Universidad
Rey Juan Carlos

Escuela Técnica Superior de Ingeniería de Telecomunicación

Grado en Ingeniería en Sistemas de Telecomunicación

Trabajo Fin de Grado

Aplicación de estimulación cognitiva en Android

Autora: Evelin Fuster Varillas
Tutor: José María Cañas Plaza

Curso Académico 2014/2015

Trabajo Fin de Grado
Aplicación de estimulación cognitiva en Android

Autora
Evelin Fuster Varillas

Tutor
José María Cañas Plaza

La defensa del presente Trabajo Fin de Grado se realizó el día de de 2015, siendo evaluado por el siguiente tribunal:

Presidente:

Vocal:

Secretario:

y habiendo obtenido la siguiente Calificación:

Fuenlabrada, a de de 2015

Agradecimientos

A mi familia por el apoyo y la confianza incondicional, por haberme soportado y ayudado en todo lo que podían y más. En especial a mis padres Nidia V. y Hector F por ser mi inspiración y ejemplo, y a mis hermanos Erick F. y Carolina F. y mi cuñada Anja F. por estar siempre ahí.

A mi novio Ivan G. por toda la ayuda que me ha brindado para hacer posible la entrega de este Trabajo de Fin de Grado.

A mis amigos Alex S, Lidia B, Bea L. y Javier E por todos los momentos que hemos pasado en nuestra etapa universitaria y el apoyo mutuo durante todo este tiempo.

A los profesores que he tenido en todas mis etapas de aprendizaje y a las personas de las cuales he aprendido. Especialmente a mi tutor José María C. por su ayuda e infinita paciencia.

A las terapeutas del *Centro de Alzheimer Fundación Reina Sofía* Carolina Mendoza, Irene Rodríguez y Almudena Pérez por la colaboración en el desarrollo de la aplicación.

"En el mundo actual se está invirtiendo cinco veces más en medicamentos para la virilidad masculina y silicona para las mujeres que en la cura del Alzheimer. De aquí en algunos años tendremos viejas de tetas grandes y viejos con pene duro, pero ninguno de ellos se acordará para que sirven."

Dr. Drauzio Varella

Contenido

Índice de figuras

Índice de cuadros

Resumen

La esperanza de vida del ser humano ha aumentado de manera significativa, lo que genera un aumento exponencial de personas mayores de 65 años. Esto ha propiciado el aumento de casos de personas que padecen enfermedades neurodegenerativas asociadas con la edad, como la enfermedad de Alzheimer o la demencia senil. Son patologías de evolución lenta que de momento no cuentan con ningún tratamiento que revierta el proceso de degeneración. Algunos fármacos pueden retrasar la progresión de dichas patologías, aunque, actualmente los ejercicios de estimulación cognitiva son el principal tratamiento para prevenir y ralentizar en la medida de lo posible la evolución de las demencias.

El presente trabajo de fin de grado tiene como objetivo poner las tecnologías de la información y la comunicación (TIC) al servicio de personas mayores que sufren deterioro cognitivo. Para ello, se ha desarrollado una aplicación Android que sirve como herramienta de estimulación cognitiva. La aplicación consta de un conjunto de ejercicios, presentados a modo de juegos, que sirven para prevenir y ralentizar el deterioro cognitivo. Están agrupados en siete categorías: Atención, Gnosias, Funciones ejecutivas, Esquema corporal, Razonamiento, Reconocer emociones y Puzzles. Cada categoría se centra en una capacidad cognitiva concreta. Cada juego de la aplicación está diseñado para ser totalmente configurable y almacenar la información relevante de cada usuario.

Para el desarrollo del proyecto se ha contado con la colaboración de tres terapeutas del *Centro de Alzheimer Fundación Reina Sofía*, las cuales nos proporcionaron la información necesaria para diseñar juegos que se ajusten a ejercicios de estimulación cognitiva existentes.

Capítulo 1
Introducción

En el presente Trabajo fin de grado se presentará el desarrollo de una aplicación Android, cuyo propósito es el de servir como herramienta de apoyo en terapias de estimulación cognitiva para personas con Alzheimer.

En este capítulo presentaremos el contexto de este TFG, desde el marco general de las enfermedades neurodegenerativas y sus terapias hasta el uso de Android en estimulación cognitiva.

1.1. Enfermedades neurodegenerativas

En apenas un siglo y gracias a los avances en medicina y nutrición, la media mundial de la esperanza de vida del ser humano ha pasado de 40 a 60 años[1]. Como se puede observar en la figura 1.1, el aumento de la esperanza de vida ha provocado un incremento exponencial de personas mayores de 60 años, lo que a su vez ha provocado el aumento de enfermedades neurodegenerativas asociadas con el envejecimiento, como el Alzheimer o la demencia senil.

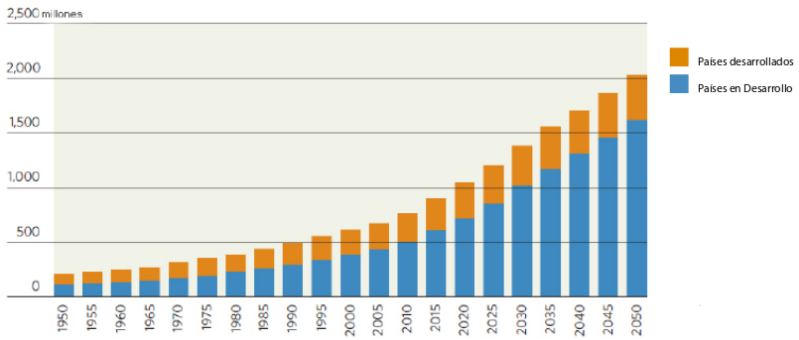

Figura 1.1. Porcentaje de personas mayores de 65 años de 1950 a 2050.[2]

[1]La esperanza de vida media en países desarrollados puede llegar a superar los 80 años.

[2]Imagen obtenida del documento Envejecimiento en el Siglo XXI: Una Celebración y un Desafío[2]

Las enfermedades neurodegenerativas se caracterizan por la degeneración y muerte de las neuronas[3], lo que se traduce en un déficit cognitivo[4] que afecta a las funciones que el cuerpo realiza, tales como el habla, el equilibrio, la respiración o la movilidad.

La demencia senil es un síndrome neurodegenerativo caracterizado por la pérdida de las capacidades psíquicas, principalmente las cognitivas. Esto produce en las personas que padecen la enfermedad desorientación, ansiedad y alteraciones conductuales que impiden el desarrollo de una vida normal.

Aunque es una enfermedad cuyo mayor factor de riesgo es la edad[5], suele ser una enfermedad derivada de otras enfermedades neurodegenerativas. El Alzheimer es la enfermedad que con mayor frecuencia causa demencia senil, lo que ha propiciado que se utilicen ambos términos de forma equivalente.

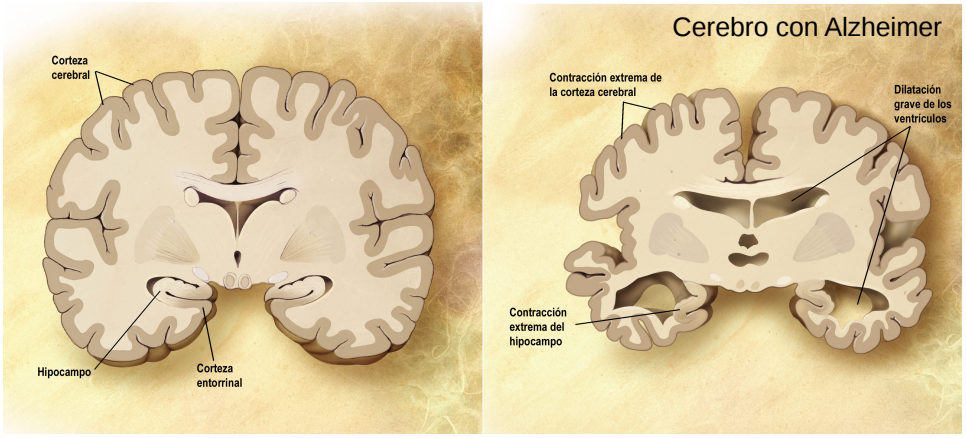

Figura 1.2. Comparativa cerebro sano vs cerebro con Alzheimer.

El Alzheimer es una patología que produce degeneración neuronal. Principalmente se ven afectadas las neuronas altamente especializadas encargadas de realizar las funciones que más nos caracterizan como seres humanos. Afecta a la memoria reciente y retrógrada, junto a otros déficits como alteraciones del pensamiento abstracto, del juicio, de la coordinación, de la planificación y organización, del habla, de la escritura y del cálculo.

Tiene un pico de aparición creciente a partir de los 70 años de edad. Aunque es relativamente rara por debajo de los 65 años, hay casos, excepcionalmente atípicos, descritos con inicio en la juventud[4].

Las personas que sufren de Alzheimer pueden vivir entre 3 y 20 años después del diagnóstico. La rapidez con la cual empeora esta enfermedad es diferente para

[3]Las neuronas son las células esenciales del sistema nervios

[4]Las funciones cognitivas son los procesos mentales que nos permiten llevar a cabo cualquier tarea[7].

[5]Las personas que más se ven afectadas por **la demencia senil** son los ancianos a partir de los 65 años.

cada persona. Si el mal de Alzheimer se presenta rápidamente, es más probable que empeore de la misma manera. La última fase de la enfermedad puede durar desde unos meses hasta varios años. Durante ese tiempo, el paciente se torna totalmente inválido[5].

Las recomendaciones de los expertos, tanto para el Alzheimer como para la demencia senil, se centran fundamentalmente en dos puntos clave: detección precoz de los primeros síntomas, y ejercitar la memoria y la función intelectual. Además, mantener una dieta equilibrada, baja en grasas, protege frente al deterioro cognitivo y la vitamina E ejerce un efecto protector[6].

Aunque por el momento no existe ningún tratamiento que revierta el proceso de degeneración que comportan estas enfermedades. Actualmente, los ejercicios de estimulación cognitiva son el principal tratamiento para prevenir y ralentizar en la medida de lo posible la evolución de las demencias.

1.2. Terapias para enfermedades neurodegenerativas

Como se ha mencionado en la sección anterior, realizar ejercicios de estimulación cognitiva es el mejor tratamiento para prevenir y ralentizar la evolución de las demencias, pero con el aumento de personas mayores que demandan cuidados especiales, aumenta la necesidad de diseñar nuevas terapias y tecnologías aptas para la asistencia de personas mayores en su día a día. Por este motivo, se están realizando estudios sobre cómo mejorar las terapias existentes y crear nuevas. El objetivo de dichas terapias no es únicamente prevenir y ralentizar el deterioro cognitivo, también buscan mejorar la calidad de vida de personas que padecen enfermedades neurodegenerativas.

1.2.1. Terapias de estimulación cognitiva

Las terapias de estimulación cognitiva consisten en realizar ejercicios diseñados para activar las diversas funciones cerebrales. El principal objetivo de dichos ejercicios es mantener y mejorar las capacidades cognitivas el máximo tiempo posible. Mejorando con esto la calidad de vida del paciente, ya que se potencia la autonomía y la autoestima, se mejora la interacción con el entorno, disminuyendo así la confusión y la ansiedad, así como otras reacciones psicológicas adversas[8].

Los ejercicios se centran en estimular las siguientes capacidades cognitivas:

Atención.- Capacidad de generar, dirigir y mantener un estado de activación adecuado para el procesamiento correcto de la información.

Cognición social.- Conjunto de procesos cognitivos y emocionales mediante los cuales interpretamos, analizamos, recordamos y empleamos la información sobre el mundo social. Hace referencia a cómo pensamos

acerca de nosotros mismos, de los demás y su comportamiento y de las relaciones sociales, y cómo damos sentido a toda esa información y emitimos comportamientos en función de ella.

Funciones ejecutivas.- Habilidades implicadas en la generación, la regulación, la ejecución efectiva y el reajuste de conductas dirigidas a objetivos.

Gnosias.- Capacidad de elaborar, interpretar y asignar un significado a la información captada por los sentidos.

Habilidades visoespaciales.- Capacidad para representar, analizar y manipular un objeto mentalmente.

Lenguaje.- Habilidad para elaborar, comunicar y entender ideas mediante sonidos, símbolos y/o sistemas de gestos.

Memoria.- Capacidad para codificar, almacenar y recuperar de manera efectiva información aprendida o de un suceso vivido.

Orientación.- Capacidad que nos permite ser conscientes de la situación en la que estamos en cada momento. Existen tres tipos de orientación, temporal, espacial y personal.

Praxias.- Habilidad para poner en marcha programas motores de manera voluntaria y, normalmente, aprendidos.

Los tratamientos que incorporan actividades lúdicas y sociales son los que mayor aceptación generan entre los pacientes. También son las que mejor estimulan las capacidades psicosociales de los pacientes.

Los ejercicios de estimulación cognitiva son muy beneficiosos en los pacientes en los que se aplican. Por ello es aconsejable que llegados a cierta edad, las personas, aunque no sufran ningún tipo de demencia ni pérdidas de memoria, realicen ejercicios de estimulación cognitiva, ya que con la edad el cerebro también pierde capacidad. No son necesarias terapias específicas, es posible estimular el cerebro mediante actividades que ayudan a mantener nuestro cerebro en forma, por ejemplo, jugar a juegos nuevos, aprender idiomas o tocar algún instrumento, ya que son actividades que ayudan a mantener el cerebro en forma.

1.2.2. Terapias asistidas con animales

Se está empezando a introducir terapias con perros como una herramienta viva, con la finalidad de contribuir a la mejora del funcionamiento físico, social, cognitivo y emocional de pacientes que sufran deterioro cognitivo. Como se puede observar

en la figura 1.3, realizar actividades con perros puede aumentar el contacto social, ya que la presencia de un animal supone un incentivo para el paciente.

Figura 1.3. Terapia asistida con perros.[6]

En la residencia y centro de día Oms-Sant Miquel, perteneciente a la Fundación de Atención a la Dependencia del Gobierno Balear, se realizan dos veces por semana sesiones de terapia con perros. Cada sesión se inicia con una presentación de las perritas a los residentes en donde se les explican las características de los animales con el objetivo de hacer trabajar a su memoria. Luego, se realizan otras actividades que completan la terapia. Así, por ejemplo, se trabaja con la fisioterapeuta la motricidad fina mediante ejercicios consistentes en peinar a las perritas, así como la motricidad gruesa a través del lanzamiento de pelotas a los perros. Otra de las actividades la realiza la psicóloga tratando de estimular cognitivamente a los participantes preguntándoles si han tenido perritos alguna vez, qué color y tamaño tenían, y cómo fue su experiencia con los animales. Si algún paciente no puede asistir a las sesiones grupales, se lleva las mascotas a su habitación y se realizan sesiones individuales[11].

1.2.3. Terapias asistidas con robots

Debido a que, por motivos de higiene y seguridad[7] no se permite la presencia de animales en algunos centros, se están explorando terapias que utilicen robots como sustitutivos de los animales. Los robots están dotados de inteligencia artificial y múltiples sensores que les permite comportarse e interactuar con los usuarios como si de un animal real se tratara. Además, aportan la posibilidad de

[6]Imágenes obtenidas del blog de AFA Parla[10]. Fotográmas del video de la sesión: "Terapia asistida con perros, 1ª sesión: Hoy ponemos guapa a Lua"

[7]Los animales pueden ser fuente de alergias, infecciones, mordiscos y arañazos.

monitorizar la evolución del paciente y de realizar terapia cognitiva.

El grupo de Robótica de la universidad *Rey Juan Carlos I* en colaboración con la *Centro de Alzheimer Fundación Reina Sofía* llevó a cabo el proyecto "ROBOTERAPIA EN DEMENCIA"[12]. El objetivo principal del proyecto era la comparación de una terapia con robots sociales frente a la terapia habitual en relación con la calidad de vida del paciente, su estado cognitivo, sus alteraciones de comportamiento, y la apatía de los pacientes hacia el tratamiento habitual.

La intención del estudio era aportar al terapeuta una herramienta nueva de trabajo, por ello fueron los mismos terapeutas quienes diseñaron las sesiones, siguiendo el guión habitual y adaptándolas a la gravedad de los usuarios.

Para la terapia con robots con aspecto de animales se utilizaron el Robot con forma de foca *Paro*[14] y el robot con forma de perro *Aibo*[15] y para la terapia con robots de aspecto humano se utilizó el robot humanoide *Nao*[16].

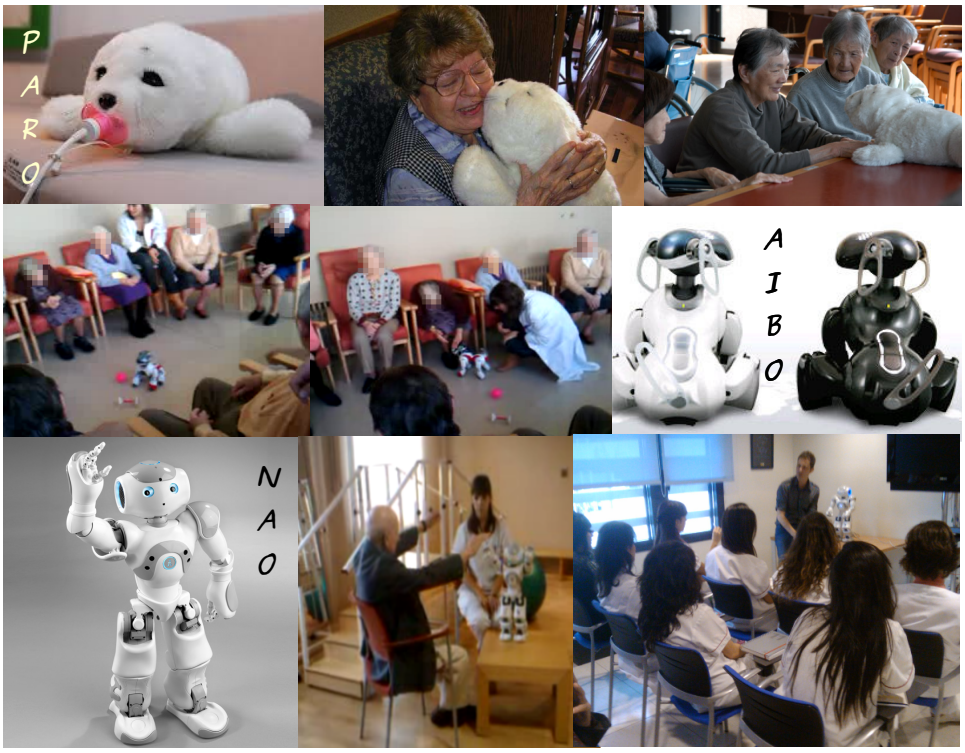

Figura 1.4. Terapia asistida con probots.[8]

Las sesiones incluían actividades de musicoterapia, fisioterapia y de estimulación cognitiva, mediante ejercicios de lógica, repetición de palabras o adivinanzas.

[8]Imágenes obtenidas de la presentación de "Roboterapia en la recuperación de Alzheimer" [13].

Los pacientes con demencia moderada-grave que participaron en este estudio mostraron una leve mejoría en la apatía tras la terapia con robots. Las alucinaciones y la irritabilidad mejoraron ligeramente. Al igual que sucede con las terapias con animales, las terapias asistidas con robots obtienen resultados positivos.

1.2.4. Terapia con música

El área encargada de la memoria musical y el área de nuestra capacidad para sentir emociones son las últimas en verse afectadas por el Alzheimer. Por ello un grupo de jóvenes ha puesto en marcha el proyecto **Música para despertar**[9], que consiste en crear una terapia que utilice la música como factor evocativo de emociones y recuerdos. Se lleva a cabo en el centro de mayores "Cáxar de la Vega", junto a la colaboración de los trabajadores, residentes, familiares y voluntarios.

La música autobiográfica es un concepto que se utiliza para definir aquellas melodías y canciones que nos han acompañado a lo largo de nuestras vidas, las que han estado presentes en los momentos más importantes, aquellas a las que recurrimos cuando estamos tristes o alegres, las que hacen que al volver a escucharlas un millón de emociones y recuerdos nos invadan. Se utiliza como punto de partida del proyecto, ya que con la información, que han proporcionado los familiares de los pacientes, sobre sus gustos musicales y sus canciones evocativas, se crean terapias personalizadas a cada paciente.

El tratamiento está obteniendo resultados sorprendentes. Ha logrado disminuir la ansiedad en los pacientes. En los momentos en los que escuchan canciones que han formado parte de su vida, las sonrisas vuelven a sus rostros y, lejos de hacer que sientan nostalgia, se llenan de vitalidad, algunos incluso cantan y bailan.

