ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Aplicación Android para interacción musical con distintos tipos de materiales

Titulación: Grado en Ingeniería en Informática

Mención: Ingeniería del software

Autor: David Suárez Suárez

Tutores: Álvaro Suárez Sarmiento y Alexis Quesada Arencibia

Fecha: julio de 2018

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/Dª David Suárez Suárez, autor del Trabajo de Fin de Título Aplicación Android para interacción musical con distintos tipos de materiales,
correspondiente a la titulación Grado en Ingeniería Informáctica (GII)
en colaboración con la empresa/proyecto (indicar en su caso) _____

S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[X] No está previsto.

Y para que así conste firma la presente.

Las Palmas de Gran Canaria, a 13 de julio de 2018.

El estudiante

SUAREZ SUAREZ DAVID - 42248448R
Firmado digitalmente por SUAREZ SUAREZ DAVID - 42248448R
Fecha: 2018.07.13 21:24:02 +01'00'

Fdo.: _____

A rellenar y firmar **obligatoriamente** por el/los tutor/es

En relación a la presente solicitud, se informa:

[X] Positivamente

QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)
Firmado digitalmente por QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)
Nombre de reconocimiento (DN): c=ES, serialNumber=42870168P, sn=QUESADA, givenName=FRANCISCO ALEXIS, cn=QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)
Fecha: 2018.07.13 18:13:18 +01'00'

Fdo.: _____

[ ] Negativamente
(la justificación en caso de informe negativo deberá incluirse en el TFT05)

SUAREZ SARMIENTO ALVARO - 43653376C
Firmado digitalmente por SUAREZ SARMIENTO ALVARO - 43653376C
Nombre de reconocimiento (DN): c=ES, serialNumber=43653376C, sn=SUAREZ SARMIENTO, givenName=ALVARO, cn=SUAREZ SARMIENTO ALVARO - 43653376C
Fecha: 2018.07.13 21:09:24 +01'00'

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Índice de contenido

Índice de figuras

Agradecimientos

En la realización de este *Trabajo de Fin de Grado* (*TFG*), he tenido el apoyo de bastantes personas o entidades, que bien me han apoyado moralmente en el proyecto o bien me han facilitado muchos materiales o herramientas necesarias para la elaboración de este. Por ello, les doy las gracias.

A mi familia por el constante apoyo y por todos los consejos que me han podido aportar, dentro de su margen de actuación.

A mis tutores de TFG, Álvaro Suárez Sarmiento y Alexis Quesada Arencibia, que han estado apoyándome y ayudándome en todo lo posible, disipando todas mis dudas y abriéndome el camino, además de facilitarme todo el proceso de gestión y elaboración del TFG.

A todos los profesores de este Grado en Ingeniería Informática por sus horas invertidas en la docencia y en mi formación, sin las cuales no habría sido posible adquirir todos los conocimientos necesarios para la realización del TFG.

A Leroy Merlín y a Víctor Acuña Santana (amigo) por facilitarme el cartón necesario para este proyecto.

Resumen

Se presentan las dos versiones del resumen, español e inglés respectivamente:

- Independientemente de sus características morfológicas y fisiológicas, la evolución le ha aportado al ser humano la capacidad de razonar a niveles inimaginables y de llevarlos a un ámbito aún más misterioso, las sensaciones.

  En este TFG se plantea una aplicación que permita manifestar y analizar la clara relación entre los sentidos del tacto y del oído. Las sensaciones que experimentamos al tocar un determinado material y al escuchar un determinado sonido (musical o no) no son incompatibles. Mediante un soporte y una aplicación móvil, se pretende obtener sonidos que expresen las sensaciones experimentadas cuando se tocan distintos materiales o texturas.

- Independent of its morphological and physiological characteristics, evolution has provided to humanity the ability to reason at unimaginable levels and to experience sensations, something even more fascinating than the first.

  In this Final Degree Project, an application to manifest and analyse the clear relationship between the senses of touch and hearing is suggested. When someone touches a material or listens to a certain sound (musical or not), experiences not incompatible sensations. Through a support and a mobile application, it is expected to get sounds which express the experienced sensations when different materials or textures are touched.

1. Introducción

En este capítulo presentamos el marco general en el que se enmarca el TFG, los objetivos y la estructura de la memoria.

1.1 Contexto

Como bien sabemos, en el planeta que conocemos y habitamos existen diferentes tipos de seres vivos (reino animal, reino vegetal, reino monera, reino protista y reino de hongos). Nos centramos dentro del reino animal para estudiar al ser humano, un ser vivo particularmente único y característico.

Concentrándonos en los seres humanos, hemos de destacar una característica vital muy importante: los sentidos. Disponemos de cinco sentidos, todos y cada uno de los cuales juegan un papel fundamental en la supervivencia: Vista, Olfato, Gusto, Tacto y Oído. Estos cinco sentidos se relacionan y se coordinan entre sí de una manera instintiva y natural para el beneficio propio (o conjunto) de la especie [1].

Es cierto que en la mayoría de las ocasiones los sentidos simplemente ayudan a los seres vivos a obtener datos del mundo real para actuar según su conveniencia, con el objetivo de sobrevivir y obtener algún tipo de beneficio. Sin embargo, el ser humano es diferente en este aspecto. Además de ayudarse de los sentidos para satisfacer los objetivos anteriormente comentados, es capaz de analizar y estudiar gran parte de los datos sensoriales para llevarlo no solo al terreno racional, sino al terreno emocional y sentimental. Esto tiene como resultado un único concepto: experimentación de sensaciones. Un ejemplo de esto es la capacidad de revivir distintos recuerdos emotivos del pasado tan solo experimentando olores, sabores, sonidos… [2].

Es aquí donde se centra este TFG. Básicamente, toda la motivación que empuja el proyecto gira en torno a desarrollar una aplicación que nos permita estudiar la relación existente entre el sentido del tacto y el sentido del oído.

Todas estas funcionalidades contribuyen en varios aspectos. Por una parte, proporcionan un medio para reflejar acciones táctiles en sonidos musicales, por otra parte, proporcionan un medio destinado principalmente a niños de corta edad para que asimilen las relaciones entre los sentidos del tacto y del oído. Con este proyecto se pueden extraer muchos comportamientos intuitivos en niños al usar nuestra aplicación/juego, constituyendo una base para el estudio relacionado con el tacto y la música. Si se lleva un poco más allá, podría extrapolarse todo a la rama de la psicología. Claro está que, para esto, sería necesario disponer de psicólogos expertos, pues no es algo que cualquiera pueda analizar con tal nivel de detalle.

1.2 Objetivos

El objetivo principal de este TFG es que una persona, escuche un sonido particular al tocar o sentir ligeramente el tacto de un determinado material, permitiéndole de esta manera asociar una sensación musical a una sensación de tacto particular. Con ello se pretende iniciar un estudio de posibilidades de las aplicaciones de las Tecnologías de la Información y las Comunicaciones (TIC) al conocimiento de la música por parte del público infantil y juvenil (aunque no estaría cerrado a esas edades).

Para cubrir este objetivo principal, se ha desarrollado la aplicación móvil *Tactile Sounds*, en exclusiva para Android. *Tactile Sounds* es una aplicación que, junto a un soporte diseñado y adaptado para su correcto funcionamiento, es capaz de detectar la interacción del usuario con diferentes tipos de materiales (mediante la cámara del dispositivo móvil) y reproducir un sonido concreto para el material que estamos tocando.

Como objetivos secundarios, se pretenden cumplir los mostrados a continuación:

- Disponiendo de una amplia selección de materiales con distintas propiedades (diferentes durezas, texturas...) se debe de emitir un sonido distinto (y adaptado) al tocar cada uno de ellos (interacción táctil).

- El sonido emitido debe corresponderse adecuadamente con la sensación real experimentada al tocarlo, es decir, si tocamos un material cuya sensación es fría y triste debe emitirse un sonido que, al escucharse, exprese también dichas sensaciones. De este modo, cada sonido musical (básicamente acordes musicales) tiene un color distinto adaptado al material que estemos tocando, el cual debe de ser estrictamente fiel a la sensación experimentada.

1.3 Estructura de la memoria

Una vez introducido el TFG, pasamos a detallar las diferentes partes que componen este documento.

En el capítulo 2 se introduce el estado del arte (ámbito del TFG), incluyendo información actual de relevancia para el proyecto, que ayuden a ubicarlo o a optimizar su comprensión en muchos aspectos: Psicología, Música y Papiroflexia.

En el capítulo 3 se incluye información sobre todas las herramientas (aplicaciones) utilizadas para poder llevar a cabo las diferentes partes del proyecto, pasando por tópicos como la programación o la edición de imágenes, entre otros.

En el capítulo 4 se incluye la información directamente relacionada con la planificación y ejecución del proyecto. La idea es presentar una descripción multinivel del software de la aplicación desarrollada: funcional, organizativa y de implementación.

En el capítulo 5 se cubre la explicación de resultados diversos al probar algunas características de la aplicación, tales como: Velocidad de respuesta, Consumo de batería y Estimación de tiempo de juego.

En el capítulo 6 se presentan las conclusiones y posibles ampliaciones de la aplicación desarrollada.

Por último, se presentan los siguientes anexos: Soporte, Manual de usuarios, Manual de instalación, Pliego de condiciones, Especificación de casos de uso, Competencias, Normativa y legislación y Presupuesto.

2. Estado del arte

En este capítulo presentamos el ámbito del TFG, el estado del arte en temas relacionados y un estudio de qué aplicaciones similares existen actualmente.

2.1 Psicología

La psicología está relacionada con el estudio del funcionamiento del órgano más representativo del ser humano, nuestro gran cerebro, y sus implicaciones personales, sociales y grupales, en general [3]. La psicología concretamente se encarga de estudiar su comportamiento en base a los datos que recibe del mundo exterior (e interior, personal), es decir, en base a la percepción y emociones/sentimientos/sensaciones del ser humano (modelo funcional).

Es evidente que todo lo que percibimos y sentimos pasa por este órgano. Por lo tanto, la psicología juega un papel muy importante en este TFG, pues tratamos de buscar la relación entre el tacto de distintos materiales y la escucha de sonidos musicales concretos. Naturalmente, el punto común y de unión de estos dos acontecimientos se da en algún lugar del cerebro.

Presentemos primero la psicología centrada en el tacto. Muchos se atreven a decir que el tacto es el sentido más importante, cosa que no es un disparate porque es un sentido presente prácticamente en cualquier forma de vida [4]. Quizás no sea el más usado por el ser humano hoy en día (conscientemente), pero no podemos negar que es uno de los más relevantes, de hecho, para un ciego, por ejemplo, es un sentido vital. Además, es la principal forma de interactuar con otros seres humanos (sin ser con el lenguaje verbal), pudiendo transmitir sentimientos hacía otros, información no verbal e incluso violencia (desgraciadamente).

Centrándonos en el tacto entre personas, muchos estudios han demostrado que, por ejemplo, el tacto influye mucho en el desarrollo de un bebé. El hecho de dormir a un niño en brazos o de transmitirles cariño a través del tacto son acciones que marcan subliminalmente su desarrollo. Esto concretamente puede influir en temas como la ansiedad, la sensación de protección, la seguridad, entre otros [5].

Esto no es solo en bebés, pues también se han realizado experimentos sociales en grupos de personas que consisten en transmitir diferentes emociones mediante el tacto a personas desconocidas. El que era tocado no veía ni conocía nada de la otra persona y disponía de una lista de emociones para adivinar las emociones transmitidas. Muchas de esas personas acertaban la emoción que la otra persona le estaba transmitiendo. Sorprendentemente se puede transmitir alegría, tristeza, depresión, nervios, miedo, entre otros [6].

Al igual que se transmite información y sensaciones a través del tacto con otros seres humanos (u otras formas de vida), se obtiene del entorno que nos rodea. El tacto de distintos tipos de materiales no es parte excluyente de esto último.

En segundo lugar, tenemos la psicología asociada al oído y a la música. Si lo comentado sobre el tacto parece asombroso, el mundo de la psicología asociado a la música es inmenso. Decíamos que no podíamos vivir sin tacto, pero el oído también es un sentido fundamental.

Tanto el lenguaje verbal entre personas como la emisión/recepción de sonidos musicales (o no musicales) son imprescindibles. Mucha gente no conoce el mundo sin música ni sin el *habla humana*, de hecho, son dos candidatos a posibles protagonistas en cuanto a psicología se trata. Solo

hace falta recapacitar en una cosa… cuando vamos al psicólogo por problemas personales ¿cuál es su principal herramienta? Efectivamente no es nada tangible, es la palabra. Al igual que la palabra, muchas personas tienen un psicólogo gratis y particular llamado música. La música es capaz de activar, de acuerdo con muchos estudios con base en monitorizaciones de impulsos nerviosos en el cerebro, gran cantidad de regiones de este. De hecho, puede tocar regiones del cerebro a las que el Alzheimer no afecta, de acuerdo con el artículo publicado en *Las Provincias* en abril de este año (2018) [7].

Si nos preguntamos el por qué la música toca tantas áreas del cerebro, llegamos a una conclusión: la música no son sonidos, son datos, emociones y sensaciones que se transmiten al oyente.

Como podemos observar, tanto el tacto como el oído son dos afluentes que desembocan en un río, la psicología. Este TFG trata de exteriorizar dicha relación.

2.2 Música

El mundo de los sonidos, particularmente el mundo de la música es gratificantemente inmenso. En el contexto de nuestro TFG y de nuestro grado, no solo tiene que ver con uno de los sentidos del ser humano (sentido del oído), sino que está impulsado en su mayor parte por las Matemáticas, otro mundo enorme y abierto de posibilidades [8].

A pesar de esto, en gran parte, el mundo de la música es inexplicable, ya que, por ejemplo, una determinada canción puede producir diferentes sensaciones, emociones o pensamientos según la persona que la escuche. Si vamos más allá, puede producir efectos distintos a una misma persona en diferentes etapas de su vida. Aquí entra en juego el estado y particularidades de la persona (estado sentimental, gustos, país de origen, …) [9].

En lo que deberíamos de estar todos de acuerdo es que, la música, al igual que las matemáticas, son universales, ya que van más allá del mundo empírico y del idioma, cubriendo la parte *inexplicable* que hace particular al ser humano [10]. Es eso lo que tratamos de reflejar en nuestro trabajo, la relación de sensaciones similares/compatibles al tocar materiales y al escuchar determinados sonidos musicales.

Para meternos en contexto, explicamos varios conceptos básicos sobre la música.

2.2.1 Pulso musical

Entendemos por pulso musical a una cantidad de tiempo mínima en la que se puede dividir la duración de una determinada pieza musical. Para entender mejor este concepto, disponemos de un ejemplo clarísimo: el reloj.

Partiendo de la Figura 2.1, afirmamos que cada *tictac* del reloj marca un pulso musical concreto. Si disponemos de un reloj analógico cualquiera y escuchamos el sonido de su aguja (*tictac*) comprendemos que, entre un *tic* y un *tac* del reloj, existe exactamente la misma cantidad de tiempo. En otras palabras, el sonido del reloj emite sonidos de manera constante y con la misma distancia entre todos y cada uno de los sonidos. Eso es el pulso musical.

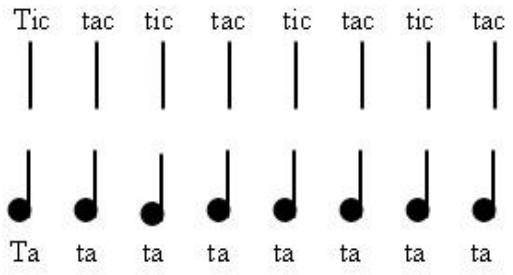

Figura 2.1: Sonidos del reloj marcando un pulso musical concreto.
Fuente: http://teoriamusi.blogspot.com/p/ritmo.html

Configurar adecuadamente un pulso musical es clave puesto que ese pulso es la base del ritmo musical que queramos imprimir a la pieza [11].

De forma aproximada, podemos considerar el pulso musical, como un reloj síncrono que marca la ausencia de uno o varios *eventos musicales* que ocurrirían simultáneamente [12]: sonido o silencio.

2.2.2 Figura musical

Una Figura musical es un símbolo que representa la duración en el tiempo de una determinada nota musical [13] , que podemos entender de manera general como un sonido característico que el oído humano es capaz de captar. También se puede usar una Figura musical para representar una duración determinada de ausencia de sonido (silencio).

Existen distintos tipos de figuras musicales, las cuales se enumeran a continuación: redonda, blanca, negra, corchea, semicorchea, fusa, semifusa. La redonda es la Figura que más duración tiene. El resto de las figuras duran la mitad que la anterior según el orden en que se enumeraron antes. Una redonda equivale a dos blancas y una blanca equivale a dos negras. De esta forma y por deducción, una redonda equivale a cuatro negras. Así, con el resto de las figuras musicales. Nótese que por tanto la duración de un sonido o un silencio nunca es absoluta, sino relativa a la cantidad de *pulsos* que dure la Figura de mayor valor.

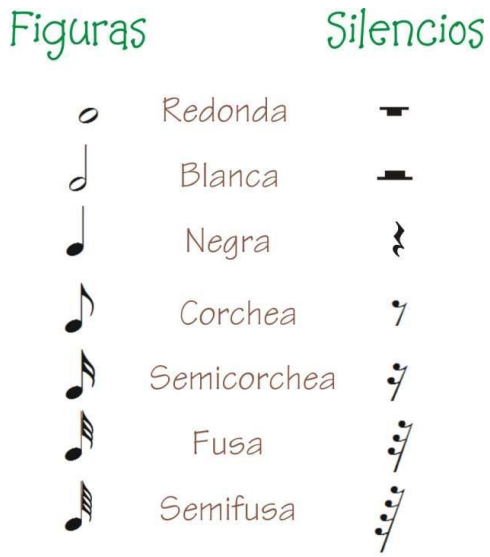

Figura 2.2: Figuras musicales y sus respectivos silencios.
Fuente: https://respuestas.tips/cules-son-las-figuras-musicales/

La Figura musical marca la cantidad de pulsos (número entero o real) que suena un sonido o existe un silencio, teniendo en cuenta que pueden sonar varios sonidos simultáneamente para conformar un evento musical complejo. En general, conformando adecuadamente figuras musicales de acuerdo con un pulso determinado se puede determinar la frecuencia con la que sonarían determinados sonidos y la presencia de silencios.

2.2.3 Nota musical

Una nota musical es un sonido predeterminado que suena como resultado del accionamiento de un instrumento musical afinado para tal tarea de forma adecuada [14]. Normalmente nos referimos a nota musical como la amplitud del sonido que se puede oír a una determinada frecuencia del espectro audible. Por ejemplo, a una frecuencia de 174,614 Hz se puede escuchar una nota que se identifica como *fa* en la música moderna occidental [15].

En la música disponemos de siete notas musicales: *do*, *re*, *mi*, *fa*, *sol*, *la* y *si*. Cada una de estas notas musicales se traducen, al fin y al cabo, en frecuencias (sonoras). En realidad, cada una de estas notas no equivalen a un único sonido, sino que, por el contrario, representan varios sonidos (fuertemente relacionados entre sí). En realidad, estas notas conforman una *octava musical*. Esto es, cuando se alcanza el sonido de la nota *si*, a continuación, vuelve a aparecer otra vez la nota *do*, pero una octava más aguda. Con el resto de las notas ocurre lo mismo (Figura 2.3). Una nota *do* tiene una determinada frecuencia. Justo el siguiente *do* en el pentagrama sería un *do* más agudo. Si nos vamos al *do* anterior ocurre totalmente lo contrario, sería un *do* más grave, ubicado a menos *altura* en el pentagrama y que tendría la mitad de frecuencia.

2.2.4 Distancias entre notas musicales

La distancia entre dos notas determinadas es, al fin y al cabo, una diferencia de frecuencia medible. En el mundo de la música, no se mide la distancia en frecuencias explícitamente, sino en lo que se conocen como tonos y semitonos. La distancia entre las notas en la escala de *do mayor* se expresa en la Figura 2.4. En esta figura, los espacios amarillos representan el medio tono adicional de diferencia entre las distintas notas. En el caso de *mi-fa* y *si-do* no existen porque entre ellas hay un solo semitono de diferencia.

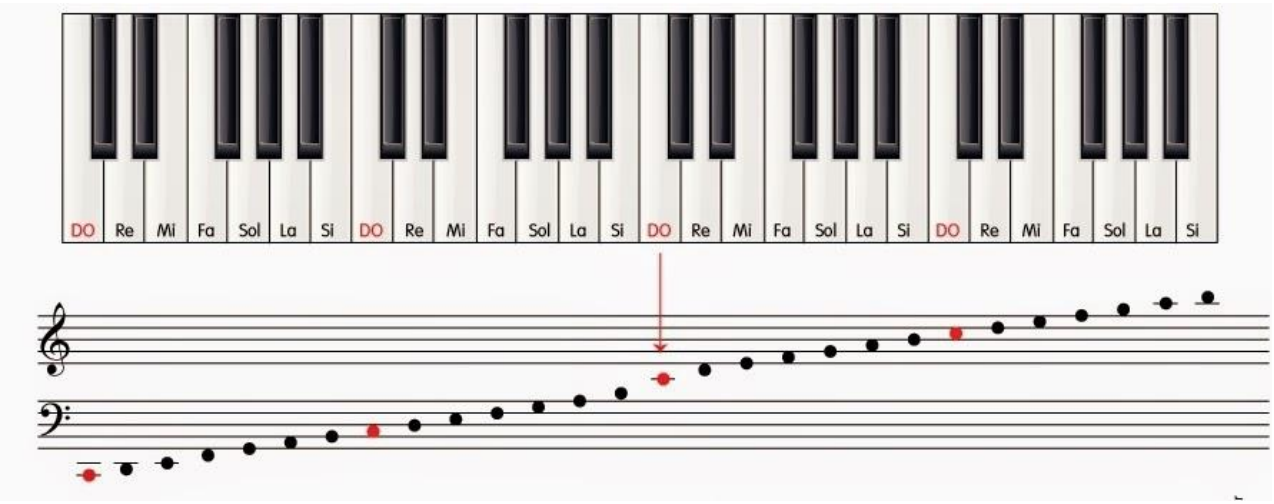

Figura 2.3: Ejemplo de octavas musicales.
Fuente: http://www.clasedelenguajemusical.com/2-1/

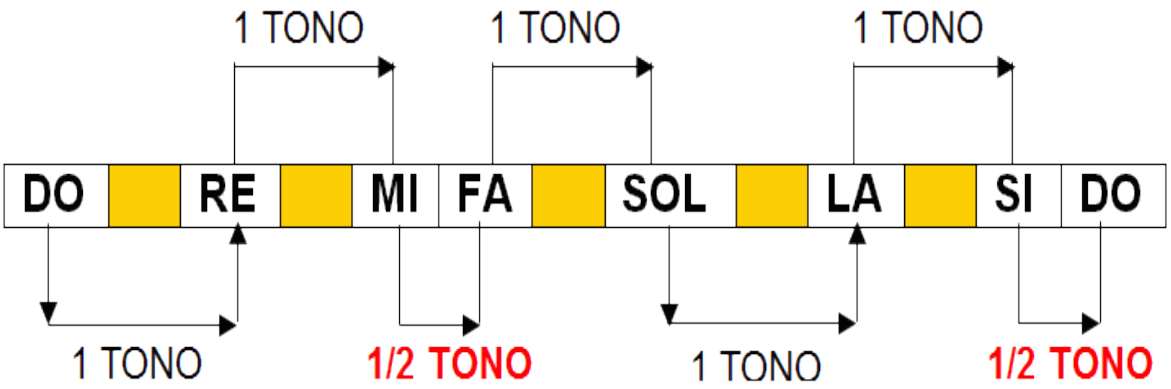

Figura 2.4: Distancia entre las notas de una octava musical.
Fuente: https://www.rockandrollparamunones.com/notas-guitarra/99-escala-mayor

A estas notas se les pueden añadir alteraciones. Las alteraciones son símbolos que afectan a las notas asociadas incrementando o disminuyendo semitonos (la frecuencia, al fin y al cabo). Las alteraciones existentes son:

- Becuadro (♮): fuerza a que una nota tenga la frecuencia natural (sin incrementos ni decrementos de ningún tipo).
- Sostenido (#): aumenta un semitono la actual nota.
- Doble sostenido (##): aumenta dos semitonos (1 tono) la actual nota.
- Bemol (b): Disminuye un semitono la actual nota.
- Doble Bemol (bb): disminuye dos semitonos (1 tono) la actual nota.

Un simple ejemplo práctico: Las notas *do* y *re* se llevan un tono de distancia. Si ponemos un sostenido al *do* (*do #*) tendmos una distancia de un semitono entre *do #* y *re*. Si, por el contrario, le ponemos un bemol al *re* (*re b*) tendmos también una distancia de un semitono entre *do* y *re b*. De esto podemos ir sospechando una cosa: efectivamente, *do #* y *re b* son la misma nota, correspondiente al espacio coloreado de amarillo en la Figura 2.4 (entre *do* y *re*).

Si le ponemos un doble sostenido al *do* (*do ##*) sería exactamente la nota *re*, al contrario que si ponemos un doble bemol al *re* (*re bb*) que sería exactamente la nota *do*. Teniendo en mente la Figura 2.4 no debería suponer un problema entender esto. No obstante, otra imagen que puede ser de ayuda es el piano, pues es un instrumento que refleja fielmente las notas, manifestando adecuadamente la distancia entre ellas (Figura 2.5).

Anteriormente se comentó que existían 7 notas musicales, pero como podemos intuir esto fue una simplificación para favorecer el aprendizaje progresivo. Con todo lo comentado hasta ahora podemos considerar que disponemos de 12 notas distintas, que son las reflejadas en cada tecla del piano de la Figura 2.5. En esta figura, podemos ver, tenemos teclas blancas y negras. Cada salto de una tecla a la siguiente (sea negra o blanca) representa un salto de un semitono. En el caso de *mi-fa* y *si-do* no apreciamos tecla negra en medio, ya que entre ellas existe un semitono.

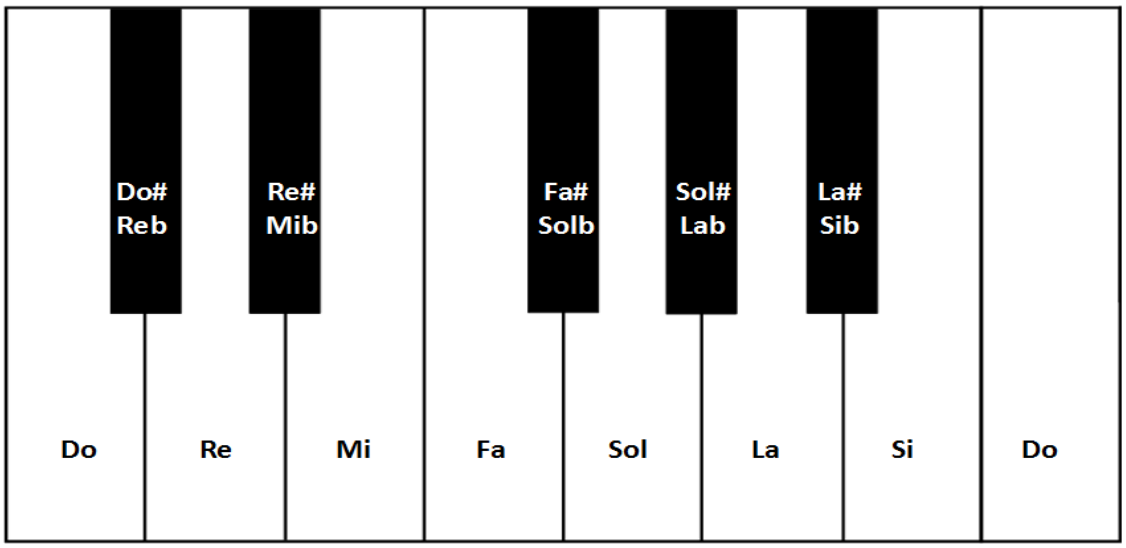

Figura 2.5: Teclas de una octava musical en el piano.
Fuente: https://www.escribircanciones.com.ar/teoria-musical/4443-teoria-musical4443-notas-musicales-html.html

En realidad, existen más notas musicales. En la cultura musical India o árabe los semitonos a su vez se pueden dividir a la mitad teniendo microtonos. Sin embargo, quedan fuera del ámbito de este TFG ese tipo de culturas musicales.

2.2.5 Intervalos musicales

Como dijimos antes, las notas (con o sin alteraciones) se distancian en un determinado número de tonos y semitonos. No obstante, cuando queremos expresar distancias entre notas se suele expresar de otra manera, esta es mediante los intervalos musicales.

Explicamos solamente lo más básico de los intervalos, ya que, por propia experiencia en lo que llevo de estudio musical, es un tema que se puede alargar innecesariamente. Tenemos intervalos de 2ª, 3ª, 4ª, 5ª, 6ª, 7ª y 8ª. Estos números lo único que indican, por resumirlo de algún modo, es el número de notas que hay entre dos notas concretas (sin tener en cuenta tonos y semitonos). Por ejemplo, entre Do y Re tenemos un intervalo de 2ª, entre Re y Fa tenemos un intervalo de 3ª, así sucesivamente. Un intervalo de 8ª es bastante peculiar, ya que es aquel que existe entre dos notas del mismo nombre (de un Do al siguiente/anterior Do, por ejemplo).

Sin entrar en mucho detalle, tenemos dos tipos de intervalos: *justos* y m*ayores/menores*. Los intervalos *justos* son los de 4ª, 5ª y 8ª, mientras que los *mayores/menores* son los de 2ª, 3ª, 6ª y 7ª. Tanto los intervalos justos como los *mayores/menores* pueden ser *aumentados* o *disminuidos*. Al ser *aumentados* se incrementa en un semitono la distancia entre las dos notas, al contrario de cuando son *disminuidos*, que se disminuye en un semitono. Visualmente tendríamos algo así:

- *Disminuidos* (-1 semitono) – *Justos* (+0 semitonos) – *Aumentados* (+1 semitono)
- *Disminuido* (-1 tono) – *menor* (-1 semitono) – *Mayor* (+0 semitonos) – *Aumentado* (+1 semitono)

Esto es quizás un poco difícil de asimilar, por lo que es más conveniente mostrar una tabla con todos los intervalos y sus equivalencias en tonos/semitonos (Figura 2.6).

Nombre del intervalo/Grados[1]	Distancia en tonos y semitonos
Unísono[2]	Mismo sonido
Segunda menor	1 semitono
Segunda mayor o tercera disminuida	1 tono
Tercera menor o segunda aumentada	1 1/2 tonos
Tercera mayor o cuarta disminuida	2 tonos
Cuarta justa o tercera aumentada	2 1/2 tonos
Cuarta aumentada o quinta disminuida (llamada *tritono*)[3]	3 tonos
Quinta justa o sexta disminuida	3 1/2 tonos
Sexta menor o quinta aumentada	4 tonos
Sexta mayor o séptima disminuida	4 1/2 tonos
Séptima menor o sexta aumentada	5 tonos
Séptima mayor	5 1/2 tonos
Octava justa	6 tonos

Figura 2.6: Equivalencia de intervalos musicales y número de tonos/semitonos de distancia.
Fuente: https://es.wikipedia.org/wiki/Intervalo_(m%C3%BAsica)

Una vez que hemos explicado los conceptos más básicos de música, afrontamos ahora otro concepto muy importante: la especificación escrita de la música mediante compases y pentagrama. Esto es necesario para poder entender de forma gráfica y sencilla el concepto de acorde que es muy importante en el TFG desarrollado.

2.2.6 Pentagrama

Un pentagrama es un gráfico que consta de conjunto de cinco líneas horizontales paralelas y cuatro espacios en blanco. En las líneas y espacios, se escriben las figuras musicales para indicar la duración de la nota musical concreta que viene dada en función del lugar que ocupa en el pentagrama. Cuando se quiere trabajar con más de dos octavas es necesario añadir líneas y espacios a ese conjunto básico de cinco líneas y cuatro espacios. Por ejemplo, considerando una nota cualquiera, por ejemplo, *sol*, afirmamos que existen varios sonidos para dicha nota, desde más graves a más agudos (Figura 2.7).

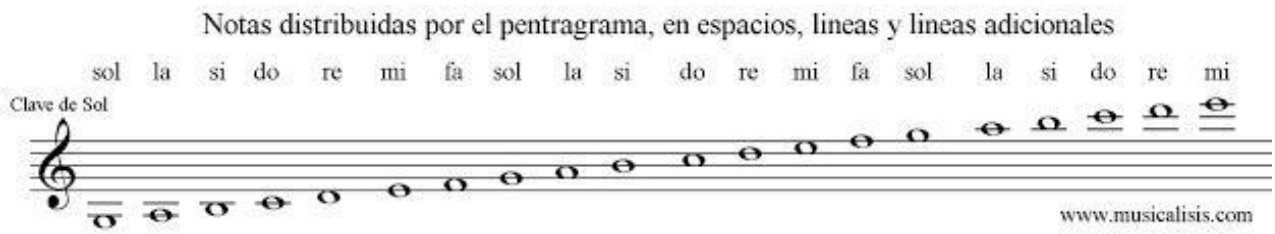

Figura 2.7: Notas musicales de sol a mi (en clave de sol).
Fuente: http://estudioyanalisismusical.blogspot.com/2011/09/aprender-la-base-del-solfeo-en-dos.html

El símbolo que aparece justo en el extremo izquierdo del pentagrama se denomina clave, y se utiliza básicamente para que, a la hora de leer o escribir las notas (componer), éstas se adapten al registro del instrumento afinado de una determinada forma. Por ejemplo, para una guitarra se utiliza *clave de sol* (justo la dibujada en la Figura 2.7) porque con dicha clave se cubren todas las notas del instrumento sin tener que desplazarnos excesivamente por debajo o por encima del pentagrama con las denominadas *líneas adicionales*. Estas líneas son líneas que se dibujan para representar notas que *se salen* del pentagrama (de esta forma, contando líneas adicionales podemos tener más de cinco líneas en el pentagrama). Las líneas adicionales se dibujan solo para la nota en cuestión, no cubriendo todo el pentagrama (pues entonces ya no sería un pentagrama).

No obstante, en lugar de añadir líneas adicionales podemos utilizar varias claves para adaptar el registro de la obra, aunque esto en ocasiones puede ser bastante lioso para el intérprete o compositor. El escenario lógico para emplear este procedimiento sería aquel en el que la obra contenga varias partes, cada una en una zona del pentagrama distinta, de forma que cada parte tendría así una *clave apropiada*.

Otro ejemplo claro en el uso de claves es el piano, en el cual se utilizan dos pentagramas, uno por cada mano (Figura 2.8). Las dos partes del pentagrama (correspondientes a cada mano) se tocan simultáneamente. En ambos pentagramas, se comienza por la nota *do,* siguiendo la escala tradicional ascendente hasta llegar al siguiente *do.* Luego se vuelve al primer *do* pasando por exactamente las mismas notas (escala descendente). En el caso de la mano derecha, tenemos la *clave de sol* (notas más agudas), mientras que, en la mano izquierda, tenemos la *clave de fa en cuarta línea* (notas más graves).

