áginas de ayuda.4.1318 áginas de ayuda.4.1418

UNIVERSIDAD
REY JUAN CARLOS

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

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

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

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

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?

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.

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.
- 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.
- Registro de usuarios. Crear cuentas de organizaciones, profesores y alumnos, donde se pueda llevar un seguimiento de los proyectos analizados.
- Análisis masivo de proyectos.

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

2.3. Planificación temporal

labelsec:planificacion-temporal

Capítulo 3

Estado del arte

Actualmente existen varias herramientas que analizan estáticamente el código fuente de programas escritos en distintos lenguajes para ofrecer retroalimentación al usuario:

- Pylint[1]: consiste en un analizado estático de código Python 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: Puntuacion general Pylint

- Scrape[2]: 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:

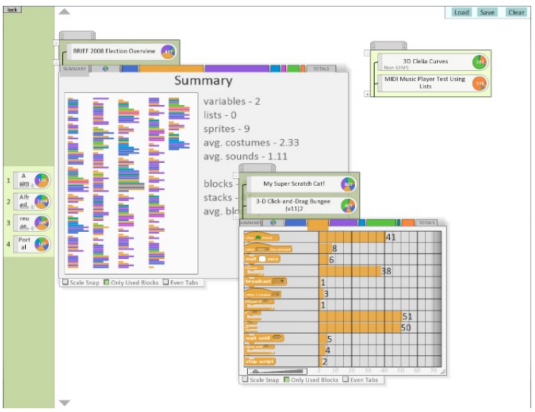

Figura 3.3: Vista general de Scrape

[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 Moreno[4]

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”. (FIXME: No sé si decir algo más.)

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[5]:

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

[3] `https://github.com/ucsb-cs-education/hairball`
[4] `https://github.com/jemole/hairball`
[5] `http://www.djangobook.com/`

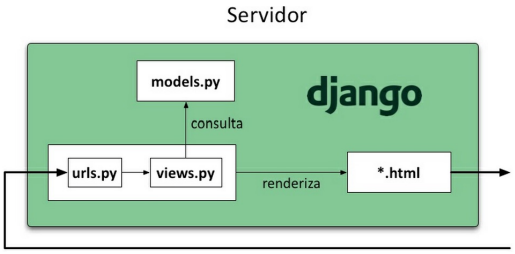

Figura 3.4: MVC de Django

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.[6]

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. (FIXME: No sé qué más decir.)

3.5. Hairball

(FIXME: Como lo he mencionado arriba no sé si volver a explicarlo.)

3.6. Scratch

Es un pseudo-lenguaje de programación basado en bloques, orientado a la enseñanza principalmente mediante la creación de videojuegos. 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.

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

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

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

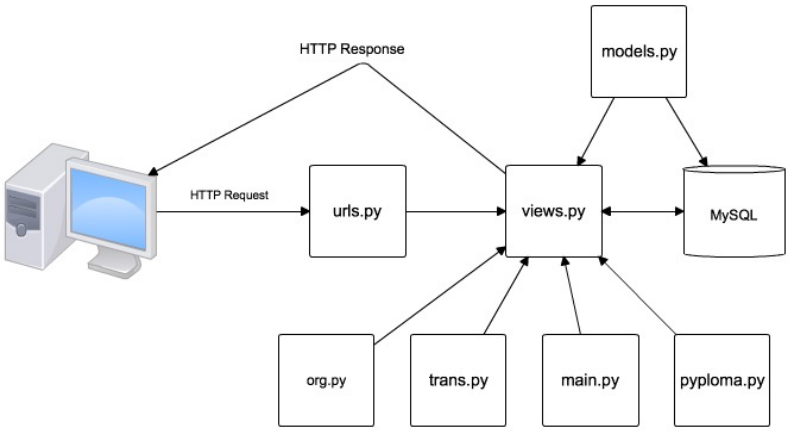

Figura 4.1: Arquitectura general

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 views.py asociada. Si no encuentra la url, optará por coger la última. Una vez encontrada la url y la función de views.py asociada, ejecuta dicha función, al final de la cual devolverá un HttpResponse y con ello una plantilla HTML.

