## **Introducción**

En el presente documento vamos a abordar los principales aspectos del proyecto Área de un triángulo, donde se representara las distintas tablas de la estructura PSP0.

En un primer paso se presentarán los puntos acerca el análisis que contiene este proyecto, así como el diagrama de Gantt, sus requerimientos y diagrama de casos de uso.

Después se continua exponiendo las acciones que cada integrante realizo por medio de las tablas del PSP0, en el que el análisis fue individual y el desarrollo fue por parte de cada uno correspondiendo a las tareas asignadas y en base a los tiempos de control que se manejaron en el controlador de versiones Github.

Para la finalización de este documento se expone el resumen plan de proyecto, donde se especifica cada avance del proyecto, por medio de tiempos contemplados por el equipo y tiempos que fueron reales, como también la codificación de esta. También cuanta con unas conclusiones exclusivas de cada integrante que desarrollo este documento, para dar énfasis a las partes del proyecto.

## **Planeación**

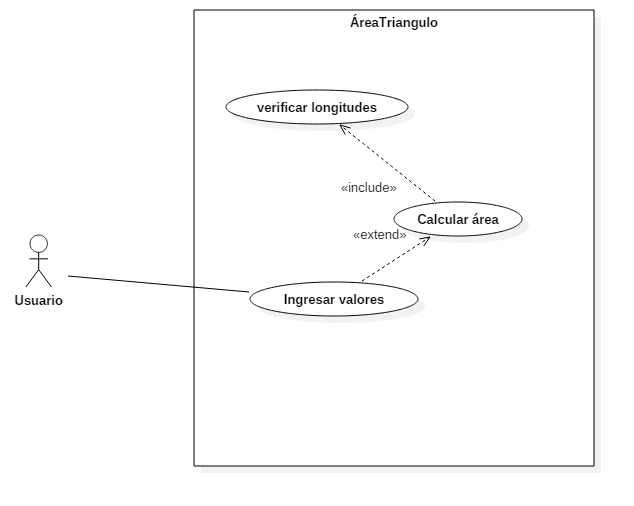
### C:\Users\electronica 63\Desktop\CalidadDeSoftwareMC\Documentación\Diagrama de gantt.pngDiagrama de Gantt:

## **Requerimientos**

### Listado de requerimientos:

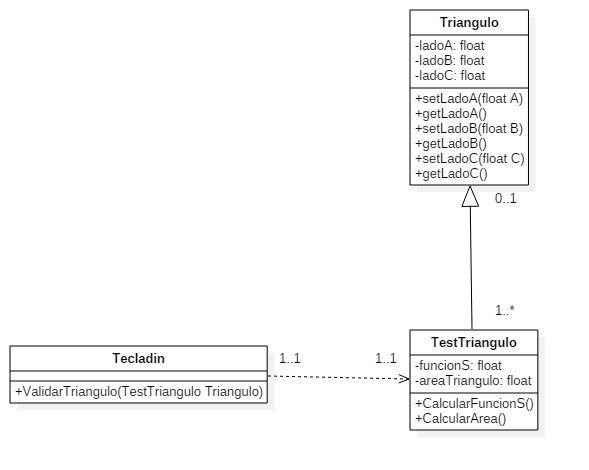
* RF001: Calcular el área.
* RF002: Validar si es un triángulo.
* RF004: Ingresar lados del triángulo.
* RNF001: Implementar las clases triangulo.java, test triangulo.java y tecaldin.java.

### Diagrama de casos de uso:



## **Diseño.**

### Diagrama de clases:



## **Cuaderno de registro de tiempos.**

**José Luis Puc Chan:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Tiempo | | | | | | |
| **Fecha** | **Inicio** | **Fin** | **Interrupción** | **Tiempo** | **Actividad** | **Comentarios** |
| 15/2 | 5:52 | 6:54 | 5 | 57 | Codificación | Codificar las clases con sus métodos. |
| 15/2 | 7:00 | 8:42 |  | 1:42 | Revisión | Revisión de la sintaxis y estructura de las clases. |
| 15/2 | 7:00 | 8:42 |  | 15 | Cambios | Corrección diagrama de clases. |

**André Salazar Figueroa:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Tiempo | | | | | | |
| **Fecha** | **Inicio** | **Fin** | **Interrupción** | **Tiempo** | **Actividad** | **Comentarios** |
| 14/0217 | 9:40 | 9:46 |  | 6 | Creación | Creación del repositorio único para el proyecto. |
| 15/02/17 | 5:52 | 6:54 |  | 62 | Creación | Creación del diagrama de clases. |
| 16/02/17 | 12:00 | 1:14 |  | 74 | Codificación | Codificación del proceso principal del proyecto. |
| 16/02/17 | 2:00 | 2:04 |  | 4 | Modificación | Agregue una parte a la documentación. |

**Diosemir Isael Nah May:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Tiempo | | | | | | |
| **Fecha** | **Inicio** | **Fin** | **Interrupción** | **Tiempo** | **Actividad** | **Comentarios** |
| 15/02/17 | 5:35 pm. | 5:40 pm. |  | 5 | Creación | Creación del proyecto en Android Studio. |
| 15/02/17 | 5:40 | 5:52 |  | 12 | Diseño | Implementación del diseño del proyecto. |
| 15/02/17 | 11:15 | 11:54 |  | 39 | Modificación | Modificación de los requerimientos funcionales y no funcionales. |
| 15/02/17 | 11:54 | 12:18 |  | 24 | Agregación | Estructura de la documentación en Word. |
| 16/02/17 | 01:14 | 1:34 |  | 20 | Testeo | Verificación del proyecto en Android Studio y modificación de colores. |