2.2.7 Acordes

Todo lo anterior para llegar a entender los acordes. Los acordes no son más que un conjunto de notas que suenan simultáneamente. Existen muchísimos tipos de acordes, pero en los que nos centramos nosotros son en los *triada* [16], formados por tres notas. La esencia de los acordes y el *cómo suenan* se debe básicamente a los intervalos musicales (distancia entre las notas del acorde), de ahí la importancia del recorrido explicativo previo.

Los acordes triada están formados básicamente por tres notas. De esas tres notas tenemos una que es la principal protagonista (se suele llamar fundamental), pues es en base a la cual se construye el acorde. Las otras dos notas del acorde tienen un intervalo de 3ª y de 5ª respecto a la fundamental, respectivamente.

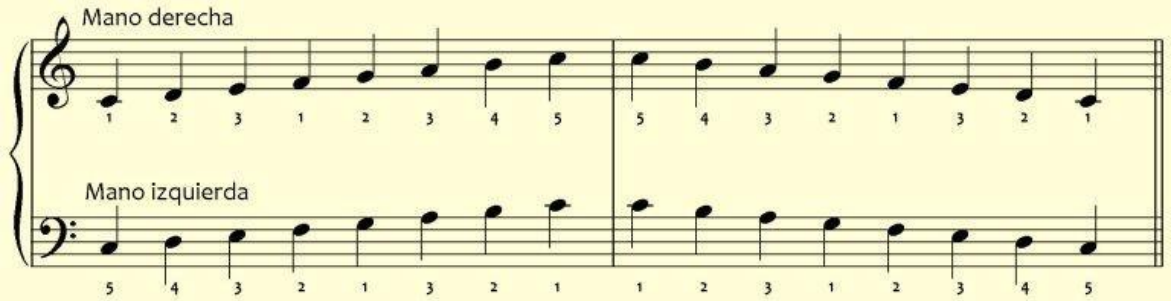

Figura 2.8: Escala de do mayor (clave de sol y clave de fa en cuarta línea).
Fuente: https://armoniacontemporanea.wordpress.com/2013/03/23/digitacion-escalas-mayores/

Nos centramos en los acordes *mayores* y *menores*. En los acordes *mayores* tenemos un intervalo de 3ª *mayor* mientras que en los acordes *menores* tenemos un intervalo de 3ª *menor* (de ahí sus nombres). En ambos casos, el intervalo de 5ª es una 5ª *justa*. Los acordes mayores se caracterizan por dar una sensación alegre, mientras que los menores una sensación más triste.

En nuestra aplicación usamos básicamente estos dos tipos de acordes, pero se jugaría con los intervalos de los acordes para buscar sonidos con mayor/menor tensión, ya que los *mayores* y los *menores* se caracterizan por una *limpieza sonora* (por expresarlo de algún modo). Por ejemplo, si en lugar de una 3ª *menor* tenemos una distancia de una 3ª *disminuida*, el acorde sonara con más tensión debido a una mayor proximidad de las notas. En algunos materiales convendría jugar con estas tensiones para acercarnos a la sensación táctil real. Obviamente, estos acordes dejarían de ser *mayores/menores*, pero tampoco conviene explicar todos los tipos de acordes ya que con estos dos (*mayores/menores*) tenemos la base para manipular los acordes y obtener sonidos que se adapten al contexto de la situación.

2.3 Papiroflexia

Existen materiales tales como el papel o el cartón que son muy usados en la actualidad para cosas muy habituales: escribir o dibujar en el caso de los distintos papeles y transportar elementos en el caso del cartón (fruta, objetos de casa en mudanzas...).

Sin embargo, hoy en día su uso se extiende más allá de aquello para lo que fueron diseñados. Efectivamente, hablo de las manualidades. En el caso del papel, existe ya un arte de diseñar y elaborar figuras, denominado papiroflexia (Origami) [17]. Este consiste únicamente en doblar el papel de una forma premeditada y precisa para obtener diferentes *esculturas a papel* que van desde barquitos hasta cisnes o rosas.

El caso del papel es un caso bastante popular, que casi todo el mundo conoce, ya que, ¿quién no ha hecho algún avión de papel para ver cuán lejos vuela? No obstante, el cartón es un caso más apartado, ya que muchas veces compramos piezas de cartón que tenemos que montar para obtener, casi siempre, una simple caja.

El cartón no sirve solo para hacer cajas de transporte de mercancías. Este material es una genialidad que sirve para elaborar un sinfín de objetos para cubrir diversas necesidades. Si se piensa que no es así, solo hay que buscar en Google *manualidades con cartón* y observar la infinidad de diseños que existen para este material. Es tal su potencial que puede emplearse cartón para amueblar una casa (mesas, estanterías...).

Es aquí donde empieza a ser bastante útil para la finalidad de este TFG. Debido a que nuestra aplicación/juego necesita de un soporte para sostener el móvil y enfocar una superficie de materiales, se pensó en elaborarlo con cartón. Esto tiene muchas ventajas: material económico, adaptable, desmontable, plegable, favorece el esfuerzo mental y la diversión al montarlo (posible parte del juego), entre otras.

Este famoso material tiene usos muy diversos precisamente porque existen distintos tipos de cartón para diferentes situaciones. Nosotros nos centramos en el más clásico de todos (el utilizado para nuestro soporte), el cartón ondulado. La resistencia, dureza y rigidez del cartón dependen de varios factores: grosor y número de *liners* y ondas que éste tenga. Los liners son láminas rectas en cuyo interior se sitúan las ondas, que son láminas onduladas. Las ondas parecen irrelevantes, pero

son las protagonistas en ofrecer la enorme resistencia característica de este material.

Los grosores de este tipo de cartón son:

- A: 4,2 – 4,8 mm
- C: 3,5 – 4,2 mm
- B: 2,2 – 3,5 mm
- E: 1,14 – 2,2 mm
- F: 0,75 – 1,14 mm
- N: 0,5 – 0,75 mm

Los tipos de cartón ondulado más básicos son los siguientes (Figura 2.9):

- Simple cara: contiene un *liner* exterior y una onda.
- Una onda: contiene dos *liners* que recubre una onda.
- Doble onda: contienen un dos *liners* exterior y otro interior que separa las dos ondas que contiene.
- Triple onda: análogo al de doble onda, pero con tres ondas y dos láminas intermedias en lugar de una, para separar las tres ondas que contiene.

En posteriores apartados de la memoria, se comenta en detalle el diseño del soporte realizado y el por qué es estable (no se cae). Naturalmente, esto se basa en principios arquitectónicos bastante básicos e intuitivos, entendibles por un perfil de persona estándar.

2.4 Sistemas similares

En este apartado presentamos la vigilancia tecnológica desarrollada para encontrar sistemas similares al que hemos desarrollado o bien relacionados con él. También mostramos la justificación de porqué hemos desarrollado el sistema tal como lo hemos hecho, observando las limitaciones tecnológicas que hemos encontrado para no realizarlo mediante sensores de tacto e interfaces cerebro-computador.

Como bien refleja el artículo titulado Tactile Perception of Roughness and Hardness to Discriminate Materials by Friction-Induced Vibration [18], estamos ante un tema bastante complejo y potente, que plantea muchas líneas de investigación abiertas a nuestros conocimientos y a nuestra imaginación.

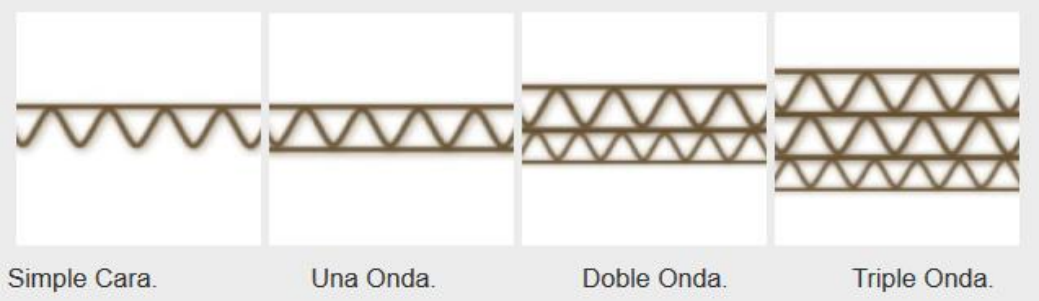

Figura 2.9: Tipos de cartón según el número de ondas y liners.
Fuente: http://cartonlab.com/tipos-de-carton-aplicaciones

En la actualidad, existen tecnologías y herramientas (básicamente, sensores) que nos permiten captar información del mundo empírico que nos rodea y trasladarlas a dispositivos de diferentes tipos. Cuesta encontrar sensores que integren sensaciones táctiles y sonoras, o al menos, no se adaptan a la especificación de nuestro problema. A continuación, describimos diferentes posibilidades teóricas a nuestro problema (identificar interacción con diferentes materiales y trasladar la información a un sistema informático capaz de procesarlo y emitir el sonido asociado a ese valor de material):

- Sensores cerebrales. Las sensaciones al tocar algún material captadas se deben traducir en impulsos nerviosos (similar a un encefalograma). Plantea complicaciones como el carácter cuantitativo de los datos obtenidos, pues los impulsos nerviosos se podrían analizar en este sentido más que en un sentido cualitativo. Esto se ha aplicado en juegos de concentración [19].
- Sensores táctiles (guantes). Usados para determinar el coeficiente de rozamiento de los materiales definiendo una velocidad de arrastre fija (controlada mediante un acelerómetro). Plantea problemas como la necesidad de cierto entrenamiento del usuario y la complejidad de implementación, aunque se han creado inventos como un robot que puede captar diferentes materiales solo con tocarlos gracias a este concepto [20]. La complejidad de esta monitorización es bastante elevada y puede implicar el uso de muchos sensores [21].
- Otra perspectiva para esto es determinar la vibración al percutir los materiales, de forma similar a las membranas de un tambor. También debería realizarse un entrenamiento para regular la fuerza/velocidad de golpeo.
- Sensores piezoeléctricos. Usados para recibir información de presión, aceleración y tensión, como los usados por los pedales de la batería electrónica.
- Sensor de ultrasonido (guantes). Utiliza la emisión y recepción de sonidos para determinar propiedades de una superficie o material [22]. En nuestro caso, puede plantear problemas con algunos materiales (las ondas pueden traspasar o reflejar según el tipo de material).
- Sensores capacitivos o inductivos. Existen implementaciones complejas y estáticas capaces de determinar el tipo de material mediante la conductividad o constante dieléctrica de los materiales (obtienes una tensión resultado tras aplicar una tensión inicial).
- Sensores fotoeléctricos (guantes). Usados para detectar materiales mediante el reflejo que emitan al ser incididos con un haz de luz blanca o RGB (*Red Green Blue*). Plantea problemas como la existencia de dos materiales del mismo color (mismo reflejo de luz). Existen implementaciones como *specdrums* [23], un anillo capaz de detectar el color.
- Sensor de campos magnéticos. Mediante la variación de campos magnéticos podemos detectar cierta interacción. Un ejemplo lo vemos en el *E-skin* [24] un anillo para manipular objetos virtuales.
- Tratamiento de imágenes. Podemos usar la comparación de imágenes, así como inteligencia artificial, para detectar cambios sobre los materiales (interacción). Esto supone limitaciones como obviar la interacción física, aunque existe remotas posibilidades como usar espejos para generar una imagen en tres dimensiones (3D), como un invento realizado para convertir la pantalla de un ordenador portátil en táctil [25].

Con todas las tecnologías *sobre la mesa*, finalmente se convierte en candidata la última de ellas, el

tratamiento de imágenes. El motivo principal de haber descartado todas y cada una de las otras tecnologías se debe a varios motivos:

- El tratamiento de datos es excesivamente complejo. Tanto la obtención como el tratamiento de datos pueden ser laboriosos, más aún considerando que nuestro dispositivo destino es un dispositivo móvil.
- El número de sensores u aparatos auxiliares para llevar a cabo la detección de interacción es excesivo. Esto complica el uso de la aplicación por parte del usuario. Muchos de los sensores requieren de cierta configuración o *montaje*, que a veces no es asumible por el usuario o simplemente es complicarle la vida. Como usuario lo lógico es disponer de un sistema cómodo de montar y de usar.
- El coste temporal y económico puede verse incrementado. Es natural que el planteamiento de un sistema que use sensores de diferentes clases pueda costar más dinero, no siendo este el objetivo principal. Además, implica mucho esfuerzo mental y técnico en cuanto a preparación o montaje se refiere.
- El tratamiento de imágenes solo haría uso de la aplicación móvil y de un soporte sencillo y asequible a todo el mundo, que es justo lo que se busca.

2.4.1 Aplicaciones similares

Existe una amplia variedad de aplicaciones que juegan con el mundo real para reproducir sonidos. Los tres ejemplos de aplicaciones existentes que comentamos se relacionan con este TFG, pero distan del objetivo de este:

- *Specdrums* [23], ya comentado anteriormente, es capaz de detectar diferentes colores y llevarlos al entorno del dispositivo móvil. Contando con su aplicación propia (*specdrums*), es capaz de registrar diferentes colores que toquemos y asociarlos a un sonido particular. A pesar de acercarse bastante a nuestro proyecto, plantea dos cosas que lo hacen alejarse un poco del objetivo de nuestro proyecto: está orientada a funcionar como un instrumento musical personalizado; nunca sentimos lo que estamos tocando, pues está el anillo de por medio; va orientado a diferentes colores, no a distintos materiales.

- *Texturas* [26] es una aplicación diseñada exclusivamente para iPad cuyo propósito es mostrar a niños (sobre todo autistas) una simulación de los sonidos que hacen objetos de la vida cotidiana como lo son piedras, tapones de plástico, arroz y pasta. Lo hace jugando con propiedades como el peso y el rozamiento.

- *Singing fingers* [27] es una aplicación diseñada para iOS (disponible en la App Store) que simplemente sirve para almacenar sonidos captados por el micrófono en un dibujo. Las posibilidades de reproducción y de creación musical con ese dibujo es bastante amplia. En gran parte, coge la esencia de nuestro proyecto con la diferencia de que juega con la percepción y con el sonido, es decir, plasma de forma visual los sonidos, que posteriormente pueden ser reproducidos de diferentes formas al interactuar con la aplicación.

3. Herramientas utilizadas

En este capítulo presentamos un análisis muy somero de las herramientas que hemos utilizado para desarrollar la aplicación.

3.1 Sourcetree

A la hora de avanzar en el proyecto es una buena práctica hacerlo contando con un control de versiones, en nuestro caso, Git. Lo mejor es contar con un repositorio remoto vinculado al local. Para dicho repositorio remoto usamos Bitbucket, herramienta que comentamos inmediatamente después a la actual herramienta.

Centrándonos en el control de versiones en Git, podríamos realizar todos los cambios mediante órdenes directas en consola. No obstante, disponemos de una herramienta para realizar todas esas operaciones mediante una interfaz gráfica, SourceTree.

Antes de comentar brevemente las operaciones de SourceTree, conviene explicar a un nivel básico y visual cómo funciona el control de versiones con Git. Partimos para ello de la propia interfaz gráfica de SourceTree. A continuación, se muestra una captura de la interfaz de SourceTree (Figura 3.1).

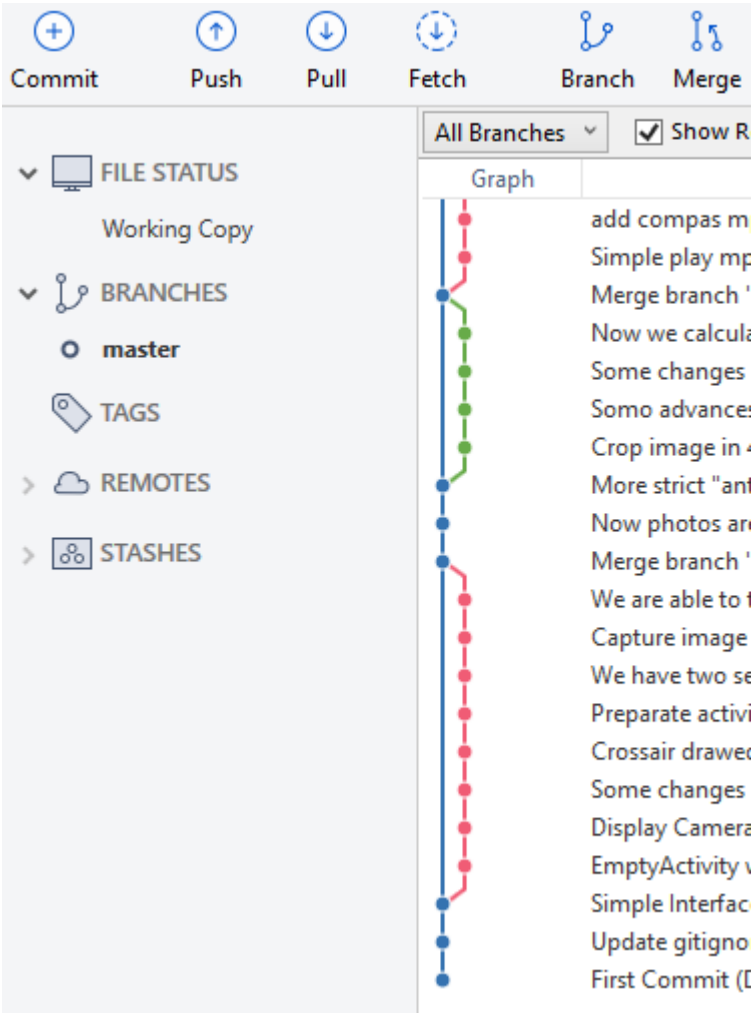

Figura 3.1: Captura de la interfaz de SourceTree.

Como se muestra, disponemos de la mayoría de las operaciones en el panel superior. Lo que nos interesa entender es el gráfico en el que se ven una serie de puntos o nodos unidos entre sí mediante líneas. Esos nodos son realmente las diferentes versiones del código (software). Cuando realizamos cambios en determinados archivos y estamos listos, los confirmamos (*Commit*) y pasan a formar parte de un nuevo nodo (junto con los otros archivos no cambiantes, aunque estos sean simples enlaces a versiones anteriores). Hay bifurcaciones en las líneas que unen los distintos nodos. Estas bifurcaciones son ramas distintas a la rama principal (*Master*, la rama de color azul). Estas ramas se suelen crear cuando se va a desarrollar una funcionalidad (o varias relacionadas).

Ahora sí, las operaciones (Git) que podemos realizar de forma gráfica con la ayuda de SourceTree son: *Commit* (confirmar cambios en el repositorio local, creando un nuevo *nodo*), *Merge* (fusionar ramas), *Checkout* (cambiar entre versiones), *Branch* (eliminar o crear nueva rama), *Push* (subir cambios al repositorio remoto) y *Pull* (bajar cambios desde el repositorio remoto).

Con todo esto, tenemos lo básico para entender Git y el control de versiones. Recalcamos que hemos empleado SourceTree como una simple herramienta gráfica para realizar las operaciones habituales de gestión de un repositorio software en un sistema de control de versiones.

3.2 Bitbucket

Bitbucket es una de las muchas herramientas disponibles para el control de versiones de un producto software mediante el uso de repositorios en Git (o Mercurial) de forma remota. Otra herramienta del mismo estilo y, quizás, bastante más conocida, es GitHub. No obstante, se descartó porque a nivel personal me parece mejor Bitbucket. Posteriormente se destacarían algunas características que Bitbucket tiene y GitHub no, a fin de encaminar la justificación de mi decisión.

Tanto Bitbucket como GitHub ofrecen la posibilidad de mantener un repositorio colaborativo remoto (fuera del entorno local) sobre el cual nuestro código evoluciona, pasando por diferentes versiones (gracias a Git). En nuestro caso, la parte colaborativa se omite al existir únicamente un desarrollador.

La forma que se ha empleado al trabajar con esta herramienta se ha basado en crear diferentes ramas que bifurcan de la rama principal (*Master*) cada vez que se desarrolla una funcionalidad o conjunto de funcionalidades significativas. Cuando se progresa de forma significativa o se finaliza el desarrollo en la rama recién creada, se actualizan los datos en el repositorio remoto. Naturalmente, la rama seguiría existiendo hasta que se acabara de trabajar en la misma, en cuyo caso se elimina tanto del repositorio local como del remoto (*Bitbucket*).

Esto permite que varios desarrolladores puedan colaborar de forma remota (cada cual en sus respectivas ramas) y disponer de un código centralizado y común, que naturalmente es el presente en la rama *Master*. La rama *Master* se actualiza con las fusiones de las ramas de cada desarrollador en un momento concreto del tiempo. En caso de que varios desarrolladores toquen en sus respectivas ramas un archivo común, a la hora de fusionar con *Master* el segundo de ellos sería notificado de un conflicto (se estaría tratando de actualizar un archivo que ya está siendo actualizado por otro desarrollador). Estos conflictos son inexistentes en nuestro proyecto al existir un único desarrollador, pero en caso de existir tienen que resolverse manualmente.

Naturalmente, cada producto impone sus ventajas respecto a la competencia, sin comentar la mayoría de las veces sus desventajas. Por lo tanto, *Atlassian* (empresa a la que pertenece Bitbucket) no es una excepción. En la parte de su Web asociada a software, concretamente la

asociada a Bitbucket, comenta una serie de ventajas del producto frente a GitHub. La mayoría son interesantes y hacen pensar al lector que la herramienta que debe usar es Bitbucket.

Sin embargo, para no alargar innecesariamente este documento, se indican los factores que me inclinaron por Bitbucket frente a GitHub:

- Repositorios privados gratuitos e ilimitados: ofrece la posibilidad de no hacer público el proyecto, en mi caso, es privado hasta que se vaya a entregar o necesite compartirse. Esta es una gran ventaja que pienso que eleva bastante el valor de Bitbucket frente a GitHub.
- Certificación *Soc 2 tipo II*: básicamente, garantiza que el código en la nube no se perdería ni sería accedido de manera no deseada. Bitbucket ha sido una de las primeras soluciones en garantizar esto.

3.3 Android Studio (IDE)

El entorno de desarrollo integrado (*IDE*, del inglés *Integrated Development Environment*) que usamos fue básicamente el Android Studio [28], ya que es el IDE oficial por excelencia para desarrollar aplicaciones para Android, además de su versatilidad y buena eficiencia. El lenguaje que finalmente empleamos fue Java [29] para la parte lógica y XML (*eXtensible Markup Language)* [30] para la parte de la interfaz (como es habitual en el desarrollo de aplicaciones en Android).

Comentamos algunas características de interés para la compresión del TFG:

- Utiliza un sistema de compilación basado en *Gradle* flexible: *Gradle* es una herramienta que se encarga de automatizar la construcción de nuestro código, esencialmente resolviendo las dependencias predefinidas por el desarrollador (de forma automática). Es bastante similar a otra herramienta llamada Maven, sobre la que no entramos en detalles.
- Posibilidad de ejecutar la aplicación en un rápido emulador o en un dispositivo Android en modo depuración conectado al equipo: el entorno ofrece la posibilidad de crear un dispositivo Android virtual bajo diferentes versiones del sistema Android (Ice Cream Sandwich, Jelly Bean, KitKat, LolliPop, Marshmallow, Nougat, Oreo...) y eligiendo el aspecto de dicho dispositivo virtual (la carcasa del dispositivo a emular, como, por ejemplo, un dispositivo móvil Nexus).
- Posibilidad de ejecución instantánea (*Instant Run*): básicamente nos ofrece la posibilidad de reflejar diferentes cambios en el código de la aplicación en el dispositivo/emulador de forma dinámica, sin necesidad de compilar de nuevo el APK (*Android Application Package*) de la aplicación. Por ejemplo, ejecutamos nuestra aplicación y nos damos cuenta de que realizamos mal una operación en el código. Lo que hacemos es corregir dicho error y, con la opción de ejecución instantánea, actualizamos nuestra aplicación rápidamente, de forma que sin pasar por todo el proceso de compilación e instalación del APK tendríamos en el dispositivo destino los cambios realizados en última instancia.
- Disponibilidad de una interfaz gráfica para la cómoda edición de la parte gráfica de la aplicación (XML): aunque muchas veces necesitemos modificar archivos XML directamente por alguna limitación de la interfaz gráfica (como en prácticamente cualquier IDE), la interfaz de edición de la parte gráfica en Android Studio es bastante útil. Permite seleccionar y arrastrar en la interfaz los elementos que deseamos y permite ajustar diferentes parámetros

en los mismos, tales como la separación respecto a otros componentes, la ubicación en la pantalla del dispositivo, el tamaño, el color de algunos componentes, entre otras propiedades.

- Estructura del proyecto: independientemente de la estructura de archivos del proyecto, dentro de Android Studio se ofrece una configuración de archivos adaptada al desarrollador y que es bastante agradable. Por ejemplo, se ofrece una diferenciación clara entre archivos de la parte lógica (Java), los recursos (XML o Layouts, valores centralizados de colores, cadenas de caracteres...) y el Manifest del proyecto (donde se ubican parámetros importantes y vitales del proyecto).
- Integración de funcionalidades para el control de versiones: como muchos otros, este IDE ofrece la posibilidad de realizar diferentes operaciones para el control de versiones en Git. Aunque nosotros usemos SourceTree para esta labor, está bien contar con esta herramienta dentro del propio IDE (muy útil cuando, por ejemplo, tenemos que crear el repositorio Git al comienzo del proyecto).
- Atajos de teclado: dispone de diferentes atajos de teclado que agilizan la ejecución de tareas por parte del desarrollador.

3.3.1 Librerías: OpenCV

En el desarrollo de nuestro proyecto en Android, cabe destacar una librería muy importante y que ayuda bastante en el proceso: Open CV (*Open Computer Vision*).

Esta librería es libre y su principal aporte está relacionado con el tratamiento de imágenes. Está disponible en varias plataformas y lenguajes, pero nosotros nos centramos en Android (Java). No obstante, algunas funciones de esta librería están desarrolladas en C, teniendo que incluirlas y adaptarlas al proyecto.

De entre las funciones que ofrece, tenemos:

- Captura y mostrado en tiempo real de la cámara de nuestro dispositivo,
- Cambiar parámetros para cambiar la captura de imágenes en la cámara (blanco y negro, negativo...),
- Dibujar sobre la captura de imágenes en la cámara, detectar diferentes movimientos y caras/objetos, entre otros.

Como se observa, no solo tiene que ver con la edición de imágenes, sino que incluye funciones relacionadas con inteligencia artificial.

Precisamente por todo lo anterior y sin extraer más detalles de esta librería, se ha optado por incluir y usar OpenCV en nuestro proyecto.

3.4 Node.js y XAMPP

Para el almacenamiento de los datos usados por nuestra aplicación, a pesar de considerar en un principio disponer de un conjunto de datos integrados en la APK de la misma, al final se optó por algo más lógico y, quizás, más estandarizado. Efectivamente, hablamos de disponer de una base de

datos de la cual se descargan todos los datos necesarios para nuestra aplicación. En nuestro caso, estos datos son sonidos asociados a cada uno de los materiales (láminas de materiales).

- *XAMPP*: aloja el servicio de la base de datos. Es un paquete (software libre) que incluye varias herramientas orientadas a proporcionar un servidor Web. Realmente nosotros solo utilizamos la base de datos (aunque para ello sea necesario el uso del servidor Web). Concretamente, sobre XAMPP instalamos MySQL (*My Structured Query Language)* para el manejo de la base de datos y PHP para el servidor Web (Apache). El servidor Web no se usa explícitamente, únicamente debemos de tenerlo activado para que la base de datos funcione, tal y como mostramos en la Figura 3.2.

  Para configurar la base de datos en cuestión utilizamos su panel de administración. En la parte de MySQL al pulsar en el botón *Admin* se dirige al navegador (*localhost*) para configurar las bases de datos del servidor. A partir de ahí, es simplemente aplicar sentencias SQL (*Structured Query Language*) y ayudarnos cuando sea conveniente de la interfaz gráfica.

- *Node.js*: basándonos en la Web oficial, es un entorno de ejecución de Javascript orientado a eventos asíncronos cuya finalidad es construir aplicaciones en red que sean escalables. Una gran particularidad de Node.js es que cuenta con el conjunto de paquetes y librerías de código abierto más grande del mundo, npm.

  Node.js se encarga de ofrecer a los usuarios las diferentes operaciones que se realizan contra la base de datos. Saliéndonos un poco del entorno formal marcado por la documentación oficial de Node.js, hay que comentar que los paquetes en Node.js (*npm*) son muy potentes. Si necesitamos de una funcionalidad concreta probablemente estaría disponible, lo único que tendríamos que hacer es incluirla en nuestro proyecto e instalarla mediante una simple orden en consola (*npm install*).

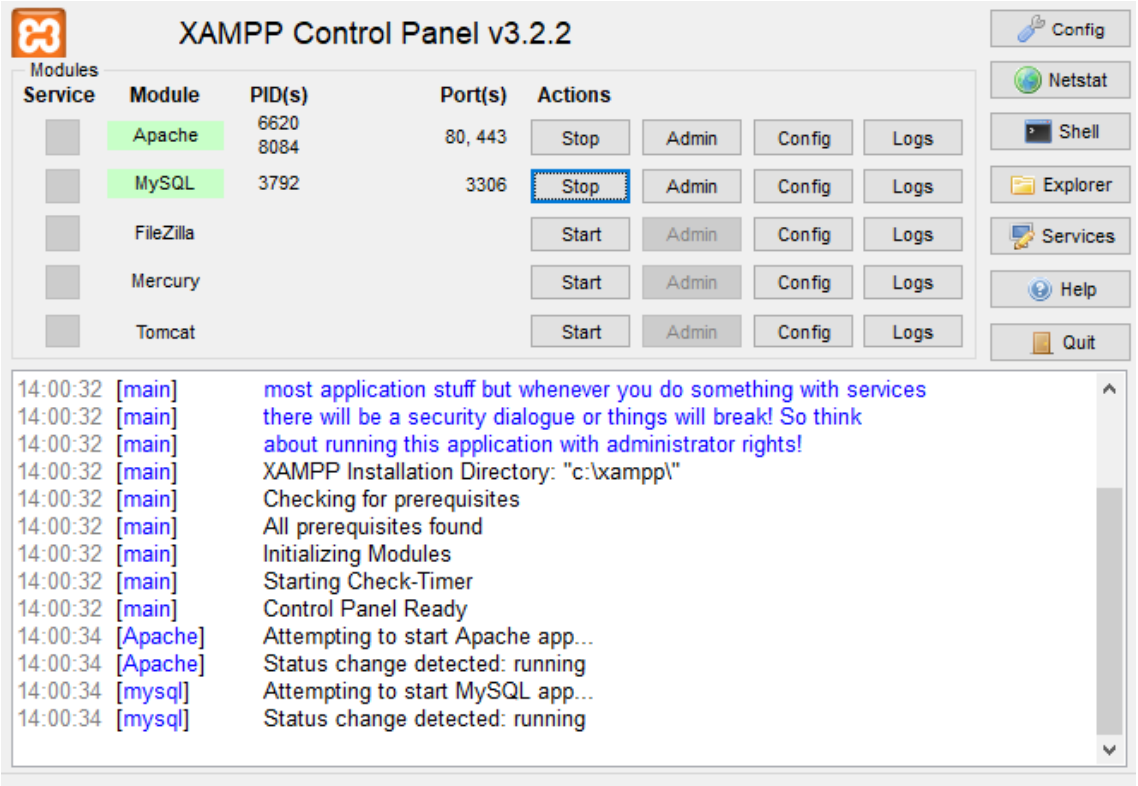

Figura 3.2: Pantalla de inicio de XAMPP. Apache (PHP) y MySQL activados.

Escucha peticiones por el puerto 3000. Dentro de nuestro proyecto en Node.js se han creado diferentes rutas para cada una de las operaciones requeridas por nuestra aplicación Android en la base de datos. El proyecto Node.js averigua automáticamente a qué base de datos acceder porque ésta se declara dentro de cada proyecto (concretamente, dentro de cada una de las rutas). Al estar el Node.js y el XAMPP ejecutándose en la misma máquina, no existen problemas entre ambos.

Una anotación importante sobre la comunicación de datos (entre cliente y servidor de Node.js) es que se realiza mediante archivos JSON (*JavaScript Object Notation)*. En un principio, al tener tanto Node.js como XAMPP en nuestra red local (en el lugar donde trabajamos), solo podemos acceder a este servicio si estamos conectados con nuestro dispositivo Android a esta red. Sin embargo, se ha asignado a la dirección IP (*Internet Protocol*) pública de nuestro encaminador principal un DNS (*Domain Name System)* dinámico que sería el usado para acceder al equipo, en el que tenemos alojado el Node.js (puerto 3000) y la base de datos. Únicamente tenemos acceso al equipo a través de este puerto, por lo que lo único que se podría realizar desde el exterior en nuestro equipo serían operaciones en la base de datos desde nuestra aplicación.

Debido a que se trata con servidores, servicios y bases de datos, es clave comentar aspectos relativos a la seguridad. Por motivos de simplicidad y, sobre todo, por el hecho de almacenar datos (en la base de datos) que no son críticos, no contemplamos servicios adicionales de seguridad.

3.5 StarUML

Cualquier buen proyecto software debería contener diagramas UML que respalden el desarrollo. En nuestro proyecto se plasman muchos de los detalles del desarrollo en estos diagramas. Para esta labor, naturalmente se requiere de herramientas de edición especializadas en ello y, aunque la que hemos usado no es la mejor de todas (bajo mi punto de vista), se ve respaldada por algunos argumentos irrefutables:

- Es la que se ha usado a lo largo del grado para elaborar todos y cada uno de los diagramas (UML).
- No excesivamente compleja y, además, es gratuita (al menos en la versión usada).
- Tiene la capacidad de abordar todos los diagramas necesarios para el proyecto. Naturalmente, unos los diseñaría mejor que otros, pero conlleva a no tener que usar aplicaciones diferentes para cada diagrama. Uno de los que peor afronta es el de la base de datos, pero afortunadamente tampoco nos perjudicaría demasiado.

El nombre aún no mencionado de la herramienta que hemos usado es *StarUML* [31]. Como hemos comentado, en ocasiones no es intuitivo, pero al no ser excesivamente complejo de usar es una herramienta totalmente asequible para nuestro proyecto.

Algunos de los diagramas elaborados con esta herramienta son: diagrama de casos de uso, diagrama de clases, diagrama de la base de datos (*Entidad-Relación*), diagrama de secuencia, entre otros.

En los sucesivos diagramas, como mostramos a lo largo de este documento, explicamos puntos que consideremos relevantes o bien, poco claros en el diagrama.

3.6 Noteflight

Al igual que la anterior herramienta, Noteflight [32] no se categorizaría como principal dentro de nuestro entorno de trabajo. No obstante, la hemos empleado para aquello relativo a los sonidos musicales, como vemos a continuación.

La función principal de Noteflight es la de servir como herramienta para la composición musical. Ofrece gran variedad de utilidades musicales para llegar a componer como si lo hiciéramos a mano, sobre un pentagrama de papel. No obstante, a pesar de no tener la libertad de *escribir cualquier cosa*, libertad que si tenemos cuando lo hacemos a mano sobre un papel, ofrece ventajas adicionales.

