Universidad
Rey Juan Carlos

ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

DR.SCRATCHJR

ANÁLISIS AUTOMÁTICO DE PROYECTOS SCRATCHJR

Autor : Sergio Pérez Mazadiego

Tutor : Dr. Gregorio Robles

Curso académico 2022/2023

Trabajo Fin de Grado

Dr.ScratchJr, Análisis Automático de Proyectos ScratchJr

Autor : Sergio Pérez Mazadiego

Tutor : Dr. Gregorio Robles

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2023, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2023

Dedicado a

mi familia y amigos

II

Agradecimientos

Agradecer a mis padres por el apoyo, la confianza y sobre todo por servirme lo que ellos no han podido tener, esa comodidad y estabilidad que siempre ha hecho que pueda tener una vida sin preocupaciones y llena de oportunidades.

Resumen

Este trabajo final de grado consiste en el desarrollo de una herramienta que analice proyectos de ScratchJr, un lenguaje de programación para niños de edades entre 5 y 7. La idea surge con el objetivo de hacer una herramienta similar a Dr.Scratch, que es una aplicación web que analiza proyectos de Scratch, otro lenguaje de iniciación a la programación.

Como resultado del trabajo fin de grado se ha creado una aplicación web llamada Dr.ScratchJr para que educadores, investigadores o padres puedan subir proyectos de niños que usen ScratchJr y vean un análisis de éstos. En el análisis de un proyecto se estudia la variabilidad en el uso de bloques, los malos hábitos que contiene, la creatividad y otros datos que podrían llegar a ser de interés como el total de páginas o personajes que se usan en él. La aplicación web ofrece la posibilidad de analizar uno, o varios proyectos a la vez, o crear una cuenta en la aplicación para poder almacenar los análisis de proyectos y tener una gestión de sus autores. En ambos casos la aplicación dispone de la opción de exportar los datos de los análisis en formato CSV.

Las tecnologías que se han empleado son: en el lado del servidor, el software de desarrollo Django en el que se utiliza Python, y Bootstrap en el lado del cliente en el que se utiliza HTML, CSS y JavaScript.

Summary

This final degree project consists on the development of a tool that analyzes ScratchJr projects, which is a programming language designed for children from 5 to 7 years old. The idea arises with the objective of making a tool similar to Dr.Scratch, which is a web application that analyzes Scratch projects, another programming initiation language.

As a result of this final degree project, a web application called Dr.ScratchJr has been created so that educators, investigators or parents can upload projects of children using ScratchJr and see an analysis of them. This análisis includes the study of the variability in the use of blocks, the bad habits it contains, the creativity and other interesting data such as the total number of pages or characters used in the project. The web application provides the option to analyze one or multiple projects simultaneously, and creating an account in the application that allows to store the analyses of the projects and manage their authors. In both cases, the application has the option of exporting the analysis data in CSV format.

The technologies that have been used are: on the server side, the Django development software utilizing Python, and on the client side, Bootstrap utilizing HTML, CSS, and JavaScript.

Índice general

Índice de figuras

Capítulo 1

Introducción

Las habilidades de programación son cada vez más necesarias que se empiecen a trabajar en las aulas desde los inicios en la educación, igual que se enseña la lengua, las matemáticas o la historia, puesto que favorecen la estructuración del pensamiento crítico y la lógica, esenciales en la resolución de problemas. Difícilmente se consigue enseñar a un niño de unos 5 o 6 años a que escriba código de programación. Pero sí es posible que comience a familiarizarse con el mundo de la programación sin que se dé cuenta, de una manera lúdica y al mismo tiempo formativa. Como en las primeras edades sus capacidades lectoescritoras todavía están formándose, por medio de herramientas vinculadas a elementos visuales como juegos, puzzles o aventuras, podemos arrancar el desarrollo del pensamiento computacional recurriendo a su creatividad. Algunas herramientas que podemos encontrar son las aplicaciones Kodable[1], Bit by bit[2], Codekarts[3] y ScratchJr[4].

De esta última, ScratchJr (que se introduce en detalle en la Sección 3.1), es donde parte este proyecto. ScratchJr está basada en Scratch pero con su interfaz y lenguaje de programación rediseñados para niños más pequeños. En la URJC se desarrolló la herramienta Dr.Scratch que permite a los usuarios de Scratch analizar sus proyectos y medir sus capacidades de programación cuantificadas por medidores de habilidades de pensamiento computacional. De esta misma idea surge Dr.ScratchJr. Pero debido a que los usuarios de ScratchJr, al ser demasiado pequeños, no tienen todavía la capacidad de poder auto-evaluar sus propios proyectos, el objetivo principal

[1] `https://www.kodable.com/`
[2] `http://rikaigames.com/bitbybit/`
[3] `https://civat.es/app/code-karts/`
[4] `https://www.scratchjr.org/`

de la herramienta Dr.ScratchJr es facilitar a los profesores o padres el proceso de enseñanza de sus alumnos o hijos. Con ella, estos o investigadores pueden analizar los proyectos que hayan hecho niños en ScratchJr y valorar el desarrollo de pensamiento computacional.

1.1. Estructura de la memoria

La memoria del trabajo está estructurada en los siguientes capítulos:

- Capítulo 1: **Introducción**. Capítulo actual donde se presenta el tema del que va a tratar el proyecto y se marca la estructura de la memoria.
- Capítulo 2: **Objetivos**. Partiendo del objetivo general, se plantean los objetivos que se han ido marcando para la realización del proyecto.
- Capítulo 3: **Estado del arte**. Aparecen pequeñas descripciones de las tecnologías utilizadas en la realización de la herramienta Dr.ScratchJr.
- Capítulo 4: **Diseño e implementación**. Se explica la arquitectura general de la herramienta Dr.ScratchJr, el algoritmo del tratamiento de datos para realizar el análisis y la vista de la aplicación web.
- Capítulo 5: **Experimentos, validación y resultados**. Se realizan pruebas con un proyecto de test para validar que la herramienta detecta todos los datos a analizar correctamente, una pequeña comprobación de la correcta carga de varios proyectos y otra de la exportación de los datos en CSV.
- Capítulo 6: **Conclusiones**. Se exponen varias secciones donde se trata la consecución de los objetivos, las asignaturas del Grado que me han servido para la realización del proyecto, aquello que he aprendido gracias a la elaboración de dicho proyecto y algunas ideas de la herramienta para futuras mejoras.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo principal de este proyecto es desarrollar una herramienta de análisis similar a Dr.Scratch para ScratchJr. En Dr.Scratch los usuarios pueden evaluar sus propios proyectos creados con Scratch, pero como los destinatarios de ScratchJr son niños, el objetivo es crear una aplicación web con la función de ser una herramienta para que profesores, investigadores o padres puedan analizar los proyectos de ScratchJr de los niños. De esta manera se facilita y agiliza el trabajo de evaluación y poder guiar a los niños a un mayor aprendizaje detectando los errores más frecuentes que suelen darse en la programación.

2.2. Objetivos específicos

Para cumplir con el objetivo principal ha sido necesario seguir los siguientes objetivos específicos:

- Comprender la herramienta Dr.Scratch.
- Ampliar y mejorar los conocimientos sobre la creación de aplicaciones web con Django e instalar las tecnologías necesarias para ello.
- Examinar la estructura de la aplicación ScratchJr: el uso y funcionamiento, los tipos de bloques y componentes que pueden emplearse y los ficheros que forman un proyecto.

- Estudiar el fichero JSON de datos que ScratchJr genera en cada proyecto.
- Extraer y almacenar en una base de datos los elementos y datos de los proyectos generados por ScratchJr.
- Crear los algoritmos para tratar y adaptar los datos para los distintos criterios del análisis en la aplicación web.
- Desarrollar la aplicación web para que se adapte a diferentes dispositivos con distintas resoluciones.
- Diseñar un sistema de administración que permita a profesores tener un control de sus alumnos y los análisis de los proyectos de estos.
- Implementar un sistema de exportación de archivos con los resultados de los análisis de los proyectos.
- Desplegar la aplicación web en un servidor web público.

Capítulo 3

Estado del arte

Para el desarrollo del proyecto se han utilizado una serie de tecnologías ya existentes que se describen a continuación.

3.1. ScratchJr

ScratchJr [8, 12] (figura 3.1) es un lenguaje de programación de iniciación que permite a los niños pequeños crear sus propias historias y juegos interactivos. Está inspirado en el lenguaje de programación Scratch[1] utilizado por millones de jóvenes en todo el mundo. ScratchJr ha rediseñado tanto la interfaz como el lenguaje de programación para adecuarlo al desarrollo de los niños más pequeños.

A través de la unión de bloques gráficos de programación, los niños hacen que los personajes se muevan, salten, bailen y canten. Los niños pueden modificar los personajes o crear nuevos en el editor de dibujos, donde incluso pueden subir sus propias fotos, y luego usar los bloques de programación para hacer que los personajes cobren vida dentro de distintos escenarios.

Las funciones están cuidadosamente diseñadas para adaptarse al desarrollo cognitivo, personal, social y emocional de los niños. Estos no sólo aprenden a programar, sino que además desarrollan destrezas que son fundamentales para su desarrollo académico.

ScratchJr está disponible como una aplicación gratuita para dispositivos iOS y Android.

[1] `https://scratch.mit.edu/about`

Figura 3.1: Aplicación ScratchJr.

Además, permite compartir los proyectos creados entre usuarios a través de correo electrónico o AirDrop.

3.2. Dr.Scratch

Dr.Scratch[2] [13] es una aplicación web gratuita de código abierto que permite, a educadores, alumnos o interesados en aprender programación, analizar automáticamente proyectos Scratch.

Utiliza la librería `Hairball`[3] para evaluar proyectos de Scratch y ofrecer una puntuación basada en el pensamiento computacional. Para ello, Dr.Scratch tiene en cuenta los siguientes siete aspectos: abstracción, pensamiento lógico, sincronización, paralelismo, control de flujo, interactividad con el usuario y representación de datos. Además de la puntuación, ofrece retroalimentación para mejorar su código en dichos aspectos y, dependiendo de la puntuación, muestra algunos malos hábitos de programación como el nombramiento inadecuado de perso-

[2] `http://drscratch.org/`
[3] `https://pypi.org/project/hairball/`

najes, repetición de código, código que nunca se ejecuta y la inicialización incorrecta de los atributos del objeto.

Para analizar un proyecto, Dr.Scratch permite cargar un archivo .sb o un .sb2, ya que la herramienta es compatible con las versiones 1.4 y 2.0 de Scratch, o directamente introduciendo la URL del proyecto.

3.3. Python

Python [7] es un lenguaje de programación de alto nivel, interpretado, dinámico, fuertemente tipado y multiplataforma. Se considera multiparadigma ya que soporta parcialmente la orientación a objetos, programación imperativa y programación funcional, además de otros paradigmas mediante el uso de extensiones.

