ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Graduado en Ingeniería Informática

Implementación de herramientas software para el enriquecimiento cognitivo para la función ejecutiva de la atención.

Software development for cognitive enrichment of the attentional control skill in the context of executive functions

Realizado por
Luis Mayo Valbuena

Tutorizado por
Beatriz Barros Blanco

Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2017

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

Resumen

Las funciones ejecutivas son unas capacidades psicológicas de alta importancia para el desarrollo sano de los niños, ya que su mal desarrollo está relacionado con determinados problemas como el trastorno de déficit de atención.

En este TFG está relacionado las funciones ejecutivas, poniendo énfasis en la atención. A partir del entendimiento de como ésta funciona, se ha trabajado en el desarrollo juegos serios para entrenar la atención con tareas como "busca las diferencias", "encuentra elementos en la escena"". Estos juegos están, destinados a enriquecer las habilidades cognitivas de niños de infantil (de 3 a 5 años) y monitorizar dichas actividades. Los juegos se desarrollan en JavaScript y se integran en el sistema de evaluación automática Siette para evaluar y monitorizar el proceso de aprendizaje de los usuarios.

En paralelo se ha desarrollado un sistema de registro de eventos de modo que guardaremos en un *log* todas las pulsaciones realizadas por un niño para que psicólogos y educadores puedan analizar el comportamiento en profundidad. Este log se almacenará en un fichero alojado en un servidor diferente de siette, usando un software de servidor implementado para tal propósito en este TFG.

Todo el software se ha desarrollado siguiendo una metodología ágil y ha sido probado y evaluado con niños de educación infantil con tablets conectados a Internet.

Palabras clave:

Funciones ejecutivas, Siette, sistemas de evaluación automática, atención, juego diferencias, juego elementos en la escena , typescript, JavaScript, Servlet

Abstract

Executive functions (EFs) are a set of highly important psychological abilities that are related to children's development. EF's bad development is related to several issues including Attention deficit hyperactivity disorder (ADHD).

This project is related to emphasis on attention. From the understanding of how it works, we have worked on the development of serious games to train the attention with tasks such as "look for differences", "find elements in the scene". These games are intended to enrich the cognitive skills of children (3 to 5 years old) and monitor these activities. The games are developed in JavaScript and are integrated into the Siette automatic evaluation system to evaluate and monitor the learning process of users.

At the same time, an event logging system has been developed so that we will keep a log of all the clicks made by a child so that psychologists and educators can analyze the behavior in depth. This log will be stored in a file hosted on a server different from siette, using server-software implemented for that purpose in this project.

All the software has been developed following an agile methodology and has been tested and evaluated with early childhood education children with tablets connected to the Internet

Keywords:

Executive functions, Siette, automatic evaluation systems, attention, game differences, game elements in the scene, typescript, JavaScript, servlet

CONTENIDO

1 INTRODUCCIÓN

1.1 Las funciones ejecutivas

Las funciones ejecutivas (EF) son una serie de procesos necesarios para el control cognitivo del comportamiento y obtención de metas (Malenka et al. 2009). Este término incluye una amplia variedad de funciones, incluyendo el control de la memoria, de la atención y de la inhibición.

El control de la atención es una de las funciones ejecutivas. Esta función se define como la capacidad de un individuo para elegir a que prestar atención y que ignorar de forma voluntaria. En general, las funciones ejecutivas, se desarrollan lentamente a lo largo de la infancia, completándose en la edad adulta (Davidson et al. 2006). Una evolución insuficiente de este control está relacionada con algunos trastornos como el Déficit de Atención con Hiperactividad (ADHD), esquizofrenia y otros problemas de salud. Para entrenarlas, profesores y educadores realizan con los niños diferentes actividades en función de la edad tales como juegos de ordenación, de búsqueda de parecidos o de memoria, entre otras.

El proyecto Patio (patio.lcc.uma.es) es un proyecto que empezó en el año 2008, realizado por el departamento de Lenguajes y Ciencias de la computación (LCC) de la Universidad de Málaga (UMA) con el propósito de establecer herramientas que permitan realizar y monitorizar el aprendizaje y analizar cómo se pueden utilizar las tecnologías de la información para mejorar los entornos educativos. Como resultado del proyecto se implementó el *framework* APRENDO con el que se podían realizar actividades sencillas de aprendizaje vinculadas al currículum de educación infantil, entre ellas tareas para el enriquecimiento de la atención y el autocontrol. Este *framework* es una aplicación stand-alone para Windows. Actualmente en los colegios hay conexión total a internet y una aplicación en la web que funciona en tablets, no como APRENDO parece más adecuada.

La continuación de PATIO es el proyecto TECHCAT (http://techcat.iaia.lcc.uma.es/) (figura 1) es el proyecto en el que se enmarca este trabajo de fin de grado, este proyecto empezó en el año 2011. En este TFG se va a trabajar específicamente en el desarrollo de tareas para el enriquecimiento de la función ejecutiva para niños de 3 a 5 años. Se pretende desarrollar un sistema para la evaluación, monitorización y mejora de las EF en niños con dispositivos móviles tipo tablets. Se quiere que sea una aplicación disponible por internet para evitar problemas de instalación y configuración, así como para realizar la gestión de los usuarios.

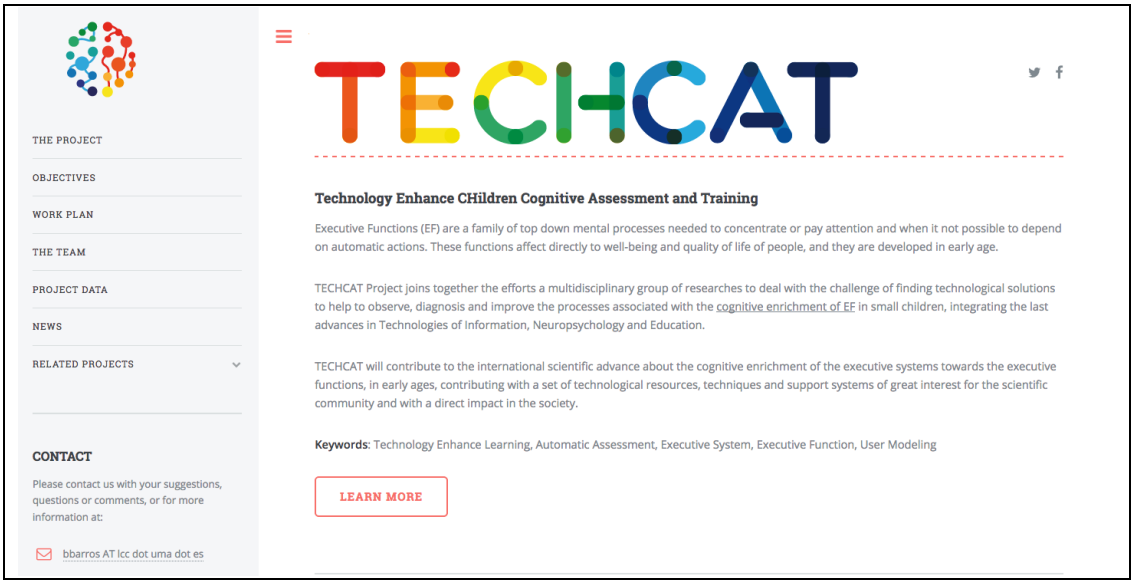

Figura 1. Página web del proyecto TECHAT en el que se enmarca este TFG

1.2 Objetivos del TFG

El objetivo del TFG es la implementación de programas que permitan el enriquecimiento cognitivo de la atención en niños pequeños, con dispositivos táctiles que puedan funcionar en Siette. De forma que puedan probarse con usuarios reales en colegios u otros entornos. En concreto se debe:

- Analizar tareas para enriquecimiento de la atención en forma de juegos: la interfaz, acciones correctas, acciones incorrectas, respuestas al usuario.
- Implementar en JavaScript las tareas en forma diferentes "juegos" sencillos
- Desarrollar un sistema de registro de eventos compartido por todos los juegos que comunique el cliente con un servidor, en el que se irán registrando las acciones relevantes que realiza el

usuario, a fin de que los investigadores sean capaces de analizarlas más en profundidad

1.3 Plan de Trabajo

En este proyecto, se trabajará con metodologías agiles tipo SCRUM o Kanban en el que se van definiendo prototipos que son probados y evaluados por el director del TFG y otros investigadores, para avanzar hacia una solución que satisfaga a los profesores y a los niños.

El Trabajo se ha realizado con las siguientes fases:

- Definición de una de aplicaciones de mejora de la atención a desarrollar: las diferencias
- Definición del registro de acciones y tipos de acción
- Desarrollo de uno o varios juegos.
- Desarrollo del mecanismo de registro del lado del servidor, e integración de los juegos con dicho mecanismo
- Desarrollo de otra aplicación: elementos en la escena
- Prueba y corrección con usuarios reales de los juegos desarrollados para lograr su mejora

1.4 Organización de la Memoria

Esta memoria está en varios capítulos. En el capítulo siguiente se describe el contexto del trabajo, el vocabulario utilizado y el tipo de aplicaciones que hay en torno al proyecto TECHCAT.

En el capítulo 3 se definirán y explicarán las tecnologías que se han utilizado en este proyecto de fin de grado para realizar la tarea.

En capítulo 4 se explicará como funcionan las aplicaciones desarrolladas desde el punto de vista funcional y del usuario.

En el capítulo 5 se explicará de forma técnica y a más bajo nivel como son los desarrollos cuyo funcionamiento se ha explicado en el apartado anterior.

En el capítulo 6 se hablará sobre el ciclo de desarrollo del TFG, indicando que aspectos se mejoraron de una versión a otra y que conclusiones se extrajeron en cada uno de los ciclos de desarrollo.

Esta memoria termina con un capítulo de conclusiones en las que ese resumen las principales tareas realizadas, así como una pequeña lista de trabajos futuros que se podrían hacer para seguir mejorando el proyecto TECHCAT.