Con Noteflight se puede elegir un instrumento y reproducir la composición con el sonido de dicho instrumento. La calidad del instrumento al reproducirse varía según el instrumento seleccionado, pero obviando ese detalle, la calidad general de la reproducción es muy buena. Es cierto que puede notarse que la obra no la está interpretando un músico con un instrumento real, pero tampoco se aleja tanto de esta sensación. Al menos algo es seguro, se reproduciría tal y como se ha compuesto. Cuando se crea una composición, se tiene la posibilidad de compartirla e incluso de exportarla en diferentes formatos.

Finalmente, terminamos diciendo que se usó esta herramienta para crear un simple compás con el acorde (o entidad musical) correspondiente y exportarlo a un archivo de audio. Así, podemos jugar con los diferentes elementos musicales para crear los sonidos apropiados para nuestra aplicación. Es mucho más fácil proceder de esta forma que buscando sonidos predefinidos por la red o crearlos mediante grabaciones, pues tenemos mayor facilidad y libertad para obtener los sonidos que queramos con los instrumentos que queramos (de los disponibles).

3.7 Gimp

A pesar de que Gimp [33] no es una herramienta fundamental en el desarrollo de la aplicación, pues se ha empleado para diseñar el icono de la aplicación y algunas imágenes de la interfaz de usuario. Esta herramienta sirve para editar imágenes de forma profesional. Podemos categorizarla como un *Photoshop* [34] *gratuito*.

La usamos para editar cualquier imagen usada en nuestra aplicación. Este grupo de imágenes recogen: el icono de nuestra aplicación (simplista y totalmente artesanal), la imagen de fondo de la pantalla principal de nuestra aplicación (ajuste de proporciones de la imagen para su correcta adaptación a la pantalla del dispositivo móvil) y diferentes iconos dentro de nuestra aplicación.

Para estos últimos iconos, se usó más el IDE de desarrollo que GIMP, ya que Android Studio ofrece la posibilidad de crear iconos predefinidos para determinadas acciones (menú, ajustes...), los cuales podemos personalizar en tamaño y color.

Como no es una herramienta protagonista en el desarrollo de la aplicación, simplemente comentar que lo más usado en Gimp ha sido la gestión de imágenes por capas, ya que nos ha facilitado la edición *por partes* de una imagen. Además, hemos sacado provecho de otras utilidades como el relleno de texturas para diseñar el icono de la aplicación (textura de madera, por ejemplo) y la herramienta de escalado y rotación de imágenes para obtener el tamaño y orientación perfectos en cada edición.

3.8 Camtasia Studio

Como parte de la aplicación, se ha elaborado un apartado de ayuda para aquel que lo requiera. No es un disparate pensar en este apartado, no sólo porque la mayoría de las aplicaciones cuentan con ello siendo prácticamente un estándar, sino porque en nuestra aplicación es totalmente necesario.

Debemos tener en cuenta que tenemos varios aspectos en los que pueden surgir dudas, ya que contamos con varios elementos: configuración de la aplicación, elaboración y montaje del soporte, uso del soporte, entre otros.

Para elaborar estos videos, no nos limitamos a grabar un video, sino que lo editamos adecuadamente para maximizar el entendimiento de este. Naturalmente, es aquí donde entra en juego la utilidad de la aplicación encargada de esta importante misión, *Camtasia Studio* [35] en su versión portable. Está claro que no es una aplicación para crear efectos especiales de películas de ciencia ficción, pero para la elaboración de videotutoriales es una herramienta adecuada.

No solo destaca por tener una interfaz simple e intuitiva, sino por aportar funciones muy útiles para la elaboración de videos de este tipo:

- Cortar videos.
- Insertar transiciones entre distintos clips de video.
- Posibilidad de congelar la imagen los segundos deseados.
- Elaboración de diálogos explicativos durante la reproducción del video.
- Posibilidad de editar el audio.
- Editar la velocidad de cada clip.
- Inserción de iconos y de textos explicativos superpuestos
- ...

A pesar de haber pensado y utilizado diferentes herramientas de este estilo, esta es sin duda la mejor (de entre las probadas), pues permite plasmar los pensamientos que se tienen sobre *cómo elaborar el video* de una manera casi inmediata, sin ser un verdadero experto en la aplicación o en la edición de video, en general.

3.9 Uso integrado de herramientas

Una vez presentadas las herramientas usadas a lo largo del desarrollo del proyecto, es necesario aclarar la interacción entre ellas. Nos centramos exclusivamente en lo que al funcionamiento básico de la aplicación se refiere.

En la Figura 3.3 mostramos tres elementos: lámina de materiales, dispositivo móvil (aplicación desarrollada en Android Studio, implicando la integración de las librerías usadas) y servidor (Node.js y XAMMP). Como podemos observar, todos y cada uno de los sonidos se reproducen del dispositivo móvil usando la aplicación desarrollada en Android Studio. Sin embargo, existen dos cuestiones importantes: de dónde se obtienen los sonidos y cómo se determina la

emisión de los sonidos.

Los sonidos se descargan comunicando con el servidor, concretamente con el Node.js, que se encarga de hacerle peticiones a la base de datos alojada en XAMPP. Por otra parte, los sonidos del metrónomo se emiten de acuerdo con el ajuste del metrónomo, simplemente. Los sonidos que llevan el pulso siempre se emiten, pero los sonidos que se emiten al interaccionar con los materiales (última parte del compás) no lo hacen siempre, solo cuando interaccionamos con los materiales.

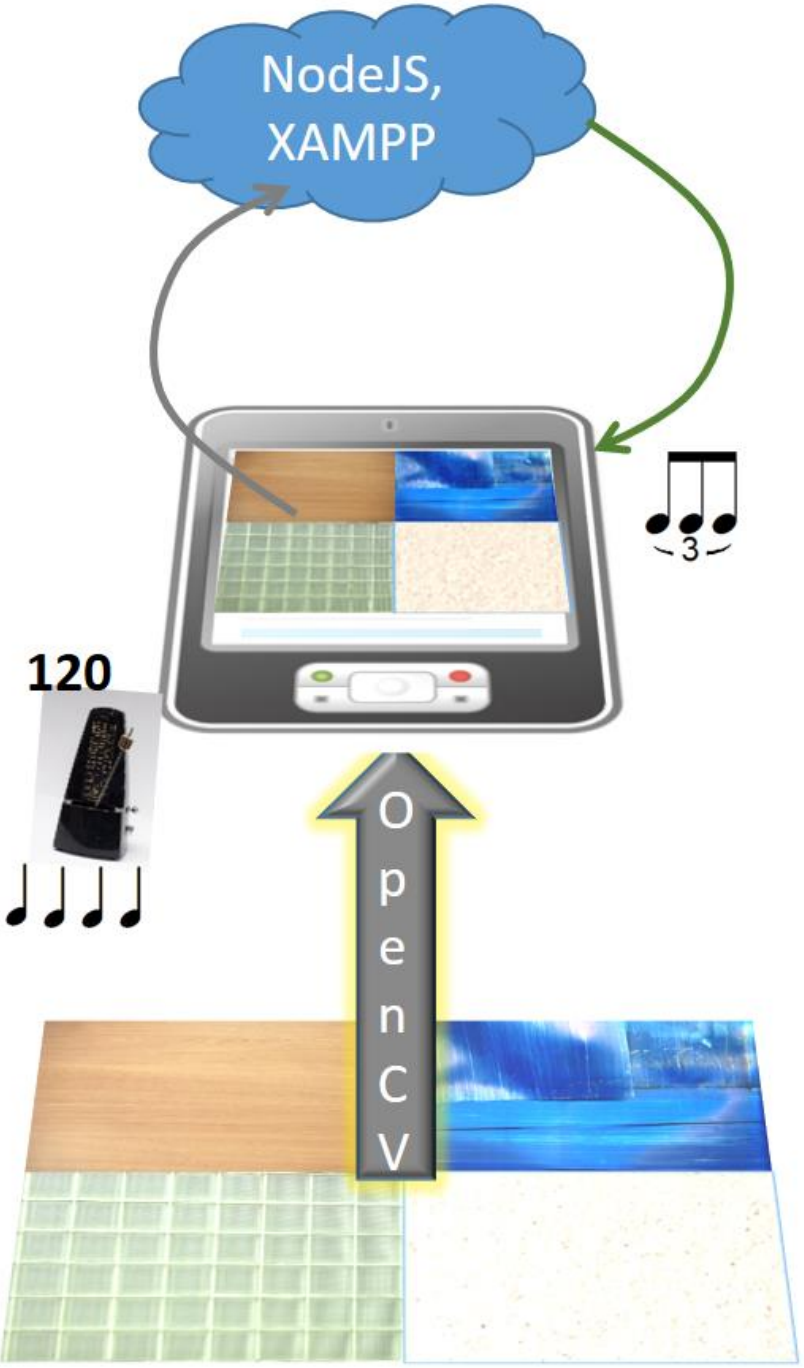

Figura 3.3: Elementos principales elaborados con diferentes herramientas.

De esta labor se encarga la librería que hemos usado como herramienta principal en el desarrollo, OpenCV, que incluye todas las funciones necesarias para el tratamiento de imágenes. En nuestro caso, detecta la interacción. Con esos datos, se determina la emisión de un sonido cuando interactuamos con un material (Figura 3.4) o la emisión de un sonido de error cuando interactuamos con varios materiales (Figura 3.5), en ambos casos, habiendo dejado al sistema tomar la captura de referencia. Si no detecta interacción, simplemente no se emite ningún sonido.

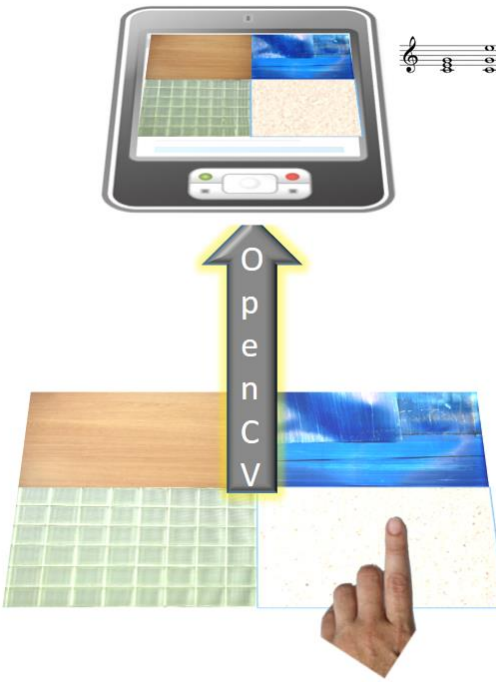

Figura 3.4: Interacción correcta con uno de los materiales.

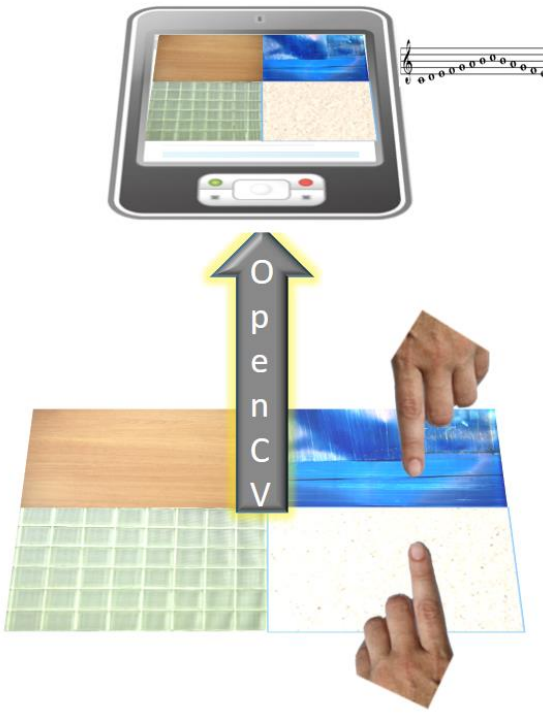

Figura 3.5: Interacción incorrecta con dos o más materiales

4. Descripción del desarrollo del proyecto

En este capítulo presentamos las descripciones: funcional, organizativa de implementación y de interfaz de usuario de la aplicación desarrollada.

4.1 Introducción

El proyecto ha seguido una evolución bastante importante. En un principio, las posibilidades de implementación de esta aplicación eran muy diversas. Hay algo claro, el proyecto tal y como está en su versión de entrega es muy similar a como se planteó en un principio, a excepción de la forma en la que detectamos la interacción con los materiales.

En cuanto a esto surgieron muchas ideas, algunas, eso sí, un poco complejas, pero en algo parecido a una *lluvia de ideas* (*Brainstorming)* esto es lo lógico (incluso ideal). Algunas de esas ideas fueron:

- Detección de interacción con un sensor de tacto por presión.
- Detección de interacción mediante un sensor capaz de detectar las vibraciones de un determinado material al desplazar nuestro dedo sobre él.
- Detección de impulsos nerviosos para determinar cuando estábamos tocando un determinado material (quizás la más compleja de todas).
- Detección mediante análisis de imágenes (mediante software). Puede contener o no detección de formas (quizás con inteligencia artificial).

La opción por la que se optó fue la comparación absoluta de pixeles disponiendo de una captura de referencia. Se descartó la detección de formas para evitar condicionar el sistema a una forma de interacción concreta (meter un solo dedo, meter la palma completa de una manera determinada...).

Esto nos llevó a descartar la detección real de interacción, por lo que sí, finalmente nuestro sistema emitiera un sonido siempre y cuando nuestra mano u otro objeto esté sobre uno de los materiales, independientemente de que se esté tocando o no. Esta decisión no fue trivial, pero los principales argumentos que la respaldan son:

- Centrarnos en la experiencia de juego y en los conocimientos que esconde, es decir, el hecho de relacionar tacto de materiales con sonidos concretos.
- Si realmente se quiere disfrutar del juego, lo ideal es tocar verdaderamente los materiales. Pensamos que, si el jugador no toca los materiales, siendo previamente concienciado de ello, es porque no quiere. Si toma esta decisión ilógica se pierde la experiencia de juego, el único perjudicado es él/ella.

Con esta idea clara, nos vemos obligados a elaborar un soporte que se detalla en apartados posteriores de esta memoria. Esta idea o planteamiento inicial es clave antes de iniciar la programación de la aplicación, pues cambia radicalmente el enfoque a la hora del desarrollo. Si esta idea se viera alterada una vez comenzado el desarrollo, debemos ser conscientes de que el trabajo realizado hasta ahora (en cuanto a desarrollo se refiere) podría no servirnos de nada, lo que se traduce en tiempo perdido.

A continuación, se explica de forma más detallada la planificación del proyecto, así como

diferentes características o propiedades de este, pasando por aspectos funcionales, organizativos y de implementación.

4.2 Funcional

En primer lugar, presentamos la metodología empleada para planificar y guiar el proyecto en su desarrollo. Si bien no se puede considerar metodología Scrum (debido a las limitaciones marcadas por el entorno educativo en el que se desarrolla el proyecto), nos hemos basado en esta metodología para planificar y guiar todo el TFG. Como consecuencia, muchos de los conceptos expuestos en la planificación nos recuerdan a esta metodología.

4.2.1 Metodología empleada

El primer paso en la ejecución de esta metodología fue elaborar la pila de producto inicial. La pila de producto consiste en el conjunto de historias de usuario (funcionalidades de la aplicación, requisitos) que se desarrollarían durante la realización del TFG. Esta pila de producto inicial no es estática, pues puede crecer a medida que surjan nuevas funcionalidades no definidas para la aplicación, pero debe de recoger la mayoría de las funcionalidades de la aplicación para permitir la planificación de los sucesivos *Sprints*.

Un *Sprint* consiste en el desarrollo de algunas funcionalidades pertenecientes al proyecto en un periodo de tiempo relativamente corto, variable normalmente entre 2 y 4 semanas. Como resultado del desarrollo de estas funcionalidades, el equipo de desarrollo obtiene un incremento, que no es más que una versión de software del producto, totalmente entregable y funcional. Como resultado de cada *Sprint*, el producto final ofrecido al cliente va creciendo.

En la elaboración de las historias de usuario, así como sus criterios de validación, el cliente tiene un papel muy importante. No obstante, en este caso y a diferencia de *Scrum*, sería una sola persona la que asuma los papeles del *Product Owner* (representante del cliente) y del equipo de desarrollo (integrando los roles de desarrollador y de *Scrum Master* en uno solo). Al ser una sola *cabeza pensante*, también se propician más errores en la estimación de las historias, pues no contamos con más puntos de vista para contrastar opiniones al respecto (como se hace, por ejemplo, al aplicar *Poker Planning*).

Con el objetivo de cubrir todas las horas estimadas para el proyecto, hemos decidido llevar a cabo 3 *Sprints* (*Sprint* 0, *Sprint* 1 y *Sprint* 2). A continuación, comentamos los distintos *Sprints*, incluyendo: historias del *Sprint* (planificación *Sprint*), *Sprint Burn Down Chart*, estimación de carga de trabajo (tiempo) tanto para el siguiente *Sprint* y conclusiones o aportaciones importantes del *Sprint*. Para no sobrecargar el documento con información redundante, la pila de producto final está formada por la fusión de las historias pertenecientes a cada *Sprint*. Naturalmente algunas historias de usuario no se introdujeron en la pila de producto inicial, sino que surgieron a posteriori.

4.2.2 Sprint 0

Este *Sprint* 0 se ejecuta en el periodo de tiempo correspondiente entre el 14 de marzo y el 01 de abril (3 semanas). El resto del mes de marzo se empleó no sólo en la elaboración de la pila de producto, sino en la planificación de este *Sprint*. Asimismo, se aprovechó antes del comienzo de este *Sprint* para revisar discretamente parte de la documentación relativa al lenguaje de programación y las tecnologías que usamos en el desarrollo.

Días	Horas/Día	Total Disponibilidad	Factor de Foco	Velocidad Estimada
9	8	72	0,6	43

Figura 4.1: Estimación de horas del Sprint 0.

Centrándonos en la planificación de este *Sprint*, para saber cuántas horas deberíamos dedicarle (horas estimadas), nos basamos en un valor denominado *Factor de foco*, cuya función es disminuir el valor de horas dedicadas al *Sprint* (horas ideales) en función del rendimiento que hayamos tenido en anteriores *Sprints* (horas realistas). Se basa en la premisa de que una persona nunca tiene un rendimiento del 100%, ya sea por su estado mental/emocional o cualquier imprevisto que pueda surgir en el desarrollo. Como podemos observar en la Figura 4.1, hemos escogido un factor de foco arbitrario de 0,6 (ni demasiado optimista ni demasiado pesimista). Al ser el primer *Sprint*, no tenemos *Sprints* anteriores de referencia para el cálculo del factor de foco.

Con todo esto, tenemos unos 9 días laborables de esas tres semanas (3 días por semana), que suman un total de 72 horas ideales a dedicar en este *Sprint*. Aplicando el factor de foco obtenemos unas 43 horas, que deberían ser las horas totales (estimadas) a dedicar en este primer *Sprint*.

Las historias que componen este *Sprint* se muestran en la Figura 4.2. Este *Sprint* es un poco especial, pues puede incluir historias de usuario *internas*, es decir, que sirven al desarrollador para obtener información de una determinada tecnología, herramienta, librería u otro recurso. En consecuencia, sirve para descartar u optar por una determinada tecnología/herramienta a la hora de desarrollar alguna/s funcionalidad/es.

Como podemos observar en la Figura 4.2, tenemos el nombre de cada historia con su descripción/definición y otros datos como son: prioridad (entre 100 y 0), validación (*prueba que considera una historia como válida*), estimación (en horas), velocidad real (en horas) y notas asociadas a la historia. A la izquierda de cada historia observamos tres posibles colores: blanco (si la estimación de la historia fue justa y correcta), verde (si se sobreestimó, es decir, se acabó antes de lo previsto) y rojo (se subestimó, es decir, se acabó la historia dedicando más horas de las previstas). Esta notación se recogería en los sucesivos *Sprints*.

El total de horas dedicadas a desarrollar estas funcionalidades fue de 39 horas, por lo que se considera que hemos sobreestimado este *Sprint*. De hecho, son más las historias en color blanco y verde que en rojo.

Id	Nombre	Descripción	Prioridad	Validación	Estimación	Velocidad real	Notas
1	Acceder a la aplicación	Como usuario, tengo que poder acceder a la aplicación para poder usarla.	100	El usuario es capaz de abrir la aplicación y ver la pantalla de inicio.	2	1	
2	Crear primer "Activity" **(Interna)**	Como usuario, tengo que poder abrir una "Activity" en Android distinta a la principal.	100	Cuando el usuario acciona un mecanismo (botón) en la actividad principal, se le redirecciona a otra activity.	2	4	Por revisar documentación y problemas en el IDE.
3	Visualizar la cámara nativa	Como usuario, tengo que poder ver una vista previa de la cámara para saber que captura las imágenes necesarias.	98	El usuario podrá ver y comprobar que las imágenes mostradas son las captadas por la cámara en tiempo real.	4	4	
4	Tomar capturas de la cámara nativa	Como usuario, tengo que poder capturar instantáneas de la cámara para poder guardarlas en el dispositivo.	98	Cuando el usuario acciona el mecanismo para capturar una foto, se muestra una vista previa con la imagen capturada.	4	2	
5	Almacenar capturas de la cámara nativa	Como usuario, tengo que poder guardar las capturas realizadas para que se puedan analizar.	98	Cuando el usuario captura una imagen, se guarda en una ruta predefinida del dispositivo.	9	5	
6	Tomar capturas de la cámara nativa en segundo plano	Como usuario, tengo que poder capturar instantáneas de la cámara para poder guardarlas en el dispositivo sin necesidad de ver la vista previa de la cámara.	96	Cuando el usuario acciona el mecanismo para capturar una foto, se captura y guarda la foto en una ruta predefinida del dispositivo.	12	11	
7	Mostrar con Opencv imágenes almacenadas	Como usuario, tengo que poder ver una imagen cualquiera almacenada en el dispositivo para garantizar que se puede analizar una imagen almacenada.	94	Cuando se abre la aplicación, el usuario es capaz de visualizar una imagen cargada de forma predeterminada.	10	12	Por problemas al integrar librería.
					43	39	

Figura 4.2: Historias de usuario correspondientes al Sprint 0.

El avance en horas durante cada día del *Sprint* se recoge de forma gráfica en la Figura 4.3. Tenemos una serie que es la estimación lineal del *Sprint* (color azul) y otra que recoge el progreso real (color rojo). En esta última se restan las horas (estimadas) de las historias terminadas el día anterior. Es lo que corresponde en Scrum a la reunión diaria, con la diferencia de que nosotros simplemente actualizamos las historias finalizadas y la gráfica que refleja el progreso. Aclarar que los días que figuran en la gráfica corresponden a los días de trabajo, no a todos los días que integran las 3 semanas del *Sprint*.

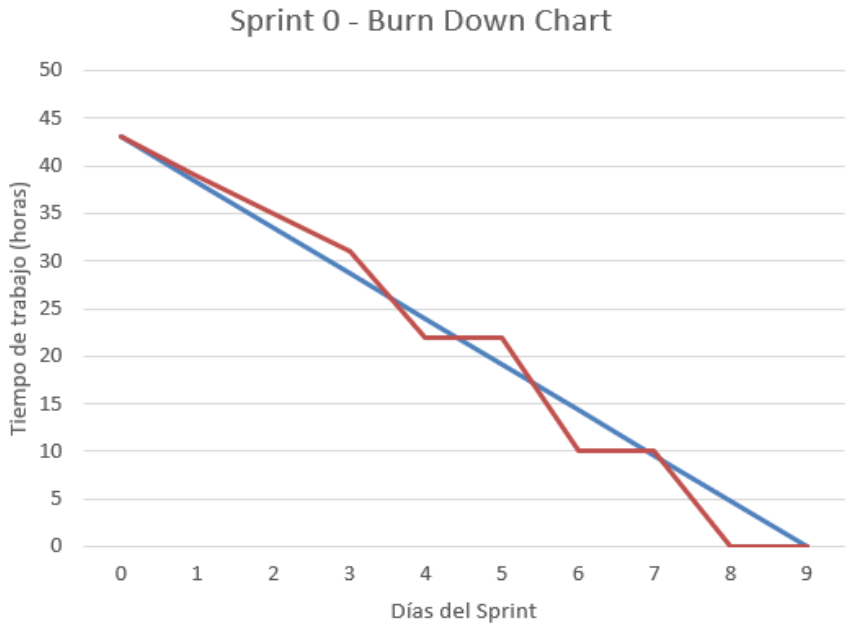

Figura 4.3: Gráfica que refleja el progreso diario del Sprint 0.

4.2.3 Sprint 1

Este *Sprint* 1 se ejecuta en el periodo de tiempo correspondiente entre el 11 de abril y el 29 de abril (3 semanas). Lo que resta del mes de abril entre la finalización del *Sprint* 0 y el comienzo de este *Sprint* se dedicó a revisar el *Sprint* 0 (lo que en Scrum equivale a la revisión y retrospectiva del *Sprint*) así como a la planificación de este *Sprint* 1.

Tal y como refleja la Figura 4.4, de acuerdo con el resultado del anterior *Sprint*, el factor de foco debe aumentar a un valor de 0,65 al haber realizado las historias previstas en menos tiempo del estimado.

Al tratar la estimación en horas en lugar de en puntos de historia (una vez calculado, se adapta mejor al tiempo productivo del desarrollador), el cálculo del nuevo factor de foco varía. Si hubiéramos empleado las 43 horas estimadas en el *Sprint* 0 para completarlo, se obtendría, lógicamente, el mismo factor de foco fruto de realizar la operación 43/73. En nuestro caso, nos sobraron 4 de esas horas (43-39), por lo que el factor de foco que se adapta a nuestra velocidad es el obtenido al realizar la operación (43+4)/72, que es efectivamente 0,65.

De las 72 horas laborables de este *Sprint*, nos quedamos tan sólo con unas 47 horas al multiplicar por el nuevo factor de foco. Estás 47 horas estimadas deberían ser las cubiertas en este *Sprint* 1.

Las historias que componen este *Sprint* pueden visualizarse en la Figura 4.5 mostrada a continuación. En general, el protagonismo en este *Sprint* recae en las funcionalidades relativas a la detección de supuesta interacción (procesamiento de imágenes) y la reproducción de sonidos (metrónomo y sonidos de materiales). De hecho, es el objetivo principal que se busca con este *Sprint*, el motivo de éste.

La estructura de la tabla de historias de usuario es idéntica a la mostrada en el *Sprint* 0. El total de horas dedicadas a desarrollar estas funcionalidades fue de 54 horas, por lo que se considera que hemos subestimado este *Sprint*, dedicando más horas de las previstas. Observamos un mayor número de historias de usuario en color rojo.

Como podemos observar, los principales responsables de esto son las funcionalidades asociadas al metrónomo y la preparación de la base de datos para que nuestra aplicación fuera capaz de descargar los sonidos automáticamente. Sin duda son tareas que esconden un esfuerzo adicional, en primer lugar, por la sincronización que implica el metrónomo en un sistema con recursos finitos, en segundo lugar, por la falta de costumbre a tratar los datos (JSON) provenientes de la base de datos, sobre todo los bytes que deben de almacenarse como un archivo de sonido.

Días	Horas/Día	Total Disponibilidad	Factor de Foco	Velocidad Estimada
9	8	72	0,65	47

Figura 4.4: Estimación de horas del Sprint 1.

Id	Nombre	Descripción	Prioridad	Área del gráfico	Estimación	Velocidad real	Notas
8	Tomar capturas de la cámara en segundo plano (Open CV)	Como usuario, tengo que poder realizar capturas de la vista actual de la cámara y almacenarlas en el dispositivo.	92	Cuando el usuario acciona el mecanismo para capturar una foto, se captura y se almacena automaticamente en una ruta predefinida sin que el usuario aprecie cambios en la interfaz.	1	1	
9	Calibrar cámara	Como asesor, tengo que poder ver una vista previa de la cámara con una cruceta central que indique el centro del cuadrado de materiales.	92	Cuando el usuario acciona el mecanismo que activa el calibrado, se muestra la vista previa de la cámara en tiempo real con la cruceta central.	5	8	
10	Descartar fotos borrosas	Como usuario, quiero que el sistema descarte fotos borrosas para evitar un análisis erróneo.	90	Cuando se captura una imagen y se detecta como borrosa no se almacena en el sistema.	4	1	Debido a la estabilidad del soporte, será prescindible.
11	Segmentación de imágenes	Como usuario, quiero que el sistema sea capaz de segmentar las fotos capturadas en cuatro partes iguales para poder analizar la interacción con cada material.	90	Cuando se captura una imagen, no solo se almacena la imagen original, sino una imagen correspondiente a cada segmento (4 imágenes).	7	9	
12	Detectar interacción con materiales	Como usuario, quiero que el sistema sea capaz de determinar si se posiciona la mano (u otro objeto) sobre alguno de los materiales.	90	Partiendo de cuatro imágenes completas y limpias de los cuatro materiales (imágenes de referencia), el sistema es capaz de calcular y mostrar por consola el porcentaje de píxeles cambiantes para cada material al capturar una nueva imagen.	5	4	Este valor sera comparado con un umbral para determinar la interacción.
13	Reproducción del metrónomo	Como usuario, quiero poder activar la reproducción del metrónomo.	88	Cuando el usuario acciona el botón "play" del metrónomo éste se activa y empieza a sonar en cada parte del compás, omitiendo siempre el sonido de la última parte del compás.	6	9	
14	Parar el metrónomo	Como usuario, quiero poder desactivar el metrónomo.	88	Cuando el usuario acciona el botón "stop" del metrónomo se desactiva y deja de emitir cualquier sonido.	3	2	
15	Ajustar parámetros del metrónomo	Como asesor, quiero regular el BPM y el número de partes del compás.	86	Cuando se cambia el BPM o el número de partes del compás, debe reflejarse al reproducir el metronomo.	2	3	
16	Reproducción de sonidos de los materiales	Como usuario, quiero que el sistema emita el sonido asignado a un material cuando se detecte la mano (u otro objeto) sobre este material.	84	Con el juego activado (metrónomo encendido), si el usuario pasa la mano sobre algún material se debe emitir el sonido predefinido para este material en la última parte del compás.	2	2	Se considera interacción si el porcentaje de píxeles cambiantes supera un umbral.
17	Descarga automática de sonidos	Como usuario, quiero que el sistema descargue automáticamente los sonidos de los materiales actuales para no sobrecargar el almacenamiento con todos los sonidos.	82	Cuando se entra en el juego, se muestran varios mensajes que nos informan del proceso de descarga, indicando en última instancia que se descargaron correctamente y se ubicaron en la carpeta predefinida para los sonidos.	12	15	Incluye integración y creación de BD. Se deben encontrar los sonidos en la carpeta de sonidos predefinida para la aplicación.
					47	**54**	

Figura 4.5: Historias de usuario correspondientes al Sprint 1

.

De forma similar al *Sprint* anterior, disponemos de la gráfica *Sprint Burn Down*, en la que recogemos el avance del *Sprint* a lo largo de los días laborables de éste. Esta gráfica se encuentra disponible en la Figura 4.6.

Podemos observar, fruto de la subestimación, que el avance en horas (serie de color rojo) está prácticamente siempre por encima de lo estimado (serie de color azul). Todo esto se traduce en un sobreesfuerzo para llegar a acabar cada una de las historias de este *Sprint*.

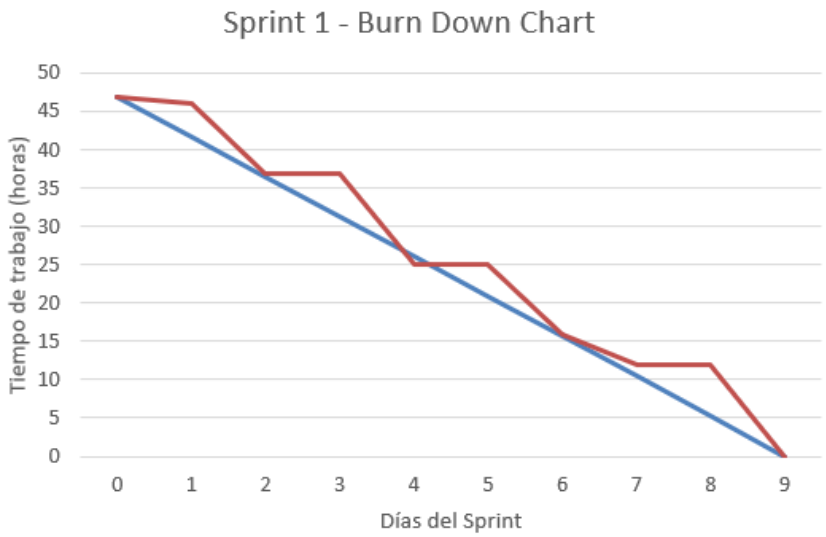

Figura 4.6: Gráfica que refleja el progreso diario del Sprint 1.

4.2.4 Sprint 2

Este *Sprint* 2 se ejecuta en el periodo de tiempo correspondiente entre el 16 de mayo y el 03 de junio (3 semanas). Lo que resta del mes de mayo entre la finalización del *Sprint* 1 y el comienzo de este *Sprint* se dedicó a revisar el *Sprint* 1 (revisión y retrospectiva del *Sprint*) así como a la planificación de este *Sprint* 2, el último *Sprint* planificado. Parte del tiempo se dedicó a plantear y remodelar el soporte para que la aplicación pueda funcionar debidamente. En este punto se amplió la pila de producto, añadiendo historias relacionadas con el enfoque de la cámara, con el icono de la aplicación o con requisitos de su diseño (interfaz de aplicación), entre otras.

Tal y como refleja la Figura 4.7, el factor de foco debe disminuir su valor a 0,55 al haber realizado las historias previstas con falta de tiempo (sobreesfuerzo). Si hubiéramos empleado justamente las 47 horas estimadas en el anterior *Sprint*, se obtendría, lógicamente, el mismo factor de foco, es decir, el valor 0,65 (operación: 47/73). En nuestro caso, excedimos el número de horas estimadas en 7 (54-47), por lo que el factor de foco final es el resultado de la operación: (47-7)/72, que es efectivamente 0,55.

De las 72 horas laborables de este *Sprint*, nos quedamos tan sólo con unas 40 horas al multiplicar por el nuevo factor de foco. Estás 40 horas estimadas deberían ser las cubiertas en este *Sprint* 2. No obstante, al ser el último *Sprint*, se ha tomado la decisión de incrementar el número de horas estimadas en 8, debiendo realizar en este *Sprint* un total de 48 horas.

Esta decisión se tomó para estimular la productividad, sabiendo que hipotéticamente realizamos las 40 horas con comodidad. Esto no es una idea que haya surgido *sin más*. En muchos proyectos reales se puede llegar a aplicar esta estrategia para estimular al equipo de desarrollo. En nuestro caso, además, se emplea para poder cubrir algunas de las nuevas historias añadidas a la pila de producto (actualizada) en este último *Sprint*.

