Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

# APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES

**Allan Marcelo Noguera Torres**
Asesorado por el Ing. Enio Fabrizio Torres Noguera

Guatemala, febrero de 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

**ALLAN MARCELO NOGUERA TORRES**
ASESORADO POR EL ING. ENIO FABRIZIO TORRES NOGUERA

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, FEBRERO DE 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA


## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANA | Inga. Aurelia Anabela Cordova Estrada |
| VOCAL I | Ing. José Francisco Gómez Rivera |
| VOCAL II | Ing. Mario Renato Escobedo Martinez |
| VOCAL III | Ing. José Milton de León Bran |
| VOCAL IV | Br. Christian Moisés De La Cruz Leal |
| VOCAL V | Br. Kevin Vladimir Armando Cruz Lorente |
| SECRETARIO | Ing. Hugo Humberto Rivera Pérez |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Pedro Antonio Aguilar Polanco |
| EXAMINADOR | Ing. Edgar Estuardo Santos Sutuj |
| EXAMINADOR | Ing. Sergio Arnaldo Méndez Aguilar |
| EXAMINADOR | Ing. William Samuel Guevara |
| SECRETARIA | Inga. Lesbia Magalí Herrera López |

# HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

**APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES**

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 27 marzo de 2019.

**Allan Marcelo Noguera Torres**

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 14 de junio de 2019

Ingeniero
Carlos Alfredo Azurdia Morales
Tutor de trabajos de graduación

Respetable Ingeniero Azurdia:

Por este medio le informo como asesor del trabajo de graduación del estudiante universitario de la carrera de Ingeniería en Ciencias y Sistemas, ALLAN MARCELO NOGUERA TORRES, carné 200715118, que he revisado el trabajo de graduación titulado: "APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES", y a mi criterio el mismo está completo y cumple con los objetivos propuestos para su desarrollo según el protocolo.

Agradeciendo su atención a la presente,

Atentamente,

Ing. Enio Fabrizio Torres Noguera
Asesor de trabajo de graduación
Colegiado: 14296

Enio Fabrizio Torres Noguera
Ingeniero en Ciencias y Sistemas
Colegiado No. 14,296

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de septiembre de 2019

Ingeniero
**Carlos Gustavo Alonzo**
**Director de la Escuela de Ingeniería**
**En Ciencias y Sistemas**

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **ALLAN MARCELO NOGUERA TORRES** con carné **200715118 y CUI 1928 73121 0101** titulado **"APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

**Ing. Carlos Alfredo Azurdia**
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24188000 Ext. 1534

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación,* ***"APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES"*** *realizado por el estudiante, ALLAN MARCELO NOGUERA TORRES, aprueba el presente trabajo y solicita la autorización del mismo.*

***"ID Y ENSEÑAD A TODOS"***

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

*MSc. Ing. Carlos Gustavo Alonzo*
***Director***
***Escuela de Ingeniería en Ciencias y Sistemas***

*Guatemala, 10 de febrero de 2020*

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

**Decanato**
**Facultad de Ingeniería**
**24189102 - 24189103**

DTG. 053.2020

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **APLICACIÓN MÓVIL EN ANDROID PARA LA TEMPRANA DETECCIÓN DEL DALTONISMO Y EL TRASTORNO DE LA DISLEXIA, DIRIGIDA A NIÑOS DE SEIS A OCHO AÑOS Y PARA VISUALIZACIÓN DE LOS RESULTADOS DEL DIAGNÓSTICO, DIRIGIDO A LOS PADRES,** presentado por el estudiante universitario: **Allan Marcelo Noguera Torres,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

Guatemala, febrero de 2020

/gdech

# ACTO QUE DEDICO A:

| | |
|---|---|
| **Dios** | Por permitirme explorar y crecer en este universo que él creo para nosotros. |
| **Mis padres** | Rosa Torres y Marcelo Noguera, por acompañarme y apoyarme en este camino llamado vida. |
| **Mi esposa** | Ana Avalos, por transmitirme las fuerzas para culminar mi carrera. |
| **Mi abuela** | María Hilda Vásquez, por preocuparse por mí en los días en que trabajaba hasta tarde. |
| **Mis abuelos** | Teodoro Torres (q.e.p.d.) y Zoila Sagastume (q.e.p.d.), cuyas enseñanzas ahora forman parte de mi persona. |
| **Mis tíos** | Enio, Brenda, Eddy (q.e.p.d.), Amabilia Torres, Adonay, Sarai, Rafael, Eliseo, Olga, Rosaura, Maura, Mirtala Noguera, Cesar Yupe y Wendy Jiménez, por los consejos y conocimientos de vida que me brindaron. |

# AGRADECIMIENTOS A:


| | |
|---|---|
| **Universidad de San Carlos de Guatemala** | Por fomentar en mí los valores de esta casa de estudios. |
| **Facultad de Ingeniería** | Por transmitirme su conocimiento y su espíritu. |
| **Mis amigos de la Facultad** | Por alegrar con su carisma nuestro día a día en la Universidad. |
| **Mi hermano** | Por motivarme con su ejemplo a seguir estudiando. |
| **Mis primos** | Por compartir los buenos y malos momentos de la vida y de esta carrera. |

# ÍNDICE GENERAL

# ÍNDICE DE ILUSTRACIONES

## FIGURAS

# GLOSARIO

| | |
|---|---|
| **Clase** | Elemento básico de programación que abstrae atributos y funcionalidades. |
| **Cache** | Tipo de memoria volátil de rápido acceso. |
| **Interfaz** | Conexión funcional entre dos sistemas. |
| **iOS** | Sistema operativo para móviles de la compañía Apple Inc. |
| **Singleton** | Patrón de diseño de programación que asegura que solo una referencia a ese objeto sea creada. |
| **Variable** | Unida elemental en la programación que conserva un valor o una referencia a una dirección de memoria. |
| **Ventana** | Componente funcional gráfico de la aplicación que se muestra en pantalla al usuario. |

# RESUMEN

El daltonismo es una enfermedad que afecta al 2 % de la población a nivel mundial, número que asciende a 7 550 millones de personas (según el último censo realizado por las Naciones Unidas en 2017), por lo que, se puede deducir que alrededor de 150 millones de personas padecen de esta enfermedad. La dislexia por su parte es estimada en más del 10 % de la población, alrededor de 755 millones de personas en el mundo.

La tecnología es la herramienta con la cual se ha resuelto muchas problemáticas. Los usuarios esperan que las nuevas tecnologías logren satisfacer una necesidad, necesidades que por lo general los usuarios tienen inconscientemente. Como las aplicaciones móviles que remueven la subjetividad humana se pueden utilizar para calificar pruebas.

Debido al desconocimiento generalizado de estas enfermedades es difícil identificarlas a una temprana edad, lo que provoca que muchos niños pasen por muchas dificultades en el sistema educativo, causando problemas emocionales en las personas que lo padecen. Con la finalidad de brindar una herramienta para promover el aprendizaje sobre estas enfermedades (para los padres) y para proporcionar una base para identificar una deficiencia en el desempeño visual de los niños, se crea una aplicación para teléfonos con sistema operativo Android que presente pruebas sencillas para que niños entre 6 y 8 años puedan realizarlas sin supervisión; y así los padres podrán ver el desempeño de sus hijos. Lo que se espera es crear una mayor concienciación sobre estas enfermedades visuales, para que los padres acudan con un médico y puedan adaptar la educación de sus hijos.

# OBJETIVOS

## General

Crear una herramienta que promueva la concientización de los padres de familia sobre la dislexia y el daltonismo que los niños en edad primaria pueden padecer, y que podrían afectar su desempeño académico.

## Específicos

1. Brindar una herramienta a los padres para que apliquen pruebas conocidas para la detección de la dislexia y el daltonismo en niños de entre 6 y 8 años.

2. Utilizar tecnología móvil de alto alcance para difundir que el daltonismo y la dislexia pueden afectar a los niños en su desempeño académico.

3. Promover la concientización sobre las enfermedades del daltonismo y la dislexia en los niños.

# INTRODUCCIÓN

Algunas personas recordarán su educación primaria como uno de los mejores tiempos de su vida, mientras que otras personas pasaran por estrés y ansiedad que harán que las personas prefieran olvidar esta etapa.

Un estudio revela “el nivel de ansiedad en niños que cursan por primera vez primero primaria, es un nivel moderado, puesto que 40 % de la muestra punteó 64,2 en el baremo de la adaptación española del CAS. Mientras que el 20 %, mostró indicios de ansiedad severa; y el resto de la muestra se ubicó entre ansiedad leve o bien no presentó sintomatología en relación a la ansiedad”[1]. Evidenciando que el 60 % de los niños presentaron ansiedad.

Considerando que la Universidad Internacional de Valencia clasifica a la dislexia como una de las principales causas del bajo rendimiento escolar. Siendo esta enfermedad padecida por un 10 % de la población mundial. Si a todo esto, se le agrega que un 2 % de la población puede padecer daltonismo, se tiene un conjunto de motivos que pueden causar o aumentar la ansiedad en los niños al aumentar el estrés por su bajo desempeño en el sistema educativo estándar.

Utilizando una de las herramientas que brinda la tecnología, como lo es las aplicaciones para dispositivos móviles Android, se plantea crear una herramienta que sirva como primer acercamiento de los padres para conocer este tipo de afecciones, las cuales pueden pasar desapercibidas incluso hasta

[1] QUIROS, A. *Ansiedad en niños que cursan por primera vez primero primaria, comprendidos entre 6-8 años en AMG Verbena zona 7*. 37p.

la educación secundaria, y esta aplicación pueda servir como primera prueba que motive a los padres a buscar ayuda médica.

# 1. ESTUDIO DE LA TECNOLOGÍA Y SU IMPACTO EN GUATEMALA

En este capítulo se relacionará la aplicación móvil a desarrollar con la teoría de investigación que mejor se adapta a la innovación, así como al impacto que esta tendría en la población objetivo de este trabajo.

Para asentar la teoría de este trabajo se tomarán en cuenta dos de las teorías utilizadas en la investigación de sistemas de información. La primera teoría que considerar es la teoría de la difusión de innovación, que trata sobre la voluntad de los individuos para adoptar nuevas herramientas tecnológicas; la otra teoría a considerar es la teoría de satisfacción sobre el cambio de rendimiento, la cual trata de medir una emoción, el sentimiento de satisfacción.

## 1.1. Teoría de la difusión de la innovación

Esta teoría trata la innovación como algo que se distribuye de persona a persona, e identifica a estas personas según el tiempo en el que adoptan el cambio. La teoría también se le conoce como DOI (*Diffusion of Innovation Theory*), esta teoría define que la adopción de la tecnología sigue una tendencia curva; esta curva muestra la tasa de adopción de la tecnología a través del tiempo.

La teoría de la difusión de la innovación hace distinción de las personas, según el momento en el que estos adoptan la tecnología y las distinciones que se hacen son: innovadores, adoptadores tempranos, mayoría temprana,

mayoría tardía, y rezagados. Y además, define que hay cinco factores que influencia la adopción de la tecnología en los usuarios innovadores.

Figura 1. **Proceso de difusión**

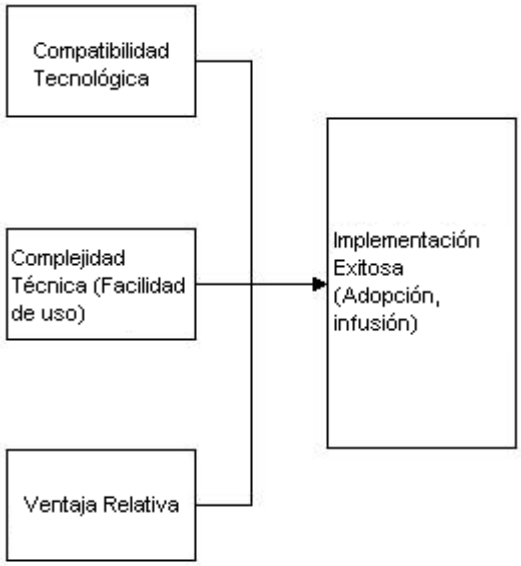

Fuente: ROGERS, Everett M. *Diffusion of Innovations*. http://is.theorizeit.org/wiki/File:Doi1.JPG. Consulta: 12 de junio de 2019.

