Tabla de Contenido

Tabla de contenido

1. Planteamiento del problema.

Para los estudiantes de diversos niveles, el estudio de las matemáticas forma parte esencial de su preparación, a lo largo de la vida estudiantil se hacen uso de diversas herramientas de apoyo, en el caso de las matemáticas una de ellas es el formulario, que concentra diversas formulas y expresiones matemáticas importantes, normalmente este material se encuentra en hojas de papel, copias de libros de texto y esto presenta un pequeño problema de movilidad, ya que no siempre se tendrá a la mano el formulario, ya sea por olvido o diversas situaciones, con tal de solventar este pequeño problema en la vida estudiantil, se propone el desarrollo de un formulario “virtual” que estará contenido en un smartphone, quitando así la necesidad de portar uno o varios papeles que se utilizan como formulario.

2. Estado del arte.

A pesar de ser un problema “simple y común”, no existen muchas aplicaciones en el mercado que tengan este objetivo o funcionen de manera similar, la mayoría de las aplicaciones que ofrecen características similares tienen como objetivo mas ayudar al calculo de las soluciones matemáticas y no ha servir como una herramienta de apoyo para el estudio de estas.

3. Soluciones existentes.

Dentro del mercado formal, existen pocas aplicaciones similares, y la mayoría se encuentra en ingles, algunas de estas son Math Ref y Math Pro, que además de contar con una interfaz poco llamativa, hacen uso de publicidad que en general causa incomodidad a los usuarios.

4. Solución propuesta (objetivo).

Desarrollar una aplicación en la plataforma Android que tenga la función de almacenar y mostrar formulas matemáticas de diversos temas de la materia, dando en ocasiones ejemplos útiles para el estudio de las matemáticas en general.

5. Recursos

Sera necesario para el desarrollo de este proyecto contar con lo siguiente:

-Equipos de cómputo.

-Equipos celulares con sistema operativo Android.

-Software de desarrollo.

-Programadores.

-Fuentes de información sobre el contenido deseado.

En cuanto a los equipos celulares con los que se realizaron pruebas constantes hasta la primera etapa de desarrollo fueron estos:

-Moto G (2013) Google Edition

Android 5.1 Lollipop®

1 GB de memoria RAM

Procesador Quad-Core a 1.2 GHz

GPU Adreno 305

Pantalla con resolución HD (1280 x 720)

-Sony Xperia ZL

Android 5.1 Lollipop®

2 GB de memoria RAM

Procesador Quad-core a 1.5 GHz

GPU Adreno 320

Pantalla con resolución Full HD

Para la finales de la primera etapa y principios de la segunda se utilizo el siguiente equipo:

-Samsung Galaxy Grand Prime

Android 5.0.2 Lollipop®

1 GB de memoria RAM

Procesador Quad-Core a 1.2 GHz

GPU Adreno 306

Pantalla con resolución qHD

Para la ultima etapa de desarrollo (pruebas y desarrollo de prototipos funcionales) se utilizo el siguiente equipo:

-Moto G Segunda generación (2014)

Android 5.0.2 Lollipop®

1 GB de memoria RAM

Procesador Quad-Core a 1.2 GHz

GPU Adreno 305

Pantalla con resolución HD (1280 x 720)

Se eligieron estos equipos por sus capacidades “comunes” dentro del mercado, el Motorola Moto G fue un móvil de gama media del año 2013 y el Xperia ZL fue un dispositivo de gama alta del año 2012, el Moto G de Segunda generación comparte características con el Moto G (2013) y el Galaxy Grand Prime es un equipo de gama de entrada del 2015, actualmente los equipos de gama media tienen características similares a los dispositivos ya mencionados por lo que se concluyo que estos equipos son los adecuados para la mayoría de las pruebas.

6. Cronograma

En este cronograma se contemplan las 12 semanas para que concluya el semestre, y en las cuales se realizan las actividades de acuerdo al modelo en cascada mejorado, que se menciona más adelante.

Se contempla una semana para requerimientos.

Dos semanas para el diseño de la aplicación.

Alrededor de cinco semanas para la parte de la codificación y pruebas unitarias.

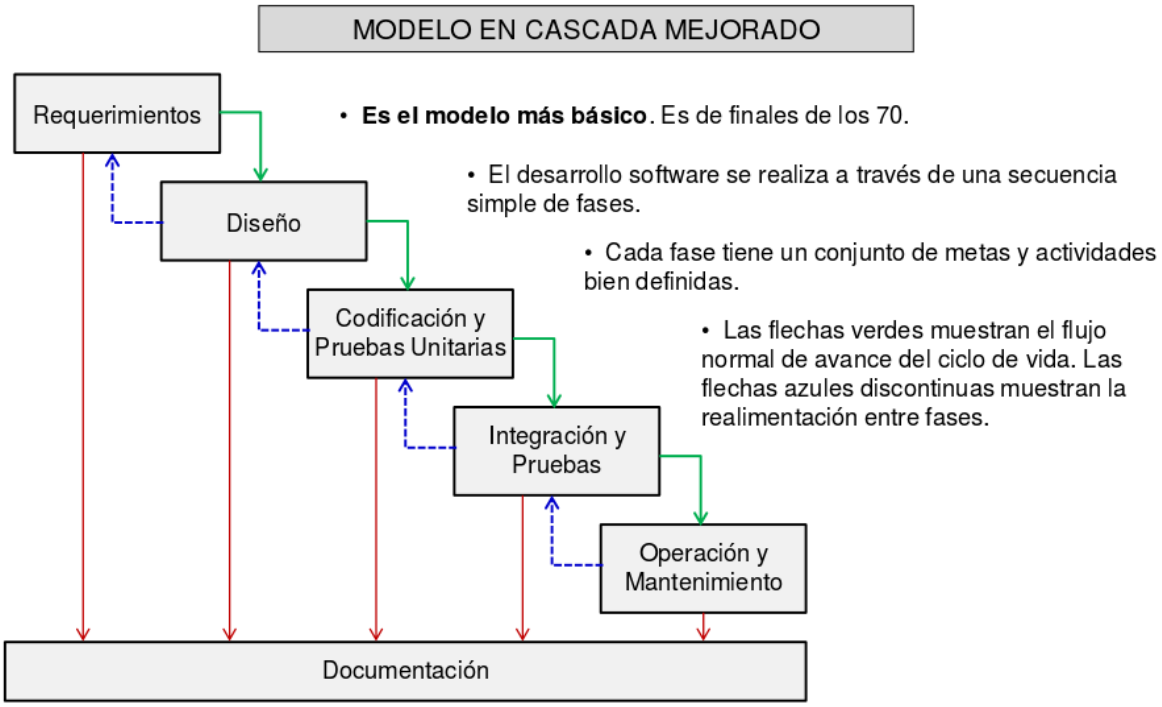
Dos semanas para la integración y más pruebas.