Días	Horas/Día	Total Disponibilidad	Factor de Foco	Velocidad Estimada
9	8	72	0,55	40

Figura 4.7: Estimación de horas del Sprint 2.

Las historias que componen este *Sprint* pueden verse en la Figura 4.8. En general, las funciones más destacadas que conforman el motivo principal de este *Sprint* son las relacionadas con la integración de códigos QR para la descarga de sonidos y las relacionadas con la ayuda mediante videos.

El total de horas dedicadas a desarrollar estas funcionalidades fue de 61 horas, por lo que hemos subestimado demasiado este *Sprint*. Otro factor que ha propiciado este sea quizás la sobrecarga de funcionalidades que se añadieron a conciencia. El número de historias de usuario en color rojo es bastante notable. No obstante, este sobreesfuerzo lo creemos necesario al tratarse del último *Sprint* que daría como resultado el producto final. Este producto final queda expuesto a la solución de posibles *bugs* que se detecten en el futuro y a optimizaciones que se crean oportunas.

Las historias que han propiciado una subestimación tan alta han sido básicamente lo relacionado con las funcionalidades más diferenciadoras, que constituyen el motivo de este *Sprint* (códigos QR y ayudas mediante videos).

La gráfica *Sprint Burn Down* para este *Sprint* se puede apreciar en la Figura 4.9. Podemos observar, debido a la subestimación, que el avance en horas (serie de color rojo) está prácticamente siempre por encima de lo estimado (serie de color azul). Ocurre de forma bastante similar al *Sprint* anterior. El sobreesfuerzo para llegar a acabar cada una de las historias de este *Sprint* es también bastante evidente al observar la gráfica.

A las horas sumadas en cada uno de los *Sprints*, es necesario sumar las horas destinadas a la elaboración y prueba del soporte, la elaboración de los videos de ayuda y la elaboración de toda la documentación. Asimismo, tanto las horas de revisión y retrospectiva del *Sprint* como las destinadas a la planificación de estos también se consideran horas dedicadas al proyecto. En cuanto a las pruebas realizadas, se han basado principalmente en los criterios de validación de las historias, pues el proyecto es a nivel general, muy práctico, por lo que las pruebas se orientaron en este sentido.

4.2.5 Diagrama de casos de uso y especificación

En la Figura 4.10 se presenta el diagrama de casos de uso correspondiente a nuestra aplicación móvil. En cierta medida refleja de manera muy compacta los requisitos recogidos en las historias de usuario de cada uno de los *Sprints* comentados anteriormente.

Una vez presentado cada uno de los casos de uso de la aplicación, se explicarían cada uno de ellos mediante el uso de una plantilla de especificación de casos de uso. Quedarían claros conceptos relativos a estos casos de uso como lo son: rol que lo ejerce (visualmente en el diagrama), número y descripción del caso de uso, flujo normal de tareas en la realización del caso de uso (añadiendo variaciones, excepciones y extensiones del flujo), precondiciones y postcondiciones para su realización y un apartado de observaciones generales del caso de uso. Todos y cada uno de los casos de uso se presentan en las figuras incluidas en el anexo 5.

Grosso modo, las funcionalidades se pueden resumir o agrupar en dos grandes ámbitos según el rol que lo ejerce:

- Acción de jugar a la aplicación (es la función principal de esta, la más importante).

Id	Nombre	Descripción	Prioridad	Validación	Estimación	Velocidad real	Notas
18	Permisos de la aplicación	Como usuario, quiero que el sistema me solicite permiso de acceso a la cámara para evitar un mal funcionamiento.	78	Al entrar en la aplicación por primera vez, se muestra la solicitud de permisos de acceso a la cámara, pudiendo el usuario aceptarlo o rechazarlo (debe aceptarlo para que la aplicación siga funcionando).	2	2	Por revisar documentación.
19	Adaptación a cuadrícula de materiales	Como usuario, quiero ver en la pantalla de calibración de la cámara el dibujo de un cuadrado para que la imagen se adapte a la cuadrícula de materiales.	76	Cuando se abre la calibración de la cámara se muestra un cuadrado perfecto que se ajusta a la cuadrícula de materiales. Será el fragmento de imagen que se analice.	4	8	
20	Identificar códigos QR	Como asesor, quiero que el sistema sea capaz de leer codigos QR mediante la cámara para poder determinar qué sonidos descargar.	74	Cuando se toma una captura de búsqueda de codigos QR se detecta el código y se informa mediante mensajes.	9	8	Aclarando conceptos, nueva API. Alternamos flash para detectar.
21	Menú de opciones	Como asesor, quiero disponer de un menú de opciones para acceder a las distintas opciones del juego.	72	Cuando el usuario acciona el botón de menú se despliegan todas las opciones disponibles.	5	5	
22	Menú de ajustes	Como asesor, quiero poder ajustar todos los parámetros del juego para obtener una experiencia optimizada y adaptada del juego.	72	Cuando el asesor acciona la opción "Ajustes" dentro del menú de opciones se le redirecciona a una pantalla de ajustes donde puede configurar el metrónomo, el procesado de imágenes y la cámara.	4	6	Integrar llamadas a calibrar, parametros,... Muchas de las cosas ya hechas pero en un menu de ajustes apropiado.
23	Usar flash de la cámara	Como asesor, quiero ajustar si se usa o no flash en la calibración y en el juego.	70	Cuando en ajustes se activa o desactiva la opción de flash, se debe reflejar a la hora de calibrar/jugar.	2	2	Creamos cámara personalizada.
24	Ajuste de sensibilidad	Como asesor, tengo que poder ajustar la sensibilidad para garantizar que se detecta la interacción correctamente.	68	Al ajustar la sensibilidad debe de reflejarse en el juego, siendo más fácil/difícil que se detecten cambios en la imagen.	1	2	
25	Descarga de sonidos con codigo QR	Como asesor, quiero descargar el conjunto de sonidos asignados a un código QR para disponer de los sonidos necesarios para los materiales actuales.	68	Dado un código QR registrado en la base de datos, el sistema informa del proceso de descarga indicando en última instancia que la descarga fue exitosa. La descarga debe incluir siempre los sonidos por defecto (sonido del metronomo y de interacción errónea).	9	12	Implicaron un reajuste de la base de datos existente.
26	Habilitar/deshabilitar menú	Como asesor, quiero poder mostrar/ocultar el icono de menú para evitar que los jugadores se vean tentados a pulsarlo.	66	Al accionar la opción de habilitar/deshabilitar menú, cambia el estado de visibilidad del icono de menú (a visible si estaba invisible y viceversa).	2	2	
27	Evitar replica de códigos	Como asesor, quiero que mis códigos solo puedan ser usados por mí para evitar cualquier copia o plagio.	64	Al utilizar un código por primera vez el sistema informa de que fue registrado, pudiendo usarlo en posteriores ocasiones. Si otro usuario intenta jugar con tu código en otro dispositivo, se le informará y se le negará el juego con ese código.	2	1	Encriptamos el identificador del dispositivo para evitar tráfico de información "sensible".
28	Migrar códigos a otro dispositivo	Como asesor, tengo que poder migrar mis códigos a otros dispositivos para poder jugar en el dispositivo que quiera.	62	Al accionar la opción de desvincular códigos en el menú, se desvinculan todos los códigos asociados al dispositivo, recuperando el estado que tenían antes de usarse. Podrás registrarlo en otro dispositivo, dejando de funcionar en el antiguo.	2	1	Simple operación a base de datos.
29	Ayuda audiovisual (vídeos)	Como usuario, quiero disponer de una ayuda básica dentro de la aplicación.	60	Cuando el usuario acciona el icono de ayuda, se muestra un reproductor que reproducirá diferentes vídeos de ayuda, pudiendo navegar hacia el siguiente/anterior vídeo hasta encontrar el deseado (cíclico). Cuando acaba un vídeo se vuelve a reproducir.	2	5	El diseño supuso tantas horas.
30	Enfoque de la cámara	Como usuario, quiero que la cámara enfoque correctamente a cualquier superficie sin perder dicho enfoque.	60	Al acceder a la calibración de la cámara, debe enfocarse adecuadamente la superficie de materiales sin que se desenfoque en ningún momento.	2	4	Este problema se detectó en el anterior Sprint.
31	Icono de aplicación	Como usuario, quiero disponer de un icono simple y adaptado al contexto de la aplicación.	58	Al localizar la aplicación en el sistema debe de tener el icono diseñado específicamente para ella.	1	2	
32	Diseño intuitivo de pantalla de juego	Como usuario, quiero que la pantalla de juego se integre visualmente con el soporte para que el dispositivo parezca parte del soporte.	58	Al entrar en el juego, la pantalla principal tendrá como fondo una imagen capturada desde la parte superior del soporte, indicando que el móvil forma parte del soporte.	1	1	
					48	61	

Figura 4.8: Historias de usuario correspondientes al Sprint 2.

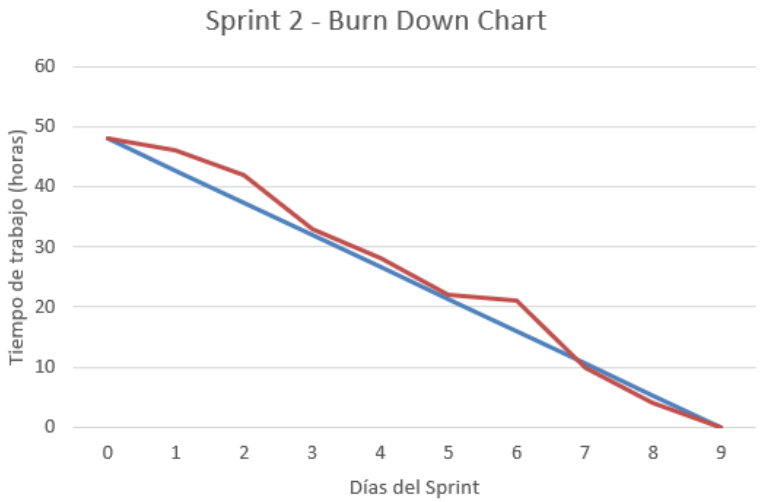

Figura 4.9: Diagrama de casos de uso de la aplicación.

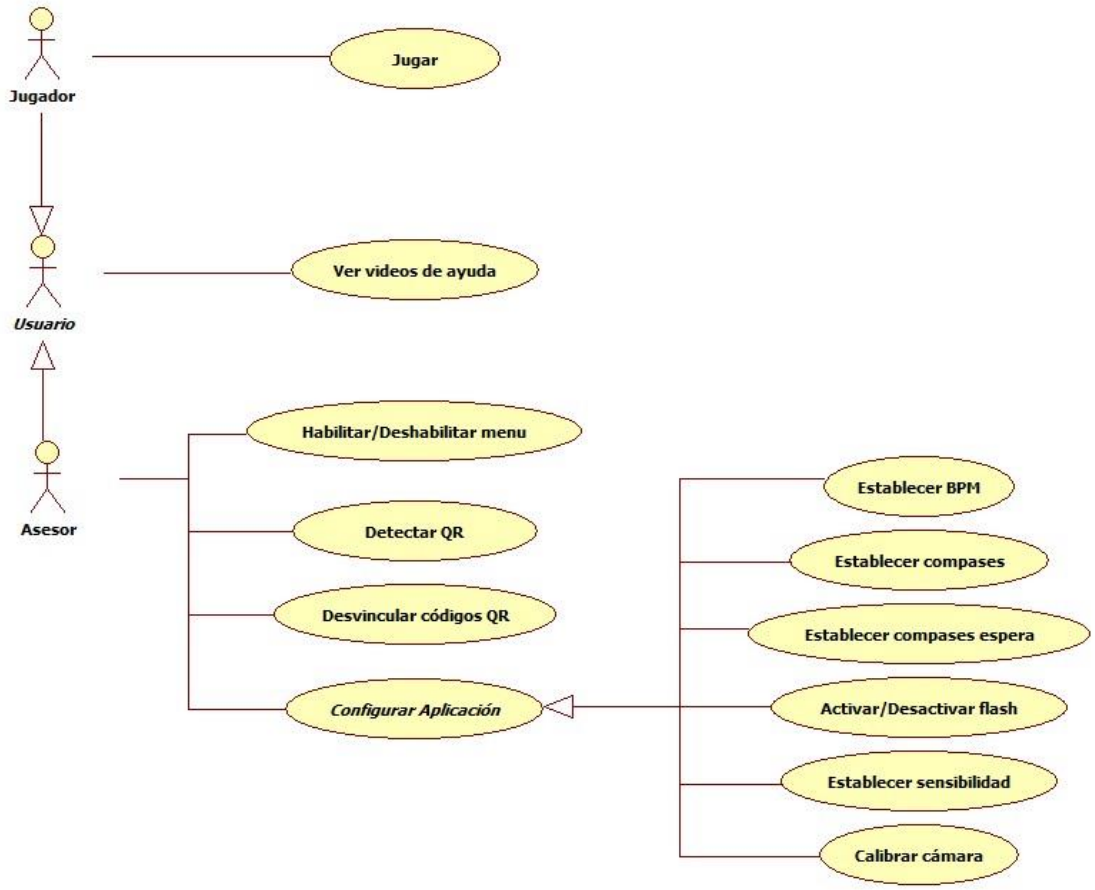

Figura 4.10: Diagrama de casos de uso de la aplicación.

- Poner a punto el juego. Entran todo el conjunto de funciones que tienen como misión fundamental personalizar los parámetros y configuraciones del juego para que no sólo funcione, sino que funcione de la mejor manera posible (habilitar juego mediante el calibrado y lectura del código QR de la lámina, parametrización del juego...). Es aquí donde

vemos cobrar importancia al rol del asesor, sin el cual no sería posible realizar la funcionalidad principal, jugar.

4.2.6 Diagrama de actividades

Ya tenemos definidas las funcionalidades de la aplicación, tanto en historias de usuario (usadas para la planificación del proyecto) como en casos de uso con sus especificaciones (para aglomerar el conjunto de funcionalidades de la aplicación). Sin embargo, falta un componente bastante importante, que viene a ser el flujo normal de uso de la aplicación, es decir, la sucesión de actividades realizadas cada vez que deseamos usarla. Esto podría considerarse parte de la lógica de negocio de la aplicación (cómo funciona el negocio, en este caso, nuestra aplicación). Para ello, hemos reflejado esto de manera muy simple e intuitiva en un diagrama de actividades, disponible en la Figura 4.11.

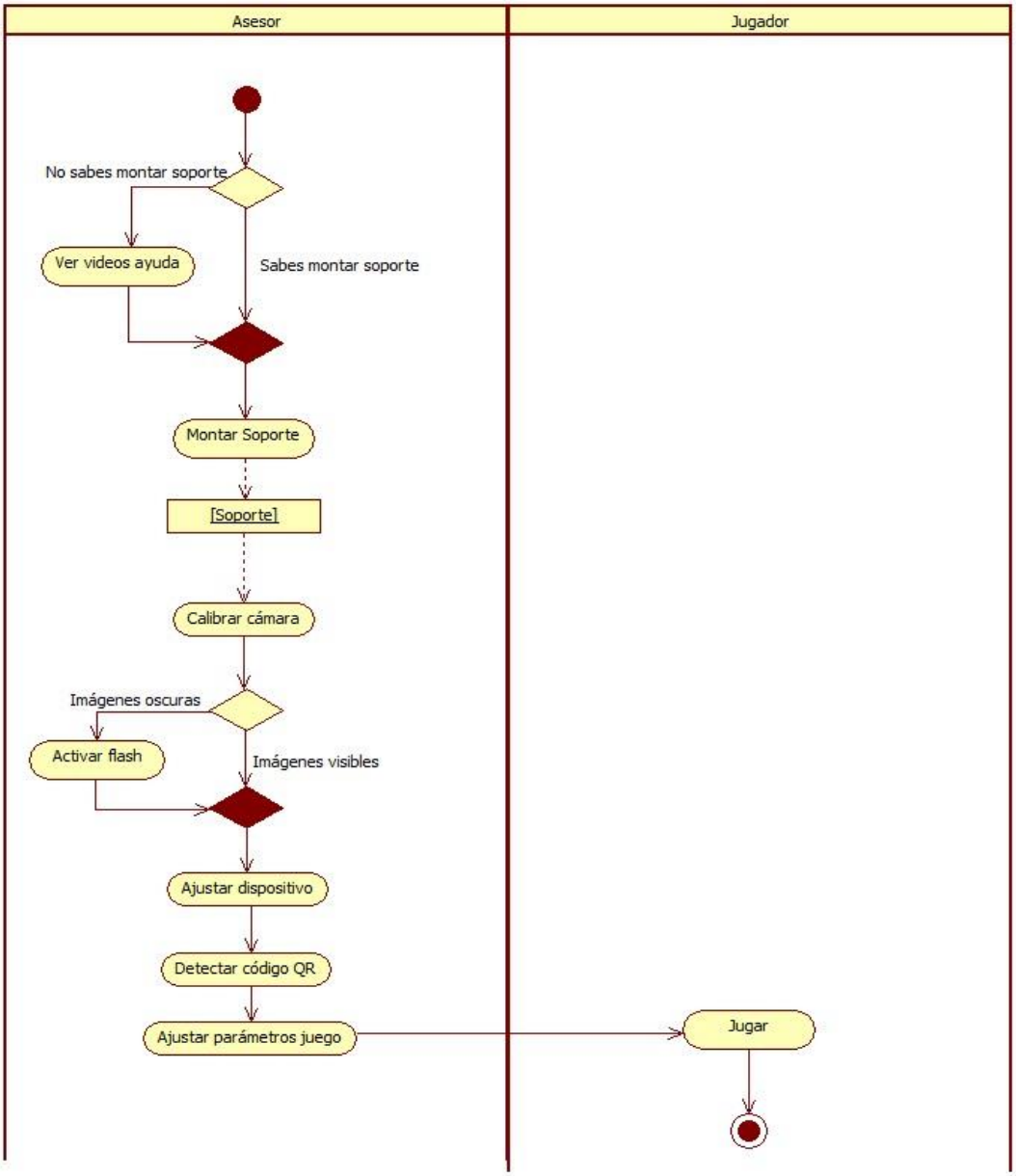

Figura 4.11: Diagrama de actividades (flujo de actividades realizadas al usar nuestra aplicación).

En general, abstrayendo el flujo principal de tareas de nuestra aplicación se comprenden mejor las funcionalidades que cubre. Lo único que quizás conviene detallar es la tarea *ajustar dispositivo*, que simplemente se refiere a mover el dispositivo hasta que, con ayuda de la calibración de la cámara recién habilitada, logramos ubicarlo en el sitio correcto (enfocando la superficie de materiales).

4.3 Organizativa

Una vez visto el factor funcional del proyecto, toca introducir la parte organizativa. En esta parte se incluye información sobre los componentes físicos que juegan algún papel en el funcionamiento de la aplicación, la estructuración del código en clases y la base de datos.

En primer lugar, debemos recaer sobre los componentes físicos implicados en el funcionamiento de la aplicación, así como los componentes software que incluyen. Esto lo podemos apreciar de forma muy simplificada en la Figura 4.12.

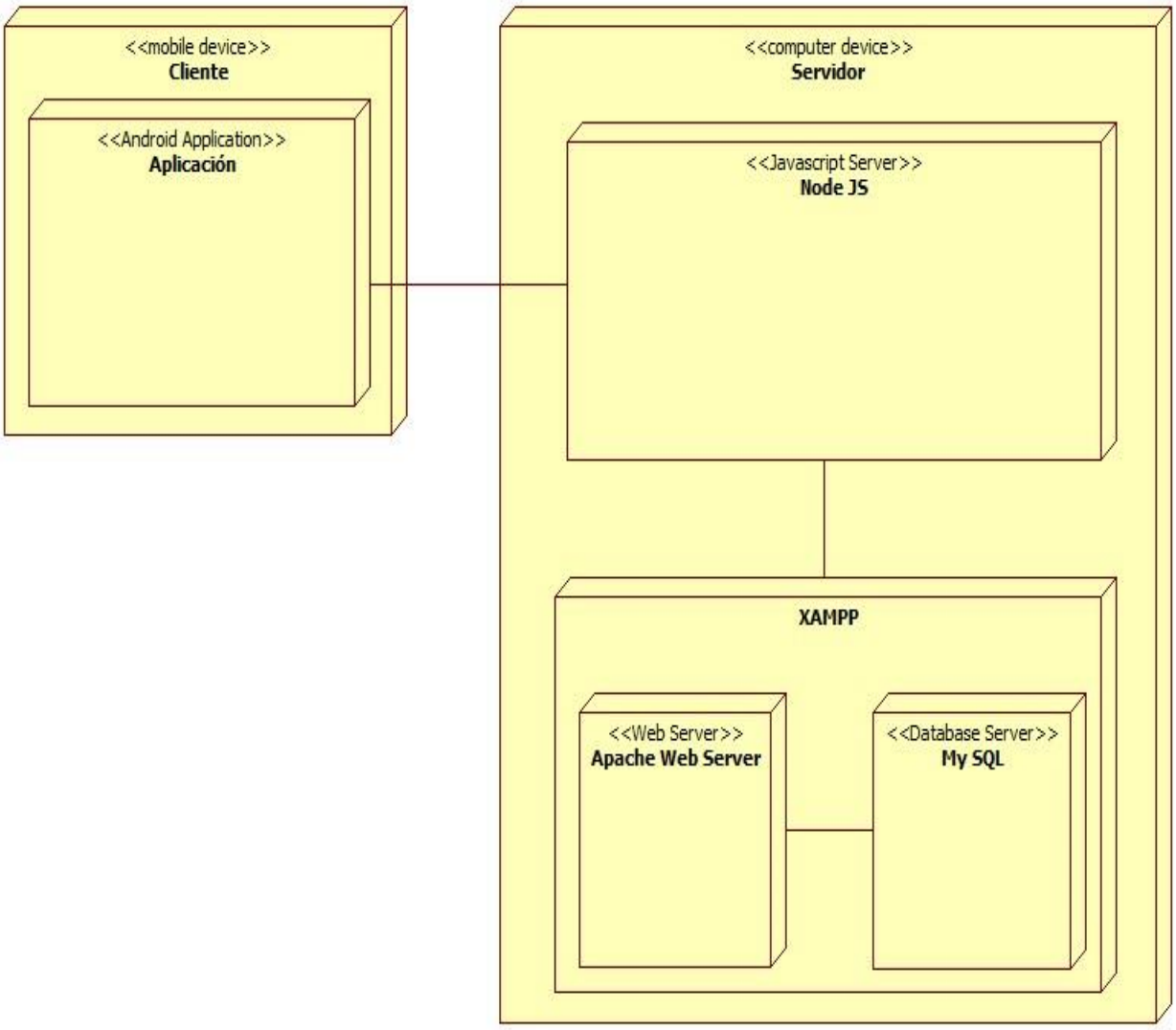

Figura 4.12: Diagrama de despliegue (minimalista, con componentes clave).

4.3.1 Diagrama de clases

A continuación, mostramos un diagrama de clases con cierto nivel de abstracción para reflejar las clases o conceptos principales que participan en nuestra aplicación a nivel de código. Como vemos en posteriores apartados, realizamos algunas modificaciones en el diagrama para adaptarlo a la estructura y características de la plataforma y lenguaje sobre la que trabajamos durante todo el desarrollo (Android). Este diagrama está en la Figura 4.13.

Como vemos, los principales componentes conceptuales de nuestra aplicación son:

- Clase controladora del juego (`Game`): es la clase principal en torno a la que gira la mayor parte de la lógica. Su función es cambiar el juego de estado (activarlo o desactivarlo). Cuando se detecta un evento (del usuario), se llama al método `turnOnGame` encargado de activar el juego. Opuesto a él tenemos el método `turnOffGame` encargado de desactivar el juego. Esto es solo posible cuando el campo `gameState` es verdadera, pues nos indica si el juego ha sido activado para jugarlo.
- Clases de control del sonido (`Metronome`, `PlaySound`): son las encargadas de llevar el ritmo (velocidad) de los sonidos y de emitirlos, tanto relativos al compás como a la interacción con los supuestos materiales. Contiene el grupo de campos que determinan su comportamiento (`bpm`, `tempo` y `waitTempo`). Contiene dos métodos invocados por la clase `Game` para activar (método `turnOnMetronome`) y desactivar su reproducción (método `turnOffMetronome`). El campo `state` es el encargado de reflejar el estado de activación del metrónomo. La clase `PlaySound` (método `playSound`) simplemente se encarga de reproducir de forma asíncrona el sonido correspondiente.

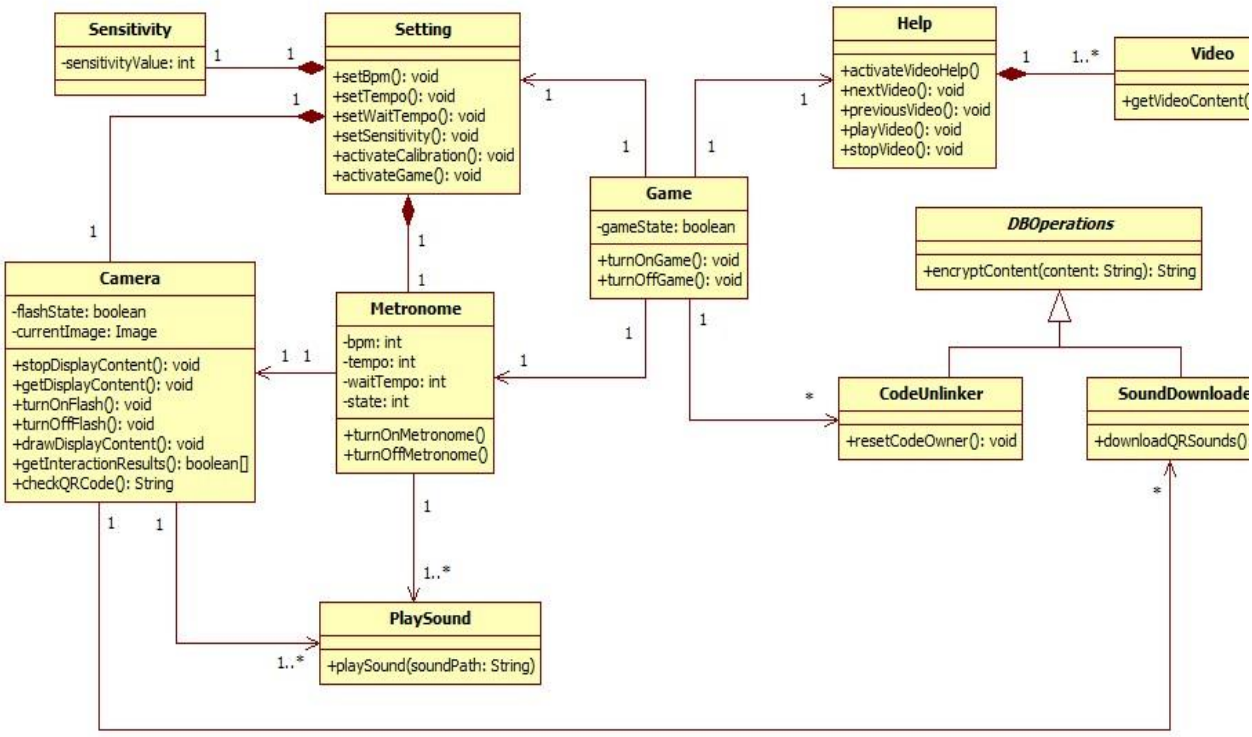

Figura 4.13: Diagrama de clases a nivel organizativo (obviando detalles de implementación particular del lenguaje empleado).

- Clases de control de imágenes (Camera): se encarga de todas las operaciones relacionadas con las imágenes de la cámara, tales como iniciar/parar de visualizar imágenes de la cámara a la hora de calibrarla (métodos `getDisplayContent` y `stopDisplayContent`), dibujar sobre las anteriores imágenes (método `drawDisplayContent`), procesar en busca de interacción (método `getInteractionResults`), comprobar códigos QR en las imágenes (método `checkQRCode`) y activar/desactivar el *flash* (métodos `turnOnFlash` y `turnOffFlash`).
- Clases relacionadas con el ajuste de parámetros (`Setting`, `Sensitivity`): en el caso de la clase `Setting` se encarga de actuar de interfaz de configuración de parámetros de juego. Los métodos más diferenciadores son `activateCalibration` y `activateGame`, que se encargan respectivamente de activar la vista previa de las imágenes de la cámara y de activar la detección de códigos QR para activar el juego (poder jugarlo). Para estas tareas necesita comunicarse con la cámara llamando a los métodos oportunos, ya comentados previamente. Los otros métodos de la clase `Setting` simplemente establecen los valores para los parámetros de juego correspondientes al metrónomo y a la sensibilidad (`sensitivityValue` de la clase `Sensitivity`).
- Clases relacionadas con el los vídeos de ayuda (`Help`, Video): controlan los aspectos de la pantalla de ayuda del usuario. La clase `Help` contiene uno o más objetos de la clase Video. A su vez, la clase `Help` contiene métodos que actúan en el control de reproducción de vídeo.
- Clases de acceso a la base de datos (`CodeUnlinker`, `SoundDownloader`): estas clases heredan a su vez de una clase abstracta, pues comparten la operación de encriptar datos. La clase `CodeUnlinker` se encarga de desvincular códigos del dispositivo actual de la base de datos, mientras que la clase `SoundDownloader` se encarga de descargar los sonidos después de que la clase Camera haya detectado un código QR correctamente.

4.3.2 Diagramas de secuencia

Comentamos muy brevemente estas clases desde un punto de vista dinámico, no estático (anterior diagrama de clases). Para ello mostramos diferentes diagramas de secuencia correspondientes a cada acción relevante dentro de la aplicación. Es ideal desglosar la aplicación en funciones para apreciar de una manera más clara la participación de componentes en cada una de las tareas, puesto que el *flujo general de uso de la aplicación* se muestra en el diagrama de actividades ubicado en el anterior apartado.

En la Figura 4.14 podemos observar el flujo de interacción principal para activar y desactivar el juego. Este diagrama de secuencia es el más relevante de todos. Debemos de tener en cuenta para entender el diagrama que todas las acciones relacionadas con el metrónomo, la cámara y los sonidos se realizan de forma asíncrona, pudiendo ser interrumpidas por eventos.

En este diagrama se muestra que, en primer lugar, (1) tras un evento activado por el usuario, se llama al método `turnOnMetronome` para activar el metrónomo (clase `Metronome`), en definitiva, el juego. Después de eso se entra en un bucle del que no se sale hasta que un evento de usuario dispare el método `turnOffMetronome`.

Durante la ejecución del bucle, (2,3) si el metrónomo no se encuentra en el último pulso del compás, simplemente emite el sonido de pulso habitual llamando al método `playSound`. Por el contrario, (4) si se encuentra en el último pulso de compás, detecta si hay interacción llamando al método `getInteractionResults` de la clase *Camera* que, (5,6) si detecta interacción, llama al

método `playSound` para emitir el sonido correspondiente. Finalmente (7,8), cuando se detecte el evento de usuario, se llama al método `turnOffMetronome` y se desactiva la reproducción del metrónomo, en definitiva, el juego.

Antes de poder jugar, como vimos en el pasado diagrama de actividades, es necesario calibrar la cámara y activar el juego (lectura de código QR). Para ello, se presenta la secuencia de acciones tanto para calibrar la cámara (Figura 4.15) como para activar el juego (Figura 4.16).

En la Figura 4.15 podemos ver que en primer lugar (1) se llama al método `activateCalibration` para acceder a la opción de calibrar la cámara dentro de los ajustes. Fruto de esta llamada, (2) se abre la visualización de la cámara llamando al método `openCamera`. Después de que (3) la cámara dibuje sobre sus imágenes en tiempo real las figuras necesarias para la calibración llamando al método `drawDisplayContent`, (4) se muestra en pantalla la vista actual de la cámara con estos dibujos. Finalmente, (5,6,7) cuando se termine de calibrar la cámara se vuelve al estado previo a llamar a la calibración de la cámara mediante la llamada al método `stopDisplayContent`.

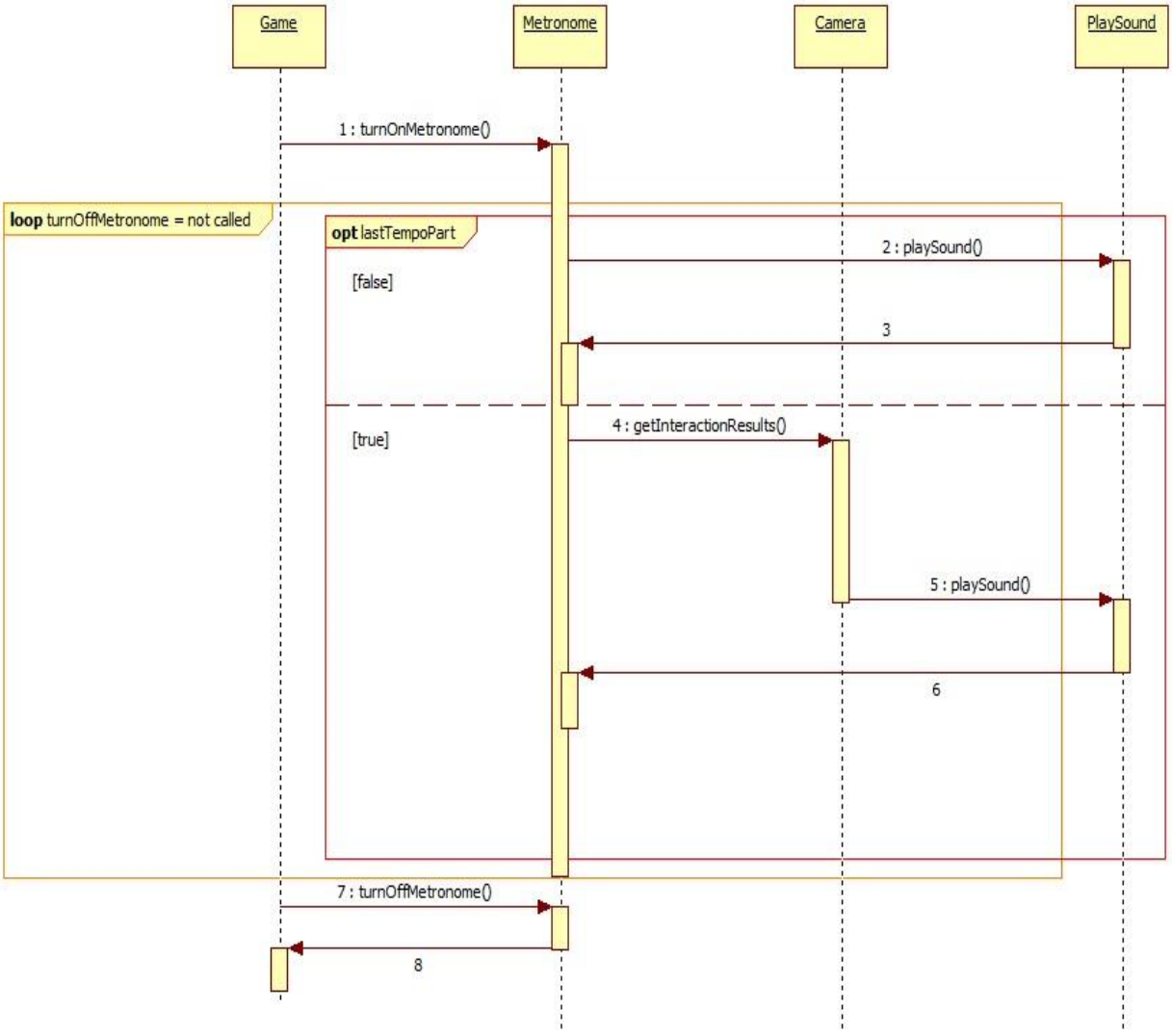

Figura 4.14: Diagrama de secuencia (Jugar).