Los factores que influencia a los usuarios innovadores son: la ventaja relativa, compatibilidad, capacidad de ser probada, visibilidad y complejidad. De estos factores la ventaja relativa, compatibilidad, visibilidad y capacidad de ser probada están recíprocamente relacionada con la adopción; pero la complejidad está relacionada de forma inversamente proporcional a la adopción.

La teoría de la difusión de la innovación también recalca que los sistemas de información proporcionan un valor, y solo los proyectos que tienen un alto valor o un valor que incrementa con el tiempo tendrán una tasa de adopción alta. Y esta teoría no descarta que la adopción se vea truncada por otra innovación.

La difusión de los sistemas de información sigue un modelo, el cual lleva a una implementación exitosa de un sistema de información, esto quiere decir que el sistema será adoptado o infundido. Los tres pilares sobre los que esta implementación exitosa se basa son: la compatibilidad técnica, la facilidad de uso y la ventaja relativa. Esto se muestra en la figura a continuación.

Figura 2. **Modelo de variación de la difusión de los sistemas informáticos**




Fuente: AGARWAL, R., Prasad, J. *The role of innovation characteristics and perceived voluntariness in the acceptance of information*.
http://is.theorizeit.org/wiki/File:Doi2.JPG. Consulta: 12 de junio del 2019.

Como se puede ver, se debe concentrarnos en la necesidad percibida, asimismo hay que tomar en cuenta la complejidad (enfocándose en la facilidad de uso), y la compatibilidad del sistema con el usuario.

## 1.2. Teoría de satisfacción sobre el cambio de rendimiento

La teoría de satisfacción sobre el cambio de rendimiento también es conocida como YTS (*Yield Satisfation Theory*). Esta teoría trata la adopción de una tecnología como un fenómeno de interés, o en otras palabras la respuesta emotiva de la satisfacción.

Pero, en esta teoría se unifican los buenos sentimientos, que anteriormente se referían a la satisfacción, y los malos sentimientos, antes referidos como insatisfacción; y se unifican en un término que es la respuesta de satisfacción. Este término se maneja entre dos límites, desde no despertado hasta despertado. Esta medida se maneja como un valor que va de un lado al otro, sin pasar por un punto medio o neutral; esto quiere decir, que una persona no puede estar en un punto intermedio, sino que directamente esta insatisfecha o satisfecha.

La conducta humana dicta que todas las metas que una persona puede tener no están siempre presentes en la conciencia; por lo que, el subconsciente maneja un grupo de metas, las cuales se conocen como conjunto de metas activas. Por lo tanto, la teoría de satisfacción sobre el cambio de rendimiento remarca que la satisfacción ocurre al percibir un cambio (positivo) en alguna de las metas del conjunto activo.

La teoría de satisfacción sobre el cambio de rendimiento se basa en cinco suposiciones.

La primera suposición, evaluación automática de la utilidad, asume que hay un mecanismo subconsciente que le asigna un nivel de utilidad al sistema para alcanzar una de las metas del conjunto activo. La suposición de la

evaluación automática de la probabilidad espera que exista un mecanismo inconsciente que evalúa la probabilidad de alcanzar una meta. Y la tercera suposición asume un mecanismo que predice el resultado del sistema, pero este último afecta inversamente a la segunda suposición.

La cuarta y quinta suposición entran en acción, ya que el sistema está siendo utilizada, a diferencia de las primeras tres suposiciones que ocurren en el momento en el que se descubre el sistema. La suposición de la detección automática del cambio de desempeño enmarca un mecanismo inconsciente que detecta la magnitud y dirección de los cambios en el rendimiento. Por último, la suposición de la respuesta afectiva a los cambios de rendimiento explica que el individuo cuando detecta el cambio en el rendimiento en el conjunto de metas activas se inicia un proceso que le asigna un valor, ya sea positivo o negativo, a este cambio.

Por consiguiente, la teoría de satisfacción sobre el cambio de rendimiento define que la satisfacción es relativa a la dirección en la que se mueva la percepción del cambio en el rendimiento. Lo que dicta una correlación directa entre el avance en el cumplimiento de las metas, y el sentimiento positivo hacia el sistema de información.

En esta teoría hay diez explicaciones para explicar qué tipo de efecto disparó el sentimiento positivo, para este trabajo se enfocará en el efecto de la confirmación, este efecto define que el usuario se siente satisfecho, el resultado coincide con su resultado esperado; en este caso es confirmar o anular su sospecha de alguna afección en sus hijos.

## 1.3. Teoría y la relación con la tecnología escogida

Para realizar la aplicación del proyecto, para ayudar a la detección temprana del daltonismo y la dislexia, nace del poco estudio que estas enfermedades tienen en Guatemala, así como la poca difusión informativa que estas enfermedades tienen en el país. Por estos motivos se desarrollará una aplicación para dispositivos con sistema operativo Android.

Según la teoría de la difusión de la innovación remarca que los sistemas de información deben de brindar una ventaja relativa, así como una baja complejidad (entre otros factores). Por lo que, la aplicación deberá brindar fragmentos de información que antes era imperceptibles para el usuario; el usuario final de esta aplicación serían los padres, los cuales tienen interés en confirmar o descartar alguna de estas enfermedades en sus hijos.

Basándose en la curva del proceso de difusión (véase figura 1), se espera que la aplicación encuentre un nicho de usuarios innovadores; quienes, al encontrar valor en la ventaja relativa del sistema, se encargarán de comunicar a otros (adoptadores tempranos). Siguiendo de esta manera la trayectoria de despegue de la curva, para difundir la información acerca de las enfermedades.

Se desarrollarán dos módulos, para la aplicación, un módulo de juego y un módulo de reportes e información. El primer módulo tiene el nombre código de ‘Explora’; este módulo será al que los niños tendrán accesos; por lo cual la facilidad de uso será la premisa; la forma en la que facilitaría la realización de estas pruebas el módulo de explora es la de selección múltiple, ya que al tener con qué comparar, los niños podrán identificar la respuesta correcta de entre las opciones.

El segundo módulo solo dará acceso para los padres, en el cual podrán ver gráficas que representan el desempeño de sus hijos e información relacionada con las enfermedades de la dislexia y el daltonismo.

La aplicación está dirigida hacia los padres de familia que están interesados por velar en el bienestar y la salud de sus hijos; pero estos usuarios solo serán usuarios esporádicos de la aplicación, por lo que, se referirá a ellos como usuarios esporádicos. Y los hijos de estas personas, que se encuentren entre las edades antes figuradas, serán los que utilicen la aplicación a diario, a estos se referirá como usuarios finales.

El módulo de Explora tendrá una variedad de juegos, los cuales estarán diseñados para una enfermedad en específico. Este módulo está diseñado para el usuario final (diseñada para niños de entre seis y ocho años, de habla hispana), debido a esto, este trabajo se guía según la teoría de la difusión de la innovación, que se enfoca en la baja complejidad de uso.

El segundo módulo, módulo de control parental, también seguirá la línea de la facilidad; aunque, en este módulo ya se puede enfocar en más aspectos, como: valor relativo, visibilidad, compatibilidad, capacidad de ser probada. Por lo que, en este módulo, se hará énfasis en la facilidad de ver los puntajes (visibilidad), y así poder obtener la respuesta a sus hipótesis sobre la salud de sus hijos.

Ahora entra en acción la teoría de satisfacción sobre el cambio de rendimiento, la cual se seguirá para que la aplicación tenga una buena aceptación entre la población. Entonces, en el módulo de control, se enfocará en mostrar la utilidad de la aplicación, de tal forma que sea fácil percibirla; de

esta manera lograr que el mecanismo inconsciente del padre pueda percibir la utilidad.

Las siguientes suposiciones de la teoría de satisfacción sobre el cambio de rendimiento son la evaluación automática de probabilidad, y la suposición del mecanismo de predicción del resultado; por lo que, el módulo de control debe mostrar la información de la forma más efectiva posible, de forma directa y concreta esto para que los padres puedan percibir la factibilidad de cumplir su objetivo.

Para cumplir las últimas dos suposiciones de esta teoría, se mostrarán los resultados del juego de los niños de manera gráfica, en forma de diagramas de sectores para mostrar la probabilidad de que el niño padezca alguna de las enfermedades, y en forma de diagramas de barras simples para mostrar de manera histórica los resultados de los niños en cada aspecto estudiado en el juego.

La aplicación entonces, estará desarrollada de tal manera que una vez se haya empezado a usar, los padres puedan percibir un cambio en el rendimiento de sus hijos. Y al desmentir o confirmar las conjeturas previamente realizadas a la utilización de este sistema, se estaría cumpliendo la última suposición de la respuesta afectiva; al poder tener de una forma relativamente más sencilla a la información que previamente era inaccesible.

# 2. IDENTIFICACIÓN DE PROBLEMA Y SOLUCIÓN QUE LA APLICACIÓN REALIZARÁ

## 2.1. Antecedentes

La enfermedad del daltonismo fue nombrada así tras las publicaciones de John Dalton que describía sus propias afecciones, estas publicaciones fueron hechas en 1794, entre estas afecciones se encontraban la de confundir el color corinto con el verde y el rosado con el azul, entre otras. De esta enfermedad no existe solo un tipo, sino que existen varios, por lo que, los individuos no siempre confunden los mismos colores; esto causa que no siempre se distinga cuando una persona es daltónica.

El daltonismo, también conocido como ceguera del color, es una enfermedad que padecen más los hombres, debido a que es una mutación que ocurre en el cromosoma X, y dado que las mujeres tienen dos de estos cromosomas y en el caso de los hombres solo uno, si una mutación ocurre en el cromosoma de los hombres los pigmentos de los conos verdes y rojos se verían afectados. Esto conlleva que un 1,5 % de hombres padezca esta enfermedad contra un 0,5 % de las mujeres.

Los tipos de esta enfermedad están directamente relacionados con los tipos de cono (conos verdes o conos rojos) que fueron afectados por la mutación del cromosoma X, por lo que, no logran captar con precisión las longitudes de onda que el cerebro interpretaría como los colores azul, rojo y verde. Por cada tipo de cono existe un tipo de daltonismo: El tipo dicromatismo, es en el que las personas solo poseen dos de los tres tipos de conos, y esto

causa confusión entre los colores azul y amarillo, así también como entre el rojo y el verde. El tipo Tricromatismo anómalo, este tipo de daltonismo solo presentan deficiencia en alguno de los tres tipos de cono, por lo que, los síntomas no son tan severos, pero existe un bajo desempeño en la diferenciación de los colores. El último tipo de daltonismo, el tipo acromatopsia, es considerado el tipo más severo de esta enfermedad, ya que la persona que la padece solo logra percibir la diferencia en escala de grises.

El método de diagnóstico más utilizado para el daltonismo es el de las cartas de Ishihara, nombradas así en honor al doctor que los diseños, Shinobu Ishihara, estas publicaciones fueron creadas en 1917. Esta prueba consiste en una secuencia de cartas con círculos de colores, estos círculos son de tamaño aleatorio, los cuales forman con patrones un número que es invisible solo para las personas que presentan la deficiencia en los conos visuales. La prueba completa está formada por 38 cartas, pero la deficiencia de la visión es fácilmente detectable con tan solo 24 de estas.

Por otra parte, la dislexia se estima que es padecida por mas del 10 % de la población mundial. La dislexia es la dificultad de la persona en la lectura y que dificulta o imposibilita la correcta comprensión. Esta enfermedad no tiene una causa definida, pero existen factores que predisponen a una persona a padecerlos, entre algunos: problemas emocionales, lesiones cerebrales, causas genéticas, entre otros. Lo que sí se ha logrado identificar es que la razón biológica que causa este déficit ocurre en el lóbulo parietal, en el hemisferio cerebral izquierdo; el cual demuestra en las personas con esta enfermedad un desempeño pobre.