**Aarón Jesús Peña Martin:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Tiempo | | | | | | |
| **Fecha** | **Inicio** | **Fin** | **Interrupción** | **Tiempo** | **Actividad** | **Comentarios** |
| 15/02/17 | 06:00 pm | 06:45 pm |  | 45 | Creación | Creación del diagrama de casos de uso. |
| 15/02/17 | 06:35 pm | 7:51 pm | 10 | 66 | Creación | Detección de requisitos. |
| 15/02/17 | 9:00 pm | 10:00 pm |  | 60 | Codificación | Codificación de clases. |

**Yair Roberto Vega Gamboa:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Tiempo | | | | | | |
| **Fecha** | **Inicio** | **Fin** | **Interrupción** | **Tiempo** | **Actividad** | **Comentarios** |
| 15/02/17 | 06:54 pm | 07:26 pm |  | 32 | Codificación | Continuidad en la codificación de las clases. |
| 15/02/17 | 07:26 pm | 07:49 pm |  | 5 | Codificación | Creación de la clase Tecladin.java |

## **Cuaderno de registro de defectos.**

**José Luis Puc Chan:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Defectos | | | | | | |
| Programa: Área de un triángulo. | | | | | Fecha: 15/02/17 | |
| Fecha | Número | Tipo | Introducido | Eliminado | Tiempo de corrección | Defecto corregido |
| 15/02/17 | 1 | 20 | Codificación | Compilación | 1 |  |
| Descripción: | Defecto en la declaración de las variables deben tratarse como float. | | | | | |
| 15/02/17 | 1 | 40 | Análisis | Prueba | 1 |  |
| Descripción: | Corrección de interacción del usuario con el sistema. | | | | | |

**André Salazar Figueroa:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Defectos | | | | | | |
| Programa: Área de un triangulo | | | | | Fecha: 15/02/17 | |
|  | Número | Tipo | Introducido | Eliminado | Tiempo de  corrección | Defecto  corregido |
| Descripción: | Mala asignación de variables. | | | | | |
|  | 1 | 40 | Codificación | Compilado | 30 minutos |  |
| Descripción: | Sistema desactualizado. | | | | | |
|  | 2 | 90 | Sistema | Prueba | 25 minutos |  |

**Diosemir Isael Nah May:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Defectos | | | | | | |
| Programa: Área de un triángulo | | | | | Fecha: 15/02/17 | |
|  | Número | Tipo | Introducido | Eliminado | Tiempo de  corrección | Defecto  corregido |
| Descripción: | Se eliminó una etiqueta de estilos | | | | | |
|  | 1 | 60 | Diseño | Compilación | 15 minutos |  |
| Descripción: | El desorden de los TextField a la hora de la virtualización del proyecto. | | | | | |
|  | 2 | 100 | Diseño | Compilación | 10 minutos |  |
| Descripción: | Cambios en cuanto a diseño general. | | | | | |
|  | 3 | 100 | Diseño | Compilación | 30 minutos |  |

**Aarón Jesús Peña Martin:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Registro de Defectos | | | | | | |
| Programa: Triangulo | | | | | Fecha:15/02/17 | |
| Fecha | Número | Tipo | Introducido | Eliminado | Tiempo de  corrección | Defecto  corregido |
| 15/02/17 | 1 | 10 | Documentación | Compilación | 10 |  |
| Descripción: | Mala definición de la documentación de casos de uso | | | | | |
| 15/02/17 | 2 | 10 | Documentación | Compilación | 1 |  |
| Descripción: | Mala estructura en el diagrama de clases | | | | | |

**Yair Roberto Vega Gamboa:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Registro de Defectos | | | | | | | |
| Programa: Área de triangulo. | | | | | | Fecha:15/02/17 | |
| Fecha | Número | Tipo | Introducido | | Eliminado | Tiempo de corrección | Defecto corregido |
| 15/02/17 | 1 | 20 | Sintaxis | | Revisión | 5 | Comentarios en la clase |
| Descripción: | Ortografía Errónea en los comentarios de la clase testTriangulo | | | | | | |
| 15/02/17 | 2 | 20 | Sintaxis | Análisis | | 10 | Sintaxis en la Variables |
| Descripción: | Sintaxis errónea de las variables (el estándar) | | | | | | |
|  | 3 | 20 | Sintaxis | Revisión | | 5 | Comentarios en la clase |
| Descripción: | Ortografía Errónea en los comentarios de la clase Tecladin | | | | | | |

## **Estándar Tipos de defectos.**

|  |  |  |
| --- | --- | --- |
| **Tipos de defectos** | | |
| **Nº de tipo** | **Nombre del tipo** | **Descripción** |
| 10 | Documentación | Comentarios, mensajes |
| 20 | Sintaxis | Ortografía, puntuación, erratas, formato de las instrucciones |
| 30 | Construir, paquetes | Gestión del cambio, librerías, control de versión |
| 40 | Asignación | Declaración, nombres duplicados, ámbito, límites |
| 50 | Interfaz | Llamadas a procedimientos y referencias, E/S, formatos de usuario |
| 60 | Chequeo | Mensajes de error, chequeos inadecuados |
| 70 | Datos | Estructura, contenido |
| 80 | Función | Lógica, punteros, bucles, recursión, computación, defectos de la función |
| 90 | Sistema | Configuración, temporización, memoria |
| 100 | Entorno | Diseño, compilación, pruebas y otros problemas que soporta el sistema |

