

ón general Pylint3.26 ón del dashboard.4.212 ón del dashboard.4.312
 ón.4.916 ón Statistics.4.1017 ón media diaria.4.1118 ón.4.1219 ágina principal de
 organizaciones.4.1320 ágina de estadísticas de organizaciones.4.1421 ísticas de organi-
 zaciones.4.1521 ú desplegable.4.1622 ón downloads.4.1822 ón changePwd.4.2024
 ña.4.2124 ón reset_password_confirm.4.2225 áginas de ayuda.4.2426 áginas de
 ayuda.4.2527 ón de formularios vía AJAX.4.2931 úmero de sesiones.5.133
 áfico en la web.5.233 áfico en la web.5.334



GRADO EN INGENIERÍA DE TELECOMUNICACIONES EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2015/2016

Trabajo Fin de Grado

MEJORA DE LA PLATAFORMA DR. SCRATCH

Autor : Eva Hu Garres

Tutor : Dr. Gregorio Robles

Co-Tutor: Jesús Moreno León

Trabajo de Fin de Grado

Mejora de la plataforma Dr. Scratch

Autor : Eva Hu Garres

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Trabajo de Fin de grado se realiza el día de
de 2016, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2016

*Dedicado a
mi abuelo Manolo.
No te conocí, pero te quiero.*

Agradecimientos

Desde pequeña, mis padres siempre me han enseñado a trabajar muy duro por lo que se quiere. Me han enseñado que las oportunidades no llegan a uno si no es uno el que trabaja en buscarlas. Gracias a ellos he crecido con una mente abierta, dispuesta a aprender de todo y de todos. Y en este párrafo quiero darles las gracias, porque se merecen todo lo bueno que hay en el mundo. Os quiero papá y mamá.

Quiero dar las gracias a mis amigos de siempre y a mis amigos de la universidad, seguro que en algún momento de mi vida me habéis hecho reír y ser feliz y eso siempre hay que agradecerlo. Gracias Lidia y Lurdes, por los buenos momentos que pasamos juntas.

También quiero dar las gracias al equipo de Dr. Scratch. A mi tutor Gregorio, por haberme dado esta oportunidad tan grande de trabajar con él. A mi co-tutor Jesús, por haberme ayudado tanto en el proceso y por la paciencia que ha tenido conmigo. Y sobretodo a mi compañera de trabajo, amiga y compitrueno, Mari Luz, que tanto me ha enseñado y soportado. Aún te queda mucho por soportar :).

Por último, quiero darle las gracias a mi amor, Alejandro. Por hacerme feliz durante estos años, te dedico esta memoria, te quiero.

Resumen

Este proyecto trata de mejorar la versión inicial de Dr. Scratch. Dr. Scratch es una herramienta web de código abierto que permite a profesores y alumnos, y usuarios en general, analizar sus proyectos Scratch y ver si lo ha programado correctamente, aprender de sus errores y obtener algo de realimentación para subir su nivel de pensamiento computacional.

Se ha realizado principalmente con el framework web Django y con el plugin Hairball, además de Bootstrap y otras tecnologías web.

El proyecto surge de una línea de investigación donde se explora la idea de que la programación puede mejorar el aprendizaje de otras asignaturas.

Summary

This project tries to improve the first version of Dr. Scratch. Dr. Scratch is a free/open source web tool that allows teachers and students, and every user, to analyze their Scratch projects to check if they have coded it correctly, learn about their mistakes and obtain some feedback in order to improve their computational thinking level.

It has been developed with the framework Django and Hairball plugin primarily, in addition to Bootstrap and other web technologies.

This project arises from a research where the idea that programming can enhance learning in other subjects is explored.

Índice general

| | |
|---|-----------|
| 1. Introducción | 1 |
| 2. Objetivos | 3 |
| 2.1. Objetivo general | 3 |
| 2.2. Objetivos específicos | 3 |
| 3. Estado del arte | 5 |
| 3.1. Python | 7 |
| 3.2. Django | 7 |
| 3.3. Bootstrap | 8 |
| 3.4. MySQL | 8 |
| 3.5. Hairball | 9 |
| 3.6. Scratch | 9 |
| 4. Diseño e implementación | 11 |
| 4.1. Arquitectura general | 11 |
| 4.2. Diseño e implementación | 12 |
| 4.2.1. Nuevos dashboards | 12 |
| 4.2.2. Estadísticas | 16 |
| 4.2.3. Organizaciones | 20 |
| 4.2.4. Páginas de ayuda | 26 |
| 4.2.5. Modificación del procesador de Dead Code | 27 |
| 4.2.6. Foro | 29 |
| 4.2.7. Validación de formularios vía AJAX | 30 |
| 4.3. Base de datos | 31 |

| | |
|---|-----------|
| 5. Resultados | 33 |
| 6. Conclusiones | 37 |
| 6.1. Consecución de objetivos | 37 |
| 6.2. Aplicación de lo aprendido | 39 |
| 6.3. Lecciones aprendidas | 39 |
| 6.4. Trabajos futuros | 41 |
| 6.5. Valoración personal | 41 |
| A. Feedback recibido | 43 |
| B. Dr. Scratch en la prensa | 45 |
| Bibliografía | 47 |

Índice de figuras

| | |
|---|----|
| 3.1. Mensajes de salida de Pylint | 6 |
| 3.2. Puntuación general Pylint | 6 |
| 3.3. Vista general de Scrape | 6 |
| 3.4. MVC de Django | 8 |
| 4.1. Arquitectura general | 11 |
| 4.2. Primera versión del dashboard. | 12 |
| 4.3. Ultima versión del dashboard. | 12 |
| 4.4. BootstrapTour. | 14 |
| 4.5. Twitter. | 14 |
| 4.6. Diploma. | 15 |
| 4.7. Link para volver a Scratch. | 15 |
| 4.8. Popup. | 15 |
| 4.9. Gamificación. | 16 |
| 4.10. Esquema de la función Statistics. | 17 |
| 4.11. Puntuación media diaria. | 18 |
| 4.12. Gamificación. | 19 |
| 4.13. Página principal de organizaciones. | 20 |
| 4.14. Página de estadísticas de organizaciones. | 21 |
| 4.15. Esquema estadísticas de organizaciones. | 21 |
| 4.16. Menú desplegable. | 22 |
| 4.17. Descargas de csv. | 22 |
| 4.18. Función downloads. | 22 |
| 4.19. Esquema de ajustes. | 23 |

| | |
|---|----|
| 4.20. Esquema de la función <code>changePwd</code> | 24 |
| 4.21. Email de restablecimiento de contraseña. | 24 |
| 4.22. Esquema de la función <code>reset_password_confirm</code> | 25 |
| 4.23. Herencia de plantillas. | 26 |
| 4.24. Acceso a páginas de ayuda. | 26 |
| 4.25. Acceso a páginas de ayuda. | 27 |
| 4.26. Esquema <code>DeadCode</code> | 28 |
| 4.27. Esquema Foro. | 29 |
| 4.28. Foro. | 30 |
| 4.29. Validación de formularios vía AJAX. | 31 |
| 4.30. Modelo de datos. | 31 |
| 5.1. Número de sesiones. | 33 |
| 5.2. Tráfico en la web. | 33 |
| 5.3. Tráfico en la web. | 34 |
| A.1. Interacciones de Twitter de Dr. Scratch. | 43 |
| B.1. Dr. Scratch en la prensa. | 46 |

Capítulo 1

Introducción

En la última década, el número de dispositivos electrónicos (móviles, tabletas, ordenadores...) ha aumentado exponencialmente, aumentando así el número de usuarios que los consumen. Es por ello que hoy en día y cada vez más, se va concienciando a los alumnos en las escuelas la importancia de la programación para un futuro a corto y largo plazo aunque uno no se vaya a dedicar específicamente a ello en un futuro. Aprender a programar es una manera ideal de estructurar los procesos mentales, ayudando a asentar conocimientos que ya se tenían y a aprender conceptos nuevos.

Actualmente, existe un movimiento mundial que está llevando la programación a las aulas desde primaria (Reino Unido, Estonia, EEUU...) haciendo que docentes de todo el mundo tengan que aprender a programar (si no saben todavía), enseñar a sus alumnos y evaluar los programas de sus alumnos. Es por ello que necesitan herramientas que les apoyen en todo el proceso. Así surge Dr. Scratch, como herramienta de apoyo a docentes y aprendices.

Un año hace desde que nació Dr. Scratch. Fue fruto de la inspiración de mi tutor de proyecto, Gregorio Robles, y mi co-tutor, Jesús Moreno, cuando buscaban un campo de investigación en el que basar su doctorado y pensaron que, ¿por qué no enseñar programación a los niños de forma divertida? Ésa fue mi motivación para realizar este proyecto. ¿Puede haber algo mejor que trabajar con niños, enseñándoles a programar y además, programar tú mismo? Si lo hay, aún no lo conozco. Este proyecto reúne todos los aspectos que me gustaría que tuviera mi trabajo ideal; combina programación web, enseñanza y niños, algo que a priori, no parece tener ninguna relación.

A raíz de elegir este proyecto y conocer a Jesús Moreno (co-fundador de la asociación sin

ánimo de lucro Programamos) se me propuso (y acepté) participar y colaborar en la asociación impartiendo distintos talleres de programación para niños, como SuperProgramadores (evento organizado con Orange), charlas de emprendimiento, o la Semana Europea de la Programación.

