UNIVERSIDAD
REY JUAN CARLOS

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.

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

En la actualidad, las nuevas tecnologías y la programación están ganando terreno en el area de la educación, la LOMCE añade una nueva competencía 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 efficiently the problems of real life 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

Índice de figuras

Capítulo 1

Introducción

En este capítulo se introduce el proyeto. 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.

1.2. Pencil Code

Pencil Code[1] 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.

[1] `https://pencilcode.net/`

Figura 1.1: 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, algorítmos, 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 progrramar 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.

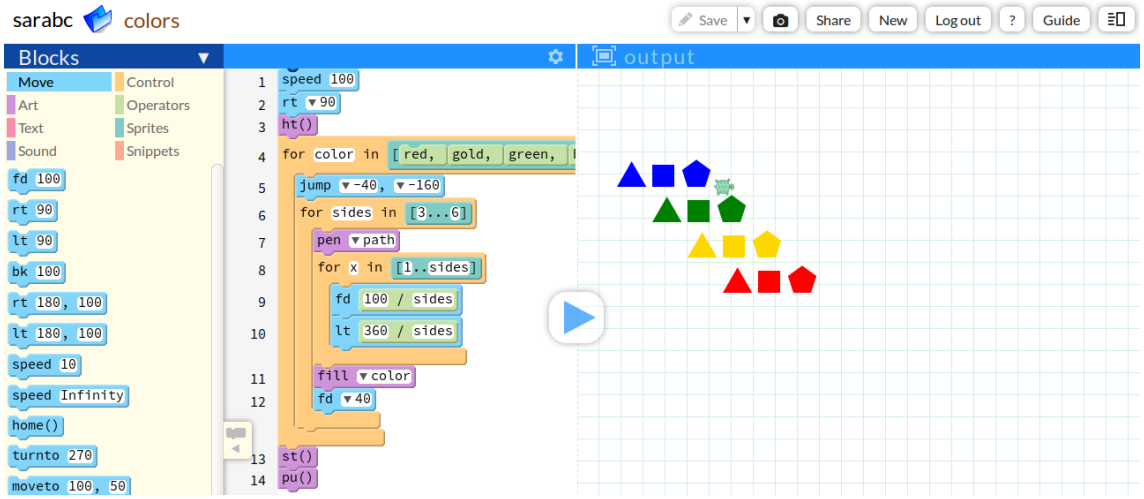

Figura 1.2: Programa creado con Pencil Code.

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 pograma 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.

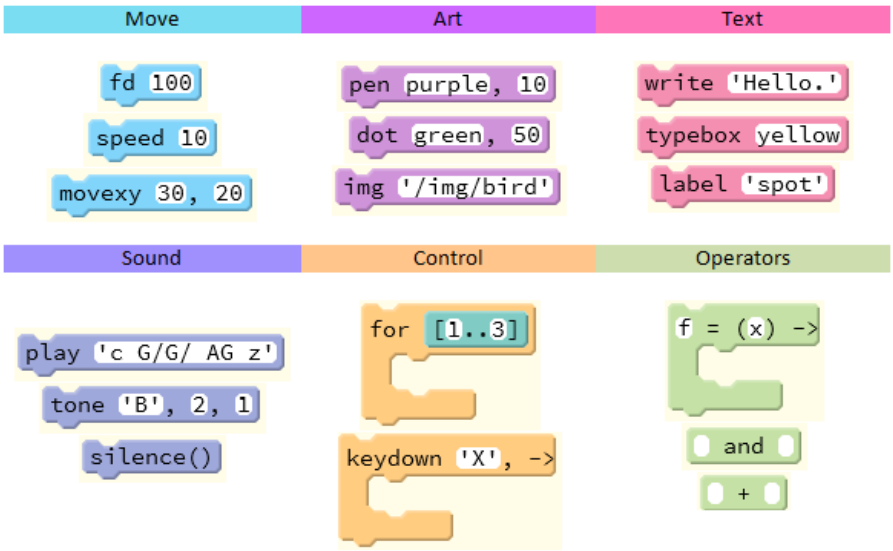

Figura 1.3: Categorías de Pencil Code

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.

1.3. Marco de Referencia

1.3.1. Dr. Pencilcode

1.3.2. 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.