En la Figura 4.16 podemos ver que primeramente (1) se llama al método `activateGame` para iniciar el escaneo en búsqueda de códigos QR a través de la cámara. Luego, (2) desde el objeto de la clase `Setting`, se llama al método `checkQRCode` para realizar el escaneo, propiamente. Si se estima dentro del método el valor de un código QR, (3) se procede a la descarga de sonidos asociados mediante la llamada al método `downloadQRSounds` del objeto `SoundDownloader`, que, a su vez, (4) se encarga de encriptar la información al acceder a la base de datos mediante una llamada al método de la clase abstracta de la que hereda, es decir, el método `encryptContent`. Finalmente, (5) se vuelve al estado previo a la inicialización de la detección de códigos QR.

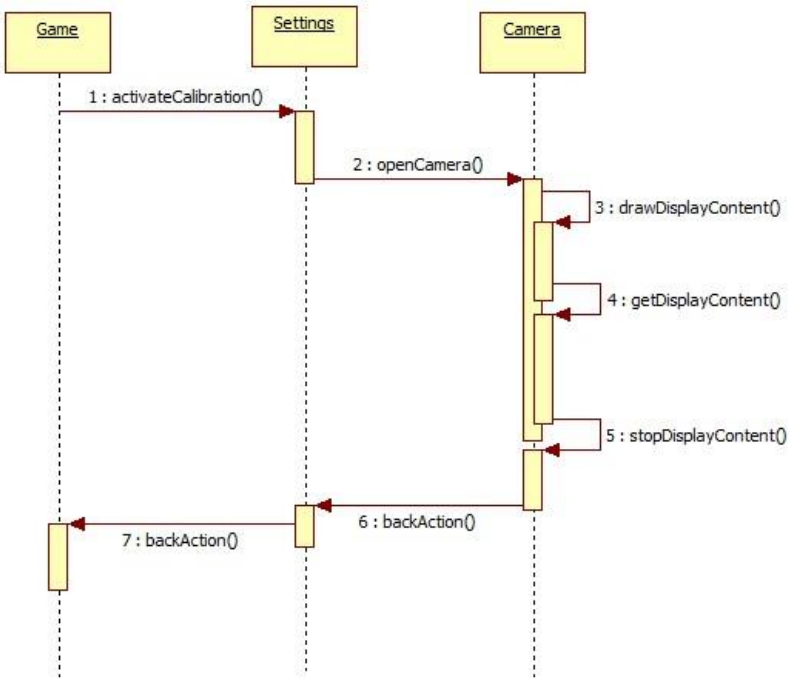

Figura 4.15: Diagrama de secuencia (Calibrar cámara).

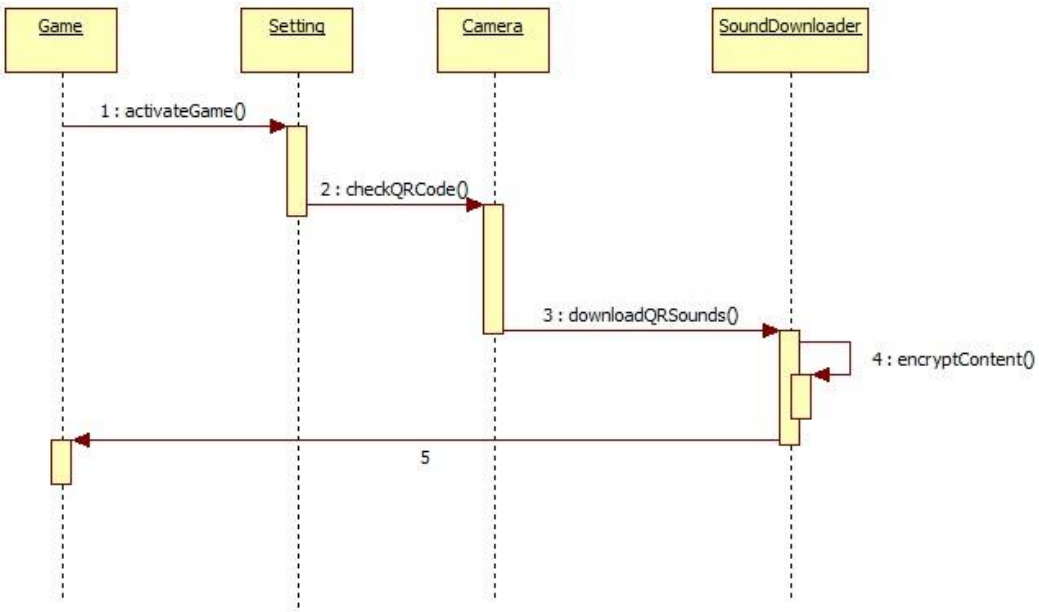

Figura 4.16: Diagrama de secuencia (lectura de código QR para activar el juego).

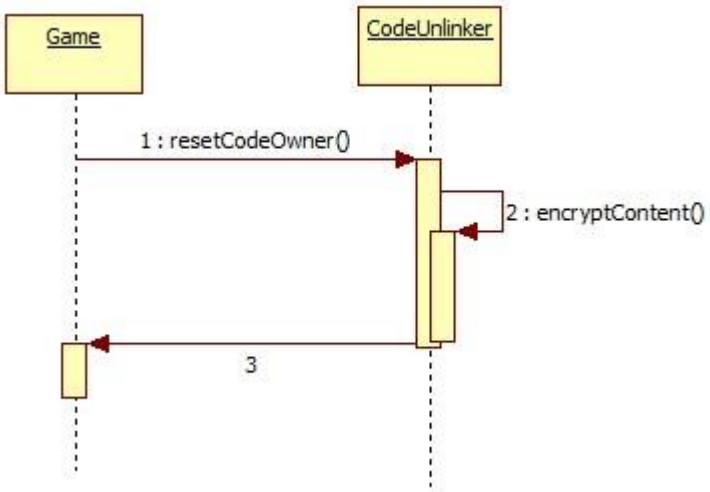

Figura 4.17: Diagrama de secuencia (desvincular códigos QR).

Así como presentamos la activación mediante la lectura de códigos QR, visualizamos en la Figura 4.17 el diagrama de secuencia para desvincular códigos previamente vinculados. Esta acción es muy simple, pues consiste en activar una operación contra la base de datos. Simplemente, (1) se llama al método `resetCodeOwner` del objeto `CodeUnlinker` para que, (2) una vez encripte la información para acceder a la base de datos, borre al dispositivo actual como propietario de cualquier código registrado a su nombre. Al final, (3), se regresa al estado inicial antes de disparar esta funcionalidad.

Sólo nos faltaría por cubrir la sucesión de acciones de lo relacionado con la configuración de la aplicación y con la ayuda (mediante videos), mostradas respectivamente en las figuras 4.18 y 4.19.

La primera secuencia (Figura 4.18) es bastante simple. Tras sucesivas llamadas a los métodos *`setter`* del objeto de la clase `Setting`, (1,3,5,7) se establece el valor de configuración para los diferentes parámetros y (2,4,6,8) se recibe la confirmación de haberse realizado correctamente.

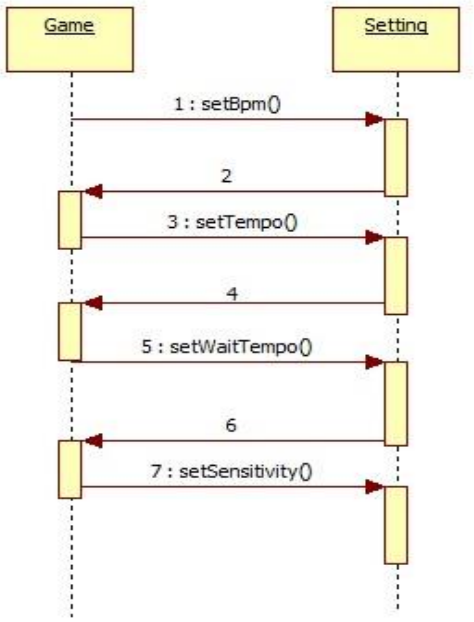

Figura 4.18: Diagrama de secuencia (Ajustes de la aplicación, diferentes parámetros).

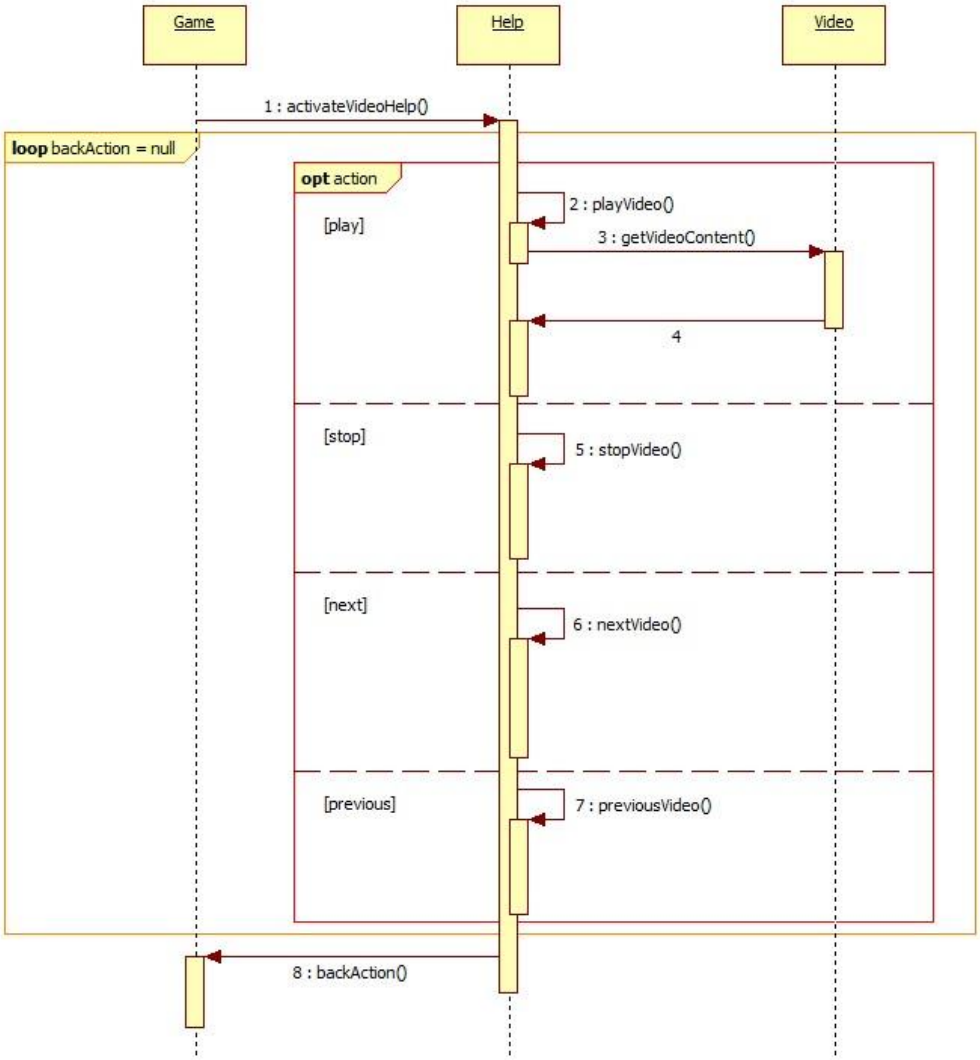

Figura 4.19: Diagrama de secuencia (ayuda mediante videos).

La segunda secuencia (Figura 4.19) consiste en, primero (1) activar o acceder mediante un evento disparado por alguna acción del usuario a la pantalla de ayuda, llamando al método `activateVideoHelp`. Luego, se entra en un bucle en el que pueden darse cuatro condiciones distintas o ninguna de ellas. (2) Se dispara la acción de reproducir un video y se llama al método `play`. *A*cto seguido, (3,4) se obtiene el contenido de este llamando al método `getVideoContent` del objeto de la clase *Video.* Para el resto de posibles operaciones, (5) se llama al método `stop` para parar el video, (6) al método `next` para pasar al siguiente video o (7) al método `previous` para pasar al anterior video. Finalmente, tras la una acción predefinida en la aplicación, definida arbitrariamente como `backAction`, (8) se vuelve a la pantalla de juego.

Como hemos observado, las tareas que quizás requieren de mayor complejidad a nivel de interacción de objetos es la acción principal de la aplicación, jugar, así como lo relacionado con la ayuda mediante videos, pues sin ser la actividad de jugar, es aquel aspecto de la aplicación que varía de forma considerable respecto al resto. Con esto último queremos decir que la mayoría de las componentes del diagrama de clases juegan un papel básico en el juego, no a nivel individual.

4.3.3 Base de datos

En cuanto a la base de datos se refiere, es bastante simple, pues únicamente se emplea para cubrir tareas como: alojamiento de sonidos en un lugar ajeno al dispositivo móvil cliente, establecer la asociación entre sonidos y materiales para cada una de las láminas y registro de la propiedad de una lámina de materiales concreta (en definitiva, su código QR asociado). El diagrama de la base de datos se muestra en la Figura 4.20.

En este diagrama visualizamos dos tablas, una correspondiente a los códigos QR (`sheet_code`) y otra correspondiente a los sonidos de juego (`game_sound`). Los campos de cada una de las tablas se muestran a continuación.

Para la tabla `sheet_code` tenemos los siguientes campos:

- `id` (clave primaria): actúa de identificador único para cada registro de la tabla. Es un valor entero y autoincremental.
- `code`: almacena el valor del código QR correspondiente. Es un valor entero y con una restricción de unicidad (con un índice asociado para ello), pues tener dos códigos iguales en esta tabla es ilógico.
- `owner`: almacena la cadena de caracteres identificativa del dispositivo propietario del código QR. Naturalmente es un valor de tipo `varchar`.
- `type`: indica el tipo de código, que al final es una clave ajena que direcciona al conjunto de sonidos descargables en la otra tabla de la base de datos (`game_sound`).

Para la tabla `game_sound` tenemos los siguientes campos:

- `id` (clave primaria): actúa de identificador único para cada registro de la tabla. Es un valor entero y autoincremental.
- `name`: almacena el nombre asociado al sonido en cuestión. Es naturalmente de tipo `varchar`. Un ejemplo clásico para este campo sería el nombre de un acorde (*domayor*, por ejemplo).
- `audio_file`: contiene el sonido que se reproduce. Es de tipo `Blob`.
- `quadrant`: partiendo de un modelo de lámina concreta, este campo indica a qué cuadrante de la lámina pertenece el sonido actual. Es un valor entero que cubre valores de 1 a 4 (o de -1 a -4 en caso de sonidos no asociados a materiales, esto último por simple arbitrariedad).
- `type`: indica el tipo de lámina de materiales a la que pertenece este sonido.

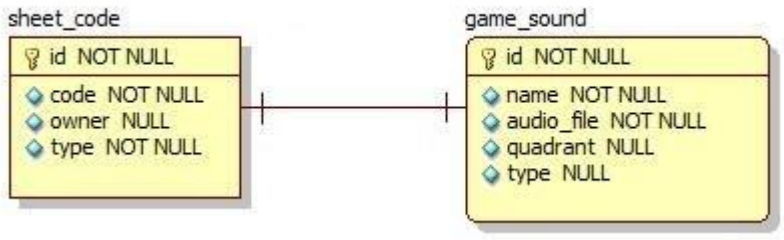

Figura 4.20: Diagrama de la base de datos.

Una restricción adicional en esta última tabla se le asigna al par de campos `quadrant` y `type`. El tipo de restricción es de unicidad, pues naturalmente, sólo debe de haber 4 sonidos (cuadrantes de 1 a 4) por cada tipo de lámina (número entero arbitrario, en principio, incremental). Para establecer esta restricción se hace uso de un índice.

4.4 Implementación

En esta última parte del capítulo presentamos detalles mucho más concretos del apartado de implementación de la aplicación. Básicamente incluimos las operaciones configuradas en el servidor de Node.js y la estructura final (más detallada) del diagrama de clases, ahora sí, mejor adaptado a la arquitectura de los sistemas Android.

4.4.1 Diagrama de clases

A la hora de desarrollar la aplicación, se adaptó ligeramente el diagrama de clases para adaptarlo al entorno en el que se programa y al desarrollador que lo lleva a cabo. Debemos tener en cuenta que el diagrama de clases presentado en el anterior apartado contiene, como se ha comentado en alguna ocasión, cierto nivel de abstracción en cuanto a la implementación real se refiere. La idea de esto es poder adaptar los conceptos que encabezan el funcionamiento de la aplicación a cualquier implementación que se decida realizar de esta.

Centrándonos en nuestra aplicación, implementada en Android (nativo), el diagrama de clases que recoge los conceptos de la aplicación es la mostrada en la Figura 4.21. A continuación mostramos algunas anotaciones de interés relativas a este diagrama.

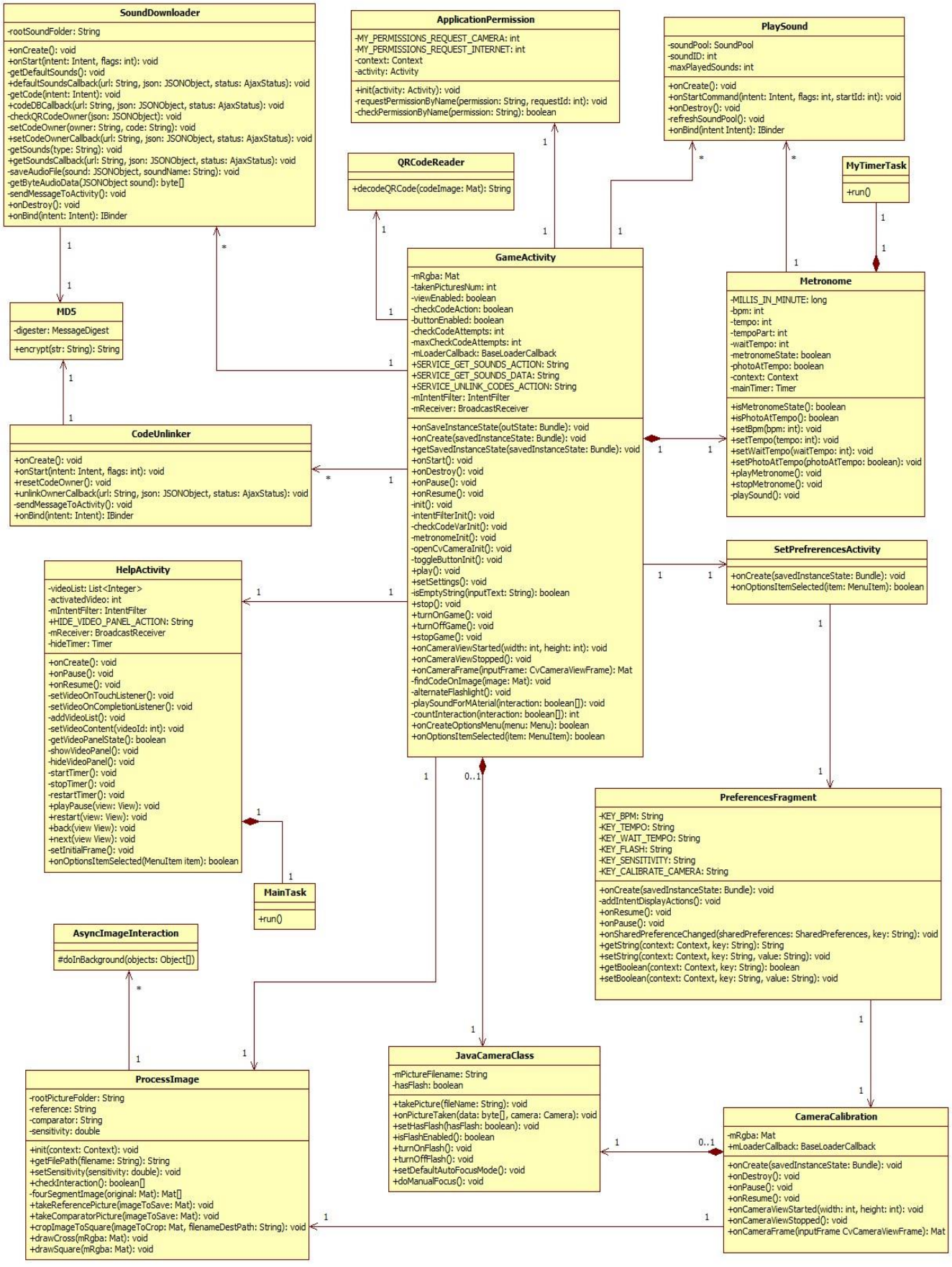

Figura 4.21: Diagrama de clases de la aplicación implementada en Android.

Pasamos a describir los roles y acciones principales de cada una de las clases del diagrama de la Figura 4.21. Naturalmente, no entramos en detalle con todas y cada una de las funciones, pues muchas de las clases contienen un gran número de métodos y campos, no obstante, los más relevante son:

- Cada una de las clases relativas a actividades (*activities*) y a servicios (*services*), que son clases muy particulares en Android que componen, respectivamente, pantallas de la aplicación y servicios/tareas en segundo plano, contiene métodos heredados como lo son: `onCreate`, `onStart`, `onPause`, `onResume`, `onDestroy`, `onBind`. Simplemente contienen código que se ejecuta cuando se crea, se inicia, se pausa, se reanuda, se destruye o se enlaza con una actividad o un servicio.

- La clase `GameActivity` es la parte lógica de la pantalla principal de juego. Está compuesta de las partes más importantes de la aplicación como un objeto de la clase `Metronome`, un objeto de la clase `JavaCameraClass`. La clase `GameActivity` se encarga de inicializar el metrónomo (método `metronomeInit`), la cámara (método `openCvCameraInit`), la interfaz de juego (método `toggleButtonInit`), las variables relativas a la detección de códigos QR (método `checkCodeVarInit`) y los ajustes de juego (método `setSettings`).

  Contiene otros métodos que se encargan de poner el juego en marcha (método `play`) y pararlo (método `stop`), activar el juego para poder jugarlo (método `turnOnGame`) así como poder desactivarlo (método `turnOffGame`), activar la búsqueda de códigos QR (método `findCodeOnImage`), reproducir un sonido cuando se detecta interacción (método `playSoundForMaterial`), entre otros. Todos los métodos que comienzan con `onCamera` son métodos heredados de la librería `OpenCV` para el control de la cámara y las imágenes provenientes de ella. También integramos métodos relacionados con el menú como `onCreateOptionsMenu` (encargado de inicializar el menú) y `onOptionsItemSelected` (encargado de establecer las acciones para los elementos del menú).

  En cuanto a los campos más importantes se encargan de llevar un control sobre los intentos de identificación de códigos QR (`checkCodeAttempts` y `maxCheckCodeAttempts`) antes de encender/apagar el *flash* para intentarlo de nuevo (método `alternateFlashlight`) y sobre el estado de elementos de la vista (`viewEnabled`, `buttonEnabled`).

- La clase `Metronome` simplemente contiene los campos que determinan el funcionamiento del metrónomo, por lo que, por simpleza, no entramos en esos detalles. Sí comentamos el campo `photoAtTempo`, pues es la encargada de indicar a la clase `GameActivity` si nos encontramos en el último pulso del compás del metrónomo, en cuyo caso se procesaría la interacción, actuando en consecuencia (emitiendo sonidos o no). Los métodos de esta clase simplemente se encargan de consultar el estado del metrónomo (método `isMetronomeState` y método `isPhotoAtTempo`), de establecer su configuración (métodos `setters`) y de encenderlo (método `playMetronome`) y apagarlo (método `stopMetronome`). El método `playSound` se encarga de reproducir el sonido usando la clase `PlaySound`.

- De la clase `PlaySound` debemos destacar que está compuesta por un objeto de tipo `SoundPool` (campo `soundPool`). Este objeto es el encargado de emitir los sonidos en el dispositivo, por lo que es importantísimo para nuestro proyecto. Existen más objetos similares, pero para sonidos de corta duración esta clase es la más adecuada.

- Entrando en las clases relacionadas con la cámara y el tratamiento de imágenes. Una de ellas es la clase `JavaCameraClass`, que no es más que una implementación personalizada de la cámara para realizar acciones no permitidas por la cámara implementada por `OpenCV`, como controlar el *flash* (métodos `setHasFlashEnabled`, `isFlashEnabled`,`turnOnFlash` y `turnOffFlash`) o controlar el modo de enfoque de la cámara (métodos `setDefaultAutoFocusMode` y `doManualFocus`). Todos estos métodos son bastante evidentes, aunque conviene recalcar la función del método `doManualFocus`, que simplemente se encarga de realizar un enfoque en la cámara (acción equivalente al pulsar con el dedo en una zona de la cámara cuando vamos a realizar fotos, para enfocar a un determinado objeto).

  Para el procesado de imágenes se aislaron las operaciones en la clase `ProcessImage`. La clase `GameActivity` y la clase `CameraCalibration` usan los métodos de esta clase para todas las tareas ya comentadas: comprobar si existe interacción (método `checkInteraction`), tomar una foto de referencia al comienzo del juego (método `takeReferenceImage`), tomar las fotos a comparar con esta última (método `takeComparatorPicture`), cortar la imagen en forma de cuadrado para evitar irregularidades en la comparación (método `cropImageToSquare`), segmentar la imagen a comparar en cuatro fragmentos iguales (método `fourSegmentImage`) y dibujar diferentes formas en la visualización en tiempo real de la cámara (métodos `drawCross` y `drawSquare`). Para el procesado de imágenes se realiza mediante una clase auxiliar (`AsyncImageInteraction`) que procesa los cuatro fragmentos de imagen (cuatro materiales) de forma totalmente asíncrona y paralela.

- Las clases `SetPreferencesActivity` y `PreferencesFragment`, no tienen demasiado jugo que extraer para esta explicación. Simplemente contienen los valores de configuración establecidos en la pantalla de ajustes de la aplicación. Estos valores se mantienen en la aplicación aun cuando esta se cierra, siendo consultados por la clase `GameActivity` para inicializar los valores establecidos para el juego. Estos valores tienen una referencia identificativa única (campos *finales* en la clase `PreferencesFragment`).

- Igual ocurre con la clase `CameraCalibration`, es la parte lógica de la pantalla de calibración y de lo único que se encarga es de mostrar las imágenes de la cámara en tiempo real aplicando previamente los dibujos de la cruceta y del cuadrado (necesarios para calibrar correctamente) llamando a los ya comentados métodos de la clase `ProcessImage`.

- La clase `HelpActivity` se encarga del apartado de ayuda mediante videos. Para ello, contiene de forma predefinida los videos en una lista (campo `videoList`). Los videos se encuentran como recursos (*resources*) de la aplicación, por lo que realmente esta lista contendría los identificadores de cada recurso de video. Sin entrar demasiado en detalles de implementación, esta clase contiene los métodos necesarios para: (1) realizar todas las operaciones de reproducción de video tales como reproducir/parar el video (método `playPause`), reiniciar el video (método `restart`), pasar al siguiente video (método `next`) y pasar al anterior video (método back), (2) establecer el Frame inicial a mostrar cuando el video no ha comenzado su reproducción (método `setInitialFrame`) y (3) controlar la visualización del panel de control de video sobre el propio video, parecido a como lo hacen aplicaciones como *Youtube* (métodos `showVideoPanel` y `hideVideoPanel`, ambos controlados por un *timer* y diversos eventos también incluidos en esta clase).

- La clase `SoundDownloader` se encarga básicamente de realizar la operación de descarga de sonidos de la base de datos una vez que la clase `GameActivity` dé por hecho la detección de un código QR. Los sonidos se descargan en la ruta inicializada en el campo `rootSoundFolder`. No se entraría en detalle de todos y cada uno de los métodos, simplemente comentar que tenemos un método que actúa a modo de *callback* para cada una de las operaciones realizadas sobre la base de datos. Por ejemplo, para la operación de descarga de sonidos por defecto (compás y sonido de error) tenemos el método `getDefaultSounds` que realiza una llamada a base de datos. La respuesta la recibiría de forma asíncrona el método `defaultSoundsCallback`, procediendo con el tratamiento de datos de la respuesta.
- La clase `codeUnlinker` es similar a la clase `SoundDownloader`, solo que incluye la operación de desvincular los códigos en la base de datos. Ambas clases dependen de la clase MD5, para encriptar la información.
- La clase `QRCodeReader` simplemente aísla la operación de obtener el código QR pasada una imagen desde la clase `GameActivity`, devolviendo como resultado una cadena (*String*) con el código.
- La clase `ApplicationPermission` contiene variables finales con los números identificadores de los permisos a solicitar por la aplicación. El más importante es el permiso de la cámara. Al instanciar un objeto de esta clase se solicita al usuario los permisos si no se ha hecho previamente.

Debemos comentar algunos aspectos, diferenciadores o no, que a simple vista podemos apreciar en este diagrama de clases (Figura 4.21):

- Al encontrarnos programando en Android, podemos observar las denominadas actividades (*activities*), en este caso para las pantallas de juego, de ajustes, de ayuda y de calibración (esta última un tanto peculiar, pues integra una simple visualización en tiempo real de la cámara).

  En Android, una actividad (*activity*) representa una pantalla de la aplicación, para la cual tenemos una parte lógica (*controller*) y una parte de vista (*view*). Las vistas asociadas a estas actividades (*activities*) no se incluyen en el diagrama, pues Android crea directamente un archivo XML vinculado a la parte lógica. De esta manera para cada vista tenemos asociado un controlador. Las vistas se pueden editar simplemente configurando el archivo XML directamente o mediante alguna interfaz gráfica (como la que Android Studio incluye).

  Las vistas XML son prácticamente tontas, es decir, simplemente contienen la información que se muestra, siendo los controladores los encargados de inyectar la información necesaria. Aunque esto puede no ser así, en nuestra aplicación lo es. De hecho, una de las acciones que actualiza la vista de la pantalla de juego principal, que viene siendo la acción de habilitar/deshabilitar menú, se lleva a cabo por completo desde el controlador (este identifica los componentes de la vista y se encarga de realizar los cambios necesarios).

  Podemos decir que esta aplicación, si bien no es exactamente un caso de MVP (Modelo Vista Presentador), es al patrón de diseño al que más se aproxima, pues el controlador (presentador) es el encargado de comunicarse con la vista y con el modelo de datos.

- La clase `GameActivity`, al igual que en el diagrama presentado en la parte organizativa del proyecto, sigue recogiendo la mayor parte de la lógica, siendo sin duda la clase central y la encargada de relacionarse con prácticamente todas las clases del modelo.
- Como requisito particular de Android, incluimos en el modelo una parte (clase) destinada a controlar lo relativo a los permisos. Los sistemas operativos Android actuales exigen solicitar permiso al usuario para usar muchos de los componentes/datos del sistema. En nuestro caso, el permiso clave para que el sistema funcione es el permiso de acceso a la cámara, que naturalmente solicitamos en el primer inicio de la aplicación.
- Muchas tareas se tuvieron que aislar en una clase propia para poder llevarlas a cabo de una manera asíncrona. Ejemplos de esto lo vemos en el procesado de imágenes y en las tareas realizadas por los *timers*, protagonistas de poner a punto los metrónomos.
- Por la estructura de las aplicaciones Android nativas, la cámara se debió crear directamente dentro de las clases que la usaban (concretamente, se añade un objeto *cámara* en la vista XML que posteriormente es referenciado, inicializado y controlado por la parte lógica).
- La librería de tratamiento de imágenes nombrada en anteriores ocasiones, OpenCV, juega un papel importante en la aplicación. Se integra en todas las clases relacionadas con la cámara y con el tratamiento de imágenes.

  Para poder exprimir al máximo la cámara, no se usó la implementación directa aportada por la librería OpenCV, si no que se creó una clase personalizada reutilizando muchas funciones de la cámara nativa de Android y la implementación de OpenCV. Uno de los motivos por lo que esto fue necesario fue para el control del *flash* de la cámara. La clase creada para ello la podemos ver en el diagrama con el nombre `JavaCameraClass`.

  Por otra parte, muchas de las operaciones de OpenCV usadas fueron: dibujar cuadro/cruceta en la calibración de la cámara para guiar al usuario en la calibración, segmentación de imágenes y tratamiento/guardado de imágenes provenientes de la cámara. De lo que sacamos bastante provecho fue de la posibilidad de tratar las imágenes de la cámara antes de mostrarlas como salida (gracias a lo cual, pudimos realizar lo anteriormente comentado: dibujar en la pantalla de calibración diferentes dibujos).

4.4.2 Node.js y XAMPP

En el anterior punto de este capítulo comentamos la estructura de los componentes físicos que juegan algún papel en nuestra aplicación, incluyendo los correspondientes elementos software. Uno de estos es el ordenador que actúa como servidor, en el cual reside XAMPP y el servidor Node.js (puerto 3000).

Respecto a XAMPP no tenemos demasiado que añadir, pues simplemente se encarga de alojar la base de datos MySQL y de mantenerla accesible mediante el servidor Apache, para que el principal interlocutor con nuestra aplicación, el servidor Node.js, pueda realizar consultas a la base de datos.

El servidor Node.js no se encarga de otra cosa que no sea recibir peticiones de la aplicación residente en el cliente para realizar consultas a la base de datos y obtener los datos adecuados. Estos datos serían retornados a la aplicación del dispositivo solicitante.

El servidor Node.js contiene diferentes rutas (*routes*) que son las que integran las diferentes operaciones que se realizan sobre la base de datos. Las operaciones que se realizan son:

- Obtener un determinado código QR (registro de la tabla asociada en la base de datos).
- Obtener el conjunto de sonidos por defecto (compás y sonido de error, básicamente).
- Obtener sonidos de la base de datos (para un determinado código, según el tipo de lámina a la que esté asociada este código).
- Establecer un propietario para un código QR concreto (el propietario del código).
- Restablecer el propietario de los códigos QR para un determinado dispositivo (borrar el dispositivo actual de todos los códigos QR donde se encuentre registrado).