Desde entonces, la plataforma ha cambiado mucho según se han ido haciendo encuestas y talleres. Gracias a la comunidad de usuarios que usan y/o han usado Dr. Scratch hemos podido seguir mejorando y creciendo en cada actualización.

En esta memoria de trabajo de fin de grado, voy a explicar los distintos objetivos marcados al inicio del proyecto. Después explicaré qué tecnologías he utilizado, cómo he desarrollado cada funcionalidad, su back-end y su front-end. Finalmente mostraré los resultados obtenidos y algunas conclusiones a las que he llegado en el último mes.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo de este proyecto es aportar realimentación sobre el nivel de pensamiento computacional de los proyectos realizados en el lenguaje de programación Scratch, enseñar buenas prácticas de programación y ofrecer una herramienta de apoyo para organizaciones y profesores a la hora de evaluar y ver la evolución de sus alumnos.

2.2. Objetivos específicos

- Mejorar los paneles mostrados al analizar, con el fin de simplificar lo máximo posible la información mostrada al usuario, teniendo en cuenta que está dirigida principalmente a niños de distinta edad y nivel de pensamiento computacional.
- Conseguir una web multilenguaje: traducir la web a las diferentes lenguas del mundo para acercarnos lo máximo posible al usuario final. Para ello, hemos contado con varios voluntarios de distintos países que han mostrado su interés en traducir la web a su idioma.
- Migrar el servidor a la nube para un ofrecer un mejor rendimiento.
- Conseguir registro de usuarios. Crear cuentas de organizaciones, profesores y alumnos, donde se pueda llevar un seguimiento de los proyectos analizados.
- Conseguir un análisis masivo de proyectos.

- Crear una página de estadísticas generales y otra página de foro.

Capítulo 3

Estado del arte

En este proyecto, hemos creado una aplicación web que permite a profesores y a alumnos analizar sus proyectos programados en Scratch. Actualmente existen varias herramientas que analizan estáticamente el código fuente de programas escritos en distintos lenguajes para ofrecer retroalimentación al usuario:

- Lint: consiste en una herramienta de análisis estático de código escrito en C, que detectaba código sospechoso o incompatible entre distintas arquitecturas en programas escritos en C; es decir, errores de programación que escapan al habitual análisis sintáctico que hace el compilador.
- Pylint¹: consiste en un analizador estático de código Python inspirado en Lint, que se puede instalar en la línea de comandos y ofrece información sobre cómo de bien está escrito nuestro código según la guía de estilos PEP-8 y muestra una serie de mensajes clasificados bajo las siguientes categorías:
 - Refactorización: Asociado a una violación en alguna buena práctica.
 - Convención: Asociada a una violación al estándar de codificación.
 - Advertencia: Asociadas a problemas de estilo o errores de programación menores.
 - Error: Asociados a errores de programación importantes, es probable que se trate de un bug.
 - Fatal: Asociados a errores que no permiten a Pylint avanzar en su análisis.

¹<http://www.pylint.org>

Messages by category

| type | number | previous | difference |
|------------|--------|----------|------------|
| convention | 969 | 1721 | ~-752.00 |
| refactor | 267 | 182 | ~+85.00 |
| warning | 763 | 826 | ~-63.00 |
| error | 78 | 291 | ~-213.00 |

Figura 3.1: Mensajes de salida de Pylint

Finalmente, Pylint nos da una puntuación general de 0 a 10, alentando al programador a subir de puntuación y con ello, mejorar su código:

Global evaluation

Your code has been rated at 7.74/10 (previous run: 4.64/10)
If you commit now, people should not be making nasty comments about you on c.l.py

Figura 3.2: Puntuación general Pylint

- Scrape²: consiste en un analizador de código Scratch en el cual se muestran los bloques de Scratch usados en el proyecto. Cuenta cuantas veces ha usado un determinado bloque, si ha usado listas, variables y cuántos objetos tiene, entre otros:

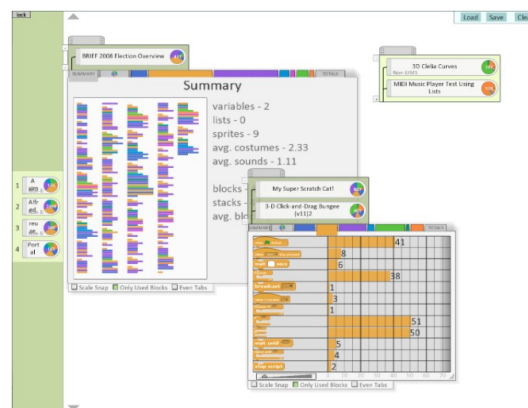


Figura 3.3: Vista general de Scrape

²<http://happyanalyzing.com/>

- `Hairball`³: consiste en un plugin de Python que analiza estáticamente proyectos de Scratch, ofreciendo a su salida la siguiente información:
 - Habilidades generales del pensamiento computacional (abstracción, paralelismo, lógica, sincronización, control de flujo, interactividad con el usuario y representación de los datos).
 - Malos hábitos de programación: programas duplicados, código muerto, nombrado de objetos e inicialización de atributos.
 - De todas las habilidades y hábitos mostrados anteriormente, los plugins Mastery (nos da una puntuación global sobre 21), código repetido y nombrado de objetos han sido desarrollados por Jesús Moreno⁴

Las tecnologías que he elegido para llevar a cabo este proyecto han sido las que se comentan a continuación.

3.1. Python

Es un lenguaje de programación interpretado, con una sintaxis sencilla y legible. Soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Fue creado por Guido van Rossum a finales de los ochenta y el nombre del lenguaje se debe a los humoristas británicos "Monty Python".

De todos los lenguajes de programación aprendidos a lo largo de la carrera, he elegido Python por ser el que tiene una sintaxis más sencilla, es multiplataforma y porque es open source.

3.2. Django

Es un framework web que permite construir aplicaciones web con Python como lenguaje de back-end más rápido y con menos código. Su patrón de arquitectura de software es MVC, Modelo-Vista-Controlador[?]:

³<https://github.com/ucsb-cs-education/hairball>

⁴<https://github.com/jemole/hairball>

- Modelo: contiene el núcleo de la aplicación.
- Vista: presenta la información obtenida del Modelo.
- Controlador: reacciona a interacciones del usuario.

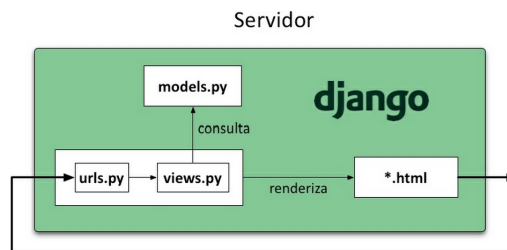


Figura 3.4: MVC de Django

Frente a otros frameworks web de Python como TurboGears o web2py, he elegido Django por ser fácil de usar, prácticamente en pocos pasos se puede tener una web en funcionamiento y porque tiene una buena documentación para poder empezar.

3.3. Bootstrap

Es un conjunto de herramientas de software libre para el diseño de sitios y aplicaciones web. Permite crear de forma muy sencilla elementos comunes de todo sitio web: botones, formularios, barras de navegación, etc.⁵

He elegido usar Bootstrap en este proyecto por que es un conjunto potente de herramientas que ayudan a que la web quede mucho más atractiva y además es bastante fácil de usar y de personalizar.

3.4. MySQL

Es un sistema de gestión de bases de datos relacional desarrollado por Oracle. Funciona bien en sitios de mucho tráfico y permite múltiples consultas al mismo tiempo.

⁵<http://getbootstrap.com>

3.5. Hairball

Como se ha mencionado antes, `Hairball`[1] consiste en un plugin de Python que analiza estáticamente proyectos de Scratch, ofreciendo a su salida la siguiente información:

- Habilidades generales del pensamiento computacional (abstracción, paralelismo, lógica, sincronización, control de flujo, interactividad con el usuario y representación de los datos).
- Malos hábitos de programación: programas duplicados, código muerto, nombrado de objetos e inicialización de atributos.

3.6. Scratch

Es un pseudo-lenguaje de programación basado en bloques, orientado a la enseñanza principalmente mediante la creación de videojuegos, historias, felicitaciones... La forma de programar en Scratch permite al usuario aprender rápidamente sin tener que aprender la sintaxis del lenguaje, centrándose en la lógica del programa. Scratch es, además, una gran comunidad de usuarios de los que aprender y con los que compartir los proyectos que se van creando, funcionando de una forma muy parecida a la que lo hace GitHub.

Capítulo 4

Diseño e implementación

A continuación se va a proceder a explicar en detalle cómo se han desarrollado las distintas funcionalidades en las que he trabajado en este proyecto.