1.4. Estructura de la memoria

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

- **Introducción:**
- **Objetivos:**
- **Estado del arte:** En esta sección se enumeran las tecnologías utilizadas en el desarrollo de este proyecto, tanto en el back-end como en el fron-end, realizando una breve descripción de cada una de ellas.
- **Diseño e implementación:**
- **Resultados:**
- **Conclusiones:**

Capítulo 2

Objetivos

2.1. Objetivo general

Basandonome en Dr.scratch crear Dr.pencilcode.

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 categorias disponibles: Move, Art, Text, Control, Sound, Operators.

3. Añadir **nuevas funcionalidades** y secciones a la página web - orientadas a Pecil 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 retroalimentacion 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 analisis 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 categorias.
- Incluir una sección donde se indiquen los bloques utilizados en cada una de las categorias 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 fórmulario 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 categoria, 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

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 interaccionan 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, interactua de forma directa con los usuarios.

- **Servidor**: funciona como proovedor de servicios, compartiendo sus recursos con los clientes. Éste espera la llegada de las peticiones, las procesa y, posteriormente, envia una respuesta a los clientes.

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

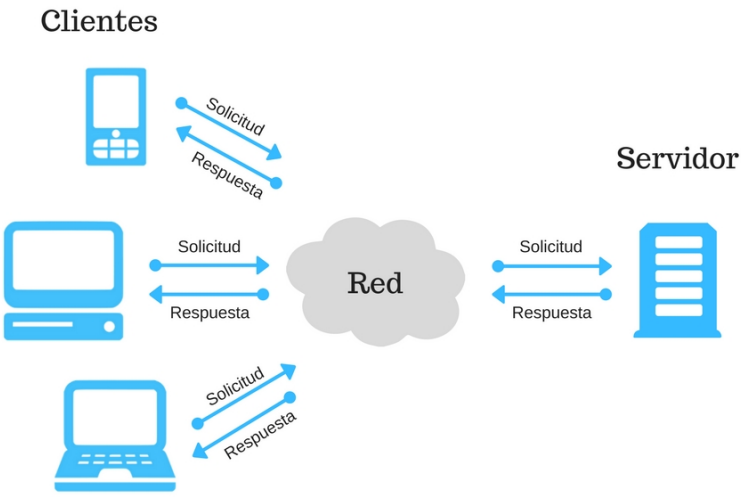

Figura 3.1: Arquitectura Cliente - Servidor

[estructura] las funcionalidades de cada aplicación puedan dividirse en tres niveles: presentación, aplicación y servicio. Según esto, la arquitecturade software de una aplicación distribuida, o una aplicación cliente-servidor, se puede describir de la siguiente manera:

Nivel de presentación. En el lado del servidor, se necesita una interfaz de usua rio. Front-end.

Nivel de lógica de aplicación: En el lado del servidor, necesita obtenerse la hora del dia del sistema y mandarla a la maquina cliente. En el lado del cliente, hay que enviar al servidor la peticions de usuario y mostrarle la respuesta al servidor.

Nivel de servicios. Back-end.

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 partes:

- **Linea 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.

- **Linea 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 linea 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 linea 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 metodos ya predefinidos. Los métodos se encuentran en la linea 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.

Códigos de respuesta como 200 - OK o 404 - Not Found entre otros.

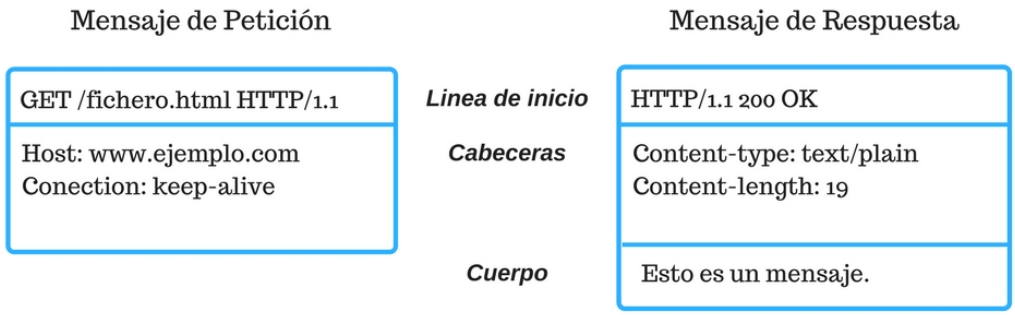

Figura 3.2: Modelo mensajes HTTP

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 estandar más útilizado 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.

```html
<!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

```css
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, conocido comunmente como JS, es un lenguaje de programación interpretado orientado a objetos que se utiliza principalmente para añadir dinamísmo 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 lenguage en nuestra página web.