2 CONTEXTO DEL TRABAJO

2.1 PLANTEAMIENTO GENERAL

En el proyecto TECHCAT se necesitan un conjunto de tareas que permitan realizar el entrenamiento de las funciones ejecutivas. En el proceso de entrenamiento los infantes juegan con una tableta. Al mismo tiempo se pretende poder representar la destreza que el niño tiene en la tarea que se está trabajando. Para ello, se quiere diseñar un sistema que recoja todas las acciones que realiza el usuario durante el juego. Ayudados por un sistema de evaluación automática, también se quiere representar el conocimiento del infante con un modelo de usuario.

El sistema de evaluación automática que se va a utilizar es Siette (www.siette.org) que gestiona los usuarios del sistema, mantiene un modelo de usuario para cada uno de ellos, así como los mecanismos para realizar las tareas en dispositivos móviles. Como contrapartida, las tareas tienen que implementarse conforme a unas restricciones y condiciones específicas.

2.2 APLICACIONES PARA EDUCACIÓN

El uso de Tecnologías con fines educativos es lo que se llaman sistemas TEL (*Technology ENhanced LEarning*) y dan lugar a un área de investigación específica que tiene diversas formas de implementarse, dependiendo del sujeto del proceso (niños, adultos, personas mayores), tipo de educación (presencial, a distancia, especialización, formación básica) o tipo de dispositivos (con móviles, en el aula…). Este TFG implementa juegos para educación, que se van a ejecutar en un sistema de evaluación inteligente implementado como un caso particular de Sistema Tutor Inteligente. En esta sección se van a describir algunos elementos relacionados el tipo de sistema en el que se utilizan los resultados del TFG.

2.2.1 Sistemas Tutores Inteligentes

Los sistemas tutores inteligentes (ITS) son unos sistemas de educación usando tecnologías de la información diseñados para que un alumno adquiera determinados conocimientos de un área a través de un entrenamiento personalizado y que se adapte a sus necesidades. Su estructura se divide en 4 componentes (Chrysafiadi, K., Virvou, M. 2013):

- Modelo de dialogo: Usando los *inputs* del alumno genera información en forma de texto, diagramas, etc. Para el profesor
- Modelo del Dominio: Incluye el conjunto de habilidades y conocimiento ideal que el alumno debe tener al final del aprendizaje, así como posibles errores y problemas que puedan surgir durante la tutorización
- Modelo del Alumno: Contiene mucha información sobre los conocimientos del alumno y otros factores afectivos o motivacionales que se pueden usar para determinar el estado del alumno
- Modelo de tutor: Este módulo mezcla la información que proviene del módulo de alumno y de dominio para generar la lista de acciones o pasos que se deben realizar para que el alumno complete la enseñanza con éxito.

2.2.2 Modelado de Usuario

Como ya se ha especificado anteriormente, un sistema tutor inteligente contiene a su vez un modelo de usuario, que incluye tanto su información estática de datos personales como información dinámica que se extrae de la interacción del alumno con el modelo. Esta información dinámica incluye el conocimiento del usuario, sus preferencias (hay usuarios que pueden aprender mejor con diferentes formas de presentar la información, por ejemplo), su estilo de aprendizaje y los errores que comete.

Esta información puede ser extraída de tareas que realiza el usuario o cuestionarios que rellene el alumno, así como de las acciones del mismo usuario. En este TFG esta información se va a obtener de los clics que cada niño realiza en la pantalla, los aciertos y los errores, así como medidas generales como el tiempo de duración. La construcción y gestión del modelo de usuario es tarea que compete a Siette y que está fuera de los objetivos de este trabajo, pero se han desarrollado las tareas para que Siette pueda procesar la información y actualizar el modelo de usuario del alumno.

2.2.3 El sistema SIETTE

Esta sección es un resumen del artículo [Conejo & Guzmán, 2019]

Siette es un sistema genérico para la evaluación que se compone de varios módulos, entre los que se incluyen:

- La base de conocimientos
- El generador de tests
- El módulo de edición
- El módulo de comprobación y activación
- El módulo de aprendizaje

La base de conocimientos del sistema Siette está a su vez compuesto por 3 tipos de objetos que a su vez se agrupan en asignaturas:

- Los conceptos, los temas en los que se descomponen las asignaturas
- Los tests, compuestos por ítems o cuestiones
- Los ítems, a su vez relacionados con conceptos.

Siette permite que estos últimos sean ítems externos que modelen tareas interactivas. Estos ítems externos son programas escritos en JavaScript y se pueden implementar como juegos sencillos. Tienen que seguir un conjunto de restricciones para poder instalarse en Siette, y poder recoger las interacciones de los alumnos como respuestas, evaluarlas y devolver un resultado al usuario. En el apartado 5.3 de esta memoria se describirá de forma detallada en los programas desarrollados como debe ser el código de cada juego o tarea para integrarse.

Respecto a los usuarios, Siette se presenta como una interfaz gráfica disponible a través de internet. Ofreciéndole a los profesores un portal web en el que editar los conceptos y los tests de sus asignaturas, y ofreciéndoles a los alumnos un aula virtual en el que poder realizar esos tests, necesitando sólo un navegador para realizarlos

Figura 2. Ventana de entrada del sistema SIETTE, en la versión 2019

2.3 DESARROLLO DE SW PARA NIÑOS

El desarrollo de software para niños, además de seguir los métodos y procedimientos habituales para el desarrollo de software, tiene unas características adecuadas para un público con unas preferencias y necesidades particulares. Esto hace que varios desarrolladoras, psicólogos y educadores hayan pensado en los elementos más adecuados para los niños, tanto para páginas web [Borse, Robles, Schwartz, 2002] como para aplicaciones y apps [Markopoulos, et al, 2008]:

- Pedir opinión u observar la reacción a niños durante el diseño: La forma en la que los niños responden a los estímulos es diferente de los adultos y por eso es necesario analizar sus reacciones
- Uso apropiado de las animaciones
- Diseños eficientes y orientados a un objetivo, no generalistas: En general páginas en las que se pueda usar el contenido lo antes posible, sin tener que esperar a una carga completa, o que al pulsar algo no tarde mucho en cargar.
- Incorporación de mecanismos de tipo social, como hacer un ranking con estrellas, o la asignación de un personaje especial, que lo distingue el resto
- Incorporar en el sistema elementos de sorpresa
- Considerar diferentes tipos de contexto de uso
- Presentar la información de diferentes formas

2.4 LAS FUNCIONES EJECUTIVAS

Bajo este término se engloban varias funciones y habilidades que pueden ser desarrolladas, entre las que podríamos incluir (lista extraída a partir de: Castillero Mimenza, O. 2019 y Cooper-Kahn, JM., Dietzel, L., 2008):

- Inhibición: Se trata de la habilidad que tenemos de detener nuestras propias acciones a tiempo y resistamos nuestros impulsos, contrarrestándose así con el concepto de impulsividad.
- Flexibilidad: La habilidad de cambiar de un contexto a otro, respondiendo a los cambios que se produzcan en el entorno para resolverlos.
- Control emocional: La capacidad de controlas las emociones y permitir resolver la situación de manera racional.
- Inicio y fin de tareas: esta función recoge la habilidad de iniciar una tarea cuando corresponda. Generando ideas y siguiendo estrategias de resolución de problemas. También incluye la capacidad de finalizar la tarea cuando así se requiera
- Memoria de trabajo: La posibilidad de almacenar información de forma temporal que vas a necesitar más adelante para resolver el problema. No se trata de memorizar a largo plazo si no de retener la información el tiempo suficiente para usarla más adelante
- Planificación: Es la capacidad de realizar planes para llegar a resolver la situación o problema siguiendo una serie de pasos.
- Organización: La habilidad de gestionar los recursos que se tienen de forma óptima para evitar perder tiempo al acceder a ellos y tenerlos disponibles para cuando sean necesarios, puede hacer referencia a la información que tenemos disponible o a recursos materiales.
- Auto monitorización: La capacidad de ser capaz de medir el rendimiento propio y contrastarlo con lo esperado.
- Razonamiento: Ser capaz de ver diferentes fuentes de información, conectarlas y extraer información lógica a través de dichas conexiones.
- Anticipación: La capacidad de predecir los resultados de una acción
- Toma de decisiones: La habilidad de, cuando se presentan muchas opciones diferentes, elegir la más adecuada para nuestro caso concreto.
- Fijación de metas: Dada una serie de objetivos, la habilidad de elegir cuantos recursos y energías dedicar a cada uno de ellos.
- Atención: Capacidad de enfocarse en una tarea y no distraerse. De esta se encarga este trabajo.
-

2.5 JUEGOS SERIOS

El concepto de "serious games", que aquí traduciremos por "juegos serios" fue acuñado por Clark C Abt en 1970 que lo definió como "juegos en el sentido de que esos juegos tienen juegos tienen un propósito educativo explícito y cuidadosamente pensado y no están pensados principalmente como diversión". Desde entonces, este concepto ha evolucionado, y ha tomado especial relevancia con la aparición de los tablets y dispositivos móviles [Laamarti, el. al 2014].

En este TFG, las tareas se van a modelar en forma de juegos, para que esta forma se más interesante y motivador para los niños. Se van a incluir gráficos adecuados para niños, sonidos para distinguir lo correcto y lo incorrecto, marcadores de puntos, indicador de inicio y fin de la tarea, y como parte final, una pantalla final con estrellas según el resultado obtenido.

3 TECNOLOGÍAS

3.1 Aplicaciones y Arquitecturas Web

Antes de hablar sobre las tecnologías que hemos utilizado durante el desarrollo necesitamos definir que es una aplicación web, una arquitectura, y explicar brevemente su funcionamiento

3.1.1 Arquitecturas cliente servidor

Una arquitectura cliente/servidor es una arquitectura en la que se ejecutan dos aplicaciones de forma diferenciada. Como su nombre indica incorpora un cliente y un servidor (Sun Microsystems, 2019):

1. El cliente que se ejecutaría en la máquina del usuario final. Este cliente se encargaría de realizar peticiones al servidor para obtener información o consumir un servicio.
2. El servidor, un programa que se ejecuta de forma permanente en una maquina remota con dirección conocida. Su objetivo es resolver las peticiones del cliente, registrando la información que el cliente publique y/o devolviéndole al cliente la información solicitada (Gough M. et al.)
3. Por último, tendríamos una API, la serie de normas y convenciones que deben seguir el servidor y el cliente al comunicarse.