4.1. Arquitectura general

A continuación se muestra la arquitectura general de Dr. Scratch:

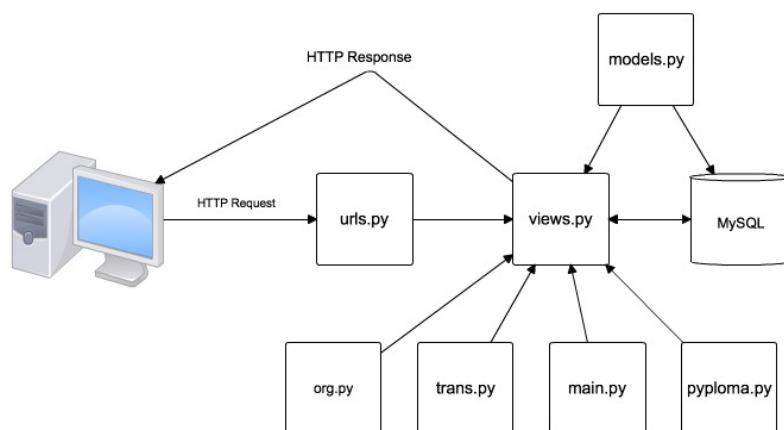


Figura 4.1: Arquitectura general

Cuando el servidor de Dr. Scratch recibe una petición HTTP, por ejemplo, `drscratch.org/alguna-url`, dicha petición es manejada por el archivo `urls.py`, que contiene todas las urls `drscratch/algo`. El archivo `urls.py` va mirando cada una de la urls que tiene en su lista y si la encuentra la enlaza con la función de `urls.py` asociada. Si no encuentra la url, optará por

coger la última. Una vez encontrada la url y la función de `urls.py` asociada, ejecuta dicha función, al final de la cual devolverá un `HttpResponse` y con ello una plantilla HTML.

4.2. Diseño e implementación

4.2.1. Nuevos dashboards

A lo largo de todo un año de trabajo, hemos realizado varios talleres tanto con alumnos como con docentes. Gracias a sus sugerencias hemos ido mejorando las pantallas mostradas al analizar proyectos. A continuación se vemos una primera versión del dashboard. Esta versión muestra para todos los niveles la misma información.

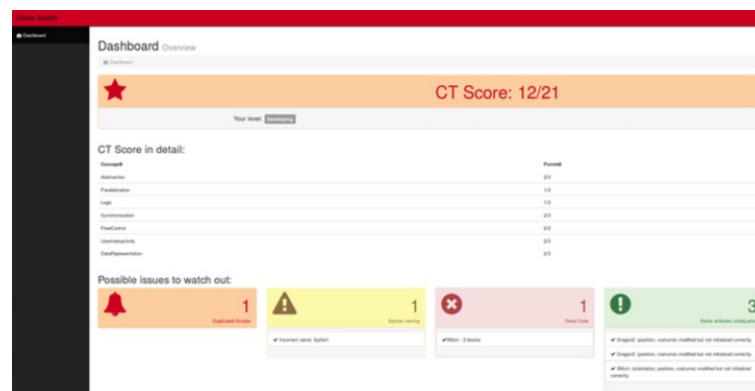


Figura 4.2: Primera versión del dashboard.

Finalmente nos dimos cuenta de que los más pequeños no sabían usar el scroll del ratón y por lo tanto, diseñamos unas nuevas pantallas que mostraran toda la información en una pantalla, sin necesidad de bajar con el ratón, y presentamos información distinta en función del nivel (más nivel, más información):



Figura 4.3: Última versión del dashboard.

Back-end

Para llegar a esta última versión sólo tuve que añadir al diccionario la URL del proyecto, en caso de que haya analizado por URL, para que el usuario pueda volver a su proyecto fácilmente. Además, reutilizamos la función `pyploa.py` para que el usuario se pudiera descargar su propio diploma. Para esta funcionalidad creamos una función `download_certificate`, en el que se llama a la función `pyploa.py` modificada pasándole como parámetros el nivel y el nombre del proyecto. Finalmente se monta una respuesta HTTP con contenido de tipo “application/pdf” y con el archivo pdf adjuntado. Con esto conseguimos que se descargue automáticamente el diploma.

Front-end

En la pantalla de la figura 4.4 se mostraba la puntuación total sobre 21 de pensamiento computacional, la puntuación parcial de cada habilidad de programación y si hay malas prácticas. Después decidimos mostrar diferentes pantallas en función del nivel (si es bajo, medio o alto), por lo que diseñamos las pantallas de la figura 4.5.

En la figura 4.5 vemos los dashboards asociados a cada nivel: básico, medio y alto (de izquierda a derecha). Para el nivel básico se muestra sólo la puntuación de las habilidades de programación, para el nivel medio, mostramos además de eso, dos malas prácticas que corregir y al nivel alto le mostramos todo. A esa versión final se le ha añadido:

- Enlaces a páginas de ayuda, donde los usuarios pueden encontrar pequeñas guías para subir de nivel.
- Una pequeña ayuda descriptiva en la que se explica cada sección del dashboard. Esta parte ha sido hecha con BootstrapTour ¹.

¹<http://bootstraptour.com>

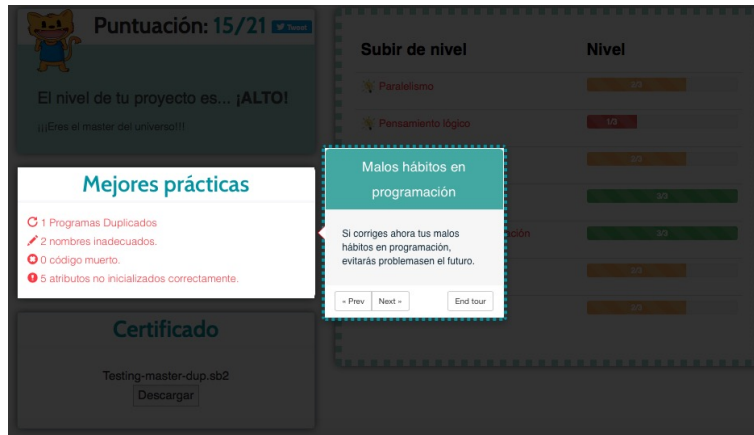


Figura 4.4: BootstrapTour.

- Posibilidad de publicar en Twitter tu puntuación ²

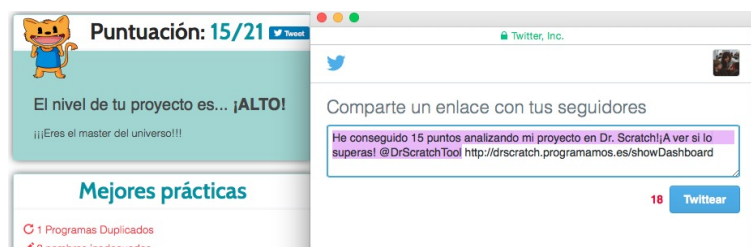


Figura 4.5: Twitter.

- Diplomas. Cada vez que el usuario analiza un proyecto, tiene la posibilidad de descargarse un diploma con la puntuación obtenida:

²<https://about.twitter.com/es/resources/buttons>



Figura 4.6: Diploma.

Dicho diploma se crea gracias al script Pyplopa (modificado) y mediante LaTeX.

- Si el usuario ha analizado el proyecto vía URL, puede volver a él fácilmente:



Figura 4.7: Link para volver a Scratch.

- Popups que informan de las malas prácticas.



Figura 4.8: Popup.

- Gamificación: en todas aquellas habilidades de programación donde ha obtenido 3/3, la barra de progreso se torna verde y se le añade una estrella.



Figura 4.9: Gamificación.

4.2.2. Estadísticas

Se ha creado una página de estadísticas donde se podrá consultar:

- Puntuación media diaria.
- Porcentaje de niveles.
- Número de proyectos analizados cada día.
- Puntuación media por habilidad.
- Puntuación media por mal hábito.

Back-end

Dicha página se actualiza cada noche mediante la creación de un nuevo comando en Django, ejecutando `python manage.py mystats` en la línea de comandos como una tarea programada. Esto se ha realizado mediante la creación de las carpetas y archivos siguientes:

```
app/
|--management/
|-----__init__.py
|-----commands/
|-----__init__.py
```

Es en `mystats.py` donde realizaremos todos los promedios y los guardaremos en la base de datos. Haciendo así las estadísticas, evitamos tener que realizar todos los promedios en tiempo real cada vez que se visita la página, quitando carga al servidor. A su vez, se ha creado una función en el archivo `views.py` llamada, `statistics`, que consultará los datos en la base de datos y formará un diccionario para devolverlo en un objeto HTTP. La función `mystats.py` está estructurada de la siguiente manera:

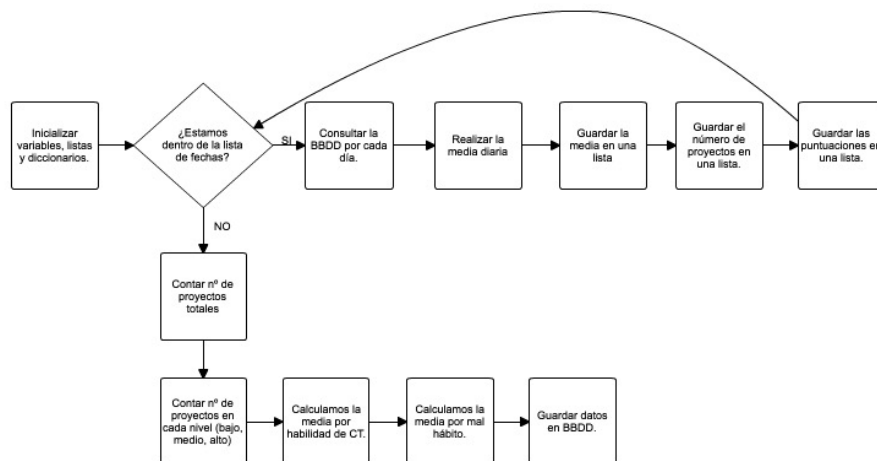


Figura 4.10: Esquema de la función Statistics.

