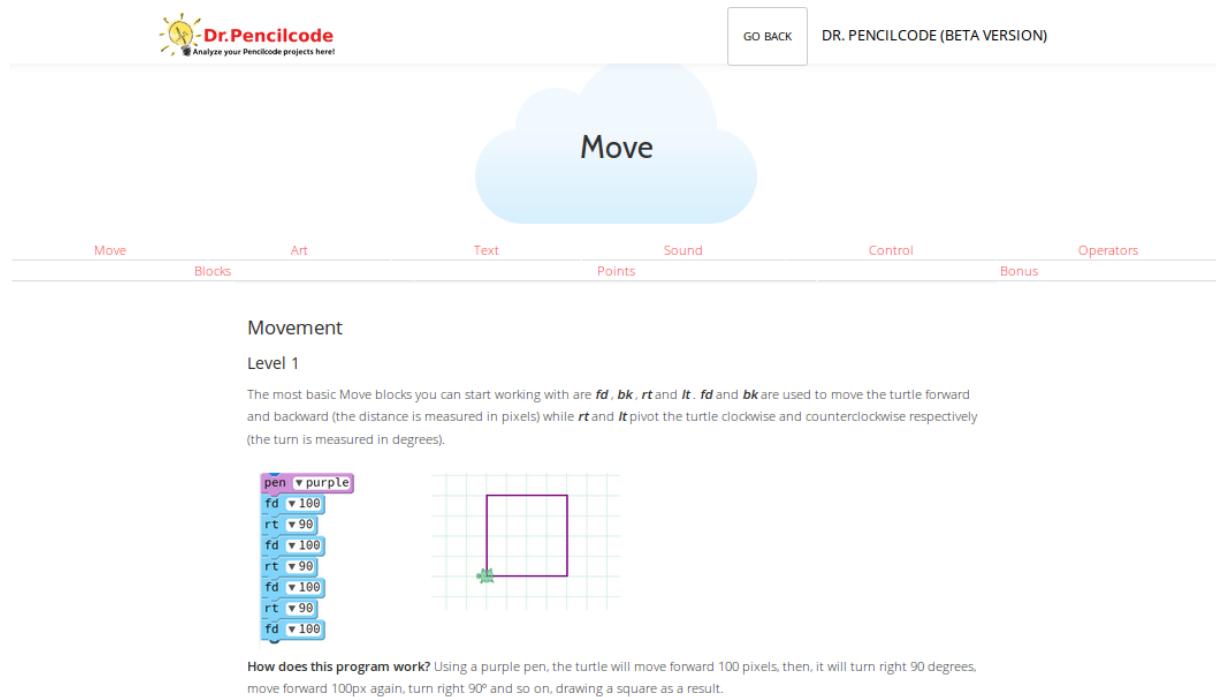


Figura 4.23: Página de ayuda Move

Por lo general, cuando una persona está siendo evaluada, ésta prefiere conocer los métodos de evaluación (rúbrica) que están siendo utilizados. Las tres páginas restantes (`Points.html`, `Bonus.html` y `Blocks.html`) están destinadas a mostrar los bloques existentes por categoría y a la explicación de cómo Dr.Pencilcode evalúa o analiza los proyectos, es decir, determina las puntuaciones y los bonus que se asignan a cada proyecto.

En la *Figura 4.24* se muestra, como ejemplo, el diseño de la página `Points.html`:

The screenshot shows the 'Dr. Pencilcode' interface with a 'Points' section. It includes a logo, navigation links ('GO BACK', 'DR. PENCILCODE (BETA VERSION)'), and tabs for 'Blocks', 'Points', and 'Bonus'. A message about the tool's purpose is displayed. Below is a section titled 'How does Dr. Pencilcode evaluate?' which explains point categories and provides three examples of progress bars:

- 1 point out of 3. (Red bar, 1/3)
- 2 point out of 3. (Orange bar, 2/3)
- 3 point out of 3. (Green bar, 3/3)

Text below the bars states: "Adding the points obtained in each category you can get a total score of **18 points** (6 categories x 3 points). You will get the maximum score when you use blocks from level 3 in all the categories and you complete all the bonuses." It also mentions levels: "According to the points you get, **your level** as a programmer can be:" followed by a list of levels.

Figura 4.24: Página de ayuda Points

Durante el diseño de esta sección se mantuvo intercambio de correos con Yana para explicarle la idea del desarrollo de las páginas de ayuda; propuesta que le pareció muy acertada.

4.7. Formulario

Con la finalidad de recibir retroalimentación y tener en cuenta las opiniones de todos los usuarios - tanto alumnos como docentes - para seguir mejorando la herramienta y las funcionalidades de ésta, se decidió realizar un formulario.

En primer lugar, el formulario se realizó en papel. Éste se centraba en conocer los niveles de usabilidad, interactividad e inteligibilidad de la herramienta desde el punto de vista del usuario. La encuesta incluía preguntas relacionadas con la apariencia de la web, el deseo del alumno de seguir mejorando su proyecto una vez analizado y conocida su puntuación, la capacidad para moverse por las diferentes páginas... concluyendo con un apartado de libre opinión. La idea

principal era que cada usuario que utilizara Dr. Pencilcode llenara el formulario y nos lo enviara por correo.

Una vez diseñada la encuesta, nos pusimos en contacto con Yana para conocer su opinión acerca de esta nueva idea. Como se puede leer en los correos del *Apéndice A.1*, había preocupación en las palabras y expresiones utilizadas en la primera versión del formulario. Al tratarse de una encuesta enfocada especialmente a niños, Yana sugirió modificar las preguntas y adaptar las palabras con el fin de que fueran comprensibles para los más pequeños.

Así mismo, para no utilizar papel y hacer todo más fácil y rápido, decidimos añadir una nueva funcionalidad implementando el formulario en la página web.

Para acceder al formulario, simplemente, se tiene que pinchar en el botón **Help us** habilitado en la página de análisis del proyecto o *Dashboard*. Al pinchar, se abre una nueva pestaña en el navegador dando paso a la página del formulario **Helpus.html**.

