



INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2016/2017

Trabajo Fin de Grado

CREACIÓN DE LA PLATAFORMA DR.PENCILCODE

Autor : Sara Blázquez Calderay

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

CREACIÓN DE LA PLATAFORMA DR.PENCILCODE

Autor : 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 agradecerles infinitamente:

En primer lugar, quiero dar las gracias a mi familia, me siento afortunada por tenerlos 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, 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, humor Blázquez, no sé que hubiera sido de mí sin eso. Sois las personas a las que más admiro del mundo, gracias por confiar en mí más que yo misma. Os quiero mucho.

A Carlos y Marivi, segundo pilar

A mis hermanos, Alex y Nuria, que siempre sacáis lo mejor de mí. Sois el regalo más grande que he podido tener. Vosotros me transmitís toda la energía que necesito para seguir, 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 poco tiempo, me ha ayudado a crecer como persona y, por haberme enseñado grandes lecciones. Aunque esteis 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 area 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 el fin 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 y retroalimentación 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 interatividad 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 support and 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
1.1.1. Impacto de la tecnología y la programación en la actualidad.	1
1.1.2. La programación en la educación.	1
1.1.3. El Pensamiento computacional.	2
1.2. Marco de Referencia	2
1.2.1. Dr. Pencilcode	2
1.3. Estructura de la memoria	2
2. Objetivos	3
2.1. Objetivo general	3
2.2. Objetivos específicos	4
3. Estado del arte	7
3.1. Modelo Cliente-Servidor	7
3.2. HTTP	8
3.3. Frontend	9
3.3.1. HTML	10
3.3.2. CSS	11
3.3.3. JavaScript	12
3.3.4. jQuery	13
3.3.5. Bootstrap	14
3.4. Backend	14
3.4.1. Django	15

3.4.2. Python	16
3.4.3. Servidor HTTP Apache	18
3.4.4. Mysql	18
3.4.5. CoffeeLint	20
3.5. Pencil Code	20
4. Diseño e implementación	25
4.1. Arquitectura general	25
4.1.1. Bases de datos	25
4.2. Scratch y Dr. Scratch	25
4.3. Diferencias entre Dr. Pencilcode y Dr. Scratch	25
4.4. Página principal	26
4.5. Rúbrica de Dr. Pencilcode	26
4.5.1. FASE I	26
4.5.2. FASE II	27
4.5.3. FASE III	27
4.6. Dashboard	28
4.6.1. FASE I	28
4.6.2. FASE II	29
4.6.3. FASE III	31
4.7. Análisis de proyectos Pencil Code FASE III	33
4.7.1. Análisis por URL	34
4.7.2. Coffee-Mastery	35
4.8. Paginas de ayuda	36
4.9. Formulario	37
4.10. Certificado	41
5. Resultados	43
6. Conclusiones	45
6.1. Consecución de objetivos	45
6.2. Aplicación de lo aprendido	45

<i>ÍNDICE GENERAL</i>	XI
6.3. Lecciones aprendidas	46
6.4. Trabajos futuros	48
6.5. Valoración personal	49
A. Manual de usuario	51
Bibliografía	53

Índice de figuras

3.1. Arquitectura Cliente-Servidor	8
3.2. Modelo mensajes HTTP	9
3.3. Tecnologías utilizadas en el front-end	10
3.4. Documento básico de HTML	11
3.5. Ejemplo de aplicación de estilo CSS	12
3.6. Document Object Model	13
3.7. Patrón de diseño MVC de Django	16
3.8. Ranking de sistemas de gestión de base de datos	19
3.9. Análisis con CoffeeLint	20
3.10. Logo de Pencil Code	21
3.11. Programa creado con Pencil Code.	22
3.12. Categorías de Pencil Code	23
4.1. Total de modificaciones	26
4.2. Dashboard Fase I	29
4.3. Dashboard Fase II	30
4.4. Sección Details Fase II	31
4.5. Dashboard Fase III	32
4.6. Página Final Dashboard	33
4.7. Detalles de los resultados	33
4.8. Errores al analizar un proyecto.	34
4.9. Nuevo error de código	35
4.10. Salida de coffee-mastery.py	35
4.11. Página de ayuda Move	36

4.12. Página de ayuda Points	37
4.13. Botón Help us	38
4.14. Pagina de formulario	39
4.15. Algoritmo del formulario	40
4.16. Mensajes del formulario	40
4.17. Certificado Dr. Pencilcode	41
6.1. Taller de superprogramadores en FICOD	48

Capítulo 1

Introducción

En este capítulo se introduce el proyecto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

1.1. Marco General

1.1.1. Impacto de la tecnología y la programación en la actualidad.

dentro del contexto actual.

1.1.2. La programación en la educación.

La competencia digital es aquella que implica el uso creativo, crítico y seguro de las tecnologías de la información y la comunicación para alcanzar objetivos relacionados con el trabajo, la empleabilidad, el aprendizaje, el uso del tiempo libre, la inclusión y la participación en la sociedad.

Esta competencia supone, además de la adecuación a los cambios que introducen las nuevas tecnologías en la alfabetización, la lectura y la escritura, un conjunto nuevo de conocimientos, habilidades y actitudes necesarias hoy en día para ser competente en un entorno digital.

Igualmente precisa del desarrollo de diversas destrezas relacionadas con el acceso a la información, el procesamiento y uso para la comunicación, la creación de contenidos, la seguridad y la resolución de problemas, tanto en contextos formales como no formales e informales. La persona ha de ser capaz de hacer uso habitual de los recursos tecnológicos disponibles con el fin

de resolver problemas reales de un modo eficiente, así como evaluar y seleccionar nuevas fuentes de información e innovaciones tecnológicas, a medida que van apareciendo, en función de su utilidad para acometer tareas u objetivos específicos.

1.1.3. El Pensamiento computacional.

1.2. Marco de Referencia

1.2.1. Dr. Pencilcode

Hablar sobre la creación de pencilcode y que esta basado en drScratch y que surge la idea de drPencilcode y que es un lenguaje de programación de propósito general por lo que es más complicado de analizar.

1.3. Estructura de la memoria

En esta sección se debería introducir la estructura de la memoria. Así:

- **Introducción:**
- **Objetivos:** Aquí se determinan los objetivos generales y específicos que se fueron fijaron durante el desarrollo de este proyecto.
- **Estado del arte:** En esta sección se enumeran la arquitectura y tecnologías utilizadas en el desarrollo de este proyecto, tanto en el back-end como en el front-end, realizando una breve descripción de cada una de ellas.
- **Diseño e implementación:** En este capítulo se habla sobre la arquitectura general de la herramienta, su diseño y su implementación.
- **Resultados:**
- **Conclusiones:**

Capítulo 2

Objetivos

El principal objetivo de todo proyecto es cubrir la necesidad que ha sido observada y que ha motivado la realización del mismo. Pero para poder llegar a dicho resultado es necesario detallar concretamente y sin ambigüedad que es lo que queremos conseguir en cada etapa. En este capítulo se describen los objetivos iniciales.

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 transmitirles que la programación es muy divertida.

En definitiva, se pretende adaptar la página Dr. Scratch al entorno Pencilcode (teniendo en cuenta que es un lenguaje de propósito general, mucho más complejo que Scratch) y avanzar en

que sea una herramienta para aprender, no sólo evaluar.

2.2. Objetivos específicos

Como se ha mencionado anteriormente, Dr. Pencilcode trata de seguir 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, 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 éstos pudieran dar retroalimentación y ayudar a mejorar la herramienta.

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, 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.

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* 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 interactúan entre sí mediante peticiones y respuestas.

- **Cliente:** es la parte encargada de solicitar las peticiones al servidor, esperar y recibir sus repuestas. 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.

