



Universidad Nacional Autónoma de México
Facultad de Contaduría y Administración
Sistema Universidad Abierta y Educación a Distancia

Alumno: López Acevedo Víctor Rafael

Grupo: 9696

Materia: INFORMATICA VI

Unidad: 1

Actividad: M1-03

Fecha: 29 de marzo de 2025

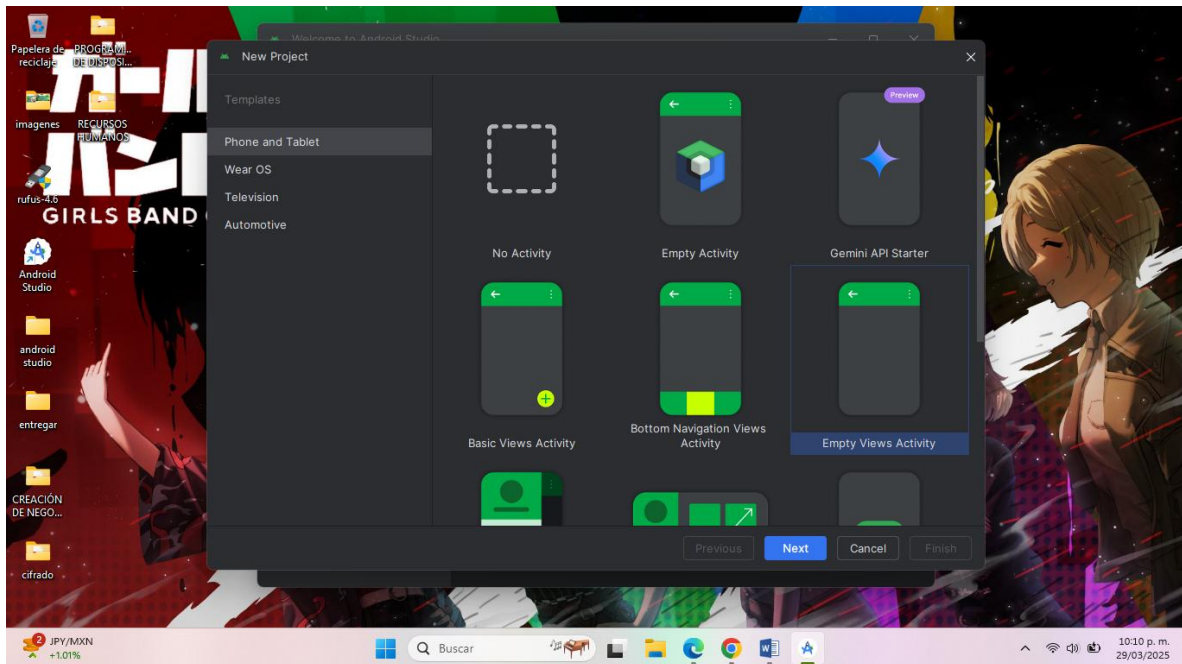
Actividad: Arreglos y tiempo de ejecución

Objetivo del aprendizaje: Ejecutar una sentencia básica de programación en lenguaje Java

Actividad sugerida:

