



# RUBRICA 2

PROGRAMACION CONCURRENTES

UNIVERSIDAD POLITECNICA DE TECAMAC  
MAESTRO: RAMIREZ FUENTES  
BENITO MATRICULA:  
1320114042  
GRUPO: 2722IS

ESTUDIANTE ERIK SAMUEL HERNÁNDEZ  
GARCÍA

## INDICE

INTRODUCCION.....	3
DESCRIPCION DEL PROBLEMA .....	4
DISEÑO .....	5
EJECUCION DEL PROGRAMA .....	8
MAPA.....	10
CONCLUSION .....	12
PORYECTO EN GIT .....	13

## INDICE DE FIGURAS

Figura 1 DISEÑO 1.....	5
Figura 2 DISEÑO 2.....	5
Figura 3 LINEA DE CODIGO 1.....	6
Figura 4 LINEA DE CODIGO 2.....	6
Figura 5 LINEA DE CODIGO 3.....	7
Figura 6 PRUEBA 1.....	8
Figura 7 PRUEBA 2.....	8
Figura 8 PRUEBA 3.....	9
Figura 9 PRUEBA 4.....	9
Figura 10 MAPA .....	10
Figura 11 RUBRIC.....	11

## INTRODUCCION

El presente documento sirve para plasmar un poco sobre el desarrollo del programa sobre los filósofos comensales, el cual se explicará con mayor detalle en la descripción del problema

Para poder realizar este tipo de problema es necesario conocer lo básico hacer de la programación concurrente, lo cual abarca desde el uso de hilos y por supuesto de semáforos. Tanto el para que sirve como el como utilizarlos.

Esto es indispensable para saber si es necesario la creación de hilos y si es así cuantos crear, lo cual es fácil de visualizar si se posee los conocimientos para ello.

También se podrá apreciar algunas funciones importantes al momento de desarrollar el código como los comentarios para identificar sobre que trata dicha línea de código

Y así que las personas que no están muy familiarizadas en este aspecto puedan comprender un poco sobre lo que se tratando el reporte. Sin mas que comentar por el momento se pone a disposición el trabajo.

## DESCRIPCION DEL PROBLEMA

La problemática que nos encontramos en el proyecto es la siguiente.

Nos encontramos una mesa la cual esta compuesta por 5 Filobarts los cuales por obvias ocasiones poseen un platillo propio y un cubierto para que puedan comer.

Pero la problemática aquí es que para que un filo-Bart pueda comer es necesario la utilización de 2 cubiertos los cuales pueden estar tanto a la izquierda como a su derecha de cada uno de ellos.

Así tomando en cuenta que para que todos puedan comer al mismo tiempo no es posible debido a que los cubiertos no alcanzarían.

Es por ello por lo que cuando uno coma es el otro se quedara esperando hasta que se desocupe para que así pueda comer

Pero al momento de intentar agarrar 2 filo-Bart el mismo cubierto esto provocara que la acción de uno de ellos no se lleve a cabo lo cual permitirá demostrar que para que puedan comer en la misma mesa es posible para 2 filósofos, entonces los otros 3 esperaran su turno en la mesa para realizar la acción

Entonces el principal problema es encontrar una forma que solo 2 filósofos puedan comer logrando que los restantes esperen el momento para así no encontrar corrupción de datos.

