

UNIVERSIDAD DE COSTA RICA
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE0499 – Proyecto Eléctrico

**Aplicación en Android para el aprendizaje de la serie I de
Ashtanga Yoga**

por

Luis Diego Pacheco Chamberlain

Ciudad Universitaria Rodrigo Facio

Octubre de 2021

Aplicación en Android para el aprendizaje de la serie I de Ashtanga Yoga

por

Luis Diego Pacheco Chamberlain

B65216

IE0499 – Proyecto Eléctrico

Aprobado por

M.Sc. Ing. Marco Villalta Fallas, MSc

Profesor guía

Ing. Teodoro Willink Castro, MSc

Profesor lector

Ing. Jorge Arturo Romero Chacón, Ph.D

Profesor lector

Octubre de 2021

Resumen

Aplicación en Android para el aprendizaje de la serie I de Ashtanga Yoga

por

Luis Diego Pacheco Chamberlain

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Profesor guía: M.Sc. Ing. Marco Villalta Fallas, MSc

Octubre de 2021

El Ashtanga Yoga es un estilo de yoga en el que se realizan distintas posturas en un orden dado. Este tipo de Yoga está compuesto por múltiples secuencias de posturas llamadas series. La Serie 1 está compuesta por más de 100 posturas. Cada una de estas posturas, además tiene un nombre en sánscrito, un enfoque de la vista específico y otros detalles técnicos. Aprender el orden en el que se realizan las posturas más sus detalles le agregan complejidad a una práctica que ya es físicamente compleja. Por lo tanto, se busca implementar una aplicación para celular que ayude a los practicantes de esta disciplina con la memorización y práctica de esta primera serie. En este proyecto se desarrolló la aplicación para el sistema operativo Android en el lenguaje de programación Kotlin. Se diseñaron dinámicas distintas que consisten en juegos mnemónicos para memorizar: el nombre de las posturas, el orden de la secuencia y el enfoque visual que se realiza en cada postura. Se buscó que las imágenes utilizadas en el proyecto son de uso libre, por lo que se podría distribuir a personas que lo quieran utilizar o modificar. Se logró implementar 5 dinámicas distintas para ayudar en la práctica y memorización de la serie I de Ashtanga Yoga. La primera dinámica ayuda a relacionar cada postura de la serie con un aspecto técnico de esta como su nombre y su enfoque. El segundo modo ayuda a aprender sobre el orden de la secuencia, aquí el usuario puede relacionar cada postura, y su nombre, con la siguiente postura. El tercer modo combina los primeros dos modos lo que ayuda a tener un aprendizaje más completo de toda la secuencia. El cuarto modo consiste en un modo de acompañamiento para alguien practicando la secuencia y le enseña al usuario cada postura con su información. Aquí las posturas están ordenadas y se puede avanzar por cada una conforme se va avanzando en la secuencia. Finalmente, en el quinto modo, la información recopilada en la aplicación se resume en fichas para que el usuario pueda repasar sobre la serie.

Palabras claves: *Android, aprendizaje, Ashtanga Yoga, Kotlin.*

Acerca de IE0499 – Proyecto Eléctrico

El Proyecto Eléctrico es un curso semestral bajo la modalidad de trabajo individual supervisado, con el propósito de aplicar estrategias de diseño y análisis a un problema de temática abierta de la ingeniería eléctrica. Es un requisito de graduación para el grado de Bachiller en Ingeniería Eléctrica de la Universidad de Costa Rica.

Abstract

Aplicación en Android para el aprendizaje de la serie I de Ashtanga Yoga

Original in Spanish. Translated as: “Android application for learning Ashtanga Yoga series I”

by

Luis Diego Pacheco Chamberlain

University of Costa Rica

Department of Electrical Engineering

Tutor: M.Sc. Ing. Marco Villalta Fallas, MSc

October of 2021

Ashtanga Yoga is a style of yoga in which different postures are performed in a given order. This type of Yoga is composed of multiple sequences of postures, or series. Each of these postures has a Sanskrit name and a specific eye focus, among other technical details. Learning the order in which the postures are performed and their details, adds complexity to an already physically demanding practice. Therefore, the need to implement a mobile application that helps practitioners of this discipline with memorization and practice of Series I. The app was developed for the Android operating system in the Kotlin programming language. Different dynamics were designed in the shape of mnemonic games to memorize details from the postures. In an attempt to allow easy distribution and third-party modification of the program, images obtained in the program were obtained from free to use sources. The project was successful in implementing 5 different dynamics to help in the practice and memorization of the series I of Ashtanga Yoga. The first dynamic helps relating each position in the series with a technical aspect of it. The second mode helps learning about the order of the sequence. The third mode combines the first two modes which helps to have a more complete learning experience. The fourth mode consists of helper mode for someone practicing the sequence and shows the user each pose, in order, with all the information. Finally, in the fifth mode, the information collected in the application is summarized in flashcards so that the user can review the series.

Keywords: *Android, learning, Ashtanga Yoga, Kotlin.*

About IE0499 – Proyecto Eléctrico (“Electrical Project”)

The “Electrical Project” (or “capstone project”) is a course of supervised individual work of one semester, with the purpose of applying design and analysis strategies to a problem in an open topic in electrical engineering. It is a requisite of graduation for the Bachelor of Science in Electrical Engineering, granted by the University of Costa Rica.

Dedicado a mi familia y amigos.

Agradecimientos

Agradezco el apoyo de mi familia y amigos. También las enseñanzas de los profesores durante mi carrera universitaria.

Índice general

Índice general	xii
Índice de figuras	xii
Índice de tablas	xv
Nomenclatura	xvii
1 Introducción	1
1.1. Descripción General	1
1.2. Alcances del proyecto	2
1.3. Justificación del proyecto	2
1.4. Objetivos	2
1.4.1. Objetivo general	2
1.4.2. Objetivos específicos	2
1.5. Metodología	3
1.6. Cronograma de actividades	4
2 Marco Teórico	5
2.1. Ashtanga Yoga	5
2.1.1. Definición e historia	5
2.1.2. Vinyasa	6
2.1.3. Asana	6
2.1.4. Ujjai	7
2.1.5. Bandha	7
2.1.6. Dristi	8
2.1.7. Surya Namaskara	8
2.1.8. Serie 1 – Yoga Chikitsa	9
2.2. Desarrollo en Android	12
2.2.1. Sistema operativo – Android	12
2.2.2. Entorno de desarrollo – Android Studio	12
2.2.3. Lenguaje – Kotlin	15

2.2.4. Diseño – Elementos de la interfaz	16
2.2.5. Estilo – Material Design	18
3 Desarrollo	21
3.1. Diseño y desarrollo de la aplicación	21
3.1.1. Diseño e ideas para la aplicación	21
3.1.2. Datos e información en la aplicación	24
3.1.3. Menús	26
3.1.4. Diseño de dinámicas	30
3.1.5. Main view model	37
3.1.6. Navegación	40
3.1.7. Temas y estilo	42
3.2. Análisis de resultados	46
3.2.1. Modo 1: Preguntas de técnica	46
3.2.2. Modo 2: Preguntas sobre secuencia	50
3.2.3. Modo 3: Preguntas de técnica y orden	50
3.2.4. Modo 4: Práctica	51
3.2.5. Modo 5: Repaso con fichas	53
4 Conclusiones y recomendaciones	55
4.1. Conclusiones	55
4.2. Recomendaciones	56
A Información del código	59
A.1. Diagramas del programa	60
A.1.1. Diagrama de flujo de preguntas	60
A.1.2. Clase Asana	61
A.1.3. DataSource	62
Bibliografía	63

Índice de figuras

2.1. Pasos del Vinyasa. Este se realiza para conectar cada postura en la serie primaria. [1] . . .	6
2.2. Durante una clase de Ashtanga Yoga, los estudiantes practican los asanas individualmente, progresando al ritmo de cada persona. [2]	7

2.3.	Asanas del saludo al sol A. [1]	9
2.4.	Asanas del saludo al sol B. [1]	9
2.5.	Asanas de pie de la serie primaria. [1]	10
2.6.	Asanas sentado, o Chikitsa de la serie primaria. [1]	10
2.7.	Asanas de cierre de la serie primaria. [1]	11
2.8.	Ejemplo de una interfaz con un botón y una imagen en el layout editor.	13
2.9.	Algunas de las plantillas disponibles en Android Studio.	14
2.10.	Emulación mediante Android Studio de un teléfono Pixel 3a XL con el API 29.	14
2.11.	Al utilizar el API 31, el más reciente, Android Studio avisa que la aplicación va a funcionar en menos del 1% de dispositivos.	15
2.12.	El API 21 es compatible con el 94.1% de los dispositivos.	15
2.13.	Distintas vistas para formar una interfaz. ImageView para el fondo, Button para el botón en el pastel y TextView para enseñar el texto.	17
2.14.	Ejemplo de un gráfico de navegación de Android Jetpack. [3]	18
2.15.	Ejemplo de algunos componentes que suministra Material. [4]	19
2.16.	Algunos iconos con licencia libre de Material. [5]	19
3.1.	Ejemplo de una dinámica de Drops para aprender persa. [6]	22
3.2.	Captura del menú principal de Anki con un fichas de Ashtanga Yoga. Se tienen distintos grupos de ficha para cada secuencia.	22
3.3.	Captura de una ficha Anki para la secuencia sentado.	23
3.4.	Ideas iniciales de dinámicas y flujo de la aplicación.	24
3.5.	Data class para el dato Asana.	25
3.6.	Algunas posturas de la secuencia de pie en el objeto DataSource.	25
3.7.	Comparación de imágenes descargadas y dibujadas.	26
(a).	Imagen descargada	26
(b).	Imagen dibujada	26
3.8.	Diseño del menú principal.	27
3.9.	Diseño de menú de selección de secuencia.	28
3.10.	Diseño de menú de selección de técnicas.	29
3.11.	Diseño de preguntas sobre técnica. Android Studio cuenta con funciones para colocar texto e imágenes que solo el desarrollador puede ver mientras trabaja con el diseño.	31
3.12.	Diseño de preguntas sobre postura.	32
3.13.	Diseño de preguntas sobre secuencia.	33
3.14.	Diseño de modo de práctica.	34
3.15.	Diseño de la carta en asana_card.xml.	35
3.16.	Diseño de la actividad de repaso review_activity.xml.	36
3.17.	Declaración de una variable de Asana Live Data en el Main View Model.	37
3.18.	En el archivo de diseño se coloca el valor de esa variable para mostrar al usuario.	37
3.19.	Desde el archivo de diseño se llama a una función de una clase y se le pasa un argumento del main view model.	38

3.20. Al terminar la secuencia reset reinicia el puntaje, la posición, la secuencia y modo seleccionado y otras variables utilizadas en distintas preguntas.	39
3.21. Gráfico de navegación del programa.	40
3.22. En esta función del menú de secuencias dependiendo del modo que se había seleccionado en el menú principal se elige a cuál fragmento avanzar y se llama al controlador de navegación para realizar la acción.	41
3.23. Al seleccionar el botón de Review en el menú principal se guarda el modo en el view model y se abre la actividad de fichas mediante un intent.	41
3.24. Para seleccionar los colores se utilizó el selector de paletas de color de material [4]	42
3.25. Las variaciones de los colores se obtuvieron mediante la Color Tool de Material Design [4].	43
3.26. En el archivo colors.xml se definen los colores que va a utilizar la aplicación.	44
3.27. theme.xml define como se van a aplicar los colores a toda la aplicación.	44
3.28. Imagen original utilizada para el ícono. Obtenida de [7]	45
3.29. El ícono de la aplicación es una flor de loto compuesta por distintas posturas de yoga. La resolución es varía con el dispositivo utilizado.	45
3.30. Navegación desde el menú principal a una pregunta sobre el nombre de una postura.	46
3.31. Ejemplo de interacción del usuario con preguntas sobre el nombre de posturas en la secuencia Suryanamaskara A. Los puntos verdes representan clics del usuario.	47
3.32. Función checkAnswer() se encarga de revisar la respuesta, cambiar el fondo de pantalla y desactivar temporalmente los botones.	48
3.33. El modo de drishtis funciona de manera similar al de nombres. Aquí se utiliza la secuencia Suryanamaskara B.	48
3.34. Ejemplo de modo de preguntas sobre el nombre pero invertido, donde se muestran 4 posturas y un nombre. Aquí se utiliza la secuencia de pie.	49
3.35. En el modo aleatorio se le puede realizar cualquiera de las 3 preguntas al usuario. Los resultados se enseñan de la misma manera que los modos anteriores. Ejemplo con Suryanamaskara A.	49
3.36. Ejemplo del usuario contestando perfectamente preguntas sobre el orden de Suryanamaskara A.	50
3.37. Ejemplo de funcionamiento con la secuencia Suryanamaskara B. Esta tiene 19 posturas por lo que se realizan 38-1 preguntas.	51
3.38. Ejemplo de navegación a través de las posturas de la secuencia completa. Aquí el usuario eligió practicar la secuencia completa.	52
3.39. El ícono de la aplicación es una flor de loto compuesta por distintas posturas de yoga. La resolución es varía con el dispositivo utilizado.	53
A.1. Diagrama de flujo para las dinámicas de preguntas.	60
A.2. Diagrama de la clase Asana.	61

A.3. Diagrama de objeto para DataSource.	62
--	----

Índice de tablas

1.1. Cronograma del proyecto.	4
---------------------------------------	---

Nomenclatura

<i>API</i>	Interfaz de programación de aplicaciones (del inglés <i>application programming interface</i>)
<i>IDE</i>	Entorno de desarrollo integrado (del inglés <i>integrated development environment</i>)
<i>U.I</i>	Interfaz de usuario (del inglés <i>user interface</i>)
<i>XML</i>	Lenguaje de Marcado Extensible (del inglés <i>eXtensible Markup Language</i>)
EIE	Escuela de Ingeniería Eléctrica de la Universidad de Costa Rica

Capítulo 1

INTRODUCCIÓN

1.1. Descripción General

El Ashtanga Yoga es un estilo de yoga basado en distintas series cada una con secuencias de posturas, llamadas asanas. Cada secuencia de posturas dentro de cada serie tiene un orden específico que debe ser seguido, por lo que poder memorizar la secuencia es una parte importante de la práctica de cada serie. Durante cada postura se debe también tomar en cuenta la respiración y la dirección en la que están apuntando los ojos. Físicamente, la práctica de este tipo de yoga ya presenta un nivel de dificultad importante; la adición de memorizar los detalles de cada postura y el orden de cada secuencia presenta otro reto para las personas que practican esta disciplina.

Este proyecto busca facilitar el aprendizaje de la primera serie de Ashtanga Yoga mediante el desarrollo de una aplicación en el sistema operativo móvil Android. El desarrollo de la aplicación en este sistema operativo se realizará mediante la herramienta de desarrollo Android Studio y el lenguaje de programación Kotlin. El IDE de Android Studio tiene una gran cantidad de herramientas y funcionalidades para facilitar el desarrollo de aplicaciones para este sistema operativo. El lenguaje Kotlin es un lenguaje de programación moderno desarrollado para la programación de aplicaciones por lo que en conjunto con Android Studio va a facilitar el proceso de desarrollo de la aplicación.

La primera serie de Ashtanga Yoga, llamada Yoga Chikitsa, es la primera de seis series de secuencias [8]. Cada series está compuesta por múltiples secuencias con muchas posturas distintas, por lo que en este proyecto se va a enfocar únicamente en esta primera serie. Mediante el uso de una aplicación móvil se busca poder crear una serie de dinámicas dentro de la aplicación que asistan a las personas a memorizar los múltiples pasos que se requieren para completar esta secuencia. Esta aplicación busca ayudar a aprender los distintos detalles de las posturas que esta primera serie tiene. Esta aplicación deberá proporcionar a los usuarios una forma eficiente y entretenida de aprender sobre esta secuencia que, a pesar de ser la primera, no es corta ni simple.

1.2. Alcances del proyecto

Se va a diseñar una aplicación para dispositivos móviles que funcionan con el sistema operativo Android. Esta aplicación va presentar dinámicas para el aprendizaje de únicamente la primera serie del Ashtanga Yoga. Estas dinámicas van a ayudar al usuario a aprender sobre distintos detalles de esta secuencia como: el nombre de cada postura, posición de la postura, respiración, orden de la secuencia, enfoque (de los ojos) y otras opciones para realizar la postura. Se busca tener al menos dos dinámicas distintas que enseñen al usuario de forma sobre esta primera secuencia.

1.3. Justificación del proyecto

La serie I del Ashtanga Yoga presenta un nivel de complejidad considerable. Esto se debe a que está compuesto por una serie de posturas, cada una con su propio nombre y detalles sobre respiración y posición de los ojos. Además todas estas posturas tienen un orden que debe ser seguido para completar la serie. A partir de estas peculiaridades de la serie, se busca diseñar una aplicación que ayude a las personas practicantes a estudiar y recordar cómo realizar estas posturas y en qué orden deben hacerlas. Una aplicación móvil permitiría a los usuarios estudiar y practicar esta serie de una forma cómoda y portátil.

1.4. Objetivos

1.4.1. Objetivo general

Crear una aplicación en Android que permita aprender la Serie I de Ashtanga Yoga con diferentes dinámicas.

1.4.2. Objetivos específicos

1. Realizar una investigación bibliográfica sobre desarrollo en Android.
2. Realizar una investigación sobre la secuencia de serie I de Ashtanga Yoga y sobre programas similares en el mercado.
3. Efectuar un levantamiento de requerimientos que identifique las necesidades de los usuarios.
4. Implementar una aplicación para Android con una interfaz intuitiva para el usuario.
5. Validar la ejecución funcional de la aplicación.

1.5. Metodología

1. Aprender sobre el desarrollo de aplicaciones mediante el IDE Android Studio y el lenguaje de programación Kotlin.
2. Estudiar la Serie I de Ashtanga Yoga para comprender que es lo que va a tener que enseñar la aplicación.
 - Estudiar el orden de la secuencia y detalles de cada postura.
 - Buscar material visual de uso libre para implementar en la aplicación, como imágenes de la realización de cada postura.
3. Estudiar sobre otras aplicaciones similares, como aplicaciones de yoga o para memorizar temas.
 - Analizar la interfaz y las dinámicas que estas aplicaciones utilizan.
4. Consultar con profesores o practicantes de Ashtanga Yoga sobre los detalles que buscan aprender en la serie y cómo aprenderlos.
5. Combinar los conocimientos estudiados sobre la serie con las necesidades de los usuarios para idear dinámicas apropiadas para el aprendizaje de esta secuencia.
6. Implementar las dinámicas programando en Android Studio enfocándose en la funcionalidad de estas.
7. Con la aplicación funcionando, trabajar en mejorar la interfaz para que sea más intuitiva, ordenada y estética.
8. Enseñar a algún practicante de Ashtanga Yoga la aplicación para recibir realimentación y así mejorar la interfaz y las dinámicas.
9. Probar la aplicación mediante un simulador de un dispositivo móvil e instalarla en uno real para detectar y arreglar errores que esta pueda presentar durante uso real.
 - Utilizar la aplicación para aprender sobre la secuencia.
 - Reflexionar sobre que tanto ayuda la aplicación a aprender los detalles de la secuencia y compararla con métodos más simples como fichas de información de la serie.