Se puede comprobar dichos resultados en su página web[17] y en su canal de *youtube*[18].

1.3. Android en estimulación cognitiva

Las tecnologías de la información y la comunicación, en especial móviles y tabletas, cada vez tienen mayor impacto social. El empleo de dichas tecnologías permite más eficacia y ahorro. También, hace posible mejorar la calidad de vida y la autonomía personal de los colectivos vulnerables.

Android es un sistema operativo, basado en el kernel de Linux, enfocado a los dispositivos móviles, cuenta con un entorno que permite construir aplicaciones y juegos innovadores para dispositivos móviles en lenguaje Java.

[9]El proyecto **Música para despertar** se encuentra entre los ganadores del *Premio Jóvenes Emprendedores Sociales.*

Los dispositivos que utilizan Android tienen una gran cantidad de recursos que se pueden aprovechar con el fin de ayudar a personas con necesidades especiales.

Una de sus mayores ventajas es la interfaz de usuario simple e intuitiva. Permite a los desarrolladores crear todo tipo de aplicaciones cuyos diseños se adapten a diversos gustos y necesidades.

Otra ventaja es el uso de sensores y *gadgets*. Por ejemplo con un móvil con GPS, se podría saber dónde se encuentra una persona mayor que está desorientada. O podríamos vincular un pulsómetro al móvil y hacer que este nos envíe un aviso si se nota que algo fuera de lo normal está ocurriendo.

A continuación describiremos algunas aplicaciones creadas con el fin de ayudar a personas mayores.

Medisafe Es una aplicación creada para recordar la hora de la medicación. Existen personas, por ejemplo las personas mayores o enfermos crónicos, que tienen necesidades especiales a la hora de tomar medicamentos. Esta aplicación da soporte a dichas personas, a los familiares y cuidadores, recordándoles que es la hora de cierta medicación y qué dosis debe tomar de la misma[21].

Mimov Es una aplicación que sirve para localizar a personas. El fin de la aplicación es poder localizar a personas que tengan cierto grado de dependencia. Por ejemplo, las personas mayores, aunque no tengan ninguna demencia, en ocasiones se sienten desorientados, lo que puede provocar que se retrasen, lo que alertaría a la familia[22].

App de Dependencia Es un aplicación desarrollada por Telefónica para el Instituto de Mayores y Servicios Sociales (Imserso) con el fin de informar, apoyar y dar autonomía a los cuidadores y a las personas mayores o dependientes[23].

En la aplicación se muestran consejos sobre cómo dar correctamente cuidados a una persona dependiente, consejos respecto a las adaptaciones que se deben hacer en el hogar y consejos para el cuidado de los cuidadores, tanto físico como emocional.

También se da información sobre los servicios y prestaciones del Sistema de Atención a la Dependencia previstos en la Ley (teleasistencia, ayuda domiciliaria, centros de día, residencias), e información y enlaces para tramitación de solicitudes por los órganos gestores de las Comunidades Autónomas.

Existen varias aplicaciones, que aunque no estén diseñadas específicamente con el fin de ayudar a personas mayores o dependientes, sirven para estimular

cognitivamente y entrenar el cerebro.

Juegos mentales Aplicación que contiene hasta 6 juegos que permiten trabajar varias habilidades mentales: memoria, razonamiento, concentración, inteligencia espacial y observación[24].

4 fotos 1 palabra Juego que ejercita las habilidades gnósicas y de cognición social, mediante un juego que consiste en ver 4 imágenes y ver lo que tienen todas ellas en común[25].

Memory Típico juego de meroria en el que tenemos que revelar las parejas[26].

Palabro Juego que ejercita las capacidades de atención y lenguaje. Consiste en crear palabras a partir de letras que se muestran en el tablero, a mayor número de palabras y de longitud de letras por palabra, mayor puntuación[27].

Clasic words Es una versión del *Scrabble*. Al igual que el juego anterior ejercita las capacidades de atención y lenguaje[28].

Brain Trainer Special Esta aplicación contiene hasta 39 juegos, de los cuales 16 sólo están disponibles en la versión Pro o de pago. En general puede ser útil para trabajar diferentes capacidades mentales como la memoria, la observación, el razonamiento, etc[29].

Como se ha mencionado al principio del capítulo, en el presente trabajo de fin de grado se desarrollará una aplicación Android de estimulación cognitiva. Para ello, se ha contado con la colaboración de terapeutas especializadas en el Alzheimer, lo que confiere a la aplicación una serie de ejercicios diseñados especialmente con el fin de la estimulación cognitiva.

Desarrollar aplicaciones de estimulación cognitvia en Android supone un ahorro frente a los robots, y ofrece versatibilidad a la hora de diseñar e implementar los ejercicios.

En los capítulos siguientes se profundizará más en el desarrollo de la aplicación. En el capítulo 2, se contarán los objetivos principales del TFG, también se hablará sobre la metodología utilizada durante el desarrollo de la aplicación. En el capítulo 3, se describirá el software utilizado, dando una breve introducción y contando para qué las hemos utilizado. En el capítulo 4, se profundizará en el desarrollo de la aplicación. Se explicarán todos los pasos que se han seguido para llegar a la aplicación final, desde el análisis de los requisitos, hasta la codificación de los mismos. En el capítulo 5 se presentarán los objetivos logrados y el tiempo empleado para realizarlos. Finalmente en el apéndice A se incluye un manual de usuario de la aplicación.

Capítulo 2
Objetivos y metodología

2.1. Objetivos

El objetivo principal del presente trabajo de fin de grado es utilizar las nuevas tecnologías para ayudar a personas mayores, en especial a las que sufren alguna enfermedad neurodegenerativa. Para ello se diseñará y programará una aplicación Android cuya finalidad es ser una herramienta práctica que sirva de apoyo en terapias dirigidas a personas que sufran alguna demencia.

La aplicación contará con un conjunto de ejercicios concretos de estimulación cognitiva, que se presentarán al paciente en forma de juegos.

Este objetivo general se ha dividido en tres subobjetivos:

- El primer subobjetivo es **capturar los requisitos** que esta aplicación debe tener para ser útil. Para ello se cuenta con la colaboración de tres terapeutas del *Centro de Alzheimer Fundación Reina Sofía* que nos proporcionaron los requisitos y la información necesaria para el desarrollo de la aplicación. Los requisitos se explicarán de forma detallada en el capítulo 4.

- El segundo subobjetivo es **diseñar y programar** la aplicación para que cumpla con los requisitos dados. Para desarrollar la aplicación tendremos que conocer las capacidades *software* básicas de las que disponemos. Primero se diseñará la estructura funcional, luego se programarán los juegos y el resto de la aplicación. Al ser una aplicación Android támbien se debe realizar un diseño atractivo para la interfaz de usuario.

- El tercer subobjetivo es la **incorporación de contenido especializado** que ha sido recomendado por los terapeutas. Esto es fundamental, ya que contar con el contenido apropiado logrará una mejor estimulación cognitiva y hará que los pacientes se interesen por la aplicación.

2.2. Metodología

Existen varias metodologías de ingeniería de software que se pueden utilizar para el desarrollo de aplicaciones. Para el desarrollo de la aplicación que nos compete en el presente Trabajo de Fin de Grado utilizaremos el modelo en espiral[19].

El modelo en espiral consiste en una serie de ciclos que se repiten en forma de bucle, cada ciclo representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que se eligen en función de las realizadas previamente. Estos ciclos se irán ejecutando hasta que la aplicación sea aceptada y no requiera otro ciclo.

Los ciclos se dividen en cuatro actividades:

1. Determinar o fijar objetivos.
2. Análisis del riesgo.
3. Desarrollar, verificar y validar (probar).
4. Planificar.

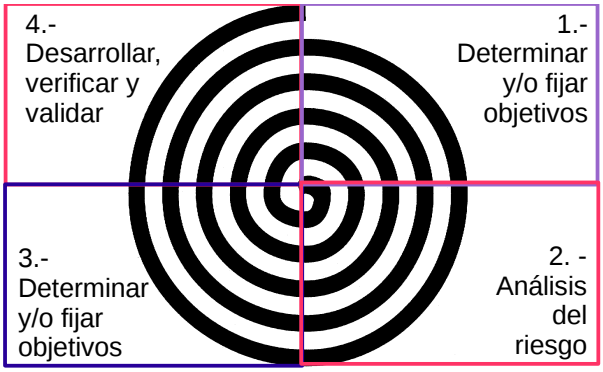

Figura 2.1. Esquema visual del modelo en espiral.

Se ha trabajado realizando reuniones periódicas con el tutor. En cada reunión primero se revisaban y validaban las tareas fijadas en la reunión anterior. Se resolvían las dudas que habían surgido. Finalmente se fijaban los objetivos para la siguiente reunión.

También se han realizado reuniones con las terapeutas en las que se mostraba la versión que hubiese de la aplicación y se fijaban los cambios necesarios.

Se ha recogido el progreso de la aplicación en la página de MediaWiki del proyecto `http://jderobot.org/Evelinfv`.

2.3. Plan de trabajo

El plan de trabajo elaborado es:

1. Familiarización con el entorno de desarrollo *Android Studio*.

2. Dasarrollo de pruebas de funcionalidad básica para el desarrollo de la aplicación. Por ejemplo, reproducir videos, reproducir sonidos, crear y utilizar bases de datos, reproducir sonidos a partir de texto, arrastrar y soltar elementos por la pantalla, comprobar el pulsado en la pantalla, etc.

3. Diseño de la apariencia final de la aplicación y de los ejercicios concretos de estimulación cognitiva.

4. Desarrollo de los juegos concretos de la aplicación.

5. Incorporación de datos persistentes para la configuracion de usuarios y juegos.

6. Búsqueda de contenido especializado para cada ejercicio de la aplicación.

Capítulo 3
Infraestructura

En este capítulo se describirán las herramientas software que se han utilizado para el desarrollo de la aplicación. Se indicará la versión utilizada de cada herramienta y se contará brevemente el motivo por el que han sido necesarias.

3.1. Android

Andrid es un sistema operativo que fue creado por Android Inc. en el año 2003, con el objetivo de desarrollar un sistema operativo de código abierto, basado en Linux y que ofreciera una interfaz de usuario innovadora.

En el año 2005 Android Inc. es adquirida por Google que continua con el desarrollo del sistema operativo, centrándose principalmente en dotar al sistema operativo de una interfaz de usuario sencilla y accesible.

En el año 2007 Google lanza la primera versión oficial de Android llamada Android 1.0 Apple Pie. La última versión estable es Android 5.2 Lollipop, aunque la última versión oficial presentada por la compañía es Android M.

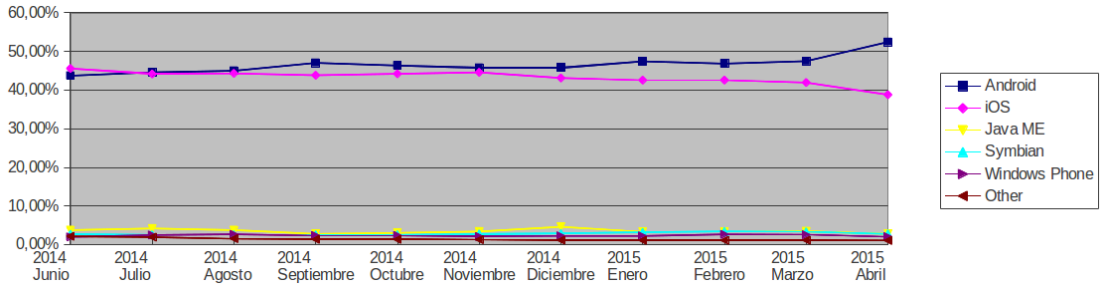

Figura 3.1. Porcentaje de uso de sistemas operativos móviles Junio 2014 - Abril 2015.[1]

Como se puede observar en la figura 3.1, actualmente Android es el sistema operativo más utilizado en dispositivos móviles como *smartphones*, *tablets*,

[1]Imagen obtenida a partir de los datos estadísticos de uso de los sistemas operativos móviles de la pagina web de Netmarketshare.[31]

smarttvs y *smartwears*.

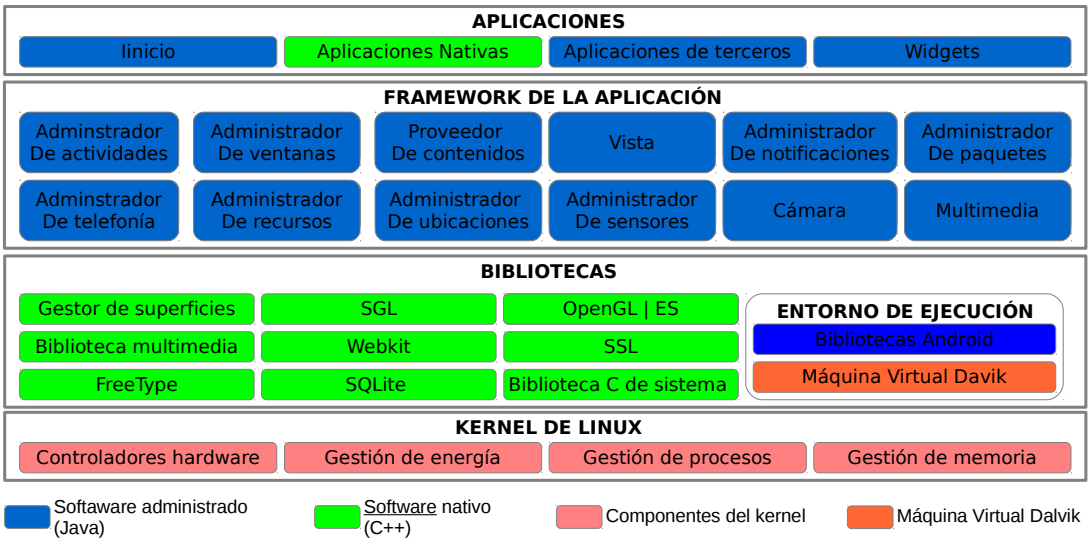

Figura 3.2. Diagrama de la arquitectura de Android.[2]

Como se puede observar en la figura3.2, Android está basado en el concepto de máquina virtual utilizado en Java, aunque debido a las limitaciones de los dispositivos donde ha de correr, se creó una nueva máquina virtual, la máquina virtual Dalvik. Todas las aplicaciones deben correr en dicha máquina virtual.

Incluye un conjunto de librerías nativas en C/C++ que están compiladas en código nativo del procesador. También cuenta con un entorno que permite el desarrollo de aplicaciones utilizando el lenguaje de programación Java. Proporciona las herramientas necesarias para desarrollar aplicaciones sin tener en cuenta los dispositivos en los que se va a ejecutar.

3.1.1. Android Studio

Android Studio[32] es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android. Está basado en *IntelliJ IDEA*[3] que ofrece:

- Un sistema flexible de construcción de proyectos basado en Gradle.
- La generación múltiples ficheros .apk. Esto significa que dependiendo de ciertas características se crearán los ficheros correspondientes. Por ejemplo si hemos puesto dos idiomas para nuestra aplicación se generarán 2 ficheros .apk, uno por cada idioma.
- Plantillas de código.

[2]Imagen obtenida de la página web para desarrolladores de android[32]

[3]**IntelliJ IDEA** es un entorno de desarrollo integrado (IDE) de Java para el desarrollo de programas informáticos.[34]

- Un editor de diseño para la interfaz gráfica de la aplicación.
- Herramientas de depuración y optimización de código.
- Facilidad para integrar nuestras aplicaciones con *Google Cloud Messaging* y *App Engine*.

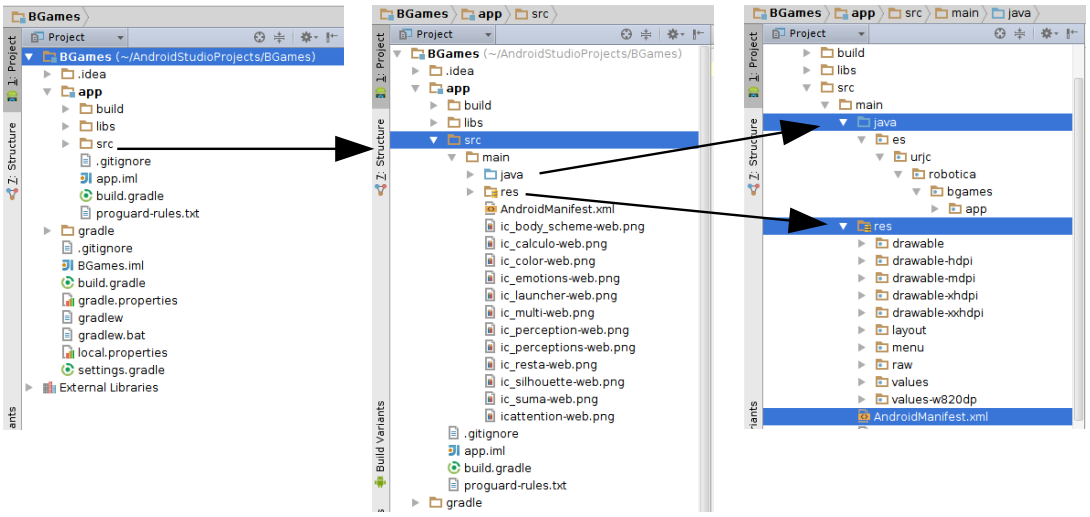

Figura 3.3. Estructura del proyecto Android.

En la figura 3.3 se muestra la estructura del proyecto Android de la aplicación.

3.1.2. una aplicación en Android

Es necesario definir los conceptos y elementos más importantes en la programación de aplicaciones Android para explicar en detalle la codificación de la aplicación en la sección 4.3.

Cuando se programa en Android se deben tener en mente cuatro elementos fundamentales, el fichero `AndroidManifest.xml`, el directorio `java`, el diretorio `res` y la clase `Activity`.

AndroidManifest

El fichero `AndroidManifest.xml` que se encuentra dentro del directorio raiz de la estructura de un proyecto Android, es el fichero de manifiesto donde se presenta la información esencial acerca de la aplicación para el sistema Android. Es la información que el sistema debe conocer antes de que pueda ejecutar cualquiera de código de la aplicación.

Directorio java

Es el directorio raiz donde se definirán todas las clases `Java` que necesitemos para realizar el código funcional de la aplicación que vamos a desarrollar.

Directorio res

Es el directorio raiz donde almacenaremos el contenido de la interfaz de usuario. El directorio se subdivide de la siguiente forma:

drawable.- Directorio en el que se almacenan las imágenes que utilizaremos en la aplicación.

raw.- Directorio en el que se almacenan los videos y sonidos de la aplicación.

menu.- Directorio en el que se almacenan los ficheros XML que representan las definiciones de los menus contextuales que utiliza la aplicación.

values.- Directorio en el que se almacenan las definiciones de constantes de la aplicación. Por ejemplo en el fichero `strings.xml` se definen los textos de la aplicación, si deseamos que la aplicación tenga dos idiomas tendremos que crear dos ficheros `strings.xml`, uno por cada idioma, donde tendremos las mismas definiciones pero con valores distintos, cada fichero con el idioma correspondiente.

layout.- Directorio donde almacenamos los ficheros XML donde se definen los diseños visuales de cada pantalla de la aplicación. Cada fichero de diseño de una pantalla se llama *Layout*. En un layout se declaran todos los elementos que se deseen mostrar en la pantalla, mediante una jerarquía de *View* y objetos *ViewGroup*.

A todos los elementos que se encuentran dentro del directorio se les asigna un identificador. Dicho identificador será accesible desde la clase autogenerada por Android Studio `app.R`.

Activity

Una *activity* es una clase java que hereda de la clase `Activity`. Se puede considerar que las *activities* se corresponden con las pantallas de la aplicación. Se dividen en parte lógica y parte gráfica.

La parte gráfica se define un fichero XML que como se ha mencionado en el apartado anterior, se almacena en el directorio layout. Por ejemplo para una pantalla de inicio de sesión, en el fichero XML declareremos dos `TextView`, una para el nombre de usuario y otra para la contraseña y un `Button`.

La parte lógica es una fichero .java en el que se declará la clase encargada de la lógica de esa *activity* dentro de la aplicación. Para seguir con el ejemplo de la pantalla de inicio de sesión, en el fichero .java definiremos los métodos encargados de comprobar si ha introducido correctamente el usuario y la contraseña.

Lo más importante es entender el ciclo de vida de una *activity*, que es el que se muestra en la figura 3.4.

