GRADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA BIBLIOTECA DE ÍTEMS INTERACTIVOS PARA SIETTE

IMPLEMENTATION OF A LIBRARY OF INTERACTIVE ITEMS FOR SIETTE

Realizado por
Manuel Díaz Rodríguez

Tutorizado por
Ricardo Conejo Muñoz

Departamento
Lenguajes y ciencias de la computación

MÁLAGA, (Diciembre de 2020)

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA BIBLIOTECA DE ÍTEMS INTERACTIVOS PARA SIETTE

IMPLEMENTATION OF A LIBRARY OF INTERACTIVE ITEMS FOR SIETTE

Realizado por
Manuel Díaz Rodríguez

Tutorizado por
Ricardo Conejo Muñoz

Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, DICIEMBRE DE 2020

Fecha defensa:

Resumen

El sistema SIETTE es una plataforma que permite realizar tareas de evaluación de forma automática en entornos virtuales de aprendizaje. Su funcionamiento se organiza en torno al concepto de Tema, subtema y pregunta. Los tipos de preguntas que pueden realizarse en este sistema pueden ser desde una selección múltiple o una respuesta abierta a tipos más complejos que pueden ir añadiéndose al sistema con desarrollos adicionales de programación según diversas necesidades. A estos últimos se les denomina ítems externos.

En este trabajo se ha ampliado la biblioteca de ítems interactivos de la que dispone la plataforma SIETTE. Para ello se ha profundizado en el concepto de aprendizaje online y de usabilidad. Concretamente, se han implementado tres ítems. El primero, denominado Emparejamiento, busca trabajar y estudiar la capacidad del alumnado para relacionar conceptos de un mismo tema y ser capaces de ver patrones y diferencias en estos. El segundo ítem, denominado Etiquetado de Imagen, consiste en dar nombre a los elementos señalados en una determinada imagen. Este ítem busca trabajar y evaluar la capacidad del alumnado de reconocer los elementos de un determinado tema de manera visual. El tercer ítem, Selección de Área, comparte el mismo objetivo que Etiquetado de Imagen, pero con la diferencia de que el alumno recibe los nombres de los elementos y debe señalar dónde se encuentran en la imagen.

Estos ítems se han configurado para su integración en SIETTE y su uso en alumnos reales. Esto permitirá mejorar las opciones de los docentes a la hora de preparar cursos de evaluación automática para conocer el nivel de comprensión de un tema o para opciones de autoaprendizaje y refuerzo de la materia impartida.

Palabras Clave:

SIETTE, JavaScript, Sistemas de Evaluación Automática, Desarrollo gráfico, Interactividad, Usabilidad.

Abstract

The SIETTE system is a platform that allows the user to perform evaluation tasks automatically in virtual-learning environments. Its way of working is organized around the concept of Topic, subtopic and question. The types of questions that can be asked in this system can be from a multiple selection or an open answer to more complex types that can be added to the system with additional programming developments according to different needs. These latter ones are called external items.

On this project, the library of interactive items available on the SIETTE system has been expanded. To this end, the concept of online learning and usability has been carefully studied. Specifically, three items have been implemented. The first one, called Match Columns, seeks to work and study the ability of students to connect concepts of the same topic and be able to spot the patterns and differences in those presented. The second item, called Image Labelling, consists of giving names to the elements marked in a certain image. This item seeks to work and evaluate the ability of students to recognize visually the elements of a certain topic. The third item, Area Selection, shares the same objective as Image Labeling, but with the difference that the student receives the names of the elements and must indicate where they are in the image.

These items have been configured to be integrated into the SIETTE system and to be used in real students. This will allow improving the options of teachers when preparing automatic assessment courses to know the level of understanding of a topic or for self-learning options and reinforcement of the taught matter.

Key Words:

SIETTE, JavaScript, Automatic Evaluation Systems, Graphic development, Interactivity, Usability.

CONTENIDO

1 Introducción 1
1.1 Motivación 1
1.2 Objetivo 1
1.3 Organización de la Memoria 2

2 Contextualización 3
2.1 E-learning 3
2.1.1 Evaluación Automática 3
2.2 SIETTE 4
2.2.1 Base de Conocimientos: Tema, Subtema y Pregunta 6
2.2.2 El concepto de ítem 7
2.3 La usabilidad en los sistemas informáticos 9
2.3.1 Evaluación de la Usabilidad 9
2.3.2 Desarrollar aplicaciones usables. Ingeniería de Usabilidad 10
2.4 Tecnologías 11
2.4.1 Aplicación Web 11
2.4.2 HTML 12
2.4.3 CSS 13
2.4.4 JavaScript 14

3 Biblioteca de ítems 17
3.1 Metodología para le creación de nuevos ítems. 17
3.2 Ítem de Emparejamiento 18
3.2.1 Solución Vista Profesor 18
3.2.2 Solución Vista Alumno 20
3.2.3 Solución Vista Resolver y Solución 22
3.2.4 Solución Técnica: Vista Alumno y Vista Profesor 24
3.2.5 Solución Técnica: Vista Resolver y Vista Solución 29
3.2.6 Aspectos de Usabilidad del ítem 31
3.3 Ítem de Etiquetado de Imagen 32
3.3.1 Solución Vista Profesor 32
3.3.2 Solución Vista Alumno 36
3.3.3 Solución Vista Resolver y Solución 37
3.3.4 Solución Técnica: Vista Profesor y Vista Alumno 39
3.3.5 Solución Técnica: Vista Resolver y Vista Solución 43
3.3.6 Aspectos de usabilidad del ítem 43
3.4 Ítem de Selección de Área 44
3.4.1 Solución Vista Profesor 44
3.4.2 Solución Vista Alumno 47
3.4.3 Solución Vista Resolver y Solución 48
3.4.4 Solución Técnica: Vista Profesor y Vista Alumno 50
3.4.5 Solución Técnica: Vista Resolver y Vista Solución 55
3.4.6 Aspectos de usabilidad del ítem 55

4 Metodología de Trabajo 59
4.1 Metodología Scrum 59
4.2 Ciclo 1. Primera Versión de Emparejamiento Alumno 59

4.3 Ciclo 2. Segunda Versión de Emparejamiento Alumno 60
4.4 Ciclo 3. Implementación de Emparejamiento Profesor 60
4.5 Ciclo 4. Adaptación del Código para que haya Múltiples Preguntas 61
4.6 Ciclo 5. Primera Versión de Etiquetas 62
4.7 Ciclo 6. Segunda Versión de Etiquetado 63
4.8 Ciclo 7. Vista Resolver y Vista Solución de Emparejamiento 64
4.9 Ciclo 8. Vista Resolver y Vista Solución de Etiquetado 64
4.10 Ciclo 9. Primera versión de Selección de Área 64
4.11 Ciclo 10. Segunda versión de Selección de Área e inclusión de ayudas 65
4.12 Ciclo 11. Pruebas de los ítems y creación de contenido 66

5 Conclusiones y Líneas Futuras 67
5.1 Conclusiones 67
5.2 Líneas Futuras 69

Bibliografía 71

1 INTRODUCCIÓN

1.1 Motivación

El presente trabajo se enmarca en el campo del E-learning o aprendizaje electrónico. Más concretamente nos centramos en la labor de SIETTE como sistema de tutorización inteligente y su aplicación de la Teoría de Test Adaptativos, concepto que se amplía en el punto *2.1.2 SIETTE* del presente documento.

El sistema SIETTE se encarga principalmente de la evaluación mediante test del aprendizaje adquirido de un alumno. Para ello, hace uso de los denominados ítems. Debido a la diversidad de conocimiento que se puede enseñar, reforzar o evaluar mediante las nuevas tecnologías se quiere ampliar la gama de ítems de los que dispone SIETTE, y esa es la tarea que aquí nos atañe.

Una mayor variedad de ítems o tipos de preguntas puede ayudar al sistema a tener una mejor información del grado de conocimiento que posee cada alumno sobre cada concepto a aprender. Del mismo modo, permite a cada alumno ver los mismos conceptos desde distintos puntos de vistas ayudando así a un mejor aprendizaje y compresión del tema.

1.2 Objetivo

Este proyecto consiste en la definición e implementación de diversos tipos de ítems interactivos de propósito general y su integración en el sistema SIETTE. En concreto se plantea la realización de los siguientes ítems:

- Preguntas de emparejamiento, en donde aparecen dos columnas y el alumno debe realizar la correspondencia entre elementos de una y otra columna. Por ejemplo, enlazar una lista de capitales con sus respectivos países.
- Preguntas de selección de áreas en imágenes, en donde se plantean al alumno la identificación de una o varias regiones sobre una imagen predefinida. Por ejemplo, localizar el pistilo en una imagen de una flor.
- Etiquetado de imágenes. En donde aparece una imagen con distintos elementos señalados y el alumno debe indicar el nombre de cada uno de estos elementos.

Para la implementación de esta biblioteca de ítems se realizarán cuatro interfaces, una para que el alumno pueda proporcionar la respuesta; otra para el

profesor, de manera que se facilite la creación de estas preguntas; y otras dos para que el alumno pueda comprobar cuál era la respuesta correcta y dónde se ha equivocado.

Para cumplir estos objetivos se ha seguido una metodología ágil en la que, por ciclos de tres a cuatro semanas, se iban entregando prototipos de los distintos ejercicios y se iban modificando, ampliando o eliminando sus especificaciones a petición del tutor del TFG, que era quien se encargaba de probarlos y evaluarlos, hasta que se considerara que cumplieran los objetivos marcados.

1.3 Organización de la Memoria

La presente memoria se estructura en torno a cinco capítulos en los que se describen las cuestiones que motivan este trabajo, cómo se ha desarrollado y con qué tecnologías, así como unas anotaciones conclusivas finales y una visión de cara al futuro.

En el capítulo segundo, llamado *Contextualización*, se habla de la educación por medio de ordenador y del sistema SIETTE, que es donde se introducirán los ítems aquí desarrollados. También se introduce el tema de la usabilidad en los sistemas informáticos y se presentan las Tecnologías usadas. Se hablará de los tres lenguajes de programación que se han utilizado para el desarrollo de los ítems, así como del concepto de Aplicación Web, para un mejor entendimiento de qué es lo que se está trabajando aquí.

En el capítulo tercero, *Biblioteca de Ítems*, se explica todo el proceso de desarrollo que ha tenido cada ítem, más concretamente, se analiza en profundidad el problema y la solución tanto para el diseño de ejercicios como para la resolución de estos mismos.

El capítulo cuarto, *Metodología*, explora la metodología que se ha usado para alcanzar las metas propuestas. En concreto, se habla de la metodología ágil Scrum y de qué se ha ido haciendo en cada ciclo del proceso.

Por último, en el capítulo quinto, *Conclusiones y Líneas Futuras*, se realizan los comentarios finales y valoraciones personales sobre el proyecto presentado, así como sobre las posibilidades futuras del tema aquí tratado.

2 CONTEXTUALIZACIÓN

2.1 E-learning

Ruth Colvin Clark y Richard E. Mayer (2011) definen el e-learning (aprendizaje electrónico o electronic learning) como un conjunto de instrucciones en un medio digital que son usadas para reforzar un aprendizaje. Además de esta definición, estos autores también añaden una serie de condiciones que debe cumplir todo proceso de e-learning para ser denominado como tal:

- El contenido es relevante para la meta a alcanzar.
- Se utilizan ejercicios y ejemplos que refuerzan el aprendizaje.
- Utiliza recursos, como imágenes o videos, que ayudan a transmitir los contenidos.
- Ayuda a adquirir nuevo conocimiento o habilidades relacionadas con el tema a aprender por cuenta propia.

Si bien existen cursos exclusivamente basados en el e-learning, el propósito de este trabajo es elaborar herramientas para la evaluación y el refuerzo del aprendizaje llevado a cabo por cada alumno.

2.1.1 Evaluación Automática

Se define la evaluación como *"una actividad o proceso sistemático de identificación, recogida o tratamiento de datos sobre elementos o hechos educativos, con el objetivo de valorarlos primero y, sobre dicha valoración, tomar decisiones."* (García Ramos, J. M., 1989).

En un sistema de e-learning implantar una evaluación automatizada presenta grandes ventajas desde el punto de vista del docente, pues permite ahorrar mucho tiempo en el proceso educativo. Es importante no confundir, en cualquier caso, el término evaluación con el término calificación, pues la calificación consiste en dar una valoración cuantitativa o cualitativa a la evaluación realizada.

Según el tipo de ítem podemos distinguir dos tipos de evaluación automática (Soler, J., 2010):

- **Evaluación automática a partir de respuestas fijas.** Es la evaluación que se realiza cuando existen un número fijo de respuestas definidas previamente y sobre las que el alumno puede elegir una o más. En esta

evaluación el sistema sólo debe comprobar si la respuesta recibida es o no correcta. Algunos de los ejercicios que se engloban en esta categoría serían las preguntas de elección múltiple, de respuesta corta o con gráficos. Por tanto, la evaluación los tres ítems que se van a desarrollar en el presente trabajo serán de este tipo, siendo que el ítem etiquetado de imagen correspondería a un ejercicio de respuestas cortas; y los ítems de Selección de Área y Emparejamiento a ejercicios con gráficos.

- **Evaluación automática a partir de respuestas libres.** En este tipo de ejercicios, la respuesta dada por el alumno se realiza sin ningún tipo de restricción. Por tanto, para la evaluación y corrección de este caso hace falta la existencia tanto de sistemas específicos como de interfaces claras que permitan la libertad suficiente al alumno como para dar su solución. Además, en los casos de ejercicios que requieran unas respuestas lo suficientemente largas, la evaluación debe tener cuidado de no dar demasiada importancia a los pequeños errores, de modo que en la calificación no salga como incorrecto un ejercicio largo con unos pocos errores pequeños. Toda esta complejidad hace que este tipo de evaluación no esté muy extendida. Algunos de los ejercicios para los que esta categoría es la adecuada son los ejercicios de programación, de diagramas o de texto libre.

2.2 SIETTE

SIETTE (Conejo et al., 2004, 2016) es un sistema web para evaluación. Para ello, ofrece herramientas online para la edición de ejercicios, de modo que los docentes puedan diseñar los suyos propios y ajustarlos a la materia concreta que se desea enseñar; y, así mismo, herramientas para la resolución de dichos ejercicios por parte del alumnado.

Figura 1. Ventana de inicio de sesión en el sistema SIETTE, 2020.

La arquitectura de SIETTE se divide así en cinco módulos:

- **Base de conocimientos.** Está compuesta por la estructura del temario, las especificaciones de test y el banco de ítems, siendo este último en el que centraremos el presente trabajo.
- **Generador de Test.** Selecciona las preguntas que se le presentan al alumno.
- **Módulo de edición.** Desde este, el docente podrá aumentar la base de conocimientos de una asignatura determinada, modificando la estructura del temario, los ítems y las especificaciones de los test.
- **Módulo de comprobación y activación.** Este módulo comprueba que toda la edición realizada por el docente es correcta y, por tanto, las nuevas preguntas son válidas para ser incluidas en test.
- **Módulo de aprendizaje.** Es el módulo sobre el que trabajan los estudiantes y en el que se le presentan las preguntas.

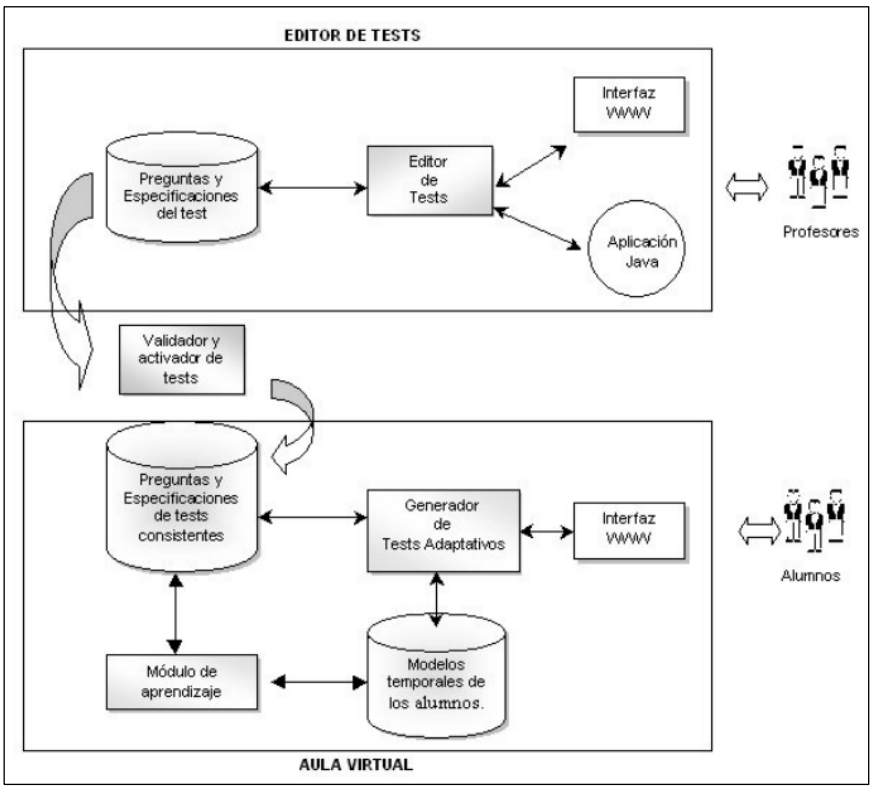

Figura 2. Arquitectura del sistema SIETTE. Ilustración de Conejo, R. y Guzmán, E. Recuperado de SIETTE: Sistema Inteligente de Evaluación mediante Test para TeleEducación.

El trabajo aquí presentado se basa en la elaboración de nuevos ítems, de modo que se amplíe la biblioteca de la que dispone SIETTE y se aumente la versatilidad de los

test al contar con una gama más amplia de opciones en las que evaluar o presentar los contenidos de una determinada materia.

2.2.1 Base de Conocimientos: Tema, Subtema y Pregunta

La base de conocimientos de SIETTE está compuesta a su vez por bases de conocimiento dedicadas a cada asignatura. Estas bases están compuestas por tres tipos de objetos:

- **Temas o conceptos**. Son los elementos en los que se divide la materia a trabajar. Los temas se organizan de forma jerárquica existiendo así la posibilidad de crear subtemas que dependen de un tema que engloba más contenido. Los nodos finales del árbol que conforman los temas de una determinada asignatura representan un concepto único o un conjunto de conceptos a evaluar como uno solo.
- **Ítems.** Este punto se extenderá en el apartado *2.2.1 El concepto de ítem.* De manera simplista, los ítems son los ejercicios que conforman los test que evalúan a los alumnos. Así pues, cada concepto tendrá ligado uno o más ítems y, del mismo modo, un ítem puede estar ligado a uno o más temas.
- **Test.** El test se compone de una serie de ítems y busca evaluar el grado de conocimiento del alumno sobre uno o más temas. En SIETTE, los temas que pueden ser evaluados por un test están restringidos a que sean hermanos.

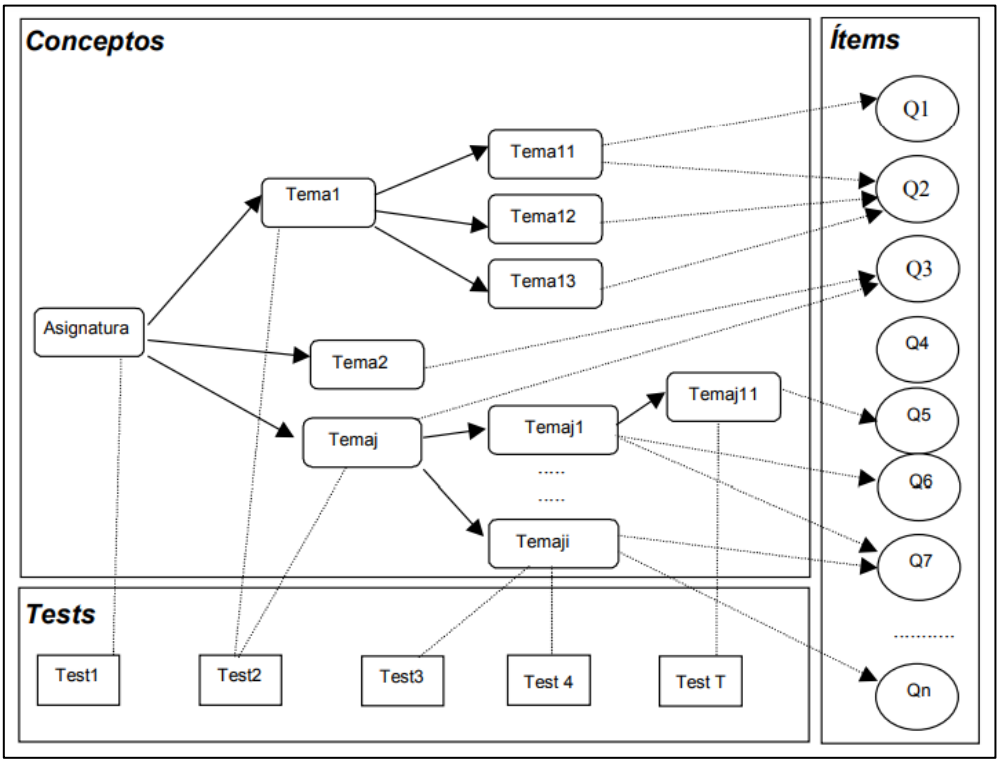

Figura 3. Estructura de la base de conocimientos de SIETTE. Ilustración de Conejo, R. y Guzmán, E. Recuperado de SIETTE: Sistema Inteligente de Evaluación mediante Test para TeleEducación.

2.2.2 El concepto de ítem

Los ítems son el elemento medidor del grado de conocimiento que tiene un alumno sobre uno o más conceptos. Así mismo, son las partes en las que se divide un test. Al inicio de la elaboración de este trabajo. Los ítems de SIETTE se pueden clasificar en distintas categorías:

Ítems básicos que corresponden a las opciones elementales de evaluación. Cualquier otro tipo de ítem se transforma en uno de estos ítem básicos:

- **Opción múltiple, respuesta única.** Son las clásicas preguntas en las que el alumno debe seleccionar una única respuesta como correcta.
- **Opción múltiple, respuesta múltiple (independiente).** En esta clásica pregunta, al alumno se le presentan varias respuestas a un determinado enunciado y debe marcar como correctas una o más de una.
- **Respuesta corta.** Al alumno se le presenta un enunciado y debe rellenar un campo de texto para responder.

Ítems Compuestos. Agrupan bajo un mismo enunciado a varios ítems. Los ítems compuestos se presentan siempre juntos y se evalúan juntos, aunque la evaluación de cada uno de los componentes se hace de forma independiente.

