TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

CURSO 2015/2016

DEDOS-PLAYER PARA TABLETAS ANDROID

AUTOR: ÓSCAR MARTÍN MARTÍN

TUTORA: ESTEFANÍA MARTÍN BARROSO

Abril 2016

Agradecimientos

Quiero expresar mi agradecimiento a todas y cada una de las personas que de una forma u otra han participado en la realización de este TFG.

En primer lugar, a mi tutora Estefanía, por todo el apoyo recibido a lo largo de este proyecto y todos estos años de carrera, que con su dedicación ha conseguido sacar lo mejor de mí.

A mi pareja, por aguantar todas las locuras que tiene un informático y darme apoyo en todos los momentos difíciles a lo largo de esta etapa.

A mi familia, por apoyarme desde pequeño para que cumpliera mi vocación, porque sin ellos esto no habría sido posible.

Y por último a todos los magníficos compañeros que he tenido, que entre todos hemos conseguido superar las dificultades que encontrábamos en el camino.

Muchas gracias.

Óscar Martín

Resumen

A lo largo de los últimos años hemos observado cómo el mundo de la tecnología ha ido evolucionando a pasos agigantados. Hechos que hace 30 años se consideraban como ciencia ficción son hoy una realidad para todos nosotros y lo vemos como algo normal.

Un área que ha tenido una revolución total ha sido el aprendizaje, que ha ido evolucionando para dar respuesta a la necesidad que tienen las generaciones actuales de nuevas maneras de aprender. Un niño ya no se conforma con un libro o una clase meramente teórica, necesita otros estímulos que las Tecnologías de la Información y la Comunicación ponen a su alcance.

Con estas nuevas herramientas el alumno puede ver el aprendizaje de una manera diferente, y no asociarlo al aburrimiento que le pueden generar las clases, de manera que puedan despertar en él otras sensaciones como la curiosidad o despertar la creatividad del alumno. Por ello es de gran utilidad desarrollar herramientas que faciliten el proceso del aprendizaje a los alumnos y de la enseñanza a los profesores. Un dispositivo muy usado en las aulas hoy en día son las tabletas, las cuales están prácticamente integradas en el día a día de cualquier estudiante.

En este Trabajo Fin de Grado (TFG) se ha desarrollado una herramienta que permite a los alumnos realizar proyectos educativos en tabletas Android. Esta herramienta se encuentra enmarcada dentro del proyecto DEDOS. Con esta versión para tabletas, el alumno podrá aprender prácticamente en cualquier lugar, y a la vez proporcionarle diversión, todo ello con actividades personalizadas creadas con DEDOS-Editor por los profesores. Además la ventaja de DEDOS con respecto otra cualquier herramienta es la capacidad de personalización, con lo que el profesor puede diseñar actividades adaptadas en función del avance del alumnado.

La versión de DEDOS-Player implementada en este TFG está diseñada para que interactúe un único alumno por tableta y soporta distintos tipos de actividades: selección, emparejamiento y matemáticas.

A lo largo de esta memoria, se irá destallando desde la explicación de la necesidad de realizar este desarrollo, pasando por la fase de diseño, desarrollo y evaluación del software desarrollado en este Trabajo Fin de Grado.

Contenido

Índice de Figuras

Índice de Tablas

1 Introducción

El proyecto DEDOS (Proyecto DEDOS, 2008) nació para facilitar al profesorado la elaboración de contenidos educativos de manera que estos pudieran ser trabajados en el aula con mesas multicontacto de una manera mucho más dinámica que con las técnicas tradicionales de enseñanza. Esta nueva metodología permite tener un alumnado mucho más motivado y conseguir a largo plazo mejores resultados académicos. Es importante tener en cuenta que el proyecto DEDOS recibió el premio a "*Mejor Proyecto TIC Educación Inclusiva, Igualdad y Diversidad*" dentro de los Premios a la Innovación Educativa del Salón de la Tecnología para la Enseñanza SIMO 2014.

Dentro del proyecto DEDOS nos encontramos con dos herramientas claramente diferenciadas:

- DEDOS-Editor es la herramienta con la cual los profesores, sin conocimientos informáticos avanzados, pueden crear contenidos educativos de manera totalmente dinámica adaptándose a las necesidades de los alumnos. El diseño de las actividades se realizará por medio de "cartas" que el profesor irá colocando según sus preferencias o necesidades en las diferentes áreas de trabajo. Una vez que el profesor haya maquetado la actividad con los requisitos deseados se generará un fichero que contendrá toda la información relativa a las actividades creadas.
- DEDOS-Player es la herramienta con la cual los alumnos podrán realizar las actividades previamente elaboradas con la herramienta DEDOS-Editor. Esta herramienta tendrá como punto de partida el proyecto generado con DEDOS-Editor, lo procesará y presentará al alumno las distintas actividades tal y como se han diseñado previamente.

El desarrollo de este proyecto surge por la necesidad de contar no sólo con una versión de DEDOS-Player para ordenadores, pizarras digitales o mesas multicontacto, sino también para que esta versión pudiera ser utilizada en tabletas con sistema operativo Android. Esto es debido a que la actual versión para ordenadores, pizarras digitales y mesas multicontacto estaba desarrollada en Flash, tecnología no soportada en las actuales tabletas Android.

1.1 Experiencias educativas con DEDOS en Educación Infantil, Educación Primaria y Educación Especial

El uso de las herramientas del proyecto DEDOS permite al docente desarrollar sus clases de una manera mucho más dinámica que como se han ido realizando de manera tradicional ya que permite la combinación de tecnologías en las aulas con métodos tradicionales. La manera que tienen de aprender con DEDOS reúne por un lado el punto de diversión, sin llegar a ser una herramienta puramente lúdica como puede ser un juego educativo, y por otro lado reúne el carácter didáctico de las clases tradicionales pero vistas desde un punto de vista más ameno, lo que favorece el aprendizaje de los alumnos enormemente.

Uno de los aspectos positivos que tiene DEDOS es la capacidad de personalización que proporciona a las clases. De esta manera si el profesor detecta que cierta materia está costando a los alumnos, éste de una manera mucho más amena podrá preparar diferentes actividades con DEDOS-Editor[1] con el fin de focalizar en los puntos débiles de los alumnos.

A lo largo del desarrollo del proyecto DEDOS, los distintos miembros que lo conforman han ido desarrollando experiencias educativas en centros educativos. Estas experiencias se han realizado con alumnos de distintos niveles, desde Educación Infantil y Primaria hasta alumnos con necesidades educativas especiales. Todas las experiencias han sido muy positivas, mejorando el ambiente del aula y demostrando que el aprendizaje con este tipo de tecnologías tiene una mayor calidad, produciéndose así una mejora significativa en el aprendizaje de los niños que utilizan métodos tecnológicos combinados con tradicionales frente a los que lo hacen exclusivamente de la manera tradicional.

A continuación se detallan algunas de las experiencias educativas realizadas hasta la fecha, exponiendo el caso práctico que se llevó a cabo y las conclusiones obtenidas del estudio:

- Cristina Fernández Gaullés (Cristina Fernández, 2014) realizó una experiencia educativa con niños de educación infantil en el que se pretendía demostrar la influencia de las tecnologías y en concreto de las pizarras digitales interactivas (PDI). En esta experiencia educativa participaron niños de 3 años del C.P María Moliner centrándose en actividades de “Conocimiento del Entorno” y Comunicación y Representación. Como conclusión se obtuvo que existe una influencia en el uso de las tecnologías durante el proceso de aprendizaje de los niños en la etapa de Educación Infantil, ya que en dicho estudio se

[1] Enlace de descarga: http://aprendecondedos.es/descargarte/aplicacion/

observó una mejora en el aprendizaje de aquellos niños que interactuaron con la PDI y por el contrario no se observó una evolución en los niños que no lo utilizaron.

- Ana Márquez Fernández (Ana Márquez, 2013) realizó un estudio en el colegio Leo Kanner con niños con TEA (Trastorno de Espectro Autista). En este estudio se utilizaron mesas multicontacto trabajando con actividades centradas en el reconocimiento de expresiones faciales, comprensión de sentimientos e interpretación de situaciones de la vida cotidiana. Como conclusión se obtuvo que no hay ningún inconveniente al utilizar este tipo de tecnologías en personas con TEA ya que se veían atraídos por ellas captando su curiosidad y mejorando su aprendizaje.
- María Rodríguez González (María Rodríguez, 2013) realizó un estudio centrado en alumnos de Educación Primaria para la asignatura de Matemáticas y Lengua Castellana y Literatura. Con este estudio se pretendía ver la influencia positiva o negativa en la adquisición de nuevos conocimientos utilizando las nuevas tecnologías con actividades diseñadas para PDI. Como conclusión se obtuvo que los alumnos que estuvieron en contacto con las nuevas tecnologías tuvieron mejores resultados que los que participaron en una clase tradicional.

A lo largo de todas las experiencias educativas se pudo observar lo siguiente:

- Uno de los aspectos que se consiguió de una mejor manera con la PDI fue captar la atención del alumnado más fácilmente, lo que impacta de una manera muy positiva en el aprendizaje.
- Los alumnos se adaptaron perfectamente a las nuevas tecnologías ya que se encuentran en contacto diario con ellas.
- Los docentes valoraron muy positivamente todas las experiencias educativas realizadas.
- La capacidad de personalización mediante la herramienta de DEDOS-Editor permite al profesor adaptar las clases a las necesidades de sus alumnos.
- Los dispositivos táctiles facilitan la manipulación de los elementos en pantalla al eliminar elementos intermedios como el ratón o el teclado.
- La combinación de elementos tradicionales con nuevas tecnologías favorece la concentración del alumno.
- Hay que controlar el número de elementos multimedia que se incluyen en las actividades para conseguir que los alumnos no se distraigan fácilmente.

1.2 ¿Por qué tabletas?

Actualmente, muchos centros educativos están invirtiendo presupuesto en dispositivos móviles como las tabletas, en concreto la mayoría de los centros están optando por tabletas con el sistema operativo Android debido principalmente al precio de las mismas. Es por ello que nace esta adaptación de DEDOS-Player a tabletas con sistema operativo Android con el fin de facilitar su incorporación al mundo docente. Además el uso de tabletas genera una serie de ventajas (Plataforma Proyecta, 2014) que no podemos dejar a un lado:

- Motivación: las tabletas proporcionan al alumno un elemento motivacional clave, y es que al tratar con ellas no generara el mismo rechazo que elementos tradicionales como son los libros. Por lo tanto el alumno estará más motivado a realizar las actividades que se le planteen y mejorará sus habilidades dentro y fuera del aula (Cristian Guzmán, 2015).
- Facilidad de uso: hoy en día las nuevas generaciones no sienten ningún rechazo al usar estos dispositivos electrónicos, sino todo lo contrario, en multitud de ocasiones les resulta más complicado realizar diversas tareas de manera tradicional. Es por ello que es idóneo poner al alcance de los alumnos distintas herramientas software de calidad con el fin de asentar los conocimientos necesarios. Además, al manipular directamente los elementos en pantalla, facilita enormemente el uso para aquellas personas que tengan ciertas necesidades especiales como pueden ser los niños o usuarios con discapacidad cognitiva (Ana Márquez, 2013).
- Autonomía: al tener esa facilidad de uso el alumno podrá llevar a cabo las actividades planteadas por el profesor sin necesidad de alguien que le guíe, otorgándole un mayor control y responsabilidad sobre su propio aprendizaje.
- Comodidad: el alumno no está limitado al aprendizaje en un pupitre, sino que éste puede llevar a cabo las actividades en prácticamente cualquier lugar, lo que elimina barreras a la hora de permitir que el alumno expanda sus conocimientos. Este concepto se conoce entre la comunidad educativa como ubicuidad.
- Flexibilidad: el uso de las tabletas proporciona una mayor flexibilidad en el aula, consiguiendo un aprendizaje más personalizado resultando clave para el desarrollo de los alumnos y en especial aquellos que tengan mayor dificultad en el aprendizaje, ya que se sentirán más integrados con el resto de la clase.

1.3 ¿Por qué Android?

En la actualidad el sistema operativo para tabletas que domina el mercado es sin duda alguna Android. Según las estimaciones del International Data Corporation (International Data Corporation, 2015), este panorama se mantendrá en el tiempo, solo perdiendo una ligera de cuota de mercado Android y iOS en favor de las tabletas con sistema operativo Windows (véase la Tabla 1).

Operating System	2014 Shipment Volumes	2014 Market Share	2015* Shipment Volumes	2015* Market Share	2019* Shipment Volumes	2019* Market Share
Android	154.7	67.3%	158.1	67.4%	169.5	62.9%
iOS	63.4	27.6%	60.1	25.6%	61.9	23.0%
Windows	11.6	5.1%	16.3	7.0%	38.0	14.1%
Total	**229.7**	**100.0%**	**234.5**	**100.0%**	**269.4**	**100.0%**

Tabla 1: Comparativo del uso de sistemas operativos en tabletas. Fuente IDC (International Data Corporation, 2015)

Debido a la mayor popularidad de este sistema operativo para tabletas, se ha considerado acertado realizar la adaptación de DEDOS para Android, con el fin de garantizar un mayor acceso a futuros usuarios. Para comprender el porqué de este dominio del mercado por parte de este sistema operativo se considera necesario analizar diferentes puntos a favor que le alzan como el sistema operativo más usado en la actualidad[2]:

- El precio es uno de los puntos a favor que tiene Android frente a su competidor directo Apple. Esta diferencia de precio hace que un mayor número de personas puedan acceder a este tipo de tecnologías. En nuestro caso este aspecto es fundamental ya que por norma general los colegios no cuentan con amplio presupuesto para invertir en material tecnológico, y las tabletas Android cuentan con precios más asequibles en comparación con sus competidores.
- La cantidad de modelos que disponen de sistema operativo Android es mucho mayor a la de iOS. En cuanto a iOS solo se puede hablar de una marca, Apple, en el caso de Android está disponible en multitud de marcas como son Samsung, BQ, LG, o Sony, entre muchas otras. Esta cantidad de dispositivos hace que la gente pueda disfrutar de un mismo sistema

[2] Ventajas de Android: http://goo.gl/gSxkC4

operativo en el modelo que desee y que además se adapte a sus necesidades o gustos. A priori esto es un hándicap para el desarrollador, ya que la misma aplicación se visualizará en multitud de dispositivos de diferentes marcas que pueden tener distintos características, con lo que el desarrollo de la misma deberá ser mucho más cuidado y elaborado, realizando multitud de pruebas para garantizar el acceso desde cualquier dispositivo.

- Si algo diferencia a Android del resto es la cantidad de desarrolladores que tiene, lo que proporciona mayor apoyo en cuanto a foros, o diversos tutoriales que los desarrolladores más nóveles pueden consultar. La Figura 1 muestra una gráfica con la evolución anual en número de desarrolladores donde se puede ver un incremento notable en los últimos años dentro de la comunidad de Android[3].

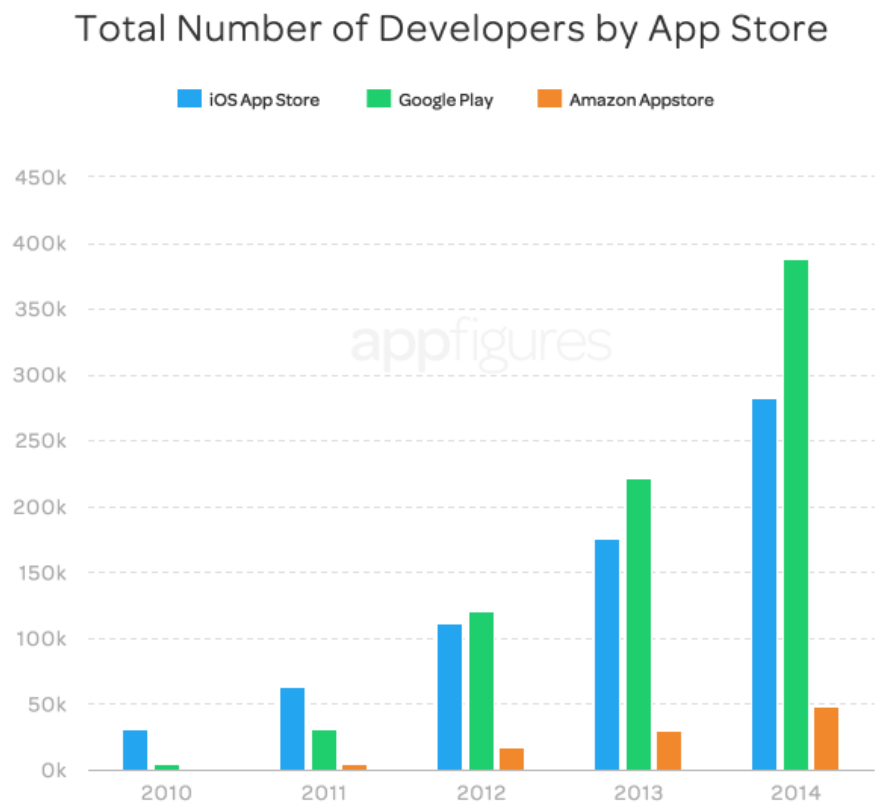


Figura 1: Número de desarrolladores por Sistema Operativo. Fuente: appfigures.

- Una de las grandes bazas a favor de Android es la gratuidad de las herramientas de desarrollo disponibles así como los diferentes manuales que proporcionan desde la página oficial, lo que permite a los desarrolladores sentirse más seguros a la hora de iniciar un desarrollo.

[3] Blog Appfigures: http://goo.gl/yI5KJp

- Otra de las ventajas es la libertad que otorga tanto a desarrolladores como a usuarios. Dentro de esta libertad podemos encontrar entre otras, la personalización, tanto el apartado visual como el apartado más puramente técnico para usuarios más avanzados.
- Especialmente importante a la hora de compartir un dispositivo, como puede ser en un colegio, es la capacidad de que el dispositivo sea multiusuario, hecho que está disponible a partir de Android 4.2 y que para sistemas operativos como iOS aún se desconoce si será posible disfrutar de esta característica en un futuro cercano. Este hecho es de gran importancia por ejemplo en colegios, donde puede ser interesante que un mismo dispositivo tenga diferentes sesiones de usuario con el fin de que pueda ser compartido entre varios alumnos de la clase o incluso de distintas clases.

1.4 Herramientas relacionadas

En la actualidad la mayoría de herramientas que existen para tabletas están más enfocadas al aprendizaje de una manera puramente lúdica, centrándose en el desarrollo de juegos interactivos y en los que el alumno puede aprender determinadas materias a la vez que juega sin supervisión por parte de docentes. Prueba de ello es que en el top de aplicaciones educativas para Android[4] sólo encontramos juegos, como los que se nombran a continuación:

- Toca Lab[5]: juego dedicado al aprendizaje de la ciencia mediante diversos mini juegos.
- Toca Nature[6]: juego dedicado al aprendizaje de la naturaleza a través de diversos mini juegos.
- El restaurante de Dr. Panda[7]: juego dedicado a introducir a los más pequeños en el mundo de la cocina.

Aun así hay algunas empresas como SMART con su tecnología Notebook[8] que permiten al docente llevar a cabo sus clases de una manera mucho más dinámica y personalizada. Esta tecnología se usa en muchos centros que disponen de pizarra digital interactiva para la realización de proyectos educativos en clase (SMART, 2016). La idea original de Smart Notebook es permitir al profesor la creación de contenidos interactivos que podrán ser utilizados en una superficie multicontacto, exportar a PDF o generar una página HTML para ser visualizada en cualquier dispositivo. El propósito de esta herramienta es conseguir que el

[4] Juegos más vendidos Play Store: https://goo.gl/eFr2Qe
[5] Toca Lab: https://goo.gl/etEikF
[6] Toca Nature: https://goo.gl/im4sNb
[7] El Restaurante del Dr Panda : https://goo.gl/hz4gLr
[8] Notebook de Smart: http://goo.gl/8eq4Sz

alumno se sienta más motivado en las clases teóricas, ya que permite adquirir diversos conceptos de una manera mucho más dinámica que en una clase tradicional.

Smart Notebook es una herramienta pensada para ser utilizada en superficies multicontacto, aunque debido a la demanda actual del mercado, tiene distintas adaptaciones para ser utilizada en tabletas.

A continuación se exponen una serie de diferencias o similitudes entre ambas herramientas.

- En Notebook no existe una categorización de actividades como en DEDOS con selección, emparejamiento o actividades de Matemáticas.
- DEDOS no soporta el uso de animaciones tal y como hace Notebook.
- Ambas han visto la necesidad de adaptar su proyecto a tabletas y no solo centrarse en pizarras digitales.
- En cuanto a la sencillez a la hora de crear actividades destacaría DEDOS, ya que al tener unos tipos predefinidos de actividades (selección, emparejamiento y matemáticas) permite al profesor centrarse en las mismas y definir mejor las actividades.
- A la hora de que el alumno reciba *feedback* por parte de la aplicación con DEDOS se consigue de una manera mucho más directa, apareciendo un *tick* verde o rojo con la respuesta, mientras que con Notebook no se consigue ese *feedback* tan directo y hace que el alumno dependa del profesor para saber las respuestas, mientras que con DEDOS es el alumno el que al contestar descubre si ha acertado o no. Con Notebook se puede conseguir este *feedback* pero no en todos los tipos de actividades y de la misma forma.

	DEDOS	NOTEBOOK
Feedback directo	x	
Categorización de actividades	x	
Material multimedia	x	x
Uso de animaciones		x
Sencillez en la creación	x	
Adaptado a varios dispositivos	x	x

Tabla 2. Comparativa DEDOS y Notebook

2 Objetivo

Este capítulo presenta el objetivo general de este Trabajo Fin de Grado (TFG) así como los objetivos específicos ligados con el mismo. También se exponen los conocimientos técnicos que son necesarios para poder realizar el trabajo.

2.1 Objetivo general

El objetivo principal de este TFG es la realización de una versión de la herramienta DEDOS-Player para tabletas con sistema operativo Android, todo ello con el fin último de que un mayor número de personas pueda acceder a esta tecnología debido a las dificultades técnicas, u económicas que puede suponer tener una mesa multicontacto o pizarra digital para los centros educativos.

El objetivo fundamental ha sido respetar la esencia original de la aplicación con el fin de que el usuario no perciba diferencias significativas al utilizar ambas aplicaciones destinadas para diferentes dispositivos. Además, es necesario que la versión que se realice para tabletas Android, interprete de forma correcta los proyectos generados con la aplicación DEDOS-Editor. De esta forma, se consigue que el destinatario final de la aplicación, realice las actividades educativas de los proyectos de una forma análoga independientemente del dispositivo que se esté usando.

La adaptación realizada en versión de tabletas ha sido destinada para que la juegue una única persona, por ello algunas de las características de la versión original no se encontrarán disponibles, como es la elección del número de jugadores en el menú de ajustes u opciones relacionadas con la posibilidad de realizar actividades de una forma colaborativa cuando los alumnos resuelven el proyecto educativo en superficies multicontacto como las mesas.

2.2 Objetivos parciales

La regla de oro de todo informático debería ser "*Divide y vencerás*" por lo que en este trabajo se procedió a descomponer el objetivo general en objetivos parciales con el fin de tener un resultado final satisfactorio. A continuación se detalla cada uno de objetivos parciales que se han ido desarrollando con el fin de conseguir el resultado final esperado.

- Había que analizar la herramienta original, con el fin de que no hubiera diferencias significativas en la jugabilidad entre ambas plataformas. Para llevar a cabo este objetivo se

tuvieron distintas sesiones de toma de requisitos con el equipo docente y desarrollador de la versión original de DEDOS-Player en las cuales se explicó cuál era la esencia original del proyecto DEDOS, y cuáles eran los requisitos funcionales y técnicos que se debían trasladar a la versión de tabletas.

- Dado que los usuarios finales iban a tener una edad significativamente inferior a la nuestra en primer lugar teníamos que comprender su mentalidad a la hora de estar en contacto con una tableta. Gracias a que hoy en día, las nuevas generaciones han nacido con la tecnología formando parte de ellos, pudimos observar que estos no tienen ningún tipo de problema ni barrera mental que les supusiese un impedimento a la hora de utilizar DEDOS-Player para tabletas.
- Con todo lo anterior analizado se desarrollaron los primeros prototipos funcionales para garantizar que la adaptación tuviera la misma funcionalidad que la aplicación original.
- Además de la funcionalidad otro requisito que debía tener la aplicación era respetar la interfaz gráfica original, para ello se modificaron los primeros prototipos a los que se agregó la estética original de DEDOS-Player.
- Como la aplicación original tenía distintos tipos de actividades: selección, emparejamiento y matemáticas, se fue desarrollando cada una de manera individual hasta conseguir la versión completa de tabletas integrada donde los alumnos pueden realizar actividades educativas combinadas de los tres tipos soportados por DEDOS-Player.
- A lo largo de todo el proceso se iría mostrando los resultados parciales a los usuarios finales con el fin de detectar posibles incoherencias o problemas de funcionalidad.

2.3 Herramientas conceptuales necesarias

A continuación se detallan cada uno de los conocimientos que han sido necesarios para la elaboración de este TFG y que se encuentran relacionados con la titulación de Ingeniería Informática.