Estas peticiones no siempre tienen porque ser pedir información, también incluye la publicación de datos por parte del cliente en dicho servidor.

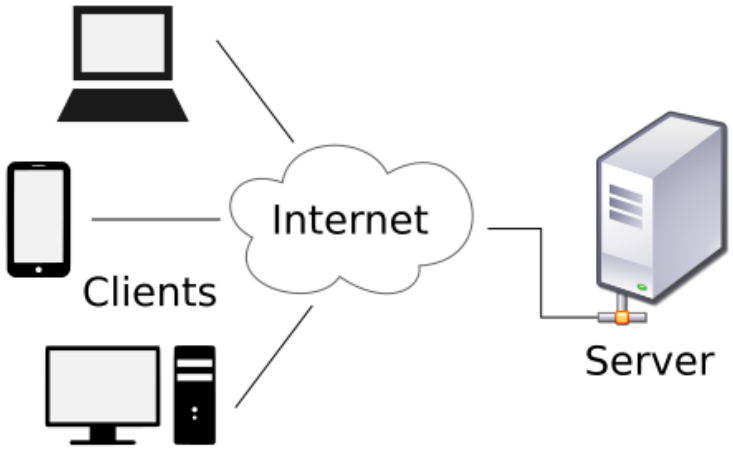

Figura 3. Esquema clásico arquitectura cliente/servidor. Ilustración por David Vignoni, licencia LGPL: https://commons.wikimedia.org/w/index.php?curid=15782858

3.1.2 Aplicación web

Una aplicación web es un sistema servidor/cliente en el que la aplicación no se instala, sino que se ejecuta a través de un navegador.

En este caso un navegador web se conecta a un servidor HTTP que le ofrece la aplicación web usando HTML (estructura de la app), CSS (parte visual de la app) y JavaScript (código funcional). Una vez descargados los recursos el navegador ejecutará la aplicación, que al igual que cualquier aplicación tradicional se conectará con el correspondiente servidor para realizar sus tareas.

Aunque a lo largo de la historia se han usado otras alternativas, como applets de Java para realizar esta función. No obstante, algunos problemas como su incapacidad para ejecutarse de forma multiplataforma, ya que sólo se ejecuta en sistemas con la máquina virtual de Java, lo que excluye dispositivos móviles, hacen que haya perdido relevancia con el tiempo.

Las aplicaciones web tienen una serie de ventajas sobre las aplicaciones tradicionales, entre ellas [Toal, R. 2019]:

- No requieren instalación por parte del cliente
- El despliegue de actualizaciones es mucho más sencillo ya que simplemente se despliega en el servidor web y los clientes se actualizarán la siguiente vez que carguen la aplicación

- El sistema es completamente multiplataforma, ya que para ejecutarse sólo necesita un navegador web compatible, lo que las hace ideales para crear aplicaciones que se ejecuten en dispositivos móviles

Estos dos últimos puntos son el principal motivo de la elección de una aplicación web para realizar este proyecto, el poder ejecutarla en tablets y su integración con Siette. JavaScript y TypeScript

JavaScript (JS) es un lenguaje de programación interpretado/compilado al vuelo (JIT) diseñado para su uso en navegadores web, tratándose, por tanto, de un lenguaje de programación del lado del cliente. No obstante, cada vez hay más esfuerzos para usarlo en el lado del servidor, con propuestas como NodeJS.

JavaScript es en realidad una implementación del estándar ECMAScript (ES), actualmente en su versión 6. La organización ECMA adoptó el lenguaje a partir del desarrollo que hizo Netscape (autor del JS original), antes de que esto ocurriera, cada fabricante tenía ligeras modificaciones del estándar, siendo la más notable JScript, utilizada por Internet Explorer (Microsoft)

ECMAScript ha ido creciendo poco a poco, los cambios más significativos produciéndose en la versión 5, donde se introdujeron los *getters* y los *setter*, pseudo-variables que se referencian en el código como si fuesen variables, pero en su lugar son funciones, y en la versión 6, con la inclusión de clases y módulos.

Dado que nuestro sistema pretendía ser una aplicación web y dado que JavaScript es el único lenguaje soportado por los navegadores de forma nativa. La necesidad de usarlo era evidente.

TypeScript (TS) es un superconjunto de JavaScript desarrollado por Microsoft desde 2012. Se trata de un lenguaje que añade azucarillos sintácticos a JavaScript. A diferencia de JS, los navegadores web no comprenden TS. Para solucionar esto TS incluye un compilador que traduce de TypeScript a JavaScript.

Entre los azucarillos sintácticos de los que hemos hablado se incluye el soporte para clases e interfaces, el cual se introdujo antes de que se introdujese en ECMAScript, ya la comprobación de errores de tipado, de modo que no puedes asignar valores del tipo X a una variable de tipo Y, cosa que en JavaScript si se puede.

Esta comprobación de errores, y permitir disponer de un sistema de clases sin tener que usar ES6 son los principales motivos que han dado lugar a la

elección de TypeScript como lenguaje principal para el desarrollo. Añadiendo la necesidad de procesar el código con el compilador de TypeScript antes de realizar la integración con Siette.

3.2 Java

Java es un lenguaje de programación orientado a objetos de propósito general creado por Sun Microsystems [Sun Microsystems, 1997], compañía más tarde comprada por Oracle. Se trata de un lenguaje de programación diseñado para ser multiplataforma. De modo que sólo sea necesario compilarlo una vez y ya se ejecute en todas las arquitecturas objetivo sin necesidad de más cambios, esto lo consigue realizando un compilado inicial a *byte-code*. Este *byte-code* será interpretado en las maquinas objetivo por la Máquina virtual de Java (JVM).

Java Enterprise Edition (JEE) es una versión de Java diseñado para su uso en servidores, en oposición a Java Standard Edition, realizado para su uso en sistemas de escritorio o la Java Micro Edition, para dispositivos móviles.

Esta versión incluye la especificación de los *servlet*. Clases Java que atienden peticiones HTTP y son capaces de atender a varios clientes de forma concurrente manteniendo una sesión con cada uno de ellos.

Se trata de unas clases muy sencillas que permiten crear una funcionalidad de forma muy rápida, en un *servlet* sólo defines las operaciones que se pueden realizar (*Get, Post…*) y una función en la que específicas la respuesta a devolver. Los *servlets,* además, también pueden guardar una sesión para que un mismo cliente realice varias operaciones.

Esto junto al hecho de que Siette ya está hecho en JEE convierte a los *Servlet* en la técnica ideal para desarrollar el servidor de logs.

3.3 JSON

JSON (*JavaScript Object Notation*) es un formato de intercambio de archivos ligero pensado para ser cómodo tanto para humanos como para maquinas. Se trata de un subconjunto de la notación de objetos de JavaScript [json.org, 2013]

Aunque fue diseñado a partir de JavaScript es un lenguaje de intercambio apto para todos los lenguajes.

JSON se basa en dos estructuras principales. Una colección de claves/valor en una estructura tipo diccionario. Nótese que en el estándar esta colección no se asume que estas claves lleven ningún orden, por lo tanto, tampoco se deberían repetir [Bray T. 2017]. Los valores pueden cadenas de texto, números, booleanos, listas u otros objetos, veremos un ejemplo de JSON en la figura 4

La otra estructura en la que se apoya JSON para su función son listas ordenadas, equivalente a los *arrays* de muchos lenguajes, incluyendo el propio JavaScript.

Debido a que JSON está integrado completamente en JavaScript, lenguaje en el que se realizan la mayoría de las aplicaciones web se ha ido convirtiendo en uno de los lenguajes claves para el intercambio de objetos.

```
{
        propiedad1: 5;
        propiedad2: 'hola';
        lista: [
                {
                        propiedad3: true;
                },
                {
                        propiedad3: false;
                }
        ]
}
```

Figura 4. Ejemplo JSON

3.4 AJAX

AJAX es el acrónimo de *Asynchronous JavaScript And XML*, el cual es una combinación de técnicas, siendo la más importante el objeto *XMLHttpRequest*

de JavaScript, que sirven para realizar peticiones asíncronas a un servidor siguiendo una estructura cliente-servidor [MDN, 2019] y, opcionalmente, modificar los datos que ve el usuario en base a la respuesta del servidor. Este método permite realizar peticiones sin bloquear la página ni tener que detener el flujo del programa.

A pesar de lo que su nombre parezca indicar, ni Ajax ni *XMLHttpRequest* están limitados a utilizar el formato de intercambio XML, si no que se puede usar, y cada vez es más común debido a su integración nativa con JavaScript, JSON como formato de intercambio.

XMLHttpRequest permite que definas una función JavaScript que se ejecutará cuando se produzca un cambio en el estado de la petición, incluyendo cuando la petición quede resuelta.

Para realizar una llamada usando *XMLHttpRequest* básica lo primero que hay que realizar es declarar un objeto del citado tipo, definir la función que manejará el cambio y abrir la conexión, esto se puede ver en la figura 5

```
const petition = new XMLHttpRequest();
petition.open('POST', 'test');
petition.addEventListener('readystatechange', (ev) => {
      if (petition.readyState === petition.DONE) {
            cookie = petition.getResponseHeader('session');
      }
});
petition.send(log + appendLogEnd());
```

Figura 5. Ejemplo envío de un POST con Ajax

Una vez la petición cambie de estado saltará la función que hemos añadido en *addEventListener*, en este caso se comprobará si la petición ha finalizado y en ese caso se almacenará un dato que se quiere conservar.

Juntándolo con las capacidades de manipulación del DOM que tiene JavaScript, esto se podría usar también para cambiar datos de la vista en

tiempo real sin necesidad de recargar la página, agilizando de forma notable la experiencia del usuario.

4 ANÁLISIS DEL PROBLEMA

En esta sección se hace un análisis del tipo de problema a resolver. Se ha elegido la tarea de las diferencias como "tarea piloto" para definir una metodología que permita cubrir un ciclo completo de desarrollo. Cuando el juego de las diferencias esté completada, se seguirá el mismo proceso para las otras tareas, cambiando aquellas funcionalidades que sean necesarias.