## **Resumen Plan del proyecto.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Programa:** Área de un triángulo | | | | | | | | **Fecha:** 15/02/2017 | | |
| **Descripción:** Software encargado de calcular el área de un triángulo realizando múltiples operaciones, tales como validar el triángulo, calcular el área en base a la función de una variable. Está desarrollado en el lenguaje Java en la plataforma Android Studio. | | | | | | | | | | |
| **Resumen** | **Plan** | | | **Real** | | | | **Hasta la fecha** | | |
| Minutos/LOC | 1.6 | | | 1 | | | | 2.6 | | |
| LOC/Hora | 95 | | | 60 | | | | 155 | | |
| Defectos/KLOC | - | | | 13 | | | | 13 | | |
| Rendimiento | - | | | - | | | |  | | |
| Valoración/Fallo | - | | | - | | | |  | | |
| **Tamaño programa (LOC)** | **Plan** | | | **Real** | | | | **Hasta la fecha** | | |
|  | | |  | | | |  | | |
| Total nuevo & cambiado | 100 | | | 150 | | | | 130 | | |
| Tamaño máximo | 300 | | |  | | | | | | |
| Tamaño mínimo | 100 | | |
| **Tiempo por Fase (min.)** | **Plan** | | **Real** | | | **Hasta la fecha** | | | **% Hasta la fecha** | |
| Planificación | 50 | | 120 | | | 100 | | | 30% | |
| Diseño | 80 | | 90 | | | 120 | | | 50% | |
| Codificación | 180 | | 200 | | | 170 | | | 60% | |
| **Revisión del código** |  | | | | | | | | | |
| Compilación | 10 | | 20 | | | 30 | | | 15% | |
| Pruebas | 50 | | 60 | | | 10 | | | 20% | |
| Postmorten | 60 | | 65 | | | 100 | | | 5% | |
| Total | 430 | | 555 | | | 530 | | |  | |
| Tiempo máximo |  | |  | | | | | | | |
| Tiempo mínimo |  | |
| **Defectos Introducidos** | **Plan** | **Actual** | | | **Hasta la fecha** | | **% Hasta la fecha** | | | **Def./Hora** |
| Planificación | - | 50 | | | 50 | | 10% | | | 0.2 |
| Diseño | - | 10 | | | 10 | | 15% | | | 0.1 |
| Codificación | - | 15 | | | 15 | | 20% | | |  |
| Revisión del código | - | 30 | | | 30 | | 22% | | |  |
| Compilación | - | 20 | | | 20 | | 25% | | |  |
| Pruebas | - | 10 | | | 10 | | 27% | | |  |
| Total | - | 135 | | | 135 | | 119% | | |  |
| **Defectos eliminados** | **Plan** | **Actual** | | | **Hasta la fecha** | | **% Hasta la fecha** | | | **Def./Hora** |
| Planificación | - | 45 | | | 45 | | 10% | | |  |
| Diseño | - | 7 | | | 7 | | 17% | | |  |
| Codificación | - | 12 | | | 12 | | 21% | | |  |
| Revisión del código | - | 25 | | | 25 | | 23% | | | 1.5 |
| Compilación | - | 15 | | | 15 | | 27% | | | 1.8 |
| Pruebas | - | 8 | | | 8 | | 30% | | | 2.3 |
| Total | - | 112 | | | 112 | | 128 | | |  |

## **Código fuente según estándar.**

**Clase del layout principal**

**public class** Principal **extends** AppCompatActivity {  
  
 TextView jTextLadoA; *// Se declaran objetos de tipo TextView* TextView jTextLadoB; *// Se declaran objetos de tipo TextView* TextView jTextLadoC; *// Se declaran objetos de tipo TextView* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_principal);  
  
 jTextLadoA = (TextView) findViewById(R.id.jTextValorA); *// Se instancia un objeto TextView* jTextLadoB = (TextView) findViewById(R.id.jTextValorB); *// Se instancia un objeto TextView* jTextLadoC = (TextView) findViewById(R.id.jTextValorC); *// Se instancia un objeto TextView* Button jButtonCalcular = (Button) findViewById(R.id.jButtonCalcular); *// Se instancia un objeto Button  
 // Se crea un evento para el botón y se realice un método creado* jButtonCalcular.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 RealizarAccion(); *// Se referencia al método a usar* }  
 });  
 }  
  
 *// Se declara el método a utilizar por el botón de la interfaz* **private void** RealizarAccion() {  
 TestTriangulo oTest = **new** TestTriangulo(); *// Instanciación de un objeto TestTriangulo* Tecladin oTecla = **new** Tecladin(); *// Instanciación de un objeto Tecladin  
 // Se crea una condicional para saber si tienen valores los TextView* **if**(!jTextLadoA.getText().equals(**""**) && !jTextLadoB.getText().equals(**""**) && !jTextLadoC.getText().equals(**""**)) {  
  
 *// Asignación de valores* oTest.setLadoA(Float.parseFloat(jTextLadoA.getText().toString()));  
 oTest.setLadoB(Float.parseFloat(jTextLadoB.getText().toString()));  
 oTest.setLadoC(Float.parseFloat(jTextLadoC.getText().toString()));  
  
 *// Se crea una condicional para saber si la longitud de los lados forman o son parte del triángulo* **if** (oTecla.ValidarTriangulo(oTest)) {  
 oTest.CalcularFuncionS(); *// Calcula la función de S* String strCadena = **"El área del triángulo es: "** + String.valueOf(oTest.CalcularArea()); *// Creación de mensaje con el resultado de la operación* Toast.makeText(**this**, strCadena, Toast.LENGTH\_SHORT).show(); *// Mostrar mensaje* }  
 **else** {  
 Toast.makeText(**this**, **"Las longitudes de los lados no le corresponden a un triángulo"**, Toast.LENGTH\_SHORT).show(); *// Mostrar mensaje si no es un triángulo* }  
 } *// Fin de la condicional* } *// Fin del método*}

