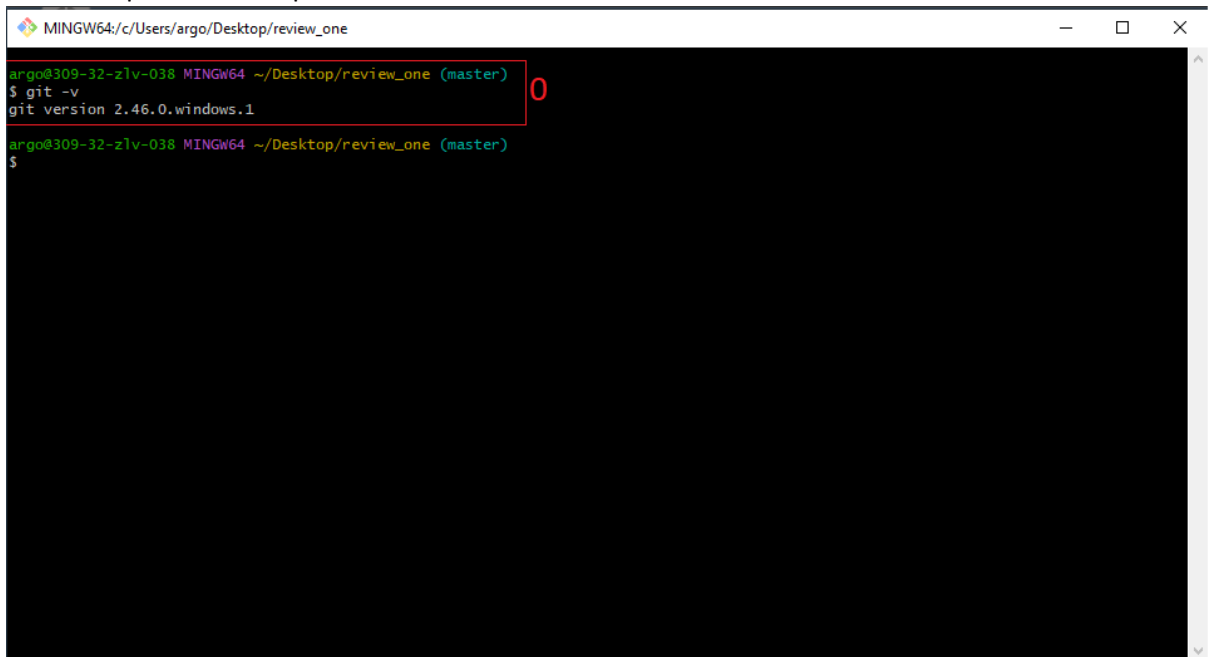


T0. Activitats de repàs I

1. Crea la carpeta de projecte “review_one” en la teva unitat i afegeix-la com a repositori de git.

PASOS

0. Para el correcto desarrollo de la actividad, primero verificamos si tenemos instalado Git en nuestro computador, para esto ejecutamos el comando mostrado en la terminal el cuál nos dará la versión instalada de Git, en caso de no reconocer git como un comando significa que tenemos que realizar el proceso de instalacion.



```
MINGW64/c:/Users/argo/Desktop/review_one
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git -v
git version 2.46.0.windows.1
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$
```

Una vez descargado Git en nuestro computador, veremos que nos descarga algunas dependencias para su correcto funcionamiento y una de ellas es la terminal Git bash, en esta terminal tendremos un entorno muy parecido a la terminal de Linux por lo que, si ya hemos usado linux previamente nos resultará amigable en cuanto a comandos e interfaz