## DISEÑO

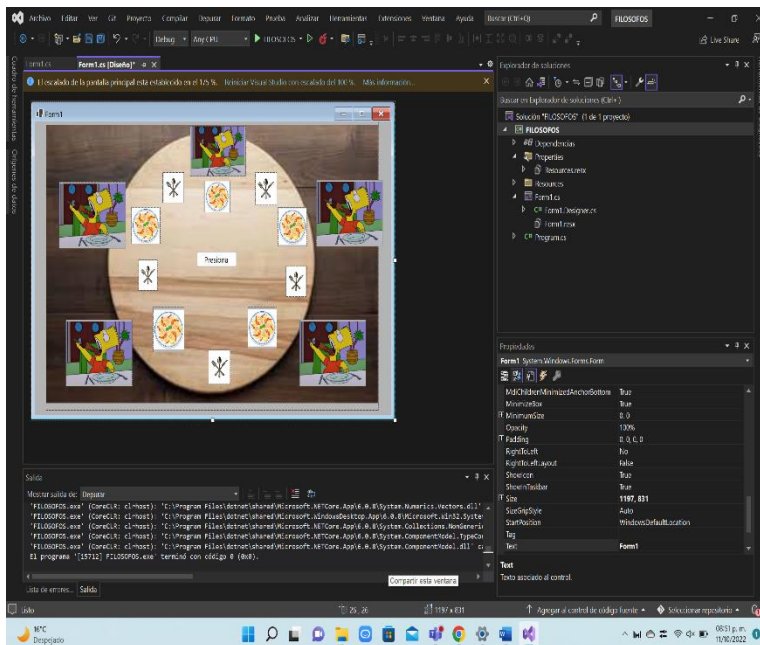


Figura 1 DISEÑO 1

Aquí podemos encontrar las pantallas finales para la ejecución del problema en esta primera imagen se muestran los bart's comelones los cuales al momento de ejecutar se esconderán para posteriormente realizar la animación que demuestre que filosofo esta comiendo

En esta otra imagen nos encontramos a los filo-Bart los cuales están esperando su turno para poder comer. Cuando se haya realizado la ejecución se realizarán las acciones pertinentes.

Las cuales podremos observar más a detalle en las pruebas

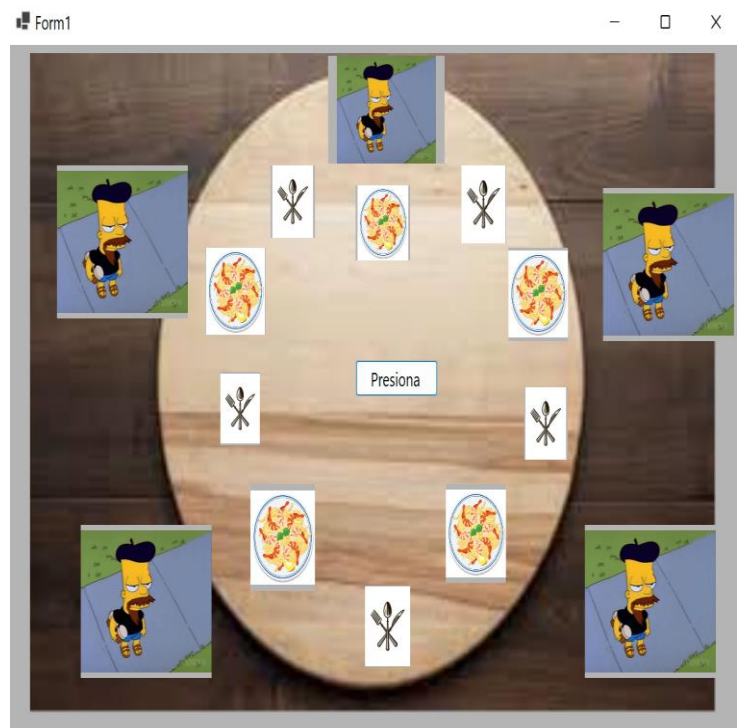


Figura 2 DISEÑO 2

```

public bool cubierto1 = true, cubierto2 = true, cubierto3 = true, cubierto4 = true, cubierto5 = true;
public Semaphore sema = new Semaphore(2, 3); // Semaphore 2, 3 sirve para indicar que dos estarn en proces
public Form1()

```

Figura 3 LINEA DE CODIGO 1

En esta parte del código podemos encontrar cuando se declara los cubiertos que se utilizaran en el código para la realización sobre la acción de comer

```

private void button1_Click(object sender, EventArgs e)
{
    //SIRVE PARA INICIAR LOS HILOS PARA PODER REPETIR 5 VECES EL PROCESO

    for (int i = 1 ; i <= 5; i++)
    {
        Thread bart1 = new Thread(filobart1);
        Thread bart2 = new Thread(filobart2);
        Thread bart3 = new Thread(filobart3);
        Thread bart4 = new Thread(filobart4);
        Thread bart5 = new Thread(filobart5);
        bart1.Start();
        bart2.Start();
        bart3.Start();
        bart4.Start();
        bart5.Start();
    }
}
//SE DECLARA LA ACCION EN CADA FILOSOFO

```

Figura 4 LINEA DE CODIGO 2

Posteriormente el Ford nos muestra la declaración de donde queremos mostrar los hilos. Nombrándolos filo-Bart.