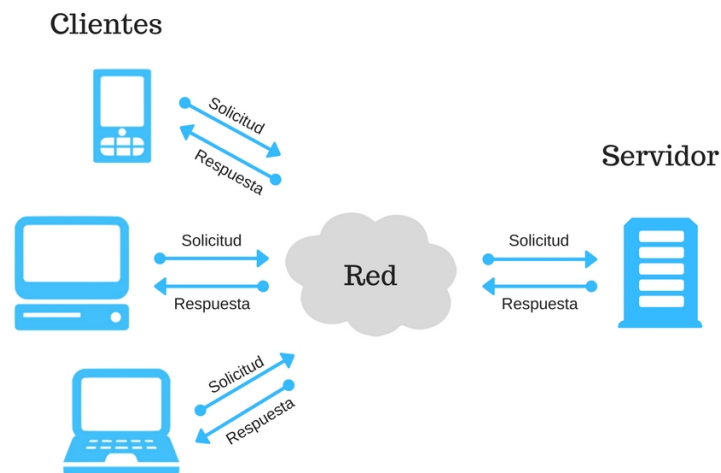


Figura 3.1: Arquitectura Cliente-Servidor

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 negocio y capa de datos (*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 [2]:

- **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 suelen 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, video, 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.

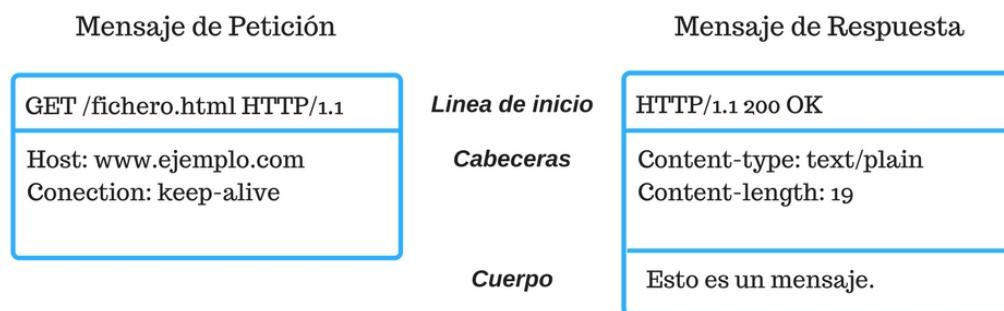


Figura 3.2: Modelo mensajes HTTP

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 (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 como 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 (cuyas siglas en inglés significan Cascading StyleSheets) describen como los elementos de un documento escrito en lenguaje de marcado, como puede ser HTML, deberían de 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 que 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.

<pre><!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></pre>	<pre>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; }</pre>
(a) html	(b) css

Figura 3.5: Ejemplo de aplicación de estilo CSS

3.3.3. JavaScript

JavaScript, conocido comunmente 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 language 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.

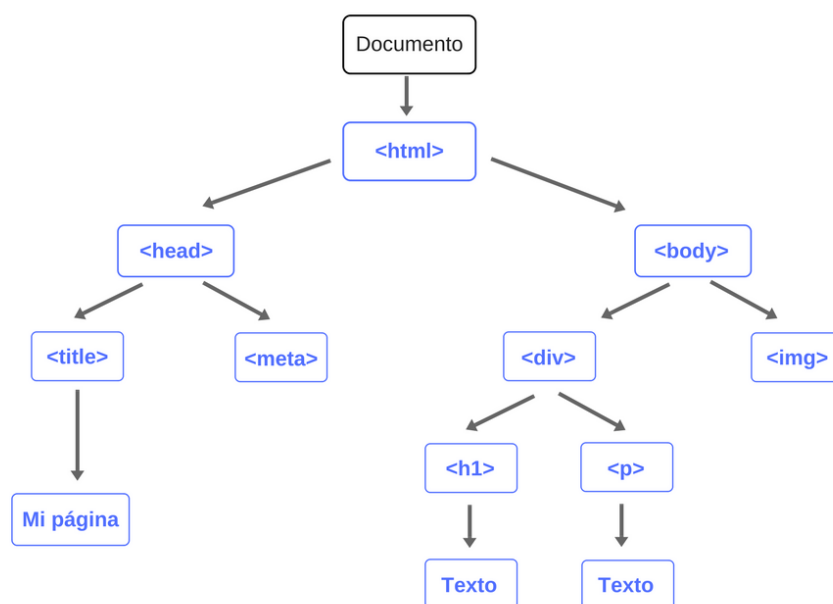


Figura 3.6: Document Object Model

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, por lo que se puede usar de forma gratuita.
- Tiene compatibilidad con la mayoría de los navegadores web.
- Desde la version 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.

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.

¹<http://getbootstrap.com/>

²<https://github.com/twbs/bootstrap>

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.
- **Sqlite3** 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:

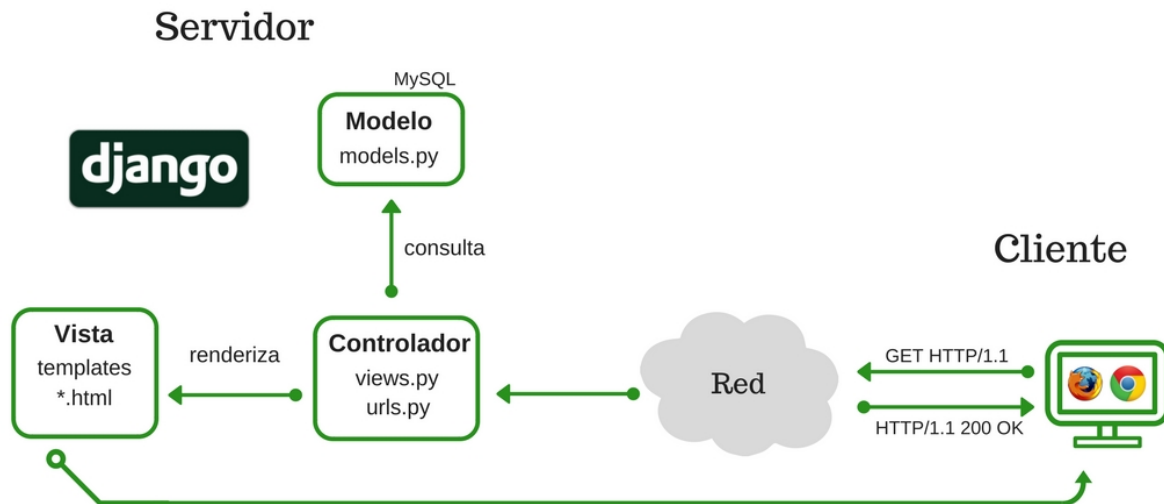


Figura 3.7: Patrón de diseño MVC de Django

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 ocheta y con el objetivo de mejorar y ampliar las funcionalidades del *lenguaje de programación ABC*, **Python**, es un lenguaje de programación de alto nivel, interpretado y orientado a objetos que se caracteriza por su concisión, legibilidad y transparencia. Debe su nombre a los humoristas británicos *Monthy 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<http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>

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 [1].

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 recolectar 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.