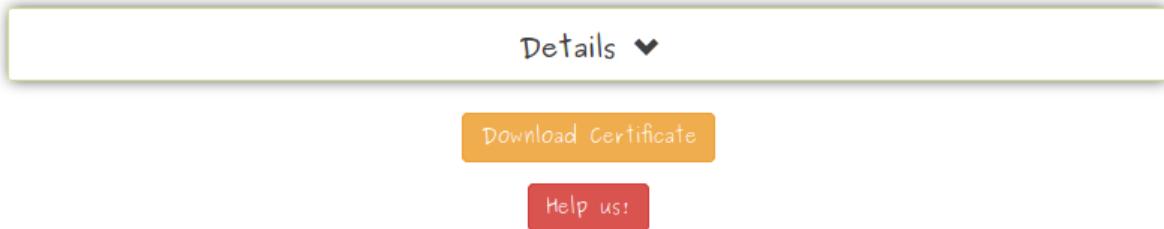


Figura 4.25: Botón Help us

El motivo de abrir una nueva pestaña, es el de facilitar a los usuarios el poder contestar a las preguntas a la vez que analizan sus proyectos, investigan y navegan por la aplicación web. El diseño de la página del formulario es el mostrado en la *Figura 4.28*.

Please, answer and complete all the questions!

Dr.Pencilcode workshop - survey

Name:

Surname:

Task 1. When visiting Dr.Pencilcode:

a. What do you think about the webpage? Do you like it?

Yes
 More or less
 No

b. Do you understand what the web is for?

Yes
 More or less

Figura 4.26: Página de formulario

Para desarrollar esta funcionalidad se crea una nueva función en views.py llamada *getFeedback()*. Una vez que el usuario pulsa el botón **Submit** se lleva a cabo una petición HTTP POST con el recurso /helpus. El documento urls.py es el encargado de enlazar con la función *getFeedback()* de views.py.

getFeedback inicializa dos variables booleanas - *form_filled* y *data_exist* - a False, dependiendo de los valores que vayan tomando estas variables a lo largo de la función, se mostrará por pantalla uno de los siguientes mensajes: success, warning o danger (warning por defecto).

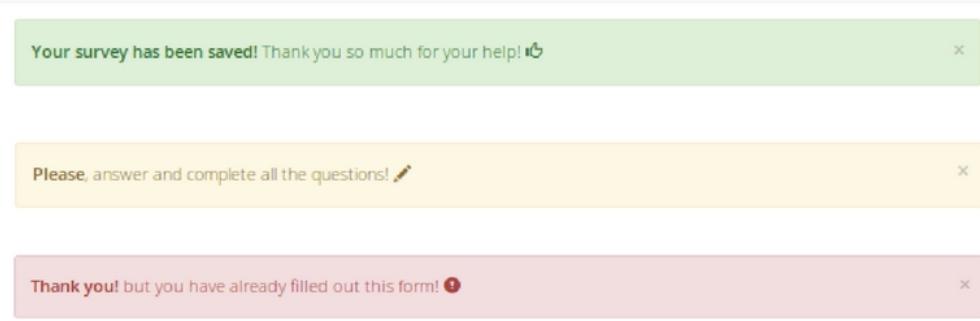
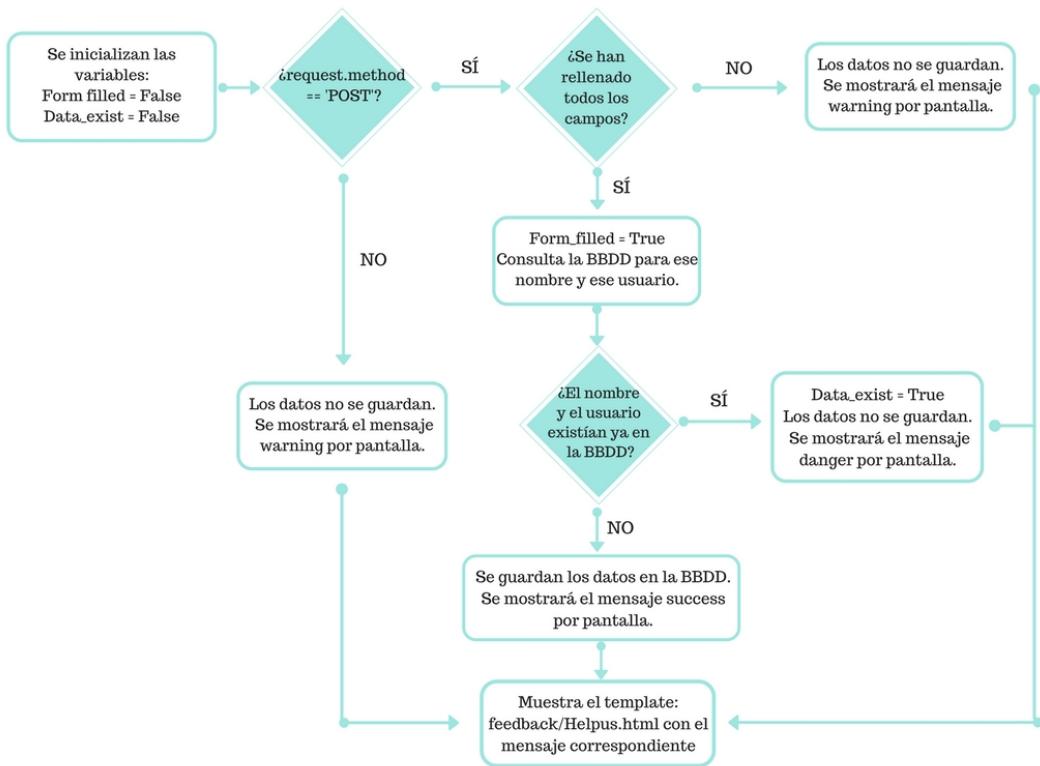


Figura 4.27: Mensajes del formulario

En **getFeedback()** se utiliza la función de Django `is_valid()`, encargada de comprobar si los campos introducidos en el formulario son válidos. En el caso de que alguno de los campos no esté completo o no coincida con lo indicado en `forms.py`, se mostrará por pantalla el *mensaje warning*. Si, por el contrario, el formulario está completo y todos los campos son correctos, esta función pondrá `form_filled` a True y obtendrá toda la información del formulario.

El siguiente paso es comprobar que los campos de `name` y `user` no existen ya en un mismo registro de la base de datos. Si todo es correcto, se guardarán los datos en la BBDD y se mostrará por pantalla el *mensaje success*, en caso contrario, se pondrá `data_exist` a True y, al llamar al template, se mostrará por pantalla el *mensaje danger* indicando el error.

Para poder comprender el trabajo de la función **getFeedback**, se muestra su algoritmo a continuación:

Figura 4.28: Algoritmo de la función `getFeedback`.