En el esquema de arriba podemos ver cómo se generan las estadísticas. En primer lugar, inicializamos las variables, diccionarios y listas y creamos una lista de fechas, desde una que indiquemos hasta el día de hoy. Después vamos recorriendo esa lista de fechas, consultando en la BBDD y recogiendo los siguientes datos:

- Media de puntuación diaria.
- Número de proyectos analizados cada día.
- Todas las puntuaciones.

Una vez hecho esto, contamos el número total de proyectos y el número de proyectos que se encuentra en cada nivel (bajo, medio, alto) para poder generar una gráfica en forma de “quesito”. Finalmente calculamos la puntuación media por habilidad del pensamiento computacional y por malas prácticas, y guardamos en la BBDD.

Haciéndolo de esta manera, la función alojada en `views.py` mencionada anteriormente, sólo tiene que consultar la BBDD para obtener las estadísticas y devolverlas en un diccionario en la respuesta HTTP.

Front-end

Para mostrar las estadísticas generadas en forma de gráficas, optamos por usar Highcharts³ dado que dispone de muchos tipos de gráficas y se adapta bien a lenguaje de plantillas de Django. Las estadísticas que se muestran son:

- Puntuación media diaria.
- Porcentaje de niveles.
- Número de proyectos analizados cada día.
- Puntuación media por habilidad.
- Puntuación media por mal hábito.

Todas estas estadísticas se generan cada noche para evitar sobrecargar el servidor.



Figura 4.11: Puntuación media diaria.

³<http://www.highcharts.com>



Figura 4.12: Gamificación.

Para usar Highcharts simplemente tenemos que descargar e importar la librería y pasarle una lista de números. Por ejemplo:

```
$(function () {
  $('#chart-panel').highcharts({
    title: {
      text: '% trans "Scores of 2015" %',
      x: -20 //center
    },
    xAxis: {
      categories: date_list
    },
    tooltip: {
      valueSuffix: '% trans " points"% '
    },
    legend: {
      layout: 'vertical',
      align: 'right',
      verticalAlign: 'middle',
      borderWidth: 0
    },
    series: [{
      name: '% trans "Total score" %',
      data: [{ dailyRate }]
    }]
  });
});
```

Los únicos datos que tenemos que rellenar son: `xAxis`, con la lista de fechas, y en series el campo “data”, la lista de puntuaciones. También rellenamos los campos `title`, con el título de la gráfica, `tooltip` con las unidades de la gráfica y `series(name)` con el nombre de los datos que estamos representando.

4.2.3. Organizaciones

Página principal

Se trata de una página donde la función principal es la de analizar. Consta de dos partes: análisis único (por subida de fichero o CSV) y análisis masivo (por subida de csv). Si analizamos un solo fichero, se nos mostrarían los dashboards explicados anteriormente. En cambio, si analizamos un fichero CSV, se nos mostraría la página de descargas para descargar el resultado del análisis.

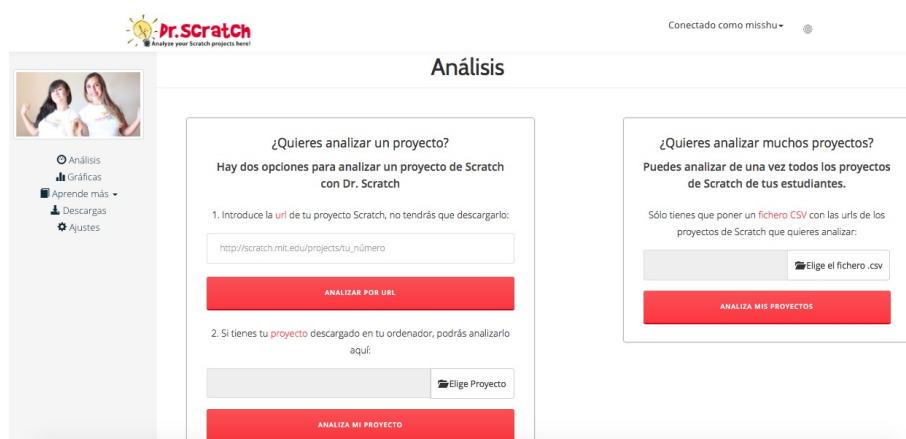


Figura 4.13: Página principal de organizaciones.

Estadísticas

En esta sección se muestran unas estadísticas individuales de todos los proyectos analizados por el usuario. De esta manera se podrá tener un seguimiento de la evolución del nivel de pensamiento computacional que adquieren los miembros de dicha organización.

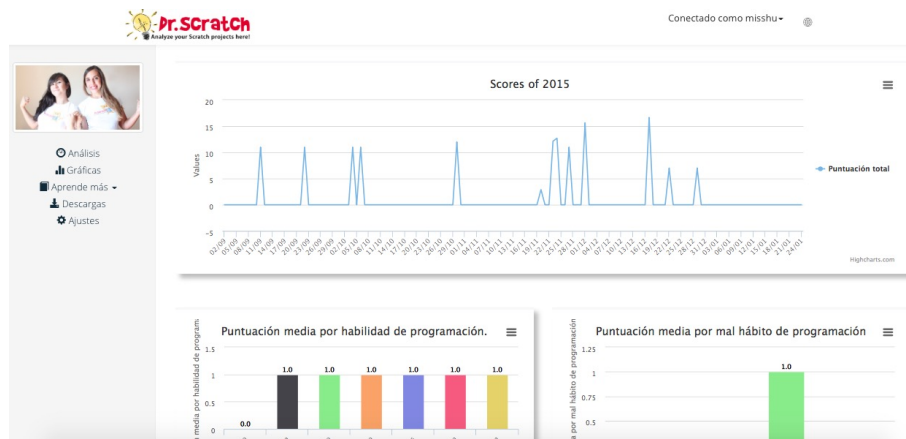


Figura 4.14: Página de estadísticas de organizaciones.

Estas estadísticas se generan de la siguiente manera:

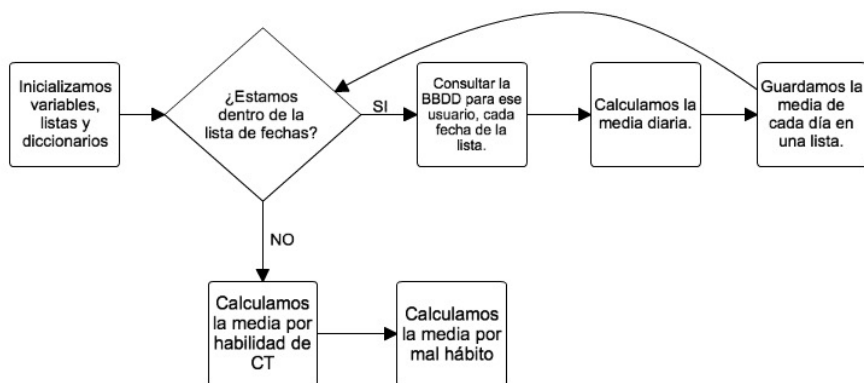


Figura 4.15: Esquema estadísticas de organizaciones.

La manera de calcularlo es muy similar a las estadísticas generales, con la diferencia de que aquí no contamos el número de proyectos analizados ni el porcentaje de proyectos que hay en cada nivel.

Páginas de ayuda

Consiste en un menú desplegable con acceso directo a las páginas de ayuda para que se pueda acceder sin necesidad de analizar un proyecto.



Figura 4.16: Menú desplegable.

Descargas

En esta sección se podrá descargar todos los resultados de los análisis por CSV, ordenados por fecha.



Figura 4.17: Descargas de csv.

La función que se encarga de esto realiza lo siguiente:

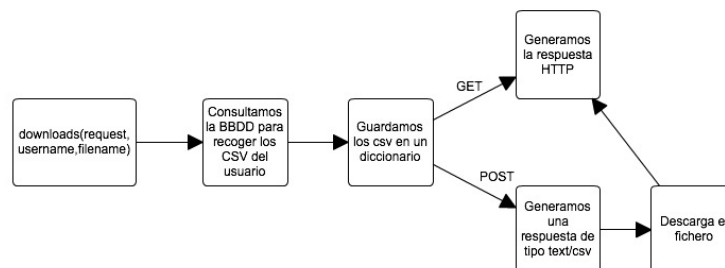


Figura 4.18: Función downloads.

Lo que hace esta función es recopilar todos los ficheros CSV. Después los organiza en un diccionario de páginas y en cada página guarda 10 ficheros CSV. Si la petición es un “POST”, montamos una respuesta HTTP donde el contenido es de tipo `text/csv` y el adjunto es el fichero CSV. Automáticamente el fichero se descargará en el ordenador del usuario. Si la petición es un “GET” simplemente se mostrará la página con las descargas disponibles.

Ajustes

Por último, incluimos un apartado de ajustes generales, donde se puede cambiar la contraseña y la foto de perfil. El procedimiento de cambio de contraseña se explica en el siguiente apartado. Para cambiar la foto de perfil nos cambiamos de directorio de trabajo a la carpeta donde queremos guardar la imagen. Después, si el usuario tiene ya una imagen de perfil, la borramos y guardamos la nueva para no consumir demasiada memoria en el servidor y lo guardamos también en la base de datos.



Figura 4.19: Esquema de ajustes.

