Universidad Autónoma Metropolitana – Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Aplicación En Android Como Complemento En El Aprendizaje De Programación Orientada A Objetos

Ingeniería en Computación

Reporte Final de Proyecto Tecnológico

Chávez Pacheco Luis Sergio
2132002872
al2132002872@alumnos.azc.uam.mx

Dr. José Alejandro Reyes Ortiz
Profesor Titular
Departamento de Sistemas
jaro@correo.azc.uam.mx

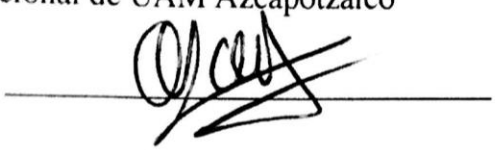
Dr. Leonardo Daniel Sánchez Martínez
Profesor
Departamento de Sistemas
Idsm@correo.azc.uam.mx

Trimestre 2017- Primavera

4 de Septiembre 2017

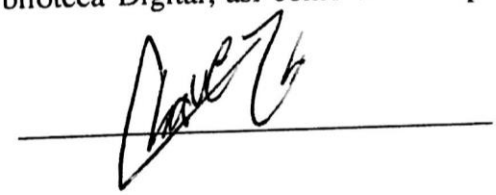
Declaratoria

Yo, Dr. José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco



Yo, Dr. Leonardo Daniel Sánchez Martínez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco

Yo, Chávez Pacheco Luis Sergio, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



RESUMEN

Este proyecto intenta ofrecer una aplicación potente y estable sin dejar de lado el factor de diversión y el de aprendizaje.
El reporte que aquí se presenta tuvo por objetivo el desarrollo de una aplicación móvil que apoye el proceso de enseñanza-aprendizaje de la materia de Programación Orientada a Objetos.

En este reporte de proyecto, se presentará el desarrollo de una aplicación para dispositivos con sistema Android. Se explicará cómo se usaron diferentes tecnologías para realizar el proyecto. Se muestran el diseño, el desarrollo del proyecto y las pruebas del proyecto.
Finalmente se reportan los resultados obtenidos y en el anexo viene el código de cada clase que se uso.

Esta aplicación tiene como finalidad generar conocimiento a los usuarios, de forma ágil, innovadora y divertida con ella se podrá aprender por cuenta propia y en cualquier lugar.

INDICE

INDICE DE FIGURAS

INTRODUCCIÓN

Los alumnos de ingeniería en computación tienen la necesidad de acreditar la UEA[1] de POO[2]. Uno de los objetivos de la asignatura es enseñar las características y conceptos del paradigma orientado a objetos.
Una forma para el aprendizaje es casi un requisito elaborar una réplica casi exacta de la información y conservarla en la memoria.[1] El problema del aprendizaje de la programación orientada a objetos se manifiesta porque es una materia compleja que implica la integración de muchos elementos como es el paradigma orientado a objetos, el lenguaje de programación, el entorno de desarrollo, la métodología de desarrollo, el lenguaje de modelado, los patrones de desarrollo y la lógica de programación. Por lo tanto, los alumnos se encuentran ante una cantidad abrumadora de conceptos en un periodo corto de tiempo, lo que dificulta su asimilación y el desarrollo de las habilidades para generar líneas de código.
En los años recientes es un hecho que la gran mayoría de los estudiantes tienen un celular inteligente a la mano, el uso de esta tecnología hace la vida cotidiana más sencilla. Android es el sistema operativo más usado para los celulares, por esa razón se ha manifestado el aumento de aplicaciones para los dispositivos con este sistema.
El uso de preguntas y los celulares pueden ser una gran herramienta que favorezca al alumno para poder llegar a un mejor entendimiento, porque se puede reforzar el conocimiento, así como a generar nuevos saberes.
Por lo tanto, en este trabajo se creará una aplicación en Android para ayudar al estudiante en la UEA de POO. El objetivo es hacer más fácil el aprendizaje de la teoría del lenguaje orientado a objetos y que pueda ser usados los conceptos en la parte práctica

ANTECEDENTES

Proyectos de integración:

*Aplicación para el apoyo a la enseñanza de la UEA Métodos Numéricos[2].
El proyecto es una aplicación realizada para la UEA de Métodos Numéricos para el apoyo y enseñanza de ésta. Es algo similar a la propuesta, pero a diferencia de esta propuesta, será hacia la UEA de POO, se creará para el uso en equipos Android y hará uso de un módulo con preguntas tipo juego, para que pueda reforzar su conocimiento.

*Aplicación Android para la práctica de verbos compuestos del idioma inglés [3].
Esta aplicación sirve como complemento para el aprendizaje para el idioma inglés. Ambas propuestas tienen similitudes, pero las distinciones a la actual se encuentran, en que la UEA a la que va dirigida, el uso de BD[3] y una interfaz más agradable para el usuario.

* EDAPIX: Una aplicación gráfica para asistir al proceso de enseñanza - aprendizaje de las Estructuras de Datos [7].
Es una aplicación que ayuda al proceso enseñanza-aprendizaje de las estructuras de datos. Las diferencias entre propuestas son: que la nuestra será una aplicación Android, la UEA a la que va dirigida es la de POO y tendrá un módulo de juego de preguntas.

[1]Unidad de Enseñanza-Aprendizaje
[2]Programación Orientada a Objetos
[3]Base de Datos

* Desarrollo de una aplicación Android de tipo quiz[8].
Es un trabajo de un estudiante de España, en el que se realizó una aplicación donde se hacen varias preguntas al usuario sobre una materia. En el proyecto propuesto también tendrá un módulo de preguntas, pero la variante está en que se tendrá un módulo de aprendizaje y un módulo de archivos, donde se podrá estudiar, repasar conceptos o ver ejemplos sencillos de su implementación, esto ayudará a tener un mejor entendimiento sobre la teoría de POO.

.

Software:

*Preguntados [4].
Es un juego de preguntas para Android, en el cual al usuario se le muestra una pregunta de alguna de 6 categorías con sus múltiples respuestas. En nuestra propuesta se usará una BD que estará dentro de la aplicación, así podrá ser usada en cualquier parte y en cualquier momento, sin necesidad de conectarse a Internet.

*Ja Sensei [5].
Es una aplicación que permite aprender japonés sin necesidad de Internet. La diferencia que tendrá con nuestra aplicación será busca facilitar el aprendizaje de POO que es una UEA obligatoria para los estudiantes de ingeniería en computación en la UAM Azcapotzalco.

JUSTIFICACIÓN

La mayoría de los alumnos tienen un problema para aprender los conceptos sobre el lenguaje orientado a objetos, esto se debe a que es demasiada información para comprender y aplicar en tan poco tiempo. Los alumnos podrían ver más atractivo aprender a través de una aplicación de celular, que ir a buscar en un libro.

En esta propuesta se obtendrá una aplicación que buscará ser un complemento para la formación en el aprendizaje de los estudiantes que estén cursando la UEA de POO.
La aplicación tratará de mejorar y hacer más flexible la manera de generar conocimiento y permitirá repasar al alumno días antes de los exámenes haciendo cuestionarios sobre los temas de la UEA.
También puede ser de ayuda para los profesores que estén enseñando la materia, porque con ella se puede hacer más rápido y eficiente la entrega del contenido y material de estudio de la materia.

Este proyecto espera facilitar la enseñanza de una manera educativa el conocimiento elemental del lenguaje orientado a objetos. Además, esta aplicación se podrá usar en cualquier lugar donde no haya Internet y tiene en cuenta que en un entorno de juego se hará más agradable y dinámico el estudiar conceptos.

OBJETIVOS

General.

- Diseñar una aplicación móvil para auxiliar en el aprendizaje de la Unidad de Enseñanza Aprendizaje de Programación Orientada a Objetos.

Específicos

- Diseñar e implementar un módulo que permita al usuario ver y revisar de que se trata cada tema de POO.
- Diseñar e implementar un juego de preguntas.
- Diseñar e implementar un módulo que facilite la búsqueda de un tema en específico.
- Implementar un módulo que permita cargar archivos con la teoría y ejemplos de las lecciones de POO.

MARCO TEÓRICO

Los avances en la ciencia y la tecnología han traído consigo un cambio sustancial en las prácticas de todas las esferas de la sociedad.

DEL E-LEARNING AL APRENDIZAJE MÓVIL [17]

El *e-learning* es un proceso de enseñanza-aprendizaje mediatizado por una computadora, orientado a adquirir ciertas competencias por parte del estudiante. Se caracteriza por el uso de la tecnología web, la interacción con la red de estudiantes, tutores y unos mecanismos adecuados de evaluación. El enriquecimiento por este conjunto de servicios de valor agregado puede ayudar a lograr la máxima interacción entre profesor y alumno, garantizando de esta forma la más alta calidad en el proceso de enseñanza-aprendizaje. Son dispositivos móviles: Tablet, iPod, Smartphone, y otros, los cuales permiten compartir material auditivo, vídeo, texto, imágenes y archivos con los cuales se facilita el proceso de enseñanza-aprendizaje.

Para los estudiantes, las herramientas de aprendizaje móvil resultan más atractivas, ya que logran interactuar con ellas todo el tiempo. La gran ventaja de estos dispositivos es que el estudiante puede llevarlos consigo a donde vaya. Así como puede ayudar a eliminar un poco la formalidad que existe en un método de aprendizaje tradicional, siendo esto más cómodo y amigable para los estudiantes, sobre todo los jóvenes que buscan siempre la oportunidad de aprender en materia tecnológica, además, ayuda a combatir la resistencia al cambio tecnológico que los adultos tienen.

Algunos recursos y aplicaciones que se encuentran disponibles y susceptibles de ser integrados en los ambientes de aprendizaje son los siguientes: blogs, sistemas de administración de cursos, mensajes instantáneos, WIKIS, PODCAST, RSS, espacios sociales y otras 9 herramientas de la web. Estos recursos tecnológicos están siendo integrados en los ambientes de aprendizaje a distancia, multimodales, combinados o de M-LEARNING. El uso y las posibilidades que pueda hacerse con ellos están en relación directa con los aprendizajes que se quiera promover, y, por ello, la creatividad en el diseño juega un papel importante, así como las condiciones de implementación que se realicen para que sean integrados en estos ambientes.

UTILIZACIÓN DE DISPOSITIVOS MÓVILES EN LA EDUCACIÓN VIRTUAL

El aprendizaje móvil se ha convertido en los últimos años en una extensión del E-LEARNING, que permite a los estudiantes planificar sus estudios en diferentes momentos y lugares, sin necesidad de estar conectados, solo necesitan un dispositivo móvil para realizar diversas actividades, tales como envío de trabajos, consulta de textos o acceder a bibliotecas virtuales.

La telefonía móvil es una de las tecnologías que más ha evolucionado en los últimos años al grado de convertirse en un elemento fundamental de las diferentes sociedades. Esta se ha convertido en una brújula indispensable para el ser humano. A diferencia de otras tecnologías, como es el caso de la computadora, los dispositivos móviles, por su bajo costo, están al alcance de la población, su amigabilidad hace que su uso sea accesible; y el avance tecnológico ha logrado que no solo sea un dispositivo de comunicación verbal, sino de envío de mensajes escritos, grabación de voz, videos, agenda, juegos, conectividad en la

web, redes sociales y mucho más; no requiere de una conexión especial, además, es de fácil traslado gracias a su pequeño tamaño.

SISTEMAS OPERATIVOS DE MÓVILES [18]

Un sistema operativo móvil o SO móvil es un sistema operativo que controla un dispositivo móvil al igual que las computadoras utilizan Windows o Linux entre otros. Sin embargo, los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

Los sistemas más usados son:

ANDROID: actualmente Android pertenece a Google, pero es un sistema abierto que permite a cualquier fabricante puede desarrollar en él sus productos.
IOS: (Anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone siendo después usado en el iPod Touch y en el iPad.

DEFINICIÓN DEL SISTEMA OPERATIVO ANDROID [19]

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como Smartphone, tablets, etc. Es desarrollado por la Open Handset Alliance2 la cual es liderada por Google.

Características del sistema operativo Android

Los componentes principales del sistema operativo de Android (cada sección se describe en detalle):

- **Aplicaciones:** Las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

- **Marco de trabajo de aplicaciones**: Los desarrolladores tienen acceso completo a las mismas Apis del *framework* usadas por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes. Cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean remplazados por el usuario.

- **Bibliotecas:** Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de traba jo de aplicaciones de Android. Algunas son: *System C Library* (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

- **Runtime de Android:** Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación

Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato DalvikExecutable (.dex), el cual está optimizad o para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que se transformaron al formato .dex por la herramienta incluida "dx".

• **Núcleo Linux:** Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

La estructura de Android está formada por varias capas: Kernel de Linux, Librerías, Frameworks y aplicaciones. Ver Figura 1.

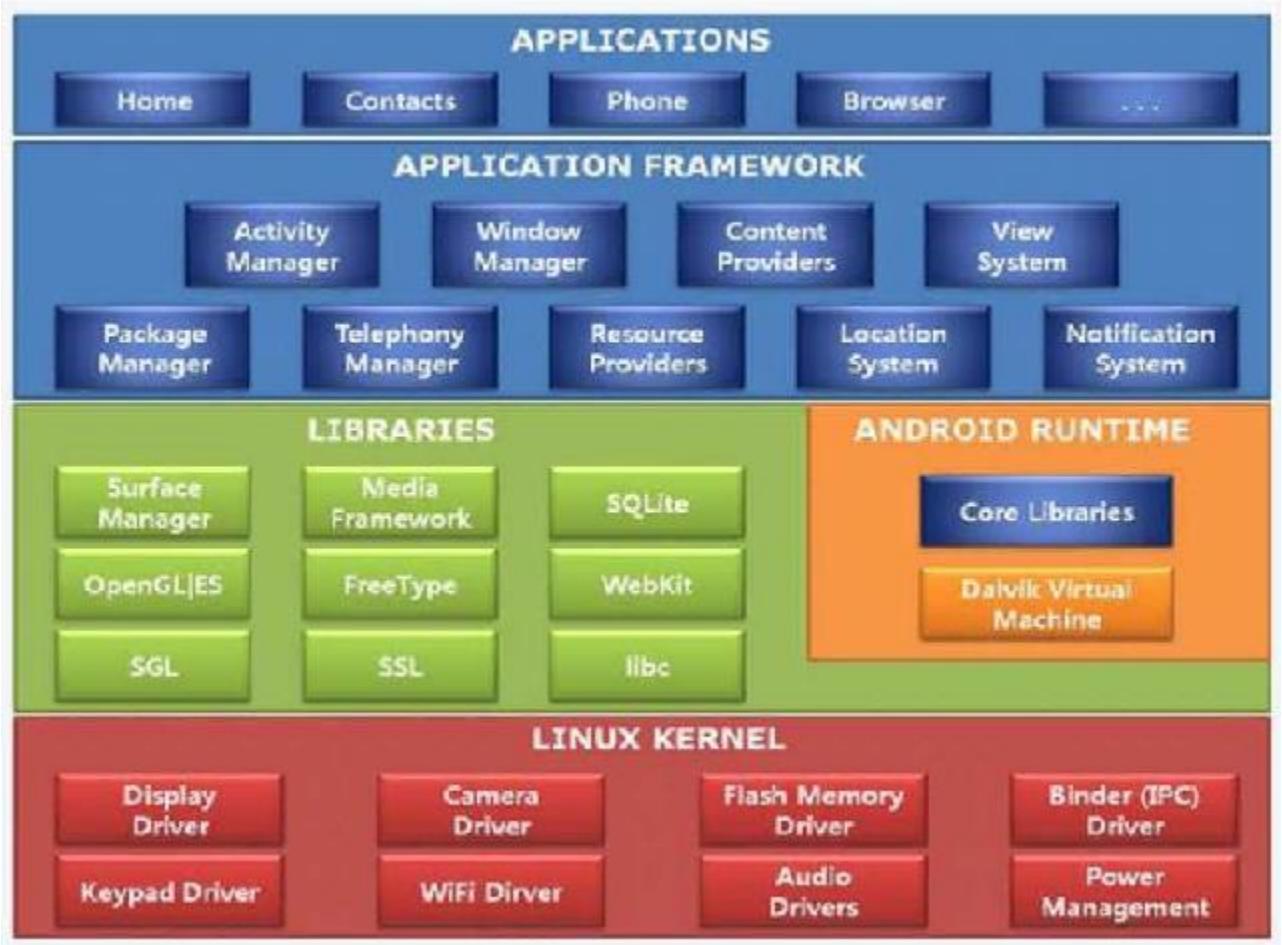


Figura 1: Estructura de Android

ESTRUCTURA DE UN PROYECTO ANDROID [20]

Para poder trabajar con Android se tiene que instalar el entorno de desarrollo Android Studio. Para poder comprender cómo se construye una aplicación Android vamos a revisar como es la estructura general de un proyecto.

Cuando se crea un nuevo proyecto Android en Android Studio se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación, esta estructura será común a cualquier aplicación, independientemente de su tamaño y complejidad.

- **Carpeta /src/** Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Android Studio genera el código básicode la pantalla (Activity) principal de la aplicación.

- **Carpeta /res/** Contiene los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos se distribuyen en las siguientes carpetas:
  - **/res/drawable/.** Contiene las imágenes de la aplicación.
  - **/res/layout/.** Contiene los ficheros de definición de las diferentes pantallas de la interfaz gráfica.
  - **/res/anim/.** Contiene la definición de las animaciones utilizadas por la aplicación.
  - **/res/menú/.** Contiene la definición de los menús de la aplicación.
  - **/res/values/.** Contiene otros recursos de la aplicación como por ejemplo cadenas de texto, estilos, colores, etc.
  - **/res/xml/.** Contiene los ficheros XML utilizados por la aplicación.
  - **/res/raw/.** Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

- **Carpeta /gen/** Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en java dirigido al control de los recursos de la aplicación. El más importante es el fichero R.java, y la case R. Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código a través de este dato.

- **Carpeta /assets/** Contiene todos los demás ficheros auxiliares necesarios para la aplicación, como ficheros de configuración, de datos, etc. La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se pondrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Se usará uno u otro según las necesidades de la aplicación.

- **Fichero AndroidManifest.xml** Contiene la definición en XML, los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono), sus componentes (pantallas, mensajes, etc.), o los permisos necesarios para su ejecución.

METODOLOGÍA DEL PROYECTO

Las funcionalidades generales de la aplicación se muestran en la Figura 2.

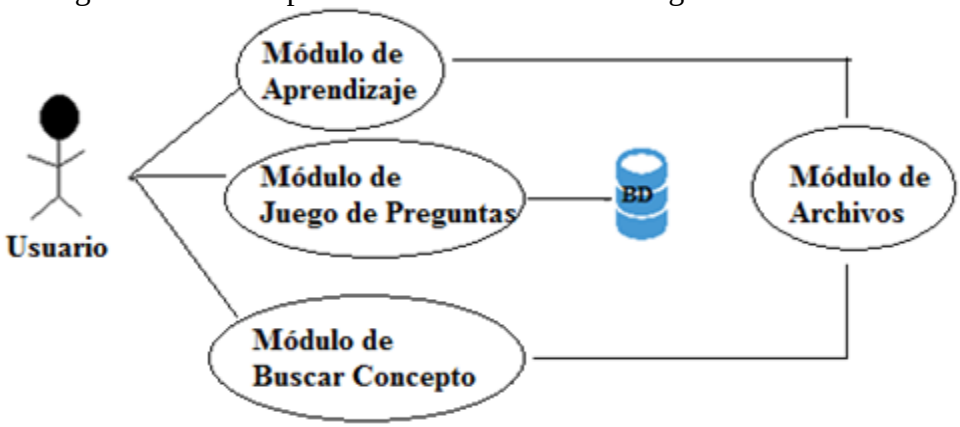


Figura 2: Diagrama de Casos de Uso General

Cada módulo tiene las siguientes funciones:

Módulo de Aprendizaje: Este módulo será el encargado de mostrar al alumno un índice con los diferentes temas y conceptos de POO, cuando le dé clic a un tema aparecerá en la pantalla información acerca del tema de interés.

Módulo de Juego de Preguntas: Su objetivo es mostrar preguntas sobre lecciones de POO, al contestar podrá ver un botón que, al apretarlo, lo llevará a una ventana donde se mostrará más información acerca del concepto del que trató la pregunta. Al terminar el juego, se hará un promedio con las preguntas totales y las correctas. En este módulo se podrán hacer algunos ajustes básicos antes de comenzar el juego.

Módulo de Buscar Concepto: Su principal función es la de obtener el concepto de POO introducido por el usuario y mostrar la información en la pantalla (si es que existe), si no existe, mostrará una pantalla con algunos conceptos válidos que tal vez era lo que quería escribir el usuario.

Módulo de Archivos: Es el módulo que se empleará para almacenar la información de los conceptos que se usarán en el Módulo de Aprendizaje y el Módulo de Búsqueda.

Como ya se ha dicho a lo largo de los puntos anteriores, los objetivos de esta aplicación Android será educar y entretener. Por tanto, debemos elegir una métodología de desarrollo que realce nuestros puntos fuertes y nos ayude con los puntos débiles.

Cada módulo se encargará de implementar la lógica de cada una de sus funciones de tal forma que, aunque todos compartan varias características, cada uno tiene características particulares.

DISEÑO DEL PROYECTO

El proyecto se desarrolla con lenguaje Java en el entorno de Android Studio, con ayuda de la biblioteca SqliteDataBase, se gestionará la BD. Como la aplicación va a leer de archivos, usará la paquetería java.io.

Para reproducir música se aprovechará de la clase MediaPlayer.

Para realizar pruebas rápidas del proyecto se utilizará el emulador BlueStack, porque simula el comportamiento de un dispositivo móvil.

Los conceptos que se contemplan en el proyecto son de los siguientes temas de POO:

1. POO
2. Clases
3. Objetos
4. Herencia
5. Abstracción
6. Encapsulamiento
7. Interfaces
8. Polimorfismo
9. Diagramas UML
10. Relaciones entre Clases

DOCUMENTOS

Para la información que contendrán los documentos de la aplicación, será sacada de libros y documentos en línea. Como los archivos estarán almacenados dentro del dispositivo, se optó porque el tipo de documento fuera .txt, con esto no ocuparía tanta capacidad de almacenaje.

PREGUNTAS

Las preguntas que se usarán en el Módulo de Juego, serán hechas a partir de la información contenida en los documentos creados previamente, se tiene pensado crear un poco más de 40 preguntas, y que cada pregunta tenga su respuesta correcta junto con otras 6 incorrectas, de las que solo se mostrarán 3 incorrectas y la respuesta correcta, haciendo que el juego no sea tan repetitivo.

BASE DE DATOS

La BD se usará en el Módulo de Juego de preguntas, ya que se necesita almacenar las preguntas y las respuestas del juego de la aplicación. La BD estará almacenada en la memoria del dispositivo.

Al querer que el juego pueda ser ejecutado en cualquier parte sin necesidad de tener Internet, elegimos la opción de crear una base de datos local en el móvil del usuario que almacenará las preguntas y sus propios resultados.

En la BD se tendrá una tabla llamada Pregunta que contiene los siguientes atributos:

Id- Es el id de la pregunta, nos ayuda para mostrar la pregunta.
Pregunta- Contiene la pregunta que puede ser mostrada al usuario.
rC- Es la respuesta correcta de la pregunta.
r1, r2, r3, r4, r5, r6- Estas son las respuestas incorrectas, de ellas se elegirán 4, sin embargo, solo 3 junto con la respuesta correcta se mostrarán al usuario.
idTema- Es el id del tema de la cual se trata la pregunta.

INTERFAZ DE USUARIO

Para cubrir los módulos anteriores, necesitaremos al menos las siguientes pantallas en la aplicación:

1. Menú Principal.
2. Pantalla para mostrar un menú con los temas.
3. Pantalla para mostrar el archivo con el tema solicitado.
4. Pantalla para hacer ajustes del juego de preguntas.
5. Pantalla para mostrar las preguntas y sus respuestas.
6. Pantalla para ver el promedio obtenido en el juego.
7. Pantalla para buscar un tema.

DISEÑO DE PANTALLAS

En la Figura 3 se puede ver el diseño de pantallas y se puede comprobar que la parte general de la aplicación ya ha adquirido forma, también se sabe cómo se va a dividir y cómo interactuará cada una de las pantallas con el resto de ellas.

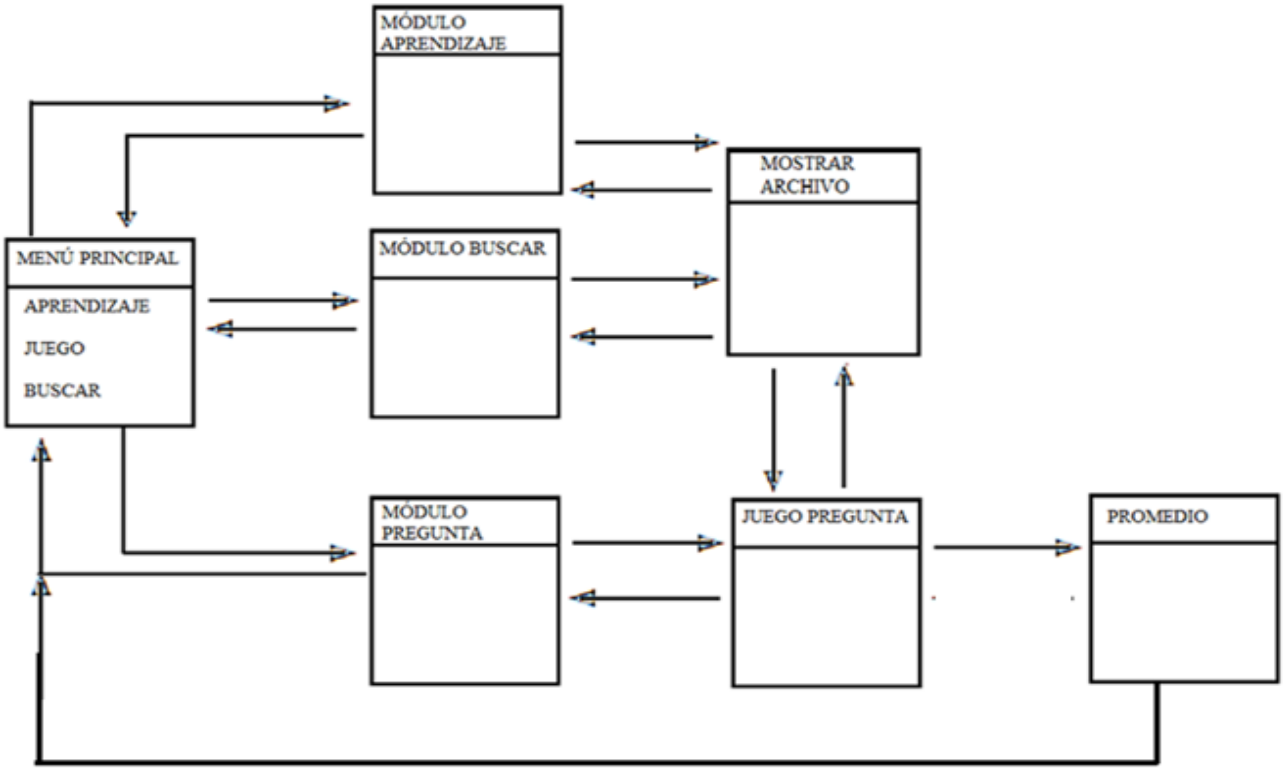


Figura 3: Diseño de Pantallas

DESARROLLO

GENERAL

En esta parte de desarrollo se van a comentar sólo algunos de los puntos más complicados, interesantes o que necesiten explicaciones:

La aplicación solo funcionará con celulares o tablets que tengan un sistema Android 4.03 (IceCreamSandwich). Las interfaces fueron creadas para dispositivos Android con pantallas de 4" o 5", si se instala en pantallas diferentes la interfaz se desformará.

Para las pruebas se tenía pensado usar el emulador BlueStack, pero no funciono, así que se optó por usar el emulador YouWave Android, ya que también simula el comportamiento de un dispositivo móvil y funciono perfectamente al momento de realizar las pruebas rápidas.

ELEMENTOS DE LA APLICACIÓN

Se crearon 7 archivos XML, los cuales servirán como las vistas para el usuario interactúe con el sistema. Se utilizaron un total de 9 clases, de las cuales 7 extienden de la clase *AppCompatActivity*.

En el módulo de archivos, se hicieron 15 documentos extensión TXT, que contienen diferentes temas de la UEA de POO. Para los fondos de pantalla, fondos de botones, iconos, etc. se usaron más de 20 imágenes diferentes, las cuales están guardadas dentro del proyecto en la carpeta *drawable*.

Considerando que los sonidos dentro de un videojuego pueden provocar un gran impacto en la experiencia del usuario, se decidió incluir sonido para indicar al usuario si la jugada efectuada es correcta o incorrecta.

Para los sonidos del juego se usaron canciones estilo 8 bits. Los nombres de los archivos son op1[9], op2[10], op3[11], op4[12], en1[13], en2[14], en3[15], en4[16].

Al querer hacer uso de botones interactivos con el usuario, las clases implementan *View.OnClickListener*() y se implementaron los métodos declarados en ella.

Para usar una BD en nuestra aplicación, lo que hicimos es crear una clase que herede de *SQLiteOpenHelper*. Esta clase nos permite crear la BD y actualizar la estructura de tablas y datos iniciales.

En la BD tiene un total de 56 preguntas de 15 temas diferentes, cada pregunta tiene su respuesta correcta y 6 respuestas incorrectas.

CASOS DE USO

Para explicar el proceso de análisis y diseño la aplicación, se tomará cada módulo y para cada uno se detallará su caso de uso de texto.

MÓDULO DE APRENDIZAJE
Descripción: Caso de uso que muestra el funcionamiento del Módulo de Aprendizaje. **Actores Primario:** Usuario. **Secundario:** N/A. **Disparador:** El actor primario solicita en el menú principal comenzar con el módulo de aprendizaje **Precondición**: El actor primario apretó el botón de Aprendizaje en el menú principal. **Pos-condición:** Se mostrará un archivo con el contenido del tema que el usuario eligió. **Flujo de eventos:** 1. El sistema carga el menú de los temas. 2. El Jugador presionará algún icono de los temas mostrados por el sistema. 3. El sistema con el id del tema buscará el archivo. 4. El sistema abrirá y mostrará en la pantalla el contenido del tema.

TABLA 1:CASO DE USO DE TEXTO- MÓDULO APRENDIZAJE

MÓDULO JUEGO DE PREGUNTAS
Descripción: Caso de uso que muestra el funcionamiento del Módulo de Juego de Preguntas. **Actores Primario:** Jugador. **Secundario:** N/A. **Disparador:** El actor primario solicita en el menú principal comenzar con el módulo de juego. **Precondición**: El actor primario apretó el botón de Juego en el menú principal. **Pos-condición:** Se mostrarán todas las preguntas que se mostraron y una calificación obtenida al término del juego. **Flujo de eventos:** 1. El sistema carga la base de datos de las preguntas. 2. El sistema muestra un menú. 3. En este menú, el Jugador debe elegir el número de preguntas que quiere que se le muestren, la dificultad para contestar las preguntas, si apagar los sonidos y si mostrar el archivo del tema al que corresponde la pregunta. 4. El sistema verifica si el número de preguntas es mayor a cero y menor a 50(el número máximo de preguntas a mostrar), si no es válido el número el Jugador no podrá seguir al evento 6. 5. El sistema crea un número aleatorio entre 1 y 56, este número será el id de la pregunta que se mostrará al jugador. 6. El sistema genera 4 números aleatorios entre 1 y 6, estos números serán las 4 de 6 respuestas de la pregunta generada en el evento 5. 7. El sistema genera un número aleatorio entre 0 y 3, este número será posición en que se mostrará la respuesta correcta. 8. El sistema busca en la BD el id de la pregunta con esto recuperará la pregunta, la respuesta correcta, 4 de las 6 respuestas incorrectas y el id del tema al que pertenece la pregunta.

9. El sistema presenta al Jugador la pregunta y 4 respuestas (1 correcta y 3 incorrectas).
10. El sistema aumenta el contador de número de preguntas mostradas en 1.
11. El Jugador debe elegir alguna de las respuestas.
12. El sistema verifica si la respuesta fue correcta o incorrecta, si fue correcta aumenta el puntaje, caso contrario el puntaje queda igual.
13. El sistema muestra el botón de siguiente.
14. El Jugador presiona el botón siguiente.
15. El sistema verifica si el contador de número de preguntas mostradas es menor al número de preguntas solicitadas por el Jugador, si el contador de número de preguntas mostradas es menor, el sistema regresa hasta el evento 3, caso contrario sigue al evento 16.
16. El sistema hace un promedio entre las respuestas correctas y las preguntas totales mostradas.
17. El sistema muestra una pantalla con la calificación y todas las preguntas que se le mostraron al Jugador.

Flujo Alternativo:

9. El jugador eligió una dificultad normal o difícil, el sistema muestra al Jugador la pregunta y 4 respuestas (1 correcta y 3 incorrectas) y una cuenta regresiva de 26 o 15 segundos, según sea el caso.
   9.1 El flujo continúa en el punto 10.

11. El Jugador eligió una dificultad normal o difícil, y el tiempo límite para contestar la pregunta se acaba.
12. El sistema da como incorrecta la pregunta.
    12.1 El flujo de eventos continua en el evento 13

13. El Jugador SÍ eligió la opción de mostrar detalles entre preguntas, el sistema muestra el botón de siguiente y el botón de detalle.
    13.0 El jugador NO presiona el botón de detalle, el flujo sigue en 14.
    13.1 El Jugador presiona el botón de detalle.
    13.2 El sistema muestra el archivo que contiene la información sobre la pregunta.
    13.3 El usuario aprieta el botón de regresar del celular.
    13.4 El sistema regresa a la pantalla donde se muestra la pregunta.
    13.5 Los eventos siguen en el evento 14.

TABLA 2: CASO DE USO DE TEXTO- MÓDULO JUEGO DE PREGUNTAS

MÓDULO DE BÚSQUEDA DE CONCEPTO
Descripción: Caso de uso que muestra el funcionamiento del Módulo de Búsqueda. **Actores Primario:** Usuario. **Secundario:** N/A. **Disparador:** El actor primario solicita en el menú principal comenzar con el módulo de Búsqueda. **Precondición**: El actor primario apretó el botón de Búsqueda en el menú principal. **Pos-condición:** Se mostrará un archivo con el contenido del tema que el usuario eligió.

Flujo de eventos:

1. El sistema muestra una lista con todos los temas que se encuentran en el Módulo de Archivos.
2. El actor principal presionará algún tema de la lista mostrada por el sistema.
3. El sistema con el id del tema buscará el archivo.
4. El sistema abrirá y mostrará en la pantalla el contenido del tema.

Flujo Alternativo:

2 El Jugador presiona el icono de Buscar que se encuentra en la barra.
- 2.1 El Jugador ingresa una palabra.
- 2.2 El sistema filtra de la lista aquellos temas que contengan la palabra ingresada por el actor principal.
- 2.3 El sistema muestra una lista filtrada al actor principal.
- 2.4 El actor principal presionará algún tema de la lista filtrada por el sistema.
- 2.5 El flujo sigue en el paso 3.

TABLA 3: CASO DE USO DE TEXTO- MÓDULO BUSQUEDA

CLASES

En la Figura 4, se muestra una representación en bloques de las clases que se generaron con ayuda del plugin de Code Iris para Android Studio.

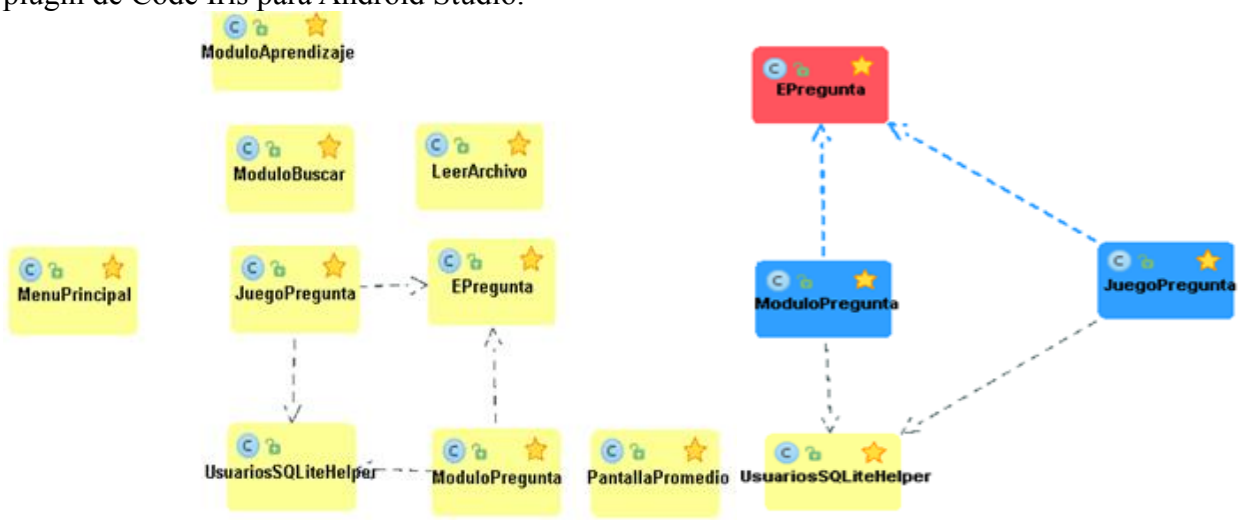


Figura 4: Diagrama de Clases con Code iris

A continuación, se explicará a grandes rasgos cada clase y los métodos usados en ella, en el anexo se puede ver con más detalle la implementación de cada clase.

CLASE *MENUPRINCIPAL*

La clase *MenuPrincipal* es la clase que nos permite controlar los eventos de los botones de la pantalla principal, para poder realizar eso debe de extender de *AppCompatActivity* y debe de hacer un implements de *View.OnClickListener*.

Para poder manipular los botones de la pantalla principal, dentro del método *onCreate()* se deben crear variables de tipo botón y se deben obtener los elementos que se desean con ayuda de *findViewById()*. Lo anterior se usa siempre que se quiera utilizar de los elementos de las pantallas.

El método *OnClickListener(),* lo que hace es que al presionar un botón lo lleve a la pantalla que se necesita.

CLASE *MODULOAPRENDIZAJE*

La clase *ModuloAprendizaje*, es la encargada de controlar la pantalla de módulo de aprendizaje, lo que debe hacer es saber que icono fue presionado, si el icono fue pulsado 1 vez, se debe mandar un mensaje con el nombre del tema al que corresponde ese icono, esto se logra con una variable llamada *Toast*, si fue presionado 2 veces se abrirá una pantalla mostrando el archivo que se solicitó. También es la que se encarga de mandar la id del tema a otra clase para que el archivo pueda abrirse y mostrarse al usuario. Extiende de *AppCompatActivity* y debe de hacer un *implements* de *View.OnClickListener*.

Tiene los métodos *onCreate()* y *OnClickListener()*, que ya se mencionó para que se usa cada uno de ellos.

Su método *irLeerArchivo(),* con el id del icono se recupera el id del tema y lo envía a la clase que maneja la pantalla en la que se mostrará el archivo, que es la que contiene el método para abrir y leer al archivo. También este método abre la siguiente pantalla.

CLASE *LEERARCHIVO*

La clase *LeerArchivo,* es la clase encargada de recibir el id del tema, abrir y mostrar el archivo solicitado en la pantalla. Extiende de *AppCompatActivity*.

Un *Layout* es la ventana o vista que el usuario ve en la pantalla: botones, textos, barras de desplazamiento, etc. La asociación se hace en el método onCreate que ya se mencionó para que se usa cada uno de ellos.

Tiene un método llamado *leerArchivo(),* con el que se abre, se lee y se muestra en la pantalla, después de usar leer el archivo lo cierra.

CLASE *MODULOPREGUNTA*

En la clase *ModuloPregunta,* se usa para crear la BD, para insertar las preguntas con sus respuestas, obtener los valores de los ajustes para el juego y enviarlos a la siguiente clase. Extiende de *AppCompatActivity* y debe de hacer un implements de *View.OnClickListener*.

Tiene los métodos *onCreate()* y *OnClickListener()*, que ya se menciono para que se usa cada uno de ellos. Su método *generarValor(),* sirve para generar un nuevo registro para que pueda ser insertado a la BD del juego.

El método *reproduceMusica(),* es para reproducir una canción aleatoria al momento de ingresar a la pantalla de ModuloPregunta.

CLASE *JUEGOPREGUNTA*

La clase *JuegaPregunta,* fue la clase más difícil de implementar, porque se debía ingresar a la BD, buscar preguntas para mostrarlas al usuario, las preguntas que se mostrarían al usuario tenían que ser diferentes y en un orden aleatorio, las respuestas también tenían que ser diferentes, también la respuesta correcta se debería visualizar en diferentes botones, es decir, si salía la pregunta 1, la respuesta correcta se podría mostrar en el botón 1, botón 2, botón 3 o en el botón 4, esto era para que el juego fuera más entretenido.

Esta clase también es la encargada de recuperar los valores de los ajustes que hizo el usuario en la clase *ModuloPregunta.* Asi que también se tendría que implementar un contador de dificultad, reproductor de música, un contador de preguntas mostradas y un enlace para mostrar archivos, todo esto por si el usuario lo deseaba. Al último, tenía que enviar las preguntas mostradas al usuario junto con el promedio sacado a la siguiente pantalla. Para que nuestro juego hiciera todo lo que queríamos, se implementaron varios métodos que a continuación se describen.

Primero esta clase extiende de *AppCompatActivity* y debe de hacer un *implements* de *View.OnClickListener*.

Para asociar los botones, editText de la pantalla con la clase y además poder tener eventos con los botones, se usaron los métodos *onCreate()* y *OnClickListener()*.

Se creo un método llamado *listaPreguntasAleatorio(),* este método lo único que hace es llenar un arreglo de enteros con numero aleatorios, que simularan las id de las preguntas en la Base de Datos, servirá para que las preguntas se muestren aleatoriamente.

El método *listaRespuestasAleatorio(),* ayuda para que se llene un arreglo de enteros con números aleatorios, que simularan las respuestas que se mostraran al usuario de la Base de Datos, servirá para que las respuestas sean diferentes.

El método *resAle(),* sirve para que se creé un entero aleatorio, el cual, será un lugar en un arreglo que contendrá a la respuesta correcta.

El método *pregunta(),* con este método se busca en la Base de Datos una pregunta, recibe un *string* con el id de la pregunta, un arreglo de enteros que servirá para que las respuestas se muestren aleatoriamente y un arreglo que contiene la pregunta que queremos que se muestre. También ayuda para guardar una cadena que contendrá las preguntas y sus respuestas correctas que será enviada a la siguiente pantalla.

resElegida(), es el método que ayudará para saber si el botón presionado por el usuario contiene la respuesta correcta o incorrecta. Mostrará un mensaje si la respuesta fue correcta o incorrecta y el botón cambiara de fondo según sea el caso. Si la respuesta fue correcta se incrementará en uno un contador de respuesta correcta.

El método *nuevaPregunta(),* este método nos servirá para que despúes de elegir una respuesta de una pregunta, se bloqueen los botones y se encargará de mostrar el botón de siguiente y si es necesario también el botón detalle. Tambien si se aprieta el botón de siguiente generará una nueva pregunta o calcule el promedio de respuestas correcta que tuvo el usuario. Enviará el promedio y las preguntas mostradas a la siguiente pantalla.

CLASE *PANTALLAPROMEDIO*

Es la clase que se encargará de recuperar el promedio y la lista de preguntas que fueron mostradas al usuario durante el juego. Extiende de *AppCompatActivity.*

Tiene un método llamado *obtenerCalificacion()* para obtener una calificación dependiendo el promedio del usuario.

Tiene el método *reproduceMusica()*, este método reproducirá(si el usuario activo esta opción) alguna de las 4 canciones que se encuentran en la carpeta raw.

CLASE *MODULOBUSCAR*

Esta clase es la que tiene los métodos de buscar y filtrar los temas que el usuario busque. Es la que se encarga de mandar la id del tema a otra clase para que el archivo pueda abrirse y mostrarse al usuario. Extiende de *AppCompatActivity.*

La función *onCreateOptionsMenu* permite instanciar el menú mediante *MenuInflater*. El método *onQueryTextChange()* es para que se modifica el texto de búsqueda. Con *onQueryTextSubmit(),* con este método se solicita realizar la búsqueda pulsando el botón "Enter" del teclado.

El método *buscarArchivo()* recuperara el id del tema y lo el método *irLeerArchivo(),* se encarga que con el id del tema lo envié a la clase que maneja la pantalla, en la que se mostrará el archivo, que es la que contiene el método para abrir y leer al archivo, este método también abre la siguiente pantalla.

CLASE *USUARIOSSQLITEHELPER*

La clase *UsuariosSQLiteHelper,* es la encargada de crear la BD y la tabla que se va a usar. Extiende de SQLiteOpenHelper.

La clase *UsuariosSQLiteHelper* tiene tan sólo un constructor, que normalmente no necesitaremos sobrescribir, y dos métodos abstractos, *onCreate()* y *onUpgrade*(), que deberemos personalizar con el código necesario para crear nuestra base de datos y para actualizar su estructura respectivamente.

Lo primero que hacemos es definir una variable llamado *sqlCreate* donde almacenamos la sentencia SQL para crear una tabla llamada Preguntas con los campos que se necesitan.

El método *onCreate(),* se llama al crear la base de datos. Es decir, una única vez (cuando la base de datos no exista). es aquí donde se crearán las tablas.

El método *onUpgrade(),* se llamará para hacer una modificación sobre la base de datos. Para que fuera más simple el manejo de la BD, en este método se decidió la opción de eliminar la tabla anterior y crearla de nuevo.

CLASE *EPREGUNTA*

Esta clase, solo es la plantilla para crear objetos del tipo Pregunta. Contiene sus respectivos constructores, sus métodos get y sus métodos set.

PRUEBAS DE LA APLICACIÓN

Las pruebas han sido muy sencillas de preparar y realizar. Tras terminar cada prototipo las pruebas realizadas han sido las mismas. Las pruebas que se hicieron fueron: Pruebas por Módulo, Prueba Completa y Prueba con Usuarios.

Prueba por Módulo : para este tipo de prueba, primero se crea un módulo (solo las funciones, sin darle una buena vista) y lo checábamos en el simulador YouWave. Así se solucionaban de forma rápida problemas que se causaban en ese módulo. Estas pruebas se hacían en todos los módulos, si la pasaban, se procedía a mejor la vista para el usuario caso contrario, se buscaban los errores y se corregían.

Prueba Completa: son pruebas donde han sido probados todos los módulos en conjunto. La forma de probarlos ha sido la siguiente: verificar que haga su función, comprobar que toda la funcionalidad que ofrece la pantalla funciona correctamente y que estén bien integradas con el resto de la aplicación. Esta prueba se realizaba en el simulador YouWave. Con esta prueba, salía un prototipo final.

Prueba con Usuarios: Una vez se tenía un prototipo finalizado, se realizaba una prueba completa, pero se les prestaba la aplicación a compañeros de Computación, para que la utilizarán como si fueran el usuario final.

RESULTADOS FINALES

Las capturas de pantalla que a continuación se muestran son los resultados obtenidos de un dispositivo con una pantalla de 5 pulgadas.

Al ejecutarse la aplicación comenzamos en la pantalla principal, ver Figura 5. Que contiene 3 botones para realizar alguna de las 3 tareas que hace nuestro proyecto.

Figura 5: Pantalla Principal

Cuando se presiona el botón de Aprendizaje del menú principal, nos lleva a la pantalla donde comienza el trabajo del módulo Buscar, como se muestra en la Figura 6. Para poder ver algún archivo de los temas que aparecen se debe apretar 2 veces el icono, si solo se hace una vez, solo se mostrará el nombre del tema como se aprecia en la Figura 7.

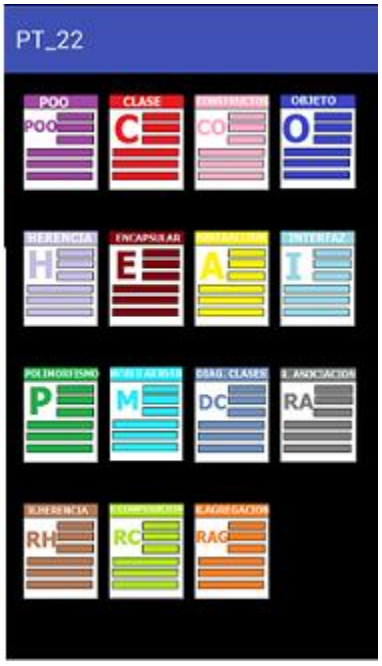


Figura 6: Menú Aprendizaje

Figura 7: Pantalla al Tocar un Icono

La pantalla para ver los archivos que usan los 3 módulos es algo simple como se ve en la Figura 8.

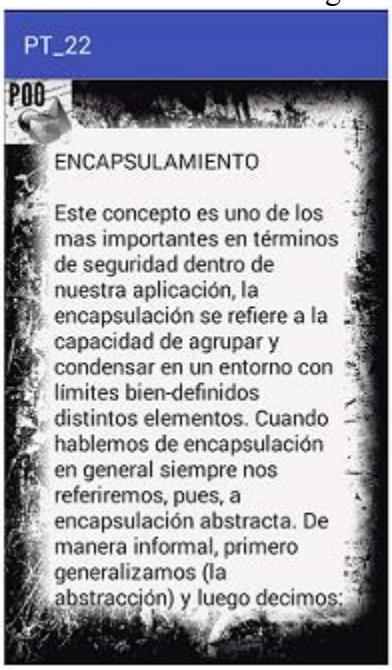


Figura 8: Pantalla para leer los Archivos

Al presionar el botón de Juego del menú principal nos lleva a la pantalla de la Figura 9, en ella se pueden hacer algunos ajustes para el juego. Para comenzar el juego solo apretamos el botón Comenzar.

Figura 9: Pantalla Para Ajustes del Juego

Cuando comenzamos el juego nos mostrará alguna de las pantallas de la Figura 10, la pantalla que se mostrará depende según los ajustes que hizo el usuario.

Figura 10: Diferentes Pantallas del Juego. Del lado izquierdo se eligió la dificultad Fácil, del lado derecho la dificultad es Normal o Dificil

Si se contesta bien la pregunta del juego, la pantalla lucirá como en la Figura 11, de lo contrario se verá como en la Figura 12, en ambos casos se desbloqueará el botón de “siguiente”.

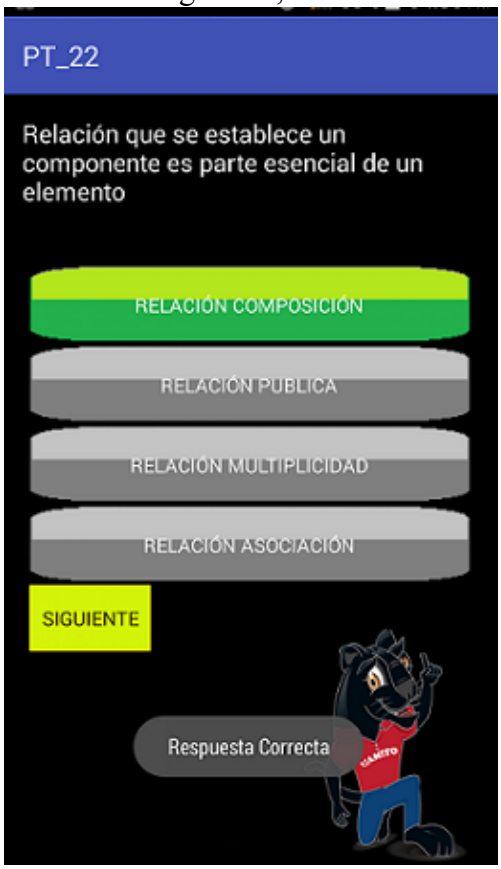


Figura 11: Pregunta contestada Correctamente

Figura 12: Pregunta Contestada Incoorectamente

Está pantalla puede verse de forma diferente, dependiendo de los ajustes que haya hecho el usuario. Ver Figura 13.

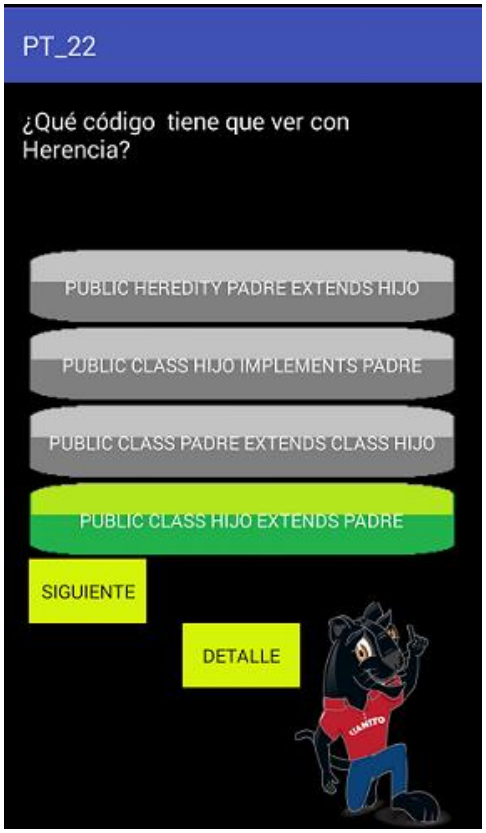


Figura 13: Pantalla con Botón "Detalle" Habilitado

La pantalla de promedio se presentá con todas las preguntas que se le mostrarón al usuario y una calificación. Ver Figura 14.

.

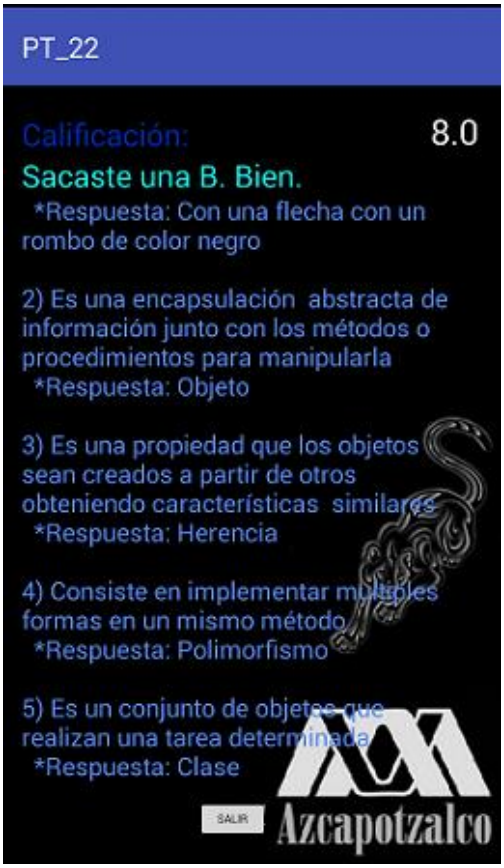


Figura 14: Pantalla Promedio

Al oprimir el botón de Buscar de la pantalla principal, nos lleva a la pantalla de la Figura 15. En esa pantalla se puede buscar una palabra tocando el icono de la lupa. Ver Figura 16.

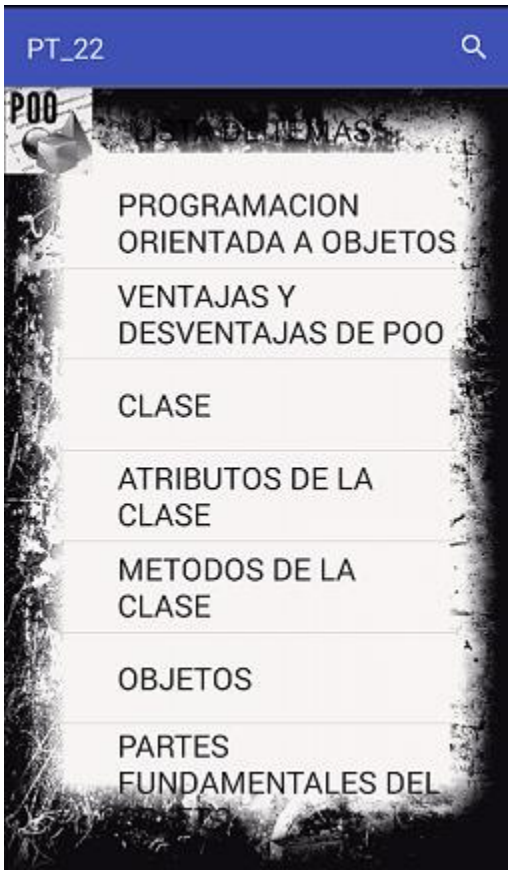


Figura 15: Pantalla para Buscar Tema

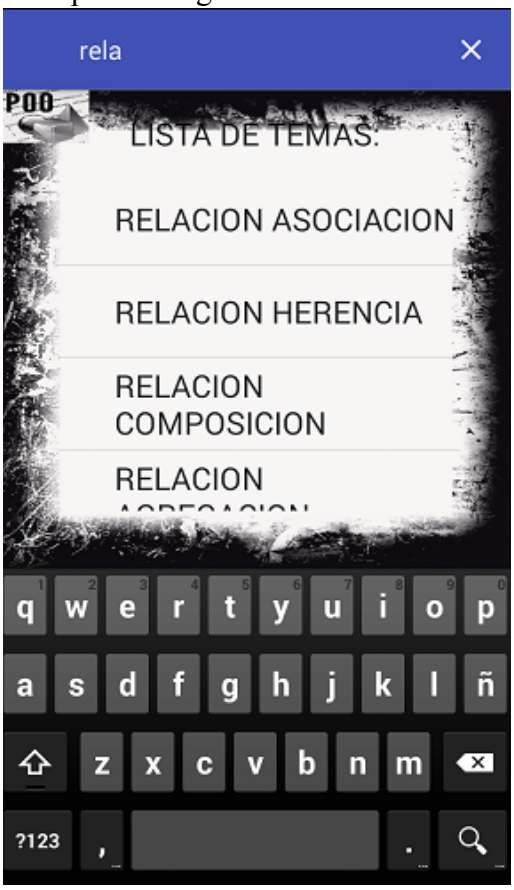


Figura 16: Pantalla al Filtrar Resultados según la Palabra Insertada

Las siguientes imágenes, son los resultados de un dispositivo con una pantalla de 7 pulgadas

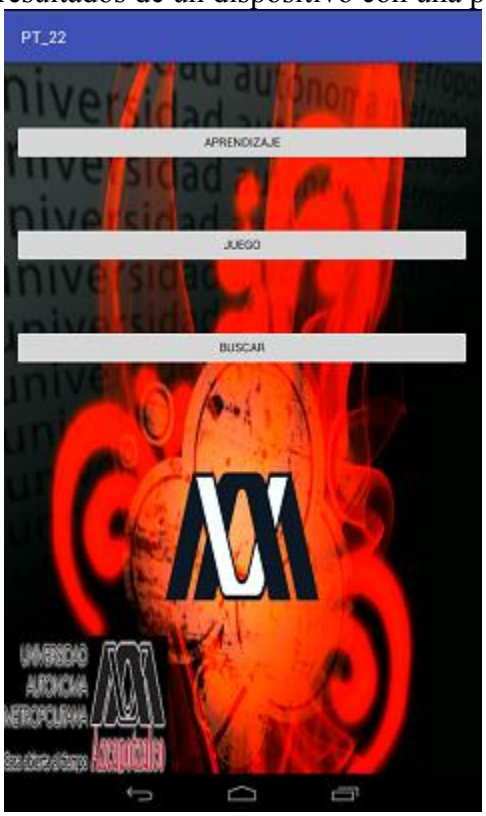


Figura 17: Pantalla Principal 7"

Figura 18: Módulo Aprendizaje 7"

Figura 19: Pantalla al Tocar un Icono 7"

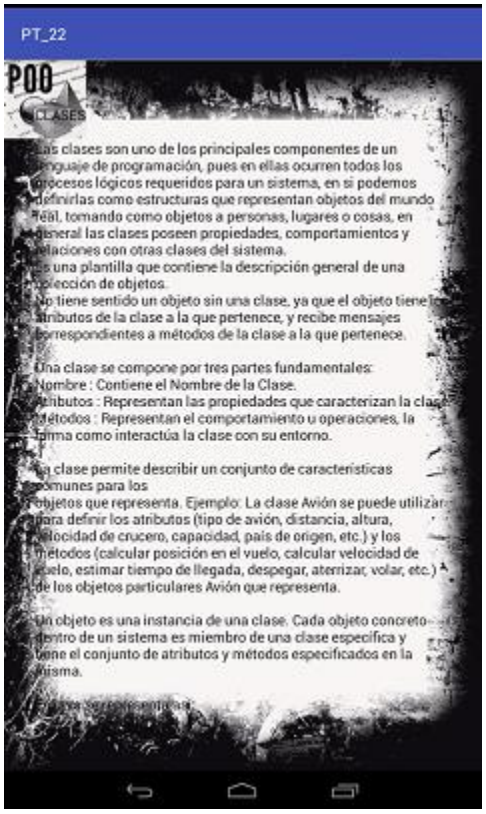


Figura 20: Pantallla para Leer un Archivo 7"

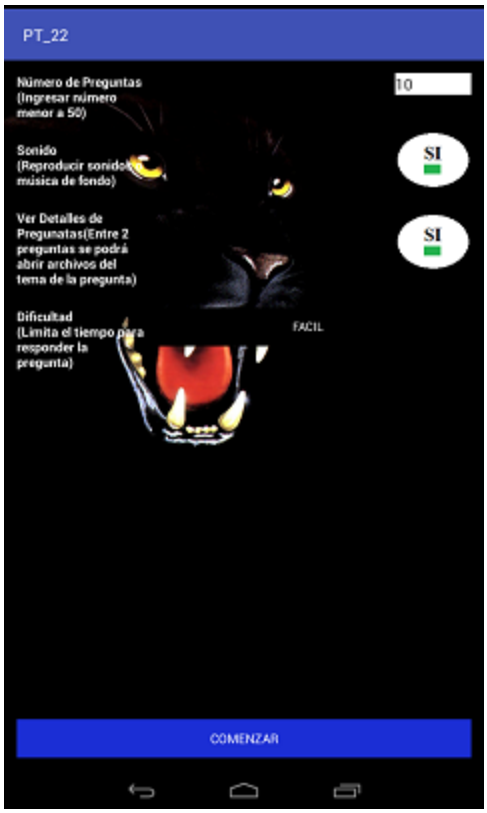


Figura 21: Pantalla para Ajustes del Juego 7”

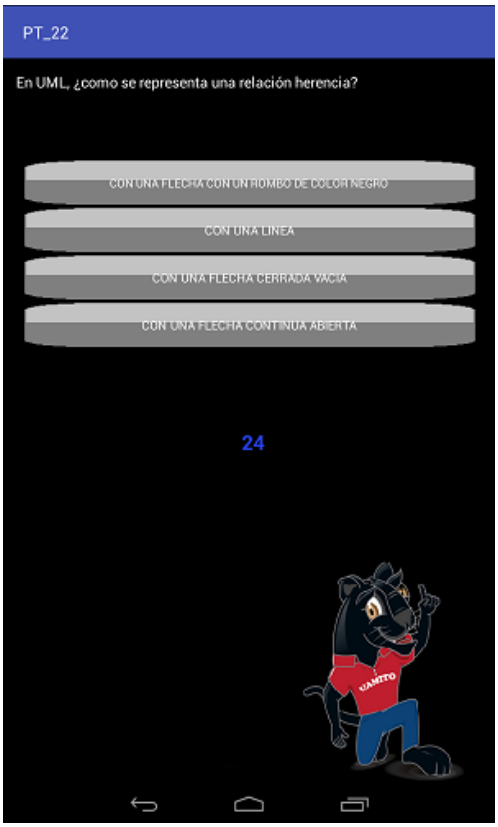


Figura 22: Pantalla de Juego con Limite de Tiempo 7"

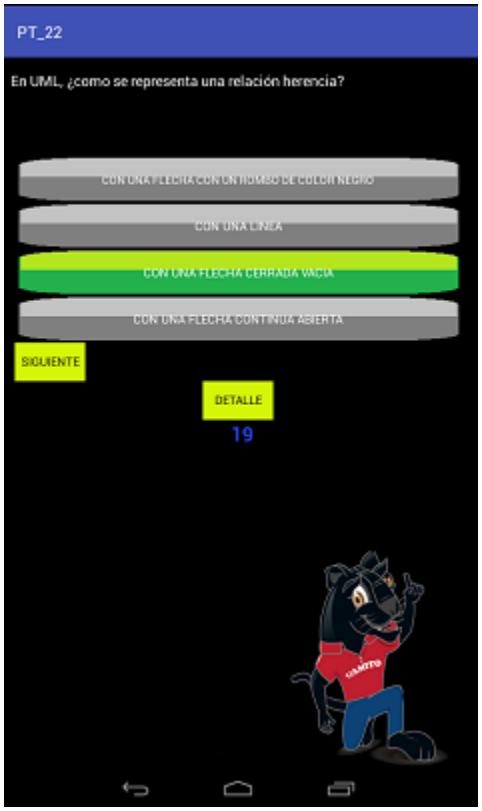


Figura 23: Pantalla al Contestar Correctamente 7”

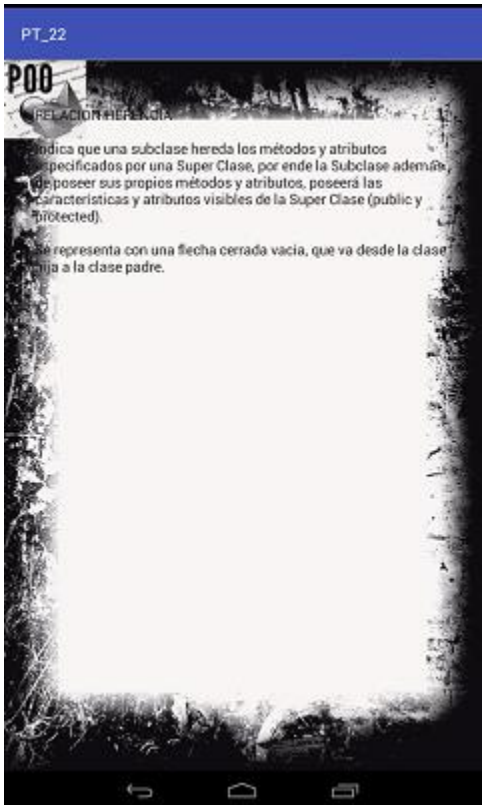


Figura 24: Pantalla al Presionar Detalle 7"

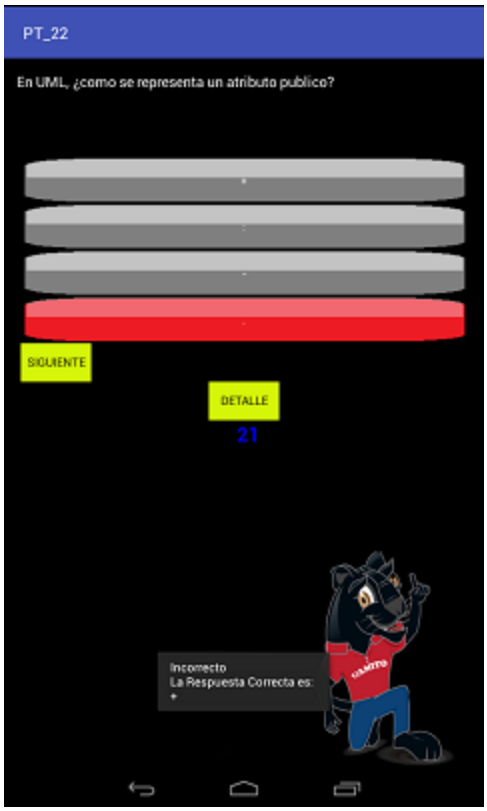


Figura 25: Pantalla Respuesta Incorrecta 7"

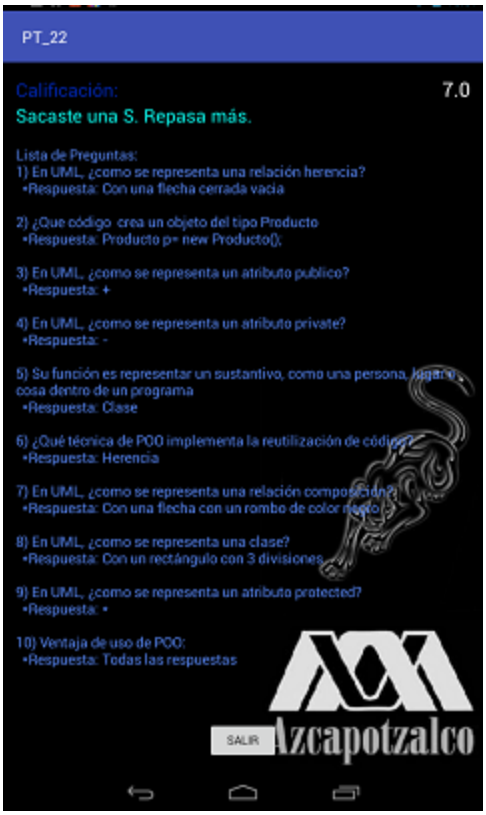


Figura 26: Pantalla con promedio y las Preguntas Mostradas 7”

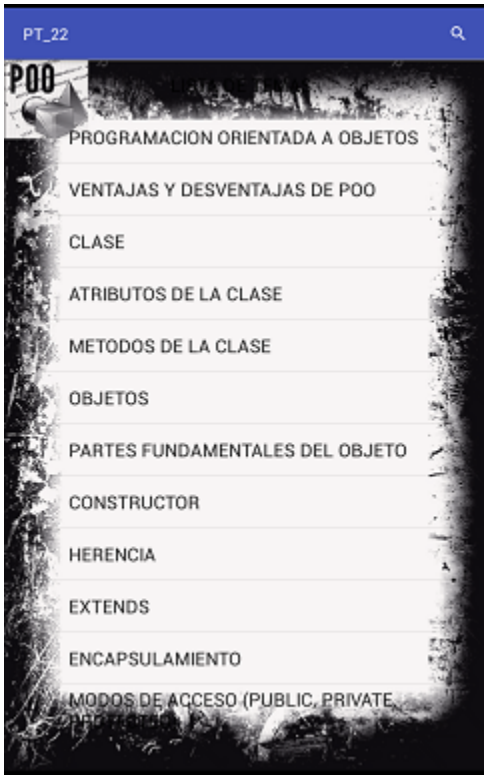


Figura 27: Pantalla para Buscar Tema o Concepto 7”

*Fig*ura 28: Pantalla al Filtrar Resultados, segun la Palabra Insertada 7”

Figura 29: 2da. Pantalla al Filtrar Resultados, segun la Palabra Insertada 7”

En las ultimas pantallas mostradas, nos damos cuenta que la aplicación no se muestra uniformemente en la pantalla del dispositivo, la razón es que solo se crearón las vistas para dispositivos con pantallas de 4 o 5 pulgadas.

CONCLUSIONES Y PERSPECTIVAS DEL PROYECTO

La finalidad de este proyecto fue generar una aplicación para dispositivo móvil, con sistema operativo Android, para la práctica de conceptos de POO. Para poderlo lograr se tuvo que aprender a programar aplicaciones Android.

La aplicación funciona correctamente, es estable y cumple sus cometidos. Espero que se empiece a utilizar como herramienta de apoyo y aprendizaje en la UEA de POO.

A la aplicación aún le queda mucho por añadir y mejorar, como puede ser añadir nuevos modos de juego, nuevos tipos de preguntas, o que el usuario pueda añadir preguntas al juego. Se puede mejorar la forma de implementar la BD y crear diferentes pantallas para que pueda verse bien en diferentes dispositivos con distintos tamaños de pantalla. También esta aplicación se puede reutilizar el código y modificar para que se use en otras UEA's y no solo en la de POO.

Al terminar el proyecto, se logró generar una herramienta de aprendizaje para la UEZ de POO.

REFERENCIAS

[1] Sánchez García, J., Urías Ruiz, M. and Gutiérrez Herrera, B. (2015). ANÁLISIS DE LOS PROBLEMAS DE APRENDIZAJE DE LA PROGRAMACIÓN ORIENTADA A OBJETOS. [ebook] pp.289-304. Available at: http://www.redalyc.org/pdf/461/46142596021.pdf

[2] M. Tapia Téllez, "Aplicación para el apoyo a la enseñanza de la UEA Métodos Numéricos", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010.

[3] J.C. Castillo García, "Aplicación Android para la práctica de verbos compuestos del idioma inglés", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[4] M. Cavazzani, Preguntados. Etermax, 2013.

[5] "JASensei", Play.google.com,2017 https://play.google.com/store/apps/details?id=com.japanactivator.android.jasensei&hl=es_419.

[6] C. Studio, "Conoce Android Studio | Android Studio", Developer.android.com, 2017. https://developer.android.com/studio/intro/index.html?hl=es-419.

[7] A. Torres Moro, "EDAPIX: Una aplicación gráfica para asistir al proceso de enseñanza - aprendizaje de las Estructuras de Datos", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2009.

[8] D. Faro García, "Desarrollo de una aplicación Android de tipo quiz", Trabajo Fin de Grado, Departamento de Ingeniería Informática, Universidad Autónoma Madrid, España, 2014.

[9] Babymetal, "Give Me Chocolate 8 Bits", https://www.youtube.com/watch?v=6nQBsCxt2ag

[10] Asian Kung-Fu Generation, "Haruka Kanata 8 Bits" https://www.youtube.com/watch?v=7Bptv3w_hrc

[11] Kanako Ito, "Seisuu 3 No Nijou 8 Bits", https://www.youtube.com/watch?v=qI9X5ze99d8

[12] Itowokoashi, "Kanadeai 8 Bits ",

[13] Jun Maeda, "To The Same Heights 8 Bits", https://www.youtube.com/watch?v=i9R9gfvSxPE

[14] Chata, "Dango Daikazoku 8 Bits ", https://www.youtube.com/watch?v=dzIcELddHC4

[15] Hari, "Gwiyomi Song 8 Bits", https://www.youtube.com/watch?v=ZznwuZ1xn7A

[16] Stereo Dive, "Daisy 8 Bits ", https://www.youtube.com/watch?v=zdugZMP2nQU

[17] J. Aparicio, “Tecnología Móvil Como Herramienta De Apoyo En La Educación Media”, Titulo de Investigación, Universidad Tecnologica De El Salvador, El Salvador, 2012.

[18] L. Aguirre Chacon, “Diseño De Una Aplicación Móvil Para La Consulta Académica De La Fiis-Utp”, Tesis, Universidad Tecnologica De Perú, Perú, 2013.

[19] R. García Padilla, “Aplicación Android para Supermercados”, Tesis de Titulación, Facultad de Informática de Barcelona, España, 2011.

[20] I. Lidó Monzón, “Aplicación Android De Movilidad De Invidentes”, Tesis De Titulación, Especialidad Telemática, España, 2011.

ANEXO

Este anexo contiene algunos fragmentos de código de los métodos usados en las clases.

MENUPRINCIPAL

```
public class MenuPrincipal extends AppCompatActivity implements View.OnClickListener{
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_menu_principal);
    btnAprendizaje = (Button)findViewById(R.id.btnAprendizaje);
    btnJuego = (Button)findViewById(R.id.btnJuego);
    btnBuscar = (Button)findViewById(R.id.btnBuscar);

    btnAprendizaje.setOnClickListener(this);
    btnJuego.setOnClickListener(this);
    btnBuscar.setOnClickListener(this);
  }
```

```
  public void onClick(View v) {
    switch(v.getId()){
      case R.id.btnAprendizaje:
        Intent intent = new Intent(MenuPrincipal.this,ModuloAprendizaje.class);
        startActivity(intent);
        break;
      case R.id.btnJuego:
       Intent intent2 = new Intent(MenuPrincipal.this,ModuloPregunta.class);
        startActivity(intent2);
        break;
      case R.id.btnBuscar:
        Intent intent3 = new Intent(MenuPrincipal.this,ModuloBuscar.class);
        startActivity(intent3);
        break;
```

MODULOAPRENDIZAJE

```
public class ModuloAprendizaje extends AppCompatActivity implements View.OnClickListener
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_modulo_aprendizaje);
    btnPOO = (ImageButton) findViewById(R.id.btnPOO);
    btnPol = (ImageButton) findViewById(R.id.btnPoli);
    btnHere = (ImageButton) findViewById(R.id.btnHere);
    btnClase = (ImageButton) findViewById(R.id.btnClase);
    btnObj = (ImageButton) findViewById(R.id.btnObjeto);

    btnPOO.setOnClickListener(this);
    btnPol.setOnClickListener(this);
```

```
    btnHere.setOnClickListener(this);
    btnRCompo.setOnClickListener(this);
```

```
  public void onClick(View v) {
    switch (v.getId()){
 /**/       case R.id.btnPOO:
           if(i == 1 ){
            if(j == 1){
              j = 2;
              irLeerArchivo(i);
            }else{
              j=1;
              Toast toast1 = Toast.makeText(getApplicationContext(), "Programación Orientada a Objetos",
Toast.LENGTH_SHORT);
              toast1.show();
            }
           }else{
            i = 1;
            j = 1;
            Toast toast1 = Toast.makeText(getApplicationContext(), "Programación Orientada a Objetos",
Toast.LENGTH_SHORT);
            toast1.show();
          }
          break;
 /**/       case R.id.btnPoli:
           if(i == 2 ){
            if(j == 1){
              j = 2;
              irLeerArchivo(i);
            }else{
              j = 1;
              Toast toast2 = Toast.makeText(getApplicationContext(), "Polimorfismo", Toast.LENGTH_SHORT);
              toast2.show();
            }
           }else{
            i = 2;
            j = 1;
            Toast toast2 = Toast.makeText(getApplicationContext(), "Polimorfismo", Toast.LENGTH_SHORT);
            toast2.show();
           }
          break;
```

```
public void irLeerArchivo(int i){
    String doc= ""+i;
    Intent intent= new Intent(ModuloAprendizaje.this,LeerArchivo.class);
    intent.putExtra("doc", doc);
    startActivity(intent);   }
```

LEERARCHIVO

```
public class LeerArchivo extends AppCompatActivity
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_leer_archivo);

    mostrar =(TextView)findViewById(R.id.txtMostrar);

    Bundle bun = this.getIntent().getExtras();
    String s = bun.getString("doc").toString();
    int i = Integer.parseInt(s);
    InputStream fraw;
    switch (i){
      case 1:
        fraw = getResources().openRawResource(R.raw.poo);
        leerArchivo(fraw);
        break;
      case 2:
        fraw = getResources().openRawResource(R.raw.polimorfismo);
        leerArchivo(fraw);
        break;
      case 3:
        fraw = getResources().openRawResource(R.raw.herencia);
        leerArchivo(fraw);
        break;
}
```

```
  public void leerArchivo(InputStream fraw){
    try {
      BufferedReader brin = new BufferedReader(new InputStreamReader(fraw));
      String texto;
      StringBuilder tex = new StringBuilder();
      while ((texto = brin.readLine()) != null) {
        tex.append(texto);
        tex.append("\n");
      }  brin.close();
      fraw.close();
      mostrar.setText(tex.toString());
    }catch (Exception e){}
```

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
      finish();
      // Si el listener devuelve true, significa que el evento esta procesado, y nadie debe hacer nada mas
      return true;
    }
    //para las demas cosas, se reenvia el evento al listener habitual
    return super.onKeyDown(keyCode, event);
  }
```

MODULOPREGUNTA

```
public class ModuloPregunta extends AppCompatActivity implements View.OnClickListener
```

```
    protected void onCreate(Bundle savedInstanceState) {
        UsuariosSQLiteHelper usdbh =
                new UsuariosSQLiteHelper(this, "DBPreguntas", null, 1);
        db = usdbh.getWritableDatabase();
        ContentValues nuevoRegistro = new ContentValues();
        EPregunta pregunta = new EPregunta(1,"¿Cual es la acción que realiza un objeto?", "Método", "Atributo",
"Constructor", "Observer", "Strategy", "Polimorfimo", "Herencia",5);
        db.insert("Preguntas", null, generarValor(pregunta));
        pregunta = new EPregunta(2, "Su principal función es inicializar las variables del objeto:", "Método Constructor",
"Método Atributo", "Método toString", "Método set", "Método get", "Método System", "Método New",6);
        db.insert("Preguntas", null, generarValor(pregunta));
        pregunta = new EPregunta(3, "¿Cúal es la entidad que se caracteriza por sus atributos?", "Objeto",
"Polimorfismo", "Herencia", "Modularidad", "Variables", "Clase", "Interface",5);
        db.insert("Preguntas", null, generarValor(pregunta));
        pregunta = new EPregunta(4, "Son algunos ejemplos de objetos:", "Reloj, Pez, Lápiz", "Lápiz, Alto, Rojo",
"Amarillo, Pez, Tabla", "Alumno, Edad, Matricula", "Persona, Alumno, Género", "Llanta, Motor, Volante",
"Computadora, Teclado, Gris",5);
        db.insert("Preguntas", null, generarValor(pregunta));
        pregunta = new EPregunta(5, "Consiste en implementar multiples formas en un mismo método:", "Polimorfismo",
"Modularidad", "Objeto", "Clase", "POO", "Atributos", "Método",2);
        db.insert("Preguntas", null, generarValor(pregunta));
        pregunta = new EPregunta(6, "¿Cuales son las características  que diferencian a un objeto de otros?", "Atributos",
"Herencias", "Polimorfismos", "Variables", "Modularidades", "Interfaces", "Extends",5);
        db.insert("Preguntas", null, generarValor(pregunta));

    }
```

```
    public ContentValues generarValor(EPregunta pregunta) {
        String id = "" + pregunta.getId();
        String preg = pregunta.getPregunta();
        String rCor = pregunta.getrC();
        String r1 = pregunta.getR1();
        String r2 = pregunta.getR2();
        String r3 = pregunta.getR3();
        String r4 = pregunta.getR4();
        String r5 = pregunta.getR5();
        String r6 = pregunta.getR6();
        String tema = ""+pregunta.getTema();
        ContentValues nuevoRegistro = new ContentValues();
        nuevoRegistro.put("idPregunta", id);
        nuevoRegistro.put("pregunta", preg);
        nuevoRegistro.put("rCor", rCor);
        nuevoRegistro.put("res1", r1);
        nuevoRegistro.put("res2", r2);
        nuevoRegistro.put("res3", r3);
        nuevoRegistro.put("res4", r4);
        nuevoRegistro.put("res5", r5);
```

```
    nuevoRegistro.put("res6", r6);
    nuevoRegistro.put("tema", tema);
    return nuevoRegistro;
```

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.btnComenzar:
            String num = numPreguntas.getText().toString();
            int num2;
            if (num.isEmpty()) {
                Toast t = Toast.makeText(getApplicationContext(), "Ingrese Número de Preguntas", Toast.LENGTH_SHORT);
                t.show();
                break;
            } else {
                num2 = Integer.valueOf(num);
            }
            if (num2 <= 0 || num2 > 50) {
                Toast t = Toast.makeText(getApplicationContext(), "Número Invalido " + num2, Toast.LENGTH_SHORT);
                t.show();
            } else {
                Intent intent = new Intent(ModuloPregunta.this, JuegoPregunta.class);
                mp.stop();
                intent.putExtra("numPreguntas", num);
                intent.putExtra("sonido", String.valueOf(i));
                intent.putExtra("detalles", String.valueOf(j));
                intent.putExtra("dificultad", String.valueOf(k));
                startActivity(intent);
                ModuloPregunta.this.finish();
            }
            break;
        case R.id.btnMusica:
            if (i == 0) {
                btnSonido.setBackgroundResource(R.drawable.si);
                reproduceMusica();
                i = 1;
            } else {
                btnSonido.setBackgroundResource(R.drawable.no);
                mp.stop();
                i = 0;
            }
            break;
        case R.id.btnDetalles:
            if (j == 0) {
                btnDetalle.setBackgroundResource(R.drawable.si);
                j = 1;
            } else {
                btnDetalle.setBackgroundResource(R.drawable.no);
                j = 0;
            }
            break;
        case R.id.btnDificultad:
            if (k == 0) {
```

```
            k = 1;
            btnDificultad.setText("FACIL");

        } else if (k == 1) {
            btnDificultad.setText("NORMAL");
            k++;
        } else {
            btnDificultad.setText("DIFICIL");
            k = 0;
        }
        break;      }
```

```
    public void reproduceMusica() {
        Random rnd = new Random();
        int cancion = (int) (rnd.nextDouble() * 4);
        switch (cancion) {
            case 0:
                mp = MediaPlayer.create(this, R.raw.op1);
                mp.start();
                break;
            case 1:
                mp = MediaPlayer.create(this, R.raw.op2);
                mp.start();
                break;
}
```

JUEGOPREGUNTA

```
public class JuegoPregunta extends AppCompatActivity implements View.OnClickListener
```

```
protected void onCreate(Bundle savedInstanceState) {
    if (segundos == 1) { //Si el usuario eligio modo fácil, el tiempo no se mostrará
        txtTiempo.setVisibility(View.INVISIBLE);
        segundos = 100;
    } else {
        txtTiempo.setVisibility(View.VISIBLE);//Si el usuario eligio modo normal o dificil, el tiempo se mostrará
    }
    tiempo = new CountDownTimer(16000 + segundos * 5500, 1000) {
        public void onTick(long millisUntilFinished) {
            txtTiempo.setText("" + millisUntilFinished / 1000);
            if (txtTiempo.getText().equals("10")) {
                txtTiempo.setTextColor(Color.RED);
            }
            if (txtTiempo.getText().equals("1")) {
                Toast t = Toast.makeText(getApplicationContext(), "La Respuesta Correcta es :" + correcta,
Toast.LENGTH_LONG);
                t.show();
            }
        }
        public void onFinish() {
            txtTiempo.setText("0");
            nuevaPregunta();
```

```
        }
    }.start();//El tiempo que se mostrará depende si se eligio modo normal o dificil
  }
```

```
  public void pregunta(String[] args, int arreglo[], int arregloPre[]) {
    // String[] campos = new String[] {"idPregunta", "pregunta","rCor","res1","r2"};
    args[0] = String.valueOf(arregloPre[numPre]);//Es la id de la pregunta que se va a buscar
    Cursor c = db.rawQuery("SELECT idPregunta,pregunta,rCor,res1,res2,res3,res4,res5,res6, tema FROM Preguntas
Where idPregunta= ?", args);
    // Cursor c = db.query("Preguntas",campos,null,null,null,null,null);
    EPregunta p = new EPregunta();
    if (c.moveToFirst()) {
      //Recorremos el cursor hasta que no haya más registros
      do {
        p.setId(c.getInt(0));
        p.setPregunta(c.getString(1));
        p.setrC(c.getString(2));
        p.setR1(c.getString(arreglo[0]));
        p.setR2(c.getString(arreglo[1]));
        p.setR3(c.getString(arreglo[2]));
        p.setR4(c.getString(arreglo[3]));
        tema= c.getString(9);
      } while (c.moveToNext());
      correcta = String.valueOf(p.getrC().toString());
      txtPregunta.setText(p.getPregunta().toString());
      btnR1.setText(p.getR1().toString());
      btnR2.setText(p.getR2().toString());
      btnR3.setText(p.getR3().toString());
      btnR4.setText(p.getR4().toString());
    }
    mandarPreguntas = mandarPreguntas + (numPre + 1) + ") " + p.getPregunta() + "\n  *Respuesta: " + p.getrC() +
"\n\n";   }
```

```
  public int[] listaPreguntasAleatorio(int arreglo[], int cantidad) {
    int i = 0, rango = 10;
    arreglo[i] = (int) (Math.random() * (cantidad) + 1);
    for (i = 1; i < cantidad; i++) {
      arreglo[i] = (int) (Math.random() * (cantidad) + 1);
      for (int j = 0; j < i; j++) {
        if (arreglo[i] == arreglo[j]) {
          i--;
        }       }     }
    return arreglo;   }
```

```
public int respAle() {
    Random rnd = new Random();
    return (int) (rnd.nextDouble() * 4);
  }
```

```
  public int[] listaRespuestasAleatorio(int arreglo[], int cantidad) {
    int i = 0, rango = 10;
    arreglo[i] = (int) (Math.random() * (8 - 3 + 1) + 3);
```

```
    for (i = 1; i < cantidad; i++) {
      arreglo[i] = (int) (Math.random() * (8 - 3 + 1) + 3);
      for (int j = 0; j < i; j++) {
        if (arreglo[i] == arreglo[j]) {
          i--;
        }       }     }
    return arreglo;
```

```
public void resElegida(Button b, int i) {
    Toast t;
    btnR1.setEnabled(false);
    btnR2.setEnabled(false);
    btnR3.setEnabled(false);
    btnR4.setEnabled(false);
    btnSig.setVisibility(View.VISIBLE);
    MediaPlayer mp;
    if (resCorrecta == i) {
      if (sonido.equals("1")) {
        mp = MediaPlayer.create(this, R.raw.correcto);
        mp.start();       }
      b.setBackgroundResource(R.drawable.botoncorrecto);
      t = Toast.makeText(getApplicationContext(), "Respuesta Correcta", Toast.LENGTH_SHORT);
      t.show();
      contaCorrecta++;
    } else {
      if (sonido.equals("1")) {
        mp = MediaPlayer.create(this, R.raw.mal);
        mp.start();
      }
      b.setBackgroundResource(R.drawable.botonincorrecto);
      t = Toast.makeText(getApplicationContext(), "Incorrecto \nLa Respuesta Correcta es:\n" + correcta,
Toast.LENGTH_LONG);
      t.show();
    }

    if(detalles.equals("1")){
      btnDetalle.setVisibility(View.VISIBLE);
    }
  }
```

```
public void nuevaPregunta() {
    btnR1.setEnabled(true);
    btnR1.setBackgroundResource(R.drawable.botonnormal);
    btnR2.setEnabled(true);
    btnR2.setBackgroundResource(R.drawable.botonnormal);
    btnR3.setEnabled(true);
    btnR3.setBackgroundResource(R.drawable.botonnormal);
    btnR4.setEnabled(true);
    btnR4.setBackgroundResource(R.drawable.botonnormal);
    if (numPre < numPreguntas - 1) {
      numPre++;
      listaRespuestasAleatorio(arregloRes, 4);
```

```
        resCorrecta = respAle();
        arregloRes[resCorrecta] = 2;
        String[] args = new String[1];
        pregunta(args, arregloRes, arregloPreguntas);
        txtTiempo.setTextColor(Color.BLUE);
        tiempo.start();
        btnSig.setVisibility(View.INVISIBLE);
        btnDetalle.setVisibility(View.INVISIBLE);
      } else {
        Intent intent = new Intent(JuegoPregunta.this, PantallaPromedio.class);
        float promedio1 = (10 * contaCorrecta) / numPreguntas;
        String promedio = String.valueOf(promedio1);
        intent.putExtra("promedio", String.valueOf(promedio));
        intent.putExtra("listaPreguntas", mandarPreguntas);
        intent.putExtra("sonido", sonido);
        startActivity(intent);
        finish();       }
```

```
public void irLeerArchivo(String i){
    String doc= ""+i;
    Intent intent= new Intent(JuegoPregunta.this,LeerArchivo.class);
    intent.putExtra("doc", doc);
    startActivity(intent);   }
```

PANTALLAPROMEDIO

```
public class PantallaPromedio extends AppCompatActivity
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_pantalla_promedio);
    Bundle bun = this.getIntent().getExtras();
    String promedio = bun.getString("promedio");
    Bundle bun1 = this.getIntent().getExtras();
    String listaPreguntas = bun1.getString("listaPreguntas");
    Bundle bunSonido = this.getIntent().getExtras();
    sonido = bunSonido.getString("sonido");
    if (sonido.equals("1")) {
      reproduceMusica();       }
    txtPromedio.setText(" "+promedio);
    txtListaPreguntas.setText(""+listaPreguntas);
    float cal = Float.parseFloat(promedio);
    obtenerCalificacion(cal);   }
```

```
public void obtenerCalificacion(float cal){
    if(cal < 6.0){
      txtCalificacion.setText("Sacaste una espantosa NA.");       }
    if( cal>= 6.0 && cal < 7.5){
      txtCalificacion.setText("Sacaste una S. Repasa más.");       }
    if(cal>=7.5 && cal < 8.5){
      txtCalificacion.setText("Sacaste una B. Bien.");
```

```
    }if(cal >= 8.5){
      txtCalificacion.setText("Sacaste una MB. Excelente.");      }
```

```
  public void reproduceMusica(){
    Random rnd = new Random();
    int cancion = (int)(rnd.nextDouble() * 4);
    switch(cancion){
      case 0:
        mp = MediaPlayer.create(this, R.raw.en1);
        mp.start();
        break;
      case 1:
        mp = MediaPlayer.create(this, R.raw.en2);
        mp.start();
        break;
```

MODULOBUSCAR

```
public class ModuloBuscar extends AppCompatActivity
```

```
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_modulo_buscar);
    String[] temas = new String[]{"PROGRAMACION ORIENTADA A OBJETOS","VENTAJAS Y DESVENTAJAS DE POO",
        "CLASE","ATRIBUTOS DE LA CLASE","MÉTODOS DE LA CLASE",
        "OBJETOS","PARTES FUNDAMENTALES DEL OBJETO", "CONSTRUCTOR",  "HERENCIA","EXTENDS",
        "ENCAPSULAMIENTO","MODOS DE ACCESO (PUBLIC, PRIVATE,      PROTECTED)",
"ABSTRACCION","INTERFACES","IMPLEMENTS","POLIMORFISMO", "MODULARIDAD",  "DIAGRAMA DE CLASES"};
    lista = (ListView)findViewById(R.id.idListView);
    arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_expandable_list_item_1,temas);
    if(lista != null){
      lista.setAdapter(arrayAdapter);
      lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
                                int position, long id) {
          // temaSelec =  (TextView) view.findViewById(R.id.t);
          temaBuscado =  lista.getItemAtPosition(position).toString();
          String temaSelec =  buscarArchivo(temaBuscado);
          irLeerArchivo(temaSelec);
        }       });
```

```
  public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu,menu);
    SearchManager searchManager = (SearchManager) getSystemService(Context.SEARCH_SERVICE);
    final SearchView serchView = (SearchView) menu.findItem(R.id.search).getActionView();
    serchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentName()));
    serchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
      @Override
      public boolean onQueryTextSubmit(String query) {
```

```
            Toast t = Toast.makeText(ModuloBuscar.this,"Resultados con la palabra "+query,Toast.LENGTH_SHORT);
            t.show();
            arrayAdapter.getFilter().filter(query);
            return false;          }
    public boolean onQueryTextChange(String newText) {
            Toast.makeText(ModuloBuscar.this,"-"+newText,Toast.LENGTH_LONG);
            arrayAdapter.getFilter().filter(newText);
            return false;
        }       });
      return super.onCreateOptionsMenu(menu);    }
```

```
    public  String buscarArchivo(String tema){
      String id="";
      if(tema.equals("PROGRAMACION ORIENTADA A OBJETOS") || tema.equals("VENTAJAS Y DESVENTAJAS DE
POO")){
          id="1";       }
      if(tema.equals("CLASE") || tema.equals("ATRIBUTOS DE LA CLASE") || tema.equals("MÉTODOS DE LA CLASE")){
          id="4";       }
      if(tema.equals("OBJETOS") || tema.equals("PARTES FUNDAMENTALES DEL OBJETO")){
          id="5";       }
       return id;    }
```

```
    public  String buscarArchivo(String tema){
      String id="";
      if(tema.equals("PROGRAMACION ORIENTADA A OBJETOS") || tema.equals("VENTAJAS Y DESVENTAJAS DE
POO")){
          id="1";       }
      if(tema.equals("CLASE") || tema.equals("ATRIBUTOS DE LA CLASE") || tema.equals("MÉTODOS DE LA CLASE")){
          id="4";       }

      return id;    }
```

USUARIOSSQLITEHELPER

```
public class UsuariosSQLiteHelper extends SQLiteOpenHelper
```

```
    String sqlCreate = "CREATE TABLE Preguntas (idPregunta INTEGER, pregunta TEXT, rCor TEXT, res1 TEXT, res2 TEXT,
res3 TEXT, res4 TEXT, res5 TEXT, res6 TEXT, tema Text)";
    private SQLiteDatabase bd;
    public UsuariosSQLiteHelper(Context contexto, String nombre,
                        CursorFactory factory, int version) {
      super(contexto, nombre, factory, version);
```

```
    public void onCreate(SQLiteDatabase db) {
      db.execSQL(sqlCreate);
    }
```

```
    public void onUpgrade(SQLiteDatabase db, int versionAnterior,
                    int versionNueva) {
      db.execSQL("DROP TABLE IF EXISTS Pregutas");
```

EPREGUNTA

```
public class EPregunta
```

```
    public EPregunta(int id,String pregunta, String rC, String r1, String r2, String r3, String r4, String r5, String r6, int tema) {
        this.id = id;
        this.pregunta = pregunta;
        this.r1 = r1;
        this.r2 = r2;
        this.r3 = r3;
        this.r4 = r4;
        this.r5 = r5;
        this.r6 = r6;
        this.rC = rC;
        this.tema = tema;
    }
```