Ítems externos. Son los que se ejecutan fuera de SIETTE, por ejemplo llamando a un programa externo del sistema operativo, o planteando la pregunta mediante una pagina web externa. El programa externo debe devolver a SIETTE una cadena de caracteres a partir de la cual SIETTE realizará la evaluación mediante el ítem básico asociado.

Items QTI. Son un tipo especial de ítems externos. El ítem contiene una especificación en XML según el formato QTI, que se ejecuta mediante un reproductor externo.

Ítems interactivos. Son ítems que requieren la realización de ciertas acciones por parte del usuario más allá de la simple elección de una opción o la escritura de un breve texto. Estas extensiones que recogen la interacción con el usuario y la transforman en un ítem básico para su evaluación

- **Ordenar textos.** El enunciado establece un criterio por el que el alumno debe ordenar los distintos textos que se le presentan. Por ejemplo:

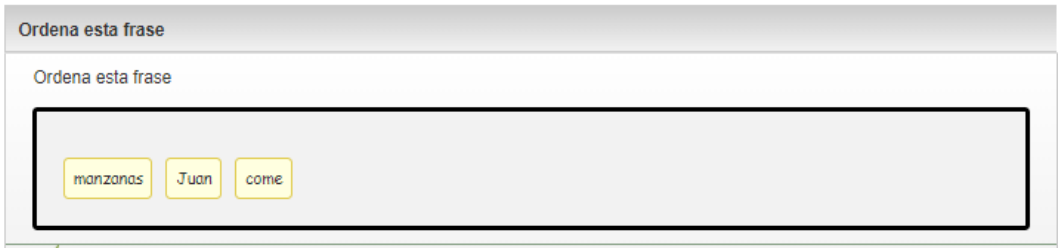

Figura 4. Ejemplo de ítem Ordenar textos

- **Ordenar imágenes.** Este ítem es similar al anterior, pero en vez de ordenar textos se ordenan imágenes.

- **Música.** Este tipo de ítem se caracteriza por que se presenta al alumno una partitura musical. En este mismo ítem se permite al docente elegir que el alumno tenga que responder mediante un teclado de piano que se muestra al alumno o un campo de texto, entre otras posibilidades.

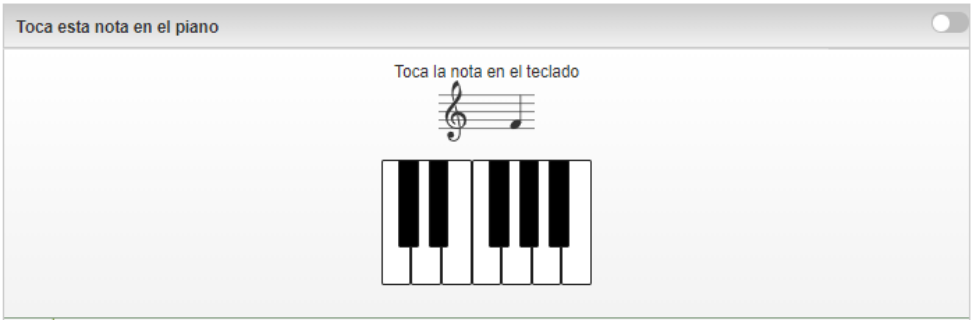

Figura 5. Ejemplo de ítem Música

- **Buscar diferencias.** El alumno debe encontrar las diferencias entre dos imágenes que se le presentan.
- **Trazar figuras.** El alumno debe trazar una figura siguiendo una línea de puntos que la delimita.

Los ítems interactivos, originalmente se desarrollaron mediante Applets de Java. Actualmente son programas escritos en JavaScript. Esta precisamente esta categoría de ítems la que vamos a ampliar en este trabajo desarrollando tres nuevos tipos de ejercicios. Dicho desarrollo, de manera general, se caracteriza por seguir los siguientes pasos:

1. Estudiar para cada tipo de ejercicio: cuál sería su correcto funcionamiento, cómo hacerlo lo más moldeable posible para adaptarlo a cualquier pregunta que se le pueda ocurrir al docente, cuál es la mejor manera de mostrar el ejercicio, qué tipos de acciones puede realizar el usuario y cómo representar la respuesta dada por el alumno al ejercicio.
2. Implementar con JavaScript cada uno de los tipos de ítems. Para ello se definen cuatro vistas: la vista del profesor, en la que se define cómo será el ejercicio y cuál la respuesta correcta; la vista del alumno, en la que el alumno se enfrenta al ítem y debe resolverlo; y las vistas resolver y solución en la que se muestra al alumno dónde se ha equivocado y cuál era la respuesta correcta respectivamente.
3. Integrar en SIETTE los ejercicios implementados para su futuro acceso a docentes y alumnos.
4. Probar lo implementado mediante la creación de gran cantidad de ítems y corregir los errores que se vayan encontrando.
5. Los pasos 2, 3 y 4 se repiten cíclicamente hasta que se llega a la conclusión de que el nuevo ítem cumple todas las especificaciones definidas en el paso 1 y las que vayan surgiendo durante el desarrollo.

2.3 La usabilidad en los sistemas informáticos

Nielsen, J. (2012) define la usabilidad como un atributo de calidad que evalúa la facilidad de uso de las interfaces de usuario. Del mismo modo, la ISO/IEC 9126 define la usabilidad como un conjunto de atributos relacionados con el esfuerzo necesitado para el uso de la aplicación y, en la valoración individual de tal uso, por un conjunto de usuarios establecido o implicado.

De la mano de estas definiciones, la Oficina de Calidad de la Junta de Andalucía (Junta de Andalucía, 2020) examina los siguientes atributos de usabilidad:

- **Eficacia:** Mide la exactitud con la que se alcanzan los objetivos especificados. Este atributo engloba la facilidad de aprendizaje, la retención en el tiempo y las tasas de error.
  - *Facilidad de aprendizaje.* ¿Cuánta tardaría un usuario medio en aprender cómo funciona, sin profundizar, la aplicación?
  - *Retención en el tiempo.* Una vez un usuario medio ha aprendido a usar la aplicación, ¿cuánto de bien recuerda su funcionamiento tras un periodo (una hora, un día, una semana, ...) sin usarla?
  - *Tasas de error.* ¿Cuántos errores comete la aplicación y cuáles son? Una aplicación que presenta errores contantemente es una aplicación poco usable.
- **Eficiencia en el uso:** Este atributo mide la velocidad que tendrían los usuarios, una vez han aprendido el diseño de la aplicación, en realizar las tareas objetivo de la aplicación. En otras palabras, mide la productividad del usuario medio.
- **Satisfacción:** mide el atractivo de la aplicación de cara a los usuarios en lo relativo a la manera de trabajar con la aplicación. Se trata de un atributo completamente subjetivo que cuanto más elevado sea más atraerá a los usuarios a que la usen.

2.3.1 Evaluación de la Usabilidad

Para comprobar si un sistema es o no usable se realizan las llamadas evaluaciones de usabilidad. Según menciona Ferré, X. (2000), estas evaluaciones se dividen normalmente en dos pruebas: test de usabilidad y evaluaciones heurísticas.

El test de usabilidad es la prueba más común y consiste en evaluar la usabilidad de la aplicación presentándole un prototipo a un usuario real. La idea detrás de esto es que es imposible medir la usabilidad de un sistema si los únicos que lo prueban son gente que ya se encuentra dentro del proyecto. Las características de la realización de este test varían mucho dependiendo de cómo sea el sistema. En general, se seleccionarán a un grupo de personas que actuarán como usuarios y probarán el sistema de modo que tengan que realizar una serie de tareas planteadas por los evaluadores. Se deberá evaluar a la hora de realizar el test si se permitirá o no hacer

preguntas al evaluador, cuánta información se les dará a los usuarios o si se podrán comunicar entre ellos. Esto último es muy común, puesto que permite ver cómo piensan los usuarios a la hora de enfrentarse al sistema.

El siguiente paso sería la realización de una evaluación heurística. Es parecida a un test de usabilidad, con la diferencia de que, en vez de utilizar posibles usuarios finales, se le presenta el sistema a un experto en usabilidad. Esto se hace así puesto que un experto será capaz de dar unas valoraciones más concisas y útiles de las que pueda dar cualquier usuario medio. Para realizar las valoraciones, el experto en usabilidad podrá basarse o bien en su experiencia personal o en alguna guía de diseño de usabilidad. La Oficina de Calidad tiene su propia guía de diseño que divide las valoraciones del experto en varias categorías: identidad e información; acceso; lenguaje y redacción; rotulado; navegación; diseño y contenidos; recursos multimedia; búsquedas y listados; formularios; ayuda; gestión de errores, control y retroalimentación; y flexibilidad y eficiencia en el uso. Cada categoría cuenta con una serie de preguntas genéricas que el experto debe responder.

2.3.2 Desarrollar aplicaciones usables. Ingeniería de Usabilidad

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. y Carey, T. (1994) define la ingeniería de usabilidad como una aproximación al desarrollo de sistemas en la que se especifican niveles cuantitativos de usabilidad a priori, y el sistema se construye para alcanzar dichos niveles, que se conocen como métricas. En otras palabras, la ingeniería de usabilidad da las herramientas necesarias para que la usabilidad sea una especificación más en el desarrollo de un sistema.

El apartado anterior habla sobre cómo la evaluación nos permite comprobar si nuestro sistema es realmente usable. Tanto el test de usabilidad como la evaluación heurística son dos de las técnicas de la ingeniería de la usabilidad. En concreto, Ferré, X. nos plantea un total de tres tipos de técnicas: especificaciones, diseño y evaluación.

Respecto de las especificaciones se definen tres técnicas distintas. La primera es *análisis de usuarios*. Para realizar cualquier sistema, es necesario conocer a quién va destinado. Saber cómo piensa el usuario permitirá elegir el modo en que el sistema debe funcionar. Este análisis puede ser de gran utilidad, además, para seleccionar a los usuarios participantes en los test de usabilidad.

La segunda técnica de especificaciones es el *análisis de tareas*. Consiste en describir los procedimientos que hace la gente para lograr realizar una determinada tarea. Generalmente esta técnica se realiza después del análisis de usuarios, de modo que el equipo de desarrollo sea capaz de entender cómo realizan las tareas a implementar los usuarios finales.

La tercera técnica de especificaciones son las *especificaciones de usabilidad*. Consiste en especificar los objetivos de usabilidad atendiendo a los atributos mencionados al principio del apartado *2.3 La usabilidad de los sistemas informáticos*: eficiencia, eficacia y satisfacción.

Respecto del diseño se definen tres técnicas distintas. La primera es el *diseño de la interacción*. El objetivo es diseñar tanto el concepto del sistema como la parte visual de la interacción. El diseño acabará convirtiéndose en un prototipo que los usuarios valorarán.

La segunda técnica de diseño es el *prototipado.* Para poder pedir a los usuarios unas valoraciones hay que entregarles un prototipo del sistema. En un primer momento, estos prototipos pueden ser de papel, un experto simulando las respuestas del sistema o incluso unas viñetas, aunque lo ideal es presentar un prototipo que se acerque lo más posible al sistema final.

La tercera técnica de diseño es *la participación de usuarios*. Es muy común que en la etapa de diseño los propios usuarios participen o bien mediante la comunicación activa con ellos o bien haciendo íntegramente responsables de las decisiones de diseño a unos representantes de los usuarios.

La clave detrás de estas técnicas es volverlas a realizar entre cada iteración o ciclo del desarrollo. De modo que tras cada nuevo prototipo se pase por todos los puntos, ya sea para evaluar las especificaciones que ya cumple o para tratar la usabilidad de los nuevos elementos que se incluirán en la siguiente iteración.

2.4 Tecnologías

2.4.1 Aplicación Web

Una aplicación web es una aplicación que sigue la arquitectura cliente/servidor, esto es, en la que en la comunicación habrá dos tipos de computadores o procesos: clientes o servidores; siendo que los clientes solicitan servicios y los servidores los proporcionan. Así, la aplicación web es un caso concreto de la arquitectura cliente/servidor en la que:

- **La comunicación** se realiza mediante un protocolo estandarizado llamado HTTP (HyperText Transfer Protocol).
- **El cliente web** es un programa (típicamente un navegador web) con el que el usuario se puede comunicar con el servidor y puede visualizar e interactuar con la información que este le proporciona.
- **El servidor web** es un programa que espera activamente la llegada de peticiones HTTP por parte de clientes web (Luján, 2001).

Los ítems elaborados en este proyecto son aplicaciones web cuyo servidor es SIETTE y el cliente son los navegadores web utilizados en los computadores de alumnos y docentes. En concreto, SIETTE se encarga de enviar los ejercicios a alumnos y docentes, así como de proporcionar un método de corrección automatizado de los mismos. El cliente se encargará de ejecutar los métodos para que el usuario pueda

resolver —en el caso de los alumnos— o diseñar —en el caso de los docentes— uno o varios ítems.

2.4.2 HTML

HTML (HyperText Markup Language) es un lenguaje de marcado que sirve para dar formato a las páginas web. Fue desarrollado por WHATWG (Web Hypertext Application Technology Working Group) y lanzado en 1993. Este lenguaje lo interpretan los navegadores web para mostrarnos las páginas con el formato deseado.

Un lenguaje de marcado es aquel que hace uso de *etiquetas* para dar información acerca del formato del texto contenido entre las etiquetas [Luján, S.]. Así, el lenguaje HTML está compuesto por una serie de etiquetas. Estas, mayoritariamente, aparecen por parejas, de modo que una indica el comienzo y otra el fin. Otras etiquetas, sin embargo, van solas, sin necesidad de que haya ningún cierre.

La sintaxis para escribir las etiquetas es el símbolo "menor que" (<), el nombre de la etiqueta, y el símbolo "mayor que" (>). De modo que, un ejemplo de etiqueta válida que se presenta sola es <img>. Esta etiqueta se utiliza para cuando introducir una imagen en el documento.

La sintaxis aquí descrita es la genérica para una etiqueta que indica comienzo, si quisiéramos escribir la etiqueta de fin, pondríamos antes del nombre de la etiqueta una barra (/). Por ejemplo <body></body>.

Además de esto, una etiqueta que no sea de cierre puede poseer uno o más atributos que se escribirán a la derecha del nombre y antes del "mayor que". La sintaxis de esto sería: *<nombre_etiqueta atributo1=valor1 atributo2=valor2 ... atributoN = valorN>*. Un ejemplo muy típico es el de la etiqueta imagen, que generalmente va a acompañada del atributo "src" con el que se indica la dirección en la que se encuentra la imagen a mostrar y del atributo "alt", que almacena un texto alternativo a mostrar en caso de que la imagen no pueda ser presentada, por ejemplo:
<img
src="https://informatica.cv.uma.es/theme/image.php/uma2015clean/theme/1605179543/evlt/logouma18" alt="Universidad de Málaga">

Los ficheros HTML están escritos en texto plano, lo que permite que sin necesidad de un programa específico, cualquiera pueda escribir uno. Así mismo, estos ficheros siguen la siguiente estructura:

```
1 <!DOCTYPE HTML PUBLIC ".//W3C//DTD HTML 4.0//EN">
2 <HTML>
3 <HEAD>
4 Cabecera de la página
5 </HEAD>
6 <BODY>
7 Cuerpo de la página
8 </BODY>
9 </HTML>
```

Figura 6. Estructura de un fichero HTML. Ilustración de Luján, S. Recuperado de Programación en Internet: Clientes WEB.

La primera línea de la figura 6 se utiliza para indicar qué versión de HTML se está usando, aunque no es obligatorio su uso y no suele incluirse en los ficheros. La segunda y novena línea indican el inicio y el final de la página, siendo que la existencia de la etiqueta <html> es obligatoria.

Dentro de la etiqueta html se engloban dos grandes bloques. El primero es la cabecera, que se indica con la etiqueta <head> y está entre las líneas tres y cinco. Dentro de la cabecera aparece la metainformación de la página, como su título, autor, etc.

Por último, entre las etiquetas <body> y </body> en las líneas de la seis a la ocho, tenemos el cuerpo de la página. Aquí será donde se pondrán todo el contenido de la página web.

2.4.3 CSS

CSS (Cascade Style Sheets) es un lenguaje de hoja de estilo desarrollado por Håkon Wium Lie, Bert Bos y el Consorcio WWW (World Wide Web) y que fue lanzado en diciembre de 1996. Se utiliza junto con HTML para poder definir mejor la presentación de cada elemento.

La separación entre CSS y HTML surge de la necesidad de separar estructura interna (que se trata en el fichero HTML) y estructura visual (de la que se encarga CSS) (Meyer, 2006). HTML permite escribir directamente código CSS mediante la etiqueta <style>, aunque lo más habitual es tener ambos códigos en documentos separados, siendo que el fichero CSS se pasa al HTML por medio de una etiqueta <link>.

CSS se encarga así de establecer un valor a una determinada propiedad, de modo que, a cada par propiedad-valor se le denomina *declaración*. Las declaraciones siguen la sintaxis: *propiedad: valor*.

Se pueden agrupar un conjunto de declaraciones colocándolas a todas entre llaves ({}) y separándolas por puntos y comas (;). La sintaxis pues de un bloque de declaraciones sería: *{propiedad1: valor1; propiedad2: valor2; ... propiedadN: valorN;}*. Lo normal es poner una línea por cada declaración para que, visualmente, sea más claro qué es lo que se está modificando.

Por último, queda definir los conjuntos de reglas CSS. Estas están compuestas por un grupo de selectores (de qué elementos se va a modificar su representación) y por un bloque de declaraciones (cómo se van a mostrar esos elementos). Los selectores irán separados por comas y seguirán una sintaxis determinada dependiendo de su naturaleza:

- **Por nombre de etiqueta.** Cuando queremos especificar cómo mostrar todos los elementos del documento que tienen la misma etiqueta, el selector será el nombre de la etiqueta sin más. Por ejemplo: *div, p {text-align: left; user-select: none;}*. En el ejemplo, todo elemento con la etiqueta div o p que no posea una declaración CSS que lo contradiga, mostrará su texto alineado a la izquierda y no será seleccionable.

- **Por id.** Cuando queremos especificar cómo se muestra un elemento del que conocemos su id, el selector será una almohadilla (#) seguida del id. Por ejemplo: *#enunciado {text-align: center; color: green;}.* La etiqueta que tenga como atributo *id="enunciado"* mostrará su texto centrado y de color verde.

- **Por clase.** Cuando queremos especificar cómo se muestran todos los elementos de una determinada clase, esto es, un elemento cuya etiqueta o algún ancestro de esta tiene un atributo *class* definido. La sintaxis que sigue el selector es un punto (.) seguido del nombre de la clase. Por ejemplo: *.textoCursiva {text-align: justify; font-style: italic;}*

- **Por etiquetas con una determinada clase.** A veces solo queremos referirnos a unas determinadas etiquetas con una determinada clase, en tal caso la sintaxis es *nombreEtiqueta.clase*. Por ejemplo, si el selector fuera *p.textoCursiva* las declaraciones solo afectarían a las etiquetas *p* que tienen como clase definida *textoCursiva*.

- **Selector universal.** Se utiliza para cuando se quieren establecer unas declaraciones genéricas que afecten a todos los elementos del HTML y se escribe con un asterisco (*).

2.4.4 JavaScript

JavaScript es un lenguaje de script, esto es, es un lenguaje que incluye y ejecuta código en páginas web. Fue inventado por Netscape Communications y Fundación Mozilla y apareció en 1995.

Luján, S. (2001) nos define JavaScript como un lenguaje que cumple las siguientes tres características:

- **Es un lenguaje interpretado.** Es decir, el código en JavaScript se ejecuta línea a línea sin necesidad de que pase todo por un compilador que traduzca el documento completo a código máquina que el procesador pueda entender.

- **Es un lenguaje Basado en Objetos**, esto es, es un lenguaje cuyo funcionamiento se basa en la utilización de objetos. Phillips, D. (2018) define estos objetos de software como modelos de cosas que tienen ciertos comportamientos y datos asociados. De modo que un objeto podrá realizar ciertas acciones y, del mismo modo, se le podrán aplicar otras. Sin embargo, no se le puede considerar un lenguaje orientado a objetos puro, puesto que no existe la herencia entre clases, esto es, no presenta mecanismos para poder crear una nueva clase que se base en otra ya implementada.
- **Es Multiplataforma.** Un mismo código JavaScript puede ejecutarse directamente en múltiples plataformas sin necesidad de adaptar el código a las especificaciones de cada una.

Para la realización de los ítems interactivos se han puesto en común las tres tecnologías expuestas (HTML, CSS y JavaScript), de modo que a la hora de resolver ejercicios o de diseñarlos, se mostrará un documento HTML cuyo estilo vendrá definido en uno o varios ficheros CSS y que será modificado mediante las respuestas programadas en uno o más documentos JavaScript a las acciones de los respectivos usuarios.

3 BIBLIOTECA DE ÍTEMS

3.1 Metodología para le creación de nuevos ítems.

En este capítulo se va a ahondar en los tres ítems que se han desarrollado para este proyecto: Emparejamiento, Etiquetado de Imagen y Selección de Área.

Para la creación de nuevos ítems, en general, la meta es que sean usables y se integren correctamente en el sistema SIETTE. Puesto que la metodología que se ha seguido en este proyecto es una metodología ágil de tipo SCRUM, se ha trabajado por ciclos, tal y como se expone detalladamente en el aparatado *4. Metodología de trabajo.* Pero esa información no es suficiente para definir cuál sería la metodología óptima a seguir a la hora de crear nuevos ítems. En concreto, se han seguido tres pasos que se repiten por cada ciclo:

1. Definición de requisitos. Por cada prototipo se definen una serie de requisitos nuevos o se remodelan los que ya existían para ajustarse mejor a los objetivos que busca cumplir el ítem, así como atendiendo a la usabilidad de la aplicación

2. Desarrollo de prototipo. Para ello tanto el programador como el administrador de SIETTE deben mantener una comunicación constante en la toma de decisiones que vayan surgiendo durante el desarrollo. Ello es debido a que la aplicación recibe y envía información a al sistema SIETTE y, debido a la intencionalidad de la integración de los ítems externos es necesario que se trabaje en la compatibilidad constantemente.

3. Estudio de usabilidad. Una vez acabado el prototipo se estudia su usabilidad atendiendo a los puntos de si es eficaz, eficiente en el uso y satisfacción. El prototipo se prueba con varios ejercicios y se ve cómo podría mejorar. Tras este paso, se vuelve al paso 1, donde las conclusiones obtenidas de este apartado se tienen en cuenta para elaboración de requisitos.

Del mismo modo, como ya se ha explicado en apartados anteriores, cada ítem constará de 4 vistas diferenciadas, cada una con su diseño propio y su función específica.

3.2 Ítem de Emparejamiento

El primer ítem que se desea desarrollar es el ítem de Emparejamiento. Este tipo de pregunta es el clásico ejercicio en el que se presentan dos columnas y se ha de trazar líneas que conecten un elemento de una columna con uno o más de la otra.