1.6. Cronograma de actividades

Semana	Actividades
12-18 de setiembre	Trabajo en Anteproyecto I
19-25 de setiembre	Practicar y estudiar desarrollo de aplicaciones en Android Studio
26 de setiembre - 2 de octubre	Realizar estudio bibliográfico sobre Ashtanga Yoga y aplicaciones similares.
3-9 de octubre	Realizar Avance I (capítulo II y secciones adicionales)
10-16 de octubre	Comenzar desarrollo de la aplicación e identificar necesidades de los usuarios
17-23 de octubre	Implementación de dinámicas para aprendizaje de la serie con base en las necesidades de los usuarios
24-30 de octubre	Implementación de dinámicas para aprendizaje de la serie
31 de octubre - 6 de noviembre	Realizar Avance II
7-13 de noviembre	Trabajar en interfaz de la aplicación
14-20 de noviembre	Validar funcionamiento de la aplicación
21-27 de noviembre	Elaboración de borrador final completo.
28 de noviembre - 4 de diciembre	Verificar funcionamiento de la aplicación y trabajar en la presentación
5-11 de diciembre	Preparación de la presentación
12-18 de diciembre	Presentación final
19 de diciembre - 14 de enero	Elaboración del reporte final.

Tabla 1.1: Cronograma del proyecto.

Capítulo 2

MARCO TEÓRICO

El Ashtanga Yoga es 99% práctica y 1% teoría.

Shri K Patthabi Jois

2.1. Ashtanga Yoga

2.1.1. Definición e historia

La palabra Ashtanga, en sanscrito significa “ocho extremidades” u “ocho miembros”. En el Ashtanga, Yoga puede significar colocar el yugo (mismo origen etimológico), unión, de la mente o del cuerpo, el alma y conectar [9]. El Ashtanga Yoga entonces se podría entender como el Yoga de los ocho miembros. Estos ocho miembros son distintas disciplinas de la práctica del yoga que involucran aspectos físicos y mentales. Estas 8 disciplinas en orden son: Yama (códigos morales), Niyama (auto purificación), Asana (postura), Pranayama (control de la respiración), Pratyahara (control de los sentidos), Dhayana (concentración), Dhyana (meditación) y Samadhi (contemplación) [9]. En el Ashtanga se comienza con el Asana, el tercer miembro, con el objetivo de estabilizar el cuerpo y la mente. Luego con el cuerpo listo se pueden comenzar a entender y practicar los primeros dos miembros. Entonces, se puede progresar hasta finalmente llegar la contemplación o Samadhi. En [10] se explica que: metodológicamente, el Ashtanga se enfoca primero en la práctica de los asanas para establecer salud, corregir desequilibrios y reforzar el sistema, estabilizando el cuerpo y la mente, sin los que no somos capaces de controlar nuestros órganos sensoriales. El Ashtanga yoga es una disciplina compleja donde físicamente se conectan posturas a través de movimientos en conjunto con técnicas de respiración y direccionamiento de la vista mientras que también se está ejercitando la mente y realizando técnicas de meditación y contemplación.

El Ashtanga Vinyasa Yoga fue elaborado por el gurú indio Shri K. Pattabhi Jois nacido en 1915. Él estableció el Instituto de Investigación de Ashtanga Yoga en el 1948, luego de haber practicado y enseñado yoga por muchos años. Jois se dedicó a la enseñanza del Ashtanga por 70 años, donde este ganó popularidad a través del mundo. El conocimiento sobre este tipo de Yoga primero fue compartido por Jois hacia sus estudiantes y familia. Esta transmisión ininterrumpida del conocimiento del gurú a

su estudiante es llamada Parampara [10]. Actualmente el Parampara del Ashtanga se mantiene a través de los familiares de Jois y sus estudiantes.

2.1.2. Vinyasa

El Vinyasa consiste en series de movimientos utilizados para transicionar entre las diferentes posturas del Ashtanga. La traducción del sanscrito de Vinyasa puede significar orden y disposición [11]. El Vinyasa es utilizado para impulsar correctamente la circulación de sanguínea en el cuerpo [10]. En el Vinyasa, el movimiento debe ir acompañado por respiración controlada, activación de ciertas partes del cuerpo y un enfoque de la vista. Esta serie de transiciones se puede realizar para cambiar entre cada postura, o se puede optar por solo realizar en ciertas partes de una serie de Ashtanga dependiendo de como se esté realizando el Ashtanga Yoga.



Figura 2.1: Pasos del Vinyasa. Este se realiza para conectar cada postura en la serie primaria. [1]

2.1.3. Asana

El Asana es uno de los ocho componentes del Ashtanga Yoga. Este es el tercer miembro del Ashtanga y consiste en la práctica de las posturas. La práctica de Asana permite preparar el cuerpo físicamente para poder avanzar por los 7 caminos restantes. Mediante el condicionamiento físico practicando los asanas, los practicantes de este tipo de yoga pueden comenzar a entender los demás miembros del Ashtanga y llegar a niveles más altos de yoga donde se puede encontrar paz y estabilidad [10]. Hasta que no se domine la práctica de un asana, no se le pueden agregar más cualidades del Ashtanga, como el control de la respiración o “Pranayama”. Este tipo de yoga tiene distintas series de asanas distintas, donde cada serie está compuesta por alrededor de 40 asanas [11]. En la Figura 2.2 se observa una estudiante realizando la asana Setu Bandhasana, que forma parte de la serie primaria de Ashtanga Yoga. Cada asana tiene un nombre en sanscrito, una posición de enfoque de los ojos, activación de puntos del cuerpo, respiración que la acompaña y tiene un orden específico en relación con las otras asanas en la serie. Se espera que los practicantes de este yoga puedan aprender todos estos detalles para poder practicar correctamente y obtener los beneficios del Ashtanga. La asana se puede ver como la postura estática mientras que el vinyasa es el movimiento entre las posturas.



Figura 2.2: Durante una clase de Ashtanga Yoga, los estudiantes practican los asanas individualmente, progresando al ritmo de cada persona. [2]

2.1.4. Ujjai

El tipo de respiración que se realiza en cada asana es llamada Ujjai. La traducción de Ujjai se puede interpretar como “respiración victoriosa”. Esta respiración requiere de una técnica particular y tiene un sonido característico. La respiración debe ser realizada únicamente por la nariz. Se busca alcanzar un balance entre la puraka (inhalación) y la rechaka (exhalación) donde ambas sean por la misma cantidad de tiempo. En cada Asana esta respiración debe ser simultánea y estar en sincronía con el movimiento del cuerpo. El Ujjai busca incrementar la cantidad de oxígeno que entra al cuerpo, esto se logra utilizando la garganta para mover el aire a través de la nariz. El sonido característico del Ujjai se obtiene por la fricción generada por el aire al pasar por la garganta y es distinto al sonido restrictivo que se realiza al forzar aire por la nariz, como cuando se está olfateando [9].

2.1.5. Bandha

En sanscrito, Bandha se puede traducir como sello, cerradura, bloqueo o cierre. En [9] se menciona que este significado es paradójico debido a que el resultado de la aplicación del Bandha más bien desbloquea energía dentro del cuerpo y permite que esta fluya a través de los canales de energía del cuerpo. A partir de su significado literal, no es tan claro como se utiliza en la práctica del yoga. Palavecino en su tesis [11], define los Bandhas como “contracciones musculares voluntarias que impiden que la energía del cuerpo se disipe”. Uniendo este significado con el original de bloqueos, se puede ver como los Bandhas consisten en el bloqueo de las salidas de energía del cuerpo para poder mantenerla y utilizarla dentro del mismo. Dependiendo de la ubicación del cuerpo de las regiones musculares que se activan en el Bandha, existen Bandhas distintos. Los tres Bandhas principales son:

- Mula Bandha: Mula significa raíz. Esta bandha involucra la contracción de músculos específicos en el suelo pélvico para mantener el flujo de energía dentro del cuerpo. Esta funciona como la raíz para tener una base firme mientras se practican los asanas.
- Uddiyana Bandha: Uddiyana se puede traducir como elevarse o volar hacia arriba [9]. Este cierre se utiliza para establecer un flujo de energía dirigido hacia arriba. Esta involucra los músculos abdominales, la caja torácica y el diafragma. El control de estos músculos permite controlar y aumentar la cantidad de oxígeno que entra a los pulmones. Este tiene una relación directa con el control de la respiración o Ujjai.
- Jalandhara Bandha: Este cierre se relaciona al movimiento de músculos por el esternón, la barbilla y el cuello. Este requiere de una posición de la cabeza específica donde la barbilla apunta al espacio entre las clavículas, dirigiendo la mirada hacia el ombligo [9].

2.1.6. Dristi

Previamente se ha mencionado la importancia del punto de foco visual en la práctica de los asanas. Este direccionamiento de la vista hacia un punto específico es conocido como Dristi. El dristi ayuda a concentrar la mente y permite avanzar a las etapas 6 y 7 del Ashtanga, Dharana y Dhyana. Estos son los miembros de la concentración y la meditación. Cada asana se practica con un dristi específico, cada uno con un nombre en sanscrito. Los dristis utilizados son los siguientes:

- Urdhva: hacia el cielo
- Nasagra: punta de la nariz
- Brumadhyya: entre las cejas (tercer ojo)
- Hastagra: mano
- Angustha: dedos pulgares
- Nabi chakra: ombligo
- Padayoragram: dedos gordos del pie
- Daksina parsva: derecha lejana
- Vamana parsva: izquierda lejana

2.1.7. Surya Namaskara

El Surya Namaskara es una serie de vinyasas (movimientos) que se realiza antes de comenzar la serie de Ashtanga. Surya Namaskara es conocido como el saludo al sol. Esta serie de movimientos no es exclusiva del Ashtanga y se realiza en otros tipos de yoga. El Surya Namaskara es el primer paso que se toma cuando se comienza el aprendizaje del Ashtanga Yoga. En este saludo cada movimiento se acompaña

2.1. Ashtanga Yoga

9

por un dristi, control de la respiración con Ujjai, enfoque en los dristis y control de las bandhas. En Ashtanga están el Surya Namaskara A y B. El Surya Namaskara A consiste en 9 vinyasas (movimientos) que se realizan al ritmo de la respiración ujjai [9]. El saludo A funciona como calentamiento y prepara el cuerpo para los siguientes movimientos. Este saludo se realiza 5 veces. Luego de realizar el saludo A, se procede al B. Este prepara al cuerpo para comenzar con la serie 1 de Ashtanga y consiste en 17 movimientos y 19 asanas (posturas).



Figura 2.3: Asanas del saludo al sol A. [1]



Figura 2.4: Asanas del saludo al sol B. [1]

2.1.8. Serie 1 – Yoga Chikitsa

El Ahstanga Yoga está compuesto por 3 series principales. La serie primaria, serie 1 o Yoga Chikitsa, es el primer nivel de asanas que se aprende. La serie primaria busca alinear y purificar el cuerpo internamente y externamente [9]. La práctica de toda la serie primaria con sus saludos al sol respectivos puede tardar alrededor de 90 minutos [12] aunque se pueden realizar variaciones más cortas. Esta serie está compuesta por 3 secuencias distintas. La primera es la secuencia de pie que consiste en 18 asanas. Luego se realiza la secuencia sentada, compuesta por 28 asanas. Para terminar se realiza la secuencia final de 18 asanas. Entre cada asana se puede realizar el vinyasa de la Figura 2.1. Estas secuencias se realizan en un orden específico y cada asana se acompaña por bandhas, dristis, y respiraciones por lo que además de tener que aprender a realizar las posturas hay un aspecto bastante importante de memorización de la secuencia. Esta secuencia a pesar de ser la primaria, presenta un alto nivel de complejidad por lo que la ayuda de un profesor es importante. Las series de Ashtanga Yoga son enseñadas en clases Mysuru, o Mysore, llamadas así porque originalmente este Yoga se enseñaba en la ciudad de Mysuru en India. Durante estas clases grupales el gurú, o profesor, guía a cada estudiante individualmente por cada una de las asanas y sus detalles. Luego de lograr la serie primaria, se puede progresar a la serie intermedia (Nadi Shodhana) para poder llegar a la serie avanzada (Sthira Bhaga).



Figura 2.5: Asanas de pie de la serie primaria. [1]

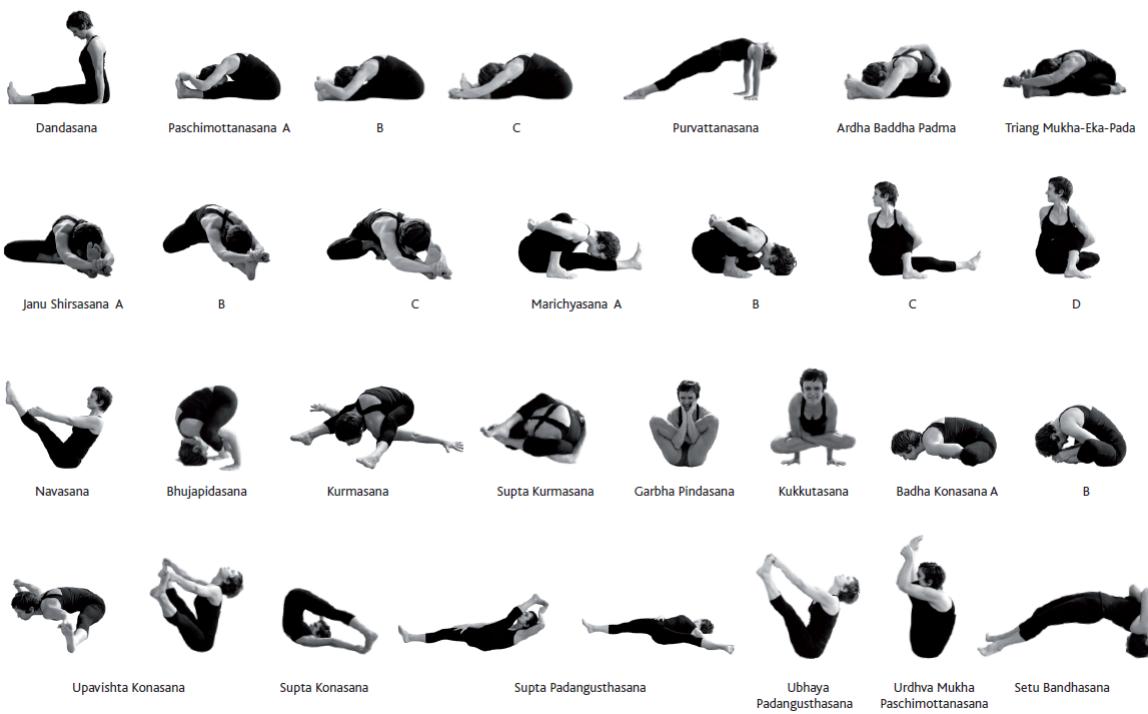


Figura 2.6: Asanas sentado, o Chikitsa de la serie primaria. [1]

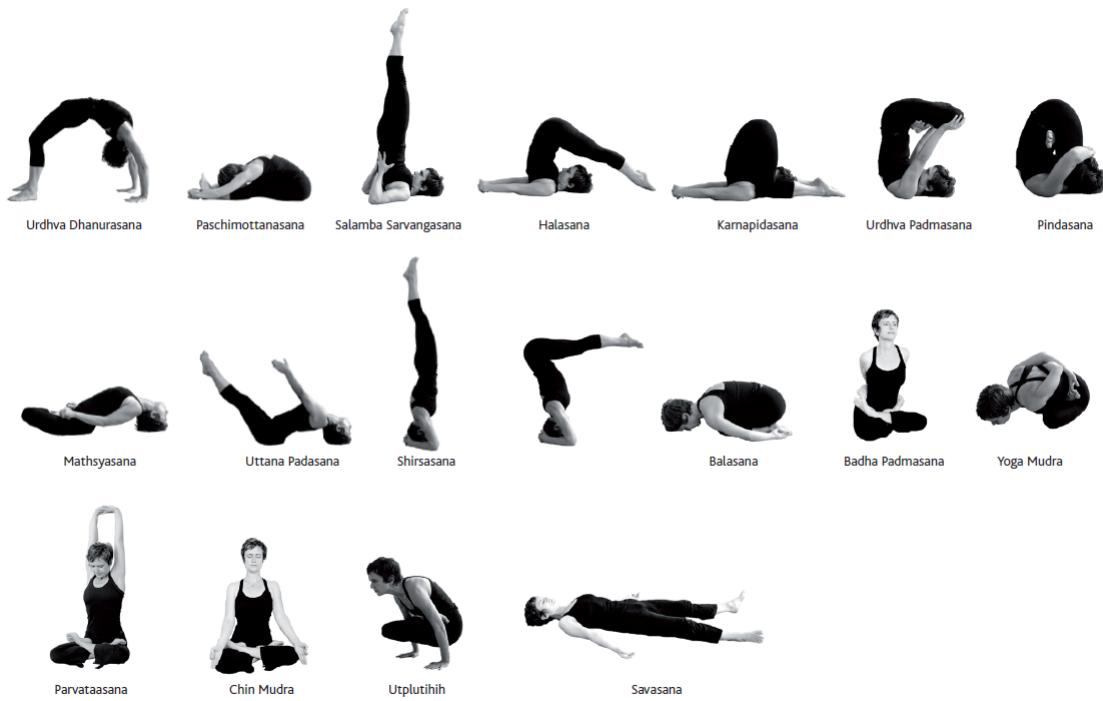


Figura 2.7: Asanas de cierre de la serie primaria. [1]

2.2. Desarrollo en Android

2.2.1. Sistema operativo – Android

Android es un sistema operativo móvil creado en el 2008, que actualmente es común en teléfonos celulares, televisores inteligentes, relojes inteligentes, vehículos y hasta refrigeradoras. Android es desarrollado por Google y es de código abierto por lo que es posible para cualquier persona ver, descargar, modificar, mejorar y redistribuir su código fuente sin necesidad de pagar nada a Google [13]. Las versiones que se encuentran en dispositivos móviles usualmente son versiones modificadas y propietarias de este software por las compañías que venden los dispositivos. Algunas compañías que comúnmente utilizan variaciones de Android en sus dispositivos son: Google, Huawei, Samsung y LG. Para el 2016 Android estaba implementado en más de 24000 dispositivos distintos y tenía más de 1 millón de aplicaciones disponibles para sus usuarios [13]. Para el 2021 ese número ha aumentado a casi 3 millones de aplicaciones [14], esto ignorando las aplicaciones que se pueden descargar fuera de la tienda de Google. La popularidad de este sistema operativo y lo abierto que es como software libre, lo hace una de las plataformas más populares para que personas desarrollen aplicaciones. Google cuenta con múltiples fuentes de información, tutoriales y herramientas para facilitar el desarrollo de aplicaciones para este sistema operativo, tanto para desarrolladores veteranos como para programadores principiantes [15].

El sistema operativo Android se ha ido actualizando con el tiempo desde que se publicó Android 1.0 en el 2008. Desde entonces con cada actualización se han agregado nuevas características y funcionalidades. Actualmente la versión más nueva de Android es Android 12, con el nombre código Snow Cone (cono de nieve). Las versiones de Android son comúnmente conocidas por sus nombres que usualmente hacen referencia a postres como Android Oreo y Android Marshmallow, aunque actualmente solo se les coloca el nombre de número de versión por la que van por ejemplo, Android 10, 11 y 12. Un dato importante que hay que saber con respecto a las versiones es el nivel de la interfaz de programación de aplicaciones (API). El API es la capa de software que funciona como intermediario para que dos aplicaciones se comuniquen entre sí [16]. En el caso de Android este permite la interacción entre el sistema operativo y las aplicaciones que se diseñan para él. Cada versión de Android tiene actualizaciones al API que agregan más funcionalidad a las aplicaciones y pueden también mejorar aspectos de seguridad u optimización del sistema. Para el desarrollo de aplicaciones en Android es importante saber el API con el que se está trabajando ya que este es el que contiene las distintas bibliotecas e instrucciones que se van a utilizar para el desarrollo. El API de Android tiene compatibilidad hacia adelante lo que hace que usualmente las aplicaciones desarrolladas en un API más viejo funcionen correctamente en las versiones nuevas. El caso opuesto (compatibilidad hacia atrás) no siempre es cierto ya que con un API más nuevo puede que se desarrollen aplicaciones con funciones que no existen en un API anterior por lo que no funcionarían.