⁴<https://httpd.apache.org/es>

⁵<https://db-engines.com/en/ranking>

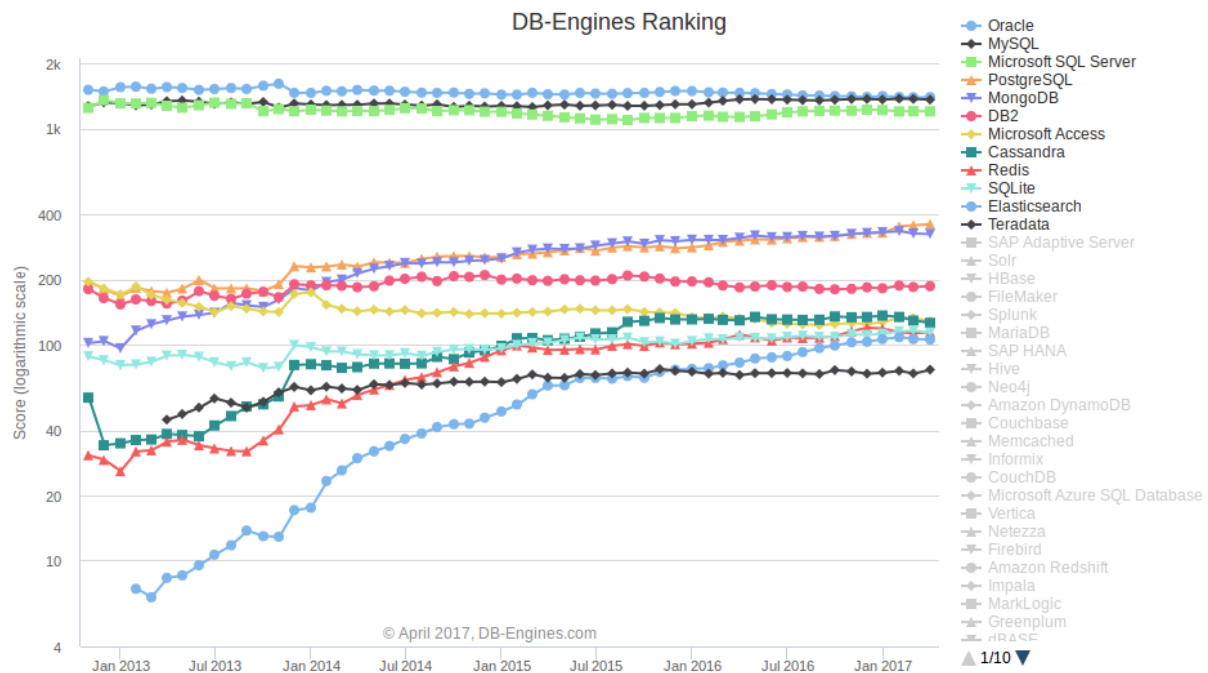


Figura 3.8: Ranking de sistemas de gestión de base de datos

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

CoffeeLint es un analizador estático de programas escritos en CoffeeScript, 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
```

(a) Programa Coffeescript

```
sara@vaio:~$ coffeelint ejemplo.coffee
✓ ejemplo.coffee

✓ Ok! » 0 errors and 0 warnings in 1 file
sara@vaio:~$
```

(b) Análisis

Figura 3.9: Análisis con CoffeeLint

Por defecto, CoffeeLint asegura que el programa esté escribiendo en CoffeeScript 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.

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.

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

⁸<https://pencilcode.net/>



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 estandar 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 quieres trabajar y el otro para pasar de la programación con bloques a texto - y viceversa.

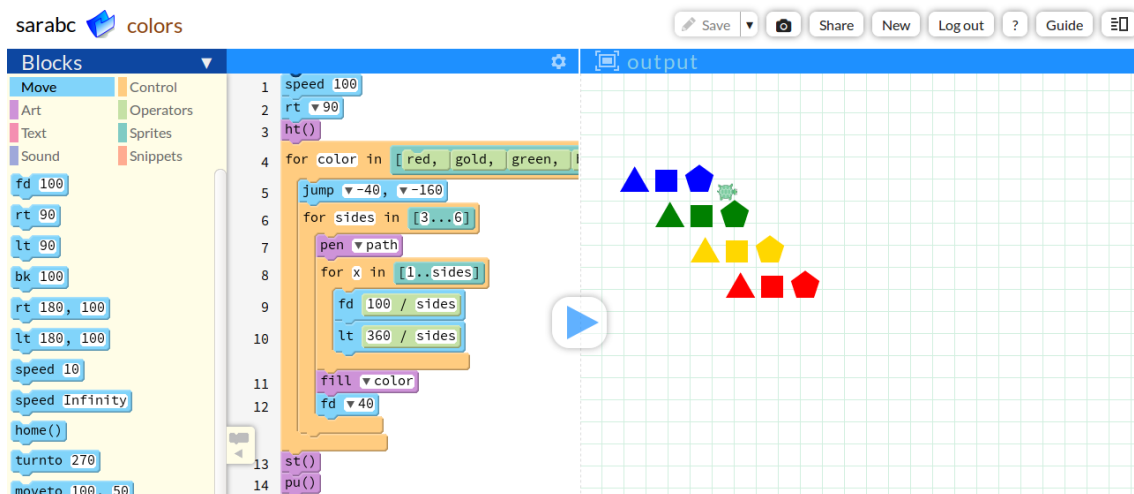


Figura 3.11: Programa creado con Pencil Code.

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 imagenes 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.

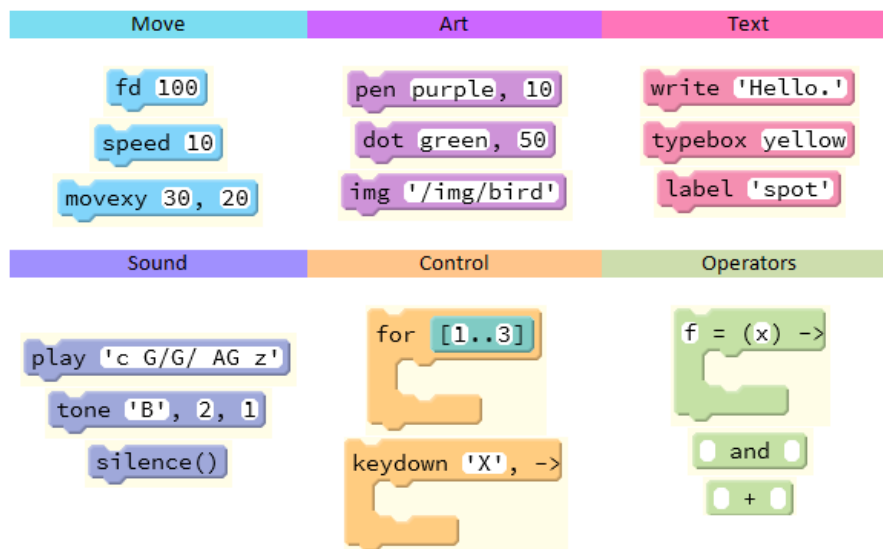


Figura 3.12: Categorías de Pencil Code

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 estos.

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

4.1.1. Bases de datos

En local sqlite ya que es la base de datos que usa django por defecto. Cuando subi la página al servidor cambié a Mysql.

4.2. Scratch y Dr. Scratch

Dr. Pencilcode se basa en scratch.

4.3. Diferencias entre Dr. Pencilcode y Dr. Scratch

Para el desarrollo de Dr. Pencilcode se ha partido de la version de Dr. Scratch disponible en la web. Se han tenido que realizar numerosas modificaciones para llevar a cabo el proyecto. En este apartado solo mencionaré las cosas que se han modificado y las que se han añadido. [Aqui menciono todo lo que se ha cambiado modificado y lo que se ha creado] y luego voy uno a uno. El resto o no se han tocado o tienen cambios insignificantes como añadir algun estilo css, cambios de nombre de variables, etc.