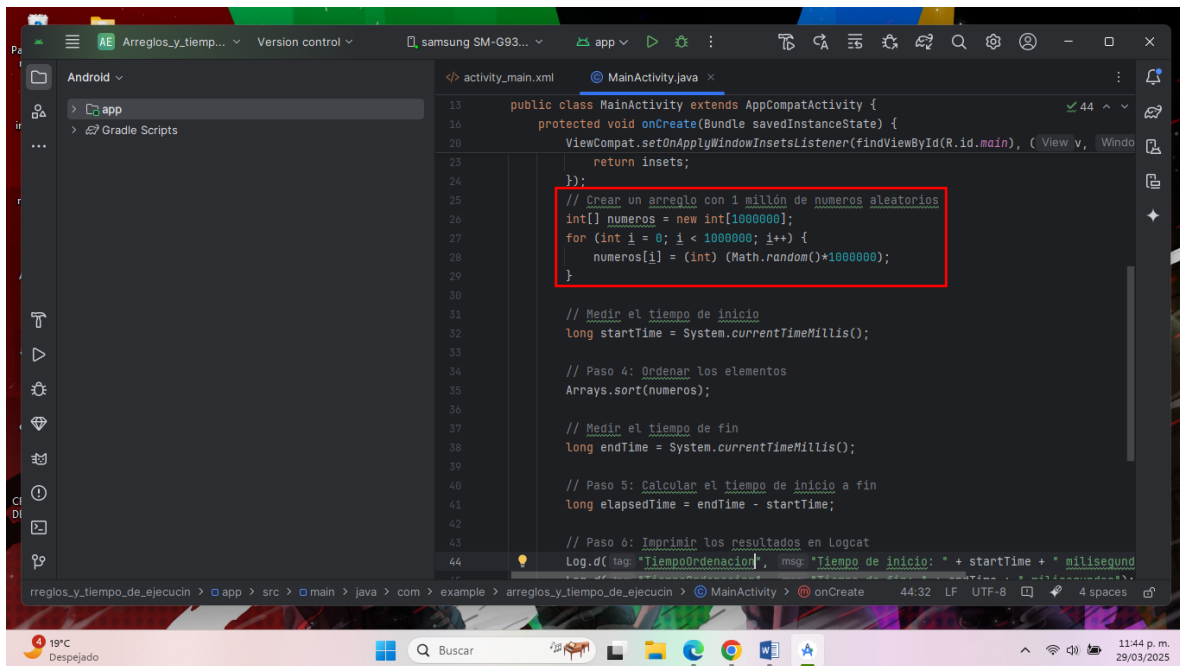
1. Crear tu primer activity

2. En lenguaje de programación JAVA dentro de Android Studio, crea una app vacía.



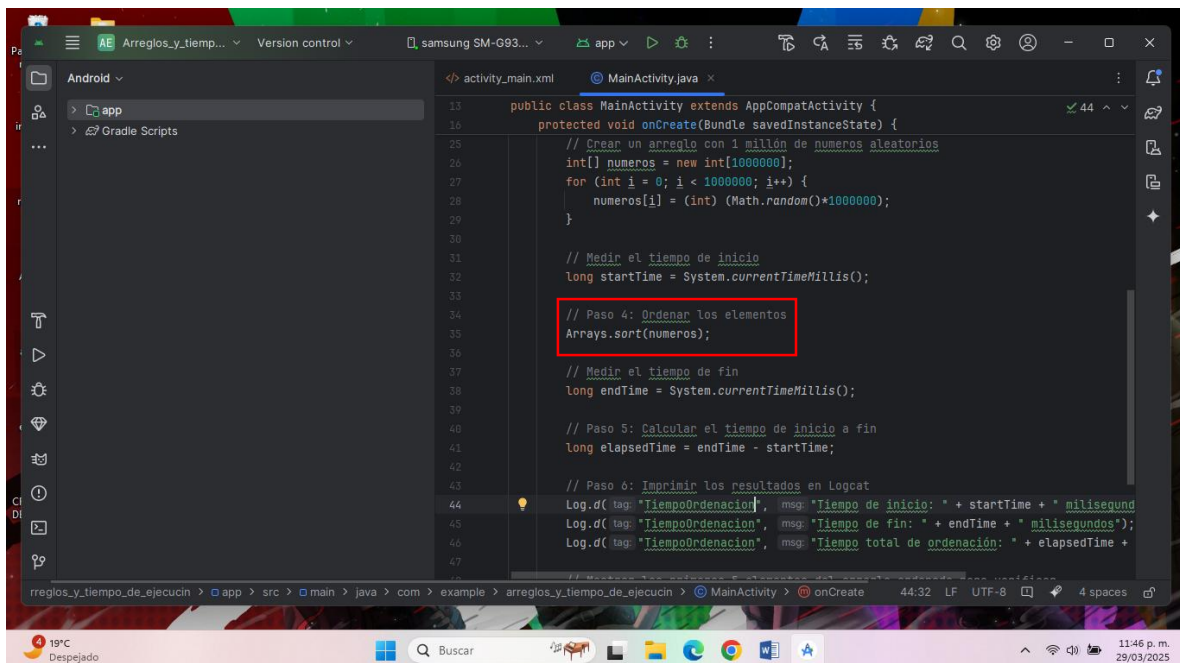
Seleccionamos la opción “empty views activity” para crear la aplicación vacía