Además inicializando los filósofos.

Ente parte del código es la fundamental en el proyecto, debido a que una vez funcionando posteriormente solo hay que copiarla y pegarla en las demás `filobart(2,3,4,5)` correspondientes

En la cual en los comentarios resaltados de color verde podemos entender para que es cada una de las líneas ingresadas en el código

```
public void filobart1()
{
    //EL SEMA ES PARA INDICAR LA ACCION DE CADA FILOSOFO
    sema.WaitOne();
    if (cubierto1 == true && cubierto2 == true)
    {
        //SE INDICA CUALES CUBIERTOS ESTAN UTILIZANDO
        cubierto1 = false;
        cubierto2 = false;
        Invoke((Delegate)new Action() =>
        {
            bart1.Visible = false;
            comelon1.Visible = true;
            cubi1.Visible = false;
            cubi2.Visible = false;
        }));
        Thread.Sleep(3500);
    }
    Invoke((Delegate)new Action() => //EL INVOKE SIRVE PARA HACER UN SUB-PROCESO
    {
        bart1.Visible = true; //ES PARA LLAMAR EL FILOSOFO
        comelon1.Visible = false; //PARA LLAMAR AL COMELON
        cubi1.Visible = true; //PARA LLAMAR LOS CUBIERTOS A UTILIZAR
        cubi2.Visible = true; //PARA LLAMAR LOS CUBIERTOS A UTILIZAR
    }));
    //Son los cubiertos con lo que comeran los filobart's
    cubierto2 = true; //HACEN REFERENCIA DE CUALES CUBIERTOS ESTAN
    cubierto3 = true;
    sema.Release(); //Con esto reinicia el semafora
}
```

Figura 5 LINEA DE CODIGO 3



## EJECUCION DEL PROGRAMA

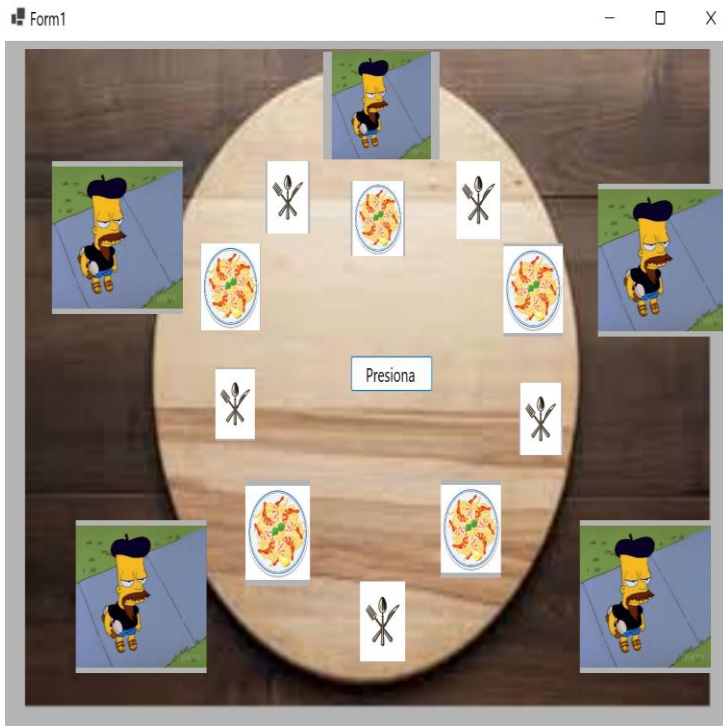


Figura 6 PRUEBA 1

En esta imagen se puede observar cual es el diseño final del problema de los filósofos.

En esta podemos observar como el Código está realizando la ejecución pertinente.