```
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  
325 files changed, 7856 insertions(+), 19604 deletions(-)
```

Figura 4.1: Total de modificaciones

4.4. Página principal

Lo único que cambia:

4.5. 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 la parte 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 ésta disponible actualmente en la web. 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:

4.5.1. FASE I

Al comienzo del proyecto se decidió realizar el análisis de la misma forma que Dr. Scratch. El objetivo era evaluar los proyectos 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 por el usuario.

4.5.2. FASE II

Se evalúa en porcentajes. Esto era inviable debido a la capacidad de Pencil Code de cambiar el modo de programación de bloques a código. El usuario podría escribir más bloques no visibles en las categorías.

4.5.3. FASE III

La última rúbrica y la que se aplica actualmente.

Se busca información sobre los principales bloques utilizados en cada categoría, incluyendo los que no se muestran en la página,

Se decide quitar la puntuación máxima.

En cada categoría se puede obtener un máximo de 3 puntos, dependiendo del tipo de bloque utilizado en dichas categorías, se pondrán 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 completando todos los bonus.

En función de los puntos obtenidos, el nivel de programación del usuario puede ser: Básico, intermedio, avanzado o experto.

puntos ¡6 -¿Básico

6 ¡puntos ¡12 - ¿Intermedio

12 ¡puntos ¡18 -¿Avanzado

puntos ¿18 -¿Experto

According to the points you get, your level as a programmer can be:

Finalmente, se decidió añadir Bonus. Por el momento hay 4 tipos de Bonus:

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

- **Bonus por repetición de bloques:** cuando se usa el mismo bloque más de una vez.

4.6. Dashboard

Se añaden nuevas funcionalidades, sección de consejos sobre el que podría mejorarse.

Categoría Bonus, que se indican con 4 estrellas, una por cada bonus.

Sección Detalles desplegable donde se pueden ver los bloques que se han utilizado en cada categoría. Botón Help us que dirige a una página con un formulario Botón Certificate para descargarse el certificado

A la par que la rúbrica iba cambiando también lo iba haciendo la apariencia de la página.

4.6.1. FASE I

Se distinguen 3 secciones:

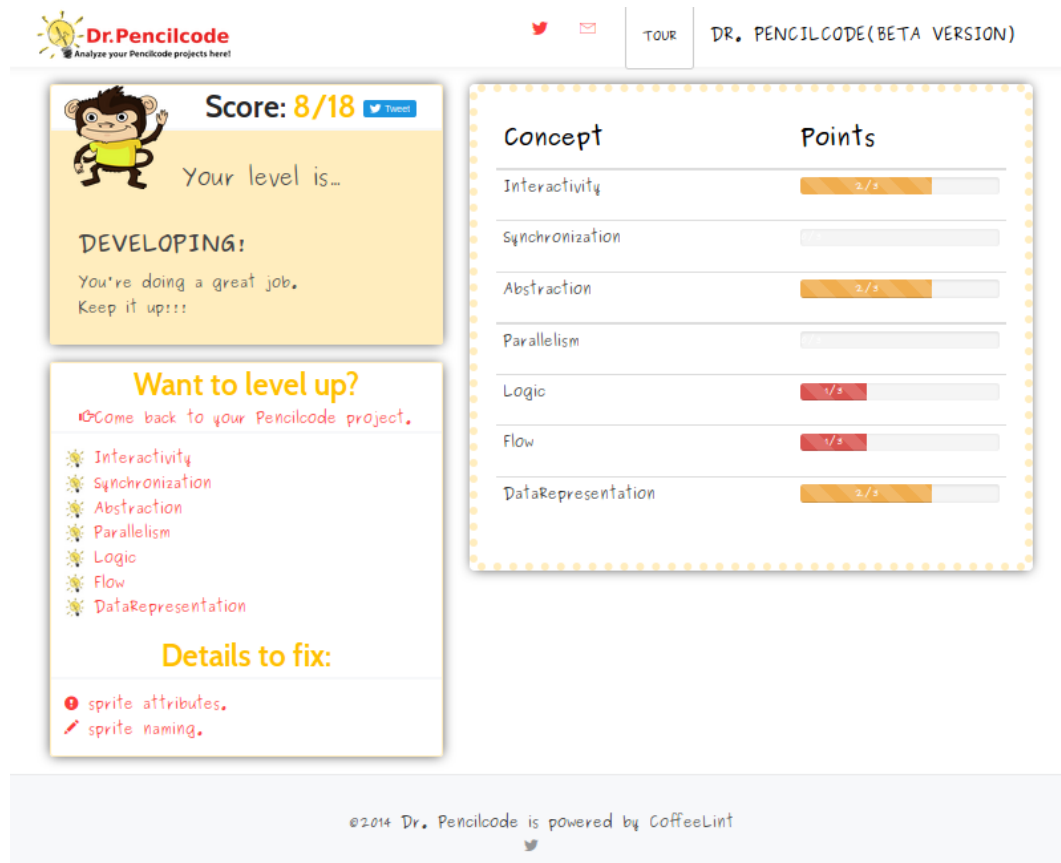


Figura 4.2: Dashboard Fase I

4.6.2. FASE II

El cambio de rúbrica pasando del análisis de los aspectos del pensamiento computacional al análisis de las principales categorías de Pencil Code conllevó la realización de varias modificaciones en el dashboard.

En cuanto a las 3 secciones principales:

- 1.
- 2.
- 3.

Se eliminaron las puntuaciones y los niveles sustituyéndose por un mensaje que motivara a los usuarios a mantener su curiosidad y seguir programando.

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.

Se modificó por completo la tabla de las puntuaciones, cambiando los nombres de los aspectos al de las categorías y cambiando las barras de progreso a porcentajes, pudiendo obtenerse un 25 %, 50 %, 75 % o 100 %.

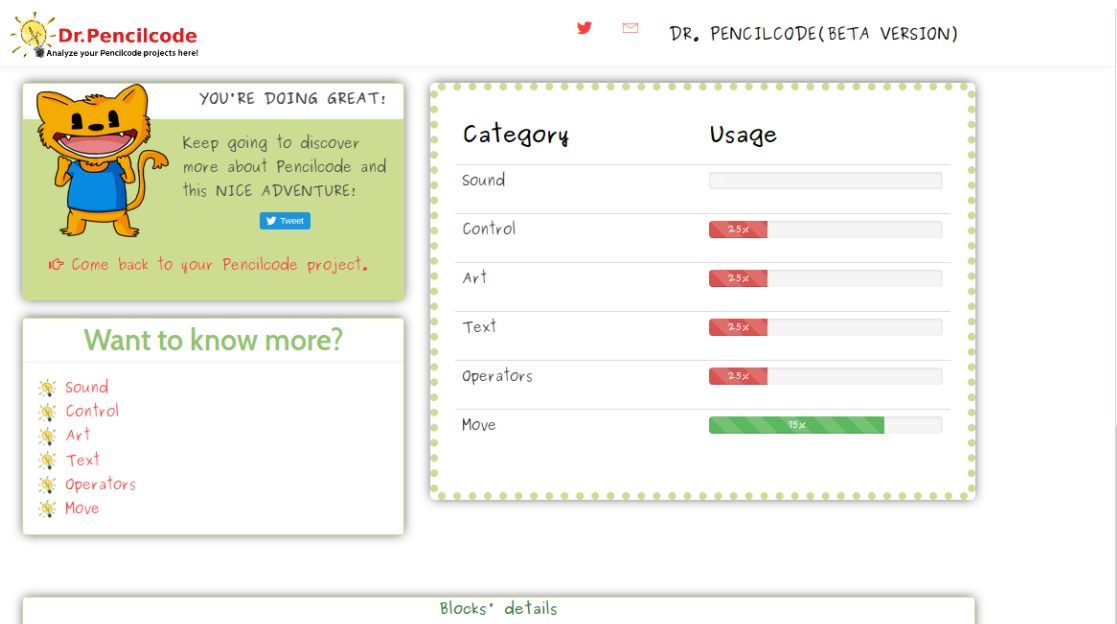


Figura 4.3: Dashboard Fase II

