



**DOBLE TITULACIÓN EN INGENIERÍA DE
TELECOMUNICACIÓN Y LICENCIATURA EN
ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS**

Curso Académico 2015/2016

Trabajo Fin de Carrera

**DR. SCRATCH: FOMENTANDO LA
CREATIVIDAD Y VOCACIÓN CIENTÍFICA
CON SCRATCH**

Autor : Mari Luz Aguado Jiménez

Tutor : Dr. Gregorio Robles Martínez

Co-Tutor : Jesús Moreno León

Proyecto Fin de Carrera

**DR. SCRATCH: FOMENTANDO LA CREATIVIDAD
Y VOCACIÓN CIENTÍFICA CON SCRATCH**

Autor : Mari Luz Aguado Jiménez

Tutor : Dr. Gregorio Robles Martínez

Co-Tutor : Jesús Moreno León

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

*Dedicado a
mi familia, pareja y amigos.*

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

Dr. Scratch es una plataforma web de software libre que permite analizar proyectos realizados en *Scratch* -lenguaje orientado a la enseñanza- aportando feedback sobre determinados aspectos relacionados con el *Pensamiento Computacional*. El **objetivo** del proyecto es dar soporte tanto a maestros como estudiantes, en sus primeros pasos en la programación con Scratch.

Este **proyecto** es el resultado del trabajo de cinco personas, no sólo el mío. Por lo que iré indicando durante toda la memoria cuál ha sido mi aportación de la forma más concreta posible. Destacar, que es un proyecto amplio, ya que ha sido desarrollado durante algo más de un año y aún está en proceso.

Las **tecnologías** utilizadas han sido diversas debido a que es un proyecto en producción, hay miles de proyectos analizados por numerosos colegios y organizaciones. Para subir a producción hemos utilizado una solución cloud de Microsoft llamada Azure y la licencia Apache. Por debajo de ello, tenemos un servidor desarrollado en Python 2.7 con la ayuda del framework Django 1.8 que nos ha facilitado multitud de tareas de la parte back-end. Además, para el registro de usuarios, organizaciones y guardar toda la información de los proyectos analizados tenemos una base de datos en MySQL. Pero el front-end es lo que más trabajo nos ha supuesto ya que hemos tratado de adaptar la plataforma a las necesidades observadas en docentes y alumnos en varias jornadas que hemos organizado a lo largo del año. De este modo, HTML, CSS con la ayuda de Bootstrap, Javascript y AJAX han sido utilizados diariamente.

Actualmente, se está introduciendo la programación en las aulas de numerosos colegios de nuestro país. Pero existe la **problemática** de falta de formación en esta materia de la mayoría de maestros de educación obligatoria. El proyecto trata de suplir esta deficiencia y guiar a dicho colectivo en sus primeros pasos. Además, sus dashboards han sido diseñados de un modo tan sencillo, que la idea es que un alumno con un nivel de comprensión lectora media, sea capaz de aprender por sí mismo utilizando la herramienta.

Índice general

Lista de figuras	IX
Lista de tablas	1
1. Introducción	1
1.1. Marco general	1
1.2. Programación con Scratch	3
1.3. Marco de referencia	5
1.4. Estructura de este documento	7
2. Objetivos	9
2.1. Objetivo general	9
2.2. Objetivos específicos	11
2.3. Planificación temporal	14
3. Estado del arte	19
3.1. Aplicaciones cliente-servidor	19
3.2. Back-end	22
3.2.1. Python	22
3.2.2. Django	24
3.2.3. MySQL	25
3.2.4. Servidor HTTP Apache	27
3.2.5. Servidor HTTP Node.js	27
3.3. Front-end	27
3.3.1. HTML	28

3.3.2. CSS	29
3.3.3. Bootstrap	29
3.3.4. JavaScript	29
3.3.5. AJAX	30
3.4. Plataforma en la nube: Azure	30
3.5. Analítica web: Google Analytics	30
4. Diseño e implementación	31
4.1. Arquitectura general de Dr. Scratch	31
4.1.1. Capa de servicio: back-end	32
4.1.2. Capa de presentación: front-end	38
4.2. Funcionalidades y evolución de Dr. Scratch	39
4.2.1. Versión Alpha	39
4.2.2. Fase I	40
4.2.3. Fase II	51
4.2.4. Fase III	70
4.2.5. Diagramas de flujo	83
5. Resultados	91
6. Conclusiones	93
6.1. Consecución de objetivos	93
6.2. Aplicación de lo aprendido	94
6.3. Lecciones aprendidas	94
6.4. Trabajos futuros	94
6.5. Valoración personal	95
A. Manual de usuario	97
Bibliografía	99

Índice de figuras

1.1.	Logo de Scratch	3
1.2.	Bloques en Scratch	4
1.3.	Estadísticas de la comunidad de Scratch	4
1.4.	Apariencia de la versión Alpha de Dr. Scratch	6
2.1.	Fase I del proyecto Dr. Scratch	15
2.2.	Fase II del proyecto Dr. Scratch	16
2.3.	Fase III del proyecto Dr. Scratch	17
3.1.	Esquema del paradigma Cliente-Servidor	20
3.2.	Capas presentes en toda aplicación distribuida	20
3.3.	Mensajes de petición y respuesta del protocolo HTTP	21
3.4.	Lenguajes de programación más utilizados en 2015	23
3.5.	MVC adaptado de Django	24
3.6.	Árbol de directorios de un proyecto en Django	25
3.7.	Ranking de las bases de datos más utilizadas actualmente	26
3.8.	Tecnologías más utilizadas en la capa de presentación	28
3.9.	Estructura básica de un código HTML	28
4.1.	Arquitectura general de Dr. Scratch	32
4.2.	Modelo de Django en Dr. Scratch	33
4.3.	Funcionamiento de la petición de recursos a Dr. Scratch: <code>urls.py</code>	33
4.4.	Distinción según el método de la petición: <code>views.py</code>	34
4.5.	Modularización del fichero <code>views.py</code>	34
4.6.	Declaración de los objetos de Dr. Scratch: <code>models.py</code>	35

4.7. Captura de pantalla de la base de datos	36
4.8. Modelos de Dr. Scratch: <code>models.py</code>	37
4.9. Estructura de las páginas de Dr. Scratch: <code>templates</code>	38
4.10. Apariencia de la versión Alpha de Dr. Scratch	40
4.11. Fase I del proyecto Dr. Scratch	40
4.12. Salidas de los plugins de Hairball	41
4.13. Primer dashboard de Dr. Scratch	42
4.14. Primera apariencia de la página principal con logo	45
4.15. Dashboard con misma gama de colores que el logo	45
4.16. Página principal de Dr. Scratch	46
4.17. ¿Por qué te encantará Dr. Scratch?	47
4.18. ¿Cómo funciona Dr. Scratch?	47
4.19. Nuevas funcionalidades de Dr. Scratch.	48
4.20. Contacta con el equipo de Dr. Scratch.	49
4.21. Configuración para la traducción de Dr. Scratch.	50
4.22. Fase II del proyecto Dr. Scratch	51
4.23. Barra de progreso mostrada durante el análisis	52
4.24. Cuadro con las dos posibilidades de análisis	53
4.25. Gestión de errores en el cuadro de análisis	54
4.26. Página mostrada al producirse un error en el análisis	55
4.27. Página de error 404	56
4.28. Página de error 500	56
4.29. Dashboard de Pensamiento Computacional bajo	59
4.30. Dashboard de Pensamiento Computacional medio	59
4.31. Dashboard de Pensamiento Computacional alto	60
4.32. Aclaración de los métodos de análisis	62
4.33. Ventana modal de qué es una URL	62
4.34. Ventana modal de cómo descargar un proyecto en Scratch	63
4.35. Panel de Google Analytics para Dr. Scratch	65
4.36. Dashboards actuales de Dr. Scratch	66
4.37. Página principal para organizaciones en Dr. Scratch	67

4.38. Formulario de registro para organizaciones	67
4.39. Dashboard de organizaciones	68
4.40. Ventana modal que explica cómo debe ser el fichero CSV	69
4.41. Fichero CSV con toda la información del análisis	69
4.42. Fase III del proyecto Dr. Scratch	70
4.43. Traducción fallida de los bloques de ScratchBlocks	71
4.44. Traducción de los bloques de ScratchBlocks mediante AJAX	72
4.45. Árbol de usuarios de Dr. Scratch	73
4.46. Traducción dinámica de la web: bola del mundo	74
4.47. Botón de la extensión	75
4.48. Extensión de Dr. Scratch para Chrome	75
4.49. Modificación en settings.py para traducir a más idiomas	76
4.50. Agradecimiento a todos nuestros colaboradores	77
4.51. Quiénes somos	77
4.52. Colaboradores que han participado en la traducción de Dr. Scratch	78
4.53. Colegios colaboradores	78
4.54. Acceso a las páginas de estadísticas, colaboradores y al blog	79
4.55. Salida antigua del plugin Dead Code	80
4.56. Salida actual del plugin Dead Code	80
4.57. Formulario para el registro de usuarios	82
4.58. Dashboard de programadores	83
4.59. Diagrama de flujo del análisis _upload	84
4.60. Lógica de la función show_dashboard()	85
4.61. Lógica de la función analyze_project()	86
4.62. Lógica de la función _url()	87
4.63. Diagrama de flujo del análisis _CSV	88

Capítulo 1

Introducción

La forma más eficaz de interesar a una persona por un tema, es hacer que se **involucre**. Y las formas de conseguir ese compromiso, dependen en gran medida, de nuestras aptitudes personales, tales como la capacidad de comunicación, creatividad, innovación y liderazgo. *Pero hoy en día tenemos un gran aliado, las nuevas tecnologías.*

En este capítulo, se trata la situación actual de la educación y cómo aprender a programar en *Scratch*, puede ayudar a desarrollar aptitudes relacionadas con el *Pensamiento Computacional*.

1.1. Marco general

Uno de los sectores más gratificantes en los que se puede trabajar, es la **educación**. Pero en sí mismo, es a la vez, todo un reto. Por descontado, decir, que no todo estudiante es igual, pero además, los estudiantes cambian junto con la *evolución de la sociedad* y hoy en día la sociedad está cambiando a un ritmo desproporcionado, ligada al *desarrollo de la tecnología*. Ya no hacemos -casi- nada como hace veinte años, y por tanto, tampoco aprendemos de la misma forma.

Como siempre, en la evolución del ser humano, lo único que está seguro es el **cambio**, y es nuestra capacidad de adaptación a dichos cambios, lo que nos hace tener éxito, o no, en los retos que nos proponemos. La educación no iba a ser diferente y en los últimos años, los currículos españoles han sufrido inmensas modificaciones con el *proceso de Bolonia*. Ahora toca ponerse manos a la obra, para tratar de adaptarnos a lo que la sociedad está demandando, para poder seguir dando una educación de calidad. Es el momento de incorporar *nuevas herramientas* y

metodologías, que permitan desarrollar las habilidades que el mercado laboral está demandando.

En unos años, la mayoría de las ofertas de trabajo serán para puestos cualificados [3], en los que se demandará -prácticamente en todos- tener nociones de **programación**. La programación, esa gran desconocida, que muchas veces tiene connotaciones negativas, sobre todo, entre la sección femenina.

Pero el mayor problema, es el citado. *No se conoce*. No se ha tratado de involucrar a la sociedad en dicha materia, hasta ahora que el día a día ha empezado a exigirlo. Todo funciona a través de internet, si no tienes tu comercio en la red estas anticuado, así que necesitas aprender sobre e-commerce. Y necesitas una web en la que se muestren tus productos, la gente no tiene tanto tiempo para desplazarse, o si lo tiene, ¿para qué ir a tu tienda si puede ver los productos de otro desde su casa? Necesitas alguien que te programe una página web propia.

Y este es el caso más sencillo, ya que la programación está tomando también un gran protagonismo en muchísimos otros sectores, tales como la industria, muchísimas fábricas poseen robots programados. Así pues, se está convirtiendo en una *tarea obligatoria a la hora de entrar en el mercado laboral*.

Pero, **¿y sí empezamos a enseñar a programar desde la infancia?** Numerosos estudios están demostrando que aprendiendo a programar se desarrolla el *Pensamiento Computacional*, el cual consta de ciertas habilidades, tales como la abstracción o lógica. Estas, no únicamente son útiles para aprender a programar, sino *en muchísimas de las demás asignaturas del currículo pueden servir para involucrar al alumnado [1]* en un proyecto que puede tocar y modificar a su gusto a través de la programación. *Puede aprender mientras experimenta*.

Un ejemplo claro es el de la abstracción, que se ve directamente relacionada con las matemáticas, si se enseña a dividir un problema grande en otros más pequeños, es más fácil reconocer cada parte del problema. Así, el alumno sabrá identificar cuándo tiene que sumar o cuándo multiplicar, cuando tiene un problema con varias operaciones.

Hace años, aprender a programar era una tarea compleja, la información para empezar a hacer tu propio programa, sólo se podía encontrar en libros complicadísimos de entender. Y los lenguajes de programación que existían no eran los más intuitivos. Hoy en día, no hay excusas.

En la siguiente subsección, se va a presentar un lenguaje de programación muy apropiado para iniciarse en el mundo de la programación, de un modo simple, motivador y divertido.

1.2. Programación con Scratch

Scratch¹ es un lenguaje de programación orientado a la enseñanza. Ha sido diseñado por el *Grupo Lifelong Kindergarten del Laboratorio de Medios del MIT*, y dirigido por *Mitchel Resnick* [8]. Su principal **objetivo**, es ayudar a los más jóvenes a pensar de forma creativa, razonar sistemáticamente, y trabajar de forma colaborativa. Aptitudes tan necesarias para la vida en el siglo XXI.

La mayor **fortaleza** de Scratch, es permitir crear historias, juegos y animaciones *sin tener conocimientos previos de programación*. La forma de programar en este lenguaje es uniendo piezas estilo *LEGO*.



Figura 1.1: Logo de Scratch

No es necesario escribir código, con su respectiva sintaxis y todo lo que ello conlleva. En lugar de utilizar tiempo en aprender los detalles específicos que tiene cada lenguaje de programación, Scratch tiene una serie de **bloques** agrupados por categorías -cada una con un color, para que sean claramente identificables- dentro de la cual tenemos las piezas del puzzle que arrastraremos para ir indicándole a nuestro videojuego o historia, cuáles son las instrucciones a seguir. En la *Figura 1.2* podemos observar cuáles son los principales bloques que pueden ser utilizados en Scratch y apreciar todo su potencial.

¹<https://scratch.mit.edu/>

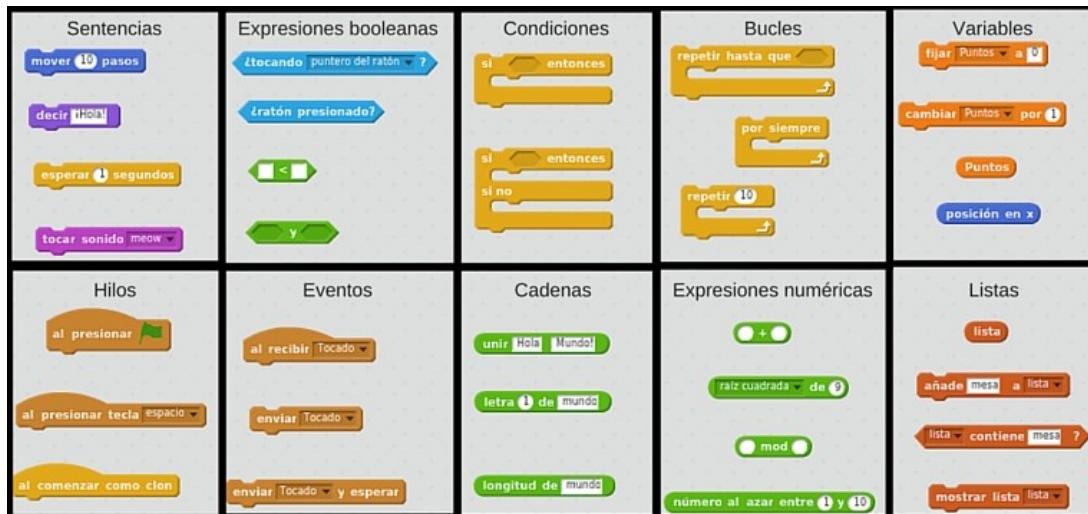


Figura 1.2: Bloques en Scratch

Pero no se queda ahí, Scratch, es además, una **comunidad** donde se pueden *compartir* los proyectos realizados y se pueden exponer dudas en su *foro*. De esta forma, se puede aprender de forma dinámica con la ayuda de otros programadores y observando lo realizado en otros proyectos. También tenemos la opción de *reinventar* otros proyectos, es decir, coger un proyecto realizado por otro Scratcher y modificarlo a nuestro gusto.



Figura 1.3: Estadísticas de la comunidad de Scratch

En la *Figura 1.3* se muestran las estadísticas de la comunidad de Scratch en enero de 2016. Este lenguaje de programación es *utilizado en más de 150 países y en los más diversos entornos* -museos, bibliotecas, escuelas o universidades son algunos- principalmente, por jóvenes entre los 8 y 16 años. El único requisito para utilizarlo es saber leer.

Actualmente la versión es Scratch 2.0 pero hubo una versión anterior (1.4) que aún está siendo utilizada, sobre todo en su **versión offline**. Porque pensando en las dificultades de conexión

que se tiene en algunos centros, Scratch posee una versión descargable que puede ser utilizada sin conexión a la red.

Scratch se ha convertido en un movimiento mundial con su propio día -Scratch Day- y una conferencia que se realiza cada verano -Scratch Conference- en la que el equipo de Dr. Scratch participó en 2015. Se organizó en Ámsterdam, donde pudimos observar las tendencias e ideas mundiales desarrolladas para potenciar las habilidades de los jóvenes programadores que utilizan Scratch.

1.3. Marco de referencia

Dr. Scratch² surge de la observación de mi tutor en este proyecto -Dr. Gregorio Robles- y del doctorando -Jesús Moreno- de la necesidad de una herramienta para Scratch que otorgue feedback sobre los diferentes aspectos que engloban el *Pensamiento Computacional*.

Tras años dedicados a la enseñanza de programación y ciencias de la computación, llegaron a la siguiente conclusión; era necesario tener una herramienta similar a Pylint -herramienta que permite *detectar malas prácticas en el código del lenguaje de programación Python*- para el nuevo lenguaje de programación Scratch.

Para poder analizar un proyecto programado con Scratch necesitamos inspeccionar su archivo. Los archivos generados en Scratch para cada proyecto, tienen un formato comprimido concreto con extensión .sb2 cuyo contenido está conformado por las distintas imágenes y sonidos utilizados en el proyecto y un *fichero JSON que contiene la información relevante* a los bloques de Scratch que se han utilizado. Es decir, para evaluar cómo de bien se ha programado un proyecto, lo que nos interesa es conocer dicho JSON y analizar su contenido.

La tarea de analizar toda la información otorgada por el fichero JSON no es fácil, pero mis tutores encontraron en Github³ un plugin que ya hacía dicha tarea. Este plugin es **Hairball**⁴ que ha sido implementado por Bryce Boe⁵ y es un analizador estático de proyectos programados en Scratch.

Mis tutores partiendo de la base de Hairball, programaron el plug-in **Mastery** que ofrece

²<http://www.drscratch.org/>

³<https://github.com/>

⁴<https://github.com/ucsb-cs-education/hairball>

⁵<https://github.com/bboe>

información de **aspectos del Pensamiento Computacional**: *pensamiento lógico, control de flujo, abstracción, paralelismo, interactividad con el usuario, representación de la información y sincronización.*

Además, Hairball, devuelve mucha más información entre la que se encuentran dos malos hábitos de programación: **código muerto y atributos no inicializados**.

Mis tutores incluyeron otros dos plugins a *Hairball* que inspeccionaban el fichero JSON buscando indicios de otros dos de los **malos hábitos que todo programador novato tiene al principio**: *programas duplicados y nombres inadecuados*. Ambos han realizado algunos papers [6] en los que explican de forma detallada cómo trabajan estos plugins y cuál es la relación que tienen los distintos aspectos del Pensamiento Computacional, además de demostrar cómo el desarrollo del Pensamiento Computacional ayuda a desarrollar habilidades que son compartidas con materias recogidas en los currículos del proceso Bolonia, por lo que aprender a programar puede ayudar a obtener mejores resultados en la enseñanza de asignaturas como Inglés, Matemáticas o Filosofía.

Dr. Scratch es un proyecto aún en desarrollo y ha tenido dos versiones hasta el momento. A continuación se muestran las dos versiones existentes de Dr. Scratch, tratando de explicar cuáles han sido las aportaciones de la versión anterior, para tener una visión del punto de partida con el que se contaba.

- La *versión Alpha* fué creada por el alumno Cristian Chusing, su principal aportación fue la creación de una página web inicial donde se pudiera subir un archivo con extensión .sb2 y mostrarse un dashboard con toda la información obtenida con los pluguins de Hairball. La apariencia de dicha web es la mostrada en la *Figura 4.10*.

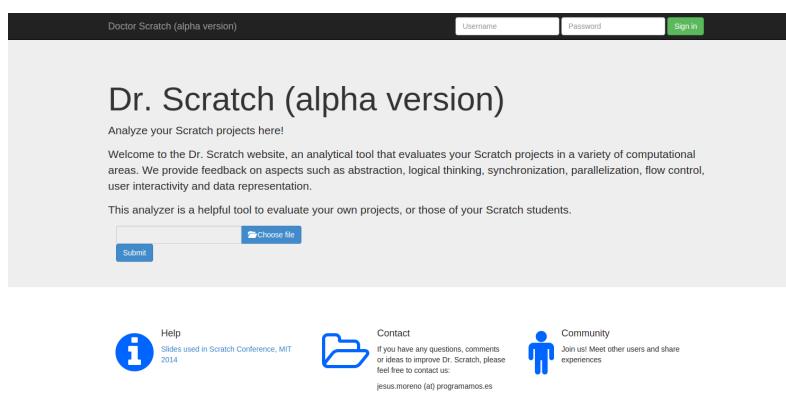


Figura 1.4: Apariencia de la versión Alpha de Dr. Scratch

- La *versión Beta* es la que he estado programando desde noviembre de 2014 con la ayuda de mi compañera Eva Hu Garres desde febrero de 2015. Ha sido realizado durante el periodo en el que fuimos contratadas a media jornada en la Universidad Rey Juan Carlos como técnicos de apoyo y difusión. Todo este proyecto se ha podido llevar a cabo gracias a la financiación otorgada por el FECyT (Fundación Española para la Ciencia y la Tecnología), además, de ser galardonado con un Google RISE Award en 2015 y del apoyo de la asociación sin ánimo de lucro Programamos. Esta versión es la que se va a detallar en este documento.

Al ser una plataforma en producción, se han realizado distintas pruebas en colegios, para comprobar el uso que realizaban los alumnos de ella, permitiéndonos observar cuáles eran las necesidades que demandaban. A lo largo de esta memoria se irán comentando los cambios introducidos en la herramienta y las razones que los han motivado.

1.4. Estructura de este documento

Intentando facilitar la lectura de esta memoria, se procede a explicar su estructura, detallando qué contenido tendrá cada capítulo:

- 1. Introducción.** Explicación del marco social en el que se enmarca este proyecto, así como el marco de referencia en el que se ha desarrollado. Se muestra cuál ha sido la motivación para su realización y las fases iniciales del mismo.
- 2. Objetivos.** Aquí se muestra cuáles han sido las metas a obtener para poder dar respuesta a la necesidad observada. Además, se indica cuáles han sido las funcionalidades a implementar. Éstas últimas, se mostrarán también, en una línea cronológica en las que se dividirá el tiempo en corto plazo, medio y largo, mediante un planteamiento temporal.
- 3. Estado del arte.** En este capítulo, se hablará de las distintas tecnologías utilizadas para poder llevar a cabo la consecución del proyecto y se dará una explicación breve de las mismas.
- 4. Diseño e implementación.** Tras conocer las tecnologías, entraremos a ver cómo las hemos utilizado para diseñar nuestra plataforma. Se explicará detalladamente cuál es la arquitectura de Dr. Scratch y todas sus funcionalidades.

- 5. Interfaz de usuario.** Repaso rápido de la web y cuáles han sido sus modificaciones. Se indicará cuáles han sido las razones para los cambios en la mismas, mostrando así la evolución de la web según las necesidades del usuario recalmando las diferencias entre el prototipo inicial y el que está ahora mismo en producción.
- 6. Resultados.** Al ser un proyecto en producción, conviene observar cuáles han sido los resultados obtenidos: número de proyectos analizados, número de organizaciones registradas e impacto en general.
- 7. Conclusiones.** Capítulo en el que se realiza una reflexión general de los resultados obtenidos y cuál ha sido la consecución de los objetivos marcados inicialmente. Se indicará lo aprendido durante la formación en la Universidad Rey Juan Carlos y que ha sido aplicado en la realización de este Proyecto Fin de Carrera. Así mismo, también se presentarán cuáles son los conocimientos adquiridos durante la realización del proyecto Dr. Scratch y cuáles serían los siguientes pasos que se podrían dar en el mismo. Y para terminar, expresar mi valoración personal de todo lo que me ha aportado Dr. Scratch.