Restauración de contraseña

Una funcionalidad básica imprescindible cuando se crean cuentas de usuario es la posibilidad de cambiar la contraseña. El procedimiento para ello se divide en dos partes:

1. El usuario pide restaurar su contraseña. Se le manda un email y en dicho email aparece una url única.
2. El usuario pulsa dicha url única que le redirige a una página donde escribe su contraseña dos veces.

Para cada una de las partes se ha desarrollado dos funciones llamadas `changePwd` y `reset_password_confirm`. A continuación se explican ambas funciones:

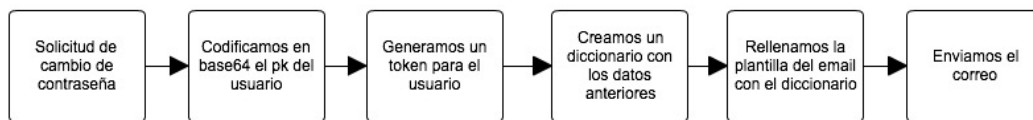


Figura 4.20: Esquema de la función `changePw`.

En la figura de arriba vemos que codificamos el pk en base 64 y generamos un token para el usuario. Ambos códigos formarán una url única para cada usuario. Después rellenamos un diccionario con el email de destino, el pk codificado, el token y el nombre de usuario. Con dicho diccionario rellenamos una plantilla HTML con el texto y lo enviamos. El usuario recibiría un email como el siguiente:

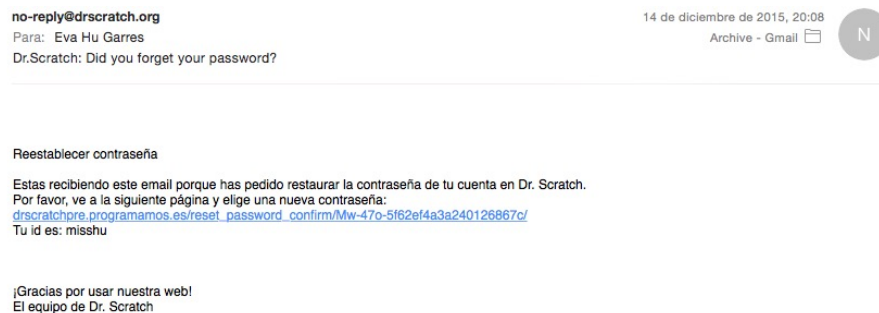


Figura 4.21: Email de restablecimiento de contraseña.

Una vez pulsado en el link que aparece en el email, se ejecutaría la función `reset_password_confirm`.

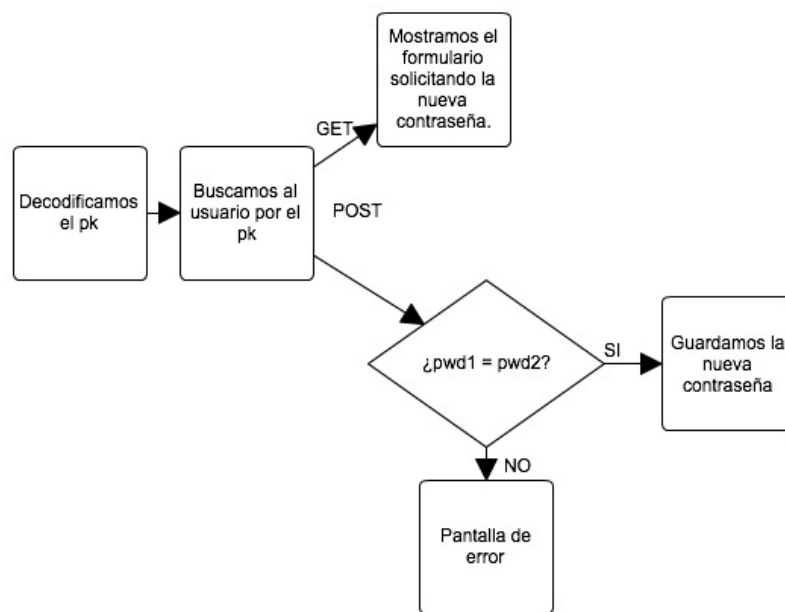


Figura 4.22: Esquema de la función `reset_password_confirm`.

Cuando se recibe el email con la url única, accedemos a un formulario con dos campos para introducir la nueva contraseña dos veces. La función decodifica el pk y con él busca el usuario. Después comprueba que ambas contraseñas son iguales. Si son iguales guarda la nueva contraseña y si no, muestra una pantalla de error.

Análisis masivo: csv(traducción de diccionarios)

Al analizar un CSV con proyectos, siempre se devuelve otro CSV con la puntuación total y la puntuación parcial de cada habilidad. Como dichos conceptos se entienden mejor en el idioma de cada usuario decidimos traducir los conceptos a varios idiomas. Esto se hizo a través de una función externa para evitar crear una función demasiado larga. Esta función se creó en un paquete externo llamado `org.py` y se llama `translate_CT`. Se le pasa como parámetro el idioma en el que está el navegador y se devuelve un diccionario con los conceptos traducidos. Así, se va rellenando las filas y columnas de la siguiente manera: `dic[«concepto>»]`.

Para crear el diseño de esta sección de la web, optamos por utilizar la herencia de plantillas que permite Django. Para ello creamos una plantilla base que contuviera la cabecera, el menú vertical (`base.html`) y el pie de página y simplemente creamos nuevas plantillas HTML donde heredaremos de la primera.

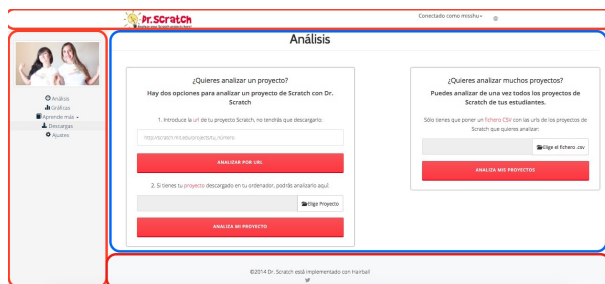


Figura 4.23: Herencia de plantillas.

En la imagen de arriba vemos la pantalla de inicio de organizaciones dividida en varias secciones. Las secciones rojas corresponderían a la parte que se encuentra en `base.html` y la parte azul sería la que hereda de `base.html` y se va rellenando. De esta forma evitamos escribir el mismo código varias veces. Es una buena opción ya que normalmente, todos los sitios web suelen tener la misma cabecera, menú y pie de página. Podemos apreciar elementos de Bootstrap tales como `navbar` (barra de navegación), `sidebar` (menú vertical), `footer` (pie de página), ventanas modales para explicar ciertos pasos, `glyphicons` para mejorar la usabilidad, `Dropdowns` (menú desplegables) para el cierre de sesión, el sistema de rejillas para organizar la parte de estadísticas, etc...

4.2.4. Páginas de ayuda

En los talleres realizados para profesores y alumnos, nos sugirieron tener algún tipo de ayuda para poder subir de puntuación. Por ello, creamos una serie de páginas de consulta en las que se podría aprender en qué consiste cada habilidad y malas prácticas y cómo mejorar. Los enlaces a estas páginas se muestran al analizar un proyecto:

| Subir de nivel | Nivel |
|------------------------------------|-------|
| ★ Paralelismo | 3/3 |
| ★ Pensamiento lógico | 3/3 |
| ★ Control de flujo | 2/3 |
| ★ Interactividad con el usuario | 2/3 |
| ★ Representación de la información | 3/3 |
| ★ Abstracción | 2/3 |
| ★ Sincronización | 3/3 |

Figura 4.24: Acceso a páginas de ayuda.

En el ejemplo, si quisiéramos mejorar la habilidad “Control de flujo” pincharíamos sobre el

y veríamos lo siguiente:



Figura 4.25: Acceso a páginas de ayuda.

Se muestran tres apartados para subir de 0 a 1, de 1 a 2 y de 2 a 3. En cada apartado veremos ejemplos visuales de lo que el usuario podría hacer para subir de puntuación en esa habilidad, y algunas explicaciones sobre los bloques que se utilizan en el ejemplo. Para mostrar los bloques hemos utilizado una librería llamada Scratch blocks⁴ que nos permite generar bloques de Scratch desde la plantilla HTML. Scratch blocks tiene una sintaxis muy parecida a la de sus bloques y está disponible para numerosos idiomas. Por ejemplo si quisiéramos mostrar el bloque .^al presionar bandera verde”sólo tendríamos que escribir:

```
<pre class="blocks">al presionar bandera verde
</pre>
```

Sin embargo, como Dr. Scratch está disponible también en más idiomas, escribimos los bloques en inglés y posteriormente se traducen:

```
<pre class="blocks">{% trans "when gf clicked" %}
</pre>
```

4.2.5. Modificación del procesador de Dead Code

Debido a que se ha modificado el plugin de Dead Code de Hairball, la salida que ofrecía dicho plugin ya no es la misma que la de antes. Por ello se ha tenido que modificar el procesador de dicho plugin para poder mostrarlo correctamente en el dashboard. Antes la salida que daba el plugin era:

```
$ hairball -p blocks.DeadCode Testing.sb2
Testing.sb2
{u'Witch': [kurt.Script([
```

⁴<https://github.com/tjvr/scratchblocks>

```
kurt.Block('forward:', 10),
kurt.Block('turnRight:', 15)], pos=(32, 287))}]}
```

Y actualmente es:

```
$ hairball -p blocks.DeadCode Testing.sb2
Testing.sb2
{u'Witch': [['move %s steps', 'turn @turnRight %s degrees']]}
```

Se modificó el procesador de Dead Code de tal manera que se convierte a diccionario todo lo que hay a partir de la tercera línea mediante la función `ast.literal_eval`.

