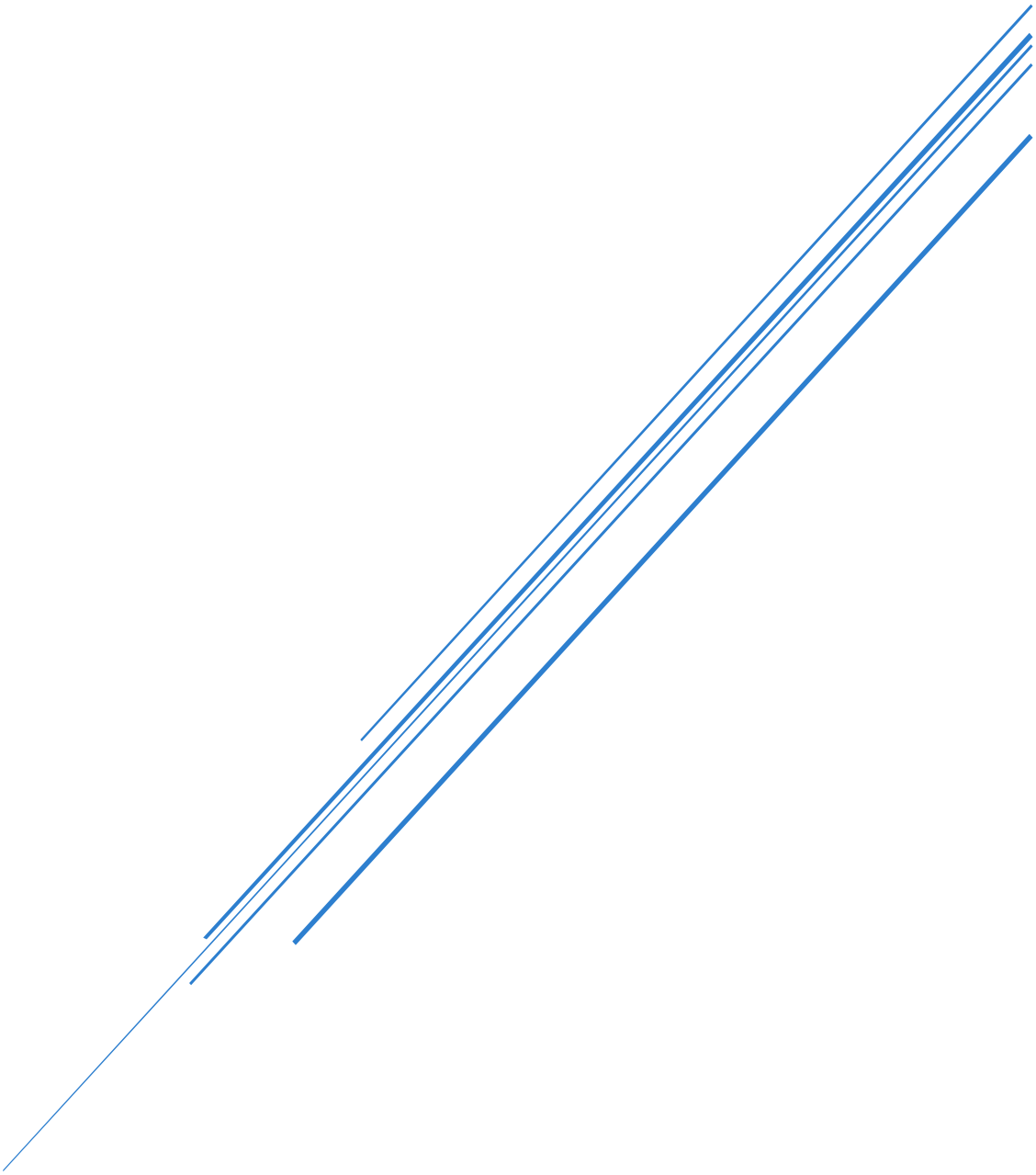
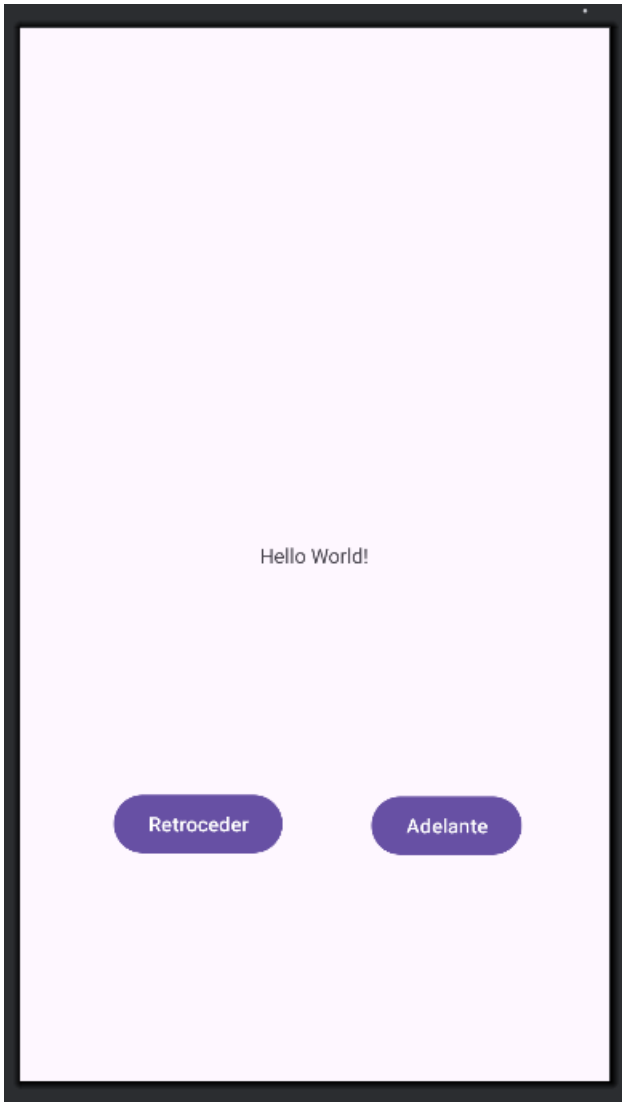
ACTIVIDAD M5-01