Aunque generalmente los programas escritos en Python muestran un tiempo de ejecución mayor respecto a programas en otros lenguajes, para el programador Python requiere un menor tiempo de desarrollo gracias a su sencillez, claridad y versatilidad.

Está desarrollado bajo una licencia de código abierto, por lo que es de libre uso y distribución. Lo que supone que además de contar con una gran biblioteca estándar, exista una colección creciente de programas, módulos, paquetes o *frameworks*, disponibles en el Python Package Index[4].

3.4. Django

Django [3] es un framework de desarrollo web de alto nivel escrito en Python (sección 3.3). Se encarga de facilitar la creación de sitios web dinámicos de manera relativamente rápida y con un diseño limpio. Es de software libre y de código abierto.

Contiene un número reducido de ficheros y carpetas. Está basado en el principio DRY (Don't Repeat Yourself o No te repitas) para evitar escribir código duplicado e invertir el menor esfuer-

[4] `https://pypi.org/`

zo posible en patrones repetitivos para poderse centrar en la parte creativa.

Está inspirado por su versatilidad y simpleza en el patrón de arquitectura software MVC (Modelo-Vista-Controlador), pero Django redefine este patrón en Modelo, Vista y Plantilla. Lo que se llamaría Controlador en un verdadero framework MVC, se llama Vista en Django, y lo que se llamaría Vista, se llama Plantilla.

- **Modelo**: es la capa de abstracción en código (`models.py`) de la base de datos, manejada como un objeto encargado de la interacción y comunicación con la base de datos.
- **Vista**: es la capa designada a la lógica de la página web, a través de la que pasarán los datos del modelo a la plantilla.
- **Plantilla**: se encarga de presentar al usuario la información disponible en un formato adecuado.

Django incluye una capa de mapeo relacional de objetos (ORM) predeterminada que se puede usar para interactuar con datos de aplicaciones de varias bases de datos relacionales, sin necesidad de conocer los comandos de la base de datos. Viene con SQLite3 por defecto, pero es posible disponer de otro motor de base de datos como PostgreSQL, MySQL o Oracle.

Aporta un servidor propio para la parte de desarrollo de una aplicación, pero se puede montar sobre otros servidores para la parte de producción.

También se toma la seguridad muy en serio y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como el método de infiltración de código intruso (SQL injection), las secuencias de comandos en sitios cruzados (Cross-site scripting o XSS), la falsificación de solicitudes entre sitios (Cross Site Request Forgery o CSRF) y el secuestro de clic (Clickjacking). Además, su sistema de autenticación de usuarios proporciona una forma segura de administrar las cuentas y contraseñas de los usuarios.

3.5. Django-registration

Django-registration [10] es una aplicación extensible que proporciona la funcionalidad de registro de usuarios para sitios web con tecnología Django (sección 3.4).

Aunque casi todos los aspectos del proceso de registro son personalizables, proporciona vistas y mecanismos predefinidos para dos casos de uso comunes:

- **Registro en dos fases:** consiste en el registro inicial seguido de un correo electrónico de confirmación con instrucciones para activar la nueva cuenta.

- **Registro de una fase:** la activación del usuario registrado y el inicio de sesión es inmediato.

3.6. HTML5 (HyperText Markup Language)

HTML5 [5] es la última versión de HTML, la quinta, lanzada en 2014. Es un lenguaje de marcado que se utiliza para estructurar y desplegar páginas web.

Ofrece una gran adaptabilidad, una estructuración lógica y facilidad de interpretación. Un documento HTML se compone en base a etiquetas, también llamadas marcas o tags, con las cuales conseguimos expresar el contenido (elementos) de una página web (cabecera, cuerpo, encabezados, párrafos, archivos...). Los elementos tienen dos propiedades básicas: atributos y contenido. Los atributos del elemento están contenidos en la etiqueta de inicio del elemento y el contenido está ubicado entre esta etiqueta y la de cierre.

Es multiplataforma siendo el estándar más utilizado por los navegadores web actuales. Además, ayuda a los motores de búsqueda a entender mejor el contenido de la web mediante sus etiquetas categorizadas, favoreciendo al posicionamiento en buscadores o SEO (Search Engine Optimization).

Dispone de etiquetas que ofrecen contenido multimedia (audio y vídeo) para poder reproducirlo en el navegador. También posee variedad de elementos para introducción de datos en formularios.

Incorpora APIs (Application Programming Interfaces o Interfaces de programación de aplicaciones) que permiten incluir, a través de JavaScript (sección 3.8) componentes complejos.

3.7. CSS (Cascading Style Sheets)

CSS [2] es un lenguaje que sirve para crear páginas web o interfaces visualmente atractivas. Permite describir las propiedades de diseño de un contenido estructurado en un lenguaje de marcado. Generalmente este contenido consiste en un archivo HTML, pero CSS también es perfectamente compatible con otras tecnologías como XML.

Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón de herencia) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

Con CSS se logra separar del archivo HTML, los contenidos y el formato. Esta separación puede darse dentro del propio archivo HTML o separándolo en una hoja de estilo .css en un fichero independiente. La separación permite modificar la estética de la página sin tener que modificar el contenido, reutilizar una sola hoja de estilos para varios archivos HTML, y reducir la complejidad y la repetición de código en la estructura del archivo.

3.8. JavaScript

JavaScript (JS) [11] es un lenguaje de programación interpretado, dinámico y débilmente tipado. Multiparadigma ya que es orientado a objetos, utiliza programación basada en prototipos, imperativa y declarativa (funcional).

Es la base de multitud de frameworks gracias a su lenguaje sencillo pero potente. Generalmente es usado en lado del cliente en las aplicaciones web, pero también se emplea para enviar y recibir información del servidor.

Integrado dentro del motor de los navegadores web más populares, favoreciendo la rápida ejecución de su sintaxis. Se relaciona de modo fluido y transparente con HTML y CSS, ofreciendo más dinamismo a las páginas web.

3.9. Bootstrap

Bootstrap [1] (figura 3.2) es un framework front-end utilizado para desarrollar sitios y aplicaciones web. Tiene como objetivo que el diseño web sea más rápido y sencillo, siendo compatible con la mayoría de navegadores web.

Contiene componentes preconstruidos de diseño, tipografías, tablas, imágenes, formularios, botones y otros elementos de presentación basados en HTML y CSS. Además, cuenta con una librería JavaScript integrada con la finalidad de brindar elementos adicionales que utilicen los componentes de manera ágil y eficiente para mejorar en la interfaz de usuario.

Bootstrap es de software libre y de código abierto. Se puede descargar compilado (archivos CSS y JavaScript) o a través del código fuente original (Sass y JavaScript), pero también se puede hacer uso de Bootstrap sin descargarlo ya que los archivos necesarios se encuentran en servidores.

Destaca por la posibilidad de crear un diseño web responsive o adaptativo. Creando interfaces que se ajustan de manera automática a cualquier dispositivo, tamaño de pantalla y resolución. Mediante reglas *media queries* adapta la representación del contenido a características del dispositivo haciendo uso del sistema de rejillas (Grid System) de forma dinámica. El sistema de rejillas de Bootstrap utiliza una serie de contenedores, filas y columnas para diseñar y alinear el contenido de manera flexible. Las filas deben colocarse dentro de un contenedor para una alineación y un relleno adecuados, y cada fila puede hasta dividirse en doce columnas.

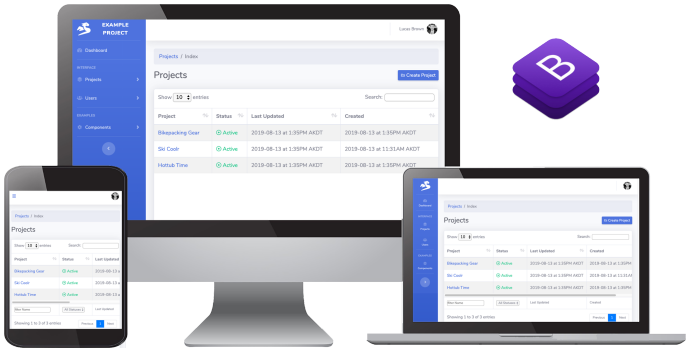


Figura 3.2: Bootstrap.

3.10. Pandas

Pandas [6] es una biblioteca construida sobre el lenguaje de programación Python de manipulación y análisis de datos. Es una herramienta de código abierto, rápida, potente, flexible y fácil de usar. Proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento.

Dispone de tres estructuras de datos diferentes:

- **Series:** estructura de una dimensión cuyos elementos tienen que ser del mismo tipo.
- **DataFrame:** estructura de dos dimensiones en formato ancho o largo (tablas). Contiene dos índices (cada uno puede ser de un tipo), uno para las filas y otro para las columnas, y se puede acceder a sus elementos mediante los índices o los nombres de las filas y las columnas.
- **Panel:** estructura de tres dimensiones (cubos).

Algunas características importantes de esta biblioteca son:

- Lectura y escritura de datos (CSV, Excel, SQL, JSON, Parquet...).
- Inserción y eliminación de columnas en estructuras de datos.
- Reestructuración y segmentación de conjuntos de datos, y manejo integrado de los datos faltantes.
- Facilita el manejo de series temporales: generación de rangos de fechas y conversión de frecuencias, desplazamiento de ventanas estadísticas y de regresiones lineales, desplazamiento de fechas...
- Hace más amigable el uso de NumPy (software de análisis numérico).

3.11. Gettext

El módulo gettext [4] de Python proporciona una implementación compatible con la biblioteca original gettext de GNU[5]. Es utilizado para proporcionar servicios de internacionalización (I18N) y localización (L10N) para módulos y aplicaciones Python como Django (incluido en django.utils.translation), mediante la traducción de cadenas de texto. Proporciona herramientas para extraer, traducir y generar archivos de catálogo de mensajes (.mo) a partir de archivos de plantillas (.po).

Separa la programación de la traducción. De esa manera, la extracción de mensajes, su traducción y su reintegración con el código es mucho más fácil y menos propensa a errores.

3.12. Visual Studio Code

Visual Studio Code [9] es un editor de código fuente ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio.

Es gratuito, de código abierto y permite ajustar su instalación a las tecnologías de desarrollo que le interese al usuario. Posibilita la descarga y gestión de extensiones con las que podemos personalizar y potenciar esta herramienta, como lenguajes, depuradores y complementos. Incluye personalización de atajos de teclado, tema del editor y la configuración de preferencias.

Viene con soporte incorporado para JavaScript, TypeScript y Node.js, pero gracias a las extensiones también agrega soporte para otros lenguajes como C ++, C#, Java, Python, etc.

Además, contiene soporte para la depuración, resaltado de sintaxis, recomendaciones de autocompletado de código, refactorización de código y control integrado de Git.