Sus aplicaciones son muy variadas, desde lingüísticas (como pedir que se relacione una palabra con su traducción a un idioma extranjero) a artísticas (como relacionar una imagen de un monumento con su nombre o autor) e incluso matemáticas (relacionar una ecuación con su posible solución). La ventaja de este tipo de preguntas es que permite a los alumnos inferir relaciones entre las columnas, descartar información no relacionada y obtener información visual de conceptos que están ligados entre sí.

3.2.1 Solución Vista Profesor

A la hora de crear un nuevo ejercicio, el docente deberá cumplimentar un formulario en el que deberá dar valor a las variables que nos permitirán mostrar el ejercicio correctamente. Aunque en una primera fase del proyecto los docentes solo podían decidir el tipo de elemento que se utilizaría en una determinada columna y qué elementos componían estas, la lista de las variables aumentó debido a que se busca conseguir la mayor versatilidad posible en la creación y presentación de preguntas. Sin embargo, estas variables *extras* no son obligatorias de cumplimentar pues el algoritmo se encarga de darles un valor predeterminado en su ausencia.

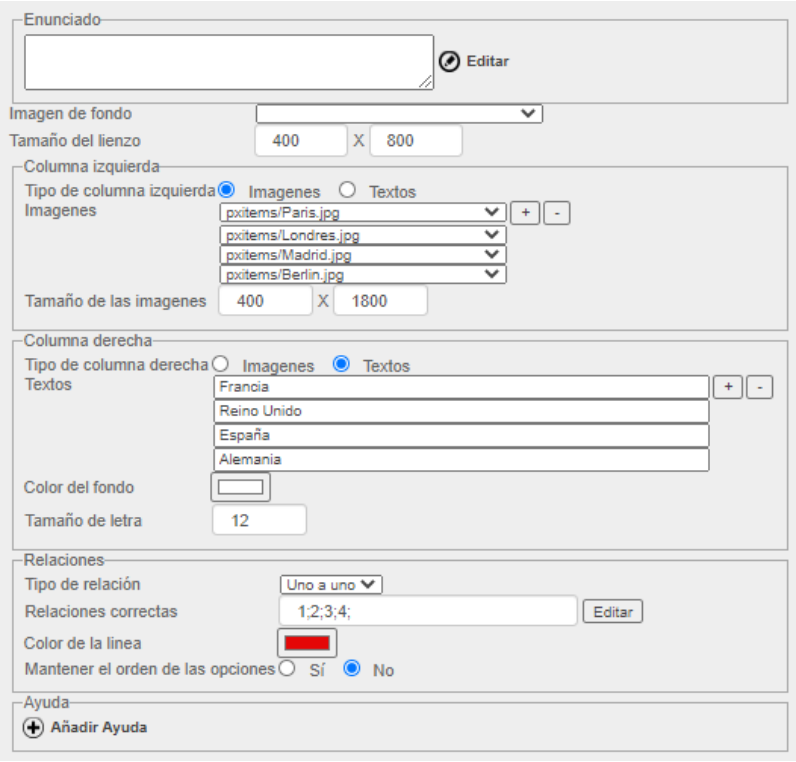

Figura 7. Formulario del Profesor para crear ejercicios de Emparejamiento

Primeramente, el docente tiene la posibilidad de escribir el *enunciado* del ejercicio. A continuación, como el ítem se engloba en un marco sin fondo, puede seleccionar una *imagen de fondo* de su galería o de las que estén ya cargadas en SIETTE. Del mismo modo, podrá elegir el *Tamaño de Lienzo*, que se refiere al tamaño que se desee que ocupe el ejercicio completo. Sin embargo, si el tamaño que se proporciona es excesivamente pequeño en comparación con el tamaño de imagen dado, las especificaciones dadas por el docente podrán ser sobrescritas por el algoritmo

Cada columna puede contener imágenes o texto. El docente debe rellenar cuál de estas dos posibilidades desea. En la Figura 7, la columna izquierda será de Imágenes, por lo que el docente deberá introducir la url de la imagen que desee para cada elemento de la columna (SIETTE le abrirá un desplegable para que elija las imágenes ya cargadas en el sistema por el profesor). En el caso de la columna derecha, el docente seleccionó que iba a ser de texto, por lo que debe introducir el texto que se presentará por cada elemento de la columna. Además, en las columnas de imagen, se podrá seleccionar cuál es el tamaño deseado de imagen; y en las columnas de texto, cuál es el tamaño de fuente deseado y de qué color.

El tamaño de las imágenes es el tamaño en píxeles en ancho por alto en el que se quiere que se muestren todas las imágenes. Como es importante conservar la relación de aspecto de las imágenes, esto es, queremos evitar que estas se deformen, estos valores constituirán el máximo de ancho y alto que puede llegar alcanzar cada imagen.

La variable color de línea permite al docente elegir el color en el que quiere que aparezcan mostradas las relaciones entre las columnas a modo de que, si eligen una imagen de fondo oscura, por ejemplo, se pueda usar un color claro en las líneas y estas sean visibles.

Otra variable de obligatorio cumplimiento y suma importancia es el "tipo de relación". Es posible que tengamos ejercicios en los que queramos hacer una relación uno a uno, como es el caso del formulario de la Figura 7, pero también se puede dar el caso en el que nos interese que haya relaciones de un elemento de una columna con varios de otra. Este segundo caso sería el de las relaciones *múltiples*.

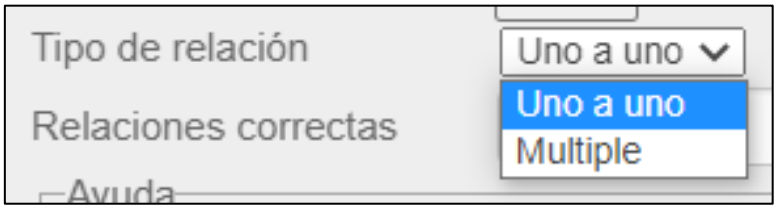

Figura 8. Tipos de relaciones en Emparejamiento

También está la variable "relaciones correctas" en la que el docente puede rellenar, en texto plano y siguiendo una sintaxis determinada, la solución al ejercicio. Debido a la complejidad que esto puede presentar, se le ha preparado al docente la

posibilidad de solucionar el ejercicio igual que lo haría un alumno usando el botón de *Editar* que aparece en la parte superior del formulario. Al pulsar dicho botón con el ratón se le mostrará una ventana en la que aparecerá el ejercicio tal y como se ha cumplimentado en el formulario. Una vez acabe de rellenar el problema con las respuestas correctas, esta solución quedará registrada en el campo "relaciones correctas".

Por último, tenemos la opción de *mantener el orden de las opciones.* Esto es, el docente puede elegir si quiere que, cuando el ejercicio se presente al alumno, se muestren los elementos de las columnas en el mismo orden en el que él las puso o si quiere que se presenten de forma aleatoria.

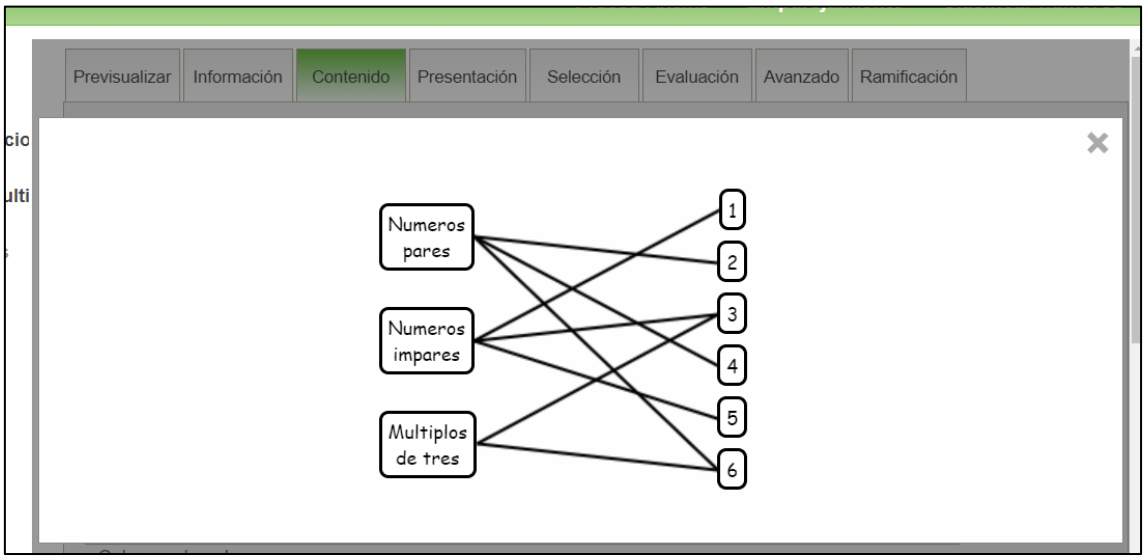

Figura 9. Vista del docente de la edición de un ejercicio de Emparejamiento

3.2.2 Solución Vista Alumno

En la siguiente imagen se muestra cómo se vería un ejercicio a realizar por un alumno. Cuando el ejercicio se le muestra, como es de esperar, las relaciones no están dibujadas. Según las especificaciones dadas por el docente a la hora de diseñar el ejercicio, los alumnos podrán o no realizar múltiples relaciones entre un elemento de una columna y los elementos de la siguiente.

Figura 10. Vista de Alumno de un ejercicio de Emparejamiento

El modo de resolución del ítem es simple. Cuando el alumno identifique una relación entre una columna y otra tendrá que hacer clic con el botón izquierdo del ratón sobre cualquiera de los dos elementos relacionados y mantener presionado el botón mientras arrastra el ratón hacia el elemento deseado de la columna contraria. Si los elementos relacionados pertenecen a columnas distintas, entonces la relación será dibujada.

En caso de que sea un ejercicio que tiene relación múltiple, no se restringirá el número de relaciones entre los elementos. Si el ejercicio, por el contrario, se trata de una relación de uno a uno, entonces si se dibuja una relación y uno de los elementos ya está relacionado con otro, la relación previa será eliminada, prevaleciendo la última que haya sido dibujada por el alumno.

Por último, si el alumno (o el docente que está diseñando la pregunta) desea eliminar una relación que ha dibujado, ya sea por inseguridad o por que se ha tratado de un error, solo tendrá que hacer doble clic sobre la línea y la relación será borrada.

Esta vista será la que trabajen los alumnos para poder resolver el ejercicio propuesto por el docente y, así mismo, una vista parecida (como la que se ve en la Figura 9) será la que se presente a los profesores para poder diseñar la respuesta correcta al ítem.

Así, el diagrama de casos de uso del ítem Emparejamiento para el alumno quedaría cómo:

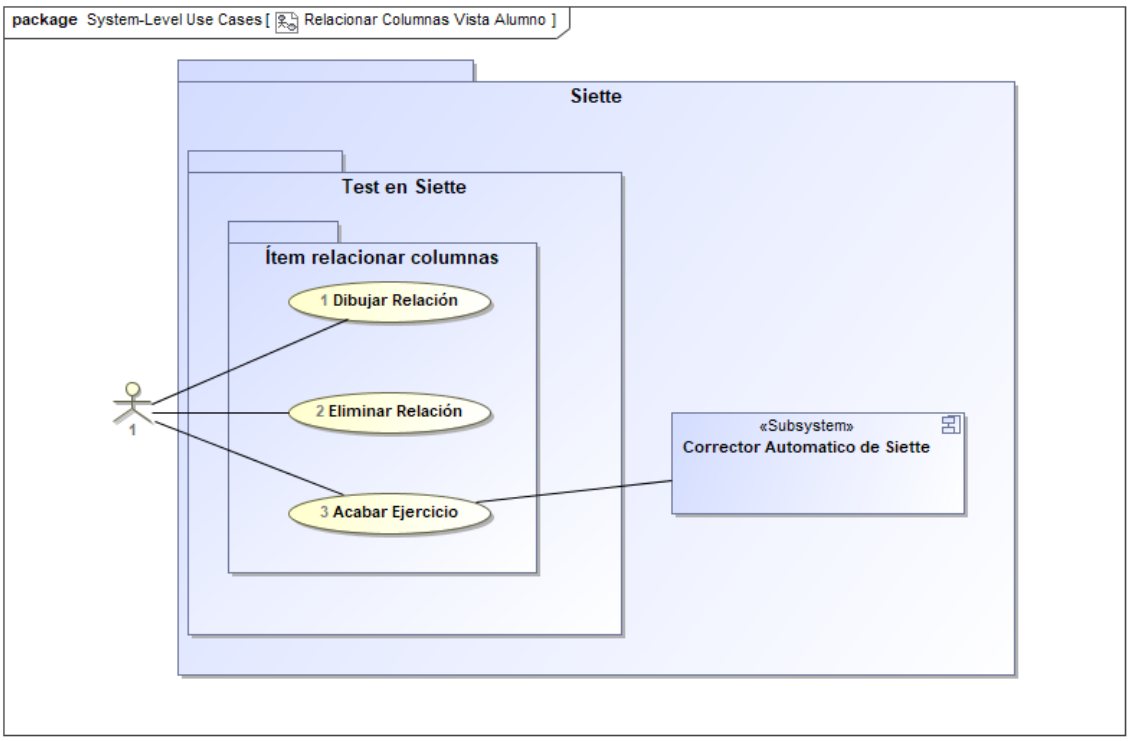

Figura 11. Diagrama de casos de uso del alumno para los ítems de Emparejamiento

3.2.3 Solución Vista Resolver y Solución

Una vez el alumno entrega su respuesta se le pueden presentar las vistas resolver y solución. Estas vistas permiten al alumno ser capaz de identificar donde están sus errores (si los hubiera) y cuál sería la solución correcta del ejercicio, para que así pueda aprender y mejorar su rendimiento en la materia.

La vista resolver es la primera en aparecer y consiste en mostrar al alumno si ha hecho bien o no el ejercicio. Por ejemplo, supongamos que tenemos el ejercicio que aparece en la figura 12 con las mismas respuestas dadas por el alumno. Claramente, el primer y último elemento de la columna izquierda están bien relacionados; el segundo no tiene respuesta dada por el alumno; y el tercero está mal relacionado. La vista resolver posee los elementos visuales apropiados para poder representar los tres casos descritos (sin respuesta, correcto o incorrecto). Para mostrarlo, a cada elemento de la columna izquierda se le colocará una imagen, que podrá ser una equis roja (incorrecto), un aspa verde (correcto) o un aspa roja (sin respuesta), como se puede observar en la figura 13.

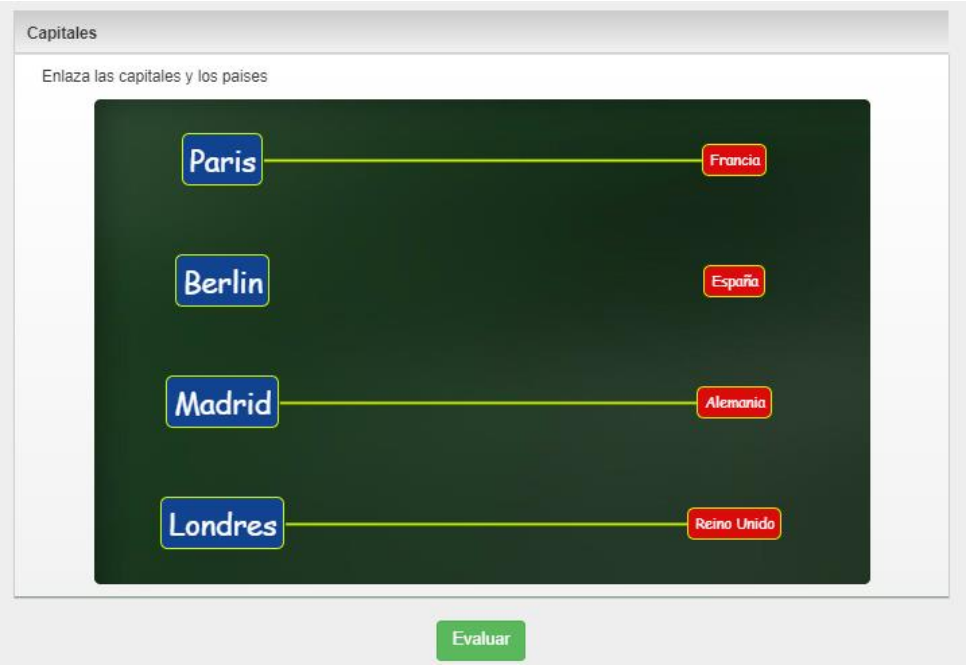

Figura 12. Ejemplo de respuesta de alumno en un ejercicio de Emparejamiento

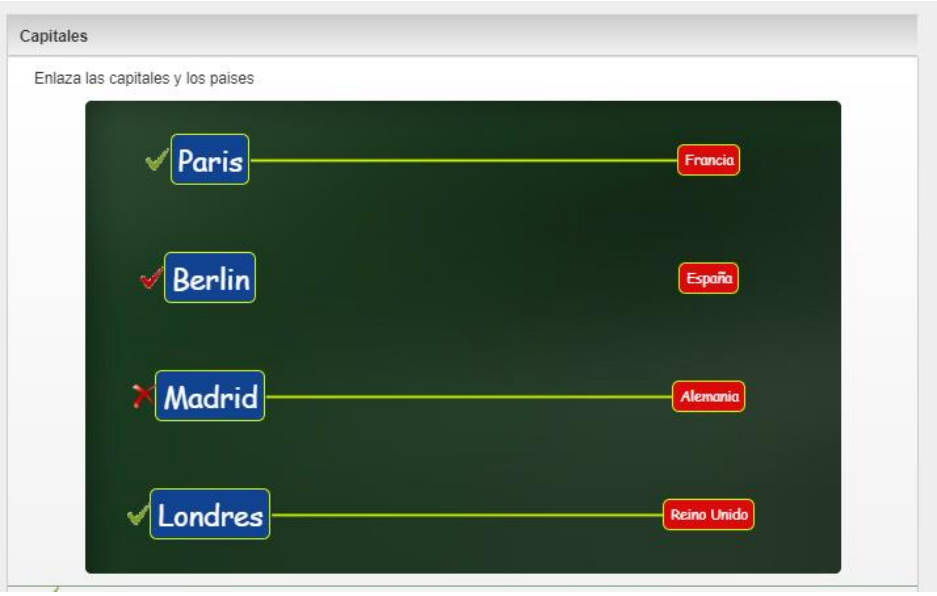

Figura 13. Visión Resolver de Emparejamiento

Respecto de la vista solución, se accede a ella clicando sobre la vista resolver. Del mismo modo, una vez en la vista resolver, se puede acceder a la vista solución haciendo clic sobre ella. Esta vista es idéntica a la vista resolver si el alumno hubiese respondido todo correctamente (Figura 14).

Figura 14. Vista solución de un ejercicio de Emparejamiento

3.2.4 Solución Técnica: Vista Alumno y Vista Profesor

A la hora de enfrentarse al problema, lo primero que hubo de tenerse en cuenta fue cómo iba a poder el alumno trazar una línea y cómo se iba a dibujar esta misma. Dos fueron las opciones que se desarrollaron, la primera, más simple pero menos interactiva, era hacer clic en un elemento de una columna, luego hacer clic sobre un elemento de otra y que, por último, la línea se dibujase por su cuenta dados estos parámetros. Sin embargo, y puesto que se buscaba la creación de ítems lo más interactivos posibles, se decidió optar por cambiar el concepto y que el usuario, haciendo clic y arrastrando el ratón, fuese dibujando una línea recta que comenzase en su clic y acabase en la posición del ratón. En el momento en que el usuario deja de pulsar el botón del ratón, se hace una comprobación de la línea dibujada y, si todo está correcto, la línea se reajusta para mejorar la visibilidad del ejercicio.

Para lograr todo ello, se tuvo que realizar una tarea de documentación sobre los *canvas* de HTML y cómo trabajar con ellos mediante JavaScript. Canvas es un elemento de HTML que permite dibujar en él, hacer animaciones o composiciones de fotos entre otras muchas cosas. Su principal función es facilitar el uso de elementos gráficos dinámicos en la programación web. Es este elemento el que se utilizó para poder pintar las líneas en pantalla. JavaScript, además, permite un buen manejo del canvas mediante instrucciones simples que permiten dibujar líneas, cambiar la resolución del canvas o incluso dibujar diversos tipos de formas geométricas.

Figura 15. Alumno dibujando una línea en un ejercicio de Emparejamiento

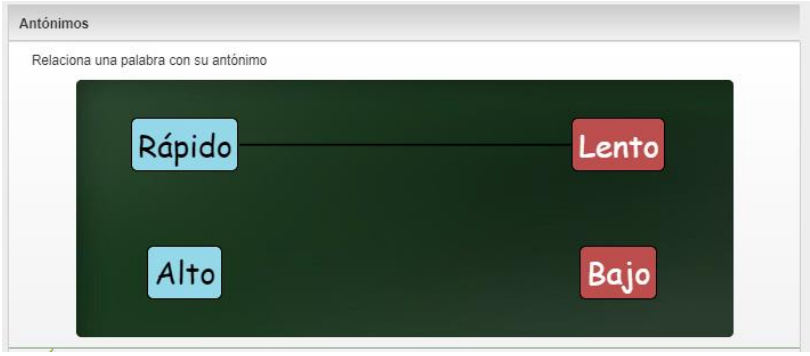

Figura 16. Línea autoajustada en un ejercicio de Emparejamiento

Antes de mostrar código relativo al funcionamiento interno del ítem, cabe destacar la implementación de la clase *LineaEmparejamiento*. Esta clase nos permite definir cada una de las líneas y, por consiguiente, relaciones que el alumno o profesor han dibujado entre las dos columnas. Además, cuenta con una serie de operaciones que nos permiten saber cuál es la distancia mínima entre un clic del ratón y el segmento dibujado.

Cada línea tendrá como atributos la posición x e y (*xinit e yinit*) en las que la línea se empieza a dibujar respecto al canvas; las coordenadas x e y (*xend* e *yend*) en las que la línea termina respecto al canvas; y los id del elemento en el que empieza la linea (*init*) y del elemento en el que acaba (*end*).