Estructuras con la sentencia For



Jorge Ernesto Ceballos Guzman

1. Lo primero que se hizo fue hacer un proyecto de Empty Views Activity
2. En el Layout se establecierons varios elementos



2.1. Un TextView que mostrar el nombre de la serie de Fibonacci, el cual posteriormente será reemplazado propiamente con la serie de Fibonacci

```
<TextView
    android:id="@+id/txv_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2.2. Un botón que será el que avance en la serie de Fibonacci

```
<Button
    android:id="@+id/btn_reinicio"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Adelante"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.802"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txv_main" />
```

2.3. Un botón que será el que retroceda en la serie de Fibonacci

```
<Button
    android:id="@+id/btnRetroceder"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Retroceder"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.224"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.773"
    tools:ignore="MissingConstraints" />
```

3. Luego, en nuestro MainActivity, comenzamos a programar el código de la serie de Fibonacci

3.1. Primero declaramos los elementos de la interfaz de usuario que vamos a usar:

```
// Declaración de variables y de los elementos de la interfaz de usuario
2 usages
Button btnAvanzar;
2 usages
Button btnRetroceder;
4 usages
TextView txvMiTexto;
```

3.2. EL contador que nos servirá para llevar el registro de los elementos

```
// Contador para la serie de Fibonacci
5 usages
int contador = 1;
```

3.3. El mapa que nos será de utilidad para el almacenamiento de los valores de la serie

```
// Mapa para almacenar los valores de la serie de Fibonacci
3 usages
Map<Integer, Long> memo = new HashMap<>();
```

3.4. En nuestro método onCreate inicializamos nuestros elementos de la interfaz de usuario, los cuales son recuperados del Layout

```
// Inicialización de los elementos de la interfaz de usuario (del layout)
txvMiTexto = findViewById(R.id.txv_main);
btnAvanzar = findViewById(R.id.btn_reinicio);
btnRetroceder = findViewById(R.id.btnRetroceder);
```

3.5. Después establecemos el texto de la serie de Fibonacci en el TextView

```
// Establecer el texto del TextView inicial
txvMiTexto.setText("Serie de Fibonacci");
```

3.6. Establecemos el comportamiento de los botones de avanzar y retroceder, respectivamente