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 qué es lo que queremos conseguir en cada etapa. En este capítulo se describen dichos objetivos iniciales, con la intención de poder reflexionar más tarde observando los resultados, cuáles han sido alcanzados y estudiar cuáles han sido los problemas o motivos que han repercutido en que no se logre alguno de ellos. Además, se indica el planteamiento temporal que se eligió para lograr alcanzar su consecución.

2.1. Objetivo general

El **objetivo** de este proyecto es el diseño y desarrollo de una herramienta web donde poder analizar proyectos programados en el lenguaje Scratch. El análisis nos reportará información de aspectos relacionados con el *Pensamiento Computacional* que debemos mejorar.

Para entender dicho objetivo, es necesario comprender qué es el pensamiento computacional y qué aspectos son los que queremos cubrir y cómo.

Cuando una persona empieza en el mundo de la programación, necesita desarrollar ciertas habilidades y cambiar en cierta medida el modo de plantear los problemas a los que se enfrenta. Necesita comenzar a pensar como lo haría un científico, lo que conlleva abordar los problemas de un modo diferente. Para ello, necesita desarrollar unas habilidades específicas que pueden ser muy útiles en otras disciplinas. Estas habilidades son las que se deben trabajar para conseguir un desarrollo del pensamiento computacional.

Jeannette Wing¹ en su artículo *Computational thinking* [10] indicó que el pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática.

La finalidad de Dr. Scratch es analizar el proyecto realizado en Scratch y evaluar dichos conceptos claves de las ciencias computacionales, en concreto, se centra en 7 aspectos:

- 1. Paralelismo.** Consiste en poder resolver varios problemas de manera simultánea en hilos diferentes.
- 2. Pensamiento lógico.** Nos permite reconocer el problema que queremos solucionar, buscar una solución que podamos programar y analizar las salidas que obtengamos para formarnos conclusiones.
- 3. Control de flujo.** Es la capacidad de poder coordinar el orden en que se ejecutan las instrucciones que tenemos en el programa.
- 4. Interactividad con el usuario.** Habilidad que permite crear proyectos interactivos, que permitan que el usuario que ejecuta el programa, pueda realizar acciones que provoquen nuevas situaciones en el proyecto.
- 5. Representación de la información.** Capacidad de un programador de detectar los datos que el programa necesita, para poder ejecutarse correctamente.
- 6. Abstracción.** Concepto que ayuda a dividir un problema en partes más pequeñas que serán más fáciles de comprender, programar y depurar.
- 7. Sincronización.** Es la comunicación entre las distintas partes del programa que permite la planificación de la ejecución de las instrucciones, en un orden determinado que ha sido elegido con anterioridad.

Todos estos conceptos han de ser evaluados de forma que el usuario al final obtenga una **puntuación total que califique el nivel de desarrollo del pensamiento computacional del programador de proyectos en Scratch**. La evaluación se realiza conforme a los bloques de Scratch utilizados.

¹<http://www.cs.cmu.edu/~wing/>

Pero Dr. Scratch va más allá. Incluye además información sobre malas **prácticas realizadas en programación** observadas en sus años de educación de mi tutor y co-tutor y que todo programador novel realiza en sus primeros proyectos -y últimos si no aprende a corregirlos-.

De esta forma, Dr. Scratch trata de dar una puntuación basada en el nivel computacional del programador, pero además sirve de apoyo para aprender a programar correctamente. Esto la convierte en una herramienta muy útil, tanto para docentes como para estudiantes.

2.2. Objetivos específicos

Una vez detallado el objetivo general del proyecto, necesitamos entender en qué momento de su desarrollo nos encontramos y cuáles son los objetivos que queremos alcanzar al final de este Proyecto Final de Carrera.

Tal como se ha comentado, ya existía una versión Alpha de la web, en la que se tenía un código, que trataba de alcanzar el objetivo e implementación básica de la web. Dicho código, mostraba una web simple e intentaba analizar el archivo .sb2 de Scratch para extraer conclusiones mediante el plugin Hairball y mostrarlas en un dashboard.

Al comienzo de este proyecto, dicho código no conseguía realizar correctamente el análisis del proyecto de Scratch, por lo que partiendo de esta base, los objetivos iniciales fueron los siguientes:

- 1. Corrección de la implementación básica.** El primer objetivo y más importante fue conseguir tener en funcionamiento el prototipo que ofrecía la funcionalidad básica.
- 2. Generación de un log.** Se quería tener información de todos los proyectos analizados en la web: quién lo había analizado -si estaba registrado-, nombre del proyecto, hora y fecha.
- 3. Plataforma en producción.** La idea principal, era que una vez el prototipo funcionase, tenerlo en producción lo antes posible para ser utilizado y poder tener feedback de cuáles eran las necesidades de los usuarios.
- 4. Cambio de apariencia de la web.** La web funcionaba, pero no era demasiado atractiva y se buscaba tener el máximo número de usuarios posibles, por lo que era necesario realizar un cambio. Teníamos además, que buscar algún logotipo que mostrase la esencia de Dr. Scratch.

5. Implementación del mecanismo de traducción de la plataforma. Se busca llegar con Dr. Scratch al mayor número de personas y para ello el idioma no debe ser un impedimento. Por ello un objetivo inmediato fue la configuración del servidor para que fuera capaz de traducir la web al idioma del navegador que hiciera la petición.

Dr. Scratch ha sido un proyecto que ha obtenido financiación por parte del **FECyT** (Fundación Española para la Ciencia y la Tecnología) y por tanto, hay una serie de objetivos iniciales que fueron fijados en la memoria de solicitud de la ayuda. Tenía objetivos referentes tanto a la plataforma web como a otros subobjetivos, tales como la realización de talleres para garantizar la difusión de la plataforma y la creación de materiales docentes, entre otros. En este documento únicamente nos centraremos en los objetivos específicos relacionados con la plataforma web:

- Llegar a conseguir más de 5.000 proyectos analizados.
- Número de alumnos registrados (>1.500)
- Número de docentes registrados (>120)
- Número de visitantes únicos diarios de la web > 25.000
- Número de descargas de la aplicación móvil (> 150)
- Número de horas del servidor caído en los últimos 6 meses de proyecto (< 100h)
- Número de lenguas de la plataforma web (≥ 5)

Nada más tener el prototipo en producción, nos llegaron las primeras peticiones de los usuarios, y por tanto, más objetivos para nosotros y algunos más que surgieron con el avance del proyecto:

- Barra de progreso del análisis del proyecto.
- Analizar proyectos de las dos versiones de Scratch (1.4 y 2.0), únicamente se podía analizar proyectos de la versión 2.0, si intentábamos analizar proyectos de la 1.4 la plataforma mostraba un error.
- Análisis mediante url del proyecto de la versión online de Scratch.

- Cuentas de usuarios.
- Implementación en el servidor para mostrar tres páginas diferentes, conforme el nivel del programador fuera: bajo, medio o alto.
- Diseño de tres páginas distintas para cada nivel computacional.
- Debido al gran número de proyectos que se analizaban diariamente teníamos numerosos problemas con las máquinas de la universidad, por lo que decidimos realizar una migración a la nube.
- Utilización de algún mecanismo de análisis web para tener estadísticas de los accesos a la plataforma.
- Páginas con información de cómo poder pasar de un nivel a otro utilizando distintos bloques de Scratch que demuestran un nivel de pensamiento computacional mayor.
- Análisis para organizaciones, permitiéndoles analizar varios proyectos de Scratch simultáneamente a través de un fichero .csv y le devolviera toda la información del análisis.
- Cuentas de organizaciones.
- Páginas de estadísticas que permitan obtener información acerca de la evolución de los usuarios y poder realizar un seguimiento.
- Extensión para los navegadores que permitiera analizar un proyecto desde la propia página web de Scratch. De forma que únicamente clicando en un botón, la extensión obtuviera la url del proyecto subido y compartido en Scratch mostrado en la página actual y abriese una nueva ventana con su análisis en Dr. Scratch.
- Además de la traducción automática detectando el idioma mediante la configuración del navegador, tener un botón que permita al usuario modificar el idioma.
- Página de agradecimiento a los colaboradores: personas que han traducido los textos de la web a distintos idiomas, colegios que nos han permitido realizar talleres en sus aulas...
- Un foro donde los usuarios pudieran expresar sus opiniones y sugerencias para poder obtener feedback.

- Dashboard para docentes, que mostraran información al maestro sobre sus alumnos.

En última instancia, se tiene en mente que la web incluya mecanismos de gamificación para conseguir motivar a los usuarios a seguir aprendiendo utilizando la herramienta, y además, al más estilo del lenguaje Scratch, tener una red social, que permitiera a los programadores poder interactuar y dinamizar la plataforma.

2.3. Planificación temporal

Al ser un proyecto tan amplio y ser desarrollado en equipo por varias personas, quiero dejar lo más detallado posible cuál ha sido el planteamiento temporal y la participación de todos los integrantes del equipo Dr. Scratch y aportaciones realizadas por Cristian Chusing en la primera versión Alpha. En este punto, únicamente desglosaré la planificación temporal, a la que en próximas secciones iré incluyendo qué miembros han participado en la consecución de cada funcionalidad y objetivo.

Este proyecto ha sido muy bien guiado y gestionado por mi tutor y co-tutor mediante **Trello** -herramienta colaborativa para organizar proyectos-. Y fueron marcados inicialmente, cuáles eran los objetivos a corto, medio y largo plazo. Según el proyecto fue evolucionando, dichas metas podían ser modificadas fácilmente gracias a la herramienta y jamás se perdía de vista cuáles eran las tareas más inmediatas.

- **FASE I.** Esta etapa del proyecto es la realizada desde noviembre de 2014, fecha de comienzo, y febrero de 2015, cuando comenzó Eva Hu. Durante este periodo los objetivos a alcanzar fueron los siguientes:

- Comprender y depurar el código realizado por Cristian Chusing en la versión Alpha de Dr. Scratch.
- Actualización de la versión 1.4 de Django a la 1.7.
- Generación del archivo log explicado con anterioridad.
- Puesta en producción del prototipo.
- Cambio de apariencia de la web.
- Implementación del mecanismo de traducción de la plataforma.

En la *Figura 4.11* se muestra dicha fase de una forma esquemática y sencilla de comprender.



Figura 2.1: Fase I del proyecto Dr. Scratch

- **FASE II.** Esta etapa del proyecto es la realizada desde febrero, cuando se incorporó al proyecto mi compañera Eva Hu, a julio de 2015.

En esta fase del proyecto los objetivos a alcanzar fueron los siguientes:

- Introducir una barra de progreso del análisis del proyecto.
- Corrección de errores en formularios.
- Análisis mediante url del proyecto de la versión online de Scratch.
- Migración de SQLite a MySQL.
- Creación de un archivo en el que se guardaran todos los proyectos en los que se producía algún error al ser analizados.
- Personalización de las páginas de error 404 y 500.
- Analizar proyectos de las dos versiones de Scratch (1.4 y 2.0).
- Implementación en el servidor para mostrar tres páginas diferentes, conforme el nivel del programador fuera: bajo, medio o alto.
- Diseño de tres páginas distintas para cada nivel computacional.
- Introducción explicaciones de qué es una url y cómo descargar un proyecto en Scratch.
- Migración a la nube en lugar de utilizar las máquinas de la universidad, las cuales nos dieron muchos problemas.

- Uso de Google Analytics para tener estadísticas de acceso a la plataforma.
- Páginas explicativas de cómo subir de nivel utilizando bloques de un mayor nivel computacional.
- Nuevo diseño de los dashboards.
- Cuentas de organizaciones.
- Botón de “Ayuda” para saber cómo navegar por el dashboard.
- Análisis multiproyecto con un archivo .csv para organizaciones.
- Traducción de la información mostrada en el archivo de organizaciones.
- Páginas de estadísticas de los proyectos analizados.
- Generación de un archivo .csv que muestra la información obtenida con el análisis realizado en Dr. Scratch y es el que se le devuelve a la organización.

En la *Figura 4.22* podemos ver otro esquema que escenifica esta fase.



Figura 2.2: Fase II del proyecto Dr. Scratch

- **FASE III.** Abarca el periodo entre septiembre de 2015 y 1 de febrero de 2016, fecha en la que empecé a escribir esta memoria.

Trás la Scratch Conference en Ámsterdam a la que acudió todo el equipo de Dr. Scratch y en la que realicé una presentación de la plataforma, pudimos obtener algunas conclusiones gracias a la realimentación obtenida. Debido a ellas, marcamos los siguientes objetivos:

- En el análisis por URL únicamente descargar el .JSON para que se realice más rápidamente.

- Traducción de los bloques pintados en las páginas para aprender cómo subir de nivel de pensamiento computacional.
- Registro de usuarios.
- Botón para traducir la página según la preferencia del usuario y no del navegador.
- Extensión que permita analizar proyectos sin necesidad de abrir en el navegador la propia página de URL, sino que en la página del proyecto de Scratch nos facilite un botón que ya muestre directamente el análisis realizado.
- Traducción de los textos que aparecen en la plataforma a más idiomas: catalán, gallego y portugués.
- Formulario de registro para usuarios y organizaciones.

En la *Figura 4.42* podemos observar estos objetivos detallados en el tiempo.



Figura 2.3: Fase III del proyecto Dr. Scratch

Actualmente, el proyecto continúa desarrollándose, con la ayuda de la Asociación sin ánimo de lucro **Programamos** en la que mi co-tutor, Jesús Moreno, es co-fundador junto con José Ignacio Huertas.

Capítulo 3

Estado del arte

En este capítulo se detallarán de forma resumida las **arquitecturas y tecnologías** utilizadas en la realización de este proyecto. Únicamente se realizará una presentación de las mismas, ya que en el próximo capítulo será donde se detalle cómo han sido utilizadas para el desarrollo e implementación de la plataforma.

3.1. Aplicaciones cliente-servidor

El paradigma **Cliente-Servidor** [5] es utilizado en diversos contextos. Centrándonos en el patrón arquitectural para software distribuido cabe indicar que es un modelo en el que existen dos tipos de elementos -cliente y servidor- y un modelo de interacción basado en un diálogo -petición y respuesta-.

El objetivo de esta arquitectura es **proporcionar el soporte escalable** para el desarrollo de aplicaciones distribuidas.

En cuanto a sus **elementos**:

- **El cliente.** Es la parte de la aplicación con la que interacciona el usuario pero no es compartida entre los mismos. No tiene restricciones especiales referentes a fiabilidad, ya que si falla un cliente, el resto del sistema no se ve afectado. Pero sí las tiene en términos de ergonomía debido a que debe adaptarse a la interacción con el usuario. Este último aspecto en concreto, será tratado con profundidad en el siguiente capítulo, ya que ha sido la mayor preocupación a la hora de diseñar la plataforma.

- **El servidor.** Es el componente que presta servicios al cliente, compartiendo sus recursos con todos ellos. Tiene una gran restricción y es que si el servidor falla, los clientes no pueden continuar. Además de presentar problemas de escalabilidad y debe ser seguro para no comprometer la seguridad de los clientes ni de los datos.

En la *Figura 3.1* podemos observar sus componentes y el intercambio de mensajes. El protocolo de intercambio de mensajes será explicado más adelante.



Figura 3.1: Esquema del paradigma Cliente-Servidor

También es importante entender que la finalidad de una aplicación distribuida es la de ofrecer funcionalidades al usuario y que su **estructura** es siempre la que se muestra en la *Figura 3.2*.



Figura 3.2: Capas presentes en toda aplicación distribuida

1. **Capa de presentación.** También conocida como front-end, suele residir en el cliente y es con la cual el usuario interactúa. Se pueden utilizar distintas tecnologías como: HTML, CSS, javascript...
2. **Lógica de la aplicación o negocio.** Puede residir en cualquiera de los componentes - cliente y servidor- incluso en ambos. Realiza acciones necesarias para desarrollar la actividad de negocio, tales como: cobro de producto, transacciones...
3. **Capa de servicios.** Denominada como back-end, es común que resida en el servidor y realiza las actividades típicas de gestión de ficheros, acceso a datos en la base de datos, procesamiento de la información...

Podemos encontrarnos distintos **tipos de servidores**:

- 1) Dependiendo de si la aplicación requiere que el servidor recuerde la historia pasada del cliente, se diferencian dos tipos:
 - a) Servidor con estados: recuerda información entre las peticiones.
 - b) Servidor sin estados: no recuerda información del cliente entre peticiones.
- 2) Atendiendo al modo de procesamiento de las peticiones, encontramos dos servidores diferentes:
 - a) Servidor concurrente: se atienden las peticiones de forma paralela, gracias a la implementación multihilo.
 - b) Servidor iterativo: se atienden las peticiones de forma secuencial, existiendo una cola de peticiones.

Hypertext Transfer Protocol

El intercambio de mensajes se realiza mediante el protocolo HTTP, el cual es un protocolo sin estado y ha tenido varias versiones. De estas versiones la más utilizada es la versión 1.1 HTTP que especifica la sintaxis y semántica que utilizan cliente y servidor para comunicarse.

La estructura de los mensajes de petición y respuesta según el protocolo HTTP se muestran en la *Figura 3.3*. Son en texto plano lo cual es una ventaja a la hora de depurar pero hace los mensajes largos.



Figura 3.3: Mensajes de petición y respuesta del protocolo HTTP

3.2. Back-end

Tal como hemos visto en la explicación de la arquitectura cliente-servidor el *back-end* se corresponde con la **capa de servicios o de acceso a datos**. Y en esta sección vamos a ver la parte teórica de las tecnologías utilizadas en este proyecto para el desarrollo de dicha capa:

1. El lenguaje de programación elegido: Python
2. El framework utilizado para la implementación del servidor: Django
3. Servidores HTTP utilizados: Apache y Node.js
4. La base de datos configurada: MySQL

3.2.1. Python

El **lenguaje Python** [2], cuyo nombre es en homenaje a la serie *Monty Python's Flying Circus*. Fue desarrollado por *Guido van Rossum* para tratar de superar las limitaciones del *lenguaje ABC*, que era comúnmente utilizado en el centro de investigación en el que trabajaba en Ámsterdam.

Desde el principio se buscó que fuera un *lenguaje divertido* a la hora de ser utilizado y es común utilizar variables que hacen referencia a sketches de los Monty Python en lugar de variables tradicionales.

En un principio iba a publicarlo bajo una licencia open source propia, pero con la versión 1.6 se decidió por cambiar a **licencia GPL** (GNU General Public Licence). Así, es posible modificar el código fuente y desarrollar código derivado sin la necesidad de hacerlo open source.

Existen **tres versiones** principales, con 1.6, 2.7 y 3.4 como últimas actualizaciones. Es destacable indicar que las versiones 2 y 3 son incompatibles entre sí, por el gran número de *diferencias entre ellas* [7].

El desarrollo y promoción se realiza a través de la organización, sin ánimo de lucro, *Python Software Foundation*, destacando su **carácter open source**.

Las **características** principales de este lenguaje son las siguientes:

- Es un lenguaje de *alto nivel*, expresando el algoritmo de forma entendible para el programador.

- *Interpretado*, se ejecuta mediante un intérprete y así es más flexible que un lenguaje compilado.
- Su filosofía se centra en tratar de conseguir un *código lo más legible posible* para el ser humano. Los programas en Python parecen pseudocódigos.
- Es *orientado a objetos*, con sus clases, propiedades, métodos.... Todo son objetos.
- *Dinámicamente tipado*: puede indicarse el tipo en cualquier momento, no es necesario declararlos en un método. Esto reduce el ciclo editar-compilar-comprobar-depurar.
- *Fuertemente tipado*: una vez declarada una variable con un tipo, esta no puede realizar una violación de tipo de datos, sino que deberá ser convertida previamente.
- *Case sensitive*: diferencia entre mayúsculas y minúsculas.

Ha sido el lenguaje elegido, debido principalmente a dos razones, es el lenguaje que me ha resultado más atractivo en mis años de aprendizaje en la universidad, y en segundo lugar, por su amplia utilización mundial. Las estadísticas mostradas por CodeEval¹ para 2015 de los lenguajes más utilizados pueden observarse en la *Figura 3.4*. En 2014, el dato para Python era de 30,3 %, por lo que sigue aumentando el número de programadores que se decantan por este lenguaje.

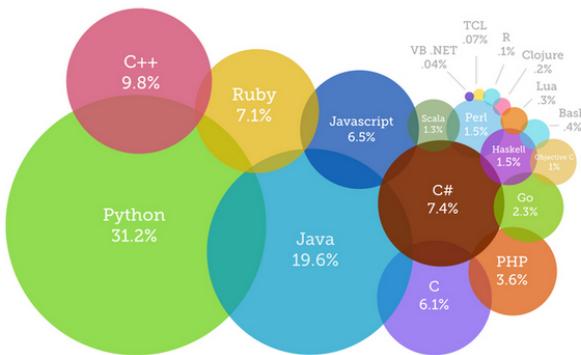


Figura 3.4: Lenguajes de programación más utilizados en 2015

Por último, destacar que Python es un **lenguaje multiplataforma** por lo que se puede utilizar en diversas plataformas. Y cuenta con varios *web frameworks*, entre los que destaca Django, que es la tecnología que trataré a continuación.

¹<https://www.codeeval.com/>

3.2.2. Django

Django es un framework que facilita al programador la creación de sitios webs complejos.

Entre sus principales **características**, destacar:

- Es de *código abierto*.
- Todo está escrito en *Python*.
- Sigue el concepto *DRY* -Don't Repeat Yourself- intentando no duplicar código.
- El programador no tiene que preocuparse de realizar las tareas repetitivas que siempre hay que realizar al crear un servidor web: abrir sockets, cerrar sockets, configuración de protocolos...
- Puede ser utilizado con varias bases de datos, tales como MySQL, SQLite o PostgreSQL.
- Su filosofía de desarrollo es MVC -Modelo Vista Controlador-, pero no en sentido estricto, sino que “hacen lo que les parece correcto”. De tal forma, podemos encontrar nombres distintos a los comúnmente utilizados en un framework MVC puro. Se podría decir que es un modelo MTV -Modelo Template Vista-.

Para explicar esta última característica y cómo funciona el modelo de Django en concreto, he creado la *Figura 3.5*².

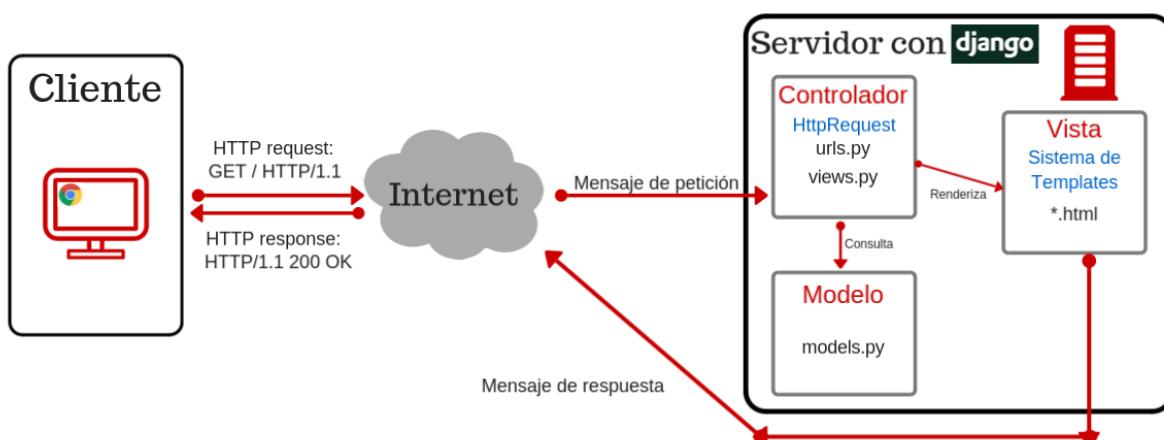


Figura 3.5: MVC adaptado de Django

²Para la creación de esta figura me he apoyado en la información encontrada en <http://es.slideshare.net/alatar/django-el-framework-web-definitivo-1362169>

Al crear un proyecto nuevo, tendremos por defecto el árbol de directorios mostrado en la *Figura 3.6*, donde se pueden observar algunos de los ficheros mencionados al describir el modelo seguido por Django.