Entre las numerosas funcionalidades que jQuery ofrece, una de las más importantes es la manipulación del arbol 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 elemetos DOM.

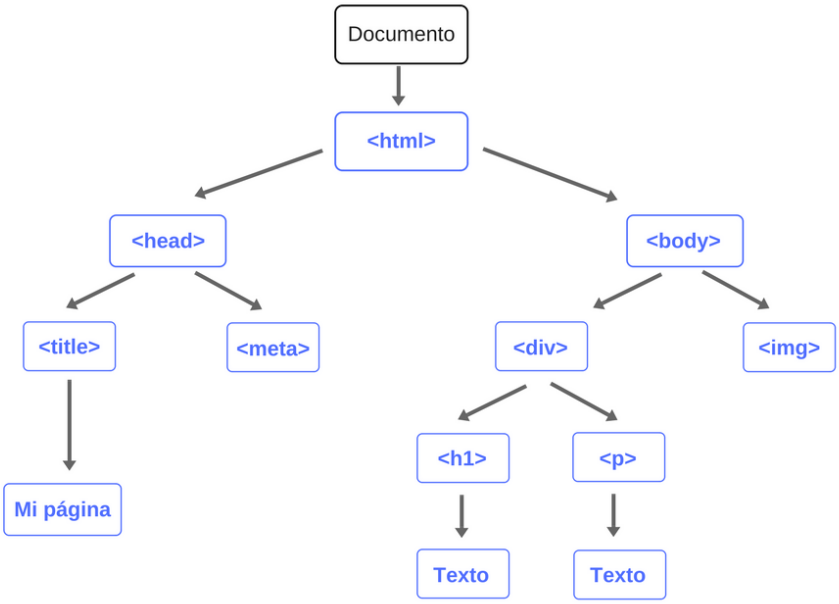

Figura 3.6: Document Object Model

Otras características de esta biblioteca multiplataforma son:

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

3.3.5. Bootstrap

Bootstrap[1] 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:

[1] `http://getbootstrap.com/`

- Se trata de una herramienta de código abierto, disponible en GitHub[2], 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.

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

[2] `https://github.com/twbs/bootstrap`

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 (interación 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 traves 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:

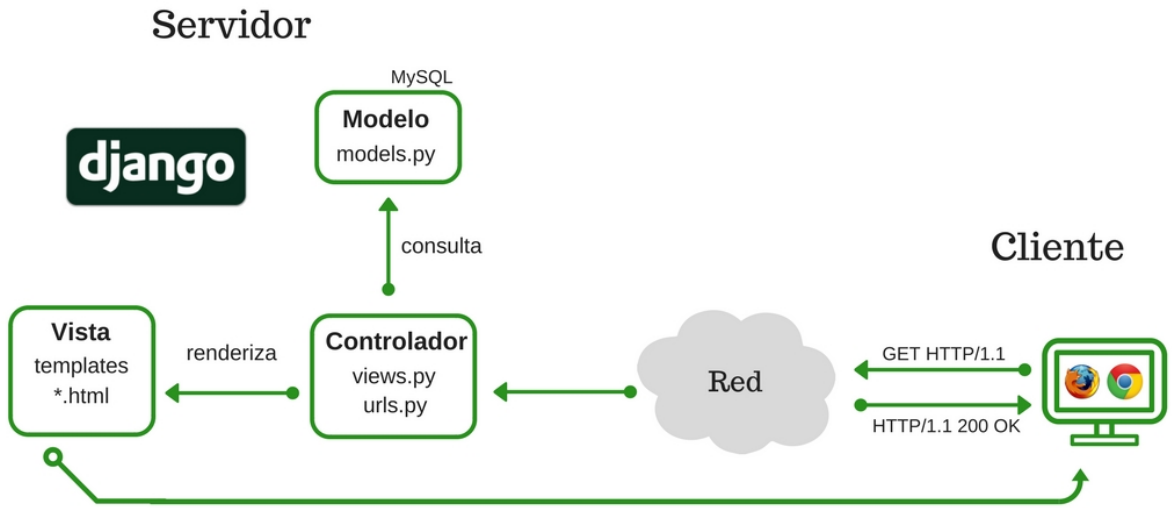

Figura 3.7: Patrón de diseño MVC de Django

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 pogramación 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 caraterísticas 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 lo 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.