4.1 Tareas para el enriquecimiento de la atención

La atención es una función ejecutiva que tiene gran importancia y ha sido estudiada por numerosos expertos en la materia. Es una función ejecutiva relacionada con Control ejecutivo, que es una función superior, específicamente en lo que se refiere al control de la interferencia. Aquí, se consideran aspectos como la atención selectiva de elementos, la diferenciación o identificación de aspectos diferentes en situaciones similares.

Para mejorar la atención hay diversos juegos o tareas que se pueden realizar, entre los que se encuentran [Gratacós, M. 2019]:

- Encontrar un estímulo concreto entre varias posibilidades. Entre las que se encuentran ejercicios como encontrar un objeto en la escena o encontrar determinadas letras o palabras en una sopa de letras.
- Encontrar las diferencias entre dos escenas aparentemente iguales.
- Agrupar diferentes estímulos en categorías en base a similitudes entre ellos.
- Dividir la atención entre diferentes estímulos. Ejercicios que te haga prestar atención a varias cosas a la vez como enlazar diferentes elementos entre sí de acuerdo con una norma determinada.

4.2 Diseño de una tarea para la atención: Las diferencias

El "juego" de las diferencias se trata de una tarea diseñada para medir y entrenar la función ejecutiva de la atención. En esta tarea se le presentan al usuario dos imágenes con un número determinado de diferencias. La tarea del usuario será identificar estas diferencias. Mientras el usuario resuelve la tarea el sistema irá registrando en un fichero todas las acciones realizadas por dicho usuario.

Esos son una pequeña muestra de los diferentes ejercicios que podemos encontrar, otro de ellos sería el ejercicio que hemos trabajado en este trabajo, el juego de encontrar las diferencias, un juego destinado para entrenar la capacidad del usuario de prestar atención a dos estímulos muy similares. Además de acostumbrarle a prestar atención a los estímulos objetivo.

El juego consiste en presentar al usuario dos imágenes muy similares como podemos ver en la figura 6. El alumno deberá en este momento prestar atención a las dos figuras sin distraerse y ser capaz de encontrar sus diferencias. En el juego que hemos diseñado se tienen en cuenta los aciertos y los errores, no obstante, otros factores como el tiempo podrían ser considerados para medir el desempeño del alumno.

Figura 6. Ejemplo juego diferencias

4.2.1.1 Descripción de la tarea para un niño

Cuando el usuario inicia la tarea, y tras su carga, al usuario se le presentará la pantalla vista en la figura 7, en esta pantalla el usuario puede ver esta información:

- Las dos imágenes
- Un contador de las diferencias que quedan
- (Usándolo desde Siette) Un pequeño texto explicándole que debe encontrar las diferencias

Al tiempo al que se le presentará una melodía inicial que dará por empezado el ejercicio.

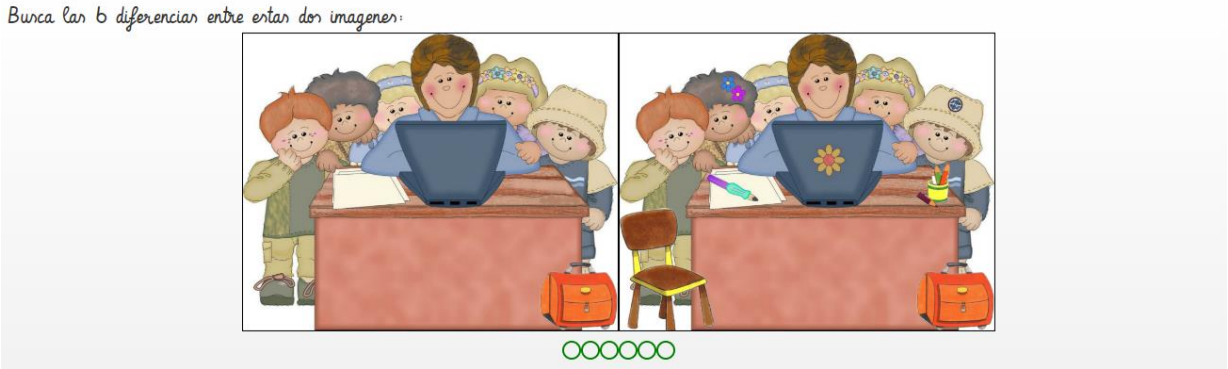

Figura 7. Ejemplo juego diferencias integrado en Siette

A partir de este momento el niño deberá encontrar las diferencias. En caso de que presione en algún punto se emitirá un pequeño tono de fallo y se mostrará una cruz en el lugar que haya fallado, eso lo podemos ver en la figura 8.

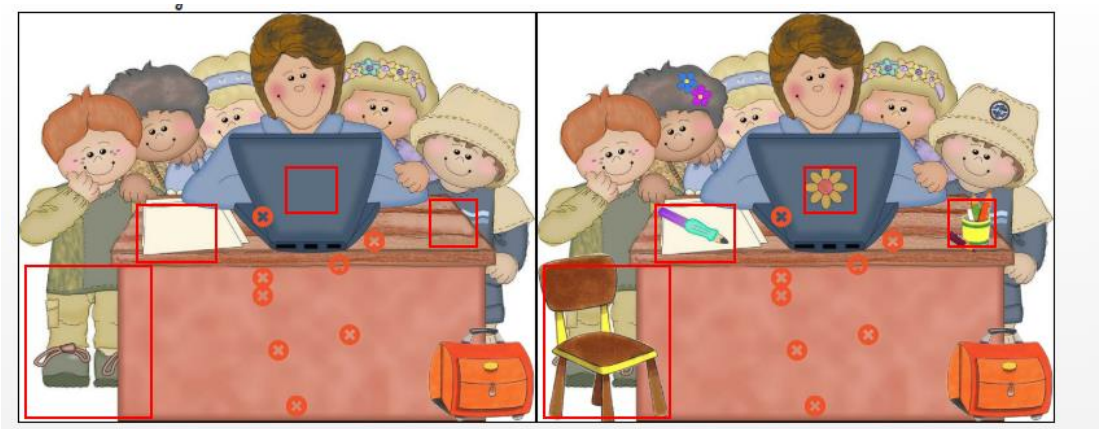

Figura 8. Ejemplo ejercicio a medio realizar

No obstante, si el estudiante pulsa una de las diferencias sonará un pequeño tono de triunfo o acierto, al tiempo que se resaltará en ambas imágenes dicha diferencia. Los sonidos se utilizan como un estímulo de recompensa al niño, tal y como explicamos en la sección de desarrollo de software para niños.

En el caso de presionar una diferencia ya descubierta con anterioridad el software no realizará ninguna acción ni emitirá ninguna melodía.

Cada vez que una diferencia sea acertada el contador de diferencias será actualizado para tener presente en todo momento cuantas diferencias han sido ya resueltas y cuales quedan por resolver. Eso lo veremos en la figura 9.

Figura 9. Ejemplo contador diferencias actualizado.

Finalmente, una vez el infante ha encontrado todas las diferencias sonará una breve melodía alegre indicando el final del ejercicio, en caso de estar ejecutándose sobre Siette además se pasará al siguiente ejercicio o al final del test, según corresponda.

4.2.2 Configuración de la tarea para un profesor

i. Tenemos dos imágenes que tienen varias diferencias, en este ejemplo serán seis

Figura 10. Imágenes con las diferencias.

ii. Entramos en Siette con un usuario que tiene el perfil profesor, y añadimos un nuevo ejercicio tipo Diferencias

iii. En el editor de la pregunta decimos cual es el fichero de cada imagen, junto a otros parámetros como los sonidos de inicio, fin, acierto, error, el ícono de fallo y las dimensiones de la imagen.

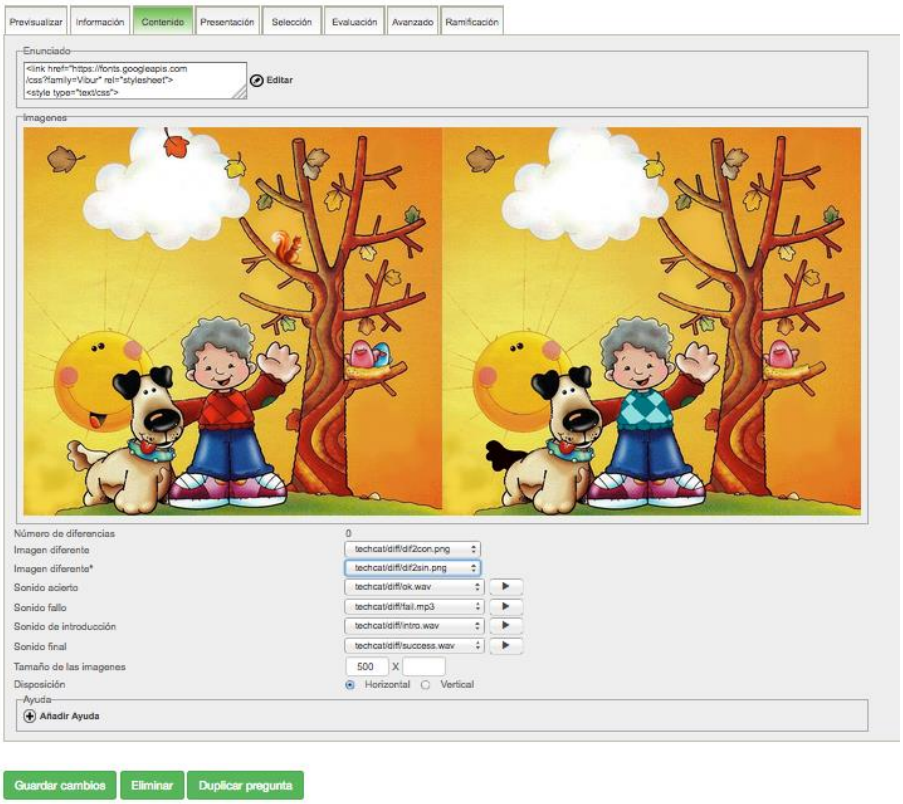

Figura 11. Ejemplo configuración en Siette de las diferencias

iv. Luego marcamos con el ratón donde están las diferencias, para ello lo que hacemos es definir los cuadros manteniendo pulsado el botón izquierdo del ratón y arrastrándolo definiendo las áreas. En caso de error el profesor puede utilizar el botón derecho para eliminar las áreas ya definidas.