4.8. Certificado

Con la intención de motivar aún más a los niños, Dr. Pencilcode, ofrece la posibilidad de conseguir tu propio certificado; en él se indica el nombre del proyecto que se ha analizado y la puntuación asociada al mismo, tal y como se muestra en la *Figura 4.30*. Son varios los documentos/funciones que participan en la generación del diploma:

- `certix.tex` es la plantilla LaTeX del certificado; en ella se insertan el nombre del proyecto y la puntuación que se pasan como parámetros a `pyploma.py`.
- `output.tex` es un fichero LaTeX que se crea cada vez que un usuario se descarga el diploma; en él se guardan todos los cambios realizados en `certix.py`.
- `pyploma.py` es el script encargado de gestionar los documentos `certix.tex` y `output.tex`, así como de realizar la conversión de `output.tex` a `output.pdf`.

- `download_certificate` es la función de `views.py` encargada de llamar a `pyploma.py` pasándole como parámetros el nombre del proyecto y su puntuación total. Una vez que se ha generado el PDF del certificado, esta función crea una respuesta HTTP que permitirá descargar el diploma automáticamente.



Figura 4.29: Certificado Dr. Pencilcode

Para descargar el diploma, como en el caso del formulario, hay que pinchar en el botón **Download Certificate** habilitado en la página de análisis. Cuando el botón es pulsado se lleva a cabo una petición HTTP POST con el recurso `/download_certificate` y `urls.py` enlaza con la función, de mismo nombre, `download_certificate` de `views.py`.

Download_certificate llama a la función `pyploma.py` pasando como parámetros el nombre del proyecto analizado y su puntuación total; ésta crea un fichero LaTeX `output.tex` para cada usuario. Una vez que los nuevos parámetros se insertan en la plantilla `certix.tex`, los cambios se guardan en el fichero `output.tex` creado para ese usuario y se pasa a documento PDF.

Finalmente, `download_certificate` crea una respuesta HTTP con MIME application/pdf y con el documento `output.pdf`, de esta forma el diploma se descargará automáticamente.

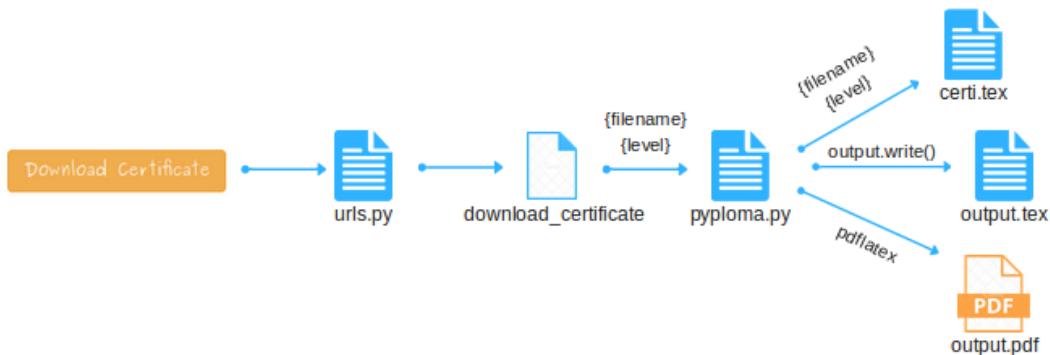


Figura 4.30: Esquema certificado

4.9. Diferencias entre Dr. Pencilcode y Dr. Scratch

Algunas funcionalidades de Dr. Pencilcode ya existían previamente en Dr. Scratch, sin embargo, para poder adaptarlas a Pencil Code, la mayor parte de los archivos se han tenido que modificar.

A estas modificaciones se incluyen también las nuevas funcionalidades y plugins desarrollados en este proyecto, como las 9 páginas de ayuda (`Move.html`, `Art.html`, `Text.html`, `Sound.html`, `Control.html`, `Operators.html`, `Points.html`, `Bonus.html` y `Blocks.html`), las nuevas secciones en la página de `dashboard.html` (bonus, botones, details, etc), el formulario y el plugin `coffee-mastery.py`.

Para poder demostrar que, realmente, ha habido modificaciones, he realizado una comparación entre el repositorio de la última versión de Dr. Pencilcode y el repositorio master de Dr. Scratch.

```
cd drPencilcode  
git remote add other https://github.com/jemole/drScratch/  
git fetch other  
git diff --shortstat other/master  
  
sara@vaio:~/drPencilcode$ git diff --shortstat other/master  
357 files changed, 10963 insertions(+), 19603 deletions(-)
```

Figura 4.31: Número total de modificaciones

En la *Figura 4.31* se puede observar el número de documentos y ficheros que se han modificado en todo el proyecto así como la cantidad de inserciones y supresiones.

Capítulo 5

Resultados

Una vez explicadas las tecnologías utilizadas para el desarrollo de la web y todas las funcionalidades que ésta dispone, es importante dejar claro cómo funciona la herramienta, qué información recibe el aprendiz del análisis de sus proyectos y qué puede hacer con dicha información, en definitiva, qué ofrece Dr. Pencilcode. Por ello, en este capítulo, se simularán distintas situaciones ejemplo de análisis de proyectos para observar el resultado.

El primer programa a analizar es el mostrado en la *Figura 5.1*. Se trata de un ejemplo sencillo por lo que se espera que la puntuación alcanzada no sea demasiado alta. El código hace que la tortuga dibuje una flor y luego la rellene de color naranja.

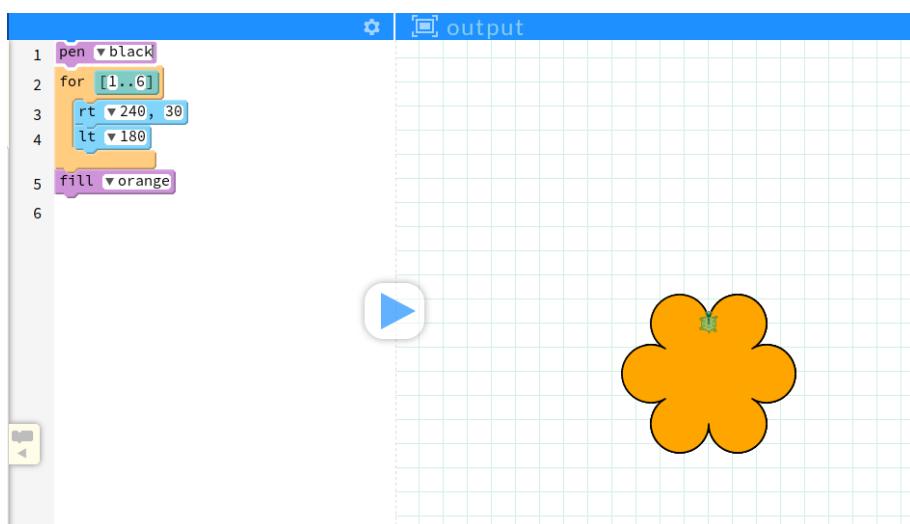


Figura 5.1: Programa Ejemplo 1

Una vez introducida la URL del programa en la herramienta, el resultado obtenido y el que se muestra en la página dashboard.html tras realizar el análisis es el mostrado a continuación:

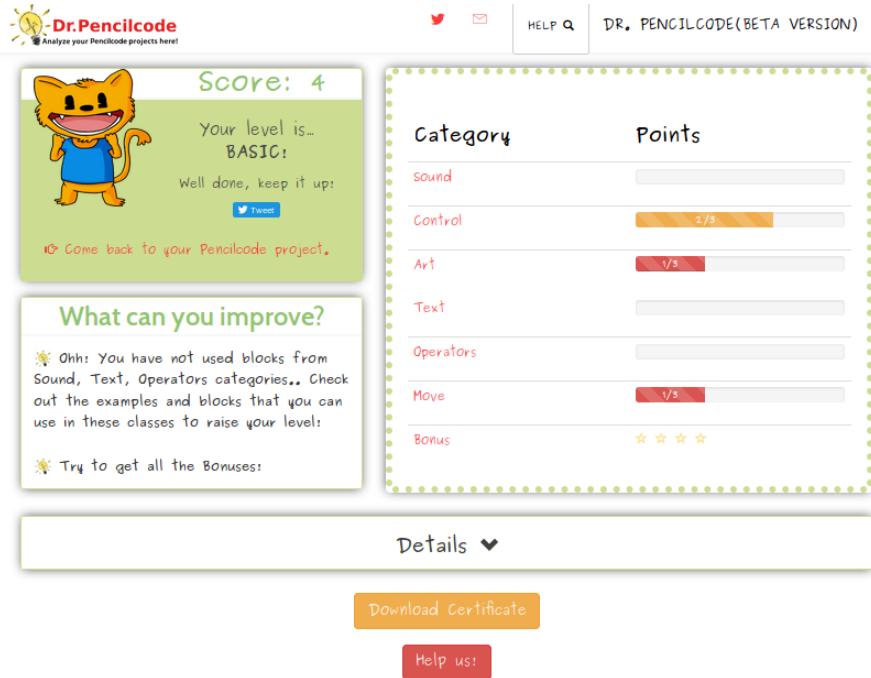


Figura 5.2: Análisis Ejemplo 1

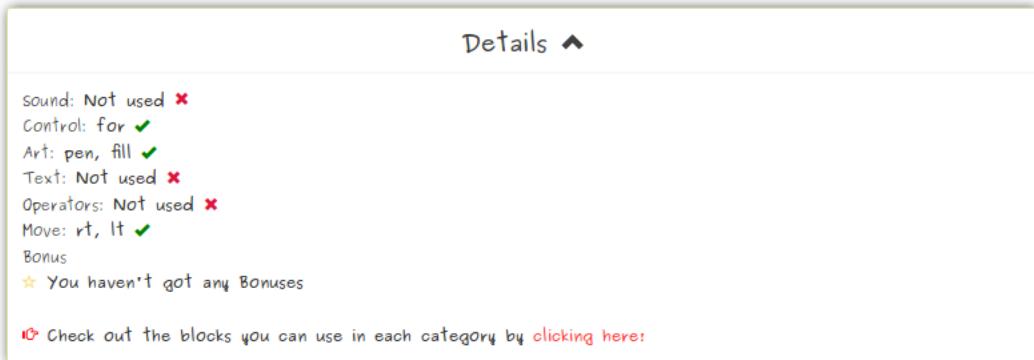


Figura 5.3: Sección Details Ejemplo 1

Con la información que se muestra por pantalla, el usuario tiene la opción de investigar y navegar por cada una de las secciones del dashboard. Se puede observar que la puntuación del proyecto es de 4 debido a la escasez de bloques utilizados. La sección *What can you improve?* da consejos sobre como mejorar tu programa; en este caso, hace hincapié en trabajar las cate-

gorías de Operators, Text y Sound y anima al usuario a conseguir todos los Bonus.

Si el usuario no sabe como mejorar su código, éste puede pinchar en los enlaces de las categorías que conducen a las páginas de ayuda; en ellas, puede encontrar ejemplos de varios programas donde se explica como utilizar los bloques, así como diversas tablas donde se indican los niveles a los que pertenece cada bloque. Por ejemplo, en la página de ayuda de la categoría **Control** se incluye un ejemplo de como llegar al nivel 3 de la misma mediante el uso de funciones. Véase la *Figura 5.4*.

Dr.Pencilcode
Analyze your Pencilcode projects here!

Functions

Level 3

With Pencil Code it's possible to create functions. A function is a list of commands for the computer to execute. There are many created functions that you can use in Pencil Code, but also you can write your own functions. In order to start a function you have to use ->. Once you have written your fuction, use do to run it.

Functions can have **arguments** which are a piece of information that you give to the function. If your function has arguments, you do not need to use do to run it.

For example:

```

list = [ red, orange, green ]
circles = (color) ->
  box ▾grey, ▾50
  dot ▾color, ▾25
  fd ▾50
for color in list
  circles color
  
```

The preview window shows a turtle on a grid, drawing three colored circles (green, yellow, red) in sequence.

How does this program work? First, there is a list with three different colors. Second, we create a function with the argument 'color'. Every time the **function** is called, the turtle will draw a grey **box**, a **dot** with the color we pass as argument and, after that, it will move forward 50px. Finally, the loop **for** calls the function **circle** 3 times, passing it the colors of list as arguments. That's why the turtle will draw red, orange and green dots.

Click for more examples of this Category:

- Mastering Control Flow.
- Basic Math Operations.
- Functions.

Figura 5.4: Nivel 3 de Control

Una vez que se han consultado las páginas de ayuda, el usuario puede regresar al código de su proyecto para seguir mejorándolo. Para mejorar el programa Ejemplo 1, se han añadido los bloques de: función -> (nivel 3 de Operators), click (nivel 3 de Control), jumpto (nivel 3 de Move)... Ahora, la tortuga dibuja flores de colores aleatorios en las coordenadas donde el usuario hace click con el ratón, *Figura 5.5*.