**Clase Triangulo**

**public class** Triangulo {  
 **private float ladoA**; *// Declaración del atributo del primer lado del triángulo* **private float ladoB**; *// Declaración del atributo del segundo lado del triángulo* **private float ladoC**; *//Declaración del atributo del tercer lado del triángulo  
  
 // Asignación de valor al lado a por medio de un parámetro* **public void** setLadoA (**float** A){  
 **this**.**ladoA** = A;  
 }*//Fin del método setLadoA  
  
 // Obtener los datos de la variable lado a* **public float** getLadoA(){  
 **return ladoA**;  
 }*//Fin del método getLadoA  
  
 // Asignación de valor al lado b por medio de un parámetro* **public void** setLadoB (**float** B){  
 **this**.**ladoB** = B;  
 }*//Fin del método setLadoB  
  
 // Obtener los datos de la variable lado b* **public float** getLadoB(){  
 **return ladoB**;  
 }*//Fin del método getLadoB  
  
 // Asignación de valor al lado c por medio de un parámetro* **public void** setLadoC (**float** C){  
 **this**.**ladoC** = C;  
 }*//Fin del metido setLadoC  
  
 // Obtener los datos de la variable lado c* **public float** getLadoC(){  
 **return ladoC**;  
 }*//Fin del método getLadoC*}

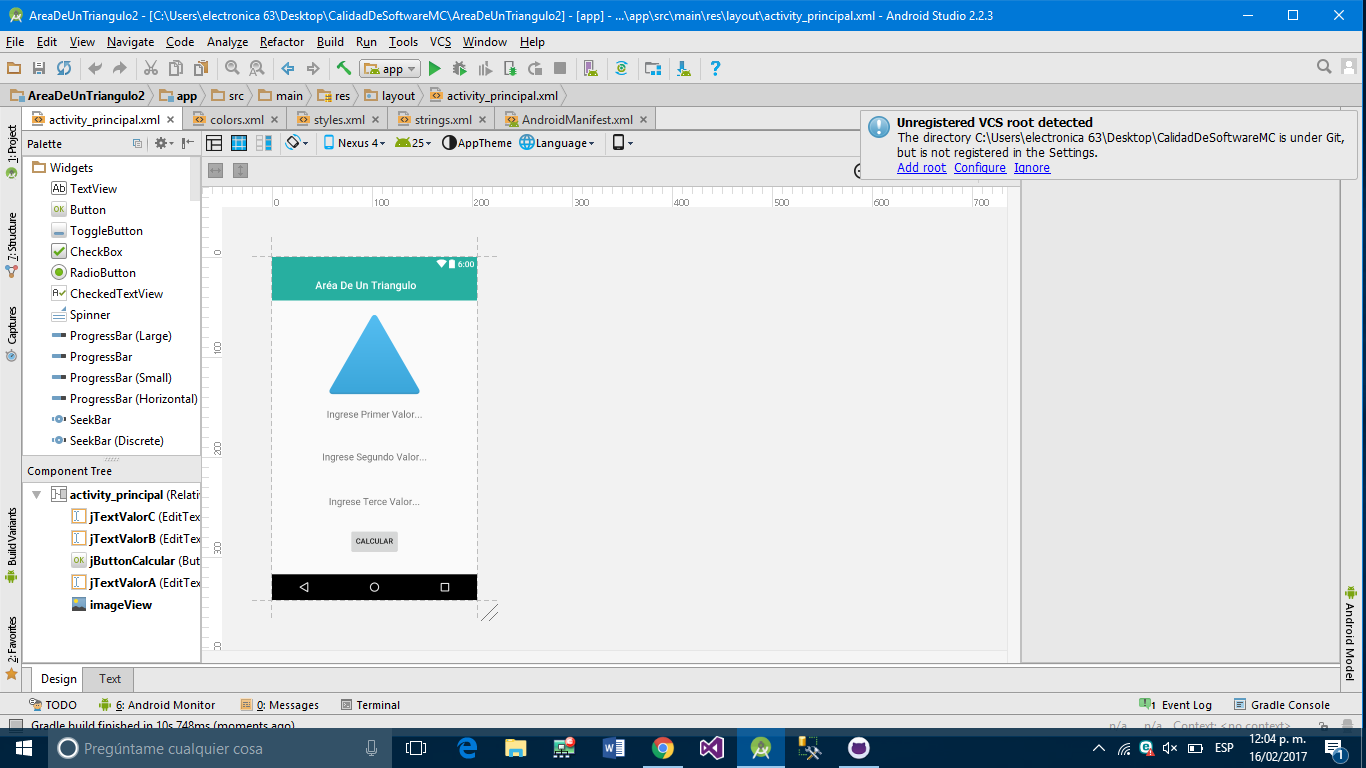
**Clase TestTriangulo**

**public class** TestTriangulo **extends** Triangulo{  
  
 **private float funcionS** ;*//Declaración del atributo funcionS del triángulo* **private float areaTriangulo**;*// Declaración del atributo área del Triángulo  
  
 // Método para calcular la funcionS* **public void** CalcularFuncionS() {  
  
 **float** ladoA = getLadoA(); *// Obtención del lado A del triángulo* **float** ladoB = getLadoB(); *// Obtención del lado B del triángulo* **float** ladoC = getLadoC(); *// Obtención del lado C del triángulo* **this**.**funcionS** = (ladoA + ladoB + ladoC)/2; *// Calculo de la funcionS* }*//Fin del método para calcular las funciones  
 //Método para calcular el área del triángulo* **public float** CalcularArea (){  
 **float** ladoA = getLadoA(); *// Obtención del lado A del triángulo* **float** ladoB = getLadoB(); *// Obtención del lado B del triángulo* **float** ladoC = getLadoC(); *// Obtención del lado C del triángulo* **this**.**areaTriangulo** = (**float**) Math.*sqrt*(**funcionS**\*(**funcionS**-ladoA)\*(**funcionS**-ladoB)\*(**funcionS**-ladoC)); *//Calculo del área del triángulo en funcion a la variable funcionS* **return this**.**areaTriangulo**;  
 }*//fin del método para calcular el área del triángulo*}