2.2.2. Entorno de desarrollo – Android Studio

La programación de aplicaciones móviles puede ser complicada debido a que se debe programar el código, desarrollar la interfaz del usuario, trabajar con distintas ventanas y probar el funcionamiento de la aplicación entre otros pasos necesarios para obtener una aplicación funcional. Para simplificar

en este proceso, Google desarrolló un IDE (entorno de desarrollo integrado) especializado para el sistema operativo Android. Esta aplicación, contiene todo lo necesario para ayudar al desarrollador a programar aplicaciones eficientemente. Una de sus funcionalidades es el “*visual layout editor*” que permite diseñar interfaces gráficas complejas mediante un editor visual sin necesidad de tener que manualmente escribir en un archivo .xml la posición de todos los objetos (aunque Android Studio también facilita la edición de estos archivos). Este editor permite colocar elementos como botones, imágenes y cuadros de texto y ajustar sus posiciones y contenidos fácilmente.

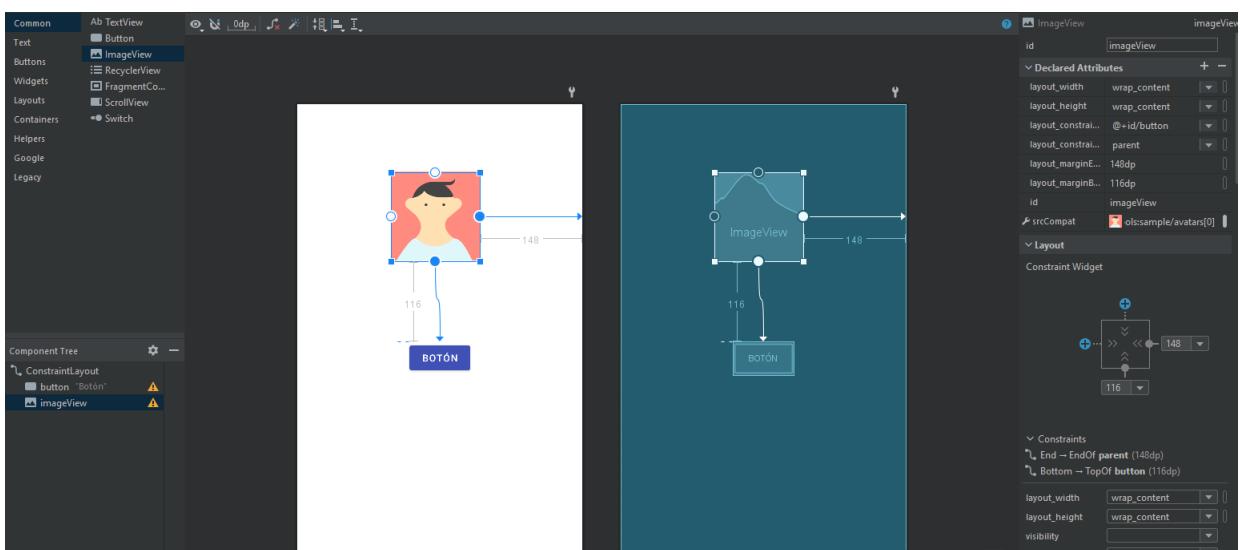


Figura 2.8: Ejemplo de una interfaz con un botón y una imagen en el layout editor.

Otra manera en la que el IDE facilita el desarrollo es mediante el uso de plantillas o *templates*. Estos funcionan como base para desarrollar distintas aplicaciones o ventanas. Estas plantillas pueden contener menús, uso de mapas y actividades de pantalla completa entre otros. Además cuenta con plantillas para desarrollar aplicaciones para televisiones inteligentes, vehículos y relojes inteligentes.

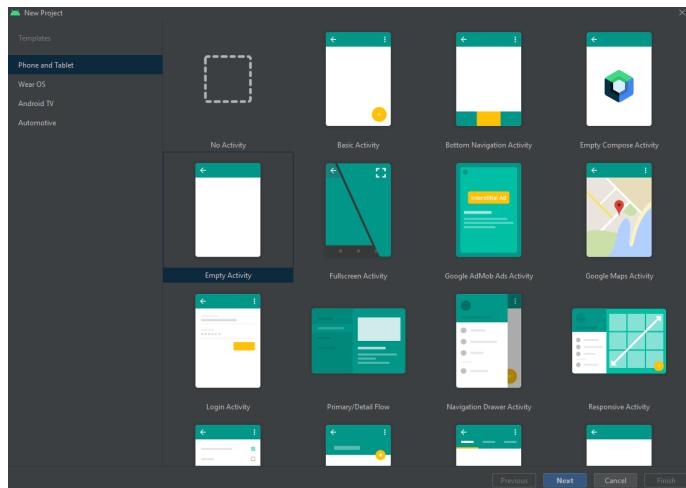


Figura 2.9: Algunas de las plantillas disponibles en Android Studio.

El IDE de Android Studio también cuenta con un emulador que permite probar las aplicaciones sin tener que instalarlas en un dispositivo físico. Este emulador es útil ya que permite probar rápidamente cambios a las aplicaciones y revisar compatibilidad con distintos dispositivos.

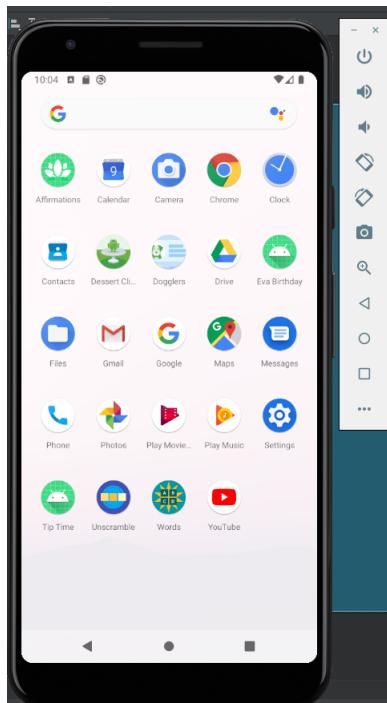


Figura 2.10: Emulación mediante Android Studio de un teléfono Pixel 3a XL con el API 29.

En el desarrollo de aplicaciones, es importante elegir el API con el que se va a trabajar. Esto puede afectar directamente la cantidad de dispositivos que van a poder utilizar la aplicación y que funcionalidad esta va a tener. Al comenzar un proyecto con Android Studio, este permite elegir el API en el que se va a desarrollar y da un estimado de la cantidad de dispositivos que van a poder utilizar la aplicación.

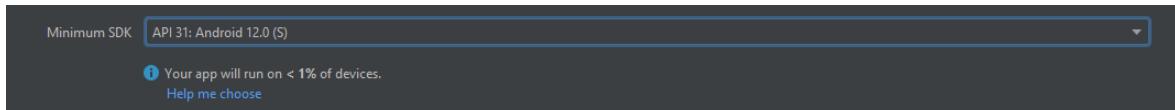


Figura 2.11: Al utilizar el API 31, el más reciente, Android Studio avisa que la aplicación va a funcionar en menos del 1 % de dispositivos.

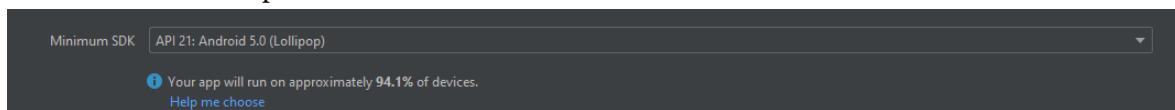


Figura 2.12: El API 21 es compatible con el 94.1 % de los dispositivos.

La programación dentro del IDE se debe realizar en algún lenguaje de programación. Los lenguajes que se han utilizado en el desarrollo de aplicaciones para Android son: C++, Java y Kotlin. Actualmente, el lenguaje preferido por Google es Kotlin. El IDE contiene soporte para Kotlin y Java.

2.2.3. Lenguaje – Kotlin

El 2017 Google anunció que iba a aceptar el desarrollo de aplicaciones para Android en el lenguaje de programación Kotlin [17]. Dos años después Google desplazó a Java como el lenguaje preferido para darle este título a Kotlin. Esto significaba que todos los nuevos avances en bibliotecas de Android iban a llegar primero a Kotlin. En la página oficial para desarrolladores de Android [18] describen a Kotlin como “un lenguaje expresivo y conciso que reduce errores comunes de código y que fácilmente se integra en aplicaciones existentes”. Técnicamente Kotlin es un lenguaje multiplataforma, multi propósito, de código abierto con tipado estático [19]. Según [18] algunas ventajas de Kotlin son:

- **Expresivo y conciso:** Debido a ser un lenguaje de alto nivel enfocado a reducir la cantidad de expresiones permite realizar funciones complicadas con pocas líneas de código.
- **Código seguro:** Contiene múltiples características que dificultan errores de programación comunes que podrían hacer que la aplicación deje de funcionar.
- **Interpolable:** En una misma aplicación se puede utilizar código de Java y de Kotlin sin problema.
- **Concurrencia estructurada:** Facilita el trabajo con código asincrónico y código bloqueante.

El soporte dado por Google a este lenguaje facilita el aprendizaje del mismo y el desarrollo de aplicaciones. Ya que Google cuenta con amplia documentación, ejemplos y tutoriales sobre este lenguaje y como utilizarlo para diseñar aplicaciones para su sistema operativo.

2.2.4. Diseño – Elementos de la interfaz

El diseño de interfaces gráficas en Android consiste en el uso de vistas o *Views* para construir lo que se desea enseñar en la pantalla. *View* es una clase de Kotlin y Java que funciona como el bloque del que se construyen todos los demás elementos de la interfaz. Las vistas son una área rectangular que se va a encontrar en la pantalla y que dibuja los elementos que se van a colocar y se encarga del manejo de eventos [20]. Algunos ejemplos de vistas son: botones, imágenes, barras de progreso y switches. Los elementos interactivos de la interfaz son llamados *widgets*. Mediante el editor de layout de Android Studio es fácil revisar todos los tipos de vistas que hay y para agregarlos a una interfaz solo es necesario arrastrarlos de un menú a la posición donde se van a utilizar. Dentro del código del programa se utilizan *event handlers* (manejadores de eventos) para controlar que sucede en la aplicación cuando el usuario interactúa con las vistas. Las propiedades de cada vista (posición, identificador, tamaño) se guardan en un archivo de extensión .xml (lenguaje de marcado extensible), que se puede editar directamente o con el editor de layout de Android Studio.

Existe una subclase de *Views* que es el *ViewGroup*. Este es una vista que es capaz de contener otras vistas. Cuando se desean agrupar componentes se utilizan *Viewgroups* y las vistas dentro de esta se llaman *children* (hijos) del *Viewgroup* en el que se encuentran. Existen distintas clases de contenedores como el *ScrollView* que contiene vistas que se pueden desplazar tocando la pantalla; *RecyclerView* que permite reutilizar vistas para enseñar distintos elementos y *Toolbar* que puede contener botones con ciertas funciones de utilidad.

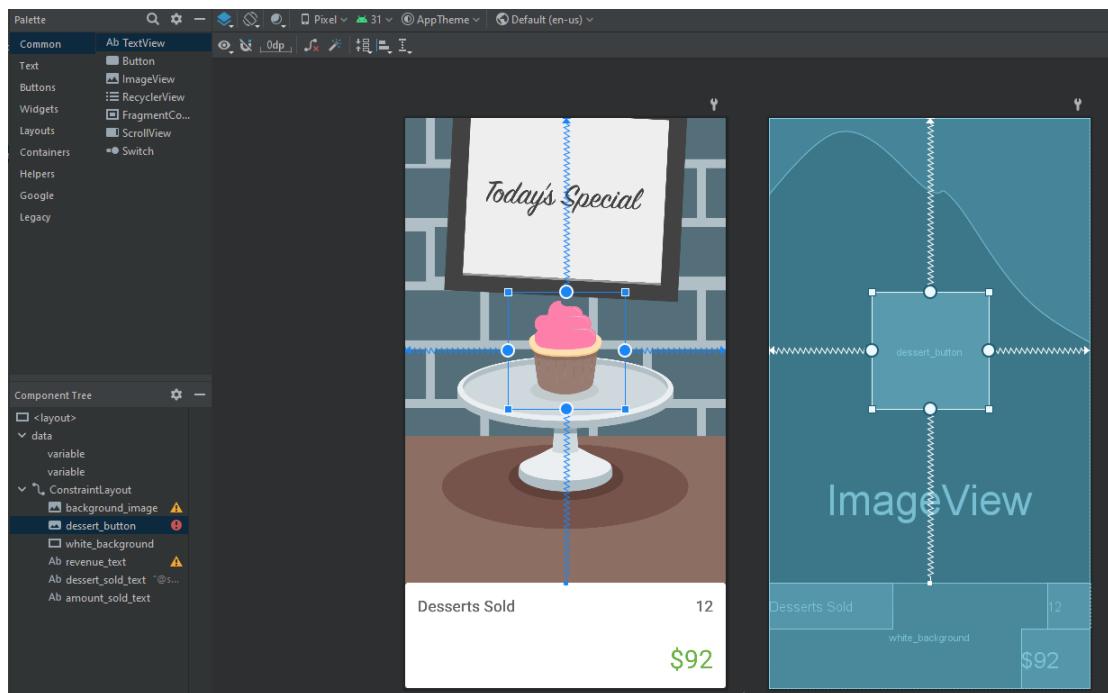


Figura 2.13: Distintas vistas para formar una interfaz. ImageView para el fondo, Button para el botón en el pastel y TextView para enseñar el texto.

En la Figura 2.13 se observa la construcción de una interfaz simple. Aquí se utiliza un layout llamado ConstraintLayout. Este es parte de un grupo de bibliotecas llamado Android Jetpack que tiene múltiples clases y funciones que facilitan el desarrollo ordenado y con buenas prácticas de aplicaciones. Este ConstraintLayout funciona como un contenedor para todos los elementos de la interfaz. A partir de este se pueden asignar posiciones relativas a él para cada vista y así tener una interfaz ordenada sin importar el dispositivo que se utilice.

Android Jetpack también facilita el uso de múltiples ventanas o pantallas y la conexión y transición entre estas. En Android Jetpack a cada una de estas ventanas se les llama fragmentos. Jetpack tiene un componente de navegación donde se utiliza un archivo XML para relacionar los destinos de los distintos fragmentos del programa. Este archivo se puede editar directamente o mediante un editor de gráfico de Android Studio.

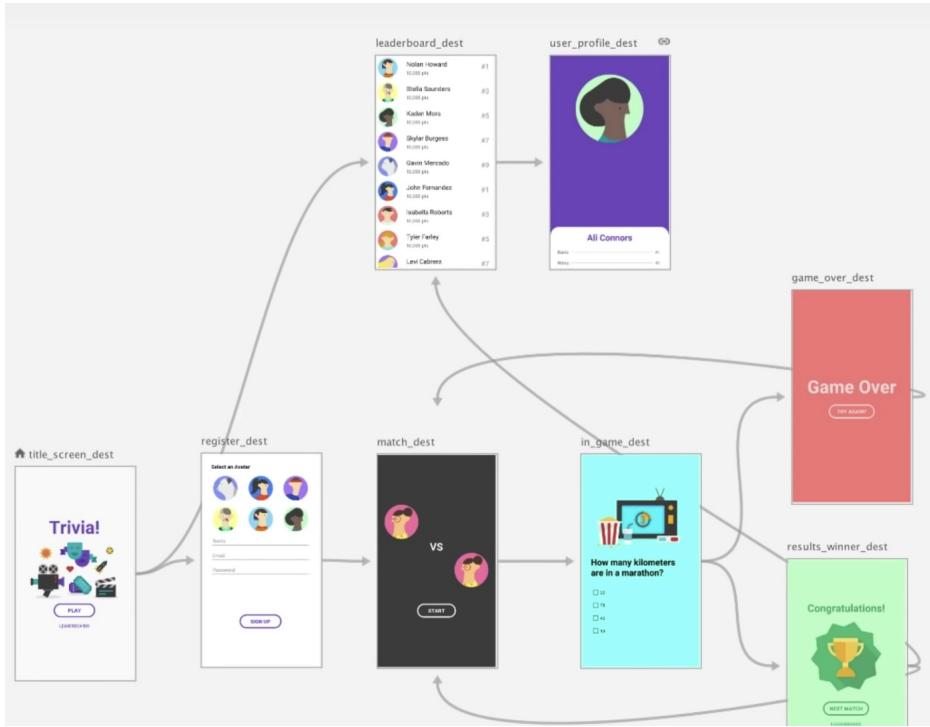


Figura 2.14: Ejemplo de un gráfico de navegación de Android Jetpack. [3]

2.2.5. Estilo – Material Design

Mediante el uso de los distintos componentes mencionados previamente es posible desarrollar aplicaciones completamente funcionales, pero funcionalidad no es lo único que se debe considerar cuando se trabaja en una aplicación. Cuando se logra obtener una aplicación funcional, se debe comenzar a mejorar el diseño visual de esta. Esto no es únicamente por razones estéticas, si no porque un buen diseño puede hacer la aplicación más accesible e intuitiva para el usuario. Para mejorar la calidad de las aplicaciones, Google creó un sistema de diseño llamado Material. Este sistema está basado en una serie de principios como tomar inspiración del mundo físico, realizar experiencias inmersivas a través de jerarquías establecidas por espacio, color imágenes y tipografía y utilizar el movimiento para dirigir la atención [4]. Material ofrece recursos, reglas de diseño y componentes para armar aplicaciones que permitan que los diseñadores puedan seguir sus principios y desarrollarlas de forma estética y cohesiva. Material suministra componentes interactivos (widgets) para poder implementar fácilmente sus guías de diseño en aplicaciones Android, iOS, web y Flutter. En la página de Material, cada componente tiene información sobre como usarlo, cuando usarlo, que principios son importantes para ese widget, que iconos utilizar y como posicionarlo. Por ejemplo, una caja de texto normal podría ser solo un bloque que cuando el usuario lo cliquea puede escribir; mientras que al utilizar un campo de texto de Material este podría ser más claro como cambiar de color en el borde para informar al usuario que puede escribir. Material también suministra iconos de distintos tipos que agrupa para que el diseñador utilice iconos

del mismo tipo en todo su programa. En el desarrollo de Android también se puede especificar tema de la aplicación. Esto va a definir los colores que se utilizan en la misma. Material también establece reglas sobre los colores que se deben elegir para el tema de la aplicación, tanto en modo claro como en modo oscuro.