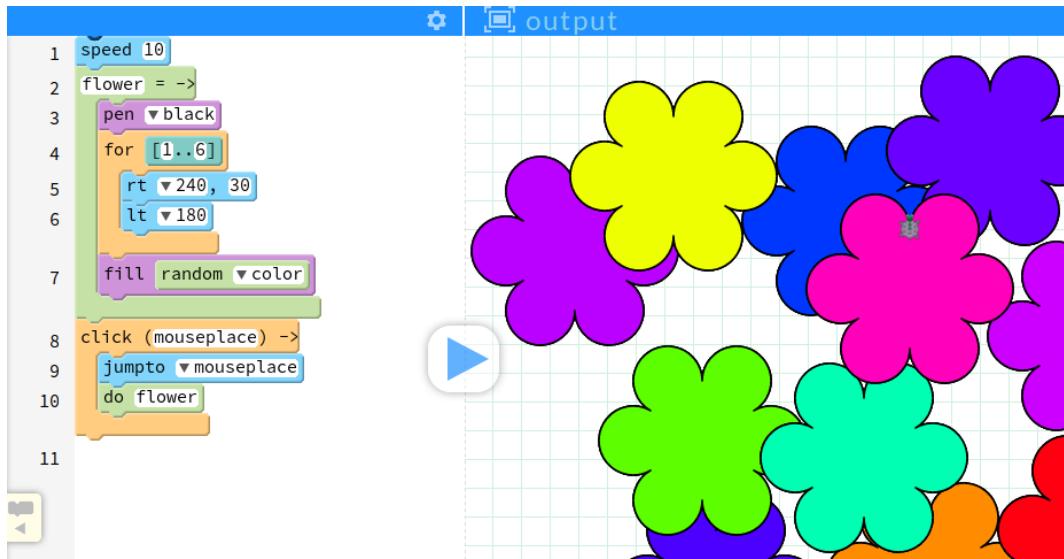


Figura 5.5: Programa Ejemplo 2

Una vez añadidos los bloques convenientes, se vuelve a analizar el proyecto. El nuevo resultado es el que se muestra en las *Figuras 5.6 y 5.7*.

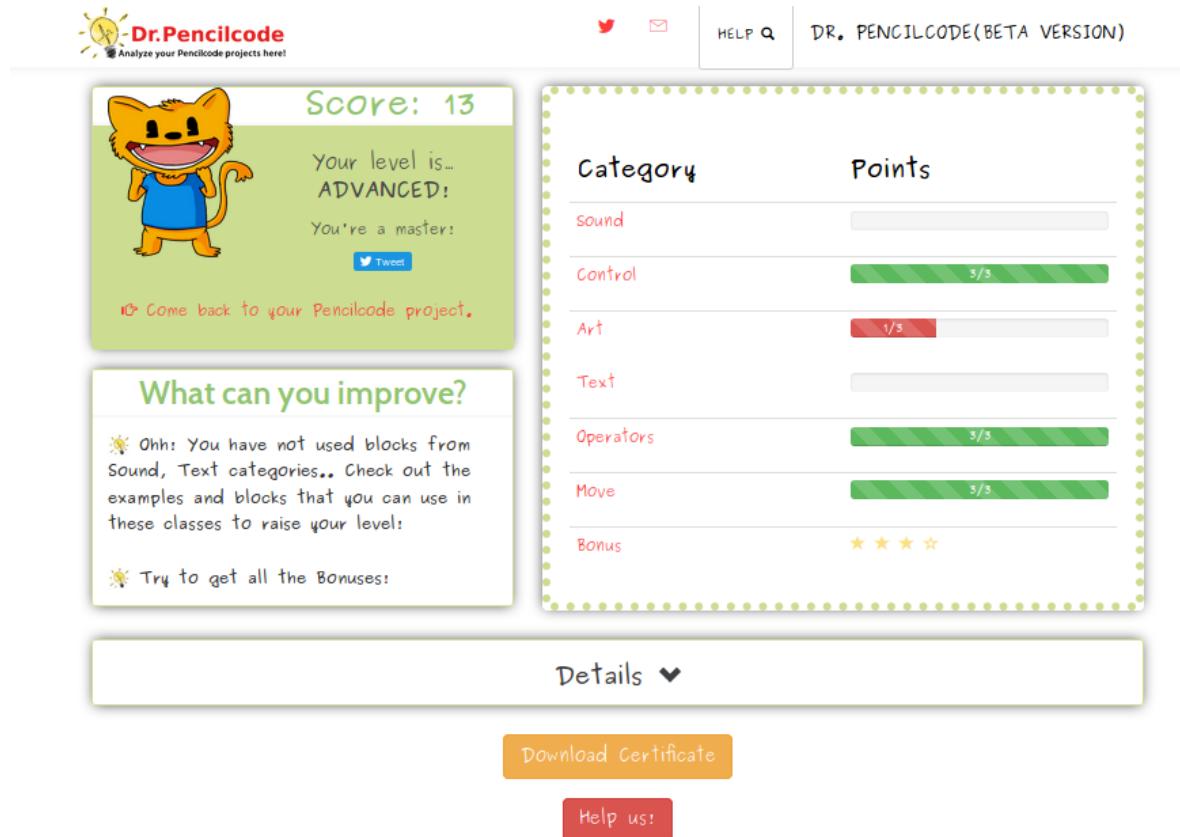


Figura 5.6: Análisis Ejemplo 2



Figura 5.7: Sección Details Ejemplo 2

Como se puede observar, la puntuación del programa ha pasado de 4 a 13. Además de mejorar las puntuaciones específicas de 3 categorías, se han conseguido Bonus por diversidad, complejidad y repetición de bloque. Aun así, el usuario podría seguir mejorando el código añadiendo bloques de Text y Sonido como se indica en la sección *What can you improve?*.