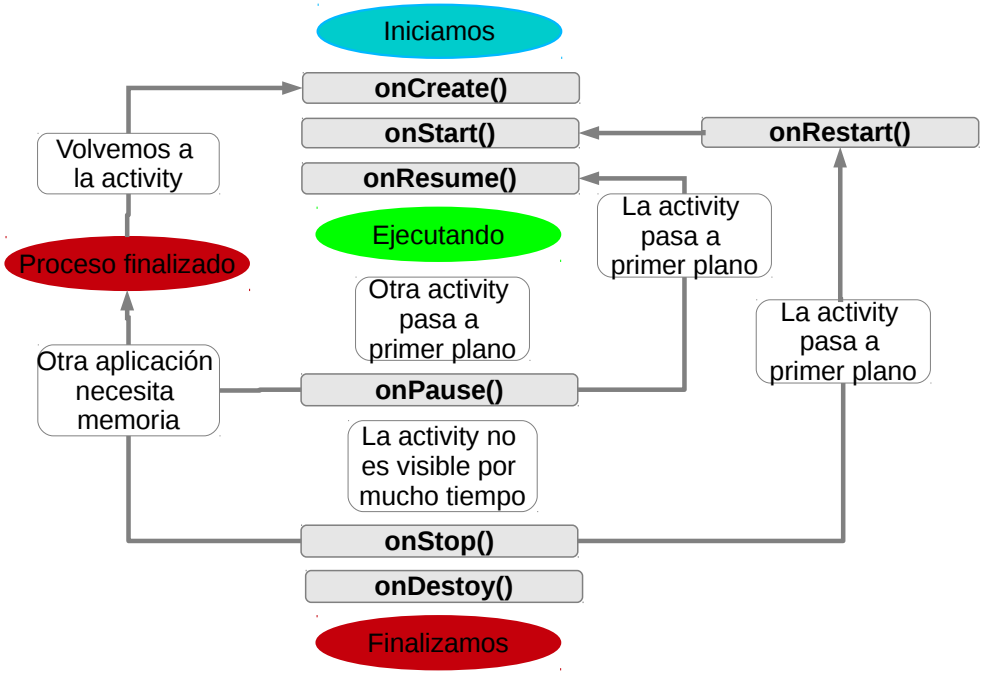

Figura 3.4. Ciclo de vida de una *activity*.[4]

Es importante comprender este concepto, debido a que los datos de una *activity* únicamente son accesibles mientras la *activity* esté ejecutándose. Por ejemplo si estamos en una pantalla y pasamos a la siguiente, los datos de la primera pantalla no se almacenarán y al regresar a la misma será como si empezáramos desde el principio.

```
public class Activity extends ApplicationContext {
  protected void onCreate(Bundle savedInstanceState);

  protected void onStart();

  protected void onRestart();

  protected void onResume();

  protected void onPause();

  protected void onStop();

  protected void onDestroy();
 }
```

Cuadro 3.1. Clase Activity

[4]Imagen obtenida de la página web para desarrolladores de android[33]

En el cuadro 3.1 se muestran los métodos que se deben sobreescribir cuando heredamos de la clase `Activity`.

Para el desarrollo de la aplicación se ha utilizado la versión 0.5.2 de Android Studio. Se eligió este entorno de desarrollo debido a que es la herramienta oficial de la que disponen los desarrolladores de aplicaciones Android.

3.2. Java

Java es un lenguaje de programación de alto nivel, orientado a objetos, robusto y fuertemente tipado. Diseñado para ofrecer a los desarrolladores herramientas para programar sin preocuparse por la plataforma. Esto se logra gracias a que una vez el código se compila, se crea un ejecutable que únicamente puede ser interpretado por la *Máquina Virtual de Java (JVM)*. Basta con contar con una *Máquina Virtual de Java* dentro de la plataforma concreta donde se desee ejecutar la aplicación.

Java se desarrolló con el objetivo de crear una nueva tecnología para programar la siguiente generación de dispositivos inteligentes. Tiene muchos elementos similares con los lenguajes C, C++ y Objetive C. Fue comercializada por primera vez en 1950 por Sun Microsystems.[35]

Para el desarrollo de la aplicación hemos utilizado el JRE 1.8.0_05-b13 amd64 de Java.

3.3. SQLite

SQLite[36] es una biblioteca de software libre que implementa un sistema de gestión de bases de datos SQL transaccional. A diferencia de otros sistemas de gestión, no necesita de una configuración ni de un servidor. Se implementa como una parte integral de la aplicación en la que se vaya a utilizar.

Ha sido necesario utilizar SQLite debido a que la aplicación debe almacenar datos persistentes de los usuarios, por ejemplo los parámetros de configuración y los datos estadístios de los juegos. Android cuenta con una biblioteca que proporciona todas las herramientas para la creación y gestión de bases de datos SQLite.

3.4. PhotoShop

Adobe Photoshop es un editor de gráficos rasterizados que se utiliza principalmente para el retoque de fotografías y gráficos. Es la principal aplicacion de edición de imágenes. Actualmente forma parte de la familia Adobe Creative Suite y es desarrollado y comercializado por Adobe Systems Incorporated.

Se ha utilizado Photoshop para editar las imágenes que forman parte del contenido de los juegos. Támbien se ha utilizado para crear el diseño de la interfaz de usuario de la aplicación. La versión empleada ha sido Adobe Photoshop CS6 versión 13.0

3.5. Edición de sonido

Para dotar de contenido a la aplicación hemos utilizado una herramienta online llamada **TwistedWave**[37] para editar las pistas de sonido que hemos utilizado en la aplicación.

Capítulo 4
Desarrollo

Como se ha mencionado anteriormente, en el presente Trabajo de fin de grado se cuenta el desarrollo de una aplicación Android, cuyo principal objetivo es servir de herramienta de apoyo en terapias de estimulación cognitiva. En este capítulo se explicará en detalle el desarrollo *software* de la aplicación.

El desarrollo de una aplicación consta principalmente de cinco partes, análisis, diseño, codificación, integración y mantenimiento. Este capítulo se va a dividir en 3 secciones, en la primera sección se contarán los requistos de la aplicación, en la segunda el diseño estructural y en la tercera el desarrollo informático de la aplicación.

4.1. Análisis - Captura de requisitos

Para poder diseñar una aplicación que cumpliese con el objetivo de servir como herramienta de apoyo a terapeutas, los requisitos nos fueron proporcionados por tres terapeutas del *Centro de Alzheimer Fundación Reina Sofía*.

Partiendo de unos requisitos base, se realizaron reuniones en las que se presentaban los avances en la aplicación y se acordaban una serie de cambios para mejorla.

De todas esas reuniones y la información recabada, se extrajeron los siguientes requisitos fundamentales que debe cumplir la aplicación.

- **Diferenciar usuarios**
  En la aplicación se debe poder diferenciar a cada usuario. Esto es necesario para realizar un seguimiento individualizado de la evolución de cada paciente y poder ajustar cada uno de los ejercicios a sus necesidades específicas.

- **Estímulos**
  Al finalizar cada ejercicio de la aplicación, se debe presentar al paciente un

refuerzo positivo, tanto a nivel visual como auditivo. Esto es importante ya que facilita la adherencia a la herramienta.

- **Parámetros de configuración**
  Cada ejercicio de la aplicación debe contar con ciertos parámetros de configuración que servirán para modificar los niveles de dificultad en función del paciente.

  — Número de niveles de un juego.

  — Tiempo de exposición del estímulo correcto. Si un ejercicio presenta un estímulo que el usuario debe buscar posteriormente.

  — Tiempo de exposición de los estímulos distractores. Por ejemplo en el juego *Simón* el sonido de los botones debe sonar cierto tiempo.

  — Número de elementos distractores, en la mayoría de los juegos se puede variar el número de elemento que se muestra.

  — Número de elementos correctos, la aplicación contará con más de un juego en los que existe más de un elemento correcto, por ejemplo en los juegos de esquema corporal se podrá buscar de uno a cuatro elementos correctos.

- **Datos estadísticos**
  Cada ejercicio debe almacenar información relevante para los terapeutas, dicha información servirá para llevar un control estadístico detallado de cada paciente y así realizar un sguimiento de su evolución. Los datos estadísticos comunes que se deben almacenar son los siguientes:

  — Fecha en la que se está realizando el juego.

  — Tiempo total jugado.

  — Número de aciertos.

  — Número de fallos.

  Cada juego debe almacenar también:

  — Datos estadísticos específicos de cada juego.

  — Los parámetros relevantes que estén configurados en el momento en el que se está realizando el juego. Esto es necesario para que el control del paciente sea más exacto.

- **Ejercicios**
  Debe contar con una serie de ejercicios diseñados para estimular las capacidades cognitivas de pacientes con Alzheimer. Los ejercicios se describirán con más detalle en el apartado 4.1.1.

- **Contenido**
  Debe contar con contenido adecuado y atractivo visualmente. Es fundamental manejar contenido adecuado para la estimulación cognitva. Para dotar de contenido a la aplicación debemos seguir los siguientes requisitos.

  — Las imágenes de la aplicación deben ser de elementos reales.

  — Los sonidos de la aplicación deben ser de animales reales.

4.1.1. Ejercicios de estimulación cognitiva

Los ejercicios de la aplicación se deben dividir por categorías. Cada categoría tiene como fin estimular una capacidad cognitiva concreta. Las categorías y ejercicios que debe mostrar la aplicación se describen a continuación.

Atención

Esta categoría se corresponde con la capacidad de generar, dirigir y mantener un estado de activación adecuado para el procesamiento correcto de la información. Se inluirán los siguiente ejercicios que se centrarán en estimular las capacidades de atención focalizada, selectiva y sostenida.

Simón.- En este juego se muestrán una serie de botones de colores distintos. Primero se pulsa un patrón aleatorio de botones. El jugador debe repetir el patrón sin equivocarse. Si el jugador realiza bien el ejercicio, el patrón se volverá a repetir, pero esta vez se pulsará un botón más. En el caso contrarío se empezará con un patrón nuevo.

Atención.- Ejercicio en el que presenta un estímulo (*target*), para la aplicación el estímulo será una letra, que el participante tiene que localizar siempre que aparezca en la pantalla. Este ejercicio se presentará de tres formas.

- **Nivel 1.-** El estímulo se presenta junto a los distractores.
- **Nivel 2.-** El estímulo se presenta previamente y los distractores se muestran juntos.
- **Nivel 3.-** El estímulo se presenta previamente y los distractores se van mostrando secuencialmente.

Reconocer emociones

Esta categoría se corresponde con la capacidad de *Cognición social*, que es la encargada de los procesos cognitivos y emocionales mediante los cuales interpretamos la información sobre el mundo social. Para estimular dicha capacidad se añadirá un ejercicio en el que se mostrarán diferentes imágenes de rostros que representan distintas emociones y el participante debe identificarlas.

Gnosias

Esta categoría se corresponde con la capacidad *Gnosica*, que es la encargada de elaborar, interpretar y asignar un significado a la información captada por los sentidos. Para estimular dicha capacidad se añadirán los siguientes ejercicios.

Sonidos.- El participante escuchará un sonido de animal y verá imágenes de animales. Tendrá que identificar qué animal se corresponde con el sonido.

Siluetas.- Se presentará una silueta y una serie de imágenes, el participante debe identificar la imagen que se corresponda con la silueta.

Colores.- Se le pedirá al participante que señale un color.

Esquema corporal

Esta categoría también se corresponde con la capacidad *Gnosica*, aunque por la forma en la que se presentan los juegos se prefiere separar las categorías. Para estimular dicha capacidad se añadirán dos siguientes ejercicios. Uno en el que se deben señalar partes del cuerpo y otro en el que se deben señalar partes de la cara.

Funciones ejecutivas

Esta categoría se corresponde con la capacidad encargada de las habilidades implicadas en la generación, la regulación, la ejecución efectiva y el reajuste de conductas dirigidas a objetivos. Para estimular dicha capacidad se añadirán los siguientes ejercicios.

Secuenciación de actividades.- Aparecerán dos o tres imágenes de una acción y el participante tendrá que tocar las imágenes en el orden que deberían ocurrir.

Categorización.- Se preguntará por una categoría y el participante debe señalar los objetos que pertenecen a dicha categoría. Por ejemplo se pedirá que encuentre todos los animales.

Razonamiento

Esta categoría también se corresponde con la capacidad de *Funciones ejecutivas*. Al igual que pasa con las categorías que se corresponden con la capacidad *Gnosica*, se ha preferido separar las categorías, ya que los ejercicios que se presentan en cada categoría pretenden estimular habilidades distintas.

Cálculo.- Se presentará una operación matemática (suma, resta o multiplicación) y varios resultados.

Comparar valores.- Se presentarán dos números y se preguntará *¿Qué número es mayor/menor?*.

Comparar.- Se presentarán las imágenes de dos objetos y se preguntará *¿Qué objeto cuesta más/menos?*.

Puzzles

Esta categoría se corresponde con las *Habilidades visoespaciales*, que son las encargadas de representar, analizar y manipular un objeto mentalmente. Para estimular dicha capacidad se añadirán los siguientes ejercicios.

Puzzle.- Se presentará un puzzle de piezas cuadradas, las piezas únicamente se pueden mover horizontal o verticalmente hacia el espacio vacío. Cuando ha colacado todas las piezas en la posición correcta se gana el juego.

Colorea.- Se muestra en un lado de la pantalla una imagen compuesta por piezas en blanco y negro. En el otro lado de la pantalla se muestran varias piezas de color desordenadas. El usuario debe colocar cada pieza de color en la posición correcta dentro de la imagen en blanco y negro.

4.2. Diseño

En esta sección se describirá el diseño de la aplicación de estimulación cognitiva que se ha desarrollado. Se describirá el diagrama de flujo de utilización de la aplicación, así como el esquema de la base de datos necesaria para almacenar la información de los usuarios dentro de la misma.

4.2.1. Diagrama de flujo

El diagrama de flujo que se ha diseñado para la aplicación de estimulación cognitiva es el que se muestra en la figura 4.1.

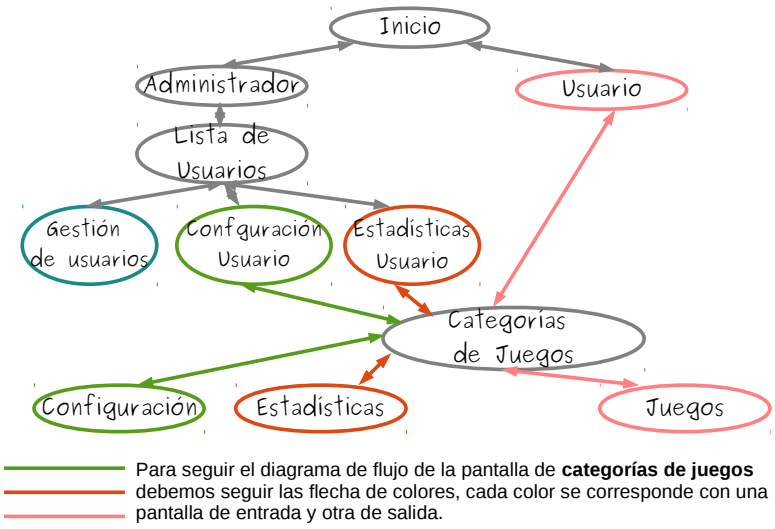

Figura 4.1. Diagrama de flujo de la apliacaión.

En la pantalla inicial se seleccionará si se accede a la aplicación como usuario (paciente) o como administrador (terapeuta). Como se puede observar en la figura 4.2 en ambos casos nos lleva a una pantalla con la lista de usuarios.

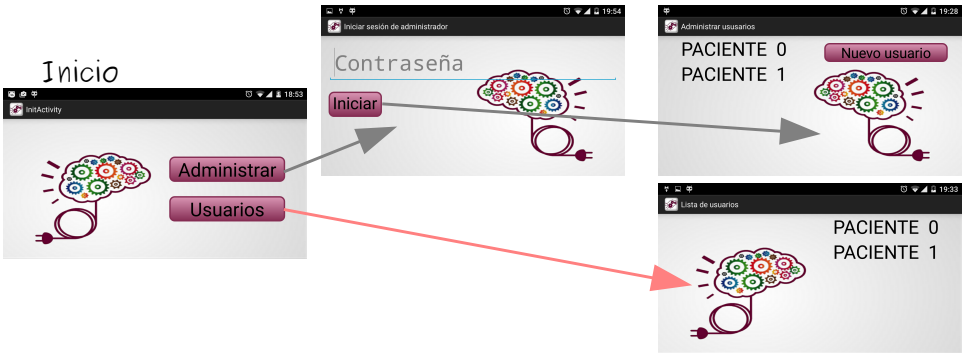

Figura 4.2. Diagrama de flujo desde la pantalla inicial.

Únicamente se podrá acceder como administrador proporcionando una contraseña a la aplicación. Desde el área de administración es posible crear y eliminar usuarios, además, por cada usuario será posible editar los parámetros de configuración y ver las estadísticas de cada juego. En la figura 4.3 se muestra el diagrama de flujo si se accede como administrador.

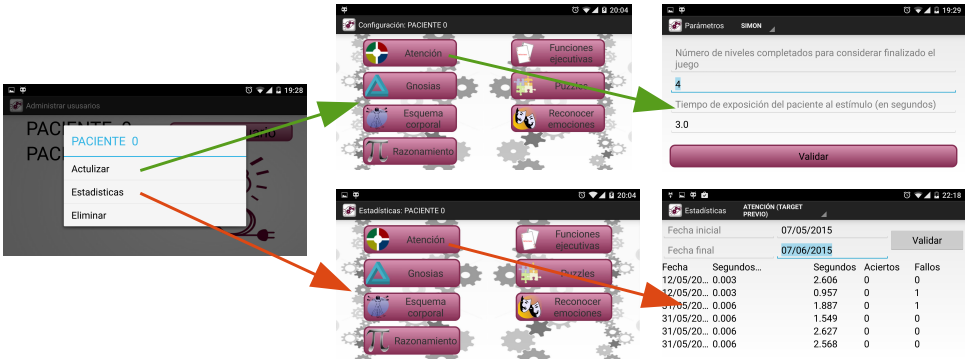

Figura 4.3. Diagrama de flujo desde la pantalla de administrador.

Si se accede como usuario, se elegirá entre la lista de todos los usuarios al usuario correspondiente. Una vez elegido el usuario, será posible acceder a las categorías y desde ahí a los juegos de la categoría seleccionada. En la figura 4.4 se muestra el diagrama de flujo si se accede como usuario.

Figura 4.4. Diagrama de flujo desde la pantalla de usuario.

4.2.2. Esquema de base de datos

Los requisitos de la aplicación nos indican que se debe almacenar a más de un usuario. Cada usuario debe tener vinculados ciertos parámetros de configuración y debe almacenar datos estadísticos. Para almacenar todo esto utilizaremos una base de datos. Para ello lo primero que debemos hacer es diseñar el esquema de base de datos que deseamos utilizar.

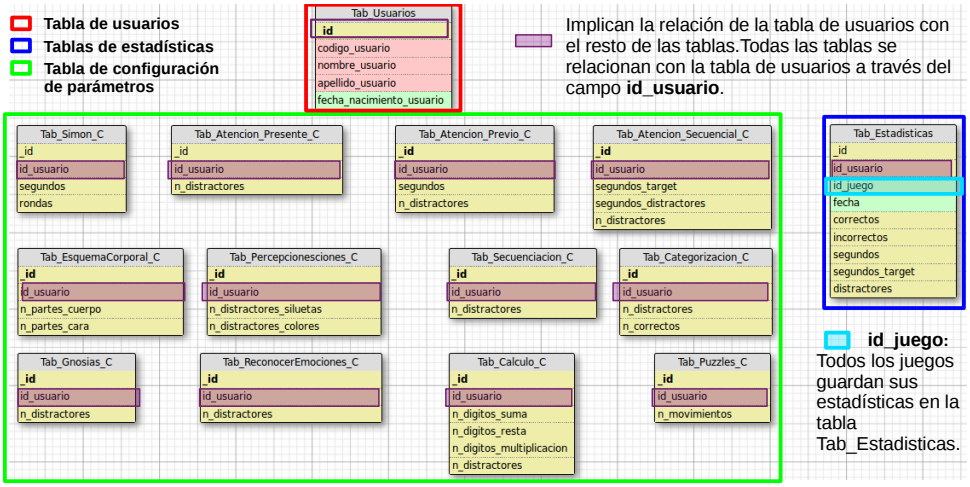

Figura 4.5. Esquema de base de datos de la aplicación.

El esquema de base de datos diseñado para la aplicación es el que se muestra en la figura 4.5. Como se puede observar en el esquema, las tablas se dividen en tres grupos:

Usuarios.- La tabla `Tab_Usuarios` es la tabla donde se almacena la información de los usuarios.

