



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE APLICACIONES MÓVILES

APLICACIÓN 1 DE EVALUACIÓN UNIDAD 2: ÁREA DE FIGURAS GEOMÉTRICAS

RECIBE:
DR. ROBERTO ELVIRA ENRÍQUEZ

ENTREGA:
JESÚS HUERTA AGUILAR – 202041509

NRC: 57340
SECCIÓN: 003

SÉPTIMO SEMESTRE

PUEBLA, PUE.

ABRIL DE 2024

ÁREA DE FIGURAS GEOMÉTRICAS

APP 1 DE EVALUACIÓN UNIDAD 2

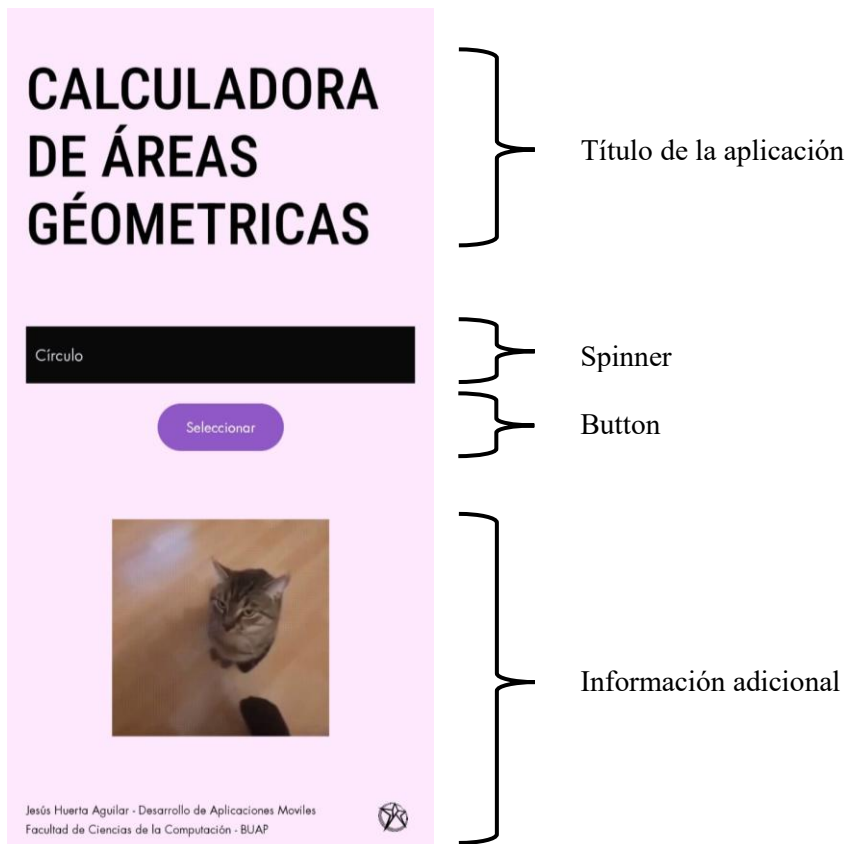
INSTRUCCIONES

Realizar una APP que calcule el área de un círculo, rectángulo, cuadrado, triángulo, rombo y un trapecio. Realizar un reporte en donde se detalle el funcionamiento de la APP, así como parte del código que sea relevante.

Nota: las vistas que se muestren al usuario para ingresar los datos deberán de cambiar conforme a la figura a calcular.