La función "*pointIn*", en un principio tomaba unas coordenadas en el canvas y calculaba si estas estaban dentro del recorrido que traza la línea. El interés de esta función era que, al hacer doble clic sobre la relación, esta pudiera ser borrada. Sin embargo, para más comodidad del usuario, se rehízo la función de modo que llamase a "*distToSegment*", que calcula la distancia mínima del punto dado a la línea. De este modo, si la distancia mínima es de diez píxeles o menos, se considera que el doble clic se ha producido sobre la línea. Del modo anterior, se requería una precisión que podría causar más problemas que soluciones.

```
class LineaEmparejamiento {
    constructor(xinit, yinit) {
        this.xinit = xinit;
        this.yinit = yinit;
        this.yend = undefined;
        this.xend = undefined;
        this.init = undefined;
        this.end = undefined;
    }

    printLinea() { ... }
    sqr(x) { ... }
    dist2(v, w) { ... }
    distToSegmentSquared(p, v, w) { ... }
    pointIn(px, py) { ... }
}
```

Figura 17. Código de la clase LineaEmparejamiento para el ítem Emparejamiento

Otra decisión de suma importancia fue la de cómo iba a ser representada la solución del ejercicio. De nuevo, dos fueron las soluciones planteadas: la primera, más intuitiva pero menos compacta, consistía en asignar a cada columna una letra. La columna izquierda sería la columna "A", y la columna derecha la columna "B". Cada elemento de cada columna tendría un número asociado que iría de 1 a *n*, siendo n el número de elementos de la columna en cuestión. Así, por ejemplo, la columna izquierda tendría los elementos A1, A2, ..., An. Cada relación hecha por el alumno sería representada con un AiBj, siendo Ai el elemento en la posición i de la columna izquierda y Bj el elemento en la posición j de la columna de la derecha. Por último, como separación entre relación y relación, tendríamos el símbolo ";".

Esta manera de representar la relación era bastante tosca y poco concisa, especialmente cuando se permite relaciones múltiples. Por ello se ideó una nueva forma de representar las relaciones en la que el número de caracteres necesarios se redujo considerablemente y, a la hora de realizar relaciones múltiples, se pudiese ver claramente cuando un elemento estaba relacionado con varios, uno o ninguno. Como el ejercicio tan solo consta de dos columnas distintas, solo nos interesan los números que representen elementos de una de las dos columnas. Esto es, simplemente posicionando correctamente los números y separándolos de manera adecuada, se puede obtener toda la información de las relaciones usando únicamente números, comas, y puntos y comas. Cada número significa la posición del elemento relacionado en la columna de la derecha; cada coma, implica que el siguiente número está relacionado con el mismo elemento de la columna de la izquierda que el anterior; y cada punto y coma implica que pasamos a ver las relaciones que tiene el siguiente elemento de la columna de la izquierda. Por ejemplo, en la figura 15 vemos cómo la columna izquierda tiene dos elementos y no hay relaciones hechas, por tanto la solución en ese momento sería ";;", que se podría leer como: "Respecto al primer elemento de la columna izquierda, no hay ninguna relación; respecto al segundo elemento de la columna izquierda, no hay ninguna relación". En cambio, en la figura 16 el primer elemento de la izquierda está relacionado con el primer elemento de la

derecha, esto es "1;;", que se podría leer como: "Respecto al primer elemento de la columna de la izquierda, está relacionado con el primer elemento de la columna de la derecha; respecto al segundo elemento de la columna de la izquierda, no tiene ninguna relación". Este método de escribir la solución, aunque algo complejo de explicar, permite un entendimiento rápido de la situación de la solución y una comparación casi inmediata por software, ya que asegura que no hay ambigüedad posible en la solución.

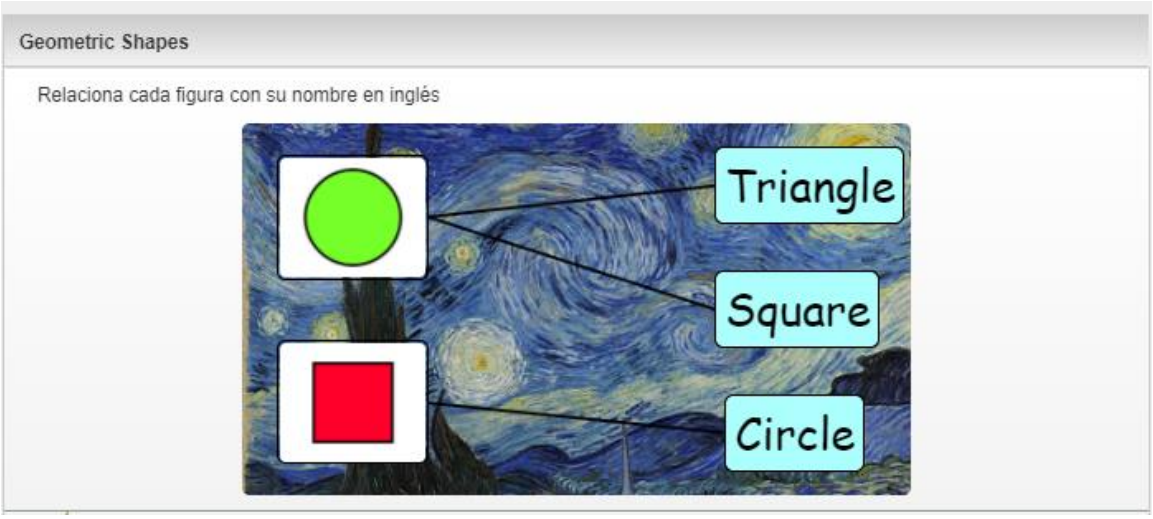

Figura 18. Ejemplo relación múltiple en Emparejamiento

En la figura 18, la solución en el primer caso podría ser en el sistema antiguo: "A1B1;A1B2;A2B3" o "A1B2;A1B1;A2B3". En cambio, en el nuevo sistema, lo único permitido es "1,2;3;", que podría leerse como "Respecto al primer elemento de la columna de la izquierda está relacionado con el primer y segundo elemento de la columna de la derecha; respecto al segundo elemento de la columna izquierda está relacionado con el tercero de la columna derecha". Como se ve, no solo se eliminaron las ambigüedades, pues se obliga en el nuevo método a que las relaciones sigan un orden creciente en su representación, sino que es una solución más compacta y manejable en la que se elimina información redundante.

Una vez tomadas todas las consideraciones, lo único que queda es programar un canvas funcional y una estructura que almacene las relaciones para poder trabajar con ellas antes de su transformación final en la cadena de caracteres explicada en los párrafos anteriores.

Para almacenar las relaciones que va realizando el alumno se utiliza un array que contiene elementos de la clase *LineaEmparejamiento*. Cuando se dibuja una nueva línea, se añade el elemento al array y cuando se borra una línea también se extrae de la estructura en cuestión. Todo ello es posible gracias a las cuatro funciones siguientes:

```
//Cuando empezamos a dibujar la linea
canvas.addEventListener('mousedown', function (event) {...});

//moviendo linea
canvas.addEventListener('mousemove', function (event) {...});

//Acabando linea
canvas.addEventListener('mouseup', function (event) {...})

//Borramos lineas
canvas.addEventListener("dblclick", function (event) {...})
```

Figura 19. Funciones de canvas para Emparejamiento

Las tres primeras funciones son las encargadas de dibujar la línea, almacenarla en caso de que sea una línea válida, y eliminar aquella o aquellas que sean necesarias para respetar los parámetros del ítem. Cuando pulsamos el botón izquierdo del ratón y estamos sobre el canvas, lo primero que se hace es comprobar si estamos en una posición que corresponda a uno de los elementos de las columnas. Si se pulsa el botón en cualquier otro sitio del canvas nada ocurre. En caso de que, efectivamente, todo esté correcto para poder iniciar el dibujo de la línea, se almacena en unas variables auxiliares la información relativa al elemento clicado y se pone una variable de control llamada "pintar" a true. Mientras esta variable esté con valor verdadero y el botón esté pulsado se dibujará una línea en el canvas que vaya desde las coordenadas en las que se pulsó el botón hasta las coordenadas actuales del ratón. Cuando el usuario levanta su dedo del botón izquierdo, se ejecuta la tercera función. Esta última función simplemente calcula si el evento ha ocurrido sobre un elemento válido y, en ese caso, reajusta la línea para que quede más estética y realiza las funciones explicadas al inicio del párrafo.

La última función simplemente elimina una relación de la estructura "lineas" cuando se realiza un doble clic a 10 píxeles o menos de esta.

Todo este proceso, requiere de un redibujado constante del canvas, de modo que se pueda ir reajustando la línea que dibuja el usuario sin dejar los trazos antiguos de esta en el movimiento del ratón. Para ello, en las tres últimas funciones, se hace siempre una llamada a la función "printAllPreviousLines" que vacía el canvas y redibuja todas las relaciones que se encuentran en la estructura "lineas".

```
function printAllPreviousLines() {

    var ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    lineas.forEach(element => {
        ctx.beginPath()
        var xinit = element.xinit - canvas.offsetLeft;
        var yinit = element.yinit - canvas.offsetTop;
        ctx.moveTo(xinit, yinit);

        var xend = element.xend - canvas.offsetLeft;
        var yend = element.yend - canvas.offsetTop;
        ctx.lineTo(xend, yend);
        ctx.strokeStyle = lineColor;
        ctx.lineWidth = '2';
        ctx.stroke();
    })
}
```

Figura 20. Función que imprime en el canvas todas las relaciones

3.2.5 Solución Técnica: Vista Resolver y Vista Solución

Para entender mejor cómo funciona el código que representa las vistas resolver y solución, primero hay que saber qué es lo que hace SIETTE y cómo llama a las funciones correspondientes.

SIETTE primero ejecutará estas tres funciones:

```
function codificar(respuesta,correccion) {...}
function resolver(respuesta,correccion) {...}
function solucion(respuestaCorrecta) {...}
```

Figura 21. Funciones codificar, resolver y solución de SIETTE para el ítem Emparejamiento

Sin entrar mucho en detalle, la función codificar toma como parámetros la respuesta dada por el alumno y la corrección de SIETTE dada a dicha solución. Esta corrección es un array de valores booleanos (true para respuesta correcta y false para respuesta incorrecta). Esta función se encarga de coger los arrays de respuesta y corrección y reordenarlos para que se presenten con el mismo orden en el que alumno realizó el ejercicio. Esta función nace de la posibilidad que tiene SIETTE de desordenar los elementos de la columna. Cuando la respuesta dada por el alumno se envía al sistema SIETTE, esta respuesta se ordena para que se pueda comparar con la solución correcta propuesta por el docente. Por tanto, cuando se quiere volver a representar el ejercicio para la vista resolver y la vista solución se han de desordenar de nuevo.

Por otro lado, la función resolver llama a la función codificar para poder obtener los parámetros bien ordenados y luego llama a la función implementada en el fichero *emparejamiento.js* "resolverEmparejamiento".

Del mismo modo ocurre con la función solución. Se llama a codificar, pasando como parámetro corrección un array cuyos elementos son todos true, y luego se llama a la función "soluciónEmparejamiento", implementada en el archivo *emparejamiento.js*.

```
function resolverEmparejamiento(respuesta,
                                correccion,
                                lineColor,
                                length,
                                myName) {...}

function solucionEmparejamiento(respuesta,
                                lineColor,
                                length,
                                myName) {...}
```

Figura 22. Funciones resolverEmparejamiento y soluciónEmparejamiento

La función resolverEmparejamiento se encarga de presentar al alumno la vista resolver. Para ello utiliza los parámetros

- **Respuesta:** Contiene la respuesta dada por el alumno siguiendo el formato explicado en el apartado 4.5.1. Esta será la respuesta que dibujará en el canvas.
- **Corrección:** Es un vector que contiene valores booleanos en función de si la respuesta es correcta o incorrecta.
- **LineColor:** El color que tendrán las líneas que se dibujarán sobre el canvas.
- **Length:** Es un valor que contiene la suma del número de elementos de la columna izquierda y derecha. Es un valor para control de posibles errores en la carga del ejercicio.
- **MyName:** Esta variable sirve para, en el caso de múltiples ejercicios en una misma pantalla, ser capaz de identificar cuál de todos los ejercicios mostrados es sobre el que tenemos que mostrar la vista resolver.

La función resolverEmparejamiento no se ejecuta completamente en el momento de su llamada. Lo mismo ocurre con la función solucionEmparejamiento. Esto se debe a que, cuando se hace la invocación de la función, el ejercicio como tal no está mostrado aún en pantalla, por lo que el canvas no puede ser dibujado. Por ello se ha creado un nuevo evento llamado "loadingCol", que cuando salta la función resolverEmparejamiento y solucionEmparejamiento lo recoge y ejecuta el resto de la función con normalidad.

Son dos tipos de elementos los que se crean en la llamada a ambas funciones. El primero, los canvas. Cada una crea su propio canvas con las relaciones correspondientes dibujadas. Sin embargo, el canvas de la vista resolver será el único que se muestre en un primer momento. El canvas de la vista solución tendrá un display none, en otras palabras, estará oculto.

También se generarán una serie de imágenes que irán colocadas a la izquierda de los elementos de la columna izquierda. Estas imágenes, ya descritas anteriormente, serán la cruz roja, el aspa verde y el aspa roja. Al igual que el canvas, solo se mostrarán en un momento las imágenes de la vista resolver.

El cambio de vista se hará mediante evento. Cuando un canvas detecta un clic sobre él, pone su display y el de todas las fotos de su vista en none y pone el display de los elementos de la otra vista en inherit.

3.2.6 Aspectos de Usabilidad del ítem

El ítem se ha desarrollado pensando en su usabilidad de cara a los posibles usuarios reales. Es por esto que siempre se ha optado por la mayor interactividad del usuario con la aplicación. Por ello, se tomaron decisiones como poder dibujar las líneas con el ratón a mano en lugar de hacerlo mediante simples clics sobre los elementos de cada columna. El objetivo es simple, hacer que el usuario pueda percibir la aplicación como completamente intuitiva, de modo que, a la hora de volver a enfrentarse con un ítem de este tipo no sienta la existencia de una barrera que le impide demostrar su conocimiento sobre un tema dado.

Uno de los aspectos de usabilidad de este ítem acaba de ser mencionado: la posibilidad de dibujar líneas directamente con el ratón. Así mismo, esto implicaba programar un comando que el usuario pueda deducir con facilidad para eliminar las líneas ya dibujadas. Se optó por la opción que parecía más intuitiva, que es el doble clic sobre la línea.

Aun habiéndose diseñado todo pensando en la inmediatez del usuario para saber cómo funciona la aplicación, se ha optado por añadir una ayuda para todos los ítems que se podrá presentar a los alumnos.

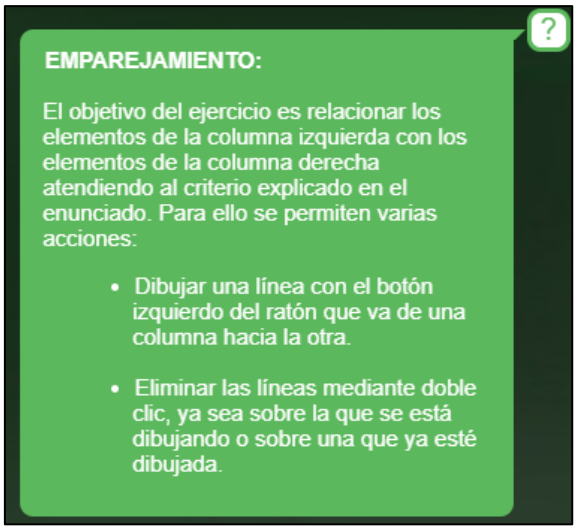

Figura 23. Texto de ayuda para el ítem Emparejamiento

3.3 Ítem de Etiquetado de Imagen

El segundo ítem en el que se ha trabajado es en Etiquetado de Imagen. Consiste en identificar determinados elementos en una imagen. Este ejercicio puede ser bastante útil cuando se busca que el alumno consiga una memoria visual respecto a un concept dado, como pueden ser las partes de una flor, los tipos de columnas griegas o los nombres de los planetas del Sistema Solar.

3.3.1 Solución Vista Profesor

Al igual que en el ejercicio de Emparejamiento, el docente deberá cumplimentar un formulario en el que describa cómo quiere el ejercicio sea visualizado. Dicho formulario se puede ver en la Figura 24.

Como podemos observar, lo primero que debe rellenar el docente es el enunciado del ejercicio, donde deberá indicar al alumno qué es lo que se debe hacer. A continuación, tenemos el campo *imagen de fondo.* La imagen que se introduzca ahí será sobre la que el alumno deberá nombrar los distintos elementos.

El campo *posición relativa de la imagen* nos permite seleccionar dónde queremos que se encuentre la imagen. El docente deberá abrir el desplegable y seleccionar entre las opciones: centrado, alineado arriba, alineado abajo, alineado a izquierda, alineado a derecha, esquina superior izquierda, esquina superior derecha, esquina inferior izquierda y esquina superior derecha.

El campo tamaño del lienzo y tamaño de la imagen sirven para indicar cómo de grande desea el tutor (en ancho por alto) que sea el ejercicio o la imagen respectivamente.

El *número máximo de caracteres* no supone un límite a la hora de escribir las etiquetas, pero sí un límite en cuánto se podrá visualizar del texto escrito. Esto es, un

número máximo de 10, implicará que a la vez, por cada etiqueta se verán diez caracteres, el resto estarán ocultos a izquierda o derecha.

El *tamaño de letra* complementa el campo anterior, pues según cuán grande o pequeña sea este, del mismo modo será el tamaño del campo de texto.

El *color de la línea* abrirá un selector de color. Esta línea será la que permita a los docentes relacionar el campo de texto o etiqueta con el punto de la imagen a etiquetar.

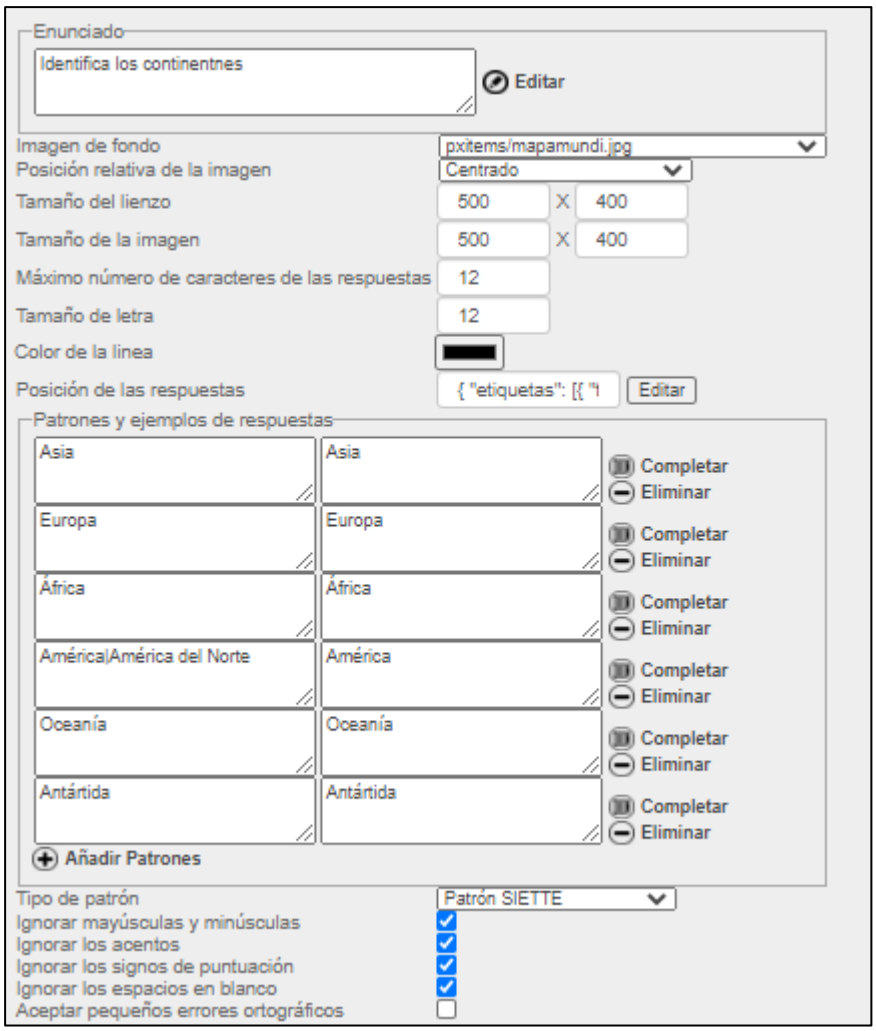

Figura 24. Formulario Profesor en ítem Etiquetado de Imagen

El campo *posición de las respuestas* es, en realidad, un campo invisible para el profesor, que solo verá el botón editar. En este campo se guarda un texto con un formato JSON que se explicará en el apartado 5.5, a grosso modo, este texto guarda toda la información relevante a la posición de las etiquetas y de sus líneas.

El botón *editar*, así mismo, despliega al profesor una vista preliminar en la que, como explicaré al final de este apartado, podrá colocar todo correctamente en el ejercicio.

El campo *patrones y ejemplos de respuesta*. Este campo está divido en dos columnas y se complementa con el campo *tipo de patrón*. La columna de la izquierda contiene los patrones, esto es, todas las posibles respuestas que se pueden dar por válidas en el ejercicio. Para ello, tipo de patrón te deja seleccionar qué tipo de expresión regular se desea usar. Como podemos ver en la figura 24, se permite al alumno escribir América o América del Norte dando ambas respuestas por válidas.

Por otro lado, en la segunda columna se pone un ejemplo de respuesta para la previsualización. Esto es, en el caso de "América|América del Norte", el ejemplo elegido por el docente es América, por tanto, en la previsualización del profesor, este será el nombre que aparezca en la etiqueta.

Por último, aparecen varias casillas que puede marcar o no el docente para que a la hora de la corrección se tengan o no en cuenta mayúsculas, minúsculas, pequeñas faltas de ortografía, etc.

En la vista editar, si no se han modificado nunca las etiquetas, estas se colocarán de manera ordenada arriba y/o abajo de la imagen, en función de la colocación de esta (ver Figura 25). Dada esta vista, el profesor puede mover las etiquetas para su colocación en el punto deseado. Para ello deberá clicar sobre la etiqueta y, sin levantar el dedo del botón izquierdo del ratón, arrastrarla hasta la posición deseada. En un inicio, esta funcionalidad era completamente diferente, el docente simplemente iba clicando sobre la imagen y las etiquetas iban apareciendo en orden. En busca de un ejercicio lo más interactivo posible, era más lógico utilizar una representación ordenada de las etiquetas que permitiera al docente una colocación totalmente al gusto y regulable en cada momento.

Una vez colocada en el lugar deseado, se hará clic en la imagen para iniciar el dibujo de una línea. Mediante los clics se irán dibujando los distintos segmentos hasta que uno de los clics se haga sobre alguna de las etiquetas. En este caso, si esta tuviera alguna otra línea se borraría y se le asignaría la nueva. Un ejemplo de trazado de línea lo encontramos en la figura 26.

Por otro lado, si el docente decidiera deja de pintar la línea, tan solo debe hacer doble clic. Esta misma operación se puede hacer sobre una línea ya dibujada por eliminarla.

Para evitar problemas y que el dibujo de los trazos quede sucio, se decidió que, si una etiqueta se movía de sitio, si tuviera alguna línea, esta desaparecería.