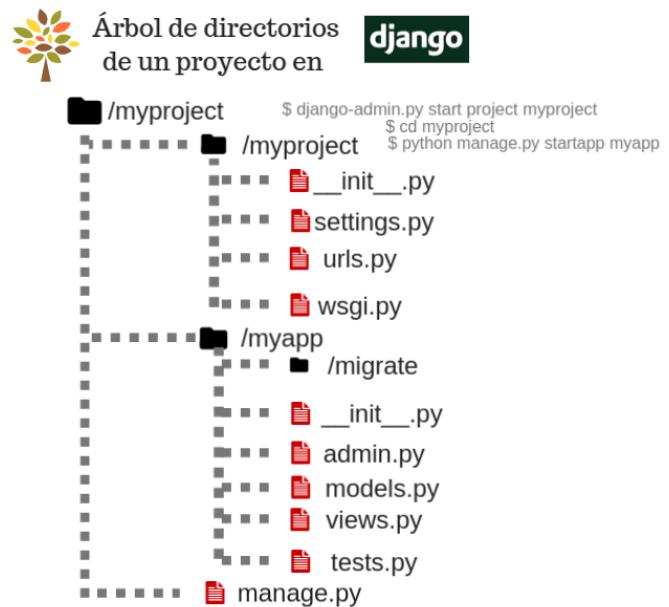


Figura 3.6: Árbol de directorios de un proyecto en Django

Django tiene una gran **comunidad** de usuarios que defienden al igual que Python las buenas prácticas de programación y que el código sea legible [4].

3.2.3. MySQL

Todo servidor web necesita un *servidor de base de datos* que administre, almacene y gestione los datos, para asegurar el acceso seguro a ellos.

MySQL [9] es uno de los sistemas de administración de bases de datos más utilizados en la actualidad, se puede observar en la *Figura 3.7*³ cuál es el ranking que ocupa relacionado con otras bases de datos muy utilizadas en el mercado, donde se puede observar que es una de las tres con más usuarios.

³Información obtenida en http://db-engines.com/en/ranking_trend

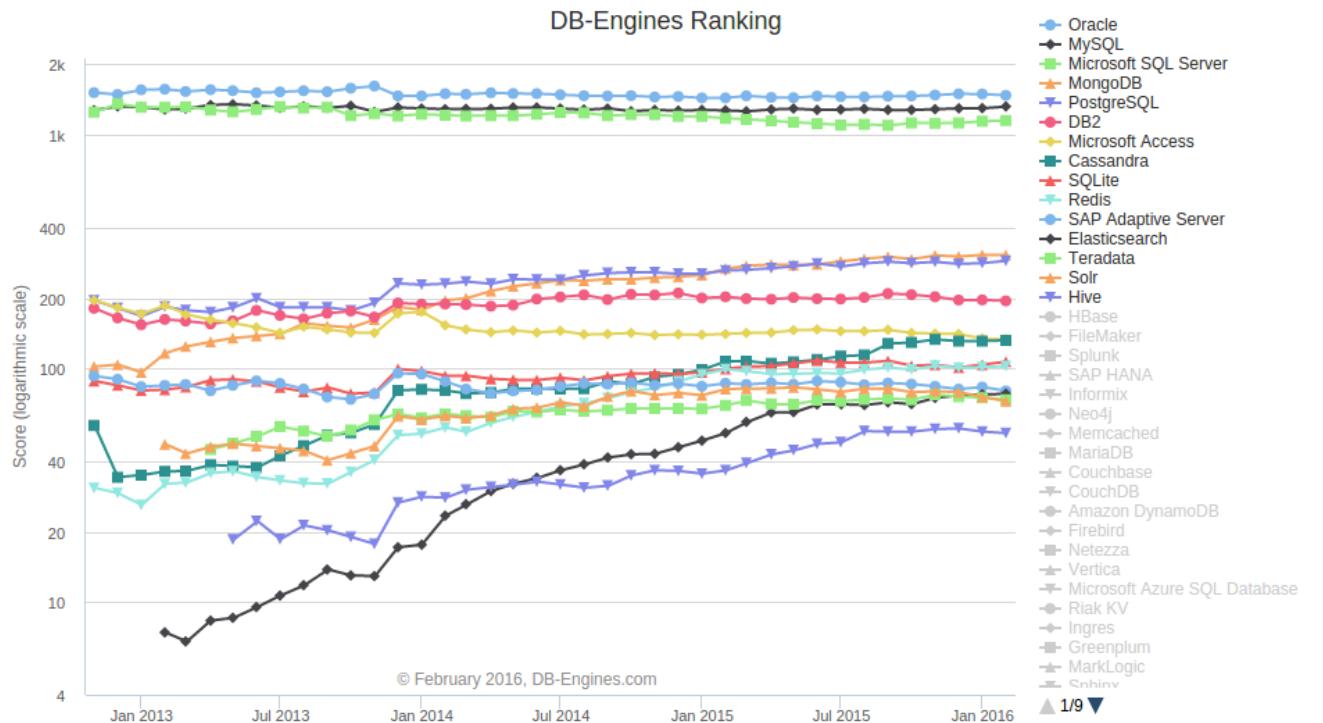


Figura 3.7: Ranking de las bases de datos más utilizadas actualmente

Las principales **características** de MySQL son las siguientes:

- Utiliza **tablas** para guardar los datos.
- Tiene un mecanismo interno de almacenamiento el cual guarda los datos salvados en dispositivos físicos.
- Es **relacional(BDR)**, por lo que permite establecer interconexiones o relaciones entre los datos ya guardados en tablas, siguiendo el *modelo relacional*.
- Es **multiplataforma**.
- Utiliza multihilos mediante hilos del kernel.
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP.
- Tiene bases de datos de hasta 50 millones de registros.

Destacar que tiene **licenciamiento dual**: se puede utilizar bajo la GNU GPL, pero si se quiere incorporar en productos privativos es necesario comprar otra licencia.

3.2.4. Servidor HTTP Apache

El **servidor HTTP Apache** es un servidor web HTTP de *software libre*. El servidor se desarrolla y mantiene bajo la *Apache Software Foundation* dentro del proyecto HTTP Server (*httpd*).

La arquitectura del servidor Apache es **altamente modular**, lo cual permite añadir más funcionalidades de modo sencillo y ser extendido fácilmente. Esta es una de las características que le hacen ser muy utilizado a la hora de poner una web en producción, ya que se pueden incluir módulos de frameworks. Como es el caso del módulo *mod_wsgi de Django*.

Algo históricamente notable es la licencia de software bajo la cual es distribuido. La **Licencia Apache** es una licencia de software libre y permite al usuario la libertad de uso para cualquier propósito, ya sea de distribución, para ser modificado y le permite además, distribuir versiones modificadas del software.

La **configuración** del mismo se realiza mediante el fichero apache2.conf o **httpd.conf** y para que los cambios en él sean agregados es necesario reiniciar el servidor.

3.2.5. Servidor HTTP Node.js

Node.js es un entorno de ejecución que permite trabajar sobre el intérprete de *JavaScript* en el lado del servidor. Utiliza el motor V8, desarrollado por Google para el navegador Chrome, compilando javascript en el lado del servidor a altas velocidades.

Señalar que es de código abierto y multiplataforma. Y la característica que lo hace destacar es el hecho de poseer un excelente *modelo de eventos*, ideal para la programación asíncrona.

3.3. Front-end

Recordar que el *front-end* es la **capa de presentación** que toda aplicación distribuida posee. Es la parte con la que el cliente interacciona. Y en las siguientes subsecciones se van a explicar algunas de las tecnologías más utilizadas actualmente en el desarrollo web y que se muestran en la *Figura 3.8*.



Figura 3.8: Tecnologías más utilizadas en la capa de presentación

3.3.1. HTML

HTML (HyperText Markup Language) no es un lenguaje de programación, sino un *lenguaje de marcado* para la creación de páginas webs. Es utilizado por todos los navegadores actuales.

Permite dar estructura e introducir contenido en las páginas webs a través de las distintas **etiquetas** que componen el lenguaje, utilizando únicamente texto. En la *Figura 3.9* se puede observar la estructura básica de un código HTML.

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Información técnica para el navegador-->
    </head>
    <body>
        <!-- Contenido que aparecerá en la página web-->
        <p>Hello, World!</p>
        
    </body>
</html>
```

Figura 3.9: Estructura básica de un código HTML

Para incluir un **elemento** en una página necesitamos indicar la etiqueta de inicio, su contenido -texto que aparecerá en la web- y la etiqueta de cierre, generalmente. Las etiquetas de inicio tienen **atributos** en los que se especifica mediante texto plano la ubicación de los contenidos(imágenes, vídeos...) que queremos introducir en la página y es el navegador quien se encarga de interpretarlo y mostrar la página web completa.

3.3.2. CSS

CSS (cascading style sheets) es utilizado para dar estilo a un lenguaje estructurado, como puede ser HTML. Se aconseja que el código CSS se incluya en otro documento separado del HTML y que éste sea incluido mediante la etiqueta adecuada (`<link>`), tratando de *no mezclar lenguajes diferentes en un mismo texto* para facilitar su legibilidad.

Al seguir la norma de estilo anterior, el modo de indicar a un elemento de HTML que debe seguir una norma de estilo concreta se realiza incluyendo en la etiqueta de dicho elemento alguno de los dos atributos de nombre: id o class. Estos dos atributos tendrán un valor determinado que será el especificado en una sección del código CSS que tendrá unas reglas propias.

El mayor potencial de CSS es su **sintaxis simple** pero hay que tener mucho cuidado con las jerarquías para incluir el estilo que deseamos a los contenidos deseados.

3.3.3. Bootstrap

Bootstrap es un framework que permite al usuario despreocuparse en gran medida del diseño y apariencia al realizar una aplicación web. Entre sus principales aportaciones están las numerosas **plantillas** que permiten tener implementado el diseño de una página web completa en pocas horas.

Está *basado en HTML y CSS* separando por completo dichos lenguajes en documentos diferentes lo que permite que una plantilla pueda ser modificada y ampliada fácilmente por el usuario para adaptarlo al diseño deseado de un modo intuitivo.

Las plantillas generadas, normalmente incluyen también ficheros con *JavaScript* para aportar a la página un diseño más atractivo para el usuario, en la siguiente subsección se explicará qué mejoras introduce este lenguaje a una página creada únicamente con HTML y CSS.

3.3.4. JavaScript

JavaScript es un lenguaje de programación interpretado utilizado de forma extendida para poder *dinamizar el lado del cliente* incluyendo así mejoras en la interfaz de usuario y aumentar su seguridad.

Aun que su principal uso es en el lado del cliente, también se utiliza con el apoyo de otras tecnologías para enviar y recibir información al servidor.

3.3.5. AJAX

AJAX (Asynchronous JavaScript And XML) es una tecnología que permite enviar y recibir información desde el servidor al cliente, permitiendo diseñar páginas webs más interactivas.

Este envío de información se realiza gracias a una *comunicación asíncrona* con el servidor siendo posible realizar cambios en la página siendo recargada automáticamente.

3.4. Plataforma en la nube: Azure

Siempre que se persigue trasladar una aplicación distribuida a producción es necesario una máquina o máquinas en las que implementar todas las tecnologías que han sido mencionadas anteriormente. **Microsoft Azure** es una plataforma alojada en los Data Centers de Microsoft que permite, entre otros servicios, la posibilidad de tener máquinas virtuales con una gran selección de sistemas operativos.

Por ello, es una herramienta muy potente en la que alojar nuestra aplicación asegurando *escalabilidad, seguridad y un 24x7 fiable*.

3.5. Analítica web: Google Analytics

El objetivo principal de una web en producción es que sea utilizada por el mayor número de usuarios posible y para poder realizar un seguimiento y medición de dicho objetivo **Google Analytics** nos permite tener numerosas *estadísticas* y herramientas que nos reportan dicha información.

Su utilización es muy simple, al crear una cuenta, *se genera un código en JavaScript que debe ser incluido en las distintas páginas de la web*.

Además, Google Analytics incluye numerosa información en texto y vídeos de cómo leer la información dada y de cómo optimizar aspectos tan importantes en las webs como es la segmentación del tráfico o publicidad online.

Capítulo 4

Diseño e implementación

En este capítulo se va a detallar *cómo se han implementado las distintas tecnologías* explícadas en el capítulo anterior.

También se irá explicando cuál ha sido la *evolución en la implementación de las funcionalidades, debido a las necesidades observadas en los test de usabilidad*, realizados en dos colegios de la Comunidad de Madrid, con alumnos de 5º y 6º de primaria: CEIP Lope de Vega y CEIP Gonzalo Fernández de Córdoba.

Y en el último apartado, una vez entendidas la estructura general y el objetivo de cada funcionalidad, se mostrarán algunos *diagramas de flujo para explicar el código de las implementaciones principales*.

4.1. Arquitectura general de Dr. Scratch

Como toda aplicación *Cliente-Servidor*, Dr. Scratch tiene las 3 capas de desarrollo: back-end, lógica de la aplicación y front-end.

Pero al ser Dr. Scratch un analizador de archivos -en este caso, el JSON del proyecto de Scratch- la lógica o negocio de la aplicación está clara. Por ello se va a prestar mayor atención a la capa de servicio y de presentación.

Señalar que esta sección no trata de explicar cómo se han implementado las diferentes tecnologías, cosa que se hará en la siguiente, sino de *mostrar una idea general de cómo todas las tecnologías han sido utilizadas en conjunto*. Busco mostrar cuál es la arquitectura final de Dr. Scratch para después poder profundizar teniendo claras cada una de sus partes.

4.1.1. Capa de servicio: back-end

En la *Figura 4.1* se puede observar la arquitectura de Dr. Scratch y la *conexión entre las distintas tecnologías utilizadas para la implementación de la capa de servicio*.

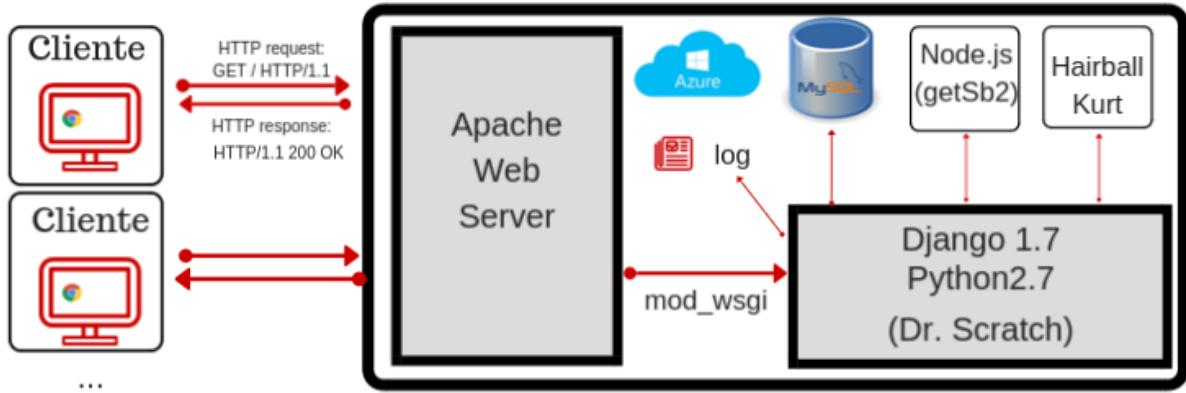


Figura 4.1: Arquitectura general de Dr. Scratch

Azure

Todo lo que se va a proceder a explicar a continuación, ha sido instalado y programado en una máquina virtual en la nube, mediante la plataforma **Microsoft Azure** con un *sistema operativo Ubuntu 14.04.2 LTS*.

Esto ha sido posible, porque Microsoft nos ha ofrecido una *suscripción de 150\$* al mes, de forma gratuita.

Apache

Actualmente, se encuentra instalada la *versión 2.4.10*, ya que es la compatible con la versión de Ubuntu de nuestra máquina. La *configuración del fichero httpd.conf* me llevó algún tiempo entenderla, y además, tuve algún problema para incluir el **módulo mod_wsgi** que permite enlazar Azure con Django. La tarea realizada por Apache nos resuelve numerosos problemas y su funcionamiento es el siguiente: cuando los clientes realizan peticiones HTTP a nuestro dominio a través de la red, estas son escuchadas por el servidor de HTTP implementado bajo la licencia Apache, la cual permite utilizar Dr. Scratch de forma simultánea por numerosos usuarios, teniendo estos la percepción de ser los únicos usándolo.

Django

La versión actualmente instalada de Django es la 1.7, ya que la 1.8 está aún en desarrollo. Como se observa, Apache se encarga de hacer llegar la petición a Django, donde tal como se puede ver en la *Figura 4.2* se pregunta en el fichero urls.py, si dicho recurso es servido por nuestro servidor. El fichero urls.py es básicamente una lista, en la que mediante expresiones regulares, se indica qué recursos son servidos por nuestra aplicación.

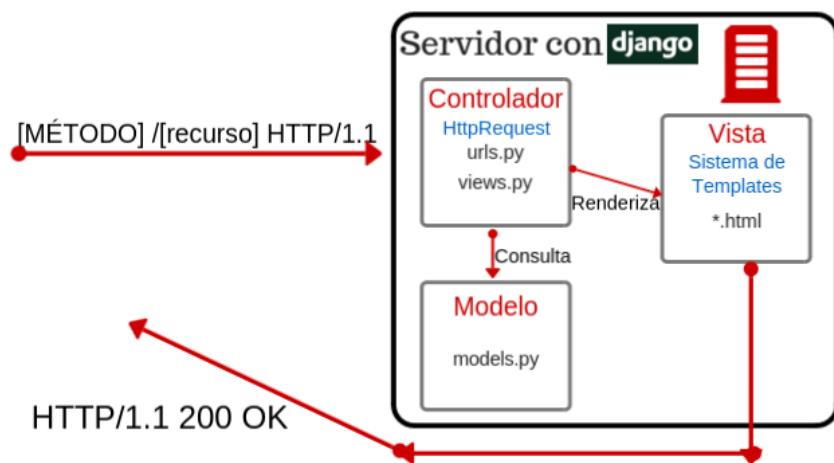


Figura 4.2: Modelo de Django en Dr. Scratch

Dr. Scratch se ha implementado de tal forma, que si se realiza una *petición a nuestro dominio con un recurso que no es servido, redirija al usuario a la página principal*. Así si el usuario, que puede ser un niño pequeño, se equivoca, pueda seguir navegando sin mayor esfuerzo o ver alguna página de error. Internamente, la lógica de este funcionamiento es el mostrado en el diagrama de la *Figura 4.3*.

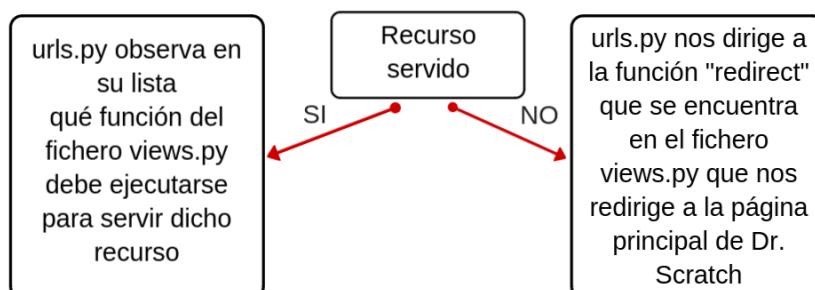


Figura 4.3: Funcionamiento de la petición de recursos a Dr. Scratch: urls.py

En el momento que `urls.py` nos dirige a una función -recordar que en Python no existen los procedimientos- las cuales se encuentran, según el modelo de Django, en el fichero `views.py`, se debe hacer distinción según si el método de la petición es: *GET, POST u otro método*. Con esto, se ejecutará un código concreto según la función y el método.

Algunos métodos únicamente servirán el recurso solicitado, si el método que acompaña a la petición es GET, y en otro caso redirigirá a la página principal de Dr. Scratch, para que al igual que en el caso anterior, el flujo de la aplicación sea lo más dinámica posible. En la *Figura 4.4* se observa el funcionamiento interno.

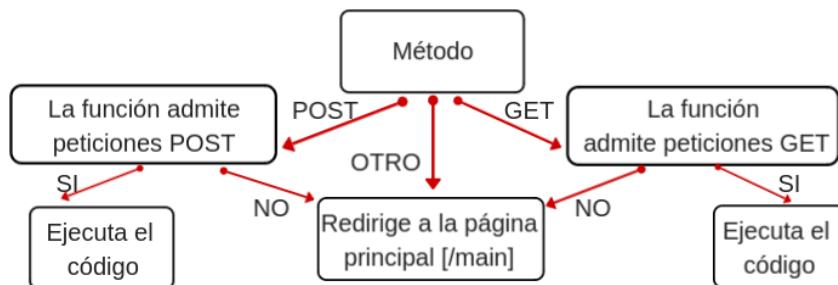


Figura 4.4: Distinción según el método de la petición: views.py

Al ser Dr. Scratch una plataforma con numerosas funcionalidades, fue necesario apoyar al fichero `views.py` con *otra serie de ficheros en los que se implementaban funciones auxiliares*. Así se permite observar fácilmente el flujo de la plataforma y **aumentar su modularidad**. Los otros ficheros que apoyan a `views.py` se pueden ver en la *Figura 4.5*.

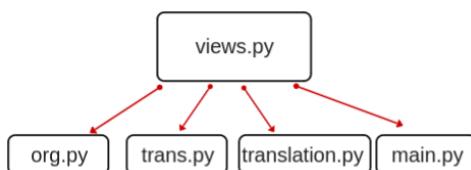


Figura 4.5: Modularización del fichero views.py

Para realizar el código de las funciones implementadas en los ficheros anteriores, hay que recordar que Django está completamente programado en Python y que este es un lenguaje orientado a objetos. Por ello, todo lo que encontraremos en nuestro código son objetos.

La declaración de los objetos que conforman nuestra aplicación se encuentra en el fichero `models.py` del modelo de Django. La declaración de uno de los objetos más utilizados por Dr. Scratch se muestra en la *Figura 4.6* donde se pueden observar sus campos. *Las distintas clases*

declaradas en este fichero serán instanciadas en las funciones para implementar las diferentes funcionalidades de la plataforma.

```

import datetime
from django.db import models
from django.contrib.auth.models import User

# Models of drScratch

class File(models.Model):
    filename = models.CharField(max_length=100)
    organization = models.CharField(max_length=100, default='drscratch')
    coder = models.CharField(max_length=100, default='drscratch')
    method = models.CharField(max_length=100)
    time = models.DateField(auto_now=False)
    language = models.TextField(default="en")
    score = models.IntegerField()
    abstraction = models.IntegerField()
    parallelization = models.IntegerField()
    logic = models.IntegerField()
    synchronization = models.IntegerField()
    flowControl = models.IntegerField()
    userInteractivity = models.IntegerField()
    dataRepresentation = models.IntegerField()
    spriteNaming = models.IntegerField()
    initialization = models.IntegerField()
    deadCode = models.IntegerField()
    duplicateScript = models.IntegerField()

```

Figura 4.6: Declaración de los objetos de Dr. Scratch: *models.py*

Una vez que ya sabe Dr. Scratch qué código debe ejecutar según lo visto ahora, procede y según la función, deberá realizar diversas acciones: consultas en la base de datos, realizar el análisis con Hairball, hacer una petición a otro servidor para obtener información, guardar el archivo... Todo este funcionamiento se irá detallando en la siguiente sección, donde se explicarán las implementaciones realizadas.

Tras ejecutar dicho código y procesar toda la información, se debe mostrar una página de respuesta, para lo que se utiliza el *sistema de plantillas* de Django llamadas **templates**. Estas plantillas son *renderizadas* con la información que se ha procesado y se quiere mostrar en dichas páginas.

Fichero log

Con el objetivo de tener un fichero que guarde información de todos los archivos analizados y poder *observar en qué momento se producen problemas* en nuestro servidor, se añadió un fichero *logFile.txt* a nuestro modelo. Este fichero es abierto, escrito y cerrado cada vez que un proyecto de Scratch es analizado agregando información relevante del mismo.

Base de datos

Dr. Scratch tiene para poder almacenar información una base de datos en **MySQL 5.5.43** a la cual se accede para realizar consultas en las propias funciones con la ayuda de Django de un modo fácil y sencillo.

En la *Figura 4.7* se muestra una captura de pantalla de todas las tablas creadas en la base de datos por la plataforma.