ACTIVITY MAIN

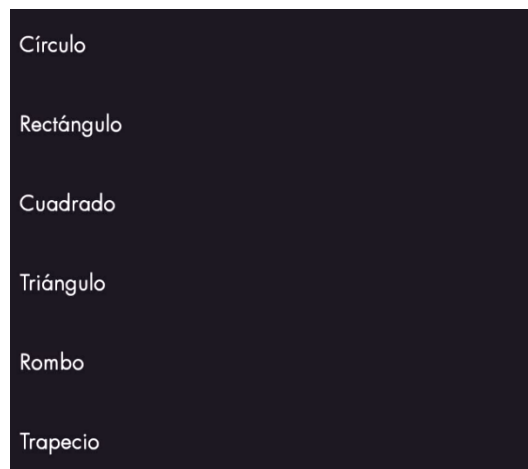
La *MainActivity* de la aplicación es la pantalla de lanzamiento que recibe a los usuarios con una interfaz intuitiva que permite la selección de diferentes figuras geométricas para calcular sus áreas. Esta pantalla es la primera interacción que el usuario tiene con la aplicación.



DISEÑO DE LA INTERFAZ

el diseño de la interfaz de usuario está configurado dentro de un `ConstraintLayout`, lo que permite una disposición flexible y adaptable de los componentes de la UI. La pantalla cuenta con un `Spinner` que ofrece una lista desplegable de figuras geométricas, un `Button` que sirve para confirmar la selección del usuario y múltiples `TextViews`

- **SPINNER**



El *Spinner*, identificado con el id *spinnerFiguras*, se inicializa con una serie de opciones que representan las figuras geométricas disponibles. Estas opciones se definen mediante un arreglo de cadenas de texto y se muestran al usuario a través de un *ArrayAdapter*.

FRAGMENTOS DE CÓDIGO IMPORTANTES

Código del `Spinner` en *activity_main.xml*

```
<Spinner
    android:id="@+id/spinnerFiguras"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#090909"
    android:backgroundTint="#000000"
    android:backgroundTintMode="add"
    android:foregroundTint="#000000"
    android:textColor="#000000"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.404" />
```

Codigo del Spinner en *mainActivity.kt*

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        // figuras geométricas disponibles  
        val figuras = arrayOf("Círculo", "Rectángulo", "Cuadrado",  
"Triángulo", "Rombo", "Trapezio")  
        val spinnerFiguras = findViewById<Spinner>(R.id.spinnerFiguras)  
  
        // mostrar las opciones en un Spinner  
        val adapter = ArrayAdapter(this,  
android.R.layout.simple_spinner_dropdown_item, figuras)  
        spinnerFiguras.adapter = adapter  
  
        ...  
    }  
}
```

- **BOTÓN SELECCIONAR**



El Button con id *btnSeleccionar* implementa un *OnClickListener* que captura la selección del usuario y, mediante una estructura condicional *when*, determina la actividad que debe iniciarse.

FRAGMENTOS DE CÓDIGO IMPORTANTES

Codigo del button en *activity_main.xml*

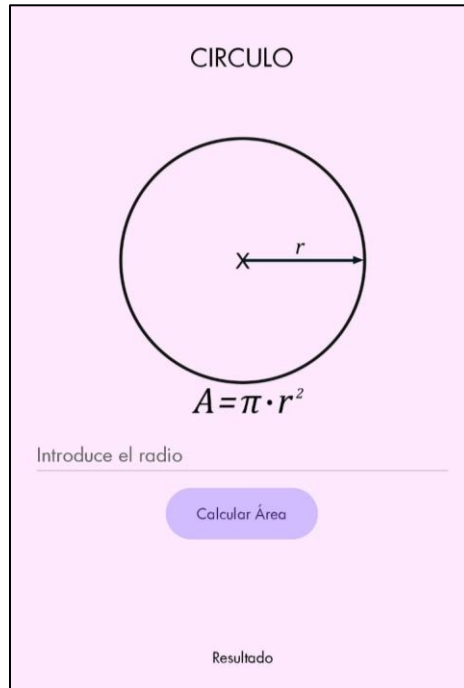
```
<Button  
    android:id="@+id/btnSeleccionar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:backgroundTint="#9058C6"  
    android:text="Seleccionar"  
    android:textColor="#FFFFFF"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.499" />
```

Código del button en *mainActivity.kt*

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        ...  
  
        findViewById<Button>(R.id.btnSeleccionar).setOnClickListener {  
            val seleccion = spinnerFiguras.selectedItem.toString()  
            // Dependiendo de la selección, iniciamos la actividad correspondiente  
            when (seleccion) {  
                "Círculo" -> startActivity(Intent(this,  
CirculoActivity::class.java))  
                "Rectángulo" -> startActivity(Intent(this,  
RectanguloActivity::class.java))  
                "Cuadrado" -> startActivity(Intent(this,  
CuadradoActivity::class.java))  
                "Triángulo" -> startActivity(Intent(this,  
TrianguloActivity::class.java))  
                "Rombo" -> startActivity(Intent(this,  
RomboActivity::class.java))  
                "Trapezio" -> startActivity(Intent(this,  
TrapezioActivity::class.java))  
            }  
        }  
    }  
}
```