Configuraciones.- Los juegos que sean susceptibles de ser configurados, para cada usuario, tienen asociados una tabla de configuración en la que almacenaremos los parámetros de configuración. Se relacionan con la tabla de usuario mediante el campo `id_usuario`.

Estadísticas.- Todos los juegos deben almacenar datos relevantes del usuario cuando está jugando. Los datos que devuelven todos los juegos son los mismos, salvo alguna excepción, por ello se ha diseñado una sola tabla de estadísticas llamada `Tab_Estadisticas`. Dicha tabla se relaciona con la tabla de usuarios del mismo modo que lo hacen las tablas de configuraciones. Para distinguir de qué juego es cada estadística, se ha incluido en la tabla el campo `id_juego` que se relaciona con la clave numérica que el sistema tiene almacenado para cada juego.

4.2.3. Contenido

Una parte muy importante del desarrollo de la aplicación es encontrar el contenido adecuado. Debe ser contenido apropiado para la estimulación cognitiva, que se ajuste al diseño visual de la aplicación y deben ser elementos que no tengan derechos de autor y puedan ser utilizados.

Debido a que en un juego se puede mostrar un número indeterminado de elementos y se deben respetar las proporciones de los mismos, las imágenes que se han utilizado para cada juego han seguido el mismo patrón de tamaño y diseño. Esto quiere decir, por ejemplo, que todas las imágenes de animales del juego "Sonidos" tienen las mismas dimensiones y se presentan de la misma forma.

Las imágenes se han editado con Photoshop y se han guardado en formato PNG. Cada juego requería un formato o tamaño específico que se describe a continuación.

- Las piezas del juego "Colorea" han sido recortadas con formas no definidas desde una imagen total.

- Las piezas del juego "Puzzle" han sido obtenidas dividiendo en nueve la imagen total.

- Las imágenes del juego "Colores" han sido creadas. Todas tienen la misma forma y se les ha dado relieve, guardando una por cada color que mostramos.

- Las imágenes del juego "Siluetas" se dividen en dos. Tenemos las imágenes completas y las siluetas que han sido creadas a partir de la imagen principal. Ambas imágenes deben tener el miso tamaño entre sí y con el resto de imágenes pertenecientes al juego.

- Para las imágenes del juego "reconcer emociones" se han buscado imágenes de una misma persona representando diversas emociones para mantener coherencia entre las imágenes que se presentan.

Los sonidos del juego "Sonidos" son de animales reales.

4.3. Desarrollo de ejercicios cognitivos

En esta sección se contará la codificación detallada de la aplicación. En especial los ejercicios de estimulación cognitiva, por ello los describiremos en primer lugar. Como ya se ha comentado anteriormente, los ejercicios se presentan en la aplicación a modo de juegos que están divididos en siete categorías.

4.3.1. Atención

En esta categoría, como su propio nombre indica, se estimularán la capacidades de atención focalizada, sostenida y selectiva con dos ejercicios que se dividirán en cuatro juegos.

Simón

En este juego se muestran cuatro botones dispuestos en la pantalla de forma circular, como se puede observar en la figura 4.6. Cada botón tiene asociado un color y un sonido. Cuando un botón es presionado debe brillar y sonar.

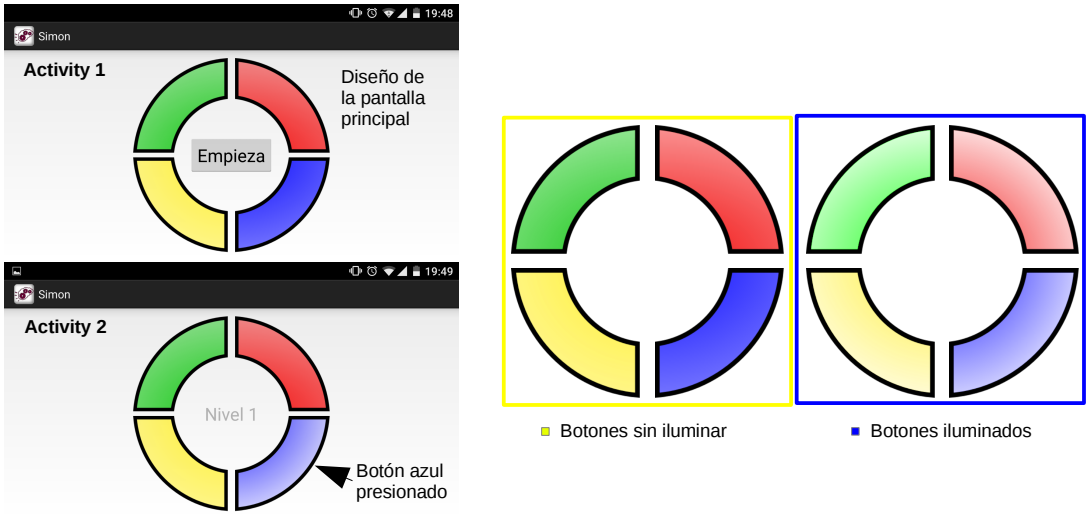

Figura 4.6. Diseño y elementos del juego Simón.

Este juego se divide en rondas. En cada ronda se simula el efecto de presionar una secuencia predefinida de botones. Se empieza la secuencia con un botón, cada vez que se supera una ronda satisfactoriamente se va sumando otro botón, lo que implica que si la primera vez fue el botón verde, la segunda debería ser verde más otro botón. Para superar una ronda el usuario deber repetir la secuencia correctamente.

Para este juego se han definido dos *activities*. Una para la pantalla del juego en el que se ejecuta la secuencia predefinida y la otra para la pantalla en la que el usuario debe repetir la secuencia.

Aunque se han definido dos *activities*, únicamente se ha definido un *layout*. Esto se debe a que visualmente los elementos son los mismos, cuatro botones de colores y un botón central.

En el *layout* el botón central será un elemento de tipo `Button` y cada botón de color como un elemento de tipo `ImageButton`. A cada `ImageButton` se le asignarán dos imágenes, una para el botón en estado no presionado y otra, de un tono

más claro para dar la sensación de el botón se ilumina, para el botón en estado presionado.

Primero crearemos un fichero XML dentro del directorio *drawable*. En dicho fichero definiremos un selector que asociará varias imágenes a un elemento `View`, dependiendo del estado del mismo. Se puede observar el ejemplo de código de selector para el botón azul en el cuadro 4.1.

```
<?xml version="1.0" encoding="utf-8"?>
<selector
  xmlns:android=
    "http://schemas.android.com/apk/res/android">
  <item
    android:state_pressed="true"
    android:drawable="@drawable/img_blue_press"/>
  <item
  android:state_pressed="false"
  android:drawable="@drawable/img_blue"/>
  <item
  android:drawable="@drawable/img_blue"/>
</selector>
```

Cuadro 4.1. Selector del botón azul *selector_simon_blue.xml*

Declararemos cuatro selectores, uno por cada botón del ejercicio. A continuación crearemos el *layout* correspondiente donde declararemos los elementos `ImageButton`. Se puede observar un ejemplo de código de asignación de un selector en el cuadro 4.2.

```
  <ImageButton
    ...
    android:src="@drawable/selector_simon_blue"/>
```

Cuadro 4.2. *Layout* del juego simón *activity_g_simon.xml*

En la parte lógica del programa, lo primero que tenemos que tener en cuenta es que los botones no se comportarán de la misma forma si estamos en una *activity* o en la otra.

En la *activity* en la que se ejecuta la secuencia de botones:

- Los botonos de colores no deben realizar ninguna acción si se hace click sobre ellos.
- Al hacer clic en el botón central se debe iniciar la ronda.

En la *activity* en la que el usuario debe repetir la secuencia:

- El botón central será únicamente informativo. Nos indica el nivel en el que nos encontramos.
- Al hacer clic sobre algún botón de color se cambiará la imagen y se reproducirá el sonido.

Es fundamental tener en cuenta que los cambios que se realizan en la pantalla mediante programación, por ejemplo cambiar el texto de un elemento `EditText`, se deben comunicar a la interfaz de usuario (UI). Para ello la clase `View` proporciona el método `post`. Mediante este método se comunica a la UI los cambios que debe realizar. En caso de no realizar los cambios mediante el método `post` es posible que no se realicen en el momento en el que los indicamos.

En la *activity* en la que se ejecuta la secuencia de botones:

- En la primera ronda se genera una secuencia aleatoria cuyo tamaño será el número de rondas que tenga asignado el usuario. En la secuencia se almacenarán enteros del 0 al 3 que se corresponden con los botones. La secuencia debe pasarse como parametro a las siguientes *activities*.
- Cuando el usuario presione el botón central se procederá a ejecutar la secuencia. Por ejemplo, si estamos en la ronda número tres, deben autopresionarse los 3 primeros botones de la secuencia. Cada botón se mantendrá presionado durate el tiempo que se haya indicado en el parámetro correspondiente a los segundos.
- Cuando se hayan presionado todos los botones se lanzará la segunda *activity* en la que el usuario debe presionar los botones en el orden correspondiente.

En la *activity* en la que el usuario debe repetir la secuencia:

- El usuario debe presionar los botones y la parte lógia de la atividad debe comprobar si ha presionado el botón correspondiente.
- Si el usuario realiza la ronda mal se muestra un mensaje de error y se vuelve a empezar. Si realiza la ronda bien, se pasa a la siguiente ronda, en caso de ser la última ronda se muestra un mensaje de juego finalizado y se vuelve a empezar. En todos los casos volveremos a la *activity* inicial.

Para reproducir el sonido de los botones usaremos la clase **android.media.AudioTrack**. A esta clase se le debe proporcionar un tono de cierta frecuencia. El tono se representa en forma de sinusoide de la frecuencia deseada. Lo que realmente se le debe pasar a la clase como parámetro son las muestras de la sinusoide almacenadas en un array.

Para generar un tono utilizaremos el código que se muestra en el cuadro 4.3.

```
/* Generamos las muestras del tono */
/* int sampleRate = 8000 */
public byte[] getGeneratedTone(double freqOfTone,
                               double secOfTone) {
  int duration = (int) (Math.ceil(secOfTone/1000));
  int numSamples = duration * sampleRate;
  double sample[] = new double[numSamples];
  byte generatedSnd[] = new byte[2 * numSamples];

  for (int i = 0; i < numSamples; ++i) {
    double div1 = 2*Math.PI*i;
    double div2 = sampleRate/freqOfTone;
    sample[i] = Math.sin(div1/div2);
  }

  int idx = 0;
  int max = 32767;
  for (final double dVal : sample) {
    final short val = (short)((dVal * max));
    this.generatedSnd[idx++] = (byte)(val & 0x00ff);
    thisgeneratedSnd[idx++] = (byte)((val & 0xff00)>>>8);
  }
}
```

Cuadro 4.3. Código para generar un tono de una frecuencia y duración concretas.

Para reproducir el tono utilizaremos la clase **android.media.AudioTrack**. En los cuadros 4.4 4.5 se muestra código básico de un ejemplo para utilizar la clase **AudioTrack**.

```
/* Inicializamos el AudioTrack */
AudioTrack audioTrack =
           new AudioTrack(AudioManager.STREAM_MUSIC,
                          sampleRate,
                          AudioFormat.CHANNEL_OUT_MONO,
                          AudioFormat.ENCODING_PCM_16BIT,
                          (2*numSamples),
                          AudioTrack.MODE_STATIC);
```

Cuadro 4.4. Declaracion del objeto AudioTrack.

```
    ...
int offset = 0;

/*Parar un tono previo.*/
audioTrack.stop();
audioTrack.flush();

/*Inicializar para el nuevo tono.*/
audioTrack.setPlaybackHeadPosition(offset);

/*Asignamos el tono.*/
byte[] tone = getGeneratedTone(freqOfTone, secOfTone);
audioTrack.write(tone, offset, tone.length);

/*Reproducir.*/
audiotrack.play();
...
```

Cuadro 4.5. Reproducir un tono.*activity_g_simon.xml*

Atención

En este juego se muestra el elemento que debe buscar el usuario, en este caso el objetivo será una letra. El usuario debe identificar el objetivo entre una serie de letras y "tocar" la letra correspondiente. Al usuario se le indicará de forma visual y auditiva la letra que debe buscar.

El ejercicio se presentará de 3 formas distintas.

- **Nivel 1 (Objetivo presente):** En el lado izquierdo de la pantalla se muestra el objetivo. En el lado derecho se muestran los distractores. Se utilizará un parámetro de configuración para saber cúantos distractores se deben mostrar (dos, tres, cuatro, seis o nueve). En la figura 4.7 se puede observar un ejemplo.

Figura 4.7. Juego "Atención focalizada (target presente)".

- **Nivel 2 (Objetivo previo):** Primero se mostrará el objetivo, y a continuación se mostrarán todos los distractores juntos. También, se utilizará

un parámetro para saber cúantos distractores se deben mostrar (dos, tres, cuatro, seis o nueve). En la figura 4.8 se puede observar un ejemplo.

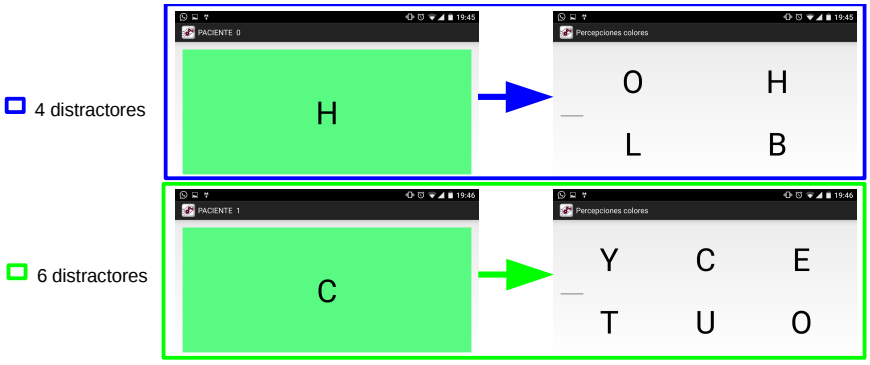

Figura 4.8. Juego "Atención focalizada (target previo)".

- **Nivel 3 (Distractores secuenciales):** Primero se mostrará el objetivo, y a continuación se mostrarán los distractores uno a uno de forma secuencial. También, se utilizará un parámetro para saber cúantos distractores se deben mostrar (desde dos hasta nueve). En la figura 4.9 se puede observar un ejemplo.

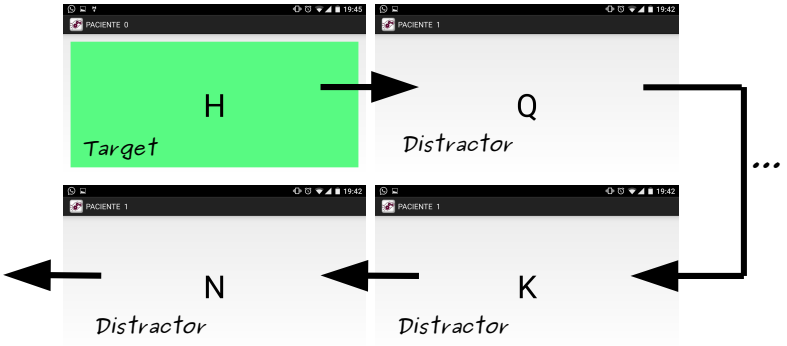

Figura 4.9. Juego "Atención selectiva" en el que los distractores se muestran de forma secuencial.

Aunque la presentación sea distinta, la lógica de los tres juegos es la misma.

1. Se genera una letra y una posición de forma aleatoria.
2. Se genera un array de letras. El array tendrá el tamaño del número de distractores que tenga asignado el usuario. Las letras no deben repetirse entre ellas, ni contener la letra objetivo.

Para indicar al usuario la letra que debe buscar se utilizará la clase `TextToSpeech`. Las activities que utilicen esta clase deben inplementar el método `TextToSpeech.OnInitListener`. Es importante finalizar la clase llamando al

método `shutdown()`, en caso contrario la aplicación perdería rendimiento. En el cuadro 4.6 se muestra un ejemplo de uso de la clase `TextToSpeech`.

```
public class AttentionTargetPrincipalActivity
             implements TextToSpeech.OnInitListener{
  TextToSpeech textToSpeech = null;
  String question;
...
  /* Inicializamos la clase */
  @Override
  protected void onStart(){
    textToSpeech = new TextToSpeech(this, this);
  }

  /* Cuando el objeto textToSpeech esté preparado
     se ejecutará el método onInit */
  @Override
  public void onInit(int i){
    if ( i == TextToSpeech.LANG_MISSING_DATA ||
         i == TextToSpeech.LANG_NOT_SUPPORTED )
      Toast.makeText(this,
                     "ERROR LANG_MISSING_DATA |
                      LANG_NOT_SUPPORTED",
                     Toast.LENGTH_SHORT).show();
    else
      textToSpeech.speak(question,
                         TextToSpeech.QUEUE_FLUSH,
                         null);
    }

  /* Es importante que al terminar
     de usar el objeto TextToSpeech
     se llame al método shutdown() */
  protected void onDestroy() {
    super.onDestroy();
    textToSpeech.stop();
    textToSpeech.shutdown();
  }
}
```

Cuadro 4.6. Código de ejemplo básico de uso de un objeto de la clase TextToSpeech.

Para presentar las letras se definirá la clase `AutoScaleTextView` que extiende

la funcionalidad de la clase `TextView`. Es una clase que ajusta el tamaño del texto al tamaño del contenedor, en este caso un `TextView`. La clase se ha obtenido desde la página web[38] de un desarrollador que se había encontrado con el problema de escalar texto para adaptarse a distintos tamaños de pantalla.

4.3.2. Puzzles

Se estimularán las "Habilidades visoespaciales" con dos ejercicios.

Puzzle de piezas cuadradas

En este juego, como su nombre lo indica, se debe resolver un puzzle. Como se peude observar en la figura 4.10, en la mitad izquierda de la pantalla se muestra la imagen completa del puzzle que se debe resolver y en la mitad derecha las piezas que forman el puzzle.

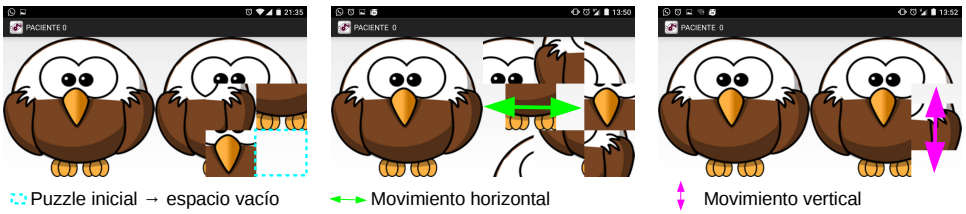

Figura 4.10. Juego "Puzzle".

Para crear las piezas, se divide la imagen principal (cuadrada) en nueve partes iguales que serán las piezas del puzzle. La imagen completa y las partes se almacenarán dentro del sistema dentro del directorio *drawable*. Como se ha mencionado en el apartado 3.1.2, a las imágenes que se encuentran dentro del directorio *drawable* se les asigna un identificador único. Dicho identificador es el que se utiliza para asignar una imagen a un objeto de la clase `view`. En el caso de este juego hemos utilizado objetos de la clase `ImageView`.

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.11.

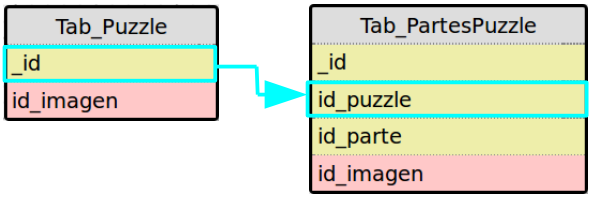

Figura 4.11. Tabla diseñada para el juego "Puzzle".

Para la tabla `Tab_Puzzle` observamos los siguientes campos:

_id.- El identificador de la imagen principal del puzzle.

id_imagen.- El identificador de la imagen dentro del sistema.

Para la tabla `Tab_PartesPuzzle` observamos los siguientes campos:

_id.- El identificador de la imagen en la tabla.

id_puzzle.- El identificador del puzzle. Tendrá que corresponderse con un elemento de la tabla `Tab_Puzzle`.

id_parte.- El identificador de la pieza (de 0 a 7).

id_imagen.- El identificador de la imagen de la pieza dentro del sistema.

Cuando inicamos el puzzle, elegimos la imagen que se va a mostrar. Para ello, primero generamos un valor aleatorio entre los valores del campo *_id* de la tabla `Tab_Puzzle` y obtenemos todos los registros de la tabla `Tab_PartesPuzzle` que se correspondan con el valor que hemos calculado.

