

Instituto Tecnológico de San Juan del Río



Tópicos de Ciberseguridad

Tema 1

R003

Documentación del proceso

P R E S E N T A:

Equipo Púrpura

PERIODO
AGOSTO-DICIEMBRE 2025

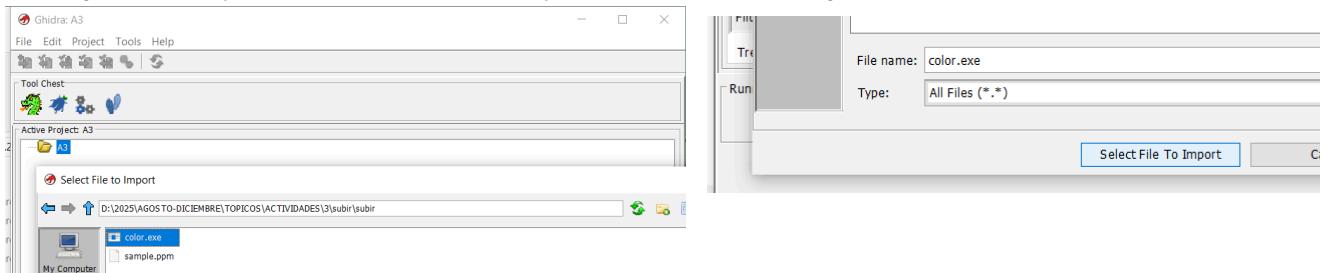


Proceso de obtención del código fuente y generación de la imagen PPM

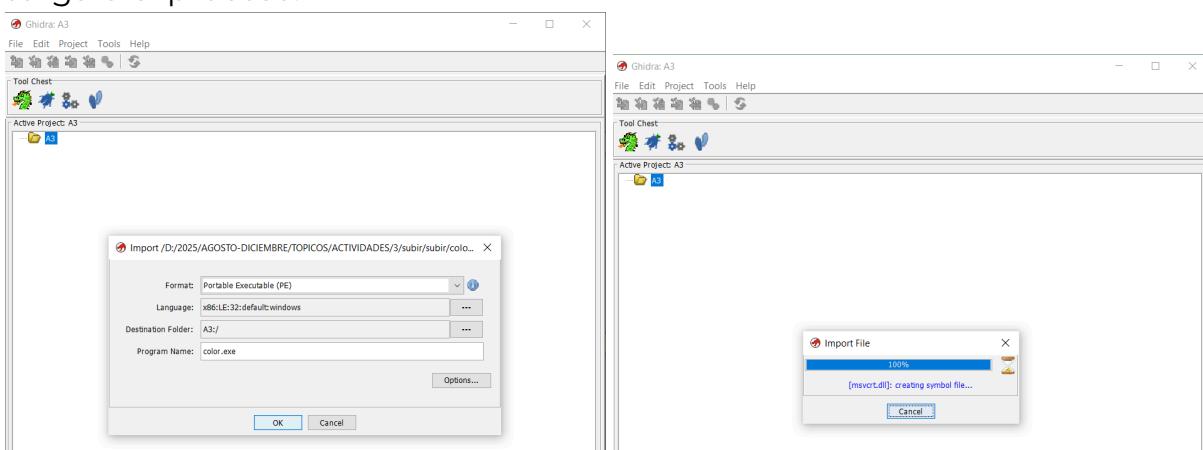
1. Reconstrucción del código con Ghidra

El profesor nos proporcionó el archivo ejecutable **color.exe** que, al ejecutarse, genera un archivo llamado **sample.ppm** (una imagen PPM que se visualizó con una herramienta en línea).

La utilización de Ghidra fue de la siguiente manera: se creó una carpeta de trabajo en la que se seleccionó importar el archivo ejecutable.



Los siguientes pasos fueron únicamente aceptar lo solicitado y esperar a que cargara el proceso.