CIRCULO ACTIVITY

La *CirculoActivity* es una actividad específica en la aplicación que se dedica al cálculo del área de un círculo. Esta actividad proporciona una interfaz sencilla donde los usuarios pueden ingresar el valor del radio y obtener el área correspondiente.



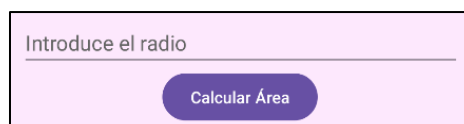
DISEÑO DE LA INTERFAZ

La pantalla está diseñada para ser auto explicativa, con etiquetas claras, un diagrama del círculo junto con su fórmula para calcular su área y un flujo lógico que conduce al usuario desde la entrada de datos hasta la visualización del resultado. Los colores de texto y los tamaños de fuente se seleccionan para una buena legibilidad en diversos fondos y condiciones de iluminación.

CÁLCULO DEL ÁREA

- INTERFAZ DE USUARIO (XML)

En el archivo `activity_circulo.xml`, se presenta al usuario un *EditText* con el id `editTextRadio`, donde se puede ingresar el valor del radio del círculo. Este componente está diseñado para aceptar únicamente números decimales, lo que previene errores de formato en la entrada.



```
<EditText
    android:id="@+id/editTextRadio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce el radio"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- **LÓGICA DE PROCESAMIENTO (KOTLIN)**

Dentro de *CirculoActivity.kt*, el botón con id *btnCalcularArea* está equipado con un *OnClickListener* que desencadena la secuencia de cálculo. Al presionarlo, el sistema recupera el texto del *EditText* y verifica que no esté vacío, lo que garantiza que se haya proporcionado un valor antes de realizar cualquier operación.

En caso de que el valor esté presente, el radio ingresado por el usuario se convierte a un *Double*. Luego, se aplica la fórmula matemática del área de un círculo $A = \pi r^2$, utilizando la constante *PI* proporcionada por el lenguaje Kotlin para asegurar la precisión del cálculo. El resultado se asigna al *TextView* con el id *textViewResultado*, presentando al usuario el área calculada de forma inmediata y legible.

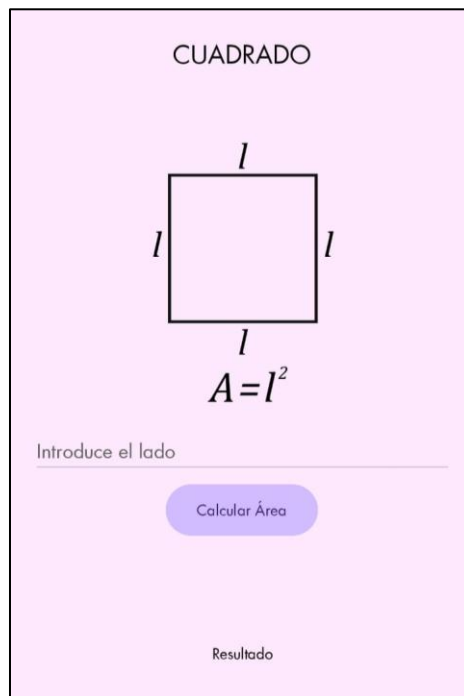
```
class CirculoActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_circulo)