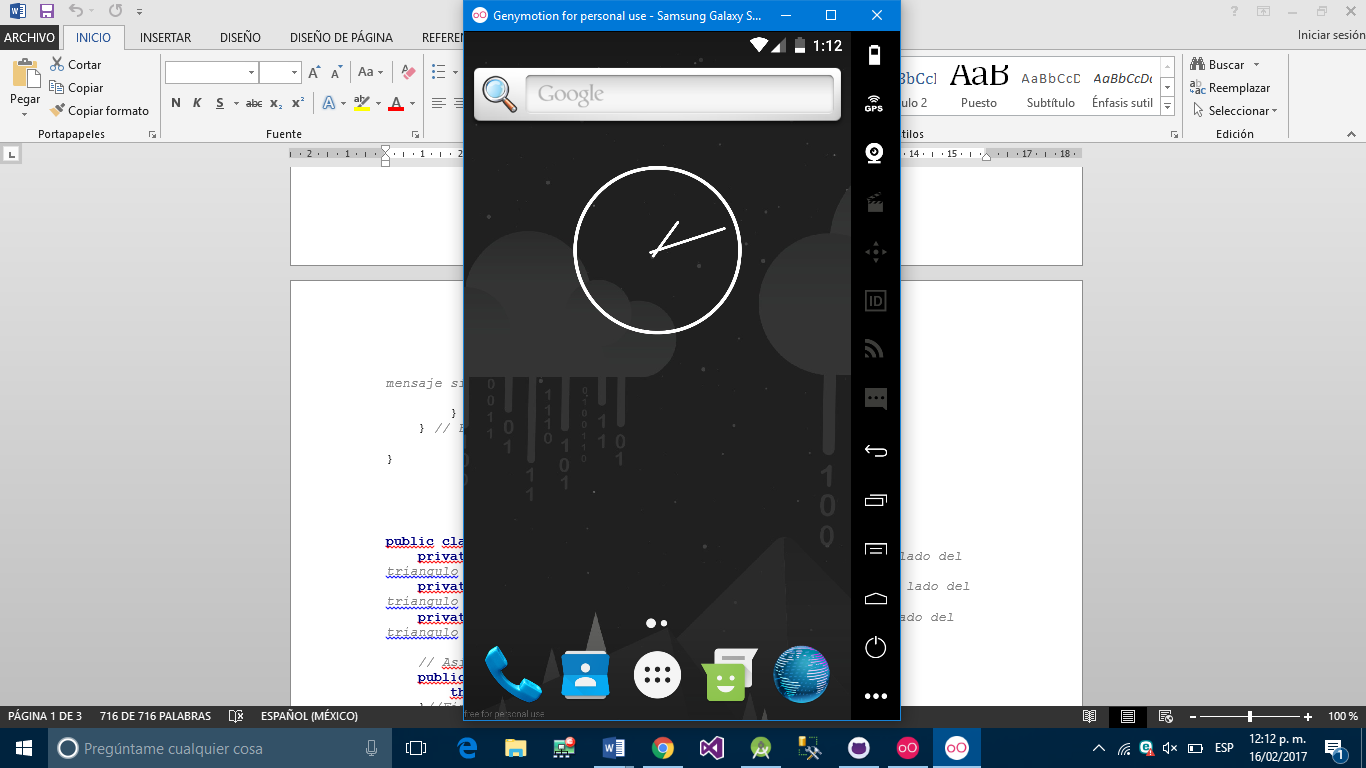
**Clase Tecladin**

**public class** Tecladin {  
 *// Método de validación de triangulo* **public boolean** ValidarTriangulo(TestTriangulo Triangulo){  
 **float** ladoA = Triangulo.getLadoA(); *// Obtención del lado A del triángulo* **float** ladoB = Triangulo.getLadoB(); *// Obtención del lado B del triángulo* **float** ladoC = Triangulo.getLadoC(); *// Obtención del lado C del triángulo  
 // si la suma de dos lados es menor al valor del tercer lado, no es un triángulo* **if** ((ladoA+ladoB) < ladoC || (ladoA+ladoC)< ladoB || (ladoB+ladoC)< ladoA) {  
 **return false**; *// Se retorna un valor falso* }  
 *// Si no, si es un triángulo* **else** {  
 **return true**; *// Se retorna un valor verdadero* }  
 } *// Fin del método de validación*}