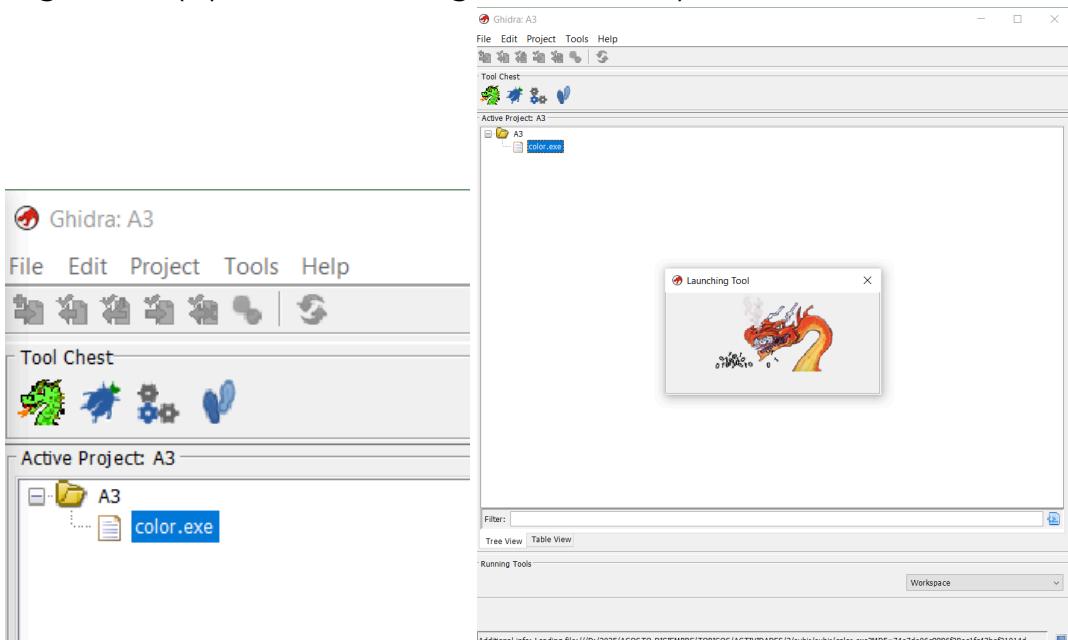
Después aparece un resumen adicional de la información del archivo.

```

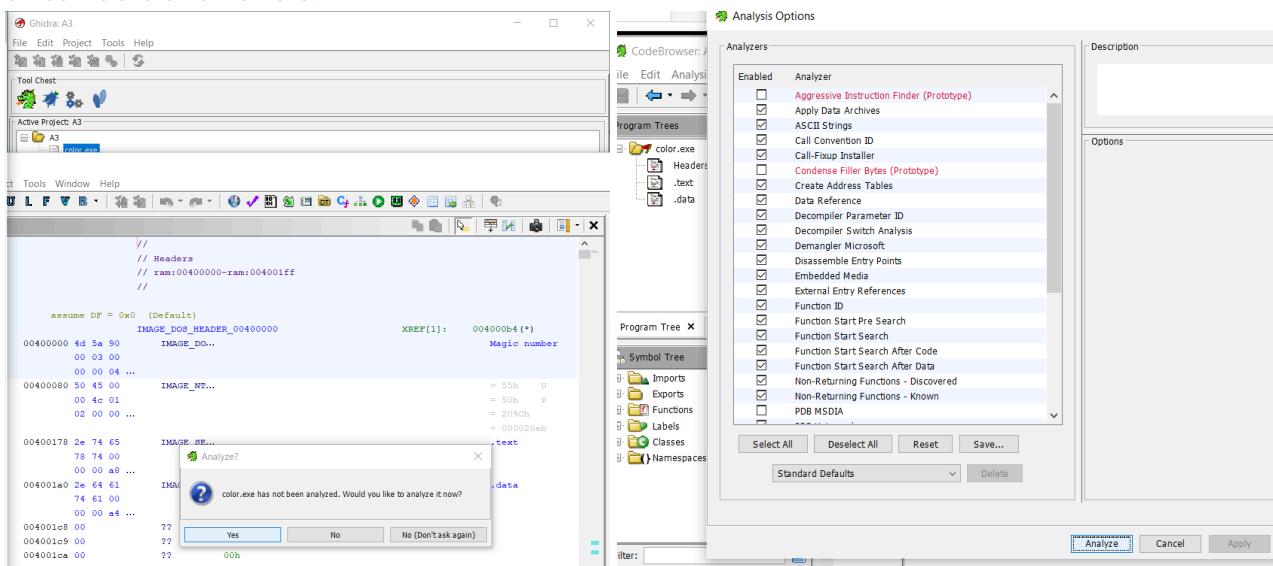
Project File Name: color.exe
Last Modified: Fri Sep 19 14:38:15 CEST 2025
 Readonly: false
Program Name: color.exe
Language ID: x86;LE;32:default (4.6)
Country ID: windows
Processor: x86
Endian: Little
Address Size: 32
Minimum Address: 00400000
Maximum Address: 004021ff
# of Bytes: 2048
# of Memory Blocks: 3
# of Instructions: 0
# of Defined Data: 74
# of Functions: 13
# of Symbols: 18
# of Data Types: 24
# of Data Type Categories: 3
Compiler: visualstudio:unknown
Created With Ghidra Version: 11.4.2
Date Created: Fri Sep 19 14:38:11 CEST 2025
Date Last Saved: Fri Sep 19 14:38:11 CEST 2025
Executable Format: Portable Executable (PE)
Executable Location: /D:/2025/AGOSTO-DICIEMBRE/TOPICOS/ACTIVIDADES/3/subir/color.exe
Executable MD5: 74e0de6c5986f28e1fc42beff21014d
Executable SHA256: ed1fd809e5604ea894732f2b0feec0852a5d2a1a9ebfcfa8e6c47746b7ed2ac
File:///D:/2025/AGOSTO-DICIEMBRE/TOPICOS/ACTIVIDADES/3/subir/color.exe?NDS=74e7de06c598
Selected Root Namespace Category: false
Relocatable: 4096
SectionAlignment: 4096

Additional Information
Loading file:///D:/2025/AGOSTO-DICIEMBRE/TOPICOS/ACTIVIDADES/3/subir/color.exe?NDS=74e7de06c598
-----
Searching 19 paths for library MAVCRT.DLL...
Loading file:///C:/Windows/system32/mavcrt.dll?NDS=c7815c121632e8ce3ad8019f2cff910...
Created export file: C:\Users\luca\appdata\Roaming\ghidra\ghidra_11.4.2_FUMLIC\symbols\win32\mavcrt
Library not saved to project.
-----
```