3. Crear un arreglo con 1 millón de elementos enteros de manera aleatoria.



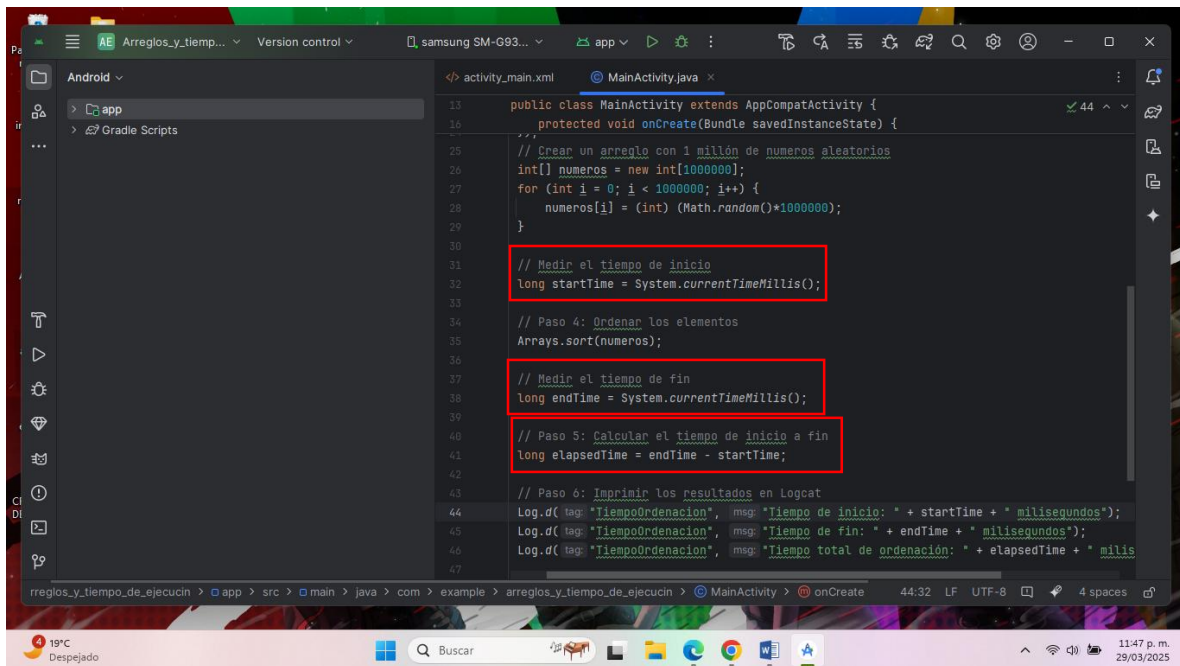
```
13 public class MainActivity extends AppCompatActivity {
14     protected void onCreate(Bundle savedInstanceState) {
15         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsets insets) -> insets);
16         return insets;
17     }
18
19     // Crear un arreglo con 1 millón de numeros aleatorios
20     int[] numeros = new int[1000000];
21     for (int i = 0; i < 1000000; i++) {
22         numeros[i] = (int) (Math.random()*1000000);
23     }
24
25     // Medir el tiempo de inicio
26     long startTime = System.currentTimeMillis();
27
28     // Paso 4: Ordenar los elementos
29     Arrays.sort(numeros);
30
31     // Medir el tiempo de fin
32     long endTime = System.currentTimeMillis();
33
34     // Paso 5: Calcular el tiempo de inicio a fin
35     long elapsedTime = endTime - startTime;
36
37     // Paso 6: Imprimir los resultados en Logcat
38     Log.d("tag: \"TiempoOrdenacion\", msg: \"Tiempo de inicio: \" + startTime + \" milisegund
39
40 reglos_y_tiempo_de_ejecucin > app > src > main > java > com > example > arreglos_y_tiempo_de_ejecucin > MainActivity > onCreate 44:32 LF UTF-8 4 spaces
```