Cuando el docente cierre esta previsualización del ejercicio, toda la información se guardará automáticamente en el campo *posición de la respuesta* ya mencionado anteriormente.

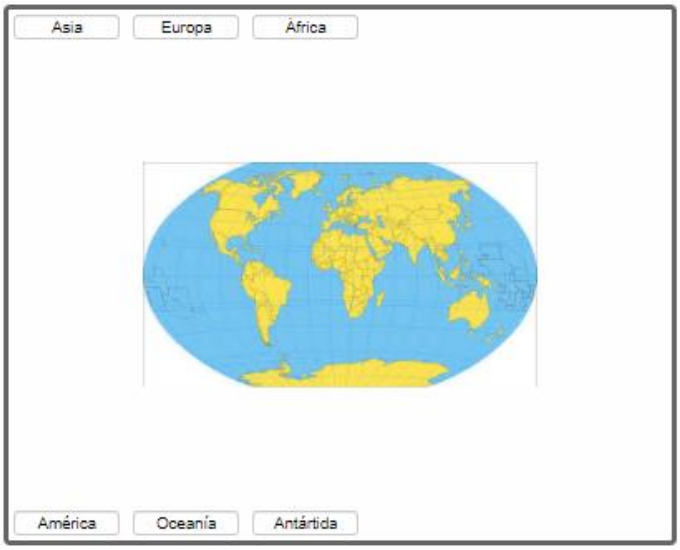

Figura 25. Vista de Profesor de un ejercicio de Etiquetado de Imagen con las etiquetas sin colocar

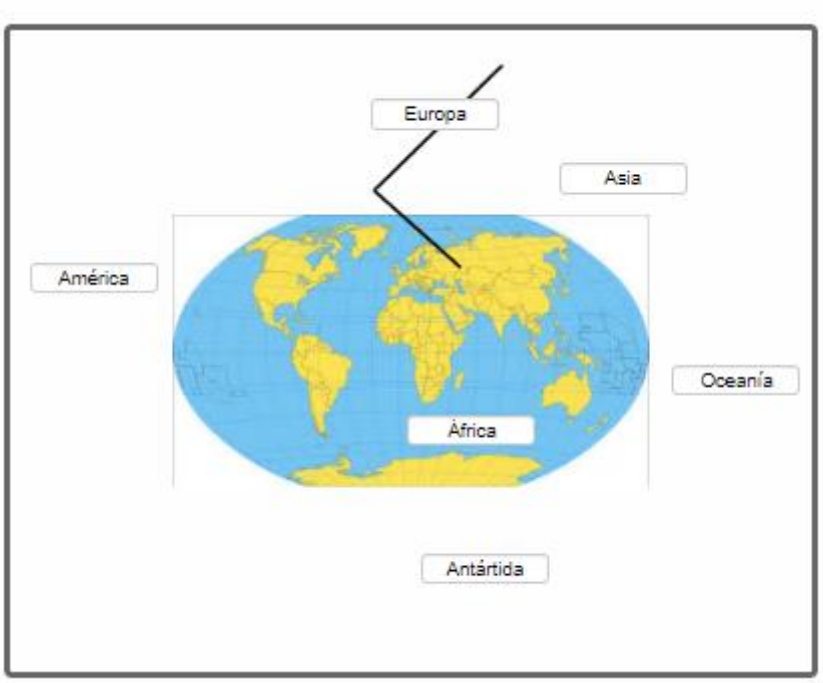

Figura 26. Trazado de línea sin terminar por docente de Etiquetado de Imagen

Así, el diagrama de casos de uso del editor del ítem etiquetado de imagen para el docente quedaría cómo:

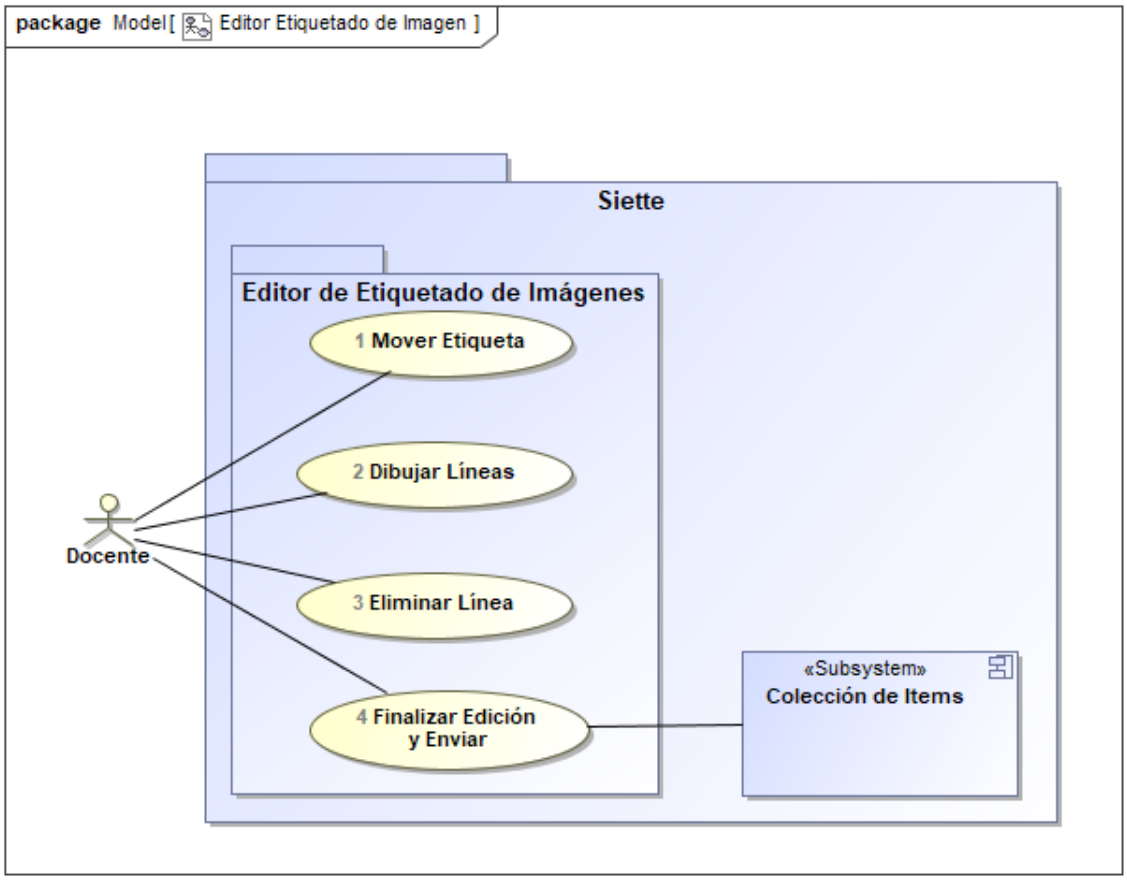

Figura 27. Diagrama de Casos de Uso de las acciones del docente en el editor de etiquetado de imágenes

3.3.2 Solución Vista Alumno

La vista del alumno simplemente consta de las etiquetas colocadas según la disposición del profesor. Las etiquetas serán campos de texto en blanco que el alumno deberá rellenar, tal y como se muestra en la siguiente imagen.

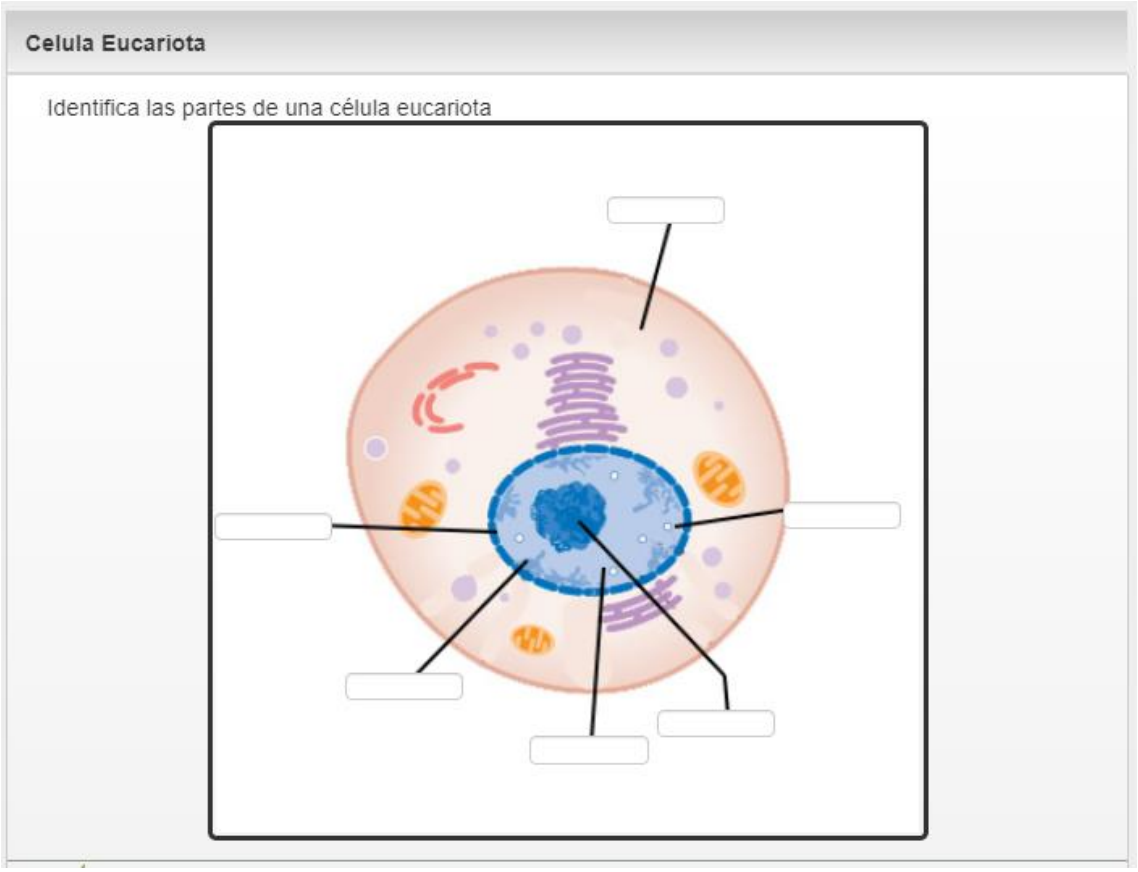

Figura 28. Vista de Alumno de un ejercicio de Etiquetado de Imagen

3.3.3 Solución Vista Resolver y Solución

La vista resolver y solución, del mismo modo que en Emparejamiento, se irán cambiando una por la otra mediante clics del usuario sobre el ejercicio.

La vista resolver usa las mismas tres imágenes que en el ejercicio Emparejamiento: una cruz roja para las etiquetas incorrectas, un aspa verde para las respuestas correctas y un aspa roja para cuando no hay respuesta. La representación de la corrección es bastante simple. Se muestran las etiquetas con la imagen de la corrección en su interior y a continuación el texto escrito por el alumno. La vista resolver se puede ver en la figura 29.

En el caso de la vista solución, el contenido de todas las etiquetas será el aspa verde seguida por el ejemplo que dio el docente en el formulario para cada una. La vista solución se puede ver en la figura 30.

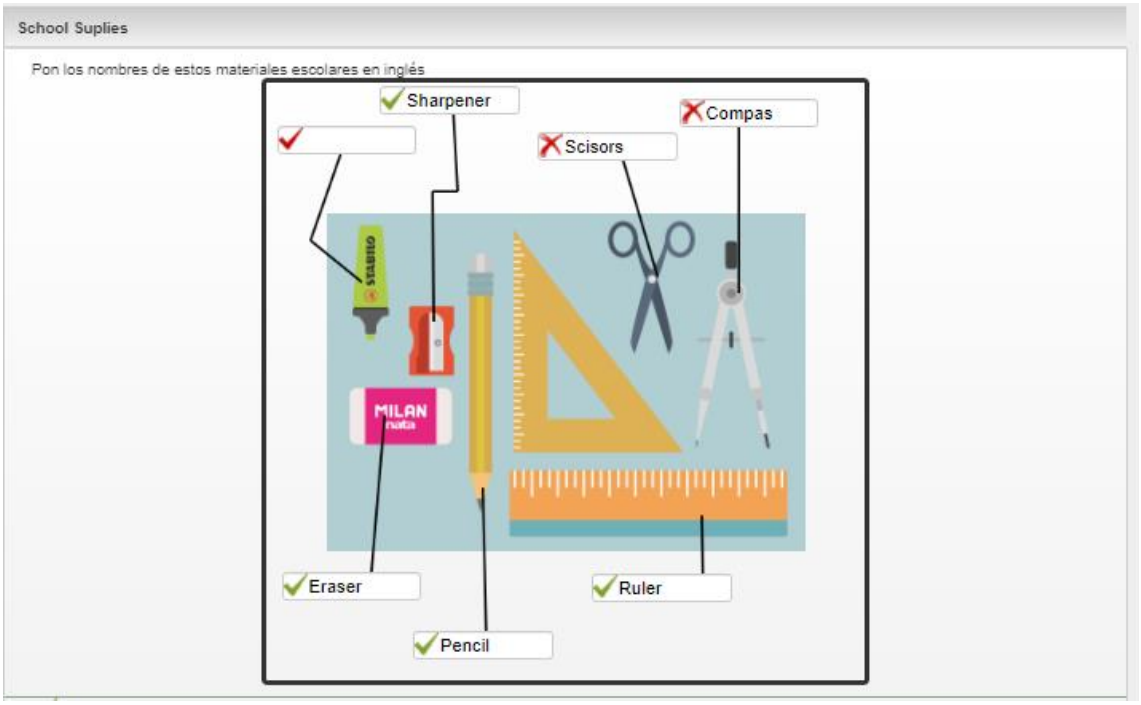

Figura 29. Vista resolver en ítem Etiquetado de Imagen

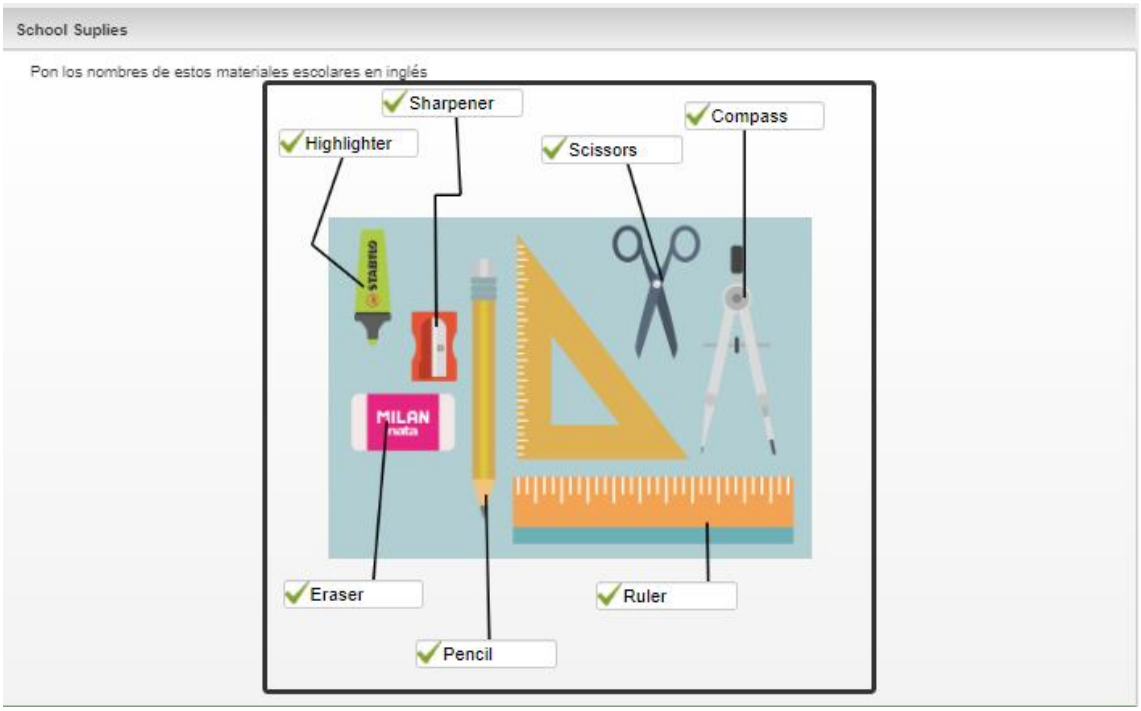

Figura 30. Vista Solución en ítem Etiquetado Imagen

3.3.4 Solución Técnica: Vista Profesor y Vista Alumno

Para la realización de este ítem, primeramente, se han definido dos clases. La primera clase es la denominada "LineaEtiqueta". Como su nombre indica, esta clase modela las líneas asociadas a una determinada etiqueta. Sus funciones son muy parecidas a las de la clase "Linea" vistas en el apartado 4.5.1, pero todo está readaptado para que cada línea conste de varios trazos. Solo almacena dos atributos, xPoints e yPoints, que son dos arrays con todas las coordenadas x e y de los trazos.

La función toStringJ() se encarga de transformar el objeto en un JSON. Veremos la utilidad de esto en la clase Etiqueta.

```
class LineaEtiqueta {
    constructor(xinit, yinit) {
        this.xPoints = [];
        if(xinit != null){
            this.xPoints.push(xinit);
        }
        this.yPoints = [];
        if(yinit != null) this.yPoints.push(yinit);
    }

    printLinea() {...}

    toStringJ() {...}

    addPunto(x, y) {...}

    sqr(x) {...}
    dist2(v, w) {...}
    distToSegmentSquared(p, v, w) {...}
    distToSegment(p, v, w) {...}

    pointIn(px, py) {...}
}
```

Figura 31. Clase LineaEtiqueta del ítem Etiquetado de imagen

La siguiente clase que se definió es *Etiqueta*. Como su nombre indica, la clase contiene todos los métodos y atributos necesarios para poder trabajar con las etiquetas.

Concretamente, los atributos que posee son:

- **tagName:** Contiene la respuesta dada a la etiqueta o el ejemplo que ha puesto el profesor.
- **linea:** Contiene un objeto LineaEtiquetado.

- **xTPos:** almacena la posición en el eje x de la esquinan superior izquierda de la etiqueta.
- **yTPos:** almacena la posición en el eje y de la esquina superior izquierda de la etiqueta
- **height:** Contiene la altura del campo de texto.
- **width:** Contiene el ancho del campo de texto.
- **et:** Contiene una referencia al campo de texto real.

Respecto de las funciones de la clase setTagName, setPos y setLine son métodos para modificar los atributos de la clase de una manera más segura, comprobando que no se pasan por parámetros valores incorrectos, como una posición negativa o un tagName undefined.

```
class Etiqueta {
    constructor(tagName, linea) {
        if (tagName != null) this.tagName = tagName;
        else this.tagName = '';

        if (linea != null) this.linea = linea;
        else this.linea = null;

        this.xTPos = null;
        this.yTPos = null;
        this.height = null;
        this.width = null;
        this.et = null;
    }

    setTagName(tagName) {...}
    setPos(x, y) {...}
    setLine(linea) {...}

    toJsonText() {...}

}
```

Figura 32. Clase Etiqueta del ítem Etiquetado de Imagen

La función toJsonText(), transforma toda la instancia del objeto en un texto estructurado que en el futuro se pueda transformar en un objeto JSON.

```
toJsonText() {
    var l = this.linea == null ? null : this.linea.toStringJ();
    var t = '{ "tagName":"' + this.tagName
            + '" , "linea":' + l
            + ' , "xTPos":' + this.xTPos
            + ' , "yTPos":' + this.yTPos
            + ' , "height": "'+this.height
            +'" , "width": "' + this.width + '" }';

    return t;
}
```

Figura 33. Función toJsonText() de la clase Etiqueta en el ítem Etiquetado Imagen

Esta función se complemente con la función soluciónToString() que hay en el fichero etiquetaProfesor.js. El código de dicho fichero se encarga de definir todas las funciones necesarias para hacer funcionar la vista del profesor del presente ítem.

Cuando se cierra la ventana de edición, se llama a la función solucionToString(), que crea un nuevo texto estructurado que contiene una lista de las etiquetas definidas por el profesor ya transformadas a un texto estructurado. La cadena de caracteres resultante de esta operación será la que se escriba en el campo *posición de las respuestas,* ya explicado en el apartado 5.1.

Dicha cadena, además, se le pasará como parámetro a la vista del alumno para poder colocar las líneas y los campos de texto en las posiciones elegidas por el profesor.

El funcionamiento interno de este ítem guarda varias similitudes con respecto al de Emparejamiento. Por ejemplo, gran parte del código de pintar líneas está muy inspirado en el código ya explicado en el aparatado 4.5, en el que veíamos 4 funciones distintas sobre el canvas, pero readaptado para poder dibujar líneas que tienen varios segmentos, que empiezan en mitad de la imagen y que acaban en una etiqueta.

```
canvas.addEventListener('mousemove', (event) => move(event));
canvas.addEventListener('mouseup', (event) => acabar(event));
canvas.addEventListener('mousedown', (event) => inicioLinea(event));
canvas.addEventListener('dblclick', (event) => eliminarLinea(event));
```

Figura 34. Funciones definidas sobre el canvas

Antes de pasar a explicar estas funciones, también cabe mencionar las funciones definidas sobre los campos de texto o etiquetas. Dichas funciones servirán para mover la etiqueta de sitio. Tanto las funciones mencionadas sobre canvas, como las que vamos a mencionar ahora sobre los inputs, solo están presentes en la vista profesor. En el resto de las vistas, el canvas no tendrá ninguna función y las etiquetas solo tendrán un evento por reescritura, en el que se modificará en la clase Etiqueta el atributo tagName.

```
//Creamos la función para poder mover las etiquetas de sitio
e.addEventListener('mousedown', (event) => inicio(event, e));
e.addEventListener('mousemove', (event) => move(event));
e.addEventListener('mouseup', (event) => acabar(event, e));
```

Figura 35. Eventos de las etiquetas en Vista Profesor para el ítem Etiquetado Imagen

Como podemos ver, tanto las etiquetas como el canvas tienen dos funciones idénticas: move y acabar. Esto se debe a que, para mover las etiquetas, si uno mueve el ratón muy rápido puede sacar el ratón de la etiqueta, lo que provocaría que el evento de mousemove no salte. Por tanto, si la misma función se le coloca a canvas, podemos hacer que haya un movimiento más fluido y menos limitado. La función move, en concreto, sirve tanto para mover una etiqueta de sitio como para pintar una línea. Se puede poner todo en la misma función sin que haya problema gracias a que no se puede pintar una línea y mover una etiqueta a la vez, por lo que usar una variable de control es medida suficiente para que el algoritmo identifique qué es lo que debe o no hacer.

```
function move(event) {
    event.preventDefault();
    if (moviendo) {...}
    else if (printingLine) {...}
}
```

Figura 36. Función move de la Vista Profesor en el ítem Etiquetado Imagen

La necesidad de tener dos variables diferenciadas hace que utilicemos también dos funciones distintas para inicializar el movimiento de la etiqueta y para pintar una línea. Además, cada una de las funciones de inicialización deberán definir otros parámetros para que todo el funcionamiento del ejercicio sea posible. Como, por ejemplo, en la función inicio(event), que es la utilizada para inicializar el movimiento de la etiqueta, se debe inicializar una variable auxiliar que tenga una referencia del elemento que estamos moviendo, por si se diera el caso de que el evento saltase en el canvas supiéramos sobre qué etiqueta debemos aplicar las operaciones oportunas.