4.2. Diseño e implementación del back-end

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 muestra una primera versión:

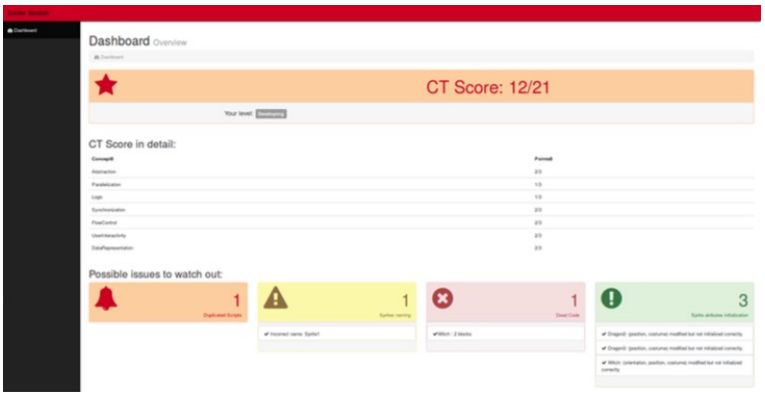

Figura 4.2: Primera version del dashboard.

Esta primera versión del dashboard se muestra para todos los niveles la misma información.

Después, se mostraban distintos dashboards y distinta información en función del nivel. Si el nivel era bajo se mostraba menos información y si el nivel era alto se mostraba más información:

(FIXME: Pedir los dashboards de Mari Luz).

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.

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.

Dicha página se actualiza cada noche mediante la creación de un nuevo comando en Django, ejecutando python manage.py mystats en la linea de comandos como una tarea programada. Esto se ha realizado mediante la creación de las carpetas y archivos siguientes: (FIXME: corregir esto -¿Duda Gregorio) app/
— management

$- {}_{i}nit_{.py\_commands|-}{}_{i}nit_{.py|\_mystats.py}$

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.[es]tá estructurada de la siguiente manera:

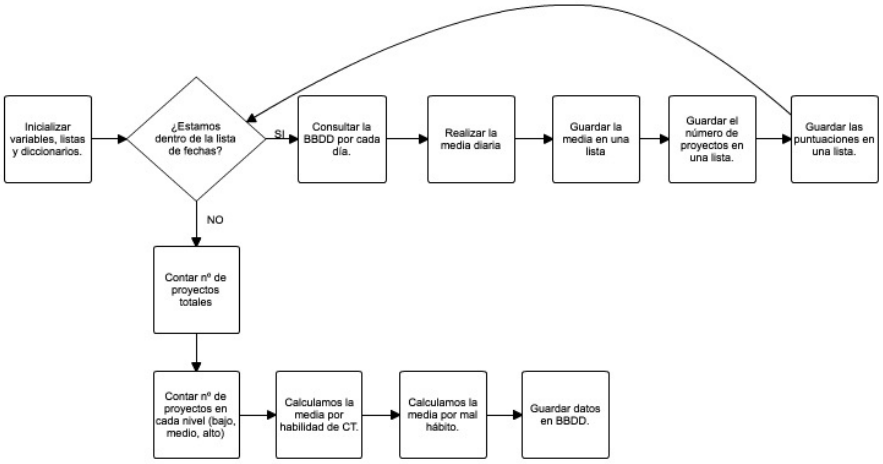

Figura 4.3: Esquema de la funcion Statistics.

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.