Además, para que los usuario 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 **Not used** asociado a dicha categoría. Para poder ver esta sección los usuarios debía hacer scroll en la pantalla.

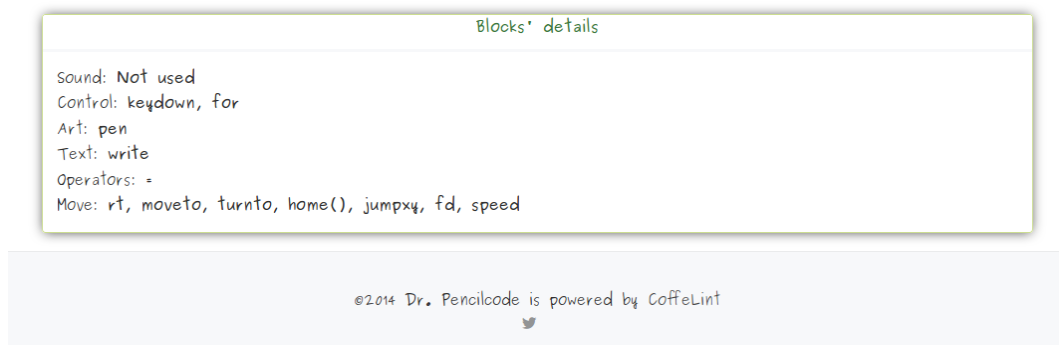


Figura 4.4: Sección Details Fase II

4.6.3. FASE III

- 1.
- 2.
- 3.
- 4.

Se vuelven a añadir niveles y puntuación, pero no puntuación máxima.

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.

Como funcionan estos mensajes? Se centran en las categorías en las que se ha conseguido el nivel más bajo.

Con la introducción de los Bonus se añade una nueva fila a la tabla de puntuaciones en la que estos quedan representados con estrellas. Se rellenaran tantas estrellas como bonus se consigan.

Además, para conocer exactamente los bonus conseguidos se añade a la sección *Details* un desplegable *Bonus* donde se especifica porqué se han obtenido. Se añaden elementos visuales para dotar a la web de mayor inteligibilidad.

Se añaden 2 botones.

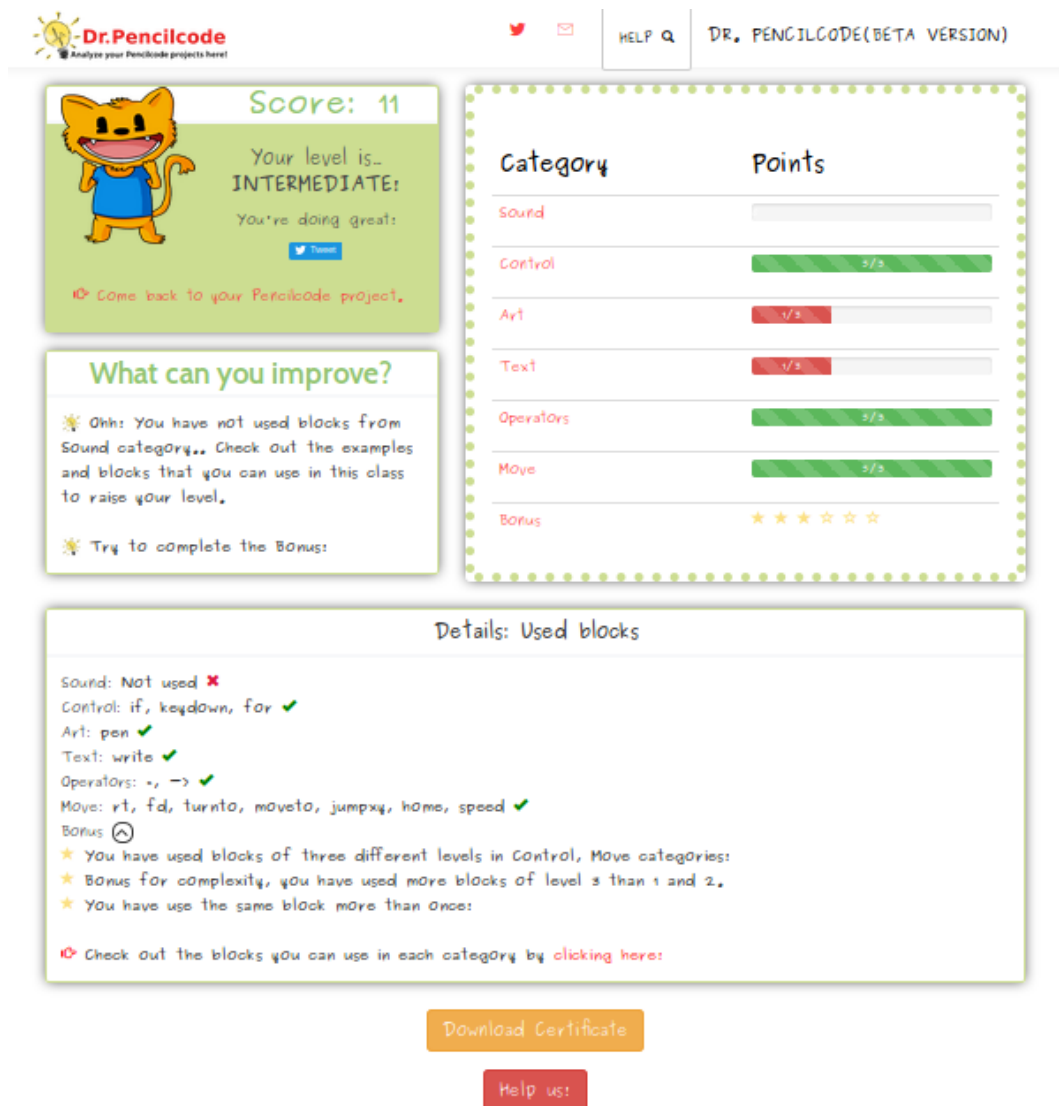


Figura 4.5: Dashboard Fase III

Como el scroll no lo hacen los niños, se decidió comprimir todo un poquito más y hacer la sección Details desplegable, si los niños quieren saber más información lo único que tienen que hacer es pinchar en la flecha de despliegue.

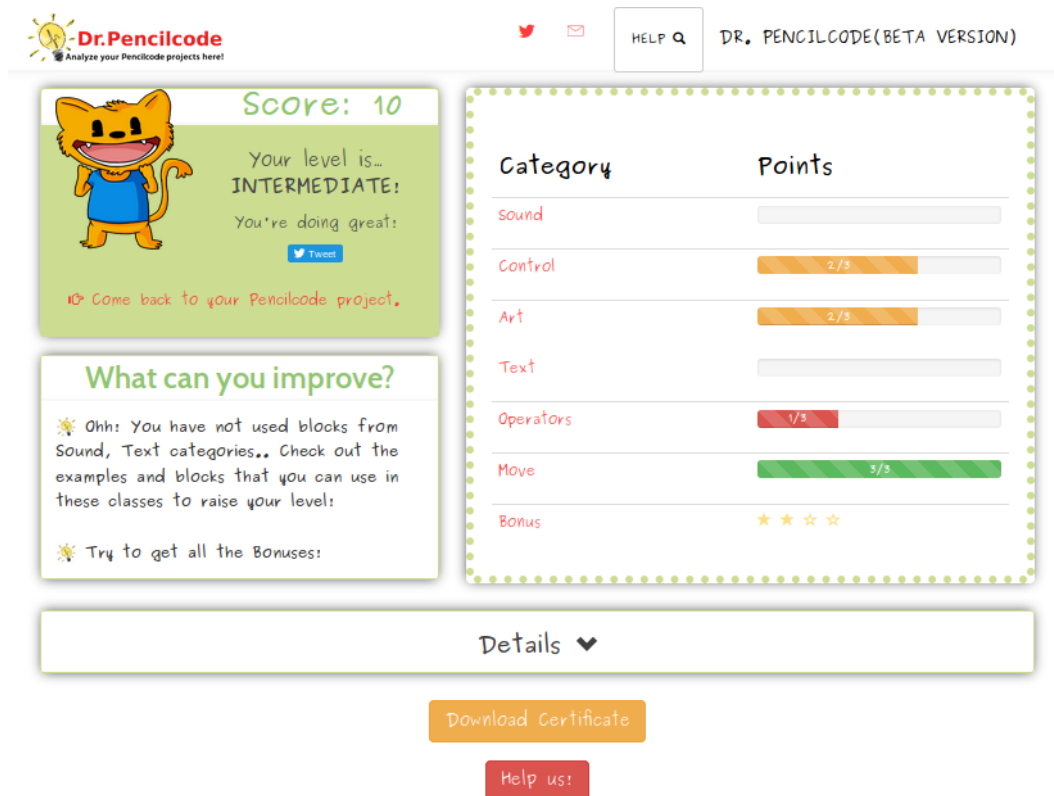


Figura 4.6: Página Final Dashboard

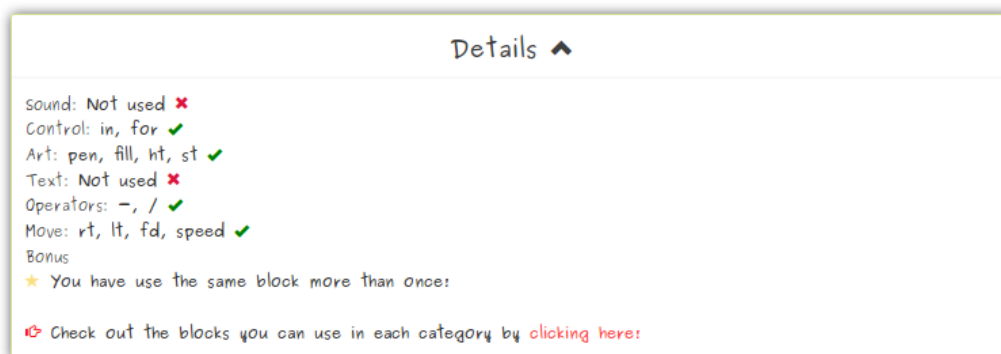


Figura 4.7: Detalles de los resultados

4.7. Análisis de proyectos Pencil Code FASE III

Al igual que Dr.Scratch realiza el análisis de los proyectos 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 como se analizan los

proyectos, desde que se introduce la URL hasta justo antes de mostrar 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*, *sendRequestJSON*, *handler_upload* y *analyzeProject* que son las principales encargadas de realizar todo el proceso de análisis.

4.7.1. 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 Pencil Code o si en la url no se ha introducido ningún valor.

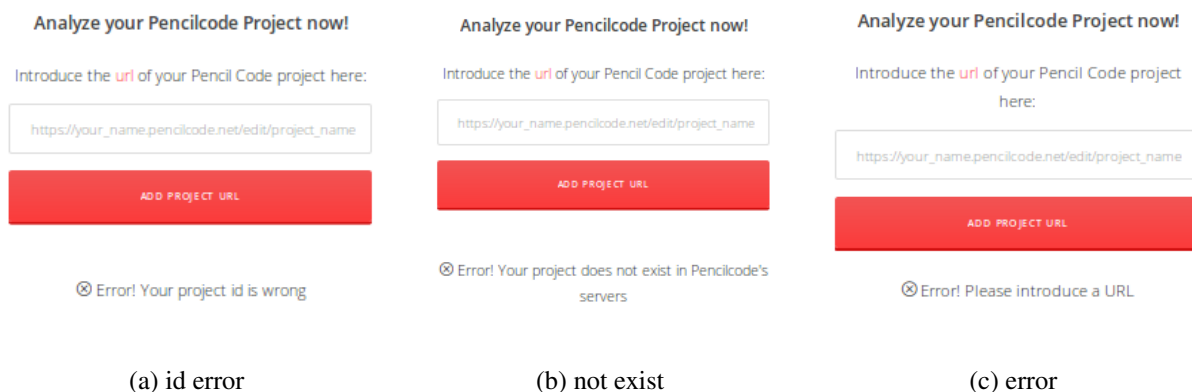


Figura 4.8: Errores al analizar un proyecto.

En Dr.Pencilcode, además de esos tres, se incluye un nuevo error, **code.error**, el cual se invoca cuando el código del programa 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.

Analyze your Pencilcode Project now!

Introduce the **url** of your Pencil Code project here:

`https://your_name.pencilcode.net/edit/project_name`

ADD PROJECT URL

⊗ Error! Your project code is wrong. Check it in Pencil Code

Figura 4.9: Nuevo error de código

4.7.2. Coffee-Mastery

Desarrollar un plugin que analice programas de CoffeeScript para medir el desarrollo del pensamiento computacional de un usuario principiante mediante la evaluación de estructuras de programación y elementos en el código.

El plugin `coffe-mastery.py` es el que, realmente, se encarga de analizar los proyectos de Pencilcode ofreciendo información sobre:

- Todos los bloques que se han utilizado en el programa y los niveles de las categorías a los que pertenecen cada uno de los bloques.
- Los bloques duplicados y el número de veces que ha sido repetido cada uno de ellos.
- El número de puntos obtenidos en cada categoría.