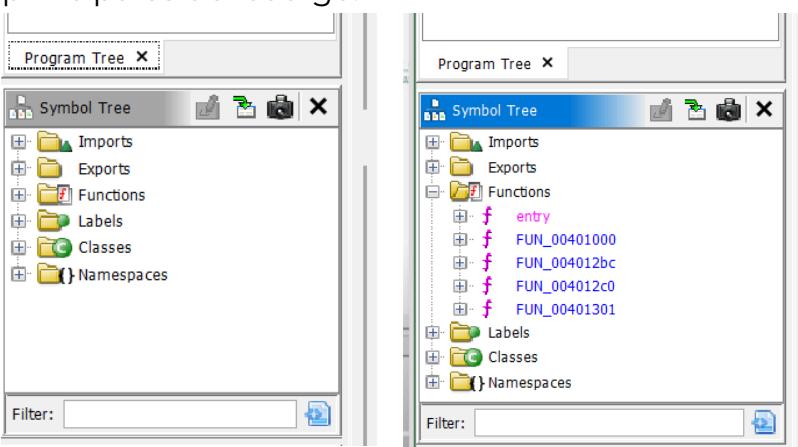
Para abrir **color.exe** sólo se hace clic sobre el archivo y se esperan unos segundos. (Aparece esa imagen de Ghidra)



Se acepta para que Ghidra reconstruya el código. Y se confirman algunos criterios del análisis.



En la sección que se muestra nos deja acceder a cada una de las funciones principales del código.



2. Análisis del código fuente C

Por ejemplo aquí se muestra cuando seleccionamos la primera función para visualizarla.