[5] `https://www.gnu.org/software/gettext/`

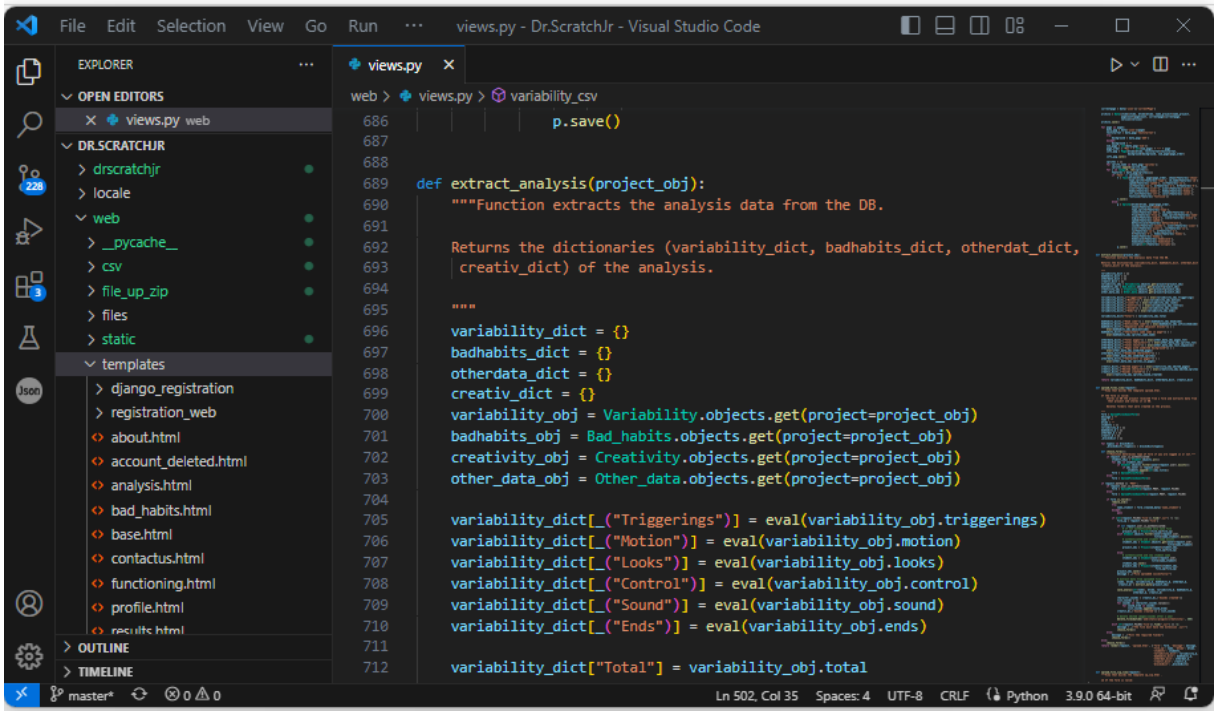


Figura 3.3: Captura de pantalla del código de Dr.ScratchJr en Visual Studio Code.

Capítulo 4

Diseño e implementación

En este capítulo se describe la arquitectura y cómo se ha diseñado e implementado la aplicación Dr.ScratchJr.

4.1. Arquitectura general

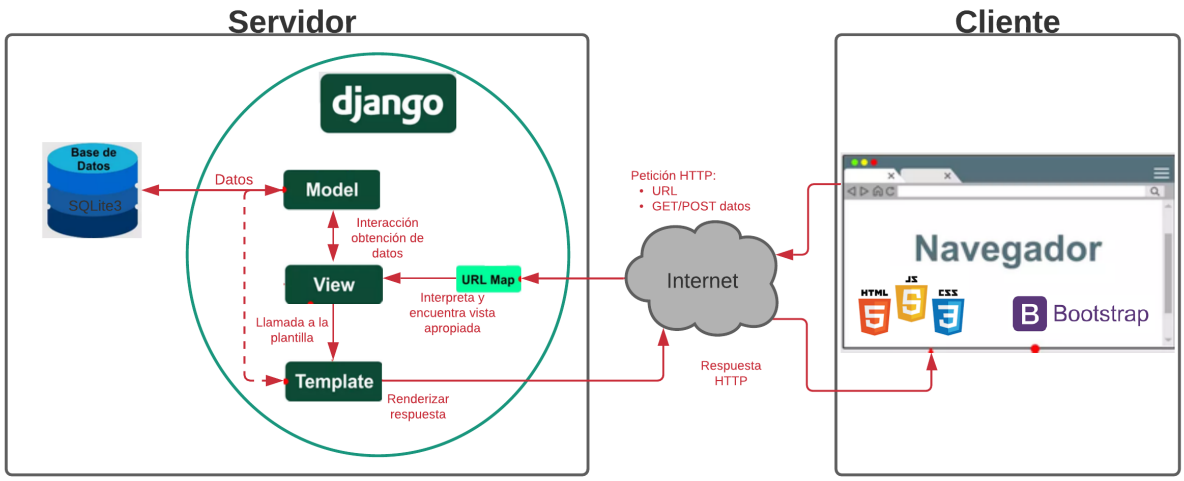


Figura 4.1: Arquitectura de Dr.ScratchJr.

La arquitectura que sigue Dr.ScratchJr es la del modelo cliente-servidor. El usuario, a través del navegador web, se conecta a la aplicación web Dr.ScrtachJr. Subirá uno o varios proyectos mediante los formularios que dispone. El navegador manda una petición HTTP con una URL que es interpretada por el mapa de URLS (URLmap o URLconf) para encontrar la vista apropiada. La vista (*View*) valida los datos de los formularios, estructura esos datos e interactúa con

el modelo para almacenar y obtener datos. El modelo (*Model*) define la estructura y Django se encarga de comunicarse con la base de datos. La plantilla (*Template*) recibe los datos de la vista y luego los organiza para la presentación al navegador a través de una respuesta HTTP.

4.2. Algoritmo aplicación

El algoritmo que se ha llevado a cabo para la realización de la función principal de la aplicación Dr.ScratchJr, que es analizar proyectos, comienza con la extracción de los datos. ScratchJr tiene la opción de compartir proyectos vía correo electrónico o AirDrop, lo que genera un archivo con extensión .sjr. De este archivo SJR es de donde partimos. Siendo este un archivo comprimido Zip que puede contener carpetas y archivos de recursos dependiendo de los elementos que se usen en la aplicación ScracthJr:

- Una carpeta *backgrounds* que contiene imágenes SVG de los fondos de páginas que han sido editados/creados con el editor de ScratchJr. No se genera si no se ha usado el editor en el proyecto.
- Una carpeta *characters* que contiene imágenes SVG de los personajes (objetos) que han sido editados/creados con el editor de ScratchJr. Tampoco se genera si no se ha usado el editor.
- Una carpeta *sounds* que contiene los archivos de audio WAV de los personajes (objetos) que han sido creados con el grabador de ScratchJr. No se genera si no se han producido grabaciones.
- La carpeta *thumbnails* contiene una imágen PNG con la vista de la página que está situada en última posición en un proyecto ScratchJr.
- Un fichero *data* que es un archivo JSON con todos los datos del proyecto. Un ejemplo de su jerarquía de datos se muestra en la figura 4.2.

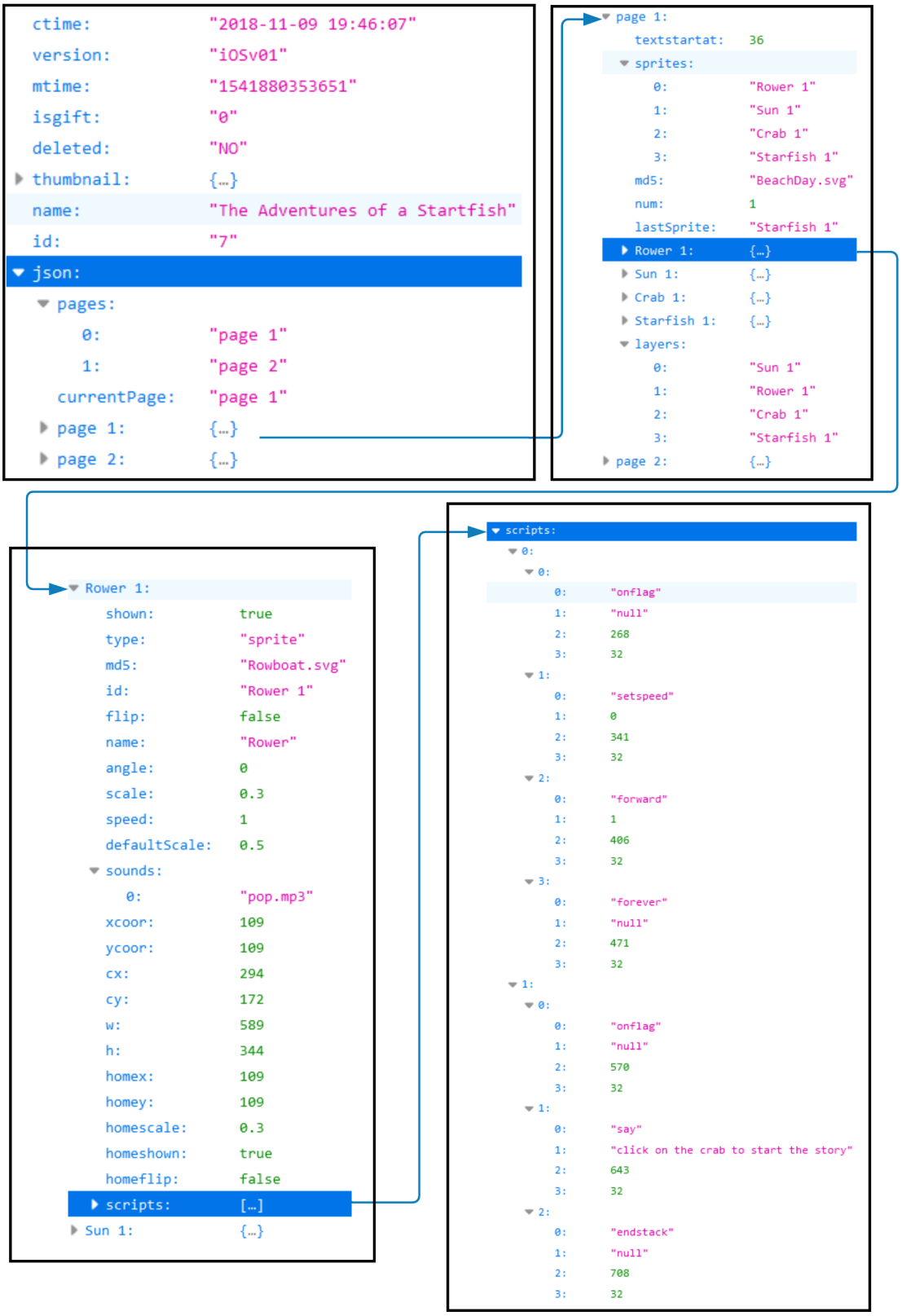


Figura 4.2: Ejemplo fichero data.json.