Sin extendernos demasiado, las rutas en Node.js funcionan asociando, valga la redundancia, una ruta (formato: */ruta*) a un determinado archivo Javascript que al fin y al cabo es el encargado de realizar la consulta SQL a la base de datos. La lista concreta de rutas (archivos Javascript) implementada está formada por los archivos:

- `getCode.js`
- `getDefaultSounds.js`
- `getSounds.js`
- `setCodeOwner.js`
- `unlinkCodes.js`

En la Figura 4.22 se muestra una captura del código Javascript de la ruta `getDefaultSounds.js`. El resto de los archivos son similares, a diferencia de los parámetros que obtiene de la *queryString* y de la sentencia ejecutada sobre la base de datos.

```
//ROUTE ADDED BY ME (CREATED FILE TO "CALL" IT FOR THIS ROUTE)

var express = require('express');
var router = express.Router();
var url = require('url');

/* GET a sound from table by its name... */
router.get('/', function(req, res, next) {
    var queryData = url.parse(req.url, true).query;
    var mysql = require('mysql');
    var config = {
        host : 'localhost',
        user : 'root',
        password : '',
        database : 'tactilemusicsense'
    }
    var connection = mysql.createConnection(config);
    connection.connect();

    connection.query('SELECT * from game_sound where type = "-1"', function(err, rows, fields) {
        if (!err) {
            if (rows.length > 0) res.send({msg: rows});
            else res.send({msg:[]});
        }
        else res.send({msg:'Access Error'});
    });

    connection.end();
});

module.exports = router;
```

Figura 4.22: Archivo getDefaultSounds.js con las sentencias Javascript para obtener todos los sonidos por defecto.

5. Resultados Empíricos

En este capítulo presentamos algunos resultados empíricos para demostrar el rendimiento de algunos parámetros clave en el diseño e implementación de la aplicación.

5.1 Experiencia de juego

La experiencia de juego depende de varios elementos. Naturalmente tenemos una mejor experiencia de juego con un móvil que disponga de *flash* y de buenas características hardware (que influyan en la velocidad de respuesta de la aplicación). Independientemente de las características del dispositivo, a nivel general podemos obtener una buena experiencia de juego. Comentamos a continuación algunos puntos clave a la hora de jugar en nuestra aplicación.

El aspecto que quizás se ve más penalizado en cuanto a rendimiento es la reproducción de los distintos sonidos. Los tipos de sonidos que se reproducen cuando jugamos son dos: el sonido de compás del metrónomo y el sonido reproducido al interaccionar con alguno de los materiales.

Al empezar el juego, es totalmente normal que *los primeros sonidos del compás suenen algo desincronizados, pero se terminan estabilizando*. Esto no es algo que afecte a la experiencia de juego, ya que una vez sincronizados podemos jugar con absoluta normalidad. Además, en el primer compás se toma la imagen de referencia (no se debe introducir la mano), por lo que no nos afecta demasiado a la hora de jugar. En condiciones normales, estando a 60 BPM (*Beats per Minute),* que equivale a un segundo entre un sonido y otro, el metrónomo se estabiliza en la segunda parte del compás, es decir, después de dos segundos.

Debemos aclarar que el tema de los sonidos es un tema bastante delicado en Android, ya que es un Sistema Operativo basado en Linux que está ejecutando multitud de hilos/procesos en segundo plano, correspondientes a las aplicaciones personales (segundo plano) y a las aplicaciones y servicios del sistema (segundo plano). Disponiendo de unos recursos finitos (memoria, tiempo de CPU...) es difícil no experimentar problemas de sincronización.

A medida que aumentamos el BPM en nuestra aplicación podemos observar que aumentan los fallos en la sincronización, aunque eso sí, siempre se terminan estabilizando acorde con el BPM marcado (pues funcionan con un *Timer* basado en el BPM establecido). Es recomendable jugar a BPM lo más bajos posibles para obtener una mejor experiencia de juego. No quiere decir que no podamos probar a un BPM más alto de lo habitual.

Por otra parte, tenemos los sonidos reproducidos al interaccionar con los materiales. Debido a que detrás de estos sonidos se esconde una comparación de imágenes (procesado de imágenes), esta última parte del compás se ve ligeramente retrasada respecto al resto, aunque eso sí, sólo para altos BPM, pues se llega al siguiente compás antes de terminar el procesado. A una configuración de 60 BPM existe un retraso de 0,4 segundos (valor medio).

5.2 Calentamiento del teléfono móvil y consumo energético

En primer lugar, debemos decir que nuestra aplicación es una gran consumidora de energía en el teléfono móvil. Aunque bien es cierto que esto depende de sus capacidades y características (en cuanto a hardware se refiere) así como de algunas configuraciones dentro de la aplicación, tenemos

varios factores que condicionan el consumo energético:

- Pantalla del dispositivo siempre encendida.
- Uso constante de la cámara (aunque no sea visible).
- Uso constante del *flash* para una mejor experiencia de juego (si lo tenemos activado durante el tiempo de juego y durante la calibración).
- Frecuencia de descarga de sonidos para cada lámina (conexiones a Internet).
- Procesado de imágenes y reproducción de sonidos (memoria y tiempo de CPU).

Al jugar y bajo estas condiciones de consumo energético, el móvil como es normal se calienta bastante. Los principales responsables de estos son el *flash* y la pantalla siempre encendida, así como la captura constante de imágenes por parte de la cámara. Esto último se puede observar también cuando usamos la cámara para grabar videos (Pantalla siempre encendida y capturando imágenes).

5.3 Batería y tiempo estimado de juego

Con el móvil sujeto a las pruebas, que concretamente es un *Moto Z Play* (3510 mAh de batería) cuyas especificaciones podemos consultar en cualquier lugar de Internet, hemos realizado una monitorización de la batería.

Esta monitorización consistió en cerrar todas las aplicaciones que estaban ejecutándose en segundo plano en este dispositivo y posteriormente abrir nuestra aplicación para comenzar a jugar. Con la batería al 100 % y hasta llegar al 86 % empezamos a jugar, anotando el tiempo cada vez que el porcentaje bajaba un 1 %.

Con esta información creamos dos gráficas relativamente idénticas en la que ubicamos los datos. En el eje x disponemos de los minutos transcurridos y en el eje y disponemos de la información de la batería, para la primera gráfica se expresa en porcentaje y para la segunda se expresa en mAh (del móvil con el que se realizaron las monitorizaciones). Estas dos gráficas corresponden a las Figuras 5.1 y 5.2, respectivamente.

Como podemos apreciar en las Figuras 5.1 y 5.2, se ha elaborado gracias a esos puntos una recta de regresión para poder estimar el tiempo de batería restante si usáramos la aplicación de manera continua. Como cada móvil dispone de una batería con mAh distintos nos conviene más usar la ecuación de la recta de la Figura 5.2. La pendiente se considera igual independientemente del móvil y lo único que habría que cambiar es la constante de la recta por el número de mAh del móvil en cuestión. A pesar de coger la misma pendiente, hay que considerar que dependiendo del móvil la batería bajaría más rápido (pendiente descendente más alta) o más despacio (pendiente descendente más baja). No obstante, considerando que son muchos factores suponemos el mismo valor para el resto de los dispositivos móviles. Las gráficas resultado de aplicar la ecuación de la recta para un dispositivo móvil de 3510 mAh y de 1510 mAh (respectivamente) son las visualizadas en las figuras 5.3 y 5.4.

Como podemos apreciar la duración es de unas 3 horas y 47 minutos para la batería de 3510 mAh y de 1 hora y 38 minutos para la batería de 1510 mAh.

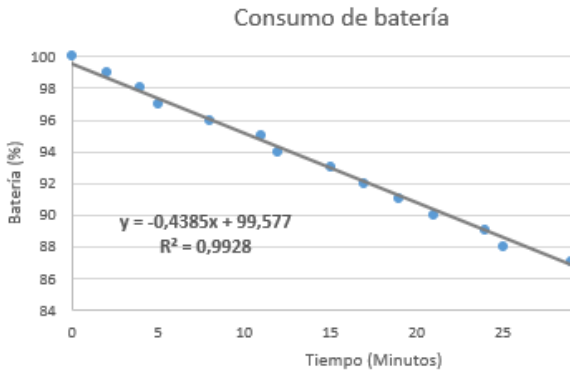

Figura 5.1: Regresión lineal del consumo de batería (%).

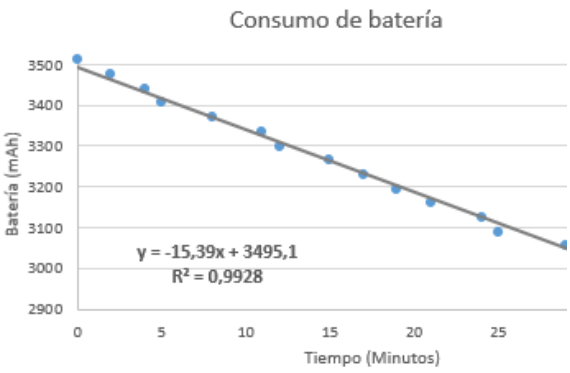

Figura 5.2: Regresión lineal del consumo de batería (mAh).

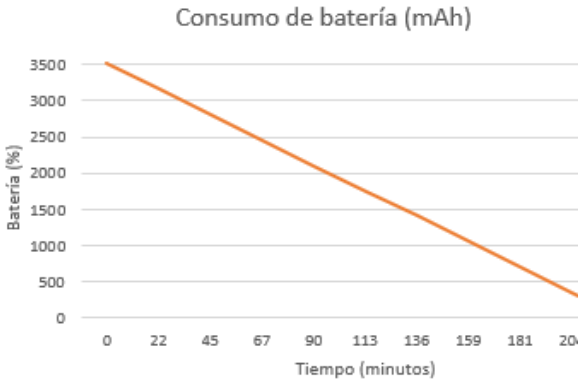

Figura 5.3: Consumo en móvil de 3510 mAh.

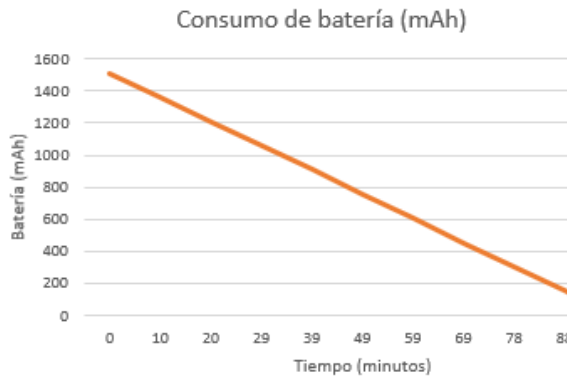

Figura 5.4: Consumo en móvil de 1510 mAh.

6. Conclusiones y trabajos futuros

En el siguiente capítulo se muestran las conclusiones obtenidas tras la realización de este TFG, así como los trabajos futuros (mejoras de la aplicación, expectativas) relacionados con este.

6.1 Conclusiones

Podemos afirmar que, con todo lo desarrollado y expuesto en este documento, se ha logrado conseguir el objetivo principal planteado en este TFG, es decir, relacionar las sensaciones captadas por el ser humano mediante los sentidos del oído y del tacto mediante la reproducción de sonidos adaptados al tipo de material que tocamos. A parte del objetivo principal, se han logrado los objetivos secundarios indicados en el apartado de introducción del documento.

En la realización de este proyecto, sin duda hemos afrontado tareas que no sólo han reforzado los conocimientos adquiridos a lo largo del cursado grado en Ingeniería Informática, tales como las distintas fases de realización del proyecto y la aplicación de metodologías de desarrollo basadas en Scrum, sino que hemos afrontado un aspecto de la Informática hasta ahora no estudiado, la programación móvil.

Este es un aspecto que, aunque sea quizás más vertical al objetivo de este proyecto, es bastante importante, pues es un sector en auge y está siendo cada vez más vital en la sociedad. Si podemos observar el entorno que nos rodea, los dispositivos móviles acompañan a las personas y las asesoran en muchísimas labores de la vida diaria. Un ejemplo claro lo vemos en la dependencia de las redes sociales, la reproducción de música, la captura de fotografías, la visualización de videos, entre muchísimas otras funciones aún más específicas.

Sin lugar a duda, este TFG nos ha aportado una síntesis de lo aprendido y, además, una visión bastante próxima a un entorno de desarrollo real y laboral, aunque eso sí, con algunos aspectos claramente diferenciadores, como el hecho de existir un único desarrollador, ser nuestro propio cliente, entre otros.

Gracias a la gran amplitud que cubre este TFG, hemos podido profundizar en temas como la música, aprender algunos conceptos sobre psicología y papiroflexia y tratar con un material que tiene mucho potencial, el cartón. Hasta la realización de este proyecto no había caído en la cuenta de los usos tan variados que ofrece este material, ya que, sí, es posible amueblar una habitación con cartón.

Dejamos de lado esto para centrarnos en el resultado obtenido al ejecutar el proyecto, que es sencillamente curioso. Parece una aplicación básica y simplista, pero debemos ser capaces de darnos cuenta de las posibilidades que ofrece, no de la complejidad que pueda llegar a sugerirnos. Personalmente, es algo en lo que sí ha recaído mi atención. Una aplicación capaz de aportar, de una manera simple y accesible a todo el mundo, la posibilidad de reflejar sensaciones táctiles en forma de sonidos musicales. Tocar un plástico y escuchar una especie de reconstrucción sonora de lo que sientes al tocarlo… me parece algo genial y con bastante potencial educativo y experimental desde el punto de vista informático, musical y psicológico.

Los resultados empíricos son más que satisfactorios: hemos podido encontrar una relación entre los BPM y la calidad del sonido obtenido al hacer el tacto de las láminas y la velocidad de procesado de la librería OpenCV dentro de Android para un teléfono móvil determinado. Por otro lado, hemos obtenido medidas del consumo de batería asociado al juego.

Para concluir, sólo puedo hacerlo con unas cuantas palabras… estoy realmente satisfecho con el resultado obtenido, no sólo a nivel de producto, sino a nivel personal.

6.2 Trabajos futuros

Entendemos que hemos desarrollado una plataforma con amplias posibilidades de mejora y evolución. Creo personalmente que esta aplicación puede ser fácilmente escalable. A pesar de encontrarse en su versión de entrega final, limitada al periodo de realización del TFG, es una aplicación con un amplio abanico de posibilidades. Algunas posibles ampliaciones son:

- La posibilidad de crear sonidos personalizados dentro de la aplicación, elegir los sonidos más adecuados para un conjunto de materiales (existiendo opciones correctas de forma predefinida o no, de forma que esté abierto a experimentar y valorar si las elecciones realizadas concuerdan con nuestras sensaciones al probarlo), entre otras muchas posibilidades.

- Permitir una configuración por voz usando órdenes muy simples que permitan al jugador interactuar de forma más directa y natural con el juego.

- Seguir profundizando en el estudio inicial realizado sobre sensores de tacto sobre superficies como las utilizadas en el TFG para abordar un problema que la Comunidad científica aún no ha resuelto eficientemente: cómo codificar las superficies que un ser humano toca sin utilizar interfaces cerebro-computador y comunicarlo eficazmente a un dispositivo de computación capaz de hacer sonidos musicales eficazmente.

Bibliografía

[1] Editorial, «Importancia,» 09 septiembre 2014. [En línea]. Available: https://www.importancia.org/cinco-sentidos.php.

[2] A. A. Fariñas, «Lamenteesmaravillosa,» 27 febrero 2016. [En línea]. Available: https://lamenteesmaravillosa.com/los-recuerdos-que-evocan-nuestros-cinco-sentidos/.

[3] «Wikipedia,» 05 febrero 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Portal:Psicolog%C3%ADa.

[4] M. Brainon, «Martin Brainon,» 07 noviembre 2016. [En línea]. Available: http://martinbrainon.com/inicio/el-tacto/.

[5] «Psytel,» 16 octubre 2017. [En línea]. Available: https://www.psytel.es/el-poder-del-tacto-en-psicologia/.

[6] Malena, «La Guia (Psicologia),» 21 septiembre 2009. [En línea]. Available: https://psicologia.laguia2000.com/psicologia-social/el-sentido-del-tacto.

[7] Europa Press (Madrid), «Las Provincias,» 27 abril 2018. [En línea]. Available: https://www.lasprovincias.es/sociedad/salud/investigacion/musica-cerebro-alzheimer-20180427175155-ntrc.html.

[8] «Wikipedia,» 05 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/M%C3%BAsica_y_matem%C3%A1ticas.

[9] G. S. Cuevas, «Lamenteesmaravillosa,» 22 abril 2014. [En línea]. Available: https://lamenteesmaravillosa.com/musica-y-emociones/.

[10] «El Confidencial,» 29 octubre 2013. [En línea]. Available: https://www.elconfidencial.com/alma-corazon-vida/2013-10-29/el-poder-de-la-musica-y-ese-momento-en-el-que-todo-parece-cobrar-sentido_45078/.

[11] Merino, Julián Pérez Porto y María, «Definicion De,» 2015. [En línea]. Available: https://definicion.de/pulso-musical/.

[12] «Wikipedia,» 11 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Pulso_(m%C3%BAsica).

[13] «Wikipedia,» 30 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Figura_musical.

[14] «Wikipedia,» 09 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Nota_(sonido).

[15] Vic, «La tecla de ESCAPE,» 19 agosto 2015. [En línea]. Available:

http://latecladeescape.com/h/2015/08/frecuencia-de-las-notas-musicales.

[16] «Wikipedia,» 12 marzo 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Tr%C3%ADada_(m%C3%BAsica).

[17] «Wikipedia,» 10 julio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Origami.

[18] Shuyang DingOrcid, Yunlu Pan Orcid, Mingsi TongOrcid y Xuezeng Zhao, «MDPI,» 28 noviembre 2017. [En línea]. Available: http://www.mdpi.com/1424-8220/17/12/2748.

[19] «The Telegraph,» 05 enero 2009. [En línea]. Available: https://www.telegraph.co.uk/technology/news/4126913/Mind-game-where-players-use-brainwaves-to-float-ball-through-hoops-unveiled.html.

[20] E. Wrenn, «Daily Mail Online,» 19 junio 2012. [En línea]. Available: http://www.dailymail.co.uk/sciencetech/article-2161678/Robot-textured-hand-detect-100-different-materials-just-fingertip--accuracy-better-man.html.

[21] AsiaHaptics2016, «Youtube,» 10 agosto 2017. [En línea]. Available: https://www.youtube.com/watch?v=MY3WoZ9NuzU.

[22] «Faculty of Managerial and Technological Engineering,» mayo 2014. [En línea]. Available: https://imtuoradea.ro/auo.fmte/files-2014-v1/Tarulescu%20Radu%202014-TEXTURE%20OF%20MATERIAL%20INFLUENCE%20AT%20ULTRASONIC%20DETECTION.pdf.

[23] «Specdrums,» [En línea]. Available: http://www.specdrums.com/.

[24] «TechXplore,» 22 enero 2018. [En línea]. Available: https://techxplore.com/news/2018-01-e-skin-virtual.amp?__twitter_impression=true.

[25] A. Raya, «Omicrono,» 05 abril 2018. [En línea]. Available: https://omicrono.elespanol.com/2018/04/convertir-un-portatil-en-tactil/?utm_source=dlvr.it&utm_medium=gplus.

[26] Á. Varona, «Generación Apps,» 03 octubre 2013. [En línea]. Available: http://generacionapps.com/aplicacion-ninos-sonidos-cotidianos/.

[27] «iTunes. App Store,» [En línea]. Available: https://itunes.apple.com/es/app/singing-fingers/id381015280?mt=8.

[28] «Android Developers,» [En línea]. Available: https://developer.android.com/studio/intro/?hl=es-419.

[29] «Wikipedia,» 10 julio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n).

[30] «Wikipedia,» 20 febrero 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Extensible_Markup_Language.

[31] «StarUML,» [En línea]. Available: http://staruml.io/.

[32] «Noteflight,» [En línea]. Available: https://www.noteflight.com/.

[33] «GIMP,» [En línea]. Available: https://www.gimp.org/.

[34] «Adobe,» [En línea]. Available: https://www.adobe.com/es/products/photoshop.html?sdid=8DN85NTQ&mv=search&s_kwcid=AL!3085!3!276539123993!e!!g!!photoshop&ef_id=WvXKKQAAAKxB9ldN:20180712191406:s.

[35] «TechSmith Camtasia,» [En línea]. Available: https://discover.techsmith.com/camtasia-brand-desktop/?gclid=Cj0KCQjw-JvaBRDGARIsAFjqkkr3NjNSpe7COx483FVhok33PPzw22HwP22hL2vuBuPKmFCRA6aJmyEaAv6OEALw_wcB.

[36] «ULPGC,» [En línea]. Available: https://www2.ulpgc.es/archivos/plan_estudios/4008_40/ObjetivosyCompetenciasdelGII.pdf .

[37] «Wikipedia,» 01 mayo 2018. [En línea]. Available: https://es.wikipedia.org/wiki/GNU_General_Public_License.

[38] «Wikipedia,» 25 mayo 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache.

[39] «Wikipedia,» 20 octubre 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Apache_License.

[40] «Wikipedia,» 13 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/MySQL.

[41] «Wikipedia,» 27 junio 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Licencia_MIT.

[42] «Wikipedia,» 25 abril 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Licencia_permisiva.

[43] «Office,» [En línea]. Available: https://products.office.com/es-es/student/office-in-education.

[44] «BOE,» [En línea]. Available: https://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf.

[45] «BOE,» [En línea]. Available: https://www.boe.es/doue/2016/119/L00001-00088.pdf.

[46] F. Tellado, «Ayuda WordPress,» 07 septiembre 2016. [En línea]. Available: https://ayudawp.com/cuanto-cobra-un-desarrollador-web-en-espana-y-en-el-resto-de-europa/ .

[47] «AG Systems,» 04 agosto 2017. [En línea]. Available: https://agsystems.es/cada-cuanto-renovar-los-ordenadores-de-la-oficina/ .

[48] J. Pablo, «Leantricity,» 01 junio 2016. [En línea]. Available: http://www.leantricity.es/cuanta-energia-gasta-un-ordenador-aproximaciones/ .

Anexo 1. Soporte

Para el correcto funcionamiento de nuestra aplicación, es requisito indispensable disponer de un soporte adecuado que nos permita enfocar nuestro móvil hacia una superficie de materiales. Este soporte debe tener una estructura resistente para aguantar el peso del móvil y permitir sacar fotos estables de la superficie de materiales.

El diseño de este soporte fue evolucionando muy rápidamente a base de ideas que se contrastaban mediante el clásico *prueba y error*, teniendo claro eso sí los conceptos sobre el soporte. El montaje de este soporte totalmente desmontable y plegable se encuentra en el anexo 2 correspondiente al manual de usuario. A modo de resumen, el montaje consiste en encajar las cuatro patas de cartón en el cuerpo de cartón, encajar el resultado en la base de cartón y atravesar el cuerpo de cartón con el tablón de cartón. Finalmente se inserta la lámina de cartón con los materiales deseados sobre el soporte ya montado.

El soporte montado se encuentra en la Figura A1.1, y está formado por los siguientes cinco elementos elaborados en cartón:

- Cuerpo de cartón (Figura A1.2): formado por cartón de una onda. Es el elemento principal del soporte. En él se pueden apreciar los huecos de encaje para las patas de cartón y para el tablón de cartón (ubicado aproximadamente a la mitad de altura de este componente, aunque puede modificarse para adaptarse al dispositivo móvil en cuestión). Aunque en la Figura A1.2 se muestran solo dos de las caras de este componente, está formado por cuatro caras (las otras dos son simétricas a las mostradas).
- Base de cartón (Figura A1.3): formado por cartón de doble onda. Se encarga de ofrecer los huecos de encaje para las cuatro patas. Con ello se logra mantener el soporte estable y sobre una superficie plana.
- Patas de cartón (Figura A1.4): formado por cartón de una onda. Estas patas de cartón se integran en el cuerpo de cartón, aportándole estabilidad (similar a la función de un trípode o las cuatro patas de una mesa). La altura de las patas puede modificarse para adaptarse al dispositivo móvil en cuestión.
- Tablón de cartón (Figura A1.5): formado por cartón de doble onda. Sobre este tablón se ubica el dispositivo móvil. Dispone de una apertura adaptada para que la cámara del dispositivo (y el *flash*) puedan apuntar hacia la superficie de materiales de una forma adecuada.
- Ejemplo de lámina de materiales (Figura A1.6): formado por cartón de una onda. Simplemente es una lámina sobre la que se distribuyen de manera equitativa cuatro materiales diferentes y el código QR asociado (en el centro). Contiene dos aperturas para facilitar el encaje en el soporte final (para evitar que la lámina se mueva al tocarse).

Figura A1.1: Soporte montado, con una lámina de materiales

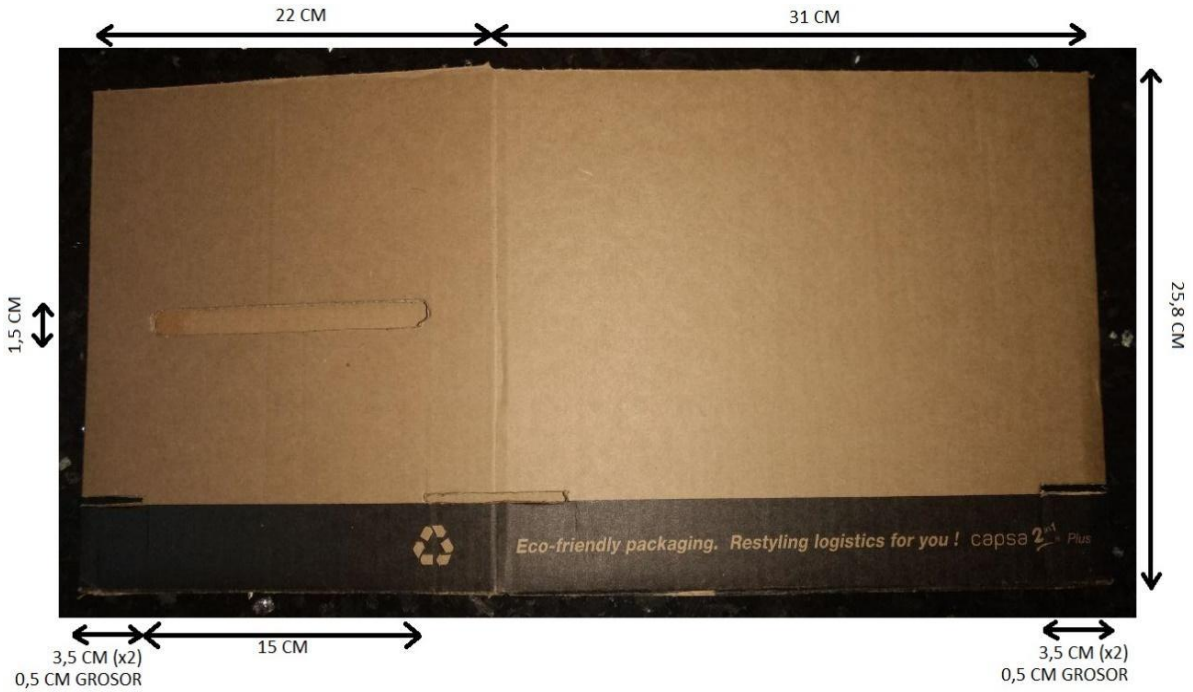

Figura A1.2: Cuerpo de cartón (x1).

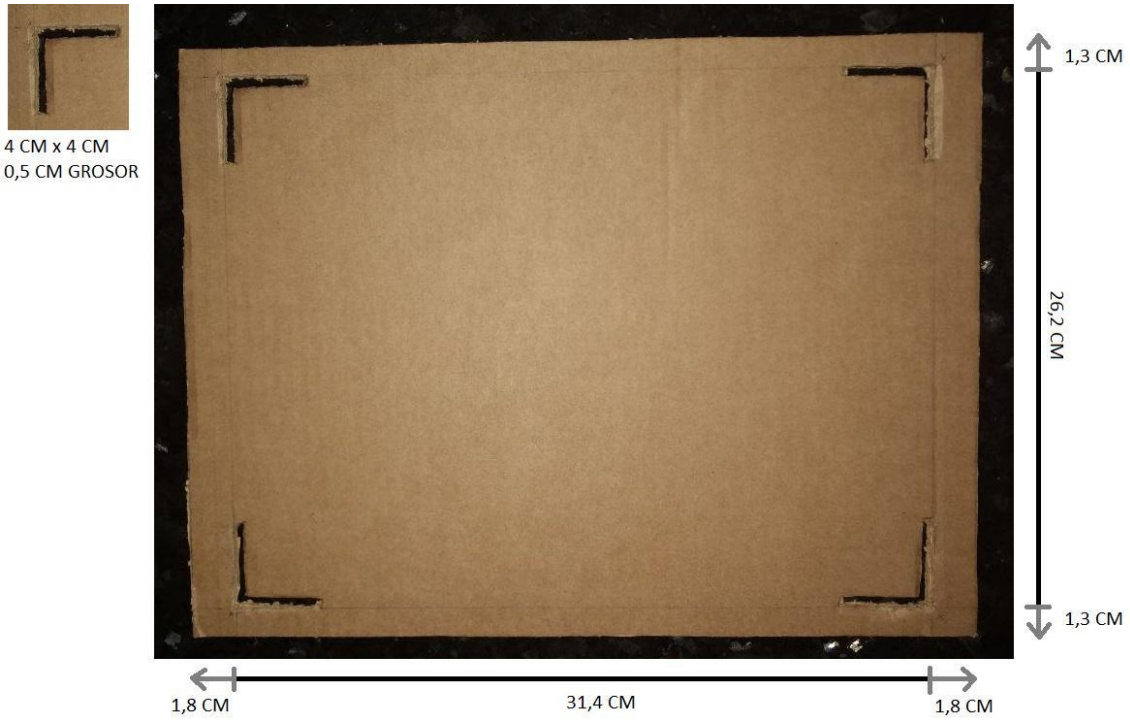

Figura A1.3: Base de cartón (x1).

Figura A1.4: Patas de cartón (x4).

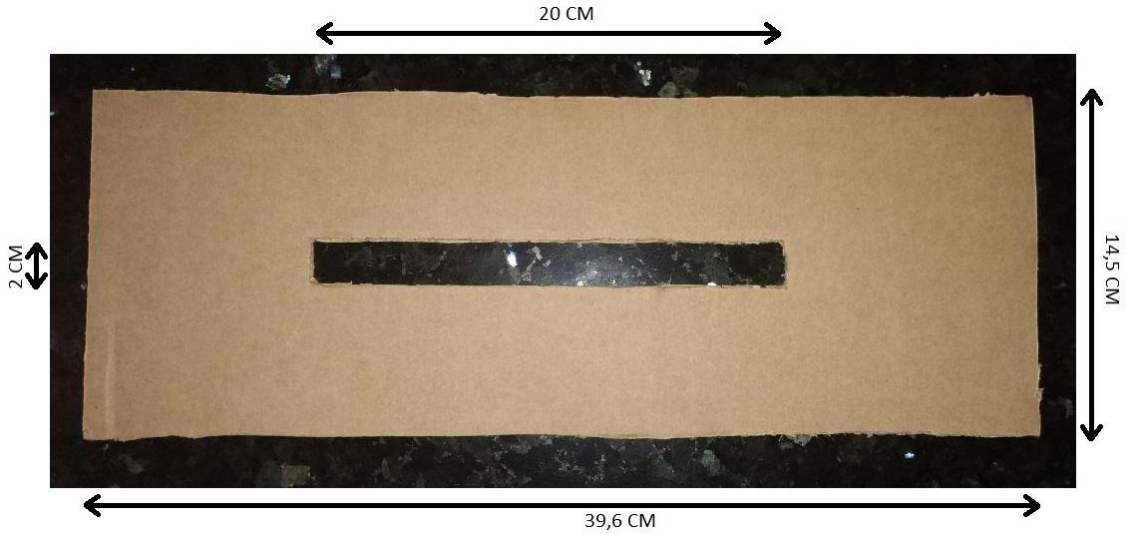

Figura A1.5: Tablón de cartón (x1).

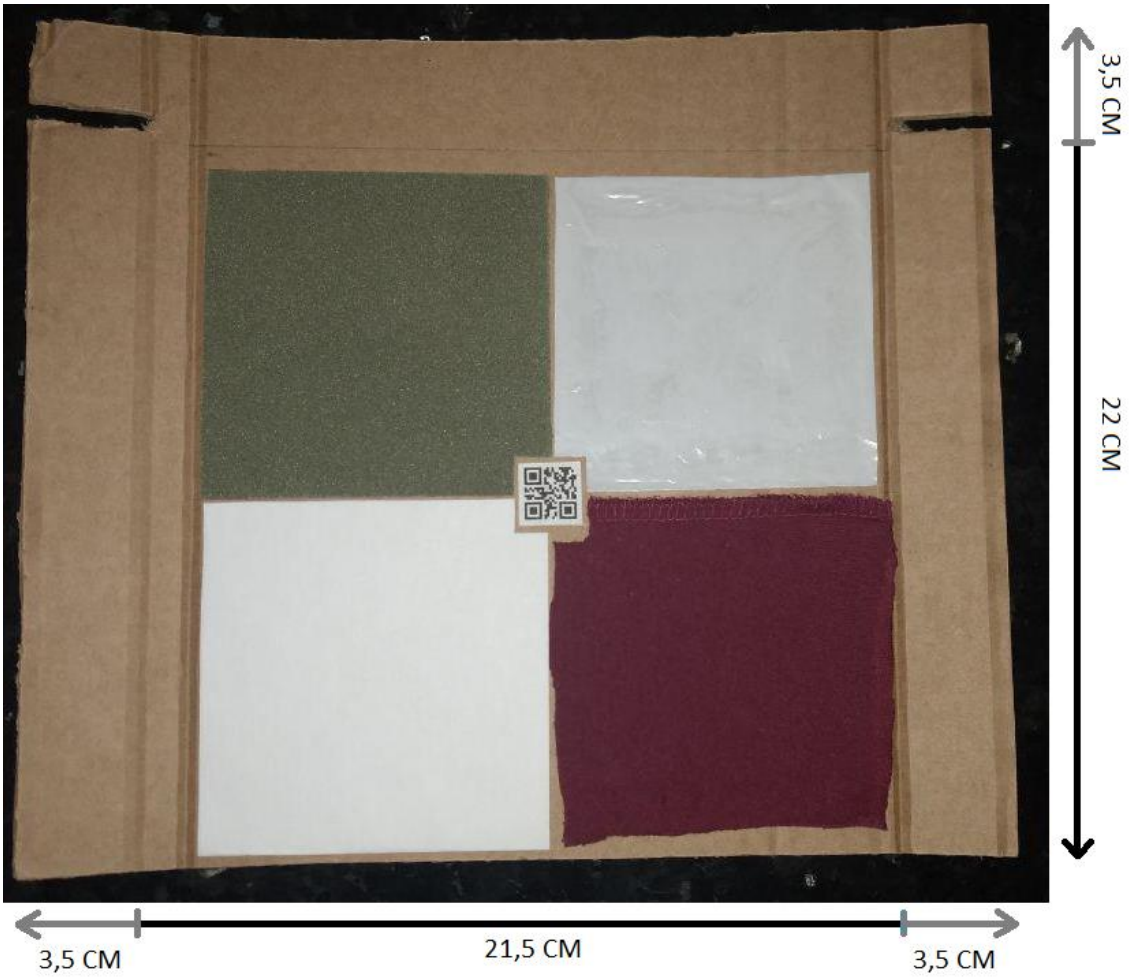

Figura A1.6: Ejemplo de lámina de materiales.