En el cuadro 4.7, se muestra el *layout* de la *activity* del puzzle. Este *layout* se ha dividido en dos `RelativeLayout` que servirán de contenedores y cada uno ocupará la mitad de la pantalla. Mediante programación añadiremos el `ImageView` en el `RelativeLayout` que se corresponde con la imagen principal del puzzle y los ocho `ImageView` que serán las piezas del puzzle. Serán ocho y no nueve piezas ya que debe haber un espacio vacío entre las piezas.

```
<LinearLayout xmlns:android=
        "http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="center"
  android:orientation="horizontal">
  <RelativeLayout
    android:id="@+id/puzzleLayoutPrincipal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
  <RelativeLayout
    android:id="@+id/puzzleLayoutParts"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
  </RelativeLayout>
</LinearLayout>
```

Cuadro 4.7. Layout puzzle *activity_g_puzzle.xml*

Esta aplicación podrá ser ejecutada en dispositivos móviles con diversos tamaños y densidades de pantallas, por lo que para posicionar las piezas tendremos que saber el tamaño real, en píxeles, de cada pieza. La forma en la que se han declarado los `RelativeLayout` garantiza que ambos contenedores tendrán las mismas dimensiones. Para saber el tamaño exacto de cada pieza primero obtenemos el tamaño de la pantalla como se muestra en el cuadro 4.8. Para saber el tamaño del contenedor tendremos en cuenta que el ancho es la mitad del ancho de la pantalla y el alto es el alto de la pantalla al que debemos restarle el tamaño de la barra superior (donde se muestra el icono de la aplicación) que obtendremos como se muestra en el código del cuadro 4.9. Para saber el ancho y el largo de cada pieza, simplemente tendremos que dividir el ancho y largo del contenedor por tres.

El tamaño de la barra superior no se puede obtener hasta que no haya sido mostrada en la pantalla, por ello tendremos que calcular el valor en una *activity* previa y pasarla como parámetro a la *activity* en la que la necesitemos.

```
Display dp = getWindowManager().getDefaultDisplay();
DisplayMetrics outMetrics = new DisplayMetrics();
dp.getRealMetrics(outMetrics);

Integer height = outMetrics.heightPixels;
Integer width  = outMetrics.widthPixels;
```

Cuadro 4.8. Tamaño pantalla en pixeles.

```
View decorView  = this.getWindow().getDecorView();
ViewGroup rootView = (ViewGroup) decorView.getRootView();
ViewGroup barContainer = rootView.getChildAt(0);
View bar =  barContainer.getChildAt(0);
float barSize = bar.getTop();
```

Cuadro 4.9. Tamaño barra superior.

Antes de añadir las piezas al contenedor debemos desordenarlas. Para ello se ha seguido un algoritmo en el que en un array de arrays *i,j* se almacenan pares de posiciones.

Como se puede observar en la figura 4.12 empezamos con el *grid* ordenado. Con ordenado, queremos decir que en la posición (1,2) tenemos almacenado el par [1,2]. Para desordenar intercambiaremos el contenido almacenado entre posiciones contíguas. Para ello seguimos los siguientes pasos.

1. Localizamos la posición en la que se encuentra el espacio vacío. En el caso inicial será la posición (2,2) y almacenará el par [2,2].

2. Sabemos que las piezas únicamente se pueden mover en dirección vertical u horizontal, si extrapolamos esta información en cuestión de las posiciones, implica que podremos sumar 1 posición en un único sentido. Esto significa que tendremos que elegir aleatoriamente entre cuatro pares [0,1], [0,-1], [1,0] y [-1,0].

3. Sumamos a la posición en la que tenemos el espacio vacío en par aleatorio que se ha generado y comprobamos que la nueva posición exista. En caso de no existir, multiplicaremos el par por -1. Por ejemplo en la primera posición (2,2) calculamos el par [0,1], la posición suma de ambos sería (2,3) que no existe, ya que las posiciones no pueden ser menores de 0 ni mayores de 2, por lo que si multiplicamos por -1 el par [0,1] el par pasa a ser [0,-1], con lo que la posición suma sería (2,1).

4. Intercambiamos el contenido de ambas posiciones (la del espacio vacío y la contígua calculada). Continuando con el ejemplo anterior, en la posición (2,2) tenemos almacenado el par [2,2] y en la posición contígua calculada (2,1) tenemos almacenado el par [2,1], los intercambiamos y nos queda (2,2)->[2,1] y (2,1) ->[2,2].

5. Repetimos el proceso tantas veces como indique el parámetro de configuración que se ha asignado al usuario.

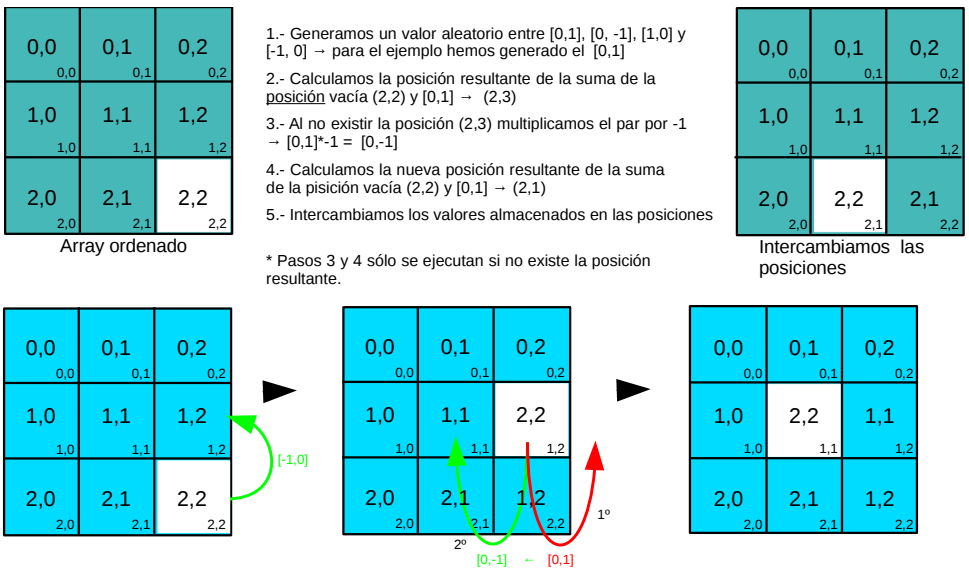

Figura 4.12. Ejemplo algoritmo desorden piezas puzzle.

Los pares de posiciones almacenados en el array se corresponden con las posiciones del array en el que se encuentran almacenadas las piezas. Para colocar las piezas desordenadas recorremos el array donde almacenamos los pares de posiciones y obtenemos el valor almacenado, con dicho valor obtenemos la pieza

que se encuentra en la posición indicada y la colocamos en la posición en la que nos encontramos. En la figura 4.13 podemos observar un ejemplo en el que se representa la forma de colocar las piezas desordenadas.

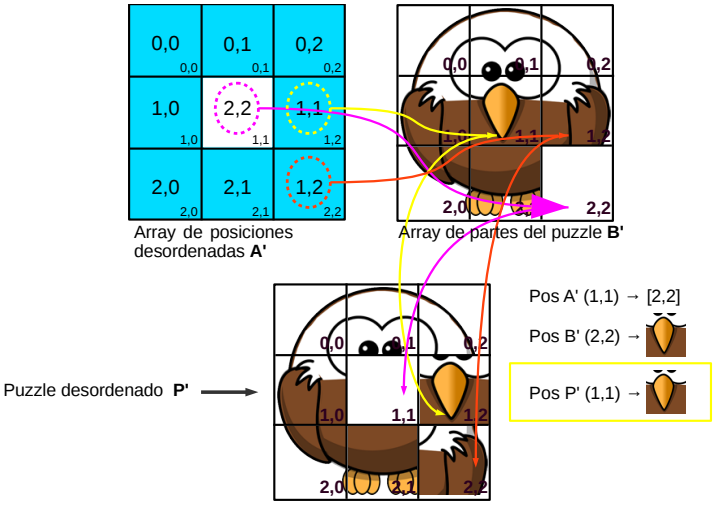

Figura 4.13. Ejemplo colocar imágenes desordenadas en la pantalla

En el cuadro 4.10, se muestra el código java necesario para añadir un elemento en un `RelativeLayout` mediante programación.

```
/* Obtenemos el contenedor */
ViewGroup lp = findViewById(R.id.puzzleLayoutParts);
/* Creamos el elemento */
ImageView iv = new ImageView(this);
iv.setBackgroundResource(id);
iv.setLayoutParams(
        new ViewGroup.LayoutParams(width, height));
iv.setX(posX);
iv.setY(posY);
/* Añadimos el elemento al contenedor */
lp.addView(iv);
```

Cuadro 4.10. Código java para añadir elementos a un RelativeLayout mediante programación.

Como ya hemos mencionado, el juego consiste en ordenar las piezas, para ello tendremos ir moviendo las piezas hacia el espacio vacío hasta colocar todas en su sitio. Para poder arrastrar una `View` por la pantalla le debemos asignar un *Listener* que sepa cuando la hemos pulsado, cuando se mueve y cuando se suelta, y tendrá que actuar en consecuencia. Para ello debemos crear una clase que implemente la interfaz `OnTouchListener`. La clase debe capturar el evento `onTouch` que nos indicará que acción se está realizando sobre la `View`. La clase debe ser capaz de realizar las siguientes acciones.

- Saber si la `View` que se está pulsando se puede mover. Para poder moverse debe ser contígua al espacio vacío.
- En caso de poder moverse, debe corregir el movimiento de la `View` para que únicamente se mueva en la dirección y sentido en la que tenga el espacio vacío. Dicho de otra forma, debemos saber cuanto nos desplazamos en ambos ejes. Únicamente se cambia de posición en el eje y el sentido permitido. Por ejemplo, si deseamos mover la pieza central y tenemos el espacio vacío a la derecha, la posición y siempre será la misma e iremos sumado las cantidad que nos hayamos desplazado en el eje X.
- Acotar el desplzamiento de una pieza entre la pieza y el espacio vacío. Esto significa que se podrá mover en el eje permitido y nunca sobrepasar los límites de su posición inical ni los del espacio vacío.
- También, debe saber cuando ordena correctamente el puzzle.

En el cuadro 4.11 se muestra el ejemplo básico de código java de creación de una clase que implemente la interfaz `OnTouchListener` y en el cuadro 4.12 se muestra la asignación del *Listener* a una `View`.

```
final class MyTouchListener implements OnTouchListener {
  public boolean onTouch(View view, MotionEvent motEv) {
    if(motEv.getAction() == MotionEvent.ACTION_UP){
      ...
    }else if(motEv.getAction() == MotionEvent.ACTION_MOVE){
      ...
    }else if(motEv.getAction() == MotionEvent.ACTION_DOWN){

    } ...
}
```

Cuadro 4.11. Ejemplo de clase que implementa OnTouchListener.

```
  ImageView iv = ...;
  iv.setOnTouchListener(new MyTouchListener());
```

Cuadro 4.12. Ejemplo de clase que implementa OnTouchListener.

Colorea

Como se pude observar en la figura 4.14, en la mitad izquierda de la pantalla se muestra una imagen en blanco y negro y en la mitad derecha las piezas de diversas formas que componen la imagen principal.

[1]Ejemplo de redimensión de una *activity* para varios tamaños y densidades de pantalla.

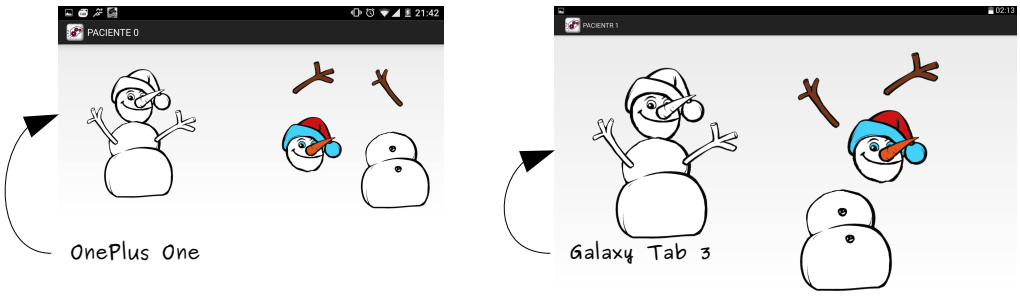

Figura 4.14. Juego "Colorea" en dos pantallas de tamaño distinto.[1]

Para crear las piezas se han dividido varias imágenes en partes sin una forma predeterminada. En la mitad izquierda de la pantalla se muestran las piezas en blanco y negro y ordenadas. En la mitad derecha se muestras las mismas piezas pero a color y desordenadas.

Como ya se ha mencionado en la descripción del juego **Puzzle de piezas cuadradas**, las pantallas de los dispositivos android pueden tiener varios tamaños y densidades. Esto es un problema debido a que las piezas de este juego son irregulares y en el caso de las piezas en blanco y negro deben estar colocadas en una posición predeterminada. Si creamos un *layout* presentando las piezas para un dispositivo concreto, en el resto de dispositivos no se ajustará del todo bien. En la figura 4.15 se muestra un ejemplo del problema de las pantallas, se ha creado un *layout* específico para los dispositivos *Nexus 4* y se ha ejecutado la aplicación en un *OnePlus One* y un *Galaxy Tab 3*, y como se puede observar, la visualización en las pantallas es distintas.

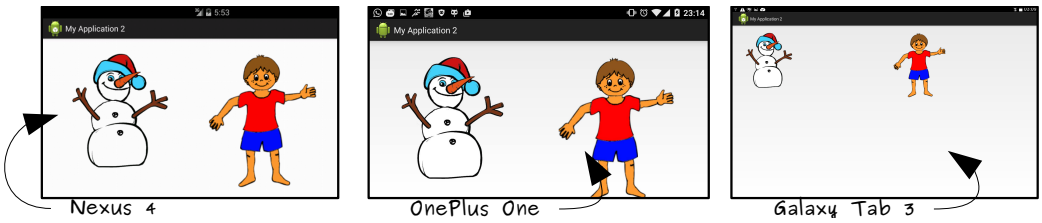

Figura 4.15. Ejemplo pantallas sin redimensión.[2]

Para solventar este problema, se utilizan posiciones y tamaños base, tanto de las `view` como de la pantalla en función de un dispositivo y se calculan los valores de posiciones y tamaños para cualquier pantalla.

Para poder redimensionar de forma eficaz se ha diseñado el algoritmo que se describe a continuación:

- Primero tenemos que obtener la densidad y los valores del ancho y largo de la pantalla base en dps (Density-independent pixel), esto se consigue

[2]Ejemplo de redimensión de una *activity* para varios tamaños y densidades de pantalla.

obteniendo los valores en píxeles y dividiendo por la densidad. Para un cálculo más exacto tendremos que restar al alto el valor de la barra superior.

- Obtenemos los valores de la pantalla en la que se va a ejecutar la aplicación.
- Los valores de posiciones y tamaños de cada pantalla se relacionan de la siguiente forma (llamaremos PB a la pantalla base y PA a la pantalla en la que se ejecuta) `valorPB*densidadPA = valorPA*densidadPB` lo que implica que `valorPa = valorPA*densidadPB/densidadPA`
- Con los valores de posición y tamaño del elemento nos queda calcular la proporción del alto y del ancho de la pantalla. Estos valores de obtienen dividiendo los valores de la pantalla actual por los valores base.
- Para hallar el valor real debemos multiplicar el valor que se correspondería con el elemento actual por la proporción correspondiente.

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.16, en la que se observan los siguietes campos.

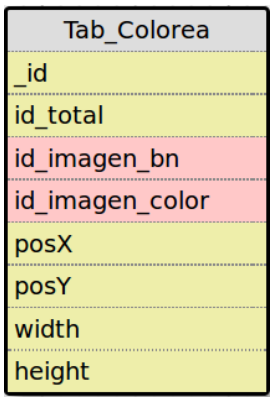

Figura 4.16. Tabla diseñada para el juego "Colorea".

_id.- El identificador de la imagen en la tabla.

id_total.- El identificador total del puzzle. Todas las piezas de un puzzle tendrán una clave común.

id_imagen_bn.- El identificador de la imagen de la pieza en blanco y negro dentro del sistema.

id_imagen_color.- El identificador de la imagen de la pieza a color dentro del sistema.

posX.- Valor base de la posición de la pieza en el eje X.

posY.- Valor base de la posición de la pieza en el eje Y.

width.- Valor base del ancho de la imagen.

height.- Valor base del alto de la imagen.

Para presentar este ejercicio se ha usado el mismo *layout* que en el ejercicio **Puzzle**, ya que la forma de presentar el juego es la misma. Para desordenar las piezas de color se ha creado un algoritmo que posiciona de forma aleatoria todas las piezas dentro del contendor sin sobrepasar los límites.

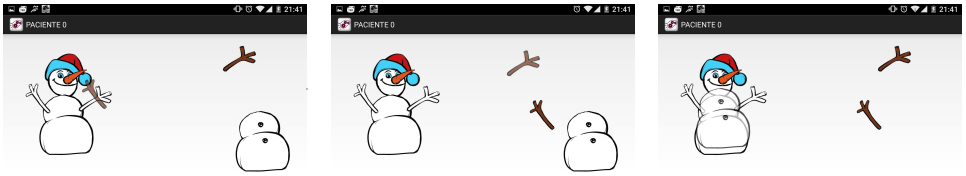

Figura 4.17. Ejemplo de *drag and drop*.

Debemos mover las piezas de color a su posición correspondiente en el lado izquierdo de la pantalla, como se muestra en la figura 4.17. Para ello y al igual que en los caso del ejercicio **Puzzle**, debemos asignar a las `views` un *listener* para el evento *onTouch*, aunque para este ejercicio de lo único que se encargará es de darle trasparencia a la imagen que se ha pulsado como se muestra en el cuadro 4.13. También debemos declarar una clase que implemente la intefaz `OnDragListener` que será la encargada de determinar si la pieza a color que hemos arrastrado y soltado sobre una pieza en blanco y negro se corresponden. A las piezas a color se les asignará la clase que implementa la interfaz de `OnTouchListener`, ya que son las que vamos a arrastrar y a las piezas en blanco y negro se les asignará la clase que implementa la interfaz de `OnDragListener`, ya que sobre ellas soltaremos las piezas de color.

```
public boolean onTouch(View view, MotionEvent motEv) {
  if (motEv.getAction() == MotionEvent.ACTION_DOWN) {
    ClipData data = ClipData.newPlainText("", "");
    DragShadowBuilder shadowB = new DragShadowBuilder(view);
    view.startDrag(data, shadowB, view, 0);
    view.setVisibility(View.INVISIBLE );
    return true;
  } else {
    return false;
  }
}
```

Cuadro 4.13. Código java de la clase que impleneta la interfaz para OnTouchListener.

Si soltamos una pieza de color sobre una pieza que no se corresponde o en cualquier posición de la pantalla que no se corresponda con su par, se devolverá la pieza a su posición original. En caso de soltar la pieza en la posición de su par correspondiente se intercambiarán las imágenes de las piezas y la pieza que antes contenía la imagen a color se hará invisible.

4.3.3. Esquema corporal

En esta categoría se estimulará la capacidad "Gnosica" con un ejercicio que se dividirá en dos juegos.

Este ejercicio consiste en presentar la imagen de un cuerpo o de un rostro, como se puede observar en la figura 4.18 y preguntar al usuario, de forma auditiva, por una o más partes de la imagen que se presenta. La cantidad de partes por las que se pregunta depende de un parámetro de configuración asignado al usuario.

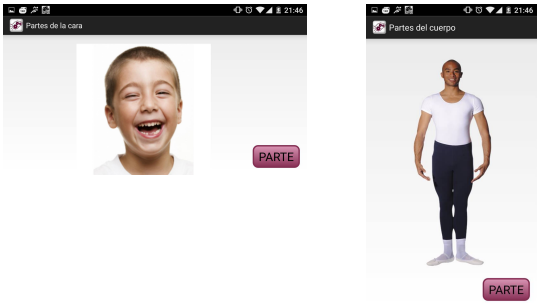

Figura 4.18. Juegos de esquema corporal.

Al igual que los juegos de la categoría **Puzzle** tenemos que tener en cuenta las dimensiones de la pantalla, ya que para controlar qué parte del cuerpo está tocando el usuario lo que se ha hecho es diseñar *layouts* de ejemplo en el que colocamos la imágen que deseamos mostrar y sobre ella botones sobre los que haremos clic en la parte correspondiente. Se muestra un ejemplo en la figura 4.19. Lo que almacenaremos en base de datos será el tamaño y posición de la imagen principal y el tamaño y posición de los botones que al presentarse en la *activity* serán invisibles.