2.3.1 Programación Orientada a Objetos

La base fundamental sobre la que se construye este desarrollo es la "*Programación Orientada a Objetos*", la cual proporciona al programador múltiples ventajas a la hora de desarrollar cualquier aplicación. Entre estas múltiples ventajas se pueden destacar las siguientes:

- Reusabilidad: si se realiza una buena estructuración de las clases, éstas pueden ser fácilmente reutilizables en cualquier parte de nuestra aplicación. En proyectos pequeños

apenas se aprecia la ventaja de la reusabilidad, pero a medida que el proyecto va adquiriendo un mayor tamaño, como es en el caso de este TFG, las ventajas son tan significativas que merece la pena invertir esfuerzo en una fase de análisis previa a la fase de desarrollo.

- Mantenibilidad: debido a la sencillez en la estructuración de la aplicación, ésta es más fácil de comprender y de leer facilitando su comprensión.
- Actualización: al estar todo más modularizado se pueden realizar modificaciones con mayor facilidad sin necesidad de modificar el concepto general de la aplicación.
- Fiabilidad: gracias a la estructura en módulos, es más fácil detectar posibles errores al realizar cambios en distintas partes de la aplicación. Además al reutilizar en la mayoría de ocasiones código a través de librerías, nos garantiza la seguridad que la mayoría han sido ampliamente testeadas por la comunidad de desarrolladores.

Dentro de la Programación Orientada a Objetos encontramos como pilares fundamentales los siguientes conceptos que se han ido viendo a lo largo de todo el desarrollo del TFG:

- Clase: define el conjunto de propiedades y el comportamiento que tendrá un determinado objeto.
- Objeto: instancia de una clase que tendrá las propiedades y el comportamiento definido en la misma.
- Método: operación definida dentro de una clase que marca el comportamiento de un determinado objeto.
- Atributo: características que tendrá la clase.
- Herencia: es la capacidad que tienen las clases de relacionarse unas con otras, pudiendo un objeto heredar características y comportamiento de una clase padre.
- Polimorfismo: capacidad de una entidad de tener valores diferentes durante la propia ejecución del programa.
- Encapsulamiento: es la capacidad de ocultar las propiedades internas de una entidad, las cuales solo podrán ser modificadas mediante la aplicación de diversas operaciones destinadas a tal fin.

2.3.2 Android

A pesar de haber tenido una variedad importante de lenguajes, como HTML5 entre otros, para desarrollar la aplicación se eligió Android (Java) al ser el más comúnmente utilizado en tabletas con sistema operativo Android y ser el más respaldado por la comunidad de desarrolladores tal y como se ha comentado previamente en el capítulo de Introducción. Esto nos facilitaría el

desarrollo de cara a la multitud de tutoriales o guías que encontramos para resolver los problemas que nos íbamos encontrando. Además, otro aspecto que se consideró cuando se inició este trabajo para seleccionar la tecnología a usar, fue el rendimiento de la aplicación dependiendo de la tecnología que se usara. Se hicieron unas pruebas de prototipos con HTML5 para que la aplicación funcionara tanto en dispositivos iOS como en dispositivos Android. Sin embargo, no se obtuvieron los resultados esperados ya que en el caso de actividades de emparejamiento, el tiempo que tardaban las tabletas Android en seguir los objetos que estaban siendo arrastrados por la pantalla no era inmediato y simultáneo al movimiento. Debido a que los usuarios del proyecto DEDOS son alumnos de etapas muy tempranas o con necesidades educativas especiales, se descartó en su momento esta tecnología puesto que no se obtenían resultados apropiados y que podían interferir en el proceso de aprendizaje de los alumnos.

Dentro de la programación en Android se incluyen diversos términos que se han tenido que comprender para un correcto desarrollo de la aplicación. Para el conocimiento de dichos términos se han tenido que utilizar diversas webs con tutoriales o libros debido a que el grado de Ingeniería Informática no dispone de una asignatura dedicada a la programación en Android. A continuación se procede a explicar brevemente cada uno de estos conceptos:

- SDK: es un *kit* de desarrollo software, en nuestro sería necesario el SDK específico para Android.
- Fichero *apk*: es el fichero generado tras la compilación del código desarrollado. Generalmente es generado a través de un entorno de desarrollo.
- *Activity*: es el componente de la aplicación encargado de mostrar las interfaces de usuario para que el usuario sea capaz de comunicarse con la aplicación a través de distintos eventos como puede ser pulsando un botón. Cada pantalla diseñada en la aplicación tendrá su propia clase que herede de la clase *Activity*, pudiendo así llamar o redefinir los métodos de esta.
- *View*: serán los componentes básicos que compondrán nuestra interfaz gráfica.
- *Intent*: componente de una aplicación que permite la comunicación entre dos componentes. Generalmente es utilizado para pasar de una *Activity* a otra.
- *TextView*: es un objeto que hereda de la clase *View* y permite mostrar en pantalla un determinado texto.
- *ImageView*: es un objeto que hereda de la clase *View* y permite mostrar en pantalla una determinada imagen.

- *Toast*: elemento de una actividad que permite mostrar un determinado *feedback* al usuario. En nuestro caso ha sido utilizado para dar retroalimentación al usuario a la hora de acertar o fallar las diferentes actividades.
- *Layout*: elementos no visuales que son utilizados para estructurar todos los elementos que aparecen en la pantalla de la aplicación.
- *Drag and drop*: es una acción por la cual un usuario podrá arrastrar un elemento y depositarlo en cualquier otra parte de la interfaz. Esta propiedad ha sido utilizada en nuestra aplicación para las actividades de emparejamiento y matemáticas.
- *Event Listener*: capturador de todos los eventos generados por el usuario al interactuar con la aplicación.

2.3.3 Entorno de desarrollo

Para el desarrollo de la aplicación se requieren nociones básicas sobre la utilización de un entorno de desarrollo con el que se ha llevado a cabo el desarrollo, compilación y ejecución de la aplicación. En nuestro caso se ha utilizado Eclipse para el desarrollo de la aplicación en Android.

2.3.4 Interacción Persona Ordenador

Todo desarrollo software tiene que tener como prioridad que los usuarios que vayan a utilizar la aplicación se sientan cómodos con la misma y el esfuerzo que inviertan en aprender a usarla les merezca la pena. Por tanto, el objetivo fue que la aplicación DEDOS-Player para tabletas fuera usable por los destinatarios de la aplicación.

Para analizar el comportamiento de los usuarios al utilizar DEDOS-Player para tabletas se llevará a cabo una evaluación heurística que evaluará distintos aspectos que la aplicación deberá cumplir. Esta evaluación es totalmente necesaria y nos permitirá verificar la usabilidad de nuestra aplicación y detectar posibles errores o mejoras. Esta evaluación se tratará en profundidad en los siguientes capítulos.

3 Solución tecnológica

En este capítulo se elaborará una descripción detallada de la solución tecnológica implementada con el fin de cumplir el objetivo principal de este TFG. Se empezará describiendo tanto el lenguaje como las herramientas usadas en el desarrollo, seguido de la explicación a nivel funcional y técnico de los diversos apartados de la aplicación desde el tratamiento del fichero generado con DEDOS-Editor hasta las diferentes casuísticas que se desarrollan en el funcionamiento de las distintas actividades.

3.1 Lenguaje utilizado

La mayoría de la gente hoy día desconoce que el hecho de quieras desarrollar una aplicación para el sistema operativo Android no tiene por qué ir relacionado con desarrollarla en lenguaje Android, sino que hay otras múltiples alternativas que pueden ser elegidas en función de tus conocimientos previos o tus preferencias.

A continuación se describe brevemente algunas de estas posibles alternativas:

- Basic4Android: sin duda alguna es una de las grandes plataformas competidoras ya que pone a su disposición la creación de aplicaciones para Android con el lenguaje Visual Basic más simple para personas que se inicien en el mundo de la programación.
- Ruboto: surge a raíz del auge ocasionado por el lenguaje de programación Ruby permitiendo al programador reducir las horas de desarrollo, debido a que los programas en Ruby requieren menos líneas de código que otros lenguajes de programación para realizar tareas similares.
- Mono: hay muchas personas que sienten rechazo a todo lo que tenga que ver con el lenguaje de programación Java, debido a sus enormes críticas en cuanto a su eficiencia. Para estas personas existe la alternativa de Mono, que permite elaborar aplicaciones para Android utilizando C# y .NET.
- App Inventor: aunque debido a su sencillez la convierte en la menos potente esto la da cierta ventaja con el resto debido a que no es necesario que la persona que esté desarrollando la aplicación tenga ningún conocimiento sobre lenguajes de programación ya que se trata de desarrollar una aplicación utilizando solo interfaz gráfica sin necesidad de escribir ninguna línea de código.

- Icenium: plataforma en la nube que permite al desarrollador olvidarse de instalación de librerías o entornos de desarrollo, todo ello mediante el uso de tecnologías como HTML5 o CSS.

Aun con todo lo anterior, para la realización de este TFG, se ha optado por usar Android de manera nativa por las siguientes razones:

- Es el lenguaje oficial propuesto por Google.
- Existen herramientas de desarrollo más completas como es el caso de Eclipse ADT.
- Hay una multitud de librerías disponibles para Java que nos facilitan la realización de cualquier tarea.
- El soporte al desarrollo es completo gracias a la extensa documentación por ser la opción más utilizada.

3.2 Herramientas tecnológicas utilizadas

En esta sección se describirá de manera breve cada una de las herramientas utilizadas en toda la elaboración de este TFG, desde herramientas de software hasta blogs o diversas webs de consulta. Las herramientas software que se han usado han sido las siguientes:

- Entorno de desarrollo *Eclipse IDE for Java Developers* para llevar a cabo el desarrollo del código de la aplicación. Actualmente existen alternativas como el propio Android Studio de Google que al inicio del proyecto carecía del apoyo suficiente entre la comunidad de desarrolladores por lo que se decidió tomar Eclipse como entorno de desarrollo para este TFG.
- *Android Development Tools* son las herramientas necesarias para desarrollar en el entorno de Eclipse.
- *Java SE Development Kit (JDK)* como componente necesario para desarrollar aplicaciones cuya base es Java.
- Dropbox: se ha utilizado esta herramienta para el almacenamiento de toda la información relativa al proyecto así como para guardar el propio código generado en el desarrollo. Con el siguiente enlace se podrá descargar el proyecto: https://goo.gl/QgO3Oc

En cuanto a las comunidades de desarrolladores utilizadas, se ha hecho uso de:

- *Stackoverflow* (http://stackoverflow.com): sitio web para desarrolladores informáticos en el que se puede encontrar solución a distintos problemas relacionados con la

programación. Para este TFG se han consultado diferentes problemáticas que han ido ocurriendo a lo largo del desarrollo.

- *Github* (https://github.com): sitio web de desarrollo colaborativo y de publicación de código abierto. Para TFG esta web ha sido utilizada para descarga de ejemplos de desarrollo de Android y para el almacenamiento del proyecto que se podrá obtener del siguiente enlace: https://github.com/omartinma/DEDOS-Reproductor.

Además, se han usado dos blogs especializados en el desarrollo de aplicaciones informáticas para dispositivos Android como son:

- http://www.sgoliver.net/: web en español con mini-tutoriales acerca de los componentes de un desarrollo en Android.
- http://www.javaya.com.ar/androidya/: web en español con mini-tutoriales acerca de los componentes de un desarrollo en Android.

3.3 Funcionamiento de la aplicación

A lo largo de esta sección se explicará de manera detallada todo el funcionamiento de la aplicación desde el tratamiento del fichero XML generado previamente con la herramienta DEDOS-Editor hasta el funcionamiento de todos los tipos de actividad que soporta la versión de DEDOS-Player para tabletas: selección, emparejamiento y matemáticas.

Dentro de cada apartado se comentará cuáles son las librerías utilizadas o los métodos principales que han sido utilizados para que nuestra aplicación cumpla con su propósito.

3.3.1 Tratamiento del fichero generado con DEDOS-Editor

El concepto del proyecto DEDOS es que el propio profesor crea los juegos para sus alumnos de manera independiente con la herramienta DEDOS-Editor. Esta herramienta una vez llevado a cabo el diseño del juego como formato de salida genera un fichero XML y una carpeta que contiene las imágenes asociadas que se usan en el proyecto.

Para llevar a cabo el reconocimiento del fichero XML se eligió realizar el parseo de dicho documento XML a través de la librería JDOM[9]. Esta librería está diseñada específicamente para el tratamiento de documentos XML en Java pudiendo realizar parseo, búsqueda, modificación o generación de documentos XML.

[9] Web de descarga de la librería JDOM: http://www.jdom.org/downloads/

El funcionamiento de esta librería es sencillo, a medida que se parsea el documento XML se va generando un árbol de nodos fácilmente recuperables con el objeto `Element` que tendrá un nombre asociado a la etiqueta del documento XML. Es por eso que existe un alto grado de dependencia entre la sencillez en la realización del parseador y la buena organización en la generación del documento XML.

A continuación se detalla cómo se realiza el parseo del documento XML generado previamente. En primer lugar se construirá el objeto de tipo `XMLParser` a través de su constructor que recibirá un objeto de tipo `String` que contiene la ruta del fichero XML a parsear:

```
XMLParser xml = new XMLParser(xmlPath);
```

Este constructor realiza tres operaciones:

1. Se llama al método `generateDocument` que recibe la ruta como tipo `String` y devuelve un objeto tipo `Document` necesario para realizar el parseo del fichero (véase la Figura 2)

```
public XMLParser(String path) throws IOException{
        docXML = generateDocument(path);
}
```

Figura 2 Código del constructor XMLParser

2. Dentro del método `generateDocument` se construirá un objeto de tipo `SAXBuilder` que nos servirá para construir el objeto de tipo `Document` (véase la Figura 3).

```
private static Document generateDocument(String path){
        Document document = null;
        try {
            SAXBuilder parser = new  SAXBuilder();
            document = parser.build(new File(path));
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        } catch (JDOMException e) {
            Log.e(TAG, e.getMessage());
        }
        return document;
    }
```

Figura 3 Método generateDocument

3. Una vez "parseado" el documento nos quedará analizar cada una de los nodos que tiene. Debido a la complejidad de este apartado, se decidió ir subprogramando los métodos que realizaban el análisis de manera descendente, es decir, desde los nodos menos profundos (padres) a los nodos más profundos (hijos), según la estructura del fichero XML generado.

En el método cuyo código se presenta en la Figura 4 se analizan los primeros nodos. Según las estructura del fichero XML formado en DEDOS-Editor los primeros nodos podrán corresponder a los tipos mostrados en la Figura 5 y que son los siguientes:

- `Resolution`: muestra la resolución predeterminada en la que se hizo el proyecto, y que servirá para que se pueda adaptar a todos los dispositivos.
- `Activity`: muestra una actividad. El proyecto educativo que contiene el fichero XML estará formado por una o más actividades.

```
public boolean parseGame(String gamePath, String xmlPath, List<Game> game) {
    try {
        this.gamePath = gamePath;
        XMLParser xml = new XMLParser(xmlPath);
        for (Element root : xml.getElements()) {
            if (root.getName().equals(IAppKeys.ATT_RESOLUTION)) {
                Double x = Double
                        .valueOf(root.getAttribute("x").getValue());
                Double y = Double
                        .valueOf(root.getAttribute("y").getValue());
                d = new Dimension();
                d.setSize(x, y);

            } else if (root.getName().equals(IAppKeys.NODE_ACTIVITY)) {
                newActivity(root.getChildren(), game);
            }
        }
        return true;
    } catch (IOException ex) {
        Log.e(TAG, ex.getMessage());
        return false;
    }
}
```

Figura 4 Método parseGame para los primeros nodos

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
  <Activity>
  <Activity>
  <Activity>
</Project>
```

Figura 5 Primeros nodos XML

A su vez dentro de cada `Activity` tendremos diferentes elementos que conformarán el diseño de nuestra aplicación. Estos elementos son tratados en el método que se muestra en la

Figura 6. A continuación se enumeran los nodos que conformarán una `Activity` mostrados en la Figura 7.

- `Objectives`: contendrá la información relativa al tipo de actividad que se va a desarrollar. En nuestro caso nos informará si el tipo de actividad es de selección, emparejamiento o matemáticas. Dentro de cada tipo de actividad nos podemos encontrar con diferentes atributos
- `Tokenlist`: contendrá aquellos objetos que el profesor a la hora de crear la actividad no los situó en un área determinada.
- `Arealist`: contendrá las áreas en las que se descompone la actividad y que contendrá los diferentes `tokens`.

```
private void newActivity(List<Element> children, List<Game> game) {
        Game activity = new Game();
        for (Element actElem : children) {
            if (actElem.getName().equals(IAppKeys.NODE_OBJECTIVES)) {
                activity = findGameType(actElem.getChildren());
                System.out.println(activity.getActivityClass().toString());
                activity.setObjectives(getObjectivesFromElement(actElem
                        .getChildren()));
            } else if (actElem.getName().equals(IAppKeys.NODE_AREALIST)) {
                activity.setAreaMap(getAreasFromElemen(actElem.getChildren()));

            } else if (actElem.getName().equals(IAppKeys.NODE_TKLIST)) {
                activity.setTokenList(getTokensFromElement(children));
            }
        }
        activity.setResolucion(d);
        game.add(activity);
    }
```

Figura 6 Método parseo nodos dentro de Activity

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
    <Tokenlist/>
    <Arealist>
  </Activity>
</Project>
```

Figura 7 Nodo Activity XML

Dentro del nodo `Objectives` tendremos la definición del tipo de Actividad que estamos parseando actualmente y los objetivos que se deberán cumplir para dar por concluida la actividad tal y como se observa en la Figura 8.

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
      <obj type="sel" obj="instance1923"/>
    </Objectives>
    <Tokenlist/>
    <Arealist>
  </Activity>
</Project>
```

Figura 8 Nodo Objectives XML

En este ejemplo anterior tenemos un objetivo tipo "*sel*" que corresponde a una actividad de tipo selección.

Dentro de cada objetivo para parsear su tipo se realiza lo siguiente:

1. En primer lugar obtenemos el tipo de Juego con el método `findGameType`. Para ello se irá comprobando qué valor tiene el atributo `type` para el objetivo tal y como se observa en la Figura 9.

```
private Game findGameType(List<Element> objectives) {
        for (Element obj : objectives) {
            if (obj.getAttribute(IAppKeys.ATT_TYPE) == null) {
                continue;
            }
            if (obj.getAttributeValue(IAppKeys.ATT_TYPE).equals(
                    IAppKeys.TYPE_SEL)) {
                return new SelectionGame();
            } else if (obj.getAttributeValue(IAppKeys.ATT_TYPE).equals(
                    IAppKeys.TYPE_PAIR)) {
                if (obj.getAttributeValue(IAppKeys.TYPE_MATH).equals("false"))
                    return new PairGame();
                else
                    return new MathGame();
            } else if (obj.getAttributeValue(IAppKeys.ATT_TYPE).equals(
                    IAppKeys.TYPE_MATH)) {
                return new MathGame();
            }
        }
        return new Game();
    }
}
```

Figura 9 Método findGameType

2. En segundo lugar en función del tipo de Juego construimos unas propiedades u otras para cada tipo de objetivo. Tendremos los objetivos para selección, emparejamiento, matemáticas, y para el objetivo de tiempo, el cual se ha parseado debido a que viene definido en la estructura del XML ya que en la versión de mesas multicontacto se tiene objetivos de tiempo al ser un juego colaborativo y por turnos en el caso de que el profesor así lo decida.

En el caso de actividades de selección se tendrá dentro del nodo `Objectives` un nodo por cada una de las respuestas que tengamos que seleccionar. Este nodo tendrá un ID que será el que almacenemos en nuestro objetivo. El tratamiento de este tipo de actividad se observa en la Figura 10.

```
/* Objetivo de Seleccion */
if (objective.getType().equals(IAppKeys.TYPE_SEL)) {
    objective.setIdDestino(objElem
            .getAttributeValue(IAppKeys.NODE_OBJ));
```

Figura 10 Tipo de actividad: Selección

En las actividades de emparejamiento tendremos que almacenar la información para el nodo origen, es decir, la tarjeta con la que iniciamos el emparejamiento, y la información para los

nodos destino, ya que una misma tarjeta podrá ser depositada en varias tarjetas si así lo requiere el juego. El tratamiento de este tipo de actividad se observa en la Figura 11.

```
/* Objetivo de Emparejamiento */
} else if (objective.getType().equals(IAppKeys.TYPE_PAIR)) {
    objective.setIdOrigen(objElem
            .getAttributeValue(IAppKeys.ATT_ORIGEN));
    if (objElem.getAttribute(IAppKeys.ATT_DESTINO) != null) {
        Target target = new Target(objElem.getAttributeValue(IAppKeys.ATT_DESTINO));
        objective.getTargets().add(target);

    } else {
        for (Element targets : objElem
                .getChildren(IAppKeys.NODE_TARGETS)) {/
            for (Element targ : targets.getChildren()) {/
                Target t = new Target(targ.getAttributeValue(IAppKeys.ATT_NAME));
                objective.getTargets().add(t);
            }

        }
    }
```

Figura 11 Tipo de juego: Emparejamiento

En el caso de objetivos de matemáticas sería similar al caso de los objetivos de emparejamiento pero con la casuística de que cada tarjeta tendrá un valor asociado que se deberá almacenar para poder resolver la actividad de suma planteada a los alumnos. El tratamiento de este tipo de actividad se observa en la Figura 12.

```
/* Objetivo de Matematicas */
} else if (objective.getType().equals(IAppKeys.TYPE_MATH)) {
    objective.setTargetValue(Integer.parseInt(objElem
            .getAttributeValue(IAppKeys.ATT_VALUE)));
    int valor = Integer.parseInt(objElem
            .getAttributeValue(IAppKeys.ATT_VALUE));
    objective.setIdDestino(objElem
            .getAttributeValue(IAppKeys.ATT_ID));

    for (Element elem : objElem.getChildren()) {
        if (elem.getName().equals(IAppKeys.NODE_ORIGIN_TOKENS)) {
            for (Element originToken : elem.getChildren()) {
                objective.getOriginTokens().add(originToken.getAttributeValue(IAppKeys.ATT_ID));
            }
        } else if (elem.getName()
                .equals(IAppKeys.NODE_ORIGIN_ZONES)) {
            for (Element originZone : elem.getChildren()) {
                objective
                        .getOriginZones()
                        .add(originZone
                                .getAttributeValue(IAppKeys.ATT_ID));
            }
        }
    }
}
```

Figura 12 Tipo de juego: Matemáticas

En el caso de objetivos de tiempo, como hemos comentado con anterioridad, lo tomaremos en cuenta sólo de cara al parseo, por si se decidiera en un futuro incluir versión colaborativa. El tratamiento de este tipo de objetivo se observa en la Figura 13.

```
/* Objetivo de Tiempo(Viene en el XML, pero no aplica en versión tablets) */
} else if (objective.getType().equals(IAppKeys.ATT_TIME)) {
    objective.setDone(true);
    objective.setTimer(Double.valueOf(objElem
            .getAttributeValue(IAppKeys.ATT_TIME)));
```

Figura 13 Objetivo de tiempo

A continuación se comentará en profundidad el contenido del elemento `Arealist`, que contiene toda la información relativa a los diferentes elementos que hay en la actividad, y la definen visualmente.

En primer lugar el `Arealist` contiene una lista de áreas tal y como se aprecia en la Figura 14, principalmente serán dos, una para la parte del Juego, en la cual el alumno podrá ver la pregunta asociada a la actividad, y por otro lado nos encontramos con la parte del Alumno, en la que él mismo tendrá las diferentes opciones para completar la actividad.

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
    <Tokenlist/>
    <Arealist>
      <Area id="instance379" type="Jugador">
      <Area id="instance230" type="Juego">
    </Arealist>
  </Activity>
</Project>
```

Figura 14 Nodo Arealist XML

La estructuración de las áreas es prácticamente idéntica, por lo que se procede a mostrar una de las áreas modo de ejemplo en la Figura 15. Para el tratamiento de cada `Area` se ha utilizado el método que aparece en la Figura 16. Dentro de cada una se pueden encontrar distintas propiedades que se enumeran a continuación:

- `Pos`: posición del área dentro de la pantalla en base a las dimensiones predeterminadas previamente.

- `Rotation`: se muestra un valor si se quiere indicar que el área se encuentra con rotación.
- `Posfondo`: se podrá definir la posición de la imagen que el usuario establezca como fondo.
- `Size`: valores para el tamaño del área dentro de la pantalla en base a las dimensiones predeterminadas previamente.
- `Bg`: nombre asociado a una imagen que se quiere mostrar como fondo en el área.
- `Tokenlist`: podrá haber diferentes listas de `tokens` o elementos que componen un área de juego.