Dos semanas para la operación y el mantenimiento del proyecto.

Para la parte de documentación, de acuerdo al modelo que estaremos trabajando, se tiene que ir documentando a la par de cada módulo del modelo, con lo cual se contemplan las 12 semanas de duración del proyecto como de documentación.

Actividades	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 1	Semana 12
Requerimientos												
Diseño												
Codificación y Pruebas unitarias												
Integración y Pruebas												
Operación y Mantenimiento												
Documentación												

Para el desarrollo de nuestro proyecto utilizaremos una variante del modelo en cascada, llamado “Modelo en Cascada Mejorado” en el que además de los módulos originales del modelo básico se agrega un módulo de documentación en el que se trabaja de modo paralelo con cada uno de los módulos del modelo.



Sus características principales son:

- Es lineal
- Las actividades están relacionadas secuencialmente
- Cada etapa tiene una entrada y una salida
- Es rígido y sistemático: La entrada de una actividad es la salida de la etapa anterior, por lo cual no se puede dar inicio a la siguiente fase.
- Es monolítico: Existe una única fecha de entrega.
- La implementación se pospone hasta que no se comprendan los objetivos.
- Los documentos a entregar rigen el proceso de software
- Las fases que contempla el modelo de la cascada son al Análisis y especificación de requerimientos, diseño, codificación, integración y pruebas, liberación y mantenimiento.

Ingeniería y Análisis del Sistema:

Debido a que el software es siempre parte de un sistema mayor el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

Análisis de los requisitos del software:

El proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

Diseño:

El diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

Codificación:

El diseño debe traducirse en una forma legible para la maquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

Prueba:

Una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Verificación:

Es la fase en donde el usuario final ejecuta el sistema, para ello el o los programadores ya realizaron exhaustivas pruebas para comprobar que el sistema no falle.

Mantenimiento:

El software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a que hayan encontrado errores, a que el software debe adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Herramientas que se utilizan para cada fase:

- Análisis de requisitos: Personal administrativo desde el jefe hasta la persona de menor rango.
- Diseño del Sistema: Arquitectura pura de donde se va trabaja teniendo dependencia a su vez de el hardware.
- Diseño del Programa: Todo el hardware y el software que se usará para el desarrollo del sistema
- Codificación: De igual manera el hardware y el software para desarrollar el programa
- Pruebas: Personal capacitado para realizar las acciones del sistema.
- Verificación: Personal capacitado para verificar que todo esté en orden.
- Mantenimiento: Desarrolladores para la actualización y estabilidad del sistema.

7. Requerimientos funcionales

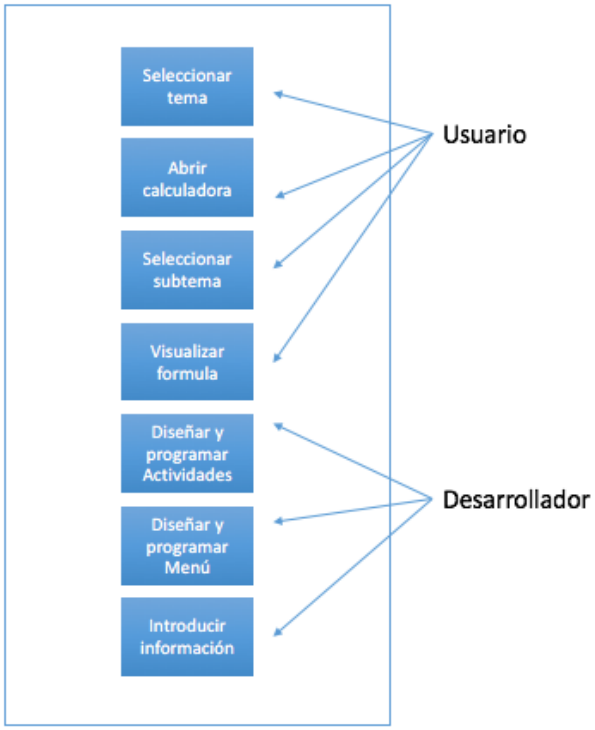
- Menú inicial con listado de temas
- Sub menús ordenados
- Visualización de formulas con imágenes

8. Requerimientos no funcionales

- Fácil de entender
- Acceso rápido a la información requerida
- Con información clara
- Sin configuraciones iniciales innecesarias
- Sin publicidad

9. Diagramas de caso de uso

Para mostrar la relación de los diversos actores con la aplicación, utilizamos el siguiente diagrama:



10. Cuadro de procesos

Actualmente el proyecto se encuentra en desarrollo, en este cuadro de procesos se ilustra el estado de los procesos, a grandes rasgos, que componen el desarrollo de la aplicación.

CUADRO DE PROCESOS		
PROCESO	ESTADO	OBSERVACIONES
PLANEACIÓN	TERMINADO	
TOMA DE REQUERIMIENTOS	TERMINADO	
REUNION DE RECURSOS	TERMINADO	
DISEÑO	TERMINADO	REVISION TERMINADA, A ESPERA DE CORRECIONES
DESARROLLO DEL CONTENIDO	EN PROCESO	DESARROLLO AL 20%
CODIFICACIÓN	EN PROCESO	DESARROLLO DEL PROTOTIPO 2
PRUEBAS	EN PROCESO	PRUEBAS SOBRE EL PROTOTIPO 1 TERMINADAS, PRUEBAS EN EL PROTOTIPO 2 EN PROCESO
CORRECCIONES	EN ESPERA	CORRECIONES DEL PROTOTIPO 1 TERMINADAS