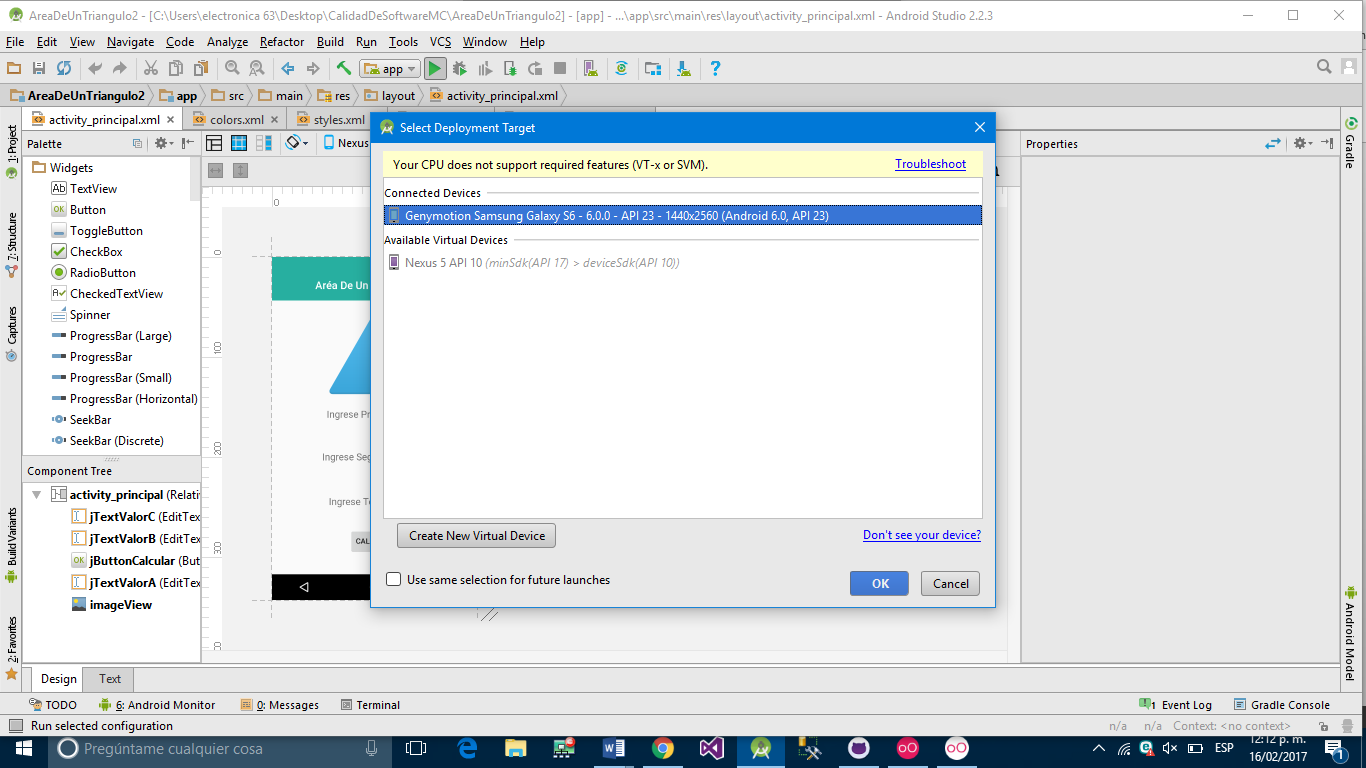
## **Aplicación ejecutable e instrucciones para ejecutar.**



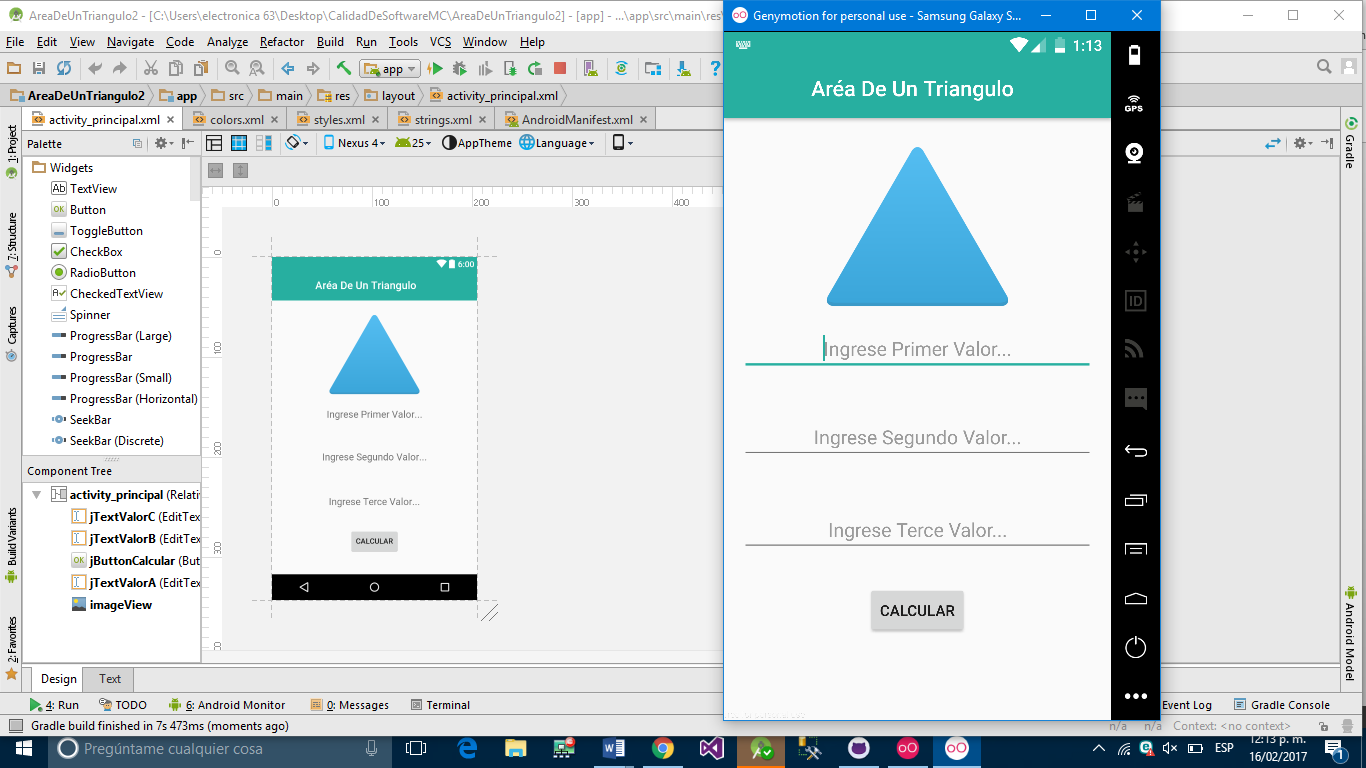
Paso 1: Abrir el Android Studio.