La cual es que mientras 2 filósofos está comiendo los demás se encuentran esperando a que se desocupen los cubiertos para así comer de igual manera

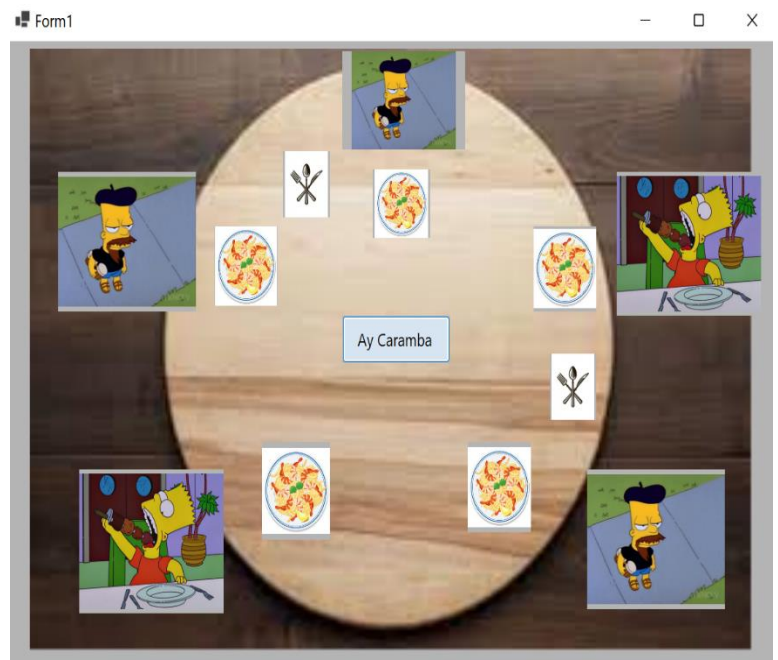


Figura 7 PRUEBA 2

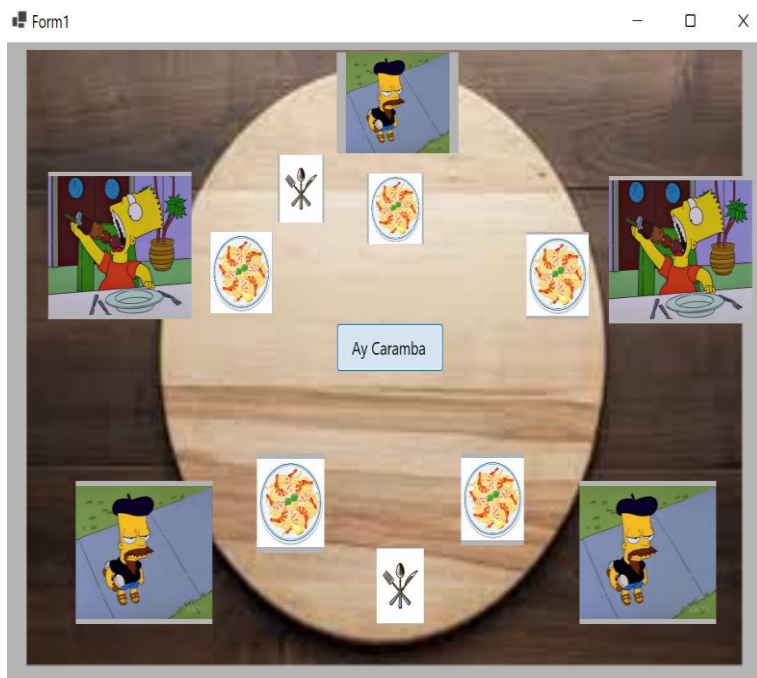


Figura 8 PRUEBA 3

Otra prueba que podemos observar es la siguiente

Para finalizar la parte de las pruebas podemos encontrar que la manera en que los filósofo está comiendo es la correcta y que cuando 2 comen se desaparecen los cubiertos correspondientes