En el caso de la función inicioLinea(event) se inicializa un objeto LineaEtiqueta que irá conteniendo toda la información de los trazos que vayamos dibujando en el canvas.

La otra función en común, *acabar*, para el caso de estar pintando una línea puede, o bien iniciar un nuevo segmento o bien acabar el dibujado del último trazo si el moseup se ha producido sobre una etiqueta. En el caso de que estemos moviendo una etiqueta, lo que se hará es almacenar la última posición en las variables xTPos e yTPos y dar valor false a la variable auxiliar "*moviendo*".

Por último, el doble clic sobre el canvas llama a una función que elimina la línea, en su caso, que se está dibujando o la línea sobre la que se ha hecho el doble clic. Al

igual que en el ítem de Emparejamiento, con que la distancia del doble clic a alguno de los segmentos de la línea sea 10 pixeles o menos, contará como que se ha producido sobre ella.

3.3.5 Solución Técnica: Vista Resolver y Vista Solución

Para representar la vista resolver y la vista solución, el archivo *etiquetaAlumno.js* contiene dos funciones:

```
function resolverEtiquetado(respuesta, correccion, name) {...}

function solucionEtiquetado(respuestaCorrecta, name) {...}
```

Figura 37. Funciones resolverEtiquetado y solucionEtiquetado del ítem Etiquetado Imagen

Los parámetros son similares a los ya descritos en el aparatados 4.5.2, con la puntualización de que, por las características de este ítem, las respuestas no se desordenan. Por tanto:

- **Respuesta:** Contiene los nombres que ha dado el alumno a las etiquetas en forma de vector de cadenas de caracteres.
- **Corrección:** Es un vector con valores booleanos que indican si la respuesta del alumno es correcta o no.
- **Name:** Es una variable que nos ayuda a identificar cuál es el ejercicio sobre el que hay que presentar la vista.
- **RespuestaCorrecta:** Es un vector de cadenas de caracteres con los nombres correctos de las etiquetas.

Para poder intercalar las respectivas vistas se utilizará una función definida dentro de resolverEtiquetado y que se ejecutará cuando se haga clic sobre el ejercicio. Esta función vaciará las etiquetas y escribirá la imagen y el nombre correspondiente.

3.3.6 Aspectos de usabilidad del ítem

Por los motivos expuestos en el subapartado homónimo a este del aparatado *3.2 Ítem de Emparejamiento*, se ha intentado tomar decisiones a la hora de elaborar este ítem que favorezcan la interactividad del usuario con la aplicación. En este caso, el usuario en el que nos hemos centrado ha sido el docente, permitiéndole mover de sitio las etiquetas que, en un primer momento, se colocaban rígidamente en el ejercicio.

El docente es el principal objetivo desde un punto de vista del atributo de usabilidad satisfacción. Es él el que decide qué ejercicios se van a realizar en los test. Si

para el docente, crear un nuevo ejercicio de un tipo determinado es demasiado complejo o molesto, lo más seguro es que opte por otros tipos de ítems, aunque no logren cubrir del todo la necesidad del usuario.

Además de la posibilidad de colocar al gusto las etiquetas en el ejercicio, también se incluye un nuevo texto de ayuda específico para el alumno. Debido a la simplicidad que presenta resolver la vista de alumno, la ayuda en este caso es bastante corta.

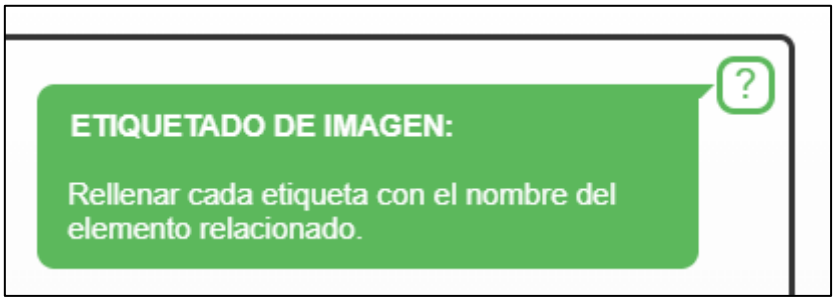

Figura 38. Ayuda para los ítems de Etiquetado de Imagen

3.4 Ítem de Selección de Área

El tercer ítem, Selección de área, tiene un objetivo similar al anterior pero su funcionamiento es a la inversa. En lugar de dar nombre a un elemento de la imagen, el alumno recibe una o más etiquetas y deberá seleccionar el elemento de la imagen con el que se corresponden dichos enunciados.

3.4.1 Solución Vista Profesor

Como se ha venido haciendo en los dos ítems anteriores, lo primero que debe realizar el docente es rellenar un formulario donde se incluirá toda la información relativa al ejercicio.

Figura 39. Formulario Profesor en ítem Selección de Área

En el campo *Imagen de fondo* el docente indica cuál es la imagen sobre la que se van a seleccionar las áreas.

En el campo *Posición relativa de la imagen*, se indica dónde se desea que se muestre la imagen dentro del contexto del ejercicio. Es el mismo campo que ya se mostró en el ítem anterior.

Los campos *tamaño lienzo* y *tamaño de la imagen* indican el tamaño en alto por ancho que deben tener ambos elementos cuando se muestren al alumno.

El campo *tamaño de texto* nos sirve, además de para el tamaño que desea el profesor que tenga la fuente, para calcular el tamaño de las etiquetas y que queden visualmente agradables.

El *color de la línea* será el color con el que se dibujarán tanto las líneas que definirán el área cuando se pulse el botón editar como para las líneas que relacionarán cada etiqueta con un área.

Por último y, antes de explicar el botón editar, tenemos los campos de *Textos*. En ellos el docente introducirá las etiquetas que desea que el alumno coloque en la imagen. Que el docente ponga una o más de una condicionará cómo funcionará el ejercicio.

Cuando tenemos varias etiquetas y el docente pusa el botón de editar se abre una ventana dentro de la página con un aspecto similar a este:

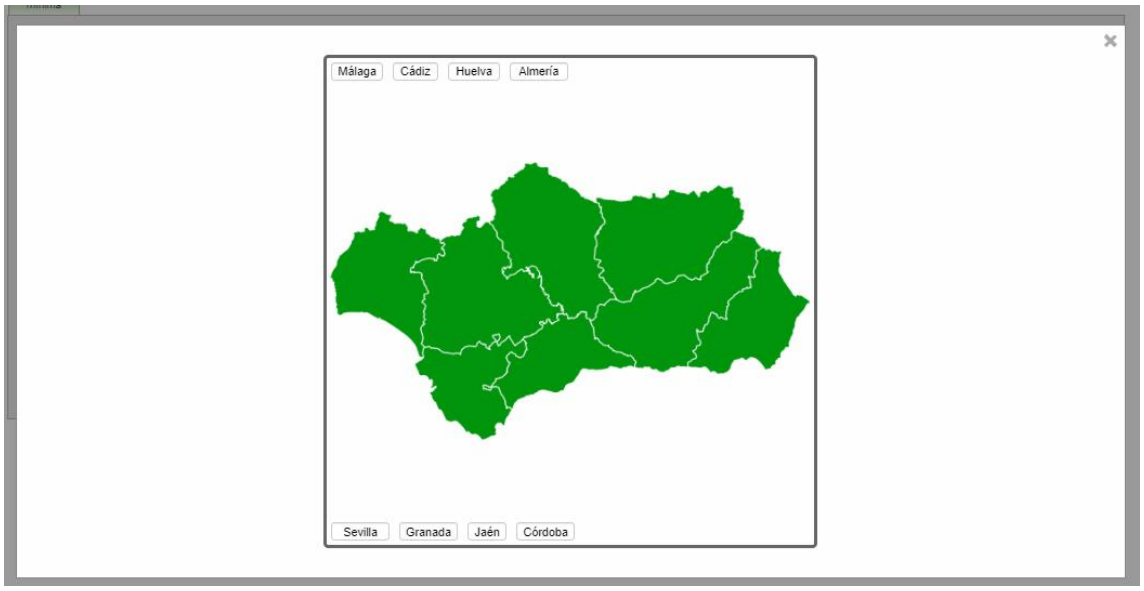

Figura 40. Vista Editar del Profesor en el ítem Selección de Área

El docente, a mano alzada y mediante clics, deberá trazar las áreas correspondientes, en este caso las áreas corresponderán a cada provincia. El algoritmo solo dibujará líneas rectas cuando interprete que se está dibujando una línea que relaciona un área con una etiqueta, esto es, cuando el clic primero para iniciar el dibujo de una línea se realice dentro de una de las áreas ya dibujadas en la imagen.

Además de dibujar las líneas, el docente podrá mover a su interés las etiquetas para que se más cómodo realizar las relaciones y para que resulte más agradable a la vista y más claro, sobre todo a la hora en que los alumnos vean la corrección de sus respuestas.

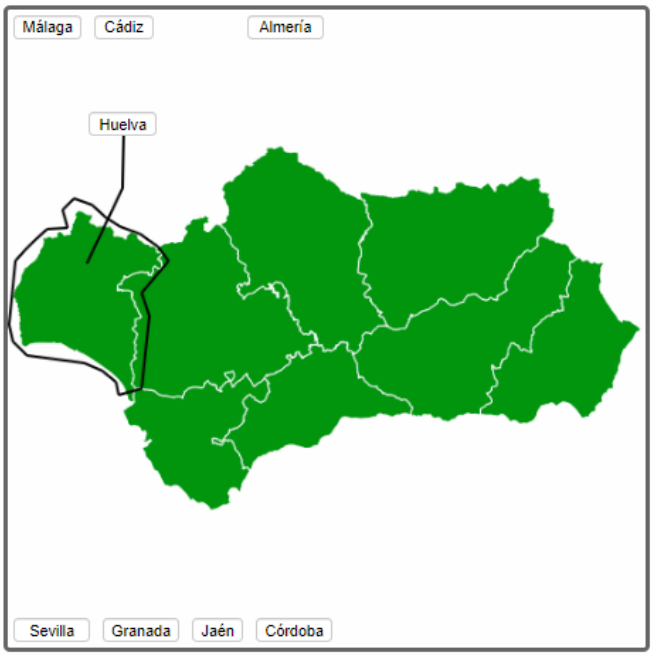

Figura 41. Dibujado de área y relación de esta en el ítem selección de área

Para el caso de una sola etiqueta el docente deberá simplemente dibujar el área del mismo modo que antes. La peculiaridad es que no aparecerá ninguna etiqueta, ni habrá que relacionar el área con dicha etiqueta, el algoritmo directamente almacena que esa es el área a tener en cuenta en la corrección.

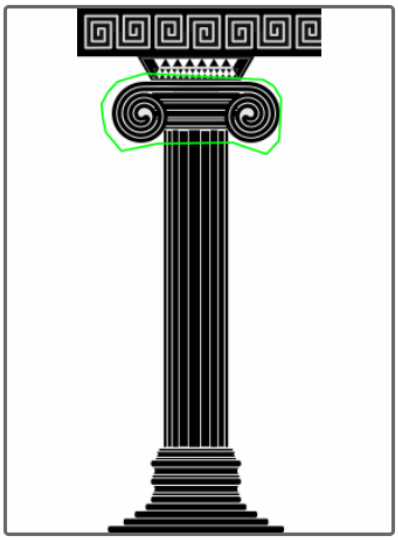

Figura 42. Ejercicio de Selección de área con una sola etiqueta en vista Profesor

3.4.2 Solución Vista Alumno

En la vista del alumno, al igual que en la del profesor, tenemos dos casos distintos. El primero es aquel en el que hay varias etiquetas. El funcionamiento aquí es igual que en el caso del profesor, con la diferencia de que el alumno no tiene que dibujar áreas, tan solo dibujar las relaciones entre la imagen y las etiquetas y, si así lo desea, cambiar la posición de dichas etiquetas.

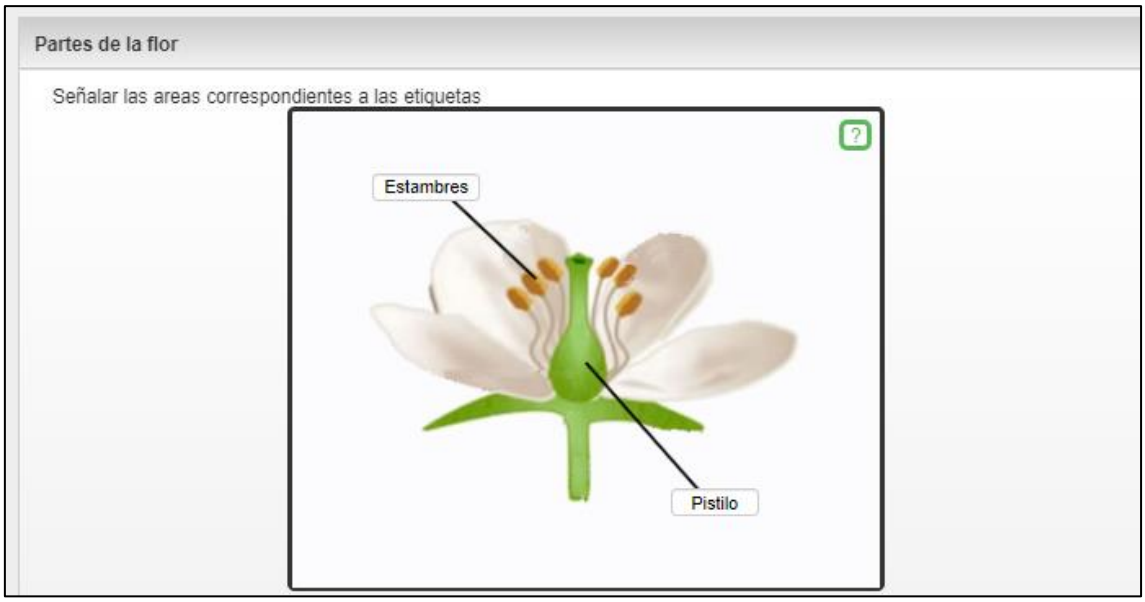

Figura 43. Ejercicio de Selección de Área resuelto por alumno

Por otro lado, cuando solo haya una etiqueta se mostrará únicamente un cursor en el ejercicio. Dicho cursor se podrá mover dejando pulsado el botón izquierdo del ratón y arrastrando la imagen, o bien clicando sobre un punto de la imagen y automáticamente se colocará el cursor en la posición seleccionada.

Figura 44. Ejercicio de una sola etiqueta de Selección de Área resuelto por un alumno

3.4.3 Solución Vista Resolver y Solución

La vista resolver y solución también tendrán dos vistas diferenciadas. La vista de varias etiquetas es idéntica a la mostrada en el Etiquetado de Imagen. En la vista resolver, las etiquetas bien relacionadas tendrán dentro del campo de texto un aspa verde a la izquierda; las etiquetas mal relacionadas una equis roja; y las etiquetas no relacionadas un aspa roja. Del mismo modo, cuando se haga clic sobre el ejercicio, se

mostrará la vista solución. Para dicha vista, se mantendrán las respuestas dadas correctamente por el alumno y solo se moverán de sitio aquellas que no hayan sido respondidas o hayan sido respondidas equívocamente.

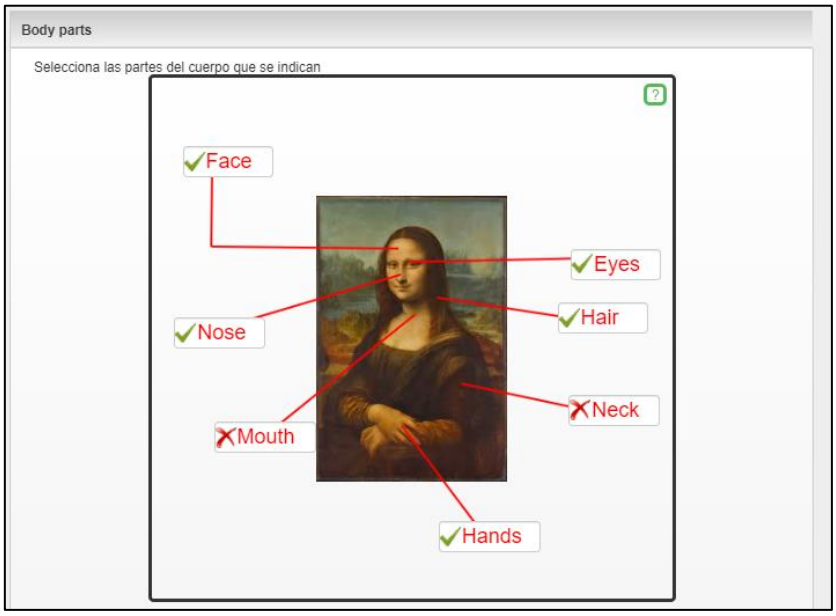

Figura 45. Vista Resolver con Múltiples etiquetas en el ítem Selección de Área

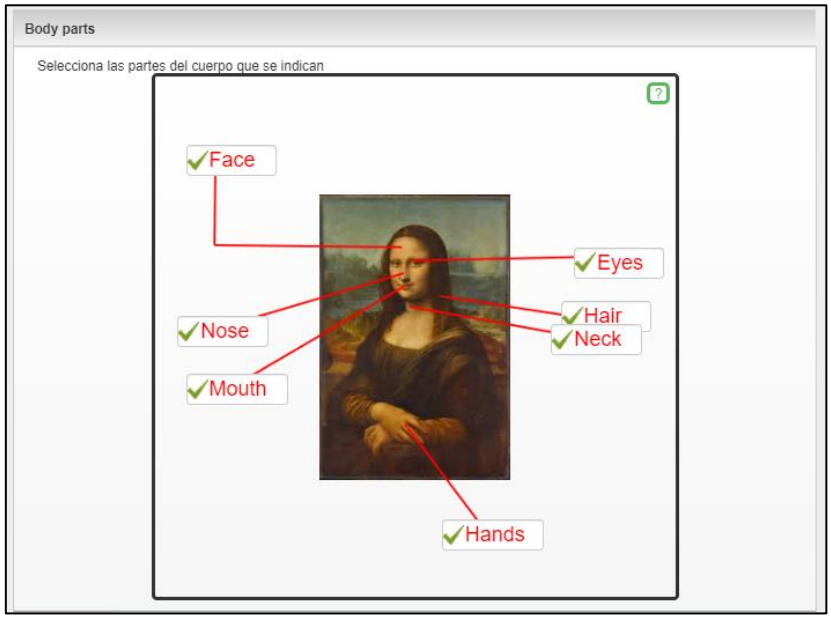

Figura 46. Vista Solución con Múltiples etiquetas en el ítem Selección de Área

En el caso de que solo hubiera una etiqueta, en la vista resolver se mostrará al alumno el cursor con un aspa verde en caso de que lo haya colocado bien y el cursor con una equis roja en caso de que lo haya colocado mal. En la vista solución, si la respuesta del alumno era correcta, se mantendrá igual, pero si es incorrecta se seleccionará un punto de dentro del área y allí se mostrará un cursor con un aspa verde.

Figura 47. Vista Resolver Selección de Área para una sola etiqueta

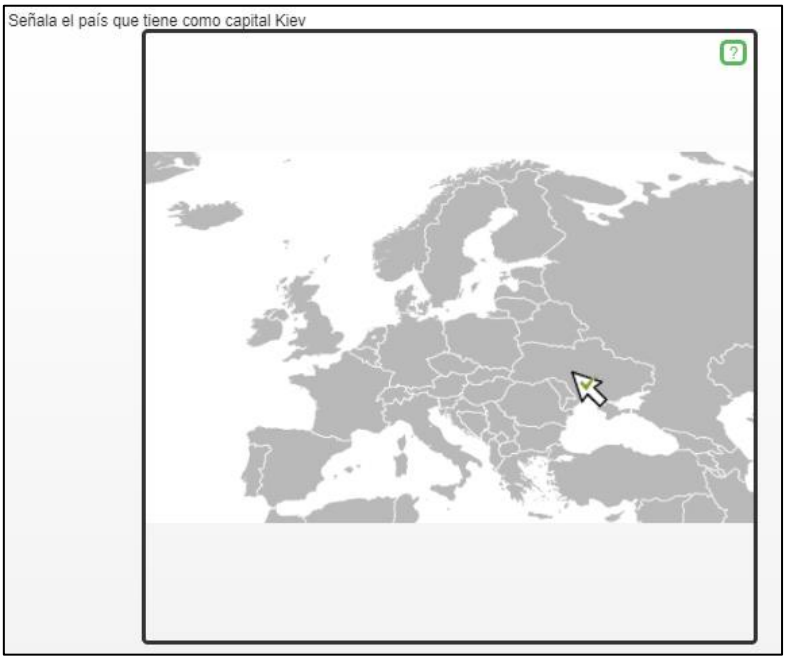

Figura 48. Vista Solución Selección de Área para una sola etiqueta

3.4.4 Solución Técnica: Vista Profesor y Vista Alumno

Para la elaboración del ítem Selección de Área, se han programado dos clases distintas. La primera es *EtiquetaArea*. Funciona exactamente igual que la clase *EtiquetaEtiquetado* explicada en el punto 5.5.1, pero con la diferencia de que

añadimos una nueva variable, denominada *area*, que contendrá un elemento de la clase *LineaArea*.

```
class EtiquetaArea {
    constructor(tagName, linea) {
        if (tagName != null) this.tagName = tagName;
        else this.tagName = '';

        if (linea != null) this.linea = linea;
        else this.linea = null;

        this.area = null;

        this.xTPos = null;
        this.yTPos = null;
        this.height = null;
        this.width = null;
        this.et = null;
    }
```

Figura 49. Clase EtiquetaArea

La otra clase desarrollada es *LineaArea*. Esta clase sirve tanto para almacenar la información de las relaciones entre un punto de la imagen y una etiqueta, como para definir un área. El código tiene bastantes cosas prestadas de la clase *LineaEtiqueta*, pero casi todo está cambiado de modo que se pueda trabajar indistintamente de un modo interno con las líneas y las áreas. Sus atributos son dos listas de coordenadas, una respecto del eje *y* y otra respecto del eje *x*.

```
class LineaArea {
    constructor(xinit, yinit) {
        this.xPoints = [];
        if (xinit != null) {
            this.xPoints.push(xinit);
        }
        this.yPoints = [];
        if (yinit != null) this.yPoints.push(yinit);
    }

    printLinea() {...}

    toStringJ() {...}

    addPunto(x, y) {...}

    sqr(x) {...}
    dist2(v, w) {...}
    distToSegmentSquared(p, v, w) {...}
    distToSegment(p, v, w) {...}

    pointIn(px, py) {...}

    changeLastSegment(px, py) {...}

    closeArea() {...}
    pointInArea(px, py) {...}
    distToFirstVertex(px, py) {...}
}
```

Figura 50. Clase LineaArea

El funcionamiento interno del presente ítem y el anterior son prácticamente idénticos. La principal diferencia - así como la mayor dificultad a la hora de programar- reside en la existencia de áreas. Se han tenido que realizar varias modificaciones en busca de la eficiencia de la representación de las mismas puesto que, al ser dibujadas a mano alzada, el número de coordenadas que tiene cada una podía llegar fácilmente a cientos para las más pequeñas y miles para un área grande. Con que hubiera dos o tres áreas a la vez, la velocidad de la ejecución del código de veía tremendamente resentida. Dos eran las causas de ello. La primera, solo existe un canvas, que debe redibujarse completamente durante todo procedo de creación de un área o de una relación entre un punto de la imagen y una etiqueta. Al tener que pintar miles de puntos por cada movimiento del ratón, el programa se ralentizaba. La segunda causa fue la función que reconocía si un punto está o no dentro del área. Dicha función será explicada más adelante en profundidad, pero a grosso modo, debía computar todos los lados y vértices que posee el polígono representante del área a seleccionar. Se tomaron pues, tres medidas diferentes para solucionar el problema:

- **Segundo canvas.** Se crea un nuevo canvas que se coloca detrás del canvas en el que se pinta, y una vez se ha dibujado el canvas principal y dicho dibujo se almacene, se pinta sobre el canvas del fondo, del que no se borrará nada a menos que se haga doble clic sobre alguna de las líneas correspondientes, y se vacía el canvas principal.