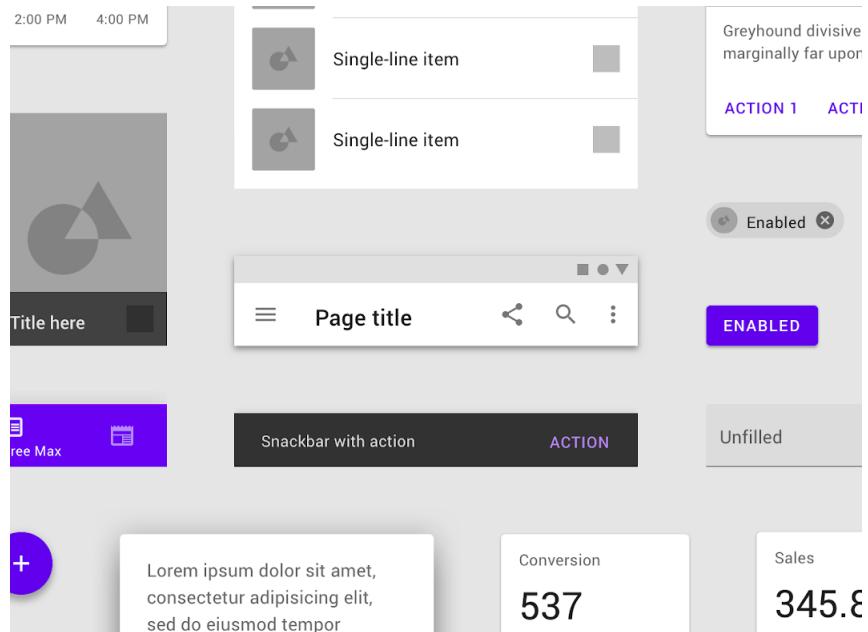


Figura 2.15: Ejemplo de algunos componentes que suministra Material. [4]

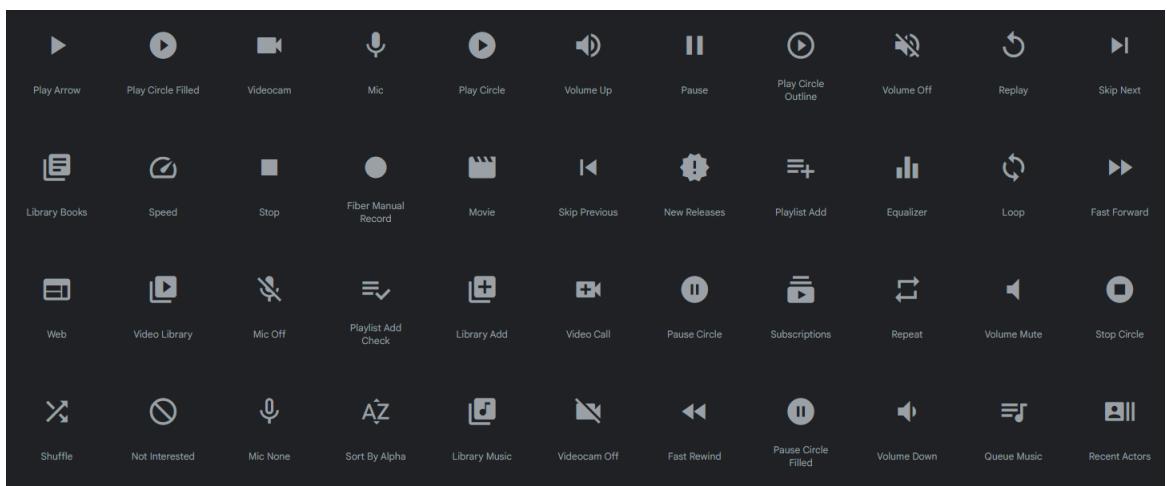


Figura 2.16: Algunos iconos con licencia libre de Material. [5]

Capítulo 3

DESARROLLO

3.1. Diseño y desarrollo de la aplicación

3.1.1. Diseño e ideas para la aplicación

El diseño de la aplicación se limitó al desarrollo de dinámicas para ayudar en tres aspectos que en [1], [9] y [10] se consideraban importantes. Estos son: el nombre de cada postura, el orden de la secuencia y el enfoque o drishti en cada posición. Por lo tanto, se buscó crear dinámicas que fomentaran la memorización de palabras o texto, la asociación de una palabra con cada postura y la relación de cada postura con la siguiente. Una característica de cada postura qué es importante y no se incluyó fue la respiración que se debe realizar en cada postura. Esta se podría desarrollar en versiones futuras de la aplicación, sabiendo la información necesaria para cada postura. Para idear dinámicas que ayudaran en este tipo de aprendizaje se investigó sobre aplicaciones similares en la Google Play Store. Una de las aplicaciones con cualidades similares a lo que se pensaba fue Drops. Esta es una aplicación utilizada para aprendizaje de lenguaje, que utiliza técnicas de memoria o mnemotécnicas, basadas en juegos que involucran relacionar dibujos de situaciones u objetos con sus palabras en el idioma que se está practicando.



Figura 3.1: Ejemplo de una dinámica de Drops para aprender persa. [6]

Otra aplicación que se estudió para idear cuáles dinámicas o funcionalidades debía tener la aplicación fue Anki. Anki es una aplicación de código abierto que utiliza un sistema de fichas o tarjetas virtuales para memorizar distintos temas. Esta se utiliza para estudiar: lenguajes, medicina, leyes, nombres, geografía, poemas y acordes de guitarra entre otros [21]. Esta aplicación permite a las personas crear y descargar mazos de cartas digitales con información de algún tema y mediante un algoritmo va diariamente enseñándole distintas cartas al usuario para que las practique.

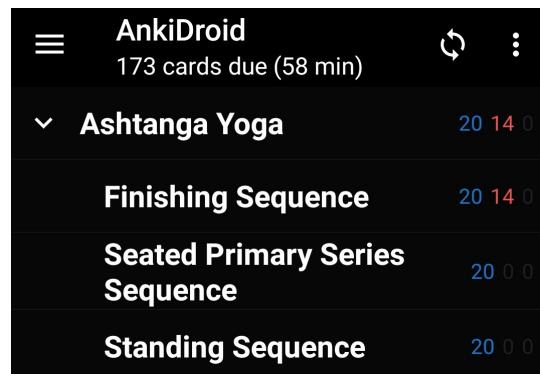


Figura 3.2: Captura del menú principal de Anki con un fichas de Ashtanga Yoga. Se tienen distintos grupos de ficha para cada secuencia.

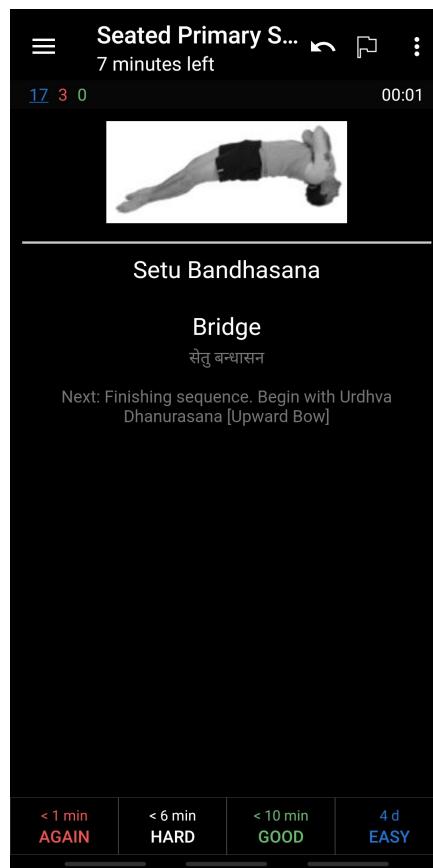


Figura 3.3: Captura de una ficha Anki para la secuencia sentado.

En la Figura 3.2 se observa la separación de cada secuencia en un grupo distinto de fichas dentro de la aplicación Anki. Luego, en la Figura 3.3 se tiene una ficha para una postura, donde está el nombre, una imagen de la postura, la siguiente postura y el nombre en sanscrito. Con base en las dinámicas de estas aplicaciones se procedió a diseñar un borrador de la estructura básica de la aplicación. Inicialmente la aplicación contaría con: Un menú principal para elegir qué se desea practicar, menús de selección de secuencia, preguntas sobre técnica (nombre y drishti) y preguntas sobre el orden de cada secuencia. Además se buscaba implementar un modo de fichas donde el usuario pudiese repasar todas las posturas sin necesidad de contestar preguntas. La estructura de las preguntas estaría basada en el tipo de preguntas visto en Drops, mientras que la información y la estructura de las fichas sería como Anki. Con base en las dinámicas planeadas originalmente, y una idea de como funcionaría la navegación por la aplicación se desarrolló el siguiente diagrama:

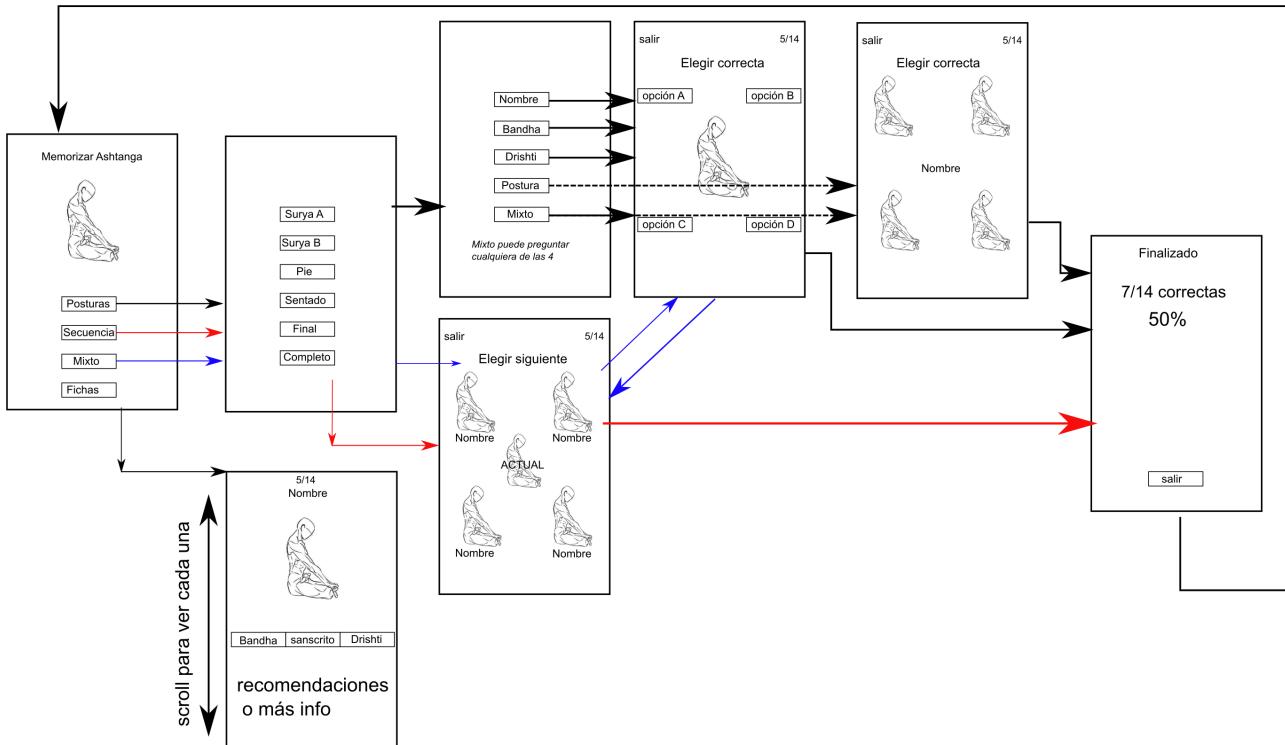


Figura 3.4: Ideas iniciales de dinámicas y flujo de la aplicación.

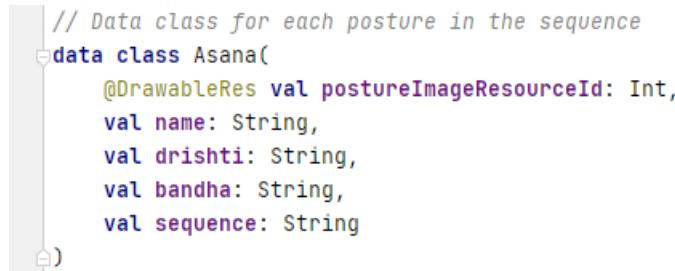
Con base en la Figura 3.4 se comenzó la programación y diseño de cada ventana de la aplicación. Además de los modos definidos aquí se iban a tener un modo de práctica que permitiera al usuario ir avanzando por cada postura en orden y con la información de cada una.

3.1.2. Datos e información en la aplicación

El programa de la aplicación iba a estar compuesto por múltiples archivos que contenían clases, fragmentos, vistas, datos, constantes, colores, temas, imágenes y archivos de diseño XML para las vistas y fragmentos. Para los datos, los archivos de mayor importancia son:

- Asana.kt: Data class para el tipo de dato Asana.
- DataSource.kt: Contiene los datos de cada postura y secuencia.

Asana.kt es una data class que define el tipo de dato Asana. Este nuevo tipo de dato está compuesto por el identificador de imagen de su postura, su nombre, su drishti, su bandha y la secuencia a la que pertenece.



```
// Data class for each posture in the sequence
data class Asana(
    @DrawableRes val postureImageResourceId: Int,
    val name: String,
    val drishti: String,
    val bandha: String,
    val sequence: String
)
```

Figura 3.5: Data class para el dato Asana.

Este dato es esencial para el funcionamiento de la aplicación, ya que este permite que se trabaje directamente con cada postura y luego acceder a datos dentro de ellas. Luego, con el tipo de dato definido, se implementó un objeto llamado DataSource. Este objeto estaría compuesto por listas que representan cada una de las secuencias de Asanas dentro de la serie 1. En este objeto se le asigna individualmente a cada postura sus información y se ordenan en listas. Este objeto es el que el resto de la aplicación utiliza para leer datos sobre las posturas.

```
object DataSource {

    val standingSequence: List<Asana> = listOf(
        Asana(
            R.drawable.padangusthsana,
            name: "Padangusthsana",
            drishti: "Nasagrai Drishti",
            bandha: "Mula Bhanda\nUddiyana Bhanda", sequence: "Standing Sequence"
        ),
        Asana(
            R.drawable.padahasthasana,
            name: "Pada Hasthasana",
            drishti: "Nasagrai Drishti",
            bandha: "Mula Bhanda\nUddiyana Bhanda", sequence: "Standing Sequence"
        )
    )
}
```

Figura 3.6: Algunas posturas de la secuencia de pie en el objeto DataSource.

Las imágenes para cada postura son imágenes vectoriales libres de regalías, descargadas de [22]. La información de las posturas se obtuvo de [1], [9], [10] y una guía suministrada por el profesor. Al haber posturas que no se encontraban en las imágenes descargadas, estas fueron elaboradas con la herramienta de diseño vectorial Inkscape.

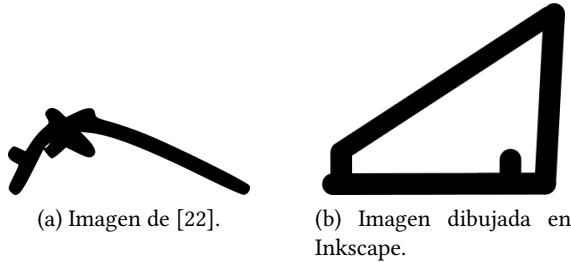


Figura 3.7: Comparación de imágenes descargadas y dibujadas.

3.1.3. Menús

Con la información para las posturas listas, se comenzó con el diseño de los menús de la aplicación. Para los menús y las preguntas se utilizó la clase de Fragment de Kotlin. Que permite un manejo fácil de ventanas que se reutilizan y la navegación entre ellos es simple mediante el uso de un NavController que permite navegar rápidamente entre pantallas. Todos los menús estaban contenidos dentro de una actividad principal, ya que los fragmentos deben estar contenidos dentro de una clase de tipo Activity. Las clases para cada menú son:

- MainFragment: Selección de modo. Preguntas sobre posturas, preguntas sobre secuencia, práctica, preguntas combinadas, práctica, repaso (fichas).
- SequenceMenuFragment: Selección de secuencia que se va a practicar.
- TechniqueMenuFragment: Selección de tipo de pregunta sobre postura.

Cada una de estas clases estaba definida en un archivo .kt, además de tener un archivo .XML que definiría su diseño.

Menú principal

Este fragmento contiene el menú principal del programa y es la pantalla de inicio. Aquí el usuario puede elegir que tipo de preguntas quieren contestar, si quieren practicar o si quieren ver las fichas informativas. Esta clase se encarga de seleccionar el modo que se va a practicar y llamar al navController para navegar a la siguiente pantalla con base en la selección del usuario. La selección de modo la realiza llamando a la función setMode() de la clase MainViewModel. El ViewModel es una clase de Kotlin utilizada para almacenar y manejar información de UI y mantenerla a pesar de cambios en los fragmentos, como navegar a otro fragmento, minimizar la aplicación y rotar el celular. En este ViewModel se tiene una variable para que el programa sepa en qué modo se encuentra. En caso de seleccionar el modo mixto, este menú también se encarga de asignar aleatoriamente cuál tipo de pregunta de técnica (nombre, drishti, postura) se va a realizar, ya que en modo mixto no se avanza al menú de selección de técnica como se observa en la Figura 3.4. El menú principal consiste en un título para la aplicación, una imagen de una postura y un linearLayout que contiene 5 botones para ir a distintas partes de la aplicación.

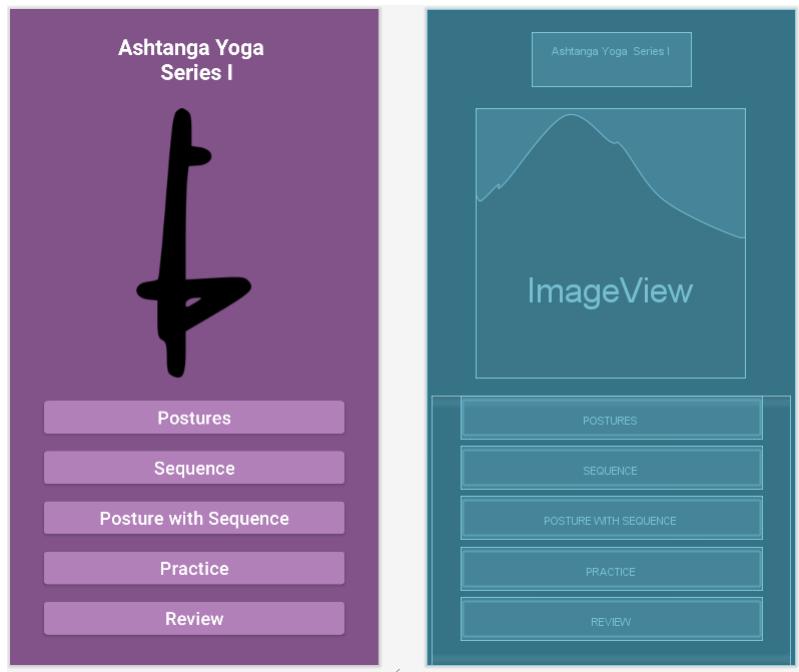


Figura 3.8: Diseño del menú principal.

Cada uno de los modos de tiene una función distinta:

- Posturas: El usuario puede probar su conocimiento sobre detalles de las posturas. Hay tres tipos de preguntas sobre: el nombre de la postura, el drishti de la postura, y la postura correspondiente al nombre. También hay un modo que combina estas preguntas. Las posturas están ordenadas en su orden de la secuencia para ayudar a recordar el orden.
- Secuencia: Al usuario se le muestra una postura y acompañada por 4 opciones. El usuario debe elegir la postura que sigue en la secuencia y así ir avanzando hasta terminar la secuencia.
- Postura y secuencia: En este modo combinado al usuario se le realiza una pregunta sobre la técnica de una postura y luego se le pregunta cuál es la postura siguiente, avanzando hasta terminar la secuencia.
- Práctica: Se avanza por cada postura de una secuencia libremente. Toda la información sobre la postura está disponible y se le permite al usuario retroceder.
- Repaso: Le enseña al usuario una lista con todas las posturas para poder rápidamente revisar datos sobre cada postura.

Menú de secuencia

Este menú permite al usuario elegir cuál de las distintas secuencias quiere practicar, en caso de que no quiera practicar la serie completa. Este menú es un linearLayout simple con un botón para cada secuencia, un botón para practicar la secuencia completa y un botón de regreso. Esta clase, funciona de forma similar a la del menú principal, pero llama a la función setSequence() del MainViewModel para asignar la secuencia elegida. Leugo la aplicación avanza al siguiente fragmento. Para saber a qué ventana avanzar el programa debe saber cuál modo se había seleccionado ya que en todos los modos se debe elegir la secuencia. Esta es la razón por la que se utilizó un ViewModel con variables que almacenaran estos datos a través de cambios en la aplicación.