```
python coffee-mastery.py colors.coffee
```

```
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 4.10: Salida de `coffee-mastery.py`

4.8. Páginas de ayuda

Al tratarse de una herramienta educativa, en Dr. Pencilcode, se incluyen nueve páginas de ayuda que dan soporte a los usuarios. Seis de ellas (`Move.html`, `Art.html`, `Text.html`, `Sound.html`, `Control.html`, `Operators.html`) contienen información sobre las diferentes categorías de Pencilcode. Estas páginas son muy útiles cuando el usuario quiere subir su puntuación, incluyen varios ejemplos, con las respectivas explicaciones del programa, de como 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 4.11*

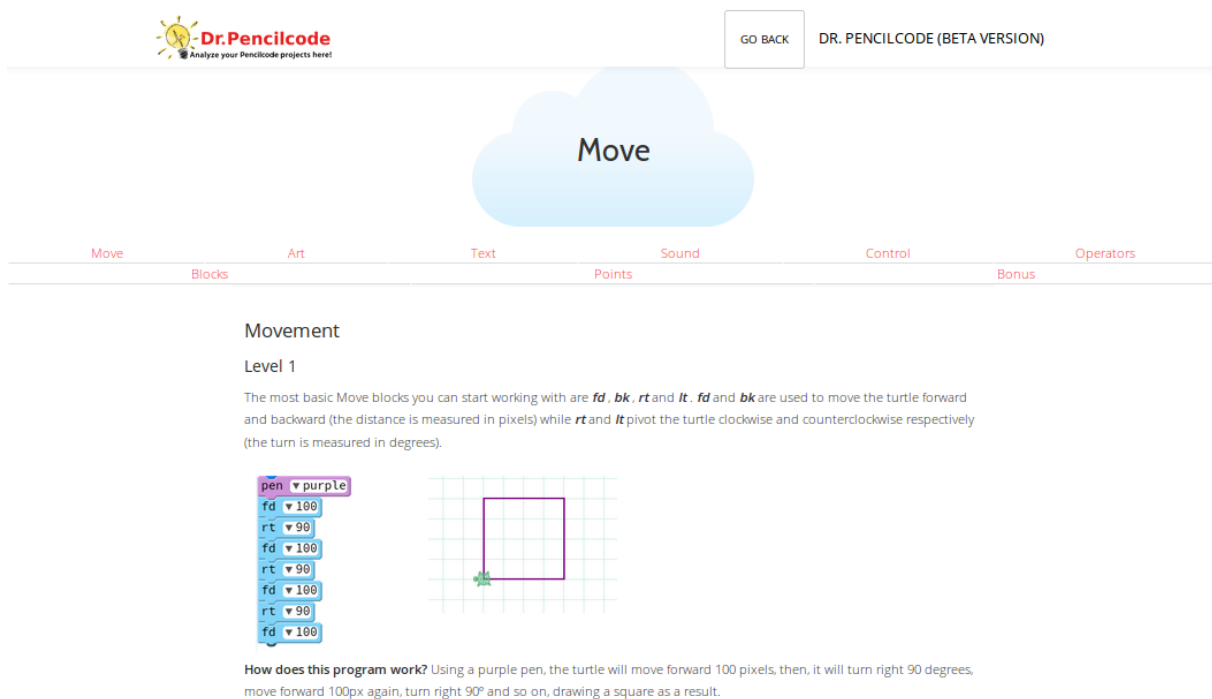


Figura 4.11: Página de ayuda Move

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 la explicación de los bloques existentes por categoría y de como Dr.Pencilcode evalúa o analiza los proyectos, es decir, determina las puntuaciones y los bonus que se asignan a cada proyecto.