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
    <Tokenlist/>
    <Arealist>
      <Area id="instance379" type="Jugador">
        <pos x="10.25" y="277.1"/>
        <rotation value="0"/>
        <posfondo x="0" y="0"/>
        <size height="343.85" width="988.15"/>
        <bg url="fondopastel.jpg"/>
        <Tokenlist>
        <Tokenlist>null</Tokenlist>
      </Area>
      <Area id="instance230" type="Juego">
    </Arealist>
  </Activity>
</Project>
```

Figura 15 Nodo Area XML

```
private List<Area> getAreasFromElemen(List<Element> children) {
    List<Area> list = new ArrayList<Area>();
    for (Element areaElem : children) {
        if (areaElem.getName().equals(IAppKeys.NODE_AREA)) {
            Area area = new Area();
            area.setId(areaElem.getAttributeValue(IAppKeys.ATT_ID));
            area.setType(areaElem.getAttributeValue(IAppKeys.ATT_TYPE));
            area.setFondo(gamePath
                    + "/contents/"
                    + areaElem.getChild(IAppKeys.ATT_FONDO)
                            .getAttribute("url").getValue());
            //Calculamos la posición
            double x = Double.valueOf(areaElem.getChild(IAppKeys.NODE_POS)
                    .getAttribute("x").getValue());
            double y = Double.valueOf(areaElem.getChild(IAppKeys.NODE_POS)
                    .getAttribute("y").getValue());
            Point pos = new Point(x, y);
            area.setPos(pos);

            //Exploramos los tokens del area
            area.setTokenList(getTokensFromElement(areaElem.getChildren()));

            //Calculamos las dimensiones del área
            double width = Double.parseDouble(areaElem
                    .getChild(IAppKeys.NODE_SIZE).getAttribute("width")
                    .getValue());
            double height = Double.parseDouble(areaElem
                    .getChild(IAppKeys.NODE_SIZE).getAttribute("height")
                    .getValue());
            Dimension d = new Dimension();
            d.setSize(width, height);
            area.setSize(d);
            //Agregamos el área a la lista
            list.add(area);
        }
    }
    return list;
}
```

Figura 16 Método parseo de áreas

Dentro de cada `Tokenlist` nos encontramos con los diferentes `Tokens` o elementos de los que está compuesta la actividad, y en concreto el área. Esta lista podrá tener de 0 a n elementos, en función de la definición que estime oportuna el profesor. En el ejemplo que mostraremos a continuación en la Figura 17 observamos como tenemos dos listas, una que contiene 3 `Tokens` y una que no contiene ninguno.

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
    <Tokenlist/>
    <Arealist>
      <Area id="instance379" type="Jugador">
        <pos x="10.25" y="277.1"/>
        <rotation value="0"/>
        <posfondo x="0" y="0"/>
        <size height="343.85" width="988.15"/>
        <bg url="fondopastel.jpg"/>
        <Tokenlist>
          <Token id="instance1724" type="txt" numValue="1">
          <Token id="instance11555" type="txt" numValue="1">
          <Token id="instance1923" type="txt" numValue="1">
        </Tokenlist>
        <Tokenlist>null</Tokenlist>
      </Area>
      <Area id="instance230" type="Juego">
    </Arealist>
  </Activity>
</Project>
```

Figura 17 Nodo Tokenlist XML

Un `Token` es el elemento último en el árbol de nodos. Éste tendrá diferentes propiedades que han sido definidas previamente por el profesor y que darán entidad a las diferentes cartas con las que el alumno podrá interactuar. Estas propiedades se detallan a continuación y se muestra un ejemplo de ellas en la Figura 18:

- `Pos`: posición que ocupará dentro de la pantalla en base a las dimensiones predeterminadas previamente.
- `Size`: tamaño del mismo en base a las dimensiones predeterminadas previamente.
- `Rotation`: se muestra un valor si se quiere indicar que el `Token` se encuentra con rotación.
- `Clickable`: se indicará con valor true en caso de que el `Token` reaccione ante un evento de clic del usuario, y con valor false en caso contrario.
- `Rotable`: se indicará con valor true en caso de que pueda ser rotado por el usuario, y con valor false en caso contrario.
- `Resizable`: se indicará con valor true en caso de que pueda ser redimensionado por el usuario, y con valor true en caso contrario.

- `Movable`: se indicará con valor true en caso de que pueda ser movido por la pantalla de la actividad por el usuario, y con valor true en caso contrario.
- `Content`: indica el contenido que tendrá el propio `token`, pudiendo ser desde una imagen hasta un texto como en el caso del ejemplo anterior.

```
<Project version="2">
  <resolution x="1024" y="722"/>
  <Activity>
    <Objectives>
    <Tokenlist/>
    <Arealist>
      <Area id="instance379" type="Jugador">
        <pos x="10.25" y="277.1"/>
        <rotation value="0"/>
        <posfondo x="0" y="0"/>
        <size height="343.85" width="988.15"/>
        <bg url="fondopastel.jpg"/>
        <Tokenlist>
          <Token id="instance1724" type="txt" numValue="1">
            <pos x="338.1" y="85"/>
            <size height="147.8" width="273.2"/>
            <rotation value="0"/>
            <clickable>false</clickable>
            <rotatable>false</rotatable>
            <resizable>false</resizable>
            <movable>false</movable>
            <content>
              <text>Murciélago</text>
              <feedback/>
            </content>
          </Token>
          <Token id="instance11555" type="txt" numValue="1">
          <Token id="instance1923" type="txt" numValue="1">
        </Tokenlist>
        <Tokenlist>null</Tokenlist>
      </Area>
      <Area id="instance230" type="Juego">
    </Arealist>
  </Activity>
```

Figura 18 Nodo Token XML

Para proceder al parseo de cada `token` se ha recurrido al método que se muestra en la Figura 19 y Figura 20, que dado el elevado número de propiedades que puede contener cada `Token`, es de una complejidad y tamaño superior al resto.

```
private Token getTokenFromElem(Element elem) {
    Token token = new Token();
    token.setId(elem.getAttributeValue(IAppKeys.ATT_ID));
    token.setType(elem.getAttributeValue(IAppKeys.ATT_TYPE));
    token.setNumValue(elem.getAttributeValue(IAppKeys.ATT_VALUE));
    Element content = elem.getChild(IAppKeys.NODE_CONTENT);
    Element feedBack = content.getChild(IAppKeys.NODE_FEEDBACK);
    Element uList = content.getChild(IAppKeys.NODE_ULIST);
    if (token.getType().equals("txt")) {
        token.getContent().setText(
                elem.getChild(IAppKeys.NODE_CONTENT)
                        .getChild(IAppKeys.NODE_TXT).getValue());
    }
    else if (token.getType().equals("img")) {
        if(uList.getChild(IAppKeys.NODE_URL)!=null){
        token.getContent().setImg(
                gamePath
                        + "/contents/"
                        + elem.getChild(IAppKeys.NODE_CONTENT)
                                .getChild(IAppKeys.NODE_ULIST)
                                .getChild(IAppKeys.NODE_URL).getValue());
         }
    }
    if (feedBack != null && feedBack.getText() != (null)) {
        token.getContent().setFeedback(
                elem.getChild(IAppKeys.NODE_CONTENT)
                        .getChild(IAppKeys.NODE_FEEDBACK).getText());
    }
    if (uList != (null)) {
        for (Element url : uList.getChildren()) {
            token.getContent().getUrlList().add(url.getText());
        }
    }
    if (content.getChild(IAppKeys.NODE_URL) != (null)) {
        token.getContent()
                .getUrlList()
                .add(elem.getChild(IAppKeys.NODE_CONTENT)
                        .getChild(IAppKeys.NODE_URL).getText());
    }
```

Figura 19 Método parsear Token(1)

```
if (elem.getChild(IAppKeys.NODE_POS) != (null)) {

    double x = Double.valueOf(elem.getChild(IAppKeys.NODE_POS)
            .getAttribute("x").getValue());
    double y = Double.valueOf(elem.getChild(IAppKeys.NODE_POS)
            .getAttribute("y").getValue());
    token.getPos().set(x, y);
}
if (elem.getChild(IAppKeys.NODE_SIZE) != (null)) {
    double width = Double.parseDouble(elem.getChild(IAppKeys.NODE_SIZE)
            .getAttribute("width").getValue());
    double height = Double.parseDouble(elem
            .getChild(IAppKeys.NODE_SIZE).getAttribute("height")
            .getValue());
    token.getSize().setSize(width, height);
}
if (elem.getChild(IAppKeys.NODE_CLICK) != (null)) {
    if (elem.getChild(IAppKeys.NODE_CLICK).getText().equals("true"))
        token.setClickable(true);
}
if (elem.getChild(IAppKeys.NODE_MOVABLE) != null) {
    if (elem.getChild(IAppKeys.NODE_MOVABLE).getText().equals("true"))
        token.setMovable(true);
}
if (elem.getChild(IAppKeys.NODE_ROTATABLE) != null) {
    if (elem.getChild(IAppKeys.NODE_ROTATABLE).getText().equals("true"))
        token.setRotatable(true);
}
if (elem.getChild(IAppKeys.NODE_RESIZABLE) != null) {
    if (elem.getChild(IAppKeys.NODE_RESIZABLE).getText().equals("true"))
        token.setResizable(true);
}
return token;
```

Figura 20 Método parsear Token(2)

La complejidad observada en la realización del parseador radica en el alto grado de libertad de creación que tiene el profesor a la hora de crear las distintas actividades, en las que prácticamente no tiene restricción alguna, mejorando la experiencia del usuario final pero dificultando la tarea para el programador al tener que adaptar el parseador a las diferentes casuísticas. Una vez realizado el parseo del fichero XML la aplicación contará con las distintas actividades creadas por el profesor y listas para ser ejecutadas.

3.3.2 Actividades de selección

A continuación se procede a explicar el funcionamiento tanto a nivel funcional como técnico de las actividades de selección en la aplicación DEDOS-Player de la versión tabletas creada durante este TFG.

Como hemos visto en el apartado sobre el tratamiento del fichero XML generado por DEDOS-Editor, las actividades de selección tendrán uno o varios objetivos, en función del número de tarjetas que el usuario deba presionar. A continuación se muestra la pantalla inicial de una de las actividades en la Figura 21.

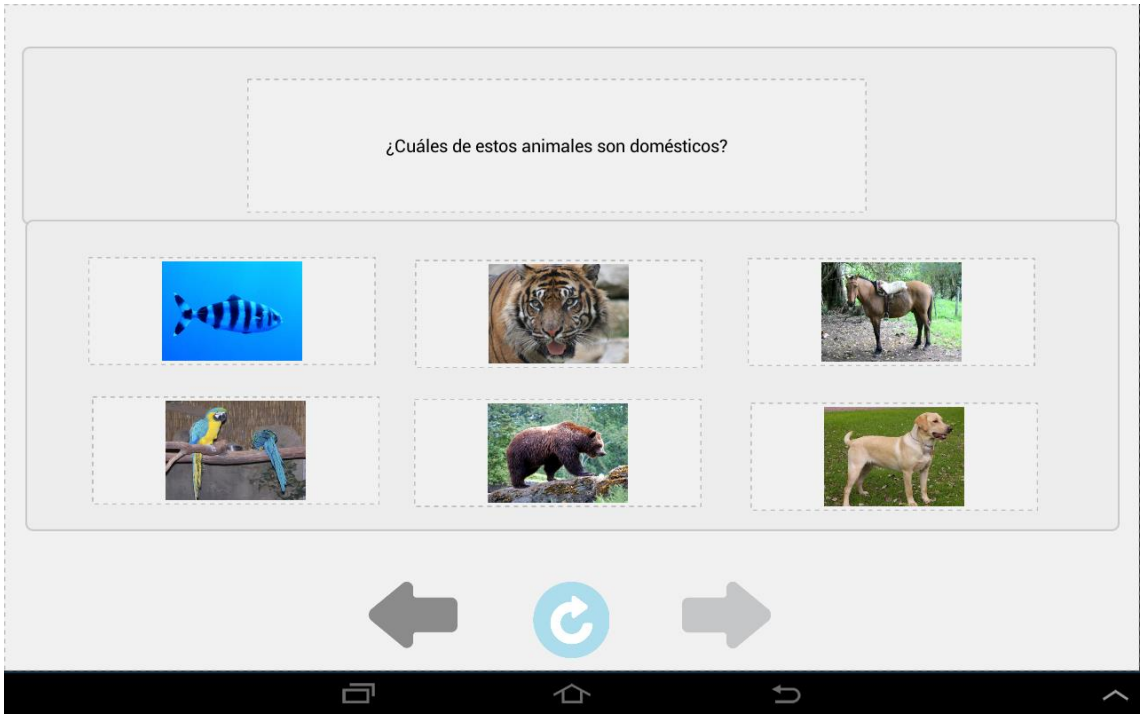


Figura 21 Inicio actividad de Selección

En concreto para este ejemplo que hemos mostrado se trata de una actividad de selección cuyas posibles respuestas son imágenes. Como se aprecia en la Figura 21 tenemos dos áreas: la que contiene el enunciado y la que contiene las respuestas posibles a la actividad. Dentro de esta última tenemos 6 `tokens` que el usuario tendrá como opciones disponibles para seleccionar.

A continuación se muestra en la Figura 22 y Figura 23 cómo se configura cada `token` para que se muestre tal y como se aprecia en la imagen, en función de las propiedades que se parsearon en el XML. Esta configuración se realiza por medio del método `setPropImg`, que recibirá un `token` y devolverá una `ImageView` configurada según las especificaciones. En el caso de tener

en la actividad tarjetas con texto, se utilizará `setPropTextView` y se devolverá un objeto de tipo `TextView`. Por continuar con el ejemplo mostraremos cómo se configura el objeto `ImageView`, ya que el objeto `TextView` se configura de manera similar a excepción de la imagen de fondo. En primer lugar asignamos id, tamaño y posición al `ImageView`.

```
//Asignamos el id
int id = Integer.parseInt(token.getId().replace("instance", ""));
imgToken.setId(id);
idTokenList.add(id);

//Asignamos un tamaño
Dimension tamanoActual = token.getSize();
Dimension tamanoEscalada = cambiarTamano(tamanoActual);
int width = (int) tamanoEscalada.getWidth();
int height = (int) tamanoEscalada.getHeight();
LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
        width, height);
imgToken.setLayoutParams(params);

//Asignamos una posición
Point p = token.getPos();
Point nuevaPosicion = cambiarPosicion(p);
p.setX(nuevaPosicion.getX());
p.setY(nuevaPosicion.getY());
imgToken.setX((float) p.getX());
imgToken.setY((float) p.getY());
```

Figura 22 Configurar ImageView (1)

Por último le asignamos la propiedad de ser clickado y le pondremos una imagen al `ImageView`.

```
//Le asignamos la propiedades de poder ser clickdado
imgToken.setOnClickListener(onClickListener);
imgToken.setBackgroundResource(R.drawable.round_rect_dashed_shape);

//Configuramos la imagen que contrendrá el objeto ImageView que devolvemos
String nombre = token.getContent().getImg();
if (!nombre.equals("")) {//hay una url
    BitmapFactory.Options options = new BitmapFactory.Options();
    InputStream is = null;
    is = new FileInputStream(nombre);
    BitmapFactory.decodeStream(is, null, options);
    is.close();
    is = new FileInputStream(nombre);
    options.inSampleSize = Math.max(options.outWidth / 200,
            options.outHeight / 200);
    Bitmap bitmap = BitmapFactory.decodeStream(is, null, options);
    imgToken.setImageBitmap(bitmap);
}
```

Figura 23 Configurar ImageView (2)

Para dotar de funcionalidad, cada vez que se produce un evento sobre las tarjetas que aparecen en pantalla, se comprueba si la respuesta es correcta o no. Con el objetivo de realizar dicha comprobación se procede a recorrer todos los objetivos de la propia actividad y se verifica que el id de la tarjeta que se pulsa en ese momento es el mismo que el id que señala la opción correcta en el XML que hemos parseado previamente.

Al presionar una tarjeta correcta se mostrará al usuario un *tick* verde para que sepa que la tarjeta que ha presionado es la correcta, sumándole una unidad a su puntación, tal y como se observa en el ejemplo presentado en la Figura 24.

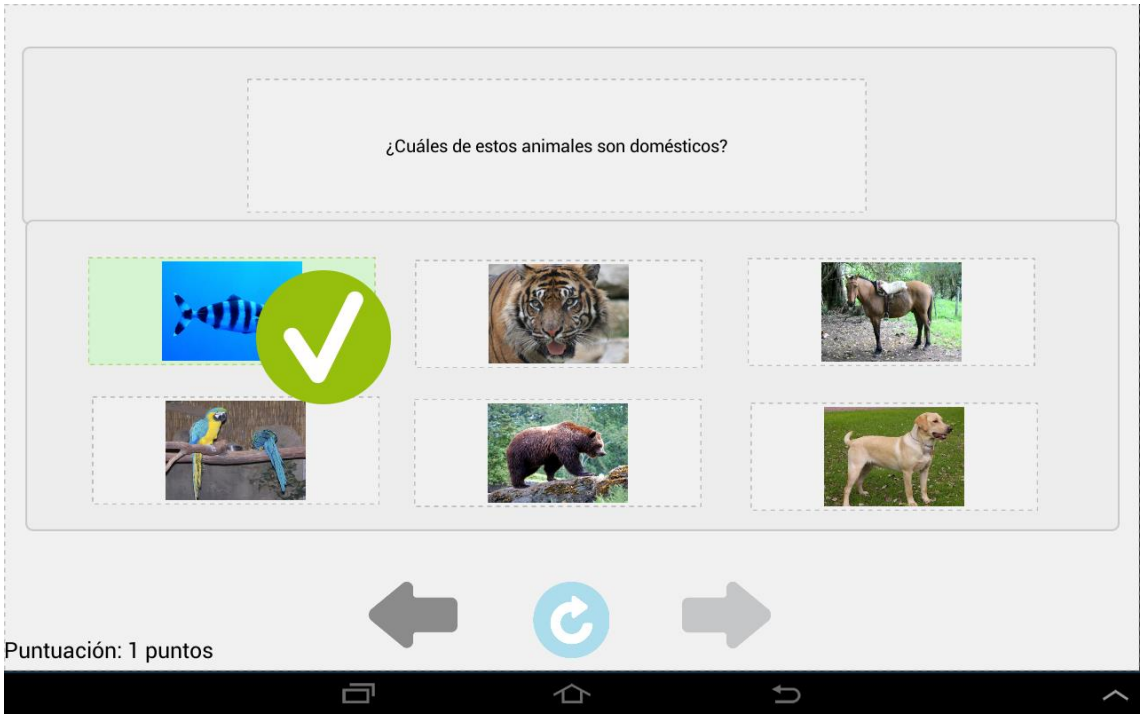


Figura 24 Acierto actividad de Selección

En caso de que la opción señalada fuera correcta se llevarán a cabo los eventos definidos en la Figura 25 y que se explican a continuación:

- Se cambiaría el color a verde de la tarjeta marcada.
- Se deshabilitaría la tarjeta para que no reciba ningún evento más.
- Se completa el objetivo de la actividad.
- Se procede a mostrar un pop-up para dar *feedback* al usuario.
- Mostramos el marcador acumulado.

```
if (objective.getIdDestino().replace("instance", "")
        .equals(String.valueOf(view.getId()))) {

    view.setBackgroundResource(R.drawable.round_rect_dashed_green_correct_shape);
    view.setEnabled(false);
    objective.setDone(true);
    createOkToast(view);
    enabledButtonNextActivity();
    find = true;
    showScore();
    break;
}
```

Figura 25 Eventos acción correcta Selección

Para proporcionar *feedback* al usuario se muestra el *tick* verde a modo de *popup* utilizando para ello la clase `Toast`.

Estos `Toast` se podrán personalizar de la siguiente manera:

- Duración: podrá tener dos valores, "1" para un instante de tiempo más largo, con duración de 3.5 segundos, y un valor "0" para un instante de tiempo algo más corte, con duración de 2 segundos.
- Icono: por medio del método `setView` se podrá personalizar el `Toast` con una imagen que deseemos, en nuestro caso la imagen de un *tick* verde.
- *Gravity*: con esta propiedad le indicaremos a la aplicación en qué lugar queremos que aparezca el `Toast`.

Por último por medio de la operación `show` permitiremos que comience la aparición del `Toast`. Esta operación se muestra a continuación en la Figura 26.

```
private void createOkToast(View view) {
    Toast toast;
    LayoutInflater inflater = getLayoutInflater();
    View okicon = inflater.inflate(R.layout.cust_toast_layout,
            (ViewGroup) findViewById(R.id.relativeLayout1));
    toast = new Toast(getApplicationContext());
    toast.setDuration(Toast.LENGTH_SHORT);
    toast.setView(okicon);
    toast.setGravity(Gravity.LEFT | Gravity.CENTER_VERTICAL,
            (int) view.getX(), 0);

    toast.show();

}
```

Figura 26 Crear Toast

Por el contrario si la tarjeta que hemos presionado es incorrecta se mostrará un *tick* rojo (véase la Figura 27) indicando al usuario que la opción que ha seleccionado es incorrecta, pero no restará puntuación al alumno para no que el alumno no pierda la motivación en el caso de que el número de fallos sea elevado. La manera de mostrar el *tick* rojo será realizar la misma operación que con el *tick* verde pero cambiando la imagen que queremos que se muestre.

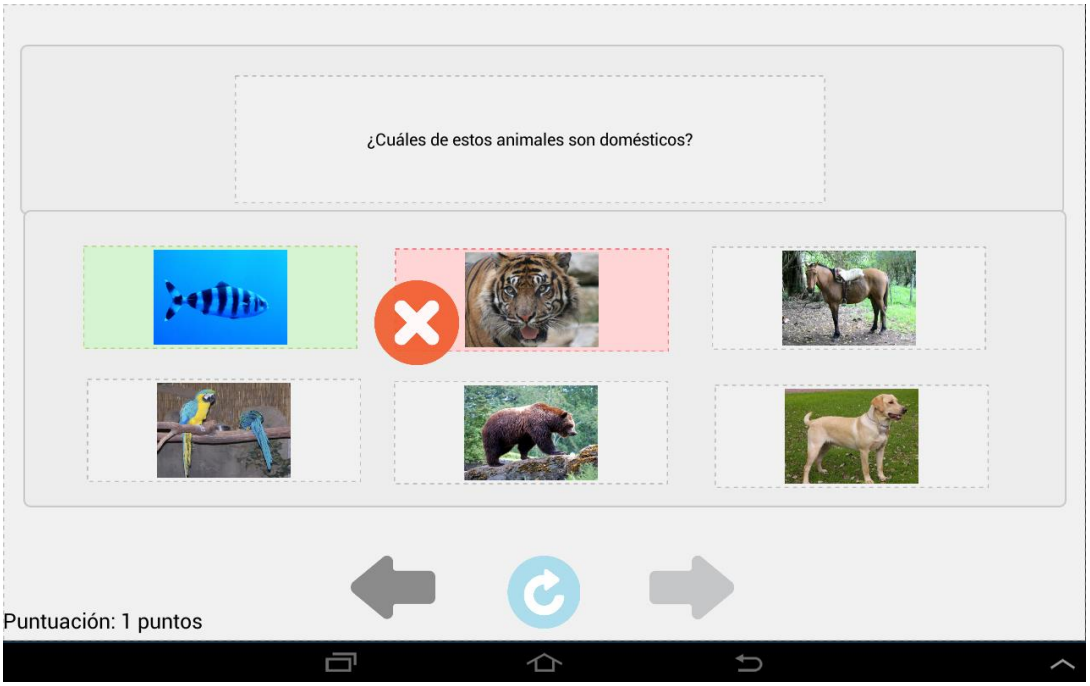


Figura 27 Error actividad de Selección

En caso de que la opción señalada fuera incorrecta se llevarán a cabo los eventos definidos en la Figura 28 y que se explican a continuación:

- Se cambiaría a color rojo la tarjeta seleccionada.
- Se deshabilita la tarjeta para que no se pueda volver a seleccionar.
- Se lanza el pop-up dando el feedback al usuario.

```
if (!find) {
    view.setBackgroundResource(R.drawable.round_rect_dashed_red_incorrect_shape);
    view.setEnabled(false);
    createWrongToast(view);
}
```

Figura 28 Eventos acción incorrecta Selección

Una vez que el alumno ha completado los objetivos necesarios, en este caso 3, se habilitará la flecha derecha (véase Figura 29) indicando que puede avanzar a la siguiente actividad. En este momento por mucho que apretemos las tarjetas, éstas se encontrarán deshabilitadas, permitiendo así que el único foco de atención sea la flecha que les permita avanzar.