Figura 12. Imagen mostrando la edición de diferencias

v. Decimos cual es el sonido de inicio, el de final, el de acierto y el de error. Así como el ícono que hay que mostrar en caso de fallo. También se indica sí que quiere mostrar en al usuario en vertical o en horizontal, así como el tamaño de la imagen (se indica o el ancho o alto, y entonces el sistema calcula el parámetro que falta, o ambos, pero con el peligro de que se distorsione la imagen).

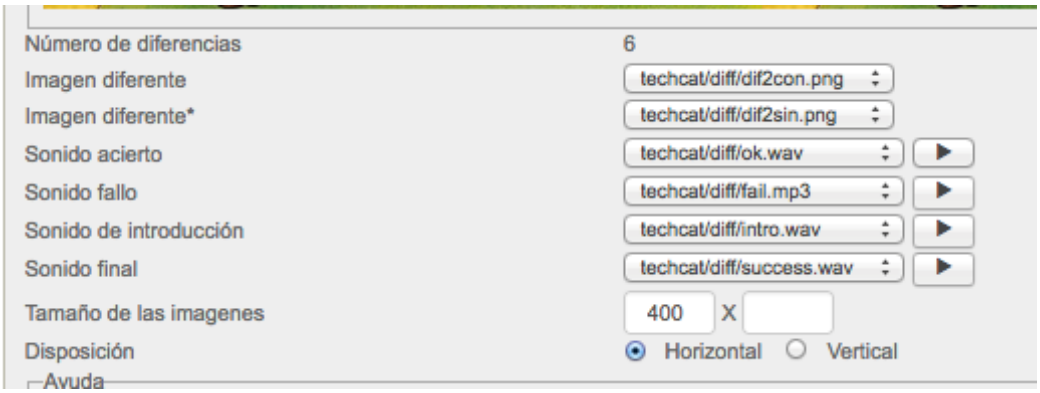

Figura 13. Formulario edición tarea en Siette

vi. Por último, el juego se prueba en modo usuario. En la figura 14 se muestra una foto de un niño resolviendo un juego de diferencias

Figura 14. Interfaz del juego en un entorno real

En el modo prueba se dan todos los eventos definidos anteriormente desde el punto de vista del niño.

4.2.3 El resultado: Evaluación automática de la tarea

Cuando el ejercicio termina, llama a Siette para que pase a la siguiente sección, ofreciéndole además una puntuación, esta puntuación tendrá en cuenta los aciertos y los fallos para obtener la puntuación final.

Ejecutándose en Siette, una vez el bloque de ejercicios ha terminado al usuario se le mostrará una puntuación sobre 5 estrellas. De nuevo dándole una recompensa tal y como hemos visto anteriormente. [Figura 15]

Figura 15. Finalización tarea.

4.2.4 El proceso: Descripción del log

Mientras todo este proceso se ejecuta, cada pulsación de ratón que el usuario haya realizado queda registrada en un log. Este registro incluirá la marca de tiempo, contada desde que empezó el ejercicio, las coordenadas, y si la pulsación fue un acierto o no.

Este log se le entrega a un servidor y queda registrado junto a un identificador del alumno que lo ha realizado, un identificador de la sesión en la que lo realizó y la información del ejercicio realizado.

Un log tiene la siguiente estructura

1. Cabecera con la información de la sesión, ejercicio, aluno fecha y tiempo
2. Información del ejercicio y las diferencias definidas en él
3. Cada pulsación de ratón, sus coordenadas, si tuvo éxito o no y el tiempo (contacto desde el inicio de la app)

En la figura 16 se muestra un ejemplo de un log, de un caso real de uso del juego en uno de los colegios donde se evaluó el TFG.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<session pupil="1" exercise="58" name="grupo1-DiferenciasCampo" date="29-nov-2010
11:43:29" time="11:43:29" mode="EVALUATION" width="1280" height="708"
type="diferencias">
 <concepts/>
<exercice>
<imageOrig path="./dif1con.png"/>
<imageMod path="./dif1sin.png"/>
<polygon>
   <vertex x="24" y="15"/>
   <vertex x="24" y="48"/>
   <vertex x="63" y="15"/>
   <vertex x="63" y="48"/>
</polygon>
<polygon>
   <vertex x="282" y="323"/>
   <vertex x="282" y="390"/>
   <vertex x="384" y="323"/>
   <vertex x="384" y="390"/>
</polygon>
</exercice>
<mouselog>
   <action time="1198345" action="1" x="47" y="30"/>
   <action time="1199105" action="2" x="267" y="27"/>
   <action time="1200161" action="1" x="103" y="149"/>
</mouselog>
</session>
```

Figura 16. Ejemplo reducido de un log

4.2.5 Mecanismo de actualización de los ficheros en el servidor

Tal y como se ha especificado, durante el sistema de log se registra periódicamente las acciones del usuario.

Cuando el sistema arranca, el juego envía al servidor un log inicial con los datos del problema y la cabecera

Cada vez que el infante realiza alguna acción de mouse el log se actualiza y se envía una copia al servidor, el servidor sigue a la espera de nuevas instrucciones

Cuando el juego termina, o en su defecto, cuando el servidor no ha recibido información del cliente durante un tiempo definido de 10 minutos el servidor procede a almacenar en un fichero con el nombre del id de sesión dicho log.

Este es el motivo por el que se envía una copia al servidor cada vez que el niño realiza alguna acción, para que en caso que el cliente dejase de funcionar o perdiera la conectividad el servidor tuviera al menos un log parcial para poder extraer algunos de los datos.

4.3 Diseño de la tarea de encontrar el objeto en la escena

La tarea de encontrar un objeto en la escena es una tarea clásica para entrenar la atención. en la que se le presenta al usuario una escena y una serie de objetos que debe encontrar en ella.

Del mismo modo que en la tarea anterior en esta tarea también todas las acciones del usuario se van a ir registrando en un *log*.

La tarea es muy similar al juego de las diferencias anteriormente expuesto. Para evitar caer en repeticiones a lo largo de estas secciones nos vamos a centrar en los factores diferenciantes del juego en lugar de explicar todo de nuevo.

4.3.1.1 Descripción de la tarea para un niño

Cuando el usuario empieza la tarea se le mostrará la pantalla de juego, la pantalla incluye la escena y todos los íconos de las cosas que debe buscar como se muestra en la figura 17.

Figura 17. Ejemplo juego diferencias

A partir de este momento el niño interactuará con el juego, de forma similar al ejercicio anterior los éxitos y fracasos se le irán notificando de forma dinámica además de quedar registrados en el *log*

4.3.2 Configuración de la tarea para un profesor

La mayoría de los parámetros son similares a los ya vistos en las diferencias, siendo la principal diferencia que ahora hay una lista con los objetos que pueden ser encontrados. Del mismo modo el editor presenta ciertas diferencias.

En el editor de ítems el profesor podrá elegir para cuál de los 3 elementos a buscar está seleccionando sus áreas, el cual quedará marcado por un borde rojo como veremos en la figura 18.

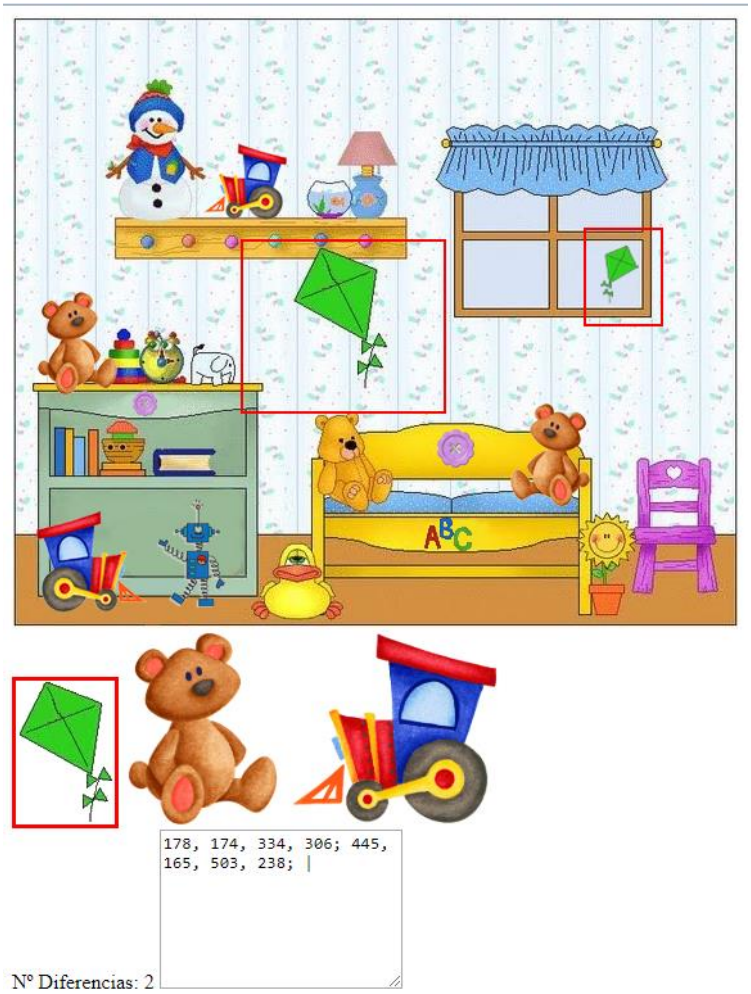

Figura 18. Ejemplo juego buscar el objeto en la escena

Conforme el profesor va cambiando se cambian en pantalla de forma dinámica las diferencias definidas. De modo que si el profesor define las áreas del objeto 1 y pasa a las del objeto 2 se dejarán de mostrar en pantalla las del primero, que se volverán a mostrar si vuelve a seleccionar dicho objeto.

5 SOLUCIONES TECNOLÓGICAS PROPUESTAS

5.1 Arquitectura general

El sistema se basa en diversos componentes, los cuales son:

- El juego de las diferencias
- Siette, donde se integran los juegos
- El servidor de guardado de log, integrado en Siette
- El editor del juego de las diferencias.
- El editor de diferencias, que se integra con el editor de diferencias de Siette.

La relación entre dichos componentes se ve en el siguiente diagrama:

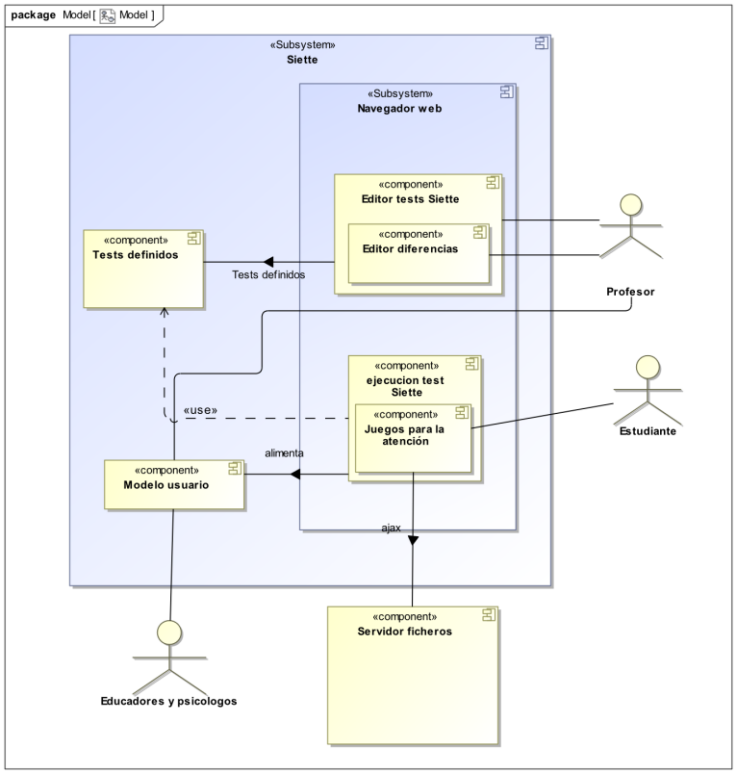

Figura 19. Diagrama UML con los componentes

5.2 Implementación de las tareas de las diferencias

Para realizar la tarea de las diferencias se ha optado por realizar una aplicación en TypeScript, traduciéndose a JavaScript. El resultado es una tarea externa a Siette, que funciona de forma independiente en un navegador web.

Esta aplicación se trata de un pequeño programa en TypeScript que se encarga, dado como entrada una serie de parámetros, incluyendo:

- Las direcciones donde se encuentran los archivos de audio
- Las direcciones donde se encuentran las imágenes
- La dirección del icono de error
- Un array con las diferencias existentes.
- Las dimensiones del array
- Un input especificando que *layout* (propiedad display en CSS) se quiere que tenga el contendor de imágenes.

El sistema tiene tantos valores de entrada porque la idea es que utilizando más adelante la capacidad de Siette de definir muchos ejercicios diferentes dado una plantilla y una serie de parámetros el sistema permita un número indeterminado de ejercicios sin cambiar el código fuente ni recompilar.

Podemos ver un ejemplo de la llamada en la figura 20