Para generar el código se realizó un análisis de las funciones. Se identificó la función principal (**entry**) y las funciones auxiliares tal y como se ven en la imagen.

```

1 undefined4 FUN_00401000(void)
2 {
3     int iVar1;
4     time_t tVar2;
5
6     tVar2 = time((time_t *)0x0);
7     DAT_00402190 = (uint)tVar2;
8     srand(DAT_00402190);
9
10    DAT_0040219c = fopen("e_sample.ppm", "w");
11    fwrite(&DAT_00402000, 1, 0x1f, DAT_0040219c);
12    for (DAT_00402010 = 0; DAT_00402010 < 0x1e0; DAT_00402010++)
13        for (DAT_00402014 = 0; DAT_00402014 < 0x280; DAT_00402014++)
14            iVar1 = rand();
15            DAT_0040200f = (undefined1)(iVar1 & 0x100);
16            fwrite(&DAT_0040200f, 1, 1, DAT_0040219c);
17            iVar1 = rand();
18            DAT_0040200f = (undefined1)(iVar1 & 0x100);
19            fwrite(&DAT_0040200f, 1, 1, DAT_0040219c);
20            iVar1 = rand();
21            DAT_0040200f = (undefined1)(iVar1 & 0x100);
22            fwrite(&DAT_0040200f, 1, 1, DAT_0040219c);
23            fwrite(&DAT_0040200f, 1, 1, DAT_0040219c);
24        }
25    fclose(DAT_0040219c);
26    return 0;
27}
28
29

```

Lo que se identificó fue que la función con nombre FUN_00401000() contiene la lógica para generar la imagen PPM.

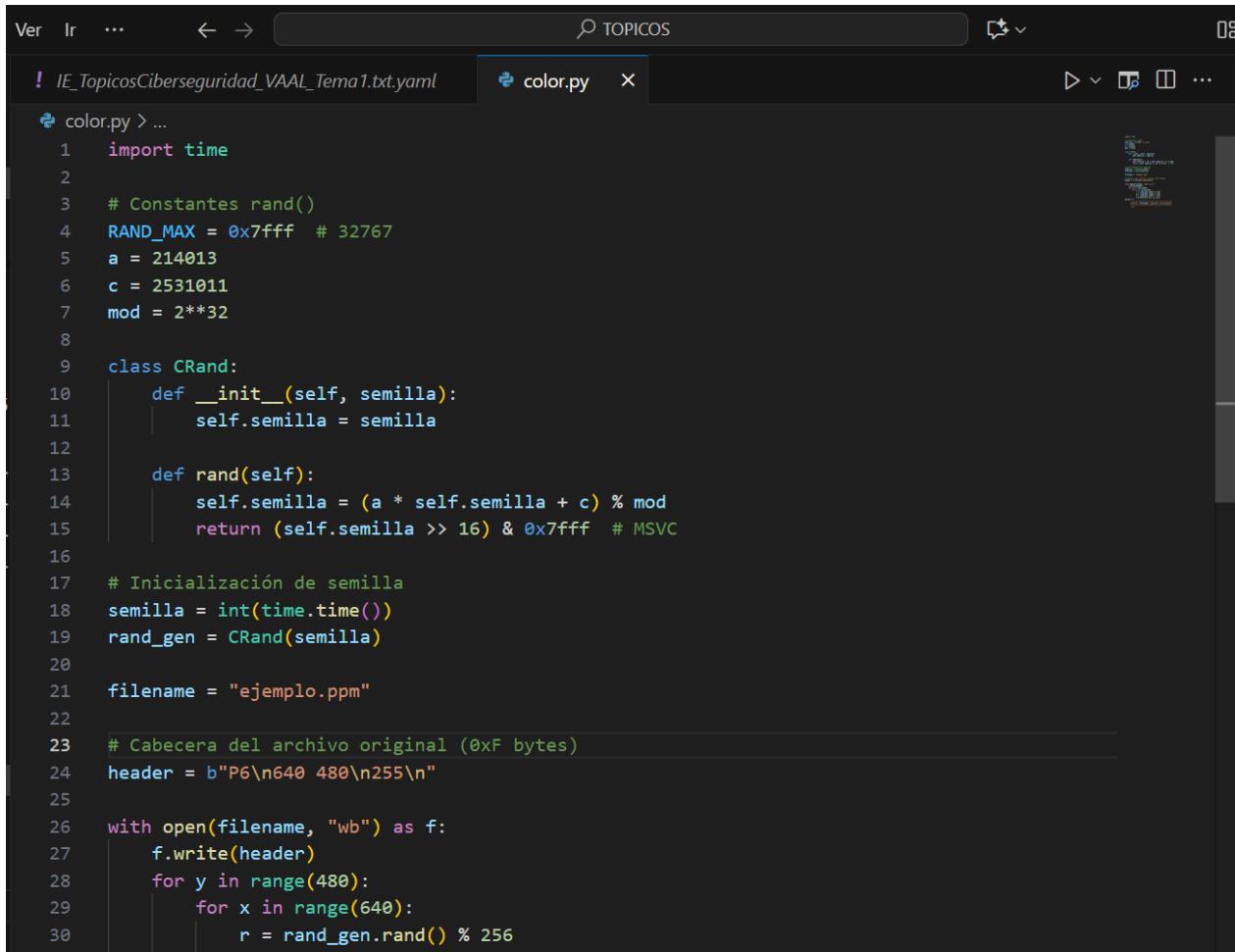
Las funciones con nombres, FUN_00401301(), FUN_004012bc(), FUN_004012c0(): son para el manejo de stack y estructuras de inicio, pero que no se relacionan directamente con la creación de la imagen.