Después, se recopilan todas las claves del diccionario, que serán todos los objetos donde hay código muerto y conforme se va recorriendo el objeto y contando los bloques que no se ejecutan, se va formando un string con el formato: “objeto: número de bloques que no se ejecutan”. Finalmente se guarda en la base de datos y se forma un diccionario con dos claves: “blocksz” “total” para entregarlo en una respuesta HTTP al analizar un proyecto. Por ejemplo:

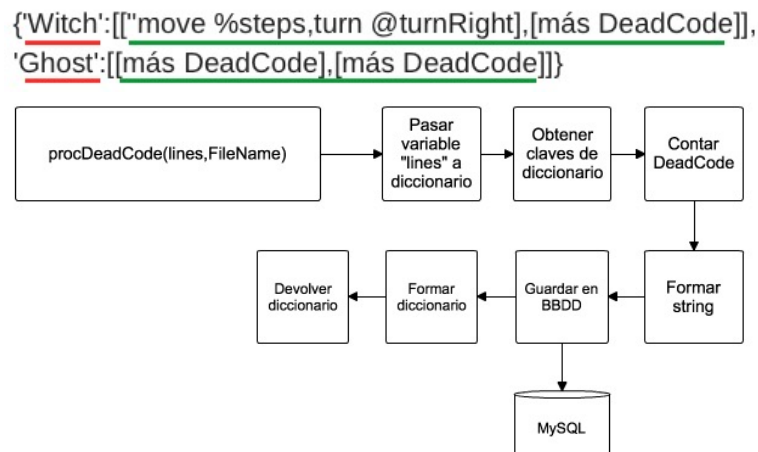


Figura 4.26: Esquema DeadCode.

En la figura de arriba vemos un ejemplo de lo que devolvería el plugin de Hairball “Dead Code”. El procesador que se ha desarrollado calcularía la longitud de la sub-lista asociada a “Witch”, en este caso, dos programas que no se ejecutan y el objeto “Ghost” tendría otros dos.

4.2.6. Foro

Back-end

Dr. Scratch tiene una cuenta Twitter y en Wordpress, donde se interactúa con los diferentes usuarios y se escriben entradas sobre los talleres que realizamos o las actualizaciones de la web. Sin embargo, queríamos incluir una parte de foro donde la gente que use Dr. Scratch pueda opinar sobre él. Para ello, se creó la función `discuss` en el fichero `views.py`. Dicha función tiene dos partes: si la petición es un GET, en el caso de mostrar mensajes, o es un POST, en el caso de comentar. Para poder comentar en el foro deberemos estar registrados. De esta manera evitamos tener que poner algún tipo de CAPTCHA.

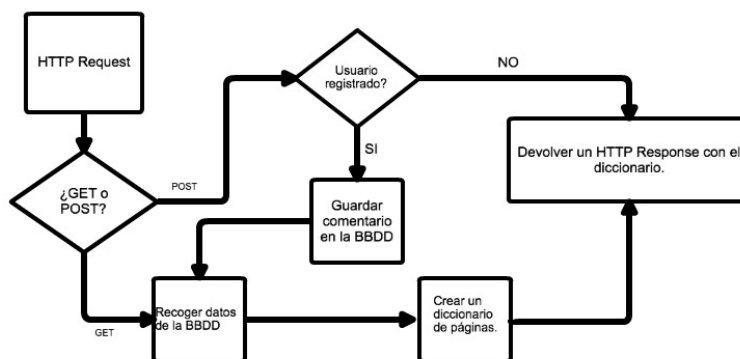


Figura 4.27: Esquema Foro.

En esta función primero comprobamos si el formulario está bien completado, se guardará el comentario en la base de datos. Cabe destacar que el usuario sólo podrá comentar si está registrado. Después se recopilan todos los comentarios y se ordenan de forma que el primero sea el más nuevo (por fecha) y el último sea el más antiguo. Finalmente se crea un diccionario de "hojas" donde la hoja 0 será la que tenga los comentarios más recientes y así sucesivamente, y se devuelve un objeto HTTP con la plantilla HTML y el diccionario.

Front-end

La función que se comenta en la parte Back-end da lugar a la siguiente pantalla:

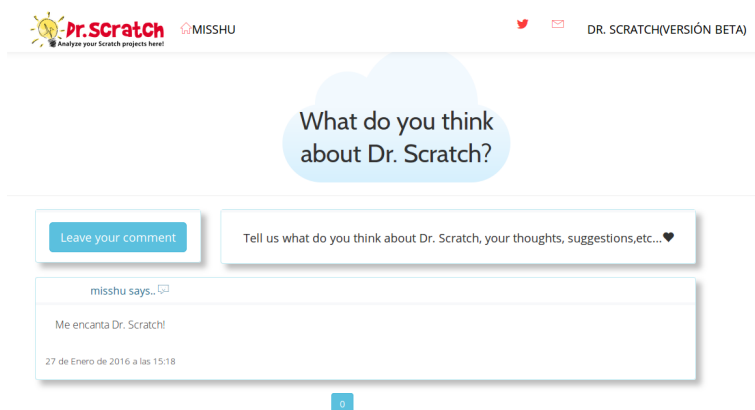


Figura 4.28: Foro.

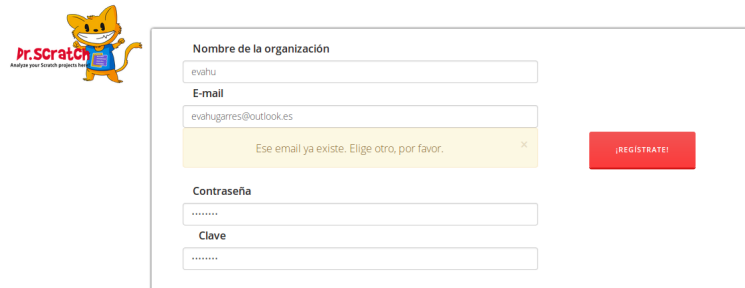
Se trata de un foro muy sencillo, donde se muestran los 10 últimos mensajes y el resto de mensajes se pueden ver cambiando de página. El cambio de página se ha realizado mediante jQuery y con la ayuda del diccionario de páginas que se entrega con la plantilla HTML.

4.2.7. Validación de formularios vía AJAX

Por último, he querido mejorar la validación de formularios, ya que inicialmente había que enviar el formulario, en el servidor se veía si estaba bien y luego se devolvía una pantalla de error en el caso de que el formulario no estuviera correctamente cumplimentado. Para evitar todo ese proceso he utilizado AJAX, que permite interactuar con el servidor sin tener que recargar la página. El proceso es el siguiente:

1. El usuario teclea en el formulario su correo (por ejemplo).
2. Mediante jQuery se detecta el evento `keyup`.
3. Cuando se detecta dicho evento, se manda por AJAX una consulta a la URL `/searchemail/` con el email tecleado.
4. La URL `/searchemail/` nos redirige a la función con el mismo nombre y en ella se busca en la BBDD.
5. Si no existe en la BBDD no hacemos nada, y si sí existe, se devuelve un JSON indicando que el email ya existe.

6. La plantilla recibe el JSON y ejecuta la función jQuery `emailexists`, que introduce un elemento `texttalert` de Bootstrap debajo del recuadro del email.



The screenshot shows a registration form with the following fields and elements:

- Nombre de la organización:** Input field with the value "evahu".
- E-mail:** Input field with the value "evahugares@outlook.es". Below this field is a yellow alert box with the message "Ese email ya existe. Elige otro, por favor." and a close button (X).
- Contraseña:** Input field with masked characters (dots).
- Clave:** Input field with masked characters (dots).
- REGISTRATE:** A red button to submit the form.

Figura 4.29: Validación de formularios vía AJAX.

4.3. Base de datos

Para guardar todos los nuevos datos se han tenido que crear nuevas tablas en la base de datos. La estructura actual es la siguiente:

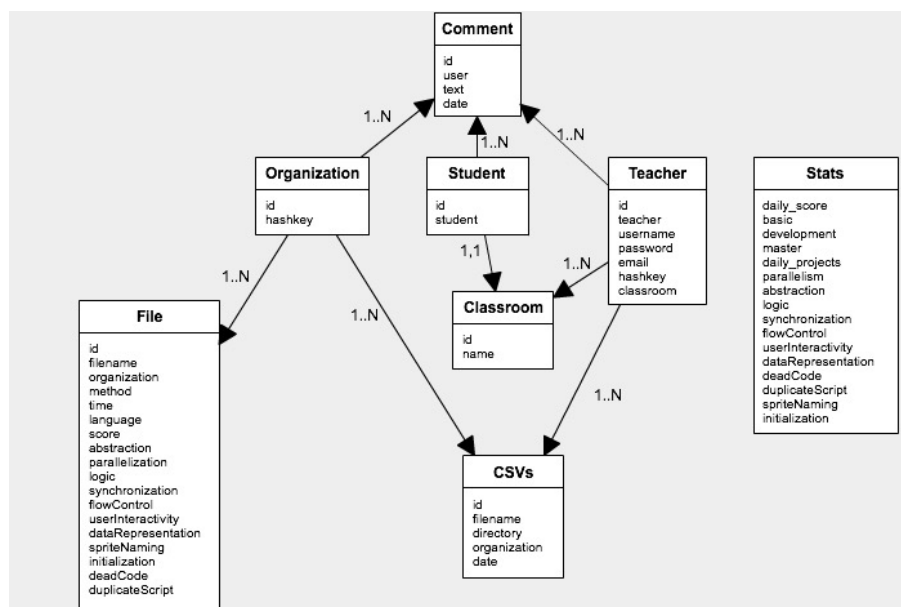


Figura 4.30: Modelo de datos.