Al igual que con el daltonismo, existen varios tipos de dislexia, pero en esta enfermedad se puede tipificar la enfermedad según varios criterios, entre

estos: según el tipo de síntoma predominante y según el momento de diagnóstico. Debido a que este proyecto trata de acelerar la detección de la dislexia se tipificará el diagnóstico por el síntoma.

Los tipos de dislexia son: dislexia superficial, en la cual la persona que la padece trata de leer las palabras a partir de segmentos más pequeños dentro de la misma, segmentos conocidos; quienes padecen de este tipo de dislexia, tienen problemas con las palabras cuya pronunciación no está directamente relacionada con su escritura. El otro tipo de dislexia es la fonológica, en la cual la persona que la padece trata las palabras como un todo, y la persona tiende a cometer errores léxicos, o de palabras derivadas.

En el momento del diagnóstico de la dislexia, hay que tomar muchos aspectos en cuenta, lo principal es descartar problemas en la visión o la audición, así como perturbaciones emocionales. Algunas de las pruebas que se pueden realizar como indicadores iniciales, para emitir un posterior diagnóstico son: la prueba de la figura humana de *Goodenough* y también se puede utilizar la prueba de *Frostig*. Con estas pruebas se puede recabar datos suficientes para identificar si un niño tiene indicios de dislexia.

## 2.2. Mercado objetivo

La aplicación estará dirigida a niños que inician la primaria, o que están por iniciarla, y los padres de estos. Las edades estimadas para los niños en esta etapa son de entre seis y ocho años; la aplicación estará diseñada para los hispanohablantes.

La aplicación estará desarrollada para dispositivos que corran el sistema operativo Android, por lo que, los usuarios deben de poseer un dispositivo para utilizar el módulo de Explora.

Tanto el encargado como el niño, compartirán el dispositivo, esto se realiza para facilitar el uso. Ya que si se separan los módulos en aplicaciones diferentes los usuarios percibirían como desperdicio de los módulos, el cual no sería utilizado todos los días (debido a la finalidad de los reportes).

Por lo tanto, los usuarios tendrán en un dispositivo Android, el módulo de Reportes que podrá mostrar los resultados en el mismo dispositivo; pero esta opción estará restringida para los niños a través de una contraseña, ya que los niños no tienen necesidad de saber esta información.

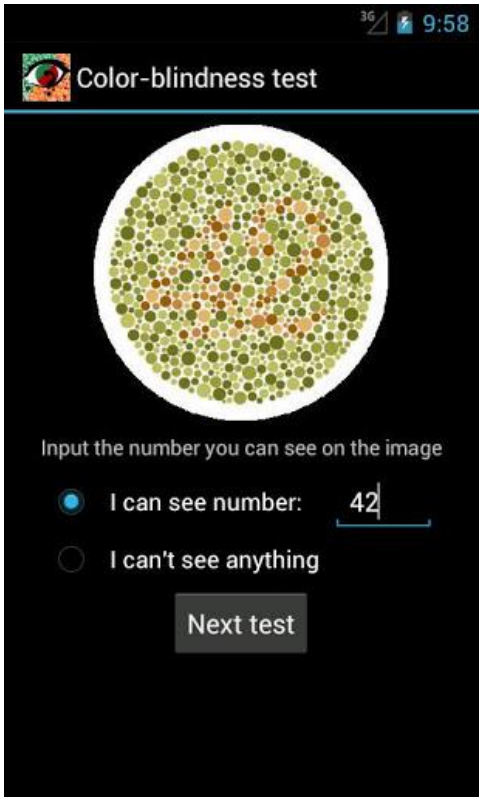
## 2.3. Evaluación comparativa de la aplicación

En estos tiempos existen muchos exámenes y pruebas para diagnosticar las enfermedades del daltonismo y la dislexia. Las pruebas que se realizan para diagnosticar estas enfermedades son mayormente visuales, y en el caso de la dislexia, algunas son auditivas. Lo único que se necesita es un evaluador objetivo que califique si la respuesta es la correcta, porque la prueba está diseñada de tal manera que elimina totalmente la subjetividad de esta, por lo que, ninguna persona con esta enfermedad podría contestar correctamente.

Debido a que las pruebas son tan precisas y quitan del medio la subjetividad, hacen que estas pruebas puedan ser programadas; por lo que, se encuentran muchas aplicaciones web, o aplicaciones móviles cuya finalidad es la diagnosticar estas afecciones. Estas aplicaciones están dirigidas a personas

que se encuentran en la adolescencia o en la adultez, que sienten la curiosidad de saber si padecen alguna de estas afecciones.

Figura 3. **Aplicación de muestra para diagnóstico de daltonismo**




Fuente: http://guiacirugiaestetica.com/wp-content/uploads/2013/05/daltonismo-test-prueba-aplicacion-medica-android.jpg. Consulta: 12 de junio del 2019.

En el internet se puede encontrar páginas en las que se realiza las pruebas de las cartas de Ishihara, esto para diagnosticar el daltonismo; y esto consta de solo imágenes estáticas en la cual uno mismo puede evaluar si logra leer o ver los números en dichas tarjetas.

Se pueden encontrar páginas en las que hay pruebas auditivas y de lectura en el internet para poder diagnosticar la dislexia en los niños, en páginas como *Lexercise* (www.lexercise.com), se prueba la habilidad de lectura y auditiva; dichas habilidades están en formación para los niños que inician la educación formal, por lo que, estas pruebas no están dirigidas al grupo objetivo de esta aplicación, niños de seis a ocho años de edad.

Por lo tanto, no hay una aplicación que realice las pruebas que esta aplicación pretende realizar, ni de la forma en que se van a realizar; por lo que, no se cuentan con un competidor directo. Esto presenta una gran ventaja para que la aplicación tenga una buena cantidad de usuarios, que encuentren beneficio y facilidad de uso en este sistema.

# 3. DISEÑO DE LA APLICACIÓN BAJO LA NECESIDAD IDENTIFICADA

En este capítulo se trata de describir cómo se podría ver la aplicación cuando esté terminada. Para esto se utilizará la técnica de prototipado, la cual ayuda a la visualización y conceptualización de la apariencia final de la aplicación como tal.

Para cada módulo de la aplicación se presentará un prototipo gráfico, para mostrar la estructura básica de cada ventana de la aplicación, y así apreciar lo importante de cada una.

## 3.1. Ventana de bienvenida juego

La primera aproximación que el niño tendrá con el juego de Mi Aventura será con la ventana de bienvenida, o *splash screen* por su nombre en inglés. Esta ventana solo servirá para alentar al niño a utilizar la aplicación, ya que esta le brindará una bienvenida al mostrar una cara sonriente.

Esta ventana solo estará unos segundos presentes, después de este tiempo, automáticamente el usuario será llevado a la pantalla de menú, la cual se describirá en la siguiente sección.

Figura 4. **Ventana de bienvenida juego**

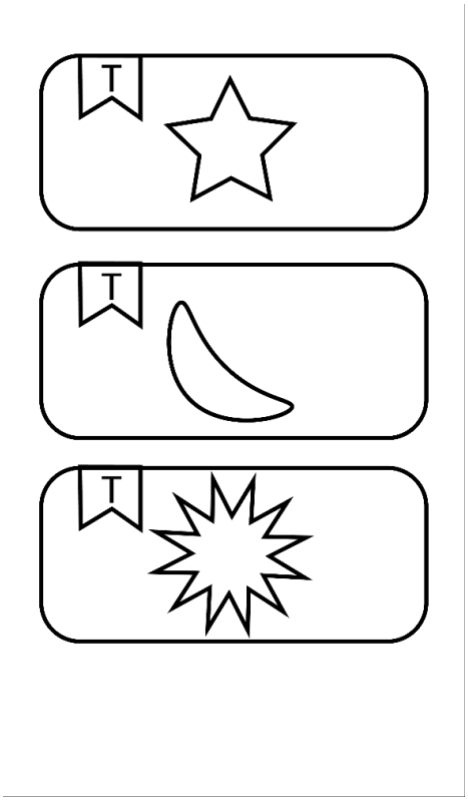

Fuente: elaboración propia, empleando Adobe Ilustrador.

## 3.2. Ventana de menú juegos

En esta ventana el usuario podrá visualizar la lista de juegos disponibles, esto se realizará con la menor cantidad de texto posible, ya que la aplicación está diseñada para niños que están aprendiendo a leer. Por esta razón, los juegos estarán representados por un tema identificador, además, el tutorial para cada juego estará fácilmente accesible desde el menú.

La ventana de menú estará dispuesta como una lista, por lo cual solo los juegos principales serán mostrados en la pantalla, al inicio, y el usuario deberá navegar hacia abajo, para poder ver los demás juegos disponibles.

Figura 5. **Ventana de menú juegos**



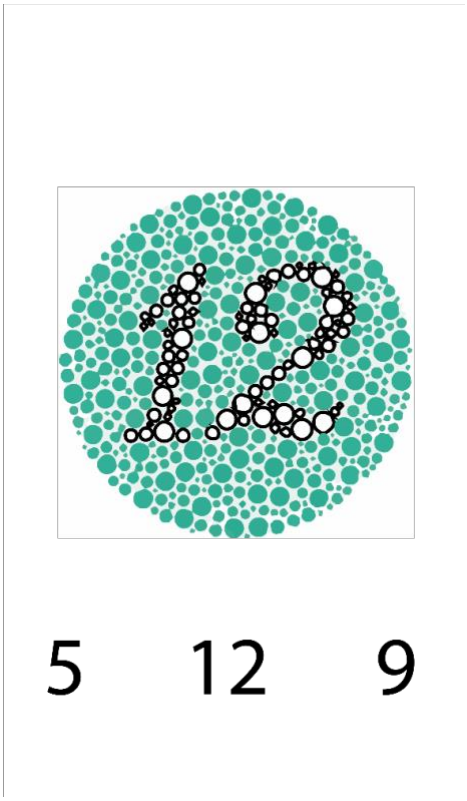

Fuente: elaboración propia, empleando Adobe Ilustrador.

Como se puede apreciar en la imagen anterior (véase figura 5), cada ícono será representativo de su juego y, además, cada uno contará con una pequeña marca con la cual podrán acceder al tutorial de dicho juego en cualquier momento. Al seleccionar el juego, o luego de pasar el tutorial del juego seleccionado, el usuario será dirigido a dicho juego.

### 3.3. Juego para la detección del daltonismo

Para esta ventana, solo se contarán con dos aspectos importantes dentro de la misma: primero la pregunta, con esto nos se refiere a la imagen de la prueba de Ishihara que será la que el usuario debe de responder y, la segunda: las posibles respuestas, para lo que el usuario contará con tres opciones de las cuales solo una es correcta. La ventana de este juego será presentada por un tiempo de hasta 15 segundos.

Figura 6. **Juego para la detección del daltonismo**




Fuente: elaboración propia, empleando Adobe Ilustrador.

Para ver la siguiente pregunta, el usuario debe responder la pregunta actual, esto será registrado por la aplicación para luego mostrar la gráfica de resultado. Mientras más preguntas correctas contesten el puntaje será mayor, lo cual motivará al usuario a seguir jugando.

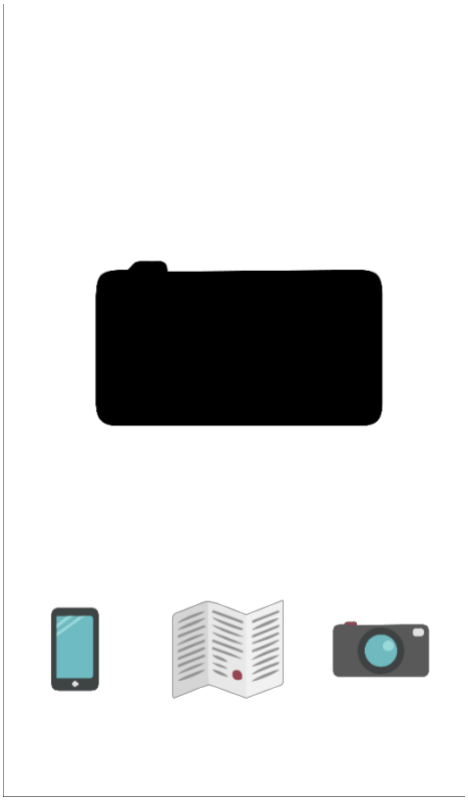
Al finalizar el tiempo, el usuario podrá ver su puntuación en la misma pantalla. Luego, el usuario será llevado nuevamente a la ventana de menú, esto para que el usuario pueda cambiar de juego, y no sienta fatiga al repetir un mismo juego.