- **Modificación de la función addPunto.** La función addPunto se encarga simplemente de añadir una coordenada X a la estructura xPoints y una coordenada Y a la estructura yPoints. La particularidad de este código es que está perparado para que no se existan puntos "redundantes". Esto es, si se añade un nuevo punto a la estructura de modo que el último que había es un punto que se encuentra en la línea imaginaria que se trazaría entre el penúltimo y el nuevo, significa que ese punto no nos está dando ningún tipo de información y no es un vértice del polígono, sino un punto intermedio de uno de los lados. Por tanto, el algoritmo elimina dicho punto y añade el nuevo.

- **Simplificación de las áreas.** Aún con todas las medidas anteriores, siguen existiendo demasiados puntos que dificultan la labor de la función que discrimina si un punto está o no dentro de un área, además, las líneas no quedan del todo bien definidas al haberse dibujado a mano alzada. Por lo que, la función *closeArea*(), además de cerrar el área se encarga de simplificar los polígonos eliminando 9 de cada 10 puntos del mismo. La visualización de área queda así mucho más clara y más agradable a la vista del docente.

Cabe destacar en la elaboración de este ítem la implementación de la función *pointInArea(px, py)* de la clase *LineaArea.* Dicha función es llamada siempre que se va a proceder a dibujar una línea en la vista profesor -para comprobar si lo que se quiere dibujar es una nueva área o una relación de un área no asignada a una etiqueta y una etiqueta) y cuando se procede a la corrección del ejercicio.

Para calcular si un punto está dentro de un polígono, se traza una línea horizontal imaginaria en el plano y si tanto a izquierda como a derecha el número de intersecciones de esa línea con el polígono es impar, entonces el punto está dentro del polígono. Esta constituye una solución llamada "Ray-casting algorithm" dada al problema de geometría computacional conocido como Punto en Polígono (PIP por sus siglas en inglés). Dicho algoritmo ya se implementó en pseudocódigo en 1962 por Shimrat, M

La implementación aquí presentada a dicho algoritmo se ha hecho totalmente usando el conocimiento teórico aportado por dicho pseudocódigo. La función pointInArea, de manera simplificada hace lo siguiente:

- Crea dos variables a modo de contador de los lados con los que se cruza a izquierda y a derecha la línea horizontal imaginaria.

- Itera los lados comprobando primero si el punto está comprendido entre las alturas de sus dos vértices y, luego, si es así, comprueba

mediante la ecuación de la recta si se encuentra a izquierda o derecha del lado del polígono.

- Si las dos variables definidas al inicio del algoritmo son impares, el algoritmo devuelve true, en caso contrario devuelve false.

```
pointInArea(px, py) {
    var countLeft = 0;
    var countRight = 0;
    if (this.yPoints[0] == py) {
        if (this.xPoints[0] < px) {
            countRight++;
        } else if (this.xPoints[0] > px) {
            countLeft++;
        }
    }
    for (var i = 1; i < this.yPoints.length; i++) {
       var min = Math.min(this.yPoints[i - 1], this.yPoints[i]);
       var max = Math.max(this.yPoints[i - 1], this.yPoints[i]);
       if (py > min && py < max) {
            var A = { x: this.xPoints[i - 1],
                      y: this.yPoints[i - 1] };
            var B = { x: this.xPoints[i], y: this.yPoints[i] };
            var m = (B.y - A.y) / (B.x - A.x);
            var x = null;
            if (m == Infinity || m == -Infinity) {
                x = A.x;
            } else {
                var b = B.y - m * B.x;
                x = (py - b) / m;
            }

            if (x > px) {
                countLeft++;
            }
            else if (x < px) {
               countRight++;
            }
       }
       if (this.yPoints[i] == py) {
           if (this.xPoints[i] < px) {
                countRight++;
           } else if (this.xPoints[i] > px) {
                countLeft++;
            }
        }
    }

    return countLeft != 0
           && countLeft % 2 != 0
           && countRight != 0
           && countRight % 2 != 0;
}
```

Figura 51. Función PointInArea de la clase LineaArea

3.4.5 Solución Técnica: Vista Resolver y Vista Solución

Son tres las funciones relevantes para estas dos vistas

```
function evaluacionArea(name, profesorAns) {...}

function resolverArea(respuesta, correccion, name) {...}

function solucionArea(respuestaCorrecta, name) {...}
```

Figura 52. Funciones relevantes para las vistas Resolver y Solución del ítem Selección de Área

La primera función es *evaluaciónArea*. Esta función es la que da a SIETTE la solución del ejercicio. Por lo general, lo único que habría que pasarle al sistema sería, en este caso, una lista con cadenas de texto con formato JSON que representen las relaciones que han realizado los alumnos entre las etiquetas y la imagen, de modo que sea SIETTE el que, comparando la respuesta dada por el profesor y la respuesta dada por el alumno, corrija el ejercicio. Sin embargo, y debido a la complejidad del tratamiento de las áreas, dicha corrección se realiza en el lado del cliente directamente.

Cuando se llama a la función *evaluacionArea,* el parámetro *profesorAns* contiene la respuesta correcta dada por el profesor. El formato de dicha respuesta es una lista que contiene cadenas de texto con formato JSON. Cada cadena de texto contendrá toda la información necesaria para poder representar idénticamente la vista del profesor. En concreto, cada elemento de la lista se refiere a cada una de las etiquetas, y, por tanto, cada cadena contendrá un área y una línea asociadas a dicha etiqueta.

Así, la corrección de la respuesta del alumno se hará comprobando si, para cada etiqueta, la línea asociada dibujada por el alumno (en caso de que la hubiera) empieza en un punto que se encuentra dentro del área definida por el profesor para dicha etiqueta. En tal caso, al string que se le envía a SIETTE se le colocará el nombre de la etiqueta antes de las llaves que definen el objeto JSON. Esto se hace para facilitar la labor de SIETTE, que mediante una expresión regular como "NombreEtiqueta*", es capaz de saber si dicha etiqueta está correctamente relacionada o no.

Las funciones *resolverArea* y *SolucionArea*, simplemente muestran las vistas resolver y solución. Como ya se explicó en el punto 6.4, dicha representación variará de si hay una o más etiquetas. Además, el código está preparado para que se intercalen ambas vistas simplemente haciendo clic sobre el ejercicio.

3.4.6 Aspectos de usabilidad del ítem

El principal reto desde un punto de vista de usabilidad que presenta este ítem es la complejidad que puede tener aprender a manejarlo. Aunque se han establecido medidas para poder facilitar lo máximo posible la libertad del docente para crear los

ejercicios de este tipo, sí que se ha observado en las pruebas un poco de dificultad para cogerle el truco a su funcionamiento.

Para simplificar las acciones del docente se optó por que tanto las áreas como las líneas que relacionan un área con una etiqueta comenzaran su dibujo del mismo modo: mediante un simple clic en la imagen. El docente será capaz de reconocer enseguida si lo que se está dibujando es una línea o un área, puesto que el área no está formada durante el dibujado por líneas rectas.

Aún con todo, el tema de dibujar varias áreas y relacionarla con cada etiqueta se puede volver algo tedioso, lo cual puede causar un nivel de satisfacción más bajo del deseado. Con la intención de mitigar este efecto, se elabora una segunda manera de poder visualizar los ejercicios que solo contienen una etiqueta. Todo su funcionamiento ya es descrito en este apartado 3.4, pero cabe mencionar que presenta un comportamiento mucho más intuitivo, fácil y rápido que el de la versión con múltiples etiquetas, con la pega de que el ítem no dará demasiada información al docente sobre cómo de bien domina el alumno un determinado concepto.

Al tener dos tipos de vista, también se hubo que elaborar dos posibles textos de ayuda, como se ve en las siguientes dos figuras.

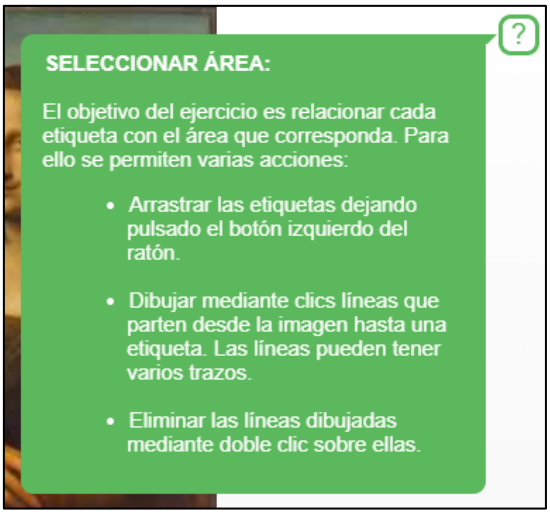

Figura 53. Ayuda para el caso de varias etiquetas en el ítem Selección de Área

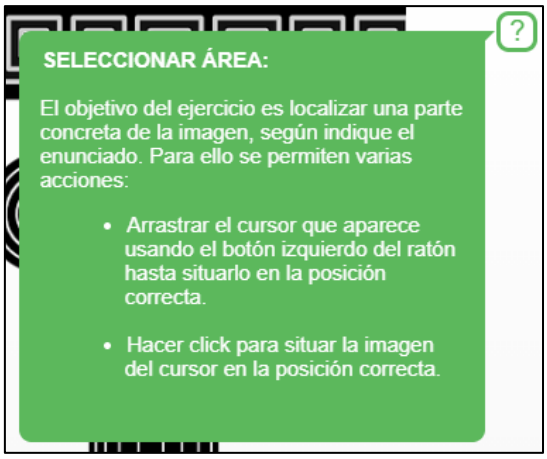

Figura 54. Ayuda para el caso de una sola etiqueta en el ítem Selección de Área

4 METODOLOGÍA DE TRABAJO

4.1 Metodología Scrum

Para la realización de este trabajo se ha seguido la metodología ágil SCRUM, en la que los ciclos tenían una duración de entre tres y cuatro semanas.

La metodología SCRUM es una metodología que se basa en la realización de *sprints* con una duración de un mes o menos en los que se plantean unos objetivos a cumplir. Acabado un sprint se realiza una reunión entre los miembros del equipo en los que compartir los avances realizados y se pasa a trabajar en nuevos objetivos marcados a partir de los ya conseguidos. Esta metodología es usada principalmente para el desarrollo software, aunque también se ha empleado en otros campos como investigación o marketing (Weiss & Kingsbury, 2006).

4.2 Ciclo 1. Primera Versión de Emparejamiento Alumno

En el primer ciclo se fijó como objetivo el desarrollo una versión temprana de la vista del alumno del ítem Emparejamiento. En dicha versión tan solo se permitían las relaciones uno a uno. Más concretamente los objetivos marcados eran:

- Permitir la existencia de columnas de imágenes y de texto.
- La manera de Emparejamiento sería que el alumno dibujase las relaciones trazando líneas usando el ratón.
- Permitir la eliminación de relaciones hechas por alumnos mediante un doble clic.
- Idear un modo de guardar en texto plano la solución aportada por el alumno al ejercicio.

De este ciclo se sacaron dos prototipos distintos. El segundo fue con el que se decidió seguir y es el que se ha explicado en el presente documento. El primero fue desechado debido a la falta de interactividad con el usuario. Su funcionamiento era simplemente clicar en un elemento de una columna y un elemento de otra y se dibujaba la línea por su cuenta. Se consideró que era preferible permitir que el usuario pudiera dibujar sobre el canvas con más libertad. Esta decisión fue crucial pues, aunque parezcan cosas bastantes similares el código cambiaba considerablemente si elegía una opción u otra.

Figura 55. Prototipo primero de Emparejamiento con el primer elemento de la columna izquierda seleccionado

4.3 Ciclo 2. Segunda Versión de Emparejamiento Alumno

Una vez finalizado el primer ciclo, en el segundo ciclo se plantea como mejora del proyecto que el ejercicio pudiera ajustarse dinámicamente al tamaño de la ventana del dispositivo. De este modo, el objetivo principal fue que las imágenes y el texto debían ajustar su tamaño según el espacio que hubiera en pantalla, creciendo (hasta cierto punto) cuando el espacio fuera mayor y disminuyendo (también hasta cierto punto) cundo este fuera menor.

Se definieron varias funciones para ello, llamadas resizeContent y resizeText para ello. La segunda función en un futuro ciclo se acabó dividiendo en resizeTextRight y resizeTextLeft para que el texto fuera ajustado dinámicamente por cada columna. Estas funciones eran invocadas cuando ocurría un evento "resize", esto es, cuando las dimensiones de la ventana eran modificadas.

4.4 Ciclo 3. Implementación de Emparejamiento Profesor

Ya implementada la vista del alumno se pasó a trabajar la vista del docente, cuyo funcionamiento era prácticamente idéntico a la del alumno. La diferencia y objetivo principal de este ciclo era que había que definir qué elementos del ítem debían ser variables modificables a la hora de crear un nuevo ejercicio y cuáles no. La lista detallada de cuáles fueron las variables seleccionadas se encuentra en el aparatado 4.2 de este documento. Otro objetivo del código era la adaptación de la vista del alumno, para integrarla dentro de la vista del profesor cuando este pulsara el botón de editar en el formulario, lo cual se logró realizar con una especie de ventana emergente dentro de la misma página, como se ve en la siguiente figura:

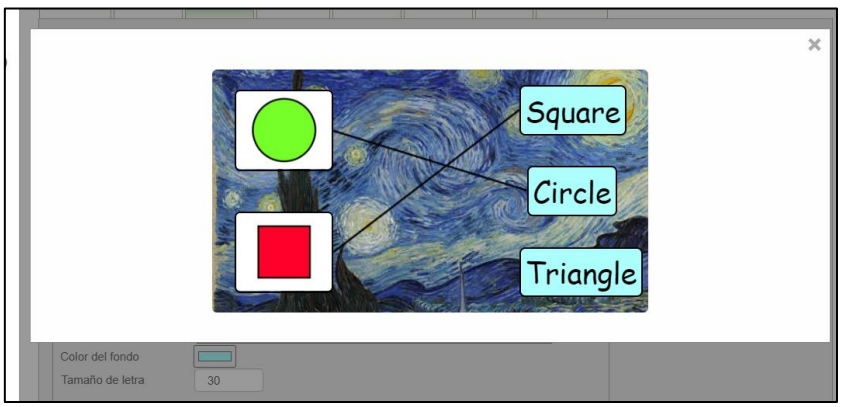

Figura 56. Modal para la vista de edición del profesor en el ítem Emparejamiento

4.5 Ciclo 4. Adaptación del Código para que haya Múltiples Preguntas

El ciclo cuarto se trabajó en dos aspectos principales de cara a su integración en SIETTE:

- Al establecer qué variables podían ser elegidas por el profesor, se planteó el primer objetivo de este ciclo: Modificar el código de la vista de alumno y profesor para que el comportamiento ante un ejercicio de relación múltiple y de relación uno a uno fuese distinto.
- El segundo objetivo fue permitir que en una misma página se mostraran varios ejercicios. Esto supuso un remodelado de gran parte del código, pues se tuvo que cambiar el modo en que los elementos del HTML se trabajaban y referenciaban. Así mismo se desechó parte del código debido a que dejó de ser necesario controlar ciertos factores como los id de ciertas etiquetas.

Durante este ciclo aparecieron varios problemas por culpa del scroll vertical de la página. Debido a un mal entendimiento de la documentación sobre los eventos de ratón de JavaScript el código empezó a complicarse mucho para el tratamiento de la excepción del scroll y cuando se solucionaba una vista, otra dejaba de funcionar. Se procedió a volver a leerse la documentación con todo sabido y una propiedad del evento parecía que se ajustaba mejor a las especificaciones del ítem. Se remodeló brevemente el código y se simplificó bastante y todo fue solucionado. En concreto, se leía event.clientY, que decía en qué posición de la pantalla (en el eje y) se encontraba el ratón en el momento del evento. Por tanto, en caso de que hubiera un scroll vertical, había que sumar a ese valor todos los scrolls que pudiera tener la página para obtener la posición exacta sobre el documento. La solución por la que se optó fue cambiar ese valor por event.offsetY, que devuelve la distancia en el eje y a la que se encuentra el ratón en el momento el evento respecto de la esquina superior izquierda del elemento sobre el que saltó el evento (el canvas en este caso).

En las siguientes dos figuras se puede apreciar la considerable diferencia entre la misma función antes y después de este cambio:

```
//Cuando empezamos a dibujar la linea
canvas.addEventListener('mousedown', function (event) {
    var top = 0, left = 0;
    var elem = canvas;
    do {
        top += elem.offsetTop || 0;
        left += elem.offsetLeft || 0;
        elem = elem.offsetParent;
    } while (elem);
    var db = null;
    if (isProfessor) db = parentTag.parentElement;
    else db = document.body;
    var x = event.clientX + db.scrollLeft - left;
    var st = isProfessor ? 0 : db.scrollTop;
    var y = event.clientY + st - top;
    if (isValid(x, y) && !acabado) {
        pintar = true;
        currentLine = new LineaEmparejamiento(x, y);
    }
});
```

Figura 57. Versión de la función para iniciar el pintado de línea en Emparejamiento con clientY

```
//Cuando empezamos a dibujar la linea
canvas.addEventListener('mousedown', function (event) {
    var x = event.offsetX;
    var y = event.offsetY;
    if (isValid(x, y) && !acabado) {
        pintar = true;
        currentLine = new LineaEmparejamiento(x, y);
    }
});
```

Figura 58. Versión de la función para iniciar el pintado de línea en Emparejamiento con offsetY

4.6 Ciclo 5. Primera Versión de Etiquetas

En ciclo quinto se comenzó con la elaboración de un segundo ítem. Dados los conocimientos ya adquiridos en los cuatro ciclos anteriores, la programación de la primera versión nuevo ítem se llevó a cabo teniendo en cuenta las especificaciones necesarias para una correcta integración en el sistema SIETTE. Así, los objetivos planteados para este ciclo eran:

- Elaborar una vista de profesor en la que se creen etiquetas mediante clics en la imagen a etiquetar.
- Elaborar una vista de profesor

- Desarrollar una sintaxis para poder representar en texto plano la solución o respuesta de un determinado ejercicio

Esta versión presentaba bastantes inconvenientes, siendo el más destacado la falta de interacción por parte del profesor para poder colocar las etiquetas donde quisiera, quedando una vista bastante rígida y a veces sucia (pues era posible encontrar una línea sobre otra). El código estaba preparado para que, mediante clics, el docente eligiera el punto de la imagen sobre la que iba una etiqueta, y se colocaba un campo de texto a izquierda o derecha de la imagen con una línea dibujada desde este input hasta el punto clicado.

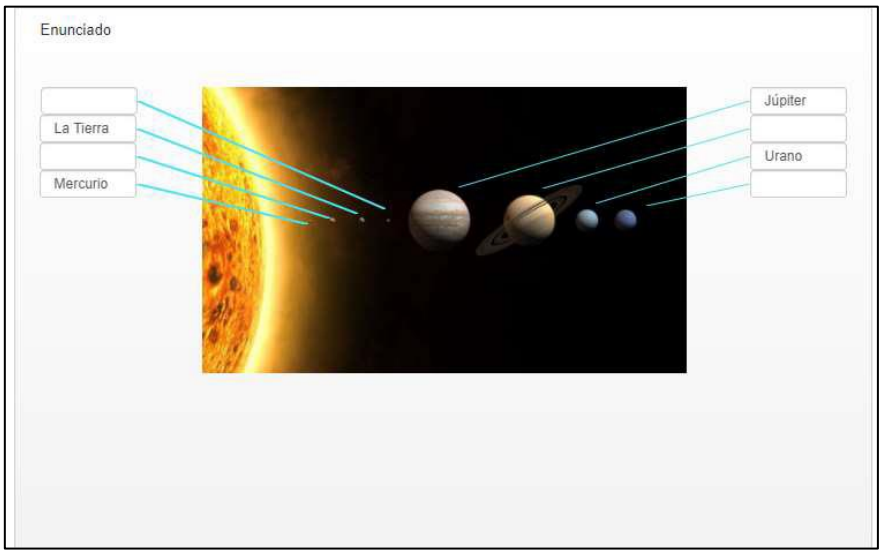

Figura 59. Versión primera del ítem Etiquetado

4.7 Ciclo 6. Segunda Versión de Etiquetado

En el segundo ciclo de Etiquetado se plantearon como metas:

- Definir en profundidad cuáles son las opciones del docente a la hora de crear un ejercicio.
- Permitir la movilidad libre de los campos de texto, así como dibujar líneas de varios trazos.
- Elaborar una vista de alumno funcional que devuelva, en forma de vector de cadena de caracteres, las respuestas dadas.

En este ciclo se trabajó sobre todo en la redefinición de clases y en programar eventos para el correcto funcionamiento de las dos vistas a desarrollar (alumno y profesor). A penas hubo incidencias en esto debido a toda la experiencia en canvas adquirida con el desarrollo del ítem Emparejamiento.