```
mysql> show tables;
+-----+
| Tables_in_drscratch |
+-----+
| app_activity
| app_coder
| app_comment
| app_csvs
| app_discuss
| app_file
| app_organization
| app_organizationhash
| app_stats
| app_student
| app_teacher
| auth_group
| auth_group_permissions
| auth_permission
| auth_user
| auth_user_groups
| auth_user_user_permissions
| django_admin_log
| django_content_type
| django_migrations
| django_session
+-----+
21 rows in set (0.00 sec)
```

Figura 4.7: Captura de pantalla de la base de datos

Como se observa, hay algunas tablas que comienzan por *app_*, otras por *auth_* y finalmente otras por *django_*. Únicamente las que empiezan por *app_* son las que creamos nosotros en el fichero `models.py`, las demás son definidas por Django de forma automática.

Para entender correctamente la estructura de la base de datos y cuáles son los objetos principales, con sus respectivos campos, conviene fijarse en la *Figura 4.8*. En ella, se detallan únicamente los modelos generados por nosotros y *auth_user*, de la cual heredan algunas de nuestras clases.

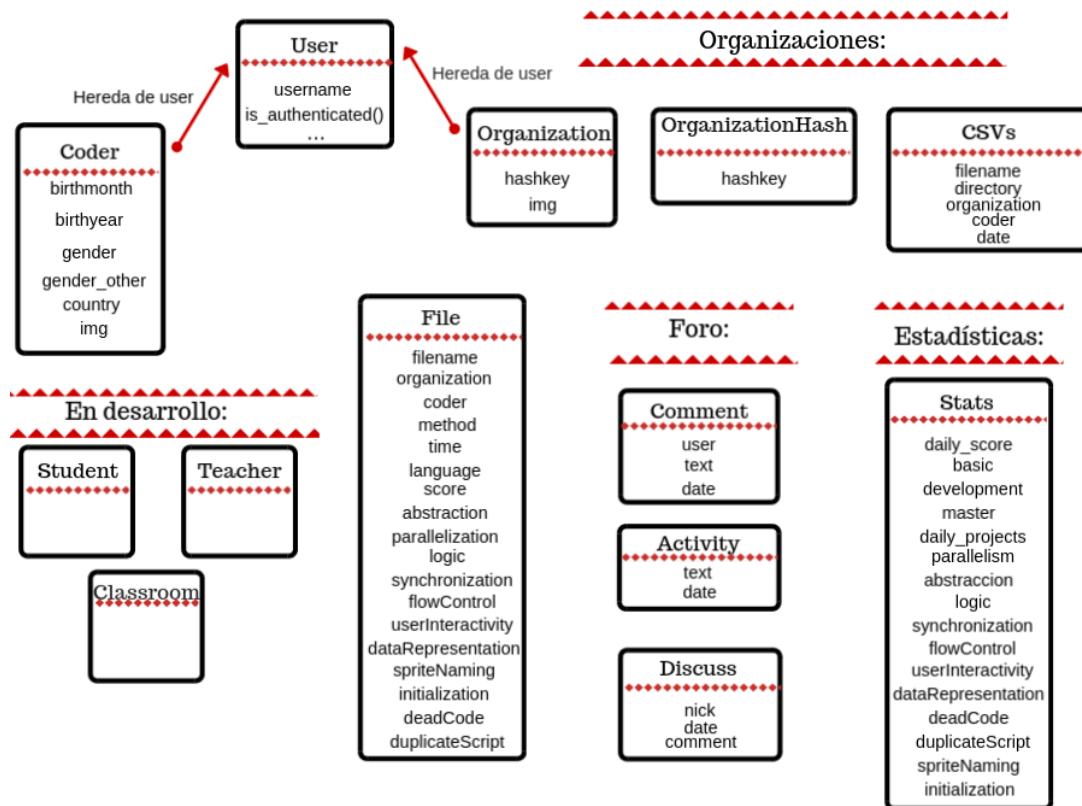


Figura 4.8: Modelos de Dr. Scratch:models.py

Node.js (getSb2)

También se ha implementado con la ayuda del proyecto de un usuario de Github, llamado *Nathan*¹, un servidor HTTP Node.js que se encarga de realizar peticiones al servidor de *Scratch*, y que será explicado más adelante.

Hairball y Kurt

El análisis del proyecto de *Scratch* se realiza mediante Hairball, el cual tenemos instalado en la máquina virtual, y que se ayuda de *Kurt* para manipular proyectos complejos de *Scratch*, con comandos sencillos de Python.

¹<https://github.com/nathan/getsb2>

4.1.2. Capa de presentación: front-end

La capa de presentación de Dr. Scratch está formada por un **conjunto de plantillas** de código HTML, que utilizan hojas de estilo CSS con la ayuda del framework Bootstrap. Estas páginas utilizan además, código JavaScript para mejorar su interactividad con el usuario, y AJAX para realizar algunas funcionalidades en segundo plano.

En la *Figura 4.9* podemos observar cuáles son las **páginas de Dr. Scratch** que son renderizadas y mostradas al usuario asociándolas a un **recurso** determinado.

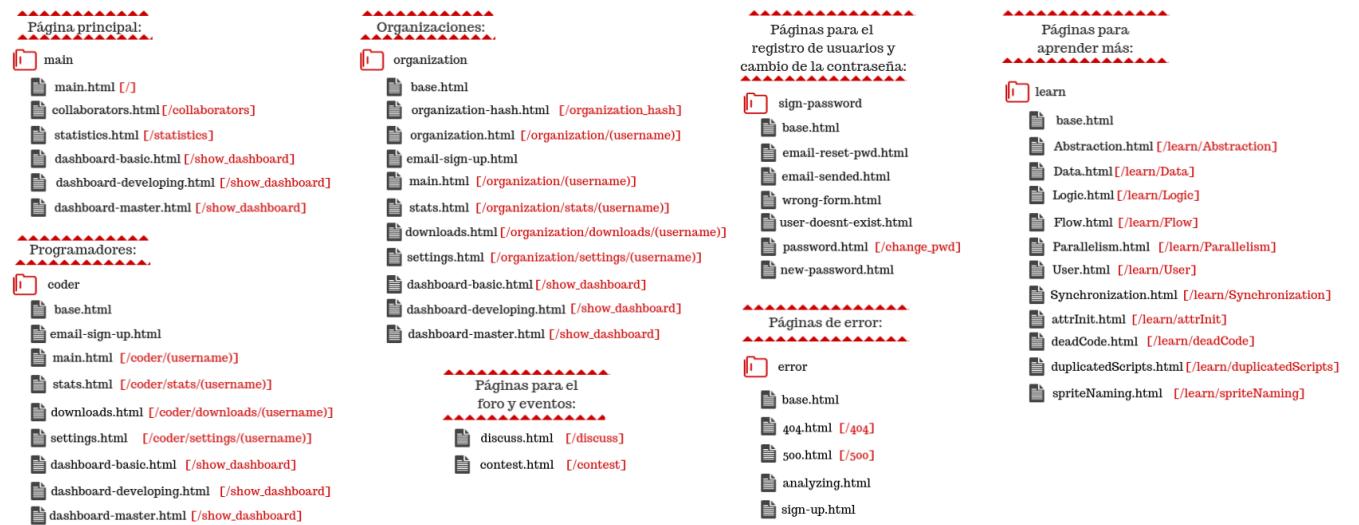


Figura 4.9: Estructura de las páginas de Dr. Scratch: templates

Se decidió realizar una clasificación de las páginas en varias carpetas para facilitar su comprensión y tenerlas organizadas de una forma útil para nuevas implementaciones.

Además, en algunas de las secciones hay páginas denominadas `base.html`, que contienen el diseño básico del conjunto de páginas que se encuentran en la misma carpeta. Las demás páginas *extienden* de su respectivo `base.html` e introducen contenido específico.

La capa de presentación ha sufrido *numerosas modificaciones provocadas por las necesidades observadas en los usuarios en los test de usabilidad, y por las peticiones realizadas directamente por varios docentes de todo el mundo*. Dicha evolución de la plataforma, se verá plasmada claramente en la siguiente sección.

4.2. Funcionalidades y evolución de Dr. Scratch

Con el objetivo de describir claramente cuál ha sido mi aportación al proyecto, quiero recordar que *Dr. Scratch ha sido desarrollado por varias personas*. Este Proyecto Fin de Carrera es la continuación de una primera implementación de Cristian Chusing (versión Alpha) pero que aún no había terminado todo su desarrollo y realizaría algunas otras tareas de forma paralela. Y además ha sido desarrollado en paralelo con las implementaciones realizadas por mi compañera Eva Hu.

Para comprender cuál ha sido el proceso de implementación, he dividido esta sección en 4 apartados:

- 1. Versión Alpha:** Muestra cuáles eran las implementaciones que debía contener el prototipo inicial.
- 2. Fase I:** Se detallan las implementaciones realizadas en los primeros meses, cuando aún Cristian y Eva no se habían incorporado al proyecto.
- 3. Fase II:** Implementaciones realizadas entre el momento en el que mis compañeros retoman o se incorporan al proyecto y la 2015 Scratch Conference.
- 4. Fase III:** Últimas funcionalidades desarrolladas antes de la presentación de este documento.

4.2.1. Versión Alpha

El **prototipo inicial** de Dr. Scratch debía implementar:

- Una página principal en la que poder indicar qué fichero quería subirse.
- Análisis del proyecto de Scratch mediante Hairball.
- Obtener toda la información útil del análisis de Hairball que quería mostrarse al usuario.
- Mostrar dicha información en un dashboard fácil de entender por el usuario.

Al comenzar este Proyecto Fin de Carrera mis tutores compartieron conmigo el código que Cristian les había hecho llegar, código en el cual la página principal tenía la apariencia

mostrada en la *Figura 4.10* y que debía realizar dichas funciones perfectamente, porque tenía todo programado, pero a la hora de ejecutar se producía un error y no conseguía mostrar la información del análisis de Hairball. Conociendo ahora completamente el funcionamiento de Dr. Scratch, puedo suponer que el problema residía en que se debió realizar algún cambio en el plugin Hairball y el código que procesaba la información de Hairball se había quedado obsoleto.

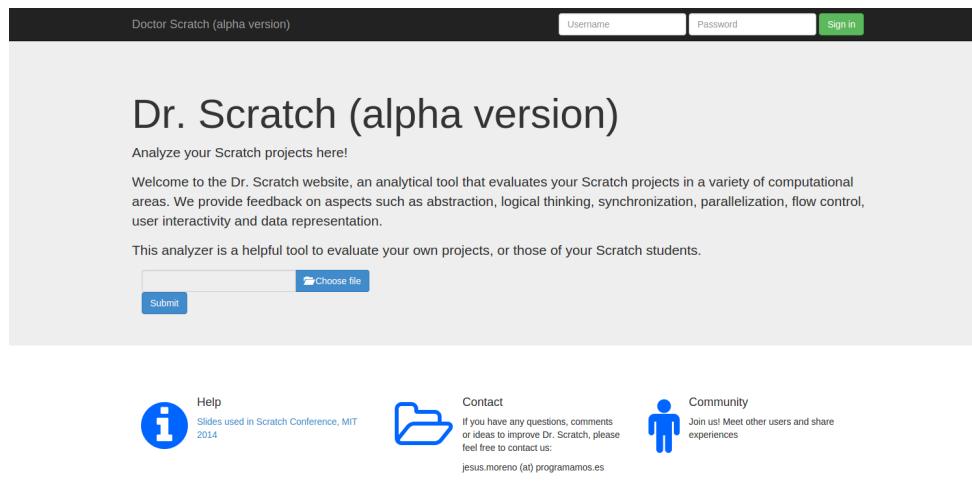


Figura 4.10: Apariencia de la versión Alpha de Dr. Scratch

4.2.2. Fase I

En el contexto anterior, se fijaron mis principales objetivos a conseguir y mis primeras implementaciones que son las mostradas en la *Figura 4.11*. Señalar que en esta fase del proyecto, no se encontraban realizando el proyecto ninguno de mis dos compañeros.



Figura 4.11: Fase I del proyecto Dr. Scratch

Estudio, depuración y modificación del código de la versión Alpha

Aunque el código estaba bastante bien estructurado y comentado y aún no había demasiada cantidad, esta primera tarea fue muy dura. Es complicado enfrentarse al código de otra persona añadido a que el código **presentaba errores a la hora de ejecutar** y al principio no pude ver el resultado final. Con la ayuda de trazas detecté que el error se producía a la hora de procesar las salidas de Hairball las cuales se pueden observar en la *Figura 4.12*.

```
[ ] Mastery:  
meryblue@malu-XPS-L701X:~$ hairball -p mastery.Mastery Trash-2.sb2  
Trash-2.sb2  
{'Abstraction': 2, 'Parallelization': 3, 'Logic': 3, 'Synchronization': 3,  
'FlowControl': 2, 'UserInteractivity': 2, 'DataRepresentation': 2}  
Total mastery points: 17/21  
Average mastery points: 2.43/3  
Overall programming competence: Proficiency  
[ ] Dead code:  
meryblue@malu-XPS-L701X:~$ hairball -p blocks.DeadCode Trash-2.sb2  
Trash-2.sb2  
{u'Contenedor verde': [kurt.Script([  
    kurt.Block('forward:', 10),  
    kurt.Block('forward:', 10)], pos=(381, 257)),  
    kurt.Script([  
        kurt.Block('pointTowards:', u'')], pos=(653.35, 345.75))),  
u'fire2': [kurt.Script([  
    kurt.Block('forward:', 10),  
    kurt.Block('turnRight:', 15)], pos=(549.5, 249.9))]}  
[ ] Sprite naming:  
meryblue@malu-XPS-L701X:~$ hairball -p convention.SpriteNaming Trash-2.sb2Trash-2.sb2  
4 default sprite names found:  
Sprite3  
Sprite1  
Sprite2  
[ ] Duplicate scripts:  
meryblue@malu-XPS-L701X:~$ hairball -p duplicate.DuplicateScripts Trash-2.sb2  
Trash-2.sb2  
1 duplicate scripts found  
[when @greenFlag clicked, 'go to x:%s y:%s', 'show', 'forever%s', 'change x by %s', 'if %s then%s',  
'hide', 'wait %s secs', 'go to x:%s y:%s', 'show', '%s < %s', 'pick random %s to %s', 'x position']  
[ ] Attribute initialization:  
meryblue@malu-XPS-L701X:~$ hairball -p initialization.AttributeInitialization Trash-2.sb2  
Trash-2.sb2  
{u'Basura mezclada sin reciclar': {'position': 1, 'size': 0, 'costume': 0, 'visibility': 2, 'orientation': 2},  
u'Contenedor verde': {'position': 1, 'size': 0, 'costume': 0, 'visibility': 2, 'orientation': 1}, u'fire2':  
{'position': 2, 'size': 0, 'costume': 0, 'visibility': 2, 'orientation': 1}, 'stage': {'background': 2}, u'Sprite4':  
{'position': 2, 'size': 0, 'costume': 0, 'visibility': 2, 'orientation': 0}, u'crystalbottle': {'position': 0, 'size':  
0, 'costume': 0, 'visibility': 2, 'orientation': 0}}
```

Figura 4.12: Salidas de los plugins de Hairball

Como se puede observar, las salidas no están normalizadas y tuve que programar métodos distintos para cada una de ellas y guardar toda la información relevante en un diccionario.

Una vez que tuve el diccionario, modifiqué la plantilla de la versión Alpha, para mostrar la información de este nuevo diccionario. Únicamente modifiqué cómo mostrar la información, el resto de la plantilla y apariencia es la heredada de la versión Alpha y que puede verse en la *Figura 4.13*.

Dashboard Overview

Welcome to Doctor Scratch. Project results.

CT Score: 17/21

Your level: Master

CT Score in detail:

Concepts	Points
Abstraction	2/3
Parallelization	3/3
Logic	3/3
Synchronization	3/3
FlowControl	2/3
UserInteractivity	2/3
DataRepresentation	2/3

Possible issues to watch out:

- Duplicated Scripts**: 2 issues
- Sprites naming**: 4 issues
 - Incorrect name: Sprite4
 - Incorrect name: Sprite3
 - Incorrect name: Sprite1
 - Incorrect name: Sprite2
- Dead Code**: 2 issues
 - fire2 : 2 blocks
 - Contenedor verde : 3 blocks
- Sprite attributes initialization**: 4 issues
 - Contenedor verde: (orientation, position) modified but not initialized correctly.
 - fire2: (orientation) modified but not initialized correctly.
 - Basura mezclada sin reciclar: (position) modified but not initialized correctly.
 - Ghoul: (costume) modified but not initialized correctly.

Figura 4.13: Primer dashboard de Dr. Scratch

En la primera sección donde pone “*CT Score: 17/21*” se mostraba la puntuación total del plugin mastery como resultado de las obtenidas en las distintas habilidades de Pensamiento Computacional que están detalladas en la sección que indica “*CT Score in detail*”, puntuando cada habilidad de 0 a 3. En la tercera sección llamada “*Possible issues to watch out*” tenemos aquellos malos hábitos que también analiza Hairball, en orden de izquierda a derecha: código duplicado, objetos con nombres inadecuados, código muerto o que nunca se utiliza y objetos que no han sido inicializados.

La gestión de subida del fichero ya estaba implementada en la versión Alpha, pero presentaba un problema: si el **nombre del proyecto de Scratch contenía espacios o caracteres raros**,

el análisis mediante Hairball no se podía realizar. Se solucionó guardando el proyecto de Scratch en lugar de con el nombre original con el **id**.

Para implementar el análisis de Hairball, se necesitó la programación de una función llamada `analyze_project`, cuya tarea es abrir un pipe por cada plugin de Hairball que se ejecuta y guardar la información del plugin en una variable llamada `result [Nombre del plugin]`. Después, esa variable se procesa en cada una de las funciones creadas para cada plugin llamadas `proc_[Nombre del plugin]`, y va guardando en el diccionario la información que se muestra en el dashboard.

Además de lo descrito hasta ahora, en este momento también **se mejoró el flujo de la página web**, ya que en numerosas ocasiones el código mostraba páginas de error con páginas HTML sin estilo. Todas estas páginas fueron eliminadas realizando redirecciones para mejorar la experiencia del usuario. Y con ello se controló el flujo de la plataforma web.

Generación del fichero log

Una vez que el prototipo funcionaba perfectamente y con un flujo entre páginas aceptable, procedí a generar un fichero `logFile.txt` dentro de la carpeta `log`. Al principio únicamente **guardaba el nombre del fichero analizado y su id**. Poco a poco se ha ido añadiendo más información: hora del análisis, método utilizado, el tipo de usuario, el resultado del análisis, si ha habido algún error en el análisis... Cada vez que se analiza un proyecto este fichero es abierto, escrito en él la información indicada y cerrado.

Este fichero es muy útil para tener un resumen rápido de lo que está sucediendo en Dr. Scratch sin necesidad de consultar la base de datos.

Primera puesta en producción

Indico “primera” ya que se ha realizado en dos ocasiones de formas diferentes. Aquí voy a detallar cómo se realizó en esta etapa del proyecto.

■ *Entorno de pre-producción:*

En primer lugar, se utilizaron los *laboratorios de la universidad donde se creó una máquina virtual* en la que pude aprender a instalar: hairball, Apache, Django... y ponerlo todo

en funcionamiento. Tras realizar la configuración del fichero httpd.conf se podía visualizar la plataforma en la dirección **http://drscratchpre.programamos.es** donde hemos estado realizando las pruebas de Dr. Scratch antes de ponerlo en producción.

Actualmente esta dirección no está disponible, ya que la máquina en la que se alojaba ha sufrido problemas.

■ *Entorno de producción:*

En esta etapa del proyecto, el entorno de producción estaba alojado en otra *máquina virtual en la universidad, idéntica a la de pre-producción*. Así, una vez que aprendí de los errores cometidos, repetir las instalaciones y configuraciones en esta máquina fue mucho más fácil.

El proceso de puesta en producción durante todo el proyecto ha sido el mismo: primero se comprobaba y aprendía a subir todo al entorno de pre-producción y después se repetían los mismos pasos en el entorno de producción. Con esto *se han acortado inmensamente los momentos en los que la plataforma no ha estado funcionando*.

La dirección de Dr. Scratch era **http://drscratch.programamos.es**, pero actualmente se encuentra en **www.drscratch.org**.

En esta etapa, mientras producción estaba alojada en las máquinas de la universidad *se produjeron numerosos problemas, por algún corte y sobre todo por falta de memoria*.

Cambio de apariencia

El diseño gráfico del logo de Dr. Scratch fue subcontratado por Programamos a una empresa externa, *TheLogoCompany*, pero la idea estaba clara: debía ser con una tipografía atractiva para niños y que tuviera una bombilla. El motivo de la **bombilla** es porque *la idea esencial de Dr. Scratch es despertar la curiosidad por aprender a programar y además poder hacerlo aprendiendo con la filosofía de código abierto utilizando las ideas de otros para generar nuevas ideas y aprender de forma dinámica e interactiva*.

En este momento, ya que teníamos un logo, empecé a estudiar el front-end que existía hasta el momento y empecé a tomar contacto con Bootstrap, para incluir el logo y cambiar los colores de la web acorde al mismo. Siendo ahora la página principal de la web la mostrada en la *Figura 4.14*.



Figura 4.14: Primera apariencia de la página principal con logo

Y quedando la apariencia del dashboard como puede verse en la *Figura 4.15*.

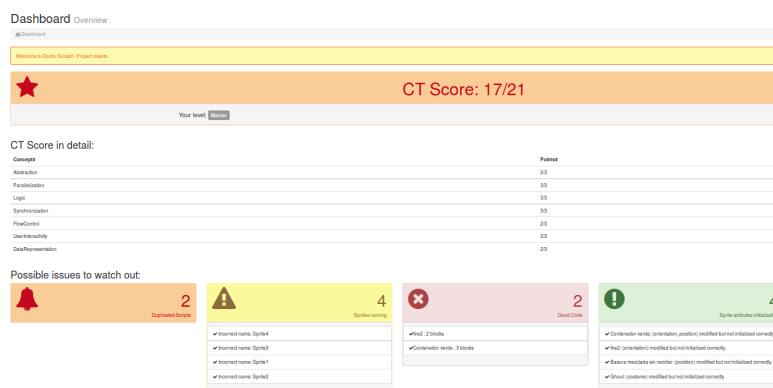


Figura 4.15: Dashboard con misma gama de colores que el logo

La web ahora se veía algo más alegre y atractiva para niños, pero mi conocimiento de Bootstrap aún era limitado. José Ignacio de Programamos me envió una plantilla de Bootstrap que consideró que tenía todas las características que estábamos buscando para Dr. Scratch y la web cambió por completo. Trabajé sobre esta plantilla, llamada *main.html*, incluyendo los textos y botones que necesitábamos y dejando las secciones que vi que nos eran más útiles siendo su apariencia final la mostrada en la *Figura 4.16*, que es la que a día de hoy se conserva con algunas pequeñas modificaciones.

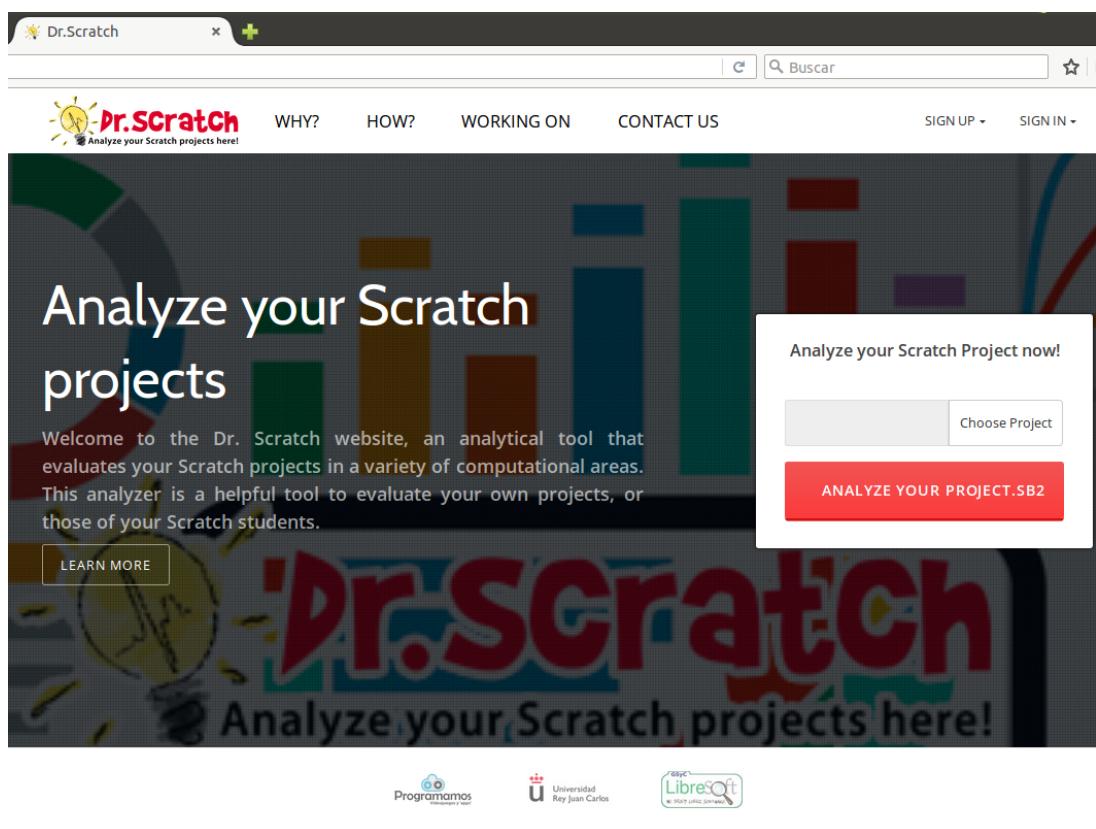


Figura 4.16: Página principal de Dr. Scratch

Esta página, además de la sección principal, donde se podía subir el fichero del proyecto de Scratch, quisimos ser más cercanos al usuario y se incluyeron 4 secciones más, que mostraban información en inglés sobre los siguientes aspectos:

1. **¿Por qué?** Explicamos brevemente cuáles son los puntos fuertes de Dr.Scratch y porque podría ser de utilidad para el usuario.Puede verse en la *Figura 4.17*.

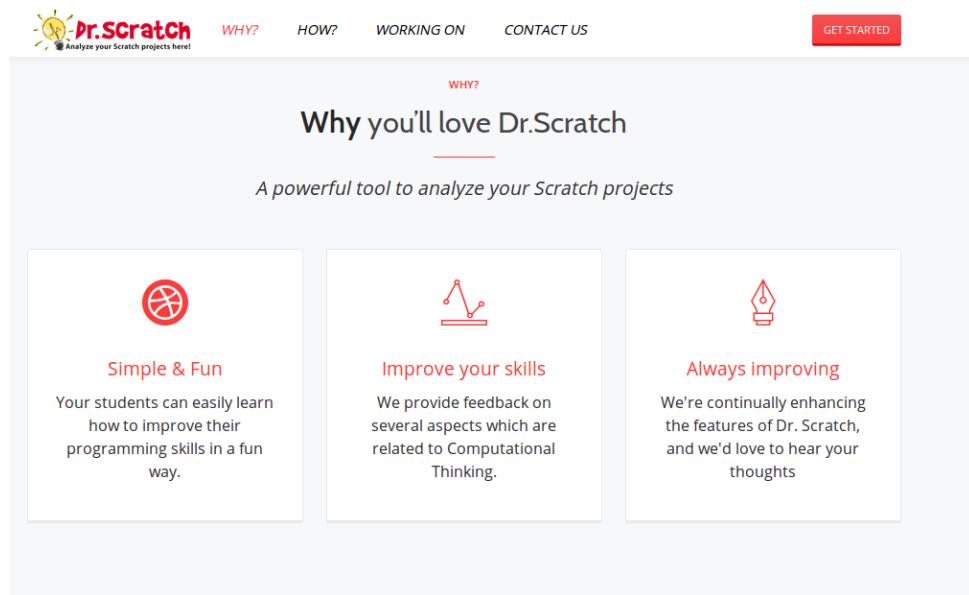


Figura 4.17: ¿Por qué te encantará Dr. Scratch?

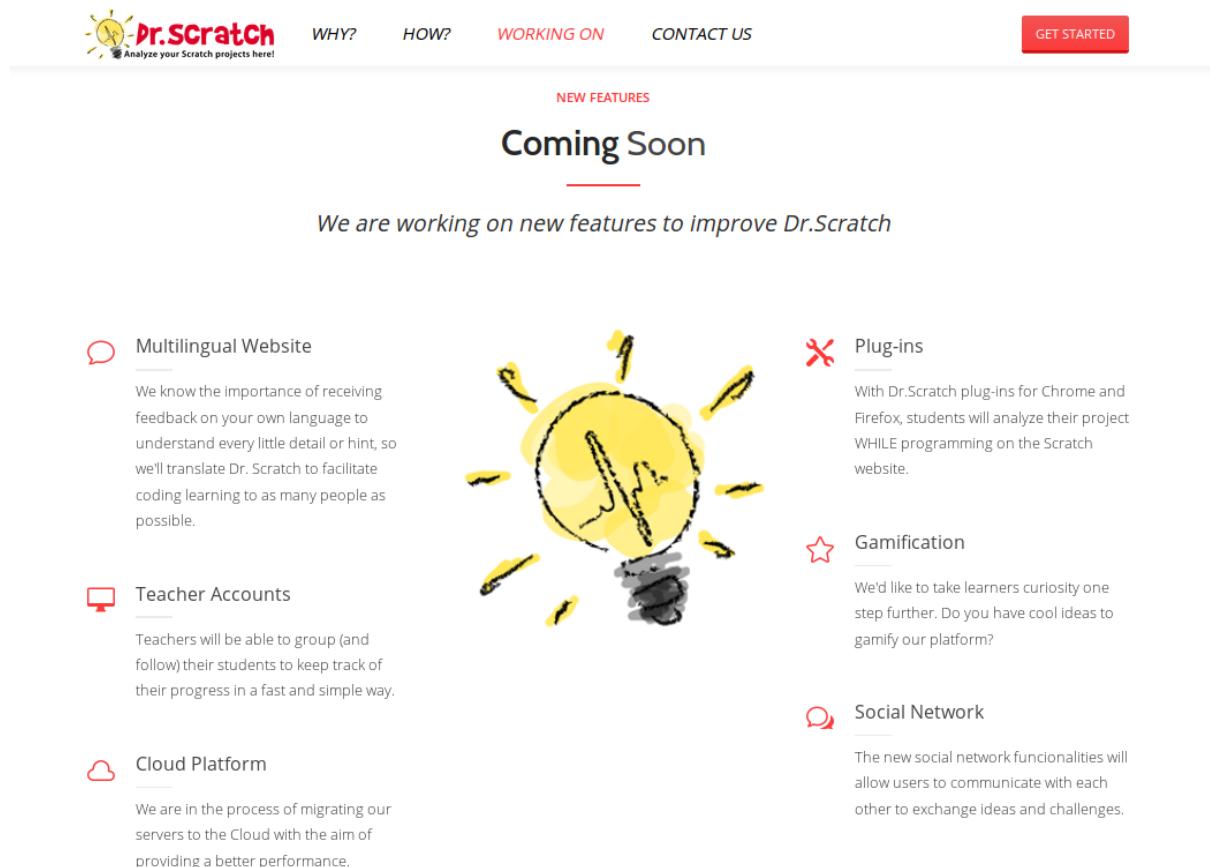
2. ¿Cómo? Mostramos mediante un vídeo y algo de texto cómo pueden empezar a probar Dr. Scratch analizando sus proyectos de Scratch. La Figura 4.18 muestra esta sección.

The screenshot shows the Dr. Scratch website's 'How it works' section. At the top, there is a navigation bar with links for 'WHY?', 'HOW?', 'WORKING ON', 'CONTACT US', and a red 'GET STARTED' button. Below the navigation, the page title 'How it works' is displayed, followed by the subtitle 'There are two options to analyze a Scratch project with Dr.Scratch'. Two options are listed: 'Choose Project' (represented by a file icon) and 'Url provided by Scratch' (represented by a URL icon). To the right, there is a screenshot of the Dr. Scratch interface showing a Scratch project with various scripts and costumes. Below the interface, there is a list of benefits:

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

Figura 4.18: ¿Cómo funciona Dr. Scratch?

3. Próximamente Indicamos cuáles eran las nuevas características que teníamos pensado incluir en la plataforma, mostrando nuestro deseo de querer recibir consejos, sugerencias y peticiones. Estas nuevas funcionalidades eran las que se pueden ver en la *Figura 4.19*.



The screenshot shows the Dr.Scratch website's "Coming Soon" page. At the top, there's a navigation bar with links for "WHY?", "HOW?", "WORKING ON", "CONTACT US", and a red "GET STARTED" button. Below the navigation is a section titled "NEW FEATURES" with the heading "Coming Soon". A sub-section below it says "We are working on new features to improve Dr.Scratch". The page is centered around a hand-drawn style illustration of a lightbulb with yellow rays emanating from it. To the left of the bulb, there's a list of five features, each with an icon and a brief description. To the right of the bulb, there's another list of three features, each with an icon and a brief description.

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

Figura 4.19: Nuevas funcionalidades de Dr.Scratch.

Al terminar de leer este documento, se verá que muchas de ellas han sido implementadas ya y deben ser actualizadas.

4. Contacto Esta sección incluye todas las vías que tiene el usuario de ponerse en contacto con nosotros, ya que nos parece indispensable tener información de nuestros usuarios, para poder hacer una herramienta que realmente sea útil. Dicha información es mostrada por la *Figura 4.20*.

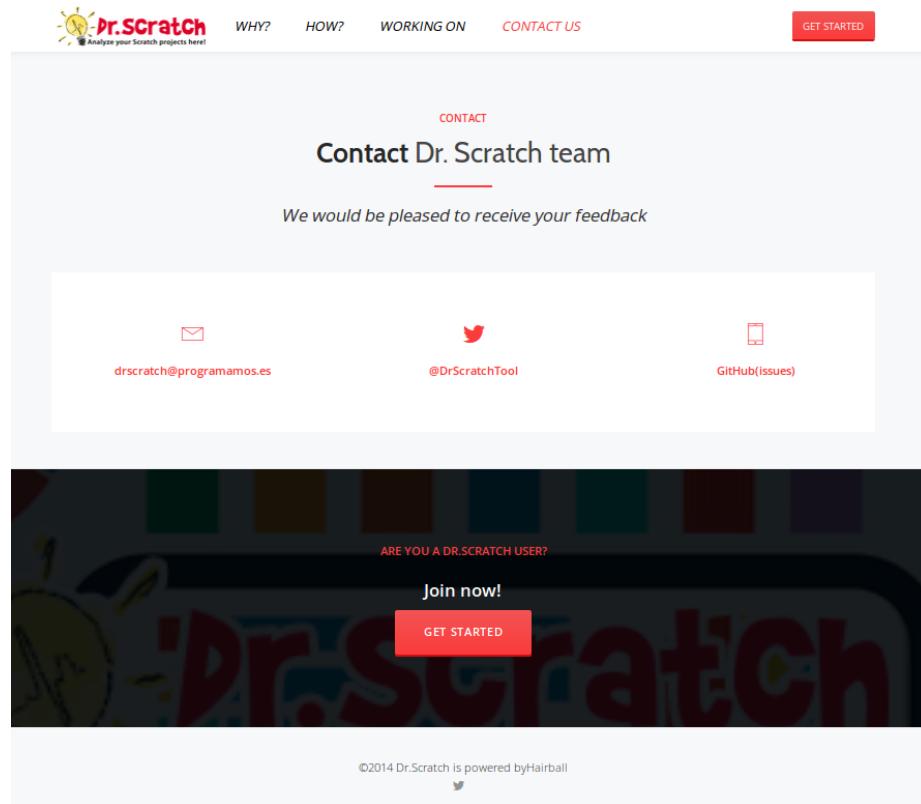


Figura 4.20: Contacta con el equipo de Dr.Scratch.

Mecanismo de traducción de Django

En este momento la página se mostraba por completo en inglés. Era necesario implementar la **traducción utilizando el mecanismo de Django** para tener la web en otros idiomas, en concreto, en español lo antes posible.

Este objetivo era bastante urgente porque queríamos empezar a utilizar la herramienta con docentes españoles y en colegios de Madrid.

Para lograr la traducción fue necesario realizar varias modificaciones en distintos ficheros del modelo de Django:

- Templates:
 1. Incluir en todos los templates la línea de código { % load i18n % }
 2. Introducir todo el contenido de la página que se quisiera traducir entre los caracteres { % trans “[contenido]” % }, tal y como se realiza en el ejemplo: { % trans “Analyze your Scratch projects” % }

- Settings.py:

Para realizar la configuración de la traducción fue necesario incluir en el fichero settings.py las líneas mostradas en la *Figura 4.21*.