Anexo 2. Manual de usuarios

Este es el manual oficial de la aplicación *Tactile Sounds*. En este manual se indican todas las explicaciones necesarias tanto para montar el soporte de juego como para configurar/usar esta aplicación. Muchas de las explicaciones contenidas en este manual se encuentran integradas en la propia aplicación en forma de vídeos de ayuda, dentro del apartado *ayuda* de la aplicación. Este apartado lo puede encontrar en la pantalla principal con el icono clásico protagonizado por el símbolo de una interrogación (*?*).

A2.1 Montaje del soporte

Ya conocidas las características y las partes que forman el soporte, explicamos paso a paso su sencillo montaje.

Lo primero que debemos hacer es encajar las cuatro patas del soporte en el cuerpo de cartón. Para ello, es necesario doblar las patas como se indica en la Figura A2.1 y para luego, introducirlas en el cuerpo de cartón como muestran las figuras A2.2 y A2.3 (estas dos últimas figuras reflejan dos pasos consecutivos e instantáneos).

Figura A2.1: Se dobla una de las pestañas de la pata de cartón.

Figura A2.2: Se atraviesa el cuerpo de cartón con una de las pestañas de la pata de cartón.

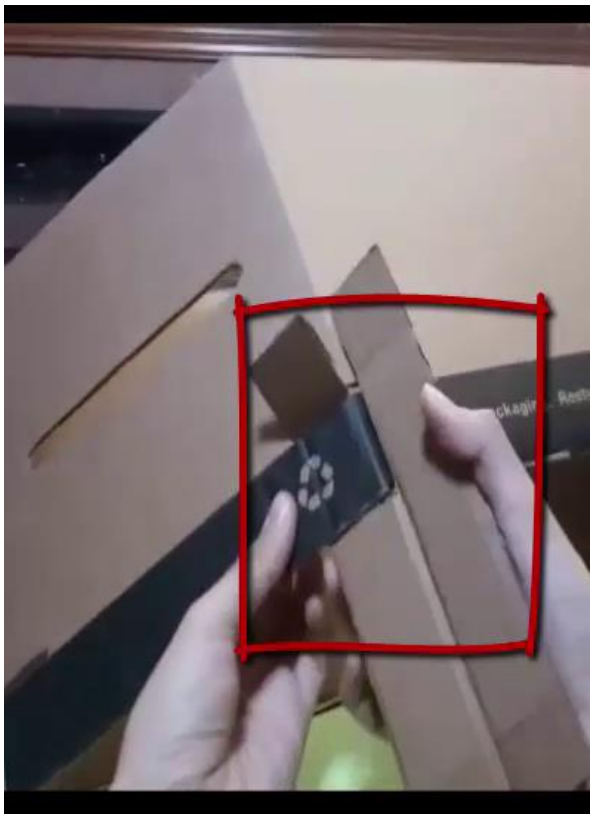

Figura A2.3: Pata de cartón insertada correctamente en el cuerpo de cartón.

Una vez montadas las cuatro patas de cartón en el cuerpo de cartón (deben quedar como en la figura A2.3), encajamos las cuatro patas en la base de cartón, como se aprecia en la Figura A2.4. Esto sirve para que las patas se mantengan alineadas y dobladas en un ángulo de 90 grados.

Figura A2.4: Se introducen las cuatro patas de cartón en la base de cartón (cada una con un ángulo de 90 grados).

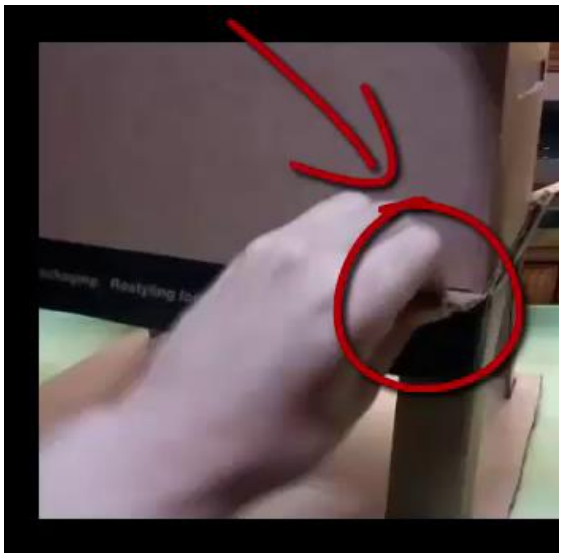

Figura A2.5: Se tira de las solapas salientes de las patas de cartón para ajustar al máximo el soporte

Con el soporte montado hasta ahora, tiramos de las solapas de cartón salientes del interior del cuerpo de cartón como se muestra en la Figura A2.5.

El último paso para montar el soporte es atravesar el cuerpo de cartón con el tablón de cartón sobre el que se ubica el teléfono móvil, a fin de mantener la cámara enfocando a la superficie de materiales ubicada en la base de cartón. Estos últimos pasos se muestran en las figuras A2.6 y A2.7, que reflejan acciones consecutivas.

A2.2 Introducción de láminas de materiales en el soporte

Lógicamente, para poder jugar necesitamos una lámina de materiales compatible, pues sin ella, el soporte no sirve para nada. Una vez elegida (o asignada) la lámina a usar, seguimos algunos pasos para introducirla.

Debemos tener el soporte de frente, es decir, con el agujero de introducción del tablón de cartón (sobre el que va el dispositivo móvil) de frente. Aclarar que el dispositivo móvil se introduce con el soporte en esta posición, quedando la pantalla orientada de la forma habitual al sujetar un dispositivo móvil con las manos (verticalmente).

Giramos el soporte en sentido horario para introducir la lámina de materiales, tal y como se muestra en las Figura A2.8 y A2.9.

Figura A2.6: Se introduce el tablón de cartón en una de las caras del cuerpo de cartón.

Figura A2.7: Se gira el soporte 180 grados para sacar el tablón introducido por el otro lado.

Figura A2.8: Se dobla y se sujeta la lámina de materiales por los dos laterales para que entre correctamente.

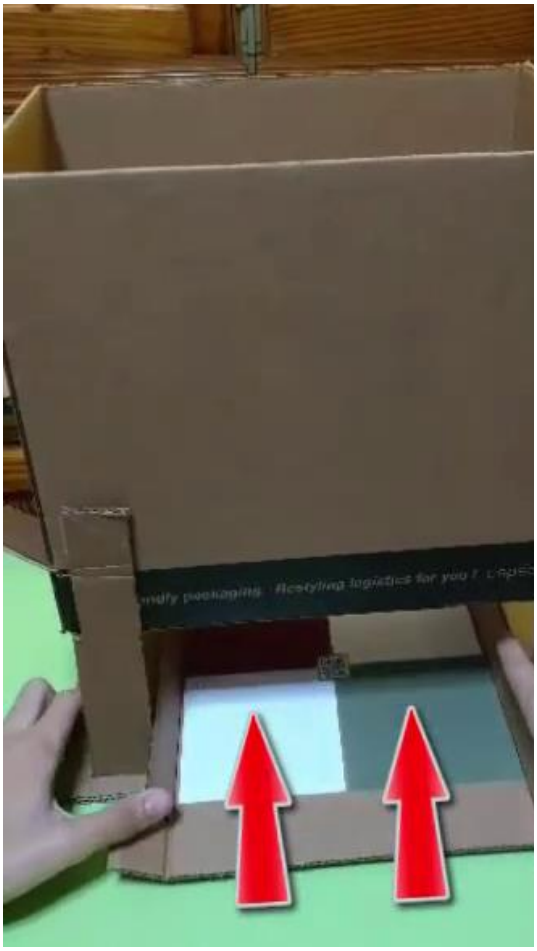

Figura A2.9: Introducción de la lámina hasta que las solapas laterales (previamente dobladas) se puedan replegar otra vez.

Figura A2.10: Se extiende la lámina de materiales dentro del soporte.

Una vez la lámina de materiales esté insertada es necesario extenderla con la mano para evitar que se quede total o parcialmente doblada en el interior del soporte. Este proceso se puede muestra en la Figura A2.10.

Ahora solo falta girar el soporte en sentido antihorario para volverlo a tener de frente. Esto es necesario para el siguiente paso del manual.

A2.3 Inicialización y configuración del juego: Calibración de la cámara

Una vez el soporte se encuentre montado y ajustado, se tiene que acceder a *Menú-Ajustes-Calibrar Cámara*. Una vez que se visualiza la imagen capturada por la cámara, se debe de colocar el dispositivo móvil sobre el tablón de cartón con la cámara apuntando hacia la lámina de materiales, tal y como se muestra en la Figura A2.11. Debemos ver a través del móvil algo parecido a lo mostrado en la Figura A2.12.

Se observa que no solo está mal ubicado el móvil sobre el tablón, sino que la previsualización es muy oscura. Debemos de activar la opción de *flash* (si el dispositivo incluye *flash*) para disponer de una visualización clara de la lámina. Para ello, nos vamos a *Menú-Ajustes* y activamos la opción *Flash*. El resultado al activar el *flash* debería de parecerse a la Figura A2.13. Ahora solo falta ajustar (mover) el dispositivo móvil hasta obtener una imagen que se ajuste a la Figura A2.14.

Figura A2.11: Colocación del móvil sobre el soporte.

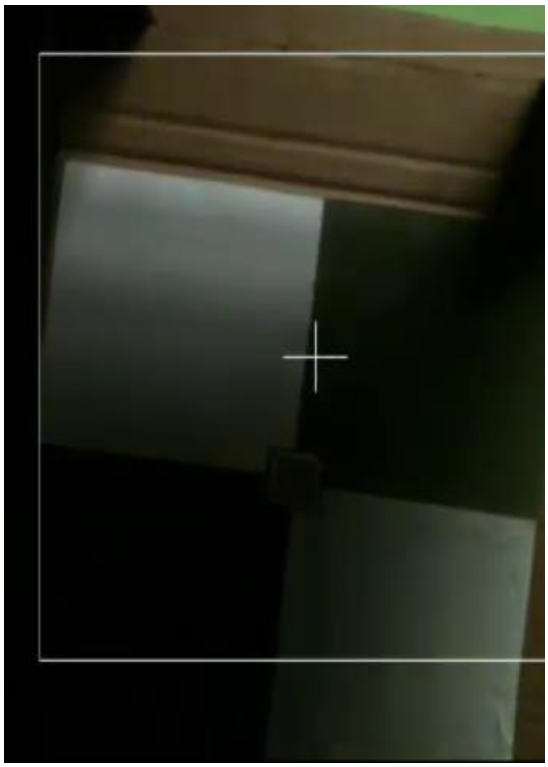

Figura A2.12: Previsualización del cámara apuntado hacia el soporte.

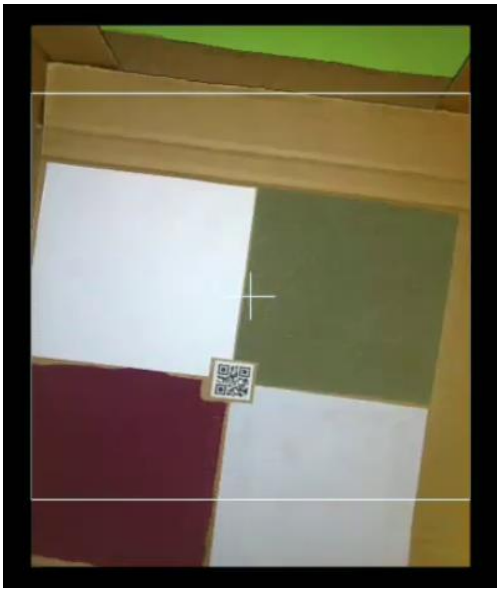

Figura A2.13: Previsualización de la cámara al activar el flash.

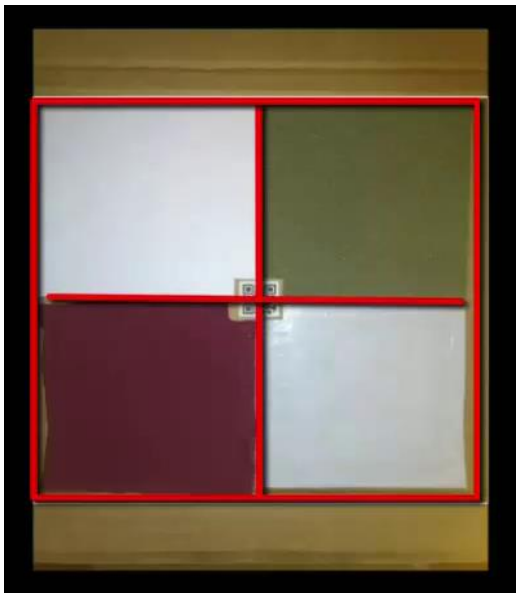

Figura A2.14: Móvil ajustado a los cuatro materiales (marco de referencia).

A2.4 Inicialización y configuración del juego: Activación del juego

A continuación, vamos a *Menú-Iniciar/Parar Identificación*. El sistema empieza a analizar las imágenes de la cámara para identificar el código QR. Si no se detecta se alterna el uso de *flash* automáticamente para optimizar la detección. Una vez captado el código debemos apreciar los mensajes que se muestran en la Figura A2.15.

En caso de experimentar algún tipo de problema, lo mejor es ir al apartado de errores de este manual o echar un vistazo al pliego de condiciones.

A2.5 Inicialización y configuración del juego: Configuración del juego

El juego tiene una opción de configuración para adaptarlo a nuestro gusto. En este apartado se comentan algunas de las opciones que se pueden ajustar. Para visualizar todas estas opciones, debemos de ir a *Menú-Ajustes*. Las opciones de configuración se muestran en la Figura A2.16, que es la pantalla de ajustes de la aplicación.

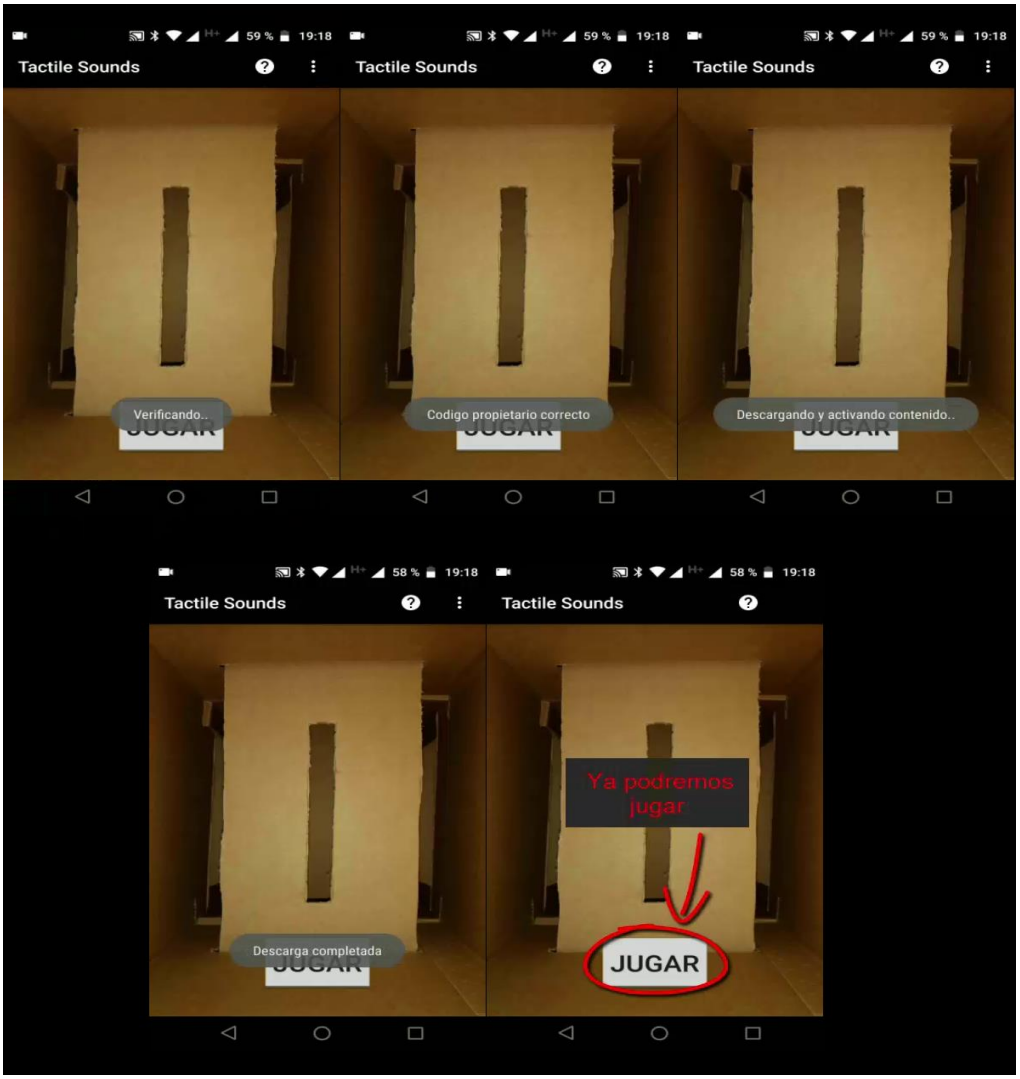

Figura A2.15: Sucesión de mensajes cuando se detecta el código QR y se activa correctamente el juego.

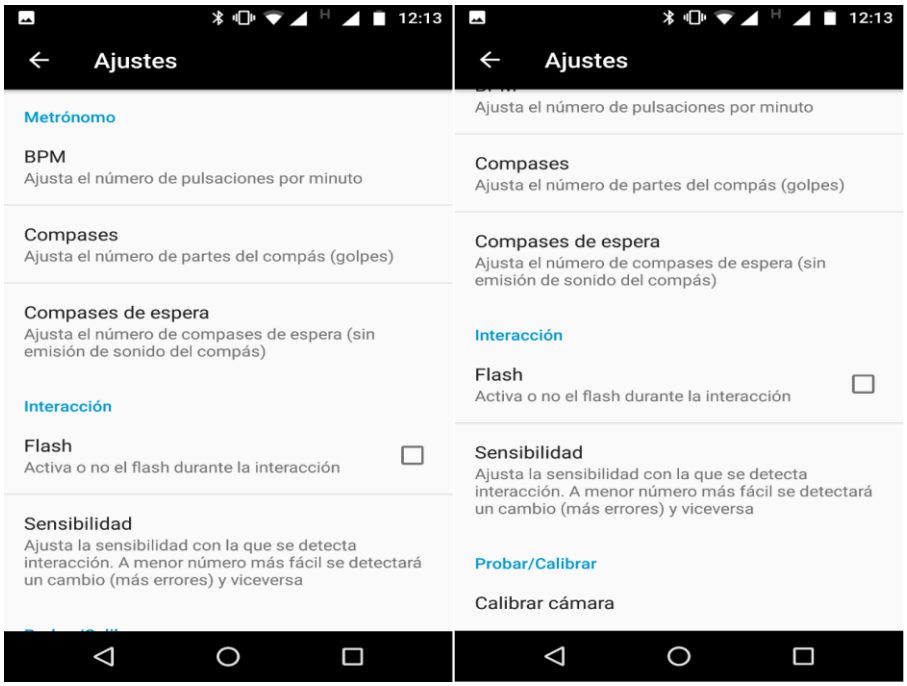

Figura A2.16: Pantalla de ajustes de la aplicación.

Comentamos los parámetros de configuración de juego con mayor detenimiento:

- Pulsaciones por minuto (BPM): a modo de resumen, nos define cuanto tiempo hay entre un sonido y otro. A menor BPM más tiempo habría entre sonidos y viceversa. Valores disponibles: 40, 60, 80, 100 y 120.
- Compases: indica el número de pulsos de ritmo que se reproducirían antes de detectar interacción (y reproducir el sonido en cuestión). Concretamente, se emitirían X-1 pulsos de percusión, siendo X el valor elegido para este parámetro. El sonido restante se reserva al sonido del material que se está tocando (si no se toca ninguno, no se emitiría sonido alguno en esta parte del compás). Valores disponibles: 4, 5, 6, 7 y 8.
- Compases de espera: se corresponde con el número de silencios antes de empezar otro compás. Para hacernos una idea, si ponemos 2 compases de espera, el resultado sería el siguiente: X-1 pulsos de ritmo, sonido o silencio asociado a la interacción con el material en cuestión y 2 silencios (como 2 compases de percusión, pero sin emisión de sonido). Valores disponibles: 0, 1, 2, 3 y 4.
- Sensibilidad: corresponde al umbral que debe superarse para que se emita el sonido asociado a la interacción con alguno de los materiales. Este valor expresa el porcentaje de píxeles que cambian respecto a la imagen de referencia (imagen de la lámina de materiales). Valores disponibles: 12, 15, 18, 21, 25, 28 y 30.
- A mayor sensibilidad, más difícil sería detectar la mano sobre un material (cambios en la imagen), a menor sensibilidad, más fácil de detectar sería. Debemos ser conscientes que, si ponemos un valor muy alto, nuestra mano puede no detectarse, pero si ponemos un valor muy bajo, se pueden detectar manos cuando no las hay (sonidos fantasmas).
- *Flash* y *Calibrar cámara*: ya se han visto en ocasiones anteriores, lo único que hacen es activar o desactivar el encendido del *flash* al jugar/calibrar o bien darte acceso a la calibración de la cámara (respectivamente).

Para una configuración óptima de todos estos parámetros de configuración, se recomienda leer el pliego de condiciones.

Una vez configurados todos los parámetros, accedemos a la opción *Menú-Habilitar/Deshabilitar vista* para ocultar el menú a los jugadores. Este menú sigue siendo accesible, pero no disponemos de un icono que nos lo haga saber explícitamente. La opción de ayuda estaría siempre accesible (Figura A2.17). Asimismo, se recomienda probar el juego para comprobar que todo funciona correctamente.

A2.6 Jugar

Con todo preparado y activado para jugar, lo único que tiene que hacer el jugador para activar el juego es dar un toque sobre el botón jugar o sobre la pantalla (mismo efecto). En ese momento, vería que el botón pasa de *jugar* a *parar* y que se empiezan a escuchar los pulsos de ritmo del compás.

En el primer compás no debemos introducir la mano, ya que la imagen que se captura es la imagen de referencia. Además, así le damos un margen al metrónomo (golpes de ritmo) a estabilizarse. En los siguientes compases, colocamos la mano sobre los materiales justo antes de

llegar a la última parte del compás (donde se detecta interacción).

Cuando se detecta la interacción debemos escuchar un sonido apropiado al material, cuando no se detecta no se reproduce nada y cuando se detecta interacción en varios materiales se emite un sonido de error caracterizado por una secuencia de notas musicales descendentes (con connotación de tristeza).

Si existiera cualquier error, lo mejor es parar el juego y volverlo a iniciar, ya que puede ser que la imagen de referencia se capturara erróneamente. Si sigue dando problemas, se recomienda contactar al asesor o configurador de la aplicación y revisar la documentación de referencia.

A2.7 Desvincular láminas/códigos registradas con el dispositivo actual

Es importante llevar a cabo este simple paso si deseamos desvincular todas nuestras láminas de nuestro dispositivo móvil. Si hemos jugado con algunas láminas de materiales, éstas quedan registradas para ser usadas exclusivamente con el dispositivo con el que fueron registradas. Si queremos jugar con otro teléfono móvil (bien porque el actual se está quedando sin batería, bien porque queremos migrar nuestros códigos a otro dispositivo móvil) debemos de ir a la opción: *Ajustes-Desvincular mis códigos.* Si todo va bien, deberíamos ver el mensaje que se muestra en la Figura A2.18.

Si ocurriera algún tipo de error, debe actuar como se le ha indicado en otras ocasiones: revise el apartado de errores del manual y/o el pliego de condiciones.

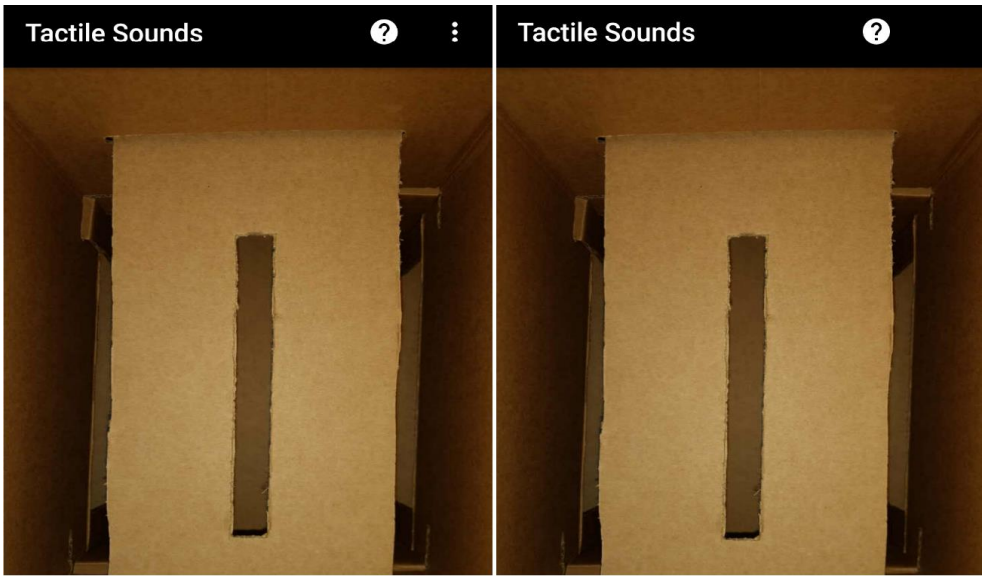

Figura A2.17: Visibilidad del menú, visible (izquierda) o invisible (derecha).

Figura A2.18: Mensaje mostrado al desvincular las láminas/códigos.

A2.8 Ayuda: vídeos

Si aún con toda la información aportada en este manual sigue teniendo dudas sobre cómo hacer algo, le invitamos a visualizar el video apropiado en el apartado de *Ayuda* de la aplicación. La mayoría de las explicaciones de este manual se encuentran de forma visual en forma de vídeo.

Al acceder a este apartado de la aplicación, se observa un menú de reproducción sobre una imagen de fondo (soporte montado con un título superpuesto), tal y como vemos en la Figura A2.19.

El título que vemos en pantalla hace referencia al contenido del vídeo actual. Si le damos a siguiente o anterior, pasaríamos al siguiente/anterior video, es decir, si le damos a siguiente pasaríamos a ver una pantalla similar, pero con otro título (y otro contenido), tal y como podemos ver en la Figura A2.20.

Cuando estemos sobre el vídeo que queremos reproducir, debemos de darle al botón reproducir (*Play)*. Una vez el video en reproducción, podemos pausarlo, reiniciarlo o simplemente cambiar de video como se hizo previamente al buscar el video deseado. Veríamos un menú de vídeo similar al de la Figura A2.21.

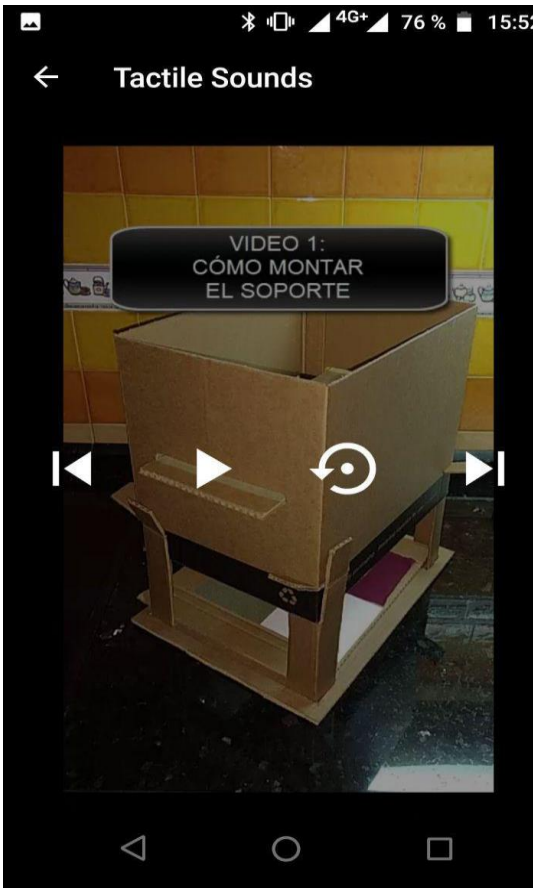

Figura A2.19: Pantalla de inicio del apartado Ayuda de la aplicación.

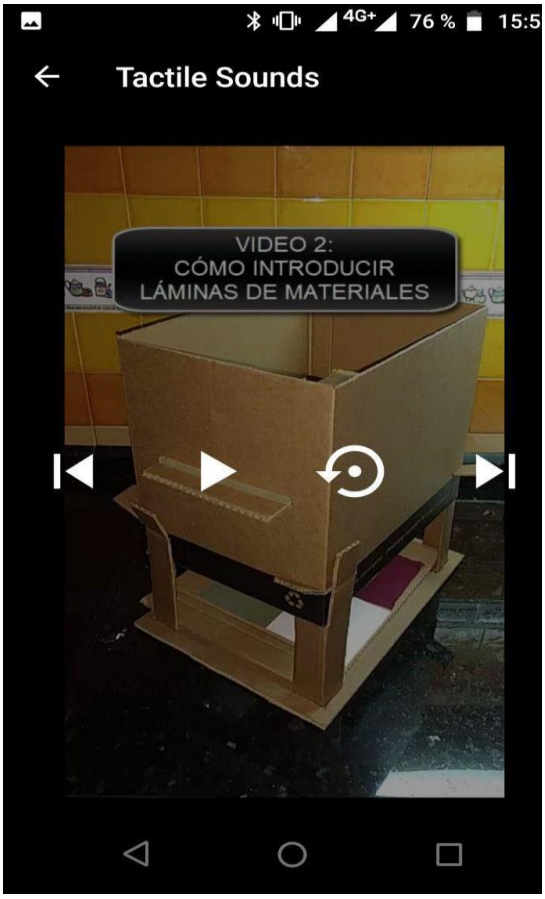

Figura A2.20: Resultado de darle al botón siguiente del panel de reproducción estando ubicados en el primer video de ayuda.

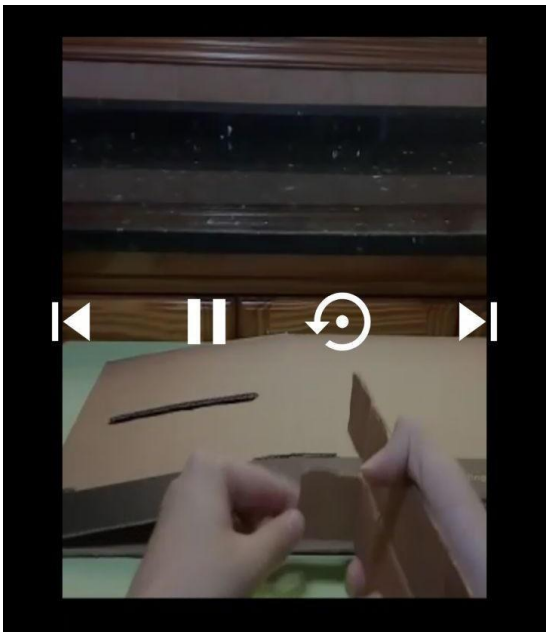

Figura A2.21: Menú de vídeo al estar reproduciendo un vídeo.

Finalmente, aclarar algunos comportamientos en este apartado de Ayuda:

- Cuando se pasa de un video a otro, independientemente de estar reproduciendo uno o no (en pausa), se muestra el siguiente vídeo pausado.
- Al finalizar la reproducción de un video se reinicia automáticamente, presuponiendo que no ha quedado claro el mensaje que transmite.
- El panel de video desaparece después de unos pocos segundos si nos encontramos reproduciendo un video o bien si accionamos con nuestro dedo sobre cualquier parte de la pantalla de reproducción de video (esto último tanto si estamos reproduciendo un video como si no).

A2.9 Errores comunes

Básicamente, tenemos un error protagonista: detección inadecuada de la interacción. Si nuestro sistema no responde adecuadamente a la interacción, debemos de asegurarnos de ajustar bien la sensibilidad.

Consecuentemente a lo descrito en el apartado de ajustes, hay que proceder de la siguiente manera para configurar de forma óptima la sensibilidad:

1. Empezar con la sensibilidad más alta posible.
2. Probar el juego con todos y cada uno de los cuatro materiales.
3. Si no tenemos problemas al interactuar con alguno de los materiales, habríamos configurado correctamente la sensibilidad. En caso contrario, siga leyendo (paso 4).
4. Si no se detecta algún material, tenemos que bajar al siguiente nivel de sensibilidad y volver al paso 2. Si estamos en el último nivel de sensibilidad y sigue dando errores al captar nuestra mano, saltar al paso 5.
5. En caso de encontrarnos en el último nivel de sensibilidad y seguir experimentando errores, debemos de revisar el pliego de condiciones. A modo de resumen, lo que nos cuenta al respecto es la alternativa de usar un guante preajustado para generar contraste con los materiales con los que experimentamos problemas. Ese guante debe tener las puntas de los dedos cortadas para poder sentir el material. El color del guante no debe parecerse a ninguno de los materiales de la lámina. Se puede observar el guante en la Figura A2.22.

Para cualquier otro error, debe leerse el pliego de condiciones, pues detalla las condiciones de uso ideales de la aplicación. A excepción del error comentado, el resto de las opciones no suelen ocurrir si se configuran bien los parámetros de juego. La sensibilidad es un caso muy particular que se debía comentar en este apartado de errores, pero otros errores comunes pueden ser: desincronización de sonidos, error de detección de QR...

Figura A2.22: Ejemplo de guante para generar contraste.

Anexo 3. Manual de instalación

Este manual de instalación va destinado al desarrollador encargado de habilitar la base de datos (servidor). Debemos de disponer en el equipo servidor de las siguientes herramientas/requisitos:

- Tener abierto el XAMPP, con el Apache (servidor Web) y el MySQL (base de datos) activados. En caso de no tener la base de datos creada e inicializada, debemos crearla.

  Para ello, abrimos el administrador de la base de datos (accionando el botón Admin del apartado MySQL de XAMPP), creamos una nueva base de datos con el nombre *tactilemusicsense* y una vez ubicados dentro de ella copiamos el contenido del archivo `tactilemusicsense.sql` (ubicado en la carpeta *nodejsDatabase*, dentro de la carpeta raíz del proyecto) en el campo de texto de la pestaña SQL.

- Antes de darle a continuar para inicializar las tablas y registros de la base de datos, debemos poder ver algo parecido a la Figura A3.1. Si todo va correctamente, deberíamos de tener la base de datos creada correctamente, en la cual podemos registrar láminas de materiales y sus sonidos asociados de acuerdo con la estructura de ésta.

- Tener instalado el paquete completo de Node.js. Aunque están incluidos en el proyecto, debemos asegurarnos de tener los paquetes necesarios instalados. Para ello, abrimos la consola *Node.js command prompt* y nos ubicamos en la carpeta `nodejsDatabase` (se encuentra en la carpeta raíz del proyecto). Posteriormente ejecutamos la orden *npm install*, que se encarga de instalar todas las dependencias definidas en el archivo `package.json` ubicado en el directorio en que nos encontramos actualmente.