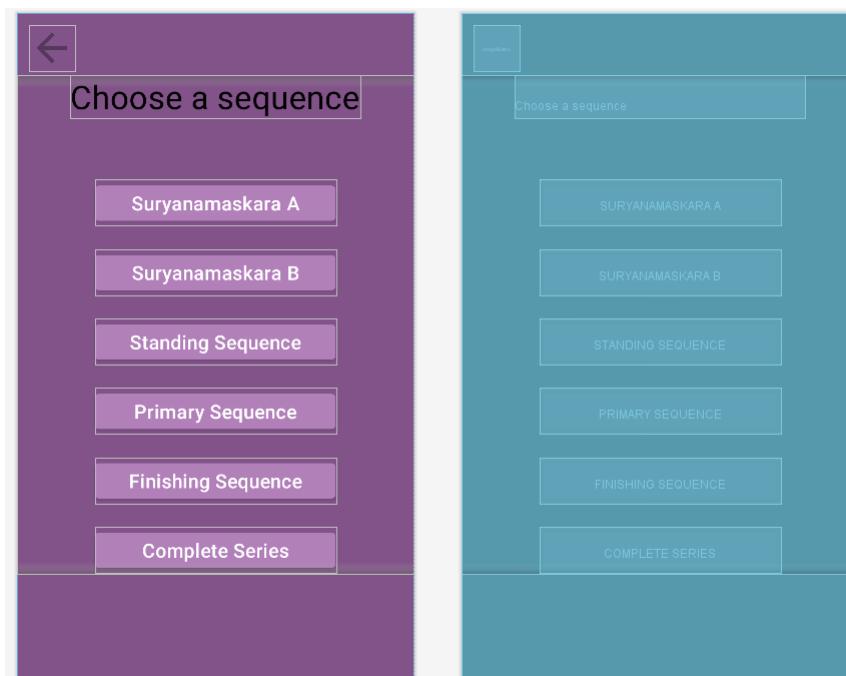


Figura 3.9: Diseño de menú de selección de secuencia.

Menú de técnica

El último menú es el de selección de técnica. Este solo se llama cuando se está en el modo de posturas. Este tiene distintos modos de preguntas sobre posturas. El primer tipo de pregunta es en el que al usuario se le muestra una imagen de la postura y debe contestar una pregunta sobre esta. Esta pregunta puede ser el nombre o del drishti. Originalmente también se iba a incluir la activación del bandha pero al observar la información de cada postura se observó que estos eran muy constantes a través de posturas consecutivas por lo que se decidió dejar este dato solo en el modo de práctica y fichas. El otro tipo de pregunta es una variación a la pregunta de nombre donde al usuario se le enseña un nombre y 4 imágenes y debe elegir la postura correspondiente. Finalmente, en el modo mixto al usuario se le hacen

preguntas aleatorias sobre las posturas, es decir le puede salir cualquier a de las opciones en el menú para cada pregunta.

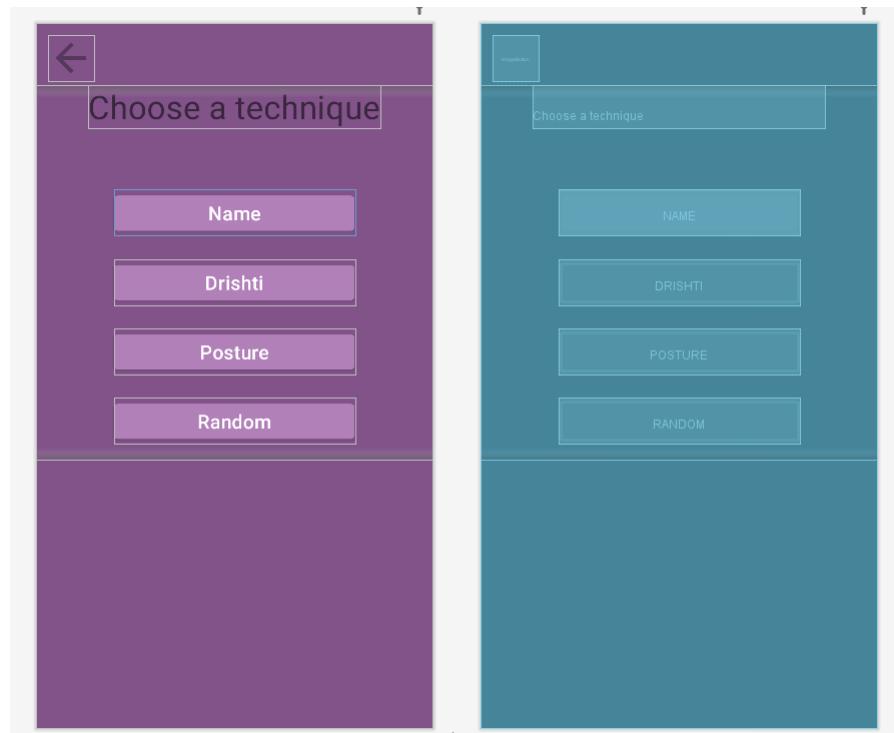


Figura 3.10: Diseño de menú de selección de técnicas.

3.1.4. Diseño de dinámicas

Las dinámicas que se diseñaron tenían la misma estructura que los menús; estaban compuestas por una clase de tipo Fragment y un archivo de diseño XML. La única excepción es la dinámica de fichas que es de tipo Activity, y no Fragment. Las clases para las dinámicas son:

- `TecnhiqueQuestionFragment`: Muestra 4 botones con texto y una imagen de una postura. Se utiliza para preguntas de nombre y drishti.
- `PostureQuestionFragment`: Muestra 4 posturas y un nombre. Se debe seleccionar la postura correspondiente al nombre.
- `SequenceQuestionFragment`: Muestra una postura y 4 opciones. Se debe seleccionar la postura siguiente.
- `PracticeFragment`: Muestra una postura con su nombre, drishti y bandha. Tiene botones para avanzar o retroceder en la secuencia.
- `ReviewActivity`: Muestra fichas con la información de las posturas. Al desplazarse verticalmente se avanza por toda la serie 1.

Pregunta de técnica

En esta dinámica se le presenta al usuario 4 opciones de texto y una imagen de una postura. Aquí se debe seleccionar la opción que corresponda a la postura. Al contestar se actualizan los botones y la imagen para tener una pregunta sobre la siguiente postura en la secuencia. Para llegar a este modo, en los menús se debe seleccionar *Postures*, luego se selecciona la secuencia que se desea practicar y finalmente se elige el tipo de pregunta que se va a hacer. Para llegar a esta ventana se debe seleccionar el modo *Name* o *Drishti*, o que este tipo de pregunta salga en el modo *Random*. Además se puede llegar a esta ventana cuando se selecciona el modo *Posture with Sequence* y la pregunta aleatoria sobre una postura corresponde a este tipo de dinámica. Las dinámicas deben tener algún tipo de retroalimentación al usuario para que esté informado sobre cuáles preguntas ha tenido correctas o incorrectas. En el caso de esta aplicación se optó por un sistema de puntos y retroalimentación visual. Cada respuesta correcta le agrega un punto la puntuación del usuario, y es representada mediante un breve cambio de color en el fondo de la pantalla antes de cambiar de pregunta. En el caso de una pregunta correcta el fondo se cambia a verde. Si el usuario tiene una pregunta incorrecta el fondo se cambia a rojo y su puntuación no cambia. Esta clase cuenta con funciones para verificar si la respuesta es correcta, revisar si el usuario se encuentra en la última pregunta de la secuencia, cargar otro tipo de pregunta en caso de que se esté en un modo combinado y una función para volver al menú principal y devolver todas las variables a sus estados por defecto para que el usuario pueda utilizar cualquier otro modo sin problemas.

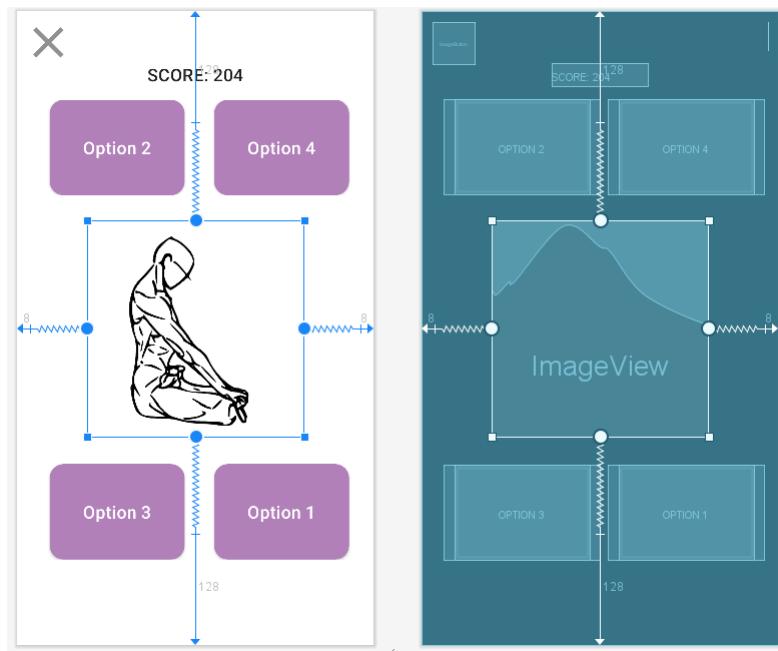


Figura 3.11: Diseño de preguntas sobre técnica. Android Studio cuenta con funciones para colocar texto e imágenes que solo el desarrollador puede ver mientras trabaja con el diseño.

Pregunta de postura

Este modo funciona igual que el modo de técnica, pero en lugar de usar botones, para las opciones se utiliza un ImageButton para colocar 4 imágenes de posturas como opciones, mientras que en el centro se coloca texto con el nombre de la postura. Este modo cuenta con las misma funciones que la pregunta de técnica. Para verificar el resultado compara el identificador de la imagen cargada en el botón seleccionado con el identificador de la postura cuyo nombre se está seleccionando. Por razones como está es que definir un tipo de dato Asana que contenga toda la información de una postura es importante. Ya que permite relacionar el texto enseñado como opción, con la imagen de la postura. Se compara el identificador de imagen ya que es posible que en las opciones esté una postura igual pero de otra secuencia por lo que el objeto no es exactamente igual, pero se utiliza la misma imagen por lo que el resultado igual va a ser correcto.

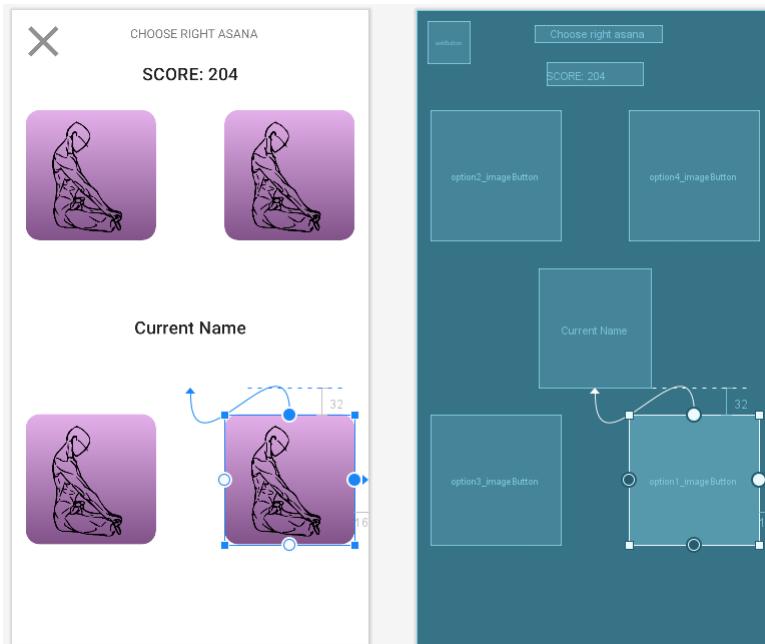


Figura 3.12: Diseño de preguntas sobre postura.

Pregunta de secuencia

La configuración de este modo es similar a la del modo de pregunta de postura. Consiste en 4 botones con imagen, rodeando a una imagen de un postura en el centro. El usuario debe elegir la postura que corresponde al siguiente paso en la secuencia. En este modo se le enseña al usuario el nombre de todas las opciones para que pueda practicarlos mientras hace la secuencia. En caso de que se esté utilizando el modo *Posture & Sequence* los nombres de la postura se ocultan, ya que podrían dar la respuesta de la siguiente pregunta. En este modo se debe cargar tanto la Asana actual como la siguiente para revisar los resultados. Aquí el ViewModel guarda el valor de la siguiente Asana con base en la actual y compara

la imagen del botón seleccionado con la imagen de la siguiente Asana para verificar si la opción es correcta. Al cargar las opciones el programa se debe asegurar que no haya opciones repetidas y que la opción correcta si esté entre las opciones. La selección de las imágenes para las posturas es realizada por el ViewModel.

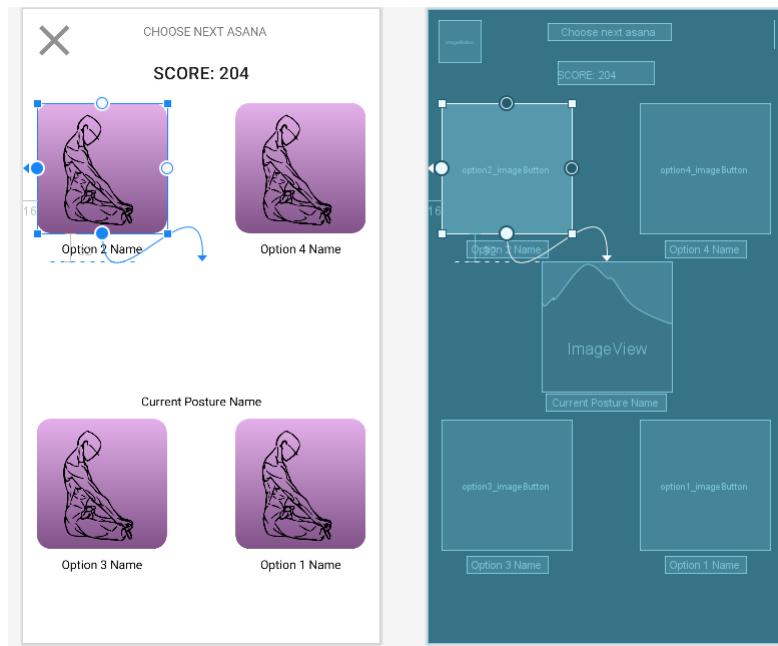


Figura 3.13: Diseño de preguntas sobre secuencia.

Modo de práctica

Debido a que el Ashtanga yoga no es únicamente sobre memorización, también se buscó implementar una dinámica que pudiese ser usada por alguien que estuviera practicando las posturas. En esta dinámica se le muestra en toda la pantalla al usuario una postura con información sobre esta. Al presionar a la derecha de la pantalla se avanza a la siguiente postura, mientras que presionar a la izquierda lo devuelve a la postura anterior. En caso de estar en la primera postura se retorna al menú de selección de secuencia, y al estar en la última se le advierte al usuario si desea terminar la secuencia. Este comportamiento al final es para evitar que se presione accidentalmente el botón de siguiente en la última postura y el usuario vuelva al menú principal, haciendo que tenga que pasar por toda la secuencia para volver a la postura en la que estaba. También se le muestra al usuario su progreso por la secuencia, y en una versión futura se buscaría implementar una forma de mostrar al usuario el tiempo que lleva en la secuencia.

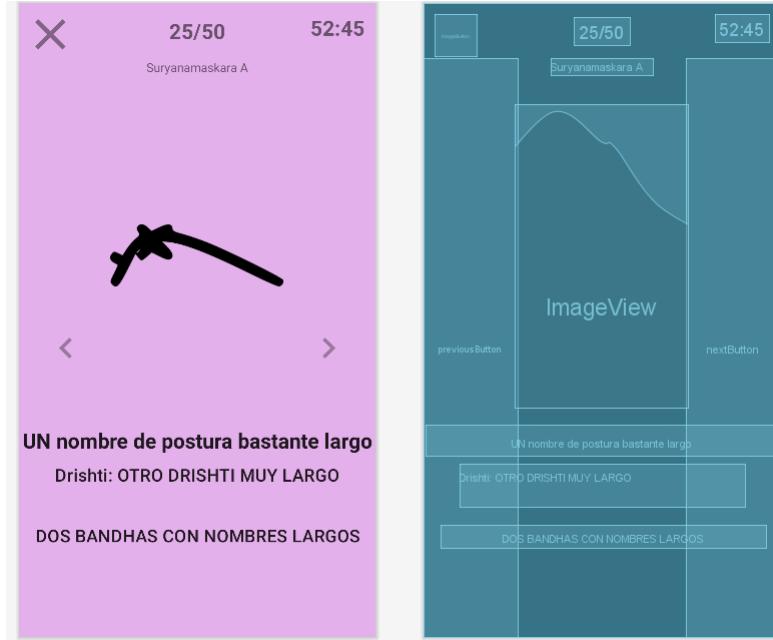


Figura 3.14: Diseño de modo de práctica.

En la Figura 3.14 se observa el uso de texto de prueba en el diseño de la aplicación. Aquí se coloca texto e imágenes solo visibles al desarrollador puede ver para poder visualizas como se vería la aplicación. Esto es útil cuando el contenido de la ventana varía constantemente.

Modo de fichas

El modo de fichas se desarrolló con una estructura distinta a los otros modos. Aquí, se buscaba poder mostrar toda la información simultáneamente, por lo que no era necesario estar constantemente refrescando datos en la pantalla. Por lo tanto, se utilizó una Activity en lugar de un Fragment. En este modo se iba a tener un archivo de diseño XML de tipo MaterialCardView. Esta carta contiene datos de una postura, y mediante una clase llamada CardAdapter, se conectan estas fichas a un archivo de diseño review_activity.xml que va a contener múltiples cartas a la vez. Para poder colocar múltiples cartas en la pantalla y desplazarse a través de ellas se utiliza un RecyclerView. La clase CardAdapter hereda de RecyclerView y permite utilizar el mismo espacio en la pantalla para mostrar cosas distintas conforme elementos salen de la vista. Por lo tanto, permite mostrar cartas y al desplazarse hacia abajo (*scroll down*) conforme cartas salen de la pantalla CardAdapter se encarga de cargar nuevas cartas.

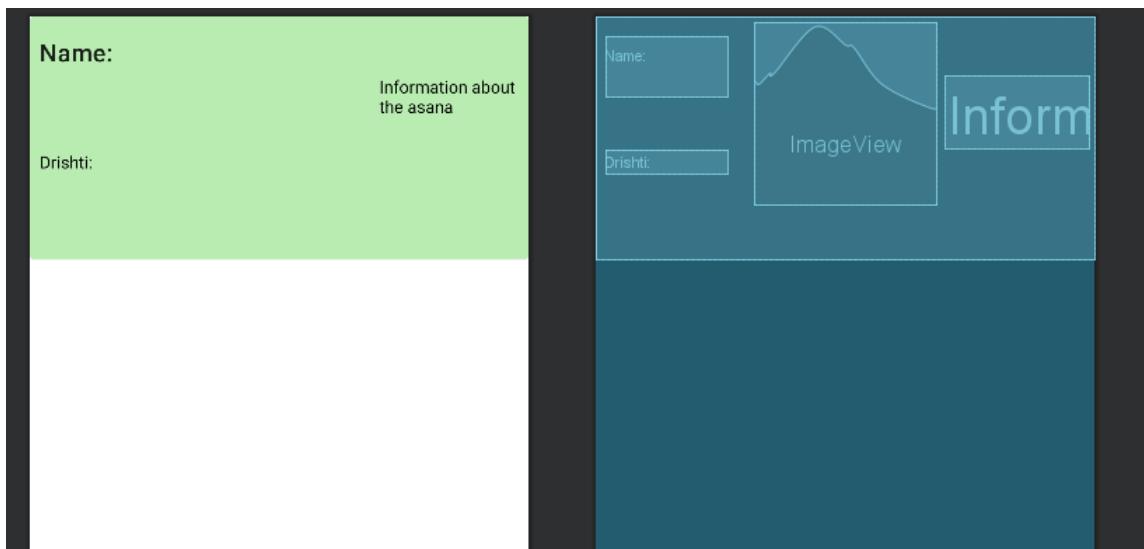


Figura 3.15: Diseño de la carta en asana_card.xml.