En la *Figura 4.12* se muestra, como ejemplo, el diseño de la página `Points.html`:

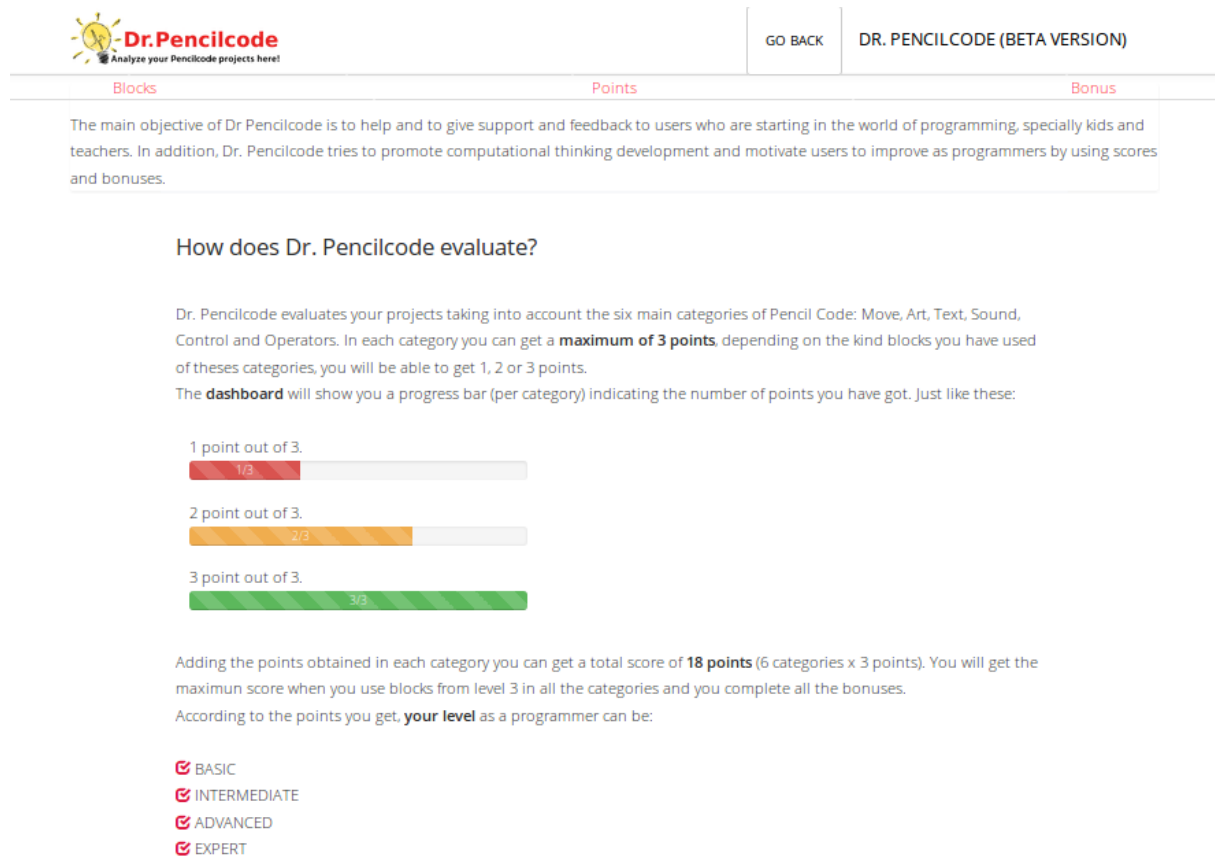


Figura 4.12: Página de ayuda Points

Cabe destacar que el diseño de estas página es totalmente nuevo, exceptuando el documento `base.html` que es el que da estructura principal a dichas páginas, debido a que no coincide con ninguna de las creadas para Dr.Scratch.

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

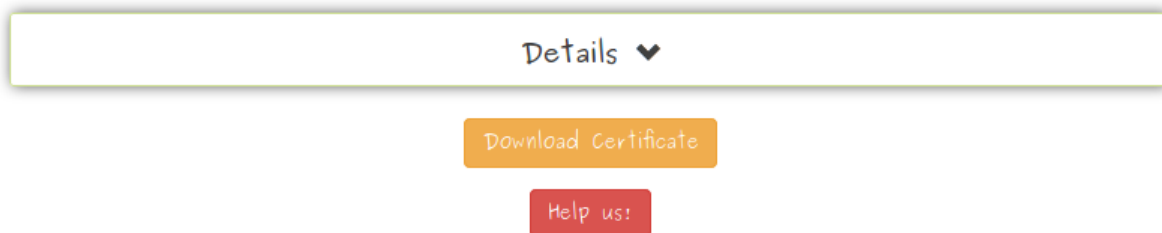
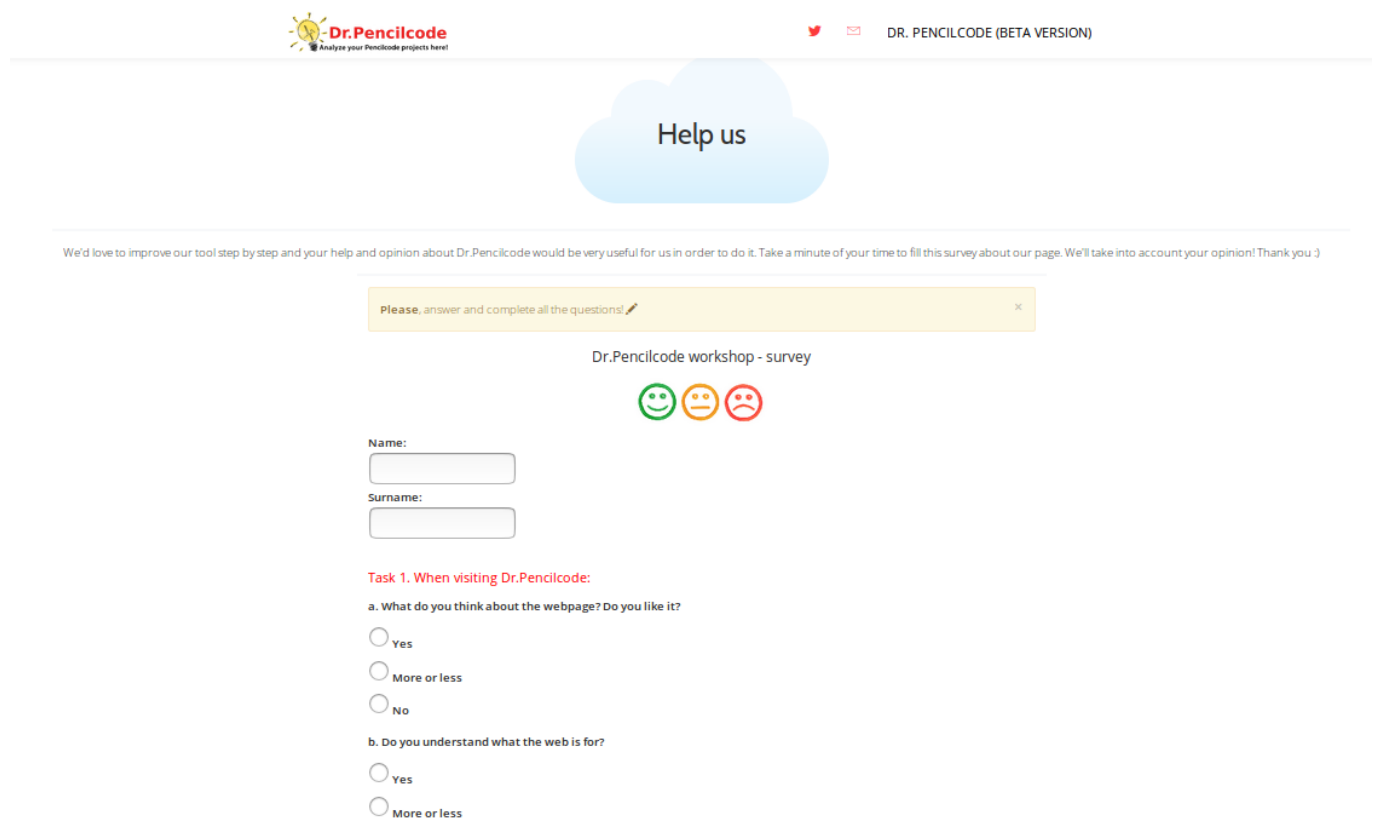