4.2.3. Organizaciones

- Registro de usuarios
- Restauración de contraseña
- Análisis masivo: csv(traducción de diccionarios)

4.2.4. 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([
    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 modificado 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:

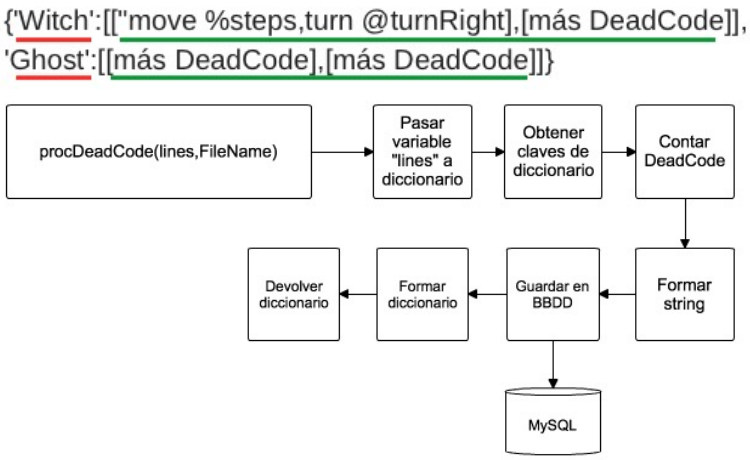

Figura 4.4: Esquema DeadCode.

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.

4.2.5. Foro

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(FIXME: existe algún ataque relacionado por recibir muchos comentarios con SPAM?Indicarlo aquí.)

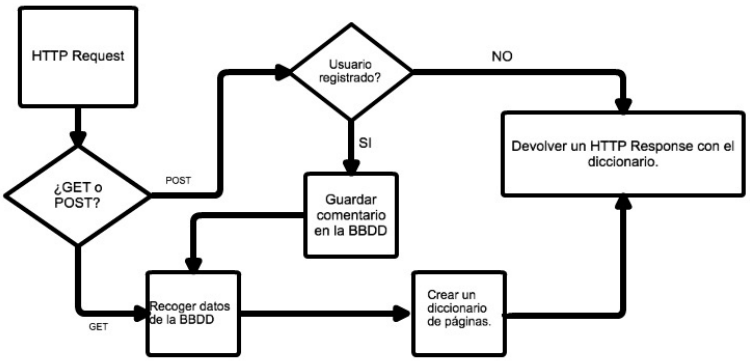

Figura 4.5: Esquema Foro.

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.

4.3. Diseño e implementación del front-end

4.3.1. Nuevos dashboards

Como se ha comentado en el apartado anterior, diseño e implementación del back-end, hemos diseñado nuevas plantillas para mostrar el resultado de los análisis. Inicialmente se mostraba lo siguiente:

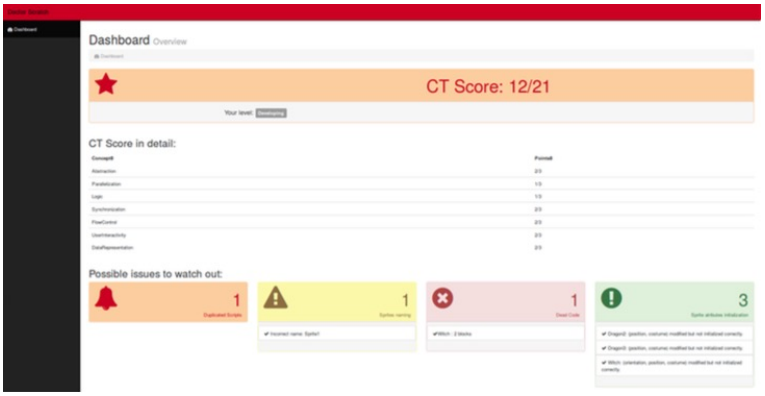

Figura 4.6: Primera version del dashboard.

En esa pantalla se mostraba la puntuación total sobre 21 de pensamiento computacional, 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 siguientes pantallas

(FIXME: incluir las versiones anteriores.)

Finalmente, rediseñamos los dashboards de tal manera que se simplificara lo máximo posible las pantallas con el fin de ver, de un vistazo, la información más importante.