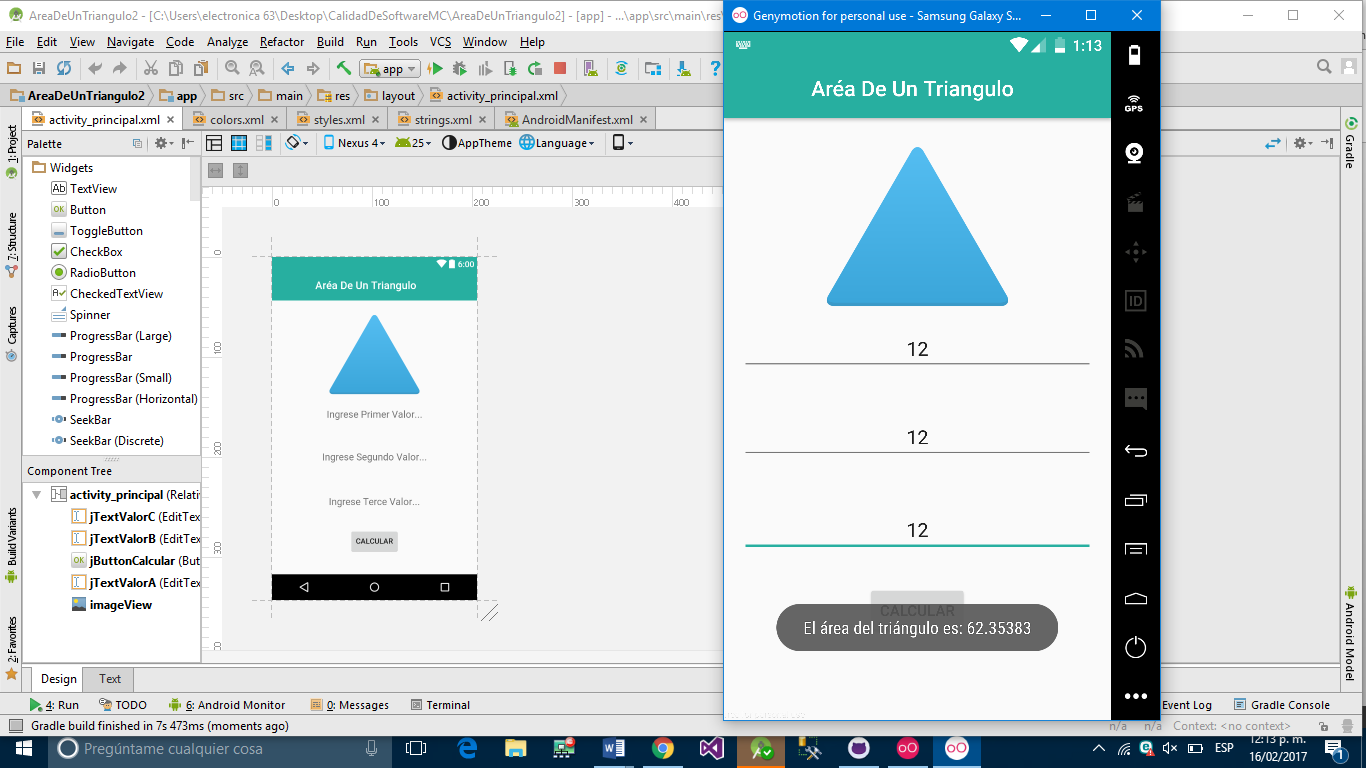
Paso 2: Abrir el emulador que en este caso usamos Genymotion.