Aunque aún no han sido creadas las cuentas de profesor, en la base de datos ya está modelado junto con las clases y los alumnos.

Capítulo 5

Resultados

Gracias al seguimiento realizado por Google Analytics de Dr. Scratch, hemos podido sacar unas estadísticas muy interesantes:



Figura 5.1: Número de sesiones.

En la gráfica de arriba podemos observar el número de sesiones que ha habido desde el 1 de abril (cuando se empezó a usar Google Analytics) hasta el día 26 de enero de 2016.

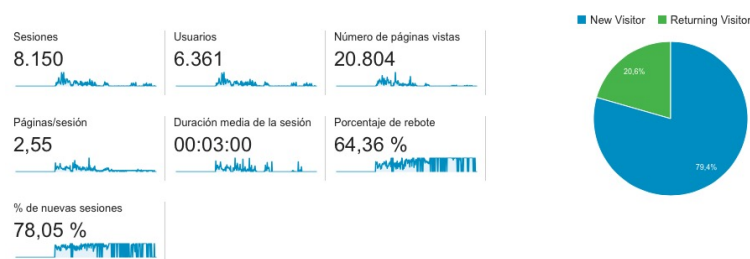


Figura 5.2: Tráfico en la web.

De forma más detallada, tenemos de forma numérica el tráfico que ha tenido Dr. Scratch:

- **Sesiones:** en total ha habido 8150 sesiones. Una sesión es el periodo durante el cual un usuario interactúa con su sitio web, aplicación, etc. Todos los datos de uso (visitas a una pantalla, eventos, comercio electrónico, etc.) están asociados a una sesión.

- Usuarios: 6367 personas diferentes han visitado Dr. Scratch.
- Número de páginas vistas: 20804 páginas. Es el número total de páginas visitadas por todos los usuarios. Las visitas repetidas a una misma página también se contabilizan.
- Páginas por sesión: 2,55. Es el promedio de páginas que se ven en cada sesión; las visitas repetidas a una misma página también se contabilizan.
- Duración media por sesión: 3 minutos.
- Porcentaje de rebote: 64,36 %. Es el porcentaje de visitas a una sola página, es decir, visitas en las que el usuario ha abandonado su sitio en la página de entrada sin interactuar con ella.
- Porcentaje de nuevas sesiones: 78,05 %. Porcentaje estimado de visitas realizadas por primera vez.

Dr. Scratch también ha tenido impacto en el mundo:

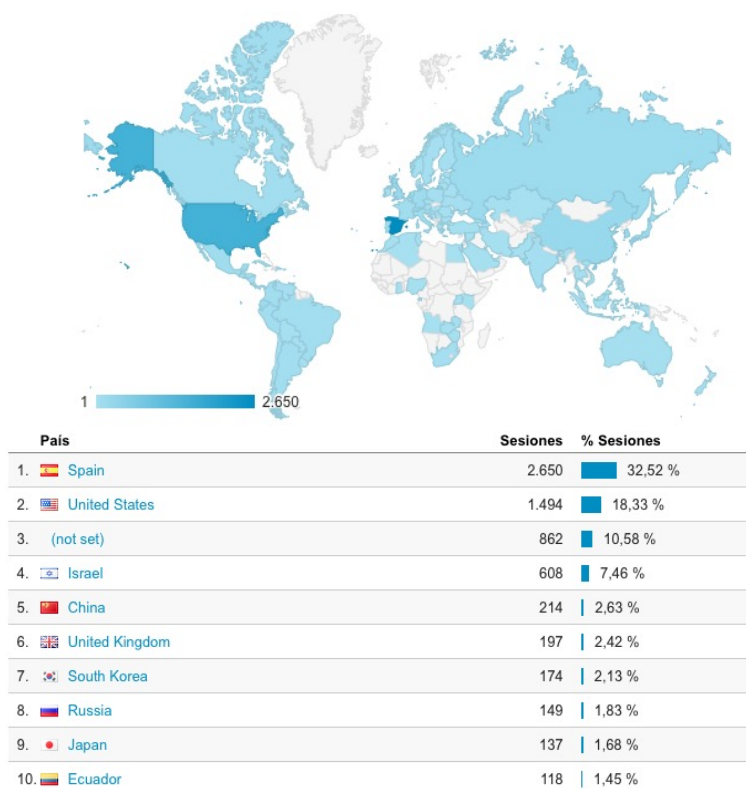


Figura 5.3: Tráfico en la web.

Finalmente destacar que el número de proyectos analizados es de 60.000 proyectos aproximadamente.

Capítulo 6

Conclusiones

Como se ha ido comentando a lo largo de esta memoria, hemos realizado distintos talleres en colegios tanto con niños como con profesores. El resultado que hemos tenido en todos ellos ha sido muy positivo. Los niños mostraban interés en mejorar su puntuación e incluso lo conseguían y retaban a sus compañeros. Cabe destacar un caso en el que una niña consiguió subir de 13 a 18 en una sola hora.[2] En cuanto a las organizaciones, son 13 las entidades que se han registrado para hacer uso de esta funcionalidad y analizar los proyectos de los alumnos participantes en sus iniciativas.

Además de los talleres realizados, también organizamos un concurso de programación en Scratch en el que se promovían vocaciones científicas, es decir, los niños tenían que entregar un proyecto en Scratch que explicara algún concepto tecnológico, matemático, físico, etc. Participaron niños de primaria y secundaria de toda España. La entrega de premios tuvo lugar en Campus Google de Madrid. Fue un emotivo encuentro entre todo el equipo de Dr. Scratch y los participantes del concurso, en el que finalmente se entregaron los premios de las dos categorías: primaria y secundaria. Los ganadores fueron tres niños y tres niñas, una de las cuales dio una pequeña charla motivadora a todas las niñas y mujeres de la sala. Cabe destacar que ésta última padece síndrome de Asperger.

6.1. Consecución de objetivos

- Mejorar los paneles mostrados al analizar, con el fin de simplificar lo máximo posible la información mostrada al usuario, teniendo en cuenta que está dirigida principalmente

a niños de distinta edad y nivel de pensamiento computacional. Esto se ha conseguido. Hemos adaptado la página de resultados lo máximo posible para los niños, simplificando la información mostrada. Nos dimos cuenta de que los niños no suelen utilizar el scroll por lo que tuvimos que condensar la información. Se introdujeron elementos de gamificación tales como las barras de progreso, enseñando a usar las fracciones de forma indirecta, estrellas en aquellas habilidades que tienen la puntuación completa, puntuación general, botón de Twitter para compartir sus resultados y se diseñó un logo identificativo de cada nivel.

- Web multilenguaje: esto se ha conseguido con creces. Gracias a la colaboración de usuarios de Dr. Scratch de todo el mundo tenemos Dr. Scratch completamente traducido a los siguientes idiomas: castellano, inglés, gallego, portugués, portugués de brasil, catalán, griego, ruso, alemán e incluso chino. Gracias a esto, Dr. Scratch sigue llegando cada vez a más sitios.
- Migrar el servidor a la nube para un ofrecer un mejor rendimiento. Esto se ha conseguido gracias a Azure mejorando así la calidad del servicio que se ofrece.
- Registro de usuarios. Crear cuentas de organizaciones, profesores y alumnos, donde se pueda llevar un seguimiento de los proyectos analizados. Hemos conseguido tener cuentas de organizaciones y alumnos. Sin embargo, no ha sido posible desarrollar las cuentas de profesores debido al tiempo y a la peculiaridad que tiene: son cuentas que pueden crear otras cuentas de usuarios(sus alumnos).
- Análisis masivo de proyectos. También se ha conseguido.
- Crear una página de estadísticas generales y otra página de foro. Se ha conseguido aunque la página de foro podría ser mucho más completa. No ha sido así debido al tiempo.

En general, puedo decir que hemos conseguido la mayoría de objetivos y sobre todo el principal y el que da título a este proyecto: mejorar la plataforma Dr. Scratch. Le hemos dado una imagen nueva, más sencilla y usable, con más funcionalidades y más información.

6.2. Aplicación de lo aprendido

En el proyecto he aplicado los conocimientos aprendidos de las siguientes asignaturas:

1. Informática I y II: fueron las primeras asignaturas de programación, donde yo aprendí a programar por primera vez. Aunque el lenguaje enseñado en estas asignaturas no fue Python ni ninguno de los utilizados en este proyecto, fue imprescindible para aprender a programar y dar los primeros pasos.
2. Arquitectura de Internet y Sistemas Telemáticos para Medios Audiovisuales: sin estas asignaturas no habría sabido cómo funciona el protocolo IP o HTTP, o el concepto de servidor entre otras muchas cosas. Ha sido muy útil de cara a subir al servidor las actualizaciones pertinentes o de cara a resolver problemas en el servidor.
3. Construcción de servicios y aplicaciones Audiovisuales en Internet: gracias a esta asignatura aprendí lo básico de HTML, CSS y Javascript.
4. Protocolos para la transmisión de audio y video en internet: en esta asignatura aprendí a programar en Python, entre otras cosas.
5. Laboratorio de tecnologías audiovisuales en la web: en esta asignatura aprendí a crear mi primer proyecto en Django.