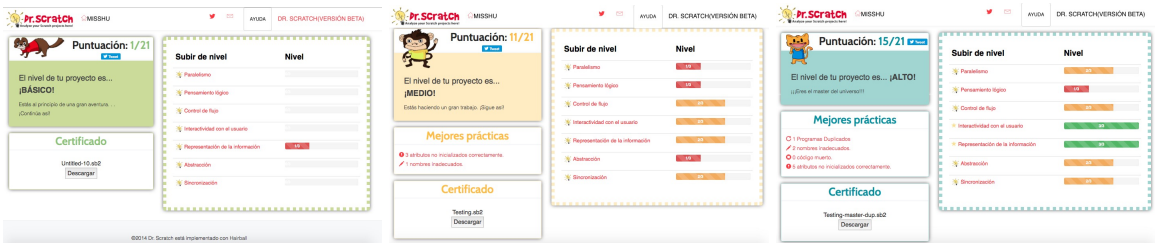

Figura 4.7: Ultima version del dashboard.

En las imágenes de arriba 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].

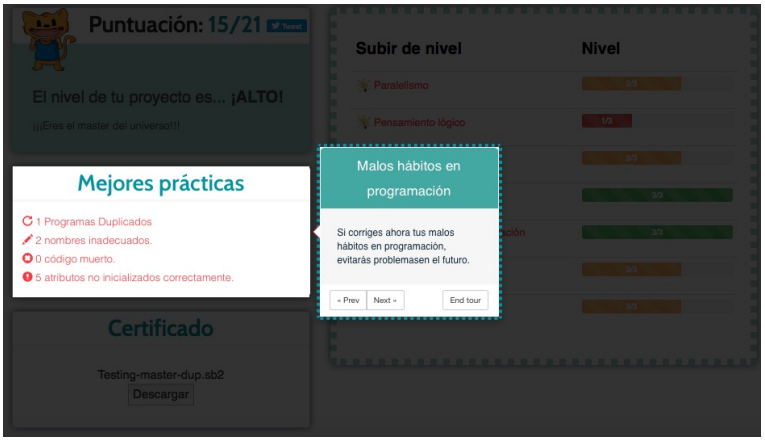

Figura 4.8: BootstrapTour.

- Posibilidad de publicar en Twitter tu puntuación [2]

Figura 4.9: Twitter.

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

[1] `http://bootstraptour.com`
[2] `https://about.twitter.com/es/resources/buttons`

Figura 4.10: Diploma.

Dicho diploma se forma gracias al script Pyploma (modificado) y mediante LaTeX.

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

Figura 4.11: Link para volver a Scratch.

- Popups que informan de las malas prácticas.

- 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 4.12: Gamificacion.

4.3.2. 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:

Figura 4.13: Acceso a páginas de ayuda.

En el ejemplo, si quisiéramos mejorar la habilidad Çontrol de flujo"pincharíamos sobre el y veríamos lo siguiente:

Figura 4.14: 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 bloque que se utilizan en el ejemplo. Para mostrar los bloques hemos utilizado una librería llamada Scratch blocks[3] 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:

[3] `https://github.com/tjvr/scratchblocks`

```
<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>
```

4.3.3. Estadísticas

Para mostrar las estadísticas generadas en forma de gráficas, optamos por usar Highcharts[4] 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

Figura 4.15: Puntuacion media diaria.

[4] http://www.highcharts.com

Figura 4.16: Gamificacion.

4.3.4. Organizaciones

4.3.5. Validación de formularios vía AJAX

4.3.6. Foro

4.4. Base de datos

Capítulo 5

Resultados

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

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

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

6.2. Aplicación de lo aprendido

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

1. Protocolos para la transmisión de audio y video en internet: en esta asignatura aprendí a programar en Python, entre otras cosas.

2. 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 los 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.

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

Apéndice A

Manual de usuario