Universidad Autónoma del Estado de México
Centro Universitario UAEM Zumpango

Ingeniería en computación

Desarrollo de una aplicación e-learning para aprendizaje de algoritmos genéticos

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

Jaaziel Isai Rebollar Calzada

DIRECTOR DE TESIS:
Dr. en C.C. Asdrúbal López Chau

Zumpango, Estado de México. Mayo 2021

Resumen

La tecnología avanza muy rápidamente, permitiendo que el acceso a la información sea más amplio y sencillo. El aprendizaje en línea es uno de los entornos que más demanda ha tenido en los últimos años, y ya que permite romper el tradicional esquema de enseñanza, descarta la necesidad estar presente para aprender. Por otro lado, la inteligencia artificial (IA) es una de las más grandes tendencias actualmente cuando de computación se trata. Los algoritmos genéticos son una de las ramas que abarca la IA y cumplen un destacable rol en la resolución de problemas de búsqueda y optimización. Al ser tendencia, es en ocasiones difícil encontrar información acerca de ella, y esto ocasiona que su aprendizaje sea complejo. El objetivo de este trabajo es el desarrollar una aplicación móvil para enseñar de una manera fácil y concisa los algoritmos genéticos.

Abstract

The tecnology advances very quickly, allowing that the access to information be wider and easier. The online learning is one of the enviroments that has had more demand in the last years, and since it allows breaking the traditional scheme of teaching, discards the need of being present to learn. On the other hand, the artificial intelligence (AI) is one of the greatest trends currently when is about computing. Genetic algorithms is one of the branches that includes the AI and achieves a remarkable role in the resolution of searching and optimization problems. As trend, it is sometimes difficult to find out information about it, and this makes it complex to learn. The objective of this work is devleoping a mobile application to teach genetic algorithms in an easier and concise way.

Dedicatoria

Dedico este trabajo de tesis al Centro Universiario UAEM Zumpango y a los alumnos de la carrera Ingeniería en Computación esperando que en el futuro pueda ser de utilidad a fin de complementar su desarrollo como futuros ingenieros en computación.

Agradecimientos

Agradezco al Centro Universiario UAEM Zumpango por brindarme las herramientas necesarias para convertirme en ingeniero y del mismo modo concluir esta tesis. A mis padres y hermano por el inmenso apoyo económico y moral. A mi asesor, el Dr. Asdrúbal López Chau por su guía en el desarrollo y culminación de este trabajo. A mi profesora Edith Herrera por inculcarme el gusto hacia la programación y desarrollo de software. A mi profesor Carlos Rojas por mostrarme que el desarrollo de aplicaciones móviles puede ser lo mejor.

Contenido

1 Introducción **1**
- 1.1 Planteamiento del problema . . . 2
- 1.2 Hipótesis . . . 2
- 1.3 Objetivos . . . 2
  - 1.3.1 Objetivo general . . . 2
  - 1.3.2 Objetivos específicos . . . 3
- 1.4 Justificación . . . 3
- 1.5 Alcance del proyecto . . . 4

2 Marco teórico **5**
- 2.1 Aprendizaje en línea . . . 5
  - 2.1.1 Tipos de aprendizaje en línea . . . 7
- 2.2 La educación a distancia . . . 11
- 2.3 Los entornos e-learning . . . 12
  - 2.3.1 M-learning . . . 12
  - 2.3.2 B-learning . . . 12
- 2.4 Plataformas para aprendizaje a distancia . . . 13
- 2.5 Educación a distancia en situaciones extraordinarias . . . 15
- 2.6 Algoritmos genéticos . . . 16
  - 2.6.1 Antecedentes teóricos . . . 17
- 2.7 Componentes de un algoritmo genético . . . 18
  - 2.7.1 Codificación . . . 18
  - 2.7.2 Cromosoma, gen y alelo . . . 19
  - 2.7.3 Operadores genéticos . . . 20
  - 2.7.4 Función de aptitud . . . 23
  - 2.7.5 Mecanismos de selección . . . 23
  - 2.7.6 Estructura general de un algoritmo genético . . . 27
- 2.8 Tecnologías involucradas . . . 29
  - 2.8.1 Android . . . 29
  - 2.8.2 Android Studio . . . 39
  - 2.8.3 Kotlin . . . 48
  - 2.8.4 Firebase . . . 51
- 2.9 Scrum . . . 57

2.9.1 Implementación 57
2.9.2 Scrum en AGtive 59

3 Desarrollo **65**
3.1 Diagramas del sistema AGtive 65
3.1.1 Diagrama de casos de uso 66
3.1.2 Diagrama de clases 67
3.1.3 Diagrama de estados 69
3.2 Desarrollo de AGtive implementando Scrum 72
3.2.1 Sprint 1 72
3.2.2 Sprint 2 83
3.2.3 Sprint 3 89
3.2.4 Sprint 4 y 5 95

Conclusiones **106**

Apéndices **108**

Apéndice A Cuestionario **109**

Apéndice B Publicación de la aplicación en la Play Store **118**

Referencias **122**

Lista de figuras

2.1 Operador de cruce por un punto 21
2.2 Cruza por dos puntos 22
2.3 Mutación en el primer descendiente de la figura 2.2 22
2.4 Pseudocódigo del algoritmo genético simple [16] 28
2.5 Ciclo de vida de una actividad en Android 37
2.6 Abrir Android Studio 39
2.7 Crear proyecto 40
2.8 Elegir API objetivo 40
2.9 Abrir SDK Manager 41
2.10 Cargar SDK 42
2.11 Herramientas de desarrollador 43
2.12 Archivos autogenerados 44
2.13 Botón agregado 44
2.14 Evento de botón 45
2.15 Iniciar aplicación 46
2.16 Aplicación en ejecución 47
2.17 Definición de una clase en Java 50
2.18 Definición de una clase en Kotlin 51
2.19 Nuevo proyecto 53
2.20 Nombre del paquete de la aplicación 54
2.21 Archivo autogenerado por Firebase 55
2.22 *build.gradle* con los repositorios de Google 55
2.23 *build.gradle* con las dependencias requeridas 56

3.1 Casos de uso en AGtive 66
3.2 Diagrama UML 67
3.3 Parte 1 del diagrama de estados 69
3.4 Parte 2 del diagrama de estados 70
3.5 Parte 3 del diagrama de estados 71
3.6 Clase Usuario 72
3.7 Clase Alumno 73
3.8 Clase Curso 73
3.9 Clase Quiz 74

3.10 Clase Pregunta 74
3.11 Clase Respuesta 74
3.12 Logueo con Gmail 75
3.13 Paleta de elementos de vista 76
3.14 Ejemplo de XML para vistas en Android 77
3.15 Menú de usuario 78
3.16 Plano del primer quiz del curso Representación entera 79
3.17 Plano del segundo quiz del curso Representación entera 80
3.18 Parte 6 del curso como ejemplo 81
3.19 Plano de la sección A del examen final de Representación Entera 82
3.20 Habilitación de Authentication 83
3.21 Confirmación de autenticación con Firebase 84
3.22 Dependencias para Firebase Authentication 84
3.23 Inicialización de objetos de tipo Firebase 85
3.24 Código de logueo con Gmail 86
3.25 Listeners para la actividad principal 87
3.26 Cuentas Gmail en Firebase Authentication 88
3.27 Habilitación de Firebase RDB en Android 89
3.28 Dependencia para utilizar Firebase RDB en Android 90
3.29 Apartado para configurar las reglas 91
3.30 Zona de prueba de reglas 91
3.31 Datos de Realtime Database en consola web 92
3.32 Función insertar en Realtime Database desde código 93
3.33 Resultados de inserción en consola de Firebase RDB 94
3.34 Temario y segunda vista del curso R.E 95
3.35 Código para continuar el curso donde se dejó 96
3.36 Ejemplo de implementación y primer quiz con datos 98
3.37 Parseo de datos XML para llenar el quiz 99
3.38 Segundo quiz y clase Cromosoma 100
3.39 Eventos de arrastre en el segundo quiz 101
3.40 Clase Individuo en diversos lenguajes de programación y ejecución conjunta de la representación entera 103
3.41 Ejercicios del examen de representación entera 104
3.42 Actualización de datos del alumno aprobado 105

A.1 Lista de personas que contestaron el cuestionario 110
A.2 Lista de personas que contestaron el cuestionario 112
A.3 Primera impresión de los usuarios sobre AGtive 113
A.4 Primera impresión de los usuarios sobre AGtive 114
A.5 Comentarios de los usuarios sobre el curso "Representación Entera" . . 115
A.6 Comentarios de los usuarios sobre el curso "Representación Entera" . . 116
A.7 Opiniones generales sobre mejoras en la aplicación 117

B.1 AGtive en la Playstore 119
B.2 Panel inicial de la Google Play Console para AGtive 120
B.3 Grafíca que muestra los dispositivos con AGtive 120
B.4 Calificación general de AGtive y errores o bloqueos presentados 121
B.5 Número de dispositivos que han adquirido AGtive y la fuente de donde se adquirió 121

Lista de tablas

2.1 COMPARATIVA DE ALGUNAS PLATAFORMAS WEB PARA APRENDIZAJE A DISTANCIA 14
2.2 EJEMPLO DE UN TABLERO BÁSICO DE SCRUM 58
2.3 TABLERO DEL SPRINT 1 60
2.4 TABLERO DEL SPRINT 2 61
2.5 TABLERO DEL SPRINT 3 62
2.6 TABLERO DEL SPRINT 4 63
2.7 TABLERO DEL SPRINT 5 64

Capítulo 1

Introducción

En la actualidad las nuevas tecnologías como Internet han permitido que el tradicional modelo de aprendizaje evolucione, pues al existir la posibilidad de aprender a distancia, se descarta la de estar en un espacio físico para aprender o enseñar. Lo anterior resuelve algunas dificultades sobre la educación presencial, como la reducción de costos económicos, materiales y tiempo.

Por otra parte, la inteligencia artificial es una de las actuales tendencias en el mundo, dentro de esta vasta área de la computación, los algoritmos genéticos (AG) juegan un papel importante en las heurísticas para optimización. Los algoritmos genéticos cumplen un rol destacable en la resolución de problemas de búsqueda y de optimización. Al estar basados en la teoría de la evolución de las especies, los algoritmos genéticos usan varios "individuos" que representan soluciones a un problema para tratar de elegir al más apto. Sin embargo, a pesar de que el estudio de esta rama de la IA puede darse por medio de libros, páginas web, videos, foros o cursos; hasta el momento de redactar este documento no se encontró un software enfocado en el aprendizaje de estos. Este documento plantea el desarrollo de una aplicación móvil para sistema operativo Android orientada al aprendizaje de algoritmos genéticos, con el fin de facilitar el entendimiento y dominio del tema.

1.1 Planteamiento del problema

El aprendizaje en línea es una tendencia a nivel mundial. Esto debido a diversas razones favorables como las siguientes: la reducción de costos tanto para las instituciones y personas que imparten cursos como para quienes los toman, pues no se requiere de una infraestructura física; el acceso a una plataforma electrónica en cualquier momento, ya que al estar disponible el contenido el usuario puede acceder a éste cuando mejor le convenga.

Actualmente existe una gran cantidad de plataformas para el aprendizaje en línea, muchas de ellas relacionadas con el área de computación; sin embargo, hay pocos recursos electrónicos (cursos, específicamente) para el aprendizaje de los AG. La necesidad de aprenderlos ha orillado a desarrollar cursos, temarios, y foros en los que se incluye información sobre el tema, pero no hay una que cubra lo fundamental y central de éste. En este trabajo, se propone llenar este hueco mediante el desarrollo de una aplicación para dispositivos móviles orientada a la enseñanza de los AG.

1.2 Hipótesis

Esta tesis tiene como hipótesis de partida que es posible desarrollar una aplicación para dispositivos móviles con sistema operativo Android que permita mostrar los temas principales y de mayor relevancia sobre algoritmos genéticos. Se considera que, mediante esta aplicación, se facilitaría su aprendizaje.

1.3 Objetivos

1.3.1 Objetivo general

Desplegar un sistema e-learning para aprendizaje de algoritmos genéticos en un dispositivo Android.

1.3.2 Objetivos específicos

Los objetivos específicos de esta tesis son los siguientes:

1. Realizar una revisión de los principales conceptos de algoritmos genéticos.
2. Realizar una revisión de las características de los sistemas e-learning.
3. Establecer los requisitos y alcances del sistema a desarrollar.
4. Describir las herramientas y *frameworks* disponibles para desarrollo de aplicaciones Android para elegir la más adecuada.
5. Diseñar la arquitectura y representarla mediante diagramas UML.
6. Implementar el tema "Representación entera" en el sistema e-learning.
7. Realizar pruebas de funcionamiento al sistema para verificar que cumple con los requisitos establecidos.

1.4 Justificación

Los AG son una parte fundamental en el estudio de la IA. El modo de empleo se basa en el proceso genético de los organismos vivos, y al repetirlo un determinado número de veces, lo que se conoce como generaciones, las poblaciones permiten producir diversas soluciones a un problema. Es importante mencionar que el uso de AG en la resolución de problemas es en muchas ocasiones la última alternativa para encontrar la mejor solución.

Este trabajo busca desarrollar un software para su enseñanza; la complejidad de estos es considerable, por lo que para facilitar su aprendizaje, es necesaria la implementación de una aplicación enfocada en el tema previamente mencionado. Además, la contingencia por COVID-19 a nivel mundial en el año 2020, ha demostrado la necesidad del uso de herramientas para aprendizaje a distancia y de auto-aprendizaje haciendo uso de las TIC, por lo que este trabajo contribuiría en este sentido.

1.5 Alcance del proyecto

El sistema desarrollado con base en este trabajo se limita a cubrir las siguientes funcionalidades básicas:

- Registro de un usuario en el sistema mediante cuenta de correo electrónico Gmail.
- Mostrar el curso: Representación entera.
- Interfaz para visualizar contenido del curso.
- Interfaz para navegar entre los temas del curso.
- Interfaz de usuario para mostrar su información personal, además de controlar el acceso a la cuenta.

Capítulo 2

Marco teórico

Este capítulo se divide en tres partes. La primera muestra a los sistemas e-learning, presentando sus características, ventajas y desventajas. La segunda se enfoca en los fundamentos de los algoritmos genéticos, dando a conocer sus principales componentes. La tercera abarca las tecnologías involucradas para desarrollar la aplicación e-learning.

2.1 Aprendizaje en línea

Internet ha sido uno de los pilares de apoyo del sector educativo y otras áreas, siendo un medio de difusión y comunicación abierto y flexible. Esta red de redes ha abierto la posibilidad de existencia a diversas aplicaciones; de entre las primeras que se dieron a conocer y más comunes están la mensajería electrónica, negocios web y entretenimiento en línea.

Una limitante para la educación es la ubicación geográfica; Internet por otro lado, rompe con esta barrera además de los esquemas tradicionales de enseñanza-aprendizaje. La Web como medio de difusión de la información, ha facilitado la creación y acceso a más contenidos de todo tipo, incluyendo el didáctico, además de servir como infraestructura para impartir educación a distancia. La implementación de dicha tecnología ha causado el surgimiento de un modelo de aprendizaje conocido como e-learning [12], que en inglés significa "Electronic learning", y en español se traduce

como “Aprendizaje electrónico”. Para implementar un sistema de este tipo, se requiere seguir algunos pasos:

- Es necesario dar un panorama general sobre la importancia de los sistemas web en el ámbito educativo a distancia.
- Se definen los diversos entornos e-learning para centrarse en el contexto del trabajo.
- Se deben mostrar algunos sistemas que intervienen en estos entornos, por ejemplo, los Sistemas de Administración de Aprendizaje. Se deben considerar y tratar las características deseables de éstos y se finaliza con un entorno integral e-learning.

A continuación se muestran algunas ventajas y desventajas de los sistemas de aprendizaje en línea.

Ventajas del aprendizaje en línea

- Se elimina la necesidad de asistir físicamente a un lugar para tomar una clase.
- El costo de una clase, curso o acceso a información puede ser menor o incluso gratis, que uno de modalidad presencial.
- La mayoría de las veces, el acceso a la información a través de la Web puede darse en cualquier momento [12].
- El tener disponibles diversas fuentes e-learning, permite comparar la información y discernir de ella más fácilmente.
- Actualmente, consultar información utilizando sistemas e-learning, puede darse desde casi cualquier lugar y casi cualquier dispositivo de comunicación como una computadora, smartphone o tableta electrónica, por mencionar algunos.

Desventajas del aprendizaje en línea

- La información disponible en Internet puede no ser correcta. Es necesario asegurarse que los cursos o información consultada sea proveniente de una fuente confiable.

- El costo inicial puede ser mayor en una plataforma e-learning que en la modalidad presencial. Sin embargo, a mediano y largo plazo el costo de las plataformas e-learning es menor con respecto a la educación presencial.

- Al estudiar de manera autodidacta mediante recursos en línea, usualmente surgen dudas, por lo que en ocasiones es más complicado resolverlas a través búsquedas en Internet a comparación de un tutor presencial que pudiera ofrecer una explicación más precisa. Para minimizar este problema, muchas de las plataformas actuales tienen un sistema para comunicación en línea con otros participantes o con el instructor.

- Algunas plataformas e-learning exigen completar cierto curso en un tiempo determinado, de modo que si esto no se cumple, la información deja de estar disponible para el usuario.

Una vez mostradas algunas de las ventajas y desventajas del aprendizaje en línea, se presentarán sus tipos.

2.1.1 Tipos de aprendizaje en línea

Antes que nada, es preciso definir que el término aprendizaje en línea se forma a partir de conceptos que son la enseñanza virtual y los estilos de aprendizaje.

Enseñanza virtual

La enseñanza virtual tiene como objetivo enfocarse en el estudiante para fortalecer y actualizar su conocimiento en combinación con distintos elementos de índole

pedagógica [9], sustentándose en las TIC. En la actualidad el tradicional modelo de interacción-aprendizaje puede darse por medio de videoconferencias, videotutoriales o chats; el contacto difiere entre tutores y compañeros a través de foros de debate y correo electrónico, y una diferente interacción con materiales de estudio por medio los multimedios.

Actualmente los paradigmas de educación pedagógica se centran en la construcción del conocimiento y de los procesos reflexivos, esto ofrece mayor oportunidad del docente hacia el discente el proporcionar actividades para reflexión y de ese modo dar soporte a su aprendizaje [9], por lo tanto puede sugerir diversas fuentes de información y ampliar el campo de explicaciones para favorecer el proceso de comprensión, es decir, el profesor tiene la oportunidad y hasta cierto punto la obligación de guiar y orientar al alumno. En la comunidad virtual las interacciones entre los individuos son mayores y se logra con el uso de diversas herramientas disponibles como por ejemplo lo es un foro, el cual se presenta como un espacio con amplias posibilidades para contribuir a las relaciones sociales a través de interacción y colaboración.

La educación es y hace referencia a *dirigir*, *encaminar* y *desarrollar*. Este concepto se puede extender a *perfeccionar* las habilidades físicas o intelectuales de un individuo a través de preceptos, ejercicios y/o ejemplos [13].

El concepto de aprendizaje puede tomar distintos rumbos según se considere al autor, por ejemplo el cognitivismo, que es considerado como un proceso de modificación interna y con cambios tanto cuantitativos como cualitativos el cual existe a partir de un proceso interactivo entre la información que procede de un medio y un sujeto activo [9]. Por otro lado, el autor John Cotton Dana afirma que el aprendizaje sólo existe cuando se adquiere un nuevo conocimiento y no sólo una retención temporal, por lo que se trata de un cambio permanente en el individuo. El autor González Romero menciona que la información son sólo datos con un significado, por lo que no se considera conocimiento, y para que esté presente, se debe asimilar e interiorizar la información a fin de lograr un aprendizaje [13].

Existen diversos factores que implican que de un proceso de aprendizaje se obtengan distintos resultados; se pueden considerar cuatro elementos que abarcan ciertos estímulos externos al momento del aprendizaje, los sociológicos que son la edad, el nivel socioeconómico y la cultura. Un concepto proveniente de la psicología es el de "Estilos de Aprendizaje" [9], que hace referencia a la manera en que las personas perciben e interactúan con un entorno de aprendizaje.

Estilos de aprendizaje

Los estilos de aprendizaje pueden definirse de diferente manera, según los autores Quiroga y Rodríguez [6] son aquellos que reflejan las diferencias cualitativas y cuantitativas de cada persona, por lo que determinan la manera en que ésta desarrolla el aprendizaje a través de la percepción, atención y pensamiento. Otros autores como Dunn y Price proporcionan una definición en donde estos estilos son la manera en que la habilidad para absorber y retener información de una persona se ve afectada por estímulos externos. Otra definición dice que son estrategias cognitivas con las que se recopila, interpreta y organiza la información adquirida [9]. Dado que no existe una definición única y total de los estilos de aprendizaje, se plantean varios de ellos.

- **Estilo activo**: Se caracteriza por tratarse de personas sin temor a nuevas experiencias y de mente abierta.
- **Estilo reflexivo**: Aplica a aquellos individuos que prefieren analizar, observar y entender la situación antes de realizar algún movimiento.
- **Estilo teórico**: Similar al estilo reflexivo, pero en este las personas observan las situaciones de una manera más lineal para buscar la racionalidad y objetividad a
- **Estilo pragmático**: Implica a aquellas personas que buscan aplicar ideas nuevas tan pronto sea posible, por lo que la planificación de actividades no es algo muy importante.

Se tienen también los estilos de aprendizaje según Kolb [6]:

- **Estilo convergente**: Busca aplicar ideas a través de un conocimiento organizado y la resolución de problemas mediante un razonamiento hipotético-deductivo.
- **Estilo divergente**: Aquel que se basa en la flexibilidad el pensamiento y por lo tanto en la imaginación teniendo la capacidad de considerar las situaciones desde muchas perspectivas.
- **Estilo asimilador**: Se enfoca en comprender los modelos técnicos dejando de lado la aplicación de lo concreto.
- **Estilo acomodador**: Está centrado en crear, experimentar y aplicar ideas en situaciones nuevas sin temor a los riesgos.

Por otro lado están los estilos según Witkin [6]:

- **Dependencia del campo**: Su base fundamental es la de un plan concreto y el trabajo en equipo es su herramienta principal. No gusta de trabajar con algo desestructurado.
- **Independencia de campo**: Se muestra interesado en trabajar con ideas abstractas y con materiales muy poco estructurados. Es riesgoso pero se compensa con la flexibilidad que proporciona.

Las definiciones para los estilos de aprendizaje difieren sutilmente. Con base en los conceptos anteriores, se pueden definir dos estilos de aprendizaje en donde uno se trata de personas que buscan un plan estructurado o lineal para aprender, mientras que el otro implica adquirir información de manera dinámica y flexible.

Los tipos de aprendizaje en línea se ven reflejados de distinta manera en el individuo, pues como se menciona anteriormente, existen diversas maneras de aprender. El aprendizaje a través de la Web puede tornarse sencillo o complicado dependiendo del estilo en que el individuo aprende.

2.2 La educación a distancia

La educación a distancia en conjunto con las nuevas tecnologías desarrolladas en los últimos años ha tenido un proceso de evolución a la par a tal grado de producir diversos medios de comunicación como la radio, la televisión, el teléfono, entre otros. Internet ha sido una de las tecnologías que mayores cambios han sufrido en cuanto a los diferentes servicios que provee [12], los cuales han causado que la educación a distancia sea una alternativa para aquellas personas que tienen limitantes de tipo geográfico, ocupacional o físicas como lo plantea el modelo tradicional.

Dejando de lado por un momento las tecnologías web, pero tomando como base que éstas han sido uno de los medios por los que la gestión de información tiene tres formas en las que es participe en un proceso de enseñanza-aprendizaje, se definen de la siguiente manera:

1. **Web para la gestión de información:** La gestión implica el almacenamiento, diseminación y recuperación. Las personas pueden consultar información o tener acceso a diversos contenidos a través de la Web.

2. **Web para enseñanza en dos medios (mixta):** Dicho de otra forma, enseñanza semipresencial; una parte de la instrucción es presencial y la otra parte por medio de la Web; se le conoce como aprendizaje mixto (”blendend learning”).

3. **Aprendizaje basado en la Web:** Se centra en llevar todo el proceso educativo a una plataforma web, haciendo nulo al modelo de aprendizaje presencial.

2.3 Los entornos e-learning

Como se ha mencionado anteriormente, la educación a distancia es una de las modalidades de enseñanza más populares. Al estar basada en la Web, los sistemas o entornos de tipo e-learning posibilitan y facilitan el proceso de enseñanza-aprendizaje.

E-learning es un término que abarca un amplio grupo de aplicaciones y procesos. El aprendizaje donde está basado en Web y en computación, las aulas virtuales y colaboración digital forman parte de éste. Dicho concepto, integra diversos tipos de contenido que pueden recuperarse vía Internet por medio de audios y videograbaciones, transmisiones satelitales, televisión interactiva, CD-ROM, entre otros más [12].

La definición de e-learning permite abrir un extenso rango de posibilidades para cualquier proceso relacionado con educación y tecnologías. En los últimos años se han implementado dos técnicas conocidas como m-learning y b-learning [8].

2.3.1 M-learning

Mobile-learning es una evolución de e-learning, pero enfocada al uso de tecnologías que no involucran un lugar estacionario como ocurre con las computadoras de escritorio. En todo caso se utilizan laptops, smartphones, tabletas o agendas electrónicas y cualquier dispositivo que tenga conectividad inalámbrica a Internet.

2.3.2 B-learning

Blended-learning o en español aprendizaje combinado, unifica la formación presencial con e-learning. Implica el uso de técnicas presenciales de enseñanza con actividades de e-learning. Para implementar este método de aprendizaje se requiere de al menos un dispositivo electrónico que permita acceder a la Web y la presencia de un educador.

2.4 Plataformas para aprendizaje a distancia

La educación a distancia ha evolucionado de distintas formas permitiendo ampliar el campo de distribución en donde los sistemas e-learning actualmente no están sólo en la Web. Con el desarrollo de tecnologías como los smartphone [1], el campo de desarrollo de herramientas y aplicaciones ha ido en aumento permitiendo así extender el aprendizaje hacia distintos enfoques. La tabla 2.1 muestra algunas de las plataformas de aprendizaje en internet más utilizadas que aplican el concepto e-learning.

TABLA 2.1: COMPARATIVA DE ALGUNAS PLATAFORMAS WEB PARA APRENDIZAJE A DISTANCIA

Plataforma	**Gratuita**	**Entrega de documento**	**Plataforma dirigida**	**Retroalimentación**	**Disponibilidad**	**Evaluación**	**Observaciones**
Opositer	No	Matrícula	Web y móvil	Sí, por parte del profesor	Total	Test de verdadero y falso	La mayoría de los cursos se enfocan en la docencia
edX	Sí	Ninguno	Web	Sí, por parte de la comunidad de alumnos	Total	Ejerciciosde evaluación	Cursos de diversos temas y casi cualquier persona puede aplicar para impartir un curso
Future Learn	Sí	Título o grado para elegir	Web	Sí, por parte del profesor y la comunidad de alumnos	Total	Quizes	Los docentes son profesores de universidades reconocidas
Campus Educación	No	Matrícula	Web	Sí, por parte de la comunidad a través de un blog	Total	Pruebas de conocimiento	Cursos homologados
Red Educa	No	Certificado en algunos cursos	Web y móvil	Sí, por parte del profesor y la comunidad de alumnos	Total	Pruebas, tareas y exámenes en línea	Profesores de la Universidad de Nebrija
Coursera	Sí	Certificado	Web	Sí, por parte del profesor y la comunidad de alumnos	Parcial. Los tiempos varían para la oferta de un curso	Tareas y cuestionarios de práctica	Los profesores provienen de universidades reconocidas. Todos los cursos son en inglés
Udacity	No	Ninguno	Web	Sí, por parte de la comunidad de alumnos	Parcial. Los tiempos varían entre los diversos cursos	Cuestionarios y ejercicios acordes al curso	Todos los cursos están en inglés. Algunos de ellos tienen apoyo de empresas mundialmente reconocidas
Formación online	Sí	Certificado	Web	Sí, por parte de la comunidad de alumnos	Total	Ninguno	Es posible descargar los cursos
Open2study	Sí	Ninguno	Web	Sí, por parte del instructor y la comunidad	Total	Diversos quizes y un examen por unidad de curso	La mayoría de cursos están en inglés. La plataforma tiene conexión con diversas universidades
Tutellus	Sólo algunos cursos	Ninguno	Web y móvil	Sí, por parte de la comunidad de alumnos	Total	Quiz	Contenido formativo de universidades y organizaciones. Cualquiera puede subir cursos
Free easy way	Sólo algunos cursos	Ninguno	Web	Sí, por parte de la comunidad de alumnos	Total	Quiz	Gran parte de los cursos los ofrecen instituciones reconocidas
Floqq	No	Certificado	Web y móvil	Sí, por parte de la comunidad de alumnos	Total	Ninguna	Los cursos son de corta duración y usualmente son de una sola clase
Udemy	No	Certificado	Web	Sí, por parte del instructor y la comunidad de alumnos	Total	Quiz por unidad de curso	Cualquiera puede subir cursos a la plataforma. Las políticas para cursos son muy estrictas
Aula Fácil	Sí	Certificado	Web	Sí, por parte de la comunidad de alumnos	Total	Cuestionarios y ejercicios para resolver	Cualquiera puede subir un curso
Harvard Online	Sólo algunos cursos	Certificado	Web	Sí, por parte de la comunidad de alumnos	Parcial	Ejercicios de evaluación	Todos los cursos están en inglés

2.5 Educación a distancia en situaciones extraordinarias

El acceso a Internet por parte de las personas alrededor del mundo abrió la posibilidad de educación a distancia usando medios electrónicos desde hace más de una década. Aunque la necesidad de un aprendizaje no presencial ha estado latente desde hace varios años, y pese a que muchas instituciones han optado por impartir educación académica de manera virtual, esta necesidad se ha incrementado gradualmente debido a situaciones extraordinarias.

En diciembre de 2019 en Wuhan, China surgieron varios casos reportados como neumonía, que fueron asociados a un virus poco conocido, y aunque esto inició como una epidemia limitada a China, pronto se convirtió en una pandemia. Hasta marzo de 2021 se estima que a nivel global hay más de 133 millones de casos de infección y más de 2 millones de muertes [21]. Esto indica que el virus nombrado SARS COVID2-19 es muy peligroso, y dado que el número de infectados y muertos es muy alto, muchos países se han visto en la necesidad de tomar medidas de extrema precaución para evitar su propagación en la población. Inglaterra [15] por ejemplo, fue uno de los primeros países que declaró el cierre temporal de cines, gimnasios, bares, teatros, restaurantes por mencionar algunos; muchos otros países del mundo han hecho lo mismo con la finalidad de implementar una cuarentena dentro del hogar, sin embargo, uno de los factores más importantes por dicha acción, ha sido el cierre temporal de escuelas, que va desde la educación básica hasta la educación superior, lo que rápidamente incrementó la necesidad de aprendizaje a distancia.

En México por ejemplo, el gobierno ha implementado dos estrategias para apoyar dicho tipo de enseñanza, por medio de la Web y a través de la televisión para aquellos que no tienen acceso a Internet. Otro factor importante son los profesores que en conjunto con los padres pueden facilitar la implementación del aprendizaje a distancia

en los alumnos/hijos. Situaciones sanitarias como la mencionada anteriormente, además de la económica y cultural, permiten ver más claramente la necesidad de crear nuevas herramientas para aprendizaje en todos los niveles educativos.

2.6 Algoritmos genéticos

Los algoritmos genéticos son métodos adaptables que se pueden usar en la resolución de problemas de optimización, se basan en la selección natural y en ciertos casos implican genética natural [2]. Una característica en particular de éstos es que se pueden obtener diversas soluciones a un problema, por lo que no hay garantía de que la mejor respuesta sea obtenida en el primer, segundo o enésimo intento [5].

La base del funcionamiento de los algoritmos genéticos es el proceso genético de los seres vivos; por lo que se manejan conceptos de la genética como: alelo, gen, cromosoma, individuo, aptitud, generación, población, cruza, selección y mutación. Las generaciones y las poblaciones a lo largo de los años evolucionan con base en los principios de la selección natural, postulado propuesto por Charles Darwin en 1859 [16], la cual menciona que los individuos más fuertes y capaces de adaptarse a un entorno sobreviven. Por medio de la imitación de este proceso, los algoritmos genéticos tienen el potencial para generar soluciones a problemas del mundo real. Es importante tener en cuenta que la cantidad y viabilidad de las soluciones, dependerá de la codificación que se emplee en las mismas, por lo que en ocasiones y dependiendo del problema que se pretenda solucionar, el primer código del algoritmo genético puede no ser el mejor, de modo que si se presenta este caso, es necesario modificarlo y mejorarlo cuantas veces se requieran para así lograr soluciones óptimas.

La naturaleza de los individuos en una población se enfoca en la supervivencia o reconocimiento por ser el mejor en algo, es por eso que existe una constante competencia por la búsqueda de diversos recursos según la necesidad del individuo. Las relaciones sociales interpersonales como el noviazgo, son un claro ejemplo en el que dos o más personas se encuentran en constante competencia, esforzándose por conquistar y cortejar a otra persona por la que siente atracción. Con base en este ejemplo, a aquellos individuos con más probabilidad éxito para sobresalir y por lo tanto atraer un número más alto de compañeros tienen mayor probabilidad de generar más descendientes [16]. Visto del lado de la genética, los genes de los individuos mejor adaptados tienden a propagarse en las futuras generaciones hacia un considerable incremento de descendientes.

La combinación de buenos rasgos entre dos individuos ancestros puede presentar el caso en el que se produzcan "súper individuos" los cuales tienen las mejores características de todas las generaciones pasadas, de esta manera se presenta el término conocido como evolución, que muestra que los rasgos del individuo se adaptan cada vez más y mejor al entorno en el que se desarrollan.

2.6.1 Antecedentes teóricos

Los algoritmos genéticos trabajan con una población de individuos en la que cada uno representa una solución factible a un problema y cada uno tiene un valor relacionado con la viabilidad de la solución. Cuanto mejor sea dicho valor, mayor probabilidad tendrá para ser seleccionado para reproducirse con otro individuo seleccionado del mismo modo, y al final producir un descendiente [2] con características de los padres. De esta manera se produce una nueva población de nuevas posibles soluciones, la cual pasa a reemplazar a la anterior, pues no habría lógica en mantenerla si ya se han seleccionado los mejores individuos para la generación futuro, y por lo tanto las mejores características se heredan. Este proceso se repite a lo largo de las generaciones para

favorecer el cruce entre los individuos mejor adaptados y la obtención de una o más soluciones óptimas [5].

Gracias a la robustez de los algoritmos genéticos, éstos tienen gran potencial para resolver problemas que otras áreas o metodologías encuentran difíciles [16]. Es de gran importancia tener en cuenta que muchos de los problemas que los algoritmos genéticos resuelven y otros métodos no, se debe a que no existe una técnica o fórmula especializada para hallar una solución; por el contrario, tener un método específico para un problema puede ser más eficiente que utilizar un algoritmo genético.

Existe un ejemplo denominado como Algoritmo Genético Simple o Canónico, el cual consiste en generar una población inicial y computar la función de evaluación o de aptitud de cada individuo; es necesario cumplir el criterio de parada, así que mientras no se obtenga un resultado esperado se debe generar una nueva generación, después comienza el proceso de reproducción, en el que se seleccionan [2] dos individuos de la población anterior con base en su función de aptitud, se cruzan con la posibilidad de obtener dos descendientes, y éstos mutan con la probabilidad de tener la función de aptitud de los individuos que les preceden, al final ambos se insertan en la nueva generación y todo el proceso se repite hasta que se cumpla el criterio de parada o bien, el número de generaciones haya finalizado.

2.7 Componentes de un algoritmo genético

2.7.1 Codificación

En los algoritmos genéticos, los individuos pueden representarse como un conjunto de valores denominados como genes [16], los cuales agrupados de cierta manera forman un cromosoma. Usualmente al cromosoma más simple se le atribuye el uso de números 1 y 0 para su representación; en términos biológicos el conjunto de parámetros que representan un cromosoma en particular se le llama fenotipo que contiene la información genética conocida como genotipo y es requerida para formar un organismo.

La resolución de un problema tiene como punto de partida la función de aptitud [2] del individuo, y puede ser programada a partir del cromosoma; cada función de adaptación debe ser diseñada de manera particular para cada problema. Entonces, cada función de aptitud le asigna un número real que refleja el nivel de adaptación al problema del individuo.

Durante la fase de reproducción se seleccionan los individuos de la población para cruzarse y generar descendientes los cuales posteriormente se mutan y forman parte de la nueva generación. La selección de los padres es aleatoria y la probabilidad de ser seleccionados es proporcional a su función de aptitud. Se dice que dicho procedimiento se basa en una ruleta.

2.7.2 Cromosoma, gen y alelo

En algoritmos genéticos existen algunos conceptos básicos que son necesarios de aprender para una mejor comprensión del tema ya que será lo primordial para comenzar a trabajar, y dado que están basados en la genética y evolución humana no hay gran diferencia en significado y sólo diferirán al momento de representarse e implementarse.

- **Cromosoma**: Es un elemento que se compone de material genético (ADN) y proteínas básicas. Contiene información genética que se presenta en la transmisión de los caracteres hereditarios [18].

- **Gen**: Es la unidad de acción y combinación del material genético presente en los cromosomas y que se forma de un segmento de ADN que es responsable de los caracteres hereditarios [18].

- **Alelo**: Es cada uno de los genes que se encuentran en el mismo lugar de los cromosomas [18].

Los conceptos anteriores aplican tanto a la biología como a los algoritmos genéticos, sin embargo, sólo cambia la manera en que se visualizan.
Un cromosoma va a estar conformado por una cadena o arreglo de parámetros en donde

usualmente son representados con bits, un gen por otro lado va a mostrar cada posición en dicha cadena, mientras que un alelo va a representar el valor de un gen.

2.7.3 Operadores genéticos

En el punto anterior se mencionaron tres elementos básicos de la genética que son necesarios para la implementación de los operadores genéticos la cruza y mutación; ambos son imprescindibles para la creación de un algoritmo genético básico.

Para efectuar dichos operadores es necesario tener dos individuos a los que les conoce como padres de los cuales se tomaran ciertos valores.

Cruza

Aunque existen varios tipos de operador de cruce, el más básico se es el "cruce por un punto", el cual consiste en dividir el cromosoma de los padres al azar para producir dos pares iniciales y finales de valores del cromosoma para después combinar los finales en dos cromosomas nuevos completos [5], de este modo ambos descendientes heredan genes de cada padre.

Usualmente la selección de padres se realiza aleatoriamente aplicando una probabilidad de 0.5 a 1.0. Cabe mencionar que si la cruza no se aplica, entonces se duplican los padres generando los mismos valores cromosómicos en la descendencia y de nada serviría generar nuevos individuos.

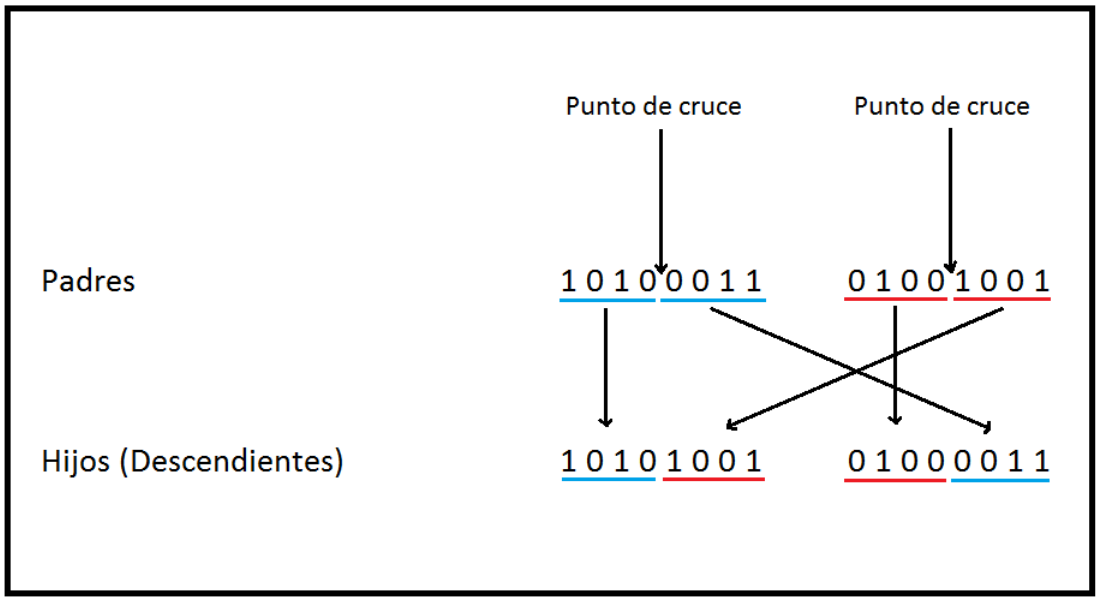

Figura 2.1: Operador de cruce por un punto

Existen otros operadores de cruce en donde usualmente se toman en cuenta más de un punto de cruce. De Jong encontró que dicho operador puede lograrse por medio del cruce de múltiples puntos, sin embargo, el cruce por dos puntos representa una mejora mientras que añadir más puntos no beneficiaba cómo se comportaba el algoritmo. La principal ventaja de tener más de un punto es que el espacio de búsqueda se puede explorar más fácilmente.

La figura 2.2 representa el modo en que la cruza por dos puntos se puede visualizar y en la que se realiza la selección aleatoria de dichos puntos obteniendo dos variantes de individuos posibles.

Mutación

La mutación es un proceso que se va a aplicar a cada hijo o descendiente de manera individual. Consiste en la alteración [5] aleatoria (usualmente con una probabilidad pequeña) de uno o más genes en el cromosoma. La figura 2.3 muestra este procedimiento.

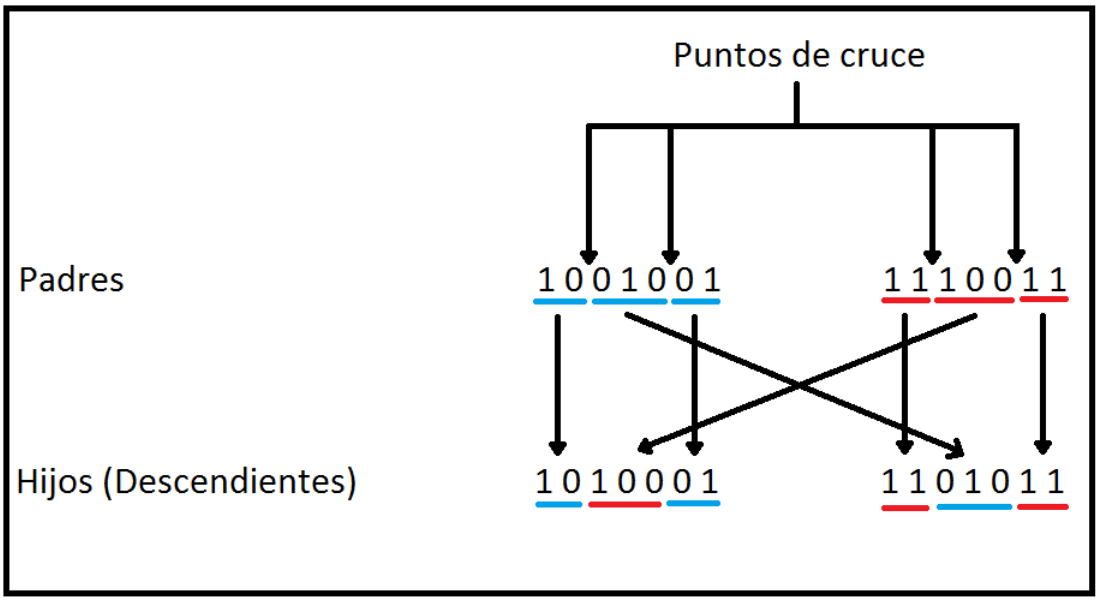

Figura 2.2: Cruza por dos puntos

Debido a que la cruza proporciona un exploración más rápida del espacio de búsqueda [16], se puede pensar que esta tendría mayor prioridad que la mutación, sin embargo, este último indica que ningún punto en el espacio de búsqueda debe quedar con cero probabilidad de ser examinado, por lo que es realmente importante tener segura la convergencia de los algoritmos genéticos.

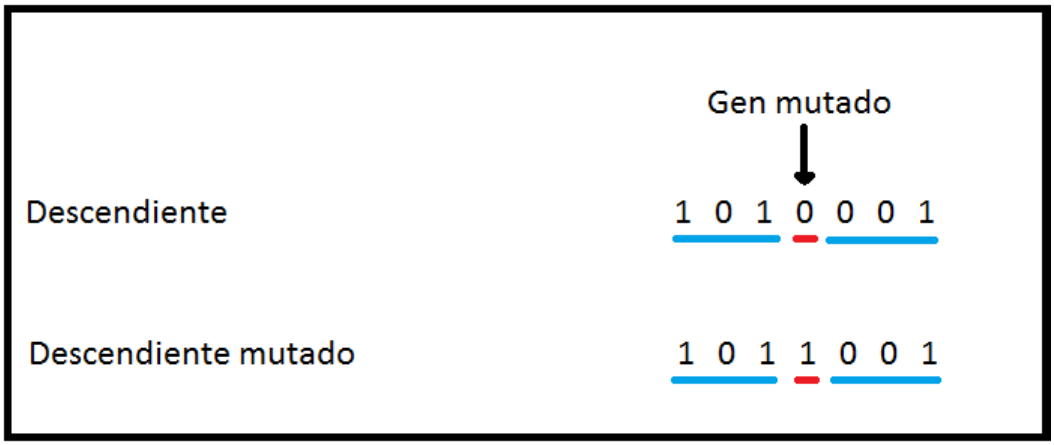

Figura 2.3: Mutación en el primer descendiente de la figura 2.2

La convergencia según De Jong [16] indica que si el algoritmo genético se implementó de manera correcta, la población evolucionará a través de las generaciones posteriores

de modo que la adaptación media será extendida a cada individuo, así mismo como la adaptación del mejor individuo irá en ascenso hacia el óptimo global. Este concepto se relaciona con la progresión hacia la uniformidad, que ocurre cuando un gen ha convergido cuando mínimo el 95% de individuos de una población, éstos tienen el mismo valor para ese gen. Una población converge cuando todos los genes han convergido.

La mutación es necesaria debido a que la selección y cruce no es tan crucial para llegar al valor óptimo [16]. La mayoría de implementaciones de algoritmos genéticos indican que la probabilidad del operador de cruce y mutación permanecen constantes.

2.7.4 Función de aptitud

Es aquella que representa la función objetivo del problema de optimización, se encarga de asignar valores reales a cada fenotipo y es la base para seleccionar a los individuos; la función será más útil en tanto discrimine más los valores [3], pues de este modo serán diferentes y descriptibles para facilitar las posteriores implementaciones en el algoritmo genético.

2.7.5 Mecanismos de selección

Una parte fundamental de los algoritmos genéticos es la selección de individuos en una población, para esto se requiere un mecanismo selectivo que permita efectuar dicha acción. Afortunadamente existen distintos mecanismos de selección que permiten facilitar esta tarea dependiendo de la solución que se esté empleando y del valor óptimo que se esté buscando. Existen algunos mecanismos selectivos que se emplean individualmente, mientras que otros pueden combinarse [5]. Es importante mencionar que todos los métodos de selección tienen un nivel de complejidad y se encuentran dentro del esquema de tres grupos que son la selección proporcional, selección mediante torneo y selección de estado uniforme.

Selección proporcional

Describe a un grupo de mecanismos de selección propuestos por John Holland. Se enfoca en elegir a los individuos con base en la contribución de su aptitud con respecto a la población total. Las técnicas de selección que abarca son lo siguientes [7]:

- **Ruleta**: Fue propuesta por De Jong en 1975, en esa época era el más utilizado en los inicios de los algoritmos genéticos. Esta técnica es simple pero no muy eficiente ya que un individuo puede ser seleccionado más de una vez y su complejidad es de $O(n^2)$.

- **Sobrante estocástico**: Su intención aproximarse a los valores esperados de cada individuo y se asigna de manera determinística las partes enteras de los valores que se esperan en cada individuo.

  Existen dos variantes del sobrestante estocástico que son con reemplazo y sin reemplazo. En la primera se construye la ruleta con partes decimales y se usa en la selección de los padres faltantes; su nivel de complejidad es de O(n$\hat{2}$).
  En la segunda se invierte el valor con las partes decimales y así se escogen los padres restantes; su nivel de complejidad es de $O(n)$.

- **Universal estocástica**: Su objetivo es reducir la mala distribución de los individuos con base en los valores esperados, sin embargo, puede presentar convergencia prematura o hacer que los individuos más aptos se multipliquen mucho más rápido. Su nivel de complejidad es $O(n)$.

- **Muestreo determinístico**: Requiere de una asignación determinística a la parte entera del valor esperado, después ordena la población de mayor a menor según las partes decimales y obtiene los padres faltantes. Al igual que el sobrante estocástico, el individuo puede ser seleccionado más de una vez. Su nivel de complejidad varía en la asignación y la ordenación, donde es $O(n)$ y $O(nlogn)$ respectivamente.

- **Escalamiento sigma**: Fue ideada para el mapeo de la aptitud original de un individuo con su valor esperado, es decir, mantener constante la presión de selección durante el proceso de evolución y así reducir la probabilidad de convergencia prematura en el algoritmo. En este mecanismo el valor esperado de un individuo se encuentra en su función de aptitud, la media de la población y la desviación estándar de la misma.

- **Selección de jerarquías**: Su objetivo es la prevención de convergencia prematura. Su complejidad es de $O(nlogn)$ más el tiempo de selección, lo que causa una convergencia más lenta.

- **Selección de Boltzmann**: Utiliza una función de variación de "temperatura" para controlar la selección. Al inicio el valor de ésta debe ser alto para que la presión en la selección sea más baja, pero al paso de las generaciones se van invirtiendo los roles, dejando así a la temperatura con un valor más bajo y la selección con uno más alto.

Selección mediante torneo

Es similar al uso de jerarquías en cuanto a la presión para selección, pero es más eficiente al computarizar y paralelizar. La idea básica de esta es seleccionar a los individuos con base en una comparación directa [7]; hay dos formas de hacerlo, de manera determinística y probabilística.

- **Determinística**: Se tienen que "barajar" los individuos de la población, elegir un número "p" de individuos (usualmente son 2), comparar sus aptitudes y el ganador del torneo se tomará como el individuo más apto. Para obtener N número de padres, es necesario "barajar" P veces la población.

- **Probabilística**: Implica los mismos pasos que la determinística, a excepción de la velocidad en que se elige al ganador. En lugar de seleccionar al individuo con la mayor aptitud, se aplica una inversión a la probabilidad "flip(p)" y si el resultado

es verdadero, se selecciona el más apto y si no, entonces al menos apto. El valor de la probabilidad "p" permanece fijo y va de 0.5 a 1.

La versión determinística asegura que el mejor individuo será seleccionado "p" veces. Cada competencia requiere de una selección aleatoria de un número constante de individuos en la población. Dado que se requieren "n" competencias [7] de comparación de individuos en tiempo constante para completar una generación, el algoritmo es de complejidad O(n).

La selección por torneo es eficiente y de fácil implementación, la comparación en la función de aptitud es directa y la presión de selección puede ser muy alta permitiendo descartar a los individuos menos aptos más fácilmente.

Selección de estado uniforme

Se usa en los algoritmos genéticos no generacionales, en donde sólo los individuos menos aptos son reemplazados en cada generación. Se aplica mayormente en sistemas basados en reglas donde el aprendizaje es incremental [7], como los sistemas clasificadores. Dicha técnica es útil en entornos donde los individuos de manera colectiva buscan solucionar un problema y no individualmente, además si es necesario "recordar" lo aprendido antes, es también una buena opción. Su complejidad es $O(nlogn)$.

El algoritmo de implementación de esta técnica requiere de seleccionar "R" individuos de una población "G" de entre los más aptos (1¡= R ¡=M), donde usualmente "R = 1" O "R = 2". Posteriormente se realiza la cruza y mutación a los R individuos para obtener "H" hijos, de los cuales se obtendrá el más apto o aptos en cuyo caso serán hijos "S", finalmente se reemplazan los peores individuos de "G" por "S" [7].

- **Selección más (+)**: Usualmente utilizada en problemas de optimización global. Consiste en unir las poblaciones, las padres con las de hijos y seleccionar la mejor mitad de esta.

- **Brecha generacional**: Es la cantidad de "traslape" que exista entre padres e hijos. Es similar a la selección de estado uniforme, pero en esta técnica las

selecciones pueden ser "traslapables" y "no traslapables". Esa primera indica que los padres van a competir con los hijos. La segunda implica un reemplazo total de los padres por los hijos.

Otras técnicas de selección

- **Selección elitista**: La selección de individuos hacia la siguiente generación está asegurada; algunos algoritmos genéticos no usan elitismo puro, sino un tipo modificado de éste.
- **Selección escalada**: Si se incrementa la aptitud media de una población, entonces la selectividad será mayor y la función de aptitud más discriminatoria.
- **Selección por rango**: Cada individuo tiene asignado un rango numérico basado en su aptitud y la de sus competidores, lo que impide a los más aptos abarcar la dominancia sobre los menos aptos, por lo que se efectúa una reducción en la diversidad genética poblacional que causa dificultades en la búsqueda de una solución aceptable [5].
- **Selección disruptiva**: Utilizada para normalizar aptitudes con respecto a cierto valor moderado. Esto se logra a través la distribución del esfuerzo al buscar las mejores y peores soluciones para de esta manera deshechar a los individuos cuya aptitud se acerca a la media [7].

2.7.6 Estructura general de un algoritmo genético

Para construir un algoritmo genético básico o canónico, se va a requerir de ciertos elementos como ha mencionado anteriormente. Para una mayor compresión de la estructura general, se utilizará el pseudocódigo como se muestra en la figura 2.4.

Como se puede ver, un AG siempre va a requerir de una población inicial que puede ser generada al azar en la cual cada individuo deberá tener una aptitud asignada, es decir, se debe efectuar la función de evaluación para cada uno. Como

```
BEGIN /* Algoritmo genético simple */
    Generar la población inicial
    Calcular la función de aptitud de cada individuo
    WHILE NOT Terminado DO
    BEGIN
        FOR Tamaño de la población / 2 DO
            BEGIN /* Inicia el ciclo reproductivo */
                Seleccionar dos individuos de la generación anterior para
                cruzarlos (probabilidad de selección proporcional a la función
                de aptitud del individuo).
                Cruzar con cierta probabilidad ambos individuos para obtener
                dos descendientes.
                Mutar ambos descendientes
                Calcular la función de aptitud de los descendientes mutados
                Insertar dichos descendientes en la nueva generación
            END
        IF La población ha convergido THEN
            Terminado := TRUE
    END
END
```

Figura 2.4: Pseudocódigo del algoritmo genético simple [16]

los algoritmos genéticos se basan en la evolución humana, entonces se van a requerir diversas generaciones [2] para llegar el óptimo; visto de otro modo, se deberá repetir el proceso una y otra vez hasta llegar a éste. Si no se ha conseguido el valor óptimo, se deberá generar una nueva generación y entonces se aplicarán "n" veces el tamaño de la población la selección de individuos para ejecutar la cruza, después mutar a los descendientes e insertarlos en una nueva generación. Al término de esto se debe comprobar si la población ha convergido para entonces finalizar el AG.

2.8 Tecnologías involucradas

En esta sección se explican brevemente las tecnologías y herramientas más relevantes que se usaron e involucran con el sistema desarrollado.

2.8.1 Android

Historia

En 2003, Android Inc. fue fundada por Andy Rubin, Rich Miner, Nick Sears y Chris White con el objetivo de crear dispositivos móviles "al corriente y preferencia del usuario". Aunque la idea inicial era el desarrollo de un sistema operativo avanzado para cámaras digitales, se cambió el enfoque a un sistema capaz de competir contra Symbian y Windows Mobile, es decir, un sistema operativo para teléfonos móviles [19].

En 2005, Google compró Android Inc. y se inició el desarrollo de una plataforma móvil basada en el kernel de Linux para así crear un sistema similar al de BlackBerry que utilizaba el teclado "QWERTY", sin embargo, para ese entonces siendo ya 2007, el iPhone había sido lanzado al mercado teniendo como innovación la pantalla táctil, por lo que Google se vio en la necesidad de hacer lo mismo. Como dato curioso el logo y mascota de Android fue diseñado por Irina Blok, una diseñadora gráfica que creó a Andy, el robot o "bugdroid" de color verde basándose en un personaje de un videojuego de los años 90.

El 23 de septiembre de 2008 se lanzó el primer teléfono con sistema operativo Android, el HTC-Dream / T-Mobile G1. Conforme pasaba el tiempo se empezaron a sacar nuevas versiones y subversiones de dicho sistema [19].

Versiones

Android, desde su lanzamiento en 2008 y hasta la actualidad ha traído consigo versiones y subversiones mejorándolo y agregando nuevas características con cada una de ellas.

- **Android 1.0**: Lanzada en 2008. Utilizaba el teclado físico, contenía algunos de los bloques de desarrollo que hasta la actualidad se siguen utilizando, incluía Google Maps, pestaña de notificaciones, Android Market, integración de Gmail, navegador web, calculadora y reloj. Esta primer versión no traía un "postre" como lo sería con los lanzamientos y actualizaciones posteriores [19].

- **Android 1.1 (Petit Four)**: Única actualización para la versión 1.0 en la que se corregían errores. Algo interesante para ese entonces fue que Android era el único sistema que podía utilizar "over-the-air", que es una característica para actualizaciones por medio de Wi-Fi.

- **Android 1.5 (Cupcake)**: El 27 de abril de 2009 llegó la versión 1.5 de Android, trayendo consigo la tradición de nombrar a las versiones mayores con el nombre de un postre y por orden alfabético. El cambio más importante fue quizás la inclusión del teclado virtual y la capacidad de implementar *widgets* de otras aplicaciones.

- **Android 1.6 (Donut)**: Lanzado el 25 de septiembre de 2009. Un cambio muy importante fue la barra de búsqueda sin la necesidad de abrir el navegador web. También se implementaron mejoras de Android Market, adaptación del Sistema Operativo (SO) a nuevos tamaños y resoluciones de pantalla, sintetizador de voz multilenguaje, cámara y galería mejoradas, se agregó el soporte para redes CDMA y conexiones VPN [19].

- **Android 2.0 (Eclair)**: En octubre 26 de 2009 se lanzó esta versión con nuevas características como las rutas en Google Maps, la posibilidad de multicuentas en el dispositivo y sincronización de cuentas de terceros como Facebook. El diseño de aplicaciones como el navegador web, calendario o teclado virtual mejoraron.

- **Android 2.0.1**: La primera actualización para Android 2.0, lanzada el 4 de diciembre de 2009 trajo consigo corrección de errores.

- **Android 2.0.2**: La segunda actualización para Android 2.0, incluyó mejoras y una revisión de la "Application Programming Interface" (API) que es un conjunto de herramientas para una versión específica de Android, lo que permitía a los fabricantes personalizar el sistema, con esto nació el programa Nexus.

- **Android 2.2 (Froyo)**: Lanzada en mayo de 2010 y con dos cambios muy importantes que eran el soporte para comandos de voz y la posibilidad de creación de puntos de acceso Wi-Fi. Otra característica fue sobre el navegador web, que mejoró con la inclusión de soporte del motor v8 de Javascript de Google Chrome, soporte para subir archivos, visualización de GIF animados y soporte para Adobe Flash. Se implementaron las notificaciones push [19].

- **Android 2.2.1, 2.2.2 y 2.2.3**: Son las tres actualizaciones para Android 2.0. Se centraban principalmente en la corrección de errores y parches de seguridad.

- **Android 2.3 (Gingerbread)**: Fue lanzada el 6 de diciembre de 2010. Dado que el sistema comenzaba a madurar, nuevas y mejores implementaciones debían mostrarse como la API para juegos, el soporte para tecnología "Near Field Communication" (NFC), visualización del uso de batería, mejoras en la interfaz y la tradición de los huevos de pascua. Esta tradición comenzó como una broma entre lo desarrolladores, el primero fue la imagen de un zombie que se mostraba tras tocar repetidas veces la versión de Android en las opciones del dispositivo [19]. Otras características muy importantes fueron el soporte para diversas cámaras como la vista frontal y soporte nativo para otros sensores como el giroscopio y el barómetro.

- **Android 2.3.1, 2.3.2 y 2.3.3**: Son tres de las siete actualizaciones en las que sólo se hizo corrección de errores.

- **Android 2.3.4**: Una actualización importante sin duda, pues gracias a que Android soportaba más de una cámara, Gingerbread implementó las videollamadas en Hangouts.

- **Android 2.3.5, 2.3.6 y 2.3.7**: Las últimas tres actualizaciones para Android 2.3 se enfocaron en cambios y correcciones menores antes de la siguiente versión.

- **Android 3.0 (Honeycomb)**: Esta versión cuyo lanzamiento fue el 22 de febrero de 2011 fue únicamente para tabletas electrónicas. Se introdujo la "System bar" similar a la barra de tareas de Windows en la que se podía ver la hora, fecha, batería restante y estado de conexión. Dado que Honeycomb se enfocó tabletas, se hicieron cambios a diversas aplicaciones: el navegador web tenía pestañas, modo incógnito y relleno de formularios, en cuanto a los contactos y correo electrónico, se incorporó la doble columna para su vista. El huevo de pascua de Honeycomb era una abeja androide [19].

- **Android 3.1 y 3.2**: Estas actualizaciones se enfocaron en la revisión de últimos cambios a la versión actual de Android como permitir las aplicaciones el acceso a la memoria SD.

- **Android 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5 y 3.2.6**: Actualizaciones enfocadas en la corrección de errores.

- **Andorid 4.0 (Ice Cream Sandwich)**: El 18 de octubre de 2011 se lanzó la versión 4.0 de Android. Algunas de las características que eran únicas de las tabletas utilizando Android 3.0 fueron implementadas en los teléfonos móviles como la barra de navegación en la pantalla. El sistema adoptó el "Holographic Design" (Holo) que es el modo en que las aplicaciones se visualizaban. Muchas otras características se implementaron como la inclusión de carpetas y estadísticas de red, captura de pantalla al presionar los botones de encendido y volumen al mismo tiempo, estadísticas de uso totales de datos y de aplicaciones permitiendo mostrar el primer y segundo plano, se incluyó el desbloqueo facial y eliminado

individual de notificaciones. El huevo de pascua de Ice Cream Sandwich eran sándwiches de helado voladores [19].

- **Android 4.0.1, 4.0.2, 4.0.3 y 4.0.4**: Estas actualizaciones corregían errores y agregaban optimización y mejoras de rendimiento.

- **Android 4.1 (Jellybean)**: Lanzada el 9 de julio de 2012 y permite ver a "Google Now" nacer; abarca la API 16, 17 y 18 para desarrolladores. Se mejoró el rendimiento, se incluyó mayor accesibilidad para usar la lupa, deslizamiento con los dedos para hacer zoom a la pantalla. En esta versión ser permitían agregar widgets a la pantalla de bloqueo, y en la versión 4.3, siendo esta la última revisión, se implementó el soporte nativo para emojis y mejoras de rendimiento en OpenGL ES 3.0. El huevo de pascua de Jellybean eran frijoles dulces.

- **Android 4.4 (Kitkat)**: El 31 de octubre de 2013 llegó Kitkat, una de las más emblemáticas versiones de Android. Se cambió la estética de diseño de la interfaz añadiendo iconos más claros, transparencias. Una característica interesante era el modo inmersivo en el que la barra de estado y de navegación se ocultaban para mostrar totalmente una aplicación.
  Kitkat incluye "Android Runtime" (ART) para reemplazar a la máquina virtual de Dalvik. Las cuatro actualizaciones que recibió, de Android 4.4.1 a 4.4.4 se enfocaban en cambios menores, mejor compatibilidad, corrección de errores y parches de seguridad. El huevo de pascua de esta versión es un homenaje a versiones anteriores de Android [19].

- **Android 5.0 (Lollipop)**: El 25 de junio de 2014 fue dado a conocer esta versión de Android. Una característica muy importante fue la llegada de "Material Design" para varias aplicaciones. Se implementó una mejora en el rendimiento de la batería, búsqueda integrada en las opciones de configuración. El huevo de pascua de Lollipop es una versión clonada del juego "Flappy bird".

- **Android 5.1**: Agregada en marzo de 2015, y siendo la única actualización para

Lollipop. Implementó la protección antirobo y soporte para más de una tarjeta SIM.

- **Android 6.0 (Marshmallow)**: La sexta versión de Android fue lanzada el 5 de octubre de 2015. El enfoque principal para Marhsmallow era la de mejorar y concretar todo el sistema. Algo nuevo que Google implementó fue el modo "Doze" que consistía en poner a "dormir" las aplicaciones y reduce el rendimiento del CPU cuando el teléfono no se está utilizando y la pantalla esté apagada, esto con la intención de prolongar la duración de la batería. Otras características fueron el soporte USB-C, modo 4K para aplicaciones, modo multiventana y soporte para lector de huella dactilar. El huevo de pascua de esta versión era el mismo que en Lollipop pero con malvaviscos y multijugador [19].

- **Android 6.0.1**: Fue la única actualización lanzada el 7 de diciembre de 2015 para Marshmallow y en ella se incluía el soporte para emojis de Unicode 7.0 y 8.0, una nueva barra de navegación y la apertura de la cámara tocando dos veces el botón de encendido.

- **Android 7.0 (Nougat)**: Lanzada el 18 de mayo de 2016 y siguiendo el enfoque de Marshmallow hacia elementos que requerían atención. Se mejoró el modo Doze, implementación de un nuevo compilador "Just In Time" (JIT) que reducía tanto la instalación de una aplicación como el espacio en memoria. Nougat permitió que las aplicaciones pudiesen agregar botones a ajustes rápidos. El huevo de pascua de Nougat es un juego que consiste en atrapar gatos.

- **Android 7.1**: Se dio a conocer el 4 de octubre de 2016 trayendo consigo cambios para desarrolladores como el soporte para iconos circulares y actualizaciones del sistema A/B. Con la actualización 7.1.1 llegaron más emojis y envío de GIF por medio del teclado. La segunda actualización 7.1.2 agregó cambios a los dispositivos Pixel y Nexus.

- **Android 8.0 (Oreo)**: Lanzado el 21 de agosto de 2017, en donde la característica

más emblemática era el "Project Treble" para actualizaciones más rápidas. El modo "Picture in Picture" se implementó en teléfonos móviles dejando de ser exclusivo de Android TV. Una novedad importante fue la API de autocompletado de formularios la cual permitía incluirse en aplicaciones y no sólo navegadores web como lo era en un principio. El huevo de pascua de Oreo es un pulpo con la idea de hacer creer a los usuarios que esta versión sería "Octopus" [19].

- **Android 8.1**: La única actualización de Oreo, lanzada en diciembre de 2017 implementó algunos cambios visuales.

- **Android 9.0 (Pie)**: Tras cinco versiones beta de esta fue lanzada en agosto de 2018. La primer cosa que se podía ver respecto a Pie era la privacidad y el modo en que limitaba a las aplicaciones a usar la cámara de fondo. Otras características se enfocaban en la modernización de Android, como el brillo, la batería inteligente, navegación por gestos, el bienestar digital para gestionar el modo en que se utiliza el dispositivo [19].

- **Android 10 o Android Q**: Esta versión que fue lanzada en 2019, trajo consigo novedades como ser la primer versión sin un postre, pero con características más atractivas siendo una de estas el modo oscuro, gestión inteligente de energía con base en el uso de inteligencia artificial y el *machine learning*, también el "Live Caption" permitiendo agregar subtítulos automáticos al reproducir archivos de audio o video incluso sin conexión a internet, también el "Smart Reply" haciendo al sistema operativo capaz de reconocer el tipo de mensaje que se recibe y sugiriendo un tipo de respuesta acorde a ésta, más emojis, mejoras en el control por gestos y en el control de privacidad y Android 10 Go para teléfonos con 1.5GB de RAM [11].

- **Android 11**: En febrero de 2020 se lanzó la primer beta de esta versión. Las principales características de Android 11 son el anclaje de aplicaciones en el menú de compartir, reducción de preguntas constantes al conceder permisos a

una aplicación, el modo oscuro se puede activar a horas específicas de manera automática, mejoras de rendimiento y mejora en la sensibilidad táctil cuando se utilicen guantes [10].

Ciclo de vida de una aplicación Android

Una aplicación Android funciona a base elementos conformados por algo llamado Activity o Actividad, que son vistas las cuales muestran y ayudan a gestionar lo que aparece en la pantalla de la aplicación.

El sistema se va a componer de una pila de actividades previamente visualizadas de modo que el usuario pueda moverse de una a otra cada vez que lo desee. Android es sensible al ciclo de vida de una actividad, por lo que es importante conocer y saber manejar las distintas etapas por las que ésta puede pasar.

Una actividad tiene distintos estados [14] en el que se puede encontrar y va a depender de la interacción del usuario con esta el modo en que permanecerá dicha actividad. Los estados de una actividad son los siguientes:

- **Activa**: Es cuando la actividad se inicia y está al principio de la pila de actividades, por lo tanto se está ejecutando y puede ser visualizada por el usuario.
- **Pausada**: La actividad pasa a estar en pausa porque una o más actividades pasan a estar activas y por lo tanto, esta primera es visible pero no es la primera en la pila.
- **Detenida**: Cuando una actividad está en segundo plano porque el usuario cerró la aplicación, esta pasa a estar detenida y no es visible.
- **Destruida**: La actividad termina su función y es destruida por el sistema. Usualmente ocurre porque el usuario cerró totalmente la aplicación.

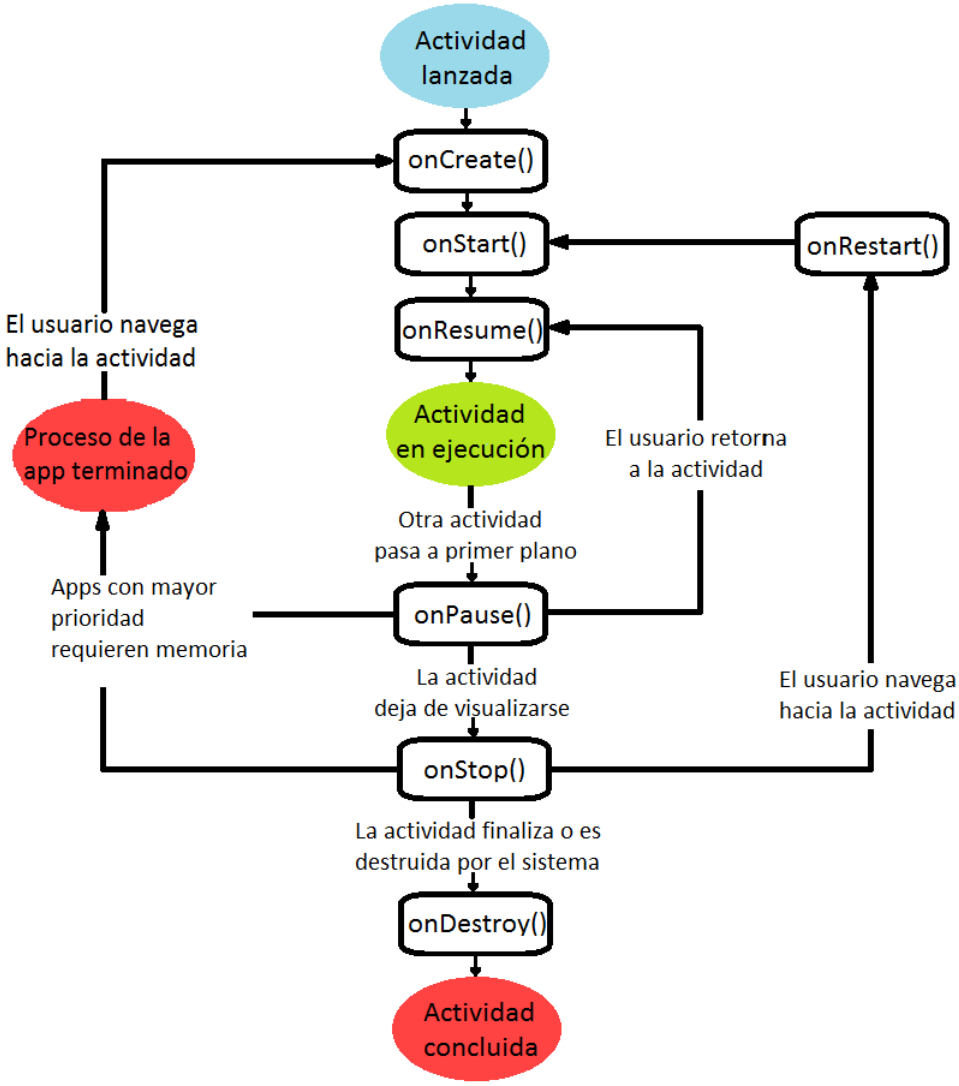

Figura 2.5: Ciclo de vida de una actividad en Android

La figura 2.5 muestra los métodos que se ejecutan dependiendo de su estado.

- **onCreate()**: Es el primero en ejecutarse al crearse la actividad, mayormente se usa para inicializaciones de datos y creación de la interfaz de usuario.
- **onStart()**: Es el segundo en ejecutarse y se encarga de indicar que la actividad está a punto de ser mostrada.
- **onResume()**: Se ejecuta cuando el usuario va a interactuar con la actividad.

- **onPause()**: La actividad está a punto de ser puesta en segundo plano debido a que otra actividad fue llamada.

- **onStop()**: La actividad deja de visualizarse. En ocasiones por falta de memoria este método no se llama y en su lugar simplemente se destruye.

- **onRestart()**: Se invoca cuando la actividad ya pasó por *onStop()* pero va a ser mostrada nuevamente.

- **onDestroy()**: Este método se invoca antes de que la actividad sea destruida.

Como se puede ver, los tres primeros métodos corresponden al estado "Activo" de la actividad, mismos que se ejecutan uno tras otro hasta que una nueva actividad sea creada y se invoca a *onPause()*, por lo que esta primera pasa a segundo plano dejando como principal a la nueva actividad, así hasta que se deje de visualizar, y ya sea que se retorne a la actividad inicial, se llamará al método *onStop()*, poniéndola en el estado "Detenido". Dependerá del usuario y su interacción con la aplicación que se invoque a *onRestart()* u *onDestroy()* para dejar a la aplicación en "Destruida", en cuyo caso se repite el ciclo a partir de *onStart()*, o bien la actividad finaliza totalmente.

Cabe mencionar que en un dispositivo en el que la memoria RAM no es suficiente para mantener abiertas varias aplicaciones a la vez y en segundo plano, aquellas con mayor prioridad (como las propias del sistema operativo) seguirán ejecutándose mientras que las de menor prioridad serán destruidas o por lo menos detenidas.

2.8.2 Android Studio

Configuración del entorno para desarrollo

Para comenzar a trabajar en una aplicación Android es necesario configurar nuestro entorno de desarrollo. Es importante definir qué tipo de lenguaje de programación será utilizado ya que con base en esto podemos saber cual IDE o herramienta es más favorable utilizar.

Esta tesis se enfoca en el uso del lenguaje Kotlin para desarrollar la aplicación, por lo que será necesario hacer uso del IDE Android Studio.

A continuación se listan los pasos para configurar el entorno de desarrollo en un sistema operativo Windows 7.

- Descargar el IDE Android Studio
- Instalar el archivo ejecutable descargado
- Tras finalizar la instalación, ubicar el programa y ejecutarlo como en la figura 2.6

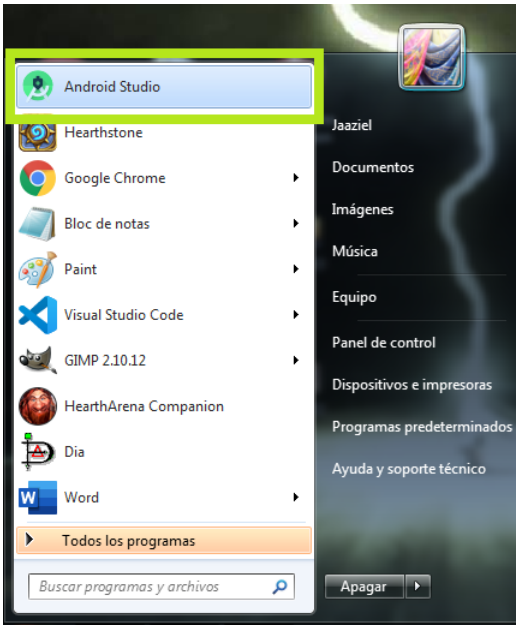

Figura 2.6: Abrir Android Studio

- Crear un nuevo proyecto como en la figura 2.7

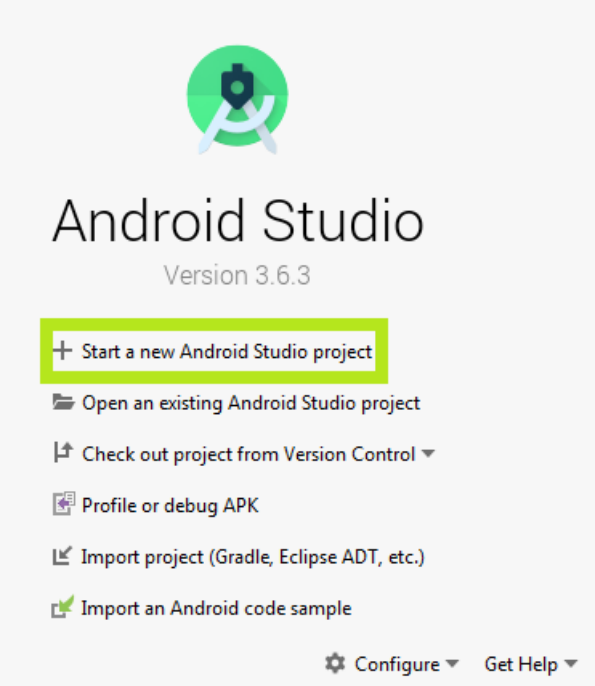

Figura 2.7: Crear proyecto

- Elegir una plantilla para nuestro proyecto

- Elegir API mínima para ejecutar nuestra aplicación

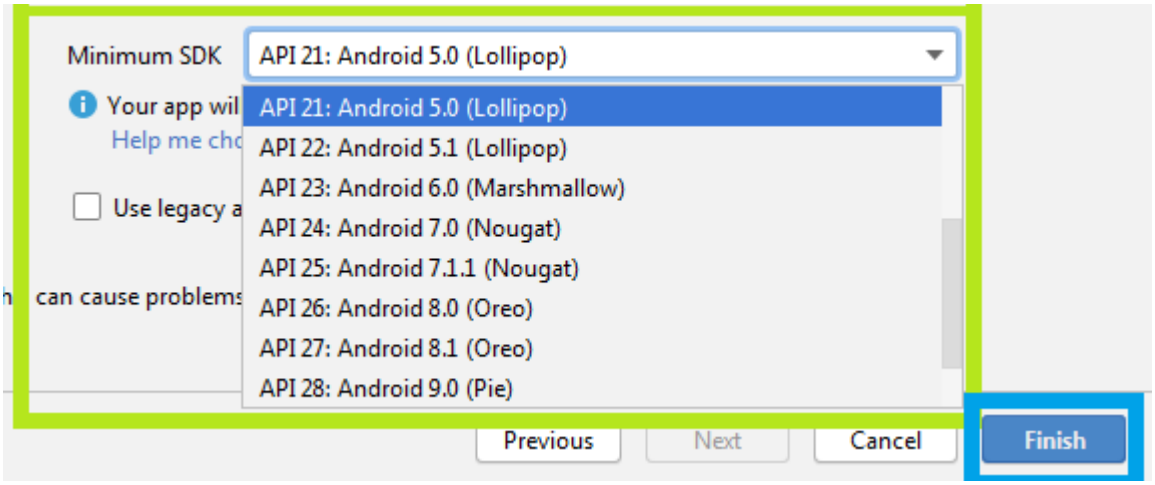

Figura 2.8: Elegir API objetivo

- Ir a "Tools" y abrir el "SDK Manager" (Software Development Kit) que se encarga de gestionar el kit de herramientas para el software.

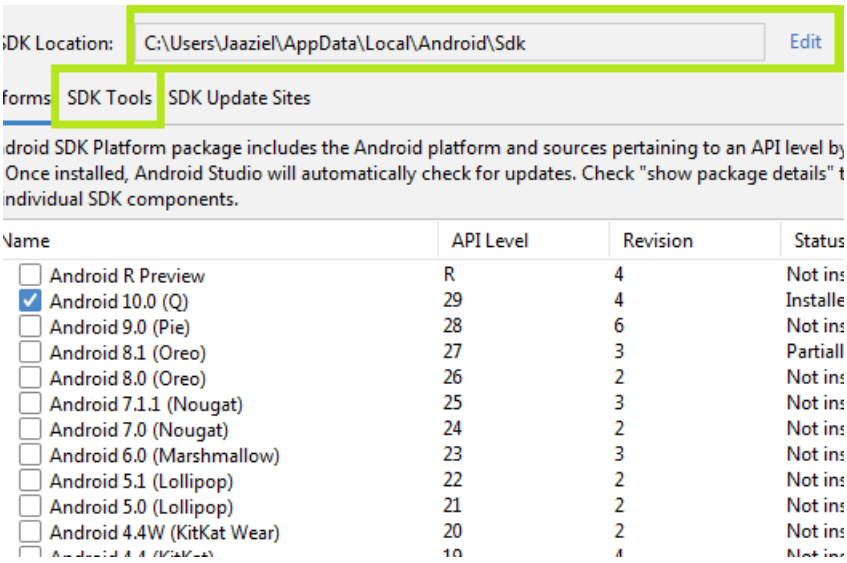

Figura 2.9: Abrir SDK Manager

- Hacer clic en "Edit" y ubicar el SDK para cargarlo. Usualmente la ruta base del archivo es la misma.

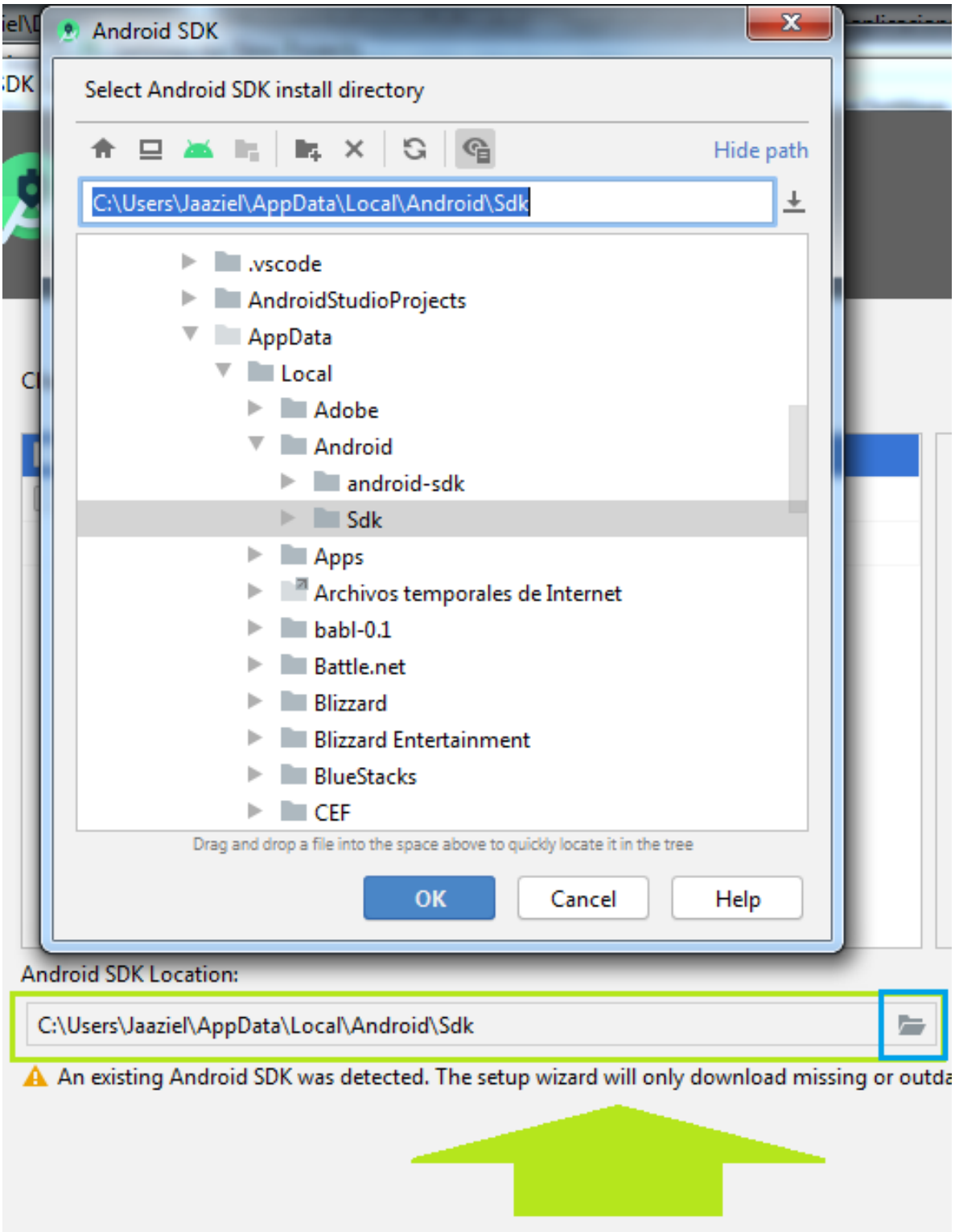

Figura 2.10: Cargar SDK

- Tras cargar el SDK es importante descargar las herramientas de desarrollador como se muestra en la figura 2.11

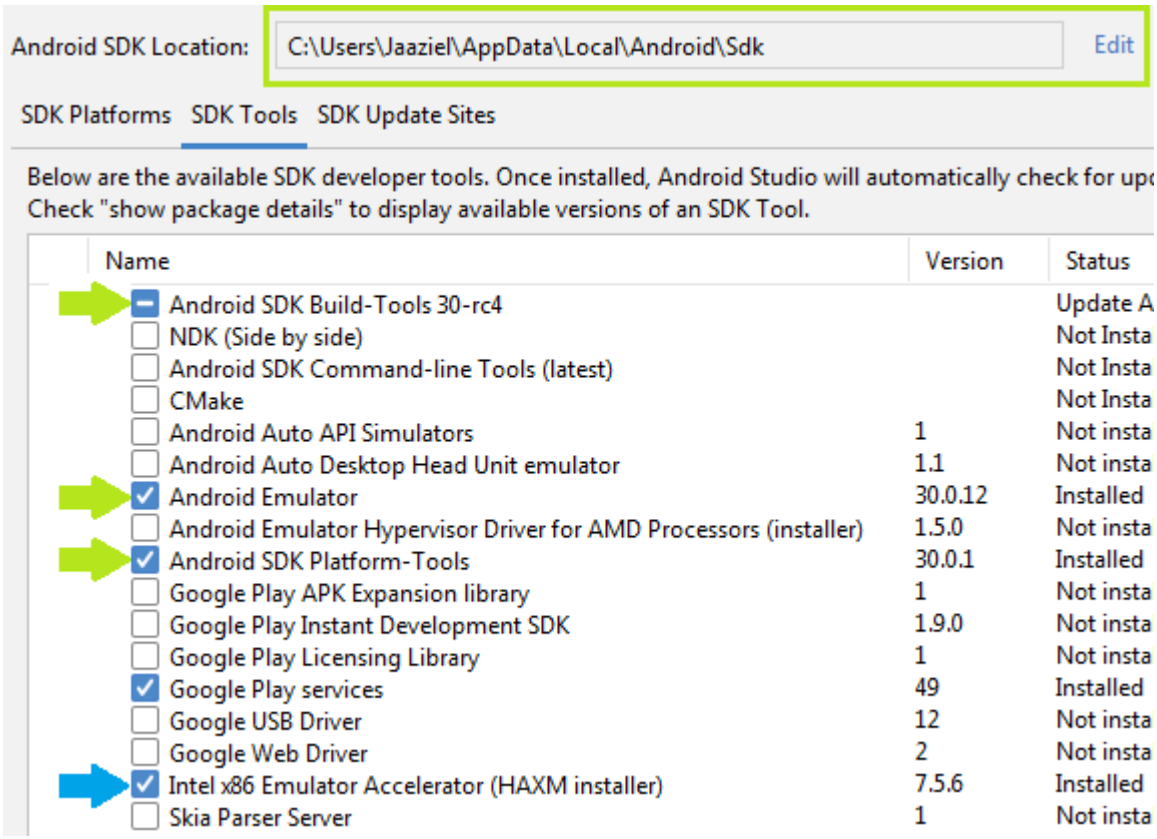

Figura 2.11: Herramientas de desarrollador

Cuando se crea un proyecto nuevo, Android Studio genera dos archivos básicos para trabajar llamados *MainActivity.kt* y *activity_main.xml*, donde el primero es el que contendrá todo el código escrito por el usuario y el otro la vista con los elementos a mostrar en la aplicación.

La figura 2.12 muestra la estructura base de ambos archivos.

Para comenzar a insertar elementos de vista para la aplicación es necesario ir al archivo *activity_main.xml*.

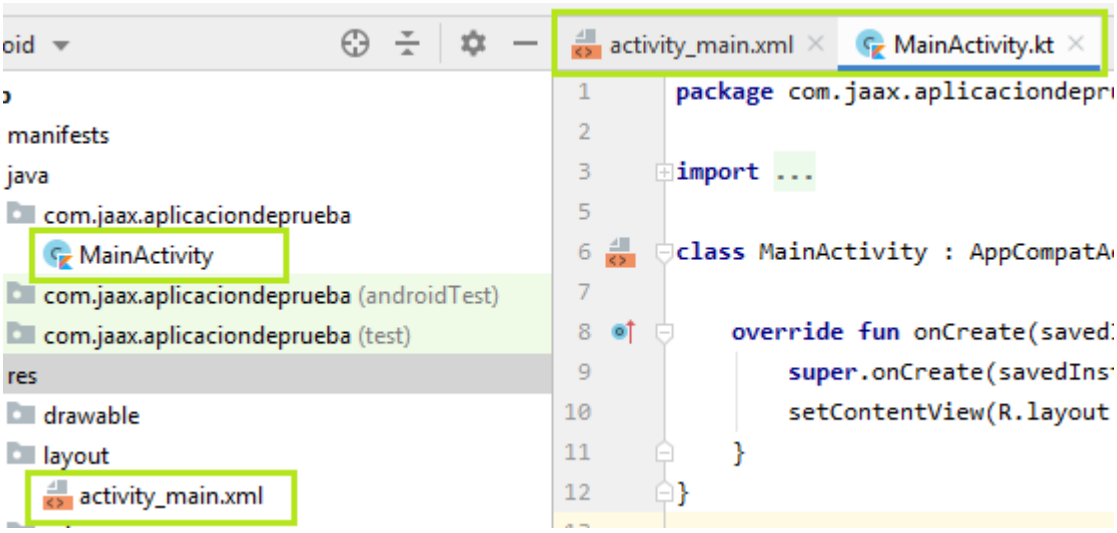

Figura 2.12: Archivos autogenerados

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.an
    xmlns:app="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tool
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/botonDePrueba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SALUDOS !"
        android:textColor="#000000"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true">
    </Button>
</RelativeLayout>
```

Figura 2.13: Botón agregado

Para hacer funcional el botón agregado como en la figura 2.13 se debe abrir *MainActivity.kt*

La figura 2.14 muestra la composición básica de una actividad desde código fuente, la línea 14 muestra la asignación del contenedor de vistas *activity_main.xml* a la clase, también como se encuentra el botón de la vista por medio de un identificador y el recuadro verde muestra el evento asignado a ésta, el cual mostrará una notificación de tipo "Toast" al ser presionado.

activity_main.xml | MainActivity.kt

```
3  import androidx.appcompat.app.AppCompatActivity
4  import android.os.Bundle
5  import android.widget.Button
6  import android.widget.Toast
7
8  class MainActivity : AppCompatActivity() {
9
10     lateinit var botonDePrueba: Button
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_main)
15
16         botonDePrueba = findViewById(R.id.botonDePrueba)
17
18         botonDePrueba.setOnClickListener { it: View!
19             Toast
20                 .makeText(
21                 applicationContext,
22                 text: "Saludos !, esta es una aplicación de prueba",
23                 Toast.LENGTH_LONG)
24                 .show()
25         }
26     }
27 }
```

Figura 2.14: Evento de botón

Para probar una aplicación en Android Studio se puede hacer uso del emulador que provee el IDE o por medio de un dispositivo físico utilizando un cable USB.
La figura 2.15 muestra la manera de ejecutar la aplicación creada en el emulador.

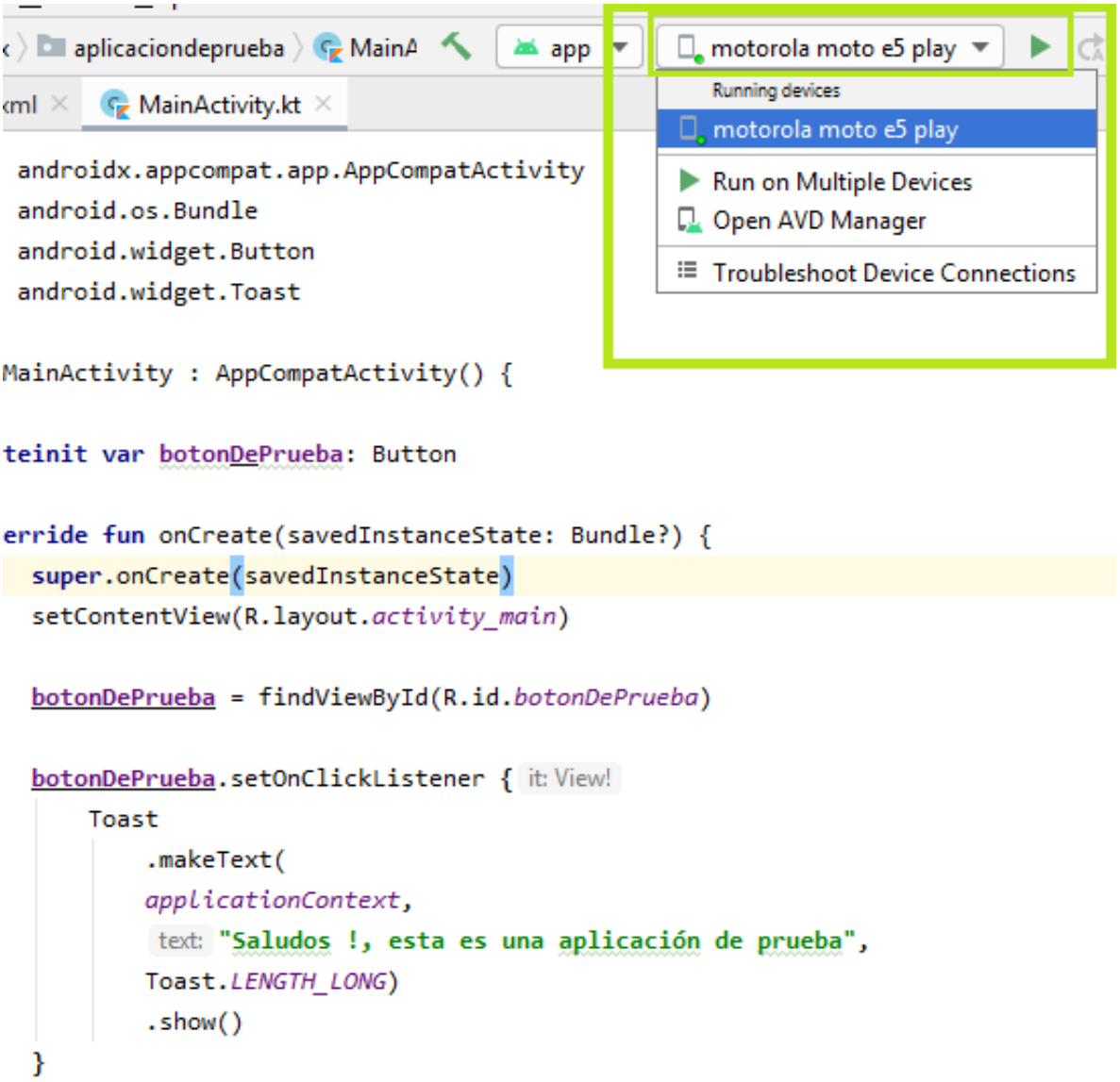

Figura 2.15: Iniciar aplicación

Figura 2.16: Aplicación en ejecución

La figura 2.16 muestra el mensaje de respuesta asignado al evento del botón.

2.8.3 Kotlin

Kotlin es un lenguaje de programación basado en la máquina virtual de Java (JVM). Fue creado en 2010 por la empresa Jetbrains [4], misma que desarrolló el IDE para Java Intellij IDEA y Android Studio.

La idea de crear un nuevo lenguaje basado en la JVM era la de resolver los problemas en Java, como por ejemplo, optimizar el número de líneas de código que se tienen que escribir para implementar una tarea. Lo que Jetbrains buscaba era reducir el tamaño de los códigos fuentes, y hacerlo interoperable con Java. Dado que en 2010 no había ningún lenguaje que posibilitara esto, optaron por mejorar Java creando un nuevo lenguaje llamado Kotlin.

Inicialmente Kotlin fue hecho para aplicaciones de escritorio haciendo uso del IDE Intellij IDEA; sin embargo, no se limitó a eso y se volvió multiplataforma con soporte para Android, compilación a Javascript y compilación nativa con LLVM para entornos donde la opción más óptima no sea la JVM.

Kotlin al ser una "mejora de Java" tiene ciertas ventajas de implementación:

- **Opcionales**: Es la declaración del tipo de variable que llevan el símbolo de interrogación (?) al final para volverla opcional.
- **Programación funcional**: Kotlin tiene la opción de trabajar con colecciones y sets de datos, es decir, se puede llamar a '.flatMap' directamente al igual que con '.map' o '.filter' por ejemplo.
- **Lambdas y expresiones de orden superior**: Se pueden definir funciones que reciban como parámetros otras funciones.
- **Nulabilidad de tipos**: Esta es posiblemente la característica más conocida de Kotlin. Al existir el soporte para opcionales, el número de posibles variables con valor nulo disminuye, y aunque puede haber la posibilidad de que se presente

alguno, se tienen dos alternativas: el operador *safe-call '?'* o el *safe-cast 'as?'* para evitar nulos al momento de hacer un casteo.

- **Enlazado de vistas**: Aunque el problema es propio de Android, Jetbrains provee *Kotlin Android Extensions* que es una librería oficial de Android que ayuda a simplificar dicho problema.

- **Verbosidad del lenguaje**: La cantidad de código que requiere Java para ejecutar un programa es más extensa que con Kotlin, mientras que el uso de lambdas y expresiones de orden superior simplifican esto.

- **Variables genéricas**: En Kotlin la declaración de una variable sólo puede iniciar con *var* para las variables cuyos valores pueden cambiar en tiempo de ejecución o *val* que corresponde a valores finales y no pueden cambiarse [20].

- **Funciones a nivel de clases**: Se puede definir una función sin tener que declarar una clase para ella.

- **Acceso directo a los atributos**: Se evita la redundancia al declarar *setter* y *getter* para acceder a los atributos de los objetos sin limitar o perder la característica de encapsulación de la POO.

- **Extensiones a clases existentes**: La declaración de funciones nuevas para una clase existente está permitida sin tener que crear una subclase.

- **Valores por defecto en funciones**: Kotlin permite asignar valores por defecto en funciones, por ejemplo, se puede llamar a la función con un sólo valor cuando la definición de esta tiene para recibir dos valores, esto evita la declaración de la función dos veces con diferentes parámetros.

```java
public class Ejemplo2 {
    int id;
    String nombre;
    String contenido;

    public Ejemplo2(int id, String nombre, String contenido){
        this.id = id;
        this.nombre = nombre;
        this.contenido = contenido;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getContenido() {
        return contenido;
    }
    public void setContenido(String contenido) {
        this.contenido = contenido;
    }
}
```

Figura 2.17: Definición de una clase en Java

```
class Ejemplo(
    val int: Int,
    val nombre: String,
    val contenido: String
)
```

Figura 2.18: Definición de una clase en Kotlin

En la figura 2.17 podemos observar una clase en Java que contiene tres atributos con sus respectivos *setter* y *getter*, que a diferencia de Kotlin (como se muestra en la figura 2.18) requiere de un menor número de líneas de código para efectuar la misma función.

2.8.4 Firebase

Otra de las tecnologías involucradas en este sistema e-learning es Firebase, que es una plataforma digital desarrollada por Google y que facilita el desarrollo de aplicaciones web y móviles a través de diversas funciones como:

- **Authentication**: Permite identificar a los usuarios por medio del correo electrónico o redes sociales.
- **Realtime Database (RDB)**: Es una base de datos no relacional para sincronizar los datos de los clientes conectados en tiempo real.
- **Storage**: Almacenamiento de archivos del usuario sin tener que usar código de servidores.
- **Hosting**: Alojamiento y despliegue para páginas web y APIs.
- **Crashlytics**: Análisis y generación de informe de errores en aplicaciones.
- **Performance**: Registro del uso de una aplicación conforme al número de usuarios.

- **Test Lab**: Espacio de pruebas de aplicaciones y juegos alojados en Google.
- **Cloud Messaging**: Envío de notificaciones segmentadas entre los usuarios para aumentar su interacción.
- **Remote Config**: Se emplea para modificar aspectos de un servidor para probar y personalizar el comportamiento de una aplicación
- **Dynamic Link**s: Facilita la inclusión de hipervínculos de aplicaciones incluso si no están instaladas en el dispositivo.

En las secciones posteriores se muestra con mayor detalle las funciones *Authentication* y *Realtime Database*

Comenzar con Firebase

Para una configurar e implementar adecuadamente esta tecnología, se deben seguir los siguientes pasos:

- Ir a la página de Firebase y loguearse con una cuenta de Gmail,
- añadir proyecto, seguir los sencillos pasos que te pide hasta llegar a una vista general como en la figura 2.19
- seleccionar la opción "Android", que posteriormente pedirá el nombre del paquete de nuestra aplicación y cambiará al paso 2 tras seleccionar "Registrar aplicación" (figura 2.20),
- descargar el archivo generado (*google-services.json*) y agregarlo a nuestro proyecto en Android Studio (figura 2.21),
- modificar el archivo *build.gradle(EjemploFirebase)* y *build.gradle(:app)* para implementar el SDK de Firebase como se observa en las figuras 2.22 y 2.23 respectivamente.

Al completar los pasos anteriores se redireccionará a la consola general de Firebase.

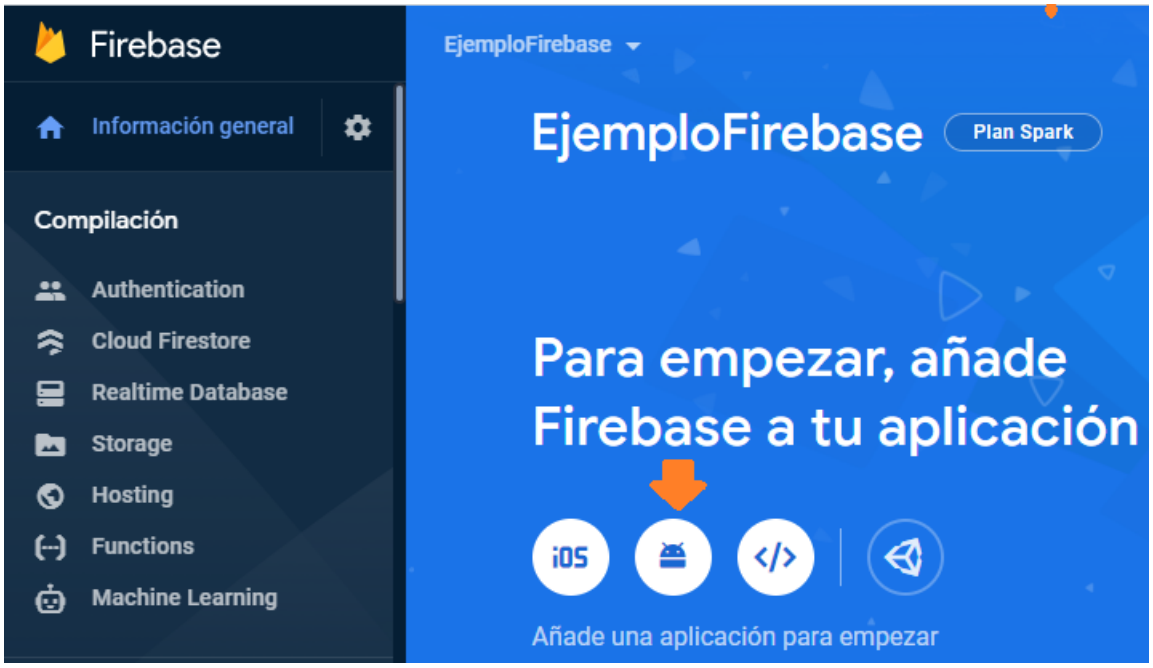

Figura 2.19: Nuevo proyecto

Figura 2.20: Nombre del paquete de la aplicación

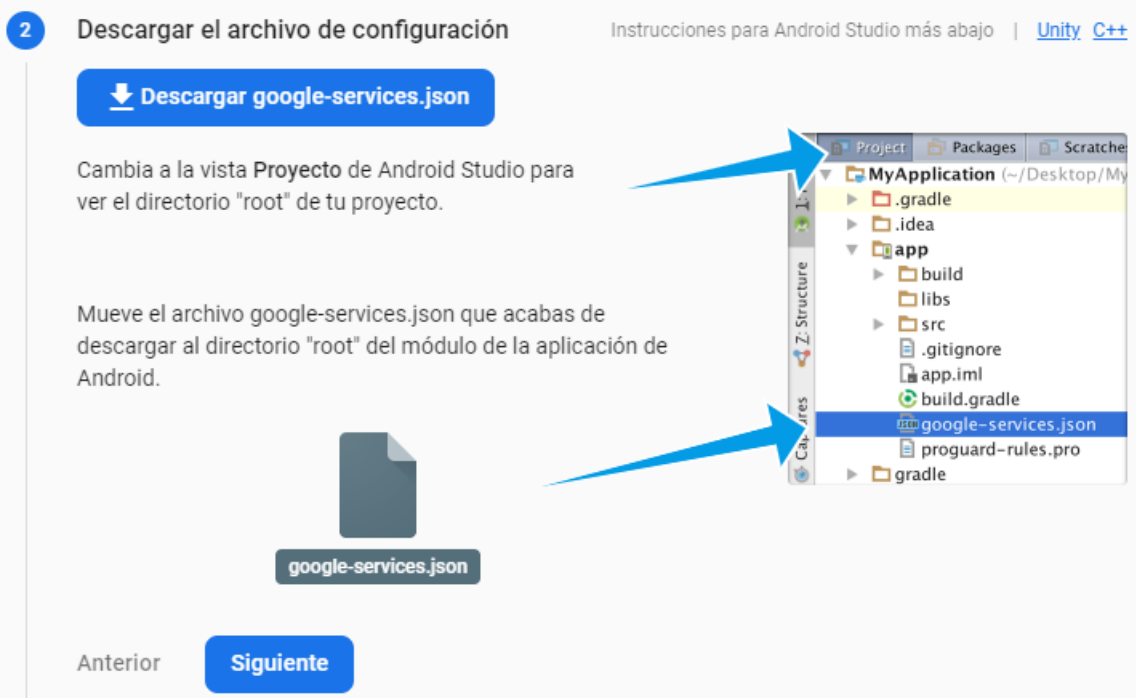

Figura 2.21: Archivo autogenerado por Firebase

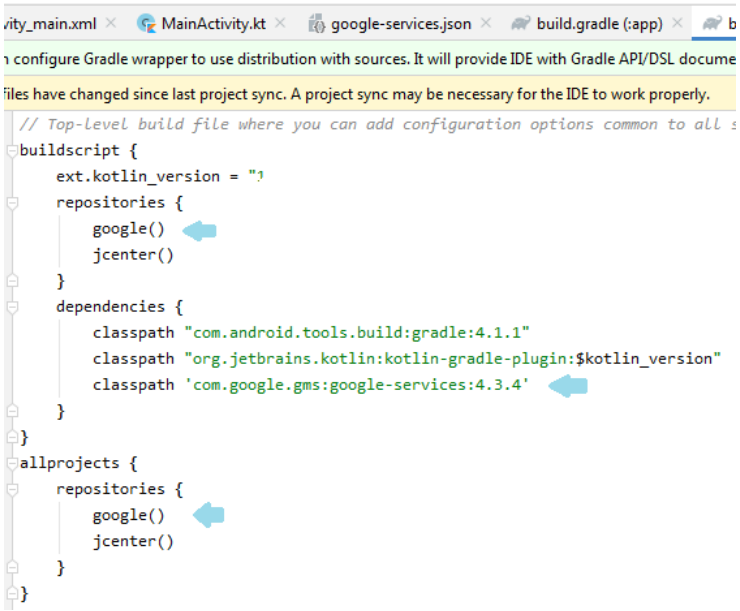

Figura 2.22: *build.gradle* con los repositorios de Google

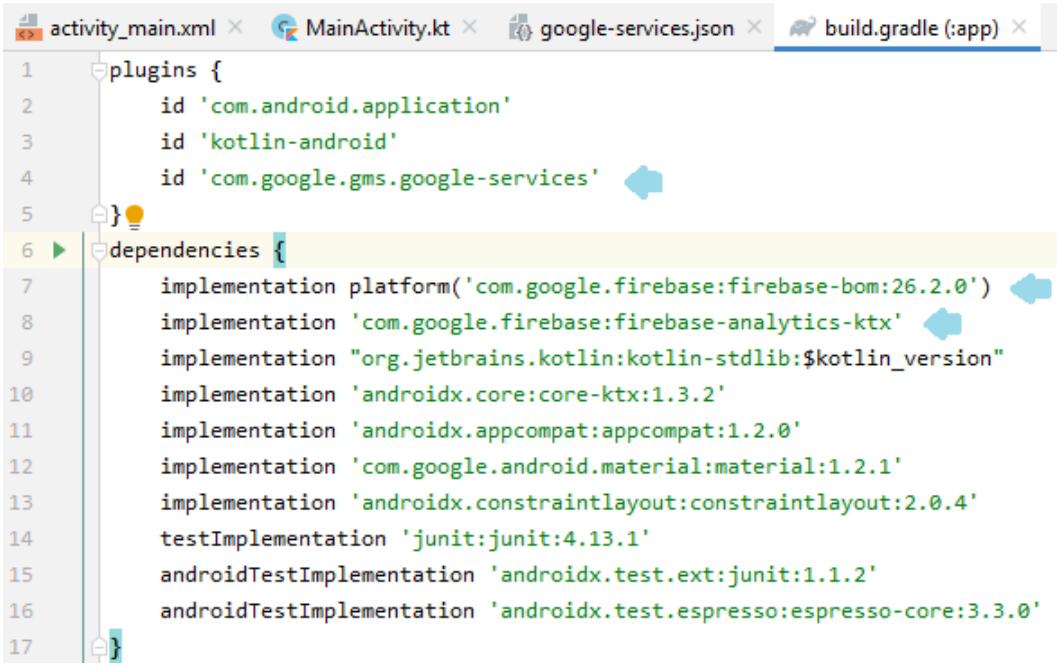

Figura 2.23: *build.gradle* con las dependencias requeridas

2.9 Scrum

Es un proceso que se aplica en el desarrollo de proyectos con la finalidad de agilizar, conseguir flexibilidad y mayor productividad en el flujo de trabajo a través de buenas prácticas [22]. Una de sus principales características es la división de la carga de trabajo por entregas parciales y regulares que poco en poco van construyendo el producto final.

2.9.1 Implementación

Scrum conlleva un proceso sencillo y puntual para efectuarse correctamente. Es de suma importancia conocer los pasos básicos requeridos para implementar dicha metodología según lo requiera el producto. Primeramente hay que definir a los usuarios.

- **Product owner**: Es el cliente, es quien se encarga de dar las especificaciones de cómo es que desea el producto final.
- **Scrum master**: Su función consiste en gestionar el tablero de Scrum con base en los requerimientos del cliente, se encarga de crear las historias y tareas además de asignarlas al equipo de trabajo.
- **Scrum team**: Son las personas encargadas de llevar a cabo las tareas que se generen. En este rol también puede incluirse al *Scrum master*.

Tras asignar los roles, se continúa con la planificación de sprints, que son plazos estimados de trabajo para completar las partes del proyecto.
Se sugiere hacer reuniones cortas cada cierto tiempo entre los usuarios para dar a conocer los avances, contratiempos o mejoras del producto y si se logró completar el sprint en tiempo y forma.
Los sprints abarcan otros elementos de Scrum que son las historias y las tareas [22]. Éstas primeras son funcionalidades generales del proyecto, mientras que las tareas reflejan aspectos técnicos de las historias.

Se sugiere asignar un valor de prioridad a las tareas para acelerar las entregas. Por ejemplo, si todas las tareas tienen la misma prioridad, puede que una parte esencial del proyecto no funcione porque no está completa debido a que otras tareas que debieron tener menor prioridad si lo están.

TABLA 2.2: EJEMPLO DE UN TABLERO BÁSICO DE SCRUM

Por hacer	**En proceso**	**Completada**
Tarea 7	Tarea 3	Tarea 1
Tarea 8	Tarea 4	Tarea 2
Tarea 9	Tarea 5	
Tarea 10	Tarea 6	
Tarea 11		
Tarea 12		
Tarea 13		
Tarea 14		
Tarea 15		
Tarea 16		

La tabla 2.2 representa de manera general como las tareas pueden avanzar de un estado a otro conforme se trabaja en ellas. Cabe mencionar que pueden haber más estados además de los básicos según sea requerido, por ejemplo, un estado llamado "testing" antes de completar la tarea, para hacer pruebas y verificación de tareas, entonces se mueve al estado de completada, o bien moverla a otro estado "con errores" para solucionar los fallos encontrados.

2.9.2 Scrum en AGtive

El desarrollo de la aplicación e-learning concluyó gracias a la implementación de Scrum. No se trabajó al pie de la letra en las especificaciones tradicionales que se plantean debido a que los tipos de usuarios que conlleva la planificación y requerimientos del producto recaían en una sola persona, pero en cierta manera fue una ventaja, pues las sugeridas reuniones para mostrar el avance y las asignaciones de historias y tareas se descartaban. En conclusión el uso de Scrum para desarrollar AGtive fue más que favorable.

Sprint 1

Con una estimación de tiempo de 3 semanas este sprint implicaba el modelado de datos y el diseño básico de las interfaces gráficas.

La tabla 2.3 representa el primer tablero de Scrum con las tareas de las historias del primer sprint, además de los estados en que éstas pueden estar. Como se puede observar, contiene dos estados más de los que comúnmente se implementan, esto debido a que se requirió de un mayor número de pruebas tras haberse presentado diversos errores, mismos que no se resolvieron de inmediato pues no todas las tareas tenían la misma prioridad.

TABLA 2.3: TABLERO DEL SPRINT 1

Por hacer	**En proceso**	**Con errores a resolver**	**En pruebas**	**Completadas**
Listar clases a utilizar				
Modelar clases usando UML				
Crear clases en Kotlin				
Hacer bosetos de todas las vistas				
Agregar vista de arranque				
Agregar vista de logueo				
Agregar vista general para el usuario				
Agregar vista general de un curso				
Agregar vista de datos del usuario				
Agregar vista de opciones de deslogueo				
Agregar vista de un temario				
Agregar vista general de cada parte del curso				

Sprint 2

El tiempo estimado de este sprint fue de 1 mes e implicaba el implementar Firebase en el proyecto y su codificación a la par de las vistas diseñadas en el primer sprint.
Se presentaron diversos errores que causaron ligeros contratiempos, los cuales se debieron a fallos en la configuración de los archivos base de AGtive, también por dificultades al conectar Firebase con Android y la falta de experiencia de uso con la función de Firebase Authentication.

TABLA 2.4: TABLERO DEL SPRINT 2

Por hacer	En proceso	Con errores a resolver	En pruebas	Completadas
Investigar acerca de Firebase				
Conectar Firebase con AGtive				
Conectar Api de logueo				
Hacer pruebas de logueo con gmail				
Codificar la transición de logueo a la vista principal				
Juntar opciones de usuario con datos obtenidos de logueo				
Codificar menús de la barra de usuario				
Hacer pruebas unitarias				

Sprint 3

La estimación de tiempo para este sprint fue de 1 mes y se planteó implementar la base de datos en el proyecto y parte del contenido del curso. En la tabla 2.5 se muestra con mayor detalle cada tarea realizada.

TABLA 2.5: TABLERO DEL SPRINT 3

Por hacer	En proceso	Con errores a resolver	En pruebas	Completadas
Investigar los requerimientos de Firebase Database				
Conectar Firebase Realtime Database con AGtive				
Realizar pruebas muy simples de CRUD en la BD				
Insertar datos del alumno en la BD				
Agregar opciones de reiniciar curso				
Conectar el reinicio de curso con la BD				
Implementar los primeras tres fragment de contenido del curso				
Investigar información extra sobre la representación entera				

El tiempo estimado fue suficiente para concluir en forma las tareas planteadas.

Sprint 4

Este sprint se enfocó en el desarrollo restante del curso de representación entera, es decir, la información como contenido de éste y el examen final con la finalidad de probar los conocimientos de el alumno respecto al tema presentado. Su tiempo estimado para fue de 3 semanas.

TABLA 2.6: TABLERO DEL SPRINT 4

Por hacer	En proceso	Con errores a resolver	En pruebas	Completadas
Diseñar primer quiz (fragment 4)				
Implementar lógica del primer quiz				
Implementar los fragment 5 y 6				
Diseñas segundo quiz (fragment 7)				
Implementar lógica del segundo quiz				
Implementar el fragment 8				
Implementar ejemplo de representación entera con Java (fragment 9 y 10)				
Implementar ejemplo de R.E con C++ (fragment 9 y 10)				
Implementar ejemplo de R.E con C# (fragment 9 y 10)				
Implementar ejemplo de R.E con Kotlin (fragment 9 y 10)				
Diseñar vista del examen				

Sprint 5

El sprint 5 corresponde a las últimas implementaciones del proyecto refiriéndose a la integración de resultados de los quiz y el examen en la base de datos que a su vez debía actualizar los datos del alumno ya sea que aprobase o no. Por otro lado, se planteó mejorar el diseño de la interfaz gráfica en general; agregar colores llamativos, formas en los elementos y animaciones, por mencionar algunos.

El tiempo estimado para concluir el sprint fue de 4 semanas, sin embargo, hubo retrasos en los tiempos debido a errores dentro del curso cuando se intentó modificar la base de datos, por lo que al menos una semana extra fue requerida para resolver dichas dificultades.

TABLA 2.7: TABLERO DEL SPRINT 5

Por hacer	En proceso	Con errores a resolver	En pruebas	Completadas
Implementar lógica del examen				
Modificar BD de Firebase por avance del alumno en quiz 1				
Modificar BD de Firebase por avance del alumno en quiz 2				
Modificar BD de Firebase por resultado del alumno en el examen				
Actualizar diseño de la vista de arranque				
Actualizar diseño de la vista general del usuario				
Actualizar diseño de la vista del curso de R.E				
Implementar opciones de administrador en el menú general				

Capítulo 3

Desarrollo

El desarrollo de la aplicación e-learning AGtive requirió del uso de las herramientas y tecnologías presentadas en el capítulo anterior. En este capítulo se presentan las funcionalidades del sistema, la estructura estática de éste y el comportamiento del mismo. Para ello, se usaron diagramas UML de casos de uso, diagramas de clases y de estados. Es importante mencionar que en este capítulo se desglosa conforme a la planificación de la metodología Scrum vista en el capítulo previo, pues con base en ella es que se logró el desarrollo.

3.1 Diagramas del sistema AGtive

Se utilizó un diagrama de casos de uso para mostrar de manera general (y muy sencilla) lo que el software hace, además de los "actores" que son las entidades que interactúan dentro y fuera sistema.

El diseño de la arquitectura de la aplicación comienza con un diagrama de clases de UML a fin de especificar las entidades involucradas en el sistema y representadas por las clases.

Por último, se implementó un diagrama de estados para dar a conocer la

funcionalidad de la aplicación con mayor profundidad cada una de las posibles acciones que éste puede efectuar cuando el usuario interactúa con ella.

3.1.1 Diagrama de casos de uso

Una manera sencilla de representar el funcionamiento y las tareas que se cumplen en un sistema es a través de un diagrama de casos de uso. El diagrama de casos de uso pra el sistema desarrollado se presenta en la figura 3.1.

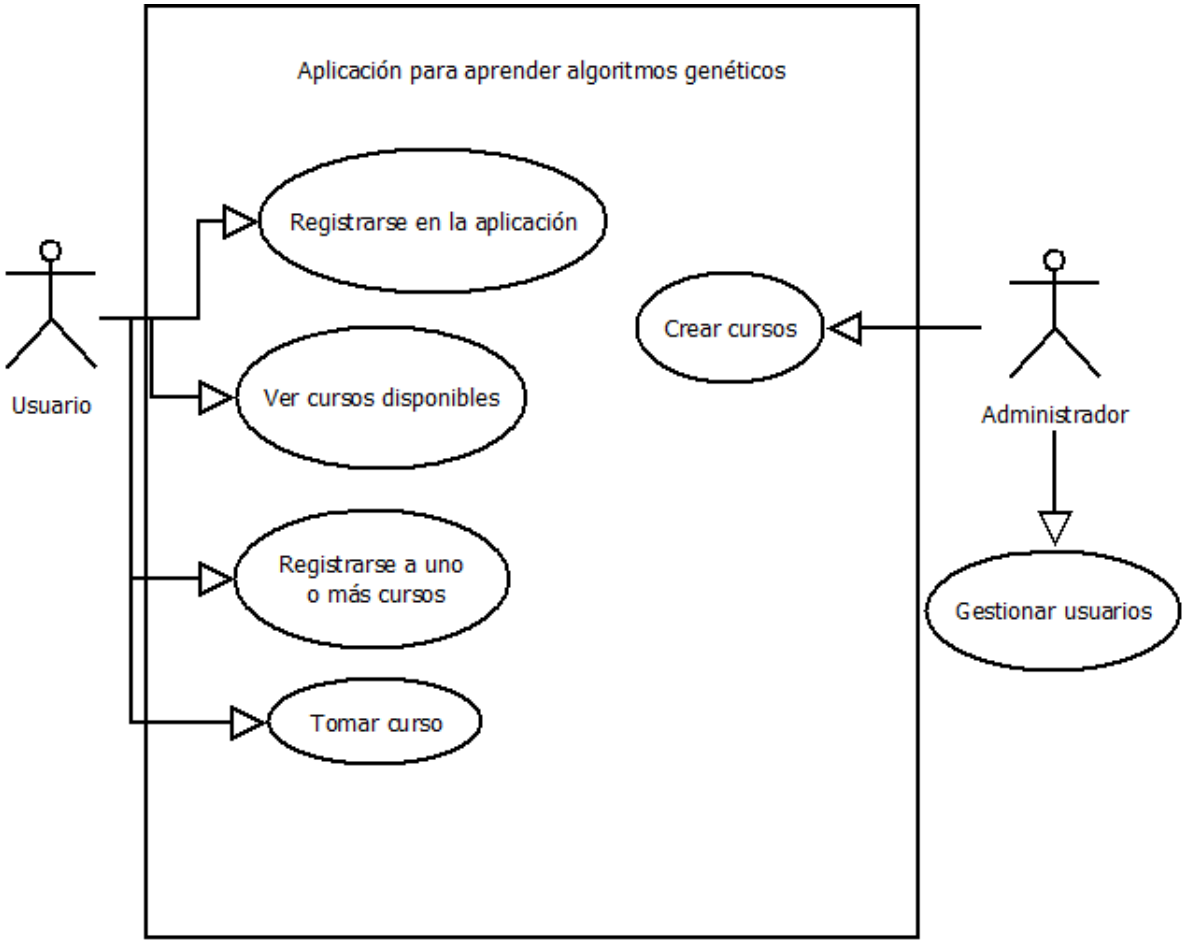

Figura 3.1: Casos de uso en AGtive

El usuario es quien puede realizar la mayoría de acciones, iniciando con el registro a la aplicación. Si elige convertirse en alumno al registrarse a un curso, entonces tiene acceso a otras acciones. El administrador, por otro lado, es un usuario que tiene control

dentro y fuera del sistema.

3.1.2 Diagrama de clases

Este tipo de diagrama UML permite explicar y entender la estructura estática de un sistema orientado a objetos. La figura 3.2 muestra el diagrama de clases del sistema desarrollado.

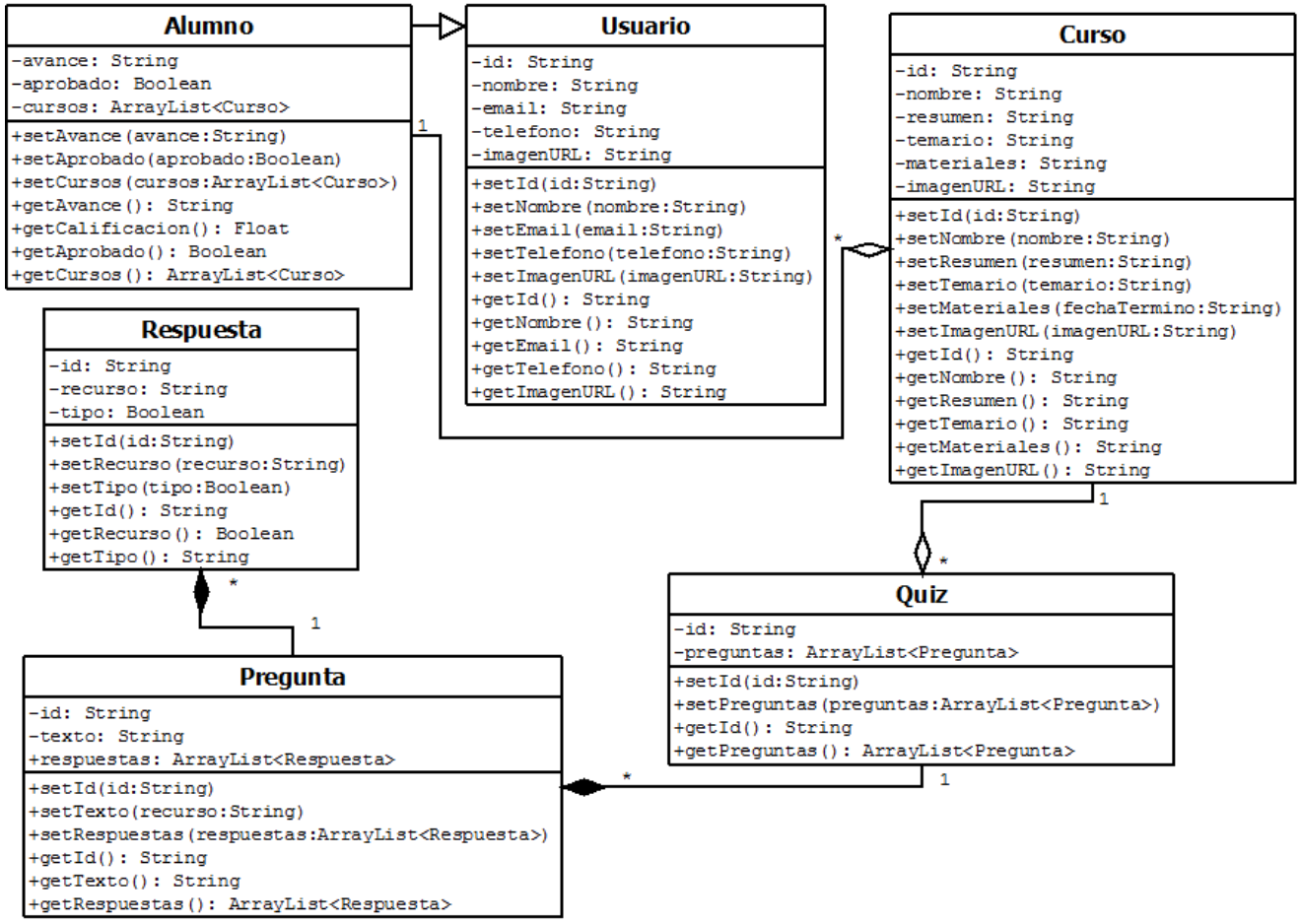

Figura 3.2: Diagrama UML

A continuación se describe cada una de las clases y su propósito.

- **Clase Usuario**: Esta clase modela a un usuario del sistema de e-learning. Entre los atributos principales figuran el *id* (un identificador único y propio de la cuenta de Gmail del usuario) y el *email* (usado para tener la referencia de Gmail en *Firebase Authentication*).

- **Clase Alumno**: Es una subclase de la clase Usuario. El atributo *avance* es de tipo String y se utiliza para indicar cuanto ha avanzado un alumno en un curso. Está también el atributo *aprobado*; que utiliza un valor Boolean para identificar si un alumno aprobó o no un curso. Finalmente están los *cursos* de tipo ArrayList que son aquellos a los que el alumno se ha registrado.

- **Clase Curso**: Esta clase modela los datos de un curso para el alumno. Sus componentes principales son el *id* (identificador único), un *nombre*, un *resumen* (descripción breve del contenido) y el temario.

- **Clase Quiz**: Es una clase que sólo contiene un *id* y las *preguntas* de la clase Pregunta.

- **Clase Pregunta**: Esta clase sirve para figurar las preguntas de los quiz y el examen en un curso. Contiene un *id*, un *texto* (la formulación de la pregunta) y una lista de *respuestas* de tipo Respuesta.

- **Clase Respuesta**: Esta clase modela los datos de una posible respuesta en una pregunta. Se compone de un *id*, un *recurso* (es el contenido, puede ser texto o una imagen) y el *tipo* de respuesta, ya sea correcta o no.

3.1.3 Diagrama de estados

Un diagrama de estados UML representa aquellas acciones que el usuario puede realizar, y los estados del sistema de acuerdo a las acciones que realiza el usuario. Las figuras 3.3, 3.4 y 3.5 muestran sus posibles estados.

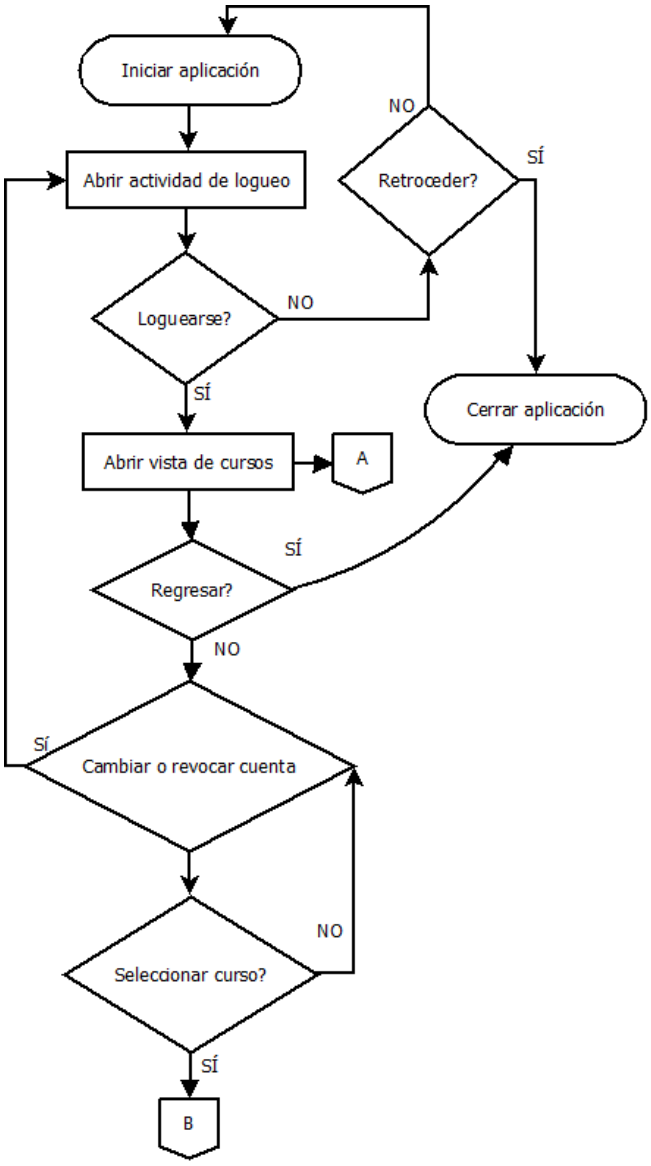

Figura 3.3: Parte 1 del diagrama de estados

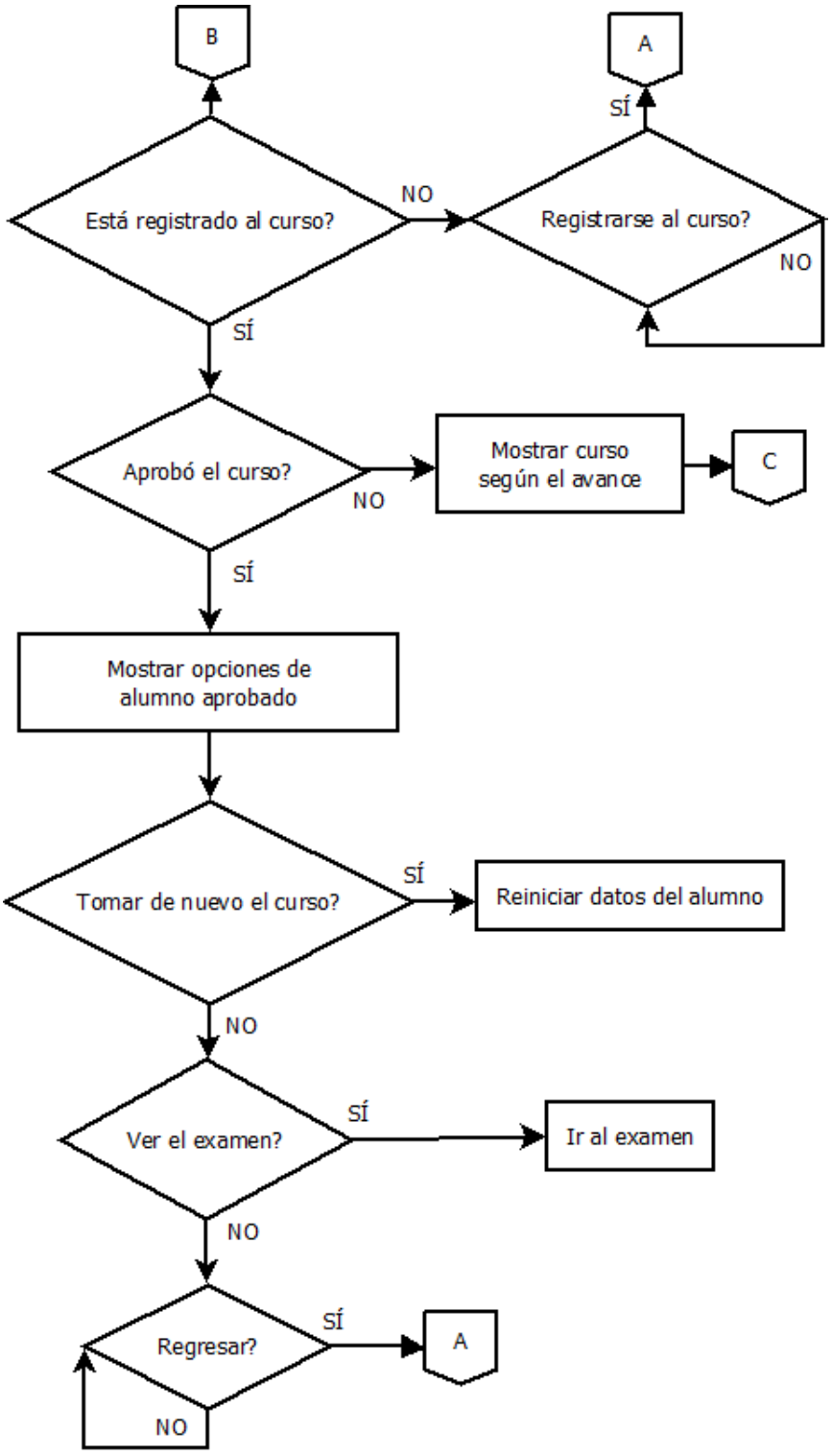

Figura 3.4: Parte 2 del diagrama de estados

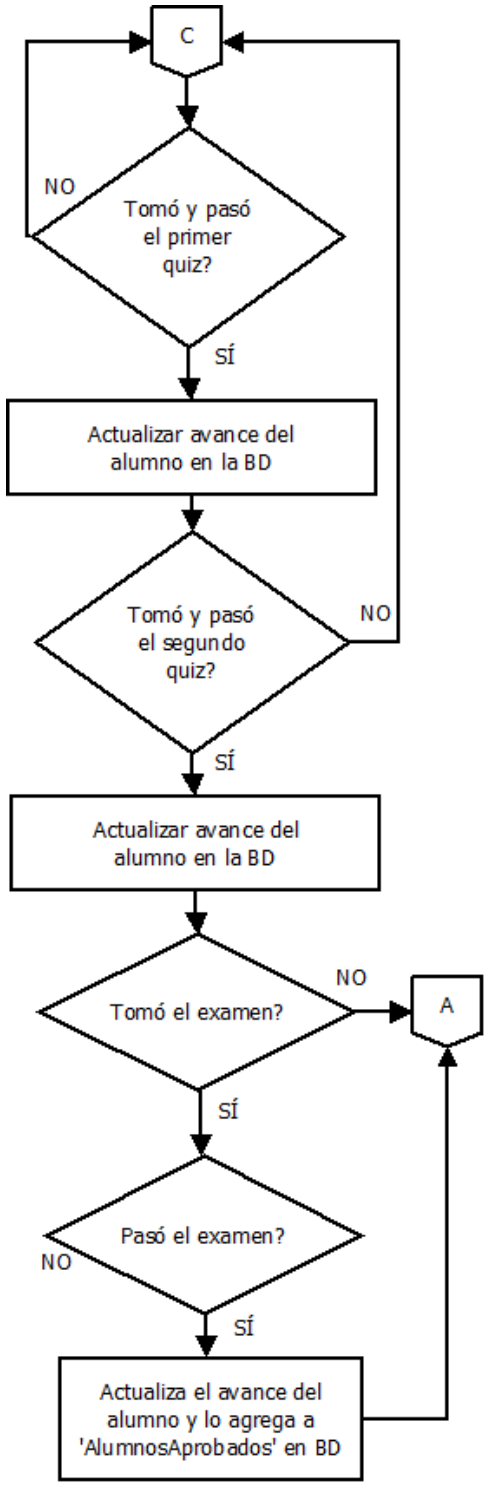

Figura 3.5: Parte 3 del diagrama de estados

3.2 Desarrollo de AGtive implementando Scrum

3.2.1 Sprint 1

En la sección 2.9, se mencionaron algunas de las tecnologías involucradas en el desarrollo de AGtive, también se habló sobre el marco de trabajo Scrum. A continuación se describe con mayor detalle el desarrollo del primer sprint (tabla 2.3).

Clases en Kotlin

Las figuras 3.6, 3.7, 3.8 muestran las clases utilizadas a lo largo del desarrollo de la aplicación, las cuales gracias a Scrum se redujeron y mejoraron de modo que los datos (como se muestra en las figuras) son exactos. Cabe mencionar que el diseño original de algunas clases como lo fue el caso de Alumno, tenían campos innecesarios que en un principio se creyeron útiles, pero que al final no fue así, y sin embargo se terminaron añadiendo algunos otros. La misma situación ocurrió con el tipo de dato de los atributos en las clases.

Usuario.kt | Alumno.kt | Curso.kt

```
1  package com.jaax.agtive.modelo
2
3  open class Usuario(
4      open val id: String,
5      open val nombre: String,
6      val email: String,
7      val telefono: String,
8      open val imagenURL: String
9  )
```

Figura 3.6: Clase Usuario

Una de las ventajas de Scrum al estar presente en el desarrollo de este proyecto fue la capacidad de mantenerse al margen para no avanzar sin antes haber terminado las tareas planteadas, lo cual se aplicó tras percatarse de cambios necesarios y forzados. Como se mencionó anteriormente, el modelado de clases se pensaba que era el correcto en un principio, pero no fue hasta que hubo problemas de codificación que se pudo observar el fallo en ellas, y esto se debió a un mal seguimiento de la metodología, por suerte no fue tan grave, ya que Scrum es conciso en cuanto a cambios durante el proceso de desarrollo.

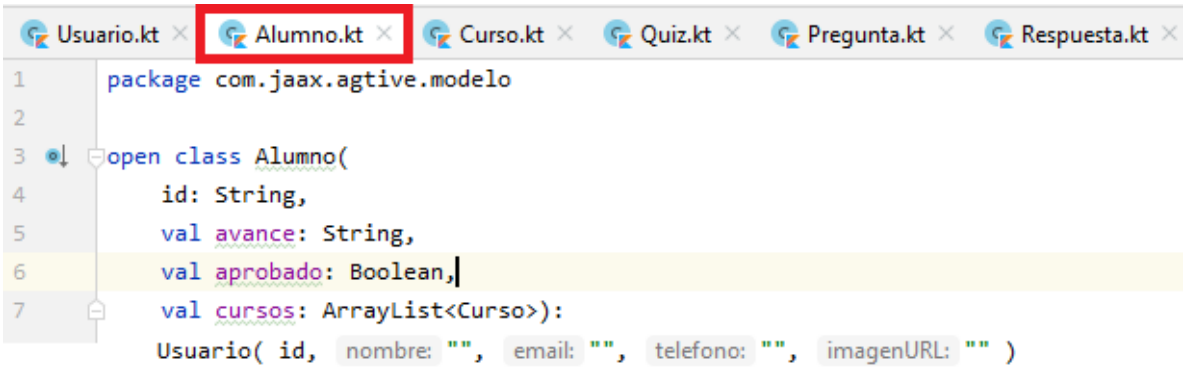

Figura 3.7: Clase Alumno

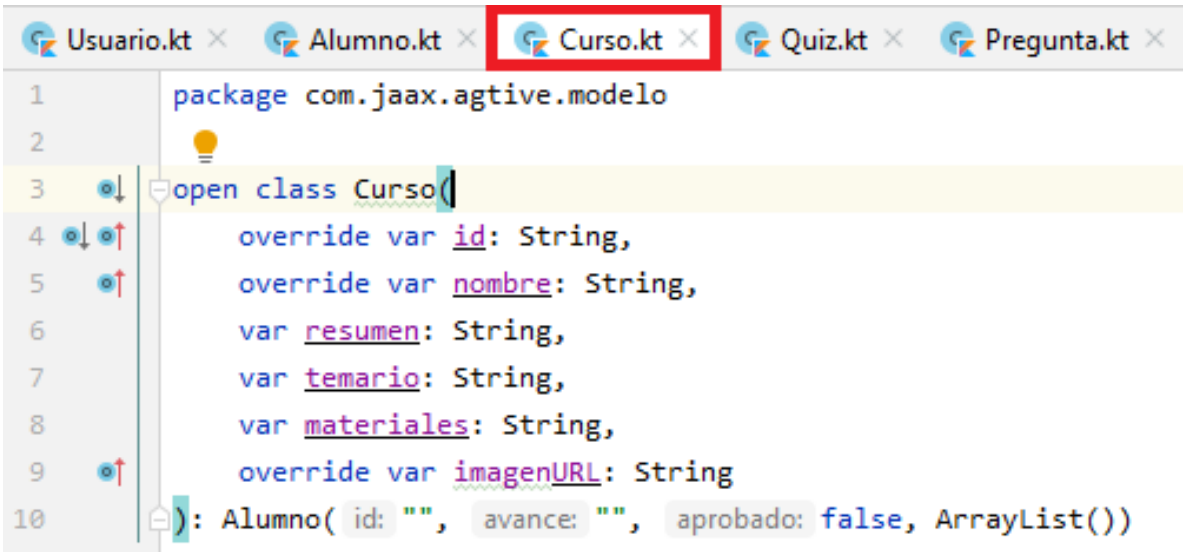

Figura 3.8: Clase Curso

Usuario.kt | Alumno.kt | Curso.kt | Quiz.kt | Pregunta.kt | Respuesta.kt

```
1  package com.jaax.agtive.modelo
2
3  open class Quiz(
4      override var id: String,
5      var preguntas: ArrayList<Pregunta>
6      ): Curso(id, nombre: "", resumen: "", temario: "", materiales: "", imagenURL: "")
```

Figura 3.9: Clase Quiz

Usuario.kt | Alumno.kt | Curso.kt | Quiz.kt | Pregunta.kt

```
1  package com.jaax.agtive.modelo
2
3  open class Pregunta(
4      id: String,
5      var texto: String,
6      var respuestas: ArrayList<Respuesta>): Quiz(id, ArrayList())
```

Figura 3.10: Clase Pregunta

Usuario.kt | Alumno.kt | Curso.kt | Quiz.kt | Pregunta.kt | Respuesta.kt

```
1  package com.jaax.agtive.modelo
2
3  open class Respuesta(
4      id: String,
5      var recurso: String, //texto o URi
6      var tipo: Boolean ): Pregunta(id, recurso, ArrayList())
```

Figura 3.11: Clase Respuesta

Diseño de la interfaz gráfica

En esta subsección se muestran los blueprint también conocidos como planos o bocetos y diseños concretos de algunas de las vistas más relevantes de AGtive.
Es importante mencionar que las figuras presentadas se encuentran en su estado final, es decir, se implementó el sprint 1 que son los bocetos de vistas y el sprint 5 que implica mejorarlas para hacerlas ver más atractivas. Se decidió de esta manera, ya que es más sencillo tener primero la lógica implementada y después "adornar" la aplicación. Un ejemplo un tanto burdo sería preparación de un pastel, en donde lo más importante es la combinación de ingredientes y al final sólo decorar.

Figura 3.12: Logueo con Gmail

El diseño de interfaces gráficas en Android se puede realizar de dos maneras: utilizando la paleta de herramientas o por medio de código XML.

La primera consiste en seleccionar y arrastrar los elementos dentro de un layout que es un contenedor para visualizar los componentes agregados. La figura 3.13 muestra algunos de los elementos disponibles.

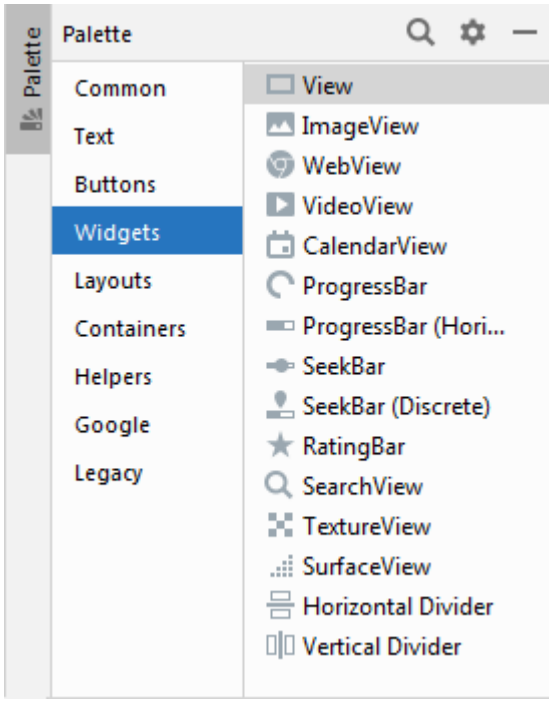

Figura 3.13: Paleta de elementos de vista

Por otro lado, se puede utilizar código XML para diseñar las vistas. Esto puede ser una ventaja, ya que se tiene mayor control de los elementos agregados. La figura 3.14 muestra un ejemplo de esto para modelar lo que es la vista de la figura 3.15.

```
12      <include
13          layout="@layout/swipe_layout"
14          android:layout_width="match_parent"
15          android:layout_height="match_parent"/>
16
17      <com.google.android.material.navigation.NavigationView
18          android:id="@+id/nav_view"
19          android:layout_width="wrap_content"
20          android:layout_height="match_parent"
21          android:layout_gravity="start"
22          android:fitsSystemWindows="true"
23          app:headerLayout="@layout/nav_header"
24          app:menu="@menu/nav_menu_drawer"/>
25
26      <ImageView
27          android:id="@+id/img_no_internet"
28          android:layout_width="match_parent"
29          android:layout_height="match_parent"
30          android:src="@drawable/nointernet2"/>
31      <Button
32          android:id="@+id/btn_refresh"
33          android:layout_width="wrap_content"
34          android:layout_height="100dp"
35          android:text="REINTENTAR"
36          android:gravity="center_horizontal|bottom"
37          android:paddingBottom="150dp"
38          android:visibility="gone"
```

Figura 3.14: Ejemplo de XML para vistas en Android

Muchas de las vistas de la interfaz de usuario para AGtive tuvrieron como base a elementos de otras aplicaciones como la misma Play Store de Google. En la figura 3.15 se utilizó un "DraweLayout" y "NavigationView" para construir lo que en ella se observa, misma que es muy similar a la barra de herramientas de la ya mencionada de la aplicación de Google. También se puede apreciar una cabecera con una imagen, el nombre de la aplicación y dos líneas que corresponden al nombre del usuario y su correo electrónico respectivamente. Después se tiene el menú para cambiar de cuenta, mostrar un mensaje corto y conocer un poco acerca de la aplicación.

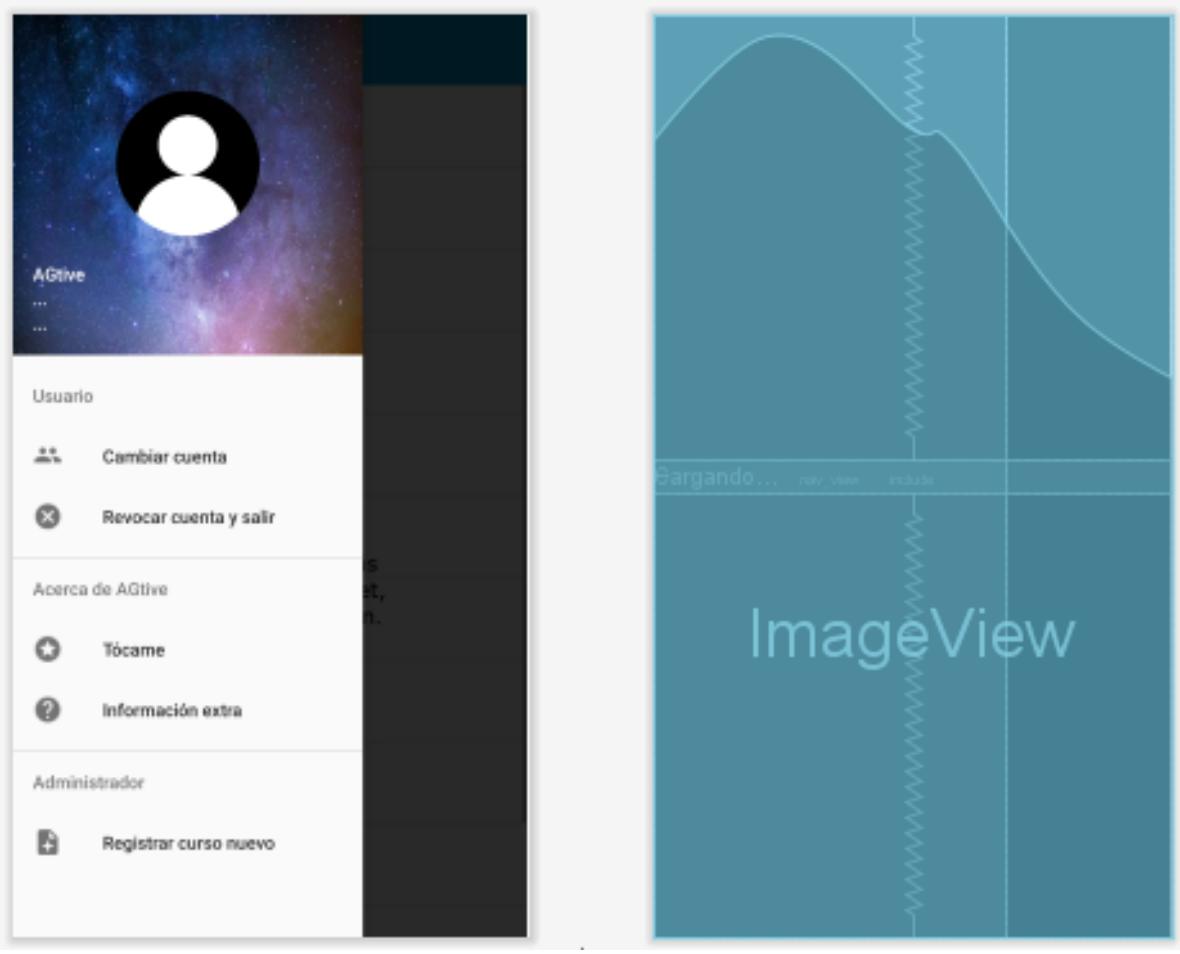

Figura 3.15: Menú de usuario

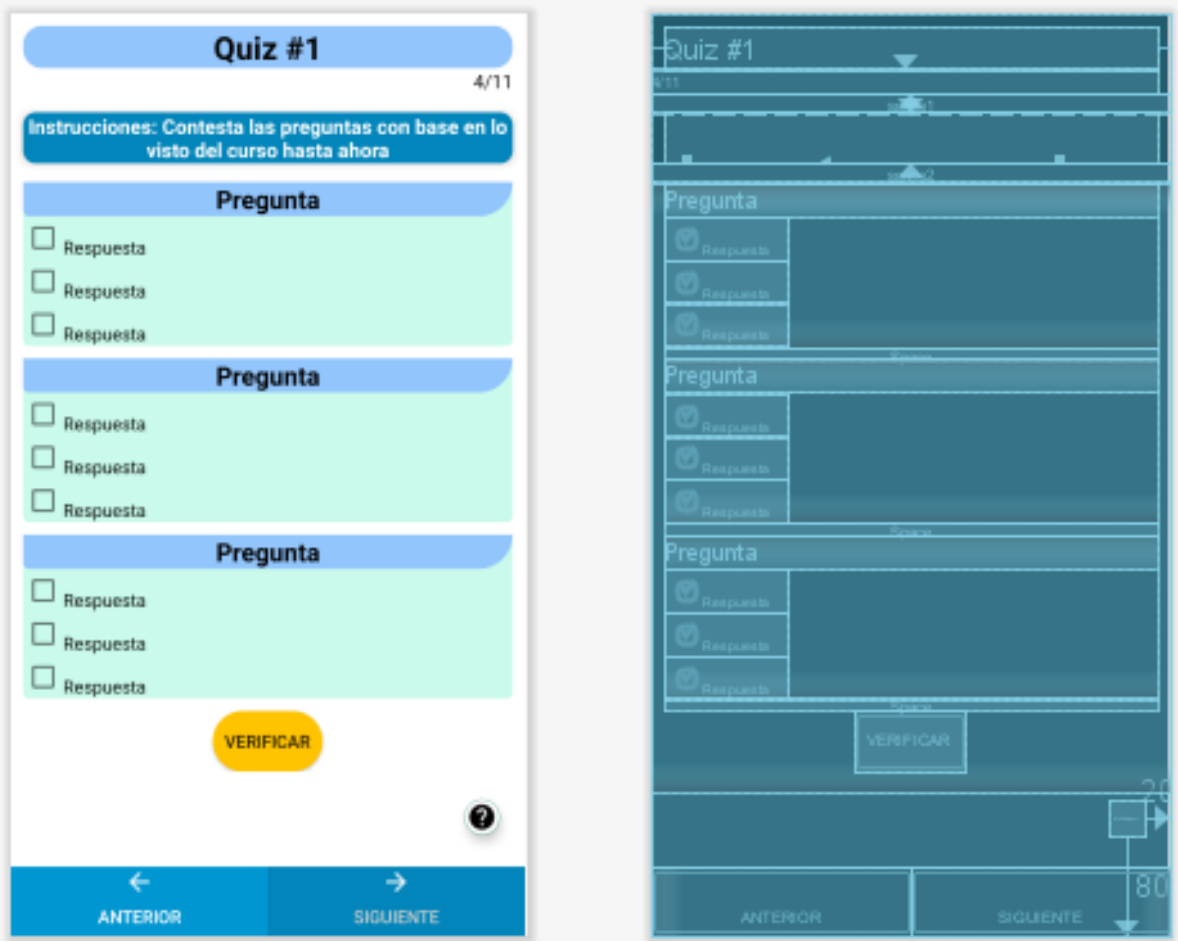

Figura 3.16: Plano del primer quiz del curso Representación entera

La idea de implementar un quiz durante un curso y no al final de éste, viene de una aplicación llamada "Sololearn", la cual ofrece cursos sobre programación de software que incluyen quizes y ejercicios de programación.

En el quiz número dos se pretendía hacer algo más dinámico y no sólo implementar las típicas preguntas y respuestas, lo que dio como resultado el arrastre y colocación de palabras en el lugar correcto conforme a lo visto en el curso hasta ese punto.

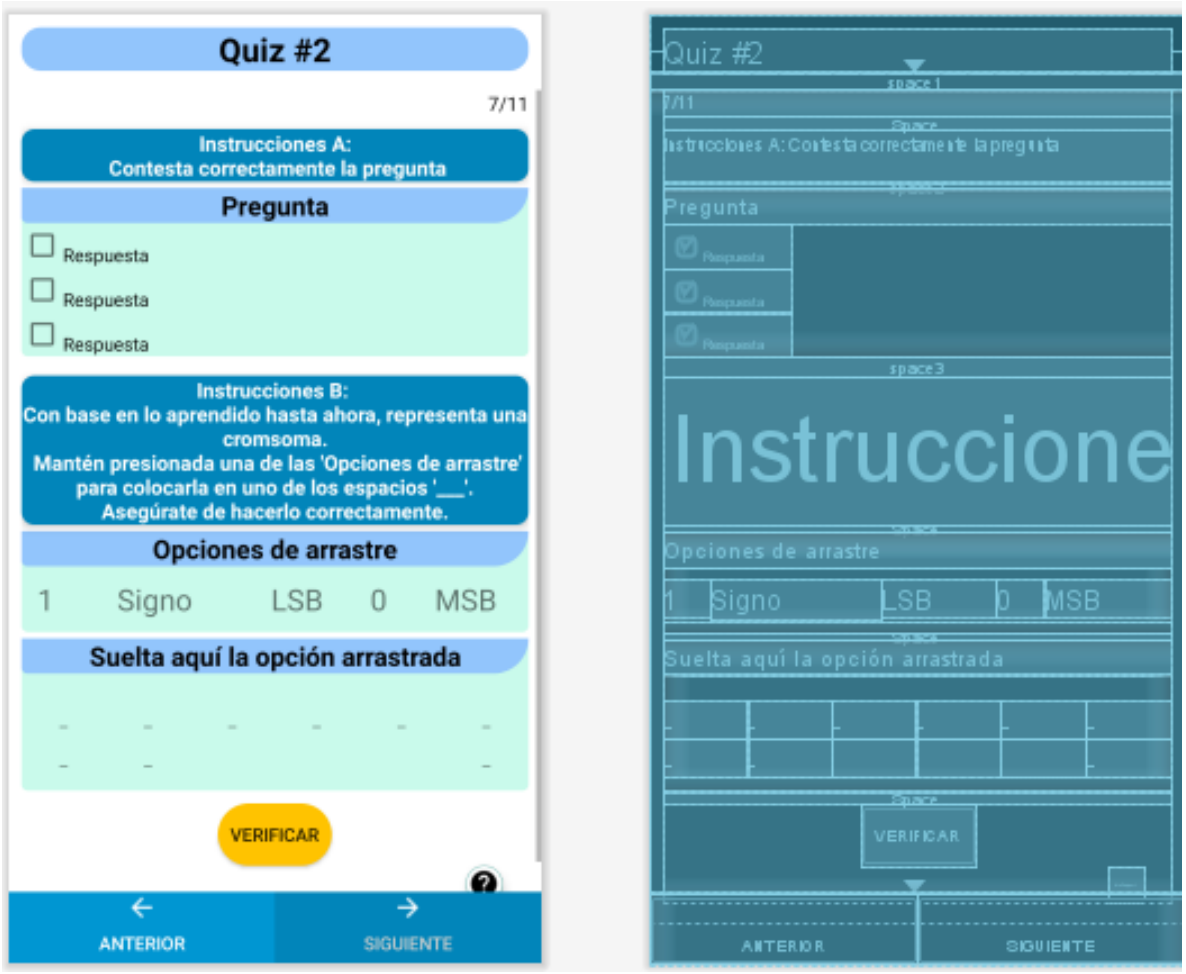

Figura 3.17: Plano del segundo quiz del curso Representación entera

La figura 3.18 muestra la vista seis de once en donde se describe la representación entera de un cromosoma y las partes de éste. La idea de desarrollar AGtive la de ser preciso, sencillo y concreto en lo que se quiere enseñar; esto con la finalidad de que el alumno logre el mejor entendimiento posible. Es cierto que en algunas vistas se abarca mucha información y puede llegar a ser molesto a la vista, pero dado que se explica con detalle, además de que es concreto en lo que menciona, se compensa.

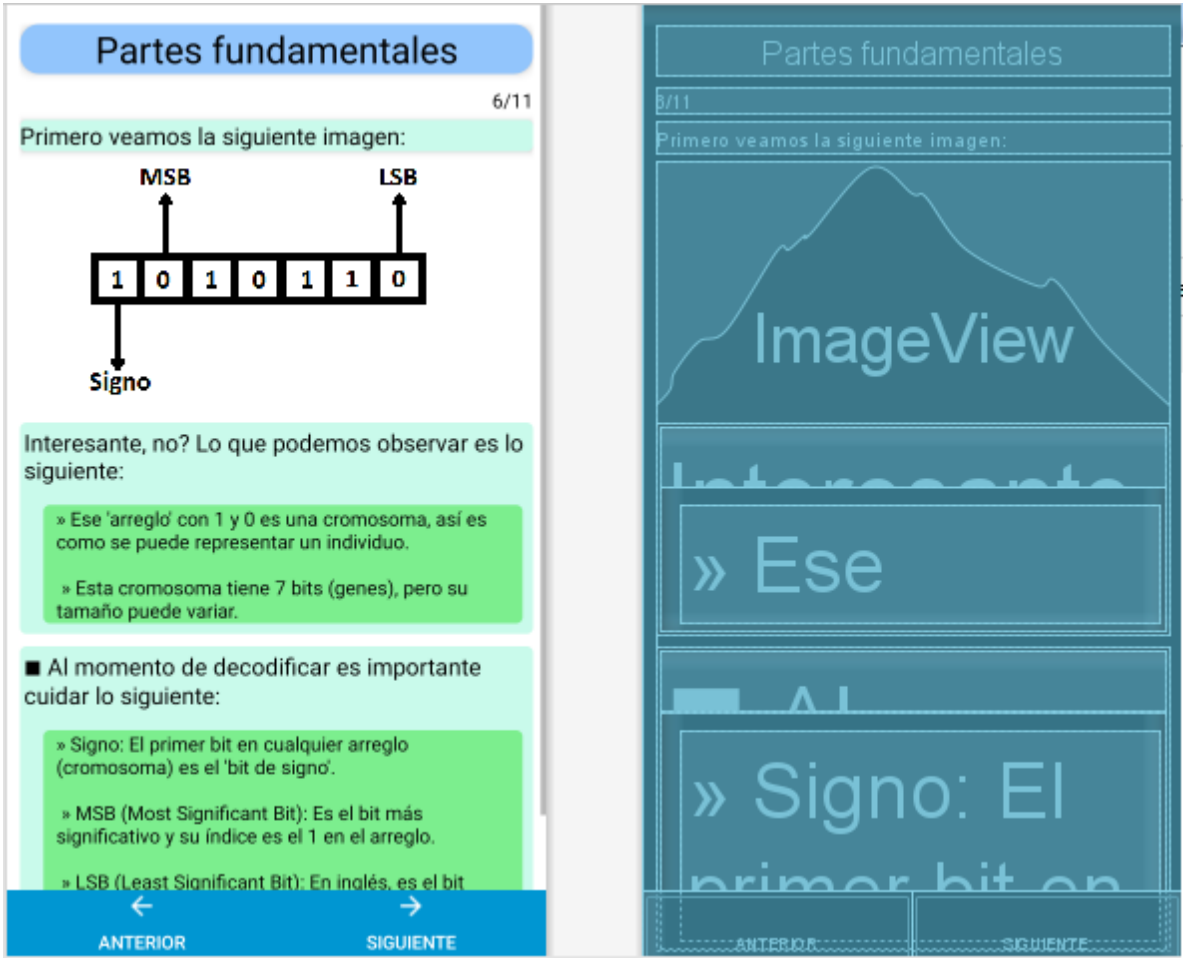

Figura 3.18: Parte 6 del curso como ejemplo

El diseño del examen tuvo como base los quiz uno y dos, además de partes que se deben completar escribiendo siguiendo la lógica de programación, la cual se muestra en el curso.

Para no hacer del examen algo molesto a la vista, se dividió en secciones, donde la sección A corresponde a sólo preguntas y respuestas, la B a opciones de arrastre y las secciones C y D igualmente opciones para arrastrar y soltar, además de completar código utilizando el teclado del teléfono.

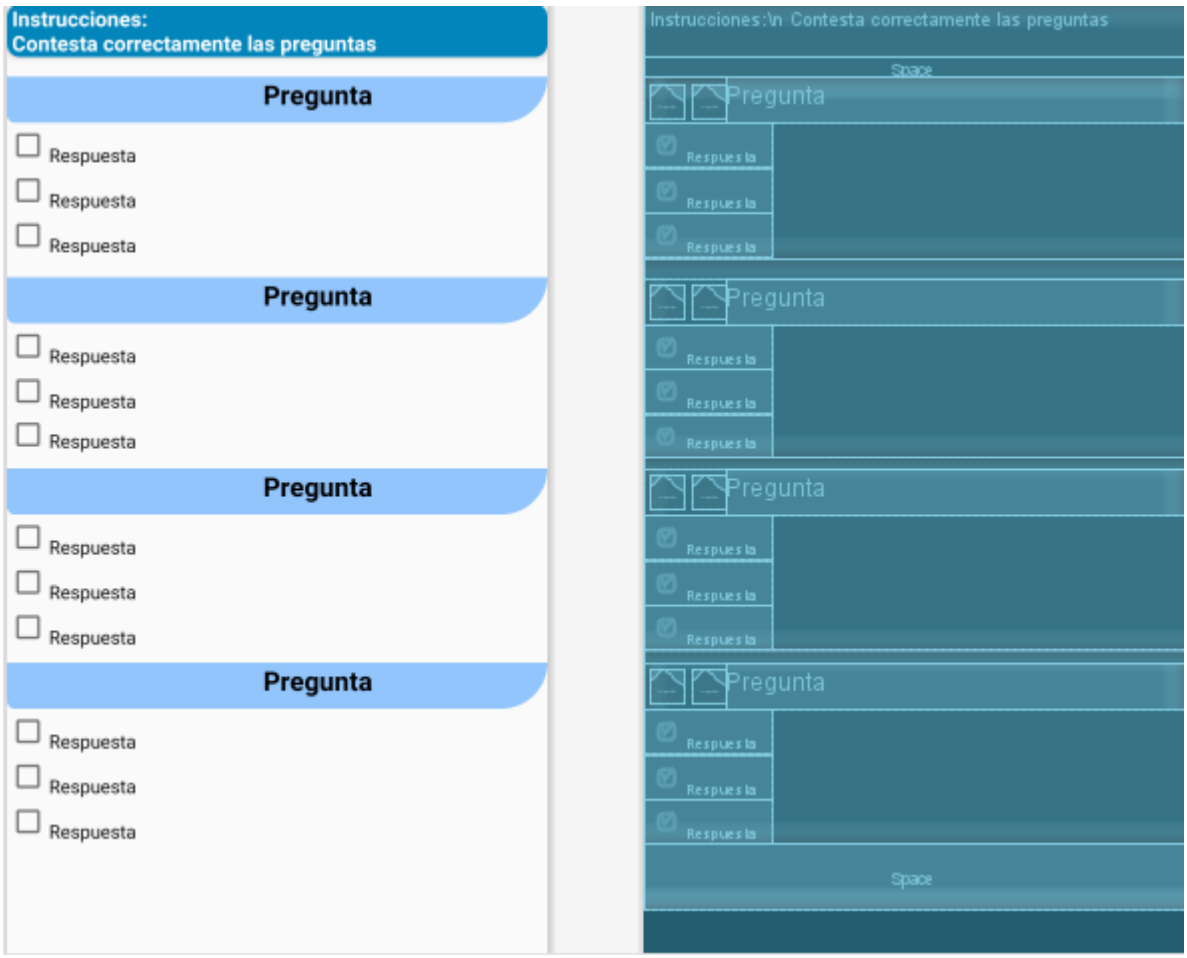

Figura 3.19: Plano de la sección A del examen final de Representación Entera

3.2.2 Sprint 2

Este sprint corresponde a las primeras implementaciones de Firebase en AGtive como se ve en la tabla 2.4. En la sección 2.8.4 se desglosan los pasos para conectar Firebase con Android Studio para después continuar a la conexión de Authentication utilizando una cuenta de Gmail. Finalmente queda agregar el menú de usuario y llenar sus datos con los obtenidos a través del logueo.

Autenticación con Firebase

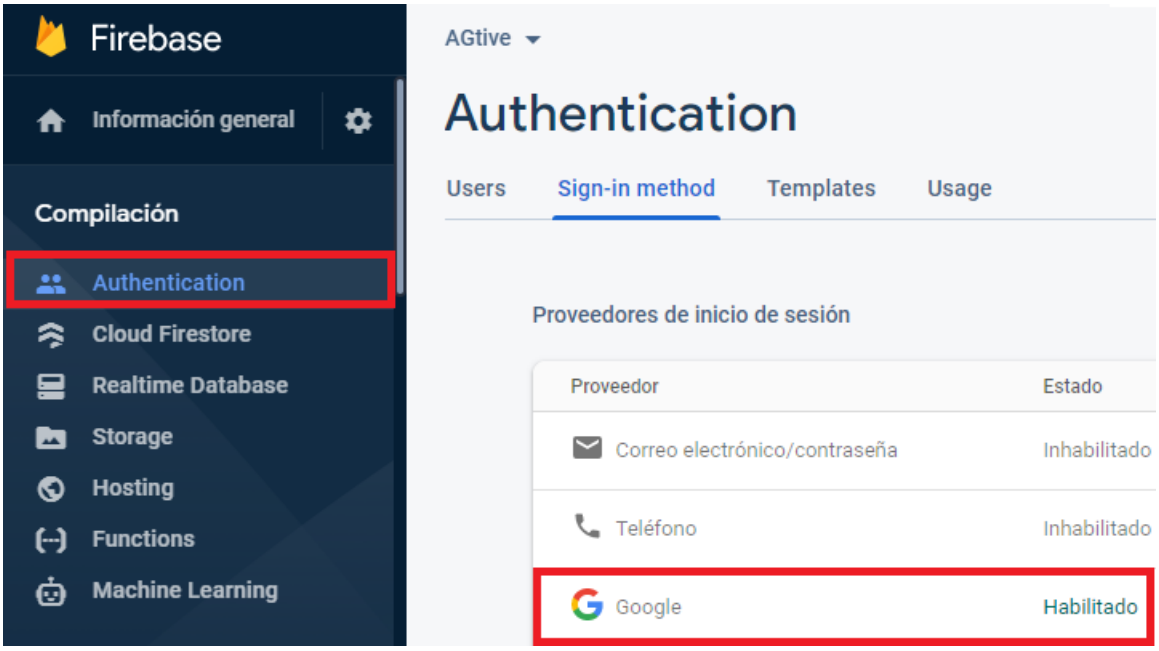

Figura 3.20: Habilitación de Authentication

La figura 3.20 muestra como habilitar la autenticación con Gmail en nuestro proyecto con Firebase desde su consola web.

En la figura 3.21 se observa la opción de Firebase para visualizar si se han implementado las dependencias necesarias para utilizar la autenticación, mismas que se deben incluir en el archivo *build.gradle* como en la figura 3.22.

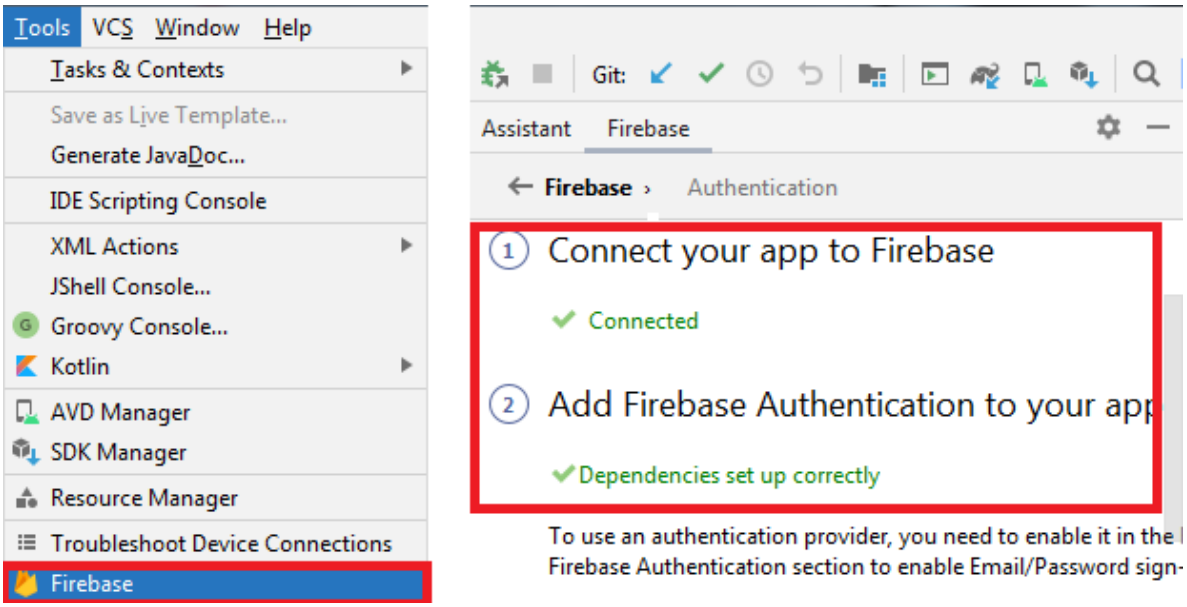

Figura 3.21: Confirmación de autenticación con Firebase

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.github.bumptech.glide:glide:4.11.0'
    implementation 'com.google.android.gms:play-services-auth:19.0.0'
    implementation 'com.google.firebase:firebase-database:19.6.0'
    implementation 'com.google.firebase:firebase-auth:20.0.2'
    implementation 'com.google.firebase:firebase-analytics-ktx'
    implementation platform('com.google.firebase:firebase-bom:26.3.0')
    testImplementation 'junit:junit:4.13.1'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Figura 3.22: Dependencias para Firebase Authentication

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.login_google)
    signInBtn = findViewById(R.id.btn_signIn)
    progressBar = findViewById(R.id.progress_bar)

    //opciones para recuperacion de datos de usuario logueado
    val signInOptions = GoogleSignInOptions
        .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken("1054293294523-7tmtk9kr4bgsio48aoh42r9icfe5jq20.apps.goo...")
        .requestEmail()
        .build()

    signInClient = GoogleSignIn.getClient(this, signInOptions)
    firebaseAuth = FirebaseAuth.getInstance()
    firebaseAuthStateListener = FirebaseAuth.AuthStateListener{ firebaseAuth ->
        val user = firebaseAuth.currentUser
        if( user != null ){
            goMainScreen()
        }
    }

    signInBtn.setOnClickListener{ it: View!
        val intent = signInClient.signInIntent
        startActivityForResult( intent, SIGN_IN_CODE )
    }
}
```

Figura 3.23: Inicialización de objetos de tipo Firebase

La figura 3.23 muestra la construcción de los objetos necesarios para loguearse al crear la actividad. Lo más importante a resaltar es la variable *signInOptions* que contiene los datos de Firebase y el correo del usuario para conceder el acceso. También el "listener" o "escuchador" que ayuda a decidir si abrir el menú de logueo o ir directo a la actividad principal. El evento de botón *signInBtn* se invoca para obtener un resultado tras lanzar una petición a la actividad a través de un "Intent" (objeto para abrir actividades o servicios), y dependiendo de la respuesta, será la actividad la que se muestre.

La figura 3.24 proporciona una muestra del código inicial para loguearse a través de Gmail. El método "onActivityResult()" se llama cuando obtiene una respuesta del "Intent" lanzado. Si el resultado es nulo, se ejecuta "goMainScreen()" y dirigirse al logueo; pero si es distinto de null, es decir, el usuario no cerró su sesión anteriormente o es la primera vez que se registra, entonces llama a "handleSignInResult()" para invocar "firebaseAuthWithGoogle()" en caso de que no ocurra error alguno en el logueo y entonces abrir la actividad con el curso.

En éste último método se crea un "listener" que mientras se esté ejecutando la aplicación, va a estar presente para proporcionar los datos del usuario en cualquier momento que se requiera.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if( requestCode == SIGN_IN_CODE ){
        val result = Auth.GoogleSignInApi.getSignInResultFromIntent(data)
        if (result != null) {
            handleSignInResult(result)
        }
    }
}
private fun handleSignInResult(result: GoogleSignInResult) {
    if( result.isSuccess ){ firebaseAuthWithGoogle(result.signInAccount) }
}
private fun firebaseAuthWithGoogle(signInAccount: GoogleSignInAccount?) {
    progressBar.visibility = View.VISIBLE
    signInBtn.visibility = View.GONE
    val credential = GoogleAuthProvider.getCredential( signInAccount!!.idToken, null )
    firebaseAuth.signInWithCredential( credential ).addOnCompleteListener {  it: Task<AuthResult!>
        progressBar.visibility = View.GONE
    }
}
private fun goMainScreen() {
    val intent = Intent( packageContext: this, MainActivity::class.java)
    intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TOP or
            Intent.FLAG_ACTIVITY_CLEAR_TASK or
            Intent.FLAG_ACTIVITY_NEW_TASK
    startActivity(intent)
}
```

Figura 3.24: Código de logueo con Gmail

La figura 3.25 muestra parte del código de la actividad principal enfocándose en el logueo del usuario. Como puede observarse, *onStart()* (el cual se ejecuta después de crear la actividad) agrega un "listener" de autenticación mientras que *onStop()* (que se invoca si la actividad pasa a segundo plano) remueve dicho "listener".
Se tienen dos métodos restantes que son *onConnectionFailed()* y *onAuthStateChanged()*, donde el primero muestra un mensaje de error si no hay conexión a internet y el segundo llenará los campos de los datos del usuario (figura 3.15) si el logueo fue exitoso, o bien, regresará a la actividad de acceso con Gmail si no lo fue.

```
override fun onStart() {
    super.onStart()
    firebaseAuth.addAuthStateListener(this)
}
override fun onStop() {
    super.onStop()
    firebaseAuth.removeAuthStateListener(this)
}
override fun onConnectionFailed(p0: ConnectionResult) {
    Toast.makeText(
        applicationContext,
        text: "No se puede acceder a tu cuenta\nVerifica tu conexión a Internet",
        Toast.LENGTH_SHORT
    ).show()
}
override fun onAuthStateChanged(firebaseAuth: FirebaseAuth) {
    if( firebaseAuth.currentUser == null ){
        goToLoginScreen()
        return
    }
    firebaseAuth.currentUser!!.getIdToken(true)
        .addOnSuccessListener {  it: GetTokenResult!
            setDatosUsuarioFirebase(usuario)
        }
}
}
```

Figura 3.25: Listeners para la actividad principal

Para concluir esta subsección, se tiene la figura 3.26 que contiene algunos de los correos de cuentas Gmail de usuarios logueados en AGtive, con lo que se comprueba el correcto funcionamiento de la autenticación en AGtive.

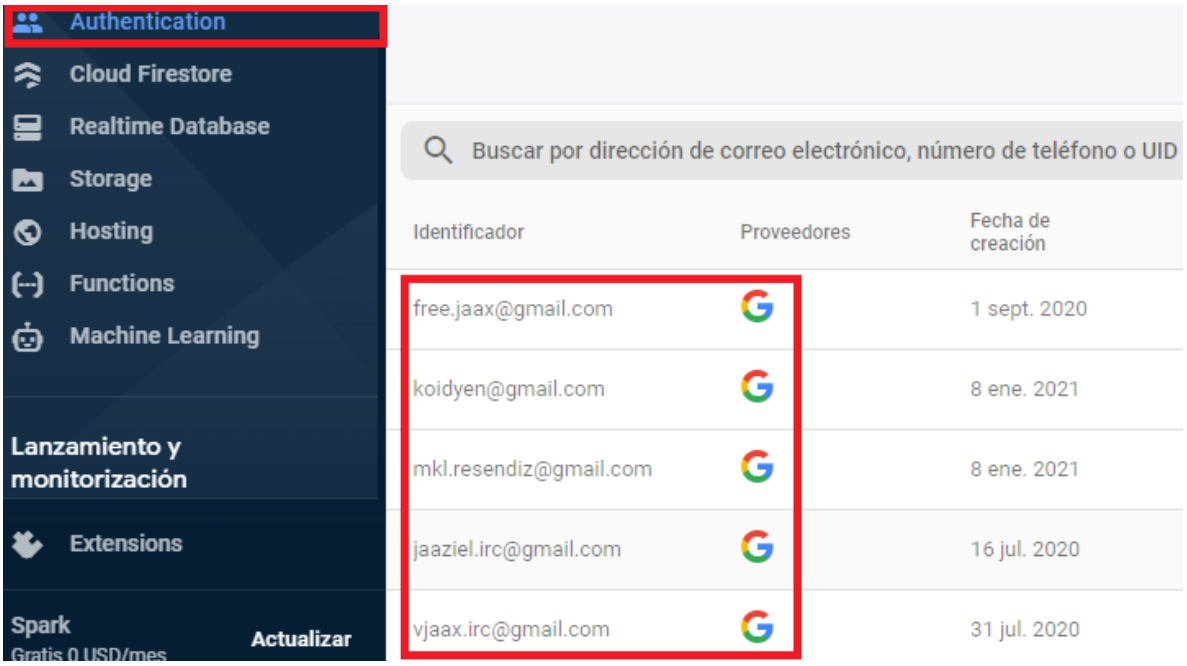

Figura 3.26: Cuentas Gmail en Firebase Authentication

3.2.3 Sprint 3

A grandes rasgos este sprint se enfoca en la base de datos de Firebase y la implementación de las primeras vistas del curso de representación entera como se indica en la tabla 2.5.

Conexión con Firebase Realtime Database

Una parte fundamental del funcionamiento de la aplicación e-learning fue el uso la base de datos en tiempo real de Firebase. En la subcección 2.8.4 se muestra la implementación de esta tecnología en un proyecto Android, sin embargo, es necesario habilitar esta función dentro del proyecto.

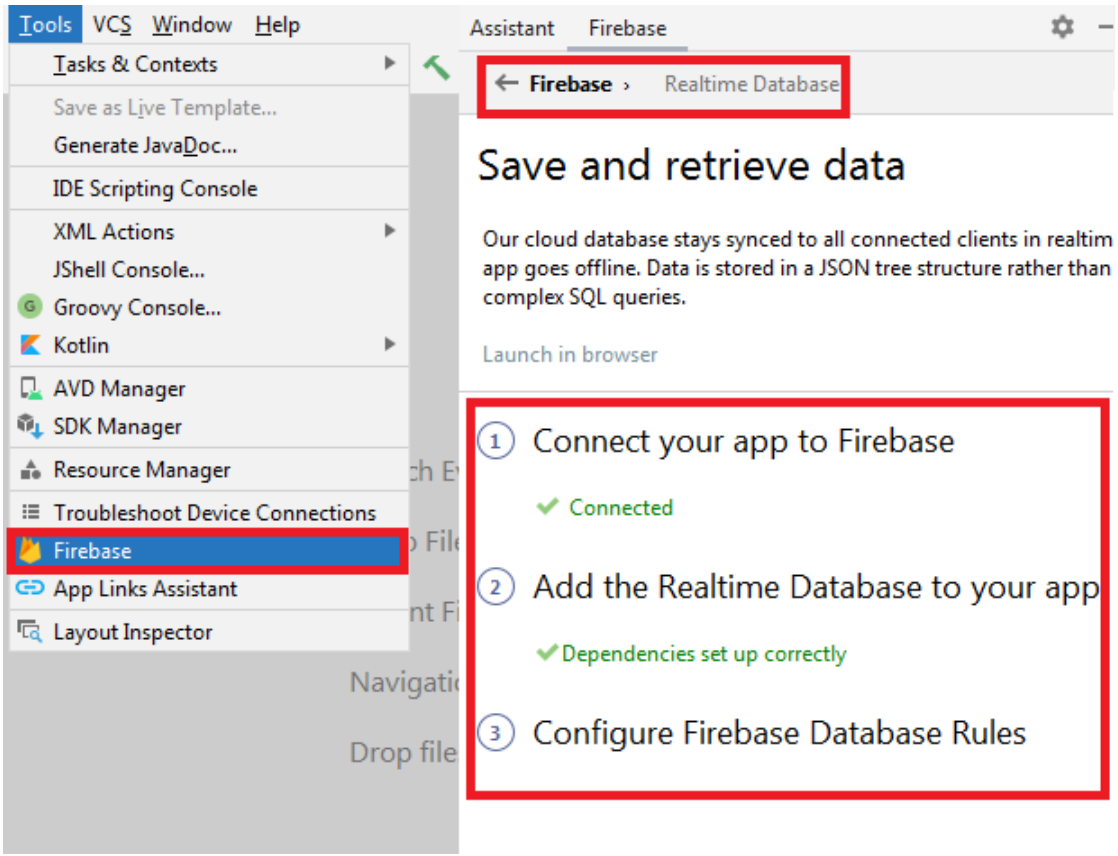

Figura 3.27: Habilitación de Firebase RDB en Android

La figura 3.27 muestra las instrucciones que indica la herramienta de Firebase para habilitar su base de datos, lo cual se logra incluyendo las dependencias correspondientes como lo indica la figura 3.28.

Es importante resaltar el punto número 3 que se muestra en la primer imagen, ya que apunta a configurar las reglas de Firebase Realtime Database.

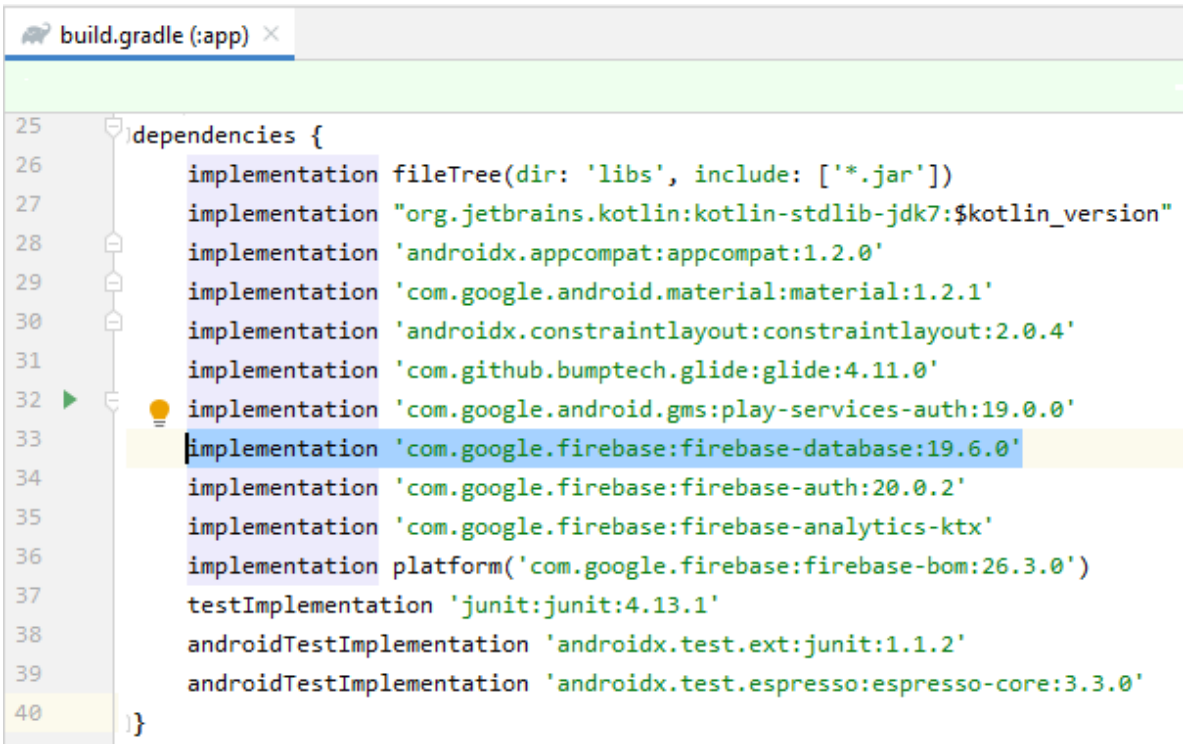

Figura 3.28: Dependencia para utilizar Firebase RDB en Android

Configurar reglas

La configuración de reglas en la base de datos es algo esencial para su correcto funcionamiento, pues con éstas Firebase decide que permisos tienen los usuarios logueados respecto a los datos.

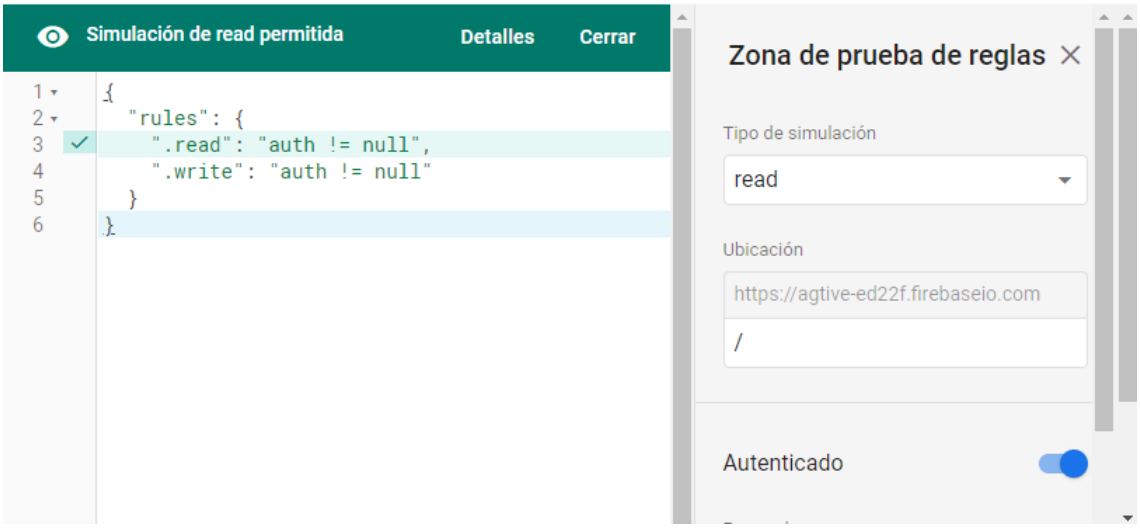

Figura 3.29: Apartado para configurar las reglas

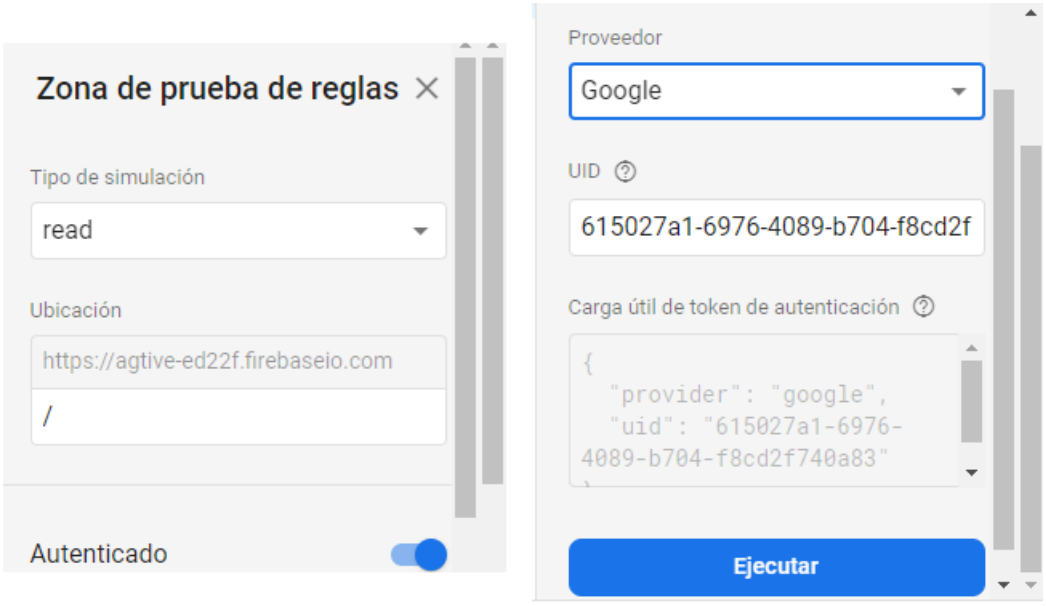

Figura 3.30: Zona de prueba de reglas

Las figura 3.29 muestra la consola para configuración de reglas de Realtime Database con una simulación permitida que indica que la lectura y escritura de datos está disponible si y sólo si, el usuario está autenticado con una cuenta Gmail como en la figura 3.30.

CRUD en AGtive

Las siglas CRUD corresponden a "Create", "Read", "Update" y "Delete" que son las funciones básicas que se pueden realizar en una base de datos. Firebase proporciona dos maneras de aplicar dichas acciones; una es utilizando la consola web (figura 3.31) y la segunda dentro de Android Studio por medio del código.

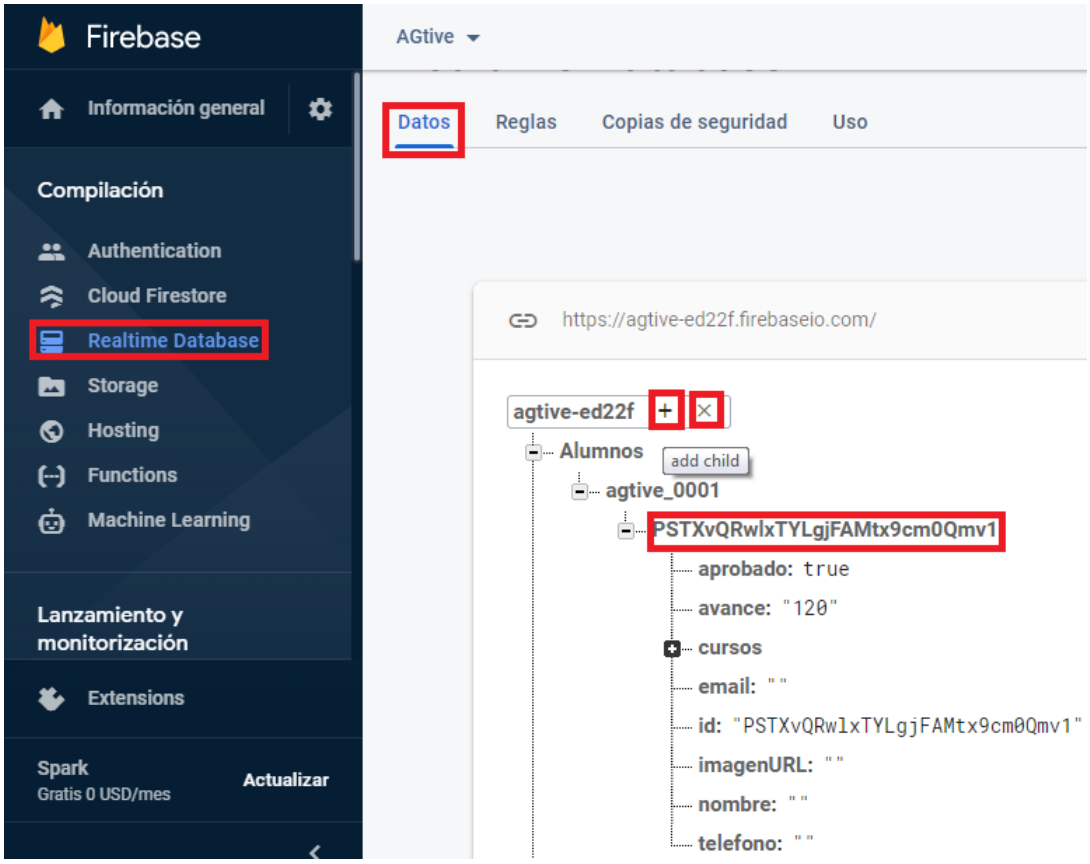

Figura 3.31: Datos de Realtime Database en consola web

```
95
96      1   firebaseAuth = FirebaseAuth.getInstance()
97          dbreference = FirebaseDatabase.getInstance().reference
98
99          if( firebaseAuth.currentUser != null ){
100             usuario = Usuario(
101         2       firebaseAuth.currentUser!!.uid,
102                 firebaseAuth.currentUser!!.displayName.toString(),
103                 firebaseAuth.currentUser!!.email.toString(),
104                 firebaseAuth.currentUser!!.phoneNumber.toString(),
105                 firebaseAuth.currentUser!!.photoUrl.toString()
106             )
107     3   dbreference.child( pathString: "Usuarios").child(usuario.id).setValue(usuario)
108         } else {
109             val intent = Intent( packageContext: this, LoginActivity::class.java)
110     4       startActivity(intent)
111         }
```

Figura 3.32: Función insertar en Realtime Database desde código

La figura 3.32 muestra cuatro recuadros que engloban las funciones de insertar y actualizar que utilizan de la misma función para ejecutarse.

El primer recuadro indica la inicialización de los objetos que necesarios para insertar un objeto de la clase Usuario (figura 3.6); *firebaseAuth* va a proporcionar acceso a los datos del usuario ya autenticado, mientras que *dbreference* obtiene la instancia de la base de datos de Firebase que se conectó al proyecto.

El segundo recuadro hace referencia a si existe el usuario por medio de una autenticación, y de ser así, entonces se crea un objeto Usuario con sus parámetros correspondientes que se obtienen de *firebaseAuth*.

El tercer recuadro realiza la función insertar. Firebase RDB maneja los datos en forma de ramas, por lo que la inserción inicial será en un "hijo" llamado "Usuarios" el cual tendrá otro "hijo" que es el *id* del usuario y finalmente se agrega el objeto Usuario con *setValue()*.

El cuarto recuadro sólo indica la apertura de la actividad de logueo en caso de no existir un usuario con credenciales almacenadas (figura 3.25).

Los resultados finales se pueden visualizar en la consola web como en la figura 3.33.

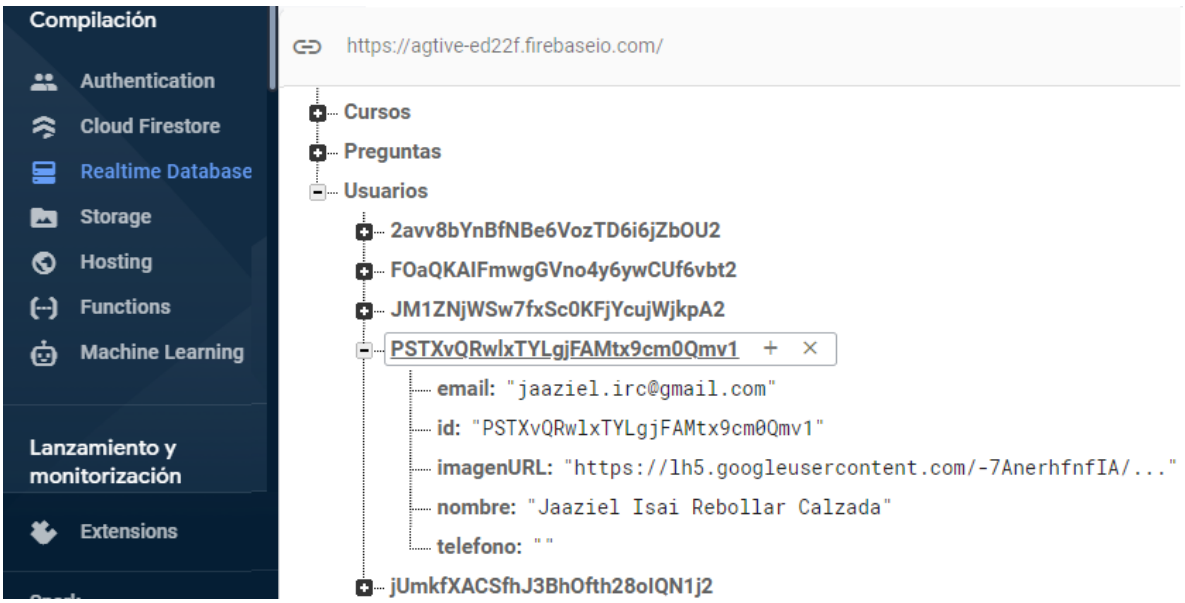

Figura 3.33: Resultados de inserción en consola de Firebase RDB

3.2.4 Sprint 4 y 5

Este sprint en conjunto con el anterior se enfocaron en el desarrollo concreto del contenido del curso de Representación Entera (RE), por lo que se muestran algunas de las vistas más relevantes y parte del código fuente con que se implementó.

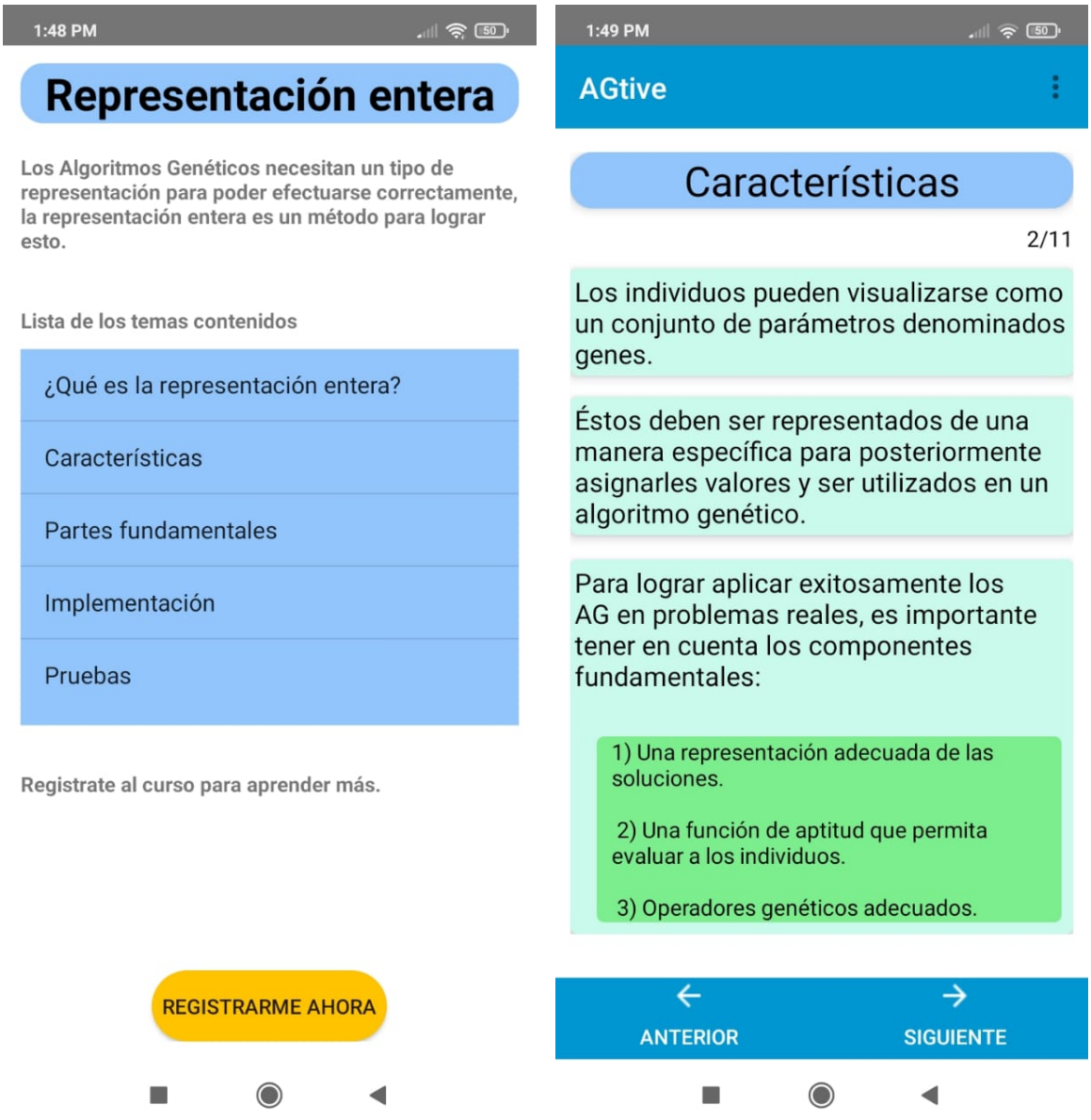

Figura 3.34: Temario y segunda vista del curso R.E

```
private fun setAvanceCurso(db: DatabaseReference, usr: FirebaseAuth) {
    db.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {
            if (snapshot.hasChild( path: "Alumnos")) {
                val currentAlumno = snapshot.child( path: "Alumnos/agtive_0001/${usr.currentUser!!.uid}")

                if (currentAlumno.child( path: "id").value.toString() == usr.currentUser!!.uid) {
                    val avance = currentAlumno.child( path: "avance").value.toString()
                    val transaction = manager.beginTransaction()
                    val fragment: Fragment?
                    toolbar.visibility = View.VISIBLE

                    when (avance) {
                        "10" -> {
                            try{
                                supportFragmentManager.popBackStackImmediate()
                                fragment = RepEntera1()
                                if (manager.backStackEntryCount != 1) {
                                    transaction.replace(R.id.repEntera_base, fragment)
                                        .addToBackStack( name: "repEnt1").commit()
                                }
```

1

2

3

Figura 3.35: Código para continuar el curso donde se dejó

La figura 3.34 muestra el diseño del temario y la segunda vista del curso. La mayoría de las vistas son similares en cuanto a estructura, es decir, dos botones en la parte inferior, el menú en la parte superior y el contenido en medio.

La figura 3.35 muestra una parte del código de la actividad que se encarga de contener las vistas del curso, e indica que según sea el avance del alumno se cambie de vista.

Se muestran tres recuadros donde el primero apunta a que la referencia a la base de datos va a tener un ”listener” el cual activa al método *onDataChanged()* siempre que este primero esté activo.

El segundo recuadro muestra una estructura de control *if* que evalúa si existe algún alumno registrado, si no es así, se redirige a la vista de introducción (figura 3.34), de lo contrario se obtiene su avance por medio del acceso a los ”hijos” y entonces avanza al recuadro tres. Cabe resaltar que en este recuadro se observa la función de lectura

"Read" para RDB.

El tercer y último recuadro indica la estructura de control *when* (propia de Kotlin) que evalúa un valor según sea la condición requerida, que para este caso, se trata del avance del alumno.

AGtive utiliza cuatro estados para el avance del alumno:

- **10**: El alumno recién comenzó a tomar el curso, ó reprobó el examen y se reinició su avance.
- **40**: El alumno logró pasar el primer quiz.
- **70**: El alumno logró pasar el segundo quiz.
- **110**: El alumno llegó a la última parte del curso y puede tomar el examen.
- **120**: El alumno pasó el examen y por lo tanto concluyó el curso.

Figura 3.36: Ejemplo de implementación y primer quiz con datos

```
124    private fun preguntasParseXML(): ArrayList<Pregunta> {
125        val parserFactory: XmlPullParserFactory = XmlPullParserFactory.newInstance()
126        val parser = parserFactory.newPullParser()
127        val inputstream = activity!!.assets.open( fileName: "preguntas_rEntera.xml")
128
129        parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false)
130        parser.setInput(inputstream, null)
131
132        val preguntas = ArrayList<Pregunta>( initialCapacity: 0)
133        var eventType = parser.eventType
134        var preguntaActual: Pregunta? = null
135
136        while (eventType != XmlPullParser.END_DOCUMENT) {
137            var dato: String
138            when (eventType) {
139                XmlPullParser.START_TAG -> {
140                    dato = parser.name
141                    if ("pregunta" == dato) {
142                        preguntaActual = Pregunta( id: "",  texto: "", ArrayList())
143                        preguntas.add(preguntaActual)
144                    } else if (preguntaActual != null) {
145                        if ("id" == dato) { preguntaActual.id = parser.nextText() }
146                        if ("texto" == dato) { preguntaActual.texto = parser.nextText() }
147                        if ("respuestas" == dato) {
148                            val tmp = separarAnswers( parser.nextText() )
149                            preguntaActual.respuestas = tmp
150                        }
151                    }
152                }
153            }
154            eventType = parser.next()
155        }
156        return preguntas
157    }
```

Figura 3.37: Parseo de datos XML para llenar el quiz

El código en la figura 3.37 corresponde al parseo de datos de un archivo XML que después se inserta en la vista del primer quiz. Este proceso se aplica para el segundo quiz y el examen.

Las vistas presentadas en la figura 3.38 corresponden al segundo quiz y el ejemplo de un Cromosoma con código en distintos lenguajes de programación.

Figura 3.38: Segundo quiz y clase Cromosoma

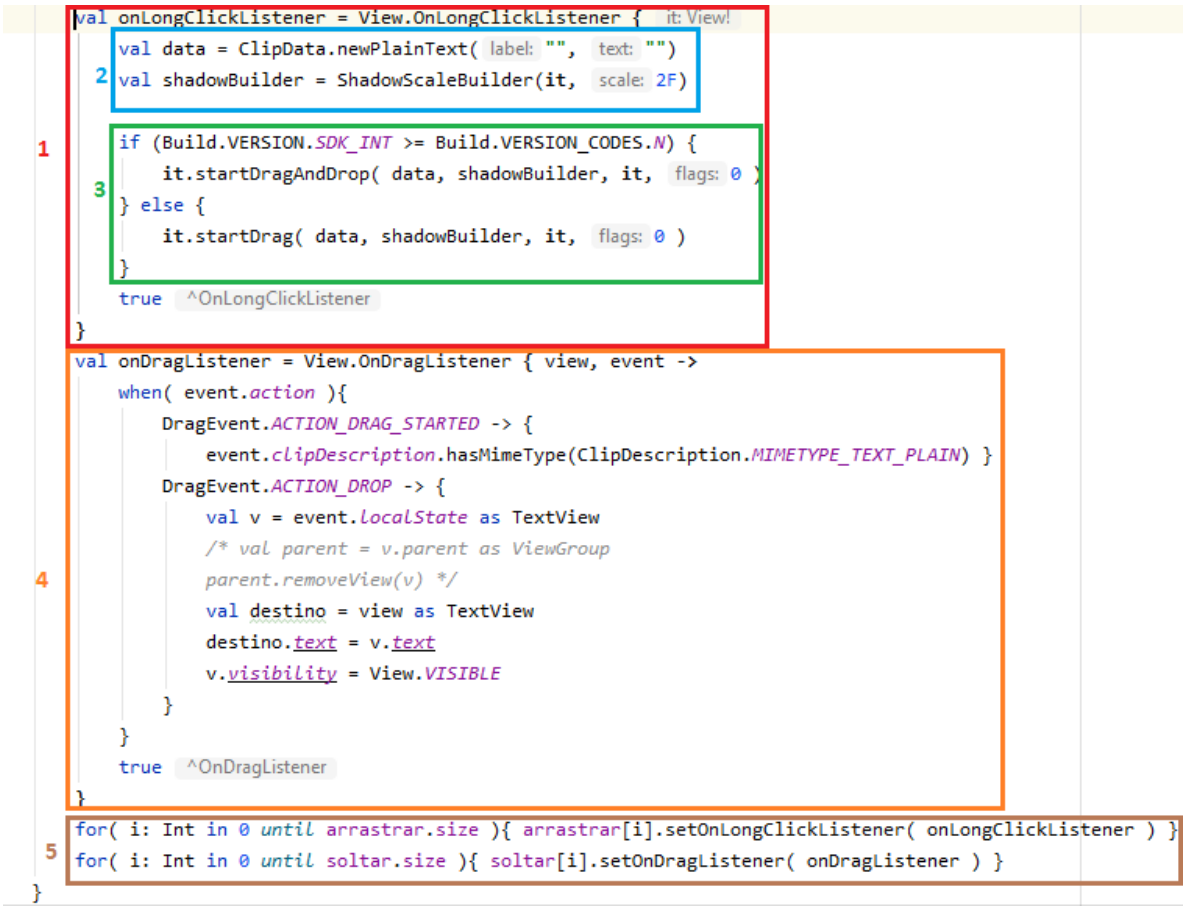

Figura 3.39: Eventos de arrastre en el segundo quiz

La figura 3.39 muestra los eventos de arrastre de los elementos utilizados en el segundo ejercicio del segundo quiz. En este se le pide al alumno que coloque las opciones disponibles en sus lugares correspondientes. Como se puede observar, se tienen cinco puntos los cuales describen lo siguiente:

- **Punto 1**: Se le asigna a una variable "onLongClickListener" un evento de clic largo.
- **Punto 2**: Se crea una variable "data" con un objeto "ClipData" que sirve como referencia al elemento presionado en el evento. La variable "shadowBuilder" va a permitir mostrar una sombra al momento de presionar el elemento.
- **Punto 3**: Como todos los dispositivos móviles con Android tienen versiones

diferentes, es necesario verificarlo para de ese modo invocar el método para permitir el arrastre a un elemento cliqueado.

- **Punto 4**: Se le asigna a una variable "onDragListener" el evento de arrastre, el cual puede tener distintos estados. Aquí sólo se muestra cuando arrastra y suelta el elemento en donde éste último cambia su texto por el que se le coloque.

- **Punto 5**: Se asignan los eventos de clic largo y arrastre a los elementos correspondientes.

La primer vista en la figura 3.40 muestra la codificación de un Individuo en distintos lenguajes de programación mientras que la segunda, las pruebas para verificar el funcionamiento conjunto de las clases y como resultado obtener un fenotipo.

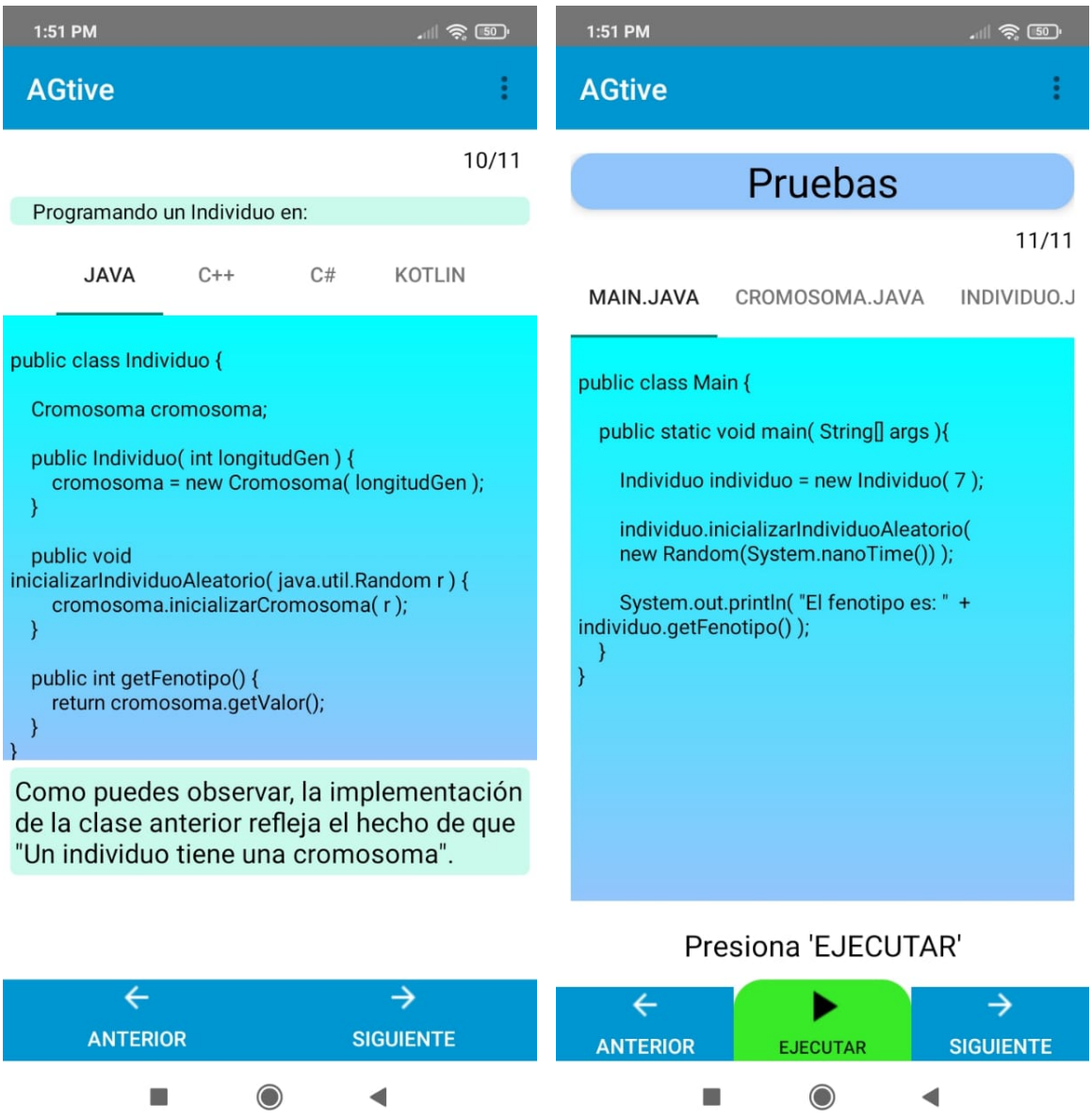

Figura 3.40: Clase Individuo en diversos lenguajes de programación y ejecución conjunta de la representación entera

Finalmente se tienen las vistas de las secciones C y D del examen del curso. Los métodos implementados en esta prueba final fueron la selección múltiple (figura 3.19), arrastre y colocación de respuestas (como el quiz 3.38) y completado de código por medio del teclado como se aprecia en la figura 3.41.

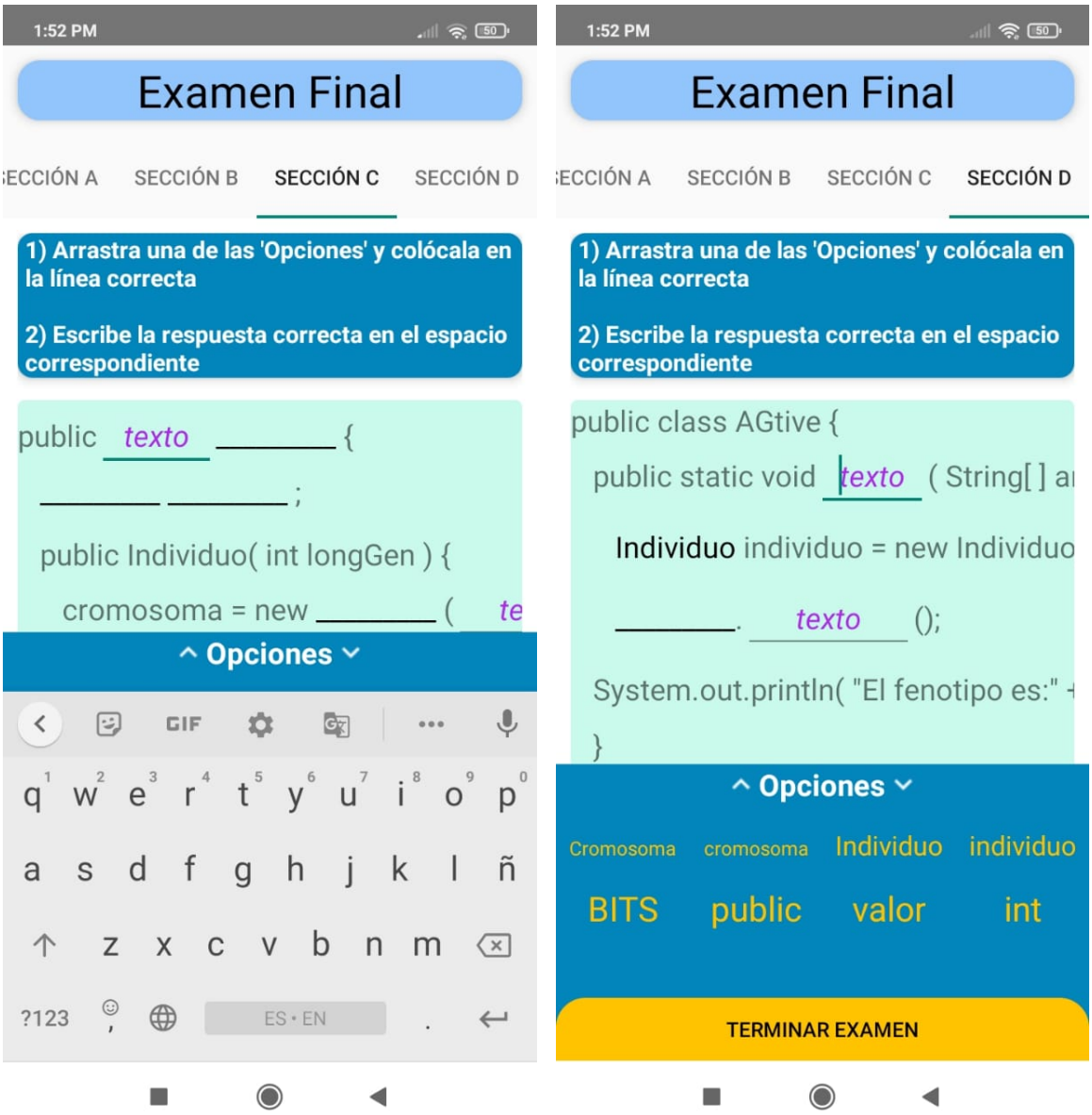

Figura 3.41: Ejercicios del examen de representación entera

Para concluir esta subsección, resta solamente mostrar un poco del código fuente del examen final en el que se da conocer la función "Update" para Realtime Database cuando un alumno aprueba o reprueba el examen. El "listener" que se ve en la figura 2.15 se vuelve a repetir como en la figura 3.35; la diferencia recae en el "hijo" en que se posiciona y el valor que actualiza, además de que esta vez sólo "escucha" para recibir una única respuesta por medio del método *addListenerForSingleValueEvent()*.

```
private fun updateAvanceAlumno(
    db: DatabaseReference,
    user: FirebaseAuth,
    score: Float
) {
    if( score >= 6.0F ){
        db.child( pathString: "Alumnos/agtive_0001/${user.currentUser!!.uid}")
            .addListenerForSingleValueEvent(object : ValueEventListener {