        val btnCalcular = findViewById<Button>(R.id.btnCalcularArea)
        val editTextRadio = findViewById<EditText>(R.id.editTextRadio)
        val textViewResultado = findViewById<TextView>(R.id.textViewResultado)

        btnCalcular.setOnClickListener {
            val radioString = editTextRadio.text.toString()
            if (radioString.isNotEmpty()) {
                val radio = radioString.toDouble()
                val area = PI * radio * radio
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce el radio."
            }
        }
    }
}
```

CUADRADO ACTIVITY

La *CuadradoActivity* es una interfaz dedicada al cálculo del área de un cuadrado. Esta actividad proporciona un entorno específico donde el usuario puede calcular de forma sencilla el área ingresando la medida de uno de sus lados.



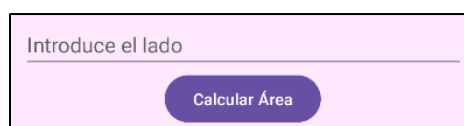
DISEÑO DE LA INTERFAZ

El archivo *activity_cuadrado.xml* organiza visualmente los componentes para alinearlos con la función que representan, resaltando la importancia del flujo de entrada de datos, acción y visualización de resultados. La imagen que contiene la fórmula del área de un cuadrado no solo decora la interfaz, sino que también funciona como referencia didáctica para el usuario.

CÁLCULO DEL ÁREA

- **INTERFAZ DE USUARIO (XML)**

La interfaz gráfica de usuario (GUI), definida en el archivo *activity_cuadrado.xml*, proporciona un campo *EditText* donde los usuarios pueden introducir la medida del lado del cuadrado. Este campo está configurado para aceptar entradas numéricas decimales, evitando así entradas no válidas que podrían resultar en cálculos erróneos.




```
<EditText
    android:id="@+id/editTextLado"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce el lado"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- **LÓGICA DE PROCESAMIENTO (KOTLIN)**

En el archivo *CuadradoActivity.kt*, la interacción del usuario con el botón 'Calcular Área' desencadena un evento que ejecuta el método *setOnClickListener*. Dentro de este método, la aplicación recoge el valor ingresado por el usuario desde el *EditText*, lo convierte a un número *Double*, y realiza la operación de cálculo del área utilizando la fórmula matemática para el área de un cuadrado $A = l^2$, donde l representa la longitud del lado del cuadrado.

Si no se introduce ningún valor, la aplicación muestra un mensaje de error en el *TextView* para el resultado, instando al usuario a proporcionar la medida del lado del cuadrado antes de realizar el cálculo.

```
class CuadradoActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cuadrado)

        val editTextLado = findViewById<EditText>(R.id.editTextLado)
        val textViewResultado = findViewById<TextView>(
            R.id.textViewResultadoCuadrado)
        val btnCalcular = findViewById<Button>(R.id.btnCalcularAreaCuadrado)

        btnCalcular.setOnClickListener {
            val ladoString = editTextLado.text.toString()
            if (ladoString.isNotEmpty()) {
                val lado = ladoString.toDouble()
                val area = lado * lado
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce el lado del
cuadrado."
            }
        }
    }
}
```

RECTANGULO ACTIVITY

La actividad *RectanguloActivity* está diseñada para facilitar el cálculo del área de un rectángulo. Esta pantalla permite al usuario introducir las dimensiones del largo y el ancho del rectángulo para obtener el área correspondiente.

RECTANGULO

h

b

$A = b \cdot h$

Introduce el largo

Introduce el ancho

Calcular Área

Resultado

DISEÑO DE LA INTERFAZ

La interfaz de usuario está compuesta por dos campos *EditText* para ingresar el largo y el ancho del rectángulo, un *Button* para realizar el cálculo y un *TextView* donde se muestra el resultado. Si los campos de largo y ancho están vacíos, la aplicación proporciona retroalimentación directa solicitando al usuario que introduzca ambas medidas antes de calcular.