4.2.1. Procedimiento

El procedimiento que se ha seguido para obtener los datos necesarios ha sido:

- Descomprimir el archivo zip.
- Copiar el contenido de las carpetas *characters*, *backgrounds* y *sounds* en local.
- Obtener del archivo *data.json* todos los valores que pueden llegar a ser de utilidad en futuros tipos de análisis de proyectos y almacenarlos en la base de datos (figura 4.3).
- Adaptar dichos datos para realizar las distintas variedades del análisis.

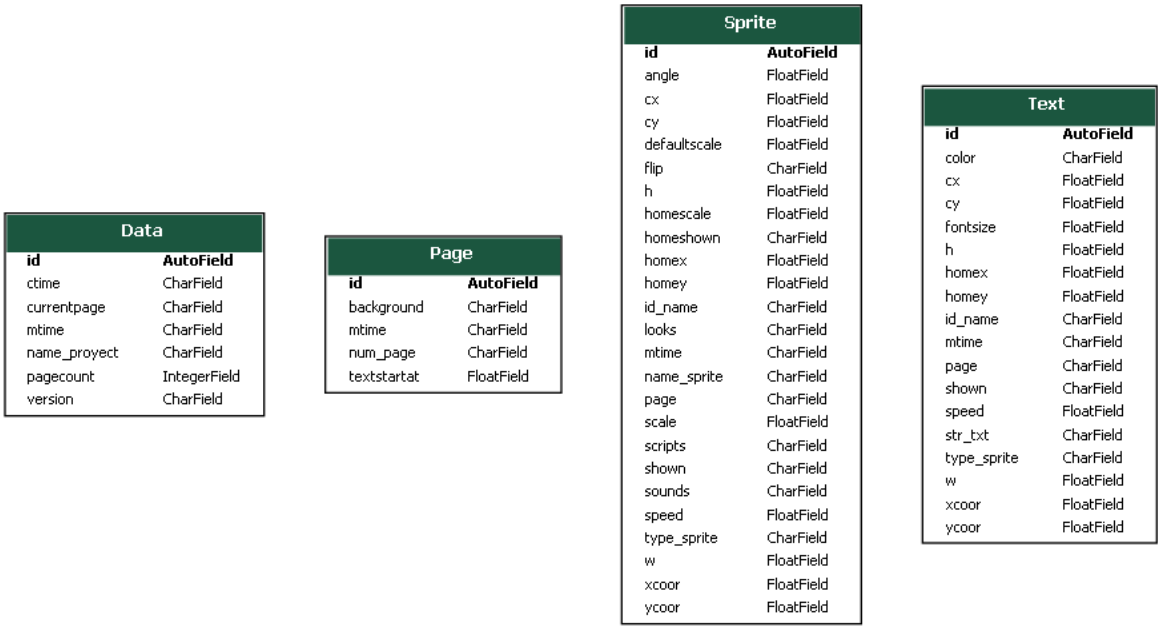


Figura 4.3: Datos de la estructura JSON en la base de datos.

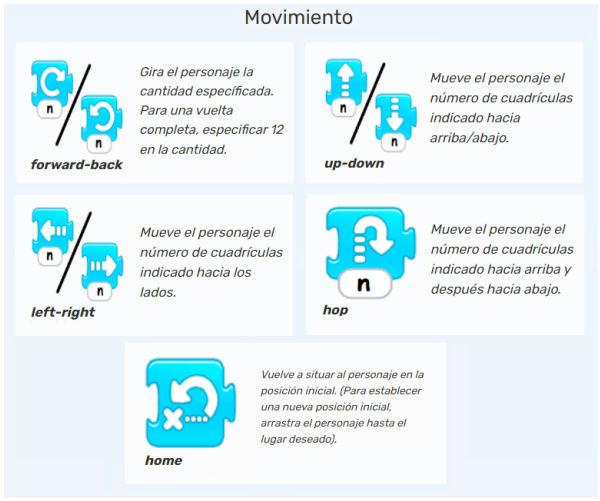
4.2.2. Criterios

Para realizar dicho análisis se han llevado a cabo los siguientes criterios: Variabilidad, Creatividad, Malos hábitos y Otros datos.

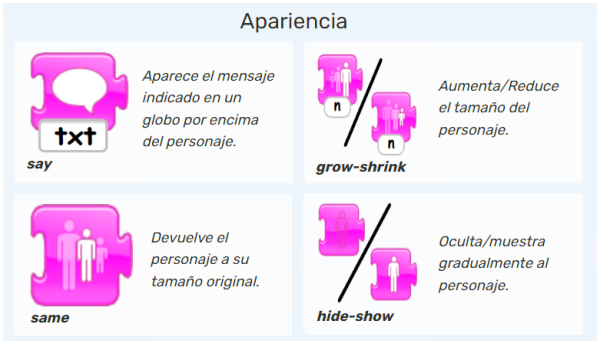
- En el aspecto *Variabilidad* se ha querido evaluar el uso de bloques en el proyecto, es decir la cantidad de bloques distintos que se han utilizado. Para ello se distinguen, como en la aplicación ScratchJr, cinco tipos de bloques: Eventos (figura 4.4a), Movimiento (figura 4.4b), Apariencia (figura 4.4c), Sonido (figura 4.4d), Control (figura 4.4e) y Finalización (figura 4.4f). El algoritmo devuelve, para cada uno de los tipos, los bloques que se han empleado al menos una vez en todo el proyecto.

  En Dr.ScratchJr se ha tomado como la misma clase de bloque, como se puede observar en las figuras 4.4b y 4.4c, aquéllos que hacen la misma función pero de manera opuesta. Como es el caso de los bloques *Girar a la derecha* y *Girar a la izquierda*, *Mover a la derecha* y *Mover a la izquierda*, *Subir* y *Bajar*, *Crecer* y *Disminuir*, y *Mostrar* y *Ocultar*.

(a) Evento



(b) Movimiento



(c) Apariencia

(d) Sonido

(e) Control

(f) Finalización

Figura 4.4: Tipos de bloques en Dr.ScratchJr

- Para el criterio *Malos hábitos* se tienen en cuenta algunos errores que se comentan a continuación en las siguientes funciones del código (retornan tanto listas como diccionarios que posteriormente son utilizados para mostrar el análisis completo):

  - *dead_code(adapted_dict):* función que devuelve una lista con las secuencias de bloques que no comiencen con un bloque de Eventos. También están incluidas en la lista las secuencias que comienzan con el bloque "Comenzar con mensaje" de un color, pero solo cuando no aparece en el proyecto ningún bloque "Enviar mensaje" del mismo color, puesto que esa secuencia nunca se ejecutaría y debe ser considerado código muerto.

    Para ello, el algoritmo busca el primer bloque de cada secuencia y si no coincide con ninguno de los bloques de Eventos (figura 4.4a), se añade a la lista la secuencia junto con el personaje que la utilice y la página donde se encuentre. En el caso de coincidir el primer bloque de una secuencia con el bloque "Comenzar con mensaje" de un color, se busca en todas las secuencias de todo el proyecto si existe el bloque "Enviar mensaje" de ese mismo color, y si no se encuentra, se añade la secuencia a la lista del mismo modo que se ha comentado antes.

  - *unfinished_code(adapted_dict):* localiza el último bloque de cada secuencia del proyecto y si éste no es un bloque de Finalización (figura 4.4f), añade dicha secuencia, junto con el personaje que la utilice y la página donde se encuentre, a una lista que es devuelta al finalizar la búsqueda en todas las secuencias.

  - *equal_adjac_blocks(adapted_dict):* busca secuencias con bloques repetidos unidos que puedan ser reemplazados por un solo bloque con atributos. Los bloques con atributos son aquellos que se puede indicar el número de veces que se quiere que el personaje haga la acción que indican dichos bloques. Por ejemplo, un personaje que utiliza el bloque Subir con atributo 3, hace que dicho personaje se desplace tres cuadriculas hacia arriba.

    La función también busca las secuencias que se encuentran dentro de un bloque "Repetir". Si encuentra en algún caso bloques repetidos unidos añade la secuencia entera a una lista que es devuelta al acabar la búsqueda.

- *sprites_same_name:* el algoritmo busca si en una página hay más de un personaje con el mismo nombre y si es así, se añade el número de ocasiones que se repite el nombre del personaje como valor en un diccionario cuya clave es el nombre de personaje en concreto. Este diccionario, a su vez, se añade como valor de otro diccionario ("sprites_same_name") y con clave la página en la que se encuentran los personajes del mismo nombre.

  Ej: {'Pos 1-page 1': {'Bat': '3 ocasiones'}}.

- Para el criterio *Creatividad* se busca si el usuario ha generado nuevo contenido respecto al contenido que proporciona la aplicación ScratchJr.

  El propio programa de ScratchJr aplica el algoritmo MD5 a sus archivos multimedia que tiene por defecto para saber si han sido modificados. Los archivos que han sido modificados por el usuario son nombrados con 32 dígitos hexadecimales. Entonces el algoritmo de Dr.ScratchJr considera imágenes editadas o grabaciones de audio, a aquellas cuyos archivos tienen un nombre con una longitud mayor a 30 caracteres. Estos datos modificados se almacenan en los siguientes diccionarios.

  - *edited_pages:* (páginas editadas) es un diccionario en el que se almacenan como claves las páginas que tienen una imagen de fondo editada y como valores los nombres de las imágenes de fondo.
  - *edited_sprites:* (personajes editados) es un diccionario en el que se almacenan como claves los personajes que han sido editados y como valores los nombres de las imágenes de los personajes editados.
  - *sprites_sound_created:* (personajes con grabaciones) es un diccionario en el que se almacenan como claves los personajes que tienen alguna grabación de audio creada y como valor de cada clave una lista con los nombres de todos las grabaciones de audio del personaje.

- En el apartado *Otros datos* se señalan los siguientes parámetros:

  - *sprites_tot:* lista con los nombres de todos los personajes. Los nombres de los personajes tienen esta estructura:

“ ‘**nombre del personaje**’ (‘**nombre identificador del texto**’) ”

- *pages_tot:* lista con los nombres de todas las páginas. Los nombres de las páginas tienen esta estructura:

  “Pos **n1** - page **n2** ”

  Siendo ‘n1’ el número que corresponde a la posición que tiene la página en Scratch cuando fue guardado el proyecto y ‘n2’ el número que corresponde al orden en el que fue creada esa página por el usuario en ScratchJr.

- *unedited_sprites:* lista con todos los personajes que no han sido editados. Estos son almacenados con los nombres de igual forma que la estructurada en la lista “sprites_tot”.