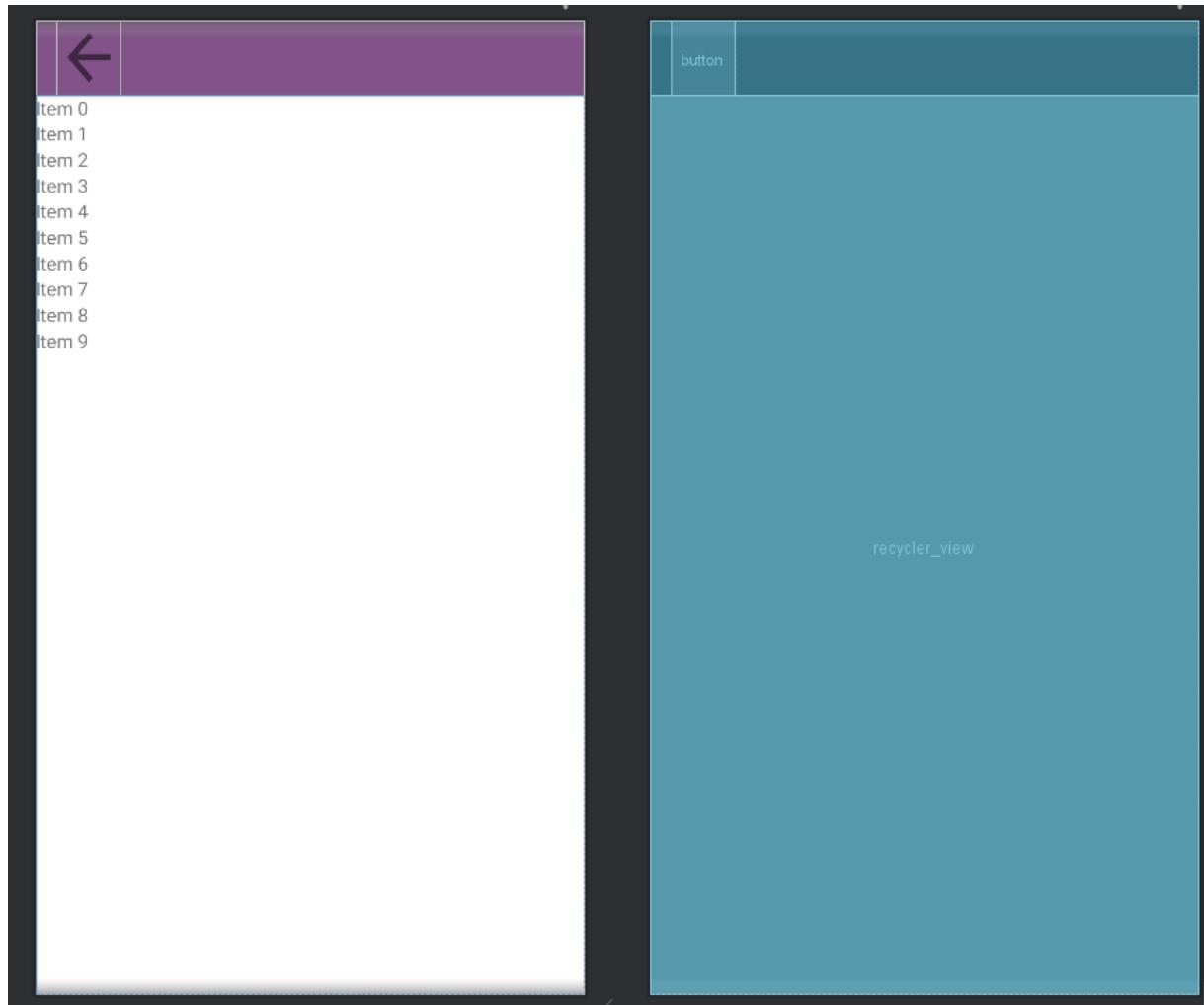


Figura 3.16: Diseño de la actividad de repaso review_activity.xml.

En las figuras 3.15 y 3.16 se observa que, a diferencia de diseños anteriores, no es tan claro como se va a ver el diseño de la ventana. Esto se debe a que la vista previa de Android Studio no toma en cuenta el trabajo realizado por la clase CardAdapter para colocar múltiples fichas como la Figura 3.15 dentro de la actividad de la Figura 3.16. En el funcionamiento de la aplicación el adaptador se va a encargar de llenar la pantalla con la cantidad de fichas que pueda colocar dentro.

3.1.5. Main view model

La clase de Main View Model es la que se encarga de manejar todos los datos con los que funciona la aplicación, además de los que se enseñan en el UI. Todos los fragmentos del programa tienen acceso a esta clase, y las variables de esta se pueden ligar a elementos en el UI a través del archivo de diseño xml. Esto se logra a través de variables LiveData. Ejemplos de variables que están siendo utilizadas como LiveData en el programa son: el modo principal, la secuencia elegida, la técnica elegida, la longitud de la secuencia, el contador de posturas, el puntaje y la postura actual. Estos son datos que se deben mostrar al usuario o controlan aspectos sobre el contenido de los botones o imágenes. Esta conexión entre los datos del view model y los datos dentro del archivo xml se le llama data binding.

```
// Current posture
private val _asana = MutableLiveData<Asana>()
var asana = _asana
```

Figura 3.17: Declaración de una variable de Asana Live Data en el Main View Model.



```
<ImageView
    android:id="@+id/Current_posture_imageView"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="128dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="128dp"
    app:imageResource="@{viewModel.asana.postureImageResourceId}"
    app:layout_constraintBottom_toBottomOf="@+id/SeqQuestConstraintLayout"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
```

Figura 3.18: En el archivo de diseño se coloca el valor de esa variable para mostrar al usuario.

En la Figura 3.17 se declara una variable de MutableLiveData de tipo Asana dentro del view model. Aquí se utiliza una propiedad llamada *backing property*. Esta se utiliza cuando se quiere tener una variable privada que otras clases no puedan modificar, pero si puedan leer. Por convención se le asigna un “_” antes del nombre a la variable privada y luego la pública se le asigna el valor. La LiveData es bastante útil ya que permite simplemente asignar la variable a un elemento de la interfaz y los cambios en esta variable se actualizan automáticamente. En la Figura 3.18 se le asigna el identificador de la postura actual a un ImageView, permitiendo que esta siempre muestre la imagen de la postura actual. En la aplicación se utilizan variables en el view model para representar los elementos del UI y así poder trabajar por separado el manejo de los datos y el diseño de la interfaz. El data binding también permite el llamado de funciones desde el archivo de diseño. Lo que permite simplificar el proceso de asignar acciones a los botones cuando son presionados. En la Figura 3.19 se observa como desde el archivo de diseño se llama a una función de la clase FragmentSequenceQuestion y se le pasa un argumento que es

una variable del view model. Esto tipo de conexión entre el archivo de diseño, la clase del fragmento y el view model se utilizó a través de todo el programa para poder actualizar la información mostrada al usuario y que el programa reaccione a lo que realiza el usuario.

```
<ImageButton
    android:id="@+id/option1_imageButton"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:adjustViewBounds="true"
    android:background="@drawable/roundcorners"
    android:onClick="@{() -> fragmentSequenceQuestion.checkAnswer(viewModel.asanaOptions.get(0).postureImageResourceId)}"
    android:clickable="@{viewModel.buttons}"
    android:padding="2dp"
    android:scaleType="fitCenter"
    android:textAppearance="?attr/textAppearanceHeadline6"
    app:imageResource="@{viewModel.asanaOptions.get(0).postureImageResourceId}"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Current_posture_imageView"
    tools:ignore="SpeakableTextPresentCheck,ContentDescription,ImageContrastCheck"
    tools:srcCompat="@drawable/asanad" />
```

Figura 3.19: Desde el archivo de diseño se llama a una función de una clase y se le pasa un argumento del main view model.

El main view model cuenta con funciones esenciales para el funcionamiento del programa. Debido a la backing property, cuenta con funciones para asignar valores a las variables como el modo o la secuencia seleccionada. También cuenta con funciones que generan las opciones que se van a mostrar en los botones. Para esto, dependiendo del modo seleccionado carga 3 asanas aleatorias y revisa que no contengan la respuesta a la pregunta que se está realizando. Luego genera una lista con las 3 opciones incorrectas y la correcta. Esta lista es la variable asanaOptions que se llama en la Figura 3.19. Aquí cada botón corresponde a un índice en la lista de posibles resultados. Cuando se presiona el botón se llama a una función que compara el valor de la lista de opciones con el valor correcto. El view model también cuenta con funciones que avanzan y retroceden entre posturas y cuentan el puntaje del usuario. En caso de que el usuario termine la secuencia o desee salir, hay una función reset() que retorna todas las variables utilizadas a su valor original.

```
fun reset() {
    _score.value = 0
    _posCounter.value = 0
    _seqLength.value = 0
    _questionPosition.value = 1
    _sequencePosition.value = 1
    _sequenceIndex = 5
    _randomT.value = false
    _combined.value = false
    bothAnswered = false
    finalScoreVar = "0"
    _buttons.value = true
}
```

Figura 3.20: Al terminar la secuencia reset reinicia el puntaje, la posición, la secuencia y modo seleccionado y otras variables utilizadas en distintas preguntas.

3.1.6. Navegación

La navegación a través del programa se realizó mediante el uso de la biblioteca Jetpack de Android. Esta permite tener un archivo xml que define funciones para las distintas transiciones entre fragmentos. Luego desde el código se pueden llamar estas funciones para navegar entre los elementos del programa. Estas conexiones entre los fragmentos se definieron en una archivo llamado navgraph.xml. Este se puede modificar cambiando el código que lo compone o mediante una interfaz visual de Android Studio. En la Figura 3.21 se observan las distintas conexiones entre cada fragmento del programa, aquí cada flecha definía una función que se podía llamar dentro del programa para cambiar de fragmento, como se observa en la Figura 3.22

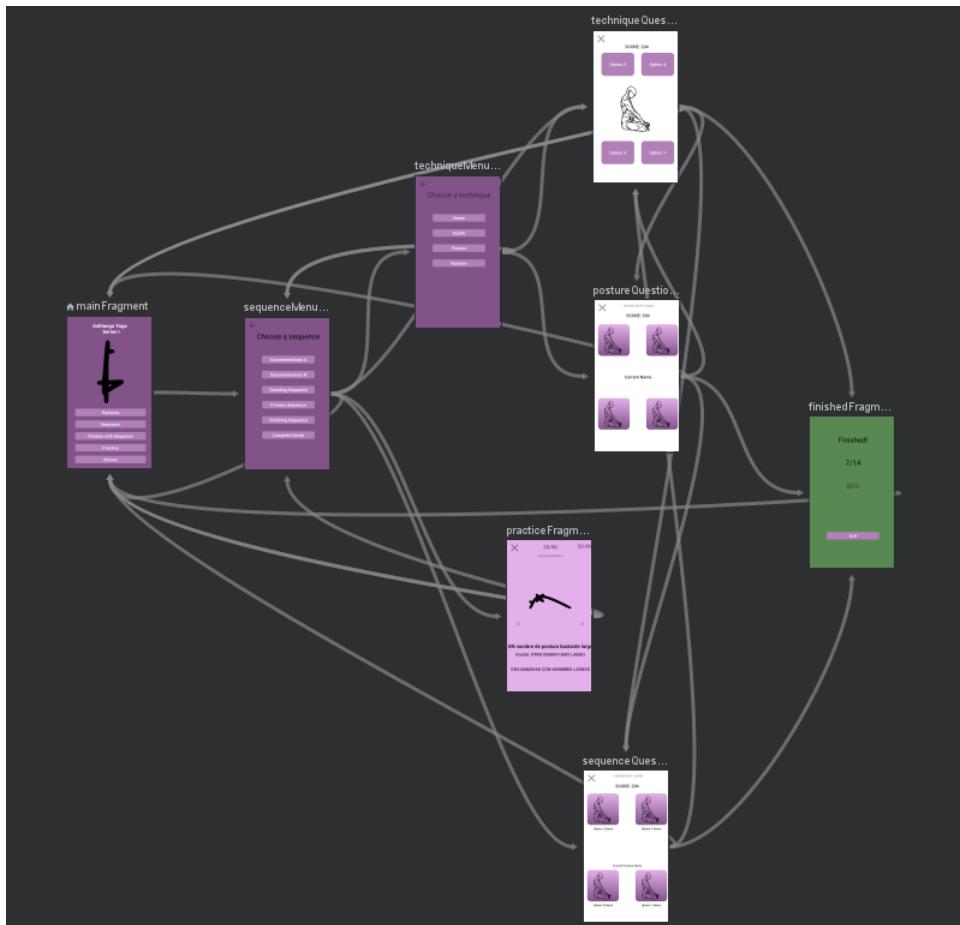


Figura 3.21: Gráfico de navegación del programa.

```
private fun selectNextScreen(){
    when (sharedViewModel.mode.value.toString()){
        sharedViewModel.mainMenu[0] -> findNavController().navigate(R.id.action_sequenceMenuFragment2_to_techniqueMenuFragment)
        sharedViewModel.mainMenu[1] -> findNavController().navigate(R.id.action_sequenceMenuFragment2_to_sequenceQuestionFragment)
        sharedViewModel.mainMenu[2] -> findNavController().navigate(R.id.action_sequenceMenuFragment2_to_techniqueQuestionFragment)
        sharedViewModel.mainMenu[4] -> findNavController().navigate(R.id.action_sequenceMenuFragment2_to_practiceFragment)
        else -> exit()
    }
}
```

Figura 3.22: En esta función del menú de secuencias dependiendo del modo que se había seleccionado en el menú principal se elige a cuál fragmento avanzar y se llama al controlador de navegación para realizar la acción.

Una de las principales razones para utilizar fragmentos fue para simplificar la navegación entre las pantallas. La pantalla de fichas no es un fragmento, si no que es una actividad por lo que no se conectaba mediante el gráfico de la Figura 3.21. Para navegar entre actividades se utiliza un objeto llamado Intent. Este solicita una acción de otro componente de la aplicación, por ejemplo una actividad. Mediante el Intent es posible llamar a una actividad desde un fragmento y hacer que la actividad comience, cambiando la pantalla. En la Figura 3.23 se utiliza un intent para comenzar la actividad que contiene las fichas desde el menú principal

```
// Review
3 -> {
    sharedViewModel.setMode(3)
    val intent = Intent(context, ReviewActivity::class.java)
    context?.startActivity(intent)
}
```

Figura 3.23: Al seleccionar el botón de Review en el menú principal se guarda el modo en el view model y se abre la actividad de fichas mediante un intent.

3.1.7. Temas y estilo

Para la selección de colores de la aplicación se utilizó la herramienta de selección Color Tool de de Material Design [4]. Esta permitía elegir un color y mostraba colores complementarios y asignaba valores a otros colores que se utilizarían dentro de la aplicación.

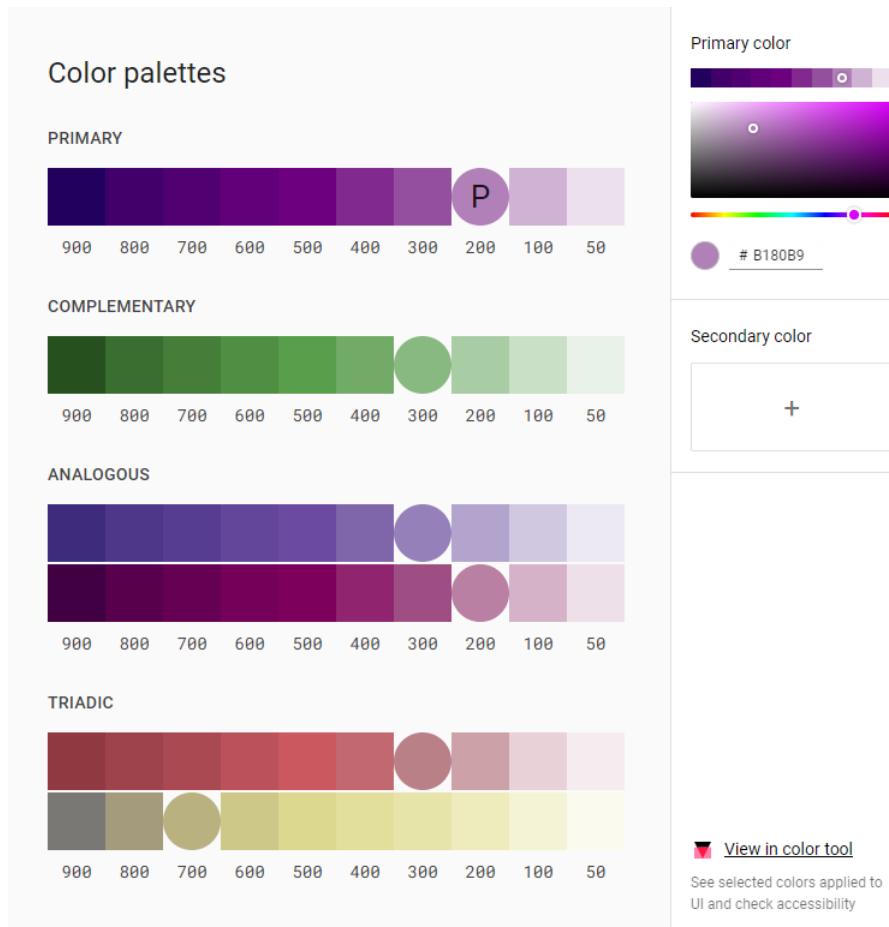


Figura 3.24: Para seleccionar los colores se utilizó el selector de paletas de color de material [4]

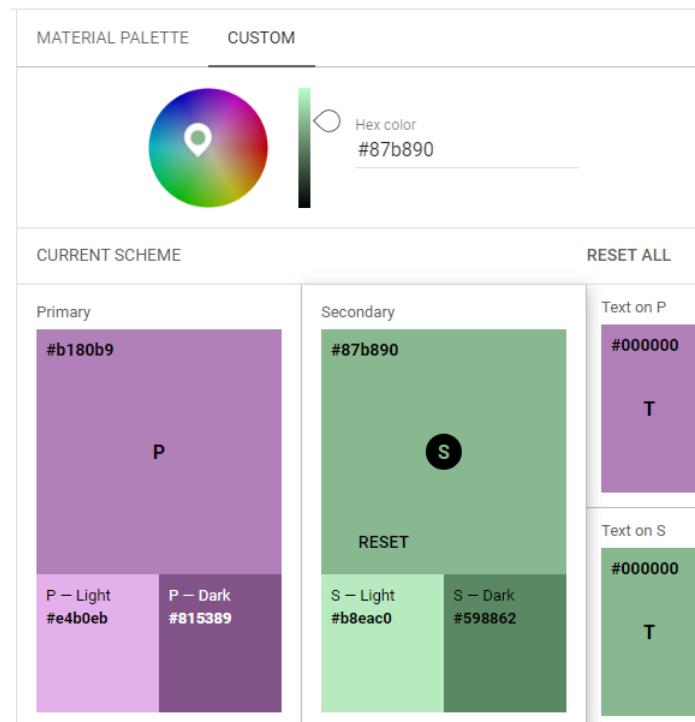


Figura 3.25: Las variaciones de los colores se obtuvieron mediante la Color Tool de Material Design [4].