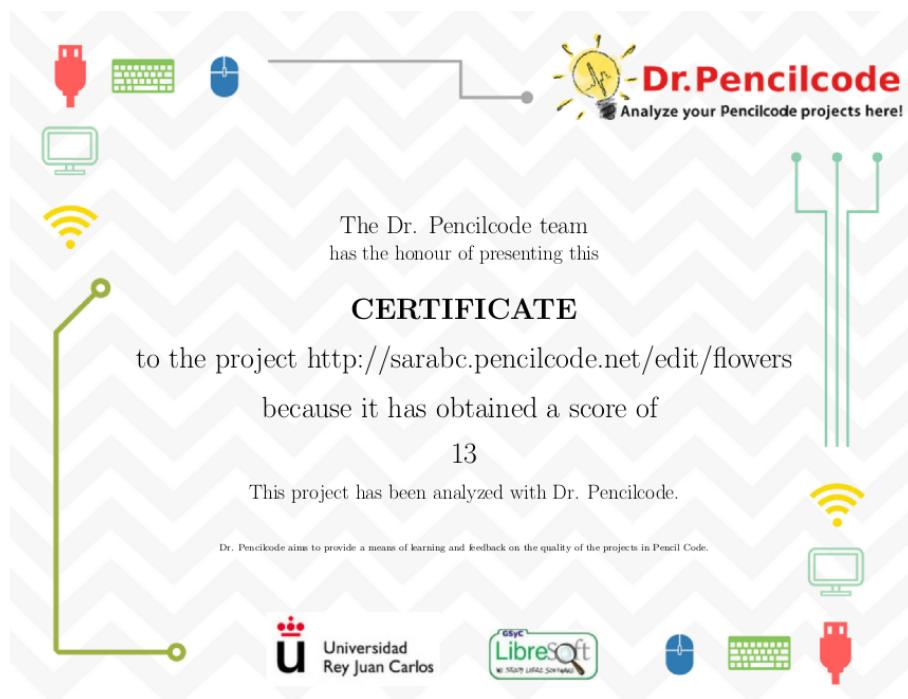


Figura 5.8: Certificado Ejemplo 2

Capítulo 6

Conclusiones

En esta sección hablaré sobre los objetivos que se han logrado alcanzar y los que no durante el desarrollo de la aplicación, explicando el motivo por el cual, estos últimos, no han podido llevarse a cabo. Además, realizaré una pequeña reflexión sobre todo lo que este proyecto me ha aportado y como lo aprendido a lo largo de la carrera ha sido de gran utilidad para el desarrollo del mismo. Finalmente, indicaré nuevas funcionalidades que podrían incluirse, en un futuro, para mejorar la plataforma y aportaré una breve valoración personal.

6.1. Consecución de objetivos

En primer lugar, el objetivo general de este proyecto era crear una aplicación web capaz de analizar los programas escritos en Pencil Code (CoffeeScript) a la par que se daba soporte y consejos para que los usuarios siguieran mejorando sus habilidades como programadores. A pesar de que la herramienta no ha sido probada en aulas reales, se puede decir que este objetivo sí se ha cumplido.

En cuanto a los objetivos específicos:

- **Adaptación de la implementación básica a Pencil Code.** Como se ha ido mostrando en los diferentes capítulos de la memoria, este objetivo se ha alcanzado con creces. Para llegar a ello, se han tenido que realizar numerosas modificaciones en el código de Dr.Scratch además de incluir nuevas funciones.
- **Creación de un Plugin.** Ligado al primer objetivo. Para poder realizar el análisis de los

programas Pencil Code se tuvo que crear el plugin `Coffee-mastery.py` encargado de ofrecer información sobre los bloques utilizado y las puntuaciones; objetivo que también se ha cumplido.

- **Añadir nuevas funcionalidades y secciones a la página web.** En este objetivo no se especificó un número concreto de adiciones. Las novedades que se llevaron a cabo para alcanzar dicho objetivo fueron: la implementación de un formulario, el diseño de 9 páginas de ayuda, la inserción de bonus y el cambio de apariencia del dashboard incluyendo 3 nuevas secciones.
- **Aplicación en producción.** Con gran esfuerzo y después de varios intentos, se consiguió subir la web al servidor. A pesar de haber estado disponible durante todo este tiempo, no ha sido posible completar al 100 % este objetivo debido a que la aplicación no ha sido probada todavía por ningún usuario y, por lo tanto, no se ha recibido ningún tipo de retroalimentación por parte de ellos. La idea principal era que, una vez terminada la web, los alumnos de Yana probaran nuestra plataforma ya que, al fin y al cabo, son ellos los que trabajan con Pencil Code. Al terminar el desarrollo, nos pusimos en contacto con Yana pero, por desgracia, no obtuvimos respuesta a tiempo para poder incluir sus resultados en esta memoria.

6.2. Aplicación de lo aprendido

A medida que he ido trabajando y avanzando en el proyecto, he sido testigo del valor de los conocimientos adquiridos en las asignaturas cursadas a lo largo de la carrera. Este proyecto ha sido un claro ejemplo de ello ya que, durante su desarrollo, he aplicado gran parte de lo aprendido en asignaturas como:

1. **Informática I:** me permitió un primer contacto con el mundo de la programación, gracias a ella, aprendí, desde cero, las técnicas y conceptos básicos de la programación de computadores. Qué era una función, qué era un procedimiento, un bucle, un array, una condición, etc. Términos que han sido necesarios para poder llevar a cabo este proyecto.
2. **Informática II:** Al igual que Informática I, esta asignatura continuó con mi formación como programadora, aunque el lenguaje con el que trabajé fue Ada y no el utilizado en este

proyecto (Python), fue muy útil para seguir mejorando mis habilidades y conocimientos.

3. **Arquitectura de Internet:** fue la primera que cursé sobre el campo de las Redes de Ordenadores e Internet. En concreto, la asignatura me enseñó cómo está estructurada la red, así como las arquitecturas de los principales protocolos de redes de ordenadores.
4. **Sistemas Telemáticos para Medios Audiovisuales:** siguiendo con los protocolos, aquí conocí el funcionamiento del protocolo HTTP, entre otros.
5. **Protocolos para la transmisión de Audio y Vídeo en Internet:** fundamental para la realización de este proyecto ya que fue aquí donde aprendí a programar en Python y gracias a la cual he podido llevar un seguimiento más sencillo en el desarrollo del fichero views.py.
6. **Construcción de Servicios y Aplicaciones Audiovisuales en Internet:** En la cual, adquirí los conocimientos básicos de HTML, CSS y JavaScript, todos ellos empleados en el desarrollo de Dr.Pencilcode.
7. **Gráficos y visualización en 3D:** Aunque ésta estaba orientada a la creación de videojuegos tanto en 2D como 3D, los lenguajes utilizados para ello fueron WebGL y Javascript, lo que me permitió seguir puliendo mi pensamiento computacional.