### 3.4. Juego para la detección de dislexia

Para esta ventana solo se contarán con dos aspectos importantes dentro de la misma: primero la pregunta, con esto se referie a la imagen del objeto que el usuario debe de determinar y, la segunda: las posibles respuestas; las respuestas, en este caso, estarán representadas por las imágenes completas de los posibles objetos que dibujen la sombra presentada en la pregunta; para lo cual el usuario contara con tres opciones, de las cuales solo una es correcta. La ventana de este juego será presentada por un tiempo de hasta 15 segundos.

Para ver la siguiente imagen o pregunta, el usuario debe de responder la pregunta actual, esta respuesta será registrada al igual que en el juego anterior para dar un resultado. Mientras mayor sea el número de preguntas correctas que contesten el puntaje será mayor.

Figura 7. **Juego para la detección de dislexia**



Fuente: elaboración propia, empleando Adobe Ilustrador.

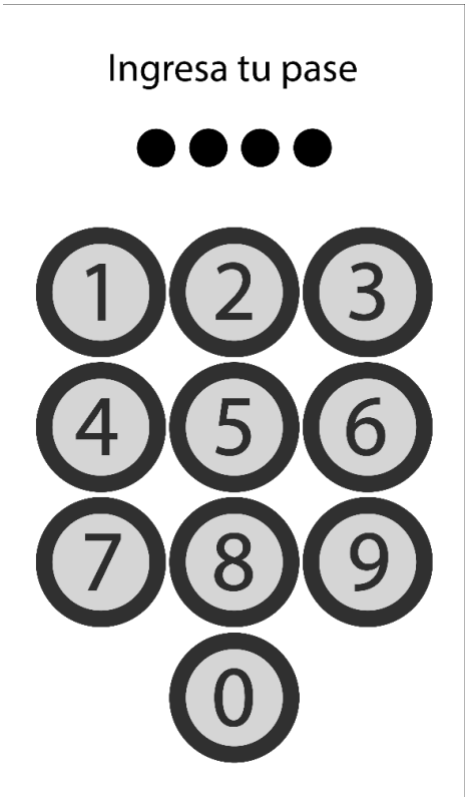
Al finalizar el tiempo el usuario podrá ver su puntuación en la misma pantalla. Luego, el usuario volverá nuevamente a la ventana de menú, como en el juego anterior.

## 3.5. Ventana de seguridad

Esta ventana ocultará la información de los reportes a los infantes, la idea es que solo la persona encargada de los niños tenga acceso. Esta ventana presentará una capa de seguridad a través de una contraseña numérica, la cual

debe ser definida por el encargado de los infantes. El patrón de esta contraseña será numérico con una longitud de cuatro caracteres.

Figura 8. **Ventana de seguridad**




Fuente: elaboración propia, empleando Adobe Ilustrador.

De esta ventana solo se podrá proceder si la persona proporciona correctamente la contraseña.

El propósito de esta pantalla es la de ocultar todo el propósito final de los juegos a los infantes, y que solo los padres puedan acceder a la siguiente ventana que muestra los reportes.

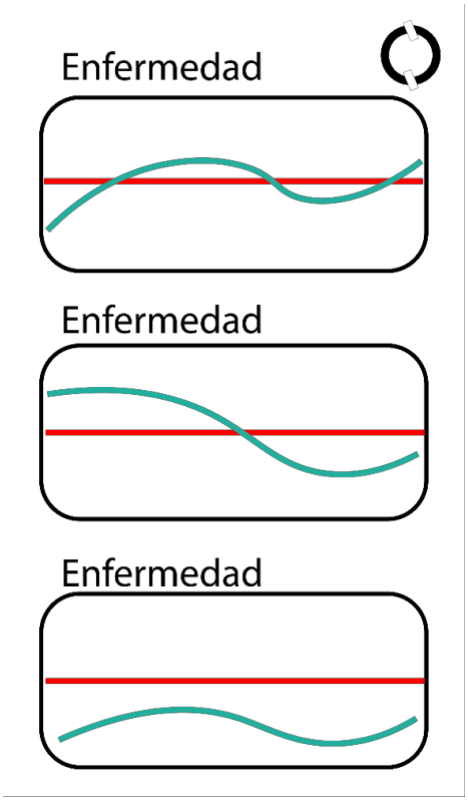
### 3.6. Ventana de reportes

Esta ventana, al igual que la ventana del menú de juego, presentará la información en forma de listas; lista de gráficas de los resultados de los juegos que cuentan con datos que pudieron ser descargados. Esta lista de gráficas tendrá un título el cual indicará la enfermedad que trata de describir.

Las gráficas serán sencillas, por lo que, solo indicarán el punto medio esperado como resultado de los usuarios, y el histograma de resultados. Esto con la finalidad que el padre o tutor, tenga una herramienta para ver progresos o retrocesos en los resultados y así poder tomar una decisión acerca de que sería pertinente, dados los resultados observables.

Dado que la idea no es la de sustituir el diagnóstico médico con esta aplicación, sino solo brindar una observación más detallada e histórica de las manifestaciones físicas de los síntomas. Por lo tanto, no se debe confundir el resultado con una aseveración, sino más como una sugerencia. Por lo que, el resultado de la aplicación es siempre muy susceptible a la subjetividad del observador; esta ayuda solo dejará ver más allá de los indicadores espontáneos en el diario vivir de una familia. Y de este modo facilitar la detección de las enfermedades antes mencionadas.

Figura 9. **Ventana de reportes**




Fuente: elaboración propia, empleando Adobe Ilustrador.

Para salir de esta ventana, el usuario solo deberá cerrar la aplicación, debido a que este módulo solo sirve para mostrar los resultados.

# 4. DOCUMENTACIÓN Y TUTORIAL DE PROGRAMACIÓN DE LA APLICACIÓN

## 4.1. Sistema operativo para teléfonos móviles

Android es un sistema operativo para dispositivos tan diversos como teléfonos, *tablets*, relojes, televisores, entre otros. Basado en Linux, Android es un sistema operativo de código abierto que fue creado por la compañía Android Inc., que Google luego adquirió en 2005. Es un sistema operativo ampliamente utilizado, ya que provee una seguridad robusta. Debido a todo esto, la tienda de aplicaciones de Google ya cuenta con más de un millón de aplicaciones.

## 4.2. Lenguaje de programación para la aplicación

El lenguaje de programación seleccionado para realizar la aplicación es Kotlin, ya que este fue adoptado por Google para ser el lenguaje de preferencia para desarrollar aplicaciones para su sistema operativo de código abierto conocido como Android.

Kotlin no es un lenguaje nuevo, fue inicialmente desarrollado alrededor del 2010, en Rusia, en una isla de la cual proviene su nombre. Concebido por un equipo de la compañía JetBrains. Kotlin es un proyecto de código abierto, con la finalidad de hacer el desarrollo de aplicación que corran en el ambiente de máquina virtual de Java, o JVM por sus siglas en ingles. El jefe de proyecto, Andrey Breslav, menciona que la intención de Kotlin es ser un lenguaje compatible al 100 % con Java, más conciso, flexible y simple.

Según el jefe de proyecto de Kotlin debe de ser un lenguaje fácil de aprender, debido a su similitud con otros lenguajes para JVM conocido como *Groovy* y *Scala*. En el sitio oficial para desarrolladores de Android se describen algunas de las ventajas que Kotlin provee a los desarrolladores y al ecosistema de aplicaciones.

Kotlin promete reducir líneas de código, Por ser un lenguaje de programacion más expresivo permite que las líneas de código sean menos repetitivas y que el código sea mas facil de comprender, y por lo tanto de mantener. Kotlin también promete que las aplicaciones sean más estables, debido a la manera en que los apuntadores de memoria son manejados. Por último, menciona que Java y Kotlin son interoperables, lo que quiere decir, que tanto el código Java funciona dentro de Kotlin y el código de Kotlin puede ser utilizado dentro de las clases en Java.

Figura 10. **Clase en Kotlin**


Fuente: elaboración propia, empleando terminal.

En la figura 10 se muestra un ejemplo de una clase en Kotlin, primero se ve como en principio es similar a Java al definir la clase; pero luego en la definición de los parámetros se nota que el tipo de variable se define después del nombre de esta, separados por dos puntos. En la siguiente línea se ve la definición de una variable privada, casi en la misma forma que en Java, pero se

ve que el tipo de la variable es implícito. Luego se define una función (fun es una abreviatura del término función en ingles), y lo que cabe resaltar aquí es que el compilador transforma la sentencia *println* en su equivalente ejecutable en Java *System.out.println*, además, de incluir las librerías para utilizar esta función.

Está publicado en *GitHub*, el código fuente de la aplicación para Android. En las siguientes unidades se explicará el código, el patrón de programación, y la arquitectura de diseño de la aplicación.

## 4.3. Arquitectura de desarrollo de la aplicación

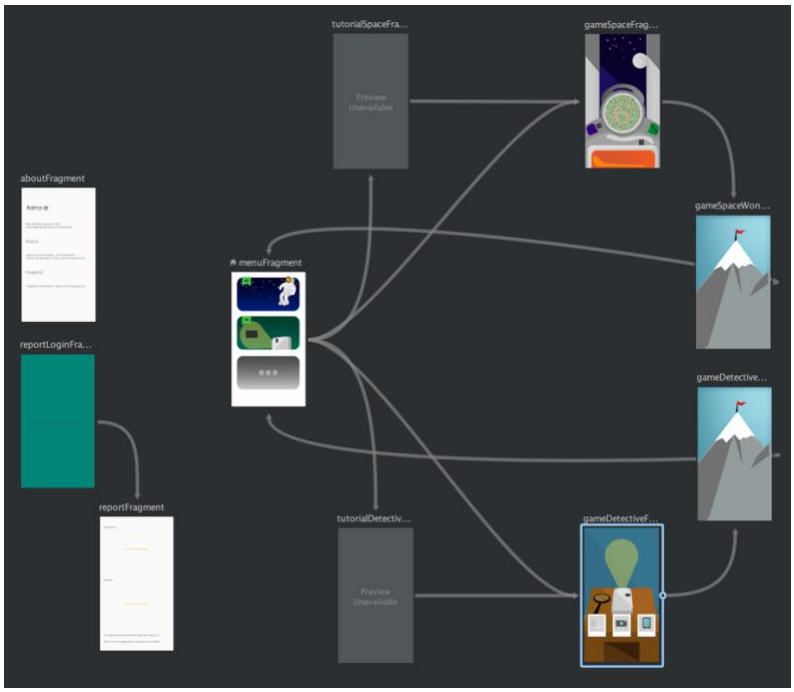
El desarrollo de esta aplicación se divide en dos partes: el despliegue gráfico y la lógica de la aplicación. Con las nuevas mejoras que proporciona el equipo de desarrollo de software para Android, esta división es claramente visible en el momento de crear la aplicación. A continuación, se describirá el patrón recomendado por Google para lograr la separación gráfica-lógica antes descrita.

### 4.3.1. Componente de navegación

Lograr una navegación fluida en las aplicaciones Android era una tarea difícil, los componentes gráficos (actividades) se encargaban de invocar nuevos destinos y estos a su vez los siguientes. Esta forma de crear el flujo de la aplicación complicaba ver todas las posibles rutas que un usuario podía seguir en la navegación de la aplicación. Por lo tanto, el desarrollador podía perder de vista algunos caminos que creaban confusión en el usuario.

Google en su versión del SDK 3.3, lanzada en enero 2019, agregó el componente de navegación. Este componente está diseñado para representar gráficamente los caminos que el usuario puede seguir en la aplicación. Se Puede apreciar en el gráfico de navegación de la aplicación Mi Aventura (véase figura 11).