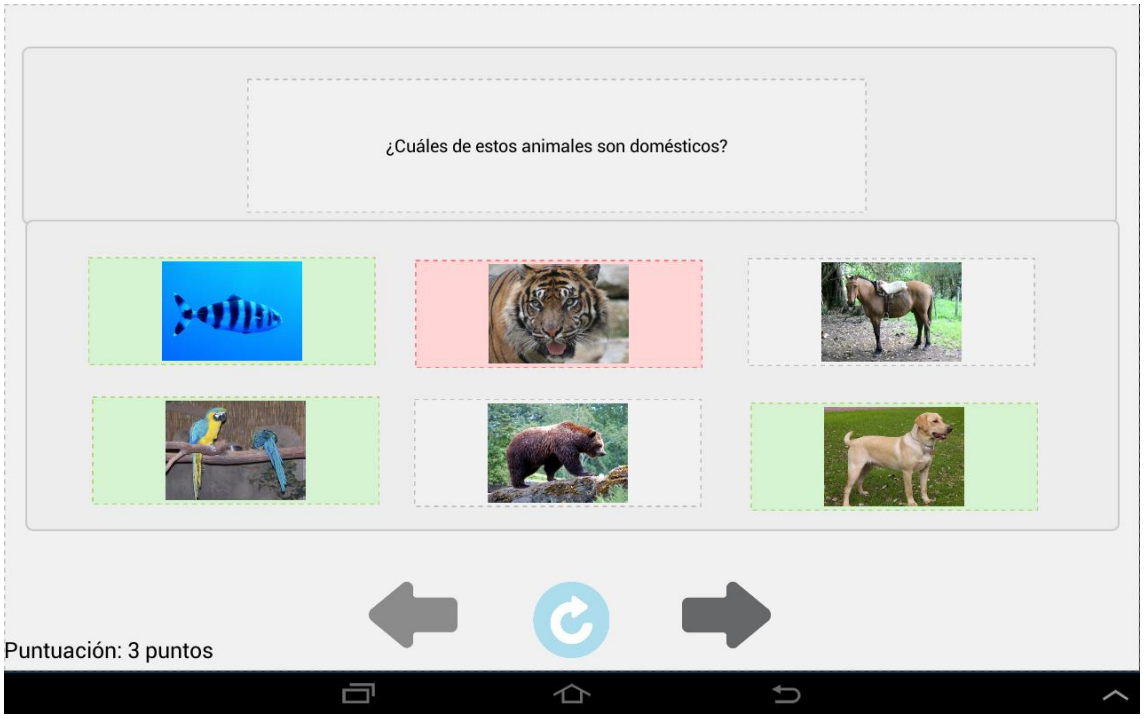


Figura 29 Actividad completa de Selección

3.3.3 Actividades de emparejamiento

Cada actividad de emparejamiento podrá tener uno o varios objetivos a completar, en función de lo que se haya definido previamente con la herramienta DEDOS-Editor. El ejemplo que se muestra en la Figura 30 se trata de una actividad de emparejamiento con 3 relaciones.

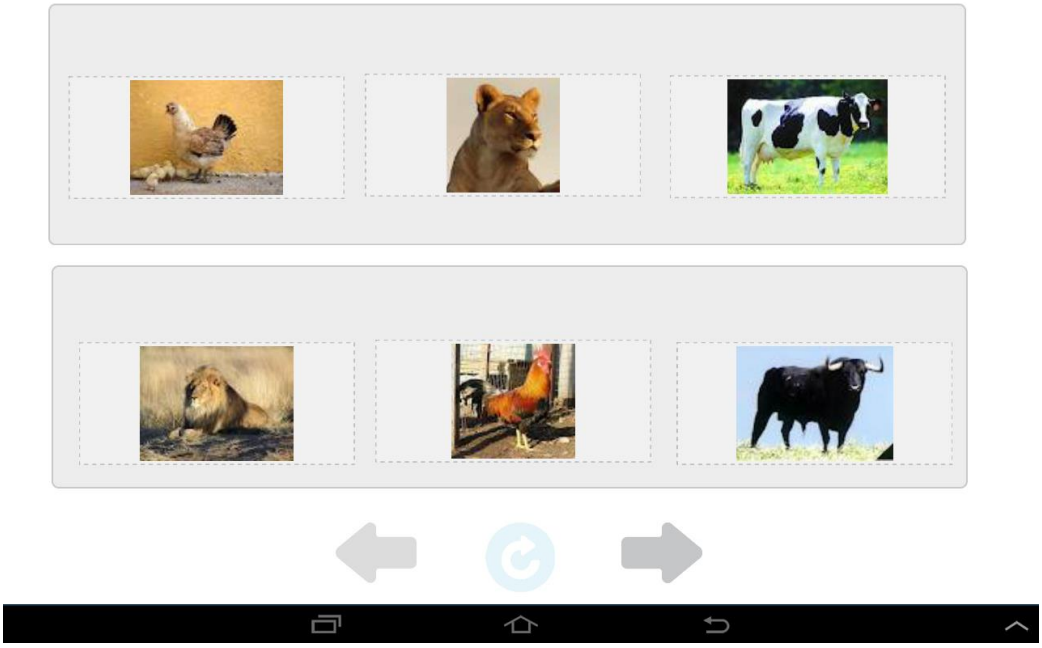


Figura 30 Inicio actividad de Emparejamiento

Para otorgar las propiedades necesarias a cada `token` se ha recurrido a un método similar al utilizado en la actividad de Selección, con algunas diferencias como la inclusión de una propiedad que hace posible realizar el "*drag and drop*" sobre los elementos de esta actividad.

Por "*drag and drop*" entendemos aquella propiedad que tienen determinados elementos de ser arrastrados y soltados en determinados espacios, y detectar todos los eventos que ello genera. Esta propiedad es una de las partes fundamentales de la actividad de emparejamiento y que explicaremos su funcionamiento con más detalle a continuación.

Para poder trabajar con esta propiedad se deberá activar un *listener* de la clase `OnDragListener` y definir los métodos correspondientes, en nuestro caso se redefine el método `onDrag`, que recibe el objeto arrastrado de tipo `View` y el evento correspondiente.

Dentro de los diferentes eventos que se puedan recibir a nosotros nos interesará quedarnos con el evento `ACTION_DROP`, que es aquel que indica que un objeto ha sido depositado encima de otro, y realizar las operaciones oportunas para verificar si el evento se ha realizado de manera correcta o no. En este evento se podrán dar 3 posibles casuísticas:

- Se deposita la carta sobre la carta correcta. En esta situación, el destino quedaría marcado en verde y la carta arrastrada, si no tiene más objetivos asociados como es el caso, desaparecerá para no crear confusiones. La Figura 31 muestra un ejemplo de la casuística comentada.

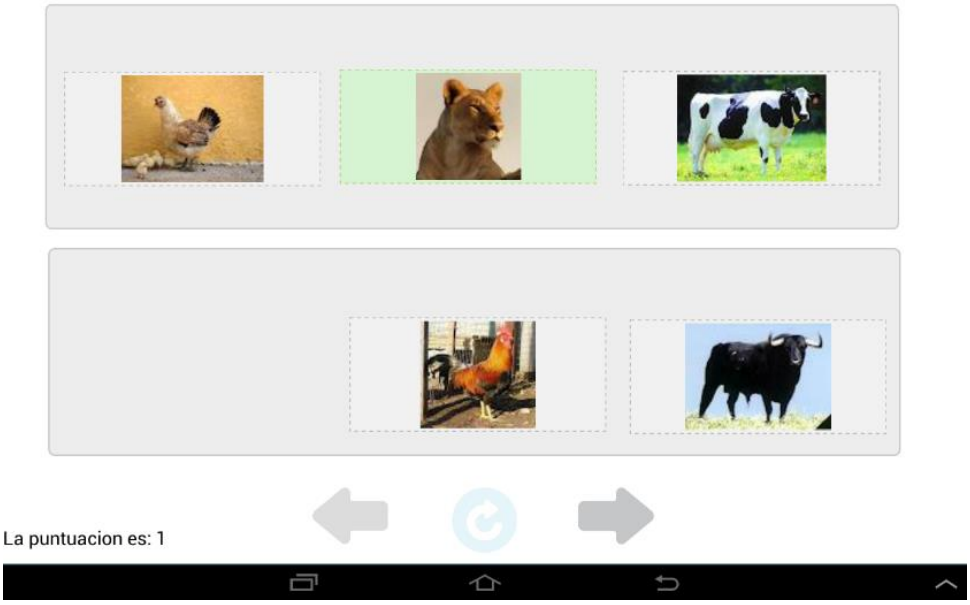


Figura 31 Emparejamiento: Respuesta correcta

- Se deposita la carta sobre la carta incorrecta. La carta de origen volverá a su posición original y aparecerá un *tick* rojo indicándonos que se ha cometido un fallo. En la Figura 32 se muestra un ejemplo de la casuística comentada.

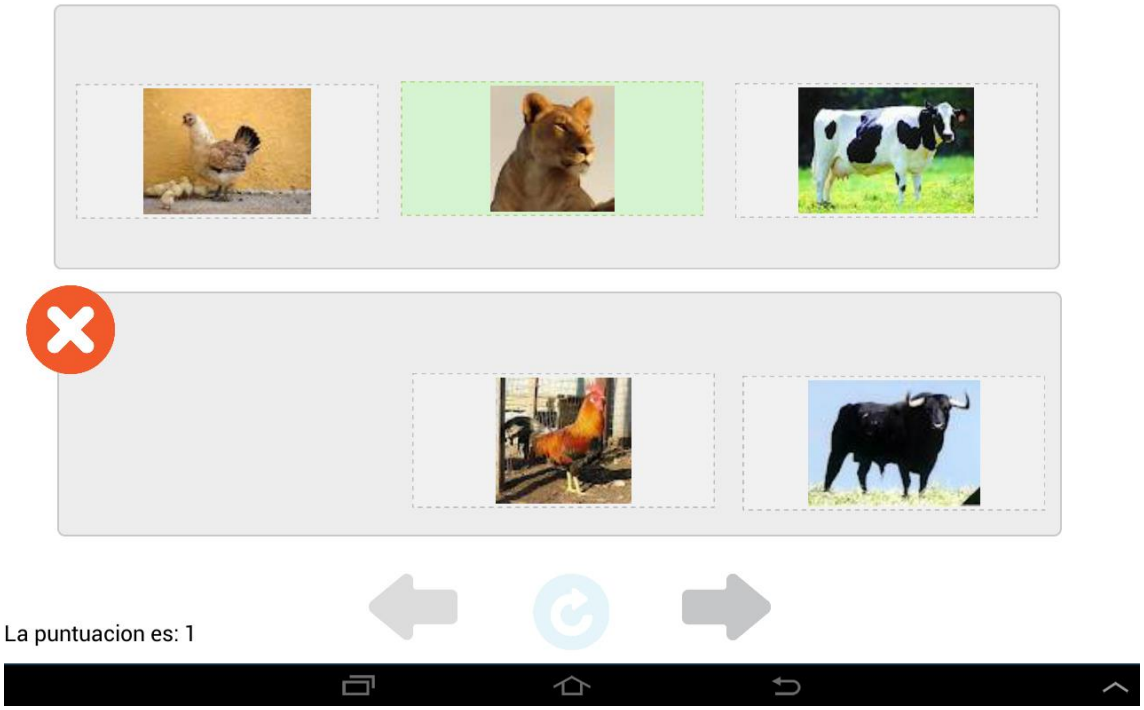


Figura 32 Emparejamiento: Respuesta incorrecta

- Se deposita la carta en otro objeto distinto. Por último si por error se deposita la carta en cualquier otro lugar que no sea una de las cartas del juego, como por ejemplo el botón de reiniciar la actividad, se indicará al usuario por un tick rojo que la respuesta es incorrecta y la carta arrastrada volverá a su posición inicial, al igual que se comentaba en el ejemplo anterior.

Finalmente, si se ha conseguido completar la actividad se procederá a activar el botón con la flecha de adelante para que el usuario pueda avanzar en el juego (véase la Figura 33).

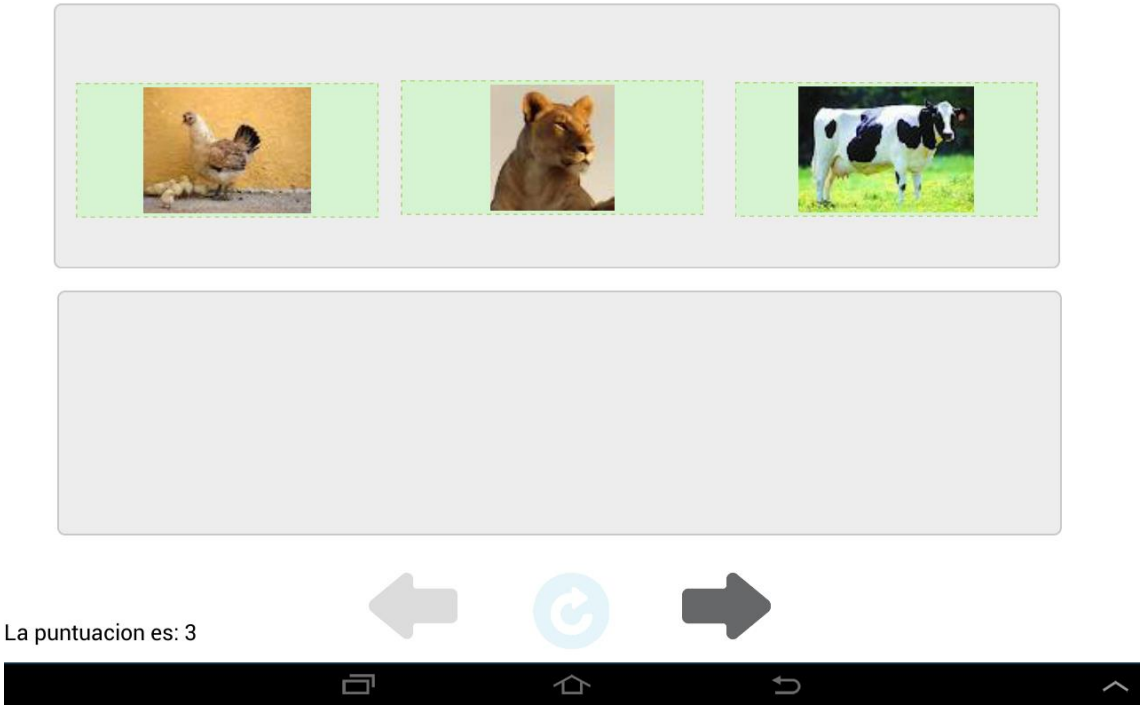


Figura 33 Emparejamiento: Actividad completada

3.3.4 Actividades de Matemáticas

Las actividades de matemáticas podrán tener uno o varios objetivos asociados. Cada objetivo tendrá un valor, es decir, una cifra que se tendrá que alcanzar arrastrando las cartas correspondientes sobre la carta objetivo. Se muestra como ejemplo en la Figura 34 una actividad en la que se tiene un único objetivo con valor igual a 7.

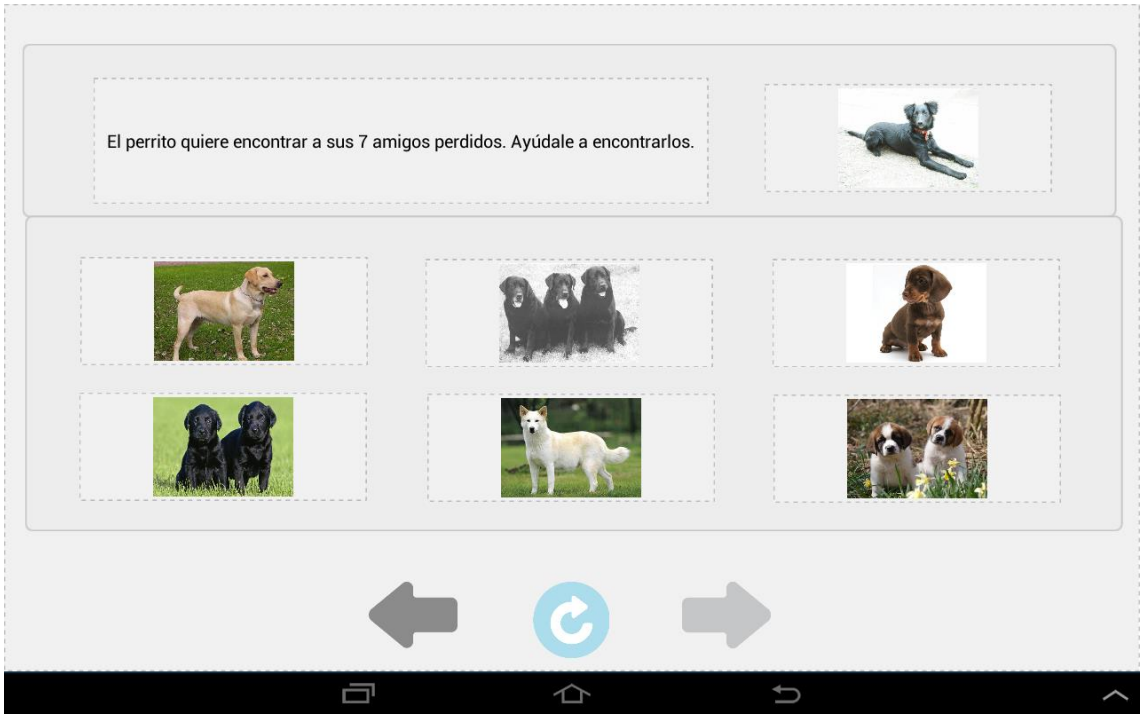


Figura 34 Matemáticas: Inicio de Actividad

Para cumplir este objetivo se tendrá que depositar sobre la carta del perro que está la primera área las cartas que se encuentran en la segunda. Con el fin de no dar pistas al alumno sólo se dará *feedback* en las dos siguientes casuísticas:

- Se ha alcanzado el objetivo justo.
- Nos hemos pasado del valor objetivo.

En el caso de alcanzar el objetivo propuesto aparecerá un tick verde que nos informará que hemos cumplido el objetivo y podremos avanzar a la siguiente actividad (véase la Figura 35).

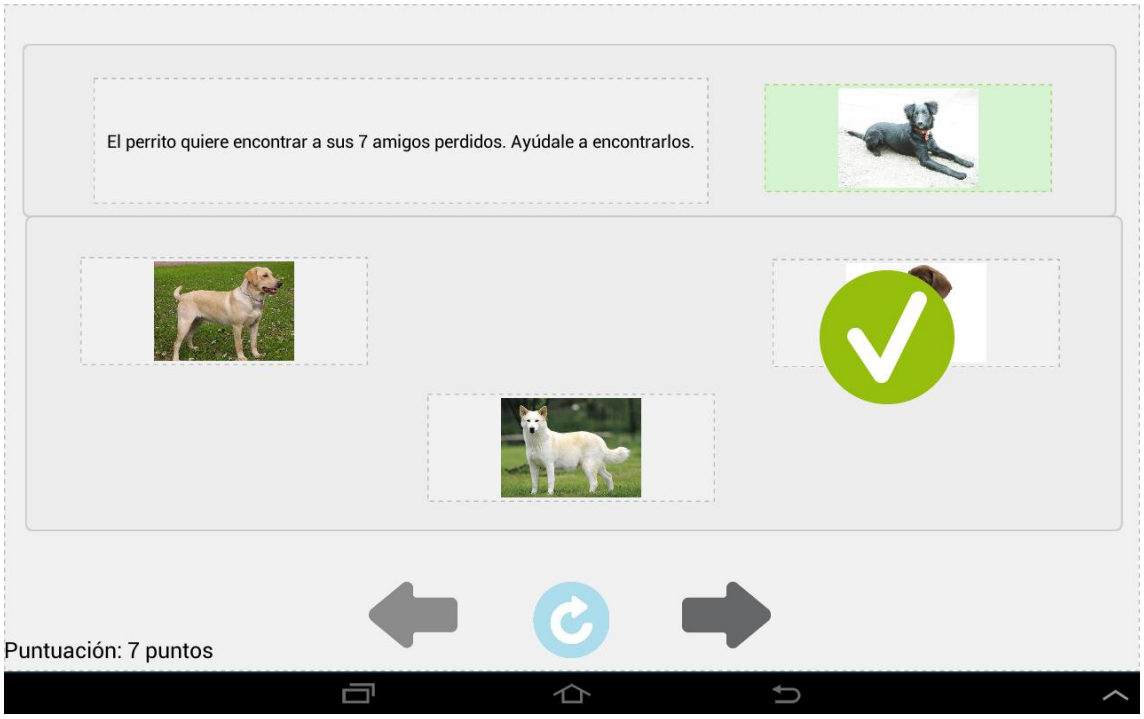


Figura 35 Matemáticas: Completar actividad correctamente

Por el contrario, puede suceder que si por ejemplo nuestro contador acumulado está a 6 y nuestro objetivo es 7, si proporcionamos una carta con valor 2 habremos excedido el contador por lo que se contará la actividad como incorrecta. Se nos dejará pasar de actividad pero no se sumará un punto a nuestro marcador tal y como se muestra en la Figura 36.

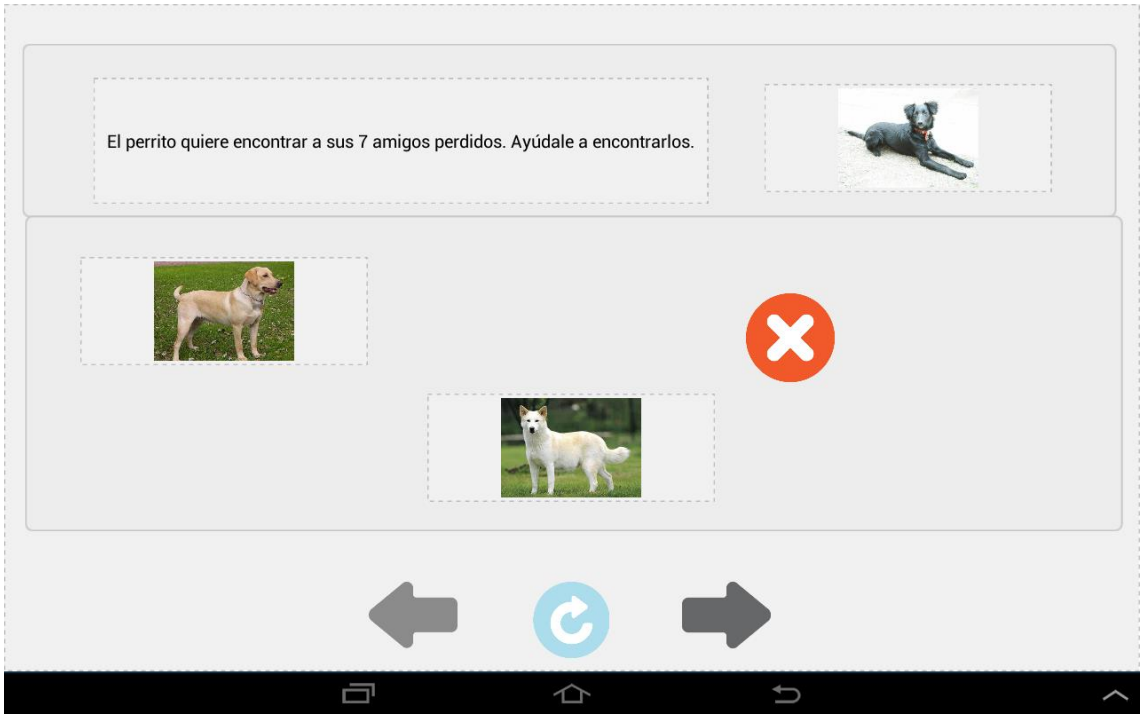


Figura 36 Completar actividad incorrectamente

3.3.5 Funcionalidad común

A continuación se comentan una serie de funcionalidades agregadas a la herramienta que funcionarían de manera exactamente igual en los 3 tipos de actividades diseñadas: selección, emparejamiento y matemáticas.

Es posible que a lo largo del juego queramos volver a una actividad anterior. Según el diseño del sistema operativo Android, es el propio sistema el que te posibilita la opción de volver a atrás con su propio botón, ya sea físico o virtual dependiendo del modelo de tableta que se esté usando. Aun así, con el fin de que la aplicación fuera lo más similar a la que se había diseñado previamente se diseñó un botón exactamente igual al de la versión para mesas multicontacto. El funcionamiento de este método (véase la Figura 37) es sencillo, primero se utiliza el método `finish` privado de la clase `Activity` de Android, que "finaliza" el estado actual de la aplicación, y a continuación se procede a lanzar un nuevo `Intent` que genera una nueva interfaz con la actividad que había cargada previamente. Cabe destacar que el funcionamiento del botón de *click* hacia adelante funciona de manera similar pero cargando con el `Intent` la actividad siguiente.

```
public void clickBack(View v) {

    finish();
    Intent intent = new Intent(this, GameDataController.getInstance()
            .getPreviousActivity());
    intent.putExtra("ACTIVITY_ID", GameDataController.getInstance()
            .getActivityId());
    startActivity(intent);

}
```

Figura 37 Eventos botón atrás

De manera similar se dota de funcionalidad a nuestro botón de reinicio, a través del método creado *restart* (véase Figura 38). En él se obtendría el *Intent* lanzado actualmente a través del método *getIntent*, y lo volveríamos a llamar con el método interno *startActivity*.

```
public void restart(View v) {
    Intent intent = getIntent();
    startActivity(intent);

}
```

Figura 38 Eventos botón reiniciar

Para prevenir posibles errores y no dejar bloqueada o congelada la aplicación se ha diseñado un mecanismo de por el cual en el momento que ocurriera un error en la aplicación se lanzará una pantalla de error como la mostrada en la Figura 39 que permite no perder el control sobre lo sucedido en la aplicación dando más robustez a la aplicación. Para llevar a cabo esta funcionalidad se ha recurrido al método cuyo código se presentan en la Figura 40.