```
const differences = [];
    differences.push({ topLeft: { x: 24, y: 15 }, bottomRight: { x: 63,
y: 48 } });
    differences.push({ topLeft: { x: 93, y: 131 }, bottomRight: { x: 116,
y: 162 } });
    differences.push({ topLeft: { x: 154, y: 310 }, bottomRight: { x:
193, y: 346 } });
    differences.push({ topLeft: { x: 247, y: 19 }, bottomRight: { x: 277,
y: 45 } });
    differences.push({ topLeft: { x: 282, y: 323 }, bottomRight: { x:
384, y: 390 } });
    differences.push({ topLeft: { x: 489, y: 324 }, bottomRight: { x:
528, y: 352 } });

differencesGame(['./assets/dif1con.png', './assets/dif1sin.png'],
      'containerForGame', './assets/ok.wav', './assets/fail.mp3',
      './assets/intro.wav', './assets/success.wav', null, null, 'inline-
      block', './assets/cancel.png', 'http://localhost:8080', 'id',
      differences);
```

Figura 20. Ejemplo de llamada para ejecutar el juego con parámetros

Lo primero que hará el juego es cargar las imágenes en el contenedor y ponerse a la escucha de los eventos del ratón (lo que incluye pulsaciones en pantalla táctil), así como reproducir el audio de inicio del juego. De este modo, una vez terminada la inicialización el sistema se quedará en espera y no hará nada hasta que el navegador ejecute.

Una vez se recibe el evento de pulsación, lo primero que se hace es calcular las coordenadas de este. Ya que, aunque en la propiedad *offset* del *MouseEvent* vienen las coordenadas[MDN], estas coordenadas vienen relativas al elemento hoja del árbol DOM, en lugar del elemento que está escuchando el evento. Esto es un problema en nuestro caso porque recordemos que tanto los aciertos como los fallos se le muestran al usuario, usando, tal y cómo se ha explicado en el apartado cuatro, creando elementos HTML que se cuelgan del mismo contenedor que la imagen y que son los que usará el navegador como referencia para determinar las coordenadas.

Por ese motivo, lo primero que se hace es sumar las coordenadas especificadas en *offset* a las coordenadas del elemento hoja (*target*), como vemos en la figura 21.

```
function normalizePoint(ev: MouseEvent, nameID: string) {
    const parent = <HTMLElement>ev.target;
    let point: Point;
    if (isDrawnSquare(parent, nameID)) { // El usuario ha pulsado un cua-
dro que ya existía
        point = { x: ev.offsetX + parent.offsetLeft, y: ev.offsetY + pa-
rent.offsetTop };
    }
    else {
        point = { x: ev.offsetX, y: ev.offsetY };
    }
    return point;
}
```

Figura 21. Normalización del punto pulsado

Una vez calculada la posición real del evento se comprueba si está dentro de una de las diferencias definidas. Si está fuera de todas las diferencias definidas se incrementa el contador de fallos, se reproduce un mensaje de error y se muestra el ícono definido en el apartado 4.

En el caso que este dentro de alguna diferencia tenemos a su vez dos posibilidades:

- Está dentro de una diferencia ya encontrada con anterioridad: En ese caso no hacemos nada (excepto registrarlo en el log).
- Está dentro de una diferencia que no se ha encontrado, en ese caso:
    - Marcamos en una variable booleana del propio objeto que almacena las diferencias para indicar que la hemos encontrado.
    - Reproducimos el sonido de éxito
    - Actualizamos el contador de aciertos, tanto el valor interno como el contador final.
    - Dibujamos un rectángulo señalando la diferencia en ambas imágenes, en la figura 22 podemos ver como se ajusta el posicionamiento del rectángulo
    - En caso que sea el final reproducimos el sonido de fin del juego, en caso de estar ejecutándose en Siette a los cinco segundos llamaremos a su método *RespuestaActiva* para acabar el juego.

```
function editDifferenceSquare(differenceDom: HTMLElement,square: Square)
{
    differenceDom.style.cssText = `position: absolute; top:
${square.topLeft.y};
    left: ${square.topLeft.x}; width: ${square.topRight.x -
square.topLeft.x};
    height: ${square.bottomLeft.y - square.topLeft.y}`;
    differenceDom.style.border = '2px solid red';
    return differenceDom;
}
```

Figura 22. Ajuste del rectángulo señalando la diferencia.

Una vez el usuario ha conseguido todas las diferencias y tras un tiempo de espera de cinco segundos se llama al método RespuestaActiva de Siette, el cual usará la puntuación obtenida (aciertos – fallos / 2) para incorporarla al modelo de usuario.

5.3 Editor de diferencias.

Tal y como se especificó en el apartado cuatro, el profesor es el encargado, de tras elegir las imágenes a usar, definir sus diferencias.

Para definir se escribió un pequeño editor de diferencias con funcionalidad básica que permite definir las diferentes áreas de diferencias.

Para inicializar el editor se le pasan los identificadores de los elementos del DOM que contienen las imágenes, el identificador del elemento del DOM que contiene el número indicando las diferencias que están seleccionadas y un *string* con las diferencias previamente seleccionadas serializadas, que se usa en caso de que estemos editando un ejercicio previamente definido en lugar de creando uno nuevo.

A continuación, se muestra el formato del string, en la figura 24 se mostrará el código responsable de su interpretación

```
square1.topLeft.x, square1.topLeft.y, square1.bottomRight.x,
square1.bottomRight.y; square2.topLeft.x, square2.topLeft.y,
square2.bottomRight.x, square2.bottomRight.y
```

Figura 23. Formato serialización diferencias

El mismo formato con el que recibimos las diferencias pasadas, es el formato en el que entregamos a Siette las diferencias para que las almacene y puedas mostrar posteriormente la imagen.

```
   function parseKnownDifferences() {
      const squaresCoordinates = serializedString.split(';');
      squaresCoordinates.forEach(cordinateString => {
         const coordinates = cordinateString.split(',');
         const square = new Square(new Point(+coordinates[0],
+coordinates[1]), new Point(+coordinates[2], +coordinates[3]));
         createSquareInDOM(square);
         pushSquareToList(square);
      });
      selectedSquare = null;
      selectedSquare2 = null;
   }
```

Figura 24. Código para des serializar las diferencias

Una vez el editor ha sido invocado lo que hace es subscribirse a los eventos de ratón de las imágenes.

Cuando detecta un nuevo clic, lo primero que hace es determinar si se trata del botón derecho o izquierdo, en caso de que sea derecho borramos la diferencia sobre la que se encuentra el ratón (si hay alguna).

En caso de que sea el botón izquierdo creamos un nuevo rectángulo en el DOM y apuntamos su referencia en variables internas, de este modo conforme el profesor mueva el ratón se irá actualizando en tiempo real dicho rectángulo. Nótese, que tanto en la primera pulsación de ratón, como conforme el usuario mueve el ratón, debemos calcular las coordenadas reales del cursor usando la lógica especificada en el apartado anterior.

Cuando el usuario suelte el clic, la diferencia quedará registrada, el contador también se actualizará y el sistema se quedará a la espera de más pulsaciones.

5.4 Juego del objeto en la escena

En este caso, la implementación del juego del objeto en la escena es muy parecida a la del juego de las diferencias. El principal cambio consiste en que hay sólo una escena y que además debemos mostrar los objetos que el niño debe buscar. Creando para ello dos contenedores HTML diferentes.

Del mismo modo la invocación al juego es diferente, en este caso se ha preferido utilizar tres parámetros con objetos para no sobrecargar mucho la firma de la función. La firma es la siguiente:

```
function spotTheObjectGame(gameSettings: GameSettings, domSettings:
DomSettings, serverSettings: ServerSettings)
```

Figura 25. Firma de la llamada para empezar el juego

gameSettings se trata de un objeto con todos los atributos del juego, incluyendo la escena, los objetos a buscar, la altura y la anchura, los sonidos de éxito, etc.

domSettings se trata de un objeto con algunos ajustes técnicos para incrustar correctamente el juego en una página web.

serverSettings se trata de un objeto conteniendo la URL y el id de sesión para guardar el log.

A continuación, podemos ver un ejemplo de dichos objetos de configuración:

```
const gameSettings: GameSettings = {
        cancelPath: './assets/cancel.png',
        endAudioPath: './assets/success.wav',
        failAudioPath: './assets/fail.mp3',
        imagePath: './assets/escena3.png',
        imgHeight: null,
        imgWidth: null,
        objects: objects,
        okAudioPath: './assets/ok.wav',
        startAudioPath: './assets/intro.wav'
    };

    const domSettings: DomSettings = {
        blockLayout: 'inline-block',
        containerID: 'containerForGame'
    }

    const serverSettings: ServerSettings = {
        sessionID: '5',
        url: 'http://localhost:8080'
    }
```

Figura 26. Objetos de configuración

5.5 Editor del objeto en la escena

El editor del juego también ciertas distinciones respecto al de las diferencias, en particular, y como definimos en el apartado cuatro, la ocultación y mostrado de las diferencias pertenecientes a un objeto que no es el que estamos modificando.

Para realizar esto lo que se ha hecho es añadir un *eventListener* a los íconos de los objetos para cuando el usuario clique poder realizar los cambios.

En concreto a nivel interno, mientras que en el editor de diferencias teníamos un *array* con las diferencias definidas, en este editor tenemos un *array* de *arrays*, de este modo tenemos un listado de diferencias para cada uno de los objetos. Así, cuando el usuario cambie de diferencia es tan "sencillo" como eliminar del DOM los cuadros asociados al objeto anteriormente seleccionado y poner los asociados al id nuevo, un pequeño fragmento del código responsable se muestra a continuación:

```
        function selectME(elementNumber: number) {
            if (selectedSquare) {
                pushSquareToList(lastGoodSquare);
                onEnd();
            }
            id = 0;
            listSquares[selectedItem].forEach(item => {
                document.getElementById(ID_DIV_PREFIX +
    item.numID).parentElement.removeChild(document.getElementById
    (ID_DIV_PREFIX + item.numID));
            });
            selectedItem = elementNumber;
            listSquares[selectedItem].forEach(item => {
                createSquareInDOM(item.originalSquare);
                item.numID = id
                id++;
            });
            selectedItem = elementNumber;
```

Figura 27. Código para alternar el ítem seleccionado

5.6 Integración de la tarea en SIETTE

Cuando la tarea de las diferencias (así como el editor) se desarrollaron, aunque se tenía en mente que acabarían siendo incorporadas en Siette se desarrollaron como aplicaciones web independientes. Más tarde, junto al equipo de Siette se realizaron los cambios pertinentes para su correcta integración dentro del sistema.

Entre estos cambios probablemente el más importante fuese llamar a la función *RespuestaActiva* de Siette.

RespuestaActiva es una de las funciones predefinidas de Siette, en este caso, RespuestaActiva se encarga de comunicar a Siette que el ejercicio ya ha terminado, para que Siette recoja la puntuación y pase al siguiente paso.

También hubo que cambiar la forma en la que se colocaban las imágenes en el DOM. Ya que la propiedad display en *inline-block*, como estaba cuando

estaba como aplicación independiente, causaba problemas, de modo que se cambió para que ahora sea una propiedad configurable al invocar el juego.

En cuanto al editor de diferencias, también se realizaron algunos cambios:

- Se eliminó el input que definía el botón para acabar la edición. Esto es así ya que a partir de la integración con Siette, el string de diferencias serializadas se actualiza en tiempo real, en lugar de actualizarse cuando se le daba a botón de finalizar, cómo hacía cuando era una aplicación web independiente.
- Se introdujo un input conteniendo un *span* HTML donde el programa va actualizando, en tiempo real, el número de diferencias que hay registradas.
- Se introdujo otro input, esta vez con un *textarea* en el que se muestra, y se va actualizando conforme se van modificando, las diferencias el *string* de diferencias serializado.

5.7 Ejecución de la tarea en SIETTE

Cuando un infante va a participar en una actividad del juego de las diferencias en Siette mantiene el siguiente flujo de trabajo:

1. El alumno utiliza un código QR (figura 28) para navegar a la URL indicada, este QR lleva incluido el ID del experimento y del alumno, por lo que desde ese momento su sesión queda monitorizada.
2. El navegador web le lleva hasta el test de Siette donde se ejecuta el juega de las diferencias.
3. El niño empieza a jugar al juego. Este alumno no conoce la existencia de Siette ni es consciente de que con su actuación se ayuda a alimentar el modelo de usuario del sistema, no obstante, por detrás los profesores si están disfrutando de toda la potencia que Siette provee. Al mismo tiempo, las acciones quedarán registradas en el *log* para poder más tarde estudiar su comportamiento.
4. Finalmente, el niño acaba la actividad y se le muestra su puntuación en estrellas, durante la actividad su actividad fue monitorizada y ahora se guardará un registro de la misma.

Figura 28. Código QR de ejemplo de acceso a un ejercicio

5.8 Mecanismo de escritura de ficheros externos

Para escribir el log en un servidor lo que se hizo fue un pequeño *servlet* en Java que atendía una petición post. Este *servlet* lo que hace es atender una petición HTTP Post con el cuerpo conteniendo una versión preliminar del log. En los parámetros de la petición se encuentran también:

1. sessionID: El ID de la sesión a registrar.
2. end: Marca si la sesión ha finalizado.

Al ejecutar el post, el servidor crea una sesión y devuelve un objeto JSON como el de la figura 29, además devuelve una Cookie con el id de sesión. El ID de sesión es la sesión vinculada y deberá ser enviada en forma de cookie o poniendo en la URL *";jsessionid=ID"* en todas las sucesivas peticiones,

El parámetro *timeout* es cuando caduca la sesión en segundos desde la última petición, en caso que el cliente quiera saberlo. Aunque este parámetro no se llegó a usar en las versiones desarrolladas, podría usarse para realizar un *keep-alive,* es decir, que el cliente le comunique al servidor que sigue conectado aunque el usuario no haya introducido nada el cliente sigue en línea.