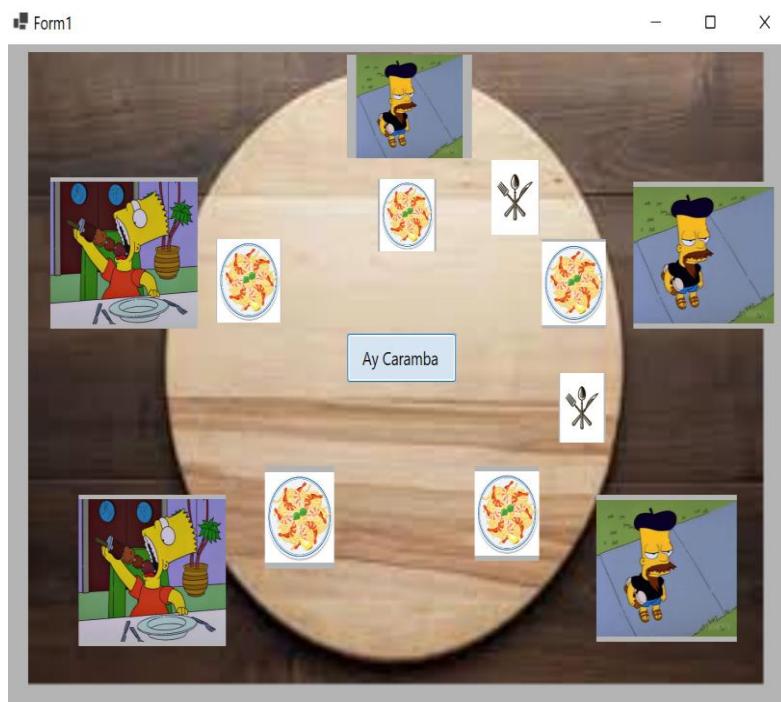


Figura 9 PRUEBA 4

## MAPA

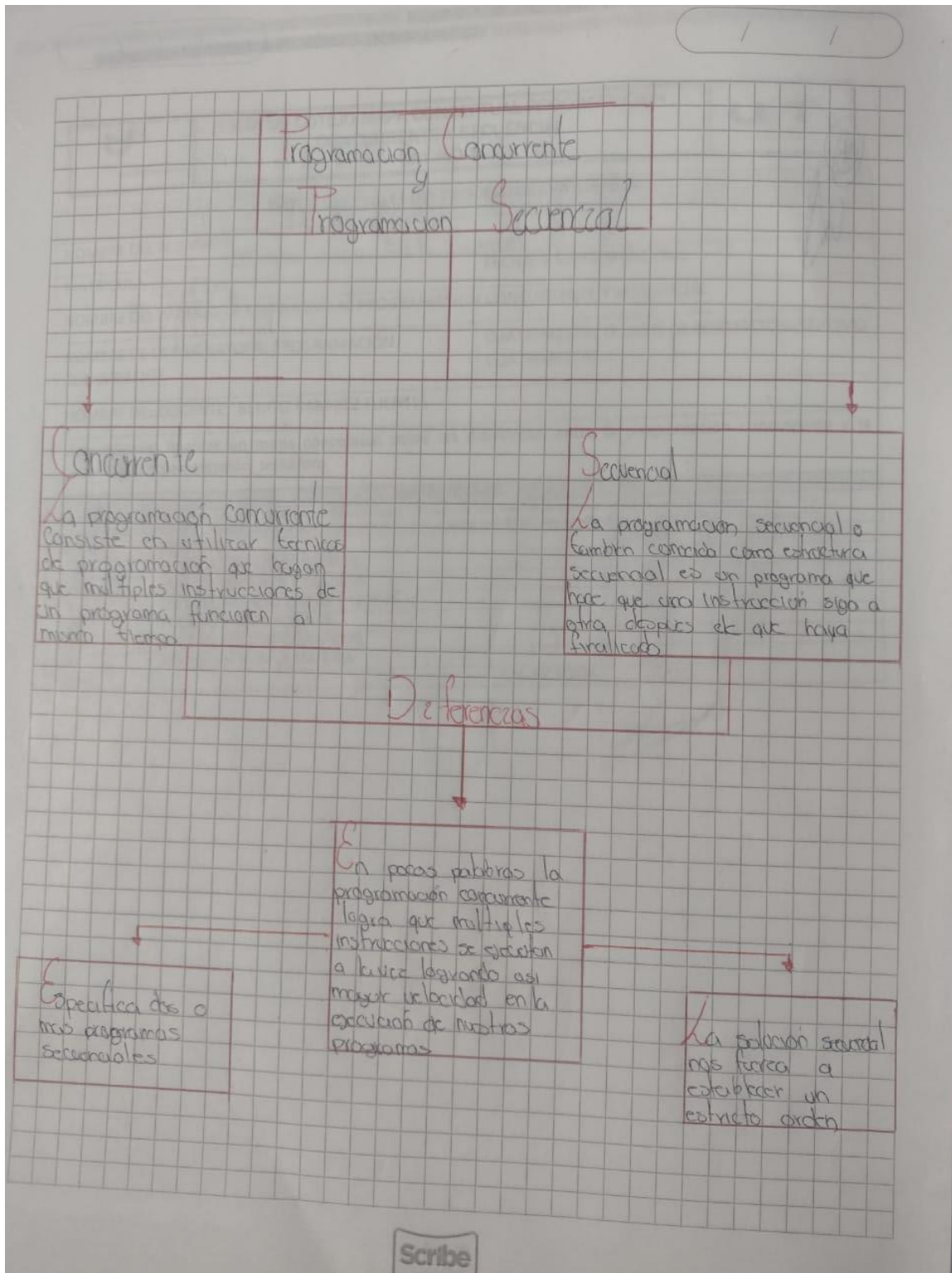




Figura 10 MAPA

 Universidad Politécnica	<b>LISTA DE COTEJO MAPA CONCEPTUAL PROGRAMACION CONCURRENTE Y SECUENCIAL</b> UNIDAD 1	 UPT UNIVERSIDAD POLITÉCNICA DE TLAXCALA	
<b>DATOS GENERALES DEL PROCESO DE EVALUACIÓN</b>			
NOMBRE DEL ALUMNO: <u>Herandez Goraa Erik Samuel</u>			
MATRICULA: <u>1320114042</u>		FECHA: <u>9 SEPTIEMBRE 2021</u>	
NOMBRE DEL PRODUCTO: <u>MAPA MENTAL PROGRAMACION CONCURRENTE Y SECUENCIAL</u>			
NOMBRE DE LA ASIGNATURA: <u>PROGRAMACIÓN CONCURRENTE</u>		CUATRIMESTRE O CICLO DE FORMACIÓN: <u>SÉPTIMO CUATRIMESTRE</u>	
NOMBRE DEL DOCENTE: <u>BENITO RAMIREZ FUENTES</u>			
Instrucciones: Realizar un mapa conceptual sobre las diferencias entre la programación concurrente y la secuencial implementación semáforo			
<b>INSTRUCCIONES</b>			
<small>Revisar las actividades que se solicitan y marque en los apartados "SI" cuando la evidencia se cumple; en caso contrario marque "NO". En la columna "OBSERVACIONES" indicaciones que puedan ayudar al alumno a saber cuáles son las condiciones no cumplidas, si fuese necesario.</small>			
VALOR	REACTIVO	CUMPLE	OBSERVACIONES
40%	CLARIDAD DE LOS CONCEPTOS	✓	
20%	USO DE IMÁGENES Y COLORES	✓	
5%	USO DEL ESPACIO LINEAS Y TEXTOS.	✓	
10%	ENFASIS Y ASOCIACIONES.	✓	
5%	SIN ERRORES ORTOGRAFICOS	✓	
20%	RELACIONA CON LA MATERIA	✓	
100%		100	