4. Ordenar los elementos.



```
13 public class MainActivity extends AppCompatActivity {
14     protected void onCreate(Bundle savedInstanceState) {
15         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsets insets) -> insets);
16         return insets;
17     }
18
19     // Crear un arreglo con 1 millón de numeros aleatorios
20     int[] numeros = new int[1000000];
21     for (int i = 0; i < 1000000; i++) {
22         numeros[i] = (int) (Math.random()*1000000);
23     }
24
25     // Medir el tiempo de inicio
26     long startTime = System.currentTimeMillis();
27
28     // Paso 4: Ordenar los elementos
29     Arrays.sort(numeros);
30
31     // Medir el tiempo de fin
32     long endTime = System.currentTimeMillis();
33
34     // Paso 5: Calcular el tiempo de inicio a fin
35     long elapsedTime = endTime - startTime;
36
37     // Paso 6: Imprimir los resultados en Logcat
38     Log.d("tag: \"TiempoOrdenacion\", msg: \"Tiempo de inicio: \" + startTime + \" milisegund
39     Log.d("tag: \"TiempoOrdenacion\", msg: \"Tiempo de fin: \" + endTime + \" milisegundos\");
40     Log.d("tag: \"TiempoOrdenacion\", msg: \"Tiempo total de ordenación: \" + elapsedTime +
41
42 reglos_y_tiempo_de_ejecucin > app > src > main > java > com > example > arreglos_y_tiempo_de_ejecucin > MainActivity > onCreate 44:32 LF UTF-8 4 spaces
```