```

LANGUAGE_CODE = 'en-us'

_ = lambda s: s

LANGUAGES = (
    ('es', _('Spanish')),
    ('en', _('English')),
)

LOCALE_PATHS = (
    os.path.join(BASE_DIR, 'locale'),
)

TIME_ZONE = 'UTC'

USE_I18N = True
USE_L10N = True
USE_TZ = True

```

Figura 4.21: Configuración para la traducción de Dr.Scratch.

- Crear una carpeta llamada “locale” dentro de nuestro proyecto.

- Ejecutar en la shell:

\$ python manage.py makemessages -l es

Con esto se generan dos ficheros django.po y django.mo y debemos modificar el primero.

Este fichero contendrá varias líneas como las siguientes y en las que se debe incluir el texto traducido:

- msgid “Analyze my project”
- msgstr “Analizar mi proyecto”

Una vez traducidos todos los contenidos, se ejecuta:

\$ python manage.py compilemessages

En esta etapa del proyecto, los únicos idiomas que estaban disponibles eran inglés y español. Más tarde se añadirían el catalán, gallego, griego, portugués y actualmente se está trabajando en el vasco.

Con las modificaciones detalladas, Django es capaz de detectar el idioma del navegador y nos mostrará la página de Dr. Scratch traducida si nuestro navegador está configurado en español y en inglés para cualquier otro idioma que detecte.

4.2.3. Fase II

En esta etapa Cristian retomó el proyecto -el cual presentó en junio- y Eva se incorporó. En la *Figura 4.22* se muestran las tareas que he desarrollado en esta etapa o en las que he realizado alguna aportación. Si la implementación no ha sido programada completamente por mí, indicaré cuál ha sido la aportación de mis compañeros y cuál la mía.



Figura 4.22: Fase II del proyecto Dr. Scratch

Barra de progreso

Con el cambio de apariencia de la página principal, había conseguido entender como funcionaba *Bootstrap*. Y me encontraba aprendiendo *JavaScript*, ya que había numerosos ficheros utilizados en la plantilla que estaban escritos en este lenguaje y lo iba a necesitar para seguir añadiendo funcionalidades a la web.

En concreto una de las funcionalidades que se querían añadir y que se necesitaba *JavaScript*, era la implementación de una **barra de progreso** que se mostrara mientras que se estuviera analizando el proyecto de Scratch. Así el usuario no veía una página blanca que podía dar la impresión de que la plataforma no funcionaba o se aburriera.

Esta tarea estuve trabajándola durante algún tiempo antes que mi compañera Eva se incor-

porará. Cuando ella comenzó, le mostré el código que había realizado y conseguimos conjuntamente llegar a la siguiente solución.

Se incluyó en la página main.html el código HTML, que ya tenía programado con la ayuda de Bootstrap para mostrar la barra con un estilo determinado, y se le asignó un id para poder diferenciar esta sección de código del resto de la página.

Con una función de JavaScript que se encuentra en el fichero progress-bar.js se programó que al mostrar la página se pudiera visualizar todo el contenido, excepto el de la barra de progreso, que puede verse en la *Figura 4.23*.



Figura 4.23: Barra de progreso mostrada durante el análisis

Al botón de análisis se le añadió también un id determinado, que estaría asociado a la función de JavaScript que se encargaría de ocultar el contenido de la página al hacer click en el botón y mostraría el contenido de la barra de progreso.

Análisis por URL

La idea de incluir también el análisis por URL surgió de la *peticIÓN realizada por varios usuarios de Dr. Scratch que mostraron la dificultad que tenían algunos de sus alumnos para descargarse el proyecto* de la página de Scratch.

Cristian se encargó de poner el código abierto de Nathan (getSb2) en Heroku² lo cual nos

²<https://www.heroku.com/>

dió numerosos problemas más tarde. Mediante el servidor getSb2 se realizan las peticiones de descarga al servidor de Scratch, en la petición se indica el identificador numérico que aparece en la URL del proyecto.

Y también realizó la implementación del análisis, pero repetía mucho código que ya se estaba utilizando en el análisis por la subida del proyecto. Por este motivo, modifiqué lo implementado por él para incluirlo al código.

Además añadí a la página la posibilidad de utilizar esta funcionalidad quedando el cuadro como se observa en la *Figura 4.24*.

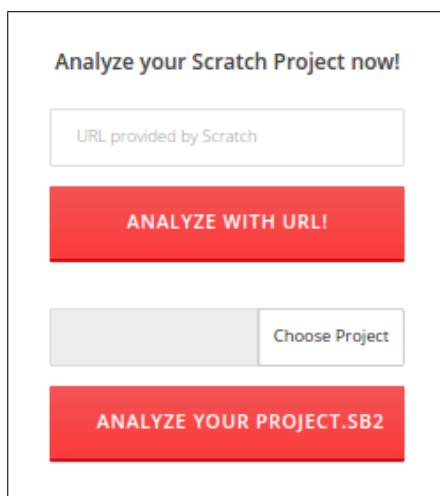


Figura 4.24: Cuadro con las dos posibilidades de análisis

Como la aplicación getSb2 subida a Heroku se caía cuando se realizaban numerosas peticiones simultáneas, tuve que **migrar el servidor Node.js a nuestra propia máquina de producción**.

Corrección de errores en el cuadro

El cuadro anterior permitía dar a cualquiera de los botones anteriores sin haber introducido ninguna información, por lo que la página nos mostraba error, ya que *intentaba analizar un proyecto inexistente*.

Para solucionar el problema, se utilizó una clase de Bootstrap que se encarga de gestionar este tipo de situaciones, mostrando la página el cuadro mostrado en la *Figura 4.25* cuando se pulsa alguno de los botones de análisis sin ningún proyecto concretado.



Figura 4.25: Gestión de errores en el cuadro de análisis

En el servidor esto se controla en la función `show_dashboard`, en ella se añade un flag según el error detectado y se renderiza con la plantilla `main.html` que se devuelve al cliente en la respuesta.

Migración de SQLite a MySQL

SQLite es la base de datos que viene por defecto al crear un proyecto en Django, pero MySQL es más profesional y más potente. Por este motivo, ahora que empezábamos a tener mayor número de usuarios, fue necesario realizar el cambio.

Para ello necesité realizar cambios en el fichero `settings.py` y aprender a utilizar las bases de datos de MySQL. Puede accederse desde la shell y se pueden realizar diversas acciones desde la misma: borrar tablas, ver el contenido de una tabla, eliminar algún usuario...

Carpetas para proyectos que producen error al ser analizados

En algunas ocasiones, bien Hairball o Kurt, *no son capaces de analizar un proyecto de Scratch concreto por diversos motivos que necesitan un estudio individual*.

Por ello, se implementó una página que fuera mostrada al usuario en esta situación, la cual puede verse en la *Figura 4.26*.



Figura 4.26: Página mostrada al producirse un error en el análisis

Además, se gestionó la *copia del archivo a otra carpeta* diferente a la que es utilizada para guardar todos los proyectos analizados.

Así tenemos a mano, sin que el usuario nos lo indique, todos los proyectos que han provocado error por no ser posible su análisis mediante Hairball.

Personalización de las páginas 404 y 500

A veces, se producen errores, porque el usuario solicita un recurso que no es servido por nuestro servidor. Como ya se ha explicado en Dr. Scratch esto está gestionado mediante redirección a la página principal para facilitar la usabilidad. Aún así se diseñó una página por si en algún momento era necesaria y puede verse en la *Figura 4.27*



Figura 4.27: Página de error 404

Otra página que si es más útil en nuestra plataforma, es la página de error que se muestra cuando se produce algún error en el servidor. Es muy desconcertante ver una página HTML sin estilo que le indica “500 Server error”. Por ello se diseño la siguiente página personalizada de error 500 que permite fácilmente volver a la página de inicio. Dicha página es la mostrada por la *Figura 4.28*.



Figura 4.28: Página de error 500

Análisis de proyectos de la versión 1.4 de Scratch

Al realizar el análisis subiendo el proyecto desde el propio ordenador, *algunos usuarios continúan utilizando la versión 1.4 offline de Scratch*. El JSON de estos proyectos es algo diferente al de la versión 2.0 y el análisis de Hairball devolvía error.

Fue necesario en el análisis por subida de fichero revisar el tipo de versión de Scratch, lo cual se hizo en la función `check_version()`, es muy simple porque:

- Los proyectos de la versión 2.0 tienen extensión .sb2
- Los proyectos de la versión 1.4 tienen extensión .sb

Y si la extensión corresponde a la versión 1.4 se realiza el cambio de versión, realizado en la función `change_version()`. Esto se consiguió gracias a una función de Kurt sencilla, `convert()`.

Servir páginas diferentes según el nivel computacional

Algunos docentes que estaban utilizando la herramienta en este momento, nos aconsejaron que tratáramos de *simplificar de algún modo el cuadro de resumen con las puntuaciones mostradas en el dashboard*. Consideraban que eran demasiado complicados para algunos alumnos.

El equipo de Dr.Scratch consideró oportuno probar a mostrar un resumen de la información diferente en función del nivel de Pensamiento Computacional obtenido.

Para ello, se modificó la función `show_dashboard()`, que hasta ahora únicamente comprobaba si había algún error en la subida del proyecto de Scratch o en la url. Si había algún error, activaba el flag concreto del error y lo renderizaba con el template `main.html` para enviarlo al usuario. Si no había error, mostraba el dashboard con toda la información del diccionario.

Este último paso es el que se vió modificado, mostrando ahora 3 dashboards distintos para diferenciar tres niveles de Pensamiento Computacional:

- Si el número de puntos devueltos por el plugin mastery de Hairball es mayor o igual a 15, muestra el `dashboard_master.html`
- En el caso en el que los puntos devueltos por el plugin mastery es mayor a 7 y menor que 15, la función nos muestra el `dashboard_developing.html`

- Y si la puntuación es menor o igual que 7, devuelve el dashboard_basic.html

La idea era mostrar diferentes dashboards, donde si el nivel obtenido era básico se mostraría muchísima menos información que si el nivel obtenido era alto, siendo el nivel medio algo intermedio.

Diseño de los dashboards

En producción se siguió manteniendo el resumen mostrado en la versión Alpha. Pero en preproducción, *se diseñaron tres dashboards distintos para cada nivel de Pensamiento Computacional*. De esta forma pudimos ver cuál era su aceptación en dos talleres que fueron realizados, uno con docentes y otro con niños de 5º de primaria.

El diseño de los dashboards fue realizado con poco tiempo, ya que queríamos probarlo para comprobar su aceptación y sacar conclusiones que fueron de gran utilidad. Además, era necesario realizar su traducción, para que tanto docentes como alumnos, pudieran realizar los análisis en castellano.

Como se va a ver a continuación, se me ocurrió además de hacer la distinción mediante el nivel, añadir colores diferentes ya que en muchos colegios utilizan el código de colores para motivar a los alumnos.

- **Dashboard si el pensamiento computacional es bajo:** se muestra, como se observa en la *Figura 4.29* tres secciones en la página.

1. Muestra el nivel de pensamiento computacional, con una frase motivadora y la puntuación total adquirida.
2. El cuadro inferior izquierdo indica cuáles son las habilidades de programación que debemos trabajar con más urgencia ya que son las que han obtenido una puntuación de 0. Además añadí un link junto a un ícono de una bombilla en la que al clicar redirigiera a otra página para aprender mediante ejemplos, cómo aumentar esa puntuación.
3. En el cuadro inferior derecho se muestra en un carrousel algunos trucos que pueden ser útiles para una persona que empieza con Scratch.



Figura 4.29: Dashboard de Pensamiento Computacional bajo

- **Dashboard si el pensamiento computacional es medio:** muestra otras tres secciones con algunos cambios que pueden verse en la *Figura 4.30*.

The screenshot shows the Dr. Scratch dashboard for a medium level user. It features a large "MEDIO" grade at the top. Below it, a message says "Estás haciendo un gran trabajo, ¡¡Sigue así!!!" and the score is "Puntuación: 12/21". On the left, a section titled "Algunos consejos:" shows a dark brown background with the text "Nombres Inadecuados" and a large number "1". On the right, a section titled "Puntuación en detalle:" provides a detailed breakdown of scores across various concepts. A "Subir" button is at the bottom right.

Concepto	Puntos
Secuenciar	0/3 <small>¡¡¡Buen trabajo!!!</small>
Lógica	0/3 <small>¡¡¡Aprende más...!</small>
Representación de los datos	1/3
Interactividad del Usuario	1/3
Paralelización	0/3 <small>¡¡¡Buen trabajo!!!</small>
Abstracción	2/3
Control del flujo	0/3

Figura 4.30: Dashboard de Pensamiento Computacional medio

1. La primera sección es idéntica al dashboard para nivel básico pero mostrando otra frase motivadora diferente.

2. El cuadro inferior izquierdo muestra en este caso en el carrousel, dos de los malos hábitos de programación que realizan todos los programadores al principio: nombres inadecuados y atributos inadecuados, por no haber sido inicializados correctamente. Y debajo del concepto se muestra el número de errores de cada tipo cometidos.
3. En el cuadro inferior derecho muestra ahora, todas las habilidades de programación, junto a la puntuación que han obtenido.

Y además, si la puntuación es de 3/3 se añade una estrella y la frase de **¡¡¡Bin hecho!!!** Y si la puntuación es de 0/3 se añade un link para mejorar como en el dashboard anterior.

La idea de la estrella, ha sido introducida con la intención de ser utilizada más adelante como un elemento de gamificación.

- **Dashboard si el pensamiento computacional es alto:** la información mostrada es la que se ve en la *Figura 4.31*.

The screenshot shows the Dr. Scratch dashboard interface. At the top, it says "Tu nivel es A LTO" (Your level is ALTO) and "¡¡¡Eres el master del universo!!!". The total score is "Puntuación: 11/21". Below this, there's a section titled "Puntuación en detalle:" (Score details) with a table:

Concepto	Puntos
Sincronización	3/3 ¡¡¡Bien hecho!!!
Lógica	3/3 ¡¡¡Bien hecho!!!
Representación de los Datos	2/3
Interactividad del Usuario	2/3
Paralelización	3/3 ¡¡¡Bien hecho!!!
Abstracción	2/3
Control del Flujo	2/3

Below the table is a section titled "Posibles cuestiones a tener en cuenta:" (Possible issues to consider) featuring a brick wall background with the text "Scripts Duplicados" and a red number "2". There's also a blue button labeled "Aprende Más" (Learn More).

Figura 4.31: Dashboard de Pensamiento Computacional alto

1. Como en los casos anteriores, lo primero que se muestra es el nivel alcanzado, con una frase motivacional y la puntuación total.

2. Y al igual que el caso anterior, se pueden ver todas las habilidades de programación, junto a la puntuación que han obtenido. Y si tienen 3/3 obtienen la estrella, mientras si la puntuación es de 0 se mantiene el link para aprender a mejorar.
3. La diferencia con el caso anterior es que ahora el carrousel contiene todos los malos hábitos analizados por Dr. Scratch y que deben corregirse al empezar a programar.

El resultado fue poco atractivo porque me centré en la programación de las plantillas de forma que cada una mostrará únicamente la información que deseábamos, ya que en la propia plantilla es donde se gestiona cuál es la información del diccionario que es mostrada al usuario.