Figura 4.13: Botón Help us

El motivo de abrir una nueva pestaña para el formulario, es el de facilitar a los usuarios contestar a las preguntas a medida que analizan sus proyectos, investigan y navegan por la aplicación web. El formulario se centra en conocer los niveles de usabilidad, interactividad e inteligibilidad de la herramienta dependiendo del usuario. La encuesta incluye preguntas relacionadas con la apariencia de la web, el deseo de seguir mejorando el proyecto una vez analizado y conocido su puntuación, la capacidad para moverse por las diferentes páginas... concluyendo con un apartado de libre opinión. El diseño de la página del formulario es el que se muestra en la *Figura 4.17*



Dr.Pencilcode
Analyze your Pencilcode projects here!

DR. PENCILCODE (BETA VERSION)

Help us

We'd love to improve our tool step by step and your help and opinion about Dr.Pencilcode would be very useful for us in order to do it. Take a minute of your time to fill this survey about our page. We'll take into account your opinion! Thank you :)

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 4.14: Pagina de formulario

Una vez que el usuario pulsa el boton **Submit** se lleva a cabo una peticion HTTP POST con el recurso /helpus. El documento urls.py se encarga de enlazar con la función `getFeedback()` de `views.py`.

En `getFeedback()` se utiliza la función de django `is_valid()`, encargada de comprobar si los campos introducidos en el formulario son válidos. Si alguno de los campos del formulario no está completo o no coinciden con lo indicado en `forms.py`, ésta función devolverá un booleano `False`, además, el html mostrará por pantalla el *mensaje warning*

Si el formulario es válido, obtendremos la informacion del mismo para comprobar que los campos de *name* y *user* no existían previamente en un mismo registro de la base de datos. Si todo es correcto, se mostrará por pantalla el *mensaje success*, en caso contrario, se mostrará el *mensaje danger* indicando el error.

Para poder comprenderlo de un manera más simple, se muestra el algoritmo a continuación:

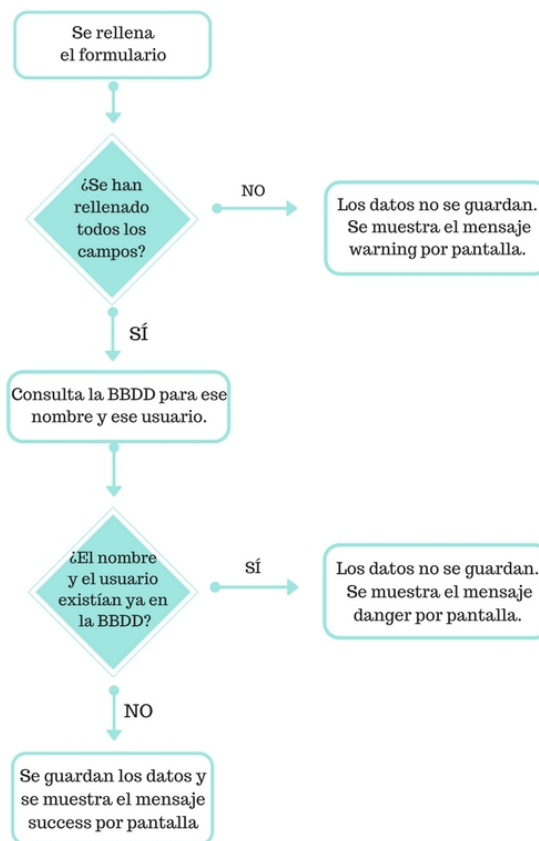


Figura 4.15: Algoritmo del formulario

Estos son los tres mensajes, mencionados anteriormente, que podrían aparecer por pantalla cuando se pulsa el botón **Submit**:

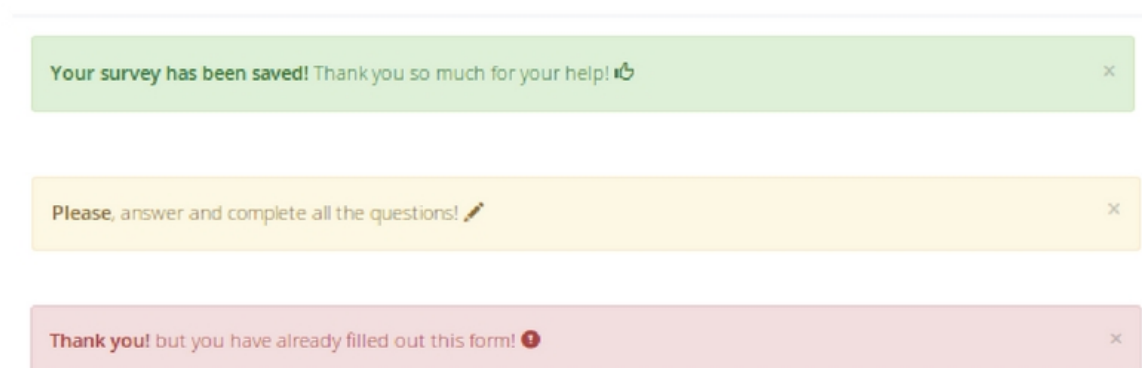


Figura 4.16: Mensajes del formulario

4.10. Certificado

Este apartado fue creado por mis compañeras de Dr. Scratch. Lo único que he hecho ha sido modificar el pdf adaptandolo a Pencil Code cambiar las las variables que se introducen como parámetro al llamar a la funcion `pyploma.py` encargada de generar el certificado.



Figura 4.17: Certificado Dr. Pencilcode

Capítulo 5

Resultados

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

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 como 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 Video 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 a cabo 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. Json

6. Servidor Apache y maquinas virtuales

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 habia 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, organizandola, identificando posible 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 aprendiendo 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. Esto me ha ayudado a ver las cosas desde el punto de vista del docente y no del alumno.

6.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

- Traducción a otros idiomas (español)
- Cuenta de usuarios en el que haya un apartado con el seguimiento de sus proyectos analizados, si ha mejorado
- Mejorar la rúbrica en cuanto a los bonus, tener en cuenta más aspectos del código y combinaciones de los bloques.

6.5. Valoración personal

Finalmente (y de manera opcional), hay gente que se anima a dar su punto de vista sobre el proyecto, lo que ha aprendido, lo que le gustaría haber aprendido, las tecnologías utilizadas y demás.

Apéndice A

Manual de usuario

Bibliografía

- [1] M. J. Kabir. *La Biblia de Servidor Apache 2*. Anaya Multimedia, 2003.
- [2] B. Totty and D. Gourley. *HTTP: The Definitive Guide*. O'Reilly Media, Inc., 2002.