6.3. Lecciones aprendidas

En este proyecto he aprendido numerosas cosas:

1. Gestión y trabajo en equipo. La coordinación y la ayuda entre compañeros ha sido la clave para que el proyecto se desarrollara con éxito. Dicha coordinación se ha llevado a cabo principalmente por la herramienta Trello y gracias a videoconferencias semanales.
2. Realizar eventos y workshops. Gracias a la realización de numerosos talleres, he aprendido a hablar en público y he perdido el miedo (y la vergüenza) que ello supone. También he podido ver cómo aprende un niño a programar. Este punto ha sido el más emocionante para mí.

3. Administración del servidor. Aquí he aprendido que hay varios entornos de desarrollo en un proyecto de software. Dr. Scratch cuenta con dos: preproducción (entornos de pruebas) y producción (entorno real). Este punto es muy importante ya que las pruebas han ocupado casi el 80 % del tiempo. He aprendido a poner en marcha un servidor Apache con Django y a realizar actualizaciones en él.
4. Usabilidad y diseño web. Una de las cosas más importantes a la hora de crear una web orientada a niños es que sea fácil de usar. Por ello hemos creado pantallas sencillas, con colores pastel y combinando las distintas componentes de Bootstrap para su fácil uso.
5. Nuevas tecnologías nunca antes usadas. He aprendido a usar tecnologías como Bootstrap, jQuery o MySQL. También he aprendido cómo funciona Apache, lo que completa bastante mi formación en la universidad.

Además, se han ido haciendo hackathones mensuales con todo el equipo de Dr. Scratch, donde hemos aprendido conceptos más avanzados de Python y Django:

- Unit Tests. Una parte muy importante y que como ya comentaba, ocupa un 80 % del tiempo, es la parte de testing. En el desarrollo de software podemos hablar de dos tipos de pruebas: prueba unitarias y pruebas integradas. Las primeras consisten en probar cada función por separado y comprobar que la salida es la correcta. Las segundas, en cambio, consisten en comprobar que todas las funciones juntas, funcionan bien, como unos engranajes. En Django existe un fichero específico llamado `tests.py` que facilitan la tarea a la hora de realizar pruebas unitarias.
- Estructuras avanzadas de Python: más allá de las clásicas listas y diccionarios a los que estamos acostumbrados ver en cualquier lenguaje de programación, Python dispone de estructuras más avanzadas, como las "List Comprehensions" que permiten realizar operaciones dentro de una lista de forma rápida y limpia. También tenemos los llamados "sets", que consiste en una colección no ordenada de valores no duplicados.

En resumen, creo que he aprendido muchas cosas nuevas y he mejorado en las que ya sabía.

6.4. Trabajos futuros

Dr. Scratch puede ser extendido y optimizado de muchas formas:

1. Cuentas de profesores. Actualmente existen en Dr. Scratch cuentas de usuarios y organizaciones. Una buena vía para seguir desarrollando y mejorando Dr. Scratch es creando cuentas para profesores donde éstos puedan agrupar a sus alumnos y ver su progreso. En estas cuentas podrían crear las cuentas de sus alumnos, ver quién va mejor o peor, sacar estadísticas. Semanalmente se podría poner un apartado de actividades, en el que Dr. Scratch sugerirá actividades para trabajar en clase algunas competencias de pensamiento computacional.
2. Crear una red social dentro de Dr. Scratch. Esto incrementará el grado de gamificación que actualmente tiene la web. Los usuarios tendrían la posibilidad de comentar las puntuaciones de sus amigos, lanzarse retos entre ellos, e incluso se podría incluir algún juego online por parejas o grupos en el que se practique lo que se sabe sobre pensamiento computacional.
3. Aplicación móvil. En esta aplicación se podría acceder a la cuenta de usuario y consultar las medallas o retos que nuestros amigos nos han lanzado.
4. Migración a Python 3 y optimización de código.
5. Dr. AppInventor. Una vez que un niño aprende Scratch y siente la necesidad de aprender otra cosa, puede ser bueno pasar a AppInventor. Para aprender a programar correctamente en AppInventor sería una gran idea disponer de un analizador como el de Dr. Scratch.

6.5. Valoración personal

Hace poco más de un año, justo antes de empezar este proyecto, recuerdo que me dijeron: “El proyecto de fin de grado no sirve para nada, no te esfuerces mucho...” Hoy, me doy cuenta de lo que ha significado para mí participar en este proyecto. Ha significado saber qué me gusta y qué no. Ha significado saber dónde me gustaría estar dentro de 10 años (la pregunta más difícil que me han hecho). Ha significado saber qué quiere decir la frase “encuentra un trabajo que te guste y no tendrás que trabajar nunca más”.

Aquí acaba una etapa de mi vida que no podría haber terminado mejor que habiendo realizado este proyecto, aunque estoy segura que la siguiente superará a esta y así continuamente...

Apéndice A

Feedback recibido

Dr. Scratch cuenta con un perfil en Twitter (@DrScratchTool) al que le siguen un total de 798 personas. Algunos comentarios motivadores que hemos recibido han sido:



Figura A.1: Interacciones de Twitter de Dr. Scratch.

Apéndice B

Dr. Scratch en la prensa

Dr. Scratch no solo ha tenido repercusión en redes sociales, sino que también se ha hecho eco en la prensa:



sinc La ciencia es noticia

PORTADA CIENCIAS NATURALES TECNOLOGÍA BIOMEDICINA Y SALUD MATEMÁTICAS FÍSICA Y QUÍMICA HUMANIDADES CIENCIAS SOCIALES POLÍTICA CIENCIA INNOVACIÓN

NOTICIAS REPORTAJES ENTREVISTAS MULTIMEDIA AGENDA ESPECIALES OPINIÓN ENVIAROS INVESTIGADORES

INNOVACIÓN: Innovación

La plataforma Dr. Scratch ha sido premiada por Google

Una nueva herramienta ayuda a enseñar programación a niños desde los seis años

Investigadores de la Universidad Rey Juan Carlos han desarrollado una herramienta para que los niños puedan aprender técnicas de programación desde los seis años de edad. Según sus creadores, el aprendizaje temprano de la programación mejora el rendimiento escolar en disciplinas tan distintas como las matemáticas o los idiomas. La plataforma, denominada Dr. Scratch, acaba de ser galardonada con el premio Google RISE Awards 2015.

Más información sobre: Dr. Scratch programación niños código

URJC / SINC | 04 mayo 2015 09:30

Este reconocimiento les permitirá continuar investigando y promocionando el desarrollo del pensamiento computacional desde edades tempranas.

Con estos premios, que en la edición de 2015 han repartido 1,5 millones de dólares a 37 organizaciones de 17 países, Google reconoce su labor a organizaciones y entidades que colaboran en la difusión de la informática entre los más jóvenes.

Un grupo de investigadores y docentes de la Universidad Rey Juan Carlos, en colaboración con la asociación sin ánimo de lucro Programamos, ha sido reconocido por Google con uno de sus premios RISE Award, un galardón que les permitirá continuar con su labor de investigación y enseñanza de la programación a niños.

Dr. Scratch
Analyze your Scratch projects here!

Etiquetas

EDUCACIÓN PROGRAMACIÓN

JÓVENES ACTUALIDAD

PREMIOS

PENSAMIENTO COMPUTACIONAL

DIFUSIÓN INFORMATICA

iEureka!

Dr. Scratch: aprendiendo a programar desde los tres años de edad

10 may 2015

Me gusta Compartir 132

Compartir: 11

1 comentario

Por Christian González

Que los niños aprendan inglés u otros idiomas desde pequeños ya no se considera una novedad, sino algo básico. Algo así podría pasar en pocos años con el verdadero idioma del futuro: **el lenguaje de la programación informática**. En muchos países, la programación ya forma parte del currículo en la educación secundaria o incluso en primaria. Los expertos consideran que fomentar el pensamiento computacional desde edades tempranas fomenta la creatividad y

iEureka!

Eureka

Como uno de los colectivos menos valorados por las instituciones y el tejido empresarial y muy afectado por la crisis económica, la mayoría de innovadores e investigadores españoles lucha cada día por sobrevivir en el mercado nacional, mientras otros no tienen más remedio que poner tierra de por medio y emigrar a otros países en busca de mejores oportunidades.

Público pone en marcha esta sección, con el objetivo de servir de escaparate para la divulgación de la riqueza de ideas que fluye en España. Nuestros investigadores son innovadores ingeniosos, pero también emprendedores valientes, con capacidad para generar empleo y riqueza.

Si eres investigador y quieres que se conozcan tus proyectos envíanos una propuesta a

Figura B.1: Dr. Scratch en la prensa.

Bibliografía

- [1] B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin. Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 215–220. ACM, 2013.
- [2] A. HOLOVATY and J. KAPLAN-MOSS. The django book: Version 2.0. *The Django Book*, 16, 2009.
- [3] J. Moreno and G. Robles. Dr. scratch, evaluación automática del desarrollo del pensamiento computacional analizando el código de proyectos scratch. *Virtual USATIC, Ubicuo y Social: Aprendizaje con TIC*, 2014.