Los docentes nos sugirieron algunas implementaciones, como poder descargar el un diploma y poder gestionar ellos mismos la información que se mostraba a sus alumnos. Y en el taller realizado con alumnos pudimos observar que no les llamaba nada la atención el carrousel con los malos hábitos de programación.

Modales explicativos en el cuadro de análisis

En estos talleres, notamos que muchos de los alumnos no sabían cómo utilizar el cuadro de análisis y que tenían los siguientes problemas:

- En el análisis mediante la URL no sabían qué es una URL.
- En el análisis subiendo el proyecto de Scratch desde su ordenador, ocurría que no sabían cómo descargar el proyecto en la plataforma de Scratch.

Actualizamos el cuadro al que se encuentra actualmente en producción. En él se añadió una breve explicación de cómo funciona cada análisis para que se vea claramente que son dos formas completamente distintas.

Hay palabras que están resaltadas en rojo debido a que eran los conceptos que necesitaban explicación extra. Para ello se programaron en la plantilla `main.html` dos ventanas modales, que aparecen al clicar en las palabras resaltadas en rojo.

El resultado final del cuadro de análisis se puede ver en la *Figura 4.32*.



Figura 4.32: Aclaración de los métodos de análisis

Dichas ventanas modales, se muestran al clicar en las palabras **url** y **proyecto**, teniendo al final dos ventanas diferentes:

- Una ventana explicativa de qué es una URL cuya apariencia puede verse en la *Figura 4.33*.

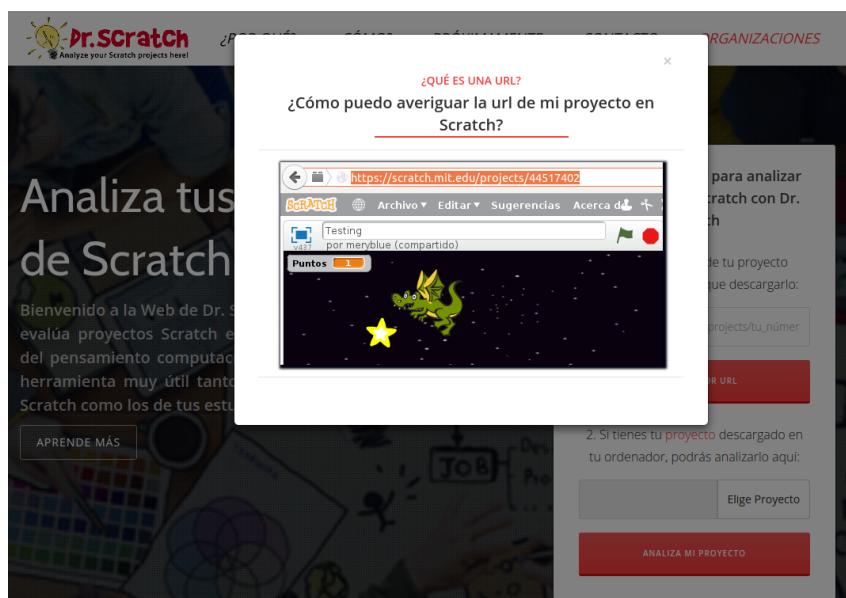


Figura 4.33: Ventana modal de qué es una URL

- Otra ventana que muestra cómo se puede descargar un proyecto a nuestro propio ordenador desde la plataforma de Scratch y así poder subirlo a Dr. Scratch para analizarlo. Esta ventana se muestra en la *Figura 4.34*.

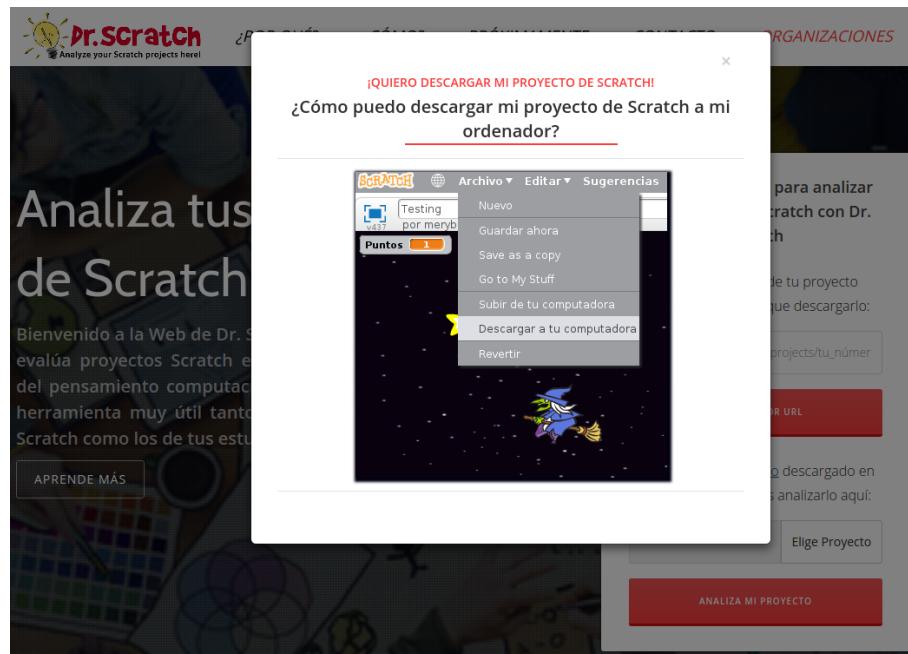


Figura 4.34: Ventana modal de cómo descargar un proyecto en Scratch

Migración a la nube: Microsoft Azure

Como ya se explicó en la Fase I del proyecto tanto el entorno de producción como el de preproducción se encontraban alojados en máquinas de la universidad, lo cual nos dió algunos problemas y debían ser solucionados.

Microsoft nos concedió una suscripción de 150\$ mensual para que pudiéramos alojar Dr. Scratch en la nube en su plataforma Azure.

Tras asistir a varios cursos de la plataforma en las instalaciones de Microsoft y de Plain Concepts, empezamos a tratar de buscar nuestra mejor solución. Ya que el crédito era limitado, al principio pensamos que era mejor utilizar la opción *website* de la plataforma. Esta decisión se tomó porque nos indicaron que esa opción consumía menos crédito.

Publiqué un website de Dr. Scratch con su base de datos MySQL enlazada a través de *Visual Studio* pero el problema que se encontró fue que **en los websites de Azure no hay posibilidad de instalar otras aplicaciones** y por ello no teníamos la posibilidad de instalar Hairball. Intentamos implementarlo ya que Python ofrece la posibilidad de ser ejecutado indicando el path completo en el que se encuentran los ficheros importando todos los módulos que había en los ficheros de Hairball y sus librerías de forma absoluta, pero había librerías dinámicas que presentaban más problemas.

En este momento, probamos a crear una máquina virtual instalando absolutamente todo lo que necesitábamos dentro de ella y controlar el consumo del crédito. La sorpresa fue que todos los meses, aún teniendo miles de proyectos analizados, el crédito *no se consumía en más de un 70 % del total*.

Gracias a esta migración Dr. Scratch no ha vuelto a sufrir problemas por cortes o por falta de memoria.

Google Analytics

Una de las tareas pendientes, era conseguir llevar un seguimiento de los usuarios que visitaban la plataforma. La opción website de Azure nos habría resuelto este problema ya que ofrece este tipo de estadísticas. Pero al alojar Dr. Scratch en una máquina virtual, tuve que buscar otro método.

El más sencillo y más recomendado que encontré leyendo varios artículos de analítica web, fue *Google Analytics*. Para ponerlo en marcha únicamente tuve que crear una cuenta en google para el equipo de Dr. Scratch y registrarla.

Facilitan un código de JavaScript que copié en el fichero `googleAnalytics.js` y lo incluí en todas las páginas de Dr. Scratch. Destacar que este fichero no ha sido subido a Github para que no sea utilizado por otras herramientas y se alteren las métricas.

Accediendo al panel se pueden observar numerosos datos como los que se ven en la *Figura 4.35*.

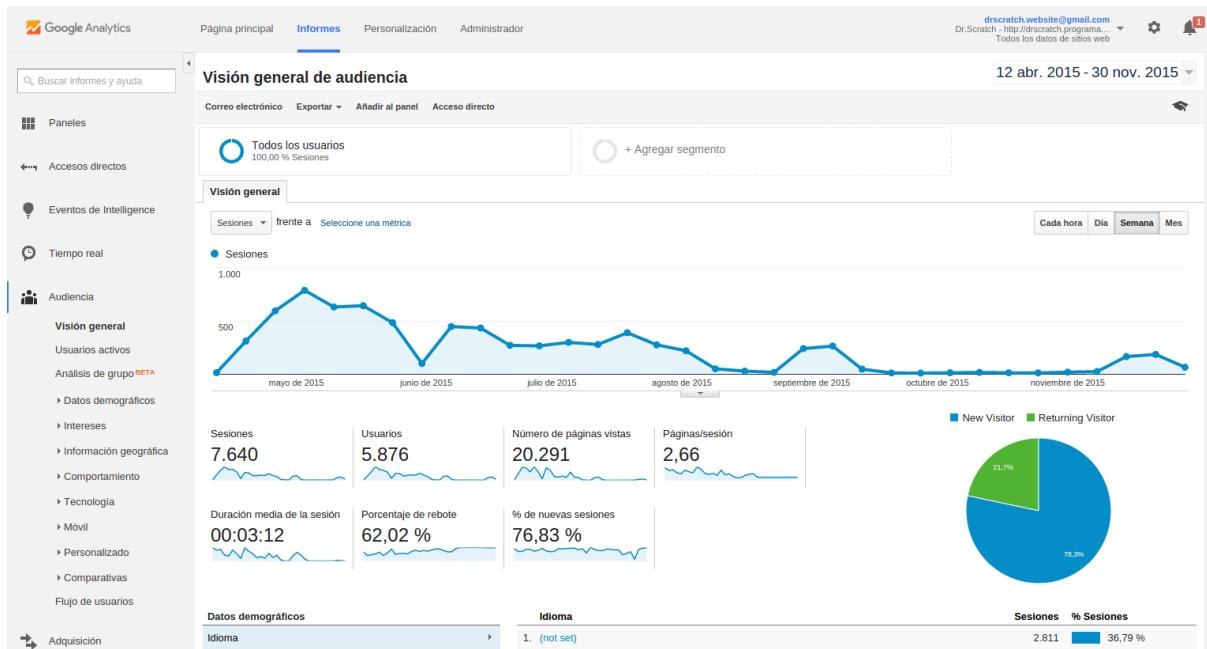


Figura 4.35: Panel de Google Analytics para Dr. Scratch

Páginas para mejorar el nivel computacional

En el diseño de los dashboard se explicó que eran necesarias la implementación de algunas páginas que mostrasen, mediante ejemplos, cómo mejorar en cada aspecto, para conseguir aumentar la puntuación de nuestro proyecto.

El diseño de dichas páginas fue realizado por mi compañera Eva. Mi aportación ha consistido en realizar la traducción de las palabras de las distintas habilidades: paralelismo, lógica, representación de la información, control de flujo... El mecanismo es el siguiente, cuando clicas en una de las habilidades, el servidor trata de ir al recurso /learn/[habilidad] y urls.py le indica que la función que debe ejecutar es learn(request, page).

Esta función lo que hace es ver en la petición, cuál es el idioma del navegador, si el idioma es uno de los traducidos por Dr. Scratch, crea un diccionario en el que incluimos como llave los nombres de las habilidades en el idioma que sea y como valor su traducción al inglés.

Al llamar a la función se le pasa como parámetro la página que queremos mostrar, que será una de las habilidades en cualquier idioma de los traducidos por Dr. Scratch, entonces se recorre el diccionario anterior y se obtiene su traducción al inglés y se redirige al usuario a dicha página.

Todas las plantillas HTML de aprendizaje se encuentran nombradas en inglés, por ello es necesario su traducción, sino tendríamos tantas plantillas repetidas como idiomas implementados para cada habilidad.

Nuevo diseño de los dashboards

Ya se ha mencionado que había que realizar cambios en las páginas mostradas al usuario tras realizar el análisis y que pudieron observarse con la implementación anterior. Por ello se decidió cambiar el diseño de la apariencia de lo cual se encargó mi compañera Eva. Se pueden apreciar los cambios en la *Figura 4.36*.



Figura 4.36: Dashboards actuales de Dr. Scratch

Mi aportación ha sido la traducción de las mismas y de la funcionalidad de twitter, que era diferente del resto. Para esta funcionalidad tuve que utilizar el mecanismo de Django: { % blocktrans % }[contenido]{ % endblocktrans % }, en lugar de utilizar el que venía usando desde el principio.

Cuenta de organizaciones

La implementación que quisimos tener realizada antes de llegar a la *2015ScratchConference* fue ofrecer la posibilidad de registrar organizaciones. Esta iniciativa surgió de la petición de varias organizaciones que realizan numerosos campamentos de programación y tienen cientos de proyectos que analizar en pocos días.

Presenté la plataforma en la conferencia e hice énfasis en la funcionalidad. Esto nos permitió lograr que varias organizaciones, además de las que habían solicitado el servicio, quisieran probar la funcionalidad y nos reportaran errores para seguir mejorándola.

Para el desarrollo de la funcionalidad se añadió en la página principal un botón rojo, en la parte superior derecha, que indica “ORGANIZACIONES”. Al clicar en dicho enlace, nos redirige al recurso /sign_up_organization que nos muestra la página de la *Figura 4.37*.



Figura 4.37: Página principal para organizaciones en Dr. Scratch

Haciendo scroll, tenemos un formulario para poder registrarnos y las instrucciones que debemos seguir para poder hacerlo. Todo esto se muestra en la *Figura 4.38*.

DR. SCRATCH PARA ORGANIZACIONES

Cómo disfrutar Dr. Scratch para organizaciones.

Si quieres ser una de las 10 primeras organizaciones que prueban Dr. Scratch envíanos un email a team@drscratch.org y te daremos una clave para poder registrarte en la web

Nombre de la organización
 ID
 E-mail
 Contraseña
 Clave

REGÍSTRATE!

EMPEZAR

team@drscratch.org

@DrScratchTool

GitHub(issues)

Figura 4.38: Formulario de registro para organizaciones

El mecanismo que hemos seguido para registrar organizaciones, ha sido muy controlado. Cuando nos escribían, introducíamos en la base de datos, con una página muy primitiva, los *hashkeys* que les indicábamos en el correo. Así podían registrarse y dicha hashkey se eliminaba de la base de datos para que no pudiera ser utilizada por otro usuario.

Así, hemos conseguido tener un control total de las organizaciones registradas y cononocer los errores que se han producido para solventarlos.

Una vez que se han registrado e ingresado en Dr. Scratch para organizaciones, se les muestra la página de la *Figura 4.39* y como se puede ver, además de los dos análisis por URL y por subida de proyecto, hay otro más.

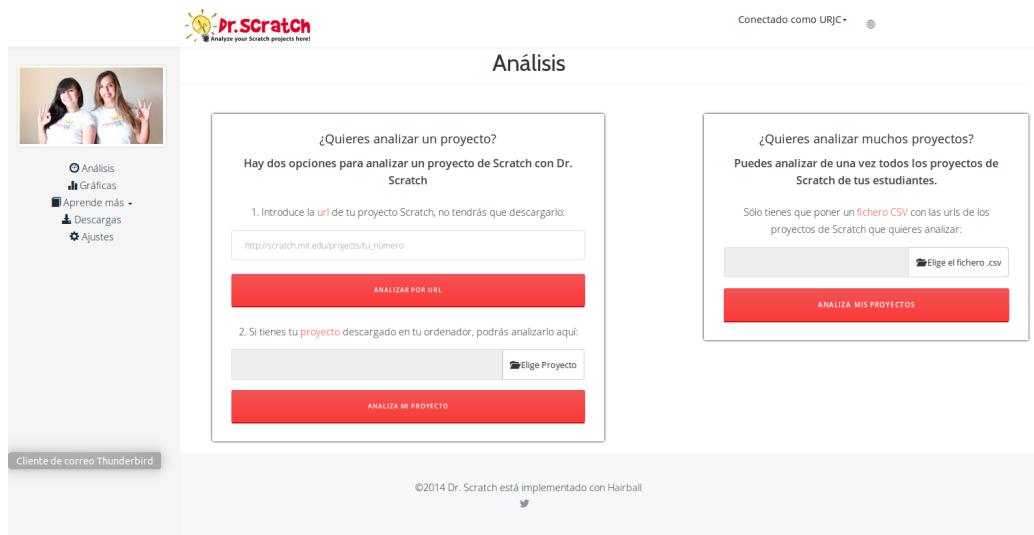


Figura 4.39: Dashboard de organizaciones

Este otro análisis permite subir un fichero CSV con un listado de proyectos Scratch a analizar. Se realiza el análisis de todos los proyectos y se devuelve la información obtenida en otro fichero CSV. Se admiten archivos CSV con 4 formatos de archivo diferentes:

- Archivo en el que se puede poner en una misma columna el número de identificador de los proyectos de Scratch.
- Otro que permite poner en una columna la URL completa de los proyectos.
- Fichero en el que hubiera una primera columna en la que hubiera un identificador para el alumno que ha creado el proyecto y otra columna para el identificador del proyecto.
- La misma idea anterior, pero con la URL completa del proyecto.

Por si hay alguna duda o confusión de cómo debe ser este archivo, programé una ventana modal que se muestra en esta página al clicar sobre la palabra **fichero CSV** y que puede verse en la *Figura 4.40*.

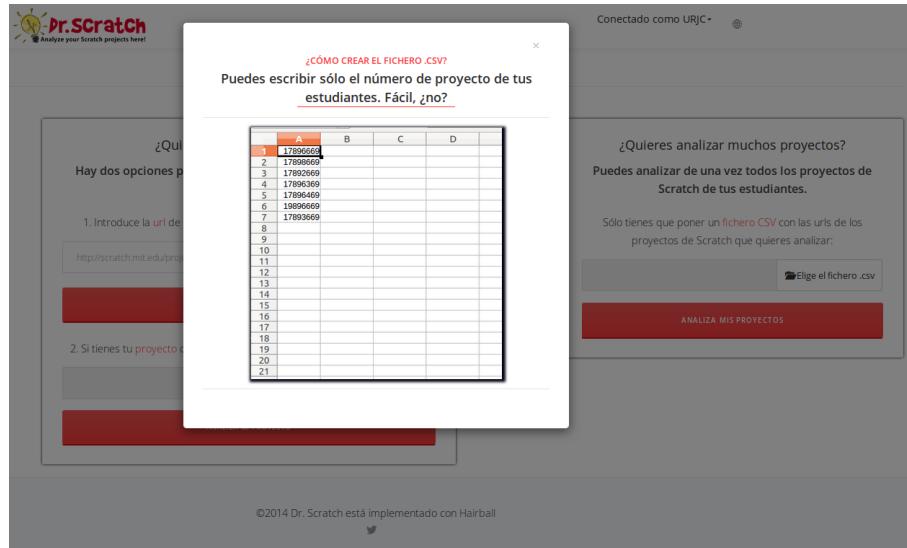


Figura 4.40: Ventana modal que explica cómo debe ser el fichero CSV

Señalar que se tuvo en cuenta para la implementación, que *podía producirse error en alguno de los proyectos al ser analizados por Hairball y podía hacer que el análisis de todo el fichero se detuviera y mostrara error*. Por ello, se trató dicha situación como una **excepción** y en el fichero CSV que se devuelve, muestra qué proyectos han tenido problemas para ser analizados, pero permitiendo tener una salida para el resto de proyectos.

El fichero que se devuelve y que el usuario puede descargarse tiene la estructura que se muestra en la *Figura 4.41*, con una columna para el identificador del alumno -si es que se indicó esta columna-, otra para el identificador del proyecto y después el resto de columnas con toda la información obtenida con Hairball.

CODE	URL	Mastery	Abstraction	Parallelism	Logic	Synchronization	Flow control	User interactivity	Data representation	Duplicate scripts	Sprite naming	Dead code	Sprite attributes
student2	17895669	Error analyzing project											
student12	44517402		14	2	2	1	2	2	3	2	1	0	1
student7	98960349	Error analyzing project											
student8	98033154		15	2	1	3	2	3	3	1	0	1	0
student5	83135936		19	2	3	3	3	3	2	2	0	1	6
student1	17896669		7	2	1	0	1	1	1	1	0	0	1
student6	94188004		12	2	3	0	2	2	2	1	0	1	1
student9	97871099	Error analyzing project											
student4	100353773		17	2	3	3	3	2	2	1	4	1	3
student3	100344829		1	0	0	0	0	1	0	0	1	0	0
student11	90523739		1	0	0	0	0	1	0	0	0	0	0
student10	94365058		12	2	3	0	2	2	1	0	0	0	1

Figura 4.41: Fichero CSV con toda la información del análisis

Se puede ver cómo, algunos proyectos han sufrido algún error durante el análisis y por ello muestran la frase “Error analizando el proyecto”, pero se ha podido seguir analizando el resto de proyectos.

Estos proyectos que sufren error son guardados en una carpeta diferente, para poder estudiar cuál es el error que se produce en el análisis con Hairball o si es debido a algún problema con Kurt.

4.2.4. Fase III

Después de la Scratch Conference, pudimos obtener muchísimo feedback gracias a todas las organizaciones y usuarios que quisieron probar la plataforma. Y realizamos nuevas implementaciones, gracias a las observaciones que nos realizaron y otras que aún teníamos pendientes y que pueden verse en la *Figura 4.42*.



Figura 4.42: Fase III del proyecto Dr. Scratch

Cambio en getSb2: descargar únicamente el JSON

Una de las primeras observaciones que nos hicieron llegar las organizaciones, fue que *el análisis del CSV tardaba mucho tiempo* en realizarse y en algunas ocasiones terminaba mostrando error.

Se decidió *modificar el servidor Node.js* que nos permitía descargar los proyectos desde la plataforma de Scratch a nuestro servidor. De forma, que no descargara el proyecto completo, sino únicamente el fichero JSON, ya que es lo único que necesitamos para realizar el análisis con Hairball.

Comencé a estudiar el código del *servidor getSb2* y conseguí modificarlo, para que efectivamente, únicamente se descargara el fichero JSON.

La velocidad de análisis mediante esta opción aumentó enormemente y desaparecieron los errores.

Al modificar el *getSb2* se afectó también al análisis mediante URL el cual se realiza a mayor velocidad y con menor tráfico. Esto es perfecto para el consumo del crédito de Azure.

Traducción de los bloques “pintados”

Al empezar a traducir la plataforma a otros idiomas me encontré con un problema que no se había detectado anteriormente: los bloques que se muestran en las páginas para aprender más y poder mejorar en las distintas habilidades, no se traducían correctamente a idiomas que no fueran el español o el inglés.

Esto tiene una explicación, mi compañera Eva había tratado de obtener el idioma para traducir los bloques con JavaScript con la función mostrada en la *Figura 4.43*.

```
<script>
$(document).ready(function() {
    scratchblocks2.parse();
});

var choose_lang = x=window.navigator.language||navigator.browserLanguage;
choose_lang = choose_lang.split("-")[0];

scratchblocks2.reset_languages();
var lang_dict = scratchblocks2._translations[choose_lang];

if (lang_dict) scratchblocks2.load_language(lang_dict);
</script>
```

Figura 4.43: Traducción fallida de los bloques de ScratchBlocks

Al observar por consola, cuál era la salida de la línea que detecta el idioma vi que siempre era la misma: *es(español)*.

Lo que estaba pasando, era que cuando el idioma del navegador era inglés(en), Django no activaba el mecanismo de traducción, por lo que funcionaba perfectamente. Y cuando el idioma era español(es), se activaba el mecanismo de traducción de Django y dicha línea obtenía algo coherente. Pero cuando se implementó para el catalán, se activaba el mecanismo de traducción pero esa línea devolvía español(es).

Tuve que cambiar la función por otra que utiliza AJAX, para poder llamar en segundo plano a una función que implementé en nuestro servidor. Esta función llamada `blocks()` devuelve correctamente el idioma en el que se realiza la petición para que la traducción funcione. El cambio puede apreciarse en la *Figura 4.44*.

```
$.ajax({
    url: "http://www.drscratch.org/blocks",
    dataType: 'jsonp',
    success: function(headers) {
        language = headers['Accept-Language'];
        language = language.split(",")[0];
        scratchblocks2.reset_languages();
        if (language) var lang_dict = scratchblocks2._translations[language];
        if (lang_dict) scratchblocks2.load_language(lang_dict);
        $(document).ready(function () {
            scratchblocks2.parse();
        });
    }
});
```

Figura 4.44: Traducción de los bloques de ScratchBlocks mediante AJAX

En el servidor añadí al fichero `urls.py` una línea para que se admitiera el recurso `/blocks` el cual esta asociado a la función `blocks()`. Esta función cambia el contenido de la variable `headers['Accept-Language']` por lo almacenado en `request.LANGUAGE_CODE` que es el idioma en el que está configurado el navegador que realiza la petición.

Registro de usuarios

Una vez que se había realizado el registro de organizaciones, parecía que el registro de usuarios iba a ser fácil. Y lo que es el registro es idéntico al de organizaciones. Por lo que ahora mismo en nuestro servidor tenemos **organizations** y **coders**. Se eligió la palabra *coder* en lugar de *user*, porque en Django ya hay un objeto user que nos ha sido muy útil. Ya que tanto organizaciones como programadores, extienden de este objeto usuario con campos extras mediante **herencia**.

Pero además, tuve que añadir más código en todo el servidor, para controlar constantemente si el usuario que estaba registrado era una organización o un programador. Esto lo he hecho mediante una función que se llama `segmentation()` y que es llamada en numerosas partes del código para decidir qué página es mostrada y el contenido de la misma.

El objeto user de Django tiene campos y funciones implementadas que hemos utilizado tales como:

- `User.objects.filter(username = username)`
- `request.user.is_authenticated()`
- `request.user.username`
- `user.set_password`

Lo próximo será tener **students**, **teachers** y **classrooms**, que se extenderán de coder con los campos añadidos que sean necesarios, quedando el árbol de objetos como muestra la *Figura 4.45*.

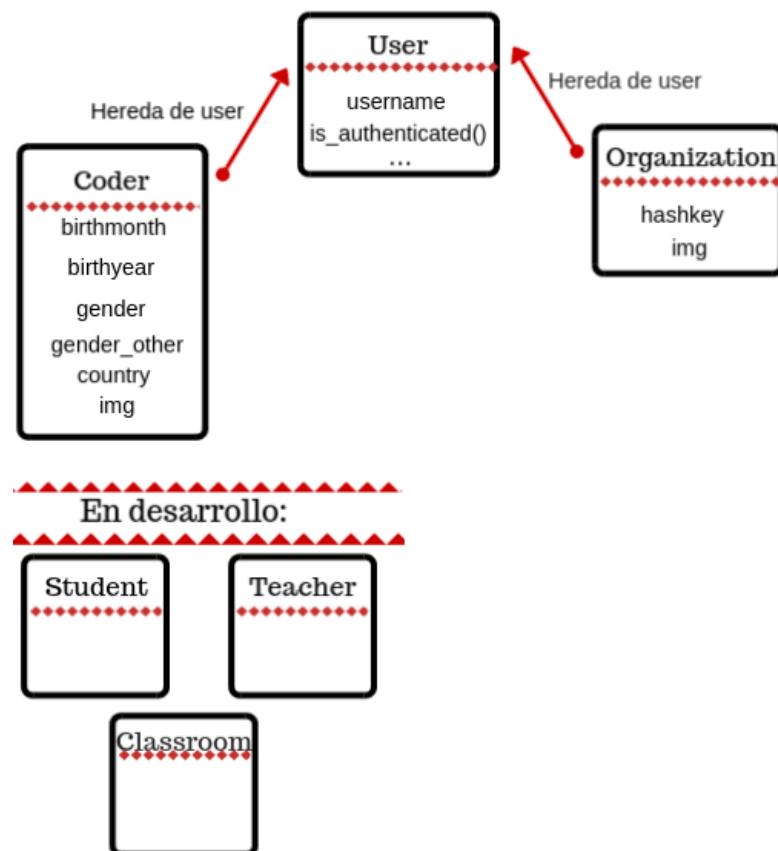


Figura 4.45: Árbol de usuarios de Dr. Scratch

De esta forma, *tenemos en la base de datos los distintos tipos de usuarios clasificados y es más fácil realizar un control de los mismos*.

Traducción dinámica: bola del mundo

Otra de las peticiones que nos realizaron fue la posibilidad de cambiar de idioma sin necesidad de modificar la configuración del navegador.

Incluí en la página principal de Dr. Scratch el icono de una bola del mundo que al clicar en ella muestra un desplegable con los idiomas que están disponibles actualmente. Esto puede verse en la *Figura 4.46*.



Figura 4.46: Traducción dinámica de la web: bola del mundo

Para implementar esta funcionalidad he utilizado la vista de redirección `set_language`.

Al elegir un idioma y clicar en el botón “**Cambiar idioma**” se pide al servidor el recurso `/i18n/setlang/` y se observa en el fichero `urls.py` la siguiente línea:

```
url(r'^i18n/', include('django.conf.urls.i18n')), la cual activa la vista.
```

Una curiosidad es que aunque es un formulario, la vista `set_language` espera ser llamada con un **método GET** y no POST como es común en los formularios.

El mecanismo interno que realiza la vista es el siguiente:

1. Fija la preferencia de idioma al que se ha seleccionado en el parámetro *language*.
2. La vista guarda dicho idioma en la sesión del usuario.
3. Redirecciona al usuario a la misma página.

Extensión para Chrome

Como ya hemos visto, los niños a veces tienen problemas para identificar cuál es la URL que deben introducir en el cuadro de análisis o cómo descargarse el proyecto a su ordenador para subirlo después a Dr. Scratch y analizarlo.

Para poder facilitar esta tarea, tanto a niños pequeños como a personas con discapacidad intelectual, pensamos en realizar un **plugin** que permitiera realizar el análisis desde la propia página de Scratch.

El plugin que he realizado es una *extensión para el navegador Chrome*, que una vez instalado, como puede verse en la *Figura 4.47*, se ve en la esquina superior derecha del navegador con el símbolo característico de Dr. Scratch: la bombilla.

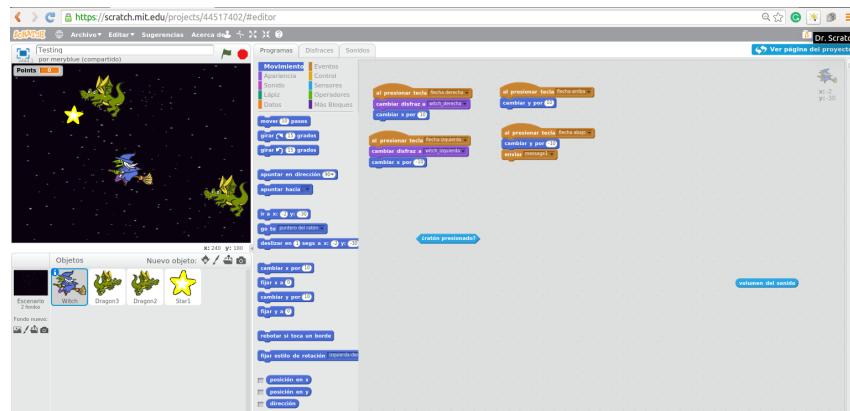


Figura 4.47: Botón de la extensión

Al pulsar en el ícono, nos muestra el desplegable que vemos en la *Figura 4.48*, en el cual ya se ha insertado la URL actual sin la parte final (#editor). Nos permite pulsar directamente en el botón ANALYZE! que abrirá otra pestaña y nos mostrará el resultado del análisis.

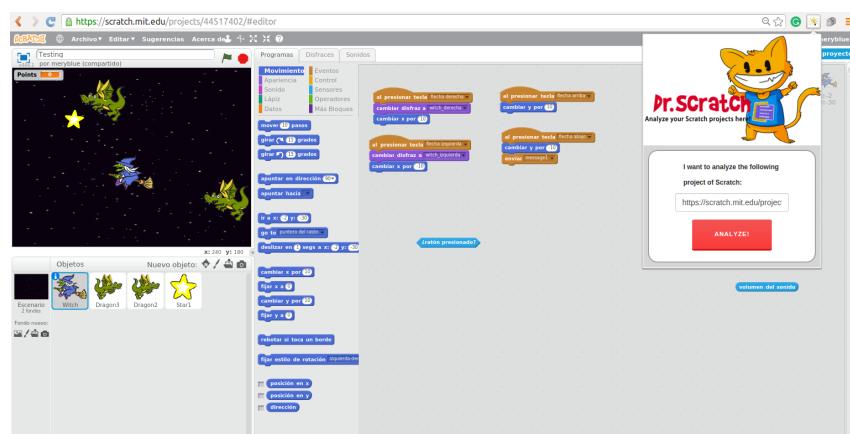


Figura 4.48: Extensión de Dr. Scratch para Chrome

Destacar que se ha dejado la *opción de poder modificar el contenido de la URL por si se quiere utilizar el plugin para analizar diversos proyectos*, en lugar de utilizar la página principal.

Traducción de los contenidos a más idiomas

Para lograr la traducción de más idiomas, intentamos motivar a los usuarios de Dr. Scratch a ser los propios traductores. A aquellos usuarios que han querido participar en la traducción, se les envió el fichero *django.po* de su respectivo idioma, que fue generado ejecutando en la shell:

```
~$ python manage.py makemessages -l [código_idioma]
```

Donde el código del idioma ha sido:

- es: para español.
- ca: para el catalán.
- gl: en el caso del gallego.
- pt: para el portugués.
- el: ha sido utilizado para el griego que estará próximamente en producción.

Y el fichero *settings.py* ha sido modificado para añadir todos ellos, como puede observarse en la *Figura 4.49*.

```
LANGUAGE_CODE = 'en-us'
_ = lambda s: s

LANGUAGES = (
    ('es', _('Spanish')),
    ('en', _('English')),
    ('ca', _('Catalan')),
    ('gl', _('Galician')),
    ('pt', _('Portuguese')),
    ('el', _('Greek')),
)

LOCALE_PATHS = (
    os.path.join(BASE_DIR, 'locale'),
)

TIME_ZONE = 'UTC'

USE_I18N = True
USE_L10N = True
USE_TZ = True
```

Figura 4.49: Modificación en settings.py para traducir a más idiomas

Página de agradecimientos

Para agradecer a todos los colaboradores, tanto en las traducciones, como a los colegios que nos han permitido realizar talleres en sus centros, diseñé una página de agradecimientos, con las siguientes secciones:

- **Principal:** en la que se muestra el logo de Dr. Scratch y damos las gracias y explicamos la finalidad de esta página, la cual puede verse en la *Figura 4.50*.



Figura 4.50: Agradecimiento a todos nuestros colaboradores

- **Quiénes somos:** tratando de ser cercanos, para invitar a que todo el mundo siga participando en la creación de la web.



Figura 4.51: Quiénes somos

- **Traducciones:** como se ve en la *Figura 4.52* se muestra una foto de aquellos colaboradores que quieren aparecer en la web, indicando sus nombres y a qué idioma han realizado la traducción, así como una pequeña biografía.



Figura 4.52: Colaboradores que han participado en la traducción de Dr. Scratch

- **Talleres:** como se ve en la *Figura 4.53* se muestra una foto de aquellos colegios en los que se ha realizado algún taller para poder realizar test de usabilidad con alumnos de 5º y 6º de primaria.

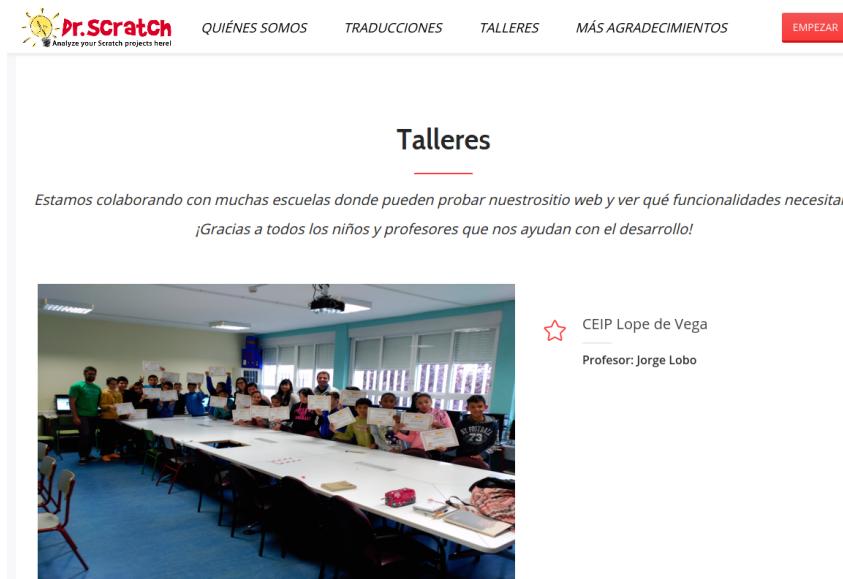


Figura 4.53: Colegios colaboradores

Botones en la página principal: estadísticas, colaboradores y blog

Debido a que se implementaron nuevas páginas y no había forma de acceder a ellas. Se añadieron a la página principal, en la sección de contacto, 3 botones para poder visualizarlas. La *Figura 4.54* muestra como queda dicha sección tras la modificación.



Figura 4.54: Acceso a las páginas de estadísticas, colaboradores y al blog

Los botones de estadísticas y colaboradores redireccionan a las páginas /statistics y /collaborators, respectivamente. Y el fichero urls.py indica que se encuentran enlazadas con las funciones statistics() y collaborators() que muestran las páginas statics.html y collaborators.

Pero la redirección al blog se realiza mediante un mecanismo especial de Django, para ello se añadió al fichero urls.py la siguiente línea:

```
url(r'^blog$',  
    RedirectView.as_view(url='https://drscratchblog.wordpress.com'))
```

Modificación del procesador del plugin Dead Code

Como comenté, en la Fase I se reprogramaron todos los procesadores de los plugins de Hairball. Recordar que entonces, la salida del plugin Dead Code era la mostrada en la *Figura 4.55*.

```
Dead code:
meryblue@malu-XPS-L701X:~$ hairball -p blocks.DeadCode Trash-2.sb2
Trash-2.sb2
{u'Contenedor verde': [kurt.Script([
    kurt.Block('forward:', 10),
    kurt.Block('forward:', 10)], pos=(381, 257)),
    kurt.Script([
        kurt.Block('pointTowards:', u")], pos=(653.35, 345.75))],
u'fire2': [kurt.Script([
    kurt.Block('forward:', 10),
    kurt.Block('turnRight:', 15)], pos=(549.5, 249.9))]}
```

Figura 4.55: Salida antigua del plugin Dead Code

Pero se realizaron algunos cambios en el plugin y la salida ha pasado a ser la que se observa en la *Figura 4.56*.

```
Dead code:
meryblue@malu-XPS-L701X:~$ hairball -p blocks.DeadCode Trash-2.sb2
Trash-2.sb2
{u'Contenedor verde': [['move %s steps', 'move %s steps'],
    ['point towards %s']],
u'fire2': [['move %s steps', 'turn @turnRight %s degrees']]}
```

Figura 4.56: Salida actual del plugin Dead Code

Por este motivo, el procesamiento que realicé en la función `proc_dead_code()` quedó obsoleto.

Mi compañera Eva intentó solucionarlo, pero para ello hizo uso de la función `ast.literal_eval()`. Esta función, lo que hace es convertir líneas en conjuntos **clave:valor** de un diccionario. Esto estaría bien, si la salida de Dead Code pudiera convertirse en todos los casos, pero no es así. Cuando un en un mismo objeto, hay varios conjuntos de bloques que no se ejecutan nunca, hay líneas en la salida que no muestran el objeto, tal como puede verse en el caso del *Contenedor verde*.

La forma que en un principio propuse para solucionarlo y poder contabilizar los bloques que no se ejecutaban era haciendo uso de la función `split()` pasándole como parámetros las cadenas claves `kurt.Block` y `kurt.Script`.

Como estas cadenas ya no aparecen en la salida, he tenido que modificar el procedimiento inicial, diferenciando entre:

- Líneas en las que aparece el carácter `:` en las cuales utilizo `split()` con los argumentos: `', [[y]]`.
- Líneas en las que no está el carácter `:` que se puede obtener la información utilizando `split()` pasándole como parámetros: `[y]`.

Todos esto se introduce en un `for` que recorre todas las líneas de la salida del plugin y va contabilizando mediante contadores el número de bloques que no se ejecutan. Y además, también se va creando un lista de cadenas en las que se indica qué objetos tienen bloques que no tienen ningún evento que provoque que se ejecuten y cuáles son esos bloques.

Toda la información procesada es guardada, tal como se hacía ya, en el diccionario que se pasa a la correspondiente plantilla HTML que muestra cada uno de los 3 dashboards.

Formulario de registro para usuarios

La forma de registro de las organizaciones, fue muy controlada mediante un **hashkey** que debían solicitarnos por correo electrónico y que nosotros introducíamos en la base de datos con la ayuda de una página muy primitiva.

Pero esta forma de registro no es la ideal y ya solventados los problemas de registro, he diseñado un formulario del mismo estilo que el que se realiza en Scratch, ya que hay que tener en cuenta que los usuarios pueden ser menores de edad.

La apariencia del formulario se puede ver en la *Figura 4.57*.

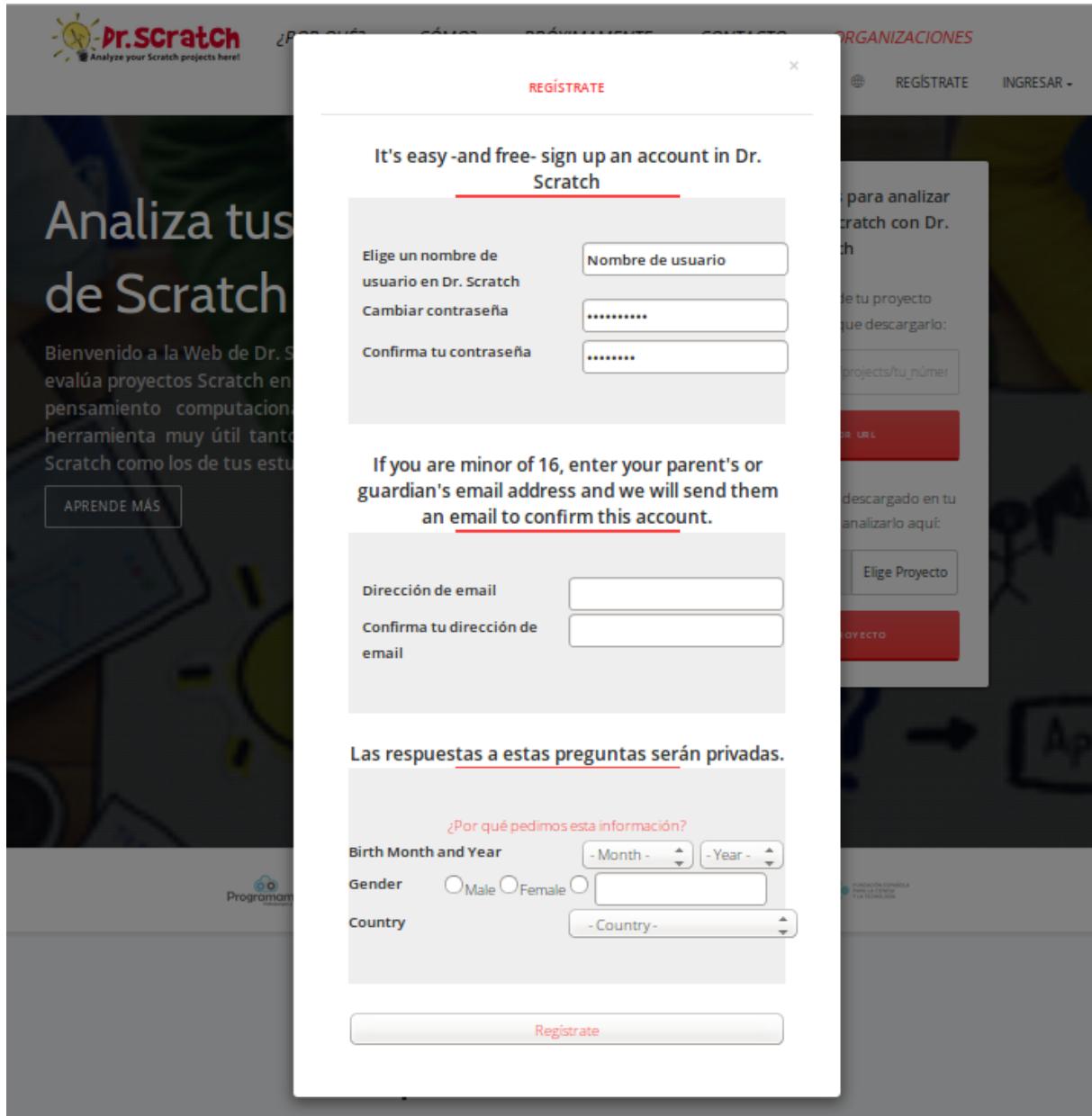


Figura 4.57: Formulario para el registro de usuarios

Cuando un usuario clica en el botón *Regístrate* se hace una **petición POST** con el recurso `/sign_up_coder`. Entonces, el fichero `urls.py` enlaza con la función `sign_up_coder()`.

En la `sign_up_coder()` se hace uso del mecanismo de formularios de Django para validar el formulario. Esto se ha realizado en todos los formularios utilizados en la plataforma, pero no se ha explicado hasta ahora para no repetir información. Esto consiste en indicar en el fichero `forms.py` los campos que debe tener un formulario para ser correcto, si alguno de los campos no coincide la función `form.is_valid()` devolverá FALSE.

Si el formulario es válido, obtendremos la información del mismo para comprobar que los campos de email y email_confirm coinciden, del mismo modo que password con password_confirm. En caso contrario, se mostrará una página indicando el error cometido. Esto último se realiza mediante distintos flags. Lo mismo ocurre si el nombre de usuario ya existe o intenta utilizar una dirección de correo que ya ha sido registrada.

Si el registro se realiza con éxito, se realiza la autenticación y login del usuario de forma automática y se le redirige al recurso /coder/ (username) que será su propio espacio que puede verse en la *Figura 4.58*.

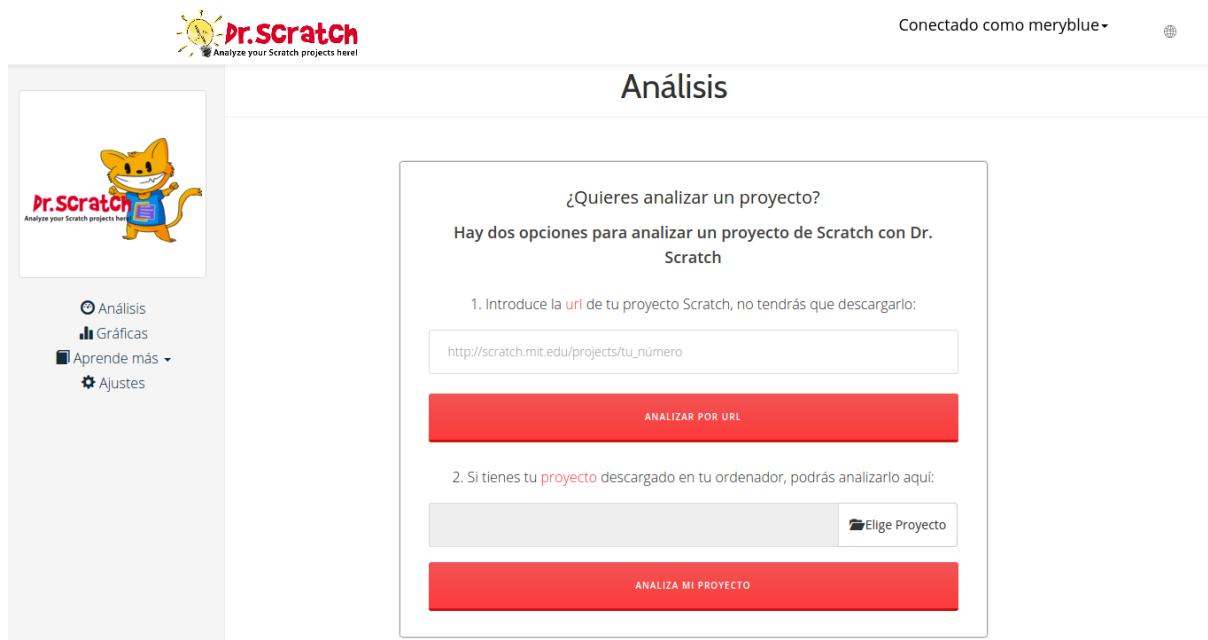


Figura 4.58: Dashboard de programadores

Este dashboard, aún está en desarrollo. La idea es que el usuario pueda ver más información de la mostrada, sobre los proyectos que ha analizado con anterioridad, además de incluir mecanismos de gamificación.

4.2.5. Diagramas de flujo

Con el objetivo de clarificar cómo es el interior del servidor y cuál es el flujo entre las distintas funciones implementadas, voy a mostrar 3 diagramas de flujo para explicar los distintos tipos de análisis.

Análisis subiendo el proyecto: upload

En el diagrama de flujo de la *Figura 4.59* se explica el uso y la interconexión de las funciones que intervienen en el análisis de proyectos de Scratch mediante la opción de subida del archivo SB2 desde nuestro propio ordenador.

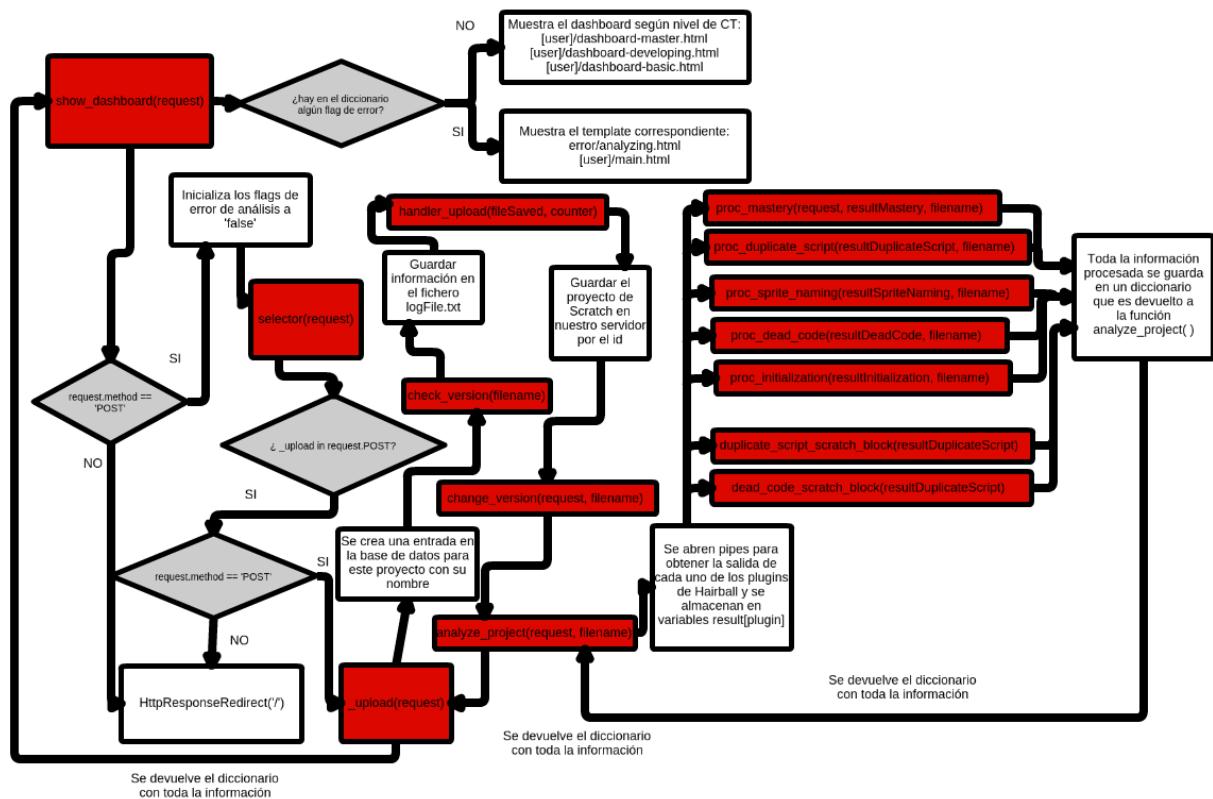
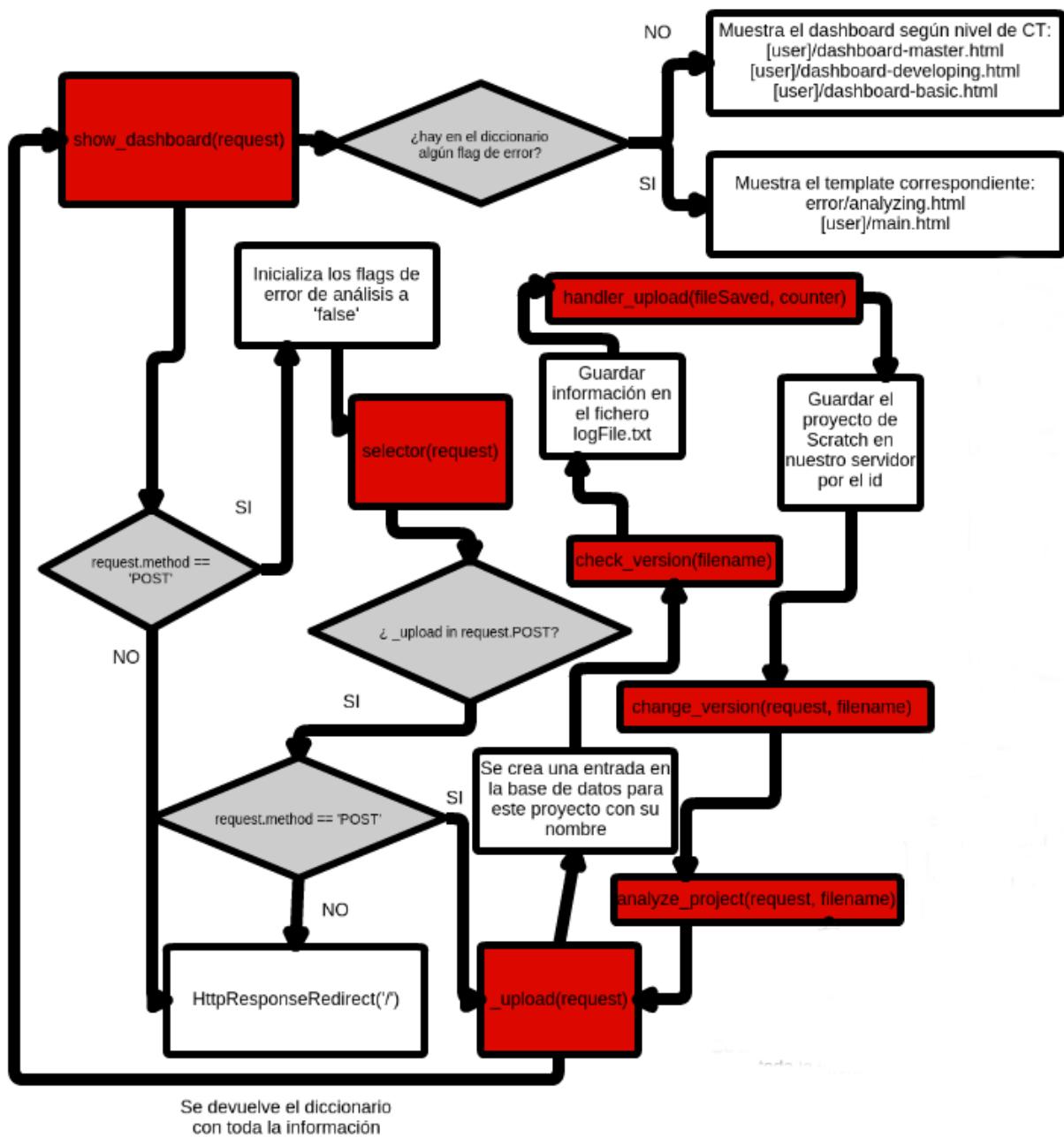


Figura 4.59: Diagrama de flujo del análisis *upload*

Vamos a ir viendo cada parte del diagrama por separado para facilitar su comprensión y evitar repetir explicaciones en las siguientes implementaciones:

1. La función `show_dashboard()` es la función a la que se llama cuando se elige un proyecto y se clica en el botón de analizar. Su lógica se presenta en la *Figura 4.60* y se encarga de mostrar al usuario la información referente al análisis:
 - Si se ha encontrado algún problema en el análisis muestra páginas en las que se detalle el error.
 - Si el análisis se realiza correctamente, muestra los diferentes dashboards acorde al nivel de Pensamiento Computacional del usuario.

Figura 4.60: Lógica de la función `show_dashboard()`

Y las tareas que realiza son las siguientes:

- Comprobar si el método de la petición es POST realizará el análisis. En caso contrario redirige a la página principal.
- Al comenzar el análisis lo primero que hace es inicializar los flags de error a `false` y llama a la función `selector()` que es la que se encarga en distinguir entre análisis por subida de proyecto o por URL.

- Como en este caso el análisis es por subida de proyecto, `selector()` llama a la función `_upload()`. Esta función es la que se encarga de guardar toda la información del análisis en la base de datos: proyecto y análisis. Y además llama a las funciones `check_version()` y `change_version()` para comprobar si el proyecto es de la versión 1.4 de Scratch, y en caso afirmativo cambiarla a la versión 2.0 antes del análisis.
- Cuando todo lo anterior ha sido realizado y se ha gestionado la subida del proyecto a nuestro servidor mediante la función `handler_upload()`, se llama a `analyze_project()`, que se explica a continuación.

2. La función `analyze_project()` es llamada para realizar el análisis de Hairball y procesar su información.

En el diagrama de flujo de la *Figura 4.61* podemos ver que hace uso de distintos procesadores para obtener la información que nos interesa del análisis y la almacena en un diccionario que es el que se renderiza en los dashboard que se muestran al usuario.

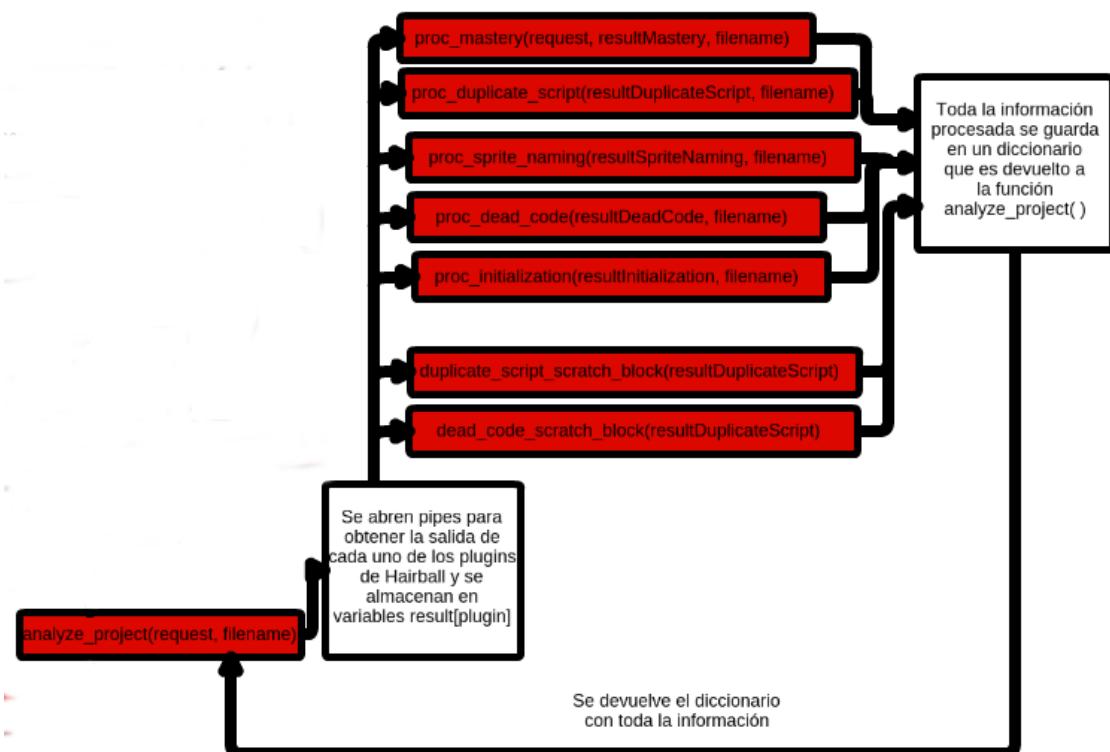


Figura 4.61: Lógica de la función analyze_project ()

Análisis con la URL: `_url`

En este caso, nos vamos a centrar en la función `show_dashboard()`, ya que como se puede entender, `analyze_project()` sigue teniendo la misma forma.

En este análisis hay algunas modificaciones en la lógica, que se observan en la *Figura 4.62* y la principal es que ahora la función llamada es `_url()`.

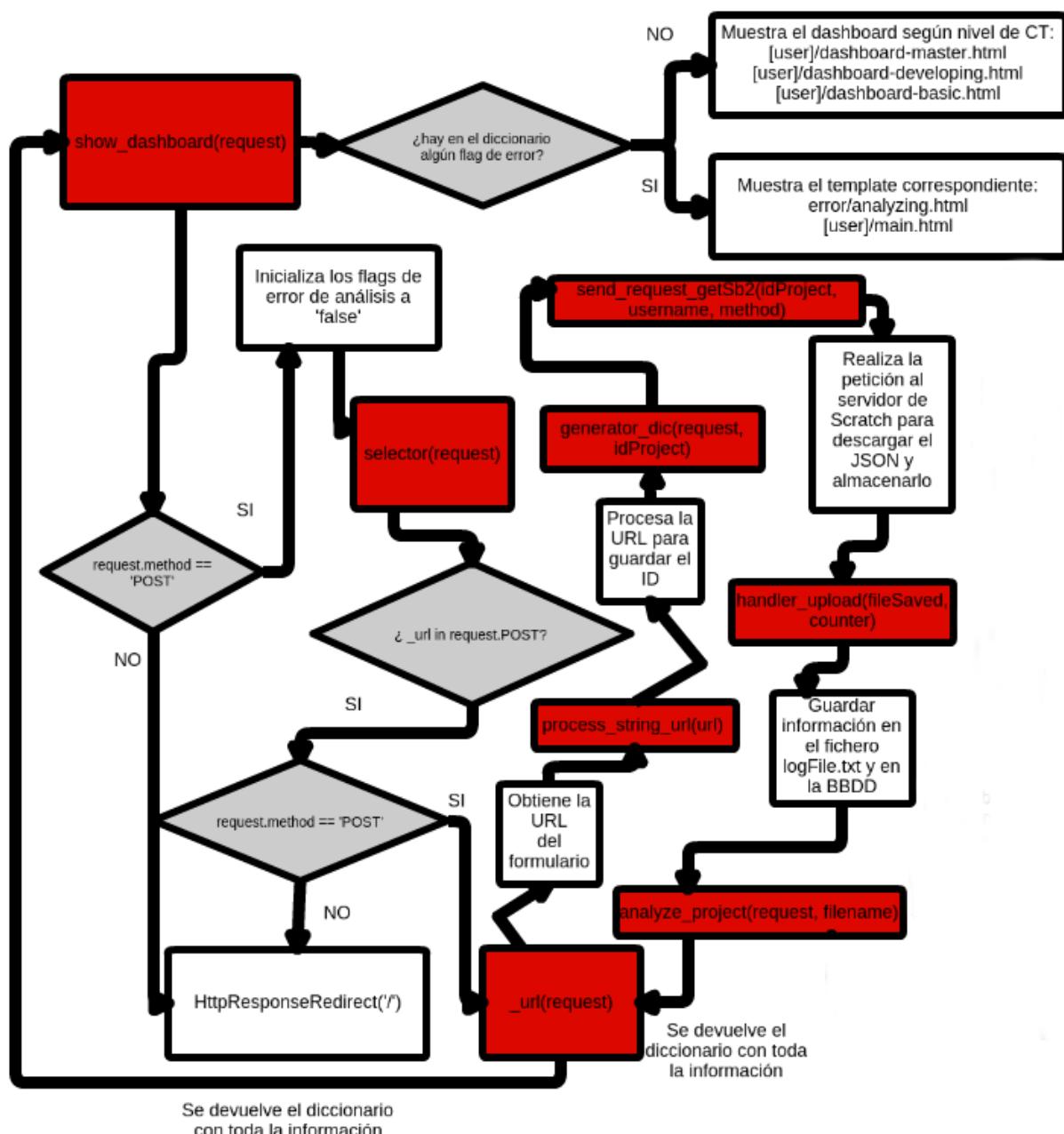


Figura 4.62: Lógica de la función `_url()`

La función `_url()` se encarga de llamar a `process_string_url()` para poder obtener el número de identificador del proyecto a través de la URL. Cuando lo tiene, se llama a `generator_dic()` que realiza las siguientes tareas:

- Lo primero que hace es descargar el archivo JSON mediante la función `send_request_getSb2()`.

Esta función, además de descargar el archivo, guarda en la base de datos y en el log toda la información relevante.

- Cuando ya tiene el archivo lo analiza llamando a `analyze_project()`.

Análisis de múltiples proyectos con un fichero: `_csv`

Cuando utilizamos el cuadro de análisis específico para organizaciones, el mecanismo cambia. Al clicar en el botón de análisis se llamará a la función `analyze_CSV()` y la lógica del método es la que se muestra en *Figura 4.63*.

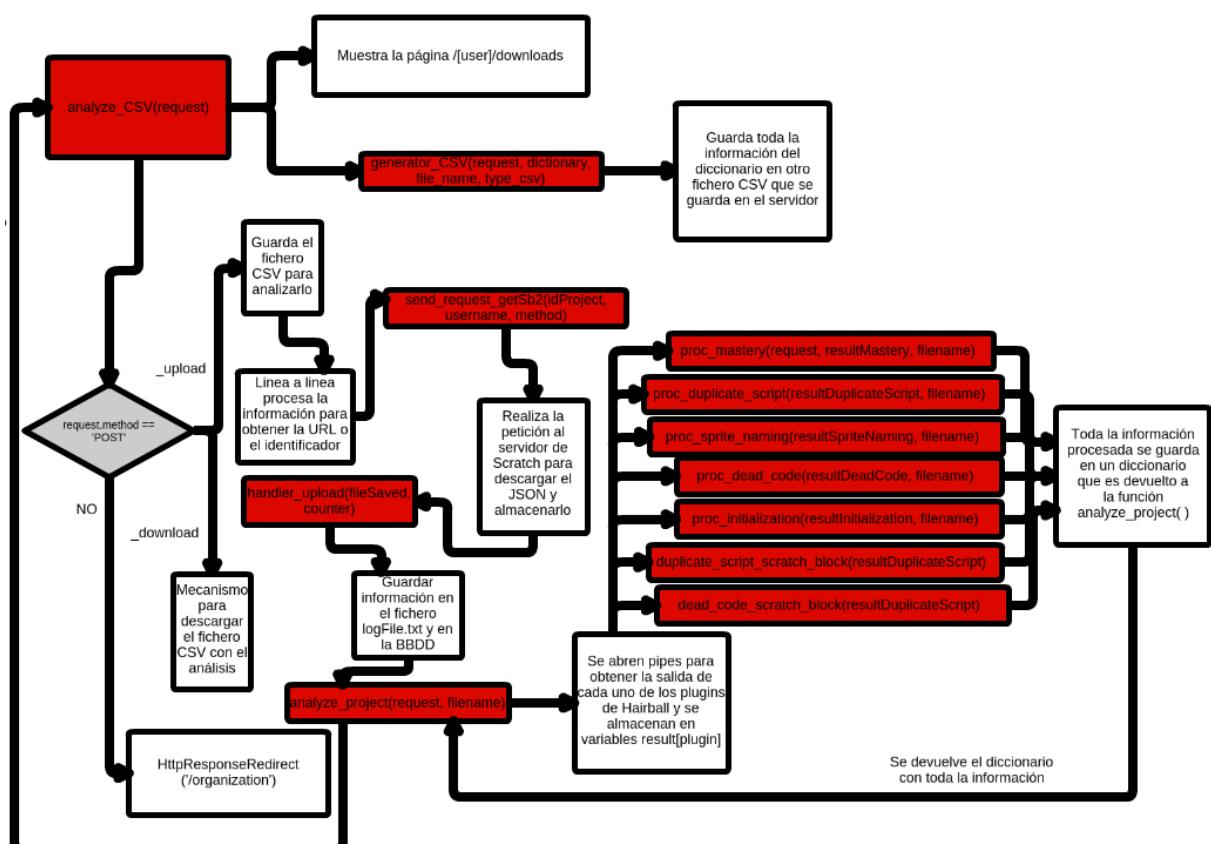


Figura 4.63: Diagrama de flujo del análisis `_CSV`

Como se observa lo primero que hace es ver si el método con el que se realiza la petición es POST, ya que si no lo es redirige a la página principal. En caso afirmativo, pasa a comprobar si se encuentra `_upload` o `_download` en `request.POST` para distinguir entre los dos casos: subir un nuevo CSV para analizar o descargar el CSV con los resultados del análisis.

- **_upload:** Lo primero que hace es guardar el fichero CSV en nuestro servidor para después procesarlo línea a línea obteniendo el identificador de cada proyecto. Cuando lo tiene, llama a la función `send_request.getSb2()` para realizar la petición al servidor de Scratch y descargar el JSON. Para guardarlo, gestiona el nombre con el que se va a guardar con la ayuda de `handler.upload()` y guarda información del proyecto en el fichero `logFile.txt` y en la base de datos. Y finalmente, llama a `analyze_project()` que le devuelve el resultado del análisis y lo guarda en el diccionario.

Cuando se realizan los análisis de todas las líneas del fichero CSV subido, `analyce_CSV()` llamará a `generator_CSV()` que es el que se encarga de pasar la información del diccionario a un fichero CSV para ser mostrado al usuario en la página mostrada mediante el recurso `/[user]/downloads`.

- **_download:** Lo único que ejecuta es un pequeño código, que hace descargar al ordenador del usuario el CSV generado con el resultado del análisis.

Capítulo 5

Resultados

Capítulo 6

Conclusiones

En este capítulo quiero realizar un análisis de cuáles han sido los objetivos alcanzados y cuáles no han podido realizarse por algún motivo particular. Además, realizaré una reflexión de cómo lo aprendido durante mis años en la universidad me han ayudado en la realización de este PFC y también lo que he aprendido con el desarrollo del mismo. Finalmente, indicaré cuales son las líneas futuras que pueden ser seguidas para la mejora de Dr. Scratch y una valoración personal de qué es lo que me ha aportado este año y medio.

6.1. Consecución de objetivos

Al ser un proyecto tan grande y con tantas personas involucradas, hay que tener en cuenta el esfuerzo de coordinación que esto supone, además de otras tareas realizadas sumadas al desarrollo de la plataforma.

En primer lugar, el **objetivo general** que era tener una plataforma web en producción 24x7 que no presentara problemas y realizara la funcionalidad básica ha sido alcanzada con creces. En los 6 meses de julio a diciembre de 2015, la plataforma únicamente ha estado caída unas 2 horas. Y estas horas en las que no ha estado funcionando ha sido debido a que se estaban realizando cambios, para añadir implementaciones al entorno de producción.

Además, puede concluirse que como se ha ido viendo a lo largo de este documento, los **objetivos específicos** que se marcaron en cada una de las diferentes fases han sido alcanzados, teniendo Dr. Scratch numerosas funcionalidades que lo hacen atractivo para el usuario.

Destacar de nuevo, la importancia de los talleres y jornadas que hemos ido realizando a lo

largo del proyecto, lo cual nos ha ido haciendo pivotar según las necesidades reales observadas. A estas necesidades han sido a las que se ha ido dando prioridad una vez logrado el prototipo inicial.

Y hemos podido observar, de forma muy gratificante, que Dr. Scratch es una herramienta muy útil para numerosos docentes y alumnos. En el caso de los docentes hemos recibido inmensidad de correos dándonos las gracias

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

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a

2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

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

- [1] L. A. Calao, J. Moreno-León, H. E. Correa, and G. Robles. Developing mathematical thinking with scratch. In *Design for Teaching and Learning in a Networked World*, pages 17–27. Springer, 2015.
- [2] R. G. Duque. Python para todos. pages 7–9, 2011.
- [3] K. Gareis, T. Husing, S. Birov, I. Bludova, C. Schulz, W. B. Korte, et al. E-skills for jobs in europe: Measuring progress and moving ahead. 2014.
- [4] D. Greenfeld and A. Roy. Two scoops of django. 2013.
- [5] M.-L. Liu and J. M. P. Sánchez. *Computación distribuida: fundamentos y aplicaciones*. Pearson Educación, 2004.
- [6] J. Moreno and G. Robles. Automatic detection of bad programming habits in scratch: A preliminary study. In *Frontiers in Education Conference (FIE), 2014 IEEE*, pages 1–4. IEEE, 2014.
- [7] M. Pilgrim and S. Willison. *Dive Into Python 3*, volume 2. Springer, 2009.
- [8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.
- [9] S. Suehring. *Mysql bible*. John Wiley & Sons, Inc., 2002.
- [10] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.