Figura 11. **Gráfico de navegación de la aplicación**



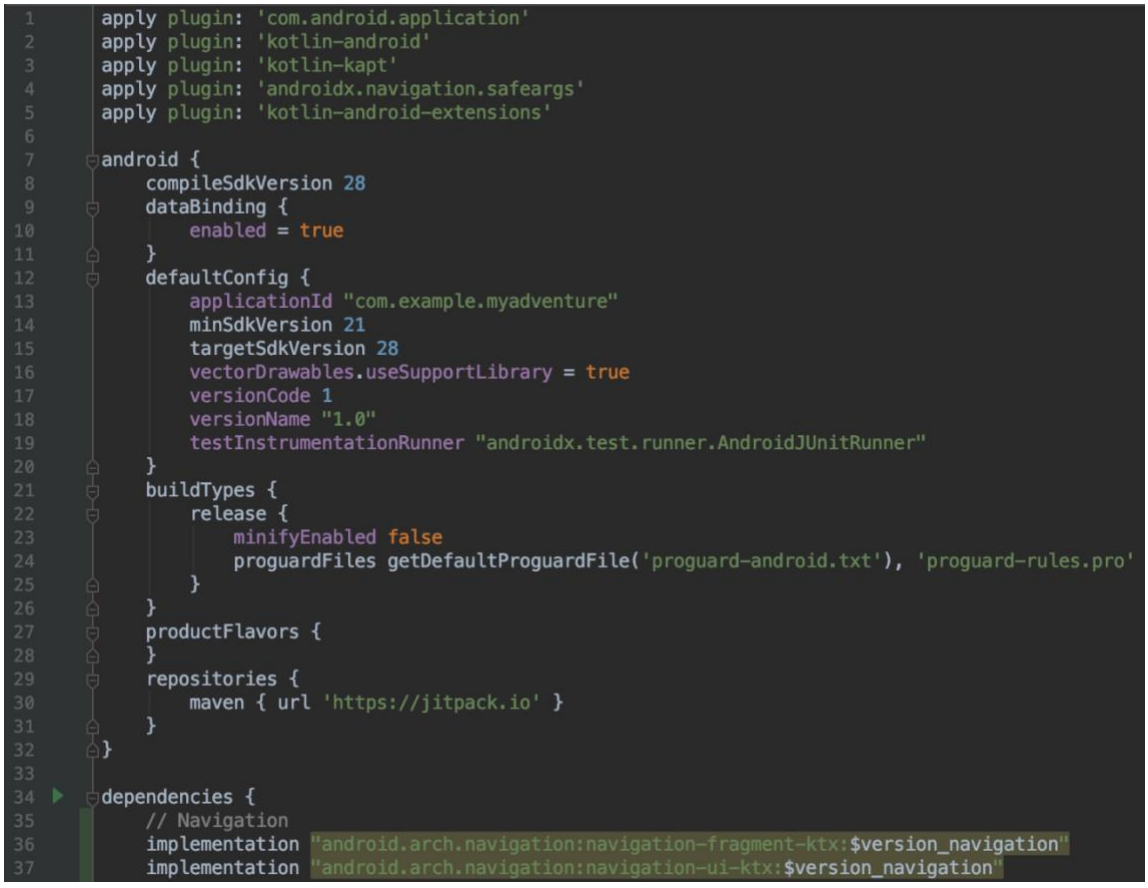

Fuente: elaboración propia, empleando Android Studio.

Se puede apreciar que el flujo que el usuario puede seguir está claramente definido (véase figura 11), nótese que las flechas de flujo solo tienen una dirección.

Cuando se inicia un nuevo proyecto en Android Studio se genera automáticamente algunos archivos que se tendrá que modificar para poder

utilizar el componente de navegación en nuestra aplicación. Los cuales se detallarán en la figura.

Figura 12. **Dependencias para utilizar el componente de navegación**



```
1   apply plugin: 'com.android.application'
2   apply plugin: 'kotlin-android'
3   apply plugin: 'kotlin-kapt'
4   apply plugin: 'androidx.navigation.safeargs'
5   apply plugin: 'kotlin-android-extensions'
6
7   android {
8       compileSdkVersion 28
9       dataBinding {
10          enabled = true
11      }
12      defaultConfig {
13          applicationId "com.example.myadventure"
14          minSdkVersion 21
15          targetSdkVersion 28
16          vectorDrawables.useSupportLibrary = true
17          versionCode 1
18          versionName "1.0"
19          testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
20      }
21      buildTypes {
22          release {
23              minifyEnabled false
24              proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
25          }
26      }
27      productFlavors {
28      }
29      repositories {
30          maven { url 'https://jitpack.io' }
31      }
32  }
33
34  dependencies {
35      // Navigation
36      implementation "android.arch.navigation:navigation-fragment-ktx:$version_navigation"
37      implementation "android.arch.navigation:navigation-ui-ktx:$version_navigation"
```

Fuente: elaboración propia, empleando Android Studio.

La figura 12 presenta las modificaciones que se necesitan en el archivo *build.gradle* (véase figura 12), como se puede ver en la línea 4, se agrega el adaptador para la navegación. Luego, en la parte de dependencias, empezando en la línea 14, se implementa la herramienta para navegación de fragmentos y navegación gráfica. Además de estos cambios, es necesario agregar una

actividad: *MainActivity* que se llama actividad principal, en la cual se agrega un componente que permitirá colocar los fragmentos que serán desplegados dentro de la actividad principal, según el flujo que la gráfica de navegación describe.

Figura 13. **Fragmento de navegación en la actividad principal**

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <layout xmlns:android="http://schemas.android.com/apk/res/android"
3          xmlns:app="http://schemas.android.com/apk/res-auto">
4
5          <androidx.drawerlayout.widget.DrawerLayout
6                  android:id="@+id/drawerLayout"
7                  android:layout_width="match_parent"
8                  android:layout_height="match_parent">
9
10             <LinearLayout
11                 android:layout_width="match_parent"
12                 android:layout_height="match_parent"
13                 android:orientation="vertical">
14                 <fragment
15                         android:id="@+id/myNavHostFragment"
16                         android:name="androidx.navigation.fragment.NavHostFragment"
17                         android:layout_width="match_parent"
18                         android:layout_height="match_parent"
19                         app:defaultNavHost="true"
20                         app:navGraph="@navigation/navigation" />
21             </LinearLayout>
22             <com.google.android.material.navigation.NavigationView
23                 android:id="@+id/navView"
24                 android:layout_width="wrap_content"
25                 android:layout_height="match_parent"
26                 android:layout_gravity="start"
27                 app:menu="@menu/navdrawer_menu" />
28         </androidx.drawerlayout.widget.DrawerLayout>
29 </layout>
```

Fuente: elaboración propia, empleando Android Studio.

Iniciando la línea 14 (véase figura 13), se puede ver que se agregó un fragmento al cual le asigna un identificador y un nombre como a cualquier elemento grafico, pero en la línea 20 (véase figura 13) se puede ver que se agrega una referencia a la gráfica de navegación. Esto es lo que le indicará a la actividad principal qué fragmento presentar y en qué orden se puede navegar

por todos los otros fragmentos. Ahora solo se tiene que crear un fragmento compatible con el componente de navegación, el cual será descrito a continuación.

La herramienta de desarrollo que proporciona Google para aplicaciones Android, Android Studio, permite agregar nuevos fragmentos a la aplicación; el único problema es que si se esta utilizando el componente de navegación, se tiene que realizar una pequeña modificación al código generado automáticamente, para que se pueda agregar a la gráfica. El único cambio que se debe realizar es cambiar el tipo de fragmento, el cual es *FragmentLayout* y debe ser sustituido por *layout*. Luego que se realiza este cambio y después de compilar la aplicación, ya se utilizá este fragmento dentro del gráfico de navegación.

Como algo adicional en la actividad principal, se agrega un componente que envuelve al fragmento, *DrawerLayout*, este elemento permitirá mostrar las opciones que se coloquen en el menú desplegable. Estas opciones serán visibles en un menú expandible que será accesible desde la esquina superior izquierda en la actividad principal. El único campo indispensable es la referencia al archivo de menú que se utilizará, como se ve en la línea 27 (véase figura 13).

Ahora, dentro de la clase que está asignada a la actividad principal (*MainActivity*) se debe agregar el código que asignará el componente de navegación, y nuestro menú despegable para que estos desplieguen los fragmentos y elementos requeridos.

Figura 14. **Código de inicialización de la actividad principal**

```
16  class MainActivity : AppCompatActivity() {
17      private lateinit var drawerLayout: DrawerLayout
18      private lateinit var appBarConfiguration : AppBarConfiguration
19      override fun onCreate(savedInstanceState: Bundle?) {
20          super.onCreate(savedInstanceState)
21          val binding : ActivityMainBinding! = DataBindingUtil.setContentView<ActivityMainBinding>( activity: this,
22                                                                                                    R.layout.activity_main)
23          drawerLayout = binding.drawerLayout
24          val navController : NavController = this.findNavController(R.id.myNavHostFragment)
25          NavigationUI.setupActionBarWithNavController( activity: this, navController, drawerLayout)
26          appBarConfiguration = AppBarConfiguration(navController.graph, drawerLayout)
27          // prevent nav gesture if not on start destination
28          navController.addOnDestinationChangedListener { nc: NavController, nd: NavDestination, _: Bundle? ->
29              if (nc.graph.startDestination == nd.id) {
30                  drawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLOCKED)
31                  supportActionBar?.show()
32              } else {
33                  drawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CLOSED)
34                  supportActionBar?.hide()
35              }
36          }
37          NavigationUI.setupWithNavController(binding.navView, navController)
38      }
39
40      override fun onSupportNavigateUp(): Boolean {
41          val navController : NavController = this.findNavController(R.id.myNavHostFragment)
42          return NavigationUI.navigateUp(navController, appBarConfiguration)
43      }
44  }
```

Fuente: elaboración propia, empleando Android Studio.

Se tienen que definir variables que contengan a al menú desplegable y la configuración de la barra de navegación. En la variable *binding* se obtienen los elementos de la actividad principal, que están listados (véase figura 13). Luego en la línea 24 (véase figura 14), se ve como se obtiene la referencia a la gráfica de navegación basándose en el ID del elemento *myNavHostFragment*, y se la pasa a la barra de navegación junto con la referencia al menú desplegable. Se agrega también un evento que verifica cada vez que se cambia el fragmento mostrado, esto con la finalidad de ocultar o mostrar la barra de navegación solo en el fragmento de inicio. Para terminar de enlazar la gráfica de navegación con la actividad se asigna el control de navegación con el elemento *navView* de la actividad principal.

El método que se define en la línea 40 (véase figura 14), es el que se encarga del botón conocido como atrás, reaccione de acuerdo con lo definido en la gráfica de navegación.

Figura 15. **Archivo de navegación**

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:app="http://schemas.android.com/apk/res-auto"
            xmlns:tools="http://schemas.android.com/tools" android:id="@+id/nav_root"
            app:startDestination="@id/menuFragment">

    <fragment android:id="@+id/menuFragment" android:name="com.example.myadventure.MenuFragment"
              android:label="@string/menu_fragment" tools:layout="@layout/fragment_menu">
        <action android:id="@+id/action_menuFragment_to_gameSpaceFragment" app:destination="@id/gameSpaceFragment"
                app:popUpTo="@+id/menuFragment" app:popUpToInclusive="false" app:enterAnim="@anim/slide_in_right"
                app:exitAnim="@anim/slide_out_left" app:popEnterAnim="@anim/slide_in_left"
                app:popExitAnim="@anim/slide_out_right"/>
        <action android:id="@+id/action_menuFragment_to_gameDetectiveFragment"
                app:destination="@id/gameDetectiveFragment" app:popUpTo="@+id/menuFragment"
                app:enterAnim="@anim/slide_in_right" app:exitAnim="@anim/slide_out_left"
                app:popEnterAnim="@anim/slide_in_left" app:popExitAnim="@anim/slide_out_right"/>
        <action android:id="@+id/action_menuFragment_to_tutorialSpaceFragment"
                app:destination="@id/tutorialSpaceFragment" app:popUpTo="@+id/menuFragment"
                app:enterAnim="@anim/slide_in_right" app:exitAnim="@anim/slide_out_left"
                app:popEnterAnim="@anim/slide_in_left" app:popExitAnim="@anim/slide_out_right"/>
        <action android:id="@+id/action_menuFragment_to_tutorialDetectiveFragment"
                app:destination="@id/tutorialDetectiveFragment" app:popUpTo="@+id/menuFragment"
                app:exitAnim="@anim/slide_out_left" app:enterAnim="@anim/slide_in_right"
                app:popEnterAnim="@anim/slide_in_left" app:popExitAnim="@anim/slide_out_right"/>
    </fragment>
```

Fuente: elaboración propia, empleando Android Studio.