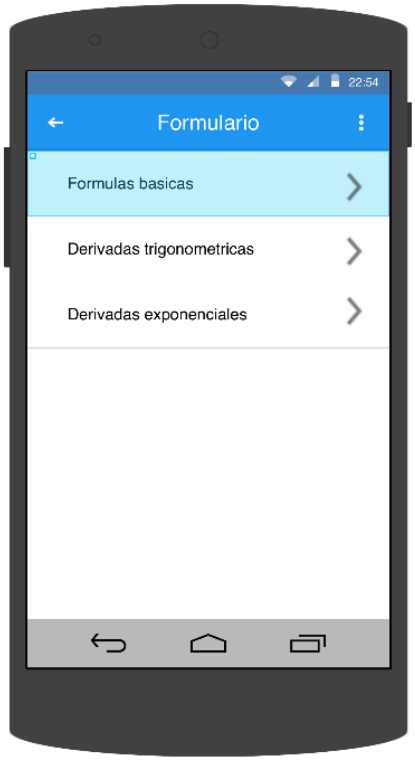
Después de las pruebas y correcciones del prototipo 1, se regreso a las etapas de diseño, y se continuo con el desarrollo del contenido tomando en cuenta el nuevo diseño propuesto, esto supone la necesidad de la realización de las demás etapas de nuevo, por lo que este cuadro ilustra el estado actual del proyecto, mas no las etapas totales por las que ha pasado, las observaciones pueden brindar una idea general de esto.

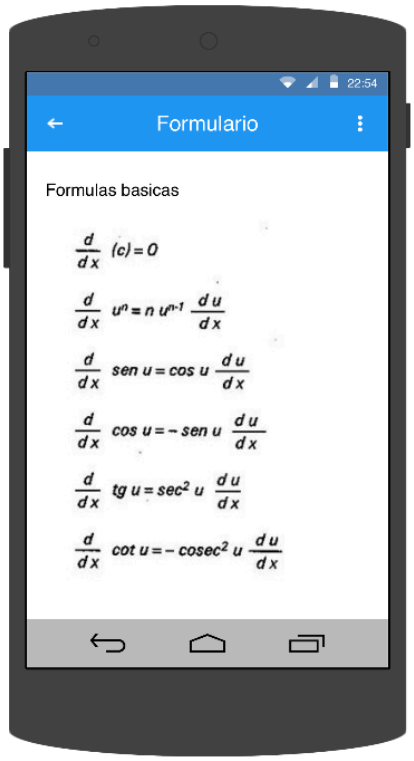
11. Diseño

Para el diseño de la aplicación se tomaron en cuenta las ultimas tendencias seguidas por los desarrolladores actualmente, el uso de la línea de diseño de Material Design propuesta por Google, es decir, la tendencia hacia un diseño minimalista y atractivo visualmente, será una de las características importantes de la aplicación.

Por otro lado, el uso de estas nuevas posibilidades presenta un importante problema de compatibilidad, ya que dispositivos con versiones anteriores a Android 4.1 no serán compatibles con la aplicación.

A continuación se muestran maquetas que ilustran de manera general el aspecto final que se espera conseguir con el desarrollo de la aplicación:





Es importante mencionar que el diseño final difiere un poco de lo mostrado en las maquetas de diseño propuestas, esto es resultado de las pruebas de usabilidad y diseño que se realizo después de la creación del primer prototipo funcional.

11.1 Problemáticas de diseño

Para el desarrollo de una aplicación enfocada a un gran publico, con diversos gustos y puntos de vista, se vuelve un reto el desarrollo de una interfaz grafica que satisfaga tanto los requerimientos funcionales como los no funcionales y que satisfaga a gran parte de la población que será el usuario final, durante el desarrollo de la aplicación se plantearon dos tendencias, que dieron como resultado dos prototipos funcionales.

Uno de ellos se centra en el contenido, dando por hecho que el usuario tiene ciertos conocimientos sobre los temas tratados por la aplicación, por otro lado, el otro prototipo es mas grafico e ilustrativo, presentando problemáticas en crear un diseño uniforme entre las secciones, donde se requiere un esfuerzo mayor en el desarrollo del material (la información).

12. Codificación

12.1 Consideraciones importantes

Existen varias puntos importantes que tienen que tomarse en cuenta para esta sección, primero que nada se tiene que hacer hincapié en que durante el desarrollo del proyecto se obtuvieron dos prototipos funcionales, donde se exploraron las posibilidades de diseño y se corrigieron o cambiaron algunas cosas, particularmente en el diseño.

En esta sección se hablara a grandes rasgos de la forma en que se fueron codificando las diversas partes del proyecto sin dividir o especificar el prototipo en el que se desarrollo tal código ya que los cambios no son lo suficientemente drásticos para requerir esa separación.

12.2 Creación de la interfaz grafica

Para la creación de la interfaz grafica podemos decir que se utilizaron los elementos comunes y clases ya disponibles dentro del SDK de Android, específicamente se hizo uso de los objetos ListView, ScrollView, ImageView, EditText asi como las clases y métodos para la creación de los elementos de la barra de titulo, se utilizaron otras clases y elementos para poder completar la fase de codificación de la interfaz, pero se detallaran mas adelante.

Para cada actividad (activity), se uso la plantilla de Blank Activity proporcionada por Android Studio, para la parte visual para cada actividad se utilizaron dos archivos XML, uno que establecía los parámetros principales de la actividad y otro que especificaba el contenido que muestra cada actividad. En el primer archivo se declaraba la inclusión del segundo, si bien esto puede resultar en una cantidad bastante amplia de archivos la posibilidad futura de utilizar el segundo archivo como plantilla (en caso de la implementación de un base de datos), puede ayudar a agilizar el proceso de escalamiento o actualización del contenido.

```
</android.support.design.widget.AppBarLayout>

<include layout="@layout/content_calculo" />

<android.support.design.widget.FloatingActionButton android:id="@+id/fab"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_gravity="bottom|end" android:layout_margin="16dp"
    android:src="@drawable/abacus6" />
```

Como un pequeño elemento adicional, se añadio un boton flotante que actualmente es muy comun encontrar en aplicaciones con diseño Material, este boton, situado en la parte inferior derecha de la pantalla se utilizara para abrir una calculadora auxiliar, facilitanto asi el acceso a una herramienta externa relacionada con el tema de la aplicación.