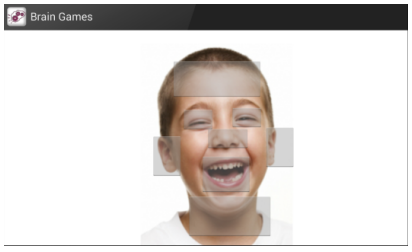

Figura 4.19. Ejemplo de *layout* en el que se muestran la posición de los botones.

Para la asignación y elección de la imagen que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.20, en la que se observan los siguietes campos.

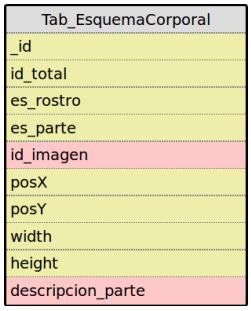

Figura 4.20. Tabla diseñada para los juegos de la categoría "Esquema corporal".

_id.- El identificador de la imagen en la tabla.

id_total.- El identificador total del juego. La imagen principal y los botones correspondientes tendrán una clave común que servirá para hacer la búsqueda.

es_rostro.- Booleano que indica si el grupo se corresponde con un rostro o un cuerpo.

es_parte.- Booleano que indica si el registro se corresponde con la imagen principal o con las partes.

id_imagen.- El identificador de la imagen dentro del sistema.

posX.- Valor base de la posición en el eje X.

posY.- Valor base de la posición en el eje Y.

width.- Valor base del ancho.

height.- Valor base del alto.

id_imagen_color.- El identificador de la imagen de la pieza a color dentro del sistema.

Para la elección de la imagen seguiremos los siguientes pasos:

1. Calculamos un valor aleatorio entre los elementos del campo `id_total` que se correspondan con cuerpo o cara.
2. Obtenemos todos los registros que se correspondan con el identificador previamente calculado.

3. Asignamos y redimensionamos la imagen principal y los botones (invibles).
4. Calculamos n descripciones aleatorias entre los registros que no son el principal.

Para indicar de forma auditiva que emoción debe buscar el usuario, se utilizará la clase `TextToSpeech`.

4.3.4. Reconocer emociones

En esta categoría se estimulará la capacidad de "Cognición social" con un ejercicio de reconocer emociones.

Imágenes

En este juego se mostrarán dos o tres imágenes de una misma persona que expresa distintas emociones. Al usuario se le indicará de forma visual y auditiva la emoción que debe señalar. El aspecto visual del juego es el que se muestra en la figura 4.21.

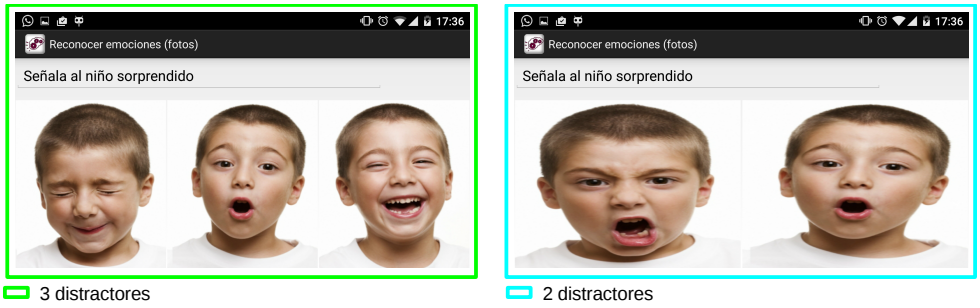

Figura 4.21. Juego "Reconocer emociones: imágenes".[3]

Como podemos observar en la figura 4.21, en una iteración del juego siempre se muestra el rostro de la misma persona, aunque en la aplicación hay almacenados varios rostros de personas distintas.

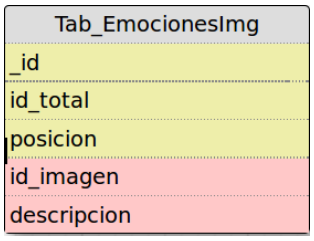

Figura 4.22. Tabla diseñada para el juego "Reconocer emociones: imágenes".

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.22 donde observamos los siguientes campos:

[3]Juego "Reconocer emociones: imágenes". En la pantalla de la derecha se muestra el juego para un usuario al que se le deben mostrar dos elementos distractores y en la pantalla de la izquierda el juego con tres elementos distractores

id_total.- El identificador del rostro.

posicion.- El identificador de la posicion del rostro empezando en 0.

id_imagen.- El identificador de la imagen dentro del sistema.

descripcion.- En este campo almacenamos la pregunta correspondiente al rostro y la expresión del mismo.

Para elegir las imágenes que se van a mostrar en cada iteración se siguen los siguientes pasos:

1. Generar un valor aleatorio entre 0 y el mayor valor del campo *id_total*.
2. Obtener todos los registros que se correspondan con el valor aleatorio para el campo *id_total*.
3. Elegir de forma aleatoria dos o tres registros y almacenarlos en un array.
4. Generar una posición aleatoria en el array. Se corresponderá con la posición del objeto `ImageView` correspondiente en el *layout* (de izquierda a derecha).

Al igual que en el juego "Atención focalizada", para indicar de forma auditiva que emoción debe buscar el usuario, se utilizará la clase `TextToSpeech`.

Videos

En este juego se mostrarán dos videos de distintas personas que expresan distintas emociones. Al usuario se le indicará de forma visual y auditiva la emoción que debe señalar. El aspecto visual del juego es el que se muestra en la figura 4.21.

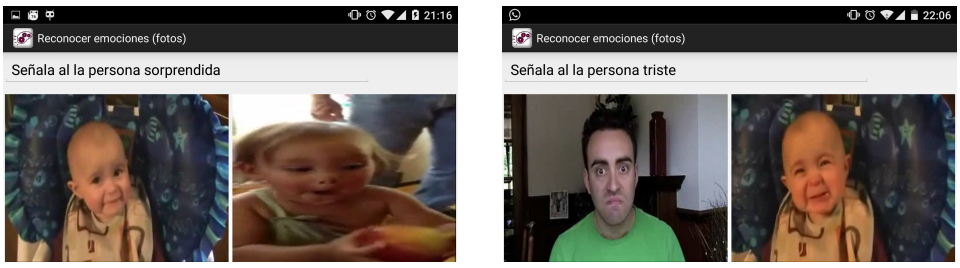

Figura 4.23. Juego "Reconocer emociones: videos".

Para mostrar los videos, primero debemos tenerlos almacenados dentro del sistema. Para ello las almacenaremos dentro del directorio *raw*. Como se ha mencionado en el apartado 3.1.2, a los videos que se encuentran dentro del directorio *raw* se les asigna un identificador único. Dicho identificador es el que se utiliza para asignar un video a un objeto de la clase `view`. En el caso de este juego hemos utilizado un objeto `VideoView`.

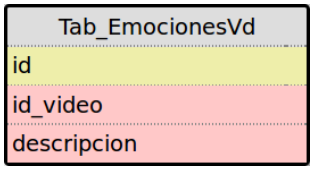

Figura 4.24. Tabla diseñada para el juego Reconocer emociones videos.

Para la asignación y elección de los videos que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.24 donde observamos los siguientes campos:

id.- El identificador del video.

id_video.- El identificador del video dentro del sistema.

descripcion.- En este campo almacenamos la pregunta correspondiente ala expresión de la persona del video.

Para elegir los ideos que se van a mostrar, se siguen los siguientes pasos:

1. Generar dos valores aleatorio entre los valores del campo *id*.
2. Almacenar los registros correspondientes a los valores del campo *id* y almacenarlos en un array.
3. Elegir de forma aleatoria el valor 0 ó 1 que se corresponderá con la posición del objeto `VideoView` correspondiente en el *layout* (de izquierda a derecha).

Al igual que en el juego "Reconocer emociones: imágenes", para indicar de forma auditiva que emoció debe buscar el usuario, se utilizará la clase `TextToSpeech`.

Este juego no se incluirá en la versión final de la aplicación para las terapeutas debido a que han considerado que el nivel de dificultad es muy alto.

4.3.5. Gnosias

En esta categoría se estimulará la capacidad "Gnosica" con tres ejercicios.

Sonidos

En este juego se mostrarán dos o tres imágenes de distintos animales. Al usuario se le indicará el animal que debe señalar de forma auditiva, de modo que si debe buscar un león, se escuchará el rugir de un león. El aspecto visual del juego es el que se muestra en la figura 4.25.

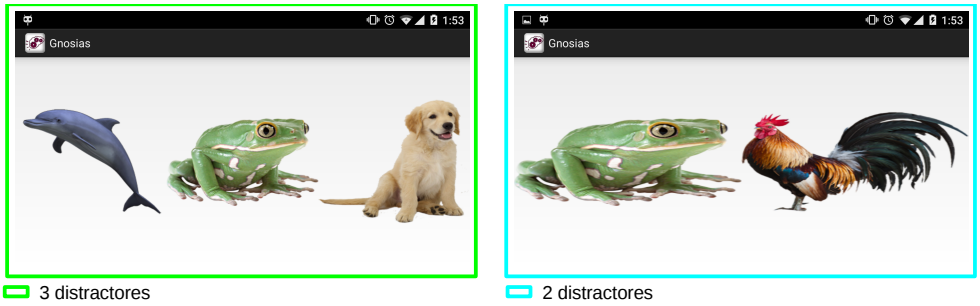

Figura 4.25. Juego "Sonidos".

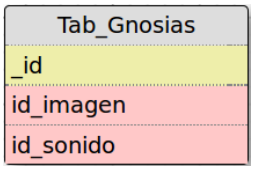

Figura 4.26. Tabla diseñada para el juego Sonidos.

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se diseñó la tabla que se muestra en la figura 4.26 donde observamos los siguientes campos:

_id.- El identificador de la imagen del inimal.

id_imagen.- El identificador de la imagen de animal dentro del sistema.

id_sonido.- El identificador del sonido de animal asociado a la imagen.

Para elegir las imágenes que se van a mostrar, se siguen los siguientes pasos:

1. Generar un valor aleatorio entre 0 y el mayor valor del campo *_id*, para obtener el identificador de la imagen que debe señalar el usuario.

2. Generar los valores aleatorios necesarios (número de elementos menos uno), que serán los distractores.

3. Almacenar, de forma aleatoria todos los elementos en un array.

4. Las posiciones del array se corresponderán con las posiciones de los objetos `ImageView` correspondiente en el *layout*.

A diferencia de otros juegos, en este se reproducirá un sonido. Para ello utilizaremos la clase `MediaPlayer`. En el cuadro 4.14 se muestra un ejemplo de uso de la clase.

```
public static void Sound(Integer idSound, int sec){
  (new Thread(){
    public void run(){
    mPlayer =  MediaPlayer.create(this, idSound);
    mPlayer.start();
    SystemClock.sleep(sec * 1000);
    /* Es importante liberar recursos */
    try {
      mPlayer.stop();
      mPlayer.reset();
      mPlayer.release();
      finalize();
      System.gc();
    } catch (Throwable e) {
      ...
    }
  }}).start();
}
```

Cuadro 4.14. Código de ejemplo básico de uso de un objeto de la clase MediaPlayer.

Siluetas

En este juego se mostrará una silueta y varias imágenes de diversos elementos. El usuario tendrá que señalar el elemento que se corresponde con la silueta mostrada.

Figura 4.27. Juego "Siluetas".

Como se puede observa en la figura 4.27, en la parte izquierda de la pantalla mostramos una silueta y en la parte derecha mostramos dos, tres, cuatro o seis imágenes de diversos elementos.

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se ha diseñado la tabla que se muestran en la figura 4.28, con los siguiente elementos:

_id.- El identificador del elemento dentro de la tabla.

id_imagen.- El identificador de la imagen dentro del sistema.

id_imagen_silueta.- El identificador de la silueta correspondiente para la imagen dentro del sistema.

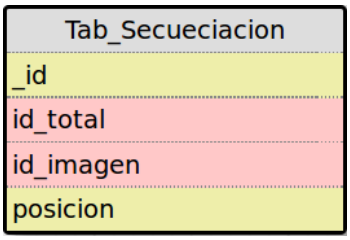

Figura 4.28. Tabla diseñada para el juego "Siluetas".

Para elegir las imágenes que se van a mostrar, la lógica es la misma que en el juego "Sonidos".

Colores

En este juego se mostrarán una serie de colores y se indicará al usuario de forma visual y auditiva el color que debe buscar. En la figura 4.29 se muestra el aspecto visual del juego. Se mostrarán dos, tres, cuatro, seis o nueve colores.

Figura 4.29. Juego "Colores".

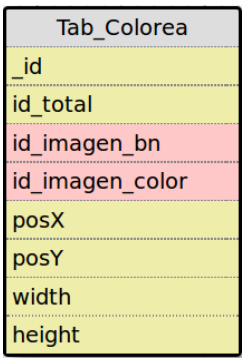

Figura 4.30. Tabla diseñada para el juego "Colores".

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se ha diseñado la tabla que se muestran en la figura 4.30, con los siguiente elementos:

_id.- El identificador del elemento dentro de la tabla.

id_imagen.- El identificador de la imagen dentro del sistema.

id_imagen_silueta.- El identificador de la silueta correspondiente para la imagen dentro del sistema.

Para elegir las imágenes que se van a mostrar, la lógica es la misma que en el juego "Gnosias".

4.3.6. Funciones ejecutivas

En esta categoría se estimulará la capacidad que se encarga de las "Funciones ejecutivas" con dos ejercicios.

Categorización

En este juego se mostrarán varias imágenes de distintos elementos. En el juego se pedirá al usuario que señale todos los objetos que pertenezcan a una categoría concreta. Por ejemplo, se muestran cinco objetos, se pregunta *¿Qué elementos son animales?* y el usuario debe señalar todos los elementos que pertenezcan a la categoría de animales.

El aspecto visual del juego es el que se muestra en la figura 4.31. Como podemos observar, en el juego se pueden presentar varios elementos distractores, desde dos hasta nueve, y varios elementos correctos. Los elementos correctos pueden ser desde 1 hasta el número de elementos distractores.

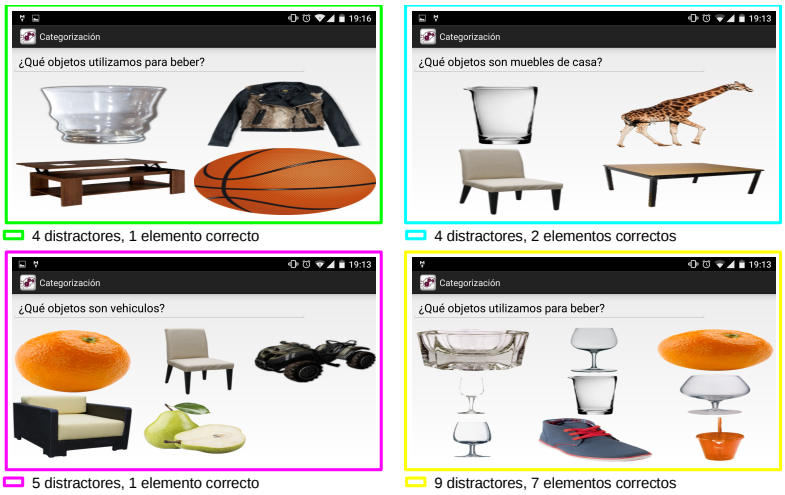

Figura 4.31. Juego "Categorización".

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se diseñaron las tablas que se muestran en la figura 4.32.

Para la tabla `Tab_Categorias` observamos los siguientes campos:

_id.- El identificador de la categoría.

descripcion.- La pregunta que se realizará en caso de ser la categoría seleccionada.

Para la tabla `Tab_ElementosCategorias` observamos los siguientes campos:

_id.- El identificador del elemento en la tabla.

id_categoria.- El identificador de la categoría. Tendrá que corresponderse con un elemento de la tabla `Tab_Categorias`.

id_imagen.- El identificador de la imagen dentro del sistema.

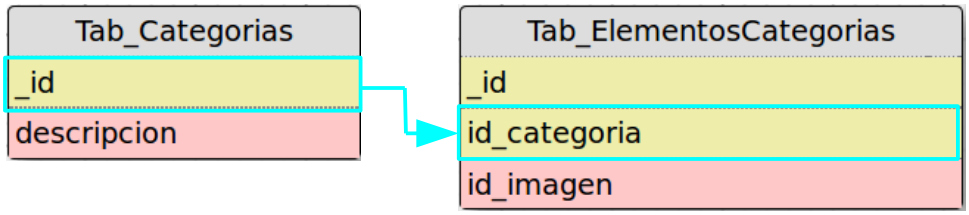

Figura 4.32. Tablas diseñadas para el juego "Categorización".

Para elegir las imágenes que se van a mostrar, se siguen los siguientes pasos:

1. Generar un valor aleatorio entre 0 y el mayor valor del campo *_id* de la tabla `Tab_Categorias`, para obtener la categoría.
2. Obtener todos los registros que se correspondan con el valor de la categoría.
3. Elegir entre todos los elementos de la categoría, los elementos que se van a mostrar.
4. Elegir, aleatoriamente, elementos que no se correspondan con la categoría seleccionada.
5. Almacenar, de forma aleatoria, todos los elementos en un array.
6. Las posiciones del array se corresponderán con las posiciones de los objetos `ImageView` correspondiente en el *layout*.

Cuando el usuario señala un elemento, la aplicaión reproduce un sonido que indica si lo ha hecho bien o mal.

- Si se señala un elemento que no se corresponda con la categoría, se reproduce un sonido que indica que se ha fallado.
- Si se señala un elemento que se corresponda con la categoría, y es la última, se reproduce un sonido que indica que se ha ganado y el juego vuelve a empezar.

- Si se señala un elemento que se corresponda con la categoría, pero no es la última, se reproduce un sonido que indica que se ha acertado y se sigue jugando.

Al igual que en el juego "Buscar la letra", para indicar de forma auditiva que emoció debe buscar el usuario, se utilizará la clase `TextToSpeech`.

Secuenciación

En este juego se mostrarán dos o tres imágenes que representan una actividad, en cada imagen se muestra una acción concreta de la actividad. El usuario tendrá que señalar las acciones en el orden en el ocurren en la actividad. Por ejemplo, en la pantalla de tres elementos de la figura 4.33, se puede ver a una niña, el orden en el que se deben señalar las imágenes es el siguiente, primero siembra las semillas, luego las riega y finalmente brotan las flores.

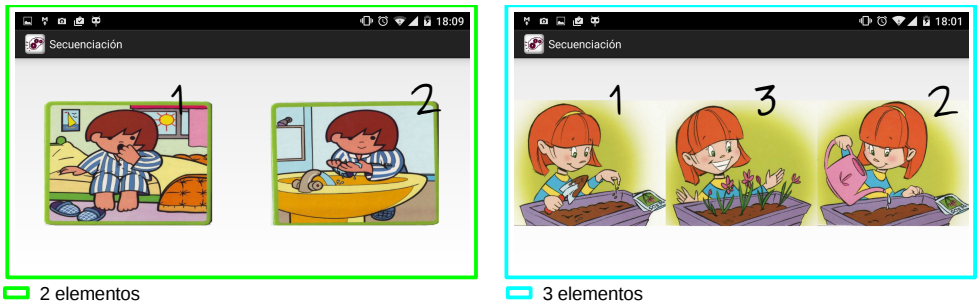

Figura 4.33. Juego "Categorización".

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se ha diseñado la tabla que se muestran en la figura 4.34.

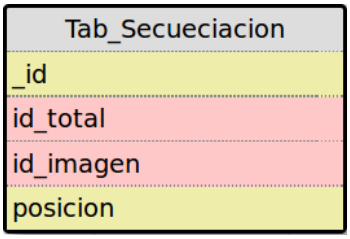

Figura 4.34. Tabla diseñada para el juego "Secuenciación de actividades".

La tabla tiene los siguientes campos:

id_total.- El identificador de la actividad.

id_imagen.- El identificador de la imagen dentro del sistema.

posicion.- El número de orden de la acción representada en la imagen.

Para elegir las imágenes que se van a mostrar, se siguen los siguientes pasos:

1. Generar un valor aleatorio entre 0 y el mayor valor del campo *id_ total*, para obtener el identificador de la actividad.

2. Obtener los registros que se correspondan con el valor de la actividad.

3. Almacenar, de forma aleatoria los registros en un array.

4. Las posiciones del array se corresponderán con las posiciones de los objetos `ImageView` correspondiente en el *layout* y el campo *posicion* del elemento se corresponde con el orden en el que debe ser seleccionado.