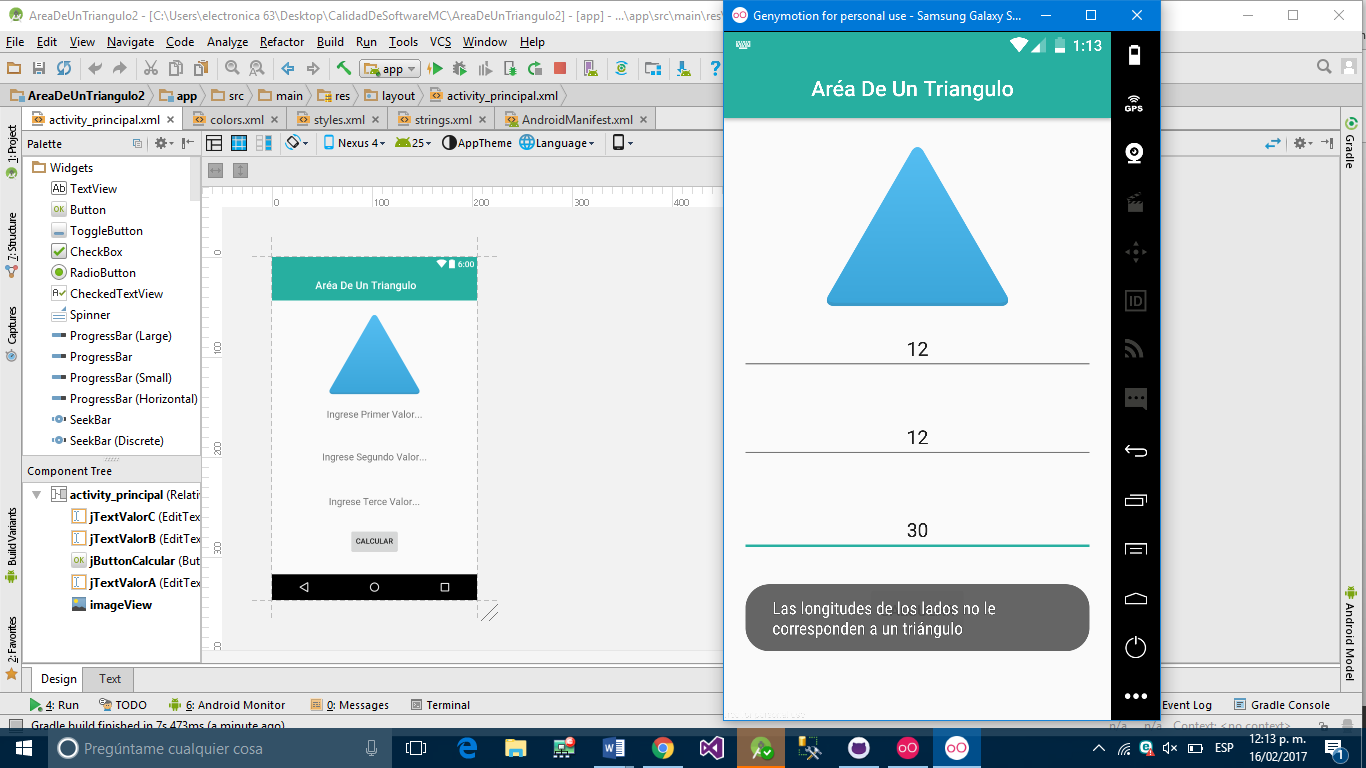
Paso 3: Ejecutar el programa y seleccionar el emulador de su preferencia, para este caso se toma el primer emulador que ofrece genymotion.



Paso 4: Una vez realizado esos pasos, se visualiza el dispositivo o emulador que genymotion con la interfaz desarrollada.



Paso 5: Se ingresan valores en las cajas de números que la aplicación te dice donde escribir, una vez ingresado los tres valores se presiona el botón calcular y este verifica si es un triángulo, en este caso se ingresó valores 12,12,12 y el mensaje muestra el área del triángulo ya que este si es un triángulo.



Paso 6: En este caso se ingresaron distintos valores y el mensaje muestra que estos valores no corresponden a un triangulo.

## **Conclusiones:**

**José Luis Puc Chan:**

Al principio es muy difícil adaptarse y poder medir los procesos y las tareas puesto que uno se acostumbra al mal manejo de nuestro tiempo, recursos e incluso del personal que uno dispone para poder realizar las tareas que tiene un desarrollo de software, esta práctica me demostró la importancia de documentar cada módulo la manera de cómo trabajar con estándares ya definidos y que en un futuro me va a servir para hacer mis horas más productivas y poder tener un mayor ingreso mediante una buena planeación de recursos.

Acostumbrarse a trabajar de esta manera podrá minimizar la interacción innecesaria con integrantes de tu equipo que tienen dudas sobre el desarrollo del mismo.

**José André Salazar Figueroa:**

El uso de estándares y metodologías de desarrollo de software ayudan a llevar un control y darle un buen seguimiento al proyecto a realizar. Conociendo estas herramientas he logrado aprender un poco más a cómo debo documentar, planificar y desarrollar un software, tomando en cuenta los estilos y estructuras de desarrollo, para así asegurar que el producto final obtenga la calidad necesaria como para que el usuario o el cliente quede satisfecho y evalúe de manera favorable el trabajo realizado.

**Diosemir Isael Nah May:**

Gracias a esta nueva implementación pude notar el cambio y la nueva restructuración del equipo en cuanto al desarrollo de esta actividad, adoptando así un nuevo control y manejo adecuado de las diversas tareas o actividades que se llevan acabó dentro de un proyecto. De igual manera es increíble notar las facilidades que un controlador de versiones puede tener sobre un equipo de trabajo, dando así una gran capacidad de conocimiento en cuanto a que se realiza en el proyecto y quien lo realiza, es increíble ver como en un proyecto pequeño se pueden tener algunos conflictos en cuanto a la repartición de tareas, ya que no es muy extenso, pero gracias a métodos que se manejan se pueden dar las soluciones para que el equipo trabaje a la perfección cuidando, asegurando y dándole un gran software de calidad al usuario.

**Aarón Jesús Peña Martin:**

En el desarrollo del sistema el PSP mejoró el trabajo de desarrollo de software porque se concentra en el trabajo individual de cada miembro del equipo de desarrollo de software, mostrándonos cómo manejar la calidad de software desde el principio del desarrollo del sistema, muestran nuestras propias métricas, que permiten estructurar y organizar de una manera más eficiente nuestro desarrollo de software, el cual lo organiza un líder de proyecto quien evalúa los resultados y avances de los miembros del equipo.

**Yair Roberto Vega Gamboa:**

El PSP nos sirvió para el trabajo individualmente y fue efectiva porque nos muestra una manera más eficiente de cómo controlar los tiempos y la gestión de errores, que tiene cada miembro del equipo y en base a eso nos ayuda a subir la productividad a la hora desarrollo de software. En mi opinión las tablas que maneja el PSP para el control de tiempos nos ayuda mucho a medir que productivos fuimos individualmente en ese tiempo transcurrido y como fuimos desarrollando la aplicación. La tabla de registro de errores fue la que más le busque efectividad porque nos muestra que defectos o errores cometemos individualmente y nos sirve en un futuro para analizar en que estamos “fallando” más y estar más pendiente de esos errores.