- *unedited_pages:* lista de todas las páginas que no han sido editadas, dichas páginas guardadas con el nombre de igual manera que se hacía en la lista “pages_tot”.

- *sprites_in_pages:* diccionario que tiene como claves los nombres de todas las páginas, y como valor de cada clave una lista con el nombre de los personajes que aparecen en esa página.

- *text_sequences:* lista que tiene como elementos tantas listas como textos haya en el proyecto. Cada lista tiene la siguiente estructura:

  “ [‘Pos **n1** - page **n2** ’, ‘**nombre identificador del texto**’, [‘str_txt’, ‘**cadena del texto**’], [‘fontsize’, ‘**tamaño de fuente**’]] ”.

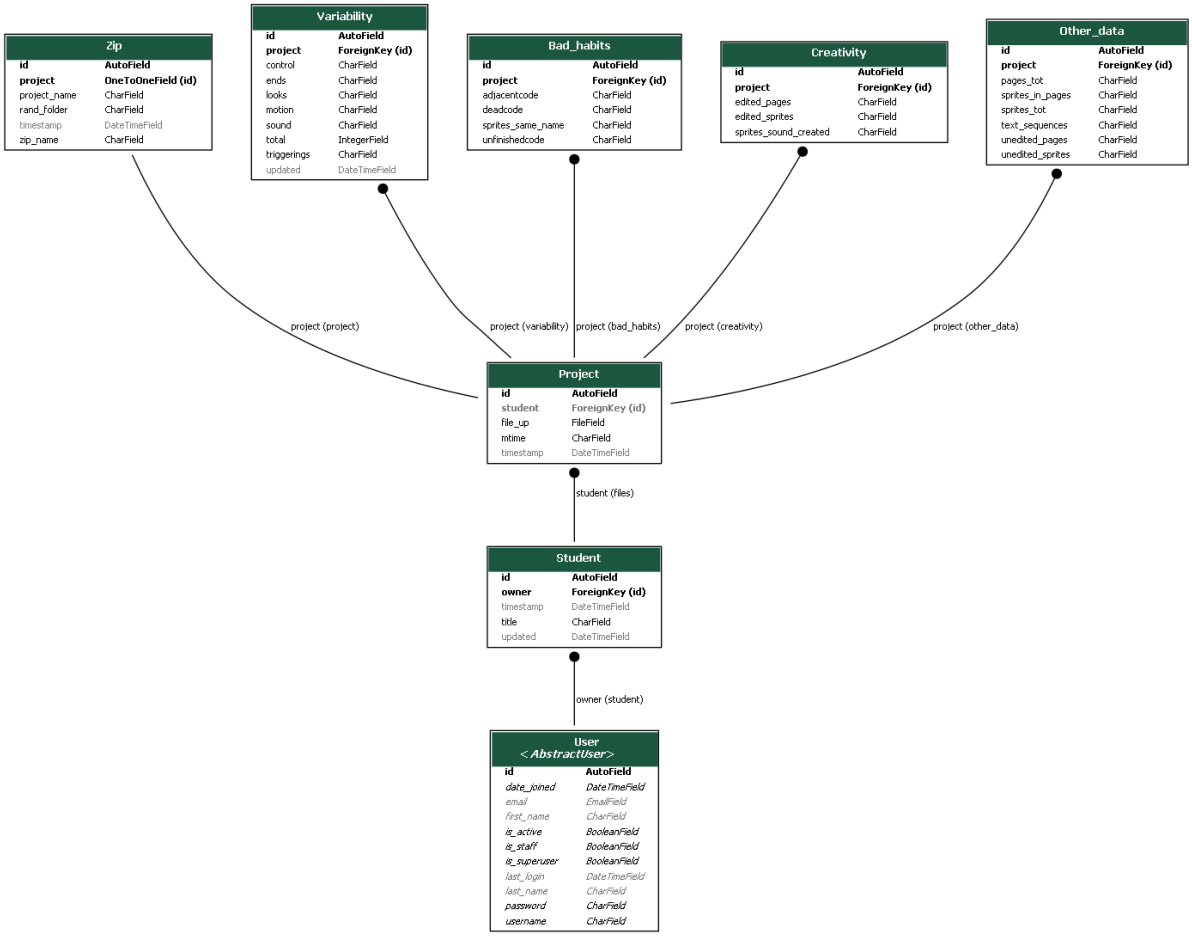


Figura 4.5: Esquema entidad-relación de los criterios del análisis.

Una vez el algoritmo ha extraído los datos para su análisis, se almacenan y se introducen en diccionarios que se utilizan como contexto para representar (renderizar) las plantillas que muestran los resultados del análisis. Estos diccionarios se almacenan igualmente en base de datos (figura 4.6) de manera temporal para posteriormente utilizarlos también para renderizar las plantillas que muestran los detalles de un tipo de análisis en concreto.

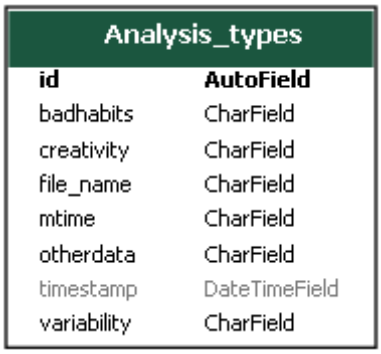


Figura 4.6: Tabla análisis en la base de datos.

El proceso para el análisis de múltiples proyectos es el mismo salvo que en lugar de representarse primero la plantilla del análisis de un proyecto, se renderiza una plantilla con un listado de los proyectos subidos junto al estudiante asociado a ese proyecto (figura 4.7). A partir de ese listado se puede ver el análisis de cada proyecto individualmente.

Figura 4.7: Ejemplo de un listado del análisis de un zip.

En cualquier listado con proyectos en la aplicación web Dr.ScratchJr existe, gracias a la biblioteca Pandas (sección 3.11), la posibilidad de exportar los datos del análisis de los proyectos. Al exportar se genera un zip, los datos se almacenan en tres ficheros CSV (badhabits-1, otherdata-1, variability-1) y en una hoja de cálculo de Microsoft Excel llamada data.xlsx con los mismos datos de los tres ficheros CSV, pero agrupados en pestañas en un solo documento.

Dr.ScratchJr dispone de la opción de cambiar a su pagina web el idioma a español o a inglés gracias al módulo `gettext` (sección 3.11), disponible en Django. Para ello, se han extraído

las cadenas de texto y se han traducido en los archivos de traducción .po que hay para cada uno de los lenguajes.

4.3. Aplicación web

Al acceder a la aplicación web sin estar identificado desde un ordenador se muestran dos cuadros donde el usuario puede introducir uno o varios proyectos a la vez para que sean analizados, una breve presentación donde sugiere que el usuario se registre para poder almacenar los análisis que suba a la herramienta y un panel lateral en el lado izquierdo para ayudar al usuario.

El primer cuadro sirve para cargar un proyecto de ScratchJr, con extensión .sjr, a fin de visualizar su análisis al instante. En el caso de que se esté registrado en Dr.ScratchJr aparecerá también un recuadro donde se podrá asociar dicho proyecto con su autor, si este ya existe, o crear uno nuevo, y poder almacenar su análisis para posteriores visualizaciones.

El segundo cuadro da la posibilidad de cargar varios proyectos de una sola vez. Para ello, es necesario haber almacenado previamente los proyectos con extensión .sjr en un archivo comprimido .zip. Y es este archivo comprimido el que hay que cargar en este cuadro. Los archivos deben de ser nombrados con la estructura "**'Nombre del proyecto'**-**'Nombre del estudiante'**.sjr" para poder ser almacenados correctamente y poder asociarlos a sus autores. Al cargar el zip y pinchar en analizar se muestra una página con un listado de los proyectos que contiene el zip con la opción de modificar el nombre con el que se desea guardar el proyecto en la aplicación. En el caso de haber iniciado sesión, también existe la opción de cambiarle el nombre al estudiante en cada proyecto (figura 4.8).

(a) Sin loguear (b) Logueado

Figura 4.8: Página tras subir un zip de proyectos.

4.3.1. Direcciones URLs de la aplicación

La tabla 4.1 hace un breve resumen de la funcionalidad de las vistas que están asociadas a las URLs de la aplicación web Dr.ScratchJr.

Path URL	**Vista**	**Función**
“ ”	home_view	Página principal de la aplicación con formularios para cargar proyectos
profile/	profile_view	Información sobre el usuario registrado y el número de estudiantes que tiene almacenados. Permite modificar algunos datos del usuario
settings/	settings_view	Opciones para cambiar o restablecer contraseña, eliminar cuenta o cerrar sesión
functioning/	functioning_view	Información sobre el manejo por Dr.ScratchJr
contactus/	contactus_view	Formulario para contactar con responsables de Dr.ScratchJr
about/	about_view	Información como correo y enlace a Github de Dr.ScratchJr
variability/	variability_view	Información sobre el criterio Variabilidad
bad_habits/	bad_habits_view	Información sobre el criterio Malos hábitos
create_csv_zip/ <str:zip_name>/ <str:rand_folder>/	create_csv_zip	Vista que no renderiza ninguna página, sino que crea un fichero CSV con los datos de los proyectos de un zip
create_csv_student/ <str:student>/	create_csv_student	Vista que no renderiza ninguna página, sino que crea un fichero CSV con los datos de los proyectos de un estudiante
create_csv_user/	create_csv_user	Vista que no renderiza ninguna página, sino que crea un fichero CSV con los datos de todos los proyectos de todos los estudiantes que tenga almacenado un usuario
albums/upload/	upload_files_view	Si no se ha cargado ningún proyecto previamente aparecerá un formulario para cargar un proyecto. En el caso de que si, muestra el resultado del análisis

Path URL	**Vista**	**Función**
albums/upload_zip/	upload_file_zip_view	Si no se ha cargado ningún archivo zip previamente con proyectos aparecerá un formulario para cargar un zip. En el caso de que se acabe de subir el zip, muestra un listado de los proyectos con opción a renombrarlos. Y una vez finalizado muestra una tabla con los proyectos que contiene el zip con opción de ver cada análisis individualmente
analysis/ <student>/ <name_file>/	analysis_view	Análisis del proyecto perteneciente a un estudiante
analysis2/ <student>/ <project>/ <file_name>/ <rand_folder>/	analysis2_view	Muestra el análisis de un proyecto individualmente cuando se accede desde el listado de proyectos al cargar un zip
results/ <file_name>/ <mtime>/ <type1>/ <type2>/	results_view	Muestra con más detalles el resultado de un criterio del análisis
review/	review_view	Tabla de los estudiantes que están almacenados por el usuario, junto con el número de proyectos de cada uno. Opciones de editar el nombre, eliminar estudiantes y exportar todos los proyectos
students/ <str:student>/	student_view	Tabla con los proyectos de un estudiante en concreto. Opciones de editar el nombre, eliminar proyectos y exportar todos los proyectos
delete/ <str:student>/ <str:file_name>/ <str:times>/	delete_regist	Misma tabla que en “student_view”, pero sin el proyecto eliminado