Figura 11 RUBRIC

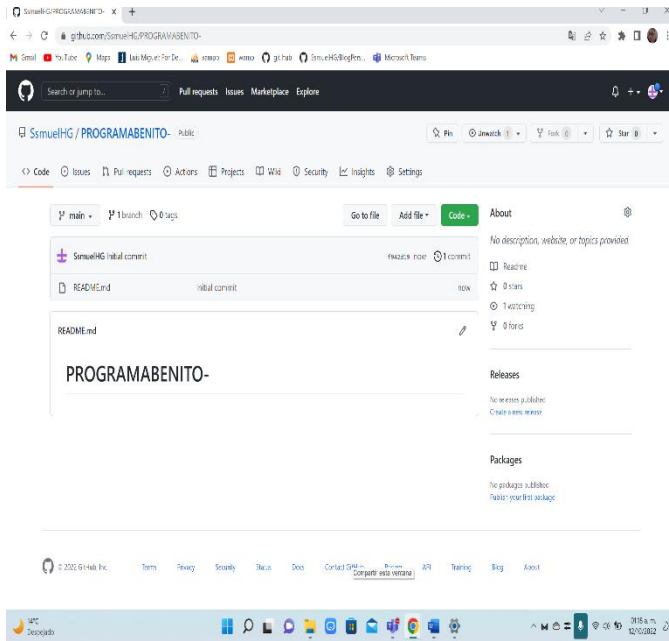
## CONCLUSION

En conclusión, podemos encontrar que para la realización de este tipo de proyectos es necesario tener los conocimientos adquiridos durante el trayecto del parcial para así poder terminar de manera satisfactoria el proyecto.

También es importante saber en qué momento es preferible la utilización de hilos y así no sobrecargar los equipos utilizados

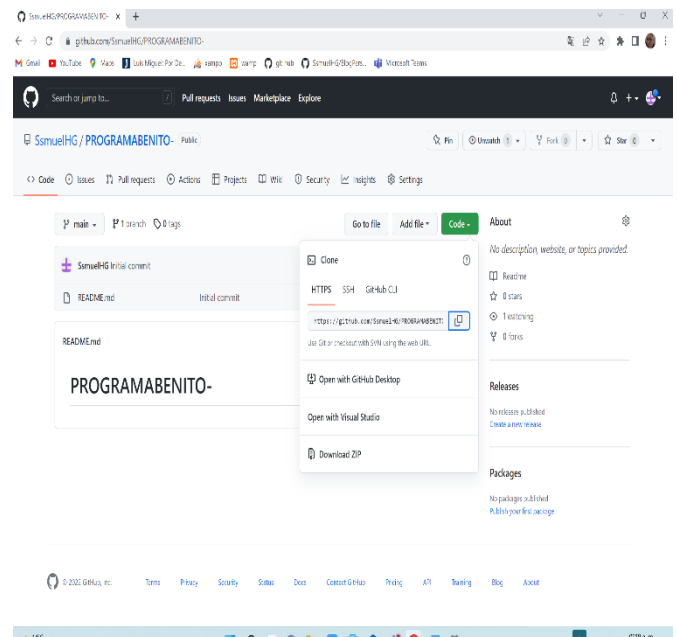


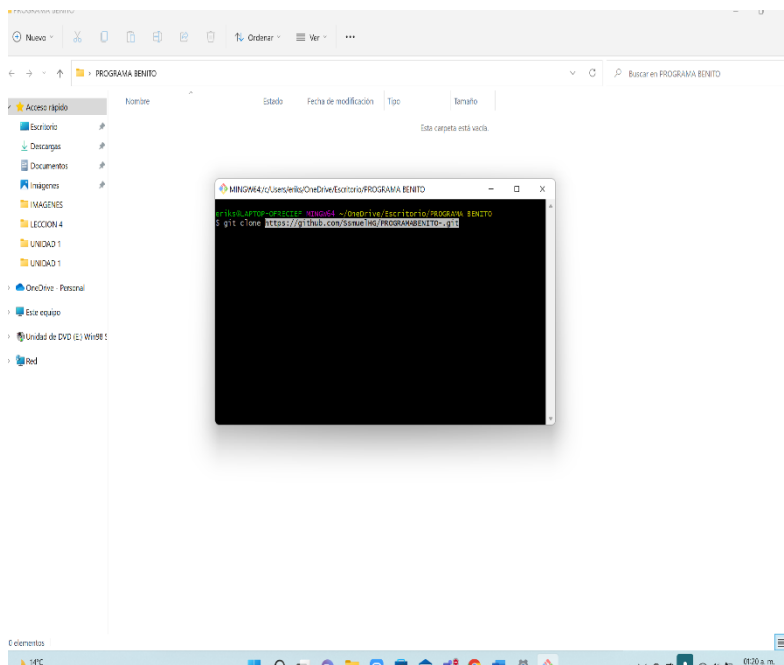
## PORYECTO EN GIT



Para poder subir un trabajo a los repositorios que git provee es necesario primero crear un nuevo repositorio con el nombre que quieras