Luego de seleccionar los colores, estos se definen en un archivo xml que contiene los colores utilizados por la aplicación, esto facilita el acceso a los mismos colores desde distintas clases.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="purple_200">#FFBB86FC</color>
4      <color name="purple_500">#FF6200EE</color>
5      <color name="purple_700">#FF3700B3</color>
6      <color name="teal_200">#FF03DAC5</color>
7      <color name="teal_700">#FF018786</color>
8      <color name="black">#FF000000</color>
9      <color name="white">#FFFFFF</color>
10
11     <color name="purple_p">#b180b9</color>
12     <color name="purple_p_light">#e4b0eb</color>
13     <color name="purple_p_dark">#815389</color>
14     <color name="purple_p_triad">#9580b9</color>
15     <color name="red_p_triad">#9e434b</color>
16     <color name="wrong_red">#e53935</color>
17     <color name="right_green">#9ccc65</color>
18
19     <color name="green_s">#87b890</color>
20     <color name="green_s_light">#b8ecb0</color>
21     <color name="green_s_dark">#588953</color>
22 </resources>

```

Figura 3.26: En el archivo colors.xml se definen los colores que va a utilizar la aplicación.

```

1  <resources xmlns:tools="http://schemas.android.com/tools">
2      <!-- Base application theme. -->
3      <style name="Theme.Ashtanga1" parent="Theme.MaterialComponents.DayNight.NoActionBar">
4          <!-- Primary brand color. -->
5          <item name="colorPrimary">@color/purple_p</item>
6          <item name="colorPrimaryVariant">@color/purple_p_dark</item>
7          <item name="colorOnPrimary">@color/white</item>
8          <item name="colorOnBackground">@color/purple_p_triad</item>
9          <item name="backgroundColor">@color/purple_p_dark</item>
10         <item name="colorError">@color/red_p_triad</item>
11         <!-- Secondary brand color. -->
12         <item name="colorSecondary">@color/green_s</item>
13         <item name="colorSecondaryVariant">@color/green_s_dark</item>
14         <item name="colorOnSecondary">@color/black</item>
15         <!-- Status bar color. -->
16         <item name="android:statusBarColor" tools:targetApi="l">?attr/colorOnBackground</item>
17         <!-- Customize your theme here. -->
18     </style>
19
20     <style name="Theme.Ashtanga1.NoActionBar">
21         <item name="windowActionBar">false</item>
22         <item name="windowNoTitle">true</item>
23     </style>
24
25     <style name="Theme.Ashtanga1.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
26
27     <style name="Theme.Ashtanga1.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />

```

Figura 3.27: theme.xml define como se van a aplicar los colores a toda la aplicación.

La aplicación está diseñada para soportar desde el API 25 de Android, por lo que tiene un archivo

de temas compatible con el modo oscuro. En este se utilizan variaciones de los colores primarios del tema que no sean tan brillantes, para que sea más cómodo de utilizar en ambientes oscuros. En el caso de esta aplicación no se ha implementado el modo oscuro, ya que las imágenes utilizadas tendrían que ser modificadas para tener una variación de color que tenga un buen contraste con las variaciones de los colores utilizados en modo oscuro. Para desactivar el modo oscuro se utilizó el mismo archivo del tema claro en el archivo del tema oscuro, haciendo que los colores no cambien.

Los iconos para cerrar las ventanas (Figura 3.11) y las flechas para avanzar y retroceder (Figura 3.14) son iconos de material que vienen incluidos en Android Studio. Los botones redondeados utilizados para las preguntas con imágenes como opciones, se obtuvieron de [23] y estan definidos mediante un archivo xml. El ícono de la aplicación es una imagen de uso libre obtenida [7]. Se eligió este ícono por su enfoque en las posturas, ya que es lo que esta aplicación busca ayudar a aprender. Además esta imagen era un svg de alta calidad y de uso libre.

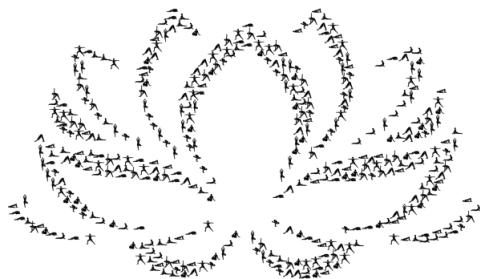


Figura 3.28: Imagen original utilizada para el ícono. Obtenida de [7]



Figura 3.29: El ícono de la aplicación es una flor de loto compuesta por distintas posturas de yoga. La resolución es varía con el dispositivo utilizado.

3.2. Análisis de resultados

La verificación del funcionamiento de la aplicación consistió en realizar las diferentes dinámicas que esta tiene con distintas secuencias y asegurarse que los datos correctos estuvieran en la pantalla, la puntuación funcionara correctamente y que no hubiese problemas de navegación o que la aplicación dejara de funcionar. Las pruebas fueron realizadas en el emulador de Android Studio en un celular Android Pixel 3a XL con el API 29.

3.2.1. Modo 1: Preguntas de técnica

El modo 1 consiste en preguntas sobre el nombre o el drishti de una postura. Para verificar el funcionamiento se probaron distintas secuencias en cada uno de los distintos tipos de pregunta de este modo: nombre, drishti, postura y aleatorio. Para llegar a la selección de tipo de pregunta se selecciona *Postures* en el menú principal, una secuencia en el menú de secuencias y finalmente el tipo de pregunta. Este proceso de navegación se observa en la Figura 3.30

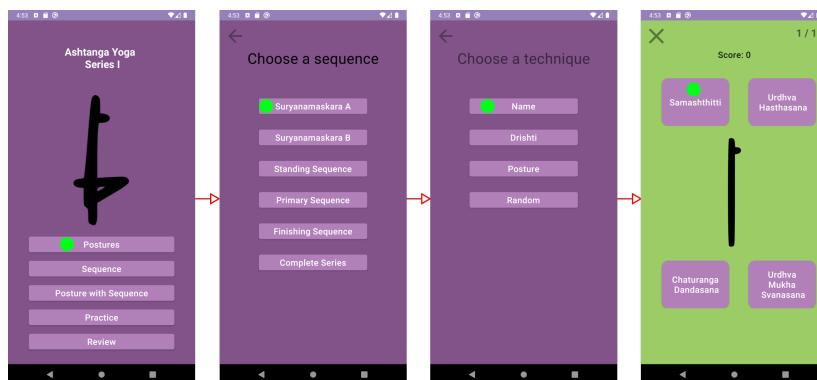


Figura 3.30: Navegación desde el menú principal a una pregunta sobre el nombre de una postura.

Se logró navegar exitosamente desde los menús hasta todos los distintos modos diseñados. Un ejemplo de esta navegación es la Figura 3.30. Además los botones de retroceso en los menús funcionaban correctamente y al devolverse se deshacían las selecciones previas para el correcto funcionamiento de la aplicación. De la misma manera el botón de salir (X) en las preguntas funciona adecuadamente, retornando al usuario al menú principal y reiniciando todas las variables de la aplicación para que no interfieran con los demás modos que vaya a usar el usuario.

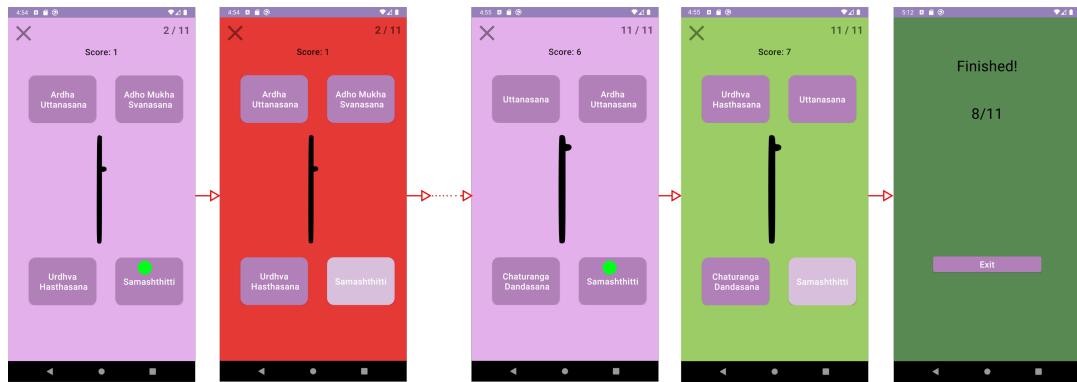


Figura 3.31: Ejemplo de interacción del usuario con preguntas sobre el nombre de posturas en la secuencia Suryanamaskara A. Los puntos verdes representan clics del usuario.

En la Figura 3.31 se observa como las preguntas sobre el nombre de postura funcionan. Aquí se le enseña al usuario una postura y debe elegir el nombre correspondiente. Al elegir incorrectamente la pantalla se pone roja antes de actualizar los puntos y avanzar a la siguiente pregunta. Para lograr el cambio de fondo cuando el usuario elige una respuesta se utilizó la función de Kotlin `postDelayed()` que permite hacer un retraso en la aplicación sin hacer que esta deje de responder. Debido a que la aplicación sigue respondiendo, también se agregó que los botones se desactivaran mientras ocurre esta transición, ya que originalmente era posible presionar los botones mientras esto ocurría, haciendo que el contador de postura avanzara sin que las posturas avanzaran haciendo que ocurriera un error de índice al leer las figuras. La función `checkAnswer()` maneja estos cambios al revisar la respuesta, variaciones de esta función se encuentran en los demás fragmentos de preguntas. La función `checkAnswer()` compara el identificador de imagen de la postura seleccionada en el botón con la que se está preguntando, aunque se esté preguntando sobre el nombre o drishti. Esto es útil ya que hay posturas repetidas en distintas secuencias pero con pequeñas variaciones en sus nombres. Pero al comparar imágenes, es de alta importancia que el programa coloque la postura correcta en las opciones elegibles y no solo una postura con el mismo nombre o drishti. Por lo que el programa, al elegir estas opciones se asegura que la respuesta correcta esté en la lista y que las demás opciones no compartan nombre y drishti con la respuesta correcta. Al cargar las opciones también solo se eligen asanas presentes en la secuencia que se está preguntando para aumentar la dificultad de las preguntas. La dificultad disminuiría ya que, por ejemplo, se podría mostrar una postura de pie cuando el usuario está practicando la secuencia sentado.