Cuando el usuario señala una acción, la aplicación reproduce un sonido que indica si ha señalado la acción correspondiente.

- Si se señala una acción que no es la que corresponde, se reproduce un sonido que indica que se ha fallado y se reinicia el juego.

- Si se señala la acción que corresponde, y es la última, se reproduce un sonido que indica que se ha ganado y el juego vuelve a empezar.

- Si se señala la acción que corresponde, pero no es la última, se reproduce un sonido que indica que se ha acertado y se sigue jugando.

4.3.7. Cálculo y razonamiento

En esta categoría se estimulará la capacidad de "Razonamiento" con tres ejercicios que se dividen en cinco juegos.

Cálculo

En este ejercicio se mostrará e indicará auditivamente una operación matemática con dos, tres o cuatro posibles respuestas. El aspecto visual del juego es el que se muestra en la figura 4.35. Como podemos observar, el ejercicio se dividirá en tres juegos, uno por cada operación que se puede realizar, **Sumas**, **Restas** y **Multiplicaciones**. Cada juego puede mostrar dos, tres o cuatro posibles resultados y las operaciones pueden tener valores de unidades, decenas o centenas. En el caso de las multipliaciones, el segundo operando siempre será de un dígito.

Al igual que en los juegos de "Busca la letra" los valores de presentan en elementos `AutoScaleTextView` y se reproducen las preguntas con la clase `TextToSpeech`.

Figura 4.35. Juegos "Sumas", "Restas" y "Multiplicaciones".

Compara valores

Como se puede observar en la figura 4.36, en este juego se presentan dos números enteros aleatorios y se pregunta al usuario *¿qué número es mayor/menor?*

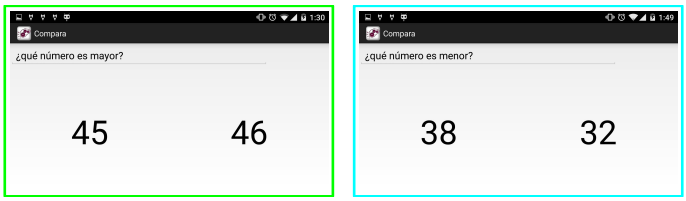

Figura 4.36. Juego "Compara".

Al igual que para los juegos "Sumas", "Restas" y "Multiplicaciones" los valores de presentan en elementos `AutoScaleTextView` y se reproducen las preguntas con la clase `TextToSpeech`.

Compara precios

Como se puede observar en la figura 4.37, en este juego se presentan dos objetos, uno cuesta mucho más caro que el otro y se le pregunta al usuario *¿qué objeto cuesta más/menos?*.

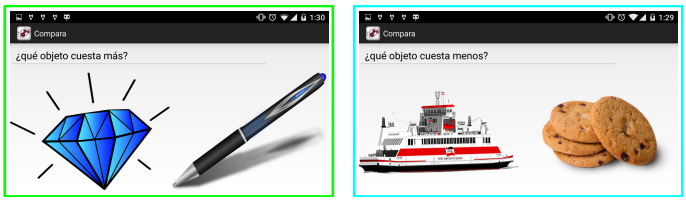

Figura 4.37. Juego "Compara precios".

Para la asignación y elección de las imágenes que se van a mostrar en el juego, se ha diseñado la tabla que se muestran en la figura 4.38. La tabla cuenta con los siguientes campos:

_id.- El identificador de la imagen en la tabla.

id_imagen.- El identificador de la imagen dentro del sistema.

es_barato.- Es un boleano que indica si el objeto de la imagen es caro o barato

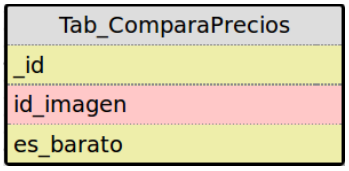

Figura 4.38. Tabla diseñada para el juego "Compara precios".

Para elegir las imágenes que se van a mostrar, se siguen los siguientes pasos:

1. Se elige un elemento barato y otro caro.
2. Se elige aleatoriamente una posición para cada elemento.
3. Se elige aleatoriamente si se va a preguntar por el elemento barato o caro.

Al igual que para el juego "Compara"las preguntas se reproducen mediante un objeto de la clase `TextToSpeech`.

4.4. Bases de datos

En la sección 4.2.2 se muestra el *Esquema de base de datos* que utiliza la aplicación para almacenar los datos necesarios de los pacientes, los parámetros de configuración y las estadísticas de los juegos. En esta sección se contará la gestión y visualización de los datos.

4.4.1. Codificación

Para codificar las bases de datos, Android proporciona la librería SQLite **android.database.sqlite** nos permitirá utilizar bases de datos mediante el lenguaje SQL, de una forma sencilla y utilizando muy pocos recursos del sistema.

Las clases encargadas de gestionar las bases de datos tendrán que extender la clase `SQLiteOpenHelper`. En el cuadro 4.15 se muestra parte de la clase que se encarga de gestionar la tabla de usuarios.

```
/* Clase encargada de la getión de los usuarios */
public class DBUsers extends SQLiteOpenHelper {
  public final static String TAB_NAME = "Tab_Usuarios";
  ...
  public final static String CAM_USER_ID = "_id";
```

```
private final static String CREATE =
    "CREATE TABLE if not exists " + TAB_NAME + " (" +
    CAM_USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
    CAM_USER_DATE + " DATE , " +
    CAM_USER_CODE + " TEXT , " +
    CAM_USER_FIRST_NAME + " TEXT , " +
    CAM_USER_LAST_NAME + " TEXT);";

public DBUsers(Context context) {
  super(context, FILE_NAME, null, version);
}

@Override
public void onCreate(SQLiteDatabase db) {
  db.execSQL(CREATE);
  /* Permite utilizar el campo _id
     como foreignkey en otra tabla */
  db.execSQL("PRAGMA foreign_keys = ON;");
}

@Override
public void onUpgrade(SQLiteDatabase db,
                      int oldVersion,
                      int newVersion) {
  // TODO Auto-generated method stub
  doReset(db);
}

private void doReset(SQLiteDatabase database) {
  database.execSQL("DROP TABLE IF EXISTS " + TAB_NAME);
  onCreate(database);
}

/* Insertamos un usuario */
public boolean insert(String userCode,
                      String userFirstName,
                      String userLastName,
                      String userBirthday) {
  String auxUserCode = userCode.toUpperCase();
  String auxUserFirstName = userFirstName.toUpperCase();
  String auxUserLastName = userLastName.toUpperCase();

  boolean existsUser = existsUser(String userFirstName,
```

```
                                        String userLastName)
    if(!existsUser){
      SQLiteDatabase database = getWritableDatabase();
      ContentValues values = new ContentValues();
      values.put(CAM_USER_CODE, auxUserCode);
      values.put(CAM_USER_FIRST_NAME, auxUserFirstName);
      values.put(CAM_USER_LAST_NAME, auxUserLastName);
      values.put(CAM_USER_DATE, userBirthday);
      database.insert(TAB_NAME, null, values);
      cursor.close();
      database.close();
    }
    return existsUser;
  }

  ...

  /* Comprobamos si el usuario existe */
   public boolean existsUser(String userFirstName,
                             String userLastName) {
    boolean exists = true;
    String auxUserFirstName = userFirstName.toUpperCase();
    String auxUserLastName = userLastName.toUpperCase();
    SQLiteDatabase database = getReadableDatabase();
    String query = "SELECT * FROM " + TAB_NAME +
                   " WHERE " + CAM_USER_FIRST_NAME + "=? " +
                   " and " + CAM_USER_LAST_NAME + "=?";
    String selection[] = {auxUserFirstName, auxUserLastName};
    Cursor cursor = database.rawQuery(query, selection);
    if (cursor.getCount() == 0 && auxUserFirstName != null)
      exists = false;
    cursor.close();
    database.close();

    return exists;
  }
  ...
}
```

Cuadro 4.15. Código java de la clase que extiende la funcionalidad de la clase SQLiteOpenHelper para gestionar los datos de los usuarios.

Como se puede observar en el cuadro 4.15 en el constructor de la clase se ejecuta el método `super(context, FILE_NAME, null, version)`, donde debemos pasarle el nombre del fichero donde almacenamos la tabla, en el ejemplo la constante `FILE_NAME` almacena el nombre del fichero `Usuarios.db`.

Guardar los datos estadístios de todos los juegos y todos los usuarios en la misma tabla, supone que es posible un crecimiento exponencial del contenido de la tabla. Para solventar este problema se ha decidido almacenar los datos en más de un fichero. Se creará un fichero cada mes que se nombrerá de la siguiente forma `"Statics_Games_[yyyyMM].db`. Cuando deseemos guardar o reuperar estadíticas comprobaremos que exita el fichero que se corresponda con el mes y año del fichero y lo utilizaremos para inicializar la clase.

Para gestionar las fechas sa ha utilizado las clases `Calendar` y `Date` de la librería `java.util`. La clase `Date` crea una instancia de esta clase representa un instante concreto en el tiempo, con una precisión de milisegundos. Independientemente de la zona horaria del sistema los valores se darán en UTC. Instancias de esta clase son adecuados para la comparación, pero poco más. La clase `Calendar`, al igual que la clase Date, una instancia de esta clase representa un instante concreto en el tiempo con una precisión de milisegundos. Es una clase abstracta que se utiliza para la conversión entre un objeto Date y un conjunto de campos enteros como año, mes, día, hora, y así sucesivamente. Se puede obtener una instancia de la clase `Date` a partir de la clase `Calendar` con el método `getTime()`.

Para almacenar las fechas se guardará el tiempo en milisegundos y para mostrarla se utilizará la clase `SimpleDateFormat` para formatear una instancia de la clase `Date`. En el cuadro 4.16 se muestra un ejemplo de formateo de fechas.

```
/* Obtener una instancia de la clase Calendar
   para la fecha actual */
protected Calendar date = Calendar.getInstance();
/* Obtener un string de la fecha formteada */
String dateFormat = (new SimpleDateFormat("dd/MM/yyyy")).
                    format(date.getTime());
```

Cuadro 4.16. Código java de ejemplo de formateo fechas.

4.4.2. Gestión y visualización

En este apartado contaremos cómo acceden los juegos a los datos de cada usuario, cómo almacenan las estadísticas y cómo los terapeutas puedes acceder y modificar dicha información.

Desde la sección de administración de la aplicación se crean y eliminan usuarios, se gestionan los parámetros de configuración de los juegos correspondiente a cada usuario y se visualizan las estadísticas.

Cuando se crea un usuario, también se crean todos los registros correspondientes con los parámetros de configuración en sus respectivas tablas y se les asigna unos valores base. Para gestionar dichos valores se accede a la pantalla de *Actualización* donde se presentarán primero las categorías. Cuando se selecionamos una categoría accedemos a una pantalla en la que se muestran una pestaña de configuración por cada juego de la categoría. En cada pestaña se muestran y editan los parámetros de configuración del juego correspondiente. En la figura 4.39 se muestran dos ejemplos de pestañas de gestion de parámetros configuración.

Figura 4.39. Ejemplo de pestañas de gestión de parámetros de configuració de juegos.

A medida que los usuarios realizan ejercicios de estimulación cognitiva, los juegos capturan datos estádisticos sobre el progreso de cada usuario. Dicha información se presenta al terapeuta de dos formas. La primera es mediante una visualización a modo de tabla, donde se presentan por columnas los datos más importantes para cada juego. La segunda es mediante la visualización de una gráfica.

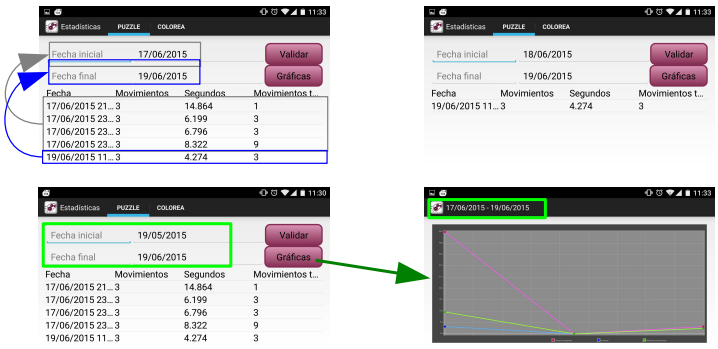

Figura 4.40. Ejemplo de presentación de estadísticas.

En la figura 4.40 se muestran las estadísticas del juego “Puzzle”. Como se puede observar en la figura, las estadísticas se filtran por fecha. Para poder realizar el filtro, primero se comprueba qué ficheros coinciden con los intervalos seleccionados en los filtros y luegos se obtinen los registros correspondientes del fichero.

Para mostrar las gráficas se ha utilizado la versión 0.6.1 de la librería externa `androidplot-core` [39].

Capítulo 5
Conclusiones

En este Trabajo de Fin de Grado se ha realizado una aplicación de estimulación cognitiva para el sistema operativo Android. En los capítulos anteriores se han contado los factores que nos motivaron para realizarla y se ha descrito el desarrollo de la misma. En este capítulo se presentarán las conclusiones referentes al desarrollo de la aplicación.

En las figuras 5.1 y 5.2 se representa el tiempo utilizado para la realización de este Trafago de Fin de Grado.

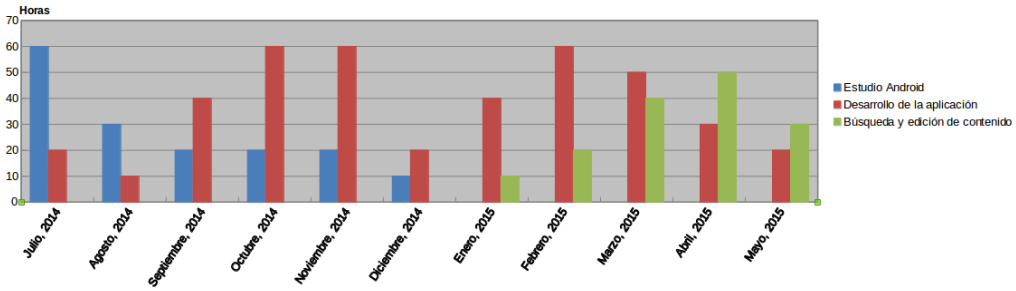

Figura 5.1. Horas por mes empleadas en las actividades necesarias para la realización del Trabajo de Fin de Grado.

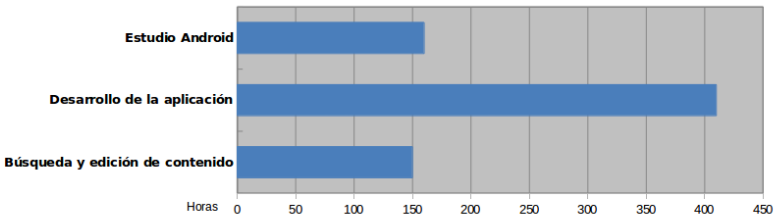

Figura 5.2. Horas totales empleadas en cada actividad necesarias para la realización del Trabajo de Fin de Grado.

5.1. Conclusiones

Tras presentar y realizar cambios acordados con las terapeutas del *Centro de Alzheimer Fundación Reina Sofía*, se ha cumplido el objetivo de desarrollar a una aplicación que cumple con todos los requerimientos para ser utilizada como herramienta de apoyo en terapias de estimulación cognitva en un centro con pacientes reales. Es una aplicación que cuenta con una serie de ejercicios específicos para estimular capacidades cognitivas concretas y que proporciona, tanto a los pacientes como a los terapeutas, una interfaz gráfica sencilla que facilita su uso. Los terapeutas tendran acceso a una sección de administración en la que podrán gestionar los parámetros de configuración de los juegos correspondientes a cada paciente. Además, podrán seguir y medir la evolución de los pacientes mediante la representación visual y gráfica de una tabla de valores estadísticos.

Como se ha mencionado en la sección 2.1, para desarrollar la aplicación debíamos cumplir con tres subobjetivos principales. En primer lugar, se recabaron los requisitos necesarios de la aplicación. Esto se logró mediante una serie de reuniones en las que las terapeutas exponían una serie de peticiones, dudas y cambios que fueron analizados para buscar soluciones satisfactorias y factibles a los problemas que se planteaban. En segundo lugar, se creó un diseño funcional y una base de datos en la que almacenar la información relevante de los pacientes, además de una serie de ejercicios específicos para la estimulación cognitiva. Para ello se utilizó el entorno de desarrollo Android Studio, Java como lenguaje de programación y SQLite como sistema de gestión de bases de datos. En último lugar se debía dotar a la aplicación de contenido, para lo que se buscaron imágenes y sonidos sin derechos de autor y que se ajustaran a las necesidades de cada juego.

Se han realizado pruebas, tanto internas como por las terapeutas, para comprobar que el funcionamiento de la aplicación es el correcto y cuente con todos los elementos necesarios para ser una herramienta útil y funcional.

5.2. Líneas futuras

A pesar de que se han realizado pruebas para comprobar que la aplicación funciona correctamente, queda pendiente la realización de una prueba piloto de seis meses de duración, en colaborción con las terapeutas del *Centro de Alzheimer Fundación Reina Sofía*, que servirá para evaluar la utilidad real de la aplicación. Se utilizarán tabletas que tengan instaladas la aplicación en sesiones de terapia reales con pacientes que sufren algún grado de deteriorio cognitivo.

Aunque los cambios en la aplicación realmente relevantes, se desarrollarán a medida que avance la prueba piloto, existen líneas de desarrollo que se podrían

realizar para mejorar la aplicación, como por ejemplo:

- **Agregar más ejercicios** de estimulación cognitiva a la aplicación.
- **Centralizar la informacion** mediante la creación de una base de datos centralizada que permita el acceso mediante un servicio web.
- **Portar la aplicación a otros sistemas operativos.** Por ejemplo IOS o Windows Phone.

Apéndices

Apéndice A

Manual de usuario BGames

BGames es una aplicación Android que pretende servir como herramienta de apoyo en terapias de estimulación cognitiva. En este manual describiremos la funcionalidad de la aplicación.

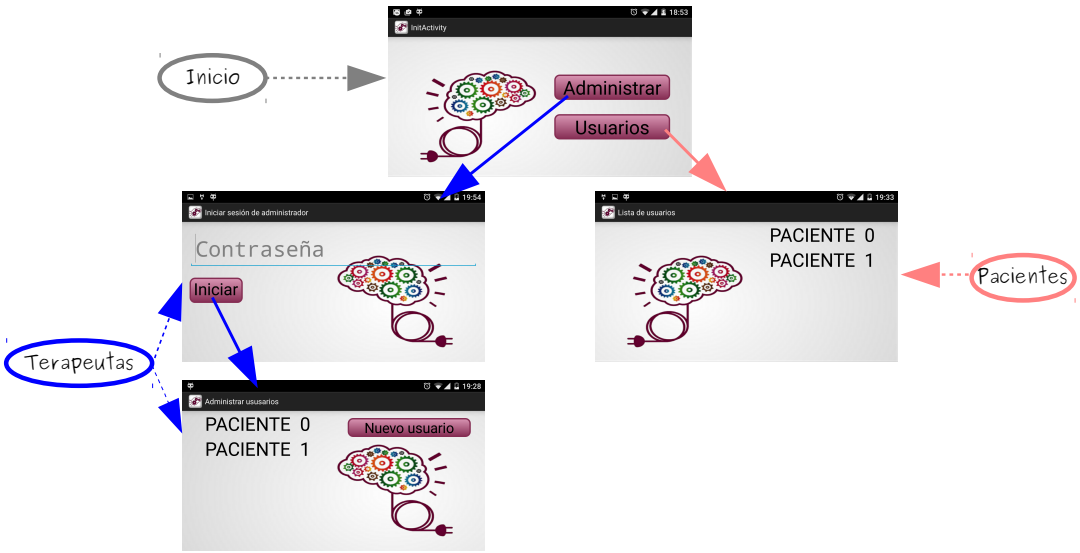

Figura A.1. Pantallas iniciales de la aplicación.

Como se puede observar en la figura A.1, la pantalla inicial de la aplicación muestra dos botones, uno para acceder a la aplicación como terapeuta y el otro para acceder como paciente.