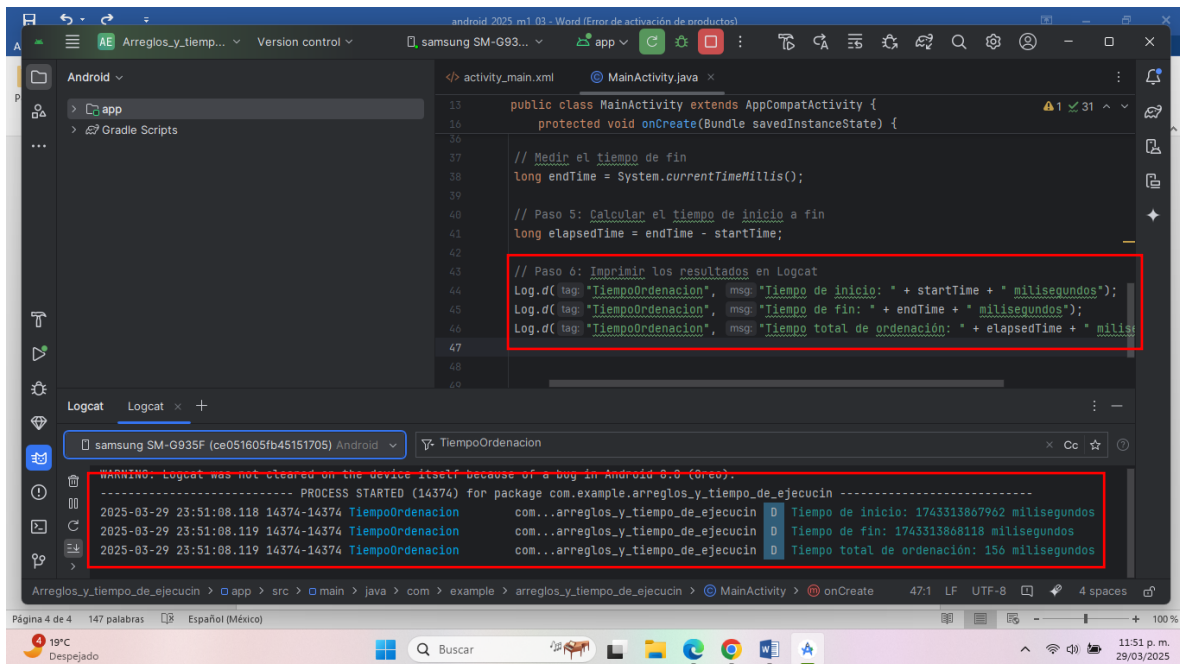
5. Medir el tiempo de inicio a fin.



```
13 public class MainActivity extends AppCompatActivity {
14     protected void onCreate(Bundle savedInstanceState) {
15
16         // Crear un arreglo con 1 millón de numeros aleatorios
17         int[] numeros = new int[1000000];
18         for (int i = 0; i < 1000000; i++) {
19             numeros[i] = (int) (Math.random()*1000000);
20         }
21
22         // Medir el tiempo de inicio
23         long startTime = System.currentTimeMillis();
24
25         // Paso 4: Ordenar los elementos
26         Arrays.sort(numeros);
27
28         // Medir el tiempo de fin
29         long endTime = System.currentTimeMillis();
30
31         // Paso 5: Calcular el tiempo de inicio a fin
32         long elapsedTime = endTime - startTime;
33
34         // Paso 6: Imprimir los resultados en Logcat
35         Log.d("TiempoOrdenacion", msg: "Tiempo de inicio: " + startTime + " milisegundos");
36         Log.d("TiempoOrdenacion", msg: "Tiempo de fin: " + endTime + " milisegundos");
37         Log.d("TiempoOrdenacion", msg: "Tiempo total de ordenación: " + elapsedTime + " milisegundos");
38     }
39 }
```

Aquí ponemos el tiempo de inicio y el tiempo final para realizar la resta y que muestre el tiempo que tardó en ejecutarse y realizar la ordenación

6. Imprimir en el logcat los resultados obtenidos.



```
2025-03-29 23:51:08.118 14374-14374 TiempoOrdenacion com...arreglos_y_tiempo_de_ejecucin D Tiempo de inicio: 1743313867962 milisegundos
2025-03-29 23:51:08.119 14374-14374 TiempoOrdenacion com...arreglos_y_tiempo_de_ejecucin D Tiempo de fin: 1743313868118 milisegundos
2025-03-29 23:51:08.119 14374-14374 TiempoOrdenacion com...arreglos_y_tiempo_de_ejecucin D Tiempo total de ordenación: 150 milisegundos
```

Aquí podemos ver que se imprimió en el logcat los tiempos de inicio y fin y muestra el resultado que es el tiempo total

Código usado

```
int[] numeros = new int[1000000];  
for (int i = 0; i < 1000000; i++) {  
    numeros[i] = (int) (Math.random()*1000000);  
}  
  
long startTime = System.currentTimeMillis();  
  
Arrays.sort(numeros);  
  
long endTime = System.currentTimeMillis();  
  
long elapsedTime = endTime - startTime;  
  
Log.d("TiempoOrdenacion", "Tiempo de inicio: " + startTime + " milisegundos");  
Log.d("TiempoOrdenacion", "Tiempo de fin: " + endTime + " milisegundos");  
Log.d("TiempoOrdenacion", "Tiempo total de ordenación: " + elapsedTime + "  
milisegundos");
```