CÁLCULO DEL ÁREA

- INTERFAZ DE USUARIO (XML)

Dentro de *activity_rectangulo.xml*, se proporcionan dos campos *EditText*, uno para el largo (*editTextLargo*) y otro para el ancho (*editTextAncho*) del rectángulo. Estos campos están diseñados para capturar entradas numéricas, guiando al usuario a través de pistas visuales que mantienen la coherencia con el esquema general de diseño de la aplicación.

Introduce el largo

Introduce el ancho

Calcular Área

```
<EditText
    android:id="@+id/editTextLargo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce el largo"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.498" />

<EditText
    android:id="@+id/editTextAncho"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce el ancho"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.567" />
```

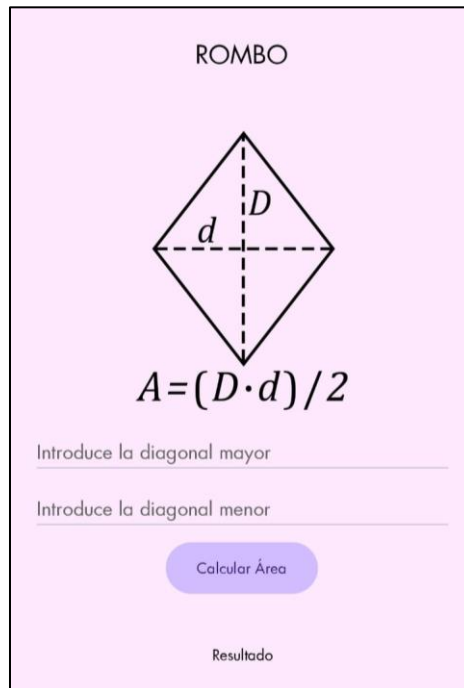
- **LÓGICA DE PROCESAMIENTO (KOTLIN)**

En *RectanguloActivity.kt*, los dos campos de texto, identificados como *editTextLargo* y *editTextAncho*, recogen las entradas del usuario. Al presionar el botón *btnCalcularAreaRectangulo*, la aplicación verifica que ambos campos estén llenos. Si se cumplen estas condiciones, la actividad toma los valores, los convierte en números de tipo *Double* y realiza el cálculo del área utilizando la fórmula del área del rectángulo $A = b \cdot h$, donde *b* es la base y *h* es la altura. El resultado se muestra instantáneamente en el *TextView textViewResultadoRectangulo*.

```
class RectanguloActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_rectangulo)
        val editTextLargo = findViewById<EditText>(R.id.editTextLargo)
        val editTextAncho = findViewById<EditText>(R.id.editTextAncho)
        val textViewResultado =
            findViewById<TextView>(R.id.textViewResultadoRectangulo)
        val btnCalcular = findViewById<Button>(R.id.btnCalcularAreaRectangulo)
        btnCalcular.setOnClickListener {
            val largoString = editTextLargo.text.toString()
            val anchoString = editTextAncho.text.toString()
            if (largoString.isNotEmpty() && anchoString.isNotEmpty()) {
                val largo = largoString.toDouble()
                val ancho = anchoString.toDouble()
                val area = largo * ancho
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce el largo y el ancho."
            }
        }
    }
}
```

ROMBO ACTIVITY

La *RomboActivity* de la aplicación está dedicada a facilitar el cálculo del área de un rombo, guiando al usuario a través de la inserción de las longitudes de las diagonales. Esta funcionalidad es esencial para quienes buscan una herramienta educativa o una ayuda rápida en geometría.