A.1. Acceso como terapeuta

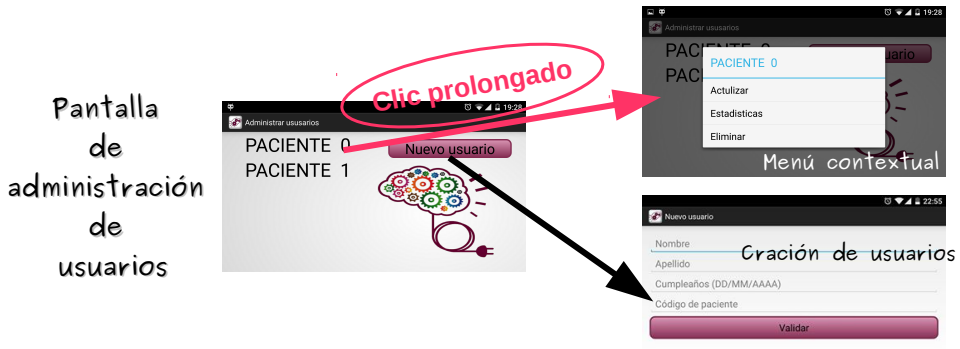

Figura A.2. Pantalla de administración de usuarios.

Para acceder como terapeuta se debe hacer clic en el botón "Administrador" de la pantalla inicial. A continuación, se presentará una pantalla de inicio de sesión en la que se tendrá que proporcionar una contraseña. Si se introduce una contraseña correcta se accederá a la pantalla de administracion de usuarios que se muestra en la figura A.2.

Desde la pantalla de administracion de usuarios, podremos crear, eliminar, gestionar los datos y ver las estadísticas de los pacientes. Si pulsamos durante más de un segundo sobre el nombre de un paciente nos aparecerá un menú contextual con tres opciones que se puede observar en la figura A.2.

- Si hacemos clic en el botón "Nuevo usuario" nos llevará a la pantalla de creación de usuarios, en la que tendremos que introducir los datos del paciente y guardarlos pulsando el botón "Validar".
- Si hacemos clic sobre el item "Eliminar" del menú contextual se eliminará el usuario.
- Si hacemos clic sobre el item "Actualizar" del menú contextual se accederá a una pantalla en la que se muestran las categorías de la aplicación. Si hacemos clic en alguna categoría accederemos a las pestañas de gestión de los parámetros de configuración, en las que podremos editar los valores y guardarlos haciendo clic en el botón "Validar". Si deslizamos el dedo de izquierda derecha o de derecha a izquierda cambiaremos de pestaña. Se muestra un ejemplo en la figura A.3

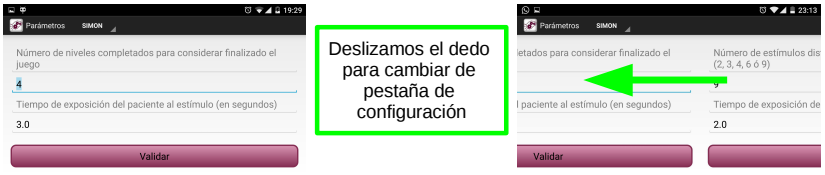

Figura A.3. Pantalla de gestión de parámetros de confuración de los juegos para cada usuario.

- Si hacemos clic sobre el item "Estadísticas" del menú contextual se accederá a una pantalla en la que se muestran las categorías. Si hacemos clic en alguna categoría accederemos a las pestañas donde se muestran los datos estadísticos que almacena cada juego para el usuario. Como se muestra en la figura A.4, se puede filtrar por fechas. Primero tendremos que pulsar por más de un segundo sobre la fecha que deseemos cambiar y aparecerá un panel para cambiar la fecha. Cuando las fechas sean las deseadas se pulsará sobre el botón "Validar". Para acceder a la las gráfcias se pulzará sobre el botón "Gráficas". Se puede observar un ejemplo en la figura A.4

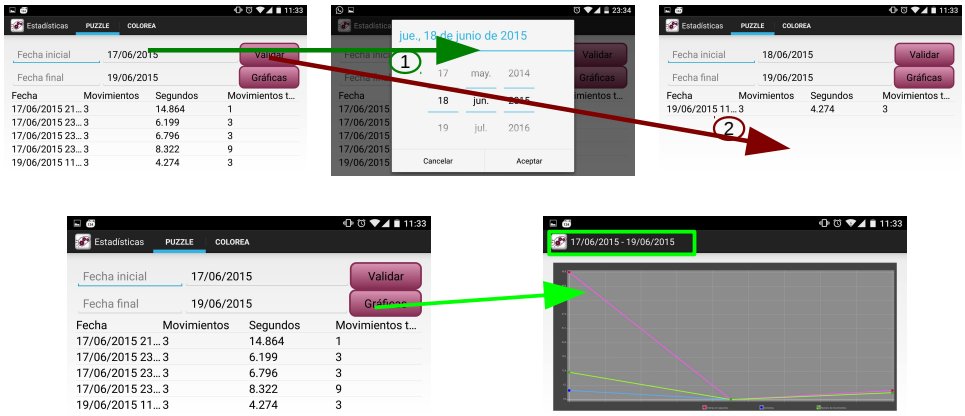

Figura A.4. Pantallas de visualización de datos estadísticos del usuario.

A.2. Acceso como paciente

Figura A.5. Pantalla de acceso como paciente.

Para acceder como paciente se debe hacer clic en el botón "Usuarios" de la pantalla inicial. Como se muestra en la figura A.5, se presentará una lista con todos los usuarios almacenados en la aplicación. Pulsamos sobre un usuario y accedemos a una pantalla que muestra las categorías, hacemos clic sobre una categoría y accedemos a una pantalla que muestra la lista de juegos que se corresponden con la categoría. A continuación describiremos los juegos a los que accedemos desde cada categoría.

Atención

Simón: En la primera pantalla debemos pulsar el botón "Empieza" para que se presionen automáticamente una secuencia de botones. En la siguiente

pantalla el usuario debe pulsar los botones en el orden anteriormente mostrado. Ejemplo de las pantallas del juego en la figura A.6.

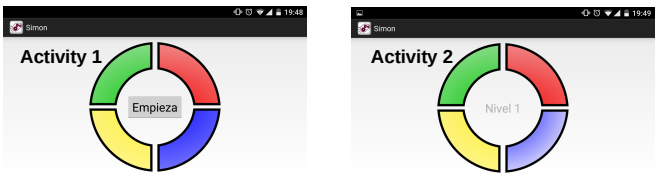

Figura A.6. Pantallas del juego Simón.

Atención nivel 1 (*target* presente): Como se muestra en la figura A.7, se presenta la letra objetivo en un lado de la pantalla y los distractores en el otro, el usuario debe presionar sobre la letra que se corresponda con el *target*.

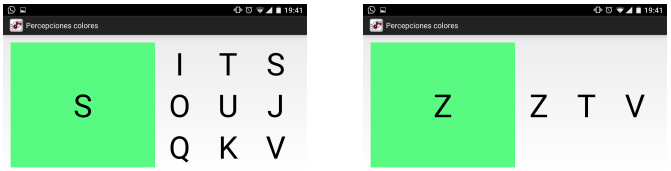

Figura A.7. Pantallas del juego Atención nivel 1 (*target* presente).

Atención nivel 2 (*target* previo): Como se muestra en la figura A.8, primero se presenta la letra objetivo durante un tiempo predeterminado y a continuación se muestran los distractores en la misma pantalla. El usuario debe presionar sobre la letra que se corresponda con el *target*.

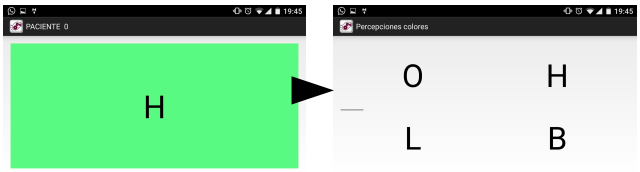

Figura A.8. Pantallas del juego Atención nivel 2 (*target* previo).

Atención nivel 3 (secuencial) Como se muestra en la figura A.9, primero se presenta la letra objetivo durante un tiempo predeterinado y a continuación se muestran los distractores uno a uno. El usuario debe presionar sobre la letra que se corresponda con el *target*.

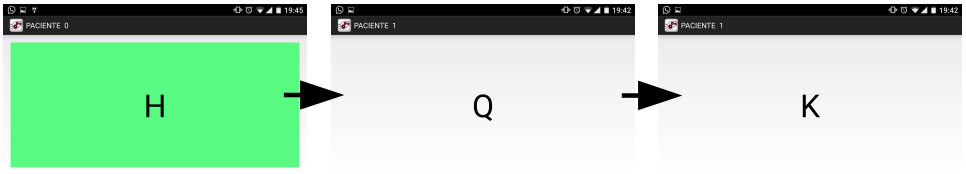

Figura A.9. Pantallas del juego Atención nivel 3 (secuencial).

Gnosias

Sonidos: El usuario debe pulsar sobre la imagen que se corresponda con el sonido del animal que se presenta. Ejemplo de las pantallas del juego en la figura A.10.

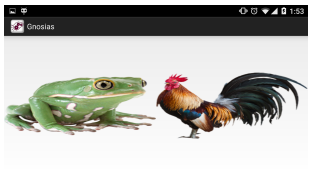

Figura A.10. Pantallas del juego Sonidos.

Colores: El usuario debe pulsar sobre la imagen que se corresponda con el color que se le pide. Ejemplo de las pantallas del juego en la figura A.11.

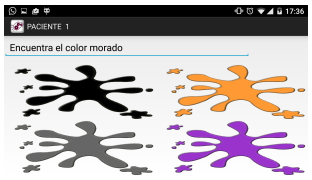

Figura A.11. Pantallas del juego Colores.

Siluetas: El usuario debe pulsar sobre la imagen que se corresponda con la silueta que se muestra en la parte izquierda de la pantalla. Ejemplo de las pantallas del juego en la figura A.12.

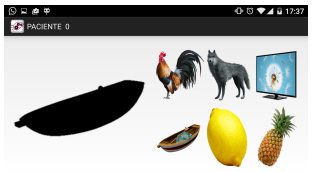

Figura A.12. Pantallas del juego Siluetas.

Esquema corporal

Partes del cuerpo: Se muestra la imagen de un cuerpo humano y el usuario debe tocar las partes que se le piden.

Partes de la cara: Se muestra la imagen de un rostro humano y el usuario debe tocar las partes que se le piden.

Se muestran ejemplos de ambos juegos en la figura A.13.

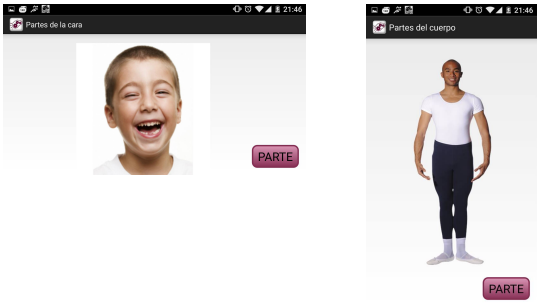

Figura A.13. Pantallas de los juegos de esquema corporal.

Razonamiento

Sumas: Como se puede ver en la figura A.14 se presenta visual y auditivamente una suma y el usuario debe pulsar sobre la respuesta correcta.

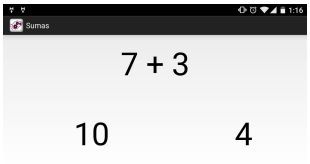

Figura A.14. Pantallas del juego Sumas.

Restas: Como se puede ver en la figura A.15 se presenta visual y auditivamente una resta y el usuario debe pulsar sobre la respuesta correcta.

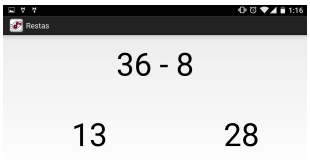

Figura A.15. Pantallas del juego Restas.

Multiplicaciones: Como se puede ver en la figura A.16 se presenta visual y auditivamente una multiplicación y el usuario debe pulsar sobre la respuesta correcta.

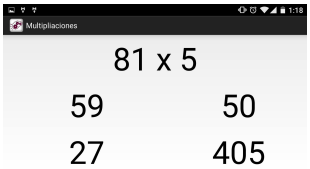

Figura A.16. Pantallas del juego Multiplicaciones.

Compara valores: Como se puede ver en la figura A.17 se presentan dos valores y se pregunta por el mayor o menor, el usuario debe seleccionar la respuesta correcta.

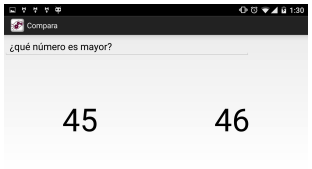

Figura A.17. Pantallas del juego Compara valores.

Compara precios: Como se puede ver en la figura A.18 se presentan dos imágenes y se pregunta por el que cuesta más o menos, el usuario debe seleccionar la respuesta correcta.

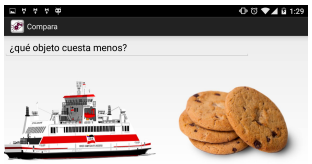

Figura A.18. Pantallas del juego Compara precios.

Funciones ejecutivas

Categorización: Como se puede ver en la figura A.19 se presentan varias imágenes y se pregunta por los elementos de una categoría, el usuario debe seleccionar todos las respuesta correcta.

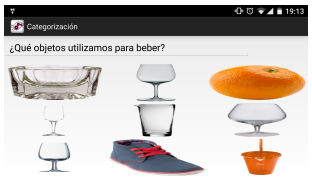

Figura A.19. Pantallas del juego Categorización.

Secuenciación: Como se puede ver en la figura A.20 se presentan dos o tres imágenes de la ejecución de un acción y el usuario debe seleccionar todas las acciones en el orden en el que se deben realizar.

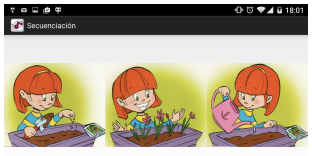

Figura A.20. Pantallas del juego Secuenciación.

Puzzles

Puzzle de piezas cuadradas: Como se puede observar en la figura A.21 se presentan ocho piezas que se corresponden con una imagen principal. El

usuario debe desplazar las piezas en sentido vertical u horizontal ocupando el espacio vacío, para así poder ordenar el puzzle.

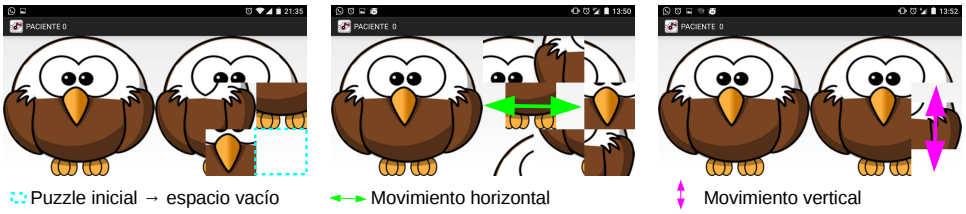

Figura A.21. Pantallas del juego Puzzle de piezas cuadradas.

Colorea: Como se puede observar en la figura A.22 se presenta a la izquierda de la pantalla una imagen y a la derecha las piezas que la conforman. El usuario debe arrastrar las piezas hasta la posición correcta dentro de la imagen completa.

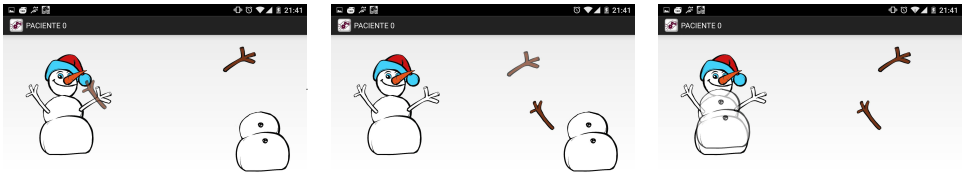

Figura A.22. Pantallas del juego Colorea.

Reconocer emociones

Reconocer emociones: Como se puede observar en la figura A.23 se presentan dos o tres fotos de una persona representando distintas emociones. El usuario debe señalar la imagen que se corresponda con la emoción por la que se le pregunta.

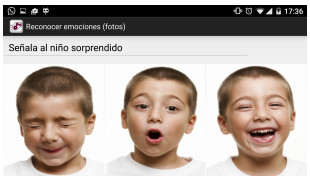

Figura A.23. Pantallas del juego Reconocer emociones.

Bibliografía

[1] Centro de Alzheimer Fundación Reina Sofía https://www.fundacioncien.es

[2] (2012) Envejecimiento en el siglo XXI. Una celebración y un desafío. [Online]. Available: http://www.unfpa.org/sites/default/files/pub-pdf/AgeingReportExecutiveSummarySPANISHFinal_0.pdf

[3] Enfermedades neurodegenerativas http://www.neurorhb.com/enfermedades-neurodegenerativas.html

[4] Alzheimer http://knowalzheimer.com/todo-sobre-el-alzheimer

[5] Alzheimer http://www.dmedicina.com/enfermedades/neurologicas/alzheimer.html

[6] Alzheimer http://www.nlm.nih.gov/medlineplus/spanish/ency/article/000760.htm

[7] Funciones cognitivas https://www.neuronup.com/es/areas/functions

[8] Alzheimer http://www.ncbi.nlm.nih.gov/pubmed/22573599

[9] (2011) Ana Rey Cao, Inma Canales Lacruz y María Inés Táboas Pais, Calidad de vida percibida por las personas mayores. Consecuencias de un programa de estimulación cognitiva a través de la motricidad «Memoria en movimiento» http://www.sciencedirect.com/science/article/pii/S0211139X11000424

[10] Blog AFA Parla - Terapia asistida con animales. http://afaparla.blogspot.com.es/p/terapia-perros.html

[11] Terapia con perros para el tratamiento de la enfermedad de Alzheimer http://cuidadoalzheimer.com/terapias-alternativas-alzheimer/terapia-con-perros-para-el-tratamiento-de-la-enfermedad-de-alzheimer

[12] Roboterapia en Demencia, Ministerio de Sanidad, Política Social e Igualdad, Instituto de Mayores y Servicios Sociales (IMSERSO). [Online]. Available: http://www.imserso.es/InterPresent1/groups/imserso/documents/binario/231_11idi.pdf

[13] (2012) José María Cañas y Francisco Martín, Roboterapia en la recuperación de Alzheimer. [Online]. Available: http://gsyc.es/jmplaza/papers/vodafone-2012.pdf

[14] Robot foca Paro - Nuka http://www.adelerobots.com/es/nuka

[15] Robot perro Aibo http://www.therobotmuseum.eu/perros-sony-aibo

[16] Robot humanoide Nao http://www.turobot.es/robot-nao-humano

[17] Proyecto "Música para despertar" http://www.musicaparadespertar.com/

[18] Canal de youtube del proyecto "Música para despertar" https://www.youtube.com/channel/UCKge2KrICcXTGTONQSKl0yg

[19] (2009) INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA. [Online]. Available: https://www.incibe.es/file/N85W1ZWFHifRgUc_oY8_Xg

[20] (2014) Antonio Abellán García, Juan Vilches Fuentes y Rogelio Pujol Rodriguez, Un perfil de las personas mayores en España, 2014. Indicadores estadísticos básicos. [Online]. Available: http://envejecimiento.csic.es/documentos/documentos/enred-indicadoresbasicos14.pdf

[21] Aplicación Android *Medidafe* http://www.medisafe.com

[22] Aplicación Android *Mimov*

[23] Aplicación Android *App de Dependencia* http://www.dependencia.imserso.es/dependencia_01/documentacion/app/index.htm

[24] Aplicación Android *Juegos mentales* https://play.google.com/store/apps/details?id=com.thepapership.braingames.espanol

[25] Aplicación Android *4 Fotos 1 Palabra* https://play.google.com/store/apps/details?id=app.fotopalabra

[26] Aplicación Android *Memory* https://play.google.com/store/apps/details?id=com.berniiiiiiii.logomatchup

[27] Aplicación Android *Palabro* Palabro-Juegodepalabras

[28] Aplicación Android *Clasic Words* https://play.google.com/store/apps/details?id=com.lulo.scrabble.classicwords

[29] Aplicación Android *Brain Trainer Special* https://play.google.com/store/apps/details?id=brain.trainer

[30] Mediawiki del proyecto `http://jderobot.org/Evelinfv`

[31] Netmarketshared, servicio de estadísticas de uso de sistemas operativos y navegadores `http://www.netmarketshare.com`

[32] Página web para desarrolladores de Android - Android Studio `http://developer.android.com/tools/studio/index.html`

[33] Página web para desarrolladores de Android - Activity `http://developer.android.com/reference/android/app/Activity.html`

[34] Página web JetBrains `https://www.jetbrains.com/idea`

[35] Página web de Java `www.java.com`

[36] Página web SQLite `https://www.sqlite.org`

[37] Herramienta online de edición de pistas de sonido. `https://twistedwave.com/online/`

[38] Página web desarrollador ankri - AutoScaleTextView `http://ankri.de/autoscale-textview`

[39] Página web descarga AndroidPlot-Core `http://androidplot.com/download`