Path URL	**Vista**	**Función**
delete_student/ <student>/	delete_student	Misma tabla que en "review_view", pero sin el estudiante eliminado
edit_student/ <old_student>/ <new_student>/	edit_student	Misma tabla que en "review_view", pero con el nombre del estudiante modificado
edit_file/ <old_file>/ <student>/ <new_file>/	edit_file	Misma tabla que en "student_view", pero con el nombre del proyecto modificado
delete_account/	delete_account	Página de confirmación de cuenta de usuario recién eliminada
accounts/login/	LoginView	Página para iniciar sesión, restablecer contraseña o Registrarse
accounts/logout/	LogoutView	Página de cierre de sesión del usuario
accounts/ password_change/	PasswordChange-View	Formulario para cambio de contraseña
accounts/ password_change/ done/	PasswordChange-DoneView	Página de confirmación de contraseña cambiada
accounts/ password_reset/	PasswordResetView	Formulario donde se introduce el correo de la cuenta para restablecer la contraseña
accounts/ password_reset/ done/	PasswordReset-DoneView	Página de confirmación de correo correcto para el envío de las instrucciones para restablecer contraseña
accounts/ reset/done/	PasswordReset-CompleteView	Página de confirmación de contraseña restablecida

Cuadro 4.1: Tabla explicativa de las URLs

4.3.2. Vista del análisis

La página que muestra el análisis de un proyecto consta de 4 secciones:

- Variabilidad (Uso de bloques) (figura 4.9): esta sección contiene un listado con los tipos de bloques que existen, con unos niveles asociados a cada uno. Se distinguen, como en la aplicación ScratchJr, seis tipos de bloques: Eventos (figura 4.4a), Movimiento (figura 4.4b), Apariencia (figura 4.4c), Sonido (figura 4.4d), Control (figura 4.4e) y Finalización (figura 4.4f). El nivel individual es la cantidad de bloques que se han empleado al menos una vez en todo el proyecto sobre los bloques totales que contiene cada tipo. Cabe destacar que Dr.ScratchJr toma como el mismo bloque aquellos que hacen la misma función, pero de manera opuesta, como se puede observar en las figuras 4.4b y 4.4c. Además, esta sección contiene el nivel total, que es la suma de todos los bloques que se han usado al menos una vez sobre la suma del total de bloques de cada tipo.

Figura 4.9: Sección Variabilidad.

Existe la opción de obtener más detalles pinchando sobre cualquier nivel obtenido de los

tipos de bloques, y aparecerá una vista que mostrará aquellos bloques que no han sido usados en el proyecto (figura 4.10).

Figura 4.10: Ejemplo detalles del apartado Movimiento de la sección Variabilidad.

- Malos hábitos (figura 4.11): aquí se muestra la cantidad de ciertos tipos de errores encontrados en el proyecto que el creador debe evitar:
  - Código muerto: cantidad de secuencias de bloques que no se van a ejecutar porque no comienzan por un bloque de Evento. También es considerado código muerto cuando una secuencia comienza por el bloque de Evento 'Comenzar con mensaje' de un color, pero no hay en el proyecto un bloque 'Enviar mensaje' del mismo color.
  - Código inacabado: cantidad de secuencias de bloques que no terminan con un bloque de 'Finalización'. Estas secuencias se van a ejecutar, pero es conveniente cerrar la secuencia por calidad del código y aprendizaje hacia buenas prácticas de programación.
  - Secuencias con bloques adyacentes: cantidad de secuencias que tienen bloques contiguos iguales. Es un mal hábito porque se debe usar un solo bloque e indicar el número de veces que se repite la acción del bloque.
  - Personajes con mismo nombre en una página: aparece, para cada página donde ocurra esto, el personaje que se repite y el número de ocasiones que lo hace. Aunque el personaje se visualice en ScratchJr, se debe nombrar correctamente al crearlo para que no haya varios con el mismo nombre en una única página.

# Malos hábitos	Cantidad
Código muerto	4
Código inacabado	3
Secuencias con bloques adyacentes	2
Personajes con mismo nombre en una página	{'Pos 1-page 1': {'Chica': '2 ocasiones'}}

Figura 4.11: Ejemplo de Malos hábitos.

Salvo en este último caso de malos hábitos (personajes con mismo nombre en una página), se encuentra la opción de pinchar en los resultados para obtener más detalles mostrados en forma de tabla con la siguiente estructura: página, personaje, nombre secuencia y secuencia (figura 4.12).

Figura 4.12: Ejemplo detalles de bloques adyacentes de la sección Malos hábitos.

- Creatividad (figura 4.13): muestra si el usuario ha generado nuevo contenido para hacer su proyecto en relación a:
  - Páginas editadas: cantidad de páginas que tienen su imagen de fondo editada respecto a las imágenes que vienen por defecto, o ha creado un fondo completamente nuevo en una página en blanco con el editor de pintura de ScratchJr.

- Personajes editados: número de personajes que han sido editados a través del editor de ScratchJr. Con este editor se puede cambiar el nombre del personaje, por lo que al cambiar el nombre también se considera un personaje editado, aunque su apariencia no haya cambiado.
- Sonidos creados: cantidad de sonidos que han sido grabados en el proyecto mediante el grabador de ScratchJr.

Figura 4.13: Ejemplo sección Creatividad.

Al pinchar sobre las cantidades de páginas o personajes editados, se puede ver la imagen correspondiente a cada página o personaje editado (figura 4.14). Además pinchando en la imagen se puede ver en tamaño completo en una pestaña nueva.

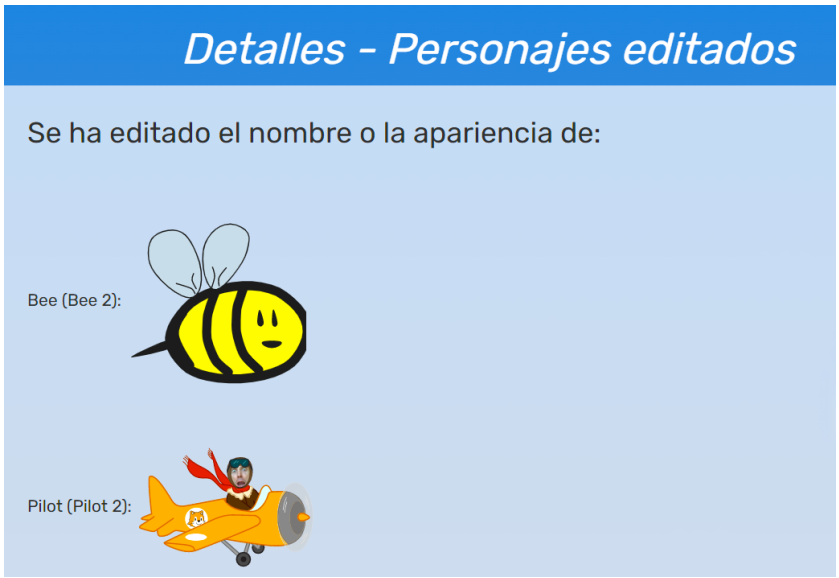


Figura 4.14: Ejemplo detalles de personajes editados de la sección Creatividad.

- Otros datos (figura 4.15): contiene un listado con datos que pueden llegar a ser de interés para analizar un proyecto:
  - Total de páginas: número de páginas que se han utilizado para todo el proyecto.
  - Total de personajes: número total de personajes empleados en todo el proyecto.
  - Total de textos: cantidad de textos creados en el proyecto.
  - Páginas con fondo sin editar: número de páginas que han sido utilizadas tal cual están disponibles en la aplicación ScratchJr por defecto.
  - Personajes sin editar: cantidad de personajes que han sido utilizados sin cambiar su nombre y sin editar su apariencia, respecto a los personajes por defecto que dispone ScratchJr.
  - Personajes en páginas: aparece, para cada página, los personajes utilizados en cada una de ellas.

Figura 4.15: Ejemplo sección Otros datos.

4.3.3. Vista de listados

Los tipos de listados que puede haber son:

- De un archivo zip: listado mencionado anteriormente (figura 4.7) de análisis de múltiples proyectos. Contiene un listado de los proyectos que están comprimidos en el zip, junto a los estudiantes asociados a dichos proyectos. Posibilidad de exportar los datos del análisis de todos los proyectos que contenga el zip.
- Estudiantes (figura 4.16): se accede principalmente desde el panel lateral. Muestra todos los estudiantes que tenga el usuario y el número de proyectos de cada estudiante. Se encuentra la posibilidad de editar el nombre y/o eliminar a cualquier estudiante junto con todos los proyectos con los que esté asociado. En este listado se pueden exportar los datos del análisis de todos los proyectos de todos los estudiantes que tenga el profesor de una sola vez.

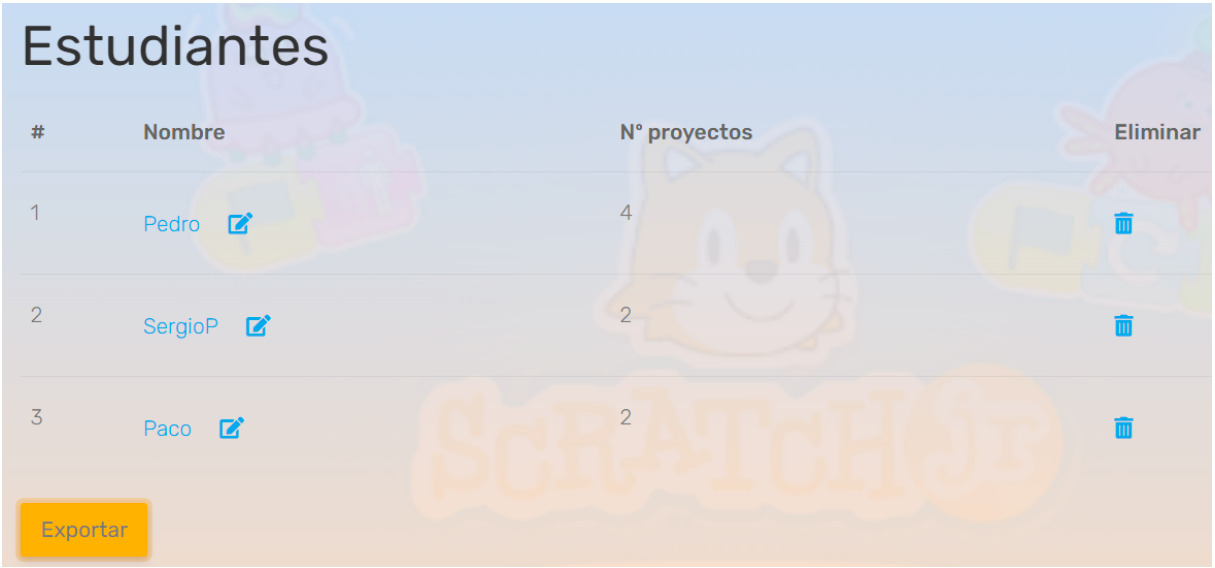


Figura 4.16: Ejemplo de un listado de los estudiantes de un usuario.