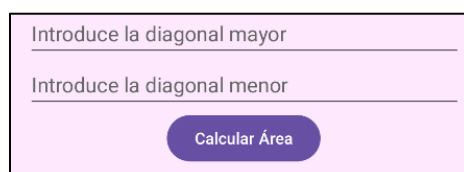
DISEÑO DE LA INTERFAZ

La captura de pantalla de `activity_rombo.xml` muestra una interfaz clara y directa, compuesta por dos campos *EditText* para que los usuarios introduzcan las longitudes de las diagonales del rombo, un botón para ejecutar el cálculo y un *TextView* que muestra el resultado. Un *ImageView* proporciona una representación gráfica de un rombo y la fórmula para el cálculo del área, sirviendo como referencia visual.

CÁLCULO DEL ÁREA

- **INTERFAZ DE USUARIO (XML)**

La interfaz para calcular el área de un rombo, definida en `activity_rombo.xml`, provee al usuario dos campos de texto (*EditText*) para introducir las medidas de las diagonales. Estos están etiquetados como "Introduce la diagonal mayor" y "Introduce la diagonal menor", correspondientes a `editTextDiagonalMayor` y `editTextDiagonalMenor`. Estos campos están diseñados para recoger entradas numéricas.



```
<EditText
    android:id="@+id/editTextDiagonalMayor"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la diagonal mayor"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/editTextDiagonalMenor"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la diagonal menor"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.568" />
```

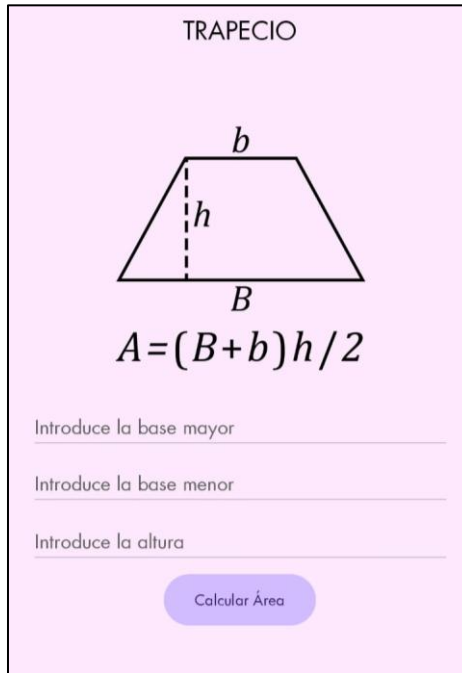
• LÓGICA DE PROCESAMIENTO (KOTLIN)

La clase *RomboActivity*, contenida en el archivo *RomboActivity.kt*, maneja la lógica operativa del cálculo. Una vez que los usuarios proporcionan las longitudes de las diagonales y presionan el botón de cálculo (*btnCalcularAreaRombo*), el evento *setOnClickListener* es disparado. Este evento invoca un bloque de código que verifica si ambos campos *EditText* tienen valores. De ser así, los valores se leen como cadenas, se convierten a tipo *Double* y se calcula el área usando la fórmula estándar para el área de un rombo $A = D \cdot d / 2$, donde *D* es la longitud de la diagonal mayor y *d* es la longitud de la diagonal menor. El resultado se asigna al *TextView textViewResultadoRombo* para su visualización.

```
class RomboActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_rombo)
        val editTextDiagonalMayor = findViewById<EditText>(R.id.editTextDiagonalMayor)
        val editTextDiagonalMenor = findViewById<EditText>(R.id.editTextDiagonalMenor)
        val textViewResultado = findViewById<TextView>(R.id.textViewResultadoRombo)
        val btnCalcular = findViewById<Button>(R.id.btnCalcularAreaRombo)
        btnCalcular.setOnClickListener {
            val diagonalMayorString = editTextDiagonalMayor.text.toString()
            val diagonalMenorString = editTextDiagonalMenor.text.toString()
            if (diagonalMayorString.isNotEmpty() && diagonalMenorString.isNotEmpty()) {
                val diagonalMayor = diagonalMayorString.toDouble()
                val diagonalMenor = diagonalMenorString.toDouble()
                val area = (diagonalMayor * diagonalMenor) / 2
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce las diagonales."
            }
        }
    }
}
```

TRAPECIO ACTIVITY

La TrapecioActivity proporciona una interfaz específica para el cálculo del área de un trapecio. Esta actividad permite a los usuarios ingresar las medidas de las bases y la altura, y calcula el área utilizando la fórmula apropiada.