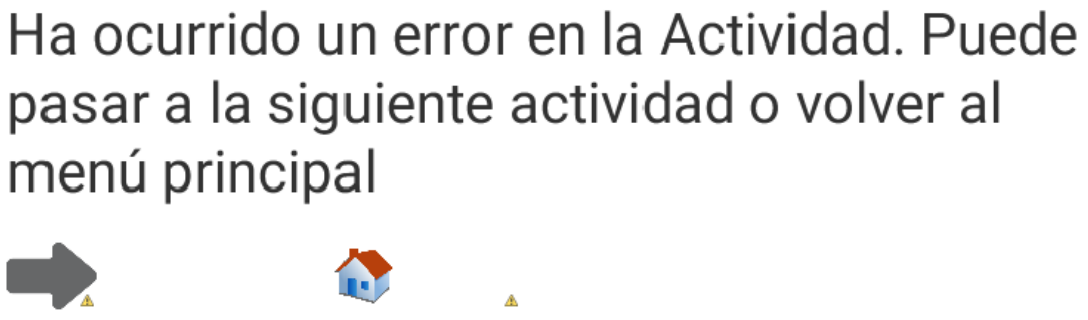


Figura 39 Error actividad

```
public void launchErrorActivity() {
    Intent startIntent = new Intent(getApplicationContext(),
            ErrorActivity.class);
    startIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    getApplicationContext().startActivity(startIntent);
}
```

Figura 40 Eventos error actividad

Cada vez que se produce un acierto en la actividad se comprueba si quedan objetivos por completar dentro de la aplicación. Esto se comprueba para proceder a habilitar la flecha que permite avanzar de actividad. En el caso de que todos los objetivos se encuentran completados se procedería a la deshabilitar todos los `token` que hubiera en pantalla, para que el alumno centre toda su atención en avanzar de actividad. Esta casuística es tratada en el método de la Figura 41.

```
private void enabledButtonNextActivity() {
    buttonNext = (ImageView) findViewById(R.id.forward);
    Boolean isDoneAllObjectives = selCont.objectivesDone(activity);

    if (isDoneAllObjectives) {
        GameDataController.getInstance().disableTokens(area, idTokenList);
        buttonNext.setImageResource(R.drawable.forwarddark);
        buttonNext.setEnabled(true);

    } else {

        buttonNext.setImageResource(R.drawable.forwardlight);
        buttonNext.setEnabled(false);

    }

}
```

Figura 41 Deshabilitar elementos de la pantalla

Por último tras completar todas y cada una de las actividades del juego aparecerá un menú final como el que se muestra en la Figura 42 informándonos de nuestra puntuación y preguntándonos si queremos jugar a otro juego o salir de la aplicación.

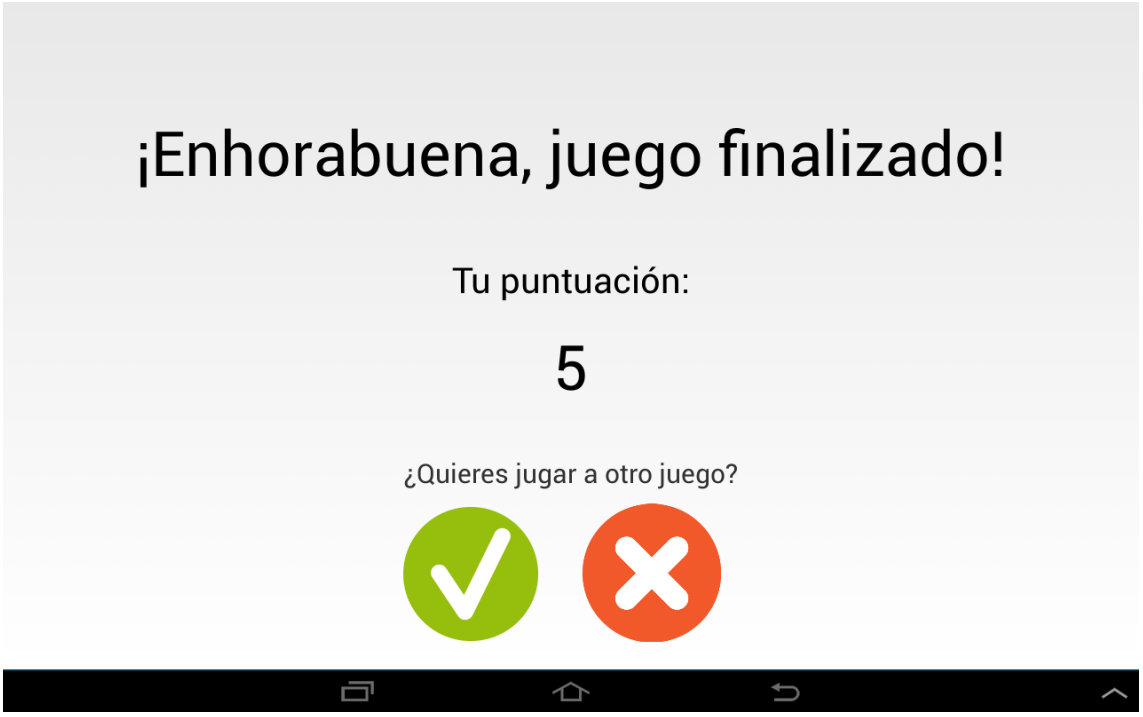


Figura 42 Puntuación final Selección

3.4 Diagrama de clases

Esta sección recoge los diagramas de clases correspondientes al desarrollo del proyecto explicando el porqué del uso de cada uno de ellos. En primer lugar se muestra en la Figura 43 el diagrama de paquetes correspondiente con el que se han subdividido las diferentes clases según el uso de estas. Los paquetes en los que se ha dividido el desarrollo son los siguientes:

- Paquete *DEDOS*: es el paquete que englobará a todos los paquetes de la aplicación. En su interior se tendrán las clases que contienen la parte visible de la aplicación, es decir, cada una de los tipos de actividad.
- Paquete *Browser*: es el paquete que contiene todas las clases utilizadas para cargar todos los proyectos educativos que contiene la aplicación.
- Paquete *Controller*: contiene las clases que forman la lógica general de la aplicación independientemente del tipo de actividad.
- Paquete *Interfaces*: contiene las interfaces utilizadas a lo largo del desarrollo.
- Paquete *Model*: contiene todas aquellas clases que representan la información con la que se está trabajando.
- Paquete *Util*: contiene las clases auxiliares para el desarrollo del proyecto.

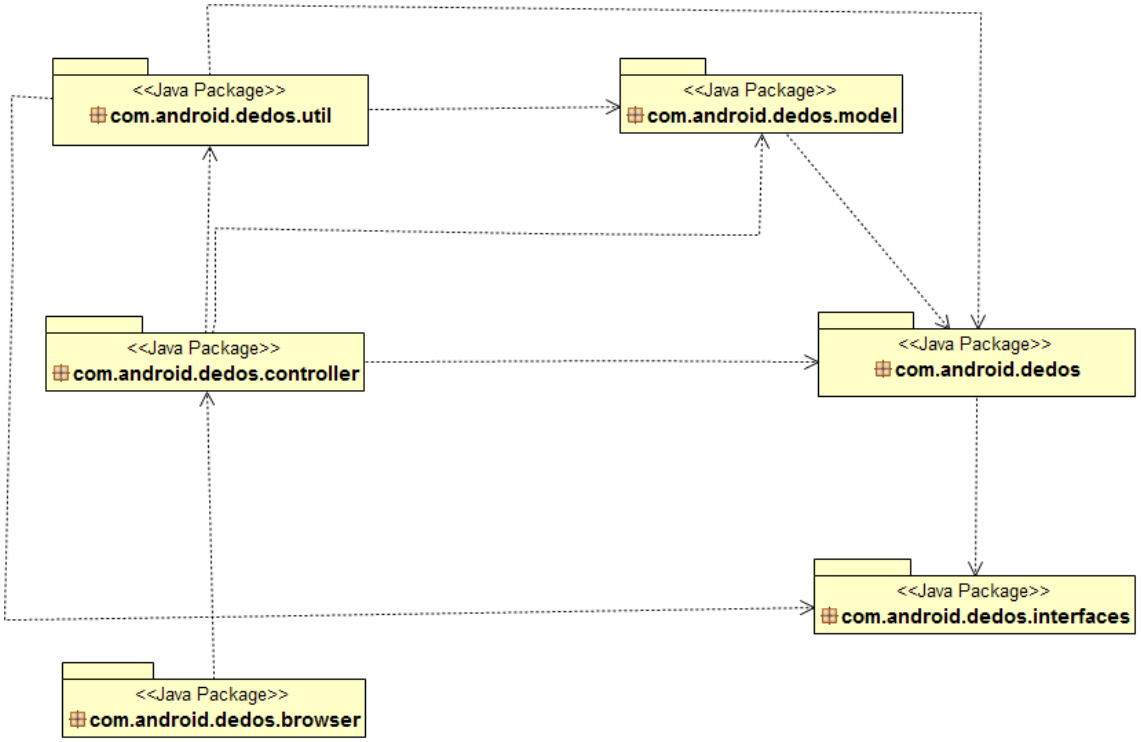


Figura 43 Diagrama de paquetes

A continuación se procede a comentar el contenido de cada paquete en profundidad.

3.4.1 Paquete DEDOS

Este paquete contendrá toda la lógica visible de la aplicación, en cuanto que contendrá todas las clases tipo *Activity* que son con las que el usuario podrá interactuar. Como se observa en la Figura 44 este paquete se encuentra formado por las siguientes clases:

- *MainActivity*: muestra el contenido del menú principal.
- *SelectionActivity*: muestra el contenido de una actividad de tipo selección.
- *PairActivity*: muestra el contenido de una actividad de tipo emparejamiento.
- *MathActivity*: muestra el contenido de una actividad de tipo matemáticas.
- *ResultActivity*: muestra el resultado final de un proyecto educativo
- *ErrorActivity*: se muestra en caso de existir algún error al cargar la actividad.
- *SplashScreenActivity*: se muestra como pantalla de bienvenida mostrando todos los colaboradores del proyecto DEDOS.

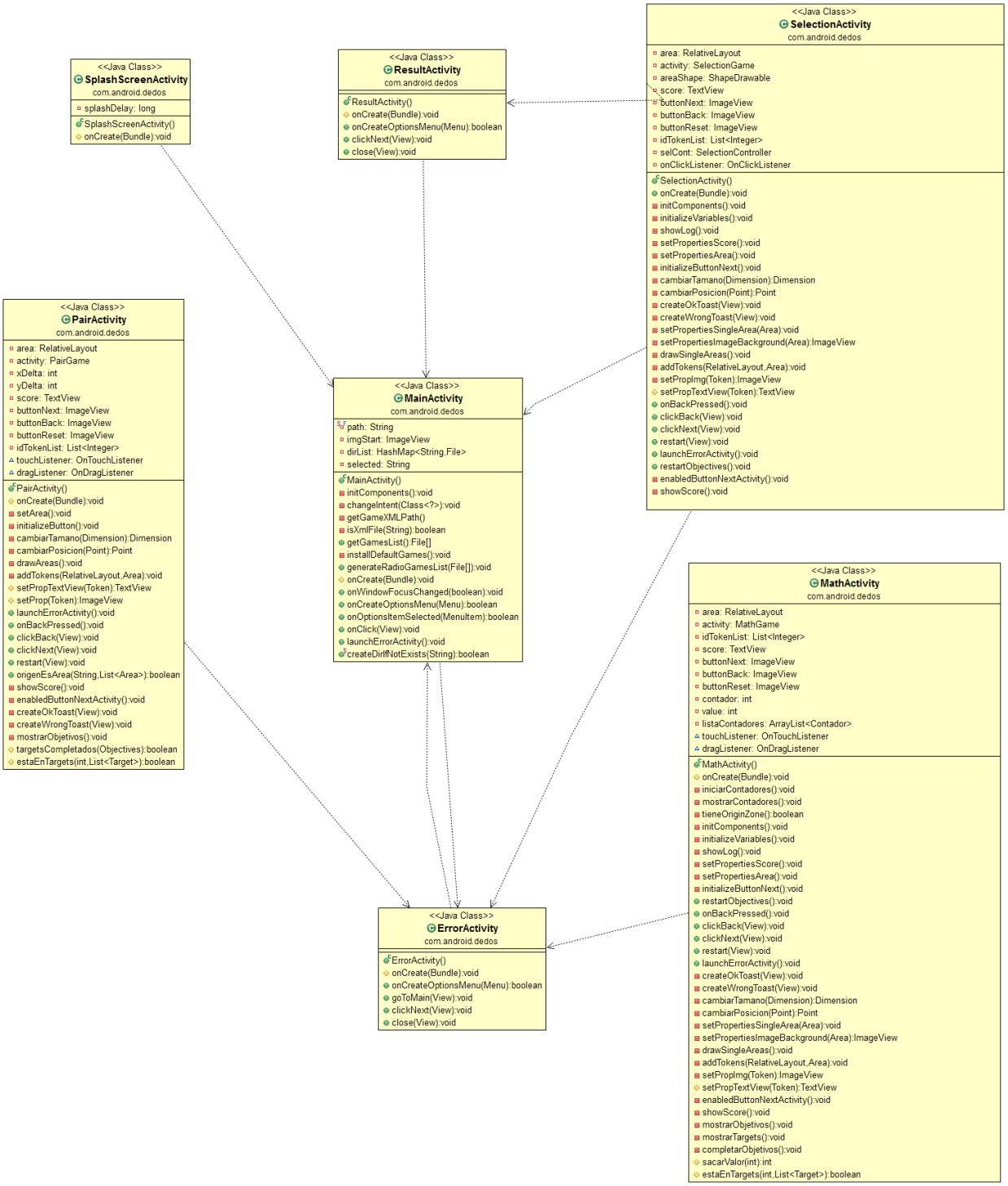


Figura 44 Diagrama de clases-DEDOS

3.4.2 Paquete Browser

Este paquete (véase la Figura 45) se utiliza para llevar a cabo todas las operaciones necesarias para mostrar todos los proyectos educativos que se encuentran almacenados en la memoria del dispositivo. La clase que soporta toda la lógica de esta operación es *FileChooserActivity*.

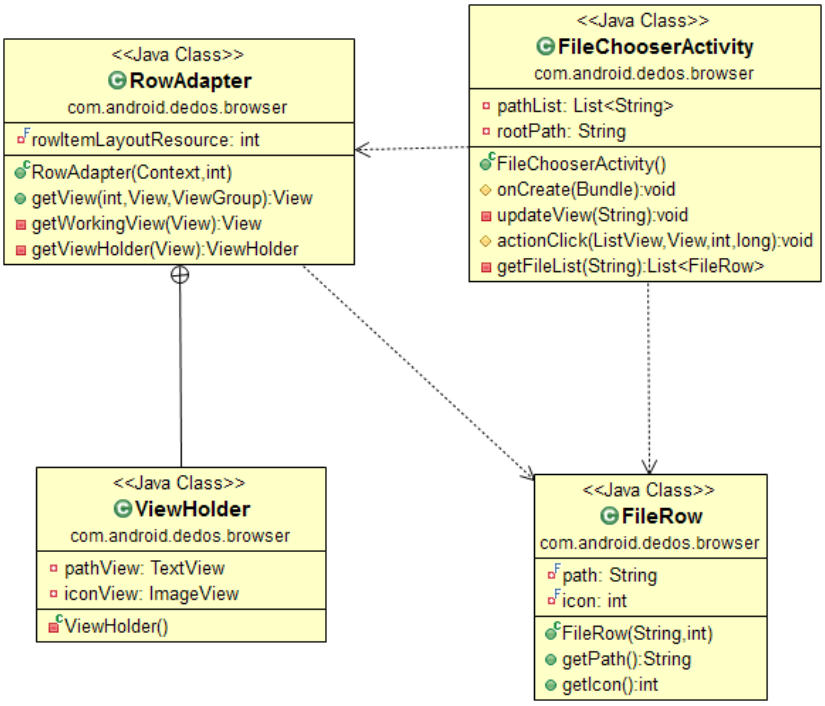


Figura 45 Diagrama de clases-Browser

3.4.3 Paquete Controller

Este paquete formado por una sola clase, tal y como se muestra en la figura Figura 46, se encarga de relacionar todos los tipos de actividad que se tengan disponible con operaciones comunes a todas ellas.

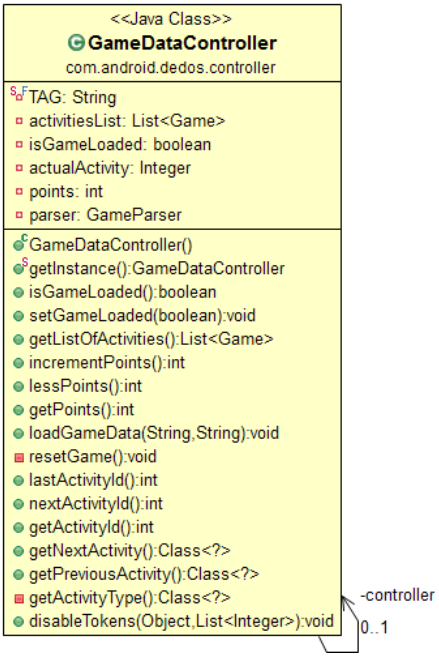


Figura 46 Diagrama de clases-Controller

3.4.4 Paquete Interfaces

Este paquete formado por una sola clase, tal y como se muestra en la Figura 47, se encarga de agrupar todos los nombres de atributos que son más utilizados en todo el desarrollo, como por ejemplo el nombre de los nodos del fichero XML y que nos permitiría, en el supuesto de que se realizase un cambio en los nombres de los nodos, tener solo que modificar esta clase en lugar de tener que modificar todo el desarrollo completo.

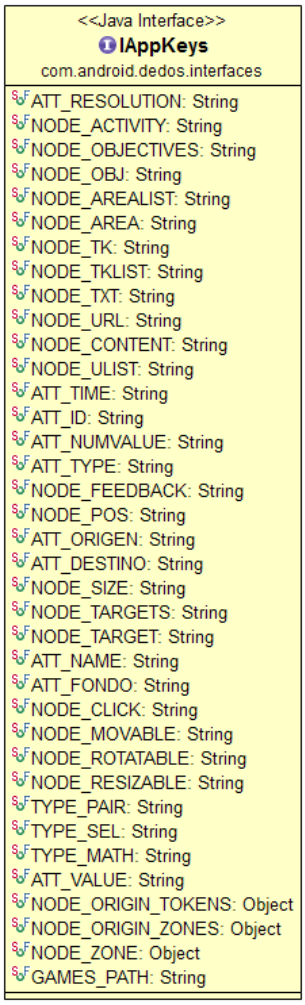


Figura 47 Diagrama de clases-Interfaces

3.4.5 Paquete Model

Este paquete (véase la Figura 48) está formado por todas aquellas estructuras que permiten representar la información de toda la aplicación de una manera correcta y eficiente. Tendrá la información de las áreas, *tokens* y diversos objetos a mostrar.

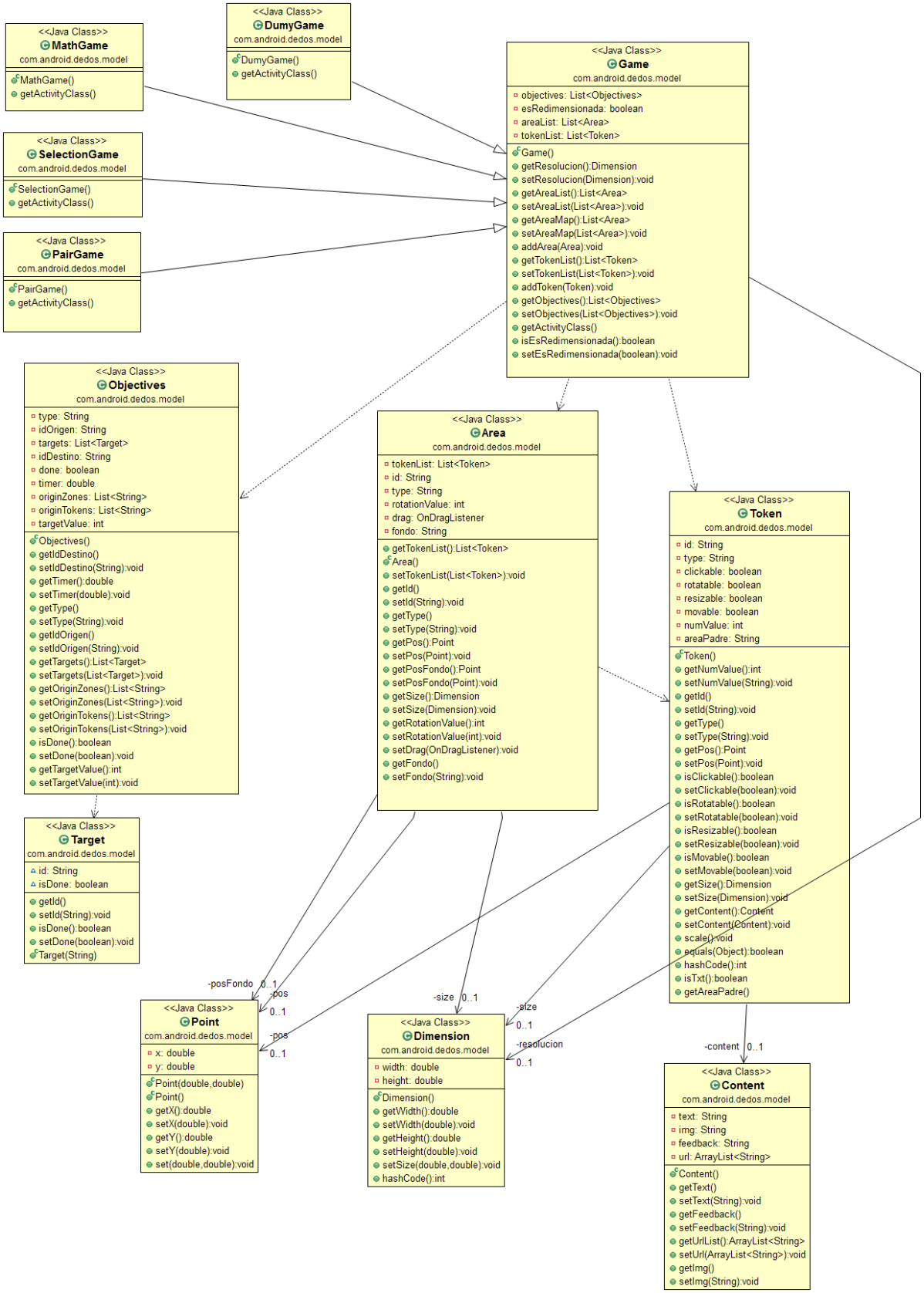


Figura 48 Diagrama de clases-Model

3.4.6 Paquete Util

La utilidad de este paquete (véase la Figura 49) radica en almacenar todas las clases auxiliares que se han ido utilizando a lo largo de todo el proyecto. Principalmente almacena las clases que se encargan de la lectura del fichero XML que contiene el proyecto educativo a cargar.

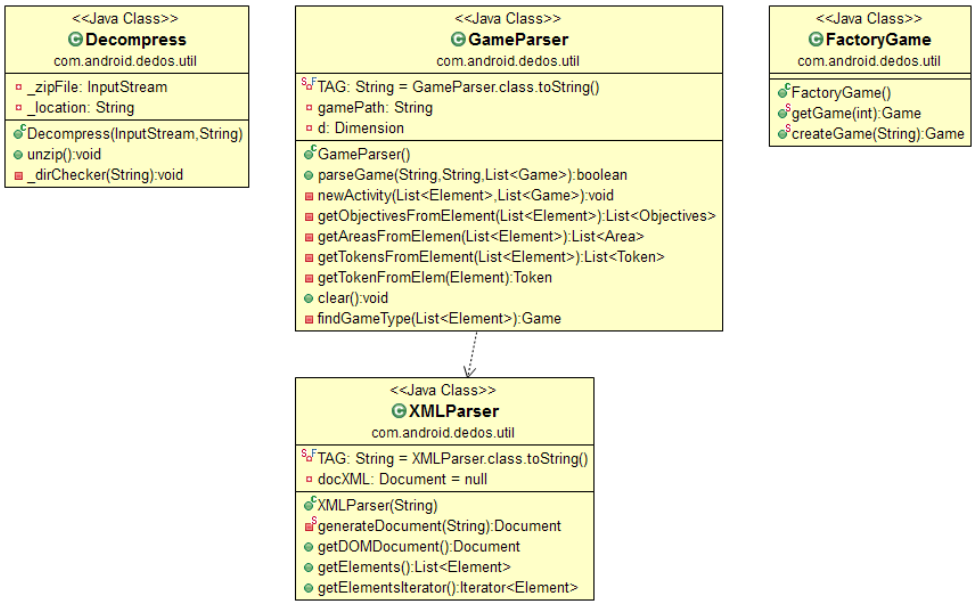


Figura 49 Diagrama de clases-Util

3.5 Problemas surgidos en el desarrollo y cómo se solucionaron

En esta sección se pretende exponer aquellos problemas tecnológicos que se tuvieron a lo largo de todo el desarrollo de la aplicación y la manera en la que fueron solucionados:

- Redimensión de los componentes de la aplicación: uno de los problemas más importantes que se tuvieron que afrontar fue el redimensionado de los elementos creados en el proyecto educativo con la herramienta DEDOS-Editor. Por defecto, Android permite esta funcionalidad, pero solo para aquellas aplicaciones en las que el tamaño de la interfaz es predeterminada. En nuestro a priori se desconoce qué elementos configurarán la pantalla de juego, ya que estos provienen del fichero XML generado con DEDOS-Editor. Para solucionar este problema al cargar el juego (fichero XML), se obtienen la resolución de la pantalla actual y se procede a realizar una proporción de cada elemento en función del tamaño original para el que fue pensado.
- "Desaparición de tarjetas": otro de los problemas encontrados fue en las actividades de emparejamiento al depositar una tarjeta sobre cualquier otro elemento de la pantalla. Al

realizar esta operación las tarjetas desaparecían debido a que en el método que se utilizaba para la funcionalidad del *"Drag and drop"* se estaba dando por hecho que siempre se iba a realizar de tarjeta a tarjeta. Se realizó un *cast* de los elementos que intervenían en la operación y se solucionó esta problemática.