```
{ "timeout": 1800, "sessionID": "E1193536DCCECC890D604947A390BD68" }
```

Figura 29. Objeto devuelto por el servidor

De este modo, el cliente nada más arrancar realizaría la primera petición y recibiría el ID de sesión. A partir de este punto para cada acción realizada por el niño, la aplicación JavaScript actualizaría su variable interna de log y enviaría una nueva copia al servidor, hasta la finalización del ejercicio donde enviaría el *flag end* a *true* y el servidor almacenaría, en un fichero con el ID de sesión de Siette, el log final.

6 METOLOLOGÍA DE DISEÑO

6.1 DESCRIPCION DE LA METODOLOGÍA

Para el desarrollo de la aplicación se ha seguido la metodología de desarrollo ágil: SCRUM, con ciclos de 3-4 semanas de desarrollo.

SCRUM consiste en un conjunto de técnicas para el desarrollo de productos, especialmente de Software [Wykowski, T. & Wykowska, J. 2018]. A diferencia de otros métodos como el método de desarrollo en cascada en SCRUM no se centra directamente en un objetivo si no que se van estableciendo diversas metas pequeñas que se resuelven en los llamados sprint, periodos de trabajo de 2-4 semanas

6.2 CICLO 1. Implementación en JavaScript de la tarea de las diferencias

En esta primera versión del juego se hizo la implementación del juego de las diferencias en JavaScript, centrándose la tarea en la localización de las áreas que diferenciaban a un dibujo de otro. En este caso, se partía de la configuración de un juego en el que se daba como entrada las áreas que designaban una diferencia en el juego. En la figura 30 se muestra la interfaz de esta primera versión.

Después de probar esta versión con el equipo de desarrollo se observaron los siguientes puntos a mejorar:

- Incorporación de sonido
- Añadir un indicador del número de diferencias que quedan por descubrir para dar

- Incluir una puntuación del ejercicio.

Estos elementos se consideran en el siguiente ciclo.

Además, en esta versión aún no se había realizado la integración con Siette.

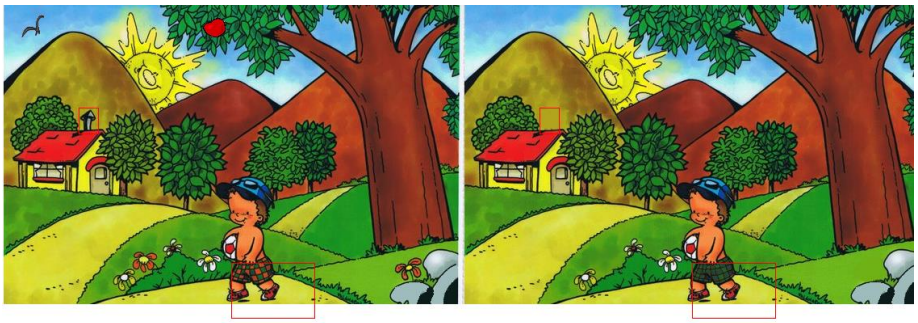

Figura 30. Versión inicial de las diferencias sin contador de fallos

6.3 CICLO 2. Implementación de la segunda versión de las diferencias y del editor del profesor. Primera prueba con usuarios

- Se incluye sonido al iniciar y al finalizar el ejercicio
- Integración del editor en Siette
- Prototipo del contador de diferencias
- Primera versión de la puntuación.
- Primera versión del editor de diferencias

En esta versión, al igual que la anterior detectamos algunos problemas:

- El usuario no tenía suficiente información sobre su fallo, se decide meter algún tipo de aviso.
- El sistema no funciona en tablets viejas, en concreto en Android 4.

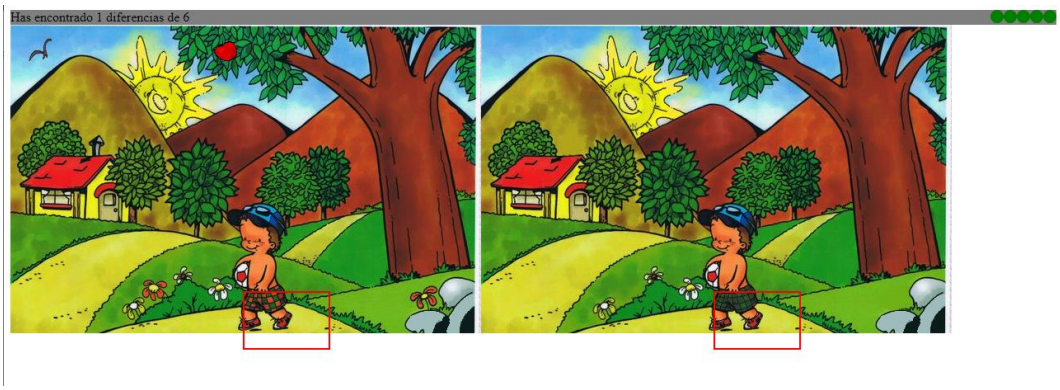

Figura 31. Versión preliminar del contador de fallos

6.4 CICLO 3. Implementación de la tercera versión más un generador del log. Segunda prueba con usuarios reales

En este ciclo se implementó el sistema de registro, tanto del cliente como del servidor, que se ejecuta en Siette, también se mejora la apariencia del contador de diferencias:

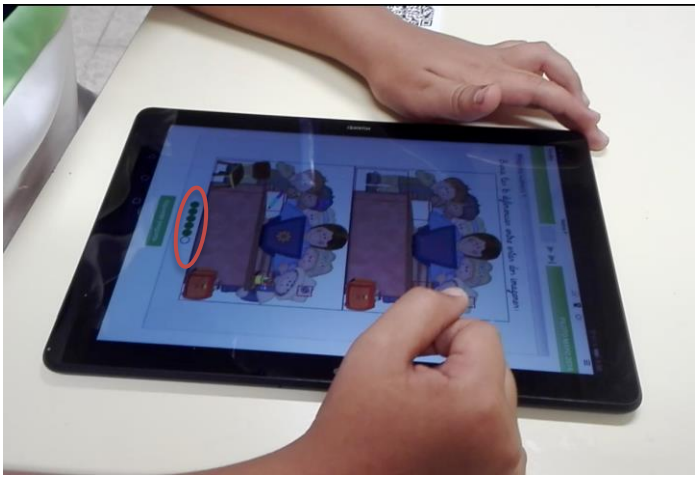

Figura 32. Ejemplo con usuario real donde se ve el nuevo contador de fallos

6.5 CICLO 4. Versión final y prueba con usuarios reales

En este ciclo se realizaron algunos ajustes finales al sistema:

- Se incluyó un icono de una cruz cuando el usuario intentaba pulsar una diferencia que no existe. De este modo el usuario tiene más claro que no debe pulsar aleatoriamente, esto se hace en consonancia con lo visto en el apartado dos, en la sección de desarrollo de Software para niños, en la que se específica que se deben utilizar las animaciones para darle información al niño. De este modo reforzamos la percepción del usuario sobre el fallo, que ya venía también por el sonido de error.
- Se adaptó a sistemas más viejos (Android 4/Chromium 2) para permitirle funcionar en tablets viejas.

6.6 CICLO 5. Implementación de la tarea y el editor de buscar el objeto en la escena

En el último ciclo se implemento la tarea de buscar el objeto en la escena, así como el editor de dicho juego.

Además, se corrigieron algunos pequeños errores en el editor de las diferencias

7 CONCLUSIONES y FUTUROS TRABAJOS

En este TFG se han diseñado e implementado un conjunto de programas para el enriquecimiento cognitivo de la atención en niños pequeños. Estos programas se han integrado en el sistema Siette y se han evaluado con usuarios reales en colegios de educación infantil. Todo este desarrollo ha sido realizado en siguiendo una metodología ágil, combinando la implementación, con la integración en Siette y el uso/prueba con niños.

7.1 Conclusiones

Concretamente podemos resumir esta conclusión general en los siguientes puntos:

- Se ha estudiado qué es una función ejecutiva, para que sirven y como se pueden mejorar con programas informáticos. Concretamente, se ha estudiado la atención y tipos de tareas que pueden ayudar a mejorarla.
- Partiendo de una tarea clásica que ayuda a la atención, el juego de las diferencias se ha diseñado un juego desde cero y se ha implementado en JavaScript para un navegador normal: la interfaz, acciones correctas, acciones incorrectas, respuestas al usuario, según las necesidades del proyecto TECHCAT.
- Se ha desarrollado un sistema de registro de eventos compartido por todos los juegos de TECHCAT que comunique el cliente con un servidor, en el que se irán registrando las acciones relevantes que realiza el usuario, a fin de que los investigadores sean capaces de analizarlas más en profundidad.
- Se ha trabajado con el equipo de Siette, adaptando el juego implementado para incorporarlo como ítem externo en Siette. Concretamente se incorporaron los mecanismos para que Siette pueda recoger las interacciones de los niños para que sean evaluadas, las acciones correctas, las incorrectas, así como la interpretación que el psicólogo quiere dar a esas acciones. Con todos estos elementos, Siette

hace un modelo de usuario que se utiliza a lo largo de proceso de aprendizaje de los niños.

- Se ha probado el sistema con niños en dos colegios en entornos de aprendizaje, durante una prueba piloto del proyecto TECHAT. El juego fue utilizado por niños de 3, 4 y 5 años, y su uso ha permitido validar la idoneidad de la interfaz diseñada, los sonidos y el feedback. Actualmente, el equipo de psicólogos del proyecto está estudiado los videos y las respuestas de los niños, en relación al plan de enriquecimiento diseñado para el proyecto. Desde el punto de vista informático, el diseño de la tarea es adecuado para los niños, cumple las restricciones establecidas por diseñadores de software para niños mencionadas a largo de esta memoria.

7.2 Futuros Trabajos

Hay varias ideas y mejoras que se podrían aplicar a este proyecto con respecto a la definición de nuevas tareas y a la incorporación de nuevas funcionalidades para el profesor que permitan hacer un seguimiento del proceso de enriquecimiento cognitivo:

En el grupo de nuevas tareas podemos citar:

- Juegos que permitan agrupar elementos en base a ciertas similitudes.
- Juegos en los que haya prestar atención a dos elementos a la vez, por ejemplo, un juego en el que haya que prestar atención a dos personajes en los que uno hace una cosa mientras otro estropea la escena.
- Ejercicios de atención selectiva como hacer puzles con diferentes niveles de dificultad.
- Ejercicios de sopas de letras, entre otros.

Otra futura línea de trabajo, ya planificada como parte del proyecto TECHAT, es la implementación de un programa que procese un log generado por los juegos, permita definir indicadores, y represente gráficamente los datos o el procedimiento de enriquecimiento.

8 REFERENCIAS

Borse J., Robles E., and Schwartz N., "Designing for Kids in the Digital Age: Summary of research and recommendations for designers," Proceeding in the First interaction design & children conference in Eindhoven, The Netherlands, 2006.

Bray, T. 2017. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. https://tools.ietf.org/html/rfc8259#section-4

Castillero Mimenza, O. (Consulta: 2019) Las 11 funciones ejecutivas del cerebro humano. https://psicologiaymente.com/inteligencia/funciones-ejecutivas

Chrysafiadi, K., Virvou, M. (2013), Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications* 40(11):4715–4729.

Conejo, R., Guzmám E. (consulta: 2019). SIETTE: Sistema Inteligente de Evaluación mediante Test para TeleEducación. XII cursos de verano de la UNED, España, 2001

Cooper-Kahn, JM., Dietzel, L., (2008) Late, Lost, and Unprepared: A Parents' Guide to Helping Children with Executive Functioning

Davidson, M. C., Amso, D., Anderson, L. C., & Diamond, A. (2006). Development of cognitive control and executive functions from 4 to 13 years: evidence from manipulations of memory, inhibition, and task switching. *Neuropsychologia, 44(11), 2037–2078.*

Gratacós, M. (Consulta 2019) 10 Ejercicios para Mejorar la Atención (Niños y Adultos). *lifeder* https://www.lifeder.com/ejercicios-para-mejorar-la-atencion/

https://cs.lmu.edu/~ray/notes/webapps/

Json.org (consulta 2019). Introducing JSON. https://json.org/

Laamarti,F., Eid, M. El Saddik, A. (2014) An Overview of Serious Games," *International Journal of Computer Games Technology*, vol. 2014.

Markopoulos, P. Read, J.C., MacFarlane, S. & Hoysniemi, J. (2008) *Evaluating Children's Interactive Products: Principles and Practices for Interaction Designers*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Malenka R. C., Hyman, S. E., Nestler, E. J. (2009). Molecular Neuropharmacology: A Foundation for Clinical Neuroscience. Mcgraw-Hill Publ.Comp.

Mozilla Development Network (consulta: 2019) https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

Pavlik Jr., P. I., Brawner, K. W., Olney, A., & Mitrovic, A. (2013). A Review of Learner Models Used in Intelligent Tutoring Systems In R. A. Sottilare, A. Graesser, X. Hu, & H. K. Holden (Eds.), Design Recommendations for Adaptive Intelligent Tutoring Systems: Learner Modeling (Vol. 1, pp. 39-68): Army Research Labs/ University of Memphis.

Sun Microsystems. (1997) The Java Language Environment

Sun Microsystems. (consulta: 2019) Distributed Application Architecture

Toal, R. (consulta: 2019) Web Application Architecture.

Urretavizcaya Loinaz, M. (2001), Sistemas inteligentes en el ámbito de la educación. Inteligencia Artificial. *Revista Iberoamericana de Inteligencia Artificial*, 5 (primavera)

Wykowski, T. & Wykowska, J. 2018 Lessons learned: Using Scrum in non-technical teams

Listado de Figuras