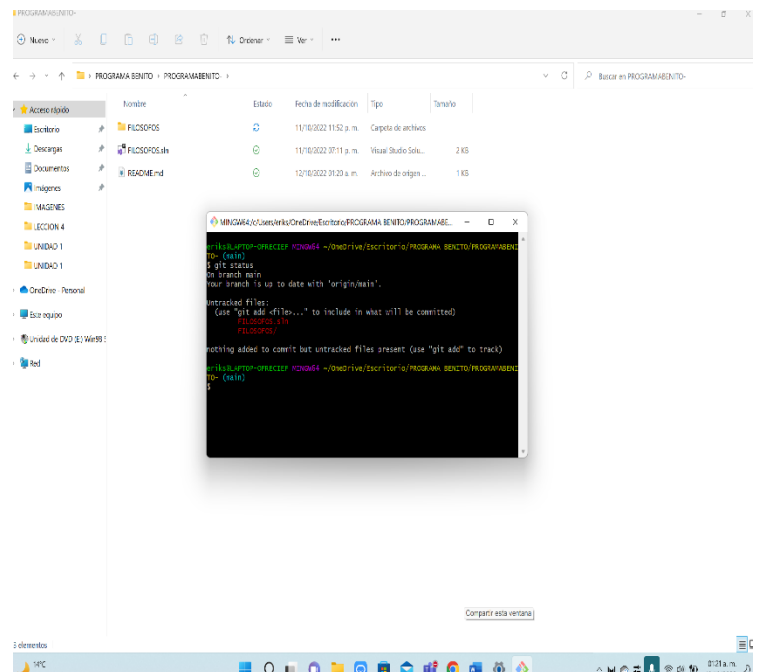
Posteriormente ir al apartado de código y ahí copiar el repositorio para después copiarlo en git bash

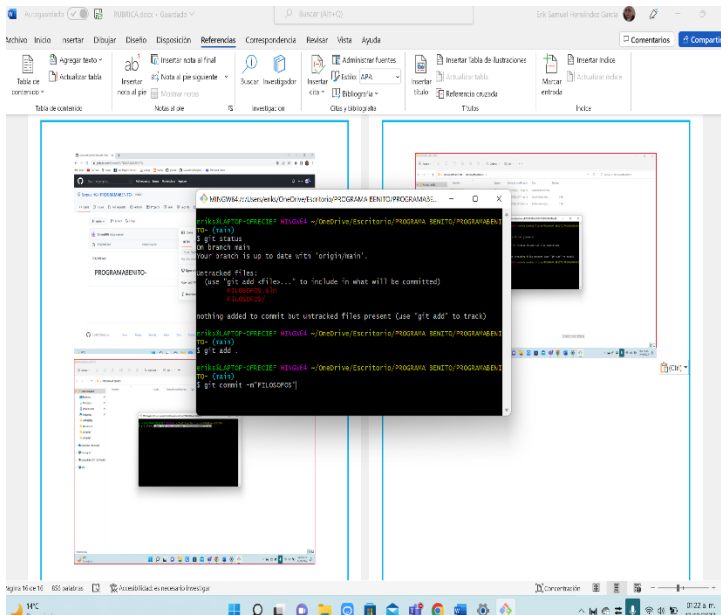




Para ellos se la da clic derecho a la carpeta creada en cualquier sitio de tu computadora y ahí colocar el comando git clone y pegar lo que acabas de copiar

Posteriormente solo ingresar git status y ahí te saldrá lo que quieres subir le das enter





Después una vez que aparezca debes de ingresar el comando git add . y se agregaran los componentes de la carpeta.

Después solo se ingresa git commit -m"nombre que quieras colocar"

Y por ultima git push para subir los trabajos de la carpeta