- *"Out of memory":* al cargar varias imágenes con gran resolución en pantalla a la vez se daba la problemática de que si relanzábamos muchas veces seguida la misma actividad la aplicación se quedaba "congelada" y en los *logs* aparecía el mensaje de *Out of memory* por lo que el dispositivo utilizado se quedaba sin memoria y paraba su ejecución. Este error era tan comentado en foros como *Stackoverflow* que la propia web de Android tuvo que publicar una pequeña guía[10] de cómo cargar eficientemente varias imágenes en pantalla. La solución radica en que en un principio cuando se cargaban las imágenes se hacía respetando la resolución original, con lo que si se cargaban varias imágenes tomadas con una resolución muy grande estas eran adaptadas al tamaño de la pantalla pero sin embargo seguían ocupando el mismo espacio en memoria por lo que llegaba un momento que ésta podía desbordar. Por ejemplo si incluimos una imagen con resolución 1024x768 cuando en realidad se está visualizando como una imagen de 128x96 se perderían muchísimos recursos si se produce su incorporación en la aplicación con resolución 1024x768 y luego adaptarla al tamaño de la pantalla, por lo que sería mejor tratarla desde el principio como una imagen de 128x96 e incluirla en el aplicación. Este aspecto es tratado con frecuencia en aplicaciones tipo Facebook dónde hay muchas imágenes en pantalla, o incluso en la propia galería fotográfica del dispositivo. Finalmente con la ayuda de esta guía se solventaron los problemas comentados.

[10] Entrada en la web de Android sobre carga de imágnes: http://goo.gl/ELqZYu

4 Evaluación

A lo largo de este capítulo se procederá a evaluar la aplicación desde tres puntos de vista distintos. Primero se evaluará desde un punto de vista funcional, a continuación desde el punto de vista de la interacción persona ordenador y por último desde un punto de vista más técnico.

4.1 Pruebas funcionales

A lo largo de esta sección se presentarán una serie de pruebas realizadas sobre la aplicación para verificar que cumple todos los requisitos funcionales necesarios. Estas pruebas se dividirán en cuatro grandes bloques: selección, emparejamiento, matemáticas y funcionamiento general de la aplicación.

4.1.1 Actividades de selección

A continuación se detalla brevemente cada prueba realizada para la actividad de selección.

- Elegir opción correcta:
  - Descripción: se observará el *feedback* que recibirá el usuario al elegir una opción correcta.
  - Resultado: aparecerá una *tick* verde, el marcador sumará un punto y la tarjeta cambiará a color verde.
- Elegir opción incorrecta
  - Descripción: se observará el feedback que recibirá el usuario al elegir una opción incorrecta.
  - Resultado: aparecerá un *tick* rojo, el marcador permanecerá estático, y la tarjeta cambiará a color rojo.
- Pulsar dos tarjetas a la vez
  - Descripción: se observará qué ocurre al intentar elegir dos opciones a la vez.
  - Resultado: se recogerá solo la opción realizada en primer lugar, al no permitirse dos acciones al mismo tiempo.

4.1.2 Actividades de emparejamiento

A continuación se detalla brevemente cada prueba realizada para la actividad de emparejamiento.

- Emparejar la opción correcta:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta sobre la tarjeta correcta.
  - Resultado: aparecerá un *tick* verde indicando que la operación se ha realizado correctamente, la carta arrastrada desaparecerá y la carta destino quedará bordeada de color verde.
- Emparejar la opción incorrecta:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta sobre la tarjeta incorrecta.
  - Resultado: aparecerá un *tick* rojo indicando que la operación se ha realizado de manera incorrecta, y la tarjeta arrastrada volverá a su posición original.
- Tarjeta de origen no válida:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta diseñada como destino a una tarjeta diseñada como origen.
  - Resultado: al tener la actividad el objetivo de X a Y, si arrastramos Y sobre X lo detectará como error, cumpliéndose el mismo resultado que en el caso anterior.
- Arrastrar tarjeta a lugar no válido:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta sobre cualquier otro punto de la aplicación que no sea una tarjeta.
  - Resultado: aparecerá un *tick* rojo y la tarjeta arrastrada volverá a su posición original.

4.1.3 Actividades de matemáticas

A continuación se detalla brevemente cada prueba realizada para la actividad de matemáticas.

- Arrastrar tarjeta sobre otra tarjeta válida:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta sobre otra que esté configurada como destino.
  - Resultado: la tarjeta arrastrada desaparece y se suma al contador interno el valor de la tarjeta arrastrada.
- Arrastrar tarjeta sobre posición no válida:
  - Descripción: se observará qué ocurre al arrastrar una tarjeta sobre otra que no esté configurada como destino.
  - Resultado: aparecerá un *tick* rojo y la tarjeta arrastrada volverá a su posición original.
- Contador interno de la actividad excedido:

  - Descripción: se observará qué ocurre al sobrepasar el contador asociado al objetivo de la actividad.
  - Resultado: aparecerá un *tick* rojo y se habilitará la flecha para avanzar de actividad aunque ésta se haya completado erróneamente.
- Contador interno de la actividad excedido:
  - Descripción: se observará qué ocurre al alcanzar el contador asociado al objetivo de la actividad.
  - Resultado: aparecerá un *tick* verde y se habilitará la flecha para avanzar de actividad.

4.1.4 Funcionamiento general

A continuación se detalla cada prueba realizada indistintamente del tipo de actividad llevada a cabo.

- Completar una actividad
  - Descripción: se observará qué ocurre al completar una actividad una vez que todos los objetivos de la actividad estén seleccionados correctamente.
  - Resultado: se habilitará la flecha que permitirá avanzar de actividad.
- Retroceder de actividad
  - Descripción: se observará qué ocurre al pulsar sobre el botón de atrás en la actividad.
  - Resultado: se volverá a la actividad anterior pudiendo completarla de nuevo.
- Reiniciar la actividad
  - Descripción: se observará qué ocurre al pulsar en el botón de reiniciar la actividad.
  - Resultado: se volverá a cargar la actividad actual en su estado inicial.

4.2 Evaluación heurística

En todo proceso de evaluación de un desarrollo software es de gran importancia evaluar su Interfaz de Usuario (IU). Todo este proceso de evaluación es importante debido que permite garantizar los siguientes aspectos:

- Verificar si los requisitos tomados son los correctos.
- Verificar que la IU no resulta un impedimento para que el usuario realice su tarea.
- Comprobar la usabilidad y accesibilidad de la aplicación.
- Permite la corrección de fallos.

4.2.1 Métodos de evaluación utilizados

Para llevar a cabo la realización de esta evaluación se realizará primero una evaluación analítica y posteriormente una evaluación empírica. A continuación se detalla en qué consistirá cada una:

- Evaluación analítica:

Con este método se conseguirá evaluar la IU desde un punto de vista más técnico, y será realizada por personas con conocimientos sobre usabilidad utilizando las heurísticas de Nielsen (Jakob Nielsen, 1995).

Para llevar a cabo esta evaluación se cada persona comprobará una a una las heurísticas y argumentará los motivos que le han llevado a indicar el porqué de su decisión. A continuación se enumeran las heurísticas utilizadas:

1. Estado del sistema siempre visible.
2. Utilizar el lenguaje de los usuarios.
3. El usuario tiene control y libertad.
4. Hay consistencia y se siguen estándares.
5. Existe prevención de errores.
6. Minimizar la carga de la memoria del usuario.
7. Existe flexibilidad y eficiencia de uso.
8. Diálogos estéticos y diseño minimalista.
9. Ayudar a los usuarios a reconocer, diagnosticar y recuperar errores.
10. Existe ayuda y documentación.

- Evaluación empírica:

Con esta evaluación se conseguirá obtener el *feedback* desde un punto de vista más funcional, realizando para ello los pasos que se explican a continuación:

- Nube de palabras para saber cuáles son aquellas más recordadas una vez realizada las pruebas con los usuarios (Ver Anexo 3: Cuestionario nube de palabras)[11].
- Cuestionario para verificar los conceptos de usabilidad más importantes: facilidad de aprendizaje, flexibilidad y robustez (Ver Anexo 4: Encuesta de usabilidad)[12].

[11] Nube de palabras: https://goo.gl/30QRHa

- Cuestionario para verificar que la aplicación cumple los estándares de accesibilidad (Ver Anexo 5: Encuesta de accesibilidad)[13].
- Cuestionario de satisfacción con algunas preguntas libres para obtener el *feedback* de los usuarios al probar la aplicación (Ver Anexo 6: Cuestionario de satisfacción)[14].

4.2.2 Análisis de los resultados

En la realización de esta evaluación heurística han participado cinco personas con perfiles muy diversos. Dos de las personas que han realizado la evaluación no tienen conocimientos técnicos avanzados y su edad oscila entre los 50-60 años. Los otros tres participantes si tienen un perfil más técnico y están más familiarizados con el mundo de la tecnología y los dispositivos electrónicos, como tabletas, en particular y cuya edad oscila entre los 20 y 30 años.

4.2.2.1 Heurísticas de Nielsen

- **Estado del sistema siempre visible.**

En todo momento se puede observar el estado del sistema, ejemplo de ello tenemos el marcador que informa de la puntuación al usuario o la habilitación o inhabilitación de los botones para avanzar de actividad.

- **Utilizar el lenguaje de los usuarios.**

Todo el lenguaje que se utiliza en la aplicación proviene de los ficheros generados por DEDOS-Editor, es decir, el lenguaje que aparezca en la misma dependerá de lo establecido por los creadores de las actividades.

- **El usuario tiene control y libertad.**

El usuario tendrá total libertad para retroceder o reiniciar la actividad en la que se encuentre, así como avanzar en el caso de que haya completado los objetivos asociados a la misma.

- **Hay consistencia y se siguen estándares.**

Toda la aplicación en consistente en cuanto a su diseño independientemente del tipo de actividad en la que se encuentre el usuario.

- **Existe prevención de errores.**

[12] Encuesta usabilidad: https://goo.gl/AIYR03
[13] Encuesta accesibilidad: https://goo.gl/zkf8va
[14] Encuesta de satisfacción: https://goo.gl/0zbWWf

Toda acción realizada por el usuario tiene una retroalimentación asociada ya sea en caso de acierto o fallo en una actividad.

- **Minimizar la carga de la memoria del usuario.**

Debido a que el estado del sistema es en todo momento visible para el usuario éste no tendrá la necesidad de memorizar las acciones a realizar.

- **Existe flexibilidad y eficiencia de uso.**

Para que sepa los logros conseguidos, así como los mensajes con el resultado final una vez completadas todas las actividades del proyecto.

4.2.2.2 Nube de palabras

Para realizar la nube de palabras (véase la Figura 50) pedimos a los entrevistados que de dos listados con adjetivos, en el que cada adjetivo positivo tiene su contrario, intentasen elegir 5 adjetivos con el fin de obtener un muestreo importante.

El resultado de esta evaluación es bastante positivo ya que de las 3 palabras más repetidas, 2 son adjetivos positivos: fácil de usar y familiar; y la tercera corresponde a un adjetivo negativo: impredecible.

Figura 50 Nube de palabras

Al ser una aplicación destinada a niños principalmente una de las claves que se tienen que valorar es su facilidad de uso, ya que al tratarse de una aplicación educativa, si además se necesita una gran habilidad para utilizarla esto generaría cierto rechazo en los alumnos. Relacionado con este adjetivo encontramos la palabra familiar. Al preguntar a los encuestados por qué habían seleccionado esta opción nos trasmitieron que sobre todo en el caso de las

actividades de selección resultaba similar a otras aplicaciones tipo pregunta y respuesta, por lo que no les resultaba ajeno.

En cuanto al adjetivo negativo que más se repite, impredecible, los encuestados nos argumentaron que habían elegido esta opción debido a que en ciertas ocasiones no se sabían a priori las consecuencias de una acción, como por ejemplo depositar una tarjeta en un lugar que no fuese otra tarjeta o actividades que no contenían enunciado, y que esto aunque a ellos no les causaba confusión quizás a los niños sí en el caso de que realizaran la actividad sin ayuda de un profesor que conociese la herramienta. Para mejorar esto, al margen de contar con que el profesor debería introducir un breve texto indicando lo que hacer en la actividad, se podría introducir algún texto de ayuda inicial o contar con un botón de ayuda para usuarios menos avanzados.

A continuación se muestra el resultado global de los adjetivos positivos (véase Figura 51) y negativos (véase Figura 52).

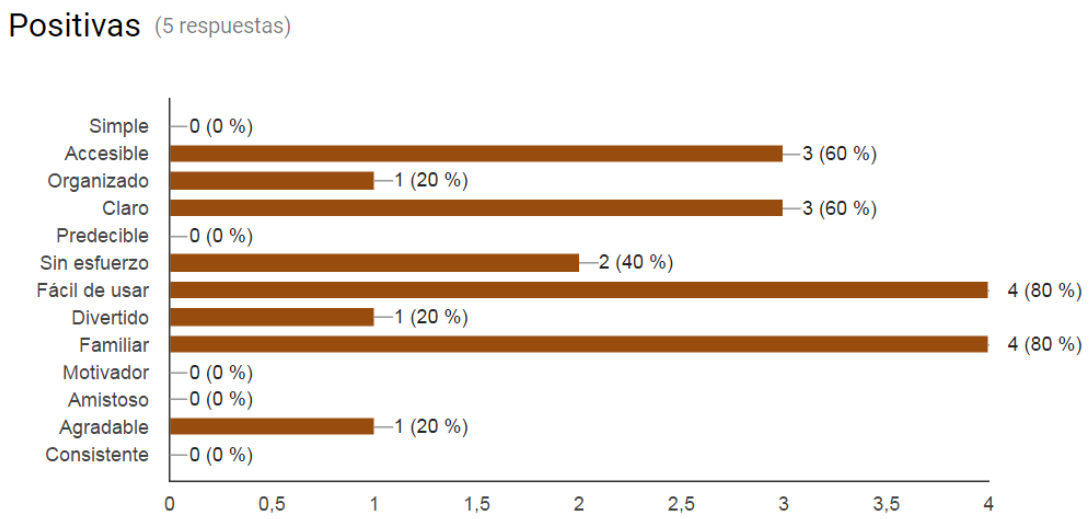


Figura 51 Adjetivos positivos nube de palabras

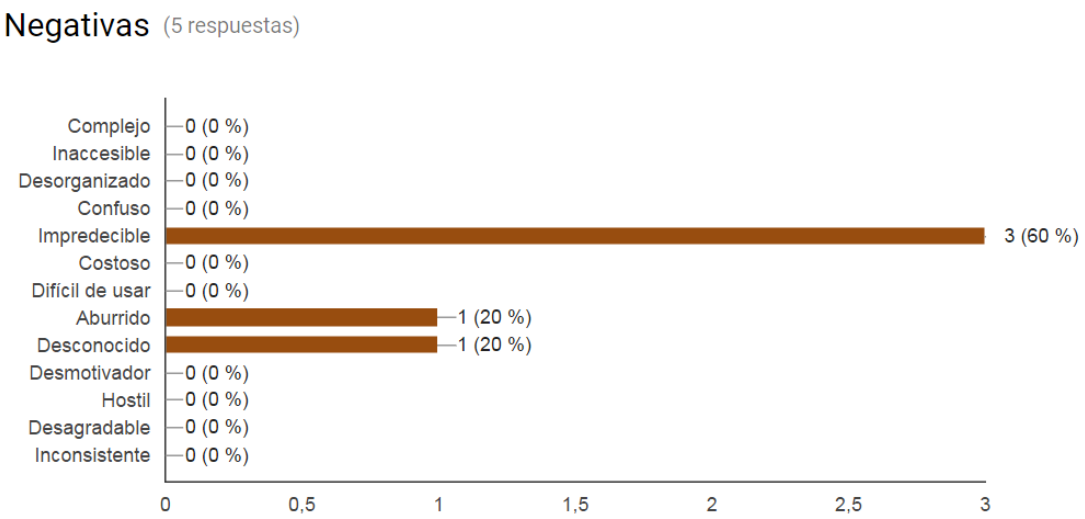


Figura 52 Adjetivos negativos nube de palabras

4.2.2.3 Cuestionario usabilidad

Con este cuestionario se ha evaluado la usabilidad de la interfaz gráfica con distintas preguntas diseñadas por instituciones de referencia, entre las que se encuentran la Universidad de Maryland o IBM[15].

De cara al análisis de todas las preguntas se van a analizar aquellas que hayan obtenido unos resultados más homogéneos en cuanto a las puntuaciones otorgadas y por tanto los encuestados estaban más de acuerdo.

Tal y como se ha analizado en la nube de palabras una de las más repetidas fue fácil de usar, y en este cuestionario cuatro de cinco encuestados consideran que están "*Totalmente de acuerdo*" con la afirmación "*Fue sencillo aprender cómo usar este sistema*" con lo que se consigue nuestro objetivo de tener una aplicación fácil de usar y que no requiera un esfuerzo extra para que el usuario aprenda a usarla.

Por otro lado a la afirmación "*El formato es amigable para la vista*" cuatro de cinco encuestados consideran que están "*Totalmente de acuerdo*". Al tratarse sobre todo de un público más infantil se considera esta una afirmación bastante importante en el análisis, debido a que este público a priori necesita tener una aplicación vistosa y agradable para que no les produzca rechazo incluso antes de usarla en profundidad.

[15] Web con preguntas sobre usabilidad: http://garyperlman.com/quest/

En cuanto a aspectos a mejorar principalmente se pueden obtener dos recomendaciones establecidas por los usuarios. En primer lugar a la afirmación "*Hay ayuda para el usuario*" los encuestados han considerado que la ayuda depende demasiado de cómo cree el profesor la actividad con DEDOS-Editor, es decir, si el profesor crea una actividad sin ningún texto de ayuda los alumnos no tendrán ese apoyo extra. Más adelante veremos las recomendaciones que nos han sugerido para mejorar este aspecto.

Otro aspecto a mejorar se deriva de la pregunta "*Se necesita hacer más de una acción para volver al menú principal de la app*", ya que algunos encuestados consideraron que en ocasiones en las que quizás haya un número elevado de actividades dentro de un mismo proyecto sería tedioso dar al botón de atrás hasta llegar al menú principal. Por otro lado con esto se consigue que el alumno se centre en terminar todas las actividades y no abandone, o que quizás realice en otro momento el resto de actividades que le queden por completar.

Como conclusión de este cuestionario los encuestados de manera general se encuentran satisfechos con el sistema, ya que a la pregunta "*Estoy satisfecho con este sistema*" dos encuestados están "*Totalmente de acuerdo*" y tres "*Están de acuerdo*".

4.2.2.4 Cuestionario accesibilidad

Con este cuestionario se pretende evaluar en qué medida nuestra aplicación puede ser usada por un mayor rango de personas con capacidades distintas. Como se ha comentado en el apartado de Experiencias educativas con DEDOS en Educación Infantil, Educación Primaria y Educación Especial, la aplicación de DEDOS-Player ha sido probada en alumnos con diferentes necesidades y con resultados muy positivos, por lo que a priori cumpliría con este aspecto.

A continuación se procede a analizar los resultados obtenidos con este cuestionario de accesibilidad.

Una de las preguntas que ha tenido unanimidad ha sido "*El diseño se puede usar eficientemente y confortablemente con un mínimo de fatiga*" que estaría relacionado con la facilidad de uso que se comentaba en apartados anteriores, y los 5 encuestados han contestado que están "*Totalmente de acuerdo*" con dicha afirmación.

A pesar de que los usuarios eligieron en la nube de palabras la palabra "*Impredecible*" como uno de los adjetivos negativos destacados, a la afirmación "*El diseño minimiza posibles incidentes por azar y las consecuencias adversas de acciones no previstas*" la mayoría de encuestados está "*Totalmente de acuerdo*" con esta afirmación. Con esto se deduce que a pesar de que en ciertas ocasiones el usuario no sabe a priori qué consecuencia tendrá alguna

de sus acciones, éstas a pesar de resultar erróneas, el sistema le provee mecanismos suficientes para solventar el error.

4.2.2.5 Encuesta de valoración

Con esta encuesta se pretende que los encuestados de manera libre expresen su opinión acerca de la aplicación con el fin de aportar tanto comentarios positivos como negativos acerca de la misma.

Aquí vuelve a surgir el aspecto que más se ha repetido a lo largo de toda la evaluación, y es la facilidad de uso que han encontrado los usuarios a la hora de utilizar la aplicación. Por otro lado también les ha parecido interesante la variedad de actividades que hay disponibles o la buena retroalimentación que se tiene al resolver las actividades.

Como aspectos a destacar opinan que sería interesante añadir una manera para cargar los proyectos desde Dropbox o alguna aplicación similar, o que hubiera algún tipo de botón para volver al menú principal de elección de juegos. También se consideraría útil si en las actividades que no haya enunciado por parte del profesor, existiera un tipo de botón de ayuda en el cual se detectara en qué tipo de actividad nos encontramos y poder dar una orientación a los alumnos.

4.3 Validación técnica

Para la realización de la validación técnica nos ayudaremos de una herramienta de software llamada Kiuwan[16] con la que se procederá a analizar el código generado en el desarrollo de nuestra aplicación.

4.3.1 Complejidad ciclomática

Con esta métrica se obtiene la medición cuantitativa de la complejidad algorítmica del código, definiendo el número de caminos independientes o saltos en el código. Esa métrica la propuso Thomas McCabe estableciendo los siguientes rangos:

- Menor o igual que 10: método sencillo, sin mucho riesgo.
- Entre 10 y 20: métodos medianamente complejos, con riesgo moderado.
- Entre 20 y 50: métodos complejos y con alto riesgo.
- Mayor que 50: métodos inestables, de altísimo riesgo.

[16] Web de Kiuwan: https://goo.gl/pjk0iW

En nuestro caso la complejidad por función es de 1.84 (véase Figura 53) por lo que entraría dentro del primer grupo con métodos sencillos y sin mucho riesgo.

Complexity by function	1.84	Cyclomatic complexity	562
Maximum value in a file	7.38	Maximum value in a file	106
Minimum value in a file	1	Minimum value in a file	0
Average per file	1.73	Average per file	15.19

Figura 53 Complejidad código

4.3.2 Código repetido

Con esta métrica se puede detectar la proporción total de código que se encuentra repetido a lo largo de la aplicación. Un código totalmente limpio de código repetido tendría un valor de un 0% y en nuestro caso se obtiene un 1,3% de código repetido (véase Figura 54).

Duplicated code ratio	0.13
Maximum value in a file	0.81
Minimum value in a file	0.09
Average per file	0.34

Figura 54 Código repetido

4.3.3 Indicadores de cumplimiento

Con estos indicadores se pretende mostrar el porcentaje de cumplimiento con respecto a diferentes características: nivel general, eficiencia, mantenibilidad o seguridad (véase Figura 55).

Global compliance indicator	95.76
Efficiency compliance indicator	95.37
Maintainability compliance indicator	92.9

Figura 55 Indicadores de cumplimiento

Como se ha podido observar a lo largo de todo este capítulo de Evaluación los resultados obtenidos son bastante satisfactorios. Por un lado se ha verificado con las pruebas funcionales que todos los requisitos funcionales que debe tener la aplicación se encuentran presentes. Por otro lado con la evaluación heurística se ha podido obtener el *feedback* de los distintos usuarios encuestados y aunque existen mejoras de cara a futuras versiones que toda aplicación ha de tener, los resultados han sido bastante satisfactorios y en los que se resalta la opinión de los usuarios en cuanto a la facilidad de uso de la aplicación, aspecto fundamental en nuestra aplicación al tratarse de una herramienta educativa. Y por último se ha verificado con diferentes métricas que además de cumplir todos los requisitos funcionales nuestra aplicación no obvia el aspecto técnico.

5 Conclusiones

Como se ha podido ver a lo largo de todo este trabajo de fin de grado se han cumplido los objetivos que se plantearon en un principio, ya que se ha conseguido la migración de la versión DEDOS-Player a tabletas con un resultado bastante satisfactorio tal y como demuestran las distintas evaluaciones que se han llevado a cabo en todo el proceso de desarrollo. En él se ha conseguido adaptar las actividades de selección, emparejamiento y matemáticas.

Al ser un proyecto bastante más prolongado en el tiempo que por ejemplo una práctica de alguna asignatura, con él se han aprendido conceptos de planificación o análisis que son pilares fundamentales de todo el desarrollo software.

Todo este proyecto se ha llevado a cabo con bastante esfuerzo, a pesar de conocer el lenguaje Java, se han tenido que obtener diversos conocimientos de programación en Android, llevados a cabo mediante un profundo autoaprendizaje con la ayuda de foros o webs especializadas. En todo este desarrollo se han tenido que aprender conceptos totalmente nuevos, o adaptaciones de librerías actuales para las necesidades del proyecto, lo que ha supuesto un reto ya que a lo largo de la carrera no ha habido este autoaprendizaje.