- Estudiante (figura 4.17): contiene un listado de los proyectos de un estudiante en particular. Además de la fecha, una nota del criterio de Variabilidad y el número de Malos hábitos del proyecto. También se puede editar el nombre de un proyecto y/o eliminar un proyecto en concreto. Posibilidad de exportar los datos del análisis de todos los proyectos del estudiante.

Figura 4.17: Ejemplo de un listado de los proyectos de un estudiante de un usuario.

Como ya se ha mencionado, al exportar se genera un zip que contiene tres ficheros CSV (badhabits-1, otherdata-1, variability-1) y una hoja de cálculo (data.xlsx) con los datos del análisis de tantos proyectos como tenga el listado desde donde se haya exportado.

Capítulo 5

Experimentos, validación y resultados

5.1. Ejemplo de un análisis

Para validar los distintos aspectos del análisis, se ha usado un proyecto de ejemplo llamado Test1.sjr cuyo contenido se presenta en la figura 5.1. Para ello, seguimos todos los aspectos que incluye un análisis de Dr.ScratchJr y comprobamos que concuerdan los resultados del análisis con los esperados.

Test1.sjr posee un total de dos páginas, la primera con una imagen de fondo del interior del océano y la segunda con una imagen de una playa. La primera página contiene los personajes “Buceador” y dos “Caballito de mar” y la segunda los personajes “Perro” y “Remero”, haciendo un total de 5 personajes en el proyecto. Hay un texto “Playa Blanca” en la página 2, siendo el único texto del proyecto. La primera página con una imagen de fondo del interior del océano no ha sido cambiada por el editor de ScratchJr y es de las que dispone el programa por defecto. Al igual ocurre con cuatro de los cinco personajes que tiene el proyecto, que son los que proporciona la aplicación y no han sido editados.

(a) Buceador

(b) Caballito 1

(c) Caballito 2

(d) Perro

(e) Remero

Figura 5.1: Proyecto de ejemplo Test1.sjr

En la figura 5.2 confirmamos estos datos con los mostrados en el análisis de Dr.ScratchJr.

 # Otros datos

Total de páginas	2
Total de personajes	5
Total de textos	1
Páginas con fondo sin editar	1
Personajes sin editar	4
Personajes en páginas	• *Pos 1-page 1: ['Buceador', 'Caballito de mar', 'Caballito de mar']* • *Pos 2-page 2: ['Perro', 'Remero']*

Figura 5.2: Otros datos del analisis de Test1.sjr.

Para validar el aspecto de Variabilidad hacemos un recuento de las ocasiones que aparece cada tipo de bloque en todo el proyecto (figura 5.3):

	BLOQUE	OCASIONES	TOTAL
BLOQUES DE EVENTOS	Al presionar bandera verde	5	1/5
	Comenzar al pulsar	0	
	Comenzar al tocar	0	
	Comenzar con mensaje	0	
	Enviar mensaje	0	
BLOQUES DE MOVIMIENTO	Mover a la derecha/izquierda	3	3/5
	Subir/bajar	5	
	Girar a la derecha/izquierda	0	
	Saltar	1	
	Ir al inicio	0	
BLOQUES DE APARIENCIA	Decir	2	2/4
	Crecer/disminuir	1	
	Restablecer tamaño	0	
	Ocultar/mostrar	0	
BLOQUES DE SONIDO	Pop	0	1/2
	Reproducir sonido grabado	1	
BLOQUES DE CONTROL	Esperar	1	2/4
	Parar	0	
	Fijar velocidad	0	
	Repetir	1	
BLOQUES DE FINALIZACIÓN	Finalizar	4	2/3
	Repetir indefinidamente	0	
	Ir a la página	1	

Figura 5.3: Recuento total de bloques de Test1.sjr.

Comparando los resultados con el análisis del aspecto de Variabilidad de Dr.ScratchJr que se muestra en la figura 5.4, confirmamos que coinciden.

Figura 5.4: Variabilidad del análisis de Test1.sjr.

Consideramos código muerto a secuencias de bloques que no se van a ejecutar en un proyecto porque no comienzan con un bloque de evento. En nuestro ejemplo Test1.sjr, hay una secuencia de bloques donde ocurre esto. En la figura 5.1c el personaje Caballito de mar posee una secuencia de bloques donde no se encuentra ningún bloque de evento, por lo tanto, no se va a realizar dicha secuencia y se considera que el proyecto contiene un *Código muerto*.

Por el contrario, una secuencia de bloques que no acaba con un bloque de finalización se considera *Código inacabado*. Es el caso que ocurre en una de las secuencias de la figura 5.1e.

En la figura 5.1b, aparecen dos bloques “subir” seguidos en una misma secuencia. Esto se considera un mal hábito de *Secuencias con bloques adyacentes* porque ScratchJr posee ciertos bloques que disponen de un atributo que representa el número de veces que quieres que se realicen dichos bloques de forma consecutiva sin la necesidad de estar repitiéndolos en el programa.

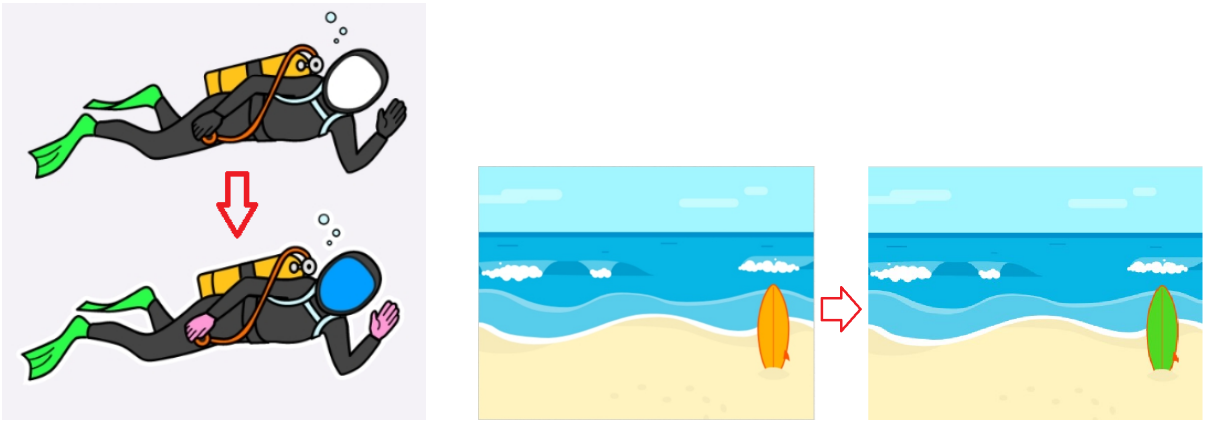
Otro mal hábito es el uso de *Personajes con mismo nombre en una página*. En la página 1 de Test1.sjr (figura 5.1) aparecen dos personajes con el nombre Caballito de mar, por lo que este nombre se encuentra en la página 1 en dos ocasiones.

Los resultados del análisis del aspecto de Malos hábitos coinciden con lo esperado (figura 5.5).

# Malos hábitos	Cantidad
Código muerto	1
Código inacabado	1
Secuencias con bloques adyacentes	1
Personajes con mismo nombre en una página	• *Pos 1-page 1: {'Caballito de mar': '2 ocasiones'}*

Figura 5.5: Malos hábitos del análisis de Test1.sjr.

De las dos páginas que tiene el proyecto Test1.sjr, la página 2 ha sido modificada con el editor de ScratchJr. Podemos ver en la figura 5.6b como la tabla de surf era de color naranja en el fondo de página que proporciona ScratchJr por defecto, y ahora es verde. De tal manera el personaje "Buceador" ha sido editado cambiando de color los guantes y la máscara (figura 5.6a), siendo éste el único personaje que ha sido modificado.



(a) Personaje editado

(b) Página editada

Figura 5.6: Personaje y página editada de Test1.sjr

En cuanto a sonidos creados, Test1.sjr contiene un bloque en la secuencia que tiene el personaje “Perro” de la página 2, con una grabación de audio emulando el sonido de la onomatopeya de ladridos.

Por último y para validar completamente el análisis de Dr.ScratchJr, los resultados del aspecto de Creatividad (figura 5.7) también concuerdan con lo esperado.

Figura 5.7: Creatividad del análisis de Test1.sjr.

5.2. Cargar varios proyectos

Ahora vamos a comprobar la funcionalidad de cargar varios proyectos simultáneamente en la aplicación. Los archivos de los proyectos se comprimen en un zip y éstos deben estar nombrados con la estructura “**‘Nombre del proyecto’**-**‘Nombre del estudiante’**.sjr” para poder almacenarlos y asociarlos a sus autores correctamente en la aplicación. Para ello usamos como ejemplo cuatro proyectos con los nombres modificados como se ha comentado anteriormente y almacenados en un zip llamado “Test_zip”. (figura 5.8).

Figura 5.8: Contenido del archivo Test_zip.zip.

Una vez cargado el zip y opcionalmente cambiado internamente en la aplicación el nombre de los proyectos o de los autores, aparece un listado como en la figura 5.9 y donde comprobamos que se han cargado correctamente los proyectos para la posterior visualización del análisis de cada uno de ellos o exportando todos los análisis en formato CSV.

Figura 5.9: Listado de proyectos de Test_zip en Dr.ScratchJr

5.3. Exportar datos en CSV

Si se cargan varios proyectos a la vez a través de un zip o si se ha iniciado sesión y están almacenados los proyectos, Dr.ScratchJr tiene la opción de exportar los análisis de los proyectos. En la figura 5.10, vemos como se exportan correctamente en tres ficheros CSV distintos los datos del análisis del proyecto del ejemplo anterior Test1.sjr. Aunque también se exportan estos mismos datos en un archivo Excel dividido en pestañas.