```
//Comportamiento del boton de avance: siguiente numero de la serie de Fibonacci
btnAvanzar.setOnClickListener( View v -> {
    long valor = fibonacci(contador);
    txvMiTexto.setText(String.valueOf(valor));
    contador++;
});
```

```
//Comportamiento del boton de retroceso: numero anterior de la serie de Fibonacci
btnRetroceder.setOnClickListener( View v -> {
    if (contador > 1) {
        contador--;
        long valor = fibonacci(contador);
        txvMiTexto.setText(String.valueOf(valor));
    }
});
```

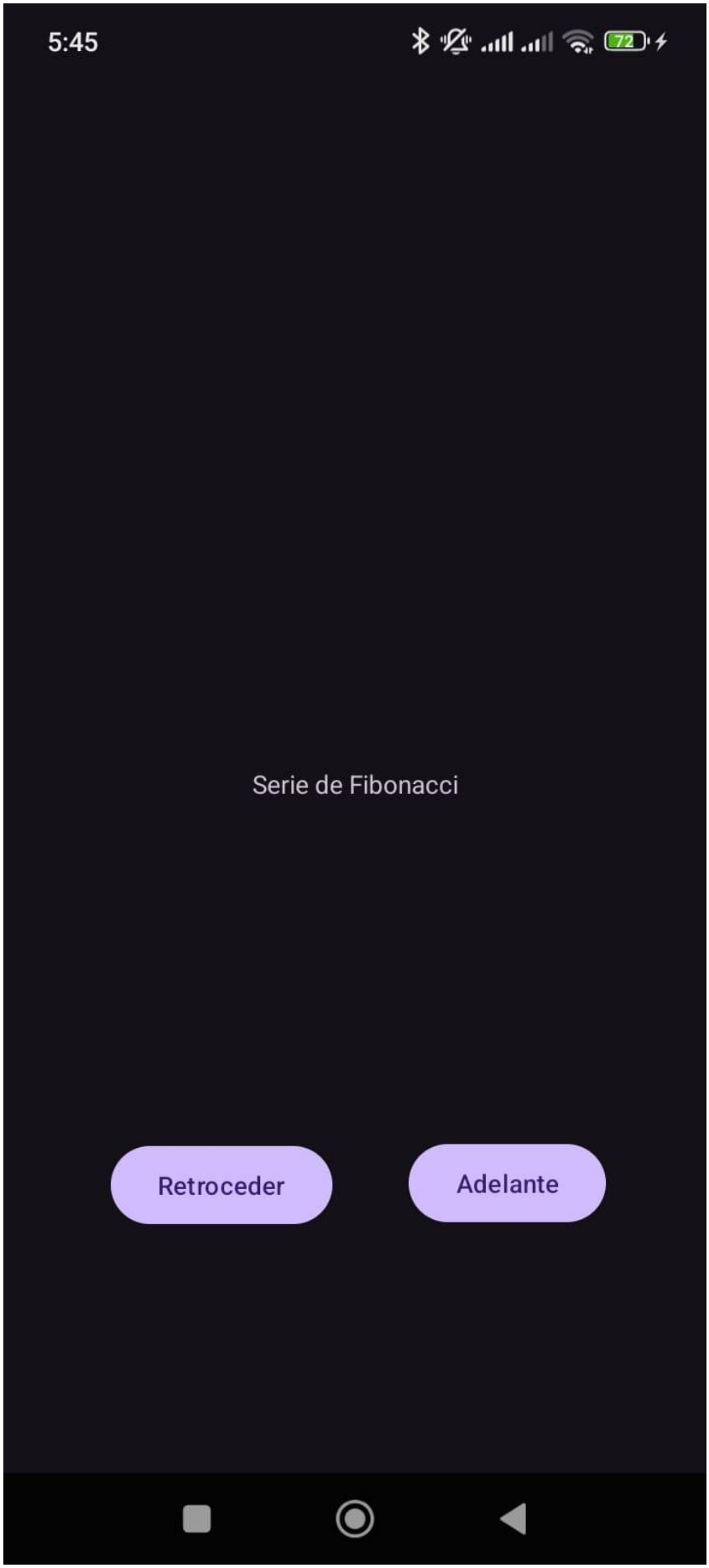
3.7. Finalmente, programamos el método que nos ayudara a calcular los números de la serie de Fibonacci

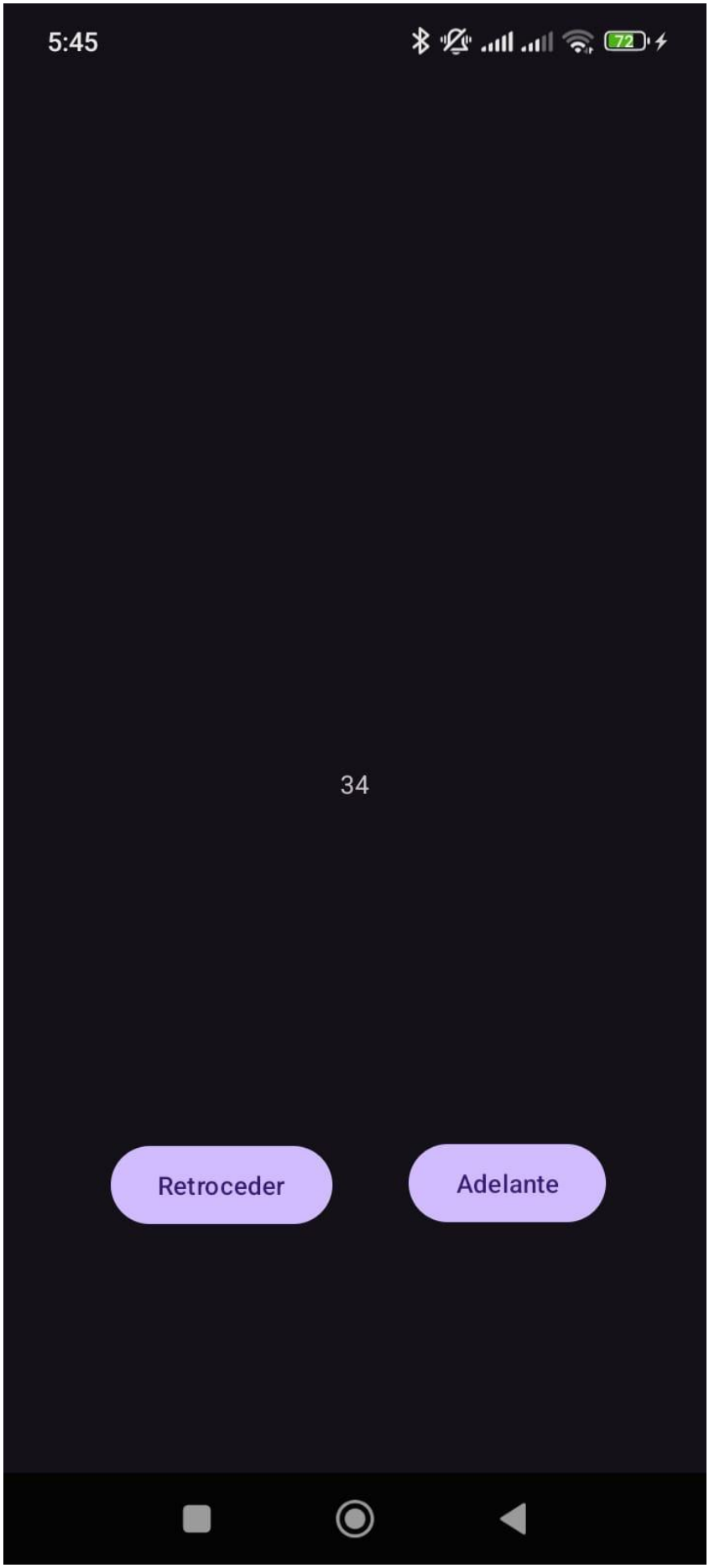
```java
public long fibonacci(int n) {
    if (n <= 1) return n;