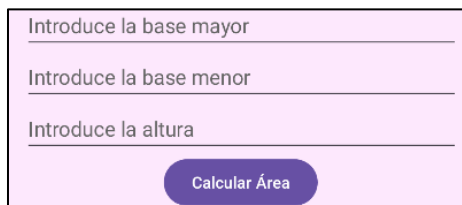
DISEÑO DE LA INTERFAZ

La interfaz, mostrada en activity_trapecio.xml, es coherente con el esquema de diseño general de la aplicación. Presenta tres campos EditText para que los usuarios ingresen las medidas de la base mayor, la base menor y la altura, además de un botón Calcular Área que inicia el cálculo. La imagen del trapecio y la fórmula proporcionada actúan como una herramienta didáctica visual que apoya al usuario en la comprensión del proceso de cálculo.

CÁLCULO DEL ÁREA

- **INTERFAZ DE USUARIO (XML)**

La actividad presenta un diseño intuitivo con campos de entrada claramente marcados en activity_trapecio.xml, donde se pide a los usuarios que introduzcan las longitudes de la base mayor, la base menor y la altura del trapecio. Estos campos EditText son accesibles y están etiquetados de forma que el usuario pueda ingresar los datos sin equivocaciones.



```
<EditText
    android:id="@+id/editTextBaseMayor"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la base mayor"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/editTextAlturaTrapezio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la altura"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.637" />

<EditText
    android:id="@+id/editTextBaseMenor"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la base menor"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.568" />
```

- **LÓGICA DE PROCESAMIENTO (KOTLIN)**

El archivo *TrapezioActivity.kt* contiene la lógica para el cálculo del área. Cuando el usuario ha introducido los valores y presiona el botón *btnCalcularAreaTrapezio*, el método *setOnClickListener* se activa. Este método verifica que los campos de las bases y la altura no estén vacíos y realiza la conversión de las entradas a números decimales. Luego, implementa la fórmula del área de un trapezio $A = (B + b) \cdot d / 2$, donde B y b representan las bases y h la altura, para calcular el área. Este resultado se muestra directamente en el *TextView* destinado para resultados.

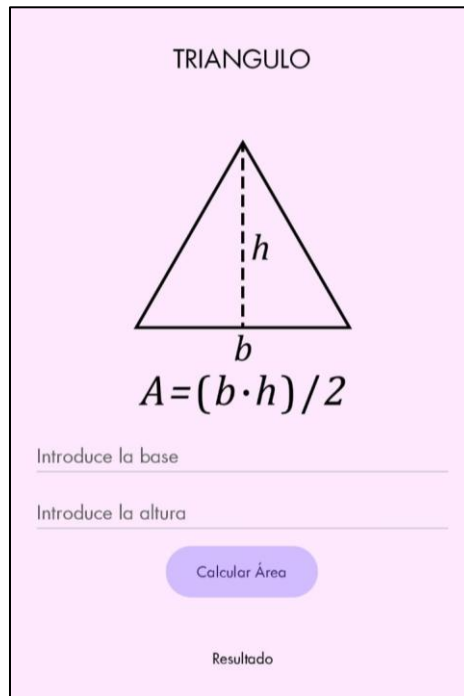
```
class TrapecioActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_trapecio)

        val editTextBaseMayor =
            findViewById<EditText>(R.id.editTextBaseMayor)
        val editTextBaseMenor =
            findViewById<EditText>(R.id.editTextBaseMenor)
        val editTextAlturaTrapecio =
            findViewById<EditText>(R.id.editTextAlturaTrapecio)
        val textViewResultado =
            findViewById<TextView>(R.id.textViewResultadoTrapecio)
        val btnCalcular =
            findViewById<Button>(R.id.btnCalcularAreaTrapecio)

        btnCalcular.setOnClickListener {
            val baseMayorString = editTextBaseMayor.text.toString()
            val baseMenorString = editTextBaseMenor.text.toString()
            val alturaString = editTextAlturaTrapecio.text.toString()
            if (baseMayorString.isNotEmpty() &&
                baseMenorString.isNotEmpty() && alturaString.isNotEmpty()) {
                val baseMayor = baseMayorString.toDouble()
                val baseMenor = baseMenorString.toDouble()
                val altura = alturaString.toDouble()
                val area = ((baseMayor + baseMenor) * altura) / 2
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce las
bases y la altura."
            }
        }
    }
}
```