El archivo de navegación puede editarse de forma gráfica y escrita. En la figura 15 muestra como se agrega un fragmento de navegación, en este caso el fragmento de menú del juego. Se debe agregar un ID que servirá para referenciarlo dentro del mismo archivo, y que *layout* se le asigna; esto es lo que la actividad mostrará cuando se le indique que debe navegar a este ID. Del fragmento de menú se puede ir a cuatro pantallas (véase figura 11), en ningún orden establecido:

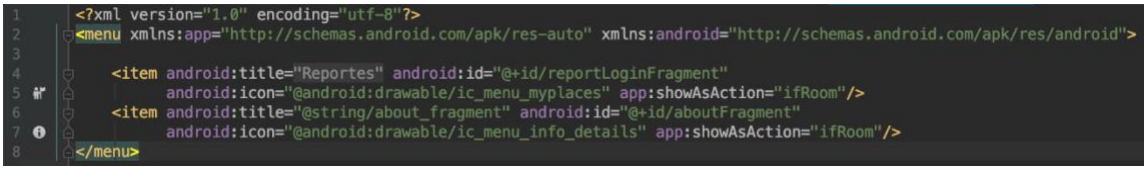
- Tutorial de juego para dislexia.
- Juego para dislexia.
- Tutorial de juego para daltonismo.

- Juego para daltonismo.

Lo único que se necesita en el fragmento de menú del juego para agregar estas transiciones son los bloques de acciones, en la cual se coloca un ID (generalmente la palabra acción seguida del fragmento origen y luego el fragmento destino) y el ID del fragmento destino.

Por último, para agregar la navegación desde el menú despeglable se tiene que agregar un archivo XML del tipo menú, en el cual se pueden agregar los elementos que se necesite para cada fragmento accesible desde este. En el caso de la aplicación Mi Aventura se tienen dos fragmentos, el fragmento de acceso a los reportes y el fragmento de Acerca de, los cuales se describirán en la siguiente figura.

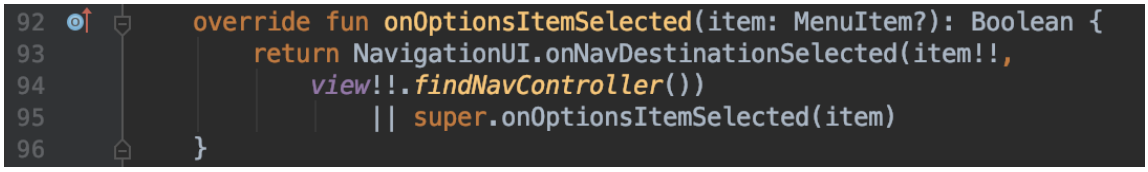
Figura 16. **Archivo de menú desplegable**




Fuente: elaboración propia, empleando Android Studio.

Lo único que se debe tener en cuenta es que el ID del elemento concuerde con el ID que se colocó en el archivo de navegación para el fragmento que se desea desplegar. Y la única lógica que se necesita para que esto sea presentado y accionado desde la actividad principal es el que se presenta a continuación.

Figura 17. **Código de navegación del menú despegable**




Fuente: elaboración propia, empleando Android Studio.

Este código debe ir dentro de lo clase que está asignada al fragmento marcado como punto de inicio de la aplicación, en el archivo de navegación, Para esta aplicación es el fragmento de menú del juego (véase figura 15). Y lo único que hace este código es invocar un elemento de la gráfica de navegación con base en el ID del elemento que se definió en el archivo XML de menú despegable.

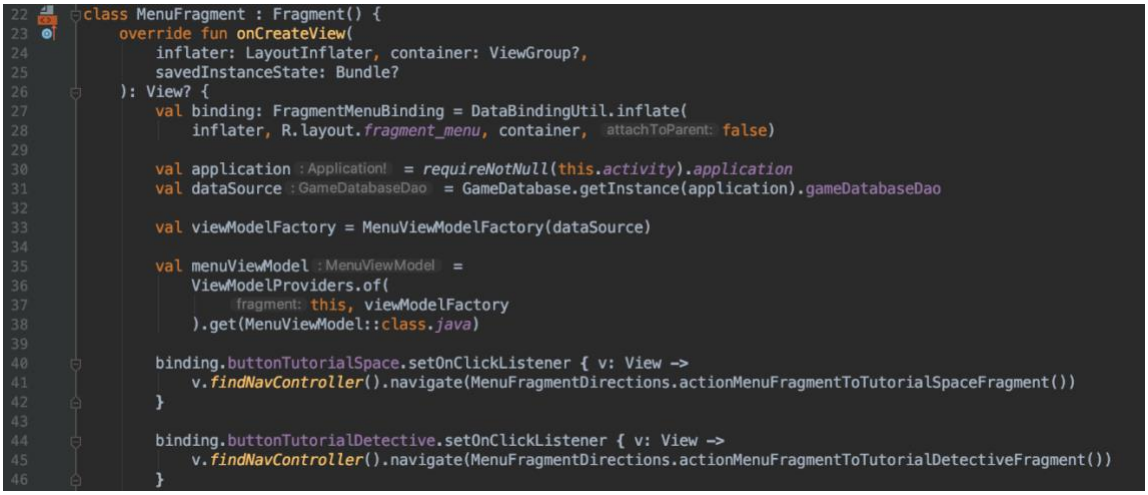
El tipo de navegación descrito en esta sección es meramente reactivo, lo que quiere decir, que no tiene ningún proceso lógico, nos lleva de un punto A hacia un punto B por simples indicadores. En la siguiente unidad se tratará el tema de transiciones entre fragmentos que requieren lógica, pero cabe resaltar que estas transiciones lógicas utilizan el diagrama de navegación para funcionar; debido a que la transición del punto A hacia el punto B es lo que se quiere lograr, pero agregando una condición, o serie de condiciones. Un ejemplo seria decidir ir al Tutorial del juego para dislexia en lugar del Juego para la dislexia si es la primera vez que se quiere iniciar el juego.

### 4.3.2. Patrón Vista-Modelo

El equipo de desarrollo de Google proporciona un componente que permite abstraer la lógica de la interfaz gráfica de la aplicación y moverla a una

clase que se encargará que la lógica sobreviva a todas las etapas del ciclo de vida de una actividad o un fragmento (Componentes gráficos de las aplicaciones para Android). El componente *ViewModel*, es una clase que esta diseñada para realizar el manejo de la información, principalmente manipulación y almacenamiento de esta. Esto debido a que el sistema Android puede crear y destruir los elementos gráficos de la aplicación basado en las acciones realizadas por el usuario; provocando que información pueda ser perdida antes de llegar a la siguiente etapa.

Figura 18. **Asignación de acciones a elementos en el fragmento**



```
22  class MenuFragment : Fragment() {
23      override fun onCreateView(
24          inflater: LayoutInflater, container: ViewGroup?,
25          savedInstanceState: Bundle?
26      ): View? {
27          val binding: FragmentMenuBinding = DataBindingUtil.inflate(
28              inflater, R.layout.fragment_menu, container, attachToParent: false)
29
30          val application : Application! = requireNotNull(this.activity).application
31          val dataSource : GameDatabaseDao = GameDatabase.getInstance(application).gameDatabaseDao
32
33          val viewModelFactory = MenuViewModelFactory(dataSource)
34
35          val menuViewModel : MenuViewModel =
36              ViewModelProviders.of(
37                  fragment: this, viewModelFactory
38              ).get(MenuViewModel::class.java)
39
40          binding.buttonTutorialSpace.setOnClickListener { v: View ->
41              v.findNavController().navigate(MenuFragmentDirections.actionMenuFragmentToTutorialSpaceFragment())
42          }
43
44          binding.buttonTutorialDetective.setOnClickListener { v: View ->
45              v.findNavController().navigate(MenuFragmentDirections.actionMenuFragmentToTutorialDetectiveFragment())
46          }
```

Fuente: elaboración propia, empleando Android Studio.

En el fragmento de menú se agregan las referencias a los elementos que aparecerán en pantalla. Se incluye también una referencia a la base de datos. Utilizando un método de fábrica se instancia un objeto *MenuViewModel*, aquí es donde reside la lógica de la pantalla del menú del juego. A los botones que dirigen a los tutoriales de juego se les puede agregar las acciones de manera directa; como se ve en la línea 41 y 45 (véase figura 18).

Para agregar la navegación desde el *MenuViewModel*, se utilizará una nueva funcionalidad que la herramienta de desarrollo ofrece conocida como variables vivas. Las variables vivas son objetos que se crean y manejan desde los *ViewModels* pero que pueden ser observadas desde el fragmento, lo que permite generar acciones como navegar a otras pantallas, cambiar imágenes basados en la lógica interna.

Desde la lógica en el *MenuViewModel* se definen cuatro variables, una para cada fragmento al que se puede llegar desde el fragmento de menú. Estas variables servirán como banderas para navegar basado en si la base de datos está vacía o no. Como se describió anteriormente se asignará un observador por cada variable dentro del fragmento para así decidir a qué pantalla navegar.

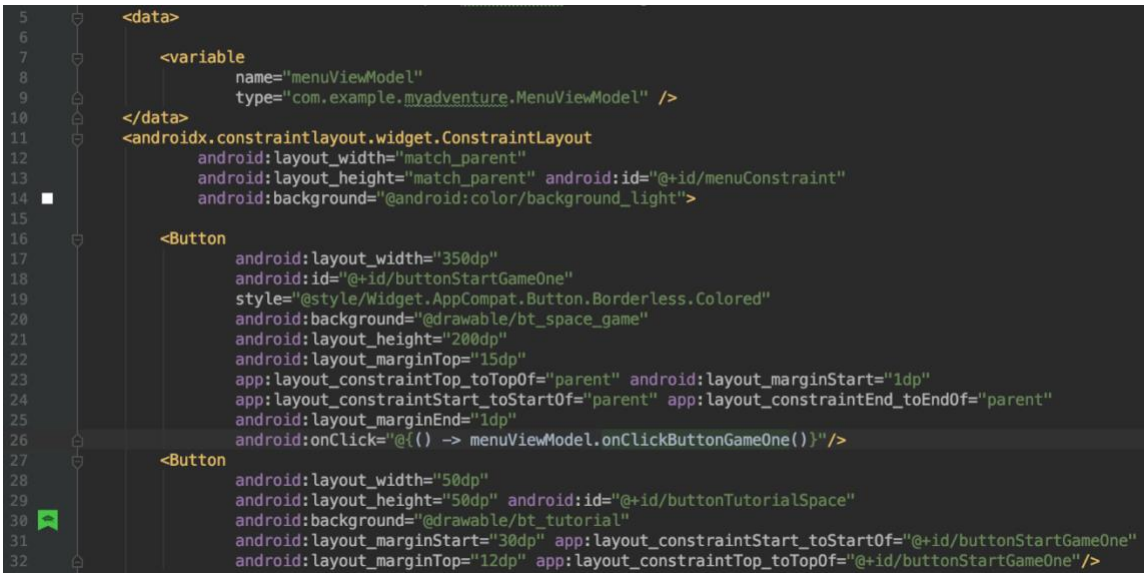
Figura 19. **Variables vivas**

```
class MenuViewModel(
    val database: GameDatabaseDao) : ViewModel() {


    private var viewModelJob :Job = Job()

    private val uiScope :CoroutineScope = CoroutineScope( context: Dispatchers.Main + viewModelJob)

    private val _navigateToGameOne = MutableLiveData<Int>()

    val navigateToGameOne: LiveData<Int>
        get() = _navigateToGameOne

    private val _navigateToGameTwo = MutableLiveData<Int>()

    val navigateToGameTwo: LiveData<Int>
        get() = _navigateToGameTwo

    private val _navigateToGameTutorialOne = MutableLiveData<Int>()

    val navigateToGameTutorialOne: LiveData<Int>
        get() = _navigateToGameTutorialOne

    private val _navigateToGameTutorialTwo = MutableLiveData<Int>()

    val navigateToGameTutorialTwo: LiveData<Int>
        get() = _navigateToGameTutorialTwo
```

Fuente: elaboración propia, empleando Android Studio.