- Tener el puerto 3000 de nuestro encaminador abierto y direccionando a nuestro equipo servidor (o los sucesivos encaminadores hasta llegar al equipo en cuestión). Esto es necesario para acceder a nuestro servidor desde cualquier red.

- Asignarle a la dirección pública de nuestro router el siguiente DNS dinámico (usado por la aplicación para contactar con el servidor de la base de datos sin depender de la IP dinámica): *dvdhack.ddns.net*.

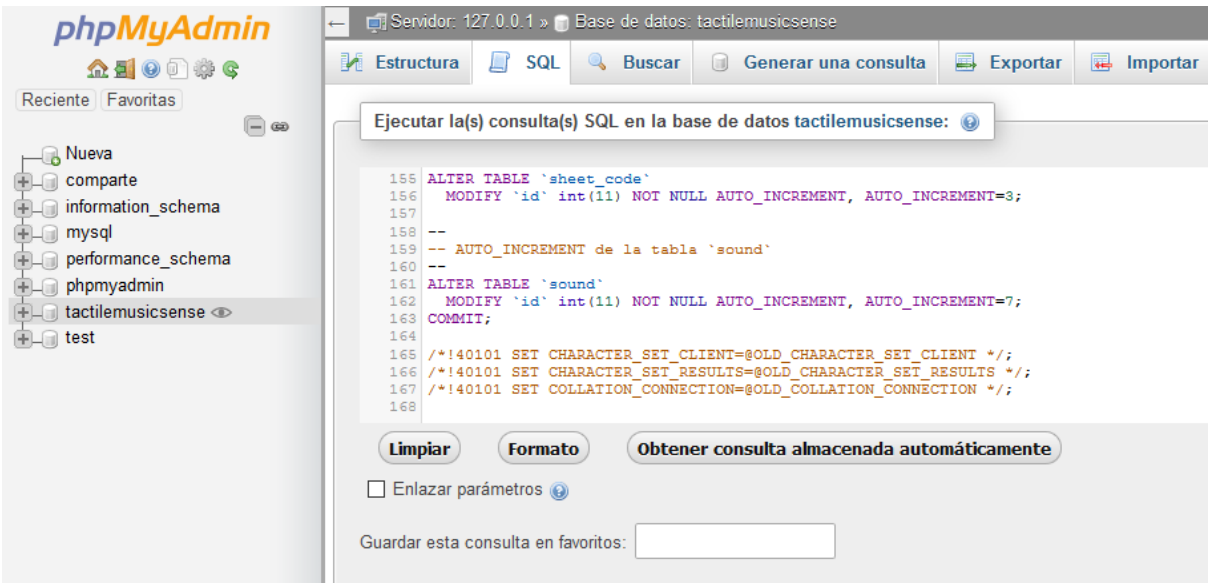

Figura A3.1: Base de datos tactilemusicsense creada, abierta y lista para inicializarse (contenido copiado y pegado en la pestaña SQL)

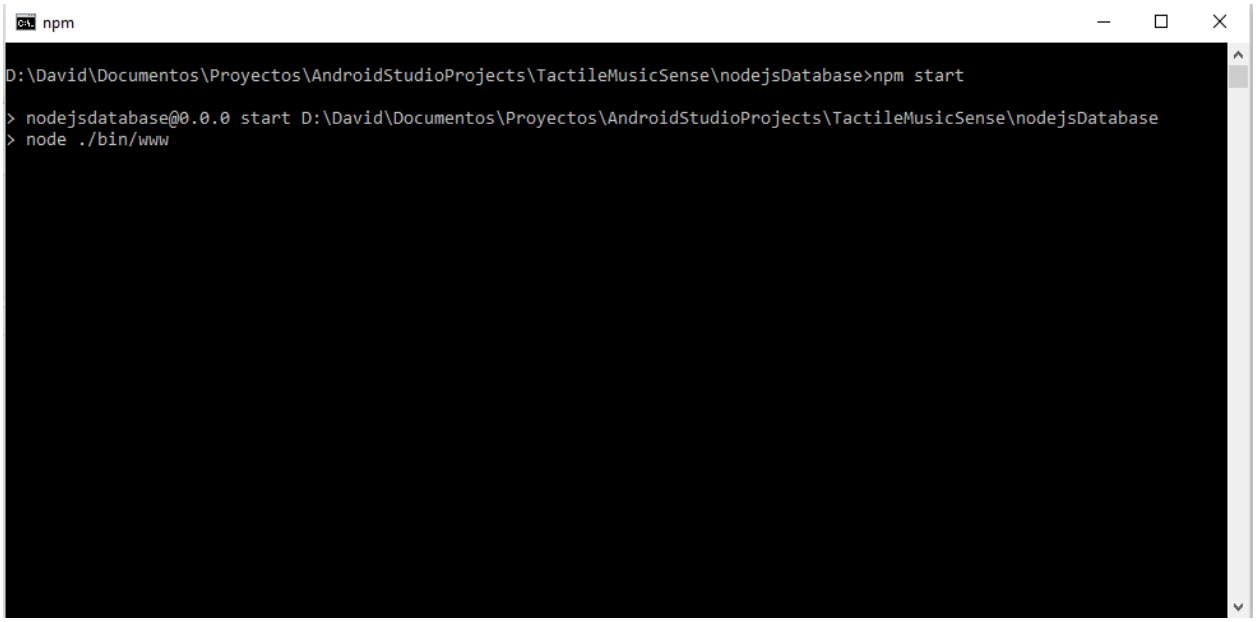

Figura A3.2: Ejecución de la sentencia npm start.

Anexo 4. Pliego de condiciones

Para que la aplicación funcione adecuadamente de acuerdo con lo previsto, es necesario respetar el siguiente conjunto de indicaciones:

1. Disponer de una lámina totalmente válida y legal que permita inicializar el juego con los sonidos predefinidos para esa lámina. De lo contrario, sería totalmente imposible comenzar el juego y tener una experiencia óptima, pues el juego no estaría activado o bien, en caso de estarlo (activado a conciencia usando otra lámina válida para la descarga de sonidos y activación del juego), no experimentamos la experiencia óptima y deseada por el desarrollador de esta aplicación.
2. Haber configurado correctamente la aplicación de acuerdo con su manual y/o los videos explicativos de ayuda. En caso contrario, lo más probable es que no encontramos el entorno de ejecución adecuado para nuestra aplicación, teniendo una experiencia muy alejada de la esperada.
3. Debido al potencial consumo de recursos que conlleva la aplicación, se recomienda dejar intervalos de descanso para el dispositivo móvil. Esto evita comportamientos inadecuados y propicia el no sobrecalentamiento del dispositivo. El tiempo de descanso adecuado varía según el dispositivo, pero una apuesta segura en cuanto a estos tiempos es alternar el estado del móvil (juego activado y pantalla encendida o juego apagado y pantalla apagada) cada 5-7 minutos. Podemos aprovechar este tiempo para que el asesor cambie de láminas de materiales, evitando también la monotonía táctil y sonora.
4. En adición a lo anterior, haber configurado bien la sensibilidad para que la experiencia sea lo más *natural y fluida* posible. En caso de ser víctima de errores de la aplicación al interactuar con la lámina de materiales (habiendo probado/ajustado correctamente la sensibilidad) es requisito disponer de un guante como el especificado en la parte de *Errores* (en manual y/o videos explicativos). Este guante consiste en un guante con las puntas de los dedos cortadas para poder tocar los materiales con la yema de los dedos y poder generar contraste. Es una forma de optimización del sistema frente a falta de contrastes, no un fallo en sí mismo, pues no se da en todos los materiales.
5. Se recomienda altamente el uso de un BPM bajo para obtener una respuesta óptima del sistema y apreciar mejor los sonidos asociados a un material. A un BPM relativamente bajo podemos experimentar con el material de una forma más pausada para asimilar mejor el sonido (asociado a este) que se emite.
6. En caso de usar un BPM relativamente alto, es recomendable establecer al menos uno o dos compases de espera para evitar la mezcla de sensaciones del jugador, pues tocaría y escucharía más sonidos en menos tiempo. Se puede considerar como un nivel *superior de juego*.
7. Si el reconocimiento del código QR no funciona correctamente, se recomienda probar a parar el reconocimiento y volverlo a activar. En caso de que siga sin funcionar, por favor, revise que la calidad de su cámara y su enfoque sean capaces de solventar esta situación.
8. En caso de tener errores con la activación del código QR, es decir, que se reconozca el código QR pero que no sea capaz de pasar al siguiente paso, bien por un *QUERY ERROR* o por errores de propiedad de la lámina contenedora del código QR. Esto se debe a errores en el servidor, en el primer caso se debe a que no se está conectando bien con el servidor (revise su conexión a Internet), en el segundo caso se debe a un error en la información almacenada en las bases de

datos (revise que la lámina realmente le pertenece y en caso afirmativo, lo mejor es contactar con el desarrollador de la aplicación).

9. Siempre que sea posible, debemos de jugar con la opción de *flash* activada. Con esto, no dependemos de estar en un entorno no lumínico.
10. Si existe algún tipo de error o anomalía no recogida en el apartado de errores del manual o en este pliego de condiciones, lo mejor es que se ponga en contacto con el desarrollador de esta aplicación.

Anexo 5. Especificación de casos de uso

CASO DE USO	1	Jugar		
Descripción		El jugador comienza a jugar con la aplicación interaccionando con los materiales y escuchando resultados sonoros.		
Actores		Jugador		
Precondiciones		El asesor debe haber calibrado la cámara (CU:11) y activar el juego mediante la lectura de QR de la lámina (CU:4).		
Flujo normal		**Paso**	**Acción**	
		1	El usuario acciona la opción de jugar en la pantalla principal de la aplicación.	
		2	El sistema enciende el flash del dispositivo móvil y activa la reproducción del metrónomo.	
		3	El usuario espera la reproducción completa de un compás sin introducir la mano en el soporte.	
		4	El sistema captura una imagen de referencia de la lámina de materiales.	
		5	El usuario introduce la mano sobre la lámina de materiales antes del último pulso del compás.	
		6	El sistema captura una nueva imagen de referencia (en el último pulso del compás).	
		7	El sistema detecta nuestra mano sobre un material y reproduce el sonido asociado.	
		8	Se repiten los pasos 5,6 y 7 hasta que el usuario acciona la opción de parar el juego.	
Postcondiciones		Estado inicial antes de jugar (opción de jugar disponible, ninguna emisión de sonido y flash apagado).		
Variaciones		**Paso**	**Acción**	
		2	El sistema activa la reproducción del metrónomo (flash desactivado).	
		7	El sistema no detecta nuestra mano y no reproduce ningún sonido.	
		7	El sistema detecta nuestra mano sobre varios materiales y emite un sonido de error.	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones		3	El usuario no espera el primer compás e introduce la mano. Con la foto de referencia mal, el sistema funciona erróneamente (no existe una imagen "limpia" de la lámina de materiales).	
Observaciones				

Figura A5.1: Especificación del caso de uso Jugar.

CASO DE USO	2	Ver videos de ayuda		
Descripción		El usuario visualiza videos que le ayudan a configurar y/o usar la aplicación/soporte.		
Actores		Usuario		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El usuario accede a la pantalla de ayuda desde la pantalla principal de la aplicación.	
		2	El sistema muestra panel de control de videos (algo transparente) superpuesto al primer video de ayuda.	
		3	El usuario selecciona el video deseado usando las acciones "siguiente" o "anterior" del panel de video.	
		4	El usuario activa la reproducción del video seleccionado.	
		5	El sistema reproduce el video indefinidamente (bucle).	
		6	El usuario pausa el vídeo actualmente en reproducción.	
		7	El usuario sale de la pantalla de ayuda.	
Postcondiciones		Se muestra la pantalla de inicio en el estado previo a entrar en la pantalla de ayuda.		
Variaciones		**Paso**	**Acción**	
		6	El usuario reinicia el vídeo actual para comenzar a verlo desde el principio (no entendió algo).	
		6	El usuario cambia de vídeo usando las acciones "siguiente" o "anterior" del panel de video y se repiten los pasos 4, 5 y 6.	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones		Antes de empezar a reproducir cada video, se observa una pantalla con el título relativo al contenido del mismo.		

Figura A5.2: Especificación del caso de uso Ver vídeos de ayuda.

CASO DE USO	3	Habilitar/Deshabilitar Menú		
Descripción		El asesor cambia el estado de visibilidad del menú de opciones de la pantalla principal.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal (visible o invisible).	
		2	El sistema muestra el menú de opciones.	
		3	El asesor acciona la opción correspondiente a "Habilitar/Deshabilitar Menú".	
Postcondiciones		Si el menú estaba visible ahora estará invisible y viceversa.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones		El menú, visible o no, siempre está disponible para accionarse.		

Figura A5.3: Especificación del caso de uso Habilitar/Deshabilitar Menú.

CASO DE USO	4	Detectar QR		
Descripción		El asesor activa la detección de códigos QR para descargar los sonidos de la lámina y activar el juego.		
Actores		Asesor		
Precondiciones		La cámara del móvil esta calibrada y existe una lámina de materiales en el soporte (enfocada por la cámara).		
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a "Iniciar/Parar la identificación" (detección QR).	
		4	El sistema empieza a detectar el código QR.	
		5	El sistema detecta el código QR.	
		6	El sistema descarga los sonidos y habilita la opción de jugar.	
Postcondiciones		La opción de jugar deberá estar habilitada y funcional con los sonidos asociados a la lámina actual.		
Variaciones		**Paso**	**Acción**	
		5	El sistema no detecta el código QR y alterna la activación y desactivación del flash hasta detectarlo.	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones		5	El sistema no detecta nunca el código QR. Debemos empezar desde el principio o intentar recalibra la cámara y la lámina de materiales.	
Observaciones				

Figura A5.4: Especificación del caso de uso Detectar QR.

CASO DE USO	5	Desvincular códigos QR		
Descripción		El asesor desvincula los códigos QR (láminas) del dispositivo actual para poder usarlos en otro dispositivo.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a “Desvincular códigos QR”.	
		4	El sistema realiza una llamada a la base de datos para borrar los sonidos asociados al dispositivo.	
		5	El sistema muestra un mensaje confirmando la desvinculación de los códigos QR asociados.	
Postcondiciones		Los códigos desvinculados ya no pertenecerán a este dispositivo, pudiendo vincularse a cualquier otro (solo a uno).		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones		5	El sistema no desvincula correctamente los códigos QR. No debe ocurrir nunca. Si se diera el caso, se debería contactar con el desarrollador para eliminarlo manualmente.	
Observaciones		La utilidad de este caso de uso está en poder migrar láminas de materiales a otros dispositivos.		

Figura A5.5: Especificación del caso de uso Desvincular códigos QR.

CASO DE USO	6	Establecer BPM		
Descripción		El asesor establece/regula el BPM (Beat Per Minute) del juego.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a “Ajustes”.	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona la opción relativa al BPM.	
		6	El sistema muestra las opciones de BPM disponibles.	
		7	El usuario selecciona el valor de BPM deseado.	
		8	El sistema notifica mediante un mensaje de que el valor fue actualizado correctamente.	
Postcondiciones		Al jugar debemos escuchar el metrónomo al BPM establecido.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Figura A5.6: Especificación del caso de uso Establecer BPM.

CASO DE USO	7	Establecer compases		
Descripción		El asesor establece el número de partes de cada compás del juego.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a “Ajustes”.	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona la opción relativa a los compases.	
		6	El sistema muestra las opciones de compases disponibles.	
		7	El usuario selecciona el valor de numero de partes de compás deseado.	
		8	El sistema notifica mediante un mensaje de que el valor fue actualizado correctamente.	
Postcondiciones		Al jugar deberemos escuchar el metrónomo correspondiente al valor establecido (menos uno, que será el sonido de interacción con los materiales).		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Figura A5.7: Especificación del caso de uso Establecer compases.

CASO DE USO	8	Establecer compases de espera		
Descripción		El asesor establece el número compases de espera (ubicados justo después de finalizar cada compás del juego).		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a “Ajustes”.	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona la opción relativa a los compases de espera.	
		6	El sistema muestra las opciones de número de compases de espera disponibles.	
		7	El usuario selecciona el valor de número de partes de compás deseado.	
		8	El sistema notifica mediante un mensaje de que el valor fue actualizado correctamente.	
Postcondiciones		Al jugar debemos de apreciar los compases de espera (después de cada compás de juego) de acuerdo al valor establecido.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Figura A5.8: Especificación del caso de uso Establecer compases de espera.

CASO DE USO	9	Activar/Desactivar flash		
Descripción		El asesor habilita o no el uso de flash durante la calibración de la cámara y el juego.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a "Ajustes".	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona activa/desactiva la opción relativa al flash.	
		6	El sistema notifica mediante un mensaje de que el valor fue actualizado correctamente.	
Postcondiciones		Al calibrar la cámara (CU: 11) o jugar, debe encenderse el flash si se activó la opción y viceversa.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Figura A5.9: Especificación del caso de uso Activar/Desactivar flash.

CASO DE USO	10	Establecer sensibilidad		
Descripción		El asesor establece la sensibilidad del sistema de juego al detectar cambios en la imagen (introducción de la mano sobre la lámina de materiales).		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a "Ajustes".	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona la opción relativa a la sensibilidad.	
		6	El sistema muestra los valores disponibles para la sensibilidad.	
		7	El usuario selecciona el valor de sensibilidad deseado y que mejor se adapte al contexto actual.	
		8	El sistema notifica mediante un mensaje de que el valor fue actualizado correctamente.	
Postcondiciones		Al jugar e introducir la mano, se detecta sin problema para todos los materiales con los que se interacciona, sin apreciar ningún sonido cuando no se interacciona con ninguno		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones		En la sensibilidad, conviene ir probando valores para adaptarse al contexto. Es algo menos empírico que los ajustes del metrónomo, por ejemplo (sabemos que tendrán un resultado determinado para cada configuración).		

Figura A5.10: Especificación del caso de uso Establecer sensibilidad.

CASO DE USO	11	Calibrar cámara		
Descripción		El asesor calibra la cámara para que enfoque correctamente a la lámina de materiales.		
Actores		Asesor		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	El asesor acciona la opción de menú de la pantalla principal.	
		2	El sistema muestra el menú de opciones.	
		3	El usuario selecciona la opción de menú relativa a “Ajustes”.	
		4	El sistema muestra todas las opciones de ajuste del juego.	
		5	El usuario selecciona la opción relativa a la calibración de la cámara.	
		6	El sistema abre una nueva pantalla con imágenes de la cámara en tiempo real.	
		7	El usuario mueve el dispositivo hasta que la cámara enfoque céntricamente la lámina de materiales.	
		8	El usuario cierra la pantalla de calibración cuando termina con el ajuste.	
Postcondiciones		Al jugar e interaccionar con los materiales se emiten 4 sonidos distintos (se segmenta correctamente la imagen analizada).		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones		Para enfocar céntricamente, se añade una referencia en la visualización de la cámara (cuadrado para la lámina y cruceta para el centro de materiales).		

Figura A5.11: Especificación del caso de uso Calibrar cámara.

Anexo 6. Competencias

Este TFG satisface todas las competencias del grado y comentamos las que se especificaron en la redacción de la solicitud de este TFG.

La competencia específica de la mención de Ingeniería del Software de la cual provengo, IS01, especifica lo siguiente en el documento oficial [36]: *Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software*.

Enumeramos las especificaciones seguidas en este TFG:

- Efectivamente todos los requisitos de usuario se han planteado mediante métodos/herramientas muy importantes dentro de la ingeniería de software, como lo son las historias de usuarios. Se han recogido los requisitos de esta forma, asignándoles una determinada prioridad y unos criterios de validación apropiados para verificar que las historias de usuarios funcionan correctamente y se han implementado de forma correcta.
- En la medida de lo posible y bajo las limitaciones del proyecto, ya que a pesar de simular un entorno de desarrollo real no deja de ser un proyecto individual limitado por las barreras del entorno educativo universitario, se ha procurado seguir la metodología Scrum.

  Básicamente, se ha segmentado el desarrollo en diferentes *Sprints*, en los cuales se han incluido las historias de mayor prioridad (bajo nuestro punto de vista). Se han anotado los tiempos a fin de autorregular la producción de incrementos (producto). Es necesario aclarar que, en este proyecto, actúo como mi propio cliente y como el único manager/desarrollador del proyecto, por lo que se pierde mucho de la metodología Scrum.
- En cuanto al desarrollo de código, se han seguido los principios o corrientes populares de la Ingeniería del Software como lo es el *Clean Code*. Se ha elaborado el código con la intención de que sea legible por otros desarrolladores y favoreciendo la Orientación a Objetos. Se ha evitado algo que bajo mi punto de vista está prohibido: el código Espagueti.

En segundo lugar, otra de las competencias directamente relacionadas con la Ingeniería Informática (independientemente de la mención a la que pertenezco) es la CII08, que de acuerdo con el documento oficial relata lo siguiente: *Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados*.

Esta otra competencia se ha llevado a cabo al pie de la letra, puesto que se ha desarrollado la aplicación en cuestión de la mejor forma posible procurando recoger todos los conocimientos adquiridos en el grado, escogiendo tanto la plataforma como el lenguaje que mejor se adapta a nuestro proyecto (programación móvil en Android usando Java y XML, además de una base de datos justamente necesaria bajo MySQL y el establecimiento de un servidor Web bajo Node.js para realizar las operaciones contra la base de datos).

En resumen, se han cumplido todas las competencias expuestas en este TFG. No obstante, es cierto que queda un largo camino a lo largo del cual debería seguir formándome y que, naturalmente, no existe la perfección (aunque siempre hay que tratar de alcanzarla).

Anexo 7. Normativa y legislación

Hablamos de varios aspectos legales, principalmente todo lo relacionado con licencias y leyes dentro del ámbito de la informática que afecten a este TFG. Naturalmente, hay leyes que afectan en más medida y otras que lo hacen en menor medida. Nos centramos exclusivamente en las más relevantes, al igual que en las licencias.

A7.1 Licencias

En este apartado se incluye la explicación para todas y cada una de las licencias de las herramientas utilizadas. Una licencia de software especifica los términos y condiciones de uso del software mediante un contrato entre el licenciatario y el licenciante/autor/titular de los derechos de distribución y explotación. Las licencias más importantes sobre las que hablamos, debido a su asociación con algunas de nuestras herramientas, son las licencias: GNU (GPL) y MIT. Hablamos también sobre otros tipos de licencias que incluyen nuestras herramientas.

A7.1.1 Licencia GNU (GPL)

La Licencia Pública General (General Public License) de GNU (*GNU's Not Unix)*, GNU GPL [37], es una licencia de tipo *copyleft,* por lo que implica que el software es totalmente libre de cara a su explotación (uso, copia, modificación, …) y, además, que el producto resultado está sujeto a las mismas condiciones. En otras palabras, implica que cualquier software que haga uso de otro con licencia GNU GPL está restringido a cualquier intento de apropiación. Garantiza que cualquier producto bajo esta licencia evolucione de manera colaborativa y totalmente abierta, creándose mejoras e incluso diferentes ramas de un mismo producto.

En nuestro caso, una herramienta muy particular utiliza esta licencia: XAMPP. Es un tanto particular en el sentido de que XAMPP está compuesto por la integración de varias herramientas, entre ellas el Apache y el MySQL. Esto implica que a pesar de que el XAMPP tenga esta licencia, en caso de uso comercial debemos consultar las licencias individuales.

En el caso del Apache [38], el tipo de licencia que tiene es la *licencia Apache* [39] (*Apache License o Apache Software License*). Como podemos imaginar es una licencia creada para el propio Apache y es una licencia de software libre permisivo. Lo más importante de este tipo de licencia es que no es del tipo *copyleft*, por lo que evita la apropiación del software.

El caso de MySQL [40] contempla las licencias GPL y de uso comercial. Básicamente, si se desarrolla con la licencia GNU GPL, el desarrollador se acoge a las características de esta licencia antes ya comentada, pero en caso de que desee no usar esta licencia, en el caso particular de MySQL puede solicitar una licencia comercial de MySQL.

Otra de las herramientas usadas que utiliza la licencia GNU GPL es el StarUML. Sin embargo, debemos indicar que esto es porque usamos la antigua versión de StarUML, puesto que ahora las nuevas versiones de StarUML están bajo una licencia de software comercial propietario (sin acceso al código fuente y pagando por su uso).

La última de las herramientas usadas bajo esta licencia es el GIMP, el software usado para editar las imágenes. Precisamente el hecho de que esté bajo esta licencia y que sea gratis, ha propiciado su uso frente a otros grandes editores de imágenes.

A7.1.2 Licencia MIT

La licencia del Instituto Tecnológico de Massachusetts (MIT License) [41], se origina como su nombre indica en el mencionado instituto. Lo que debemos saber sobre esta licencia es que es una licencia de software libre permisiva [42]. Esta licencia no es del tipo copyleft y permite que un tercero use o modifique nuestro software (software original) y elija la licencia que desee (de acuerdo con los términos de esta licencia MIT). Un ejemplo de uso de este tipo de licencia es el Node.js.

A7.1.3 Otras Licencias

En el caso de otras de las herramientas utilizadas, comentamos simplemente el tipo de licencia que integran.

En el caso de Sourcetree, contiene una licencia de software propietario, aunque eso sí, desde que Atlassian adquirió Sourcetree, a pesar de que su código fuente sigue siendo privado el uso de la aplicación es totalmente gratuito. Lo único que se requiere es un registro en la página oficial de Atlassian para obtener un ID que nos permita acceder a la aplicación (y al resto de aplicaciones de Atlassian). Bitbucket es otro producto de Atlassian que, aunque contiene una licencia de software propietario, no es totalmente gratuita, por lo que es del tipo *shareware.* En este caso particular, puedes usar la aplicación gratuitamente, pero con ciertas limitaciones, como el número de usuarios colaboradores en un repositorio privado. Para acceder a un mayor abanico de funcionalidades debemos pagar cierta cantidad de dinero.

En el caso de Camtasia Studio, la licencia es de tipo propietario, debiendo pagar completamente por el uso del producto. Sin embargo, puede considerarse del tipo *shareware,* ya que contiene una prueba gratuita. En nuestro caso, aprovechamos esta prueba gratuita para realizar la mayor parte del trabajo necesario con esta aplicación.

Para Noteflight tenemos otra licencia de software propietario, siendo además del tipo *shareware*, ya que ofrece un registro gratuito con limitaciones funcionales (límite de partituras creadas, por ejemplo), pudiendo ampliarlas con un coste adicional. Sin embargo, ofrece también una opción de prueba gratuita (con un límite temporal) para usar la aplicación sin limitaciones.

En el caso de Android Studio, simplemente comentar que tiene un tipo de licencia de software libre (*freeware*), proporcionando acceso al código fuente. Esto permite el avance colaborativo del programa a fin de mejorarlo.

Dejamos para el final una herramienta no mencionada en el apartado de *Herramientas utilizadas* por simple obviedad, el Office 365 Educación (Microsoft Office). Para la elaboración del contenido del documento actual se ha usado concretamente Microsoft Word 2016 y Microsoft Excel 2016, incluidos ambos en el paquete Office 365 Educación [43].

Este paquete es totalmente gratuito para estudiantes y profesores de cualquier centro educativo previamente registrado en el servicio. Lo único que debemos hacer es aportar nuestro correo institucional y obtendmos acceso gratuito a estas herramientas siempre y cuando se pertenezca al centro educativo (ULPGC). Office se reserva el derecho de poder revisar la validez de la inscripción al centro educativo, es decir, si sigues cumpliendo los requisitos para usar estos servicios o no.

A7.2 Leyes

En este apartado se comentaría básicamente la ley más importante y popular en los últimos meses, debido a su reciente reforma. Efectivamente estamos hablando de la Ley Orgánica de Protección de Datos.

A7.2.1 Ley Orgánica de Protección de Datos (LOPD)

La Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos (LOPD) [44], es una de las leyes más importantes y conocidas dentro del mundo del Software. Recientemente se ha reformado esta ley en España para adaptarse a la normativa europea (que entró en vigor hace dos años aproximadamente), concretamente entro en vigor este 25 de mayo de 2018. Ahora, la LOPD sería sustituida por el Reglamento General de Protección de Datos (RGPD) [45].

Aunque se mantienen algunos aspectos de la LOPD, es verdad que se han modificado algunos aspectos importantes en ella. Sin embargo, en este documento se explicarían solo aquellos puntos importantes que sean claves en nuestro proyecto, siguiendo eso sí la nueva reforma de esta ley.

A pesar de la reforma que ha sufrido, la LOPD sigue teniendo el mismo objetivo esencial, garantizar y proteger las libertades y derechos de las personas físicas en cuanto a datos personales se refiere, cubriendo aspectos como la intimidad y la privacidad. Algunas de las cosas que cambian en esta nueva reforma son: el derecho a la portabilidad, el derecho al olvido y la solicitud explicita para garantizar el consentimiento del usuario en todos los aspectos que así lo requieran. La reforma de esta ley va orientada a la evolución de la actual sociedad de la información.

En lo que a este proyecto se refiere, debemos destacar dos aspectos que debemos cumplir (y hemos cumplido). En primer lugar, en lo que a datos personales se refiere, nuestra aplicación no contiene demasiada información de carácter relevante, es decir, que tenga un nivel de privacidad alarmante. Si recordamos los datos que almacenamos en nuestra base de datos, tenemos, grosso modo, códigos QR y sonidos asociados a cada tipo de lámina de materiales. Estos datos no son importantes de cara a la privacidad del usuario, pues al fin y al cabo no son datos personales. No obstante, para garantizar la no copia de los códigos QR se utiliza el identificador del dispositivo con el que se juega. Este dato si es bastante más sensible que el resto de los datos, de hecho, es el único en nuestra base de datos (aun así, sigue sin ser vital). Para evitar el apropiamiento de este dato se emplea un método de encriptado mediante MD5, que genera un código *Hash* del código identificador del dispositivo. Es ese código encriptado el que viaja y el que se almacena en la base de datos. Para comprobar este código se utiliza (en tiempo de ejecución) el mismo método de encriptado y se compara con el de la base de datos. De esta forma, ni teniendo acceso a la base de datos o *esnifando* la red a la que estamos conectados con el dispositivo podrían obtener el acceso a este dato.

En segundo y último lugar, de acuerdo con las nuevas versiones de Android y siendo fiel al reglamento de esta ley, es necesario solicitar el permiso de la cámara en el primer inicio de la aplicación. Si este fuera negado por el usuario, nuestra aplicación no funcionaría, pero seguiríamos cumpliendo el reglamento, ya que solicitamos explícitamente el permiso, siendo responsabilidad del usuario concederlo para que funcione correctamente la aplicación. Aclaramos que esto, en anteriores versiones de Android, no era así, ya que al declarar los permisos de los componentes que deseabas usar (en un documento llamado *manifest*) se obtenía acceso inmediato a las funciones requeridas (acceso a la red, acceso a la cámara, acceso a los contactos, entre otros).

Anexo 8. Presupuesto

Para este proyecto en concreto, tenemos el presupuesto reflejado en la Figura A7.1. Antes de empezar a observar este presupuesto, debemos de tener en cuenta algunos aspectos para entenderlo por completo:

- Planchas de cartón: nos basamos en los precios de la tienda de cartón online Kartox.com que vende cartones de diferentes tipos a medida. Mientras más cartones pidas más baratas son las unidades. En nuestro caso se encargarían 20 planchas, obteniendo los precios reflejados en el presupuesto.
- En este proyecto se cuenta con diferentes roles de trabajo, empeñados todos por una misma persona, que al fin y al cabo es el autor de este TFG. Las horas destinadas a cada tarea son las previstas en la propuesta oficial de TFG aprobada por la Comisión de TFT (CTFG).
- El sueldo establecido para un desarrollador Web en España es de 1500 € al mes (brutos), que vienen siendo unos 6,25 €/hora [46]. A pesar de que cubrimos diferentes ámbitos de la informática (y también ajenos a la misma) hemos decidido establecer el coste/hora para cada rol (coste de mano de obra) en la cantidad mencionada previamente (6,25 €/hora).
- El tiempo ideal para cambiar de equipos informáticos en una empresa es de aproximadamente 5 años (3-4 años si no se le da soporte) [47]. Partiendo de estos datos, se ha añadido al presupuesto la parte proporcional a las horas de vida *gastadas* de los equipos informáticos. Son de 240 horas (aproximadas) para los ordenadores y de unas 20 horas para el dispositivo móvil. Suponemos que la vida útil de todos nuestros dispositivos es de 3 años y que los costes son de 600 € y 1400 € para el dispositivo móvil y los ordenadores, respectivamente.
- El coste energético diario de un ordenador corriente es de 0,437 kWh (activo) más 0,1524 kWh (apagado) [48]. Basándonos en estas referencias teóricas, los tres meses de trabajo cubiertos en el proyecto (90 días) sumarian un total de 53,046 kWh. Multiplicando esta cantidad por el coste energético en España (0,15€) tendríamos un total de 7,96 € por ordenador.
- Se han considerado también los recursos: Internet y telefonía. Aunque sean recursos típicos son necesarios para el correcto funcionamiento de la aplicación (acceso a base de datos, repositorio del código...).
- Se consideran un margen de imprevistos y un margen de beneficios en el precio final, que hacen que este aumente su valor.
- No se han considerados gastos software pues las herramientas utilizadas para la programación de esta son libres.

Sumando todo esto, nos sale un total de 2399,56 €.

PRESUPUESTO			
MATERIALES			
MATERIAL	**UNIDADES**	**COSTO UNITARIO**	**COSTO TOTAL**
Plancha cartón una onda (40x40cm)	1	2,00 €	2,00 €
Plancha cartón doble onda (50X85cm)	1	2,68 €	2,68 €
Caja desmontable Leroy Merlin	1	5,65 €	5,65 €
MANO DE OBRA (SOFTWARE Y SOPORTE)			
PERSONAL	**HORAS TRABAJO**	**COSTO UNITARIO**	**COSTO TOTAL**
Analista (David Suárez Suárez)	40	6,25 €	250,00 €
Diseñador (David Suárez Suárez)	40	6,25 €	250,00 €
Desarrollador (David Suárez Suárez)	160	6,25 €	1.000,00 €
Tester (David Suárez Suárez)	20	6,25 €	125,00 €
HARDWARE			
CONCEPTO	HORAS USO	COSTO UNITARIO	COSTO TOTAL
Depreciación o uso de los computadores usados para el desarrollo	240	0,05 €	12,79 €
Depreciación o uso de los dispositivos móviles usados para las pruebas	20	0,02 €	0,46 €
SERVICIOS			
CONCEPTO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Energía por dispositivo	3	7,96 €	23,88 €
Internet y telefonía (meses)	3	35,00 €	105,00 €

SUBTOTAL		1.777,45 €
Porcentaje Imprevistos	10%	177,75 €
Porcentaje Beneficios	25%	444,36 €
TOTAL PROYECTO		**2.399,56 €**

Figura A8.12: Presupuesto.