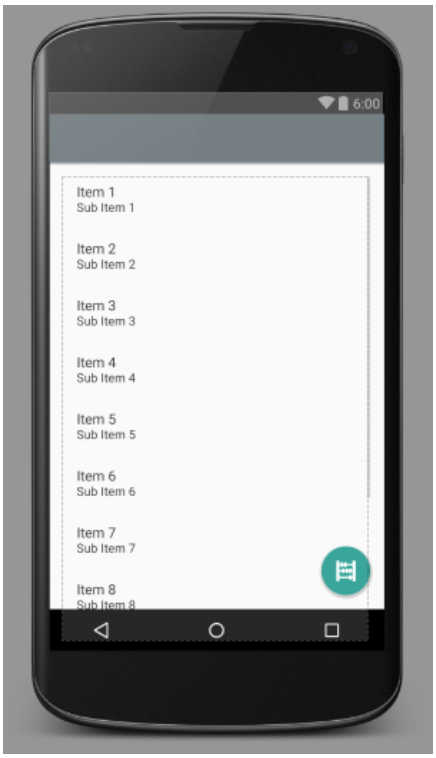
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Perímetro"
    android:id="@+id/textView"
    android:textSize="22sp"
    android:layout_marginTop="200dp"
    android:layout_below="@+id/textView2"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<ImageView
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="36dp"
    android:src="@drawable/p_cuadrado"
    android:adjustViewBounds="true"
    android:scaleType="centerInside" />
```

Para la visualizacion de las formulas se utilizaron elementos proporcionados por el SDK, configurando sus atributos manualmente para mostrarlos en la pantalla.

12.3 Parte lógica

Para la parte lógica se utilizaron clases de apoyo para rellenar la información mostrada por las actividades, a pesar de que la parte lógica esta pensada estrictamente en la manera en que funciona la aplicación, en este caso, tanto la parte lógica como la grafica tienen una relación bastante importante, por un lado, para poder mostrar la información en el ListView, se creo una clase ListViewAdapter que construye las listas y hace posible la visualización de los iconos y texto de los menús de temas que utiliza la aplicación.



Esto facilita la reutilización del código, haciendo mas sencillo el desarrollo de submenús o listas adicionales que se puedan requerir en el futuro, si bien en un principio la clase ListViewAdapter se pensó para todas las listas de la aplicación, al final fue necesaria la utilización de otra clase similar para algunos submenús.

```java
public class TemasListAdapter extends ArrayAdapter<String> {

    private final Activity context;
    private final String[] itemname;
    private final Integer[] integers;

    public TemasListAdapter(Activity context, String[] itemname, Integer[] integers) {
        super(context, R.layout.fila_lista, itemname);
        // TODO Auto-generated constructor stub

        this.context=context;
        this.itemname=itemname;
        this.integers=integers;
    }

    public View getView(int posicion,View view, ViewGroup parent){

        LayoutInflater inflater=context.getLayoutInflater();
        View rowView=inflater.inflate(R.layout.fila_lista,null,true);

        TextView txtTitle = (TextView) rowView.findViewById(R.id.texto_principal);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        TextView etxDescripcion = (TextView) rowView.findViewById(R.id.texto_secundario);

        txtTitle.setText(itemname[posicion]);
        imageView.setImageResource(integers[posicion]);
        if (posicion==0){
            etxDescripcion.setText("Formulas de integración y derivación ");
        }
        if (posicion==1){
            etxDescripcion.setText("Razones trigonométricas y formulas");
        }
        if (posicion==2){
            etxDescripcion.setText("Formulas de perimetros y areas");
        }
        //else {
        //    etxDescripcion.setText("Description " + itemname[posicion]);
        //}
        return rowView;
    }
```

Por otro lado, para poder navegar entre las diversas actividades se utilizaron varios métodos para asegurar una buena experiencia de navegación. Para los elementos y la navegación entre los temas que se muestran en la ListView se utilizaron intents para abrir las otras actividades, en este caso se utilizo una estructura bastante simple que solo declaraba un intent en una instancia y después lo iniciaba, en este caso, en vez de programar cada intent para cada elemento, se utilizo una estructura if para comprobar que elemento de la lista había sido presionado y según el lugar en el arreglo (array) local se llamaba a la siguiente actividad por medio del intent.

```
TemasListAdapter adapter=new TemasListAdapter(this,temas,imgid);
lista=(ListView)findViewById(R.id.mi_lista);
lista.setAdapter(adapter);
lista.setOnItemClickListener((parent, view, position, id) → {
        String Selecteditem = temas[+position];
        Toast.makeText(getApplicationContext(), Selecteditem, Toast.LENGTH_SHORT).show();
        if (Selecteditem==temas[0]){
            Intent intent = new Intent(getApplicationContext(),calculo.class);
            startActivity(intent);
        }
        if (Selecteditem==temas[1]){
            Intent intent = new Intent(getApplicationContext(),trigonometria.class);
            startActivity(intent);
        }
        if (Selecteditem==temas[2]){
            Intent intent = new Intent(getApplicationContext(),areasperimetros.class);
            startActivity(intent);
        }