6.3. Lecciones aprendidas

Con el desarrollo de Dr.Pencilcode he conseguido ampliar y mejorar los conocimientos adquiridos durante la carrera además de aprender otros totalmente nuevos. Entre ellos cabe destacar el aprendizaje de nuevas tecnologías - nunca antes vistas - como por ejemplo:

1. **Django** es la base de este proyecto, para poder llevarlo acabo ha sido necesario informarse y profundizar en este framework de desarrollo web desde cero.
2. **Diseño gráfico web:** Además de ampliar mis conocimientos de HTML y CSS como desarrolladora de front-end, también he aprendido a utilizar **Bootstrap** y **jQuery** para añadir mayor interacción, usabilidad y dinamismo a mis páginas web, dándoles un aire más profesional.

3. **Bases de datos:** Gracias a Dr.Pencilcode he aprendido a trabajar con bases de datos con sistemas como **MySQL** o **Sqlite** con el fin de guardar y gestionar la información de los usuarios que utilizan la aplicación para el análisis de sus proyectos.
4. **Conceptos avanzados de Python:** A medida que el proyecto iba avanzando, he mejorado mis habilidades de programadora en Python utilizando nuevas estructuras y nuevos módulos como *sets* y *expresiones regulares* (regex).
5. **Servidor Apache y maquinas virtuales.** Nunca antes había subido una web a un servidor por lo que, al igual que con Django, tuve que aprender de forma autodidacta a instalar y configurar el servidor web Apache.

No todo lo aprendido está relacionado con las tecnologías. Participar en este proyecto, ligado a *Scratch* y *Dr.Scratch*, me dio la oportunidad de formar parte de diversos talleres orientados a niños y personas con discapacidad donde daba soporte y consejos sobre programación.

Con esto, entra en juego un nuevo concepto que nunca antes había escuchado, el **pensamiento computacional**. He descubierto que con ayuda de los conceptos básicos de la programación es posible aprender a resolver problemas de la vida cotidiana, analizando la información, organizándola, identificando posibles soluciones de forma algorítmica (pasos ordenados), etc. La introducción de la programación en la educación promueve el desarrollo del pensamiento computacional.

Estos eventos, escasos pero intensos, me ayudaron a desenvolverme un poco más como persona e impulsaron mi deseo de seguir creciendo con este proyecto con el fin de enseñar lo útil que puede llegar a ser la programación en la enseñanza.



Figura 6.1: Taller de superprogramadores en FICOD

Por último, he aprendido a realizar evaluaciones objetivas de los proyectos analizados. Quizás, esta parte es la que más tiempo me ha llevado pues la creación de una **rúbrica**, la evaluación de una tarea, engloba numerosas variables que hay que tener en cuenta. Esta tarea me ha ayudado a ver las cosas desde el punto de vista del docente y no del alumno.

6.4. Trabajos futuros

Existen muchas funcionalidades que podrían añadirse a Dr.Pencilcode para que fuera una herramienta más completa y óptima, algunas de ellas podrían ser:

1. **Traducción a otros idiomas.** Actualmente, Dr. Pencilcode solo se encuentra disponible en inglés por lo que una buena opción sería traducir la plataforma a más idiomas como, por ejemplo, el castellano.
2. Una forma para seguir mejorando la plataforma sería añadir **cuentas de usuario**; con ellas, el usuario podría hacer un seguimiento de sus puntuaciones, sus proyectos analizados y su progreso y evolución como programador.

3. **Añadir nuevos bonus y medallas.** Estaría bien incluir en la rúbrica actual nuevos bonus y medallas teniendo en cuenta más aspectos del código y combinaciones de los bloques.
4. Muy similar al punto anterior, se podría añadir un **sistema de mini-retos** para tener un incentivo extra y hacer más divertida y amena la programación.
5. **Mejorar el plugin coffee-mastery.** En lugar de tratar los bonus en el fichero `views.py` se deberían calcular todas las puntuaciones en dicho plug-in.
6. Por último, una buena vía para comprobar que Dr. Pencilcode cumple su labor y da buenos resultados, sería llevar la **plataforma a las aulas** españolas aunque, para poder realizarlo, sería necesario que Pencil Code se popularizará en nuestro país.

6.5. Valoración personal

He tenido la suerte de elegir una carrera con un amplio abanico de posibilidades. Sin embargo, me ha llevado más de 5 años saber lo que realmente me motiva y mi decisión final se la debo, en gran parte, a este proyecto. Con el desarrollo de Dr. Pencilcode me he dado cuenta de que la programación es lo que verdaderamente me gusta pero, sobretodo, el hecho de poder ayudar a otras personas con ella y la gratificación y la satisfacción que eso conlleva.

Se cierra una etapa de mi vida para dar comienzo a otras nuevas, de las que estoy segura que me ayudarán a formarme para llegar a ser una buena programadora.

Apéndice A

Información Extra

A.1. E-mails con Yana

En este anexo se muestran algunos de los e-mails que se intercambiaron con David Bau y Yana Malyshева sobre el desarrollo de Dr.Pencilcode

 David Bau <davidbau@mit.edu> Responder | v
mié 15/06/2016 19:02
Para: Gregorio Robles (grex@gync.urjc.es); Yana Malysheva (yanamal@google.com)
Cc: Jesús Moreno (jesus.moreno@programamos.es); sara blazquez (sara_gr138@hotmail.com) ✉

Bandeja de entrada

Hello Gregorio! Sorry for the delay in getting back.

My research direction has taken me away from Pencil Code this year. I would love to put you in touch with a Googler Yana Malyshева (cced) who is taking up the lead with Pencil Code this year. I'll ask her to take a look and provide feedback. She is also based in Boston - would it be OK to invite her to the seminar? And perhaps while you are in town it would be a good time to meet - what days are you in Boston?

David

Figura A.1: E-mail David Bau

 Yana Malysheva <yanamal@google.com>
vie 17/06/2016, 1:58

◀ | ▾

Hi Gregorio,
Thanks! I signed up for the seminar on eventbrite, it sounds great!

Dr. Scratch/ Dr. Pencilcode is pretty cool.
I only tried out Dr. Pencilcode, since I don't have any Scratch projects on hand, but I guess they are still very similar to each other so far?