    // Si el valor ya está en el mapa, lo retornamos
    if (memo.containsKey(n)) return memo.get(n);

    //Calculo recursivo y almacenamiento del resultado
    long result = fibonacci( n: n - 1) + fibonacci( n: n - 2);
    memo.put(n, result);
    return result;
}
```

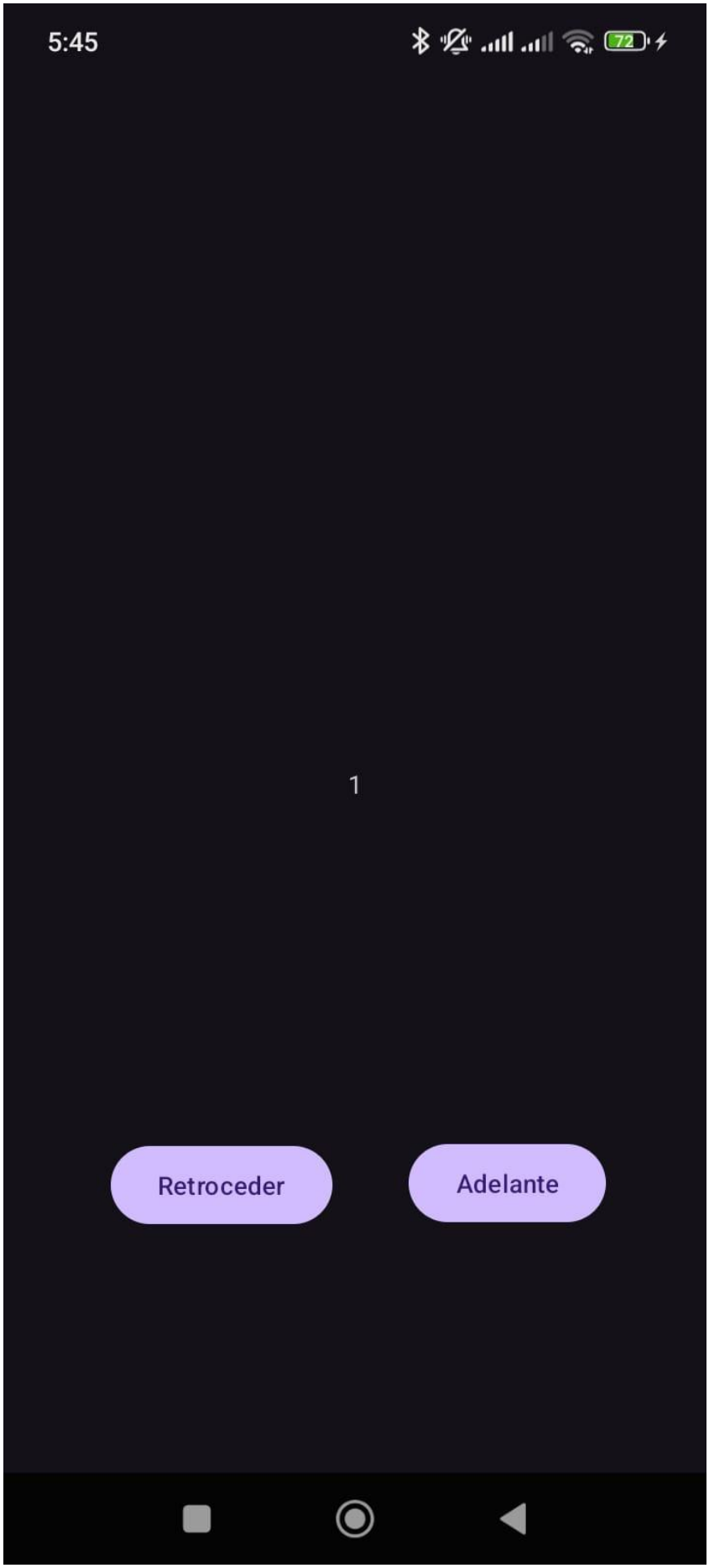
Este código hace lo siguiente:

- Cuando n es 0 o 1, regresa directamente n, ya que es el caso base de la recursión
- Memoización, se revisa si el resultado de n ya se había calculado con anterioridad, si sí, esta guardado en el HashMap llamado memo, esto para después devolverlo sin demorar en volver a calcularlo
- Se tiene una llamada recursiva que calcula el valor de la serie sumado los valores anteriores
- Posteriormente se guarda el resultado en el HashMap para no tener que volver a calcularlo
- Se retorna el resultado de n.

4. Evidencia de la aplicación


5:45
72
Serie de Fibonacci
Retroceder
Adelante


5:45
72
34
Retroceder
Adelante


5:45
72
1
Retroceder
Adelante