});
```

Para regresar a la actividad anterior, Android automáticamente hace posible esto por medio de la barra de navegación, pero aunque esto funciona sin ningún problema, existen en Material Design otra posibilidad, que aparte de añadirle una funcionalidad mas a la aplicación, agrega un elemento visual atractivo y que favorece a la usabilidad de la aplicación. En cada actividad que muestra un subtema o que no es la actividad principal se le declaro una actividad padre, esto, sumado a un pequeño fragmento de código en la clase principal de la actividad hace posible la utilización de una flecha de retroceso en la barra de titulo.

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

12.3.1 Posibilidades futuras

Para este proyecto se puede considerar varias posibilidades futuras en cuanto a la parte lógica se refiere, al contener una gran cantidad de información, en algún punto la actualización se volverá compleja y difícil de lograr con eficiencia, por lo que se propone la consideración de la posible implementación de una base de datos local para almacenar la información, si bien existe código XML que tiene a grandes rasgos considerada esta posibilidad, no existe ninguna clase o parte lógica que tenga esta posibilidad contemplada.

Por otro lado se propone la implementación de hilos para la carga de las imágenes en las actividades que lo requieran, esto se volverá importante si se llevara a cabo la implementación de la base de datos, ya que con el uso de hilos para la carga de las imágenes, se pueden evitar cierres o errores debido a los tiempos de espera prolongados en terminales poco potentes o a errores en la conexión a la base de datos, esto, junto con la base de datos, dotaran a la aplicación de mayor robustez, asi como una mayor capacidad para almacenar información.

La implementación de una base de datos supone ciertos retos que actualmente implican mas costo que beneficio, el desarrollo y codificación de una base de datos con imágenes supone esfuerzos, que para el tamaño actual del proyecto, no son necesarios y que sin embargo causarían un retraso importante en los tiempos establecidos al inicio del proyecto.

12.4 Prototipo funcional 1

Para el desarrollo del primer prototipo funcional se utilizo información disponible en internet sobre los temas a tratar en la aplicación, despues de recopilar una muestra de la información, se codifico y adapto la interfaz a dicho contenido, sin embargo los resultados no fueron uniformes y no se logro una visualizacion conjunta adecuada según las metas planteadas por el proyecto para el diseño de la interfaz.

12.4.1 Vistas de operación

A continuacion se presentan capturas de pantalla de la apliación en ejecución.

12.5 Prototipo funcional 2

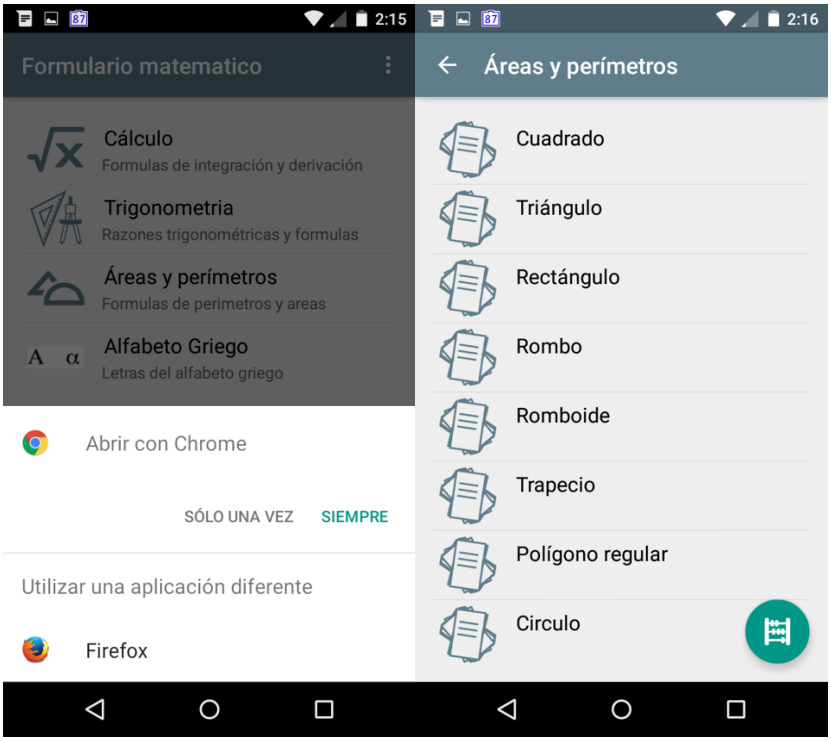
Después de realizar el primer prototipo y teniendo en cuenta los resultados obtenidos, se llego a la conclusión de que la necesidad de un aspecto visual uniforme era primordial para la buena aceptación de la aplicación, por supuesto, tomando en cuenta los lineamientos de usabilidad y rendimiento esperados y proyectados en la planeación del proyecto.

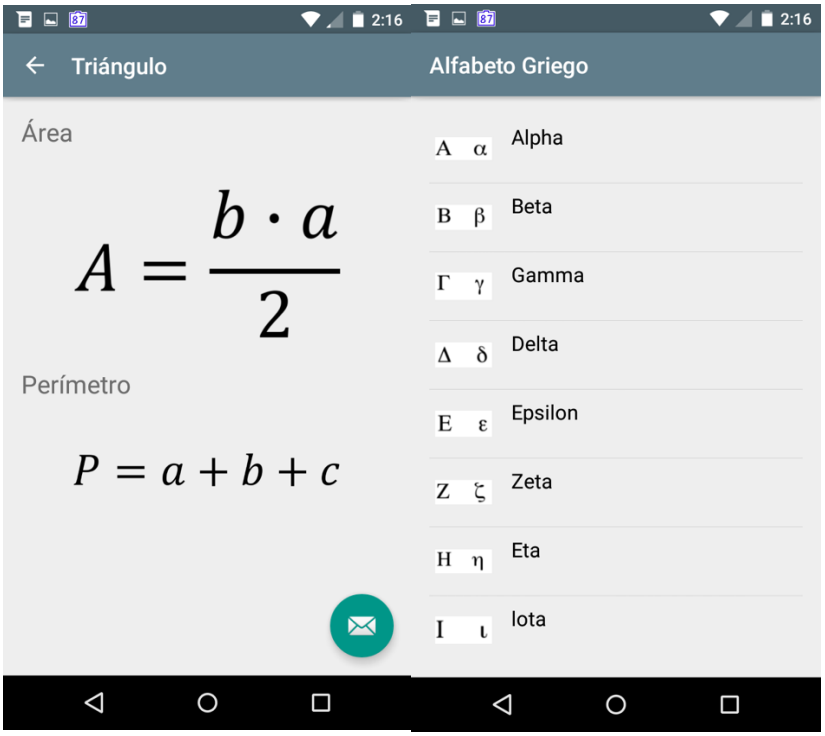
Para tal efecto, se realizo una muestra de imágenes con las formulas, elaboradas una por una, en formato PNG, con fondo transparente y tipografía y tamaño iguales, para posteriormente agregar a las actividades y así lograr un aspecto de uniformidad entre las diversas secciones que componen la aplicación.

De esta forma se obtuvo un nuevo prototipo, con un diseño mas sobrio y uniforme, donde también se incluían algunos detalles que no se habían tomado en cuenta en el primer prototipo, se propone continuar con la misma línea de diseño para completar la aplicación y llegar a la fase de pruebas con un prototipo terminado basado en este prototipo funcional inicial.

12.5.1 Vistas de operación

A continuacion se presentan capturas de pantalla de la aplicación en ejecución.





13. Plan de Pruebas

13.1 Propósito

El propósito del plan de pruebas planteado en este documento, es permitir definir los lineamientos a seguir para realizar la planeación de la etapa de pruebas sobre el proyecto “Aplicación móvil: Formulario matemático”, planteando una estrategia que conduzca al objetivo enfocado en el aseguramiento de calidad del software.

El propósito del Plan Maestro de Pruebas es:

- Proveer un artefacto central que gobierne la planeación y control del esfuerzo de pruebas. Este define el enfoque general que será empleado para probar el software y para evaluar los resultados de esas pruebas, y es el plan de más alto nivel que será usado por los administradores para guiar y dirigir el trabajo de pruebas detallado.
- Proveer visibilidad a los interesados en el esfuerzo de pruebas que han tenido las consideraciones adecuadas para varios aspectos que orientan el esfuerzo de pruebas, y dónde es apropiado que los interesados aprueben el plan.
- Este Plan Maestro de Pruebas también soporta los siguientes objetivos específicos:
    - Identificar los ítems que serán objeto de las pruebas.
    - Enmarcar la metodología de pruebas que será utilizada
    - Identificar los recursos requeridos y proveer un estimado del esfuerzo de las pruebas.
    - Elaborar un listado de los elementos entregables del plan de pruebas.

13.2 Alcance

El plan maestro de pruebas describe el detalle de las diferentes pruebas a ser aplicadas, así como también las herramientas y metodologías a utilizar en cada una de estas. Las pruebas que serán realizadas son:

- Revisión de la documentación: Consiste en revisar la calidad y completitud de los documentos insumo y casos de uso para la ejecución de las pruebas.
- Pruebas Unitarias: Se validaran las piezas individuales del software como una unidad independiente, bucles, condicionales, etc.
- Pruebas de integración: Se validara la integración entre los diferentes módulos que componen la solución con el fin de garantizar que su operación integrada es correcta.
- Pruebas Funcionales o de Procedimientos: Se validaran los procesos, reglas de negocio establecidas y los requerimientos funcionales.
- Pruebas de sistema: Las pruebas de sistema se determinarán en el momento que el Outsourcing de Desarrollo entregue el documento de Requerimientos no funcionales, y así determinar que tipos de prueba se realizarán y a que casos de uso se aplicarán.

- Pruebas de regresión: Se validara que el sistema mantenga su correcta funcionalidad debido a la incorporación de un ajuste, corrección o nuevo requerimiento.

  Adicionalmente y con el fin de centrar el plan de pruebas en ciertos factores que son críticos y de mayor relevancia para el proyecto, se determinan los tipos de pruebas que se realizarán para el proyecto, diseñando los factores de calidad y las pruebas especializadas para alcanzar estos atributos del software entregado. Con esta misión se identifican de acuerdo a las especificaciones del cliente los factores

  Para este proyecto de acuerdo a los requerimientos, se definen los siguientes factores en los que se enfocarán las pruebas:

  - Corrección.
  - Conformidad.
  - Facilidad de Uso.
  - Portabilidad.
  - Facilidad de Operación.

13.3 Audiencia

Este plan maestro de pruebas esta dirigido a todas aquellas personas involucradas en la planeación, aprobación y ejecución del mismo.

14. Misión de las Pruebas

14.1 Contexto del Proyecto y Antecedentes

Se pretende realizar un levantamiento y análisis de información de los procesos de gestión de solicitudes del sistema propuesto con el fin de plantear una arquitectura de solución tecnológica con el fin de optimizar eficiencia, tanto a nivel técnico como funcional de estos procesos de negocio que constituyen y representan valor en las objetivos estratégicos de la organización.

14.2 Misión de las Pruebas aplicable a este proyecto

La misión de la evaluación para el presente proyecto se define enfocada al aseguramiento de la calidad de los componentes y artefactos tecnológicos desarrollados, de manera que estos cumplan con la especificación de los requerimientos del cliente. Para esto se definen los siguientes lineamientos que constituyen la misión y objetivos dentro este esfuerzo de pruebas:

- Descubrir tantos errores como sea posible
- Notificar acerca de los riesgos percibidos del proyecto
- Examinar la aplicación para comprobar si hace o no lo que se supone, debe hacer. De igual forma verificar si ésta hace o no lo que se supone, no debe hacer.
- Validar y Verificar a través de la comparación del resultado de las pruebas del aplicativo con el resultado que el mismo tendría que producir de acuerdo a su especificación.

- Evaluar la calidad del producto y satisfacción de los interesados
- Cumplir con los requerimientos del cliente

El proceso de evaluación y pruebas debe permitir detectar problemas desde el inicio de la especificación de requerimientos, antes de que sean de gran impacto en fases más adelantadas del proyecto, esto con el fin de disminuir los riesgos y de obtener un producto con calidad logrando mayor satisfacción del cliente.

14.3 Motivadores de las Pruebas

Dentro de los principales motivadores de pruebas del proyecto, están, la necesidad del cliente de optimizar y gestionar la ejecución de sus procesos de negocio, verificar la confiabilidad de la información que posee sobre sus clientes y dar trámites ágiles y efectivos a las solicitudes que ellos generan a la organización

Adicionalmente existen unos motivadores puntuales que van a contribuir a que se construya un software que satisfaga los requerimientos del cliente de la manera más óptima posible y siguiendo un proceso adecuado para conseguirlo. Estos son:

- Aseguramiento de la calidad.
- Solicitudes de cambios.
- Riesgos de calidad.
- Verificación de los casos de uso.
- Comprobación de los requerimientos funcionales y no funcionales.

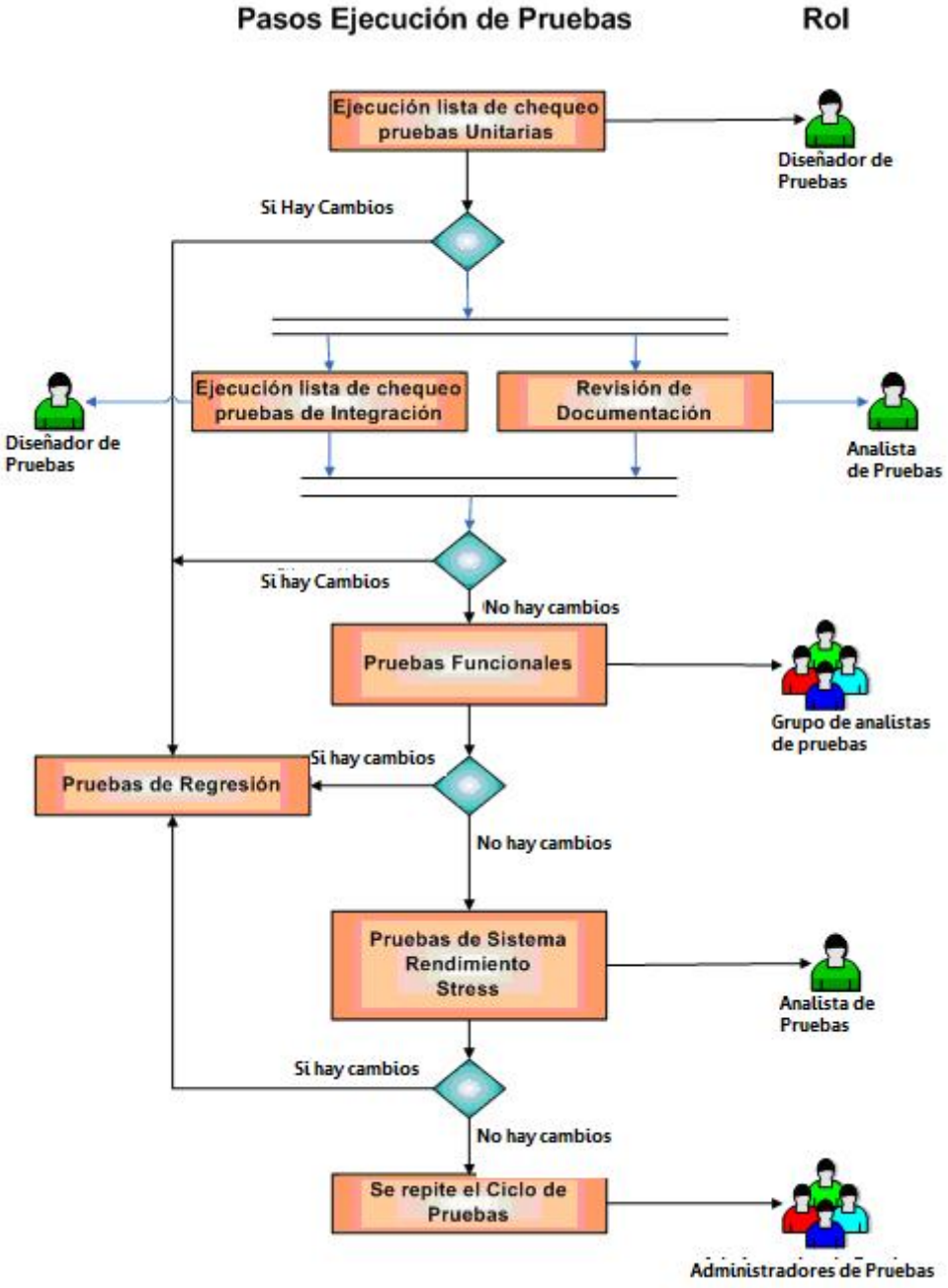
15. Elementos Objetivo de Pruebas