```

fun checkAnswer(selection: Asana) {
    val view = view?.rootView
    val handle = Handler()
    sharedViewModel.enableButtons( enable: false)
    Log.d( tag: "Test", msg: "${selection}, ${sharedViewModel.asana.value}")
    // Comparar nombres deberia funcionar con Drishti y con imagenes
    if (selection.postureImageResourceId == (sharedViewModel.asana.value?.postureImageResourceId))
        view?.setBackgroundResource(rightColor)
    view?.postDelayed({ view.setBackgroundResource(defBackg) }, delayTime)
    handle.postDelayed({
        sharedViewModel.correctAnswerTechnique()
        checkLast()
        sharedViewModel.enableButtons( enable: true)
    }, delayTime)
} else {
    view?.setBackgroundResource(wrongColor)
    view?.postDelayed({ view.setBackgroundResource(defBackg) }, delayTime)
    handle.postDelayed({
        sharedViewModel.incorrectAnswerTechnique()
        checkLast()
        sharedViewModel.enableButtons( enable: true)
    }, delayTime)
}
}

```

Figura 3.32: Función checkAnswer() se encarga de revisar la respuesta, cambiar el fondo de pantalla y desactivar temporalmente los botones.

Al seleccionar el modo de drishtis, este también funcionó correctamente.

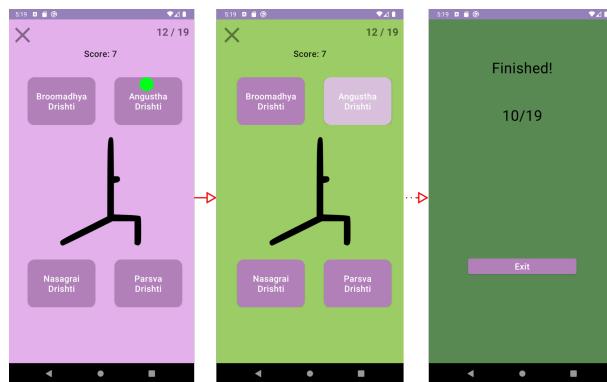


Figura 3.33: El modo de drishtis funciona de manera similar al de nombres. Aquí se utiliza la secuencia Suryanamaskara B.

En la Figura 3.33 se observa un ejemplo de una pregunta sobre drishti. Aquí el usuario va por la pregunta 12 de 19 y ha tenido 7 respuestas correctas. Al seleccionar correctamente el fondo cambia a color verde. Aquí se observa que el puntaje no aumenta aunque el usuario contestó correctamente, esto se debe a que los puntos corresponden a una variable de LiveData que no cambia hasta después de que se acaba el retraso para cambiar el fondo, por lo que la puntuación se va a actualizar hasta que se

muestre la pregunta siguiente. En la Figura 3.33 el usuario sigue contestando preguntas hasta llegar al menú de finalización donde se le muestra que tuvo 10 de 19 posturas correctas.

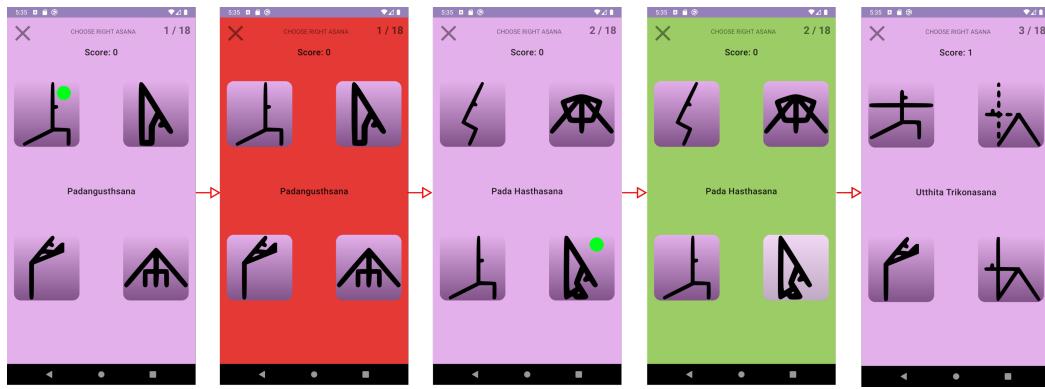


Figura 3.34: Ejemplo de modo de preguntas sobre el nombre pero invertido, donde se muestran 4 posturas y un nombre. Aquí se utiliza la secuencia de pie.

En el modo de postura se utilizan botones distintos, ya que estos tienen que contener imágenes. En la Figura 3.34 se tiene un ejemplo del usuario realizando preguntas sobre la postura. Al ser la secuencia de pie, se observa que se tienen 18 preguntas distintas. Al seleccionar incorrectamente la primera, la pantalla cambia a rojo antes de avanzar a la siguiente pregunta. Cuando la tiene correcta la pantalla se pone verde. Aquí se puede observar como luego de tener el resultado correcto, los puntos se actualizan hasta la siguiente imagen. Durante el uso de la aplicación estos cambios son bastante rápidos por lo que no es tan problemático este retraso en los puntos.

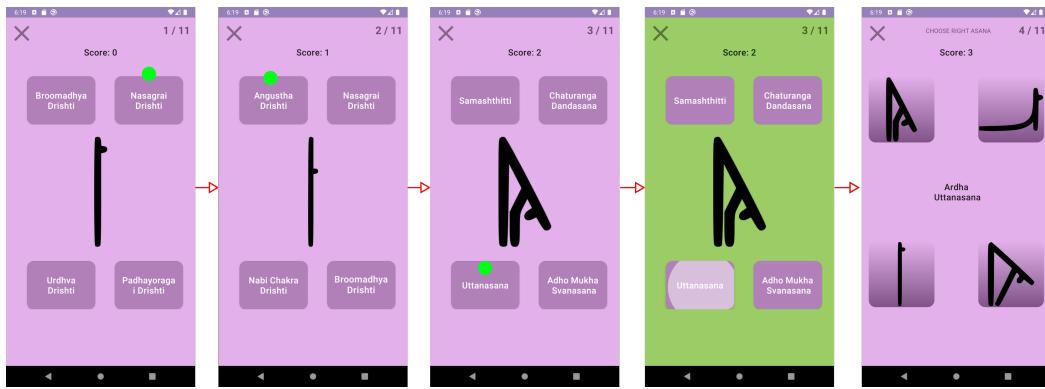


Figura 3.35: En el modo aleatorio se le puede realizar cualquiera de las 3 preguntas al usuario. Los resultados se enseñan de la misma manera que los modos anteriores. Ejemplo con Suryanamaskara A.

Finalmente, el modo 1 cuenta con una opción en la que se le elige el tipo de pregunta aleatoriamente entre los 3 tipos disponibles. En la Figura 3.35 se tienen las 3 preguntas distintas de manera consecutiva.

Primero al usuario se le pregunta sobre el drishti de la primera y segunda postura, luego el nombre de la tercera y finalmente por la postura correspondiente al nombre de la 4 asana en la secuencia.

3.2.2. Modo 2: Preguntas sobre secuencia

En este modo, el usuario debe ir avanzando por la secuencia eligiendo cual es la postura que sigue a la opción que se le muestra. Debido a que siempre se le muestra la primera postura, este modo tiene una pregunta menos que el modo 1. Esto se puede notar comparando el contador de posturas de la Figura 3.31 con la Figura 3.36 donde ambos casos son preguntas sobre el Suryanamaskara A pero en el modo de secuencia hay una pregunta menos.

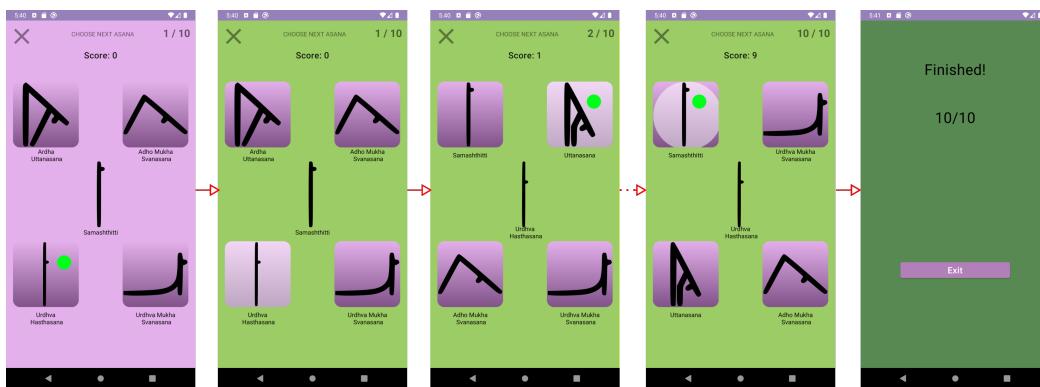


Figura 3.36: Ejemplo del usuario contestando perfectamente preguntas sobre el orden de Suryanamaskara A.

En este modo, también se utiliza el mismo cambio de fondo con retraso del modo 1. En el ejemplo de la Figura 1, se realizó una práctica del orden del Suryanamaskara A. Aquí, se obtuvo un resultado perfecto, debido a que durante el desarrollo de la aplicación la secuencia Suryanamaskara A fue la que se utilizó para realizar todas las pruebas iniciales por lo que ya se había aprendido bastante de esta secuencia a través de múltiples pruebas de funcionamiento. Este modo también complementa el modo de nombres ya que el usuario puede leer el nombre de todas las posturas que está contestando.

3.2.3. Modo 3: Preguntas de técnica y orden

En este modo, la cantidad de preguntas corresponde al doble del tamaño de la secuencia menos 1. Esto se debe a que se realizan dos preguntas por postura, exceptuando la última que no tiene una postura que la sigue.

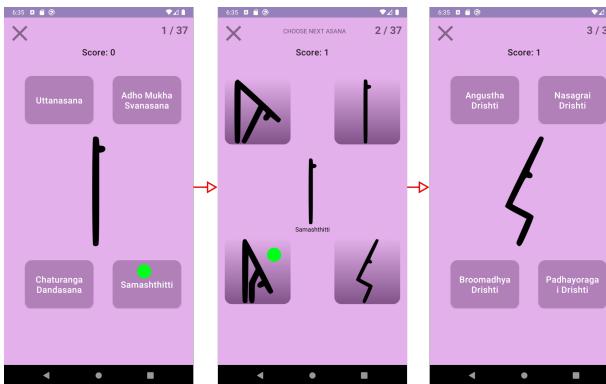


Figura 3.37: Ejemplo de funcionamiento con la secuencia Suryanamaskara B. Esta tiene 19 posturas por lo que se realizan 38-1 preguntas.

En este modo se combina el código para las preguntas de secuencia y de técnica aleatorio para poder alternar entre tipos de preguntas. Al usuario se le muestra una de 2 posibles preguntas de técnica (drishti y nombre), seguida por una pregunta de secuencia. El tipo de pregunta que muestra el nombre de la postura y el usuario debe elegir la imagen correspondiente se omite debido a que la respuesta de esta pregunta es la respuesta de la pregunta anterior (imagen de siguiente postura). El funcionamiento de este modo se exemplifica en la Figura 3.37. Primero el usuario tiene que contestar el nombre de la postura, seguido por la postura siguiente. Luego el usuario debe contestar una pregunta sobre el drishti de la postura que seguía. A diferencia del modo 2, en las preguntas de secuencia se esconde el nombre de las opciones, para que el usuario no sepa la respuesta a una posible pregunta de nombre. Este modo es más largo que los anteriores y al practicar la secuencia completa puede llegar a 204 preguntas, por lo que a la aplicación se le podría añadir una memoria que mantenga el progreso del usuario luego de salir de la secuencia o cerrar la aplicación.

3.2.4. Modo 4: Práctica

El modo 4 consiste en mostrar las posturas de una secuencia al usuario con su bandha, drishti, imagen y nombre para que puedan tenerlo de apoyo mientras practican. Aquí se buscó que alternar entre las posturas fuera fácil por lo que presionar en cualquier punto a la derecha o izquierda de la pantalla va a cambiar la postura.

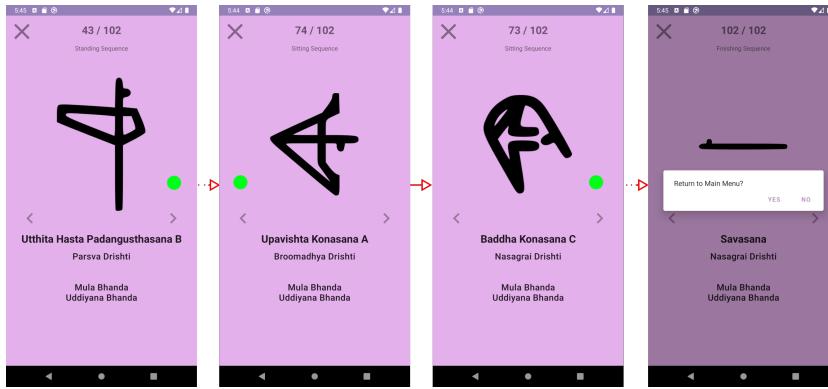


Figura 3.38: Ejemplo de navegación a través de las posturas de la secuencia completa. Aquí el usuario eligió practicar la secuencia completa.

El ejemplo en la Figura 3.38 muestra como el usuario podría navegar por las posturas mientras practica. Al presionar a la derecha de la pantalla avanza por las posturas. Al llegar a la postura 74 de la secuencia sentado se devuelve a la 73 presionando a la izquierda. Finalmente sigue avanzando hasta llegar a la última postura donde la aplicación verifica si quiere terminar luego de haber presionado el botón de siguiente en la última postura. En este modo, nombres largos como el nombre de la postura 43 en la Figura 3.38 caben correctamente en la pantalla y se muestran de la manera apropiada.

3.2.5. Modo 5: Repaso con fichas

El último modo, le muestra una serie de fichas con información de cada postura al usuario. Aquí puede revisar todas las posturas de una manera rápida desplazándose hacia abajo por todas las fichas. Para volver al menú principal, la ventana cuenta con una flecha de salida en la barra de acciones superior. En la Figura 3.39 se observa como se tienen múltiples posturas al mismo tiempo y puede rápidamente avanzar en la serie moviéndose hacia abajo. Este modo funciona como un repaso rápido de las posturas, pero por el tamaño de las fichas no es tan útil para usarlo mientras se practica como el modo 4 de práctica.

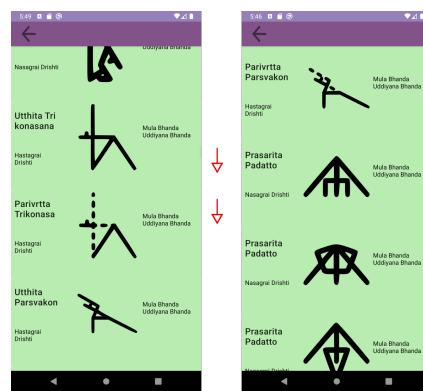


Figura 3.39: El icono de la aplicación es una flor de loto compuesta por distintas posturas de yoga. La resolución es varía con el dispositivo utilizado.

Capítulo 4

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se investigó sobre el Ashtanga Yoga y su serie primaria. Información sobre cada Asana en la secuencia se logró agregar a la aplicación para que el usuario pudiera observarla e interactuar con ella a través de distintas dinámicas.
- El lenguaje de programación Kotlin fue aprendido desde cero a partir de tutoriales diseñados por Google, y la implementación práctica de estos conocimientos adquiridos en el desarrollo de una aplicación para Android. Fue posible navegar entre pantallas, guardar datos entre estas ventanas, registrar las interacciones del usuario, manejar datos, cambiar el diseño de la aplicación y darle realimentación al usuario sobre su interacción con esta aplicación.
- Las dinámicas diseñadas funcionan para ayudar a practicantes de Ashtanga Yoga con algunos aspectos importantes sobre la serie 1 como: los nombres de las posturas, el enfoque visual de la postura, cómo se ve cada postura y el orden en que se debe realizar cada una de las 5 secuencias que componen esta serie. Estas dinámicas se desarrollaron con una interfaz intuitiva basada en el diseño de otras aplicaciones para memorización disponibles para Android.
- Luego de tener una aplicación en la que se podía navegar correctamente, se procedió a trabajar en el proceso de verificación de su funcionamiento. Aquí se encontraron distintos errores y problemas que podía tener la aplicación. A partir de estos errores se idearon soluciones para arreglarlos, desarrollando mayores habilidades en Kotlin y aprendiendo sobre las múltiples maneras en las que una aplicación de Android puede tener problemas.
- El modo 1 sobre preguntas de técnica, permitió desarrollar una dinámica que ayudará al usuario a relacionar imágenes con teoría sobre cada postura. Permitiendo al usuario conectar estas imágenes con las posturas reales cuando esté practicando Ashtanga Yoga y ayudándole en su práctica. Los modos de preguntas sobre el nombre pueden presentar una complejidad mayor en ciertas secuencias en comparación al de drishti, ya que las variaciones en drishtis no son tantas como las variaciones de posturas y nombres.

- El modo 2 facilita el aprendizaje de cada secuencia. Durante el desarrollo de la aplicación se utilizó este modo múltiples veces como prueba y el conocimiento del estudiante sobre el orden de las secuencias Suryanamaskara A y B aumentó hasta el punto que podía consistentemente obtener todas las respuestas correctas en preguntas sobre el orden de esta secuencia.
- El modo 3 presentó un reto, al tener que combinar las ventanas hechas para los dos modos anteriores y tener que hacer variaciones a como estos modos se comportaban. Aquí se aprendió sobre la importancia de hacer funciones que puedan ser muy generales para que múltiples modos distintos pudieran usar una misma función sin tener que hacer muchos cambios. Este modo también fue el límite de la utilidad de la LiveData, ya que al haber cambios rápidos entre distintos tipos de preguntas, a veces era posible observar muy brevemente los cambios incorrectos en el LiveData antes de que se avanzara a la siguiente ventana donde ya todo se veía bien.
- En el modo 4 se aprovechó el código de los modos anteriores para fácilmente mostrar al usuario toda la información sobre las distintas posturas. Aquí se aprendió sobre como mostrarle al usuario una ventana con texto que le permitiera salir del modo, sin tener que agregar botones adicionales.
- Para el modo 4 se implementó una serie de tarjetas contenidas en una actividad que se podía desplazar hacia abajo. Esta actividad reutilizaba el espacio de la pantalla para ir mostrando las nuevas posturas conforme el usuario avanzaba.
- Con los conocimientos adquiridos durante el desarrollo de esta aplicación, se podría recrear esta aplicación desde el inicio teniendo más claros todos los conceptos sobre Kotlin aprendidos mientras se trabajaba en la aplicación. Debido a que se estaba aprendiendo el lenguaje mientras se desarrollaba la aplicación, la calidad del código también fue aumentando con el proyecto.

4.2. Recomendaciones

- Esta aplicación dependió mucho en el uso de LiveData, databinding fragmentos para manejar la información a través de las ventanas. A pesar de ser una manera relativamente simple de manejar esto, un manejo mediante código pragmático podría presentar una aplicación más fácil de modificar, con menos errores visuales y con mayor control sobre los distintos elementos de la interfaz.
- Desarrollar un código más robusto y modificable, para permitir que otras personas modifiquen fácilmente la aplicación para mejorarla o cambiar su temática a preguntas sobre otros temas.
- Utilizar la información dentro de la aplicación para desarrollar más variaciones a los tipos de preguntas. Algunas posibles dinámicas comunes en aplicaciones de aprendizaje son: preguntas de asociación, desplazar imágenes a la posición correcta, encontrar el error en la ficha de información y relacionar audio con la pregunta.
- Desarrollar una dinámica que ayude a los usuarios con la respiración para cada postura. Ya que este es un aspecto importante en la práctica del Ashtanga Yoga.

- Las imágenes utilizadas son de uso libre, y por su estilo y la complejidad de ciertas posturas no siempre es muy claro cómo se debe realizar una postura con base en la imagen dada. Por lo tanto, se podría buscar imágenes más claras y pedir autorización a la persona que las diseñó ó añadir un tipo de ayuda que le explique al usuario cómo realizar la postura. Esto se podría implementar mediante un botón que busque la postura en internet y permita al usuario observar imágenes más claras de cada postura.

Apéndice A

INFORMACIÓN DEL CÓDIGO

Todo el código del programa está disponible en: <https://github.com/LuisCP20/AshtangaApp>. Para instalar la aplicación se debe descargar el archivo APK, y permitir al dispositivo usar programas externos a la Google Play Store.

A.1. Diagramas del programa

A.1.1. Diagrama de flujo de preguntas

Funcionamiento básico de las preguntas del programa.

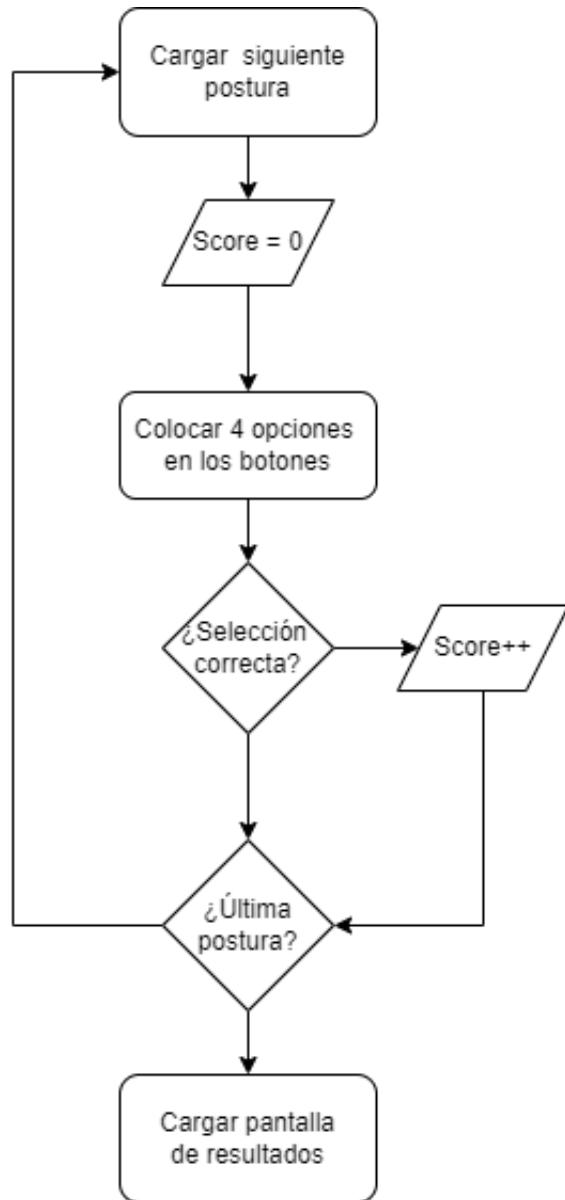


Figura A.1: Diagrama de flujo para las dinámicas de preguntas.

A.1.2. Clase Asana

Esta clase define una variable de tipo Asana. Esta contiene las características para cada postura. En caso de querer agregar más técnicas, se deben agregar aquí primero.

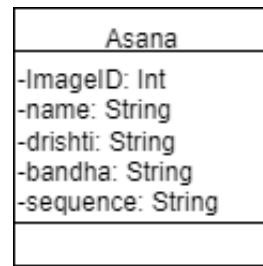


Figura A.2: Diagrama de la clase Asana.

A.1.3. DataSource

Este objeto contiene todas las posturas de la secuencia. Para modificar posutras se deben cambiar aquí. Esto se encuentra en el archivo `DataSource.kt`.

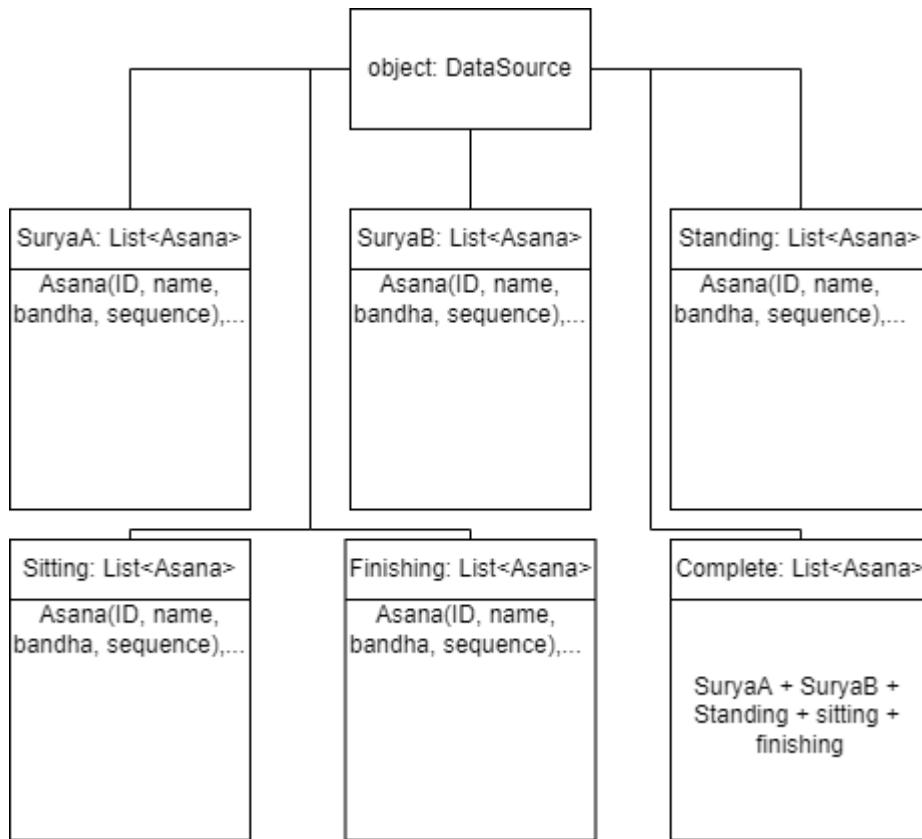


Figura A.3: Diagrama de objeto para `DataSource`.

Bibliografía

- [1] A. Lutz, *Ashtanga Yoga The Asanas of the Primary Series*. Ashtanga Studio Berlin. [Online]. Available: http://www.ashtangastudio.de/documents/primaryserieschartA4_001.pdf
- [2] B. Silver. (2009) Yoshiko at 8 months. [Online]. Available: <https://www.flickr.com/photos/gbsk/4028242162/>
- [3] S. P. Kantamani. (2020) Everything you want to know about android jetpack's navigation component. [Online]. Available: <https://betterprogramming.pub/everything-about-android-jetpacks-navigation-component-b550017c7354>
- [4] Material design introduction. [Online]. Available: <https://material.io/introduction>
- [5] Google fonts. [Online]. Available: <https://fonts.google.com/icons?selected=Material+Icons&icon.category=av>
- [6] (2021) Drops. [Online]. Available: <https://languagedrops.com/language/learn-persian>
- [7] (2020) Lotus flower yoga poses silhouettes. [Online]. Available: <https://freesvg.org/1545770750>
- [8] J. Savage. (2020) Ashtanga yoga the primary and intermediate series. [Online]. Available: <https://www.ekhartyoga.com/articles/practice/ashtanga-yoga-the-primary-and-intermediate-series>
- [9] J. Scott, *Ashtanga Yoga The Essential Step-by-step Guide to Dynamic Yoga*, 2nd ed. Gaia Classics, 2018.
- [10] Jois Yoga, *Astanga Yoga An Introduction to the Fundamentals of Astanga Yoga*. Jois Yoga, 2013.
- [11] C. Palavecino, *EL SURGIMIENTO DEL CUERPO EN LA PRÁCTICA DEL ASHTANGA VINYASA YOGA*. Universidad Nacional de la Plata, 2020.
- [12] A. Koletsou. (2020) How long does ashtanga primary series take? [Online]. Available: <https://yogamyoldfriend.com/how-long-does-ashtanga-primary-series-take/>
- [13] (2021) Android. [Online]. Available: <https://www.android.com>
- [14] (2021) Number of android apps on google play. [Online]. Available: <https://www.appbrain.com/stats/number-of-android-apps>

- [15] (2021) Android for developers. [Online]. Available: <https://developer.android.com/>
- [16] (2021) What is an api? (application programming interface). [Online]. Available: <https://www.mulesoft.com/resources/api/what-is-an-api>
- [17] F. Lardinois. (2019) Kotlin is now google's preferred language for android app development. [Online]. Available: <https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/>
- [18] Android's kotlin-first approach. [Online]. Available: <https://developer.android.com/kotlin/first>
- [19] M. Heller. (2020) What is kotlin? the java alternative explained. [Online]. Available: <https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html>
- [20] Android for Developers. View. [Online]. Available: <https://developer.android.com/reference/kotlin/android/view/View>
- [21] (2021) Anki. [Online]. Available: <https://apps.ankiweb.net/>
- [22] K. Tauber. Ashtanga yoga stick figure set. [Online]. Available: <https://stock.adobe.com/images/ashtanga-yoga-stick-figure-set/104103815>
- [23] Pei. Add ripple effect to my button with button background color. [Online]. Available: <https://stackoverflow.com/questions/40008609/add-ripple-effect-to-my-button-with-button-background-color>