                override fun onDataChange(snapshot: DataSnapshot) {
                    if (snapshot.exists()) {
                        db.child( pathString: "aprobado").setValue(true)
                        db.child( pathString: "avance").setValue("120")
                        db.child( pathString: "cursos/0/aprobado").setValue(true)
                        db.child( pathString: "cursos/0/avance").setValue("120")
                    }
                }
                override fun onCancelled(error: DatabaseError) {
                    Log.e( tag: "rEnt12_UPDATE ",  msg: "Read failed: " + error.message)
                }
            })
```

Figura 3.42: Actualización de datos del alumno aprobado

Finalmente en el sprint 5, y con base en la tabla 2.7, se indica la implementación de la lógica del examen además de mejorar el diseño de la interfaz gráfica para hacer de la aplicación un proyecto más atractivo, donde esto último se ha visto a lo largo de las secciones 3.2.1, 3.2.2, 3.2.3 y 3.2.4.

Conclusiones

En esta tesis se desarrolló una aplicación e-learning para el aprendizaje de algoritmos genéticos. El software fue creado bajo un enfoque orientado a objetos y siguiendo los lineamientos de la metodología Scrum.

Se cumplieron todos los objetivos específicos planteados. La revisión de los principales conceptos de algoritmos genéticos se presentó en al capítulo 2. Se hizo una revisión de las principales características de los sistemas e-learning. Se describieron las herramientas y frameworks disponibles para desarrollo de aplicaciones para sistema operativo Android, eligiendo la más adecuada para este proyecto. El desarrollo del software comenzó con la identificación de requisitos y delimitando los alcances. Se diseñó la arquitectura del sistema y se representó mediante diagramas UML. La implementación y pruebas en un dispositivo Android se puede apreciar en las mismas conclusiones. Dado que los objetivos específicos se lograron, el objetivo general también se logró.

Aunque en los objetivos específicos no se plantea la publicación de AGtive en alguna tienda de aplicaciones, aún así se realizó con la finalidad de obtener una calificación y comentarios de los usuarios.

La hipótesis inicial fue confirmada, de acuerdo a la aceptación del software en la plataforma Google Play al momento de escribir estas conclusiones. Además se aplicó un cuestionario a los usuarios para conocer sus opiniones del trabajo realizado. Los resultados de esto se encuentran en el apéndice A.

Con base en las respuestas del cuestionario, las calificaciones y reseñas en la

Play Store por parte de los usuarios (pues son estas las que abarcan los resultados del trabajo realizado), se puede concluir que el objetivo de esta tesis se cumplió completamente además de concordar con el alcance del proyecto.

Dado que el desarrollo de AGtive fue exitoso, puede tomarse como punto de partida para desarrollar más cursos individuales y así completar la enseñanza de algoritmos genéticos.

Por otra parte, gracias a la arquitectura de software de AGtive sería posible implementar nuevas características como pudieran ser:

- **Optimización:** Consumir menos recursos del sistema. Por ejemplo, reducir el uso de memoria interna.
- **Generalización:** Con la posibilidad de diseñar un único modelo para los cursos para de esa manera facilitar la inclusión de su contenido.
- **Escalabilidad:** Crear nuevas aplicaciones de índole educativo enfocado a los universitarios, no necesariamente estudiantes de ingeniería de software, sino de cualquier otra carrera.

Apéndices

Apéndice A

Cuestionario

A las personas que se les compartió la aplicación se les pidió contestar un cuestionario con la finalidad de conocer la opinión que tenían al respecto, además de qué aspectos se podían mejorar. Dicho cuestionario se creó utilizando la herramienta "Forms" de Google, que ofrece una diversa gama de opciones para formular preguntas y aceptar respuestas.

El cuestionario consta de siete preguntas, las cuales fueron pensadas de manera concisa apuntando a los aspectos más importantes para obtener resultados a base de pruebas de la aplicación.

La figura A.1 muestra la lista de personas que pertenecen al grupo de quienes utilizaron la aplicación, pero que únicamente contestaron el cuestionario.

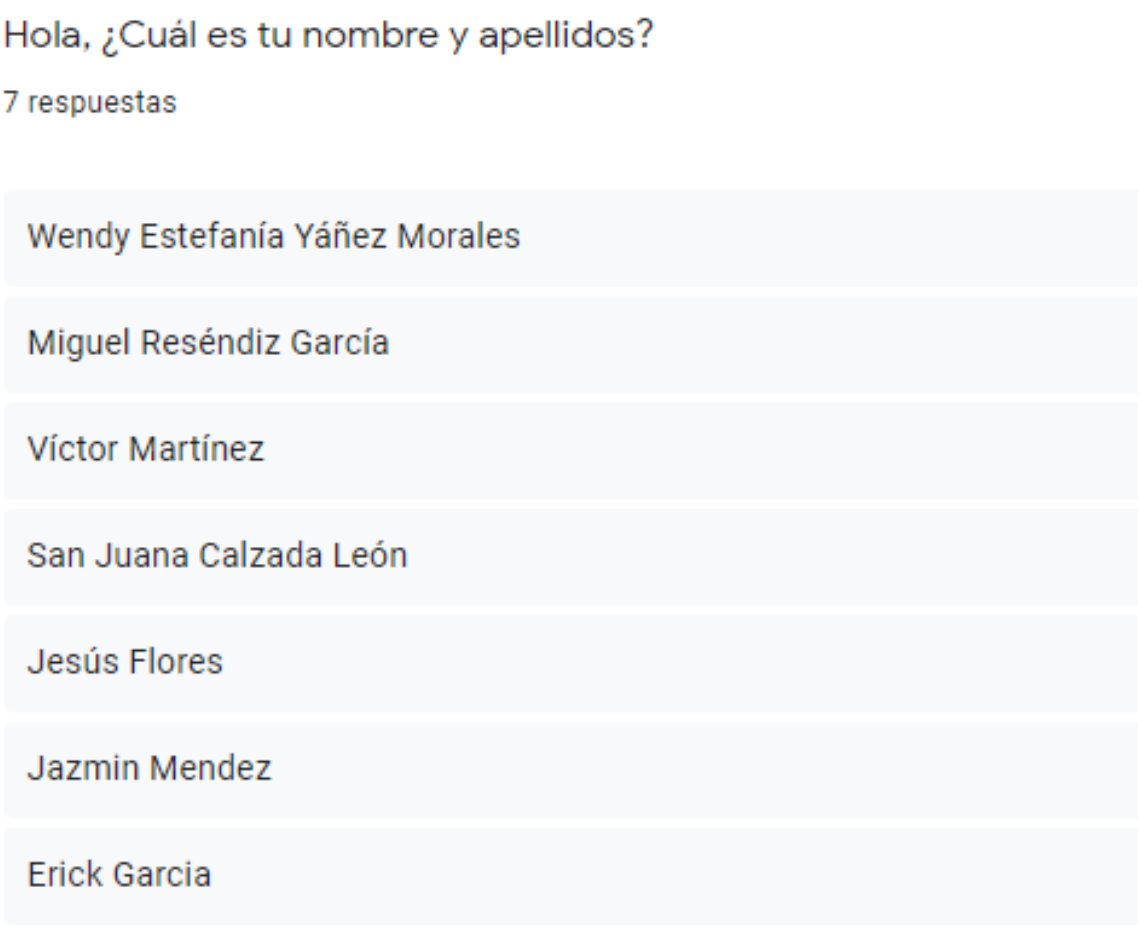

Figura A.1: Lista de personas que contestaron el cuestionario

Es importante mencionar que no todas las personas a quien se les compartió la aplicación tenían algún conocimiento sobre algoritmos genéticos o si quiera sabían algo sobre desarrollo de software. La finalidad de esto era la de obtener distintos puntos de vista de las personas y conocer la diferencia entre una que tiene indicios de la función de AGtive y una que ni siquiera entiende un poco de esto, pero que puede contribuir con su opinión.

La primer pregunta de la figura A.2 muestra el porcentaje de personas con algún conocimiento de software, siendo la mayoría. Estas estadísticas servirán en las siguientes páginas para mostrar la diferencia en el tipo de opinión que se tiene respecto

a la aplicación de una persona que tiene conocimiento de software a una que no lo tiene.

La segunda pregunta se refiere a una versión anterior de AGtive que tenía diversas debilidades en cuanto al diseño, como el uso de muchos colores que tampoco combinaban, causando incomodidad a la vista del usuario, instrucciones poco específicas o un diseño poco atractivo y conciso en el examen. La idea de esta segunda interrogante es para remarcar la diferencia de la versión anterior a la actual y de ese modo hacer notar más la opinión de los usuarios.

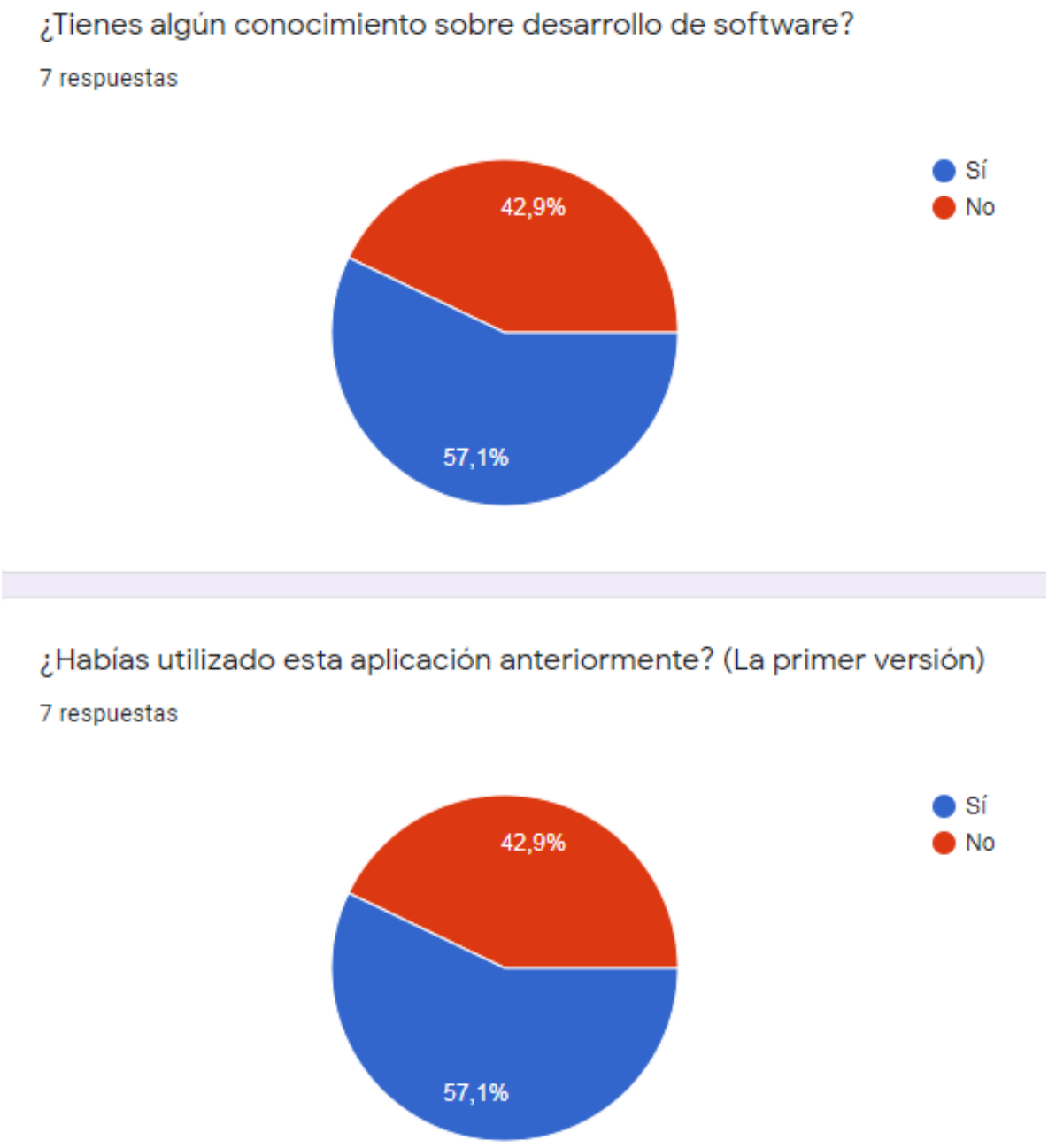

Figura A.2: Lista de personas que contestaron el cuestionario

La tercer pregunta corresponde a la primer impresión del usuario con la aplicación, ya que muchas veces es lo que define si utilizar o no un producto. La figura A.3 muestra las respuestas de los usuarios ante a esta interrogante.

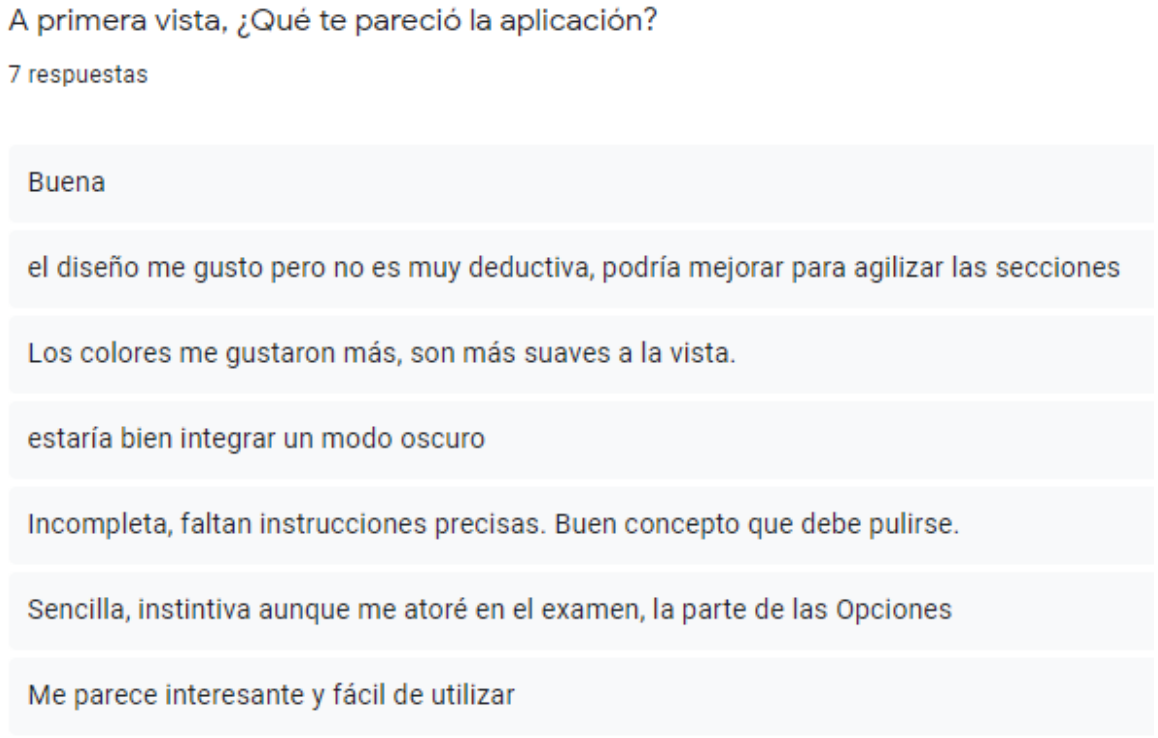

Figura A.3: Primera impresión de los usuarios sobre AGtive

Las preguntas cuatro y cinco se enfocan en conocer la dificultad de los controles iniciales y las primeras impresiones sobre el curso disponible representación entera. Como se puede observar en la figura A.4 se tienen cinco posibles para ambas preguntas. En la primer pregunta se observa que un 42.9% de usuarios tuvieron cierta dificultad para navegar dentro de la aplicación, lo cual es entendible si es la primera vez que se usa una aplicación. El mismo porcentaje aplica para quienes consideraron los controles con una dificultad esperada, mientras que un 14.3% mencionan que no fue nada complicado.

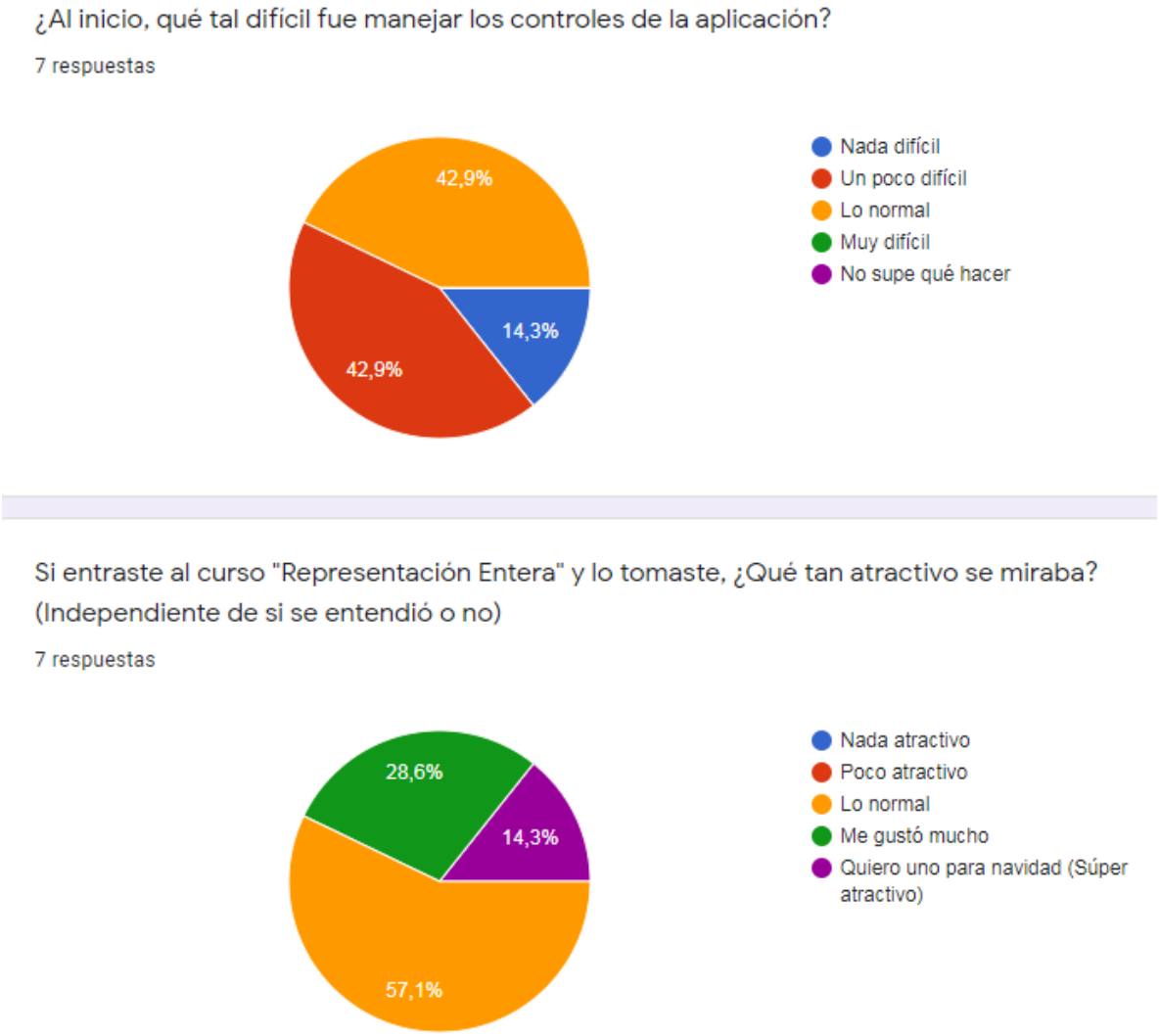

Figura A.4: Primera impresión de los usuarios sobre AGtive

Como AGtive ofrece sólo el curso de representación entera, es importante conocer qué opinan los usuarios a primera vista y si la estructura en que se presenta es adecuada al usuario y fomenta una agradable experiencia de aprendizaje. Lo que se puede observar en la figura anterior es que no hay desagrado o negatividad en la vista de la interfaz gráfica del curso; al contrario, a más de la mitad le parece adecuado, mientras que el porcentaje restante gusta de cómo luce el curso.

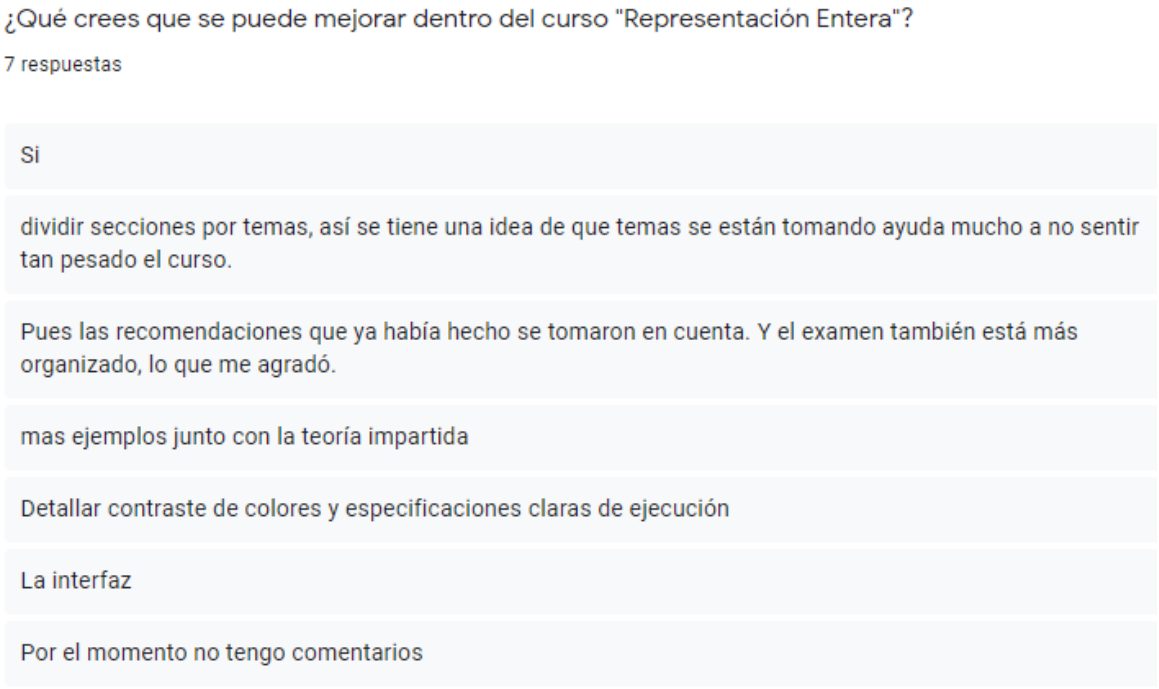

Figura A.5: Comentarios de los usuarios sobre el curso "Representación Entera"

En la figura A.5 se observa la pregunta y respuestas que muestran los comentarios de los usuarios indicando qué posibles cambios pueden hacerse para mejorar AGtive.

La figura A.6 se enfoca en el examen, tanto en el diseño como parte de su funcionalidad. La primer pregunta se enfoca conocer la opinión respecto a la interfaz gráfica del éste, ya que contiene diversas preguntas, y por lo tanto puede ser considerado como "pesado" para el alumno; es por eso que se diseñó a manera de prevenir esto, siempre y cuando se mostrase la información necesaria sin omitir detalles.

La segunda pregunta hace referencia a una característica del examen que es la retroalimentación, y que consiste en hacerle saber al alumno que errores cometió al contestarlo. Como se observa en la figura A.6 todos están de acuerdo en que dicha característica es útil y por tanto es adecuada.

Figura A.6: Comentarios de los usuarios sobre el curso "Representación Entera"

Finalmente se tiene la figura A.7 que muestra la pregunta número siete en donde se le pide al usuario hacer una descripción general sobre las cosas que se pueden mejorar en la aplicación. Esto ayuda a ver detalles que como desarrollador pueden pasar desapercibidos.

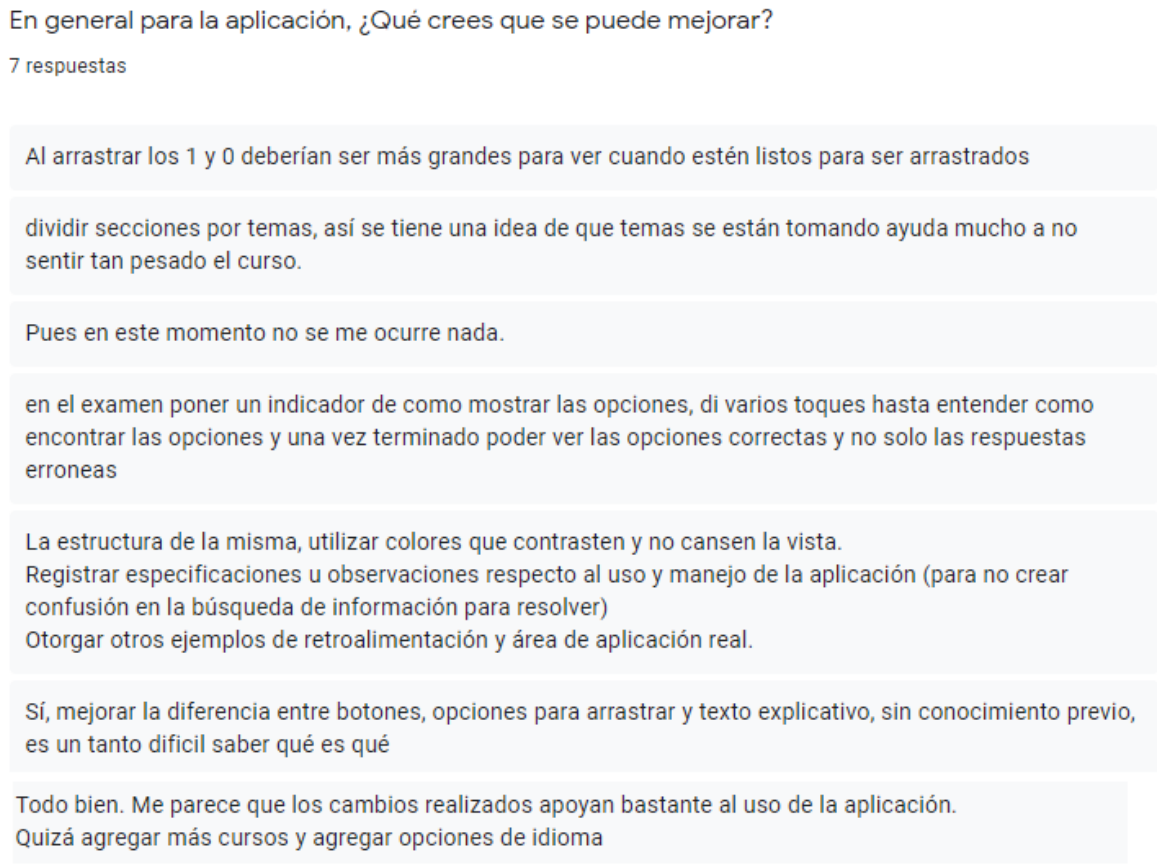

Figura A.7: Opiniones generales sobre mejoras en la aplicación

Apéndice B

Publicación de la aplicación en la Play Store

Compartir un software puede ser un tanto complicado si no se realiza de una manera adecuada. Para este proyecto el comunicarse con varias personas y compartir un enlace o un archivo uno por uno no es algo efectivo del todo, es por eso que se optó por publicar AGtive en la tienda de aplicaciones Playstore de Google, de ese modo los usuarios pueden acceder a descargar la aplicación de una manera sencilla y segura.

Como puede apreciar en la figura B.1, AGtive ya está disponible en la Play Store, y se muestran dos recuadros en color rojo, en donde el primero indica el link para compartir y que al ser cliqueado dentro de un teléfono móvil se abre la tienda de aplicaciones con AGtive listo para instalar; el indicador dos muestra el número de personas que la han calificado con estrellas lo cual se refleja en la figura B.4.

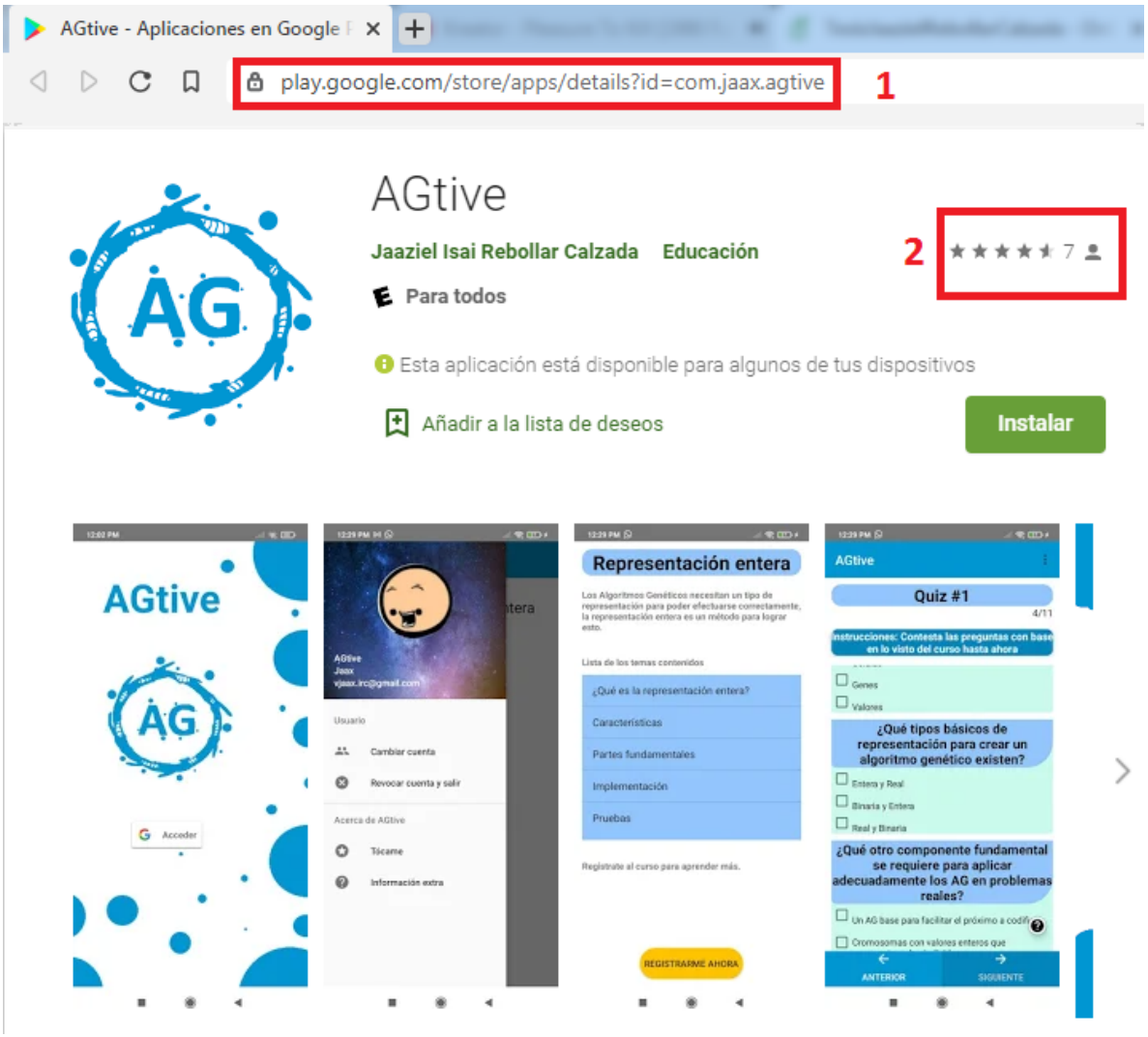

Figura B.1: AGtive en la Playstore

En la figura B.2 se observa en color azul el número de dispositivos que tienen instalada la aplicación; visto de otro modo y de mejor manera se tiene la figura B.3, que también muestra el número de dispositivos que la han desinstalado.

Figura B.2: Panel inicial de la Google Play Console para AGtive

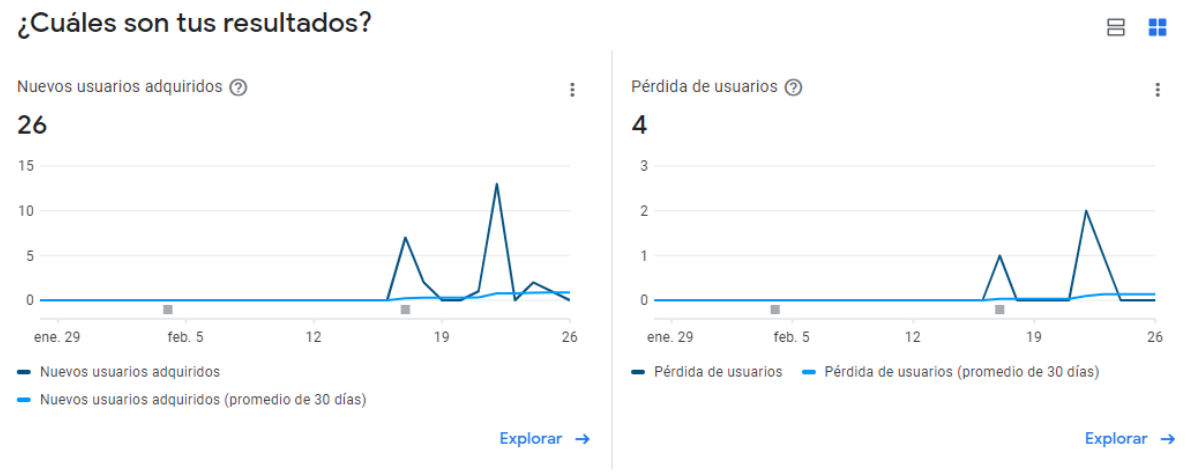

Figura B.3: Grafíca que muestra los dispositivos con AGtive

La figura B.5 indica la fuente de donde los dispositivos han adquirido AGtive, la mayoría han sido por medio del enlace como en la figura B.1 y los demás por medio de búsqueda en la Play Store. Cabe mencionar que de inicio una aplicación nueva es un poco complicado de encontrar, es por eso que se prefiere iniciar por compartir el link directo.

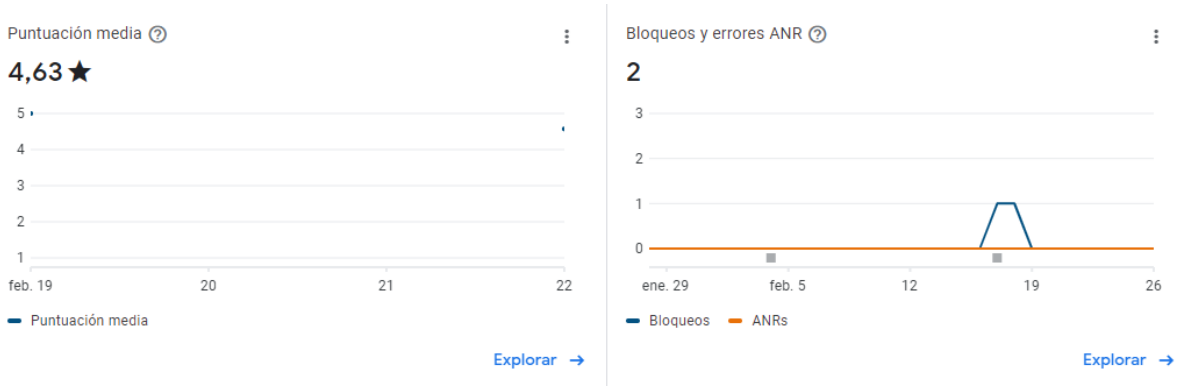

Figura B.4: Calificación general de AGtive y errores o bloqueos presentados

Figura B.5: Número de dispositivos que han adquirido AGtive y la fuente de donde se adquirió

Referencias

[1] 3.0, E. 30 plataformas educativas que ofrecen formación online,. 2018. [Online]. Disponible en: https://www.educaciontrespuntocero.com/formacion/plataformas-de-formacion-line/18508.html, [Accesado el 25-Sep-2018].

[2] Andaluz, A. M. Algoritmos evolutivos y algoritmos genéticos. [Online]. Disponible en: http://www.it.uc3m.es/~jvillena/irc/practicas/estudios/aeag, [Accesado el 27-Enero-2021].

[3] Berzal, F. Algoritmos genéticos,. . [Online]. Disponible en: https://elvex.ugr.es/decsai/iaio/slides/G1%20Evolutionary%20Computation.pdf., [Accesado el 10-Junio-2020].

[4] Cabot, J. ¿kotlin - un java mejorado,. 2017. [Online]. Disponible en: https://ingenieriadesoftware.es/kotlin-java-mejorado/, [Accesado el 02-Junio-2020].

[5] Caparrini, F. S. Algoritmos genéticos,. 2019. [Online]. Disponible en: http://www.cs.us.es/~fsancho/?e=65, [Accesado el 12-Junio-2020].

[6] Castro, S., and de Castro, B. G. *Los estilos de aprendizaje y el aprendizaje: Una propuesta para su implementación.* Universidad Pedagogica Experimental Libertador, Caracas, Venezuela, 2005.

[7] Coello, C. A. C. *Introducción a la Computación Evolutiva.* CINVESTAV-IPN, Av. IPN No. 2508 Col. San Pedro Zacatenco México, DF. 07300, 2009.

[8] Coronel, G. Cuadro comparativo e-learning, b-learning, m-learning y método presencial de educación,. 2015. [Online]. Disponible en: https://es.slideshare.net/gisellecoronel/cuadro-comparativo-elearning-blearning-mlearning-y-mtodo-presencial-de-educacin [Accesado el 15-Junio-2020].

[9] Fantini, A. C. *Enseñanza virtual y Estilos de aprendizaje. Consideraciones para el mejoramiento del rendimiento académico.* UNPSJB, 2008.

[10] Garzon, J. Android 11 a android 1.5: Cada versión de android y sus novedades [fotos],. 2020. [Online]. Disponible en: https://www.cnet.com/es/imagenes/android-11-novedades-actualizacion-android-historia-google/, [Accesado el 02-Junio-2020].

[11] González, J. Android 10: todas sus características y funciones,. 2019. [Online]. Disponible en: https://androidayuda.com/reportajes/software/android-q-android-10-caracteristicas/, [Accesado el 02-Junio-2020].

[12] Guzmán, C. L. La web y los sistemas e-learning,. 2007. [Online]. Disponible en: http://www.biblioweb.tic.unam.mx/libros/repositorios/la_web.htm#22a, [Accesado el 11-Sep-2018].

[13] Gómez, M. A., Romero, V. M. G., Otero, M. M., and Araiza, J. C. *Aprendizaje en línea.* Centro Universitario de la Costa, Universidad de Guadalajara, Puerto Vallarta, Guadalajara, 2005.

[14] Martínez, E. E. G. Ciclo de vida de una aplicación android,. 2015. [Online]. Disponible en: https://edenms.wordpress.com/2015/05/07/ciclo-de-vida-de-una-aplicacion-en-android/, [Accesado el 03-Junio-2020].

[15] Matt Reynolds, S. W. How coronavirus started and what happens next, explained,. 2020. [Online]. Disponible en: https://www.wired.co.uk/article/china-coronavirus, [Accesado el 22-Mayo-2020].

[16] Moujahid, A., Inza, I., and Larrañaga, P. *Algoritmos Genéticos*. Departamento de Ciencias de la Computación e Inteligencia Artificial, 2008.

[17] Ojeda, D. ¿por quÉ kotlin, android?,. 2017. [Online]. Disponible en: http://www.profesor-p.com/2018/08/23/kotlin-vs-java/, [Accesado el 02-Junio-2020].

[18] Puigserver, S., Prats, J. M., and Rovira, J. *Océano Uno: Diccionario Enciclopédico Ilustrado*. Barcelona Océano D.L 1994., Milanesar 21-23, EDIFICIO OCÉANO, 08017, Barcelona, España, 1994.

[19] Ramírez, I. Historia y evolución de android: cómo un sistema operativo para cámaras digitales acabó conquistando los móviles,. 2018. [Online]. Disponible en: https://www.xatakandroid.com/sistema-operativo/historia-y-evolucion-de-android-como-un-sistema-operativo-para-camaras-digitale [Accesado el 02-Junio-2020].

[20] Sánchez, J. J. P. Kotlin vs java,. 2018. [Online]. Disponible en: http://www.profesor-p.com/2018/08/23/kotlin-vs-java/, [Accesado el 02-Junio-2020].

[21] University, J. H. Covid-19 dashboard by the center for systems science and engineering (csse) at johns hopkins university,. 2020. [Online]. Disponible en: https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6, [Accesado el 22-Mayo-2020].

[22] Valdellon, L. Scrum para novatos: Cómo usar scrum para controlar el caos,. 2017. [Online]. Disponible en: https://www.wrike.com/es/blog/scrum-para-novatos-como-usar-scrum-para-controlar-el-caos/, [Accesado el 15-Enero-2021].