La estructura del código para la generación de la imagen es que el archivo se abre en modo binario, se escribe un encabezado PPM fijo y se generan los pixeles con valores aleatorios usando rand().

Se determinaron las dimensiones de la imagen del ejecutable original:

- 480 filas (0x1e0)
- 640 columnas (0x280)

Se observó que la semilla usada en srand() es la hora del sistema (time(NULL)), igual que se hace en color.exe.



```

! IE_TopicosCiberseguridad_VAAL_Tema1.txt.yaml
color.py x

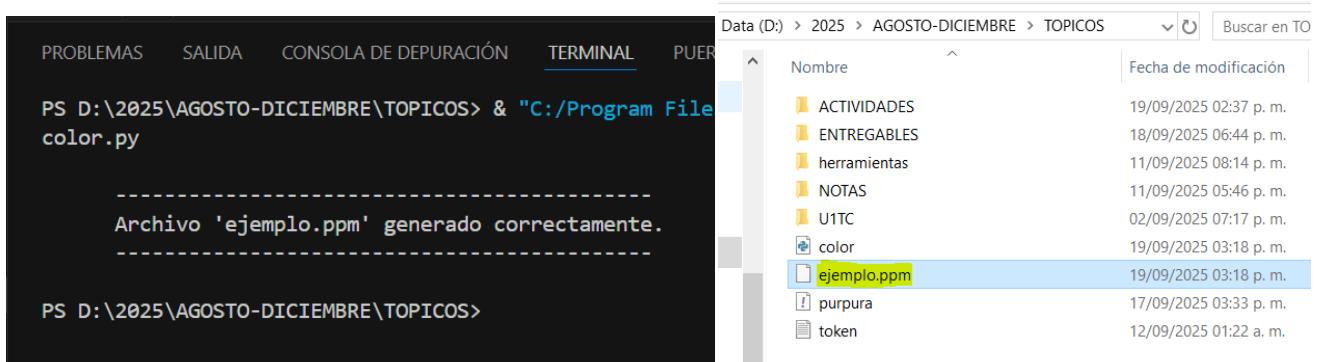
color.py > ...
1 import time
2
3 # Constantes rand()
4 RAND_MAX = 0x7fff # 32767
5 a = 214013
6 c = 2531011
7 mod = 2**32
8
9 class CRand:
10     def __init__(self, semilla):
11         self.semilla = semilla
12
13     def rand(self):
14         self.semilla = (a * self.semilla + c) % mod
15         return (self.semilla >> 16) & 0x7fff # MSVC
16
17 # Inicialización de semilla
18 semilla = int(time.time())
19 rand_gen = CRand(semilla)
20
21 filename = "ejemplo.ppm"
22
23 # Cabecera del archivo original (0xF bytes)
24 header = b"P6\n640 480\n255\n"
25
26 with open(filename, "wb") as f:
27     f.write(header)
28     for y in range(480):
29         for x in range(640):
30             r = rand_gen.rand() % 256

```

El programa conserva las funciones que identificamos como necesarias para generar la imagen: una función para abrir un archivo, una que ayuda a escribir el encabezado PPM y generar los píxeles aleatorios.

Para que el resultado fuera el esperado, se implementó el mismo generador aleatorio (rand() de MSVC).