I really like the idea of analyzing student code for certain elements/constructs and giving them ideas on how to take it a step further. I think from a gamification perspective, that kind of feedback would probably work even better as more qualitative "badges"/achievements, rather than quantitative levels. Framing them as levels makes it seem like the more complex your project is, the better. So you're almost setting the goal of coming up with a project that throws all the complex

Ideally, I guess, the feedback would be more around what kinds of constructs the student uses overall, not necessarily in a single project. But I think a good step toward that having a set of individual checkboxes, with some of them unchecked, that don't translate into a single 'score'.

Anyway, I really like the idea of looking for specific concepts and suggesting a next step for the student, so Dr. Pencilcode/Scratch seems very exciting to me. Especially if and when it connects directly to the code in one interface. Would love to chat about your plans for it!
I think I will be pretty free June 23rd-24th; I think I will see you at the seminar, as well.

Thanks,
-Yana

Figura A.2: Primer e-mail de Yana

 Yana Malysheva <yanamal@google.com>
mar 05/07/2016 21:47
Para: Jesús Moreno (jesus.moreno@programamos.es); Gregorio Robles (grex@gsc.urjc.es); sara.blazquez (sara_gr138@hotmail.com) ↗

Responder | ▾

Bandeja de entrada

Hi all,

Here are some thoughts on potential dimensions/concepts for Dr. Pencilcode.
This is basically a huge braindump so feel free to skim and pick and choose whatever seems relevant. Also, I'm happy to write little illustrative Pencil Code programs for any concepts that seem particularly interesting and/or confusing.

Starting from the Dr. Scratch dimensions:

I think some of them are directly applicable to Pencilcode, in particular **Logical Thinking** and **Flow Control** (Pencilcode also has a forever loop, and it's usually easier to start with it and transition to the for [1..5] style loop.)
Some additional advanced concepts somewhere within these two areas could be:

- nested loops/conditionals
- adding an iteration parameter and using it, i.e. "for i in [1..4]" instead of just "for [1..4]"
- while loops, looping over a data structure, other advanced type of loops?

Parallelism and **Synchronization** are probably the most Scratch-specific, and least transferrable to Pencil Code.
There is some stuff like that, but it doesn't tend to be prominent when we teach beginners.
for example:

Figura A.3: E-mail sobre las categorías de la Fase I

Hi Gregorio,

sorry, I did take a look at it back then, but then I did get pretty busy and never got around to writing back!

It's looking good! I felt compelled to click around and figure out what I needed to do to improve my various scores. (although right now, it looks like all those "want to know more" links are broken?)

I think it'd be really useful to have a snippet of text for each category, and for each level, that goes right into the chart on the left and gives me a hint about what to add to get to the next level.

On the technical side, I ran Dr. Pencilcode on this project again: <http://yanamal.pencilcode.net/edit/PhysicsGame>
And noticed that it attributed a couple of things to me that I didn't actually do. For example, it said I used a "stop" command in the sound category, but I didn't (directly) do that. It also said that I used a while loop, but there's no while loop in the code.

I wonder if that's because Dr. Pencilcode is running on some kind of expanded, instrumented version of my code, not literally what I have on the screen? The Pencilcode IDE does add quite a bit of stuff behind the scenes.

Anyway, looks exciting, thanks for keeping me updated!

-Yana

Figura A.4: E-mail sobre las páginas de ayuda

YM Yana Malyshova <yanamal@google.com>
mar.20/12/2016, 23:05

That survey sounds good to me - the only part I found confusing is question 1b, "Has the information displayed on the screen helped you to understand what the web is for?" (emphasis mine).
did you mean "what this website is for"? or is/was there part of dr. pencilcode that is supposed to talk about the web as a whole?

Separately, I'm not sure whether you've tested this wording with kids before and what age kids you're targeting, but I'd be a bit concerned about words like "information", "improvement", "sufficient", "incorporate" - I think there are simpler wordings you might be able to use? let me know if you want me to try and come up with an example alternative wording.

-Yana

Figura A.5: E-mail sobre el formulario

A.2. Formulario Dr. Pencilcode

Aquí se muestra el primer formulario a papel que se hizo de Dr. Pencilcode.

Dr. Pencilcode workshop – Survey

Task 1. Visit Dr. Pencilcode website: <http://drpencilcode.libresoft.es/> or <http://193.147.79.206/>

a. What do you think about the website? Do you think it is appealing?



b. Has the information displayed on the screen helped you to understand what the web is for?



Task 2. Analyze one of your Pencilcode projects with Dr.Pencilcode.

a. Was it easy to do the analysis?



b. Do you understand the results shown?



c. How did you feel when you saw the results?



d. Why?

Task 3. From the results page, after analyzing a project, click on any of the links “Want to know more?” to get information that can help you improve your program.

a. Write the name of the page you have clicked on:

b. Do you understand the information displayed on this new page?



c. After reading the information, Do you feel like improving your project by trying something new in Pencilcode?



Task 4. Using the information that has been shown in the selected help page, try to improve your project by incorporating something new. Is the information shown sufficient to understand how to make the improvement?



Task 5. After following the instructions on the help page, re-analyze your Pencilcode project with Dr. Pencilcode. Have you got a higher score/percentage?



Task 6. Comment whatever you want about Dr. Pencilcode: things you like, things you don't, ideas that we could incorporate...

Bibliografía

- [1] Django girls tutorial. <https://tutorial.djangogirls.org/en/>.
- [2] Python Software Foundation. Data Structures: Sets.
<http://docs.python.org/3/tutorial/datastructures.html>.
- [3] Orden ECD/65/2015, 21 de enero, de Educación. Boletín Oficial del Estado. (25):6995–6996, 29 de enero 2015.
<http://www.boe.es/buscar/doc.php?id=BOE-A-2015-738>.
- [4] M. L. A. Jiménez. *Dr. Scratch: Fomentando la creatividad y la vocación científica con Scratch*. 2016.
- [5] M. J. Kabir. *La Biblia de Servidor Apache 2*. Anaya Multimedia, 2003.
- [6] M. Liu. *Computación Distribuida: Fundamentos y Aplicaciones*. Pearson Education, 2004.
- [7] J. E. Pérez. *Introducción a CSS*. Diciembre 2008.
- [8] J. E. Pérez. *Introducción a JavaScript*. Junio 2008.
- [9] J. E. Pérez. *Introducción a XHTML*. Diciembre 2008.
- [10] B. Totty and D. Gourley. *HTTP: The Definitive Guide*. O'Reilly Media, Inc., 2002.
- [11] J. M. Wing. Computational thinking. *Communications of the ACM*, 49:33–35, 2006.
<http://www.cs.cmu.edu/15110-s13/Wing06-ct.pdf>.