Para agregar un variable viva solo se necesita agregar dos referencias, una que será para acceso interno y una referencia en la que se define un *getter*; para que la variable sea accesible desde el fragmento. En la figura 19 se puede ver que se utiliza el tipo *LiveData* y se crea con una variable entera. Después se ve como se puede agregar la referencia al *MenuViewModel* y así utilizar las funciones que se definen dentro de la clase (véase figura 20).

*Figura 20.* **Asociación de fragmento con el *ViewModel***




Fuente: elaboración propia, empleando Android Studio.

Se ve en la línea 5 (véase figura 20) como se agrega una referencia a *data*, dentro se asigna la clase *MenuViewModel*. Ahora con esta referencia se puede invocar valores y funciones dentro del fragmento con los cuales se crea la interacción entre la parte gráfica de la aplicación y la parte lógica. También se puede ver en la línea 26 (véase figura 20) que se puede llamar funciones desde los eventos de los elementos gráficos. Dentro de la función

*onClickButtonGameOne* se tiene la lógica para decidir pasar al tutorial o al juego, como se ve en la figura 21.

Figura 21. **Funciones lógicas de navegación**

```
38      fun onClickButtonGameOne(){
39          uiScope.launch { this: CoroutineScope
40              if(getSpaceScores().isEmpty()){
41                  _navigateToGameTutorialOne.value = 1
42              } else {
43                  _navigateToGameOne.value = 1
44              }
45          }
46      }
47
48      fun onClickButtonGameTwo(){
49          uiScope.launch { this: CoroutineScope
50              if(getDetectiveScores().isEmpty()){
51                  _navigateToGameTutorialTwo.value = 1
52              } else {
53                  _navigateToGameTwo.value = 1
54              }
55          }
56      }
57
58      private suspend fun getSpaceScores(): IntArray {
59          return withContext(Dispatchers.IO) { this: CoroutineScope
60              var latestScores : IntArray = database.getSpaceLatestScores()
61              latestScores ^withContext
62          }
63      }
64
65      private suspend fun getDetectiveScores(): IntArray {
66          return withContext(Dispatchers.IO) { this: CoroutineScope
67              var latestScores : IntArray = database.getDetectiveLatestScores()
68              latestScores ^withContext
69          }
70      }
```

Fuente: elaboración propia, empleando Android Studio.

Para acceder a la base de datos en una acción que proviene de la interface gráfica, se debe hacer uso de corutinas; corutinas que se encargan de ejecutar las instrucciones sin bloquear la funcionalidad de la interface gráfica. En la línea 39 (véase figura 21) se ve como se lanza una corutina, dentro de la

cual se llama a un método que accede a la base de datos. Aquí se utiliza otra corutina pero esta vez el método utilizado es *withContext* debido a que se regresa un valor. En grandes rasgos lo que se hace en el código de la figura 21 es asignar un valor a la variable para navegar al tutorial si no hay ninguna puntuación guardada en la base de datos, lo que quiere decir que no se a completado ningun juego, y si ya existe algun puntaje almacenado en la base de datos entonces se dirige al usuario al juego directamente.

Se puede ver como la interface gráfica interactua con la variable viva (véase figura 21), pero aún se tiene que agregar código para que el fragmento puede reaccionar a los cambios de estas variables, lo cual se muestra en la figura 22.

Figura 22. **Observadores para variables vivas**

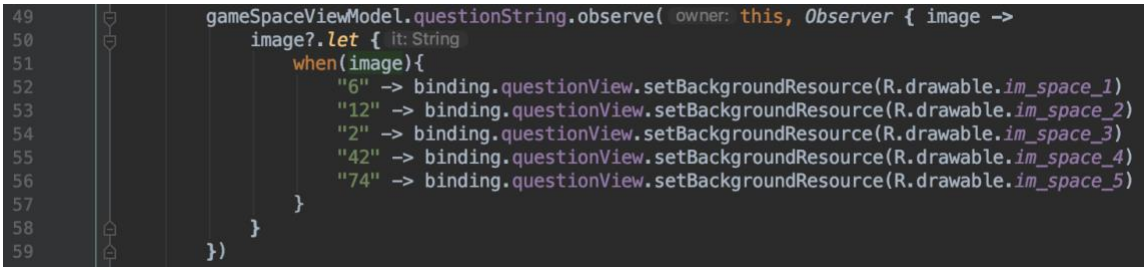
```
48      menuViewModel.navigateToGameOne.observe( owner: this, Observer { game ->
49          game?.let { it: Int
50              this.findNavController().navigate(
51                  MenuFragmentDirections.actionMenuFragmentToGameSpaceFragment()
52              )
53              menuViewModel.doneNavigating()
54          }
55      })
56
57      menuViewModel.navigateToGameTwo.observe( owner: this, Observer { game ->
58          game?.let { it: Int
59              this.findNavController().navigate(
60                  MenuFragmentDirections.actionMenuFragmentToGameDetectiveFragment()
61              )
62              menuViewModel.doneNavigating()
63          }
64      })
65
66      menuViewModel.navigateToGameTutorialOne.observe( owner: this, Observer { game ->
67          game?.let { it: Int
68              this.findNavController().navigate(
69                  MenuFragmentDirections.actionMenuFragmentToTutorialSpaceFragment()
70              )
71              menuViewModel.doneNavigating()
72          }
73      })
74
75      menuViewModel.navigateToGameTutorialTwo.observe( owner: this, Observer { game ->
76          game?.let { it: Int
77              this.findNavController().navigate(
78                  MenuFragmentDirections.actionMenuFragmentToTutorialDetectiveFragment()
79              )
80              menuViewModel.doneNavigating()
81          }
82      })
```

Fuente: elaboración propia, empleando Android Studio.

Dentro de la clase de *MenuFragment* utilizando la variable con la referencia al *MenuViewModel* se accede a la referencia pública de la variable viva, y a su instancia de observador se le agrega la lógica necesaria para navegar al fragmento requerido. Como se ve en la línea 48 (véase figura 22), en este lambda se tiene una variable *game*; la cual contendrá el valor que le asigna a la variable viva. El observador solo se ejecuta cuando hay un cambio en el valor de la variable. Al finalizar se llama a un función en el *MenuViewModel* que se encarga de limpiar o asignar nulo a todas las variables de navegación, esto dado que el fragmento será remplazado para pasar al siguiente.

Otro uso que se le puede dar a las variables vivas es el de asignar imágenes diferentes a los objetos del fragmento, se ve como se puede asignar nuevas imágenes basado en la lógica del juego (véase figura 23). Debido a que las imágenes presentadas en la aplicación deben de seguir la lógica del juego.

Figura 23. **Actualización dinámica de imágenes**




Fuente: elaboración propia, empleando Android Studio.

Ya que las variables vivas pueden contener cualquier tipo de dato, entonces en base a su valor se asignará al elemento que contiene la imagen del juego una nueva imagen cada vez que la lógica lo indique.

En la siguiente unidad se explicara cómo se almacena la informacion de de la base de datos y cómo se crean las funciones que extraen la complegidad de este proceso, el cual se lleva acabo en una clase que oculta la lógica.

### 4.3.3. Base de datos

Según la página oficial de desarrolladores de Android, la librería recomendada para manejar base de datos es *Room*, ya que esta provee una capa de abstracción sobre la base de datos que corre en el sistema operativo Android (SQLite). Esta librería proporciona más beneficios que solo abstraer la lógica de la base de datos, sino que además permite crear una copia de los datos en memoria caché, lo que hace que el acceso a los datos sea mucho más rápido.

De igual manera que con otras librerías, se debe agregar la referencia al proyecto para hacer uso de los beneficios que *Room* provee. Por lo que, se agrega dicha referencia en el archivo *build.gradle* como se muestra en la figura 24. También otras librerías que servirán para manejar las corutinas, con las cuales se permitiá que se acceda a la base de datos sin bloquear la interface gráfica.

*Figura 24.* **Dependencias de *Room***

```
58    // Room and Lifecycle dependencies
59    implementation "androidx.room:room-runtime:$version_room"
60    implementation 'androidx.legacy:legacy-support-v4:1.0.0-beta01'
61    implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0-beta01'
62    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.0.0'
63    implementation 'androidx.appcompat:appcompat:1.0.0-beta01'
64    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
65    kapt "androidx.room:room-compiler:$version_room"
66    implementation "androidx.lifecycle:lifecycle-extensions:$version_lifecycle_extensions"
67    // Coroutines
68    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:$version_coroutine"
69    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:$version_coroutine"
```

Fuente: elaboración propia, empleando Android Studio.

La primera clase que se debe crear es en la que define cómo se creará la instancia de la base de datos, la cual se muestra en la figura 25. Como se

puede ver en la línea 9 se asigna la clase de tipo *RoomDatabase*, se agrega una referencia en la línea 14 al objeto de acceso a la data, DAO por sus siglas en inglés; y luego se inicia el método con el cual se obtiene la instancia única de la base de datos. En el método *getInstance* se utiliza el método *synchronized* que permite bloquear la base de datos para que solo una referencia pueda ser leída o escrita a la vez, esto para evitar colisiones de datos. Además, se crea un Singleton que permite crear la referencia solo una vez, y si se solicita una nueva se devuelve la referencia al primer objeto creado.

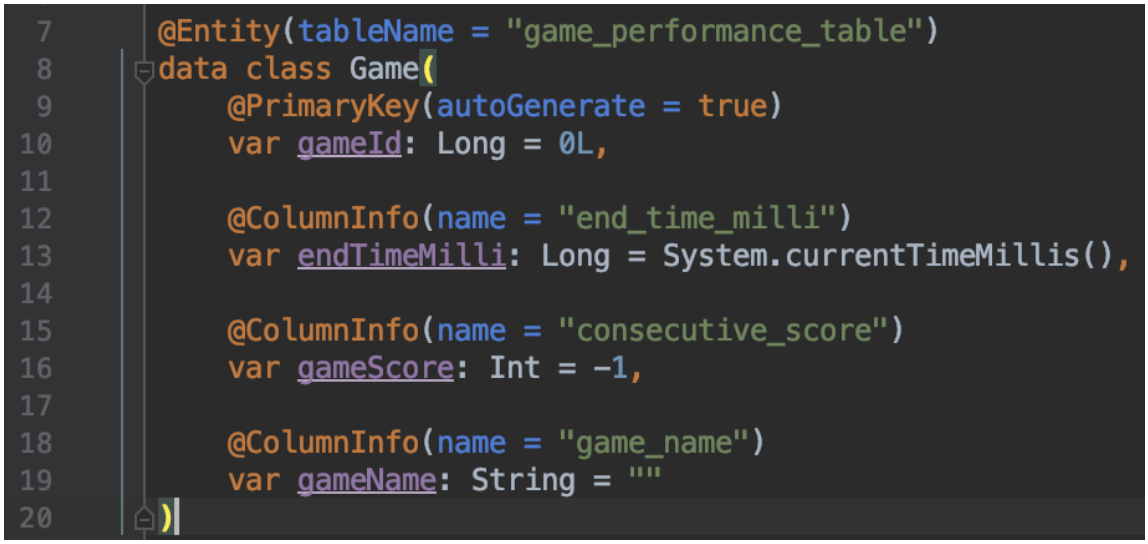
Figura 25. **Clase para instanciación de la base de datos**

```
@Database(entities = [Game::class, Pin::class], version = 3, exportSchema = false)
abstract class GameDatabase : RoomDatabase() {

    /**
     * Connects the database to the DAO.
     */
    abstract val gameDatabaseDao: GameDatabaseDao

    companion object {
        @Volatile
        private var INSTANCE: GameDatabase? = null

        fun getInstance(context: Context): GameDatabase {
            // Multiple threads can ask for the database at the same time, ensure we only initialize
            // it once by using synchronized. Only one thread may enter a synchronized block at a
            // time.
            synchronized( lock: this) {
                // Copy the current value of INSTANCE to a local variable so Kotlin can smart cast.
                // Smart cast is only available to local variables.
                var instance :GameDatabase?  = INSTANCE
                // If instance is `null` make a new database instance.
                if (instance == null) {
                    instance = Room.databaseBuilder(
                        context.applicationContext,
                        GameDatabase::class.java,
                         name: "game_history_database"
                    )
                        // Wipes and rebuilds instead of migrating if no Migration object.
                        // Migration is not part of this lesson. You can learn more about
                        // migration with Room in this blog post:
                        // https://medium.com/androiddevelopers/understanding-migrations-with-room-f01e04b07929
                        .fallbackToDestructiveMigration()
                        .build()
                    // Assign INSTANCE to the newly created database.
                    INSTANCE = instance
                }
                // Return instance; smart cast to be non-null.
                return instance
            }
        }
    }
}
```

Fuente: elaboración propia, empleando Android Studio.