Se podría destacar que la aplicación DEDOS-Player para tabletas fue utilizada en una experiencia educativo con alumnos de Educación Infantil por Cristian Guzmán (Cristian Guzmán, 2015) con resultados bastante positivos. Este estudio fue realizado a veinte niños de 5 años en el Colegio Valle del Miro tratando actividades de Conocimiento del Entorno. En esta evaluación se dividió a los alumnos en dos grupos, diez utilizando métodos tradicionales y diez utilizando DEDOS-Player para tabletas. De este estudio se pueden extraer las siguientes conclusiones:

- El grupo que utilizó DEDOS-Player vio incrementado su aprendizaje de manera notable con respecto al grupo que utilizó métodos tradicionales.
- Los niños se adaptaron sin ningún problema al uso de tabletas, ya que cada vez empiezan con el uso de estos dispositivos a una edad más temprana.
- Las actividades diseñadas para este estudio fueron muy bien valoradas por la profesora responsable una vez finalizado el mismo. Cabe destacar que antes del inicio del estudio la profesora era contraria a la introducción de estas tecnologías en edades tan tempranas para su utilización en la enseñanza ya que las consideraba como un uso más lúdico que educativo.

- Al usar DEDOS-Player mejoró notablemente el clima del aula y mejoró el ambiente de trabajo ya que se les presentó a los alumnos la utilización de la tableta como un premio por lo que mejoró su motivación.

Actualmente se encuentran en curso otras dos experiencias educativas en el ciclo de Educación Infantil en alumnos de 2 y 5 años, incluyendo en este último grupo alumnos de altas capacidades y aquellos que precisan de necesidades educativas especiales.

Además de todo lo analizado anteriormente se puede destacar que este proyecto ha sido presentado en las VI Jornadas de Innovación y TIC Educativas (JITICE, 2015), organizadas por la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos, en el campus de Móstoles durante los días 24 y 25 de noviembre de 2015 y se encuentra otro artículo enviado al XI European Conference On Technology Enhanced Learning (ECTEL 2016).

Por todo lo anterior, a pesar de que siempre hay cosas que mejorar, en líneas generales el resultado de este desarrollo ha sido muy satisfactorio tanto a nivel personal como profesional.

Como trabajo que se podrían incluir en futuras versiones, se proponen las siguientes líneas:

- Migrar el entorno de desarrollo de Eclipse a Android Studio con el fin de analizar las posibles ventajas de este, al ser un producto desarrollado por Google específico para Android.
- Permitir modo multi-jugador con partidas online.
- Incluir apartado de ayuda para usuarios menos expertos.

6 Bibliografía

Ana Márquez. (2013). *Experiencia educativa con niños con autismo utilizando mesas multicontacto.*

Cristian Guzmán. (2015). *Las tabletas digitales como recurso para el aprendizaje en educación infantil.*

Cristina Fernández. (2014). *La pizarra digital interactiva en el aula de educación infantil.*

International Data Corporation. (2015). *Worldwide Tablet Growth Hits the Brakes, Slowing to the Low Single Digits in the Years Ahead*. Recuperado el 10 de Marzo de 2016, de http://goo.gl/N9bSn0

Jakob Nielsen. (1995). *Nielsen Norman Group*. Recuperado el 10 de Abril de 2016, de https://goo.gl/zdWiJm

JITICE. (2015). *VI Jornadas en Innovación y TIC Educativas.* Obtenido de https://goo.gl/55CMwE

María Rodríguez. (2013). *Influencia del uso de las tecnologías en el aprendizaje.*

Plataforma Proyecta. (2014). Recuperado el 20 de Marzo de 2016, de http://goo.gl/q1KxRR

Proyecto DEDOS. (2008). Recuperado el 20 de Marzo de 2016, de http://aprendecondedos.es/

SMART. (2016). *) Notebook 15.2*. Recuperado el 22 de Marzo de 2016, de http://goo.gl/8eq4Sz

Anexo 1: Entorno de desarrollo

A continuación se detallan los diferentes pasos a seguir para preparar el entorno de desarrollo que nos permita codificar nuestra aplicación en Android. En primer lugar se deberá instalar el IDE sobre el que se desarrollará la aplicación. En este caso, se eligió Eclipse. Para ello se deberá ir a la web oficial de descarga (véase la Figura 56) y elegiremos la versión que se adapte a nuestro sistema operativo, en este caso Windows 64-bit de la Web: http://goo.gl/BbuQ5n

Figura 56: Web de descarga de Eclipse

Una vez descargado e instalado el IDE procedemos a descargar el SDK de Android (véase la Figura 57), para lo que se irá al siguiente enlace: http://developer.android.com/sdk/index.html

SDK Tools Only

If you prefer to use a different IDE or run the tools from the command line or with build scripts, you can instead download the stand-alone Android SDK Tools. These packages provide the basic SDK tools for app development, without an IDE. Also see the SDK tools release notes.

Platform	Package	Size	SHA-1 Checksum
Windows	installer_r24.4.1-windows.exe (Recommended)	151659917 bytes	f9b59d72413649d31e633207e31f456443e7ea0b
	android-sdk_r24.4.1-windows.zip	199701062 bytes	66b6a6433053c152b22bf8cab19c0f3fef4eba49
Mac OS X	android-sdk_r24.4.1-macosx.zip	102781947 bytes	85a9cccb0b1f9e6f1f616335c5f07107553840cd
Linux	android-sdk_r24.4.1-linux.tgz	326412652 bytes	725bb360f0f7d04eaccff5a2d57abdd49061326d

Figura 57: Web descarga SDK Android

En nuestro caso, debido a que no estamos desarrollando sobre el IDE Android Studio, procederemos a descargar solamente el SDK para la plataforma que necesitemos. Una vez descargado el SDK se procederá a su instalación.

Cuando se encuentre instalado el SDK se podrá ejecutar el SDK Manager, que nos permitirá instalar o desinstalar las diferentes API's o herramientas (véase la Figura 58).

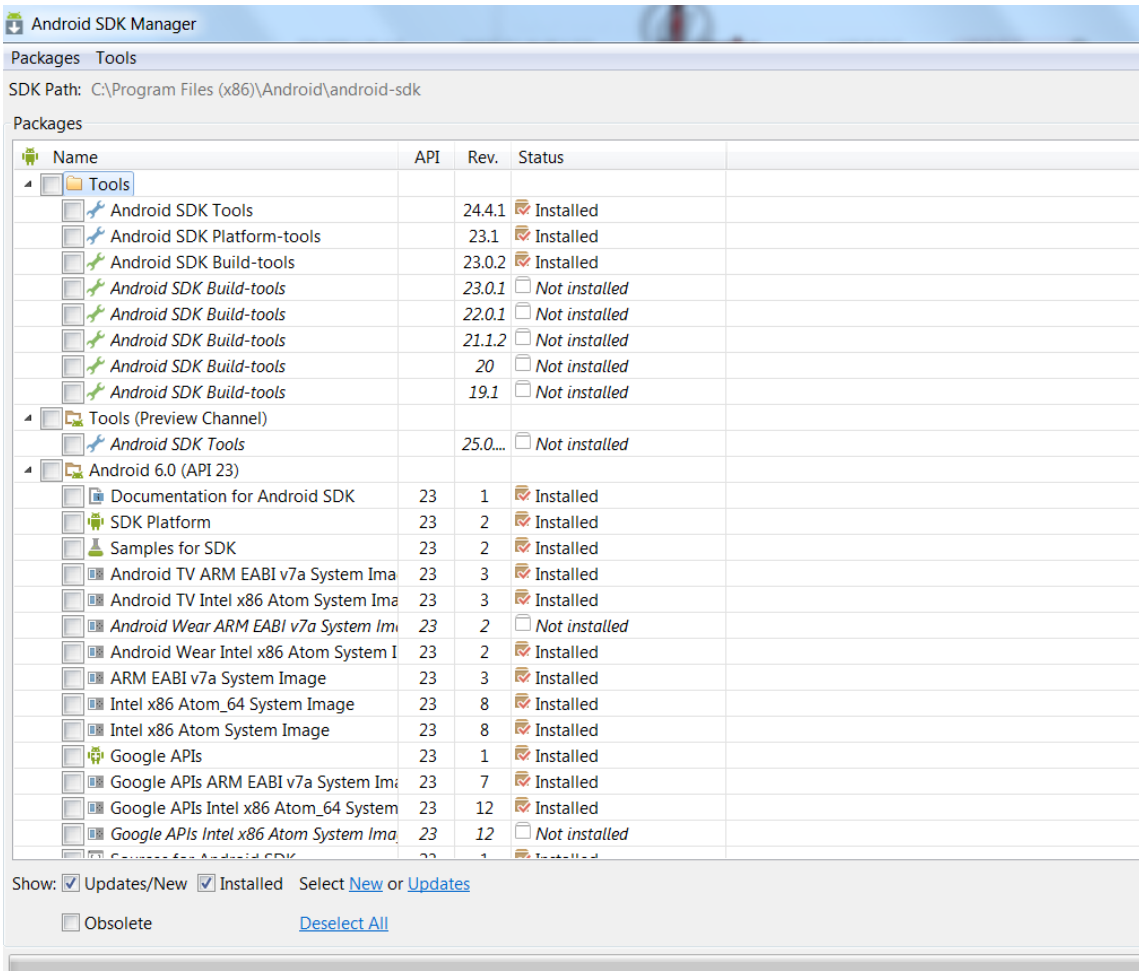


Figura 58: Android SDK Manager

Llegados a este punto tenemos dos programas individuales, pero no vinculados entre sí:

1. Eclipse
2. SDK de Android

A continuación se detallan los pasos necesarios para vincular ambos programas y poder desarrollar en Android. Se abrirá Eclipse y se hará pulsará sobre *Help/Install New Software* (véase la Figura 59).

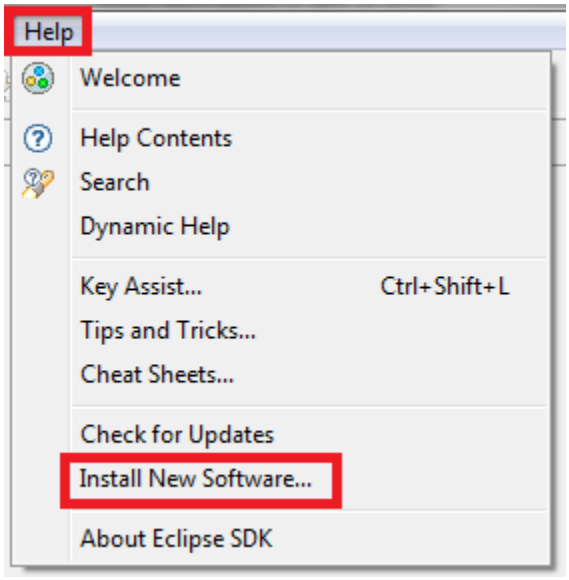


Figura 59: Pestaña Help>> Install new software de Eclipse

Aparecerá una ventana donde pulsaremos sobre el botón de añadir (Add) y aparecerá una ventana emergente en la que se elegirá un nombre para nuestro Plugin, por ejemplo "Plugin Android", y en la ruta se copiará lo siguientehttps://dl-ssl.google.com/android/eclipse/ (véase la Figura 60).

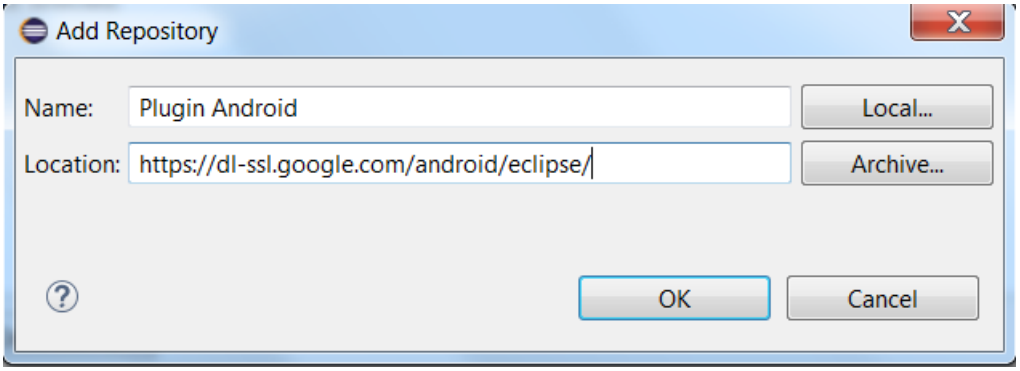


Figura 60: Repositorio de instlación

Tras esto aparecerá lo siguiente una ventana (véase la Figura 61) para seleccionar todo el software disponible para su instalación que contenga el repositorio.

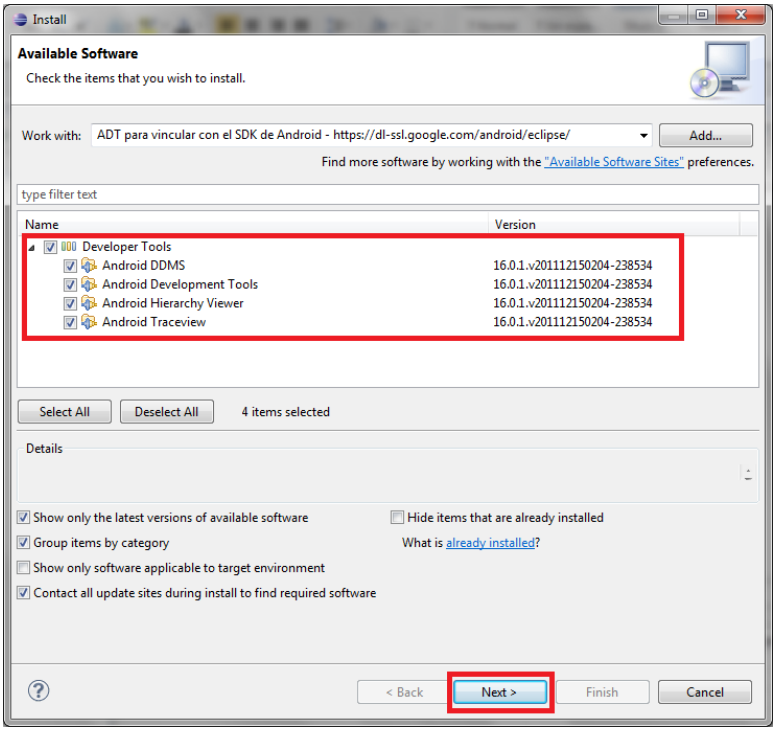


Figura 61: Selección del software disponible en el repositorio

Se llevará a cabo una instalación que pedirá el reinicio de Eclipse. Tras el reinicio se nos preguntará si se quiere instalar un SDK o utilizar una ruta existente. Se elegirá la última opción y se introducirá la ruta donde se instaló el SDK previamente.

Con todo lo anterior nuestro sistema estará listo para desarrollar una aplicación Android.

Anexo 2: Configuración de DEDOS - Player para Android

Este anexo explica cómo realizar una correcta instalación de la aplicación DEDOS – Player para tabletas Android. Primeramente habrá que comprobar que tengamos la tableta actualizada a la versión 4.0.4 o superior, ya que si no, la versión de DEDOS-Player podría no funcionar. Para ello, iremos a los ajustes de nuestra tableta y buscaremos la opción "*Acerca del Dispositivo*" (véase la Figura 62).

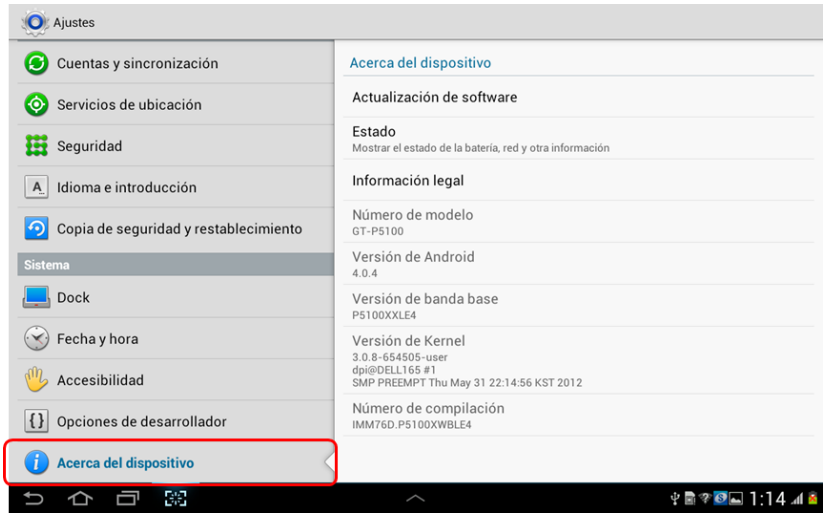


Figura 62 Información sobre la versión de Android de la tableta

Después de realizar la verificación de que la tableta cuenta con la versión adecuada se deberá descargar el APK de la web http://aprendecondedos.es/descargarte/aplicacion/.

Una vez con la aplicación instalada en nuestro dispositivo se necesitará introducir los diferentes juegos con las actividades que se hayan creado. Para ello primeramente se iniciará la aplicación y se comprobará que no hay ningún juego disponible en la misma (véase la Figura 63).

Figura 63 Pantalla inicio sin juegos

Tras esto se habrá creado automáticamente una carpeta llamada DEDOS en el almacenamiento de nuestra tableta (véase la Figura 64).

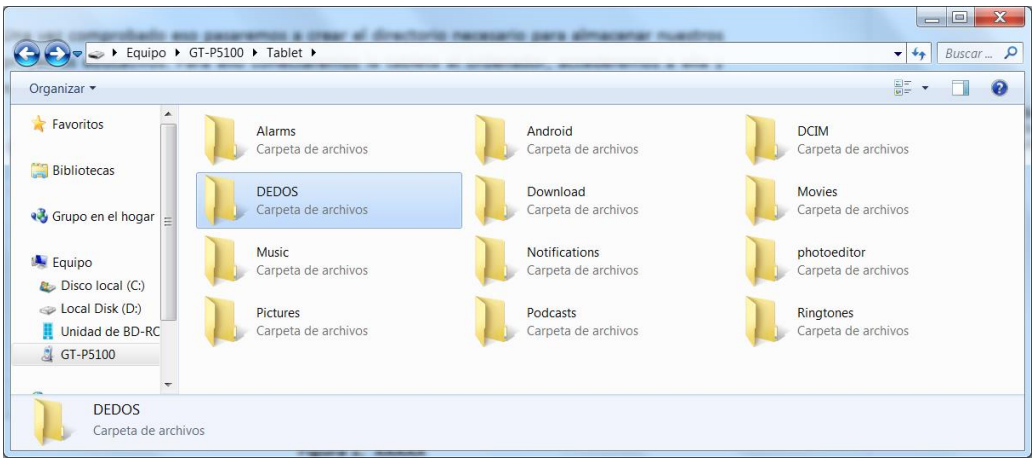


Figura 64 Carpeta DEDOS

Dentro de la carpeta DEDOS se habrá creado otra carpeta llamada Juegos (véase la Figura 65):

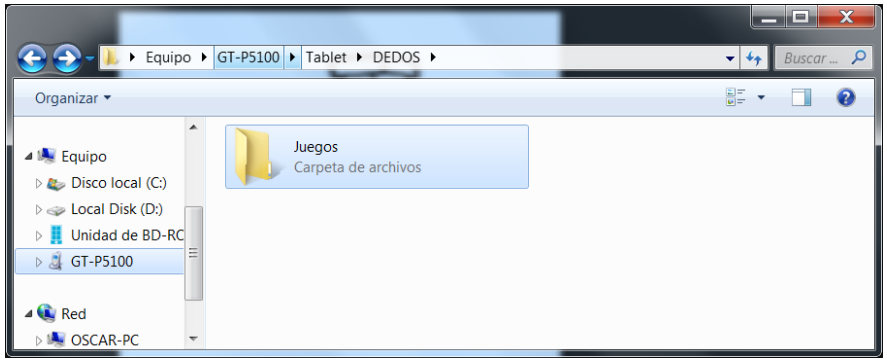


Figura 65 Carpeta Juegos

Dentro de la carpeta Juegos podremos almacenar los proyectos educativos que vayamos creando. Siempre hay que comprobar que la carpeta que contiene el proyecto educativo tiene el siguiente aspecto: nombre de la carpeta que contiene el proyecto educativo, y dentro de ésta tendrán que existir un fichero con extensión XML que es el que tiene el proyecto en sí y una carpeta *contents* con las distintas imágenes del proyecto. La Figura 66 muestra los proyectos educativos que están almacenados en nuestra tableta dentro de "DEDOS/Juegos" y en la Figura 67 se muestra la estructura de un proyecto educativo llamado "Acierta-y-gana" donde se ve que existe el fichero con el nombre exactamente el mismo que la carpeta y extensión xml y una carpeta llamada *contents* con las imágenes del proyecto.

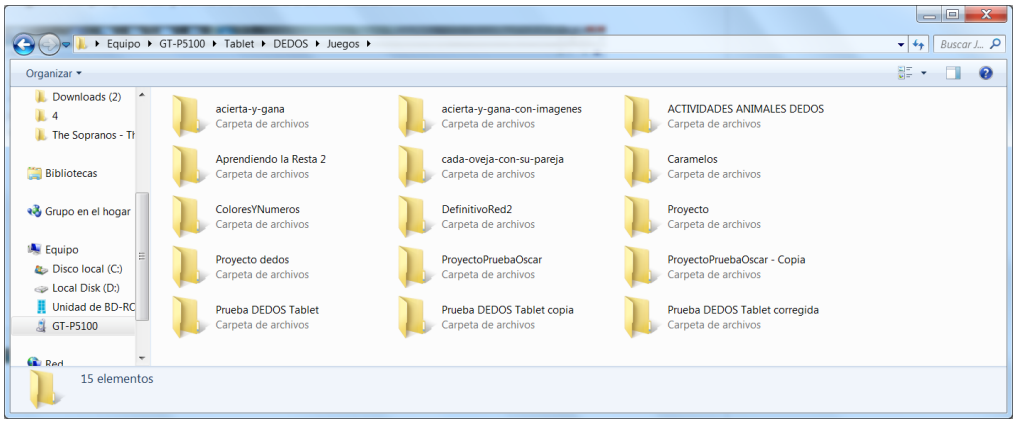


Figura 66 Proyectos educativos almacenados en nuestra tableta

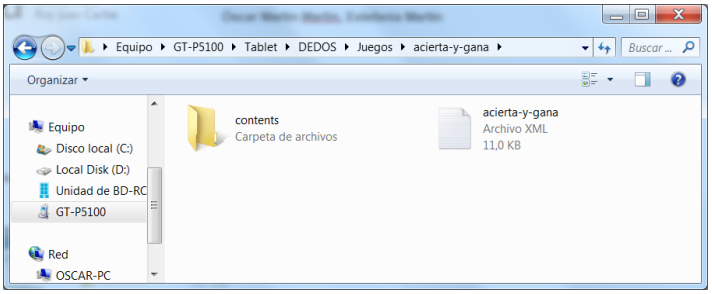
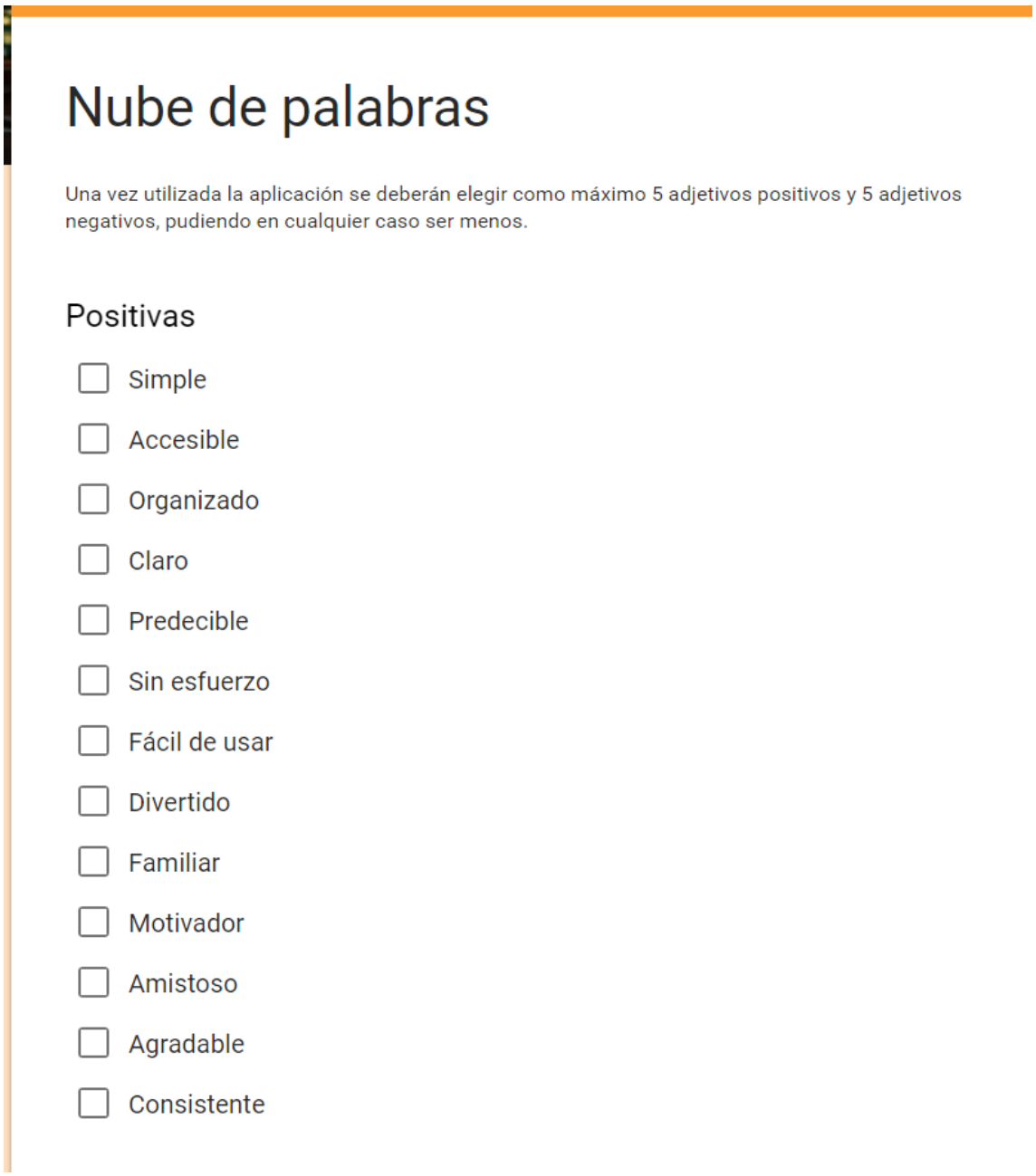


Figura 67 Estructura de un proyecto educativo

Anexo 3: Cuestionario nube de palabras



Nube de palabras

Una vez utilizada la aplicación se deberán elegir como máximo 5 adjetivos positivos y 5 adjetivos negativos, pudiendo en cualquier caso ser menos.