4.8 Ciclo 7. Vista Resolver y Vista Solución de Emparejamiento

Una vez implementadas las vistas de docente y alumno para Emparejamiento y Etiquetado se pasó a desarrollar dos vistas nuevas para el primer ítem. Estas vistas surgen de la necesidad de mostrar al alumno una corrección visual de la que pueda aprender y mejorar.

Aunque las vistas son independientes una de otra, son muy parecidas y el código empleado en una se reutilizó en otra. Además, ambas coexisten en el mismo espacio y se van turnando según las acciones del usuario.

El resultado de este trabajo fue conseguir unas vistas muy dinámicas, simples e informativas que permiten una experiencia de aprendizaje considerablemente interactiva, sobre todo cuando comparamos con la clásica evaluación numérica del medio.

4.9 Ciclo 8. Vista Resolver y Vista Solución de Etiquetado

Este ciclo se trabajó del mismo modo que el ciclo siete, pero sobre el ítem de Etiquetado.

Durante este ciclo, además, se estuvieron arreglando pequeños errores de todas las vistas ya integradas en SIETTE para dejar los ítems lo más perfectamente preparados para el siguiente y último ciclo.

4.10 Ciclo 9. Primera versión de Selección de Área

En este ciclo se inició el desarrollo del ítem Selección de Área. Puesto que este ítem tenía a nivel conceptual muchas similitudes de con el ítem Etiquetado de Imagen, no hubo mucho lugar a dudas respecto a cómo se iba a programar. La mayoría de los esfuerzos estuvieron concentrados en estudiar cómo hacer la comprobación de si un punto estaba o no dentro de un área.

Al final de este ciclo se entregó una versión completa del ítem con las cuatro vistas (profesor, alumno, resolver y solución) pero sin contar con una vista diferenciada para el caso de una sola etiqueta, puesto que fue un concepto que surgió después de esta entrega.

Por otro lado, haciendo pruebas nos dimos cuenta de que el rendimiento del ítem se resentía considerablemente a mayor número de áreas había en pantalla. A partir de tres áreas todo empezaba a ir muy lento. Además, el algoritmo de reconocimiento de si un punto está o no dentro de un área fallaba bastante. Por tanto, la siguiente entrega se caracterizaría por la intención de arreglar los problemas aquí mencionados y añadir unas nuevas vistas especiales para el caso de una sola etiqueta.

4.11 Ciclo 10. Segunda versión de Selección de Área e inclusión de ayudas

En este ciclo se desarrolló una serie de vistas para el caso de una sola etiqueta para el ítem de Selección de Área. Estas vistas se caracterizan por la eliminación de la representación visual de las etiquetas y por la inclusión de una imagen que funciona a modo de cursor. Allá a donde apunte el cursor, será donde al alumno ha colocado su respuesta. Por la parte del profesor, basta con dibujar una sola área sin tener que relacionarla a nada, puesto que, al haber una sola etiqueta, no hay necesidad de que dibuje ninguna relación.

También, se optimizó el código de modo que las áreas almacenaran menos coordenadas y dejando un canvas para las cosas fijas y otro para las cosas que se están dibujando. Desde ese arreglo, el código no ha vuelto a ir ralentizado en ninguna de las pruebas realizadas.

La parte que más costó realizar en este ciclo fue arreglar la función de pointInArea de la clase LineaArea. El código daba muchos problemas y cada vez fallaba más. Por lo que se decidió volver a implementarlo de nuevo, usando todo lo aprendido. El código reescrito parecía funcionar mejor que el anterior, pero seguía dando problemas. Tras una depuración exhaustiva del código, nos dimos cuenta de que el problema venía en que no se estaba manejando bien el caso de una línea completamente vertical. Cuando esto ocurría, la ecuación de la recta nos daba que la pendiente era infinita y, por tanto, los cálculos empezaban a darnos resultados no numéricos que no tenían sentido. La solución fue simple, reconocer que, en el caso de una línea vertical, si la coordenada en el eje y del punto a comprobar estaba comprendida entre las coordenadas en el eje y de los vértices que une el lado, haciendo una comprobación booleana de si alguna de las coordenadas en el eje x de uno de los vértices (ambas son iguales, pues es un alinea completamente vertical) es mayor, menor o igual a la del punto estudiado.

Con todo ello, el ítem de Selección de Área quedó terminado. Sin embargo, había que ocuparse de otra tarea. Los ejercicios, aun intentándose hacer lo más intuitivos posibles, necesitaban de una pequeña guía para el usuario de modo que los alumnos sean capaces de realizar sus ejercicios sin que tengan que perder demasiado tiempo en experimentar cómo funciona el ítem. Para ello, se ha incluido en todos los ítems la posibilidad de añadir una ayuda que se mostrará en la parte superior derecha del ejercicio. Si se pulsa el botón izquierdo del ratón sobre el icono que aparece, se muestra un mensaje de ayuda, tal y como se ve en la figura 60.

Esta ayuda se incorporó a todos los ítems aquí presentados y cuentan con su versión en español y en inglés.

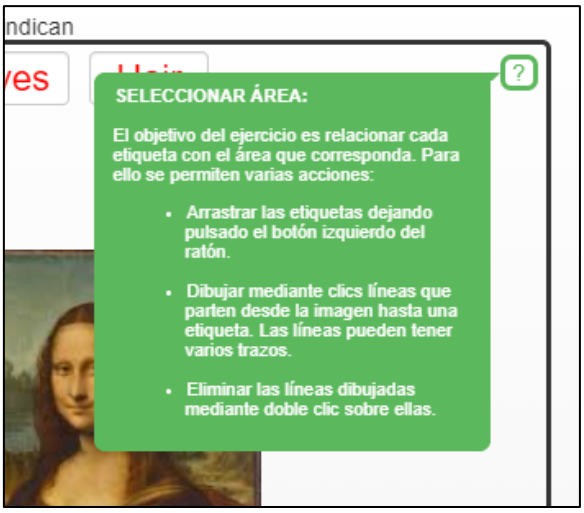

Figura 60. Ayuda para el ejercicio Selección de Área

4.12 Ciclo 11. Pruebas de los ítems y creación de contenido

Aunque las tareas de este ciclo ya se han llevado a cabo en menor grado durante todo el desarrollo del proyecto, era necesario dedicar un tiempo a probar todas las funcionalidades de los ítems en el entorno de SIETTE.

Durante este ciclo se dedicó a crear ejercicios y test con los ítems implementados. Salieron varios errores, siendo que la mayoría eran errores de carácter gráfico y pocos de carácter funcional.

Cabe destacar un problema que ocurría cuando se ponían imágenes en la columna de la derecha en Emparejamiento. En estos casos, las imágenes se guardaban como si fueran una columna por sí mismas, por lo que se provocaba un error tal que los elementos de la columna derecha aparecían al fondo de la página del ejercicio, donde no se veían sin scroll y donde, evidentemente, no se podía dibujar una línea que conectase una columna con la otra. El problema radicaba en la creación de estos elementos como imágenes y un par de errores derivados en la función mencionada anteriormente: resizeContent.

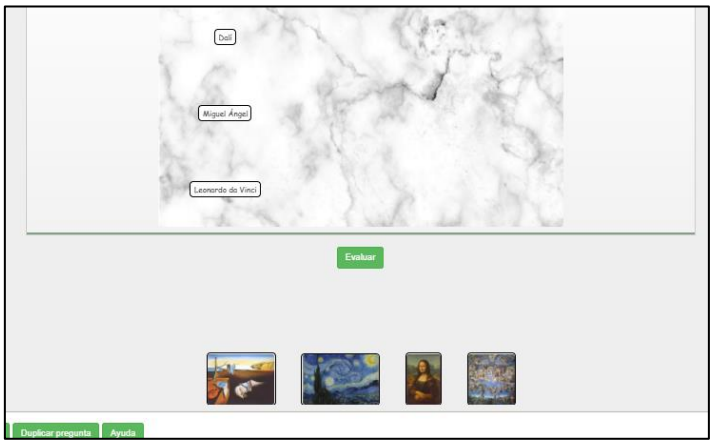

Figura 61. Error Gráfico en Emparejamiento

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

En este proyecto se han trabajado los conceptos de evaluación automatizada, ítem y usabilidad y se han puesto en común para crear una nueva biblioteca de ítems externos que se han integrado en el sistema SIETTE. La biblioteca está compuesta por tres tipos de ítems: Emparejamiento, selección de área y etiquetado de imagen.

En un primer momento, se fijó como objetivo de este TFG la implementación de cuatro ítems. Tres de ellos son los que se han desarrollado en este trabajo. Para ello se marcó la usabilidad como un requisito indispensable, de modo que cada uno de los programas debiera cumplir los atributos de eficacia, eficiencia en el uso y satisfacción. El cuarto ítem, Preguntas de Ordenación, quedó fuera de proyecto al estar ya incluido en el sistema SIETTE. Se optó, en cambio, por mejorar las interfaces y la funcionalidad de los tres ítems implementados.

Cada ítem se desarrolló en dos o más ciclos de modo que se presentaban al tutor los prototipos y sobre ellos se iban definiendo nuevos requisitos o correcciones de los que ya se habían implementado. En concreto, cada objetivo se logró mediante las siguientes fases.

1. Definición de nuevos requisitos y, en caso de ya contar con un prototipo, redefinir si hiciese falta los que ya existen en el programa. La definición de estos requisitos se hacía teniendo en cuenta que debía ser un sistema usable, por tanto, la mayor preocupación era decidir si los comandos propuestos eran o no intuitivos, si eran demasiados o muy pocos, si el programa hacía lo que debía, y si realmente era lo suficientemente satisfactorio solucionarlos. El ítem que más cambios sufrió fue el de Emparejamiento que, aun cumpliendo con sus objetivos, en un inicio presentaba una rigidez que lo hacía poco atractivo. La posibilidad de dibujar las líneas a mano alzada y, en definitiva, ser capaz de interactuar con el ítem de manera directa, permitió dar una sensación de satisfacción a la que no se podía llegar simplemente haciendo clics sobre unas cajas en pantalla. No es de extrañar pues, que la elaboración de los otros dos tipos de ítems siguiera líneas inspiradas en lo conseguido con el desarrollo de Emparejamiento, que fue el primero en ser trabajado.

2. Definidos los requisitos a cumplir y estudiadas las posibles líneas de diseño, se pasaba al desarrollo de un nuevo prototipo del ítem. Esta fase requería la comunicación activa entre el programador y el administrador de SIETTE. Pues tanto uno como otro, debían preparar código para la

adaptación en SIETTE del ítem. Eso es así, debido a que el ítem interactivo recibe los parámetros almacenados en el sistema y, del mismo modo, envía información a SIETTE como la respuesta propuesta por el docente o la solución que ha dado el alumno al ítem que se le ha presentado. Por tanto, el esfuerzo programático no ha sido solo de una parte, si no que el desarrollo de cada ítem requería modificaciones conjuntas entre las dos partes.

3. Estudio de la usabilidad del producto. Aunque no se probaba con usuarios reales entre cada ciclo, sí que se hacía una serie de pruebas creando ejercicios y probándolos. Esta fase era de gran preocupación como se ve en el punto 1, en el que las conclusiones aquí obtenidas se ven reflejadas en la redefinición de especificaciones y en la creación de nuevas.

Para cada ítem se implementaron dos vistas distintas. La primera es la vista del profesor. En ella el docente define cómo debe presentársele el ejercicio al alumno y cuál es la respuesta correcta. Ello lo hará mediante un formulario de SIETTE y un programa JavaScript que le muestra una previsualización del ítem y la posibilidad de resolverlo como si fuera un alumno. Al cerrar la previsualización, los datos introducidos por el profesor son almacenados en el sistema. Esta resolución se hará para mostrar la respuesta correcta. Una excepción a esto es Etiquetado de imagen. En dicho ítem, el profesor debía definir las respuestas correctas en el formulario siguiendo una determinada sintaxis y tan solo dibujar las relaciones entre la imagen que se etiqueta y cada etiqueta.

La segunda vista es la vista del alumno. En ella se les dan a los estudiantes las herramientas necesarias para resolver el ítem que se les presenta. La respuesta dada por el alumno es almacenada y pasada al sistema SIETTE, que se encargará de corregirla y devolver la corrección y demás información necesaria al código JavaScript, de modo que se presenten las vistas resolver y solución. Estas muestran, respectivamente, dónde ha acertado y dónde se ha equivocado el alumno, y la respuesta correcta aportada por el docente.

En definitiva, este proyecto ha logrado elaborar tres tipos de ítems, que, aunque planteados varias veces en el pasado, aún no se encontraban en el sistema SIETTE:

- **Emparejamiento.** Se presentan dos columnas de modo que, mediante líneas, se deben relacionar los elementos de la primera con los de la segunda. Este tipo de ejercicios permiten a los estudiantes inferir relaciones entre dos columnas, así como descartar información no relacionada.

- **Etiquetado de Imagen.** En los ítems de etiquetado de imagen, a los alumnos se les presenta una imagen con una serie de etiquetas vacías marcadas sobre la misma. El objetivo de estos ejercicios es que los

alumnos sean capaces de nombrar los elementos que se presentan en una imagen.

- **Selección de Área.** En este caso se presenta al alumno una imagen y una serie de etiquetas que tienen información sobre los elementos que aparecen en la imagen. El objetivo es que los estudiantes sean capaces de señalar en una imagen cuáles son los elementos sobre los que se les pone a prueba.

5.2 Líneas Futuras

Este trabajo se ha encargado principalmente de la elaboración de tres ítems para su incorporación en SIETTE. Esto no es más que una pequeña contribución a la basta base de ejercicios e ítems de los que ya dispone el sistema.

Además, y evidenciado por la pandemia que hoy día sufrimos, cada día es más necesaria la presencia de sistemas que permitan realizar un aprendizaje online de calidad; que proporcione a alumnos y docentes las herramientas necesarias para una educación eficiente y equiparable a la presencial.

Por otro lado, a los ítems aquí presentados también se le pueden añadir nuevas funcionalidades que amplíen la interacción del alumno y del docente con los ítems. Como, por ejemplo, la posibilidad de que el tutor pueda mover los elementos de las columnas en el ítem Emparejamiento a su antojo y poner el ejercicio en la disposición que desee.

Otra manera en el que los ítems podrán mejorarse será a través de las peticiones de los usuarios. Desde el punto de vista de la usabilidad, hasta ahora, solo se han hecho pruebas con personas relativas al proyecto, sin embargo, actualmente los ítems están pasando pruebas con posibles usuarios finales, cuyas valoraciones del proyecto se tendrán en cuenta. Docentes de Botánica en la Universidad de Málaga están haciendo pruebas con los tres ítems de modo que, muy probablemente, lo acaben incorporando a sus test de la plataforma SIETTE.

Por tanto, es de esperar, que se desarrollen próximamente nuevos ítems y, así mismo, los que ya estén implementados sean modificados para aumentar su interactividad con el usuario o, tras la realización de las pruebas con usuarios reales, para mejorar la experiencia.

BIBLIOGRAFÍA

Clark, R. C. y Mayer, R. E., (2011). *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning, Third Edition.* Pfeiffer.

Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-De-La-Cruz, J. L. & Ríos, A. (2004). SIETTE: A web based tool for adaptive testing. *International Journal of Artificial Intelligence in Education,* 14(1), 29-61.

Conejo, R., Guzmán, E. y Trella, M. (2016). The SIETTE automatic assessment environment. *International Journal of Artificial Intelligence in Education*, 26(1), 270-292.

Conejo, R. y Guzmán, E. *SIETTE: Sistema Inteligente de Evaluación mediante Test para TeleEducación.* XII cursos de verano de la UNED, España.

García, J. M., (1989), *Bases pedagógicas para la evaluación,* España, Editorial Síntesis, S.A.

Junta de Andalucía. Normativa usabilidad (consulta 2020) https://ws001.sspa.juntadeandalucia.es/unifica/web/gobernanza/normativa-usabilidad-web

Luján Mora, Sergio, (2001). *Programación en Internet: Clientes WEB.* Alicante, España: Editorial Club Universitario.

Nielson, J. (2012). Nielsen Norman Group. *Usability 101: Introduction to Usability*. https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Meyer, Eric A., (2006). *Cascading Style Sheets: The Definitive Guide, Fourth Edition.* O'Reilly Media, Inc.

Mozilla Development Network (consulta: 2020). *API Canvas.* https://developer.mozilla.org/es/docs/Web/HTML/Canvas

Mozilla Development Network (consulta: 2020). *Syntax - CSS: Cascading Style Sheets.* https://developer.mozilla.org/en-US/docs/Web/CSS/Syntax

Phillips, D., (2018). *Python 3 Object-Oriented Programming, Third Edition.* Packt Publishing.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. y Carey, T., (1994). *Human-Computer Interaction*. Addison Wesley.

Schwaber, K y Sutherland, J, (2017). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.*

Shimrat, M., *"Algorithm 112: Position of point relative to polygon" 1962, Communications of the ACM Volume 5 Issue 8*, Aug. 1962

Soler, J. (2010). Entorno virtual para el aprendizaje y la evaluación automática en bases de datos (Tesis doctoral). Universitat de Girona, Girona, España.

Ferré Grau, X., (2000). *Principios Básicos de Usabilidad para Ingenieros Software.* Universidad Politécnica de Madrid

Weiss, D. J. y Kingsbury, G. G., (1984). Application of computerized adaptive testing to educational problems*. Journal of Educational Measurement,* 21(4), 361-375.

Wykowski, T. y Wykowska, J. (2018*). Lessons learned: Using Scrum in non-technical teams.* Recuperado de: https://www.agilealliance.org/

Listado de Figuras

Figura 1. Ventana de inicio de sesión en el sistema SIETTE, 2020. 5

Figura 2. Arquitectura del sistema SIETTE. Ilustración de Conejo, R. y Guzmán, E. Recuperado de SIETTE: Sistema Inteligente de Evaluación mediante Test para TeleEducación. 5

Figura 3. Estructura de la base de conocimientos de SIETTE. Ilustración de Conejo, R. y Guzmán, E. 6

Figura 4. Ejemplo de ítem Ordenar textos 7

Figura 5. Ejemplo de ítem Música 8

Figura 6. Estructura de un fichero HTML. Ilustración de Luján, S. Recuperado de Programación en Internet: Clientes WEB. 13

Figura 7. Formulario del Profesor para crear ejercicios de Emparejamiento 18

Figura 8. Tipos de relaciones en Emparejamiento 19

Figura 9. Vista del docente de la edición de un ejercicio de Emparejamiento 20

Figura 10. Vista de Alumno de un ejercicio de Emparejamiento 21

Figura 11. Diagrama de casos de uso del alumno para los ítems de Emparejamiento 22

Figura 12. Ejemplo de respuesta de alumno en un ejercicio de Emparejamiento 23

Figura 13. Visión Resolver de Emparejamiento 23

Figura 14. Vista solución de un ejercicio de Emparejamiento 24

Figura 15. Alumno dibujando una línea en un ejercicio de Emparejamiento 25

Figura 16. Línea autoajustada en un ejercicio de Emparejamiento 25

Figura 17. Código de la clase LineaEmparejamiento para el ítem Emparejamiento 26

Figura 18. Ejemplo relación múltiple en Emparejamiento 27

Figura 19. Funciones de canvas para Emparejamiento 28

Figura 20. Función que imprime en el canvas todas las relaciones 29

Figura 21. Funciones codificar, resolver y solución de SIETTE para el ítem Emparejamiento 29

Figura 22. Funciones resolverEmparejamiento y soluciónEmparejamiento 30

Figura 23. Texto de ayuda para el ítem Emparejamiento 32

Figura 24. Formulario Profesor en ítem Etiquetado de Imagen 33

Figura 25. Vista de Profesor de un ejercicio de Etiquetado de Imagen con las etiquetas sin colocar 35

Figura 26. Trazado de línea sin terminar por docente de Etiquetado de Imagen 35

Figura 27. Diagrama de Casos de Uso de las acciones del docente en el editor de etiquetado de imágenes 36

Figura 28. Vista de Alumno de un ejercicio de Etiquetado de Imagen 37

Figura 29. Vista resolver en ítem Etiquetado de Imagen 38

Figura 30. Vista Solución en ítem Etiquetado Imagen 38

Figura 31. Clase LineaEtiqueta del ítem Etiquetado de imagen 39

Figura 32. Clase Etiqueta del ítem Etiquetado de Imagen 40
Figura 33. Función toJsonText() de la clase Etiqueta en el ítem Etiquetado Imagen 41
Figura 34. Funciones definidas sobre el canvas 41
Figura 35. Eventos de las etiquetas en Vista Profesor para el ítem Etiquetado Imagen 42
Figura 36. Función move de la Vista Profesor en el ítem Etiquetado Imagen 42
Figura 37. Funciones resolverEtiquetado y solucionEtiquetado del ítem Etiquetado Imagen 43
Figura 38. Ayuda para los ítems de Etiquetado de Imagen 44
Figura 39. Formulario Profesor en ítem Selección de Área 45
Figura 40. Vista Editar del Profesor en el ítem Selección de Área 46
Figura 41. Dibujado de área y relación de esta en el ítem selección de área 46
Figura 42. Ejercicio de Selección de área con una sola etiqueta en vista Profesor 47
Figura 43. Ejercicio de Selección de Área resuelto por alumno 48
Figura 44. Ejercicio de una sola etiqueta de Selección de Área resuelto por un alumno 48
Figura 45. Vista Resolver con Múltiples etiquetas en el ítem Selección de Área 49
Figura 46. Vista Solución con Múltiples etiquetas en el ítem Selección de Área 49
Figura 47. Vista Resolver Selección de Área para una sola etiqueta 50
Figura 48. Vista Solución Selección de Área para una sola etiqueta 50
Figura 49. Clase EtiquetaArea 51
Figura 50. Clase LineaArea 52
Figura 51. Función PointInArea de la clase LineaArea 54
Figura 52. Funciones relevantes para las vistas Resolver y Solución del ítem Selección de Área 55
Figura 53. Ayuda para el caso de varias etiquetas en el ítem Selección de Área 56
Figura 54. Ayuda para el caso de una sola etiqueta en el ítem Selección de Área 57
Figura 55. Prototipo primero de Emparejamiento con el primer elemento de la columna izquierda seleccionado 60
Figura 56. Modal para la vista de edición del profesor en el ítem Emparejamiento 60
Figura 57. Versión de la función para iniciar el pintado de línea en Emparejamiento 62
Figura 58. Versión de la función para iniciar el pintado de línea en Emparejamiento 62
Figura 59. Versión primera del ítem Etiquetado 63
Figura 60. Ayuda para el ejercicio Selección de Área 66
Figura 61. Error Gráfico en Emparejamiento 66

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga