UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

XOOK: APLICACIÓN PARA GEOMETRÍA ANALÍTICA EN ANDROID

TESIS

Que para obtener el título de

INGENIERO EN COMPUTACIÓN

P R E S E N T A

BONILLA PÉREZ JULIO ALFREDO

DIRECTORA DE TESIS

MTRA. IRENE PATRICIA VALDEZ Y ALFARO

Ciudad Universitaria, Cd. Mx., 2016

AGRADECIMIENTOS

A mi familia por todo el apoyo brindado a lo largo de los años, por apoyarme en las buenas y en las malas, que siempre han estado ahí cuando más los he necesitado.

A mis amigos quienes han hecho más ameno el camino y me han apoyado. Particularmente Cesar Castro Cerros por su apoyo en la realización de este trabajo.

A los participantes en el desarrollo de los ejercicios del Laboratorio Virtual de Matemáticas, de donde se adaptaron los ejercicios para el desarrollo de este trabajo: Casiano Aguilar Morales, Hortencia Caballero López, Juan Velázquez Torres, Luis Humberto Soriano Sánchez, María del Rocío Ávila Núñez, María Sara Valentina Sánchez Salinas, Mayverena Jurado Pineda, Ricardo Martínez Gómez, Rosalba Rodríguez Chávez, Sergio Arzamendi Pérez, Sergio Carlos Crail Corzas.

A Érick Castañeda de Isla Puga por el libro Geometría analítica en el espacio, que sirvió para el desarrollo de contenidos de la aplicación desarrollada.

RESUMEN

El presente trabajo abordará el desarrollo de una aplicación para dispositivos móviles que utilicen el sistema operativo Android, con la finalidad de apoyar el aprendizaje de los estudiantes de nivel universitario en la materia de Geometría Analítica.

Este trabajo se compone de una revisión del desarrollo de los dispositivos móviles, los tipos de programación para el sistema operativo Android, y los diferentes programas para desarrollar aplicaciones híbridas, entre los que eligió Corona SDK. Para el contenido, se realiza una selección de temas de Geometría Analítica que el alumno aprende en los primeros semestres de estudio en las diversas carreras de la Facultad de Ingeniería de la UNAM. Se considera la importancia del apoyo a los estudiantes en el repaso de los temas de Vectores y Superficies cuádricas como eje del objetivo principal.

Por último, se hace una evaluación sobre los contenidos, funcionalidad de la app mediante la manipulación de ésta por parte de los usuarios.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el presente trabajo se desarrolla una propuesta de aplicación para los dispositivos móviles que cuenten con el sistema operativo Android. Ésta es una aplicación enfocada a la asignatura de Geometría Analítica, la cual se imparte en los primeros semestres de cada una de las carreras con las que cuenta la Facultad de Ingeniería de la Universidad Nacional Autónoma de México. El objetivo que se persiguió es que dicha aplicación incorpore información relativa a los temas que se utilizan en los programas educativos a nivel universitario para que el alumno, apoyándose en ésta herramienta didáctica, pueda repasar sus conocimientos de una forma interactiva.

En este trabajo se muestran las diferentes formas de programación existentes en Android para los dispositivos móviles, tales como la programación nativa, el desarrollo con HTML5 (desarrollo web) y la programación híbrida. La programación utilizada está enfocada a que la aplicación pueda ejecutarse en gran variedad de dispositivos móviles con Android y sea accesible para la mayor parte de los estudiantes universitarios, asimismo se tomó en cuenta la posibilidad de que en un futuro la aplicación sea multiplataforma, con este propósito se utilizó programación híbrida.

El presente trabajo se encuentra formado por los siguientes capítulos:

- **Propuesta para apoyar el aprendizaje de la geometría analítica**. A lo largo del capítulo se establecerá por qué se propone el desarrollo de una aplicación, cuáles son los objetivos de este trabajo, sus alcances y limitaciones.
- **Marco teórico**. En este apartado se hablará de los antecedentes de los dispositivos móviles, las diferentes alternativas en tecnologías para programar en Android y el software que se eligió para realizar la aplicación (app).

- **Diseño y desarrollo de la aplicación**. En este capítulo se describe el proceso de desarrollo de la aplicación, desde la selección de los materiales que contendrá, su interfaz, la programación y las pruebas realizadas.

- **Conclusiones y recomendaciones**. Capítulo en el que se presentan las conclusiones obtenidas al finalizar el desarrollo de este trabajo y recomendaciones para trabajo futuro.

CAPÍTULO 1 PROPUESTA PARA APOYAR EL APRENDIZAJE DE LA MATERIA DE GEOMETRÍA ANALÍTICA

A lo largo del capítulo se establecerá el porqué de la aplicación. Se comentará sobre el alto índice de reprobados en la materia de Geometría Analítica, el cuál es el problema a abordar. Así mismo se destacará la importancia que hoy en día tienen los dispositivos móviles, justificando el por qué se ha de realizar un programa didáctico para hacer uso de estos. Se hablará del software para desarrollar aplicaciones móviles y se propondrá el desarrollo de una aplicación para así afrontar al problema.

1.1 Planteamiento del problema

En la Facultad de Ingeniería de la UNAM existe un alto índice de reprobación en la asignatura de Geometría Analítica, como se muestra en la Figura 1-1 y Figura 1-2. Materia que en el plan de estudios 2016 se integró a Cálculo y Geometría Analítica, por lo cual hay que sumar esfuerzos para combatir el rezago y la deserción; como parte de estos esfuerzos se busca que, a través de la creación de recursos digitales de aprendizaje, se pueda ayudar a los estudiantes a mejorar la comprensión con respecto a los temas de la materia.

Los alumnos en su quehacer cotidiano tienen una gran diversidad de actividades que los ocupan, lo que acarrea que les quede poco tiempo para el estudio de sus materias, entre el vaivén de las clases y alguna actividad deportiva; dando como resultado que el tiempo de traslado –prolongado en algunos casos- sea aprovechado para repasar los apuntes y revisar información sobre los deberes escolares. Por lo que se espera que los recursos digitales los puedan usar los alumnos de manera autónoma fuera del aula a través de un dispositivo móvil y quizá en esos momentos de transporte.

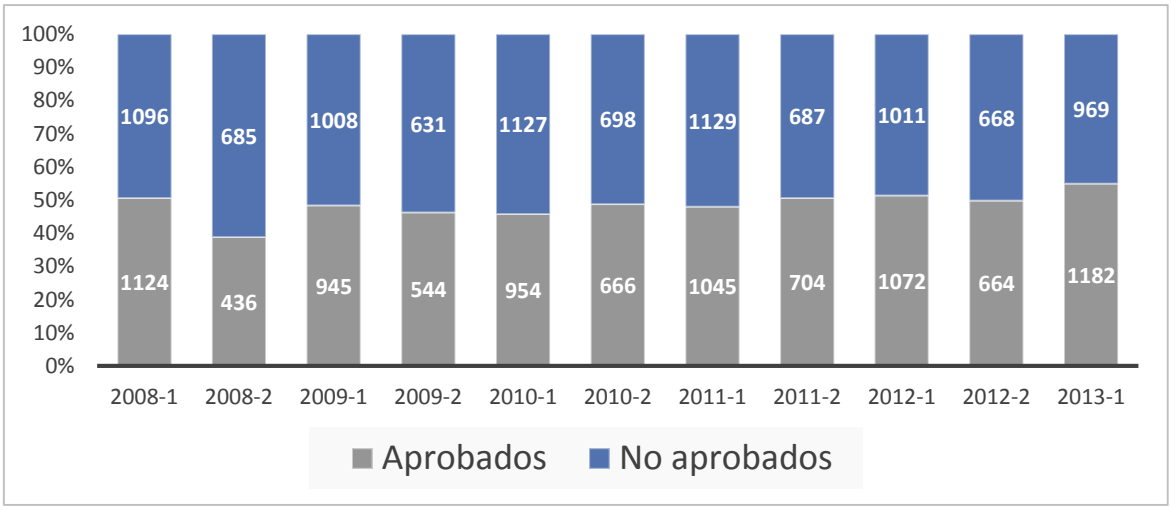


Figura 1-1 Alumnos Ordinarios de Geometría Analítica

Fuente: Elaboración propia con base en datos proporcionados por la División de Ciencias Básicas

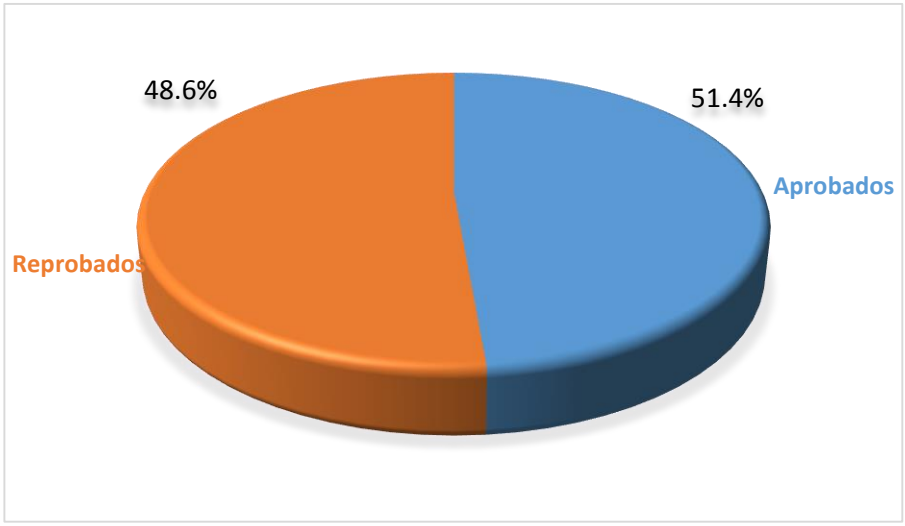


Figura 1-2 Porcentaje de Alumnos aprobados; Semestres 2008-1/2013-1

Fuente: Elaboración propia con base en datos proporcionados por la División de Ciencias Básicas

1.2 Objetivos

1.2.1 Objetivo general

Aportar un software para dispositivos móviles con el fin de apoyar la mejor compresión de algunos temas de Geometría Analítica por parte los estudiantes de la Facultad de Ingeniería de la UNAM.

1.2.2 Objetivos particulares

- Indagar cuáles son las herramientas actuales de diseño y programación, de carácter libre o gratuito para plataformas móviles, con el propósito de determinar y seleccionar las más apropiadas para el desarrollo de recursos digitales de aprendizaje de Geometría Analítica.
- Desarrollar recursos digitales de aprendizaje para Geometría Analítica destinados a dispositivos móviles.

1.3 Justificación

Hoy en día los dispositivos móviles son una herramienta indispensable para el quehacer cotidiano. La cantidad de suscriptores (no confundir con la cantidad de usuarios) de los dispositivos denominados “móviles” a nivel global era en el 2012 de 6,200 millones y ya para mayo del 2014 la cifra ascendía a alrededor de 7,000 millones, equivalente al 95.5% de la población mundial (mobiThinking, 2014).

Android es uno de los sistemas operativos con los que cuentan los teléfonos inteligentes, llegando a 1,000 millones de activaciones en el 2014 en dispositivos móviles según cifras oficiales de Google (Google, 2014). Tomando en cuenta este amplio mercado, es relativamente fácil que las aplicaciones lleguen a diferentes usuarios, entre ellos a la comunidad estudiantil universitaria.

Los estudiantes suelen auxiliarse en aplicaciones móviles, como por ejemplo para poner recordatorios del estudio de alguna tarea o examen próximo, consultar fórmulas matemáticas, etc.

Tomando en cuenta que los alumnos generalmente pasan un tiempo considerable en el transporte de su casa a su Facultad y viceversa se decidió elaborar una aplicación que permita hacer uso de estos tiempos para el repaso de algunos temas de Geometría Analítica, reforzando así los conocimientos vistos en clase.

1.4 Metodología

Para el desarrollo de la aplicación se buscarán las diferentes tecnologías para desarrollo de aplicaciones Android, una vez elegida la que se adapte más a las necesidades del presente trabajo se buscará y seleccionará un software que permita trabajar con dicha tecnología. Paso siguiente será la selección de temas que se presentarán en la aplicación.

Respecto a los temas para los que se desarrollarán los recursos digitales, se trabajará con un desarrollo previo realizado por profesores para la asignatura de Geometría Analítica, quienes de acuerdo a su experiencia docente determinaron qué conceptos de la materia se abordaron y especificaron los requerimientos didácticos de los recursos de aprendizaje.

Una vez seleccionados los temas, se diseñará la interfaz del programa, se establecerán los estándares a seguir y se procederá a desarrollar la aplicación realizando pruebas de software por cada segmento desarrollado.

1.5 Descripción del producto esperado

Una aplicación o app a desarrollarse para el sistema operativo Android, con la finalidad de ejecutarse en dispositivos móviles. Ha de ser un software que sea posible descargar vía web, tanto el archivo ejecutable (que por ser para Android será un archivo apk) como el código fuente del propio software, liberado mediante una licencia Copyleft (permitir la libre distribución de copias y versiones modificadas, exigiendo que los mismos derechos sean preservados).

1.6 Alcances y limitaciones

Al ser un programa desarrollado para Android cuenta con una gran gama de dispositivos en los que podrá usarse, aunque probablemente haya algún dispositivo en el que no se ejecute el programa correctamente. El desarrollo se enfocara en la versión 2.3 de Android (debido a la retrocompatibilidad, o compatibilidad regresiva, con la que cuenta este sistema operativo será posible ejecutar el software aun en las versiones más recientes).

Se considera cubrir diversos temas de Geometría Analítica, los cuales se detallan en la sección: 3.1.1 Selección y descripción de los contenidos de cada recurso. Aunque el programa será diseñado de tal manera que sea posible que quien esté interesado pueda agregar más temas o contenido en general, ampliando así lo comprendido en la app.

Las plataformas en las cuales se pondrá a disposición el programa ejecutable, para la instalación en dispositivos, son Google Play® (la tienda de aplicaciones de Google®), y en el sitio web CERAFIN (Centro de Recursos de Aprendizaje para las Ciencias Básicas) de la DCB (División de Ciencias Básicas de la Facultad de Ingeniería de la UNAM). El código fuente

podrá ser descargado desde Bitbucket® (plataforma de desarrollo colaborativo de software), para que la aplicación pueda ser revisada o modificada.

CAPÍTULO 2 MARCO TEÓRICO

En este apartado se hablará de los antecedentes de los dispositivos móviles, la masificación en su uso y el constante desarrollo de software para estos, lo que llevó a la creación de las tiendas de aplicaciones.

Se identificarán los elementos para la creación de un software con materiales didácticos apropiados, así como las diferentes alternativas en tecnologías para programar en Android y se describirá la tecnología con la que se realizará la app.

2.1 Antecedentes.

La llegada de los teléfonos celulares al mercado marcó un hito en la sociedad. Ya en su segunda generación (2G), además de la transmisión de voz, fue posible la transmisión de datos como los mensajes de texto corto (SMS), mensajes multimedia (MMS) y acceso al correo electrónico (email). Fue en esta generación que surgieron los llamados teléfonos inteligentes, aunque si bien este término es considerado mercadológico (Sutter, 2010) una de las características que destacó a estos teléfonos es el tener acceso a internet.

En la tercera generación de telefonía celular (3G) la compañía Apple® lanzó un servicio que modificaría el cómo se distribuyen los programas en los celulares. Se trata de la tienda de aplicaciones de Apple®, *iPhone's App Store*®, que fue lanzada junto con el iPhone 3G® (Apple, 2008). Tienda que contaba en su lanzamiento con más de 500 aplicaciones (apps), teniendo en principio dentro de sus cien aplicaciones más vendidas a dos apps educativas: Netters® (Atlas de anatomía humana) y Starmap® (aplicación de astronomía) (Pinch Media, 2008).

A la tienda de Apple® se le sumaron eventualmente las de Google®, BlackBerry®, Nokia®, Windows® y Firefox®; aumentando con ello la cantidad de aplicaciones educativas, existiendo ya en el 2014 más de 80,000 apps de este tipo (eduapps, 2014). A pesar de la gran cantidad de programas educativos disponibles, a la fecha de elaboración de este documento únicamente existían dos aplicaciones en Google Play® que hacían referencia a Geometría Analítica en español, y ambas consistían en formularios.

2.2 Elementos a considerar para la creación de materiales didácticos digitales.

El presente trabajo trata de considerar diferentes elementos para la creación de un software con materiales didácticos apropiados. Para realizar lo anteriormente mencionado se utilizarán criterios basados en la evaluación de Software Educativo propuesta por el Mtro. Miguel Ángel González Castañón.

La evaluación del software educativo se debe de orientar a ayudar al usuario; haciendo énfasis en aspectos pedagógicos, metodológicos, ideológicos y culturales (González Castañón, 1998). La evaluación propuesta no considera un método rígido sino más bien abierto a la capacidad y posibilidad de la "construcción de conceptos" como criterio pedagógico deseable.

Las partes necesarias para la evaluación de un Software Educativo son considerar los aspectos como: Objeto Material y Objeto Pedagógico.

Objeto Material

En la evaluación del Software Educativo como *Objeto Material* se consideran el Equipo Requerido y la Usabilidad del programa, utilizando estos elementos para la creación del material didáctico digital.

Respecto al Equipo Requerido, González Castañón (1998) explica que, para la evaluación, se comprendan los requerimientos del equipo mínimos para el funcionamiento del Software.

Usabilidad, será la facilidad con la que el usuario pueda aprender a utilizar el software educativo, en general trata de la amigabilidad del software: interfaz intuitiva, de fácil aprendizaje.

Objeto Pedagógico

En lo referente a la evaluación del software como *Objeto Pedagógico*, se toman en cuenta aspectos como el Contenido, Comunicación y Método.

Contenido.- Puede ser científico, pedagógico o socio-cultural e ideológico. Cantidad y calidad adecuada de información para los usuarios a los que está orientado.

Comunicación.- Es evaluar los recursos que permiten transmitir un mensaje; la dirección y control de la interacción entre el programa y el usuario (el sentido), y la forma del mensaje: los textos, animaciones, audio e imágenes.

Método.- Organización; secuencias de los ejercicios para el desarrollo de habilidades, y adaptabilidad: dinámicas impuestas en función del nivel del usuario.

En el Software a desarrollarse se considerarán los elementos que Miguel Ángel González Castañón sugiere para la evaluación de Software Educativo, considerando los aspectos materiales y pedagógicos, con la finalidad de obtener un resultado que ofrezca un contenido y usabilidad adecuados para los usuarios finales; que en el caso que nos concierne, son los alumnos de nivel superior que estudian alguna carrera de ingeniería.

2.3 Diferentes tipos de tecnología para crear una app en Android.

Existen diferentes tipos de aplicaciones Android. A continuación se brinda una descripción de los tipos de desarrollos para aplicación que existen y cuál se seleccionó.

Aplicaciones nativas.

Son aquellos programas desarrollados específicamente para una plataforma, implementados en el lenguaje nativo de dicha plataforma. En el caso de Android la combinación de lenguaje y herramienta de desarrollo recomendada por Google® es Java® y Eclipse®. Las aplicaciones nativas se ejecutan directamente en el dispositivo. Algunas ventajas de este tipo de aplicaciones es que son las que tienen un mejor desempeño, cuentan con una gran documentación y se tiene acceso a prácticamente todos los componentes que traen instalados de fábrica los dispositivos Android. Como desventajas se tiene que el tiempo de desarrollo suele ser superior comparado con el de las aplicaciones web o las hibridas, además de que en caso de que se quisiera que la app trabajara en otra plataforma se tendría que programar desde cero dada la incompatibilidad del código. Ejemplos de este tipo de aplicaciones son: Swype® e Infinity Blade 3®.

Aplicaciones web o HTML5.

Son prácticamente una página web, o una serie de páginas web. Se ejecutan en el servidor. Como puntos a favor permiten una gran modificación de la interfaz, solo requieren programarse una vez y funcionan en cualquier plataforma (idealmente). En cuanto a puntos en contra se tiene que la seguridad en este tipo de aplicaciones suele ser muy deficiente, la interfaz puede variar de sistema a sistema, otro factor es que aún no se puede hacer uso de algunas funciones nativas como es el acceso a la cámara fotográfica o la capacidad *multitáctil.* Dropbox® es un buen ejemplo de esté tipos de apps (también disponible como aplicación nativa).

Aplicaciones hibridas.

En su mayoría son programadas como aplicaciones web con la diferencia que estas se ejecutan en el dispositivo. Este acercamiento generalmente nativo-web permite el uso de funciones nativas del dispositivo, además que no requiere una conexión a internet para ejecutarse, para usarse en otra plataforma se requieren cambios mínimos. Las aplicaciones que son multiplataforma y no son web suelen ser ubicadas en esta categoría. Su principal desventaja es que suelen tener un tiempo de ejecución más alto que el de las aplicaciones nativas. Evernote®, Amazon® y LinkedIn® son ejemplos de éstas.

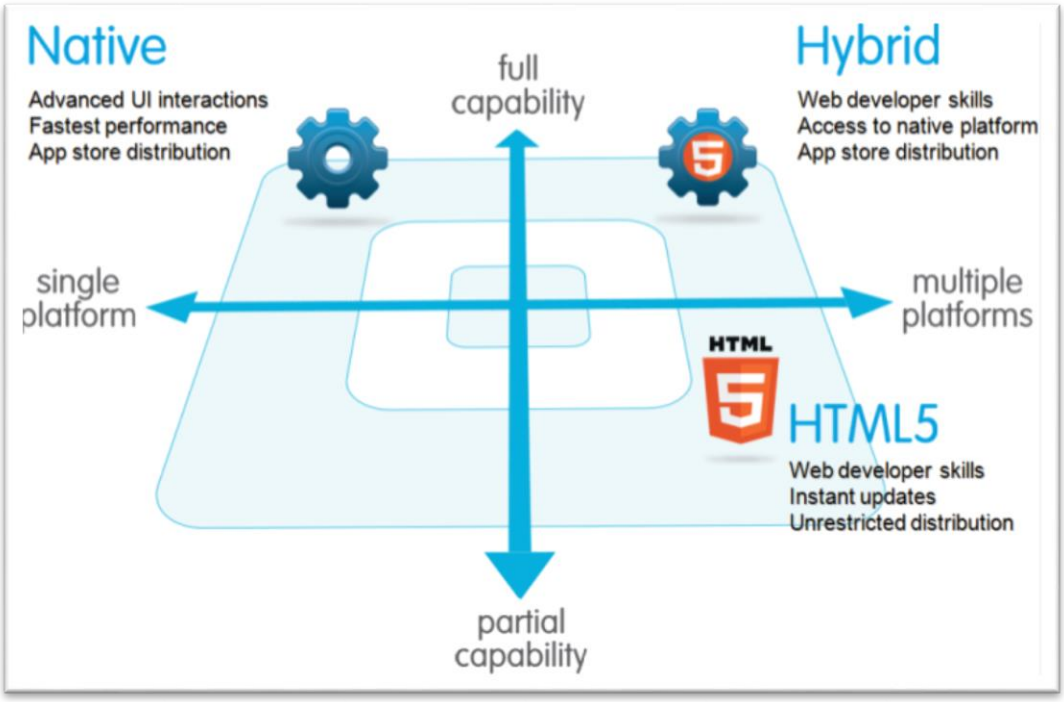


Figura 2-1 [Muestra comparativa de los distintos tipos de aplicaciones para dispositivos móviles]
Fuente: (Korf & Oksman, s.f.)

La Figura 2-1 muestra de manera resumida lo anteriormente expuesto. Con base en las características de los diferentes tipos de desarrollos para las aplicaciones en Android, y considerando que en un futuro la aplicación pueda ser multiplataforma, la programación hibrida puede ser una buena opción a tomar en cuenta para el desarrollo de la aplicación.

2.4 Programas para desarrollo de aplicaciones híbridas

Existiendo actualmente una gran variedad de programas que permiten el desarrollo de apps hibridas, el siguiente paso fue analizar y elegir el software con el que se realizaría la aplicación. Se buscó programas que no requirieran el pago de licencia, tuvieran amplia documentación y contaran con foros gratuitos para soporte. Aun acotando la búsqueda hay

un sinfín de programas que cumplen con estos parámetros, enseguida se muestran cuatro de los más populares.

Apache Flex

Figura 2-2 Logotipo de apache flex®

Fuente: (The Apache Software Foundation, 2014)

Apache Flex® es un Kit de Desarrollo de Software (SDK por sus siglas en ingles) que permite crear aplicaciones para móviles, navegadores o computadoras.

En un principio conocido como Adobe Flex, Apache Flex® fue donado por Adobe® a la Apache Software Foundation (ASF) (Jackson, 2011). La versión más reciente de Flex (4.12.1) fue liberada bajo la licencia apache versión 2, por lo que no solo es un SDK de código libre sino también permite que los programas sean desarrollados usando el Entorno de Desarrollo Integrado (IDE) que prefiera el usuario.

El lenguaje de programación que se ocupa para desarrollar en Flex es Action Script, el cual es un lenguaje de Programación Orientado a Objetos. Al compilar la aplicación se combina el MXML con el código Action Script para crear una aplicación Formato SWF . Ésta aplicación puede ser publicada para que se visualice en los navegadores o compilada con Adobe Air® para convertirla en una aplicación nativa de Windows®, Mac OSX®, Android®, iOS®, o BlackBerry®.

Apache Flex® proporciona acceso a todas las características nativas de los Sistemas Operativos por ejemplo: Interfaz de Usuario Grafica (GUI) nativa, acelerómetro, cámara, compas, etc.

La principal desventaja de este software es que es dependiente de Adobe Flash Player® o de Adobe Air®, es decir depende de un software de terceros, en este caso de Adobe®.

Apache Cordova

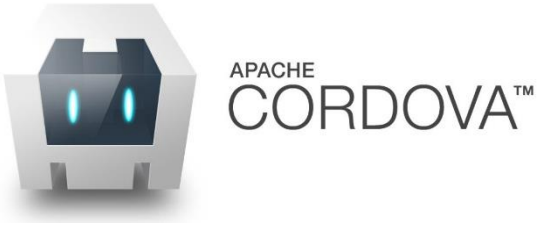


Figura 2-3 Logotipo de Apache Cordova®
Fuente: (The Apache Software Foundation, 2015)

PhoneGap fue donado a la ASF bajo el nombre de Apache Cordova® y se encuentra bajo la licencia Apache, versión 2.0. Apache Cordova® es un framework que permite el desarrollo de programas multiplataforma usando tecnologías web estándar como son HTML, JavaScript y CSS.

Las plataformas para las que se puede desarrollar son: iPhone®, Android®, Windows Phone®, BlackBerry 10®, Firefox OS®, Symbian®, webOS®, Bada, Ubuntu Touch y Tizen®. Las características nativas a las que se puede acceder en la mayoría de estos sistemas son:

Acelerómetro, cámara, compas, contactos, archivos, geolocalización, notificaciones, multimedia, almacenamiento y red.

La desventaja del uso de tecnologías basadas en web, como es el caso de los programas desarrollados en Apache Cordova®, tienden a ser más lentos que aquellos desarrollados en otras tecnologías similares.

ShiVa 3D

Figura 2-4 Logotipo de ShiVa®
Fuente: (ShiVa Technologies SAS , 2015)

Software propietario de la compañía Stonetrip®. Es un motor de videojuegos 3D que además cuenta con un editor WYSIWYG y un simulador para dispositivos.

El editor permite desarrollar para Windows®, Mac®, Linux®, Wii®, Xbox 360®, PS3®, PSP®, PSP Vita®, iPhone®, iPad®, Android®, Windows Phone®, BlackBerry QNX®, Marmalade, Bada, navegadores web, Adobe Flash®. Los lenguajes utilizados para el desarrollo con esta plataforma son Lua, C, C++ y Objective-C.

El motor de videojuegos, sirve para el manejo de: geometría de terreno, shader, luz y sombra, sonido y red. Así mismo se le pueden agregar complementos como PhysX engine y ARToolkit, entre otros.

El simulador puede ser instalado en iPhone®, iPod®, iPad®, Android® y Palm®. Su función es la de mostrarnos como se ejecuta nuestra aplicación y desplegar datos que se estén generando como pueden ser los del acelerómetro, GPS, compas y multitáctil.

ShiVa 3D® tiene una licencia web gratuita, pero para desarrollo en plataforma se requiere comprar una licencia y aunque ShiVa 3D® cuenta con licencia académica el precio de ésta es de $400 USD.

Corona SDK

Figura 2-5 Logotipo de Corona SDK®

Fuente: (Corona Labs, 2015)

Programa multiplataforma creado por Corona Labs Inc. Se compone de un depurador, un simulador y un gestor de proyectos, opcionalmente de un editor y para las versiones de paga un editor de interfaz gráfica de usuario, otros complementos (plugins) pueden comprarse por separado. El lenguaje de programación usado en este software es Lua, principalmente.

Corona SDK® cuenta con una amplia biblioteca de Interfaz de programación de aplicaciones (API). Esto aunado a su simulador de dispositivos, que carga en segundos, y al depurador que proporcionan, permite que el desarrollo de la aplicación sea más rápido. Cuenta con licencia para el desarrollo gratuita, la aplicación puede ser publicada para cualquier cantidad de plataformas.

Como desventaja está el hecho de que se crea dependencia de su API, en caso de querer reescribir el programa con otro SDK la mayoría del código quedaría inutilizable. Otro inconveniente es la falta de acceso a algunas de las bibliotecas nativas de Android, el acceso está restringido a las versiones de paga

La masificación de los dispositivos móviles ha llevado a la creación de las tiendas de aplicaciones, propiciando a su vez el constante desarrollo de software para estos. Al considerar el desarrollar un software educativo se deben considerar aspectos como: Objeto Material y Objeto Pedagógico.

Las aplicaciones en Android se pueden clasificar en tres tipos de tecnologías nativas, web e hibridas. Considerando particularmente el desarrollo de aplicaciones hibridas se cuenta con Apache Flex, Apache Cordova, ShiVa 3D, Corona SDK, entre otros programas.

CAPÍTULO 3 DISEÑO Y DESARROLLO DE LA APLICACIÓN

Este capítulo corresponde a la parte de diseño y desarrollo del programa. En el diseño de la propuesta pedagógica se seleccionará el contenido de la aplicación y se establecerá como ha de funcionar la aplicación. En cuanto a la interfaz, se indicará la cantidad de pantallas que contendrá la app y como funcionarán los elementos contenidos en éstas. Otros temas que cubre éste capítulo son herramientas de desarrollo, producción de los recursos y pruebas de software

3.1 Diseño de la propuesta pedagógica.

Entiéndase propuesta pedagógica como: Modelo de intervención en los procesos formativos de determinados sujetos (Hidalgo Collazos & Cuba Marmanillo, 1999). Se trata de aquella acción que explica las intenciones educativas y sirve de guía para orientar el proceso de aprendizaje-enseñanza (Cortina Navarro, 2013).

La intención de ésta aplicación es que el alumno pueda reforzar sus conocimientos de Geometría Analítica, por lo que se tomaron en cuenta los planes de estudio 2010 de la Facultad de Ingeniería de la UNAM. Para lo que se consideran dos aproximaciones: La primera será una sección llamada Introducción, cuenta con compendio de fórmulas matemáticas así como con textos que servirán a manera de introducción de los diversos temas que se abordarán. La segunda parte consistirá de reactivos de respuesta múltiple. La secuencia de aprendizaje se explicará con más detalle en la sección 3.1.2.

3.1.1 Selección y descripción de los contenidos de cada recurso.

Los temas que se eligieron con base en la asesoría proporcionada por profesores de Geometría Analítica son: Algebra Vectorial y Superficies. El tema Algebra Vectorial será referido en la aplicación como Vectores, por cuestiones de diseño. En la tabla Tema Vectores y en la tabla Tema Superficies se desglosan los subtemas que contienen cada uno de los temas, también se da una muestra de los textos que se incluyen en la sección de introducción y ejemplos de los reactivos contenidos en la app. Para la sección de introducción se ocupo texto del libro de Geometría analítica en el espacio de Érik Castañeda (2003).

Tabla 3-1 Tema Vectores

Vectores	
Subtemas	a) El vector como terna ordenada de números reales. b) Vector nulo. c) Vectores unitarios i, j, k. d) Vectores representados por una combinación lineal de los vectores i, j, k. e) Definición de igualdad de vectores. f) Operaciones con vectores: adición, sustracción y multiplicación por un escalar. g) Propiedades de las operaciones. h) Condición de perpendicularidad entre vectores. i) Componente escalar y componente vectorial de un vector en la dirección de otro. j) Ángulo entre dos vectores. k) Ángulos, cosenos y números directores de un vector. l) Producto vectorial: definición, interpretación geométrica y propiedades. m) Condición de paralelismo entre vectores. n) Aplicación del producto vectorial al cálculo del área de un paralelogramo o) Producto mixto e interpretación geométrica.

Ejemplo de texto contenido en la sección de introducción	• **Vector nulo.** Se llama vector nulo o vector cero a $\bar{0}$ = (0, 0, 0). El vector nulo tiene magnitud nula y no tiene definida ni su dirección ni su sentido. • **Igualdad de vectores.** Dos vectores $\bar{a}$ = (a_1, a_2, a_3) y $\bar{b}$ = (b_1, b_2, b_3) son iguales si y sólo si $b_1 = a_1$, $b_2 = a_2$, $b_3 = a_3$. • **Operaciones con vectores: adición.** La adición de dos vectores es la operación entre $\bar{a}$ = (a_1, a_2, a_3) y $\bar{b}$ = (b_1, b_2, b_3) de tal manera que $\bar{a} + \bar{b}$ = ($a_1 + b_1$, $a_2 + b_2$, $a_3 + b_3$).
Ejemplo de reactivos	➢ Un vector que es simultáneamente perpendicular a $\bar{r}$ = (2, 4, 6) y $\bar{c}$ = (2, 0, 2) es ○ (1, 1, -1) ○ (1, -1, -1) ○ (-1, 1, 1) ○ (-1, 1, -1) ➢ Dado un vector no nulo, la suma de los cuadrados de sus cosenos directores es ○ 1 ○ 0 ○ 3 ○ -1 ➢ El producto vectorial de **j** × **k** es ○ **i** ○ **j** ○ **-j** ○ **k**

Tabla 3-2 Tema Superficies

Superficies	
Subtemas	a) Clasificación de superficies. b) Superficies cuádricas. c) Definición de superficies cilíndricas, cónicas, regladas y de revolución. d) Ecuación vectorial y ecuaciones paramétricas de una superficie cuádrica. e) Obtención de la ecuación cartesiana por el método de las generatrices.
Ejemplo de texto contenido en la sección de introducción	• **Superficies cuádricas o cuadráticas** Se les llama superficies cuádricas o cuadráticas a las superficies que tienen su representación analítica del tipo: $Ax^2 + Bxy + Cy^2 + Dxz + Ez^2 + Fyz + Gx + Hy + Iz + J = 0$ • **Superficies** Se llama superficie al lugar geométrico de todos los puntos que tienen representación gráfica en el espacio de tres dimensiones y cuya relación matemática representativa cuenta con una sola ecuación del tipo: $F(x, y, z) = 0$ Es conveniente hacer notar que no todas las ecuaciones de ese tipo representan una superficie. • **Elipsoide** La forma canónica de la ecuación de la elipsoide es $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1$

Ejemplo de reactivos	➢ La ecuación de una esfera de radio 3 y centro en (-6, 4, -4) es ○ $(x+6)^2 + (y-4)^2 + (z+4)^2 = 9$ ○ $(x-6)^2 + (y+4)^2 + (z-4)^2 = 9$ ○ $(x-6)^2 + (y+4)^2 + (z-4)^2 = 3$ ○ $(x+6)^2 + (y-4)^2 + (z+4)^2 = 3$ ➢ La ecuación $(x+6)^2 + (y-4)^2 + (z+4)^2 = 3$ representa a un cilindro ○ hiperbólico ○ circular ○ elíptico ○ parabólico ➢ Las rectas contenidas en el cilindro circular $(x+6)^2 + (y-4)^2 + (z+4)^2 = 3$ son paralelas al ○ eje y ○ plano XZ ○ eje X ○ plano y = 4

3.1.2 Secuencia de aprendizaje propuesta para el usuario

Con relación a las actividades que ha de realizar el alumno se pretende que en un principio seleccione el tema Vectores, aunque también estará disponible el tema Superficies (además de que la aplicación será diseñada para que se puedan añadir más temas). Una vez seleccionado el tema con el cual desea trabajar, tendrá a su disposición tres módulos a elegir: Introducción, Ejercicios I, Ejercicios II (la app también ha de permitir que se agreguen más módulos).

Introducción le permitirá al alumno revisar diversos conceptos del tema en el que se encuentra, los cuales le pueden ayudar tanto a repasar conocimientos adquiridos, complementar ideas o bien facilitarle la realización de los ejercicios de los otros módulos.

Ejercicios I y II. Cada módulo contará con por lo menos diez reactivos de opción múltiple, cada uno con cuatro posibles respuestas. De los reactivos se le presentarán cinco al usuario, éstos reactivos serán seleccionados aleatoriamente. A su vez las respuestas de cada reactivo serán ordenadas al azar. Los subtemas que cubrirá cada módulo son:

Vectores

- Ejercicios 1. Incisos a, d, e, f, g, h de la Tabla 3-1 Tema Vectores
- Ejercicios 2. Incisos b, c, i, j ,k, l , m, n de la Tabla 3-1 Tema Vectores

Superficies

- Ejercicios 1. Los cinco incisos de la Tabla 3-2 Tema Superficies, destacando los lugares geométricos: un punto, esfera, elipse, parábola, hipérbola, paraboloide hiperbólico, hiperboloide de revolución
- Ejercicios 2. Los cinco incisos de la Tabla 3-2 Tema Superficies, particularmente las superficies cuádricas: cono circular, cono elíptico, cilindro hiperbólico, cilindro parabólico y cilindro circular.

Con el fin de hacer más atractiva la aplicación y aumentar *la dificultad* de los módulos de ejercicios se añadirá un sistema de *vidas,* es decir; el usuario contará con tres vidas, cada pregunta mal contestada hará que el alumno pierda una vida. Si llega a perder las tres vidas se le sacará al usuario del módulo y tendrá que reiniciar la actividad. Por el contrario si logra contestar las diez preguntas sin haber perdido las tres vidas se contará como módulo

completado. Con el fin de que el usuario vaya viendo su progreso aparecerá una marca indicando que el módulo ha sido completado.

Cada tema corresponde a un nivel (nivel 1: Vectores, nivel 2: Superficies), el usuario no podrá realizar ejercicios de un nivel superior al que se encuentre activo, hasta que logre completar todos los módulos de ejercicio del nivel en el que se encuentra. Para el presente caso, el alumno no podrá ingresar a la sección del tema de Superficies sin antes haber completado los módulos de ejercicios de Vectores. (La programación permitirá que haya múltiples temas por nivel).

3.2 Diseño de la interfaz.

La Interfaz gráfica de usuario se diseñará procurando que sea amigable. La aplicación llevará por nombre XOOK (que proviene del idioma maya y significa número, contar), por lo que se buscará que la interfaz sea relacionada tanto al concepto de Geometría Analítica como en cierto grado a los mayas. En los siguientes apartados se entra a detalle en cuanto a las especificaciones de ésta.

3.2.1 Storyline y guión técnico.

El Storyline: la estructura de la app, el orden jerárquico y las relaciones entre las diferentes pantallas de la aplicación a diseñar se muestra el Figura 3-1.

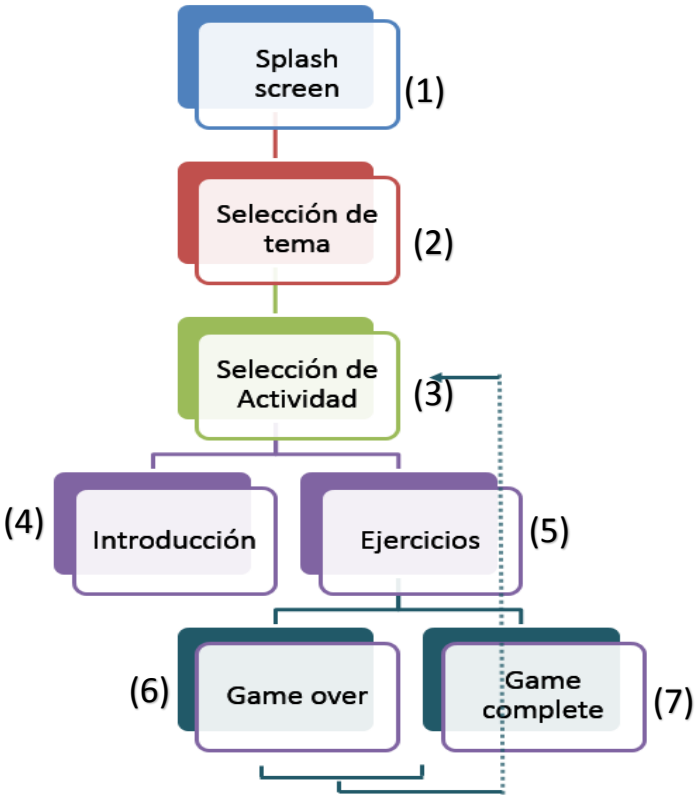


Figura 3-1 Storyline. Diagrama que muestra cada una de las pantallas que compondrán a la app.

La descripción de los elementos que conformarán a cada una de las pantallas contenidas en la app; imágenes, textos, audio, navegación, etcétera. Se muestra en la Tabla 3-3

Tabla 3-3 Guion Técnico

Nombre de la pantalla: Splash screen	Número: 1
Descripción: Pantalla que se muestra al abrir la aplicación, constará únicamente de una animación con sonido	
Texto: Ninguno	
Sonidos: Nombre de la app	Imágenes: Nombre de la app (Xook)
Animaciones: Ninguna	Navegación: pantalla 2
Nombre de la pantalla: Selección de tema	Número: 2

Descripción: Servirá para que el alumno seleccione el tema de su interés	
Texto: Nombre de la app, nombre de los temas (Xook, Vectores, Superficies)	
Sonidos: Ninguno	Imágenes: La imagen de fondo, la imagen que servirá como contenedor de los temas, la de opciones y la de salida.
Animaciones: el cambio de pantalla	Navegación: pantalla 3
Nombre de la pantalla: Selección de actividad	Número: 3
Descripción: Aquí se permitirá elegir la actividad a realizar	
Texto: Nombre del tema seleccionado, nombre de las actividades (Vectores/Superficies, Introducción, Ejercicios 1, Ejercicios 2)	
Sonidos: Ninguno	Imágenes: La imagen de fondo, la imagen que servirá como contenedor de las actividades, la de opciones y la de salida.
Animaciones: el cambio de pantalla, el desplazamiento entre actividades (deslizador)	Navegación: pantalla 2, pantalla 4, pantalla 5
Nombre de la pantalla: Introducción	Número: 4
Descripción: Pantalla que mostrará contenido, texto y formulas, a manera de introducción del tema previamente seleccionado.	
Texto: Título del tema, así como el material correspondiente.	
Sonidos: Ninguno	Imágenes: La imagen de fondo, la imagen que servirá como contenedor del texto, la de opciones y la de salida.
Animaciones: el cambio de pantalla, el cambio de texto (deslizador)	Navegación: pantalla 3
Nombre de la pantalla: Ejercicios	Número: 5
Descripción: Esta pantalla mostrará al usuario las preguntas, permitirá elegir la respuesta. Una vez seleccionada cualquier respuesta la respuesta correcta se mostrara en verde. En caso de ser acertada se procederá a la siguiente pregunta, de lo contrario el usuario perderá una vida.	
Texto: Título del tema, respuesta correcta o respuesta incorrecta, así como el material correspondiente.	

Sonidos: Acierto/error	Imágenes: La imagen de fondo, la imagen que servirá como indicador de vidas, la imagen que servirá como contenedor de los reactivos, la de módulo completado y la de salida.
Animaciones: el cambio de pantalla, el cambio de reactivo (deslizador), la aparición del texto si la respuesta fue correcta, la aparición del texto si la respuesta fue errónea, perdida de vida	Navegación: pantalla 3, pantalla 6, pantalla 7
Nombre de la pantalla: Game over	Número: 6
Descripción: Pantalla que se despliega al perder 3 vidas	
Texto: Inténtalo nuevamente	
Sonidos: fin del juego	Imágenes: La imagen de fondo, la de fin del juego
Animaciones: fin del juego	Navegación: pantalla 3
Nombre de la pantalla: Game complete	Número: 7
Descripción: Pantalla que se despliega después de haber contestado cinco preguntas y de éstas haber acertado en por lo menos tres	
Texto: Bien hecho	
Sonidos: Bien hecho	Imágenes: La imagen de fondo, la de módulo completado
Animaciones: módulo completado	Navegación: pantalla 3

3.2.2 Diseño de los aspectos funcionales.

A continuación se muestra un esbozo de las diferentes pantallas que conformarán la aplicación, con la respectiva descripción de su funcionamiento.

Pantalla de selección de tema. Conformada por una barra de título con un botón de salir (sale de la aplicación), el nombre de la aplicación y un botón de ajustes. El "cuerpo" de la ventana constaría de una especie de árbol formada por los distintos temas a abordar. Cada nivel del árbol indica una precedencia, no pudiendo realizarse dicho nivel hasta no haber concluido satisfactoriamente los anteriores, por ejemplo en la Figura 3-2 Pantalla de selección de tema, no se podrían realizar los temas 2 y 3 hasta no haber concluido los ejercicios del tema 1. Sólo se creara un árbol con todos los temas contenidos en la BD, aunque excedan los límites de la pantalla; si este fuera el caso se ha de poder desplazar el contenido de la pantalla de tal manera que sean visibles los temas que se encuentran más a la derecha.

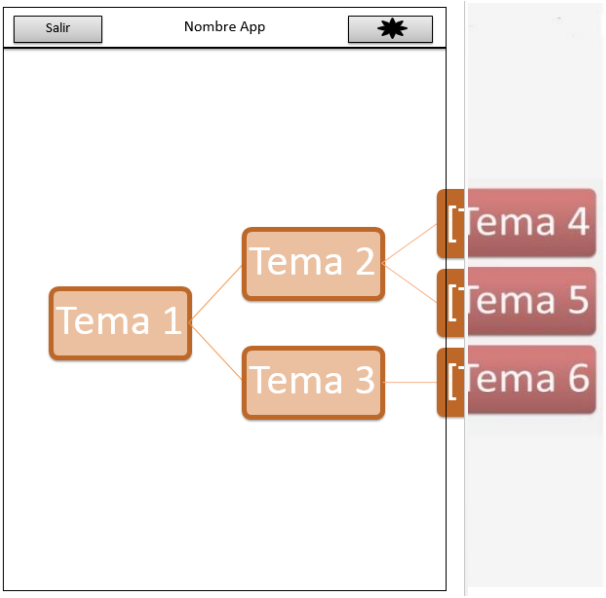


Figura 3-2 Pantalla de selección de tema

Pantalla de selección de actividades. Será la encargada de mostrar las actividades disponibles para el tema en cuestión; una introducción y una o más secciones de ejercicios. Constará de una imagen con: título y subtitulo de la actividad correspondiente, además de un indicador de si la actividad ha sido completada (cuando corresponda). También tendrá

una barra de título que contará con: un botón que permite ir a la pantalla de selección de tema, el nombre del tema en el que se encuentra el usuario y un botón de ajustes.

Figura 3-3 Pantallas de actividades

Pantalla de introducción. En la parte superior tendrá una barra de título con las mismas características que las descritas en la pantalla de selección de actividades, con la diferencia que el botón de atrás desplegaría la pantalla de selección de actividades. Además poseerá uno o más rectángulos que servirán a manera de contenedores del texto que se le quiere mostrar al usuario.

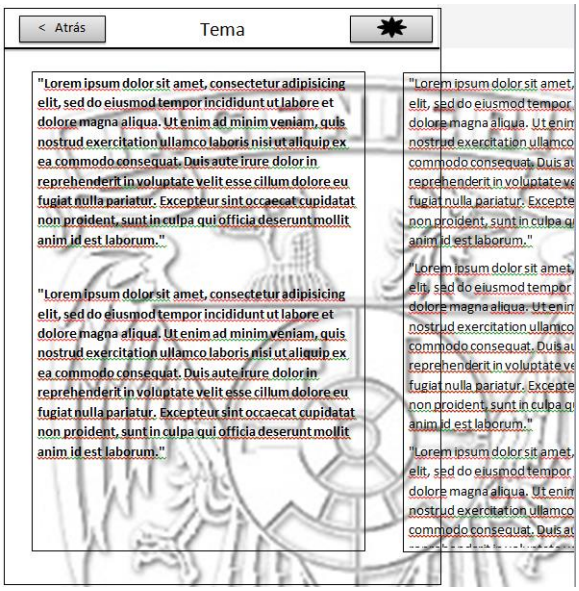


Figura 3-4 Pantalla de Introducción

Pantalla de ejercicios. La Figura 3-5 muestra en general como será la respectiva pantalla y debido a la gran cantidad de elementos con los que cuenta se describirán uno por uno en la Tabla 3-4

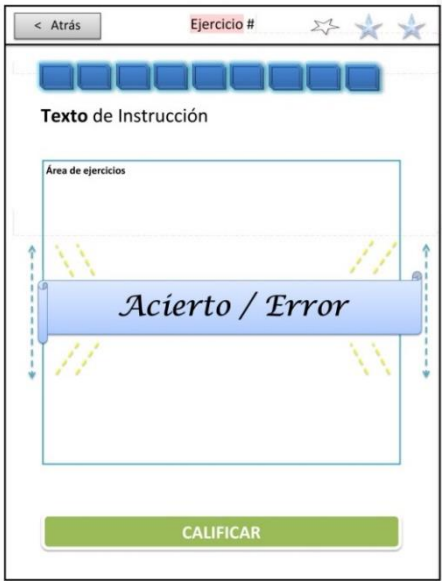


Figura 3-5 Pantalla de ejercicios

Tabla 3-4 Pantalla de ejercicios

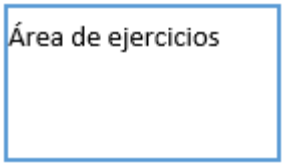
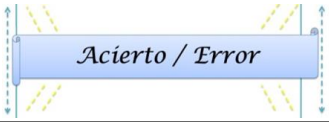
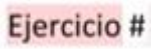
Imagen	Nombre	Descripción
	Botón Atrás	Dirige a la pantalla de selección de actividades
	Indicadores de ejercicios	Muestra cuantos ejercicios se han resuelto y cuantos faltan
Texto de Instrucción	Texto de instrucción	Señala las instrucciones para el presente ejercicio, esto es en función del tipo de pregunta
	Área de ejercicios	Varía en función del tipo de pregunta, es decir puede ser de dos formas distintas. Ver **¡Error! o se encuentra el origen de la referencia.**
	Ventana emergente	Elemento que se muestra al contestar la pregunta indicando si fue contestada correctamente
	Botón de calificar	Al presionar "calificará" la respuesta seleccionada
	Texto de título	Nombre de la sección de ejercicios que se está realizando
	Indicador de vidas	Cuántas vidas le quedan al usuario

Figura 3-6 Tipos de preguntas. El tipo uno consta de cuatro imágenes, y el tipo dos de cuatro oraciones. En ambos casos es sólo una la respuesta correcta.

Además se requerirá de otras 3 pantallas: Pantalla de inicio (o Splash screen), pantalla de fin de juego (Game over) y pantalla de ejercicios completados con éxito (Game complete).

Debido a la complejidad visual de la aplicación y para llegar a uno de los objetivos de este trabajo se decidió solicitar el apoyo para el diseño de la interfaz de la aplicación a la diseñadora gráfica: Lic. Marina Nitzague Casañas Bonilla. A quien se le proporcionará tanto el Storyline como el guion técnico para el desarrollo de la interfaz.

3.2.3 Especificaciones generales de diseño.

Con el fin de lograr una apariencia homogénea y profesional se establecerán los siguientes lineamientos:

Todas las imágenes en el proyecto serán con formato png y deberán ser colocadas en el folder denominado assets. En caso de ser una imagen correspondiente a una pregunta o respuesta habrá que ponerla en la subcarpeta adecuada (Questions o Answers).

El alto de las imágenes para las preguntas y respuestas, de la sección de ejercicios, deberán ser de 75 pixeles por cada renglón. El ancho variara conforme el contenido correspondiente a la pregunta, sin sobrepasar los 600 pixeles.

Nota: Todo renglón que contenga fracciones será considerado como renglón doble.

El tipo de letra para las preguntas, las respuestas y la sección de introducción será: Myriad Pro semibold, en color negro, tamaño de fuente 32. El tipo de letra para los botones será Montserrat-Regular, color blanco.

Los sonidos serán en formato mp3 y deberán ser ubicados en la carpeta Sounds dentro de la carpeta assets.

3.3 Selección de la herramienta de desarrollo

Tomando en cuenta los puntos a favor y en contra de cada una de las herramientas, que se revisaron en la sección 2.4 Programas para desarrollo de aplicaciones híbridas, se eligió Corona SDK® por la facilidad que presenta para programar, aunado a que cuenta con una licencia gratuita. Esta herramienta permite desarrollar para múltiples plataformas; iOS®, Android®, Kindle®, Windows Phone 8® y próximamente Mac OS X® y Windows® (estas últimas se encuentran en fase beta).

El SDK se encuentra en desarrollo continuo, actualmente cuenta con más de 1000 API'S, las que ayudan a que la programación de la app sea relativamente más sencilla. El lenguaje con el que se utilizan las API'S, y con el que se programará el software, es Lua. Lua es un lenguaje de scripting usado en distintos ámbitos, incluyendo el de videojuegos; por ejemplo el popular juego de MMORPG: World of Warcraft®.

Corona SDK® fue creado por Corona Labs®. Corona Labs® es una filial de Fuse Powered Inc. ®. La empresa cuenta con distintas herramientas para programar, o ayudar a la programación, en dispositivos móviles como Corona Enterprise®, Corona Cards®, etc. Sin embargo su producto emblemático es Corona SDK®, producto construido conforme distintos estándares en la industria, como son: OpenGL®, OpenAL®, Box2D®, Facebook®, SQLite®, entre otros. Si bien el software no es open source, sí cuenta con licencia gratuita, por lo que se podrá desarrollar la app sin necesidad de pagar cuota alguna.

3.4 Programación de la aplicación

Para facilitar la comprensión de la estructura del sistema los archivos contenidos en el apk pueden dividirse en 5 partes:

- **Manejador**: main.lua
- **Configuración.** build.settings, config.lua
- **Manejo de datos.** BD2.sqlite, loadsave.lua, data.lua
- **Escenas**. overlayOpciones.lua, overlayResult.lua, overlayZoom.lua, sceneActividades.lua, sceneCero.lua, sceneCreditos.lua, sceneEjercicios.lua, sceneGameComplete.lua, sceneGameover.lua, sceneIntroduccion.lua, sceneSplash.lua, sceneTemas.lua.

- **Carpeta recursos (Assets).** Constituida por los diferentes archivos de imágenes y sonidos.

Manejador

main.lua. Es el encargado de establecer la conexión con la base de datos, así como de extraer toda la información que contiene. En caso de que no exista el archivo con los datos del avance del usuario, main lo crea.

A continuación unos fragmentos del código:

```
-- Require widget and storyboard libraries
local widget = require( "widget" )
local storyboard = require( "storyboard" )


-- Invoke file with "global variables"
local Datos = require( "data" )
```

Estas líneas son una constante a lo largo de las distintas escenas. La biblioteca widget es necesaria para poder dibujar los botones, las imágenes, el fondo, etc. En cuanto a la biblioteca, storyboard es la que nos permitirá el manejo de escenas, como es el mandar a llamar a otra escena, abrir ventanas emergentes o recargar la escena actual.

Se abre el archivo data.lua y su contenido es almacenado en la variable local Datos, con la finalidad de poder compartir valores entre escenas. Este procedimiento se explica con detalle en el apartado de manejo de datos

```
-- Require sqlite3
require "sqlite3"

-- Set the directory for the DB and create it
local path = system.pathForFile("BD2.sqlite",
system.ResourceDirectory)
db = sqlite3.open(path)

-- Module for loading & saving data
local loadsave = require( "loadsave" )

-- Invoke file with "saved data"
local dP = loadsave.loadTable("permanentdata.json") --
datosPermanentes

---If the file permanentdata is empty create the basic
tables
if (dP == nil) then
    dP = {}
    dP.tema = {}
    dP.actividad = {}
end
```

En este segmento de código se indica que se va a ocupar sqlite3, como manejador de la base de datos. Se muestra donde se encuentra el archivo que contiene nuestra base de datos y se manda abrir el archivo que la contiene. Se solicita al archivo loadsave.lua y se almacena en loadsave, lo cual nos permitirá cargar y guardar archivos en formatos JSON. Precisamente lo siguiente que se hace es leer el archivo: permanentdata.json. En caso de que dicho archivo se encuentre vacío se crean las tablas que servirán para guardar el progreso del usuario: dp, dp.tema y dp.actividad. Estos procedimientos se explican con detalle en el apartado de manejo de datos.

```
--Load "THEME" table---------
local sql = "SELECT * FROM THEME"
local increment = 1

--Extract data from DB - temas
for row in db:nrows(sql)do

    Datos.tema[increment] = {}
    Datos.tema[increment].id = row.idTheme
    Datos.tema[increment].nombre = row.name
    Datos.tema[increment].imagen = row.image
    Datos.tema[increment].nivel = row.level


    if (dP.tema[increment] ~= nil) then
        Datos.tema[increment].activo =
dP.tema[increment].activo
        Datos.tema[increment].completado =
dP.tema[increment].completado
    else
        dP.tema[increment] = {}
        dP.tema[increment].activo = row.active
        dP.tema[increment].completado = row.complete
        Datos.tema[increment].activo = row.active
        Datos.tema[increment].completado = row.complete
    end

    increment = increment + 1
end
```

Se inicia con una consulta de SQL que regresará todas las filas contenidas en la tabla THEME. Se establece una variable llamada increment que funcionará como el índice del arreglo tema, es decir, en cada iteración estará cambiando el índice del arreglo. La parte de "if-else" es donde se verifica si dicho tema ya existía o es un tema nuevo. Si no existía se añade tanto a la tabla Datos como al archivo los valores de activo y completado. En caso contrario se toman los datos del archivo permanendata.json y se guardan en la variable Datos.

Configuración

De los 2 archivos contenidos en esta sección el de config.lua se encarga de establecer la relación de aspecto (aspect ratio), permitiendo así que el diseño se mantenga constante aunque se muestre en dispositivos de distinto tamaño. El archivo build.settings permite establecer para que sistema se está desarrollando, en este caso solo Android, cual es la versión de desarrollo, entre otras cosas.

Manejo de datos

Conformado, en un principio, por tres archivos: BD2.sqlite, data.lua, loadsave.lua. BD2 es donde se encuentra almacenada la base de datos (main.lua realiza las consultas para extraer la información de cada tabla) es importante destacar que el programa sólo extrae datos, no escribe en la base de datos. Los datos leídos se almacenan temporalmente en data.lua

```
local Datos = {}
Datos.tema = {}
Datos.introduccion = {}
Datos.actividad = {}
Datos.pregunta = {}
Datos.scenes =
{"sceneTemas","sceneActividades","sceneIntroduccion","sceneE
jercicios","sceneGameover","sceneCompleta","sceneCreditos"}
return Datos
```

data.lua consta de un par de líneas en las que se crea la tabla Datos, anexándole cinco arreglos. Pese a que cada vez que se inicia la aplicación se cargan los valores de la BD, al término de la ejecución se eliminan. Esto porque leer de la BD resultó más rápido que leer todos los datos de un archivo de texto plano o de un archivo tipo JSON. Además el archivo sirve para pasar variables entre las distintas escenas y el archivo main, evitando así el uso de variables globales.

El archivo loadsave.lua fue creado por Rob Miracle (**@MiracleMan**). Sirve para guardar tablas en un archivo con extensión json, y para leer este tipo de archivos. Se implementa tanto en main.lua como en sceneCompleta.lua, para crear y guardar el archivo permanentdata.json

Al ser ejecutado por primera vez el programa crea un cuarto archivo: permanentdata.json. En este archivo es donde se guardan los avances del usuario. Se guardan únicamente el índice correspondiente al tema, si se encuentra activo o completado, el índice correspondiente a la actividad y si es que ha sido completada. En caso de actualización de la app no se sobrescriben los avances.

Escenas

Esta sección conforma principalmente la "parte visible" de la aplicación, y es que prácticamente cada uno de los archivos corresponde a cada una de las pantallas que se le muestran al usuario. Una de las excepciones a esta relación es la de los elementos que son persistentes a lo largo de la aplicación; los contenidos en la barra de título, que son establecidos en sceneCero.lua. Eso se logra con las siguientes líneas de código:

```
--Create Title Bar Group
    local display_stage = display.getCurrentStage()
    display_stage:insert( 1,fondo )
    display_stage:insert( 2,titleBarBottom )
    display_stage:insert( 3,titleBarText )
    display_stage:insert( 4,subTitleBarBottom )
    display_stage:insert( backButton )
    display_stage:insert( settingsButton )
```

En cuanto a la escena que convoca sceneCero es la de Splash. La pantalla que muestra el inicio de la aplicación es la de sceneSplash, ésta a su vez llama a sceneTemas.lua; que es en donde se muestra el árbol y se puede seleccionar el tema a repasar. Cabe destacar que esto es hecho de manera dinámica, en cuanto se agregue o elimine algún tema se autoajustara el despliegue de estos. Al elegir un tema se guarda en una variable el identificador del tema elegido, para indicarle a la siguiente pantalla los datos a cargar. Independientemente del tema elegido se llamara a la escena Actividades.

En la escena Actividades se coloca un recuadro con el nombre de las secciones disponibles. Primero se muestra la sección llamada introducción. A continuación se muestra uno o más recuadros correspondientes a la sección de ejercicios, esto dependerá de cuantas de estas secciones se crearon en la BD. Para navegar entre estas secciones se ocupan los botones colocados debajo del recuadro que muestra la actividad. Dependiendo de cuál de estas secciones sea elegida será la ventana que se desplegará.

Si se seleccionó la sección de introducción, el archivo a cargar será el de sceneIntroduccion.lua. Aquí se muestra la información contenida en la tabla de Introducción que corresponda con el tema seleccionado. Se mostrara en rectángulos independientes cada fila obtenida de la BD.

Pero si se seleccionó la escena de ejercicios, se carga no solo la que es la escena más compleja sino la que probablemente es la principal escena del proyecto; sceneEjercicios.lua. Ésta consta de 10 funciones, 20 variables a las que se puede acceder en cada una de las funciones, y un manejo de escena distinto al usado en el resto del proyecto. Al ser esta una escena que se recarga (*refresca*) tiene una estructura que consta de 4 funciones para el manejo de escenas (event listeners), se describe cómo interactúan en la Figura 3-7.

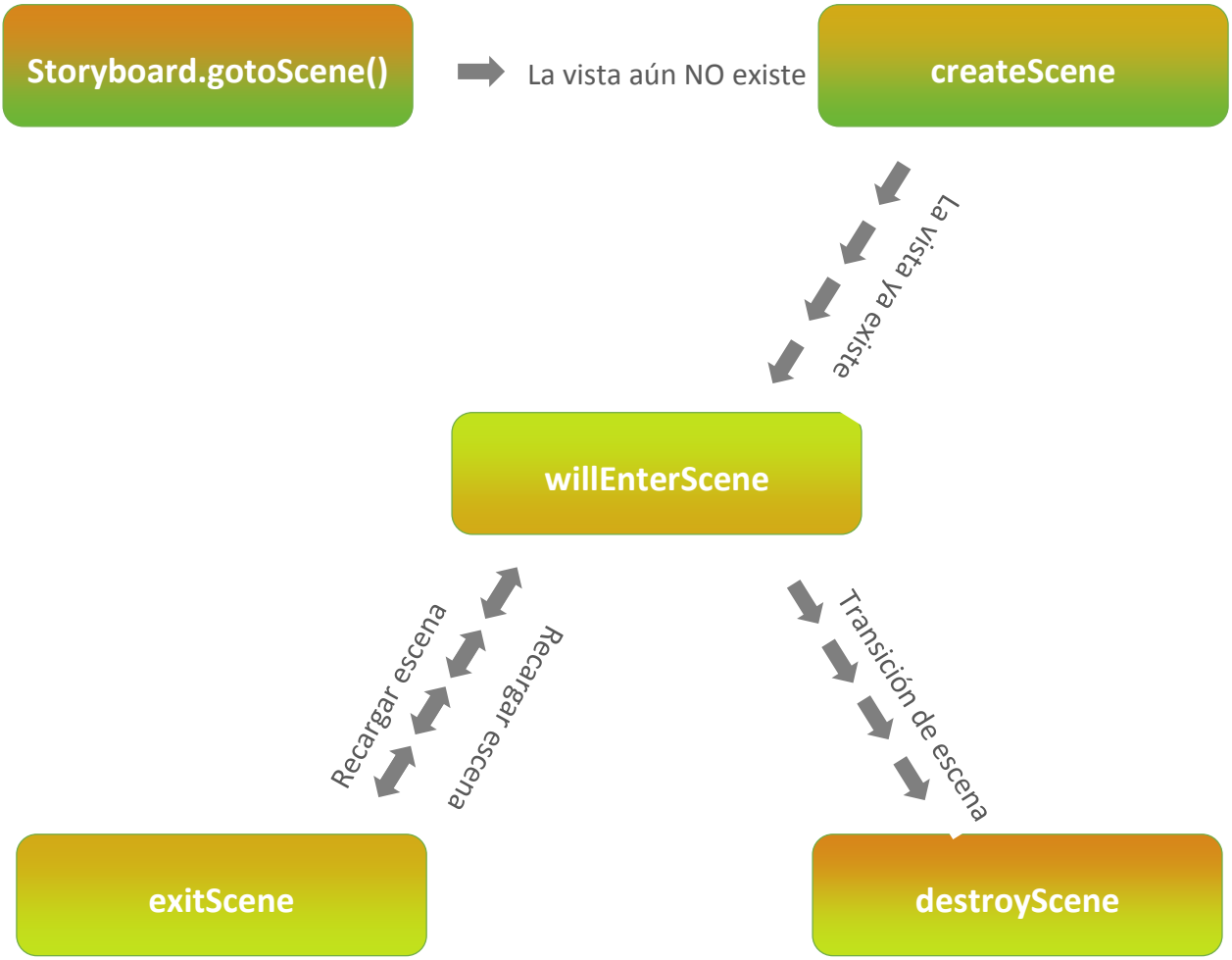


Figura 3-7 Secuencia de eventos en las escenas

Fuente: Adaptado de (Beebe, 2012)

Lo primero es crear e inicializar las variables y objetos que van a ser ocupados; las variables que están declaradas al inicio del archivo, además de las variables y objetos declarados en la función createScene. En la función se establece la cantidad de preguntas que van a ser mostradas y, en base a una selección aleatoria, se extraen los reactivos de la tabla. Acto seguido se carga la función willEnterScene.

En willEnterScene se dibuja la pregunta, se crea un arreglo para establecer de manera aleatoria el orden de las respuestas, y se despliegan las respuestas. Tanto las preguntas como las respuestas cuentan con un manejador (handler); handleTapEvent, que "llama" a la escena overlayZoom.lua, que como su nombre lo indica, es la encargada de hacer un zoom al elemento seleccionado. Para el caso de las respuestas se les adhiere otro manejador: onSwitchPress, el que permite que detectemos cuando se seleccionó una respuesta, activando el botón de Siguiente y guardando en la variable correcta su respectivo valor (1 en caso de ser la respuesta correcta, 0 de otro modo).

En cuanto el usuario presiona el botón de Siguiente se inicia la función handleButtonEvent, de la cual se muestra un fragmento del código a continuación:

```
local function handleButtonEvent( event )

      if (boton:getLabel()=="Continuar") then
            continua()
      else --First time that it's pressed
            answerBox[1]:setFillColor(0,1,0)
            if(correcta~= 1)then --If the answer was wrong
                  answerBox[choose]:setFillColor(1,0,0)
                  vidas = vidas + 1 --Loose one life
                  gaugeSprite:setSequence( "health" .. vidas )
```

Inicia verificando si la etiqueta está establecida como Continuar, con el fin de ver si es la primera vez que se presiona el botón durante la pregunta actual.

Si es la primera vez, marca la respuesta correcta con verde. Revisa si la seleccionada es la correcta, si lo es continua. En caso contrario marca la respuesta seleccionada con rojo, cambia la variable que indica la cantidad de vidas e inicia las animaciones de vidas perdidas.

Si es la segunda vez que se presiona el botón pasa a la función *continua* la cual está conformada por el siguiente código:

```
-- the following event is dispatched to continue or end the test
continua = function ()
      if(vidas==4) then
            storyboard.gotoScene("sceneGameover","slideLeft",300)
      else
            if (ciclo < cantidadPreguntas) then -- # of questions to
be shown
                  storyboard.reloadScene()
            else

      storyboard.gotoScene("sceneCompleta","slideLeft",300)
```

Lo que hace es revisar si se llegó al límite de intentos fallidos en cuyo caso manda llamar a la escena sceneGameover y a la función destroyScene. De lo contrario verifica si ya se han respondido todas las preguntas que se habrían de mostrar. Si no se cumple la condición se manda recargar la escena con lo que se ejecuta a la vez exitScene. Si esto es así se pasa a destroyScene, invocando a su vez a la escena sceneCompleta.

La escena sceneCompleta se encarga de: marcar la actividad como realizada, verificar si todas las actividades del tema fueron cumplidas, de ser así revisar si todos los temas del mismo nivel fueron completados. Guardando estos datos en el archivo permanentdata.json

Carpeta recursos

Contiene todas las imágenes del sistema; iconos y botones de la aplicación, dos carpeteas (Questions y Answers) con las imágenes de las preguntas y las respuestas. La carpeta, nombrada assets, tiene además la carpeta Sounds donde, como su nombre lo indica, se encuentran los sonidos utilizados por la aplicación.

3.5 Pruebas de software

La prueba de software, mejor conocida como *software testing*, es el proceso de ejecución de un programa con la intención de descubrir un defecto, así como evaluar la calidad del programa y mejorarlo. El principal objetivo de una prueba es detectar errores, por lo que una prueba tiene éxito si se descubre un error no detectado hasta entonces (Myers, 1979).

Algunos puntos importantes sobre las pruebas de software son:

- Las pruebas deberían planificarse mucho antes de que empiecen (Davis A. , 1995).
- Las pruebas deberían empezar por "lo pequeño" y progresar hacia "lo grande" (Pressman, 1998).
- Al generar casos de prueba, se deben incluir tanto datos de entrada válidos y esperados como no válidos e inesperados (Pressman, 1998).

Es importante que cuando se realicen las pruebas del sistema se realice tanto la verificación cómo la validación de éste.

La verificación se lleva a cabo por el desarrollador y es con el fin de comprobar que se esté construyendo el programa correctamente. Éste es un proceso objetivo.

La validación es hecha por usuarios o testers, sirve para confirmar que se esté construyendo el programa correcto. Se ejecuta el software y se ve si cumple con las expectativas. Éste es un proceso subjetivo.

3.5.1 Niveles de prueba

Existen distintos niveles de prueba, en el presente documento se describirán únicamente aquellos relativos a la funcionalidad y que fueron utilizados en este desarrollo.

Prueba unitaria. Es la prueba de componentes individuales del programa o módulos. Esto normalmente realizado por los programadores y no por los testers, por el hecho de que se requiere conocimiento detallado del diseño y código del programa. Parte de las pruebas de caja blanca, o pruebas unitarias, es la depuración.

La depuración conlleva identificar, aislar y corregir los errores (bugs). Es común que los programadores realicen debugging al encontrarse con un error en el programa.

Prueba de integración incremental (Bottom-up Integration testing). Las pruebas se van realizando conforme se van agregando nuevos componentes o nuevas características, para determinar si los cambios funcionan correctamente. Esta prueba puede ser realizada por los programadores o por los testers.

Prueba de sistema. Éste es el siguiente nivel de pruebas y es donde se verifica el sistema con todos sus componentes ya integrados. Aquí es donde se prueba que el sistema en su conjunto cumpla con los estándares de calidad.

Pruebas alfa. A la combinación de las pruebas: unitaria, de integración y de sistema, se le conoce como prueba alfa. Representan la primera etapa de las pruebas y han de ser realizadas por los desarrolladores o los equipos de calidad. Durante ésta fase se prueba lo siguiente en la aplicación: errores ortográficos, enlaces rotos, desempeño de la aplicación.

Prueba de facilidad de uso o usabilidad. Se prueba la fluidez de la aplicación; ¿qué tan sencillo le resulta al usuario el manejar la aplicación recién creada?

Pruebas beta. Ésta prueba se realiza después de que la prueba alfa se ha realizado con éxito. En la fase beta una muestra de la audiencia objetivo prueba la aplicación. Las pruebas beta también se conoce como la prueba pre-lanzamiento (*prerelease*). Las versiones beta de software son idealmente distribuidas para una amplia audiencia en la Web, en parte para dar al programa una prueba del "mundo real" y en parte para proporcionar una vista previa de la próxima versión. En esta fase la audiencia probará lo siguiente:

- Los usuarios instalarán y ejecutarán la aplicación. Enviarán retroalimentación al equipo del proyecto: errores tipográficos, si el flujo de la aplicación les resulta confuso, e incluso caídas del sistema.
- Al obtener la retroalimentación, el equipo del proyecto puede corregir los problemas antes de liberar el software para los usuarios finales.
- Entre más problemas se arreglen, mayor será la calidad de la aplicación.
- Al tener una aplicación de mayor calidad aumentará la satisfacción del cliente.

Prueba de aceptación. Probablemente la prueba más importante. Normalmente este tipo de pruebas es realizado para verificar si el sistema cumple con los requerimientos de los usuarios. Los usuarios o los clientes son los que realizan esta prueba y determinan si aceptan, o no, la aplicación.

Las pruebas de aceptación no sólo son realizadas para encontrar errores ortográficos o de interfaz, sino también para encontrar bugs en la aplicación que provoquen un mal funcionamiento.

Al realizar este tipo de pruebas se revisa que se cumpla con los requerimientos establecidos.

En la Figura 3-8 se muestra un esquema ilustrativo de los niveles de testing.

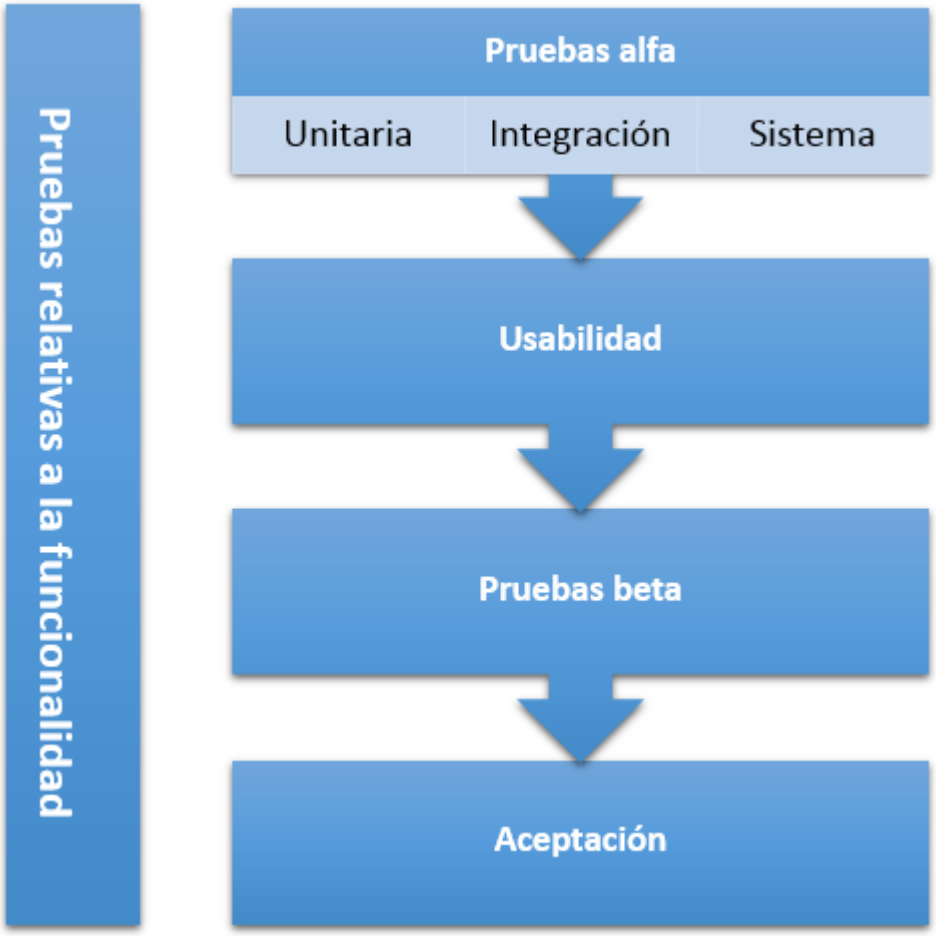


Figura 3-8 Niveles de prueba

3.5.2 Métodos de prueba

Los métodos que se usarán son: el de caja blanca y el de caja negra. Los cuales se describen brevemente.

Prueba de caja blanca. Es una técnica de verificación que los ingenieros de software pueden usar para examinar si su código funciona como se espera.

El método de prueba de caja blanca también es conocido como prueba estructural, prueba de caja clara o prueba de caja de cristal (Boris, 1995). El método de prueba de caja blanca toma en cuenta el mecanismo interno de un sistema o componente (IEEE Computer Society, 1990). Para poder realizar éste tipo de prueba en una aplicación el tester necesita tener conocimiento del funcionamiento del código. El tester necesita ver el código fuente e identificar qué sección del código se está comportando inapropiadamente.

Prueba de caja negra. A la técnica de probar una aplicación sin tener conocimiento de cómo funciona ésta internamente se le conoce como prueba de caja negra, o prueba de comportamiento.

La prueba de caja negra se centra en determinar si un programa hace lo que el cliente quiere que haga. El tester debe desconocer la arquitectura del sistema y no requiere contar con acceso al código fuente. Esto es, interactuara con la interfaz de usuario del sistema ingresando datos (inputs) y examinando las salidas obtenidas (outputs) sin saber cómo son procesados los datos que ingreso.

Sin embargo es importante destacar que, sin importar la cantidad de pruebas, no se puede asegurar la ausencia de errores o defectos en el código.

3.5.3 Pruebas

Las pruebas unitarias se realizaron conforme cada módulo (cada archivo con extensión lua) se desarrollaba. Estas se efectuaron prácticamente en conjunto con las pruebas de integración incremental.

Las pruebas de integración incremental se hicieron a partir de que se programó el primer archivo (main.lua), lo que permitió vislumbrar como se manejarían los datos y por ende como se trabajarían estos en el archivo data.lua. Durante esta fase se ocupó tanto el debugger como el simulador de Corona Labs®, el primero permitiendo que se realizaran pruebas de caja blanca y el segundo de caja negra. Conforme se fueron ejecutando las pruebas se realizaron diversas modificaciones; cambios a la base de datos, ajustes a las funciones en el módulo de ejercicios, etc.

En las pruebas de sistema se apreció que convenía cambiar la interfaz, los principales cambios se muestran a continuación.

En la pantalla de selección de actividad se eliminan las flechas para desplazamiento y en su lugar se colocan en la parte baja unos botones para navegar, como se muestra en la siguiente figura:

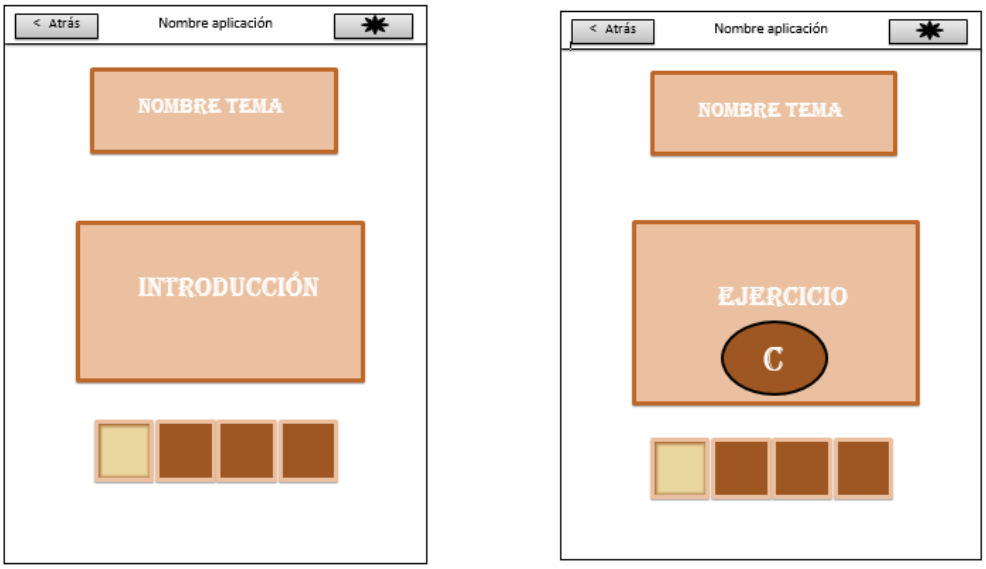


Figura 3-9 Nuevas pantallas de actividades

En la pantalla de introducción se elimina la imagen de fondo debido a que perjudicaba la lectura, quedando la pantalla de la siguiente forma:

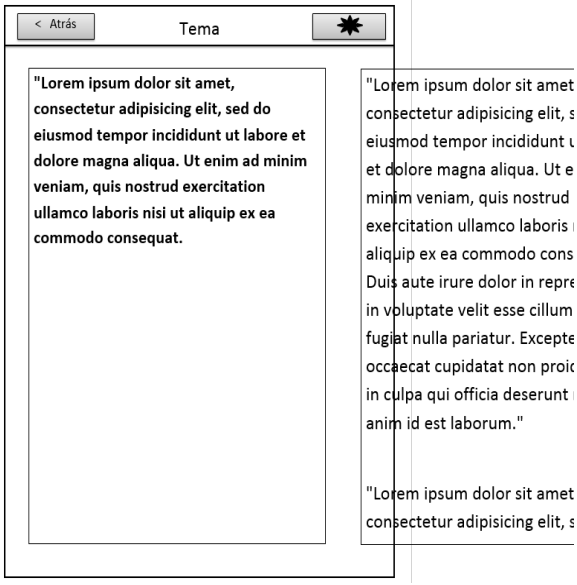


Figura 3-10 Nueva pantalla de introducción

Para la pantalla de ejercicios se decide eliminar la barra que indicaba cuantos ejercicios se habían completado, para permitir que el área correspondiente a las actividades sea mayor, como se puede apreciar en la Figura 3-11.

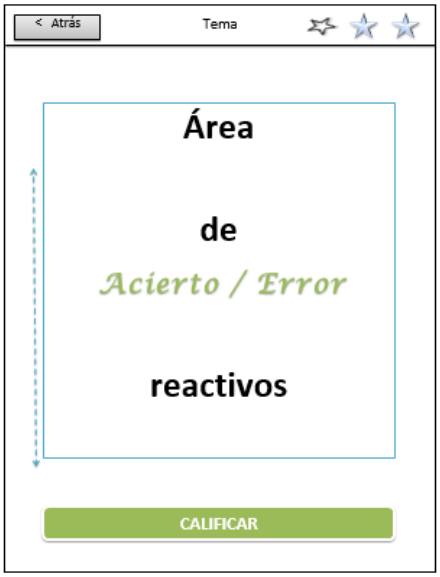


Figura 3-11 Nueva pantalla de ejercicios

Las pruebas beta, que representan únicamente pruebas de caja negra, se realizarán una vez publicada la aplicación. Los resultados de esta prueba se verán reflejados en el capítulo: 3.7Evaluación de la aplicación.

3.6 Publicación en repositorios web

La publicación de la aplicación que fue realizada en el presente trabajo se realizó en el sitio web "CERAFIN" de la División de Ciencias Básicas de la Facultad de Ingeniería: http://dcb.fi-c.unam.mx/cerafin/bancorec/descargas/xook.apk. También disponible en la plataforma Google Play® a la que se puede acceder para su descarga mediante este enlace: https://play.google.com/store/apps/details?id=mx.unam.comunidad.julio_bonilla.xook.

3.7 Evaluación de la aplicación.

En este apartado se diseñará una breve encuesta y se aplicará a los usuarios de la aplicación; se mostrarán los resultados, todo esto con la finalidad de conocer si el contenido y la usabilidad, de la aplicación, son adecuados para los usuarios.

La encuesta se aplicará a un grupo de usuarios con el perfil al que está orientada la aplicación, que son alumnos de los primeros semestres de las carreras de ingeniería de la Universidad Nacional Autónoma de México. Se aplicarán 28 encuestas que nos ayuden a orientarnos para realizar las modificaciones necesarias.

3.7.1 Proceso para la evaluación

Para poder obtener información útil de los usuarios, que permita perfeccionar y ajustar la aplicación en sus diferentes módulos, se les realizará una encuesta. Para ello, se considerará que los elementos principales a evaluar son los correspondientes a:

- Usabilidad (amigabilidad de la interfaz): Se solicitará a los usuarios que informen sobre la facilidad con la que pueden utilizar el software.
- Contenido: Se les preguntará a los usuarios acerca de si los contenidos ofrecidos en la aplicación les son útiles y comprensibles.
- Presentación: Se pedirá a los usuarios su opinión del ambiente gráfico de la aplicación.
- Aceptabilidad: Se preguntará a los usuarios si, en general, les agradó la aplicación y si la utilizarían como un método de repaso.

La encuesta tendrá 6 preguntas con respuestas de opción múltiple (con los rangos: mucho, algo, nada), y 1 pregunta abierta para obtener alguna información que pudiera ser útil.

En función de los elementos que se desean conocer y evaluar, los reactivos de la encuesta serán los siguientes:

1. ¿Fue fácil **utilizar** la aplicación (botones, menús, pantallas)?
2. ¿Fue fácil de **entender** las preguntas y los temas que se muestran en la aplicación?
3. ¿Crees que los ejercicios de la aplicación te ayudan a **repasar** los temas de Geometría Analítica que ves en clase?
4. ¿Te agradó la **presentación** de la aplicación (temática, colores, diseños)?
5. ¿Crees que la aplicación te ayudaría a repasar lo aprendido en clase, **en lugares** como el transporte, la calle, tu casa, etc.?
6. En general, ¿**Utilizarías** la aplicación **como apoyo** de repaso de lo aprendido en clase?
7. ¿Qué opinión nos podrías dar para el mejoramiento de la aplicación?

Las respuestas de las encuestas realizadas se pueden consultar en el ANEXO 4 RESULTADOS DE LA EVALUACIÓN .

3.7.2 Resultados

Los datos obtenidos son los mostrados en la siguiente tabla:

Tabla 3-5 Resultado de las encuestas

	RESPUESTAS		
PREGUNTAS	MUCHO	ALGO	NADA
1	28	0	0
2	22	6	0
3	20	8	0
4	10	14	4
5	22	6	0
6	22	6	0

Posterior al análisis de las respuestas de la pregunta No. 7 (que es una pregunta abierta), las opiniones que fueron más útiles, y que más se orientaron hacia el mismo punto, son las siguientes:

-Los colores de la aplicación deberían ser más brillantes o llamativos

Dada la similitud de las opiniones sobre esta parte de la interfaz, se tomará en cuenta para aplicar mejoras a una versión posterior.

CAPÍTULO 4 CONCLUSIONES Y RECOMENDACIONES.

Los avances tecnológicos han permitido que los dispositivos móviles permeen en los distintos estratos y sectores de la sociedad. Hoy en día es común ver en cualquier espacio público de la ciudad a alguna persona ocupando un dispositivo móvil, siendo el más común el celular. No sólo ha avanzado en el poder de procesamiento de dichos dispositivos, la cantidad de megapíxeles con los que cuenta la cámara que comúnmente traen integrada, o en general las mejoras en lo que a Hardware respecta; también se ha notado un considerable aumento en la cantidad de aplicaciones disponibles y la demanda de éstas. Por lo que se decidió utilizar en el presente trabajo semejante éxito en el mercado, para atacar una problemática que se presenta en la Facultad de Ingeniería de la UNAM que es el alto número de reprobados en la materia de Geometría Analítica.

Al optar por el desarrollo de una aplicación para dispositivos móviles hubo que investigar los sistemas operativos con los que cuentan éstos, como Android, IOS®, Firefox OS®, etc. Si bien el sistema elegido fue Android, debido principalmente a la gran cantidad de usuarios con los que cuenta, al elegir la herramienta con la que habría de programarse la app se eligió una que facilitara el lanzamiento multiplataforma; Corona SDK®.

En una primera instancia el kit de desarrollo de Corona facilitó la programación de la interfaz y la lectura de la BD. Sin embargo, al llegar a la parte correspondiente al contenido de la app, la elección de este software probó no ser la más adecuada. El contenido del programa consta en su mayoría de fórmulas matemáticas y si bien Corona presenta herramientas para texto, ninguna facilita la escritura de fórmulas; lo que procedió fue escribir las formulas en LaTeX y exportarlas cómo imágenes, sin embargo dicho problema habría sido más simple solucionarlo con un programa que soportara HTML5.

Se cumplieron los objetivos planteados en el presente trabajo. Se investigaron una serie de herramientas para el desarrollo de aplicaciones seleccionando una herramienta gratuita, no así de código libre, con la cual se programó la app. El material o contenido que se desarrolló cubre dos temas indispensables para el apoyo del estudio: superficies y vectores. Se tomó en cuanta en el desarrollo la posibilidad de agregar más temas a la aplicación, lo cual se puede realizar como se muestra en el manual que se encuentra en el ANEXO 2 MANUAL PARA AÑADIR CONTENIDO A LA APLICACIÓN

.

Se desconoce exactamente el nivel de refuerzo que este programa dará a los alumnos, pero se espera que la aplicación creada sea una herramienta que ayude a los futuros ingenieros en su formación ya que se planeó utilizando los principales temas que ellos abordan en sus estudios, según los planes de estudio que tiene la UNAM para los primeros años.

El proceso de desarrollo puede continuar indeterminadamente, siempre hay trabajo por hacer. La aplicación podrá estar disponible para que se pueda modificar y agregar otros contenidos. Para mejorar la aplicación se recomienda:

- Aumentar el banco de preguntas de los temas existentes. Se cuentan con 50 *preguntas* para cubrir los temas ya mencionados, un aumento en esta cantidad de preguntas podría ayudar al alumno a una mejor comprensión de los temas.
- Ampliar la cantidad de *temas* con los que cuenta la app. Originalmente se diseñó con temas de Geometría Analítica, posteriormente se podrían agregar temas de otras materias.

- Facilitar el incremento de contenido. Si bien para aumentar el contenido basta con agregar las imágenes a las carpetas correspondientes y agregar las entradas a la BD, esto podría simplificarse aún más. Por ejemplo, el interesado en agregar nuevos o diferentes contenidos podría hacerlo con una aplicación que al arrastrar la imagen añada el registro automáticamente a la base de datos (*Drag and Drop*).

Este trabajo me permitió, además de emplear todo el conocimiento adquirido en la carrera, reforzar o adquirir conocimientos sobre: el lenguaje de programación Lua, el desarrollo de aplicaciones para dispositivos móviles, diseño de Bases de Datos, entre otros.

GLOSARIO

Amazon.- Aplicación diseñada por la empresa Amazon.com, Inc. Compañía estadounidense de comercio electrónico.

Archivo apk.- Archivo de paquetes Android (Application package). Contiene todo lo necesario para instalar la aplicación en un dispositivo Android.

Copyleft. – Es un método general para hacer un programa (u otro tipo de trabajo) libre, exigiendo que todas las versiones modificadas y extendidas del mismo sean también libres (Free Software Foundation, 2015).

Debugging. - La depuración se refiere al hecho de poder ejecutar un programa en un ambiente controlado de manera que se pueda estudiar el flujo de ejecución, la pila de llamadas, los valores de las variables, etcétera (M.C. Edgar E. García Cano, 2015).

Dropbox. – Servicio de alojamiento de archivos en la nube. Su aplicación para Android fue lanzada al mercado en el 2010 (Ying, 2010).

Evernote. – Sistema organizador de notas, disponible en múltiples plataformas, lanzado al mercado en el 2008 (Libin, 2008).

Formato JSON (JavaScript Object Notation). – Es un formato ligero de intercambio de datos, basado en el lenguaje de programación JavaScript (JSON, s.f.).

Formato PNG (Portable Network Graphics). – Es un formato abierto basado en un algoritmo de compresión sin perdida (Roelofs, 2016).

Framework. – Es un patrón para el desarrollo o la implementación de una aplicación (Sánchez, 2006)

Infinity Blade 3. – Juego de acción desarrollado por la compañía Epic Games, publicado el en el 2013 para el sistema operativo IOS (Davis J. , 2013).

LinkedIn. – Aplicación que permite conexión con el sitio web del mismo nombre. Sitio de red profesional.

MMORPG (Massively Multiplayer Online Rolling Play Game). – Género de videojuegos que permite que admiten un amplio número de jugadores simultáneamente.

MXML (Macromedia eXtensible Markup Language). – Lenguaje descriptivo o lenguaje de marcado basado en XML

Netters (Netter's Anatomy Atlas). – Aplicación sobre la anatomía humana con ilustraciones del Dr. Frank H. Netter.

Plugins. – Software que añade características específicas a otro programa.

Retrocompatibilidad. - Capacidad un sistema de permitir la ejecución o el uso de versiones del software anteriores a la actual (colaboradores de Wikipedia, 2015)

Shader. – Es un programa definido por el usuario para ejecutarse en la Unidad de Procesamiento Gráfico (GPU) o en la tarjeta gráfica. Determinan como es que un objeto será dibujado (Valve, 2015).

Starmap. - Aplicación de astronomía. Herramienta para "mirar las estrellas" apta para todo público; desde entusiastas hasta profesionales.

Formato SWF (Small Web Format). – Formato de archivo que puede contener sonido, video, u otros datos. Este tipo de archivos pueden ser ejecutados en los navegadores mediante el complemento Adobe Flash Player (Adobe Systems Incorporated, 2012).

Swype. – Teclado virtual para dispositivos móviles. Aplicación disponible oficialmente en el mercado a partir del 2013 (Moscaritolo, 2013).

Tester. – También conocido como Ingeniero de pruebas o analista de pruebas.

World of Warcraft (WoW). – Juego desarrollado por Blizzard Entertainment, contaba con 5.5 millones de suscriptores a finales de septiembre de 2015 (BusinessWire, 2015).

WYSIWYG (What You See Is What You Get). - En español: lo que ves es lo que obtienes. Implica una interfaz que permite al usuario ver algo muy similar al resultado final, mientras se realiza el proceso de creación.

MESOGRAFÍA

IEEE Computer Society. (1990). *IEEE Standard Glossary of Software Engineering Terminology.* New York: Institute of Electrical and Electronics Engineers.

Adobe Systems Incorporated. (2012). *SWF File Format Especification.* Retrieved from https://a248.e.akamai.net/f/1953/8974/2h/wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/swf/pdf/swf-file-format-spec.pdf

Apple. (2008, Julio 10). *Apple Press Info*. Retrieved from apple: http://www.apple.com/pr/library/2008/07/10iPhone-3G-on-Sale-Tomorrow.html

Beebe, J. (2012, 03 27). *Coronalabs Blog*. Retrieved from http://coronalabs.com/blog/2012/03/27/storyboard-scene-events-explained/

Boris, B. (1995). *Black-box testing: techniques for functional testing of software and systems .* New York: John Wiley & Sons.

BusinessWire. (2015, Noviembre 02). *businesswire*. Retrieved from http://www.businesswire.com/news/home/20151102007005/en/Activision-Blizzard-Announces-Agreement-Acquire-King-Digital

Castañeda De Isla Puga, É. (2003). *Geometría analítica en el espacio.* México: Facultad de Ingeniería-UNAM.

colaboradores de Wikipedia. (2015, Julio 28). *Wikipedia, La enciclopedia libre* . Retrieved from Retrocompatibilidad : https://es.wikipedia.org/w/index.php?title=Retrocompatibilidad&oldid=84071944

Corona Labs. (2015). *Corona Labs Blog*. Retrieved from https://coronalabs.com/blog/2011/11/25/black-friday-corona-sdk-giveaways/

Cortina Navarro, C. (2013). *MODULO DE PROCESOS PEDAGOGICOS.* Retrieved from isssu: http://issuu.com/luzkarinnegonzalezjulio/docs/modulo_de_procesos_pedagogicos

Davis, A. (1995). *201 Principles of Software Development.* McGraw-Hill.

Davis, J. (2013, septiembre 17). *IGN*. Retrieved from http://www.ign.com/articles/2013/09/18/infinity-blade-iii-review

eduapps. (2014, Agosto 14). *eduapps*. Retrieved from http://www.eduapps.es/

Free Software Foundation. (2015). *El sistema operativo GNU*. Retrieved from http://www.gnu.org/copyleft/copyleft.es.html

González Castañón, M. Á. (1998). Evaluación de Software educativo: orientaciones para su uso. In C. M. Zea Restrepo, *Conexiones : ambiente tecnologico educativo .* Medellin : Universidad EAFIT. Retrieved from tecnoedu.net.

Google. (2013, 05 15). *you tube*. Retrieved from https://www.youtube.com/watch?v=1CVbQttKUIk

Google. (2014, junio 25-26). *www.google.com*. Retrieved from https://www.google.com/events/io#wtLJPvx7-ys

Hidalgo Collazos, L., & Cuba Marmanillo, S. (1999). *Construyendo la nueva escuela. Proyecto Educativo Institucional, Volumen II .* Lima: Tarea.

Jackson, J. (2011, Noviembre 17). *TechWorld*. Retrieved from http://www.techworld.com.au/article/407714/adobe_donates_flex_apache/

JSON. (n.d.). *Introducción a JSON*. Retrieved 02 25, 2015, from http://www.json.org/json-es.html

Korf, M., & Oksman, E. (n.d.). *Salesforce developers*. Retrieved from salesforce.com, inc.: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Libin, P. (2008, Junio 24). *Evernote*. Retrieved from https://blog.evernote.com/blog/2008/06/24/evernote-public-launch/

M.C. Edgar E. García Cano, I. J. (2015). *Laboratorio de Computación Salas A y B.* Retrieved from Anexo: Depuración de programas: http://lcp02.fi-b.unam.mx/static/docs/PRACTICAS_FP/Anexo%20DepuracionProgramas.pdf

mobiThinking. (2014, Mayo 16). *mobiforge*. Retrieved from http://mobiforge.com/research-analysis/global-mobile-statistics-2014-part-a-mobile-subscribers-handset-market-share-mobile-operators

Moscaritolo, A. (2013, Abril 24). *PCMag*. Retrieved from http://www.pcmag.com/article2/0,2817,2418104,00.asp

Myers, G. (1979). *The Art of Software Testing.* Wiley.

Pinch Media. (2008, Julio 15). *PinchMedia*. Retrieved from http://web.archive.org/web/20080721054229/http://feeds.feedburner.com/Top100PaidIphoneApplications-PinchMedia

Pressman, R. S. (1998). *Ingeniería del Software: Un enfoque práctico, cuarta edición.* McGraw-Hill.

Roelofs, G. (2016, 01 16). *PNG(Portable Network Graphics) Home Site*. Retrieved from http://www.libpng.org/pub/png/

Sánchez, J. (2006, Septiembre 29). *jordisan.net*. Retrieved from http://jordisan.net/blog/2006/que-es-un-framework/

ShiVa Technologies SAS . (2015). *Shiva Engine*. Retrieved from http://www.shivaengine.com/

Sutter, J. D. (2010, Octubre 21). *CNN México*. Retrieved from http://mexico.cnn.com/tecnologia/2010/10/21/que-es-y-que-debo-esperar-de-un-telefono-inteligente

The Apache Software Foundation. (2014, Mayo). *Apache Flex*. Retrieved from http://flex.apache.org/

The Apache Software Foundation. (2015). *Apache Cordova*. Retrieved from https://cordova.apache.org/

Valve. (2015, Enero 19). *Valve Developer Community*. Retrieved from https://developer.valvesoftware.com/wiki/Shader

www.bitsenimagen.com. (n.d.). Retrieved from www.bitsenimagen.com/siete-de-cada-10-celulares-sera-smartphone-en-2015

Ying, J. (2010, Mayo 5). *Dropbox Blog*. Retrieved from https://blogs.dropbox.com/dropbox/2010/05/im-just-going-to-leave-these-here/

ANEXO 1 CONTENIDO MATEMÁTICO DE LA APLICACIÓN

Tema Vectores

Sección introducción

1. De acuerdo con la Real Academia Española se puede definir a la Geometría Analítica como: *estudio de figuras que utiliza un sistema de coordenadas y los métodos del análisis matemático*.

2. La geometría analítica es un medio que permite modelar fenómenos físicos para poder entender con mayor precisión los fenómenos tales como trayectorias que siguen los planetas, cometas, lanzamiento de un objeto, etc.

3. Se llama cantidad escalar a aquella que solamente posee magnitud.

4. Una cantidad es vectorial si posee, magnitud, dirección y sentido.

5. Se llama segmento dirigido a un segmento de recta en el que se ha asignado un punto origen y un punto extremo.

6. En virtud de que un vector tiene magnitud, dirección y sentido puede representarse geométricamente con el segmento dirigido que convenga.

7. Para representar analíticamente a un vector se usa una terna ordenada de números reales $\bar{a} = (a_1,\ a_2,\ a_3)$

8. Sea el vector $\bar{v}$ = (v_1, v_2, v_3) Si dicho vector se representa por el segmento dirigido $\overline{\mathbf{AB}}$ su punto origen es **A** (a_1, a_2, a_3) y su punto extremo **B** (b_1, b_2, b_3); las componentes escalares del vector son: $v_1 = b_1 - a_1$; $v_2 = b_2 - a_2$; $v_3 = b_3 - a_3$.

9. Se llama vector nulo o vector cero a $\bar{0}$ = (0, 0, 0). El vector nulo tiene magnitud nula y no tiene definida ni su dirección ni su sentido.

10. Se llama vector de posición de un punto P (p_1, p_2, p_3) aquel que tiene su punto de origen de coordenadas y su punto extremo en el punto P.

11. La magnitud o módulo de un vector es el tamaño de cualquier segmento dirigido que lo representa. La determinación del módulo en forma analítica se obtiene por la expresión:

$$|\bar{a}| = \sqrt{a_1^2, a_2^2, a_3^2}$$

12. Un vector es unitario si su módulo es igual a la unidad.

13. Los ángulos directores de un vector son los que forma un segmento dirigido que representa a dicho vector con los ejes coordenados.

14. Los cosenos directores de un vector son los cosenos de sus ángulos directores.

$$\cos\alpha = \frac{a_1}{|\bar{a}|};\ \cos\beta = \frac{a_2}{|\bar{a}|};\ \cos\gamma = \frac{a_3}{|\bar{a}|};$$

15. (Teorema) Sea el vector no nulo $\bar{a}$ = (a_1, a_2, a_3). Entonces sus cosenos directores cumplen con $\sqrt{cos^2\alpha + cos^2\beta + cos^2\gamma} = 1$.

16. (Corolario) Un vector unitario tiene por componentes escalares sus cosenos directores.

17. Dos vectores $\bar{a}$ = (a_1, a_2, a_3) y $\bar{b}$ = (b_1, b_2, b_3) son iguales si y sólo si $b_1 = a_1$, $b_2 = a_2$, $b_3 = a_3$.

18. La adición de dos vectores es la operación entre $\bar{a}$ = (a_1, a_2, a_3) y $\bar{b}$ = (b_1, b_2, b_3) de tal manera que $\bar{a} + \bar{b}$ = ($a_1 + b_1$, $a_2 + b_2$, $a_3 + b_3$).

19. Sean el vector $\bar{a}$ = (a_1, a_2, a_3) y el escalar $\lambda \in \mathbb{R}$; se llama multiplicación de un vector por un escalar o simplemente multiplicación por un escalar a la operación: $\lambda\bar{a}$ = (λa_1, λa_2, λa_3)

20. Sean los vectores $\bar{a}$ = (a_1, a_2, a_3) $y\,\bar{b}$ = (b_1, b_2, b_3); se llama producto escalar, producto interno o producto punto de $\bar{a}\ y\ \bar{b}$ a $\bar{a} \cdot \bar{b} = a_1b_1 + a_2b_2 + a_3b_3$.

21. Sean los vectores $\bar{a}$ = (a_1, a_2, a_3) $y\,\bar{b}$ = (b_1, b_2, b_3); entonces $\bar{a} \cdot \bar{b} = |\bar{a}| \cdot |\bar{b}|$ cos Θ, donde Θ es el ángulo que forman dos segmentos dirigidos.

22. Dos vectores $\bar{a}\ y\ \bar{b}$ son perpendiculares si y sólo si $\bar{a} \cdot \bar{b} = 0$

23. Dos vectores no nulos $\bar{a}\ y\ \bar{b}$ son paralelos si $\bar{a} = \lambda\bar{b};\ \lambda \in \Re,\ \lambda \neq 0$

24. Se llama componente vectorial de un vector $\bar{a}$ en la dirección del vector $\bar{b}$ a la proyección perpendicular $\bar{a}$ sobre la dirección de $\bar{b}$.

25. Sean los vectores $\bar{a}$ = (a_1, a_2, a_3) y $\bar{b}$ = (b_1, b_2, b_3); se llama producto vectorial o producto cruz a $\bar{a} \times \bar{b}$ = $(a_2b_3 - a_3b_2)\,i - (a_1b_3 - a_3b_1)\,j + (a_1b_2 - a_2b_1)\,k$.

26. Sean los vectores $\bar{a}, \bar{b}\ y\ \bar{c}$; se llama producto mixto, o producto triple escalar, a $[\bar{a}\ \bar{b}\ \bar{c}] = \bar{a} \times \bar{b} \bullet \bar{c}$.

Sección ejercicios 1

1. Dada la igualdad $\bar{z} - \bar{r} + \bar{k} = \bar{z} + \bar{r} - \bar{k}$

   $\bar{k} = -\bar{z}$
   $\bar{k} = \bar{z}$
   $\bar{r} = \bar{k}$
   $\bar{r} = -\bar{k}$

2. Si $(1, 3, a) = (2b, 3, 4) + (3, c, b)$, la suma de $a + b - c$ es

   -2
   4
   2
   -4

3. El vector que satisface la igualdad $(1,2,-3)+\bar{z}=-\bar{z}+(1,4,-3)$ es

$\bar{z}=(0,-1,0)$
$\bar{z}=(1,0,1)$
$\bar{z}=(0,1,0)$
$\bar{z}=(1,-1,0)$

4. Un vector que es simultaneamente perpendicular a $\bar{r}=(2,4,6)$ y $\bar{c}=(2,0,2)$ es

$(-1,1,-1)$
$(-1,1,1)$
$(1,-1,-1)$
$(1,1,-1)$

5. Dados los vectores perpendiculares entre si $\bar{c}$ y $\bar{d}$ entonces

$|\bar{c}-\bar{d}|=|\bar{c}|^2-|\bar{d}|^2$
$|\bar{c}-\bar{d}|^2=|\bar{c}|^2+|\bar{d}|^2$
$|\bar{c}-\bar{d}|=|\bar{c}|-|\bar{d}|$
$|\bar{c}-\bar{d}|=|\bar{c}|+|\bar{d}|$

6. Dados los vectores $\bar{e}$, $\bar{g}$ y $\bar{r}$ con la igualdad de $\bar{e}=\bar{g}+\bar{r}$ entonces:

$\bar{r}=-\bar{e}-\bar{g}$
$\bar{r}=\bar{e}-\bar{g}$
$\bar{r}=-\bar{e}+\bar{g}$
$\bar{r}=\bar{e}+\bar{g}$

7. Sean los vectores $\bar{u}, \bar{v} y \bar{r}$, si $\bar{r} = \bar{u} + \bar{v}$ entonces

$\bar{u} = \bar{r} - \bar{v}$
$\bar{u} = \bar{r} + \bar{v}$
$\bar{u} = -\bar{r} + \bar{v}$
$\bar{u} = -\bar{r} - \bar{v}$

8. Un vector ortogonal tanto al vector $\bar{b} = (2,5,-10)$ como al vector $\bar{c} = (12,3,-6)$ es

$(0,2,1)$
$(0,2,-1)$
$(-5,0,1)$
$(1,-2,1)$

9. El vector $\bar{x}$ que satisface a la ecuación $(3,-3,4) + \bar{x} = (-1,11,11) - \bar{x}$ es

$\bar{x} = (-1,3,2)$
$\bar{x} = (-7,6,4)$
$\bar{x} = (-2,6,4)$
$\bar{x} = (1,-3,-2)$

10. **Si $(6, c, b) + (3, 1, a) = (b, 5, 2)$ entonces la suma de $a + b + c$ es**

6
2
12
-2

11. Si el vector $\bar{a}-\bar{b}+\bar{c}$ es igual al vector $\bar{a}+\bar{b}-\bar{c}$ entonces

$\bar{b}=\bar{c}$
$\bar{c}=-\bar{b}$
$\bar{a}=-\bar{b}$
$\bar{a}=\bar{c}$

12. Dos vectores son ortogonales entre sí, si y sólo si

Su producto escalar es cero
Su producto vectarial es cero
Su producto escalar es uno
Su producto vectarial es uno

13. El conjunto de todos los vectores perpendiculares al vector $\bar{r}=(-1,2)$ es

$(\frac{1}{4}a,-\frac{1}{2}a)$

$(1,2)$

$(-2,1)$

$(-\frac{1}{4}a,-\frac{1}{2}a)$

14. Un vector $\bar{r}$ simultaneamente ortogonal a los vectores $\bar{u}=(1,0,2)$ y $\bar{v}=(2,1,0)$ es

$\bar{r}=(-2,4,1)$
$\bar{r}=(-2,4,-1)$
$\bar{r}=(2,-4,-1)$
$\bar{r}=(2,-4,1)$

Sección ejercicios 2

1. La componente vectorial del vector $\bar{u} = (1,3,5)$ sobre el vector unitario **j** es

   $(0,3,0)$
   $(1,0,0)$
   $(1,1,5)$
   $(1,3,1)$

2. La componente escalar del vector $\bar{u} = (3,2,5)$ sobre el vector unitario **i** es

   3
   -3
   2
   5

3. La componente vectorial del vector $\bar{u} = (-2,-3,-5)$ sobre el vector unitario **-i** es

   $(2,0,0)$
   $(-2,0,0)$
   $(2,1,1)$
   $(1,3,1)$

4. El ángulo que forman el vector $\bar{a} = (0,2)$ y el vector $\bar{b} = (0,1)$ es

   $\cos\alpha = 1 \Rightarrow \alpha = 0$
   $\cos\alpha = .66 \Rightarrow \alpha = 48.7$
   $\cos\alpha = -.66 \Rightarrow \alpha = 131.29$
   $\cos\alpha = 0 \Rightarrow \alpha = 1$

5. Dado un vector no nulo, la suma de los cuadrados de sus cosenos directores es

1
0
3
−1

6. Qué angulo forman los vectores $\bar{a} = (2, -1, 4)$ y $\bar{b} = (-1, 0, 3)$

$-46.36°$
$45.0°$
$46.63°$
$90°$

7. La componente vectorial del vector $\bar{v} = (1,2,3)$ sobre el vector $\bar{u} = (1,2,1)$ es

$(\frac{4}{3}, \frac{8}{3}, \frac{4}{3})$

$(\frac{4}{\sqrt{14}}, \frac{8}{\sqrt{14}}, \frac{12}{\sqrt{14}})$

$(\frac{4}{\sqrt{14}}, \frac{8}{\sqrt{14}}, \frac{4}{\sqrt{14}})$

$(\frac{4}{3}, \frac{8}{3}, \frac{12}{3})$

8. Si $\bar{r} = (0,1,1)$ es la componente vectorial del vector $\bar{u} = (0,a,0)$ sobre el vector $\bar{v} = (0,1,1)$ entonces el valor de "a" es

3

$\frac{1}{3}$

-3

$-\frac{1}{3}$

9. El producto vectorial de **j** x **k** es

i
j
-j
k

10. El producto vectorial entre dos vectores paralelos entre sí es

el vector cero
0
1
i

11. El área del paralelogramo que tiene como lados no paralelos a los vectores $\bar{a} = (1,1,0)$ y $\bar{b} = (-1,0,2)$ es

3
5
$\sqrt{5}$
$\sqrt{3}$

12. Si los vectores $\bar{r}$ y $\bar{s}$ son paralelos entre sí, entonces el producto $[\bar{r} \bullet \bar{s} \times \bar{u}]$ es

indefinido
positivo
cero
negativo

13. Cuatro de los vértices de un paralelogramo son los puntos A(0,2,1), B(2,3,2), C(2,1,1) y D(0,0,0), el área de dicho paralelogramo es

$\sqrt{19}$
19
7
$\sqrt{7}$

14. Sean los vectores $\bar{r}, \bar{s}$ y $\bar{t}$ que definen tres aristas de un paralelepípedo cuyo volumen es V. El volumen de un segundo paralelepípedo cuyas aristas son $3\bar{r}, 3\bar{s}$ y $3\bar{t}$ es

27V
9V
3V
V

Tema Superficies

Sección introducción

1. Se llama superficie al lugar geométrico de todos los puntos que tienen representación gráfica en el espacio de tres dimensiones y cuya relación matemática representativa cuenta con una sola ecuación del tipo:

$$F(x, y, z) = 0$$

Es conveniente hacer notar que no todas las ecuaciones de este tipo representan una superficie.

2. Las superficies que tienen su representación analítica del tipo:

$$Ax^2 + Bxy + Cy^2 + Dxz + Ez^2 + Fyz + Gx + Hy + Iz + J = 0$$

Se les llama superficies cuádricas o cuadráticas.

Las cuádricas se clasifican a la vez en:

- Esferas
- Elipsoides
- Hiperboloides de uno y dos mantos
- Paraboloides circulares o de revolución
- Paraboloides elípticos
- Paraboloides hiperbólicos

3. Clasificación de las superficies cuádricas

Cuádricas con centro ($Ax^2 + By^2 + Cz^2 = R$)

R	A, B, C	Lugar Geométrico
POSITIVO	Todos positivos	Elipsoide
	Todos negativos	Ningún lugar geométrico
	Dos positivos y uno negativo	Hiperboloide de una rama
	Uno positivo y dos negativos	Hiperboloide de dos ramas
	Uno nulo y dos positivos	Cilindro elíptico recto, sí los dos son iguales es circular
	Uno nulo y dos negativos	Ningún lugar geométrico
	Uno nulo y los otros de signos diferentes	Cilindro elíptico recto
	Dos nulos y uno positivo	Dos planos paralelos

	Dos nulos y uno negativo	Ningún lugar geométrico
NULO	Todos del mismo signo	Un solo punto que es el origen
	Dos positivos y uno negativo	Cono recto
	Uno nulo y los otros dos del mismo signo	Un eje coordenado
	Uno nulo y los otros de signos diferentes	Dos planos que se cortan
	Dos nulos	Un plano coordenado

NOTA. Si *R* es negativo se multiplica por menos uno toda la ecuación

Cuádricas sin centro ($Ax^2 + By^2 = Rz$)

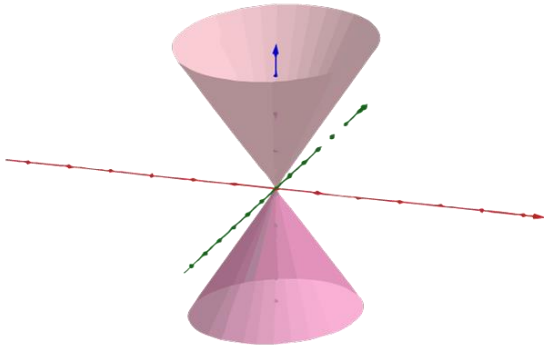
R	A, B	Lugar Geométrico
POSITIVO	Del mismo signo	Paraboloide elíptico
	Diferentes signos	Paraboloide hiperbólico
	Uno nulo	Cilindro parabólico recto
NULO	Del mismo signo	Un eje coordenado
	Signos diferentes	Dos planos que se cortan
	Uno nulo	Un plano coordenado

NOTA. Si R es negativo se multiplica por menos uno toda la ecuación

4. Cono

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z^2}{c^2}$$

Para a>0, b>0 y c>0

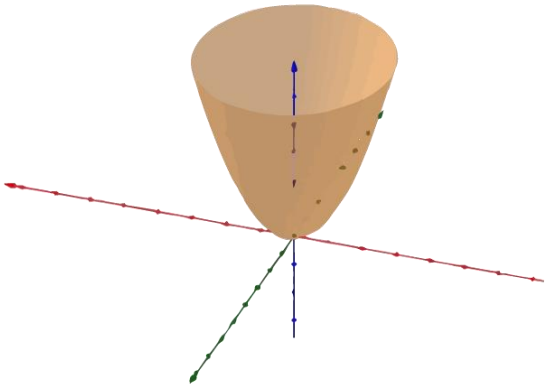


Plano de coordenadas	Sección Transversal
xy	punto
xz	rectas que se cortan
yz	rectas que se cortan

5. Paraboloide elíptico

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = zc$$

Para a>0 y b>0

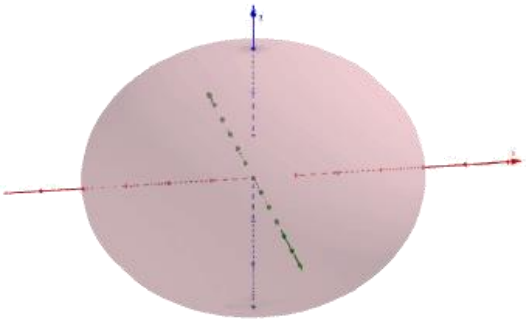


Plano de coordenadas	Sección Transversal
xy	punto
xz	rectas que se cortan
yz	rectas que se cortan

6. Elipsoide

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

Para a>0, b>0 y c>0 la ecuación representa un elipsoide.
Si a=b=c la elipsoide es una esfera.

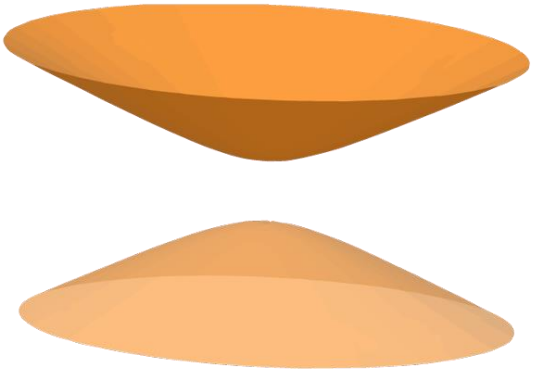


Plano de coordenadas	Sección Transversal
xy	elipse
xz	elipse
yz	elipse

7. Hiperboloide de dos mantos

$$-\frac{x^2}{a} - \frac{y^2}{b} + \frac{z^2}{c} = 1$$

Para a>0, b>0 y c>0

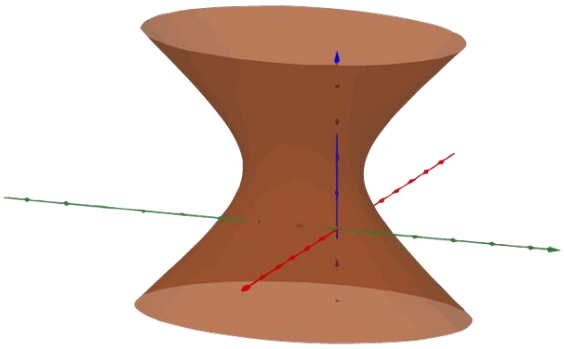


Plano de coordenadas	Sección Transversal
xy	hipérbola
xz	ninguna
yz	hipérbola

8. Hiperboloide de un manto

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

Para a>0, b>0 y c>0

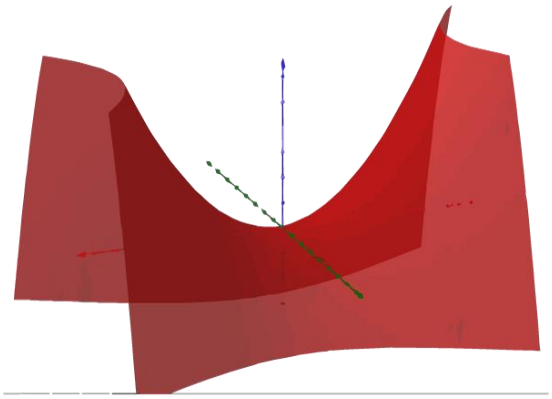


Plano de coordenadas	Sección Transversal
xy	elipse
xz	hipérbola
yz	hipérbola

9. Paraboloide hiperbólico

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = \frac{z}{c}$$

Para a>0 y b>0



Plano de coordenadas	Sección Transversal
xy	rectas que se cortan
xz	Parábola
yz	Parábola

Sección ejercicios 1

1. La ecuación $(x-2)^2+(y-2)^2+(z-2)^2=0$ representa

   un punto
   una esfera
   un elipsoide
   un hiperboloide

2. La ecuación de una esfera de radio 3 y centro en (-6, 4,-4) es

   $(x+6)^2+(y-4)^2+(z+4)^2=9$
   $(x-6)^2+(y+4)^2+(z-4)^2=9$
   $(x-6)^2+(y+4)^2+(z-4)^2=3$
   $(x+6)^2+(y-4)^2+(z+4)^2=9$

3. La ecuación que representa a un paraboloide hiperbólico con vértice en $(2,4,4)$ y eje paralelo al eje Z es

   $$\frac{(x-2)^2}{3}-\frac{(y-4)^2}{2}-\frac{(z-4)^2}{3}=0$$
   $$\frac{(x+2)^2}{3}-\frac{(y+4)^2}{2}-\frac{(z+4)^2}{3}=0$$
   $$\frac{(x-2)^2}{3}+\frac{(y-4)^2}{2}-\frac{(z-4)^2}{3}=0$$
   $$\frac{(x+2)^2}{3}-\frac{(y+4)^2}{2}+\frac{(z+4)^2}{3}=0$$

4. La ecuación $\frac{(x-5)^2}{6} - \frac{(y-7)^2}{9} - \frac{(z-4)^0}{5} = 0$ representa

una elipse
una circunferencia
una parábola
una hipérbola

5. Un ejemplo de superficie reglada es un

hiperboloide de revolución
elipsoide
paraboloide de revolución
paraboloide elíptico

6. La traza de la superficie $6y^2 - 12z^2 = 2x + 2$ en el plano x= 2 es

una hipérbola
un punto
un par de rectas paralelas
un par de rectas que se cruzan

7. La ecuación de la superficie que se obtiene al rotal la parábola de ecuación $y - x^2 = 0$ alrededor del eje Y

$x^2 + z^2 = y$
$x^2 + y^2 = z$
$-(x^2 + z^2) = y$
$-(x^2 + y^2) = z$

8. La traza entre el paraboloide hiperbólico de ecuación $\frac{x^2}{4}+\frac{y^2}{9}=z$ con el plano y = 1 es una

elipse
parabloa
circunferencia
hipérbola

9. Las trazas entre la superficie de de ecuación $25z^2-9x^2=4y$ con el plano XZ son un par de rectas con ecuaciones

$5z+3x=0$ y $5z-3x=0$
$3x+5z=0$ y $3x-5z=0$
$25z+9x=0$ y $25z-9x=0$
$9x+25z=0$ y $9x-25z=0$

10. La traza entre la superficie de ecuación $6z^2+y^2+3z+y+12x=-7$ con el plano $y=1$, es una

parábola
hipérbola
recta
circunferencia

11. La traza entre la superficie de ecuación $4x^2+5z^2-20y+8x-20z=9$ con el plano XZ, es una

elipse
hipérbola
parábola
circunferencia

Sección ejercicios 2

1. La ecuación $(x-6)^2-(z-4)^2-5=0$ representa a un cilindro

   hiperbólico
   circular
   elíptico
   parabólico

2. Las rectas contenidas en el cilindro circular $(x-6)^2+(z-7)^2=4$ son paralelas al

   eje Y
   plano XZ
   eje X
   plano y=4

3. Al cortar el cilindro de ecuación

$$\frac{(y-1)^2}{9}+\frac{(z-6)^2}{4}=7$$

   con planos paralelos al plano YZ se obtienen

   elipses
   circunferencias
   parábolas
   hipérbolas

4. El cilindro parabólico $x-1=y^2$ es simétrico con respecto a

   el eje X
   el eje Y
   el eje Z
   el origen

5. La ecuación $-x^2 - y^2 + z^2 = 0$ representa a un cono

circular o de revolución, con vértice en el origen y eje coincidente con el eje Z

circular o de revolución, con vértice en V(1,1,-1) y eje coincidente con el eje Z

elíptico, con vértice en el origen y eje coincidente con el eje Z

elíptico, con vértice en V(1,1,-1) y eje coincidente con el eje Z

6. La ecuación $(x-4)^2 + (y-3)^2 - (z-2)^2 = 0$ representa a un cono

circular o de revolución, con vértice en $V(4,3,2)$ y eje paralelo al eje Z

circular o de revolución, con vértice en $V(4,3,2)$ y eje paralelo al eje X

circular o de revolución, con vértice en $V(-4,-3,-2)$ y eje paralelo al eje Y

circular o de revolución, con vértice en $V(-4,-3,-2)$ y eje paralelo al eje Z

7. La ecuación $\frac{x^2}{2} - \frac{y^2}{3} - \frac{z^2}{4} = 0$ representa a un cono

elíptico, con vértice en el origen y eje coincidente con el eje X

elíptico, con vértice en el origen y eje coincidente con el eje Z

circular o de revolución, con vértice en el origen y eje coincidente con el eje X

circular o de revolución, con vértice en el origen y eje coincidente con el eje Z

8. La ecuación $(x+4)^2 - (y+3)^2 + (z+5)^2 = 0$ representa a un cono

circular o de revolución, con vértice en V(-4,-3,-5) y eje paralelo al eje Y

circular o de revolución, con vértice en el origen y eje paralelo al eje Y

elíptico, con vértice en el origen y eje paralelo al eje Y

elíptico, con vértice en V(-4,-3,-5) y eje paralelo al eje Y

9. La ecuación $(x+15)^2+\frac{(y-4)^2}{3}-\frac{(z-6)^2}{7}=0$

representa a un cono elíptico con

vértice en V(-15, 4, 6) y eje paralelo al eje Z

vértice en el origen y eje paralelo al eje Z

vértice en el origen y eje paralelo al eje X

vértice en V(-15, 4, 6) y eje paralelo al eje X

10. Una ecuación del cono circular con centro en el punto $C(1,-3,4)$ y eje paralelo al eje Z es

$-(x-1)^2-(y+3)^2+(z-4)^2$

$-(x+1)^2-(y-3)^2+(z+4)^2$

$(x-1)^2+(y+3)^2-(z-4)^2$

$(x+1)^2+(y-3)^2-(z+4)^2$

11. Unas ecuaciones de rectas generatrices de una superficie cónica cuyo vértice es el punto V(1, 2, 3) son:

$\frac{x-1}{\alpha}=\frac{y-2}{\beta}=\frac{z-3}{\gamma};\alpha\neq 0,\beta\neq o,\gamma\neq 0$

x-1 = y-2 = z-3

$x-\alpha=\frac{y-\beta}{2}=\frac{z-\gamma}{3}$

$\frac{x}{\alpha-1}=\frac{y}{\beta-2}=\frac{z}{\gamma-3}$

ANEXO 2 MANUAL PARA AÑADIR CONTENIDO A LA APLICACIÓN

El presente manual tiene como finalidad el asistir a todo aquella persona que quiera agregar contenidos matemáticos a la aplicación. Se considera que ya se cuenta con las imágenes que se desean añadir, ya sea para los reactivos, para la sección de introducción o bien la imagen que representa al tema. Es requisito contar con: el código fuente, un software para modificar la Base de Datos y Corona SDK.

Para descargar el código fuente, se puede realizar desde el siguiente enlace: https://bitbucket.org/jbonilla/xook.

Dado que la aplicación usa el sistema de gestión de datos SQLite se requiere usar un software para poder modificar la Base de Datos, se recomienda para dicho propósito ocupar SQLite Manager o DB Browser for SQLite(SqliteBrowser).

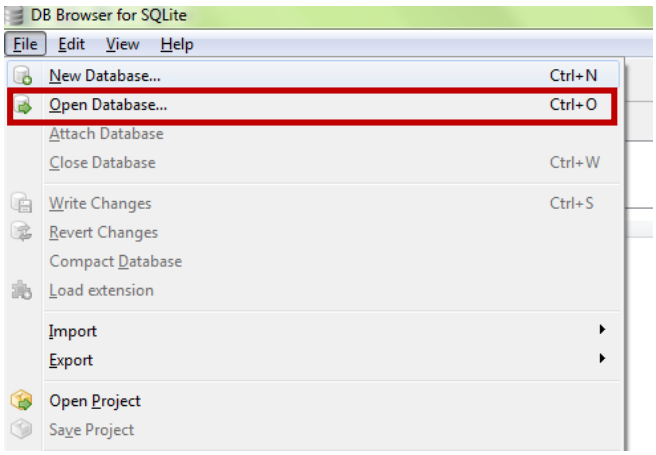
El Kit de Desarrollo de Software de Corona (Corona SDK) se puede descargar desde su página, sin costo alguno. Cabe destacar que para poder realizar la descarga se necesita contar con registro.

Abrir Base de Datos

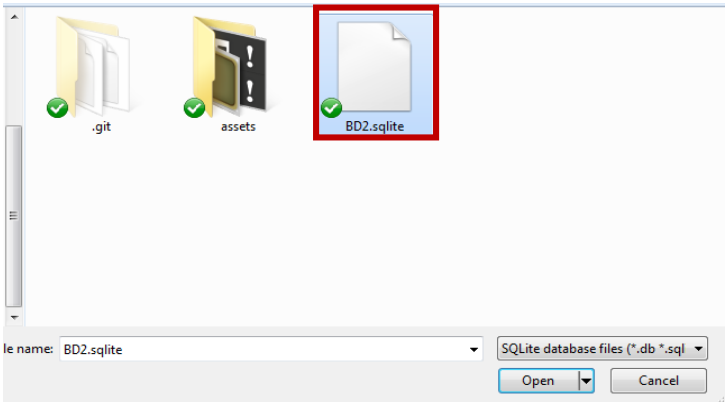
Para poder añadir contenido a la aplicación se requiere abrir la BD. Los siguientes pasos ilustran como realizarlo.

- Se abre el programa SqliteBrowser

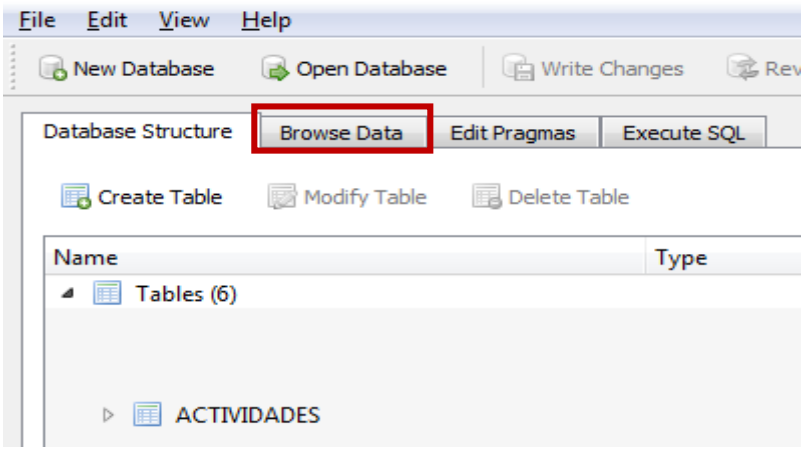
- En SqliteBrowser se selecciona File -> Open Database



- En el cuadro que se despliega se elige el archivo BD2



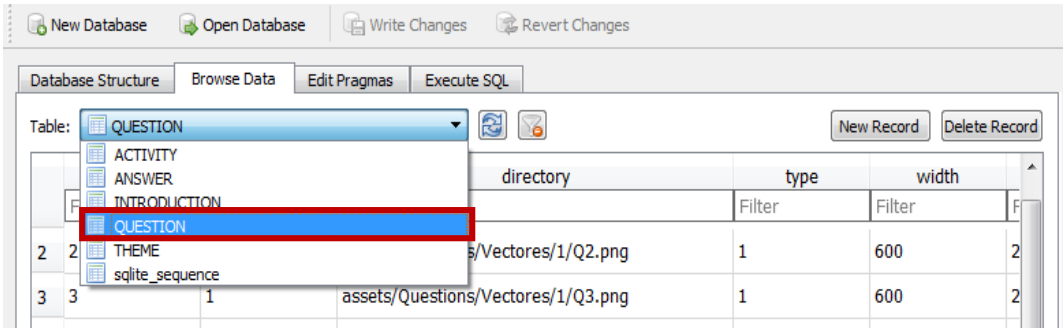
- Se procede a cambiar de pestaña a Browse Data



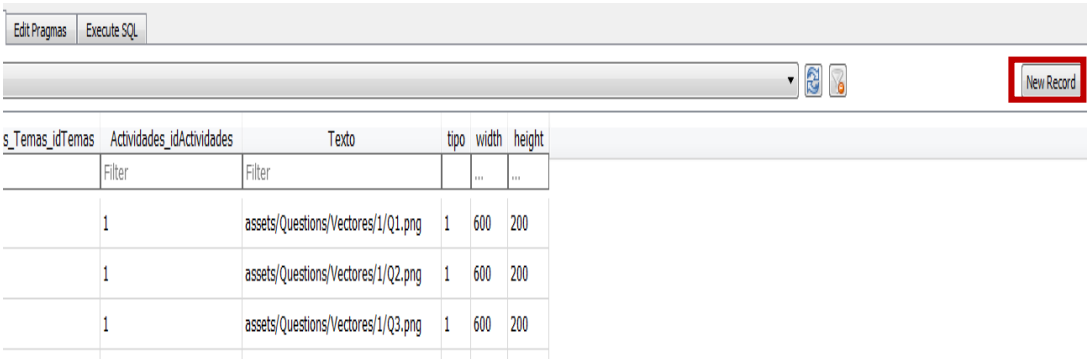
Agregar un reactivo

Para adicionar un reactivo se requiere contar con la imagen de la pregunta y sus 4 posibles respuestas.

- La imagen correspondiente a la pregunta se colocará en la carpeta assets\Questions\[tema]
- Las imágenes de las respuestas deberán ir en el directorio assets\Answers\[tema]
- Se abre la BD
- Se elige la tabla QUESTION



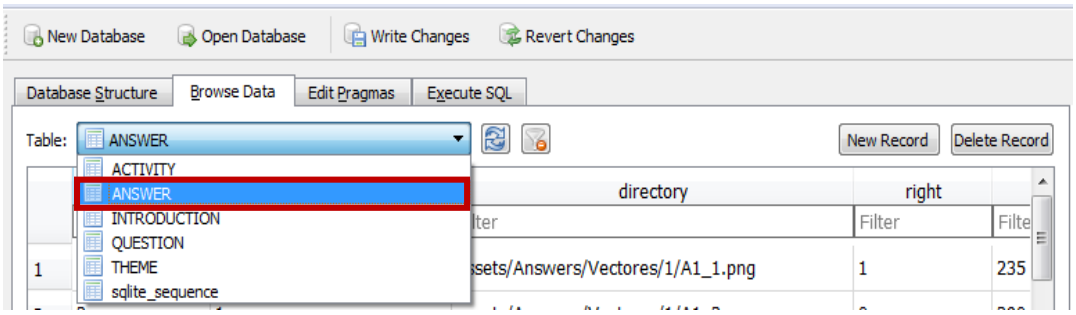
- Procedemos a dar clic en New Record



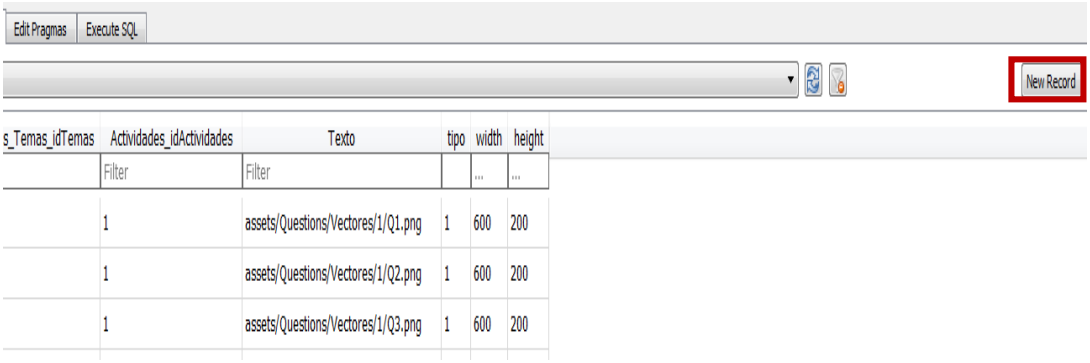
- En el nuevo registro se deben de introducir los datos correspondientes, donde:
    - ACTIVITY_idActivity: Corresponde al id de la actividad a la que se va a añadir la pregunta.
    - directory: Ruta donde se colocó la imagen correspondiente
    - type: Representa que tipo de pregunta es (1 o 2) De momento sólo se ocupó el tipo de pregunta 1.

- width: Ancho de la imagen. Máximo 600.
- height: Altura de la imagen.

- Se elige la tabla ANSWER



- Procedemos a dar clic en New Record



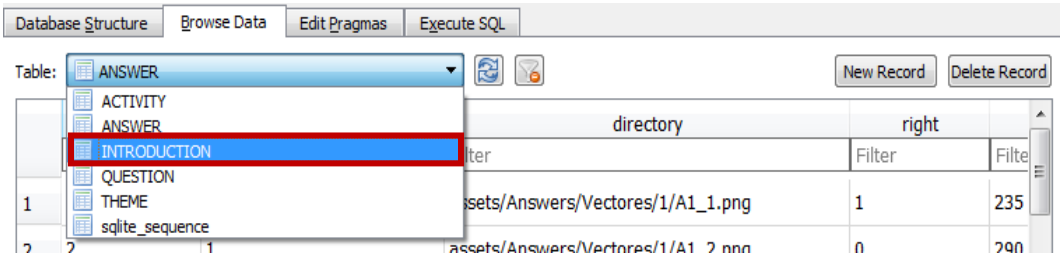
- En el nuevo registro se deben de introducir los datos correspondientes, donde:
  - QUESTION_idQuestion: Indica a que pregunta corresponde la respuesta, es decir si se está agregando la respuesta de la pregunta 1 el valor es 1, si es una respuesta de la pregunta 19 el valor es 19.
  - directory: Ruta donde se colocó la imagen correspondiente
  - right: Se usa para señalar si esta respuesta debe ser aceptada como correcta, 1 para tal caso, 0 de lo contrario.
  - width: Ancho de la imagen. Máximo 600.
  - height: Altura de la imagen.

- Por cada una de las 4 respuestas se repiten los 2 últimos pasos.

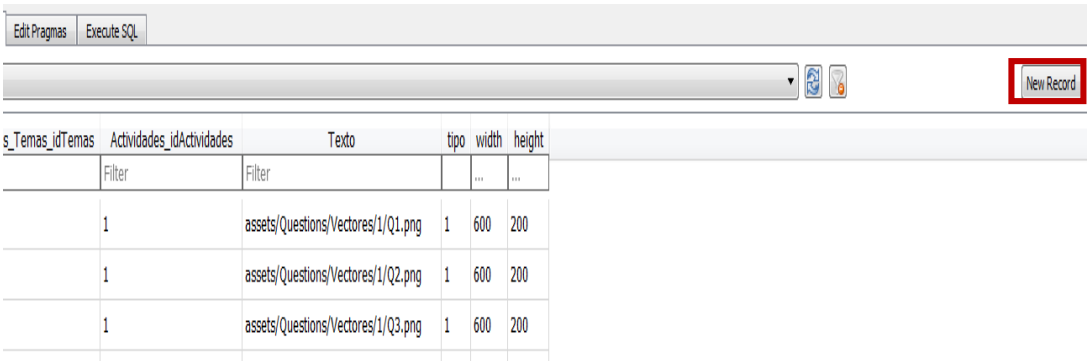
Con esto se ha agregado un reactivo, para comprobar su funcionalidad se puede utilizar el simulador de Corona (Corona Simulator) o bien se puede compilar directamente.

Agregar un elemento a la sección Introducción

- La imagen correspondiente se colocará en la carpeta assets\Intro\[tema]
- Se abre la BD
- Se elige la tabla INTRODUCTION



- Procedemos a dar clic en New Record

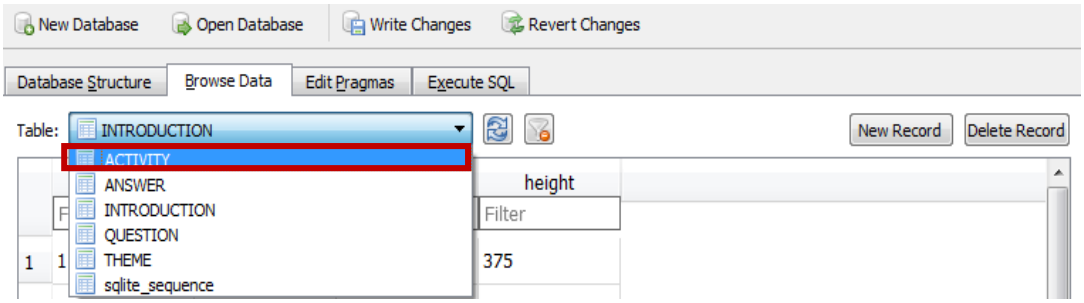


- En el nuevo registro se deben de introducir los datos correspondientes, donde:
    - ACTIVITY_idActivity: Corresponde al id de la actividad a la que se va a añadir la imagen.
    - directory: Ruta donde se colocó la imagen correspondiente
    - height: Altura de la imagen

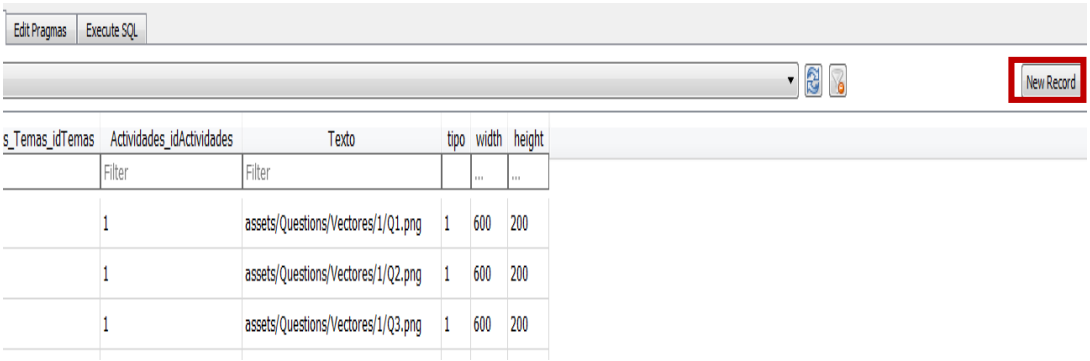
Para comprobar su funcionalidad se puede utilizar el simulador de Corona (Corona Simulator) o bien se puede compilar directamente.

Agregar un bloque de ejercicios

- Se abre la BD
- Se elige la tabla ACTIVITY



- Procedemos a dar clic en New Record

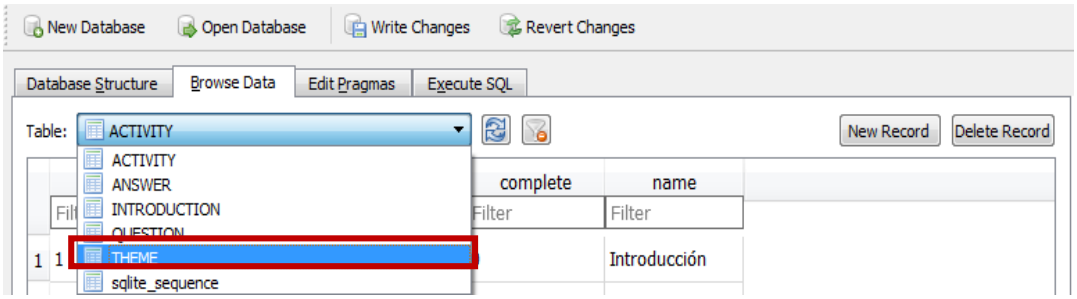


- En el nuevo registro se deben de introducir los datos correspondientes, donde:
    - Theme_idTheme: Identificador del tema.
    - Type: Se debe de ingresar el número dos para indicar que se trata de ejercicios. En el caso de introducción es uno.
    - complete: Se deja el valor por default, cero. Señala si una actividad, en el caso de ejercicios, fue completada. El programa lo cambia automáticamente a uno cuando se completa.
    - name: Nombre de la sección. Es el nombre que se le mostrará al usuario para que identifique que tipo de actividad es.
- Se procede a agregar los reactivos (mínimo 10). Por cada reactivo se siguen los pasos indicados en Agregar un reactivo

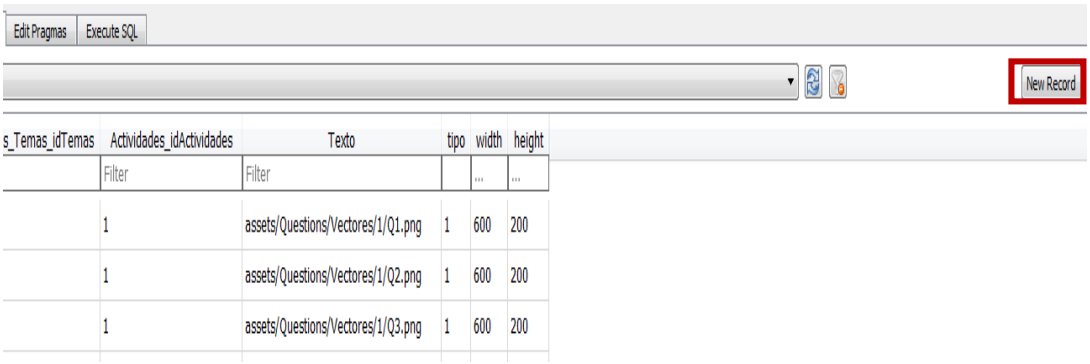
Agregar un bloque de introducción

Agregar un tema completo

- Se abre la BD
- Se elige la tabla THEME



- Procedemos a dar clic en New Record



- En el nuevo registro se deben de introducir los datos correspondientes, donde:
    - name: Nombre del tema.
    - image: Ruta de la imagen que corresponde al tema.
    - level: Número que representa el nivel al que se quiere agregar el tema. Se puede asignar al mismo nivel que otro tema o uno superior al nivel máximo existente.

- complete: Se deja el valor por default, cero. Señala si una tema, todos sus bloques de ejercicios, fue completado. El programa cambia el valor automáticamente a uno cuando se completa.

- active: Marca si se puede ver el tema que se está agregando.

- Se procede a agregar un bloque de; introducción y ejercicios.

ANEXO 3 CÓDIGO DE LA APLICACIÓN

build

```
settings = {

    orientation = {
        default = "portrait",
        supported = { "portrait", "portraitUpsideDown"}
    },

    android =
    {

        --Version 1.0
        versionCode = "10",
        allowAppsReadOnlyAccessToFiles = false,
        usesPermissions = {},

    },
}
```

config.lua

```
---------------------------------------------------------------------------
-- config.lua
-- Permite Auto ajuste para distintos tipos de dispositivos.
---------------------------------------------------------------------------

local aspectRatio = display.pixelHeight / display.pixelWidth
application = {
    launchPad = false,
    content = {
        width = aspectRatio > 1.5 and 800 or math.ceil( 1200 /
aspectRatio ),
        height = aspectRatio < 1.5 and 1200 or math.ceil( 800 *
aspectRatio ),
        scale = "letterBox",
        fps = 30,
    },
}
```

data.lua

```
---------------------------------------------------------------------------
-- data.lua
-- Permite almacenar variables disponibles para distintas escenas (En
lugar de usar variables globales)
---------------------------------------------------------------------------

local Datos = {}
```

```
Datos.tema = {}
Datos.introduccion = {}
Datos.actividad = {}
Datos.pregunta = {}
Datos.scenes =
{"sceneTemas","sceneActividades","sceneIntroduccion","sceneEjercicios","s
ceneGameover","sceneCompleta","sceneCreditos"}
return Datos
```

loadsave.lua

```
--creado por Rob Miracle (@MiracleMan)

local _ = {}
local json = require("json")
local DefaultLocation = system.DocumentsDirectory
local RealDefaultLocation = DefaultLocation
local ValidLocations = {
   [system.DocumentsDirectory] = true,
   [system.CachesDirectory] = true,
   [system.TemporaryDirectory] = true
}

function _.saveTable(t, filename, location)
    if location and (not ValidLocations[location]) then
     error("Attempted to save a table to an invalid location", 2)
    elseif not location then
      location = DefaultLocation
    end

    local path = system.pathForFile( filename, location)
    local file = io.open(path, "w")
    if file then
        local contents = json.encode(t)
        file:write( contents )
        io.close( file )
        return true
    else
        return false
    end
end

function _.loadTable(filename, location)
    if location and (not ValidLocations[location]) then
     error("Attempted to load a table from an invalid location", 2)
    elseif not location then
      location = DefaultLocation
    end
    local path = system.pathForFile( filename, location)
    local contents = ""
    local myTable = {}
    local file = io.open( path, "r" )
    if file then
        -- read all contents of file into a string
        local contents = file:read( "*a" )
```

```
        myTable = json.decode(contents);
        io.close( file )
        return myTable
    end
    return nil
end

function _.changeDefault(location)
    if location and (not location) then
        error("Attempted to change the default location to an invalid
location", 2)
    elseif not location then
        location = RealDefaultLocation
    end
    DefaultLocation = location
    return true
end

return _
```

main.lua

```
-----------------------------------------------------------------------------
-- main.lua
-- Funcion principal; inicia aplicación, carga base de datos
-----------------------------------------------------------------------------

-- Require widget and storyboard libraries
local widget = require( "widget" )
local storyboard = require( "storyboard" )

-- Invoke file with "global variables"
local Datos = require( "data" )

-- Require sqlite3
require "sqlite3"

-- Set the directory for the DB and create it
local path = system.pathForFile("BD2.sqlite", system.ResourceDirectory)
db = sqlite3.open(path)

-- Module for loading & saving data
local loadsave = require( "loadsave" )

-- Invoke file with "saved data"
local dP = loadsave.loadTable("permanentdata.json") --datosPermanentes

---If the file permanentdata is empty create the basic tables
if (dP == nil) then
    dP = {}
    dP.tema = {}
    dP.actividad = {}
end

-- Hide the status bar
```

```
display.setStatusBar( display.HiddenStatusBar )

-- Set the background to white
display.setDefault( "background", 255, 255, 255 )

--Variable for overlayOpciones
Datos.showing = false

--Load sounds
Datos.sound = true
Datos.gameComplete = audio.loadSound ("assets/sounds/steel.mp3")
Datos.gameOver = audio.loadSound ("assets/sounds/wooden.mp3")
Datos.gameStart = audio.loadSound("assets/sounds/xook.mp3")

--Load "THEME" table---------
local sql = "SELECT * FROM THEME"
local increment = 1

--Extract data from DB - temas
for row in db:nrows(sql)do

    Datos.tema[increment] = {}
    Datos.tema[increment].id = row.idTheme
    Datos.tema[increment].nombre = row.name
    Datos.tema[increment].imagen = row.image
    Datos.tema[increment].nivel = row.level


    if (dP.tema[increment] ~= nil) then
        Datos.tema[increment].activo = dP.tema[increment].activo
        Datos.tema[increment].completado = dP.tema[increment].completado
    else
        dP.tema[increment] = {}
        dP.tema[increment].activo = row.active
        dP.tema[increment].completado = row.complete
        Datos.tema[increment].activo = row.active
        Datos.tema[increment].completado = row.complete
    end

    increment = increment + 1
end


--Load "ACTIVIDADES" table---------
sql = "SELECT * FROM ACTIVITY"
increment = 1


--Extract data from DB - Actividades
for row in db:nrows(sql)do

    Datos.actividad[increment] = {}
    Datos.actividad[increment].id = row.idActivity
    Datos.actividad[increment].idTheme = row.THEME_idTheme
    Datos.actividad[increment].nombre = row.name
    Datos.actividad[increment].tipo = row.type
```

```
    if (dP.actividad[increment] == nil)then --Theme does exist but
activity doesn't
        dP.actividad[increment] = {}
        dP.actividad[increment].completado = row.completado
        Datos.actividad[increment].completado = row.completado
    else
        Datos.actividad[increment].completado =
dP.actividad[increment].completado
    end
    increment = increment + 1
end
loadsave.saveTable(dP, "permanentdata.json")

--Load "INTRODUCCION" table---------
sql = "SELECT * FROM INTRODUCTION"
increment = 1

--Extract data from DB - Introduccion
for row in db:nrows(sql)do

    Datos.introduccion[increment] = {}
    Datos.introduccion[increment].id = row.idIntroduction
    Datos.introduccion[increment].idActividad = row.ACTIVITY_idActivity
    Datos.introduccion[increment].texto = row.directory
    Datos.introduccion[increment].alto = row.height

    increment = increment + 1
end


--Load "QUESTION" table---------
sql = "SELECT * FROM QUESTION"
local idQuestion

--Extract data from DB - pregunta
for row in db:nrows(sql)do
    idQuestion = row.idQuestion

    Datos.pregunta[idQuestion] = {}
    Datos.pregunta[idQuestion].idActivity = row.ACTIVITY_idActivity
    Datos.pregunta[idQuestion].Texto = row.directory
    Datos.pregunta[idQuestion].Tipo = row.type
    Datos.pregunta[idQuestion].Ancho = row.width
    Datos.pregunta[idQuestion].Alto = row.height

end

--Load "ANSWER" table---------
sql = "SELECT * FROM ANSWER"
local idAnswer

--Extract data from DB - respuesta
for row in db:nrows(sql)do
    idQuestion = row.QUESTION_idQuestion

    if (Datos.pregunta[idQuestion] == nil) then
```

```
        error("Answer: "..row.idAnswer.."is linked to the non existent
question: "..idQuestion)
    else
        idAnswer = #Datos.pregunta[idQuestion] + 1
        Datos.pregunta[idQuestion][idAnswer] = {}
        Datos.pregunta[idQuestion][idAnswer].id = row.idAnswer
        Datos.pregunta[idQuestion][idAnswer].Opcion = row.directory
        Datos.pregunta[idQuestion][idAnswer].Correcta = row.right
        Datos.pregunta[idQuestion][idAnswer].Ancho = row.width
        Datos.pregunta[idQuestion][idAnswer].Alto = row.height

    end

end


if db and db:isopen() then
    db:close()
end
sql = nil
idQuestion = nil
idAnswer = nil
increment = nil

----- load sceneSplash.lua -----
storyboard.gotoScene("sceneSplash","slideLeft",1)



local function handleButtonAlert( event )
    if "clicked" == event.action then
        local i = event.index
        if i == 1 then --If yes
            storyboard.gotoScene(Datos.scenes[2],"slideRight",200)
        end
    end
end

local function onKeyEvent( event )
   local phase = event.phase
   local keyName = event.keyName
   local sceneName = storyboard.getCurrentSceneName()

    if ( "back" == keyName and phase == "up" ) then
        if(sceneName==Datos.scenes[1])then
            native.requestExit()
        elseif(storyboard.isOverlay) then
            storyboard.hideOverlay()
        elseif(sceneName==Datos.scenes[2] or sceneName ==
Datos.scenes[7]) then
            storyboard.gotoScene(Datos.scenes[1],"slideRight",200)
        elseif(sceneName==Datos.scenes[4]) then
            native.showAlert("Salir","El progreso se perderá, aún así
desea salir", {"Si", "No"},handleButtonAlert)
        else
            storyboard.gotoScene(Datos.scenes[2],"slideRight",200)
        end
```

```
        return true
    end
end

--add the key callback
Runtime:addEventListener( "key", onKeyEvent )
```

overlayOpciones.lua

```
-----------------------------------------------------------------------------
-- overlayOpciones.lua
-- Despliega los iconos de opciones (sonidos y creditos)
-----------------------------------------------------------------------------
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local widget = require( "widget" )

-- Invoke file with "global variables"
local Datos = require( "data" )


musik = function  ( event )
    Datos.sound = not Datos.sound
    Datos.showing = not Datos.showing
    storyboard.hideOverlay()
end

handleTapEventCredit = function  ( event )
    Datos.showing = not Datos.showing
    storyboard.gotoScene("sceneCreditos","zoomOutInFade",200)
end


-- Called when the scene's view does not exist:
function scene:createScene( event )

    local group = self.view

    local display_stage = display.getCurrentStage()

    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "subTitleBarBottom") then
            break
        end
        i = i + 1
    end

    local options =
    {
        width = 82.5,
        height = 82.5,
        numFrames = 4,
```

```
        sheetContentWidth = 165,  -- width of original 1x size of entire
sheet
        sheetContentHeight = 165  -- height of original 1x size of entire
sheet
    }
    local imageSheet = graphics.newImageSheet( "assets/adicion.png",
options )


    local musica
    if(Datos.sound) then
        musica = widget.newButton
        {
           id = "musica",
           sheet = imageSheet,
           defaultFrame = 1,
           overFrame = 3,
           onRelease  = musik
        }
    else
        musica = widget.newButton
        {
           id = "musica",
           sheet = imageSheet,
           defaultFrame = 3,
           overFrame = 1,
           onRelease  = musik
        }
    end
    musica.x = 642.5
    musica.y = display_stage[i].y + display_stage[i].height/2 +
musica.height/2

    local creditos = display.newImageRect( imageSheet, 2, 82.5, 82.5 )
    creditos.x = 730
    creditos.y = display_stage[i].y + display_stage[i].height/2 +
creditos.height/2
    creditos:addEventListener( "tap", handleTapEventCredit )

    group:insert( musica )
    group:insert( creditos )
end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )

return scene
```

overlayResult.lua

```
-----------------------------------------------------------------------
-- overlayResult.lua
-- Muestra la imagen de falso o correcto
-----------------------------------------------------------------------
```

```
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view
    local resultado = event.params.result

    if (resultado == 0) then
        dir = "assets/falso.png"
    else
        dir = "assets/correcto.png"
    end

    local imgResultado = display.newImageRect(dir,
display.contentWidth,display.contentHeight)
    imgResultado.x = display.contentCenterX
    imgResultado.y = display.contentCenterY
    group:insert(imgResultado)
end



-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )

return scene
```

overlayZoom.lua

```
-----------------------------------------------------------------------------
--
-- sceneZoom.lua
-- Amplifica la imagen que se le envíe
-----------------------------------------------------------------------------
--

local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local params


local function handleTouchEvent( event )
    storyboard.hideOverlay("zoomInOutFade", 100 )
end


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view
    params = event.params

    --Hide back and options Buttons
```

```
    local display_stage = display.getCurrentStage()
    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "salir") then
            display_stage[i].isVisible = false
        end
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
        end
        i = i + 1
    end

    local dir = params.filename
    local ancho = params.Ancho * 3
    local alto = params.Alto * 3

    if(ancho>display.contentWidth) then
        ancho = display.contentWidth
    end
    if(alto>display.contentHeight) then
        alto = display.contentHeight
    end

    --Display image zoomed
    local image = display.newImageRect(dir,ancho,alto)
    image.x = display.contentCenterX
    image.y = display.contentCenterY

    ancho = display.contentWidth
    alto = display.contentHeight
    local fondo = display.newRoundedRect( display.contentCenterX,
display.contentCenterY, ancho, alto, 7 )

    fondo:addEventListener( "touch", handleTouchEvent )
    group:insert(fondo)
    group:insert(image)
end

function scene:exitScene( event )
    local group = self.view

    --Show back and options Buttons
    local display_stage = display.getCurrentStage()
    local i=1

    if(params.Ejercicios) then --Show back Button
        while display_stage[i] ~= nil do
            if(display_stage[i].id == "salir") then
                display_stage[i].isVisible = true
            end
            i = i + 1
        end
    else --Show back and options Buttons
        while display_stage[i] ~= nil do
            if(display_stage[i].id == "salir") then
                display_stage[i].isVisible = true
            end
```

```
            if(display_stage[i].id == "opciones") then
                display_stage[i].isVisible = true
            end
            i = i + 1
        end
    end

end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )

-- "exitScene" event is dispatched before next scene's transition begins
scene:addEventListener( "exitScene", scene )

return scene
```

sceneActividades.lua

```
-----------------------------------------------------------------------------
-- sceneActvidades.lua
-- Despliega la pantalla de actividades
-----------------------------------------------------------------------------

local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local widget = require( "widget" )

-- Invoke file with "global variables"
local Datos = require( "data" )

--Variables inherited from sceneActividades
local idTemas = Datos.temaTarget

--Forward declaration
local scrollView
local F_Stage

--Variable
local themeBottom = {}
local myText = {}
local completada = {}
local navButton = {}


local function handleImageEvent ( event )
    local phase = event.phase
    local id = event.target.id
    local counter = event.target.counter
    local tipo = Datos.actividad[counter].tipo

    if ( phase == "moved" ) then
        local dy = math.abs( ( event.y - event.yStart ) )
        local dx = math.abs( ( event.x - event.xStart ) )
```

```
        -- If the touch on the button has moved more than 10 pixels,
        -- pass focus back to the scroll view so it can continue
scrolling
        if ( dy > 10 or dx > 10) then
            scrollView:takeFocus( event )
            return true
        end
    elseif ( phase == "ended" ) then
        Datos.actividadTarget = id

        if(tipo == 1) then
            storyboard.gotoScene("sceneIntroduccion","slideLeft",200)
        else
            storyboard.gotoScene("sceneEjercicios","slideLeft",200)
        end
    end
    return true
end

local function handleImageEventSelect( event )
    local counter = event.target.counter

    for i=1, #themeBottom do
        themeBottom[i].isVisible = false
        myText[i].isVisible = false
        if (completada[i] ~= nil) then
            completada[i].isVisible = false
        end
    end

    themeBottom[counter].isVisible = true
    myText[counter].isVisible = true
    navButton[4].x = 100 + (navButton[counter].width/2) +
navButton[counter].width * (counter-1)
    navButton[4].y = themeBottom[counter].y +
(themeBottom[counter].height/2) + 150

    if (Datos.actividad[counter].completado == 1) then
        completada[counter].isVisible = true
    end

end

-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view
    local buttonSheet
    local myGroupIMG
    myGroupIMG = display.newGroup()

    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    F_Stage() --Theme title
```

```
    local dir = Datos.tema[idTemas].imagen  --nombre de las imagenes BD
    local themeImage = display.newImageRect(dir,400,165)
    themeImage.x = display.contentCenterX
    themeImage.y = 100 + (themeImage.height/2)
    myGroupIMG:insert( themeImage )

    local i = 1
    for j = 1 , #Datos.actividad do --#imagenLeccion    #Datos.tema

        if (Datos.actividad[j].idTheme == idTemas) then
            local dir = "assets/themeBottom.png"  --bottom Image

            themeBottom[i] = display.newImageRect(dir,520,450)
            themeBottom[i].x = display.contentCenterX
            themeBottom[i].y = themeImage.y + themeBottom[i].height/2 +
150
            themeBottom[i].id = Datos.actividad[j].id
            themeBottom[i].counter = i
            themeBottom[i]:addEventListener( "touch", handleImageEvent )
            themeBottom[i].isVisible = false
            myGroupIMG:insert( themeBottom[i] )

            local optionsTexto =
            {
                text = Datos.actividad[j].nombre,  --Activity name
                width = themeBottom[i].width - 80,
                height = 0,
                font = "Montserrat-Regular",
                fontSize = 60,
                align = "center"
            }
            myText[i] = display.newText( optionsTexto )
            myText[i].x = themeBottom[i].x
            myText[i].y = themeBottom[i].y * .95
            myText[i].id = Datos.actividad[j].id
            myText[i].counter = i
            myText[i]:setFillColor( 1, 1, 1 )
            myText[i].isVisible = false
            myGroupIMG:insert( myText[i] )

            local options = {
                width = 110,
                height = 110,
                numFrames = 2,
            }
            buttonSheet = graphics.newImageSheet( "assets/navButton.png",
options )

            navButton[i] = display.newImageRect(buttonSheet,1, 110, 110)
            navButton[i].x = 100 + (navButton[i].width/2) +
navButton[i].width * (i-1)
            navButton[i].y = themeBottom[i].y + (themeBottom[i].height/2)
+ 150
            navButton[i].id = Datos.actividad[j].id
            navButton[i].counter = i
            navButton[i]:addEventListener( "touch",
handleImageEventSelect )
```

```
            myGroupIMG:insert( navButton[i] )

            if (Datos.actividad[j].completado == 1) then  --Agregar
indicador en caso de que ya se encuentre completado
                local dir = "assets/completo.png"
                completada[i] = display.newImageRect(dir,120,120)
                completada[i].x = themeBottom[i].x
                completada[i].y = themeBottom[i].y * 1.20
                completada[i].isVisible = false
                myGroupIMG:insert( completada[i] )
            end
            i = i + 1
        end
    end

    --Default is "screen 1" selected (intro)
    myText[1].isVisible = true
    themeBottom[1].isVisible = true
    navButton[4] = display.newImageRect(buttonSheet,2, 110, 110)
    navButton[4].x = 100 + (navButton[4].width/2)
    navButton[4].y = themeBottom[1].y + (themeBottom[1].height/2) + 150
    myGroupIMG:insert( navButton[4] )

    scrollView = widget.newScrollView
    {
        left = 0,
        top = 100,
        width = display.viewableContentWidth,
        height = display.viewableContentHeight,
        scrollWidth = display.viewableContentWidth,
        scrollHeight = display.viewableContentHeight,
        rightPadding = 100,
        leftPadding = 0,
        topPadding = 0,
        bottomPadding = 0,
        hideBackground = true,
        horizontalScrollingDisabled = true,
        verticalScrollingDisabled = false,
        isBounceEnabled = false
    }

    scrollView:insert( myGroupIMG )
    group:insert(scrollView)
end


F_Stage = function()
    local display_stage = display.getCurrentStage()

    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = true   --Options button
        elseif(display_stage[i].id == "titleBarText") then
            display_stage[i].isVisible = true   --Options button)
        elseif(display_stage[i].id == "salir") then
            display_stage[i].isVisible = true   --Options button)
```

```
        end
        i = i + 1
    end

end



-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )


return scene
```

sceneCero.lua

```
-----------------------------------------------------------------------------
-- scenecero.lua
-- Carga los elementos que serán constantes a lo largo de toda la app;
-- fondo y barra de título.
-----------------------------------------------------------------------------
local widget = require( "widget" )
local storyboard = require( "storyboard" )
local Datos = require( "data" )
local scene = storyboard.newScene()
-----------------------------------------------------------------------------
local handleButtonEvent
local handleButtonAlert
local show


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    -- Load image for bottom
    local fondo = display.newImageRect( "assets/fondo.png",
display.contentWidth,display.contentHeight)
    fondo.x = display.contentCenterX
    fondo.y = display.contentCenterY
    fondo.id = "fondo"

    -- Load image for title bar bottom
    local titleBarBottom = display.newImageRect(
"assets/titleBarBottom.png", display.contentWidth,110)
    titleBarBottom.x = fondo.x
    titleBarBottom.y = titleBarBottom.height/2
    titleBarBottom.id = "titleBarBottom"

    -- Load title bar text (XOOK)
```

```
    local titleBarText = display.newImageRect( "assets/XOOK.png",
380,110)
    titleBarText.x = display.contentCenterX
    titleBarText.y = titleBarBottom.y
    titleBarText.id = "titleBarText"

    -- Load image for subTitle bar bottom
    local subTitleBarBottom = display.newImageRect(
"assets/subTitleBarBottom.png", display.contentWidth,55)
    subTitleBarBottom.x = display.contentCenterX
    subTitleBarBottom.y = titleBarBottom.height + 22.5 --
(subTitleBarBottom.height/2)
    subTitleBarBottom.id = "subTitleBarBottom"

    --image sheet options and declaration
    local options = {
       width = 55,
       height = 55,
       numFrames = 2,
    }
    local buttonSheet = graphics.newImageSheet( "assets/backSheet.png",
options )

    -- Create backButton for the title bar
    local backButton = widget.newButton
    {
       id = "salir",
       sheet = buttonSheet,  --reference to the image sheet
       defaultFrame = 1,     --number of the "default" frame
       overFrame = 2,        --number of the "over" frame
       onRelease  = handleButtonEvent
    }
    backButton.x = 100
    backButton.y = subTitleBarBottom.y

    --image sheet declaration
    local buttonSheet = graphics.newImageSheet( "assets/optionSheet.png",
options )

    -- Create settingsButton for the title bar
    local settingsButton = widget.newButton
    {
       id = "opciones",
       sheet = buttonSheet,  --reference to the image sheet
       defaultFrame = 1,     --number of the "default" frame
       overFrame = 2,        --number of the "over" frame
       onRelease  = show
    }
    settingsButton.x = 675
    settingsButton.y = subTitleBarBottom.y

    --Create Title Bar Group
    local display_stage = display.getCurrentStage()
    display_stage:insert( 1,fondo )
    display_stage:insert( 2,titleBarBottom )
    display_stage:insert( 3,titleBarText )
    display_stage:insert( 4,subTitleBarBottom )
```

```
    display_stage:insert( backButton )
    display_stage:insert( settingsButton )

    -- load sceneTemas.lua
    storyboard.gotoScene("sceneTemas","slideLeft",1)
end


handleButtonEvent = function( event )
    local sceneName = storyboard.getCurrentSceneName()

    if(sceneName==Datos.scenes[1])then
        native.requestExit()
    elseif(storyboard.isOverlay) then
            storyboard.hideOverlay()
    elseif(sceneName == Datos.scenes[2] or sceneName == Datos.scenes[7])
then
        storyboard.gotoScene(Datos.scenes[1],"slideRight",200)
    elseif(sceneName  == Datos.scenes[4]) then
        native.showAlert("Salir","El progreso se perderá, aún así desea
salir", {"Si", "No"},handleButtonAlert)
    else
        storyboard.gotoScene(Datos.scenes[2],"slideRight",200)
    end

    return true
end

handleButtonAlert = function  ( event )
    if "clicked" == event.action then
        local i = event.index
        if i == 1 then --If yes
            storyboard.gotoScene(Datos.scenes[2],"slideRight",200)
        end
    end
end

show = function (event)

    if Datos.showing then
        storyboard.hideOverlay("fade",0)
    else
        local options =
        {
            effect = "flipFadeOutIn",
            time = 0,
            --isModal = true
        }
        storyboard.showOverlay( "overlayOpciones", options)
    end

    Datos.showing = not Datos.showing
end

-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )
```

```
return scene
```

sceneCreditos.lua

```
-----------------------------------------------------------------------
-- sceneCreditos.lua
-- Ventana de Creditos
-----------------------------------------------------------------------
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local widget = require( "widget" )


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    --Hide Settings Button
    local display_stage = display.getCurrentStage()
    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
        end
        i = i + 1
    end

    local message = [[-------------CREDITOS-------------
Creado por: Julio Bonilla

Contenidos:
Castañeda De I. P., Érik
Geometría analítica en el espacio
México
Facultad de Ingeniería-UNAM, 2003

Ejercicios. Adaptación del Laboratorio
Virtual de Matemáticas. Agradecimiento
a los participantes en el desarrollo
de los ejercicios modelo:
~ Casiano Aguilar Morales
~ Hortencia Caballero López
~ Juan Velázquez Torres
~ Luis Humberto Soriano Sánchez
~ María del Rocío Ávila Núñez
~ María Sara Valentina Sánchez Salinas
~ Mayverena Jurado Pineda
~ Ricardo Martínez Gómez
~ Rosalba Rodríguez Chávez
~ Sergio Arzamendi Pérez
```

```
~ Sergio Carlos Crail Corzas

Diseño gráfico: Marina Casañas

Sonidos:
~ 103628_benboncan_large-anvil-steel-hammer by Benboncan
https://freesound.org/people/Benboncan/sounds/103628/
~ 161592_jorickhoofd_wooden-log-falling by jorickhoofd
http://www.freesound.org/people/jorickhoofd/sounds/161592/
]]

    local options =
    {
        text = message,
        width = display.contentWidth * .80,
        height = 0,
        font = "Montserrat-Regular",
        fontSize = 35,
        align = "left"
    }
    local textBox = display.newText( options )
    textBox:setFillColor( 0, 0, 0)
    textBox.x = display.contentCenterX
    textBox.y = display.contentCenterY + 300

    local box = display.newImageRect( "assets/burbujab.png",
display.contentWidth * .90, textBox.height+20)
    box.x = textBox.x
    box.y = textBox.y

    local scrollView = widget.newScrollView
    {
        left = 0,
        top = 200,
        width = display.viewableContentWidth,
        height = display.viewableContentHeight,
        scrollWidth = box.contentWidth,
        scrollHeight = box.height,
        topPadding = 00,
        bottomPadding = 350,
        hideBackground = true,
        horizontalScrollingDisabled = true,
        verticalScrollingDisabled = false,
        isBounceEnabled = false,
    }

    scrollView:insert( box )
    scrollView:insert( textBox )
    group:insert( scrollView )
end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )

return scene
```

sceneEjercicios.lua

```
-----------------------------------------------------------------------------
---------
-- sceneEjercicios.lua
-- Selección de preguntas y respuestas al azar, despliegue de éstas,
sistema de
-- vidas.
-----------------------------------------------------------------------------
---------

local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local widget = require( "widget" )

--Modify time to hear taps
system.setTapDelay( 5 )

-- Invoke file with "global variables"
local Datos = require( "data" )


--Interface (constant objects)
local scrollView
local boton
local gauge
--Interface (variables for the questions)
local questionBox
local question
local answerGroup
local answerBox = {}
local answer = {}
local radioButton = {}
local buttonBox = {}
--Variables
local vidas = 1
local ciclo = 1
local cantidadPreguntas = 10 --How many questions will be showed
local secuencia = {} --Array that contains the questions to be shown
local numeroRespuestas
local correcta
local choose
local resultado
--Variables inherited from sceneActividades
local idActividades = Datos.actividadTarget  --Because of introduction
local idTemas = Datos.temaTarget

-----functions
local continua
local incorrecta

--Handlers
local function onSwitchPress( event )
```

```
    if (boton:getLabel()=="Continuar") then -- Used to disable the radio
button
        for k=1, #buttonBox do
            buttonBox[k]:setState({isOn=false})
        end
        buttonBox[choose]:setState({isOn=true})
        return true
    end


    if (event.phase=="began") then
        for k=1, #buttonBox do
            buttonBox[k]:setState({isOn=false})
        end
        choose = event.target.id

        correcta = Datos.pregunta[secuencia[ciclo]][choose].Correcta
        boton:setEnabled(true)
        boton:setFillColor(1,1)

        buttonBox[choose]:setState({isOn=true})
        return true
    end
end

local function handleButtonEvent( event )

    if (boton:getLabel()=="Continuar") then
        continua()
    else --First time that it's pressed
        answerBox[1]:setFillColor(0,1,0)
        if(correcta~= 1)then --If the answer was wrong
            answerBox[choose]:setFillColor(1,0,0)
            local options =
            {
                effect = "flipFadeOutIn",
                time = 400,
                --isModal = true,
                params = {
                result = 0}
            }
            storyboard.showOverlay( "overlayResult", options)
            vidas = vidas + 1 --Loose one life
            gaugeSprite:setSequence( "health" .. vidas )
            gaugeSprite:play()
            incorrecta()
        else
            local options =
            {
                effect = "flipFadeOutIn",
                time = 400,
                --isModal = true,
                params = {
                result = 1}
            }
            storyboard.showOverlay( "overlayResult", options)
        end
```

```
        --Tal vez tenga que ir arriba (en cada caso)
        if (ciclo < cantidadPreguntas) then -- # of questions to be shown
            storyboard.hideOverlay( true, "fade", 2000 )
        else
            storyboard.hideOverlay( "overlayResult", 2000)
        end

        boton:setLabel("Continuar")
    end
end

local function after( event )
    resultado:removeSelf()
    resultado = nil
end

local function handleTapEvent( event )
    if (event.numTaps > 1 ) then
        local target = event.target.id

        if(target=="question") then
            local options =
            {
                effect = "zoomOutInFade",
                time = 400,
                isModal = true,
                params = {
                filename = question.filename,
                Alto = question.height,
                Ancho = question.width,
                Ejercicios = true}
            }
            storyboard.showOverlay( "overlayZoom", options)
        else
            local options =
            {
                effect = "zoomOutInFade",
                time = 400,
                isModal = true,
                params = {
                    filename = answer[target].filename,
                    Alto = answer[target].height,
                    Ancho = answer[target].width,
                    Ejercicios = true}
            }
            storyboard.showOverlay( "overlayZoom", options)
        end
    end
end


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    if(not(storyboard.getPrevious() == nil)) then
```

```
        storyboard.removeScene(storyboard.getPrevious())
    end

    --Hide settings button, change titleBarText
    local display_stage = display.getCurrentStage()
    local i = 2
    local j
    while display_stage[i] ~= nil do

        if(display_stage[i].id == "titleBarText") then
            display_stage[i].isVisible = false
            j = i
        elseif(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
            --break
        end
        i = i + 1
    end

    -- Load theme title bar text (replace XOOK)
    local dir = Datos.tema[idTemas].imagen  --nombre de las imagenes BD
    local titleBarText2 = display.newImageRect(dir,380,110)
    titleBarText2.x = display.contentCenterX
    titleBarText2.y = display_stage[j].y

    -- Create a ScrollView
    scrollView = widget.newScrollView
    {
        left = 0,
        top = 200,
        width = display.ContentWidth,
        height = display.ContentHeight,
        scrollWidth = 1800,
        scrollHeight = 1400,
        rightPadding = 200,
        topPadding = 0,
        bottomPadding = 200,
        hideBackground = true,
        horizontalScrollingDisabled = false,
        verticalScrollingDisabled = true,
        isBounceEnabled = false
    }

    local buttonOptions = {
       width = 220,
       height = 120,
       numFrames = 2,
    }

    local buttonSheet = graphics.newImageSheet( "assets/navButtonG.png",
buttonOptions )


    boton = widget.newButton
    {
        id=i,
        label = "Calificar",
```

```
        labelAlign = "center",
        sheet = buttonSheet,  --reference to the image sheet
        defaultFrame = 2,     --number of the "default" frame
        overFrame = 1,        --number of the "over" frame
        font = "Montserrat-Regular",
        fontSize = 30,
        labelColor = { default = {0,0,0}, over = {0,0,255} },
        isEnabled = false,
        onRelease = handleButtonEvent
    }
    boton.xScale = 2

    local options = {
        width = 150,
        height = 50,
        numFrames = 4,
    }
    local gaugeSheet =
graphics.newImageSheet("assets/sheetLife.png",options)

    local sequenceData = {
       { name = "health1", start=1, count=1, time=0, loopCount=1 },
       { name = "health2", start=2, count=1, time=0, loopCount=1 },
       { name = "health3", start=3, count=1, time=0, loopCount=1 },
       { name = "health4", start=4, count=1, time=0, loopCount=1 }
    }

    gaugeSprite = display.newSprite( gaugeSheet, sequenceData )
    gaugeSprite.x = 660
    gaugeSprite.y = display_stage[4].y

    -----Which questions are going to be loaded------
    local totalPreguntas = 0
    local temporalTable = {}    -- new temporal array
    j = 1

    for key,value in pairs(Datos.pregunta) do
        if (value.idActivity == idActividades) then
            totalPreguntas = totalPreguntas + 1
            temporalTable[j] = key
            j = j + 1
        end
    end

    --Create and aleatory array that contains the index of the questions
to be shown
    for i = 1, cantidadPreguntas do
        local pos = math.random( totalPreguntas - (i-1))
        local numero =  table.remove(temporalTable,pos)
        secuencia[i] = numero
    end
    --------questions to be loaded-----------

    --Clear temporal variables
    totalPreguntas = nil
    temporalTable = nil
    j = nil
```

```
    group:toFront()
    scrollView:insert( boton )
    group:insert( gaugeSprite )
    group:insert( scrollView )
    group:insert( titleBarText2 )

end


-- Called immediately before scene is moved on screen:
function scene:willEnterScene( event )
    local group = self.view


    --button disabled until the user makes a choice
    boton:setEnabled(false)
    boton:setFillColor(.5,.5)
    boton:setLabel("Calificar")

    local numeroPregunta = secuencia[ciclo] --Id of the question to be
shown

    ----Display question-----
    local dir = Datos.pregunta[numeroPregunta].Texto --link of the image
    local alto = Datos.pregunta[numeroPregunta].Alto
    local ancho = 600

    question = display.newImageRect(dir,ancho,alto)

    question.x = ancho/2 + 100
    question.y = question.height/2
    question.id = "question"
    question.filename = dir

    question:addEventListener( "tap", handleTapEvent ) --Double tap to
zoom

    ---------Answers-------
    answerGroup = display.newGroup() --Group for the Answers
    local answersheight = question.height/2 + question.y --Variable with
the total height of the Questions & Answers
    local answersWidth = ancho

    ------Random---------------------------------------
    local totalRespuestas = #Datos.pregunta[numeroPregunta]
    local temporalTable = {}     -- new array
    for i=1, totalRespuestas do -- Array initialized with consecutive
numbers
      temporalTable[i] = i
    end

    --Create an aleatory array with de id of the answer to be showed
    local numeroRespuesta = {}
    for i=1, totalRespuestas do
        local pos = math.random( totalRespuestas - (i-1) )
        local numero =  table.remove(temporalTable,pos)
```

```
        numeroRespuesta[i] = numero
    end
    ------ End of random-------------------------------

    --Draw the answers-------
    for i=1, totalRespuestas do
        local nR = numeroRespuesta[i] --Alias

        if (Datos.pregunta[numeroPregunta].Tipo == 1) then
            print("Tipo 1")

            local dir = Datos.pregunta[numeroPregunta][nR].Opcion--image
direction
            local alto = Datos.pregunta[numeroPregunta][nR].Alto
            local ancho = Datos.pregunta[numeroPregunta][nR].Ancho
            if (ancho > 600) then
                ancho = 600
            end
            print(dir)
            print(alto)
            print(ancho)

            answer[nR] = display.newImageRect(dir,ancho,alto) --Ancho y
alto

            answersheight = answersheight + answer[nR].height/2 + 100
            answer[nR].x = answer[nR].width/2 + 120
            answer[nR].y = answersheight
            answer[nR].id = nR
            answer[nR].filename = dir

            answer[nR]:addEventListener( "tap", handleTapEvent )

            -- Box (rectangle) that will contain the answer
            answerBox[nR] = display.newImageRect( "assets/burbujab.png",
answer[nR].width+50, answer[nR].height+10 )
            answerBox[nR].x = answer[nR].x - 10
            answerBox[nR].y = answersheight
            answerBox[nR].id = nR

            if(answersWidth < answerBox[nR].width) then --If the box is
smaller make it bigger than the answer
                    answersWidth = answerBox[nR].width + 50
            end

            radioButton[nR] = widget.newSwitch
            {
                left = answerBox[nR].x-answerBox[nR].width/2,
                top = answer[nR].y-20,
                style = "radio",
                id = nR,
                initialSwitchState = false,
                onPress = onSwitchPress
            }
            buttonBox[nR] = radioButton[nR]
```

```
            answerBox[nR]:addEventListener( "touch", onSwitchPress ) --
All the elements of the answer should activate the rb

            answerGroup:insert( answerBox[nR] )
            answerGroup:insert( answer[nR] )
            answerGroup:insert( radioButton[nR] )
        else
            local dir = Datos.pregunta[numeroPregunta][nR].Opcion
            local alto = Datos.pregunta[numeroPregunta][nR].Alto
            if (alto > 330) then
                alto = 330
            end

            local ancho = Datos.pregunta[numeroPregunta][nR].Ancho
            if (ancho > 250) then
                ancho = 250
            end

            answer[nR] = display.newImageRect(dir,ancho,alto)
            local X =  question.x - question.width/2 + answer[nR].width/2
--+ 50

            if(nR<3) then
                answer[nR].x = X
            else
                answer[nR].x = X + ancho + 100
                local anchonR = answer[nR].x + answer[nR].width/2
                if(answersWidth < (anchonR - 100) ) then
                    answersWidth = anchonR - 100
                end
            end

            if(nR==1 or nR==3) then
                answer[nR].y = question.y + question.height/2  +
answer[nR].height/2 + 75
            else
                answersheight = answersheight + alto + 87.5 --Will
increment 2 times
                answer[nR].y = question.y + question.height/2 + alto*1.5
+ 175
            end

            answer[nR].id = nR

            answer[nR].filename = dir

            answer[nR]:addEventListener( "touch", onSwitchPress )
            answer[nR]:addEventListener( "tap", handleTapEvent )

            answerBox[nR] = display.newImageRect( "assets/burbuja.png",
answer[nR].width+40, answer[nR].height+60 )
            answerBox[nR].id = nR
            answerBox[nR].x = answer[nR].x
            answerBox[nR].y = answer[nR].y + 10

            radioButton[nR] = widget.newSwitch
            {
```

```
                left = answerBox[nR].x-10,
                top = answer[nR].y+ answer[nR].height/2 ,
                style = "radio",
                id = nR,
                initialSwitchState = false,
                onPress = onSwitchPress
            }
            buttonBox[nR] = radioButton[nR]

            answerBox[nR]:addEventListener( "touch", onSwitchPress )

            answerGroup:insert( answerBox[nR] )
            answerGroup:insert( answer[nR] )
            answerGroup:insert( radioButton[nR] )
        end
    end

    questionBox = display.newImageRect( "assets/burbujab.png",
answersWidth + 100, answersheight + 100 )
    questionBox.x = answersWidth/2 + 100
    questionBox.y = (answersheight/2) + 50

    boton.x = questionBox.x
    boton.y =  answersheight + 200

    scrollView:insert( questionBox )
    scrollView:insert( question )
    scrollView:insert( answerGroup )
end


-- Called when scene is about to move offscreen, when scene is reloaded:
function scene:exitScene( event )
    local group = self.view

    for numeroRespuestas=#answerBox,1,-1  do
        answerBox[numeroRespuestas]:removeEventListener( "touch",
onSwitchPress)
        answerGroup:remove( answerBox[numeroRespuestas] )
        answerBox[numeroRespuestas] = nil
        answerGroup:remove( answer[numeroRespuestas] )
        answer[numeroRespuestas] = nil
        answerGroup:remove( radioButton[numeroRespuestas] )
        radioButton[numeroRespuestas] = nil
    end

    scrollView:remove( answerGroup )
    display.remove( answerGroup )
    answerGroup = nil
    scrollView:remove( question )
    display.remove( question )
    question = nil
    scrollView:remove( questionBox )
    display.remove( questionBox )
    questionBox = nil

    buttonBox = {}
```

```
end


-- Called prior to the removal of scene's "view" (display group)
function scene:destroyScene( event )
    local display_stage = display.getCurrentStage()

    display_stage[3].isVisible = true

    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
        end
        i = i + 1
    end

end


-- the following event is dispatched to continue or end the test
continua = function ()
    if(vidas==4) then
        --storyboard.hideOverlay( "fade",100 )
        storyboard.gotoScene("sceneGameover","slideLeft",300)
    else
        if (ciclo < cantidadPreguntas) then -- # of questions to be shown
            --storyboard.hideOverlay( true, "fade",100 )
            storyboard.reloadScene()
        else
            --storyboard.hideOverlay( "fade",100 )
            storyboard.gotoScene("sceneGameComplete","slideLeft",300)
        end
        ciclo = ciclo + 1
    end
end

incorrecta = function ()
    resultado = display.newImageRect( "assets/piedra.png", 50,50)
    resultado:toFront()

    if(vidas==2) then
        resultado.x = 660 - 257/3
    elseif(vidas==3) then
        resultado.x = 660
    else
        resultado.x = 660 + 257/3
    end
    resultado.y = resultado.height/2 + 14

    transition.to( resultado, { time=1100, y=1200, rotation=720,
alpha=0.2, transition=easing.outInBack } )
    timer.performWithDelay( 1800,after)
end

-- "createScene" event is dispatched if scene's view does not exist
```

```
scene:addEventListener( "createScene", scene )

-- "enterScene" event is dispatched whenever scene transition has
finished
scene:addEventListener( "willEnterScene", scene )

-- "exitScene" event is dispatched before next scene's transition begins
scene:addEventListener( "exitScene", scene )

-- "destroyScene" event is dispatched before view is unloaded, which can
be
-- automatically unloaded in low memory situations, or explicitly via a
call to
-- storyboard.purgeScene() or storyboard.removeScene().
scene:addEventListener( "destroyScene", scene )

-- scene:addEventListener( "overlayEnded" )
---------------------------------------------------------------------------

return scene
```

sceneGameComplete.lua

```
---------------------------------------------------------------------------
-- sceneGameComplete.lua
-- All the questions were successfully answered
---------------------------------------------------------------------------
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local widget = require( "widget" )

-- Invoke file with "global variables"
local Datos = require( "data" )

-- Module for loading & saving data
local loadsave = require( "loadsave" )

-- Invoke file with "saved data"
local datosPermanentes = loadsave.loadTable("permanentdata.json")
local dP = datosPermanentes

--Handle of the timer
local htimer

--Handler
local function handleButtonEvent( event )
    storyboard.gotoScene("sceneActividades","slideLeft",600)
end

local function handleTapEvent( event )
    transition.cancel( "transition" )
    audio.stop( )
    htimer = nil
    storyboard.gotoScene("sceneActividades","slideLeft",1200)
end
```

```
local function listener( event )
    if (htimer ~= nil and Datos.sound == true) then
        audio.play( Datos.gameComplete,{channel = 1} )
    end
end



-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    local tamenX = 408
    local tamenY = 544
    local animaX
    local animaO
    local animaO2
    local animaK

    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    --Hide Settings Button
    local display_stage = display.getCurrentStage()
    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
        elseif(display_stage[i].id == "salir") then
            display_stage[i].isVisible = false
        end
        i = i + 1
    end

    local function completeListener ( obj )
    ----- load sceneActividades.lua

        fondo = nil
        letterX = nil
        letterO = nil
        letterO2 = nil
        letterK = nil

        tamenX = nil
        tamenY = nil
        animaX = nil
        animaO = nil
        animaO2 = nil
        animaK = nil

        transition.cancel()
        storyboard.gotoScene("sceneActividades","slideLeft",1400)
    end

    ----------------------Lets Update-----------------------
```

```
    --Variables de la escena Actividades----
    local idEjercicios = Datos.actividadTarget
    local idTemas = Datos.temaTarget

    dP.actividad[idTemas][idEjercicios].completado =  1  --This activity
was completed
    Datos.actividad[idTemas][idEjercicios].completado =  1

    local temaUpdate = 1

    for id=1, #Datos.actividad[idTemas] do
        if(Datos.actividad[idTemas][id].tipo == 2) then --Only if it's an
exercise
            if(Datos.actividad[idTemas][id].completado ~= 1) then --All
the activities from this theme were completed?
                temaUpdate = 0 --If not, we don't need to update
                break --go out
            end
        end
    end

    if (temaUpdate == 1) then --If all the activities in the theme are
complete
        dP.tema[idTemas].completado = 1
        Datos.tema[idTemas].completado = 1

        local nivel = Datos.tema[idTemas].nivel --Alias of the actual lvl
        local temaUpgrade = 1
        for tema=1, #Datos.tema do --Let´s check in (almost)all the
themes
            if(Datos.tema[tema].nivel==nivel) then --Checking only if
it´s the same lvl
                if(Datos.tema[idTemas].completado ~= 1) then --If one
from the lvl isn´t complete
                    temaUpgrade = 0 --We don´t upgrade
                    break --Go out
                end
            elseif(Datos.tema[tema].nivel==nivel+1) then --We are one lvl
up
                if(temaUpgrade==1)then --If the previous lvl was complete
                    Datos.tema[idTemas].activo = 1
                    dP.tema[idTemas].activo = 1
                end
            elseif(Datos.tema[tema].nivel>nivel+1) then --We are more
than one lvl up
                break
            end
        end
    end

    loadsave.saveTable(dP, "permanentdata.json")  --Save in the file
    ---------------------------- End of update ---------------------------
---------------------------

    -- Create the result text
    local optionsTexto =
```

```
    {
        text = Datos.actividad[idTemas][idEjercicios].nombre .. "
Completado",
        x = 100,
        y = 150,
        width = 550,
        height = 0,
        font = "Montserrat-Regular",
        fontSize = 80,
        align = "center"
    }
    local resultText = display.newText( optionsTexto )
    resultText.anchorX = 0
    resultText.anchorY = 0
    resultText:setFillColor(1, 1, 1)
    group:insert( resultText )

    local fondo = display.newImageRect( "assets/fondo.png",
display.contentWidth,display.contentHeight)
    fondo.x = display.contentCenterX
    fondo.y = display.contentCenterY
    fondo.id = 0
    fondo:addEventListener( "tap", handleTapEvent )
    group:insert( fondo )

    local letterX = display.newImageRect( "assets/X.png", tamenX,tamenY)
    letterX.x = -350
    letterX.y = 420
    letterX.id = 1
    letterX:addEventListener( "tap", handleTapEvent )
    group:insert( letterX )

    local letterO = display.newImageRect( "assets/O.png", tamenX,tamenY)
    letterO.x = 1200
    letterO.y = 310
    letterO.id = 2
    letterO:addEventListener( "tap", handleTapEvent )
    group:insert( letterO )

    local letterO2 = display.newImageRect( "assets/O.png", tamenX,tamenY)
    letterO2.x = -350
    letterO2.y = 1200
    letterO2.id = 3
    letterO2:addEventListener( "tap", handleTapEvent )
    group:insert( letterO2 )

    local letterK = display.newImageRect( "assets/K.png", tamenX,tamenY)
    letterK.x = 1200
    letterK.y = 1090
    letterK.id = 4
    letterK:addEventListener( "tap", handleTapEvent )
    group:insert( letterK )

    -----------------Start animating--------------------

    htimer = timer.performWithDelay( 950, listener )
    animaX = transition.to(letterX,{
```

```
    time = 1000,
    x = 230,
    y = 620,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 1900, listener ) })

    animaO = transition.to(letterO,{
    time = 1000,
    delay = 1000,
    x = 520,
    y = 510,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 2850, listener )})

    animaO2 = transition.to(letterO2,{
    time = 1000,
    delay = 2000,
    x = 310,
    y = 1000,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 3800, listener )})

    animaK = transition.to(letterK,{
    time = 1000,
    delay = 3000,
    x = 600,
    y = 890,
    transition = easing.linear,
    tag="transition",
    onComplete = completeListener})

end



-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )


return scene
```