A continuación se listan los elementos (artefactos, entregables, documentos etc.) que seran objeto de prueba dentro del esfuerzo de pruebas:

Fase Inicial

- Documentación
- Especificación de Requerimientos
- Estimaciones
- Modelos - Diagramas

15.1 PANORAMA DE PRUEBAS PLANEADAS



16. ENFOQUE DE LAS PRUEBAS

El plan de pruebas se basará en su totalidad en pruebas funcionales, instalación, regresión y otras teniendo en cuenta los requerimientos no funcionales.

Revisión de la documentación: La estrategia para realizar estas pruebas, consiste en la revisión de la documentación y casos de uso verificando su completitud y concordancia en la información que se encuentra en ellos.

- Pruebas unitarias: Las estrategias para realizar estas pruebas consiste en generar casos de prueba necesarios:
    - Para que cada sentencia o instrucción del programa se ejecute al menos una vez correctamente.
    - Para que cada condición tenga por lo menos una vez un resultado verdadero y al menos una vez uno falso.
    - Para probar varias veces el mismo bucle (en donde aplique) considerando los siguientes casos: Ignorar el bucle, pasar una vez, pasar dos veces, pasar n veces, pasar n-1 veces y n+1 veces.

- Pruebas funcionales o de procedimientos: La estrategia para realizar estas pruebas consiste en la elaboración y ejecución de Set de Pruebas, teniendo en cuenta flujo normal y flujos alternativos, usando datos validos e inválidos que permitan verificar lo siguiente:
    - Los resultados esperados ocurren cuando se usan datos validos.
    - Se despliegan mensajes de error cuando se usan datos inválidos.
    - Cada regla de negocio es propiamente aplicada.

- Pruebas de Regresión: La estrategia para realizar estas pruebas consiste en repetir las pruebas (funcionales y de carga) ejecutadas antes de corregir defectos o de añadir nuevas funcionalidades, para comprobar que las modificaciones no provocan errores donde antes no los había.

16.1 Pruebas de Aceptación

La pruebas de aceptación se basarán en su totalidad en pruebas funcionales, instalación, y otras teniendo en cuenta los requerimientos funcionales las pruebas. Adicionalmente estas pruebas serán de caja negra.

- Pruebas funcionales o de procedimientos: La estrategia para realizar estas pruebas consiste en la elaboración y ejecución de Set de Pruebas, teniendo en cuenta flujo normal y flujos alternativos, usando datos validos e inválidos que permitan verificar los

casos de prueba descritos en el diagrama de casos de uso, asi como establecidos en la retroalimentación con el cliente. Estos casos de prueba son aprobados por el cliente.

5.2.1 Técnicas y Herramientas de Prueba

A continuación se exponen las herramientas y técnicas que se usaran para llevar a cabo las pruebas enfocadas a la mitigación de los riesgos anteriormente planteados:

Factor de Prueba:	Conformidad	Técnica:	Pruebas de operación
Descripción: Con las pruebas de operación se garantiza que el usuario está bien capacitado en el manejo del software y además se lleva un registro para guardar los caminos no contemplados dentro de las pruebas previas del software, y con ello se tomarán las medidas adecuadas.			

Factor de Prueba:	Facilidad de Uso	Técnica:	Walkthroughs
Descripción: Se debe incluir al cliente y/o usuario final con un role de evaluador durante sesiones de walkthroughs en las cuales se discutirán los escenarios de calidad referentes a la usabilidad del software.			

Factor de Prueba:	Facilidad de Operación	Técnica:	Pruebas de Requerimientos
Descripción: Validar los requerimientos no funcionales de ambiente recolectados con el cliente versus las características requeridas por el ambiente de producción.			

16.2 Pruebas de Integración

Las pruebas de integración que se realizaran durante el proceso de desarrollo de los componentes de software, deben seguir las siguientes políticas y lineamientos de ejecución:

- Se tiene una fase de pruebas unitarias competa y aprobada para el inicio de las pruebas de integración.
- Se seguirá el enfoque “Bottom-UP” para la ejecución de estas pruebas, es decir, se

probaran en primer lugar los componentes o módulos individuales del software y posteriormente y de manera progresiva se Irán agrupando hacia arriba y de manera funcional estos componentes para probar escenarios que impliquen varias funcionalidades de interacción entre los componentes, y se continuará así hasta llegar al nivel más alto de funcionalidad e integración.

- Para la ejecución de estas pruebas se utilizarán las siguientes técnicas:

Finalmente y como criterio de aceptación para esta fase de las pruebas se realizará un piloto funcional de la solución construida, para el cual se debe generar un Set de casos de prueba que abarquen la mayor cantidad de interacciones que impliquen comunicación e integración entre los diferentes componentes del software, y en el deben participar tanto los usuarios finales como los desarrolladores.

17. CRITERIOS DE ENTRADA Y SALIDA

17.1 Criterios de Entrada del Plan Maestro de Pruebas