Para crear las tablas en las cuales se guardará la información se deben de definir clases, que contengan las columnas, estas se definen como atributos. En la figura 26 se ve como se define también el nombre de la tabla y los tipos de datos de cada columna.

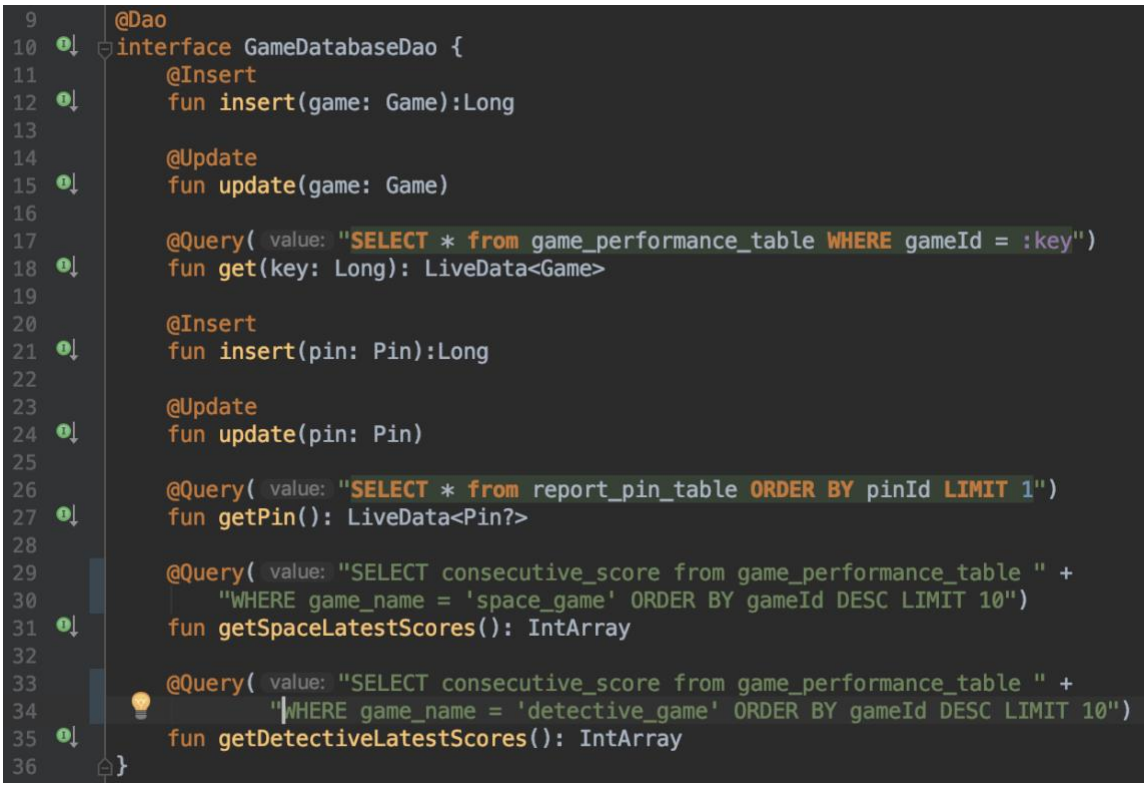
Figura 26. **Tabla de datos de juego**




Fuente: elaboración propia, empleando Android Studio.

Por último, para acceder a los datos se debe crear la interfaz que define el DAO, esta clase utiliza lenguaje SQL para acceder a los datos; pero devuelve los datos y es utilizada como si fuera un método o función. En la figura 27 se muestra como se insertan y leen datos de la base de datos. Lo primero que se debe indicar es el tipo DAO, también se puede ver que el tipo del archivo es *interface*. Para cada función se le debe asociar su etiqueta, como se ve en la línea 11 y 14 se utiliza las etiquetas *Insert* y *Update*, las cuales se encargarán de crear el comando SQL necesario para insertar y actualizar la información respectivamente. Para poder acceder a los datos se crean los *queries*, como se muestra en la línea 17.

Figura 27. **Clase para acceso de los datos**



```
@Dao
interface GameDatabaseDao {
    @Insert
    fun insert(game: Game):Long

    @Update
    fun update(game: Game)

    @Query( value: "SELECT * from game_performance_table WHERE gameId = :key")
    fun get(key: Long): LiveData<Game>

    @Insert
    fun insert(pin: Pin):Long

    @Update
    fun update(pin: Pin)

    @Query( value: "SELECT * from report_pin_table ORDER BY pinId LIMIT 1")
    fun getPin(): LiveData<Pin?>

    @Query( value: "SELECT consecutive_score from game_performance_table " +
        "WHERE game_name = 'space_game' ORDER BY gameId DESC LIMIT 10")
    fun getSpaceLatestScores(): IntArray

    @Query( value: "SELECT consecutive_score from game_performance_table " +
            "WHERE game_name = 'detective_game' ORDER BY gameId DESC LIMIT 10")
    fun getDetectiveLatestScores(): IntArray
}
```

Fuente: elaboración propia, empleando Android Studio.

Para hacer uso de las funciones que el DAO provee en nuestro código se debe utilizar corutinas para que los accesos a la base de datos no interfieran con las acciones que el usuario está realizando en la aplicación. En la figura 28 se ve como crear los métodos que relacionen las acciones en la interfaz gráfica con la base de datos, por medio de las clases *ViewModels*.

Figura 28. **Utilización del DAO**

‘


Fuente: elaboración propia, empleando Android Studio.

Como se ve en la figura 28, al terminar el juego se quiere guardar el momento en el tiempo en el que se terminó el juego y la puntuación. Estos datos se almacenan en una instancia de tipo *Game* (véase línea 89), que es la clase que define la tabla para el DAO. Cuando se esta listos para terminar la partida, se llama al método *onEndGame* y aquí se utiliza la corutina generada por el uiScope, en donde se llama al método insertar (véase línea 93) pasando como parámetro el objeto *Game* a guardar. Como se quiere regresar el ID único que identifica este objeto en la base de datos se utiliza en la función *insert* el método *withContext*, el cual permite utilizar la corutina y luego devolver un valor; que como se ve es de tipo *Long.*

De igual manera se debe proceder al leer la información de la base de datos, la cual es luego asignada a una variable viva, a un objeto *TextView* o gráfica de barras para poder presentar la información al usuario.

# CONCLUSIONES

1. Con la tecnología de Android, se pudo crear una aplicación que presenta las cartas de Ishihara y una prueba para la dislexia de forma objetiva, ayudando a los padres para tomar la decisión de llevar a su hijo a una evaluación médica.

2. Utilizando los medios proporcionados por Google se crea un canal de distribución de alto alcance para promocionar la herramienta.

3. Al realizar la aplicación utilizando las más recientes librerías proporcionadas por Google se asegura que la aplicación podrá ser utilizada en los dispositivos mas recientes y que la aplicación podrá ser mantenida por mucho tiempo.

# RECOMENDACIONES

1. Se puede aprovechar el mercado de las aplicaciones para iOS para distribuir una aplicación similar.

2. Investigar un método de calificación más tecnificado que minimice la subjetividad de este tipo de pruebas aplicadas a los niños.

3. Fomentar la creación de aplicaciones de este tipo con fines educativos, o para difundir mensajes a la población en general.

# BIBLIOGRAFÍA


1. AGARWAL, Ritu y PRASAD, Jayesh. *The role of innovation characteristics and perceived voluntariness in the acceptance of information.* EE. UU: Decision Sciences, 1997. 26 p.

2. BRESLAV, Andrey. *Language of the Month: Kotlin. Dr. Dobb's*. [en línea]. <http://www.drdobbs.com/jvm/language-of-the-month-kotlin/232600836?pgno=1>. [Consulta: 20 de enero del 2019].

3. BRIGGS, Robert O.; REINIG, Bruce A. y DE VREEDE, Gert-Jan. *The Yield Shift Theory of Satisfaction and Its Application to the IS/IT Domain*. EE. UU: Springer, 2011. 26 p.

4. Clínica Baviera. *¿Qué tipos de daltonismo existen?.* [en línea]. <http://www.clinicabaviera.com/blog/salud-visual/tipos-de-daltonismo/>. [Consulta: 15 de junio de 2019].

5. Equipo de Expertos de la Universidad Internacional de Valencia. *Dificultades de aprendizaje: dislexia, dislalia y otros problemas*. [en línea]. <https://www.universidadviu.com/dificultades-de-aprendizaje-dislexia-dislalia-y-otros-problemas/> [Consulta: 15 de junio de 2019].

6. Equipo editorial Google Developers. *ViewModel Overview*. [en línea]. <https://developer.android.com/topic/libraries/architecture/viewmodel>. [Consulta: 5 de junio 2019].

7. EVERETT, Rogers. *Diffusion of Innovation*. 4ª ed. EE. UU: Free Press, 2010. 518 p.

8. HEISS, Janice. *The Advent of Kotlin: A Conversation with JetBrains' Andrey Breslav*. [en línea]. <https://www.oracle.com/technetwork/articles/java/breslav-1932170.html>. [Consulta: 17 de abril de 2019].

9. HUNT, David M. *The Chemistry of John Dalton's Color Blindness*. EE.UU.: American Association for the Advancement of Science, 1995. 4 p.

10. INDRARATHNE, Bimali. *Accommodating learners with dyslexia in ELT in Sri Lanka: teachers' knowledge, attitudes and challenges*. [en línea]. <http://eprints.whiterose.ac.uk/143707/1/Author_accepted_manuscript.pdf>. [Consulta: 15 de marzo de 2019].

11. ISHIHARA S. *Tests for colour-blindness*. Japan: Kanehara & Co. LTD, 1985. 24 p.

12. MERINO, María y PÉREZ, Julián. *Definición de Android*. [en línea]. <https://definicion.de/android/>. [Consulta: 23 de febrero del 2019].

13. OLTRA, Vicente. *Dislexia: Información, Diagnóstico y Tratamiento de la Dislexia*. [en línea]. <http://www.psicopedagogia.com/dislexia>. [Consulta: 23 de febrero del 2016].

14. Pediatrics. *Learning Disabilities, Dyslexia, and Vision*. Vol. 124. EE. UU: Pediatrics, 2009, 7 p.

15. QUIROS, Annghy. *Ansiedad en niños que cursan por primera vez primero primaria, comprendidos entre 6-8 años en AMG Verbena zona 7*. Trabajo de graduación de Lic. Psicología. Escuela de Ciencias Psicológicas, Universidad de San Carlos de Guatemala, 2018. 60 p.

16. SHAFIROV, Maxim. *Kotlin on Android. Now official*. [en línea]. <https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/>. [Consulta: 17 de mayo de 2019].

17. VON REBEUR, Ana. *La ciencia del color*. Argentina: Siglo XXI, 2014. 126 p.

# APÉNDICES

Apéndice 1. **Repositorio del proyecto**
<https://github.com/AllanMNoguera/MyAdventure>.

Fuente: elaboración propia, empleando Android Studio