Nombre	Nombre del proyecto	Eventos	Movimiento	Apariencia	Control	Sonido	Finalización	Total
noelia	Test1.sjr	1/5	3/5	2/4	2/4	1/2	2/3	11/23

(a) Variabilidad en formato CSV

Nombre	Nombre del proyecto	Total de páginas	Páginas	Total de personajes
noelia	Test1.sjr	2	['Pos 1-page 1', 'Pos 2-page 2']	5

Personajes
['Buceador (Buceador 1)', 'Caballito de mar (Caballito de mar 1)', 'Caballito de mar (Caballito de mar 2)', 'Perro (Perro 1)', 'Remero (Remero 2)']

Total de textos	Textos en páginas	Total de páginas con fondo sin editar	Páginas con fondo sin editar
1	[['Pos 2-page 2', 'Text 1', ['str_txt', 'Playa Blanca '], ['fontsize', 36.0]]]	1	['Pos 1-page 1']

Total de personajes sin editar	Personajes sin editar
4	['Caballito de mar (Caballito de mar 1)', 'Caballito de mar (Caballito de mar 2)', 'Perro (Perro 1)', 'Remero (Remero 2)']

Personajes en páginas
{'Pos 1-page 1': ['Buceador', 'Caballito de mar', 'Caballito de mar'], 'Pos 2-page 2': ['Perro', 'Remero']}

(b) Otros datos en formato CSV

Nombre	Nombre del proyecto	Tipo de mal hábito	Existen malos hábitos	Página	Personaje
noelia	Test1.sjr	Código inacabado	SI	Pos 2-page 2	Remero (Remero 2)
noelia	Test1.sjr	Código muerto	SI	Pos 1-page 1	Caballito de mar (Caballito de mar 2)
noelia	Test1.sjr	Personajes con mismo nombre en una página	SI	----	----
noelia	Test1.sjr	Secuencias con bloques adyacentes	SI	Pos 1-page 1	Caballito de mar (Caballito de mar 1)

Nombre de la secuencia	Secuencia	Personajes con mismo nombre en una página
Sequence 2	[['onflag', 'null'], ['forward', 2], ['down', 3], ['forward', 1], ['down', 2]]	----
Sequence 1	[['repeat', 4, [['hop', 2, 657.682965648855, 61.423020515267055]]], ['endstack', 'null']]	----
----	----	{'Pos 1-page 1': {'Caballito de mar': '2 ocasiones'}}
Sequence 1	[['onflag', 'null'], ['up', 1], ['up', 1], ['down', 2], ['endstack', 'null']]	----

(c) Malos hábitos en formato CSV

Figura 5.10: Datos en CSV del análisis de Test1.sjr

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Los objetivos de este trabajo final de grado se han cumplido. Se ha creado una herramienta dónde se analizan proyectos de ScratchJr. Profesores, investigadores o padres pueden subir a la aplicación web uno o varios proyectos simultáneamente realizados por niños, ver el análisis de cada uno, almacenar y/o exportar los datos de los análisis.

6.2. Aplicación de lo aprendido

La mayoría de asignaturas que he cursado durante el Grado en Ingeniería en Sistemas Audiovisuales y Multimedia me han ayudado para la realización de este trabajo, puesto que me han ido formando en diversos aspectos, pero destacan:

- Expresión Oral y Escrita y Búsqueda de Información porque me ha ayudado a redactar, buscar información y desarrollar la memoria.

- Informática I, donde gracias al lenguaje de introducción a la programación "Picky" comencé a programar y aprender los fundamentos y las sintaxis básicas de la programación.

- Informática II me ayudó a conocer mejor la programación con el lenguaje Ada, a cometer menos errores y a enfrentarme a proyectos un poco más complejos.

- Protocolos para la Transmisión de Audio y Vídeo en Internet, pues gracias a esta asignatura comencé con el lenguaje Python que se usa en este proyecto y que, junto a Arquitectura de Internet y Sistemas Telemáticos para Medios Audiovisuales, aprendí los conceptos de arquitectura de red y protocolos, así como los métodos de interconexión de redes y encaminamiento, ayudándome a utilizar los fundamentos de la programación en internet.

- Construcción de Servicios y Aplicaciones Audiovisuales en Internet, ya que aprendí desarrollo web del lado del cliente y que me ha servido para usar HTML, CSS y JavaScript en el proyecto.

- Laboratorio de Tecnologías Audiovisuales en la Web es la asignatura que más me ha servido para poder realizar mi proyecto porque es donde aprendí más sobre las tecnologías web para la construcción de aplicaciones y servicios. Y donde adquirí los conocimientos para utilizar Django como tecnología web en el lado del servidor y los mecanismos de interacción entre cliente y servidor.

6.3. Lecciones aprendidas

Realizar este trabajo final de grado ha sido un gran reto y un verdadero quebradero de cabeza. Nunca me había enfrentado a un proyecto tan amplio y complejo partiendo de cero. La mayoría de prácticas del Grado han estado guiadas y esquematizadas, teniendo gran parte de la información necesaria disponible para realizarlas y obviamente han sido mucho más breves sin requerir de tanta elaboración.

He aprendido, aunque tarde, ha organizar mi propio tiempo y enfrentarme a los problemas y solucionarlos. Durante el grado, siempre había unas fechas para realizar los exámenes y entregar trabajos que me empujaban y obligaban a esforzarme, y como durante la realización de este proyecto no tenia ninguna presión del tiempo, en cuanto me surgía cualquier problema me estancaba, lo dejaba pasar y volver a retomar el proyecto se me tornaba más complicado. Pero al final conseguí compaginar el trabajo con el proyecto y aprovechar más el tiempo.

Gracias a este proyecto he aprendido a elaborar una aplicación web completa, a manejar el código, las funciones, la base de datos, los archivos... para que todo tenga una estructura

ordenada y enlazada. Por supuesto afianzando y ampliando todo lo estudiado en las asignaturas del Grado mencionadas en sección anterior 6.2.

También a la hora de realizar la memoria he aprendido a plasmar y redactar todo el trabajo realizado y era algo que no estaba habituado a ello.

6.4. Trabajos futuros

Puesto que es una aplicación web necesitará de contante mantenimiento y, siendo mi primera, de numerosas correcciones.

En cuanto a ampliar las funcionalidades de Dr.ScratchJr se podría:

- Mejorar el análisis de los proyectos añadiendo más criterios que den mayor información.
- Añadir otra función a la aplicación donde un usuario pueda ver estadísticas de todos los análisis que tenga almacenados y así tener mayor control e información.
- Crear otro apartado dónde haya enunciados de ejercicios específicos para hacer proyectos de ScratchJr con distintos niveles de dificultad y dónde Dr.ScratchJr evalúe con una nota si se ha realizado correctamente en exclusiva dicho ejercicio. Un ejemplo de ejercicio sería uno donde se tenga que hacer un proyecto contando una historia en ciertos lugares, que contenga ciertos personajes y que usen mínimo ciertos bloques.

Apéndice A

Manual de usuario

La aplicación web Dr.ScratchJr[1] se encuentra desplegada en el servicio de alojamiento web Pythonanywhere.

Pero además se puede instalar en local, donde mejoran considerablemente los tiempos de carga de los análisis. Para ello es necesario:

1. Descargar los archivos del repositorio de Github[2]

2. Tener instalado Python 3.9, Django 3.1.3, Django-Registration 3.1.1, Pandas 1.2.4 y Openpyxl 3.0.7.

3. Aunque no sea necesario para realizar el análisis de un proyecto, para configurar un servidor de correo propio, en el fichero settings.py situado en la carpeta descargada drscratchjr hay que cambiar los valores de las siguientes variables:

   - EMAIL_HOST: servidor SMTP que se utilizará para enviar correos electrónicos.
   - EMAIL_HOST_USER: nombre de usuario que se utilizará para el servidor SMTP.
   - EMAIL_HOST_PASSWORD: contraseña que se utilizará para el servidor SMTP.
   - EMAIL_PORT: puerto que se utilizará para el servidor SMTP.

4. Abrir el terminal en el directorio donde se encuentre el archivo manage.py y ejecutar los comandos:

[1] `http://spereztfg.pythonanywhere.com/`

[2] `https://github.com/sperezmaz/Dr.ScratchJr.git`

- `python manage.py makemigrations`
- `python manage.py migrate --run-syncdb`
- `python manage.py runserver`

5. Abrir en un navegador la siguiente URL:

`http://127.0.0.1:8000`

Ya sea en local o en Pythonanywhere, para analizar un proyecto ScratchJr sin almacenar su análisis en la aplicación web solo se debe encontrar en la página de inicio el formulario llamado "Analizar proyecto", darle al botón "Seleccionar archivo", buscar el proyecto deseado y pulsar sobre "Analizar". Por el contrario, si se quiere almacenar los resultados se debe crear una cuenta y posteriormente iniciar sesión. La única diferencia, en el formulario "Analizar proyecto" cuando se ha iniciado sesión, es que en el momento de subir el proyecto se necesita asociar dicho proyecto a un estudiante existente o crear uno nuevo escribiendo su nombre.

Si se desea analizar varios proyectos a la vez y exportar los resultados, en la página de inicio aparece otro formulario llamado "Analizar varios proyectos", en el que se debe subir un archivo .zip donde están comprimido todos los proyectos que se quieren analizar nombrados "NombreProyecto-NombreEstudiante.sjr". Al pulsar sobre analizar debe aparecer una página con otro formulario pre-rellenado por si se quiere modificar el nombre del proyecto o del estudiante. Cuando se pulsa en el botón finalizar se muestra una lista de los proyectos subidos en los que pinchando en el nombre del proyecto podemos ver su análisis. En esta lista existe la opción de descargar los resultados de los análisis dándole al botón exportar. Igualmente, para almacenar los análisis en la aplicación era necesario haber iniciado sesión previamente.

Para más detalles, dos vídeos explicativos del funcionamiento de la aplicación web Dr.ScratchJr:

- Para utilizar la aplicación sin registrarse.[3]
- Para utilizar la aplicación con las ventajas de tener un usuario.[4]

[3] `https://www.youtube.com/watch?v=wAnI3-UCVCs`
[4] `https://www.youtube.com/watch?v=ZXXOl7qqiMw`

Bibliografía

[1] Página Bootstrap.
https://getbootstrap.com/docs/.

[2] Página CSS.
https://www.w3schools.com/css/.

[3] Página Django.
https://docs.djangoproject.com/en/3.1/.

[4] Página Gettext.
https://docs.python.org/es/3/library/gettext.html.

[5] Página HTML.
https://www.w3schools.com/html/.

[6] Página Pandas.
https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html.

[7] Página Python.
https://www.python.org/doc/.

[8] Página ScratchJr.
https://www.scratchjr.org/.

[9] Página Visual Studio Code.
https://code.visualstudio.com/.

[10] J. Bennett. Django-registration.
https://django-registration.readthedocs.io/en/3.1/.

[11] J. D. Gauchat. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012.

[12] K. D. Leidl, M. U. Bers, and C. Mihm. Programming with scratchjr: a review of the first year of user analytics. In *Conference Proceedings of International Conference on Computational Thinking Education*, pages 116–121, 2017.

[13] J. Moreno-León, G. Robles, and M. Román-González. Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, (46):1–23, 2015.