- Set de pruebas completo y claro.
- Claridad en el procedimiento para el desarrollo de las pruebas.
- Tener un entorno de pruebas adecuado.
- Toda la documentación requerida para la realización de las pruebas debe estar disponible.

17.2 Criterio de Salida del Plan Maestro de Pruebas

- Que todos los set de pruebas diseñados para cada caso de uso se ejecuten de manera exitosa, cumpliendo los criterios de aceptación definidos para cada uno.

17.3 Criterios de suspensión y Reanudación

- Una característica principal tiene un error que impide probar un área importante.
- El entorno de pruebas no es lo suficientemente estable como para confiar en los resultados.
- El entorno de pruebas es muy diferente del entorno de producción.
- No se puede instalar la nueva versión o un componente

17.4 Requisitos de reanudación

- Existe consenso en el equipo acerca de la solución del problema que supuso la suspensión de las pruebas.

18. Necesidades de Ambiente

18.1 Hardware Base para ambiente de pruebas

La siguiente tabla relaciona los recursos de hardware que son necesarios para crear un ambiente inicial de pruebas en este proyecto

Equipo	Procesador	Memoria	RAM	Aplicación a Instalar
Android 4.3 o superior	Dos núcleos 1 GHz	50 mb libres	512 GB	Aplicación a probar

18.2 Software Base para ambiente de pruebas

El software necesario en el equipo de pruebas es:

- Android 4.3 o superior

19. RESPONSABILIDADES Y EQUIPO DE TRABAJO

19.1 Personas y Roles

Esta tabla muestra el personal supuesto para el esfuerzo de pruebas.

Recursos Humanos	
Rol	Responsabilidades Específicas o Comentarios
Administrador de Pruebas	• Administra el esfuerzo de las pruebas, aprueba los criterios de entrada y salida a las pruebas, monitorea avance del esfuerzo de pruebas, aprueba los casos de prueba, gestiona el alcance y misión de las pruebas, Certifica el nivel de calidad del producto construido.
Diseñador de Pruebas	• Es el responsable de diseñar los set de pruebas (estructura y enfoque) que se realizarán al sistema para una certificar que se construyó un producto que satisface los requerimientos definidos.
Analista de Pruebas	• Es el responsable de ejecutar los casos de prueba y realizar los reportes correspondientes sobre esta ejecución. • Realizar documentación técnica de las pruebas.

20. Riesgos de las pruebas

A continuación se expone una matriz en la cual se relacionan los 5 factores de prueba más críticos para el proyecto con los riesgos identificados para cada uno de ellos vs. las fases en las que se ejecutan las pruebas.

Factor de Prueba	Requerimientos	Diseño	Software
Conformidad	• Pasar por alto la prueba de requerimientos no funcionales clave que impliquen un gran impacto en la arquitectura propuesta.	• Alta complejidad en el diseño de las pruebas que evidencien la conformidad con los requerimientos de gobernabilidad y reusabilidad	• Omitir la ejecución de pruebas en las características menos comunes de utilización.
Portabilidad	• Identificar tardíamente problemas de compatibilidad con plataformas externas de alto riesgo o costo.	• No contar con la tecnología necesaria para probar aspectos del diseño enfocados a comprobar el bajo acoplamiento de la solución.	• No cubrir en las pruebas una cantidad representativa de plataformas que deben ser compatibles con la solución a futuro.
Facilidad de Uso	• No lograr captar la opinión de los usuarios finales para determinar los aspectos de facilidad de uso que ellos esperan.	• Realizar las pruebas con un enfoque muy técnico sin detectar aspectos que por diseño supongan complejidades altas en el uso del software	• Probar solo funcionalidades sin identificar problemas o mejoras en la facilidad de utilización del software
Facilidad de Operación	• No incluir en las listas de chequeo de comprobación de los requerimientos, los aspectos relacionados con la facilidad de operación, por desconocimiento en los mismos	• No detectar a tiempo aspectos del diseño que se conviertan en impedimentos para permitir las fácil instalación y administración de las solución	• Detectar tardíamente problemas relacionados con la instalación y operación del software
Corrección	• No Encontrar requerimientos en una fase temprana con algún nivel de ambigüedad.	• No Identificar problemas para corregir defectos detectados en una fase avanzada del desarrollo.	• Presencia de errores en el producto que sean muy costosos de corregir cuando este ya se encuentre finalizado.

21. Bibliografía

- http://articulos.softonic.com/crear-aplicaciones-moviles-panorama
- http://somosprogramacion.blogspot.mx/2015/04/video2brain-desarrollo-de-apps-para-android-con-material-design-2015.html
- http://www.cristalab.com/tutoriales/material-design-en-versiones-anteriores-a-lollipop-c114485l/
- http://androideity.com/2011/09/20/trabajando-con-threads-en-android-ii-runnable/
- http://academiaandroid.com/ejecucion-tareas-segundo-plano-android/
- http://www.intertech.com/Blog/android-intents-for-app-integration-call-a-calculator-play-video-open-an-editor/