TRIANGULO ACTIVITY

La *TrianguloActivity* está diseñada para calcular el área de un triángulo, una de las figuras geométricas fundamentales. Los usuarios pueden ingresar las medidas de la base y la altura del triángulo y obtener el área correspondiente.



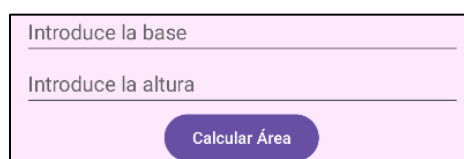
DISEÑO DE LA INTERFAZ

Los usuarios cuentan con dos campos de entrada: uno para la base (*editTextBase*) y otro para la altura (*editTextAltura*), y un botón Calcular Área que procesa el cálculo. Un *ImageView* ilustra un triángulo con la fórmula del área sirviendo como una guía visual

CÁLCULO DEL ÁREA

- INTERFAZ DE USUARIO (XML)

La interfaz diseñada en *activity_triangulo.xml* proporciona dos campos de texto *EditText* para que el usuario introduzca los valores de la base y la altura del triángulo. Estos campos están convenientemente etiquetados y posicionados para facilitar la entrada de datos de manera eficiente y sin errores.



```
<EditText
    android:id="@+id/editTextBase"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la base"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/editTextAltura"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Introduce la altura"
    android:inputType="numberDecimal"
    android:textColor="#000000"
    android:textColorHint="#656565"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.568" />
```

- **LÓGICA DE PROCESAMIENTO (KOTLIN)**

TrianguloActivity.kt contiene el corazón computacional de la actividad. Al hacer clic en el botón de calcular, definido como *btnCalcularAreaTriangulo*, el evento *setOnClickListener* se dispara. Dentro de este controlador de eventos, se recupera el texto de los campos *editTextBase* y *editTextAltura*, y se comprueba que no estén vacíos. Los valores recogidos son convertidos a números decimales y se aplica la fórmula para el área de un triángulo $A = b \cdot h/2$, donde b representa la longitud de la base y h la altura. El resultado de esta operación se muestra en el *TextView textViewResultadoTriangulo*.

```
class TrianguloActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_triangulo)

        val editTextBase = findViewById<EditText>(R.id.editTextBase)
        val editTextAltura = findViewById<EditText>(R.id.editTextAltura)
        val textViewResultado =
            findViewById<TextView>(R.id.textViewResultadoTriangulo)
        val btnCalcular = findViewById<Button>(R.id.btnCalcularAreaTriangulo)

        btnCalcular.setOnClickListener {
            val baseString = editTextBase.text.toString()
            val alturaString = editTextAltura.text.toString()
            if (baseString.isNotEmpty() && alturaString.isNotEmpty()) {
                val base = baseString.toDouble()
                val altura = alturaString.toDouble()
                val area = (base * altura) / 2
                textViewResultado.text = "Área: $area"
            } else {
                textViewResultado.text = "Por favor, introduce la base y la
altura."
            }
        }
    }
}
```