sceneGameOver.lua

```
-----------------------------------------------------------------------
-- sceneGameOver.lua
-- Animación en caso de haber contestado erróneamente 3 veces
-----------------------------------------------------------------------
local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

local Datos = require( "data" )
```

```lua
local htimer --Handle of the timer

local function handleTapEvent( event )
    transition.cancel( "transition" )
    storyboard.gotoScene("sceneActividades","slideLeft",1200)
    audio.stop( )
    htimer = nil
end

local function listener( event )
    if (htimer ~= nil and Datos.sound == true) then
        audio.play( Datos.gameOver,{channel = 1} )
    end
end

-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    local tamenX = 408
    local tamenY = 544
    local animaX
    local animaO
    local animaO2
    local animaK


    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    --Hide Settings Button
    local display_stage = display.getCurrentStage()
    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = false
        elseif(display_stage[i].id == "salir") then
            display_stage[i].isVisible = false
        end
        i = i + 1
    end


    local function completeListener ( obj )
    ----- load sceneActividades.lua

        fondo = nil
        letterX = nil
        letterO = nil
        letterO2 = nil
        letterK = nil

        tamenX = nil
        tamenY = nil
        animaX = nil
        animaO = nil
```

```
        animaO2 = nil
        animaK = nil

        transition.cancel()
        storyboard.gotoScene("sceneActividades","slideLeft",1200)
    end


    local fondo = display.newImageRect( "assets/fondo.png",
display.contentWidth,display.contentHeight)
    fondo.x = display.contentCenterX
    fondo.y = display.contentCenterY
    fondo.id = 0
    fondo:addEventListener( "tap", handleTapEvent )
    group:insert( fondo )

    -- Create the result text
    local optionsTexto =
    {
        text = "Game Over",
        x = 100,
        y = 100,
        width = 550,
        height = 0,
        font = "Montserrat-Regular",
        fontSize = 85,
        align = "center"
    }
    local resultText = display.newText( optionsTexto )
    resultText.anchorX = 0
    resultText.anchorY = 0
    resultText:setFillColor(1, 1, 1)
    group:insert( resultText )

    local letterX = display.newImageRect( "assets/X.png", tamenX,tamenY)
    letterX.x = 230
    letterX.y = 620
    letterX.id = 1
    letterX:addEventListener( "tap", handleTapEvent )
    group:insert( letterX )

    local letterO = display.newImageRect( "assets/O.png", tamenX,tamenY)
    letterO.x = 520
    letterO.y = 510
    letterO.id = 2
    letterO:addEventListener( "tap", handleTapEvent )
    group:insert( letterO )

    local letterO2 = display.newImageRect( "assets/O.png", tamenX,tamenY)
    letterO2.x = 310
    letterO2.y = 1000
    letterO2.id = 3
    letterO2:addEventListener( "tap", handleTapEvent )
    group:insert( letterO2 )

    local letterK = display.newImageRect( "assets/K.png", tamenX,tamenY)
    letterK.x = 600
```

```
    letterK.y = 890
    letterK.id = 4
    letterK:addEventListener( "tap", handleTapEvent )
    group:insert( letterK )

    -----------------Start animating--------------------

    htimer = timer.performWithDelay( 950, listener )
    animaX = transition.to(letterX,{
    time = 1000,
    alpha = 0,
    y = display.contentHeight,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 1900, listener ) })


    animaO = transition.to(letterO,{
    time = 1000,
    delay = 1000,
    alpha = 0,
    y = display.contentHeight,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 2850, listener )})


    animaO2 = transition.to(letterO2,{
    time = 1000,
    delay = 2000,
    alpha = 0,
    y = display.contentHeight,
    transition = easing.linear,
    tag="transition",
    onComplete = timer.performWithDelay( 3800, listener )})


    animaK = transition.to(letterK,{
    time = 1000,
    delay = 3000,
    alpha = 0,
    y = display.contentHeight,
    transition = easing.linear,
    tag="transition",
    onComplete = completeListener})

end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )


return scene
```

sceneIntroduccion.lua

```
-----------------------------------------------------------------------------
-- sceneIntroduccion.lua
-- Carga el módulo de introducción correspondiente
-----------------------------------------------------------------------------

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- Invoke file with "global variables"
local Datos = require( "data" )

--Forward declaration
local idTemas = Datos.temaTarget
local idActividad = Datos.actividadTarget

--Variable
local introImage = {}

--Handler
local function handleTapEvent( event )
    if (event.numTaps > 1 ) then
        local target = event.target.id

        local options =
        {
            effect = "zoomOutInFade",
            time = 400,
            isModal = true,
            params = {
            filename = introImage[target].filename,
            Alto = introImage[target].height,
            Ancho = introImage[target].width,
            Ejercicios = false}
        }
        storyboard.showOverlay( "overlayZoom", options)

    end
end



-- Called when the scene's view does not exist:
function scene:enterScene( event )
    local group = self.view

    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    local display_stage = display.getCurrentStage()
    local i = 2
    local j
    local ancho=600
```

```
    --Change titleBarText
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "titleBarText") then
            display_stage[i].isVisible = false
            j = i
            break
        end
        i = i + 1
    end

    -- Load theme title bar text (replace XOOK)
    local dir = Datos.tema[idTemas].imagen  --nombre de las imagenes BD
    local titleBarText2 = display.newImageRect(dir,380,110)
    titleBarText2.x = display.contentCenterX
    titleBarText2.y = display_stage[j].y

    -- Create a ScrollView
    local scrollView = widget.newScrollView
    {
        left = 0,
        top = 200,
        width = display.viewableContentWidth,
        height = display.viewableContentHeight,
        scrollWidth = 1800,
        scrollHeight = 1600,
        rightPadding = 150,
        topPadding = 0,
        bottomPadding = 200,
        hideBackground = true,
        horizontalScrollingDisabled = false,
        verticalScrollingDisabled = false,
        isBounceEnabled = false
    }

    -- Load and show image
    j = 1
    for idintroduccion = 1, #Datos.introduccion do

        if(Datos.introduccion[idintroduccion].idActividad == idActividad
) then
            local dir = Datos.introduccion[idintroduccion].texto --link
of the image
            local alto = Datos.introduccion[idintroduccion].alto

            introImage[j] = display.newImageRect(dir,ancho,alto)
            introImage[j].x = 400+650*(j-1)
            introImage[j].y = display.contentCenterY - 200 --
scrollView.top
            introImage[j].id = j
            introImage[j].filename = dir

            introImage[j]:addEventListener( "tap", handleTapEvent ) --
Double tap to zoom

            --local box = display.newImageRect( "assets/burbujab.png",
introImage[j].width+20, introImage[j].height+60)
```

```
            local box = display.newImageRect( "assets/burbujab.png",
introImage[j].width+20, 1100)
            box.x = introImage[j].x - 10
            box.y = box.height/2

            scrollView:insert( box )
            scrollView:insert( introImage[j] )

            j = j + 1
        end

    end

    group:insert(titleBarText2)
    group:insert( scrollView )
end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "enterScene", scene )


return scene
```

sceneSplash.lua

```
-----------------------------------------------------------------------------
-- sceneSplash.lua
-- Pantalla de inicio, muestra el logotipo
-----------------------------------------------------------------------------
local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- Invoke file with "global variables"
local Datos = require( "data" )


local function startEvent( event )
    storyboard.gotoScene("sceneCero","slideLeft",1200)
end


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view

    local fondo = display.newImageRect( "assets/fondo.png",
display.contentWidth,display.contentHeight)
    fondo.x = display.contentCenterX
    fondo.y = display.contentCenterY
    fondo:addEventListener( "tap", startEvent )
    group:insert( fondo )

    local splash = display.newImageRect( "assets/splash.png", 780,780)
```

```
    splash.x = display.contentCenterX
    splash.y = display.contentCenterY - 50
    splash:addEventListener( "tap", startEvent )
    group:insert( splash )

    if(Datos.sound)then
        audio.play( Datos.gameStart, {channel = 1} )
    end

    timer.performWithDelay( 900, startEvent )
end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )


return scene
```

sceneTemas.lua

```
-----------------------------------------------------------------------
-- sceneTemas.lua
-- Muestra los temas disponibles
-----------------------------------------------------------------------
local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()


-- Invoke file with "global variables"
local Datos = require( "data" )

-- forward declarations
local F_dibujaBoton
local F_dibujaLinea
local F_Stage
--Variables
local anchoBoton = 330
local altoBoton = 165


--Handler
local function handleButtonEvent( event )
    local target = event.target
    Datos.temaTarget = event.target.id
    storyboard.gotoScene("sceneActividades","slideLeft",200)
end


-- Called when the scene's view does not exist:
function scene:createScene( event )
    local group = self.view
    local x1,x2,y,bandera,nLinea,nBoton
    local nivel = 1
```

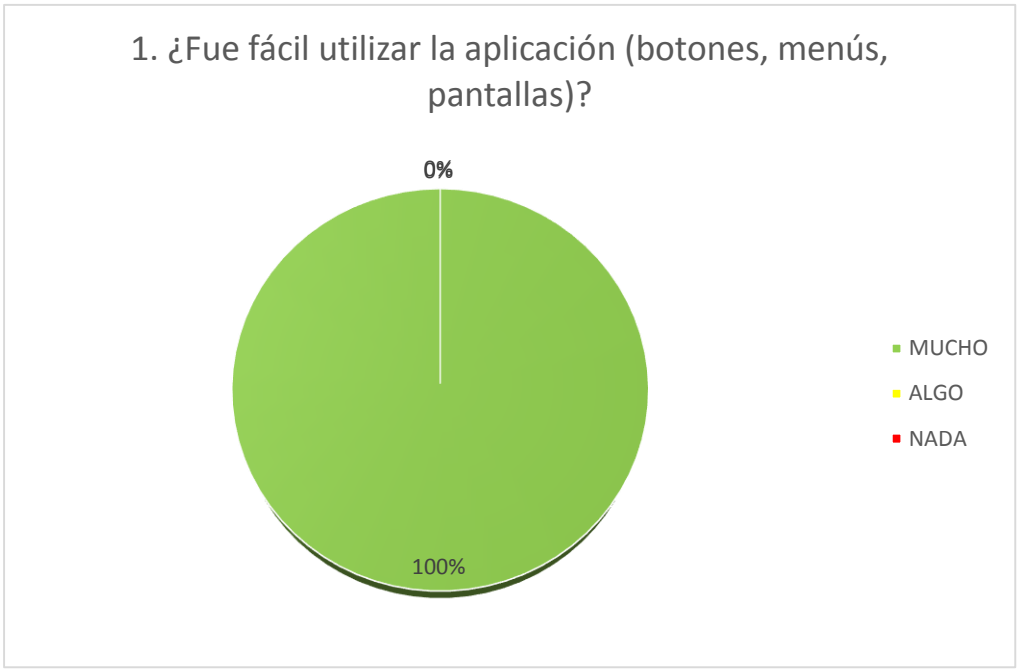
```
    local botonxnivel={}
    botonxnivel[nivel]=0
    y = altoBoton + 130 -- altoBoton + Margen


    if(not(storyboard.getPrevious() == nil)) then
        storyboard.removeScene(storyboard.getPrevious())
    end

    local display_stage = display.getCurrentStage()
    local i=1
    while display_stage[i] ~= nil do
        if(display_stage[i].id == "salir") then
            display_stage[i].isVisible = true
        end
        if(display_stage[i].id == "opciones") then
            display_stage[i].isVisible = true
        end
        i = i + 1
    end

    local groupBackground = display.newGroup()

    for i=1, #Datos.tema do

        if (Datos.tema[i].nivel == nivel) then
            botonxnivel[nivel] = botonxnivel[nivel]+1
            nBoton =
F_dibujaBoton(Datos.tema[i].id,i,nivel,botonxnivel[nivel])
            groupBackground:insert(nBoton)

            if (nivel>1) then
                --Dibuja las lineas a la izquierda
                nLinea = F_dibujaLinea( x2,y*botonxnivel[nivel],
x2+50,y*botonxnivel[nivel] )
                -- x2,(y*j),x2,(y*j)
                groupBackground:insert(nLinea)

                if(bandera) then
                    -----Dibuja la linea vertical-----
                    if(botonxnivel[nivel-1]>botonxnivel[nivel]) then
                        nLinea = F_dibujaLinea( x2,y,
x2,y*botonxnivel[nivel-1] )
                        -- x2,(y*j),x2,(y*j)
                    else
                        nLinea = F_dibujaLinea( x2,y,
x2,y*botonxnivel[nivel] )
                        -- x2,(y*j),x2,(y*j)
                    end
                    groupBackground:insert(nLinea)
                    bandera=false
                end
            end
        else
            x1=200 +(anchoBoton+100)*(nivel-1) + (anchoBoton*.5) --
boton.x + (.5*anchoBoton)
            x2=x1+50
```

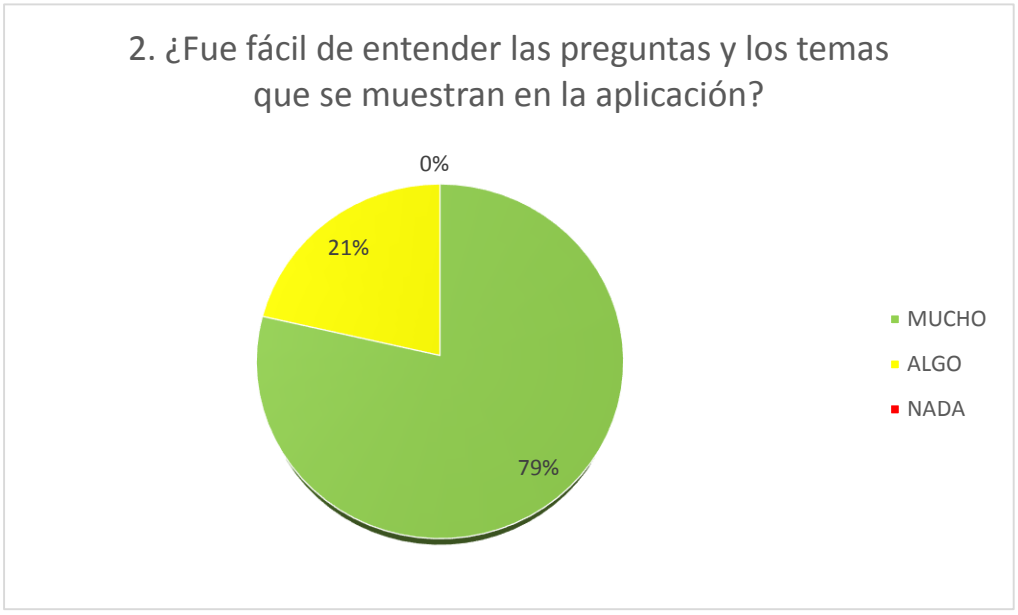
```
            --Dibuja las lineas a la derecha
            for j=1, botonxnivel[nivel] do
                nLinea = F_dibujaLinea(x1,y*j, x2,y*j)
                -- x1, (altoBoton + Margen)*j, x2,y1
                groupBackground:insert(nLinea)
            end

            nivel = nivel+1
            botonxnivel[nivel]=1
            bandera=true
            -----Dibuja el primer boton y la primer linea izq del
siquiente nivel-----
            nBoton =
F_dibujaBoton(Datos.tema[i].id,i,nivel,botonxnivel[nivel])
            nLinea = F_dibujaLinea( x2,y, x2+50,y )
            groupBackground:insert(nBoton)
            groupBackground:insert(nLinea)

            -----Dibuja la linea vertical si solo queda un elemento-----
            if((i+1)>#Datos.tema) then
                nLinea = F_dibujaLinea( x2,y, x2, y*botonxnivel[nivel-1]
)
                groupBackground:insert(nLinea)
            end

        end
    end

    if(x2==nil) then
        x2 = display.viewableContentWidth
    end

    -- Create a ScrollView
    local scrollView = widget.newScrollView
    {
        left = 0,
        top = 0,
        width = display.viewableContentWidth,
        height = display.viewableContentHeight,
        scrollWidth = x2*2,
        scrollHeight = y*6,
        rightPadding = 200,
        topPadding = 0,
        bottomPadding = 50,
        hideBackground = true,
        horizontalScrollingDisabled = false,
        verticalScrollingDisabled = false,
        --backgroundColor = {0,0,0},
        isBounceEnabled = false,
    }

    scrollView:insert( groupBackground )
    group:insert( scrollView )
end
```

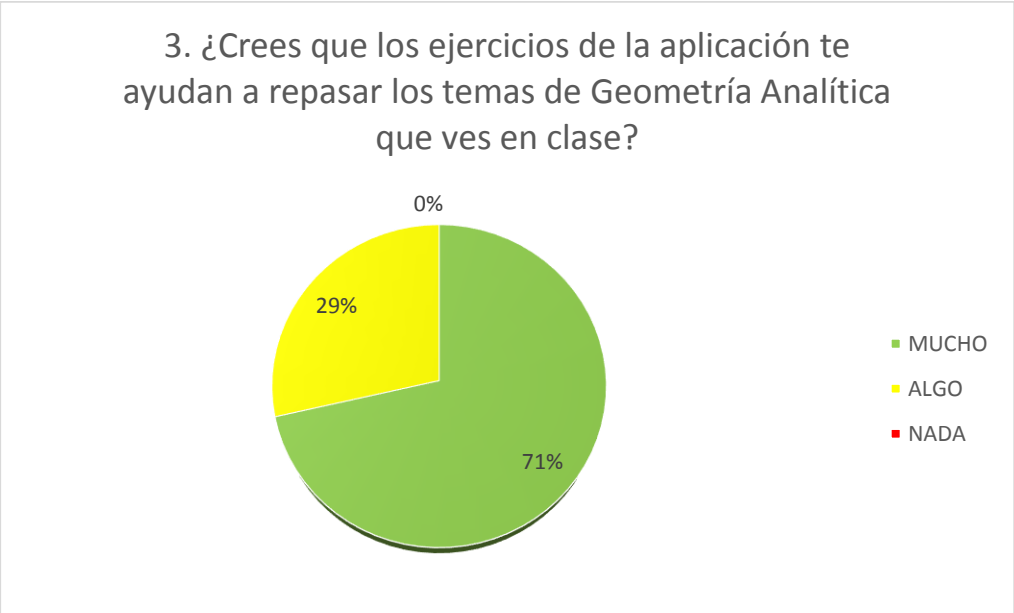
```
F_dibujaBoton = function (idTema,i,nivel,botonxnivel)
    local boton

    local options = {
       width = 330,
       height = 165,
       numFrames = 2,
    }
    local buttonSheet = graphics.newImageSheet(
"assets/blackboardSheet.png", options )

    boton = widget.newButton
        {
            id = idTema,
            label = Datos.tema[i].nombre,
            labelAlign = "center",
            sheet = buttonSheet,  --reference to the image sheet
            defaultFrame = 1,     --number of the "default" frame
            overFrame = 2,        --number of the "over" frame
            fontSize = 40,
            font = "Montserrat-Regular",
            labelColor = { default = {1,1,1}, over = {1,1,1} },
            isEnabled = true,
            onRelease = handleButtonEvent
        }

    if(Datos.tema[i].activo == 0) then
        boton:setEnabled( false )
    end
    boton.x = 200+(anchoBoton+100)*(nivel-1) -- MargenIzq + (anchoboton +
MargenDer)
    boton.y = (altoBoton+130)*botonxnivel  -- (altoBoton + Margen)

    return boton
end

F_dibujaLinea = function (x1,y1,x2,y2)
    local linea = display.newLine(x1,y1,x2,y2)
    linea:setStrokeColor( 1, 1, 1)
    linea.strokeWidth = 10

    return linea
end


-- "createScene" event is dispatched if scene's view does not exist
scene:addEventListener( "createScene", scene )


return scene
```

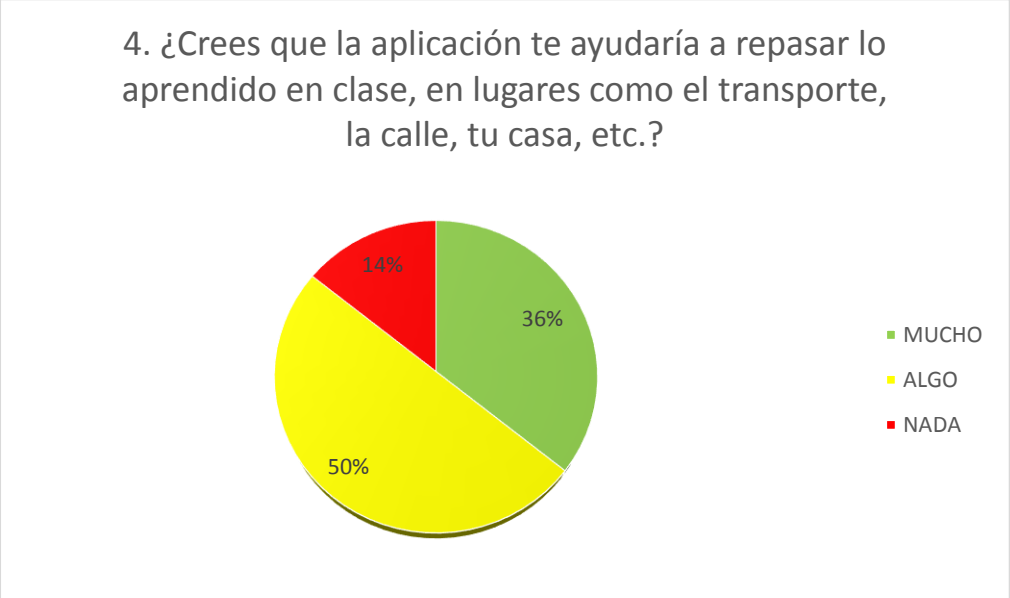
ANEXO 4 RESULTADOS DE LA EVALUACIÓN

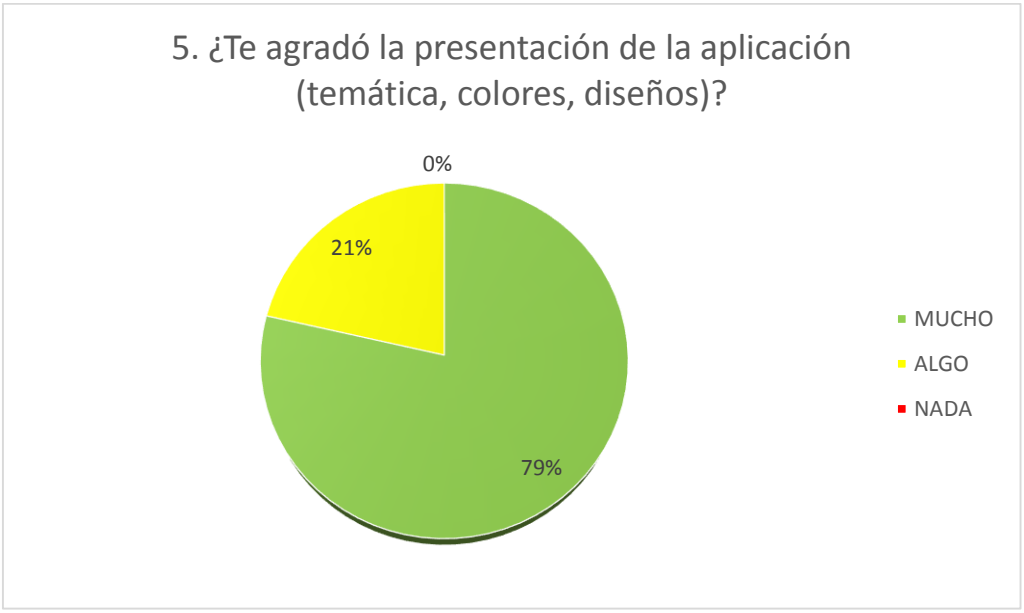
Con el fin de que se pueda obervar a mayor detalle el resultado de las encuestas realizadas a los usuarios, a continuación se muestra una gráfica por cada una de las preguntas.

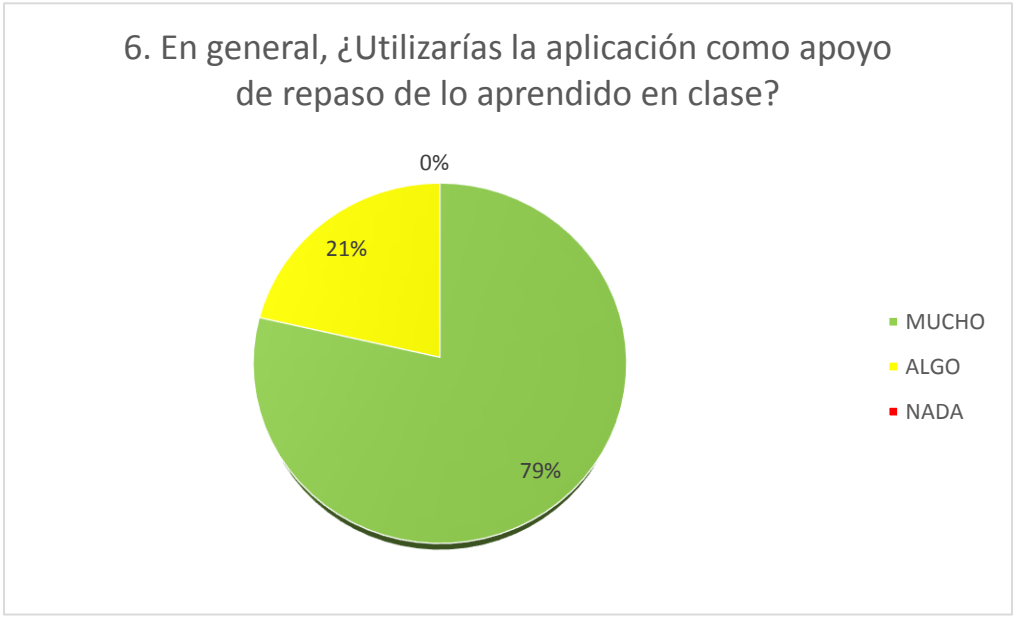





3. ¿Crees que los ejercicios de la aplicación te ayudan a repasar los temas de Geometría Analítica que ves en clase?
0%
29%
71%
MUCHO
ALGO
NADA


4. ¿Crees que la aplicación te ayudaría a repasar lo aprendido en clase, en lugares como el transporte, la calle, tu casa, etc.?
14%
36%
50%
MUCHO
ALGO
NADA


5. ¿Te agradó la presentación de la aplicación (temática, colores, diseños)?
0%
21%
79%
MUCHO
ALGO
NADA


6. En general, ¿Utilizarías la aplicación como apoyo de repaso de lo aprendido en clase?
0%
21%
79%
MUCHO
ALGO
NADA