- **Librerias**: Hay cantidad de librerias disponibles para ampliar la funcionalidad de Python.

- **Case sensitive**: sensible a mayúsculas y minúsculas.

¿Por qué Python? Está ganando terreno.

3.4.3. Servidor HTTP Apache

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 sitemas 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 gestion de bases de datos más populares y más utilizados en el ámbito mundial, como se puede observar en la *figura 3.7*[3].

[3] `https://db-engines.com/en/ranking`

323 systems in ranking, April 2017

Rank Apr 2017	Rank Mar 2017	Rank Apr 2016	DBMS	Database Model	Score Apr 2017	Score Mar 2017	Score Apr 2016
1.	1.	1.	Oracle	Relational DBMS	1402.00	+2.50	-65.54
2.	2.	2.	MySQL	Relational DBMS	1364.62	-11.46	-5.49
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1204.77	-2.72	+69.72
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	361.77	+4.14	+58.05
5.	5.	↓ 4.	MongoDB	Document store	325.43	-1.51	+12.98
6.	6.	6.	DB2	Relational DBMS	186.66	+1.74	+2.57
7.	7.	7.	Microsoft Access	Relational DBMS	128.18	-4.76	-3.79
8.	8.	8.	Cassandra	Wide column store	126.18	-3.01	-3.49
9.	↑ 10.	9.	Redis	Key-value store	114.36	+1.35	+3.12
10.	↓ 9.	10.	SQLite	Relational DBMS	113.80	-2.39	+5.83
11.	11.	11.	Elasticsearch	Search engine	105.67	-0.56	+23.09
12.	12.	↑ 13.	Teradata	Relational DBMS	76.56	+3.02	+4.30
13.	13.	↓ 12.	SAP Adaptive Server	Relational DBMS	67.46	-2.67	-5.86
14.	14.	14.	Solr	Search engine	64.37	+0.38	-1.65
15.	15.	15.	HBase	Wide column store	58.47	-0.51	+6.98
16.	16.	↑ 17.	FileMaker	Relational DBMS	57.18	+2.61	+11.07
17.	17.	↑ 18.	Splunk	Search engine	55.50	+1.42	+13.15
18.	↑ 19.	↑ 21.	MariaDB	Relational DBMS	48.72	+1.84	+17.14
19.	↓ 18.	19.	SAP HANA	Relational DBMS	48.15	-1.91	+7.80

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

Algunas característica 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 informáticas[4].

[4] `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 abierto[5] 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

By default, CoffeeLint will help ensure you are writing idiomatic CoffeeScript, but every rule is optional and configurable so it can be tuned to fit your preferred coding style

helps enforcing a coding standard

3.5. Google Analytics

[5] https://github.com/clutchski/coffeelint

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

figura 4.1.

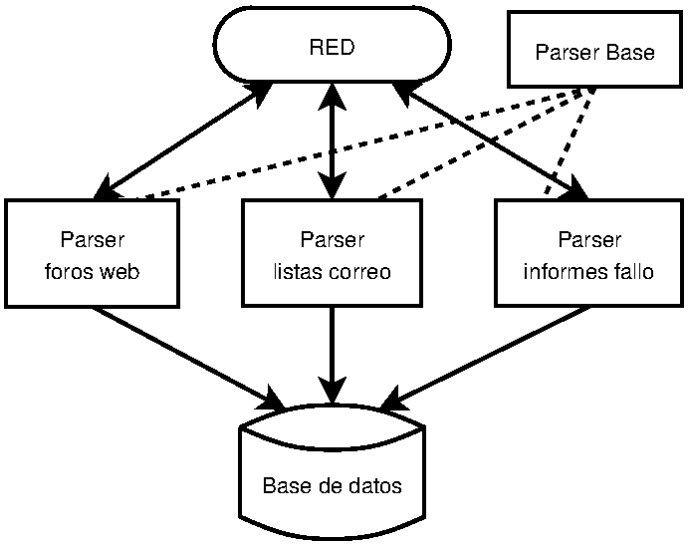

Figura 4.1: Estructura del parser básico

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 desarollo, he aplicado gran parte de lo aprendido en asignaturas como:

1. **Informatica 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. **Informatica 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 Telematicos para Medios Audiovisuales:** siguiendo con los prótocolos, 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 sequir 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. **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 solucciones 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.

Por último, he aprendendido 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.

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