Positivas

- [ ] Simple
- [ ] Accesible
- [ ] Organizado
- [ ] Claro
- [ ] Predecible
- [ ] Sin esfuerzo
- [ ] Fácil de usar
- [ ] Divertido
- [ ] Familiar
- [ ] Motivador
- [ ] Amistoso
- [ ] Agradable
- [ ] Consistente

Figura 68 Cuestionario nube de palabras (1 de 2)

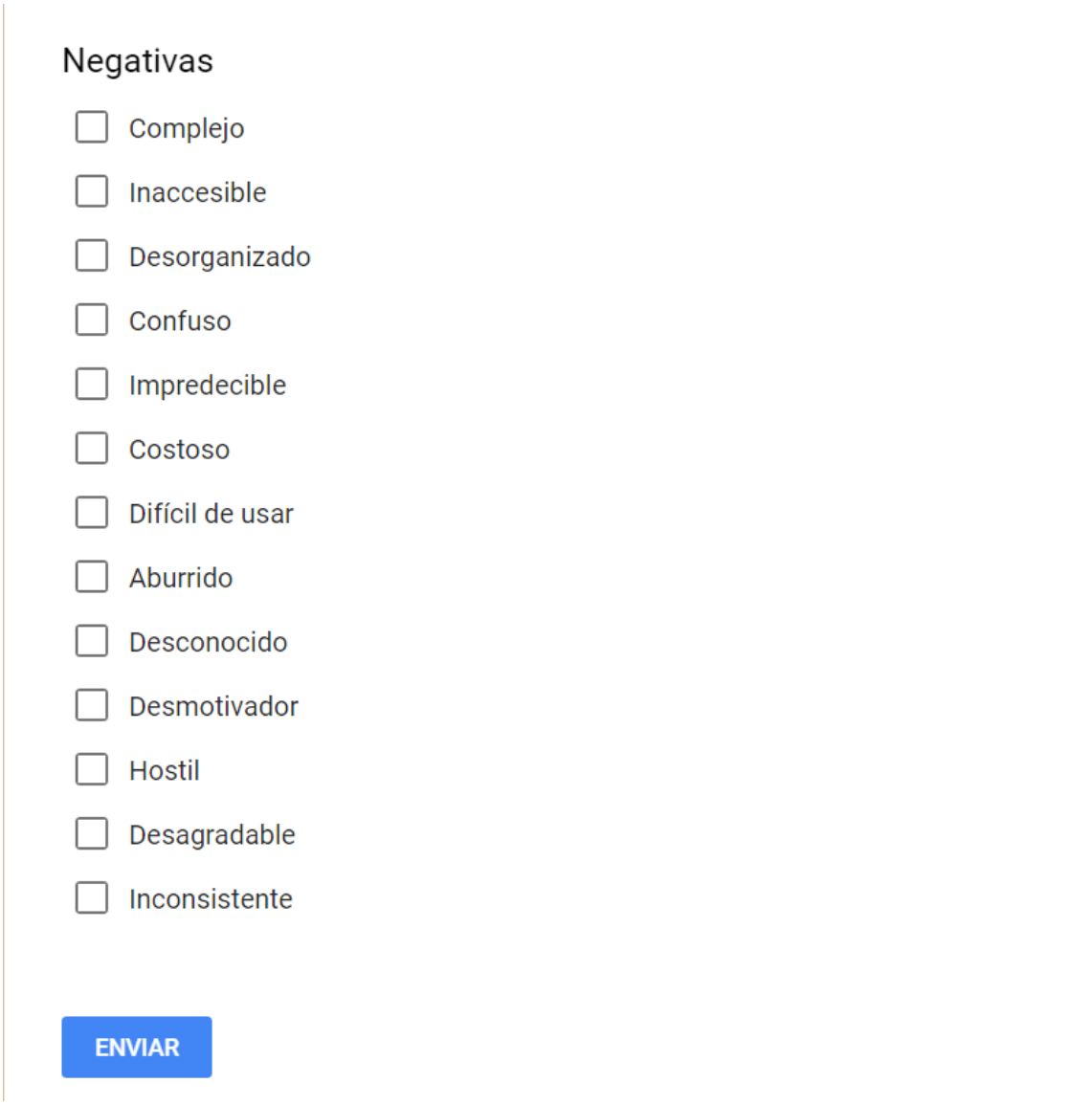
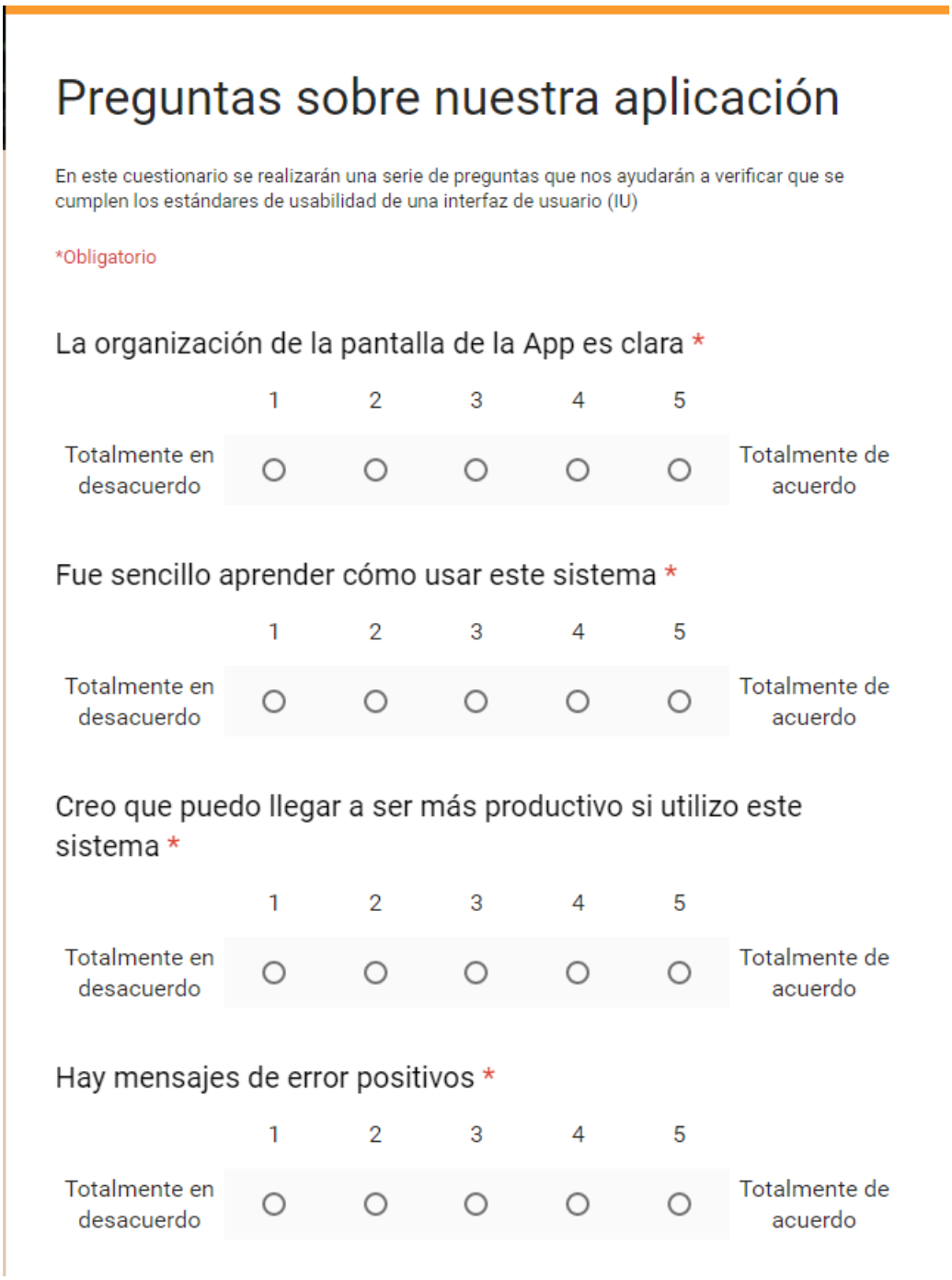


Figura 69 Cuestionario nube de palabras (2 de 2)

Anexo 4: Encuesta de usabilidad



Preguntas sobre nuestra aplicación

En este cuestionario se realizarán una serie de preguntas que nos ayudarán a verificar que se cumplen los estándares de usabilidad de una interfaz de usuario (IU)

*Obligatorio

La organización de la pantalla de la App es clara *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Fue sencillo aprender cómo usar este sistema *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Creo que puedo llegar a ser más productivo si utilizo este sistema *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Hay mensajes de error positivos *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Figura 70 Encuesta de usabilidad (1 de 3)

Se necesita hacer más de una acción para volver al menú principal de la app *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Los items en pantalla están ordenados *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Se distinguen bien los elementos de la pantalla *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

El formato es amigable para la vista *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Hay ayuda para el usuario *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Figura 71 Encuesta de usabilidad (2 de 3)

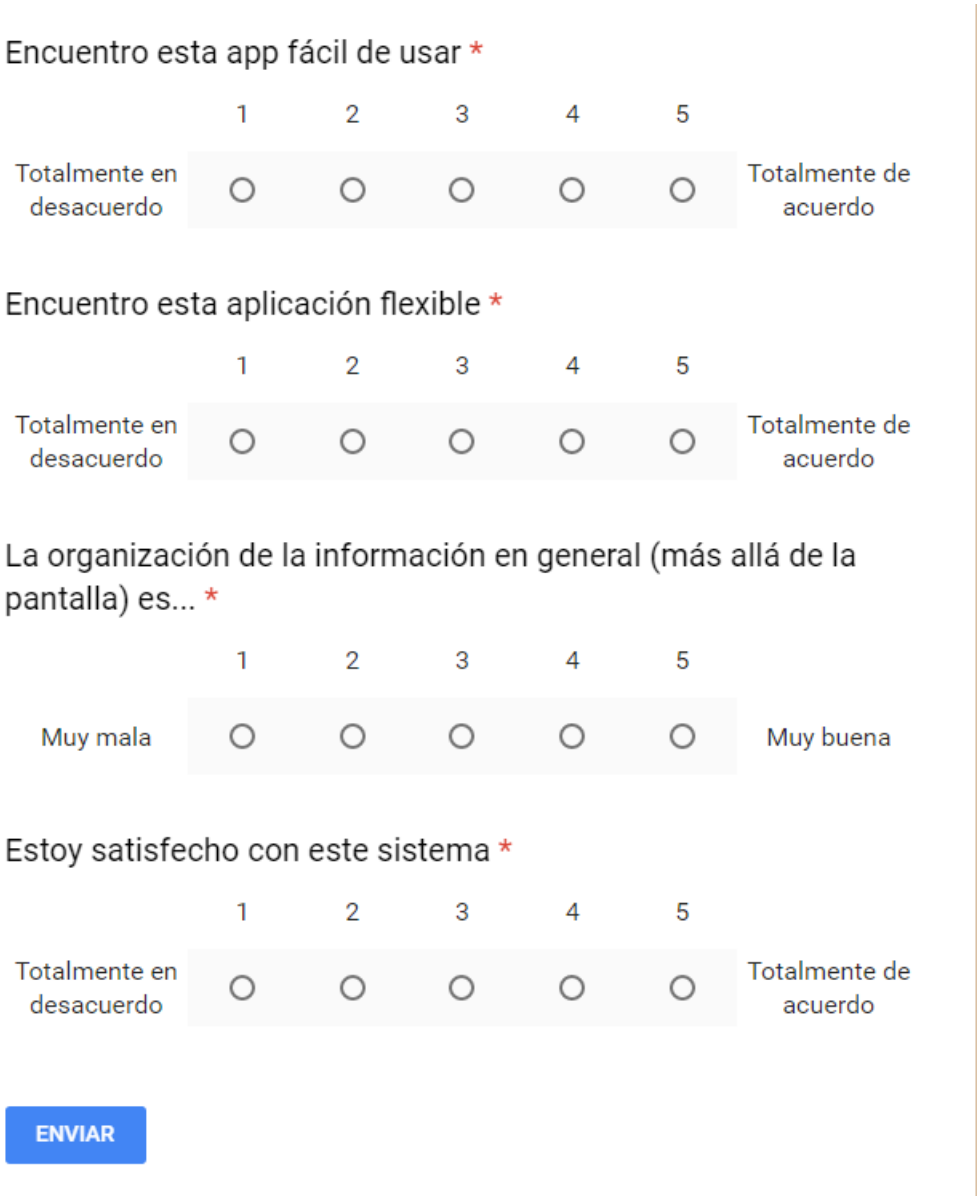
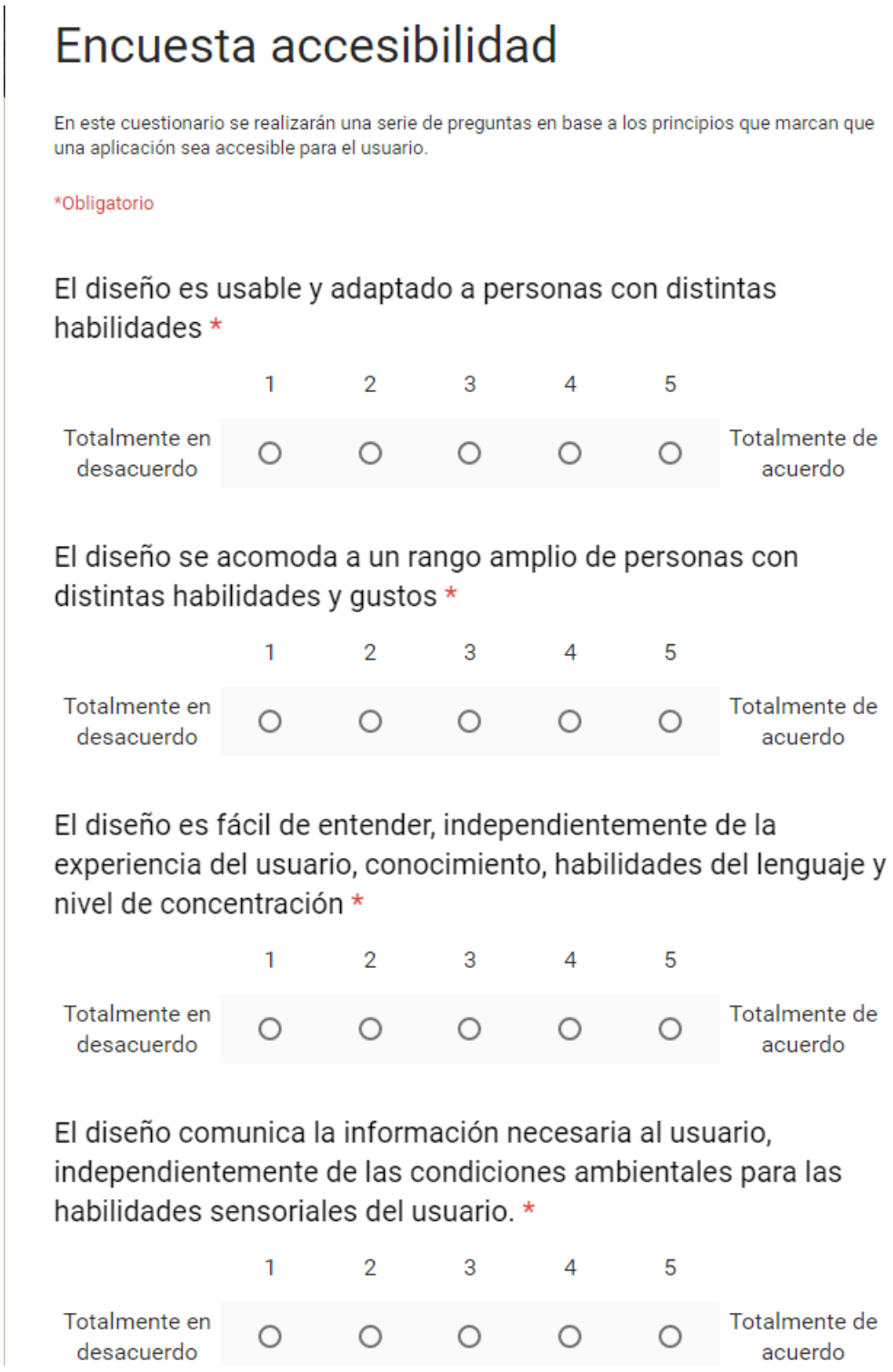


Figura 72 Encuesta de usabilidad (3 de 3)

Anexo 5: Encuesta de accesibilidad



Encuesta accesibilidad

En este cuestionario se realizarán una serie de preguntas en base a los principios que marcan que una aplicación sea accesible para el usuario.

*Obligatorio

El diseño es usable y adaptado a personas con distintas habilidades *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

El diseño se acomoda a un rango amplio de personas con distintas habilidades y gustos *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

El diseño es fácil de entender, independientemente de la experiencia del usuario, conocimiento, habilidades del lenguaje y nivel de concentración *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

El diseño comunica la información necesaria al usuario, independientemente de las condiciones ambientales para las habilidades sensoriales del usuario. *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Figura 73 Encuesta de accesibilidad (1 de 2)

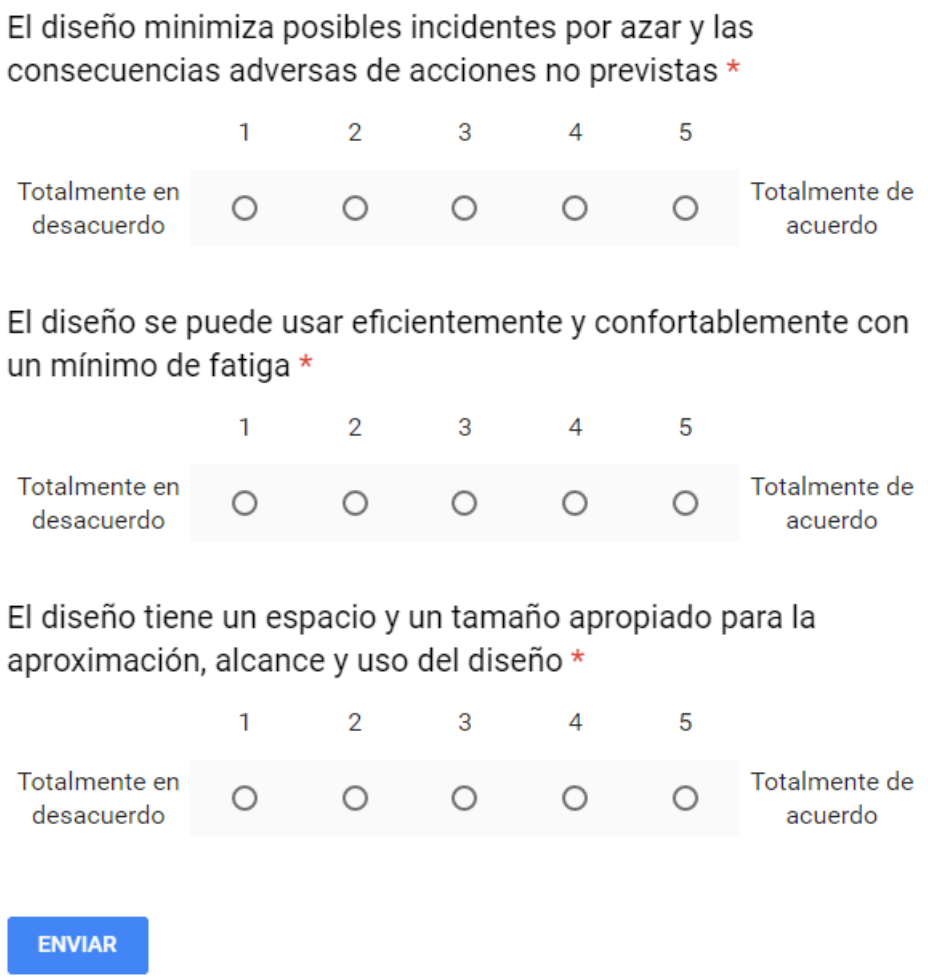


Figura 74 Encuesta de accesibilidad (2 de 2)

Anexo 6: Cuestionario de satisfacción

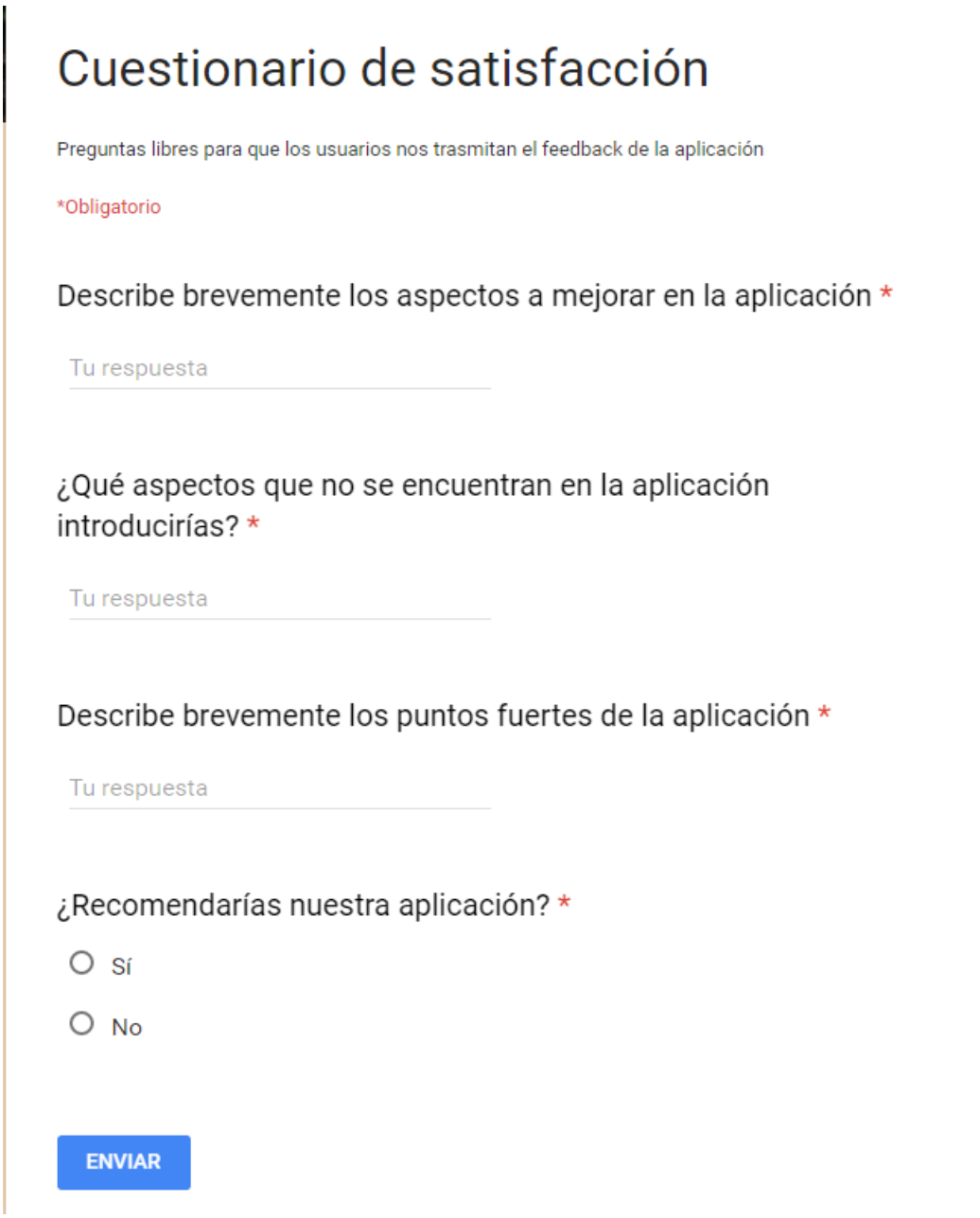


Figura 75 Cuestionario de satisfacción