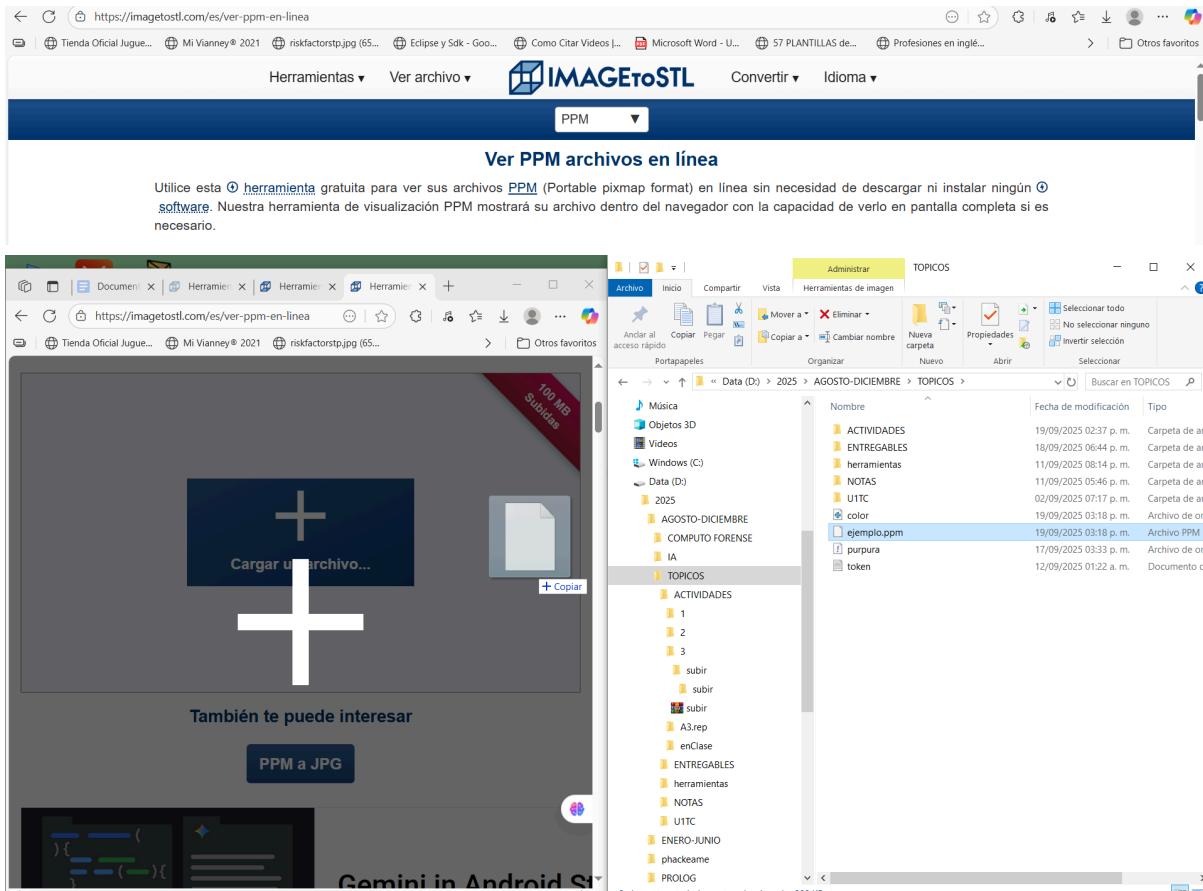
Entonces `colorNuevo.py` crea la imagen de 480×640 píxeles y al final muestra un mensaje confirmando que el archivo se generó correctamente.



Nombre	Fecha de modificación
ACTIVIDADES	19/09/2025 02:37 p. m.
ENTREGABLES	18/09/2025 06:44 p. m.
herramientas	11/09/2025 08:14 p. m.
NOTAS	11/09/2025 05:46 p. m.
U1TC	02/09/2025 07:17 p. m.
color	19/09/2025 03:18 p. m.
ejemplo.ppm	19/09/2025 03:18 p. m.
purpura	17/09/2025 03:33 p. m.
token	12/09/2025 01:22 a. m.

R003

Finalmente, como la ejecución nos genera un archivo PPM, se utilizó una herramienta en línea llamada IMAGE To STL para visualizar ambas imágenes y compararlas.



Se adjuntan ambas imágenes, la primera se generó con **color.exe** y la otra con nuestro programa **color.py** y podemos observar que la segunda imagen es como la del ejecutable original, cumpliendo con los requisitos de la actividad.