1. Se crea la carpeta “review_one” mediante comandos (mkdir <nombre_directorio>).
2. Se verifica que la carpeta se haya creado exitosamente, esto se realiza mediante el comando “ls -la” el cuál nos generará una lista detallada de los directorios y archivos existentes en nuestra ubicacion
3. Se accede a la carpeta creada (a partir de ahora la llamaremos directorio) mediante el comando “cd” el cual nos permite cambiar de ruta
4. Usando el comando “git init” generamos los archivos necesarios para que el Git bash reconozca el directorio como un repositorio Git, en este caso se crea el directorio “.git”.
5. Si usamos de nuevo el comando “ls -la” podremos ver el directorio creado .git. De igual manera un indicativo de que creamos un repositorio es que despues de la ruta en la que estamos ubicados aparecerá un “(master)”.

```
MINGW64: c:/Users/argo/Desktop/review_one
argo@309-32-z1v-038 MINGW64 ~/Desktop
$ mkdir review_one
argo@309-32-z1v-038 MINGW64 ~/Desktop
$ ls -la
total 17
drwxr-xr-x 1 argo 197121  0 Sep 18 16:26 ./
drwxr-xr-x 1 argo 197121  0 Sep 18 16:20 ../
drwxr-xr-x 1 argo 197121  0 Sep 17 19:45 LoveStone/
-rw-r--r-- 1 argo 197121 282 Feb  8 2022 desktop.ini
drwxr-xr-x 1 argo 197121  0 Sep 18 16:26 review_one/
argo@309-32-z1v-038 MINGW64 ~/Desktop
$ cd review_one/
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one
$ git init
Initialized empty Git repository in C:/Users/argo/Desktop/review_one/.git/
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ ls -la
total 4
drwxr-xr-x 1 argo 197121 0 Sep 18 16:27 ./
drwxr-xr-x 1 argo 197121 0 Sep 18 16:26 ../
drwxr-xr-x 1 argo 197121 0 Sep 18 16:27 .git/
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$
```

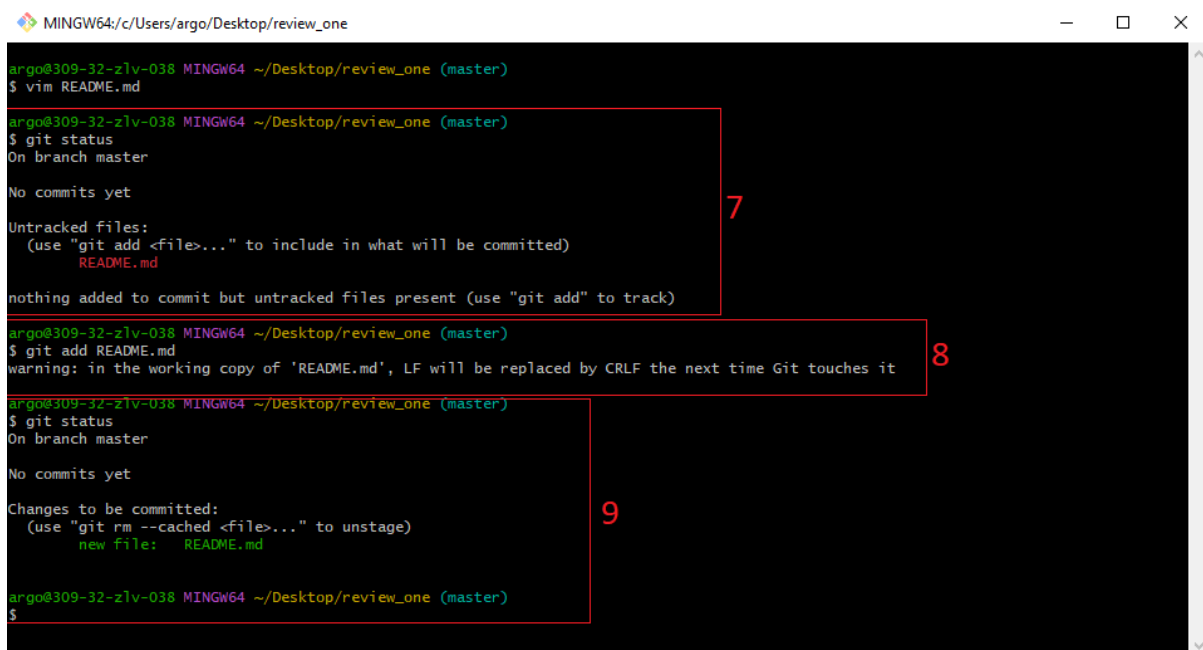


Aquest repositori conté les activitats de repàs I del mòdul de projectes

[illegible]

3. Afegeix-lo al staging area.

7. Para identificar un archivo o directorio que no se haya agregado al staging area, usamos el comando “git status” el cual nos indicará en color rojo lo que nos hace falta para añadir.
8. Una vez identificado, en este caso el archivo README.md, lo añadimos con el comando “git add” seguido del nombre del archivo/directorio
9. Si volvemos a usar el git status, este nos indicará en color verde que se ha subido pero hace falta el commit para confirmar



```
MINGW64: c:/Users/argo/Desktop/review_one
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ vim README.md

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md

nothing added to commit but untracked files present (use "git add" to track)

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git status
On branch master

No commits yet

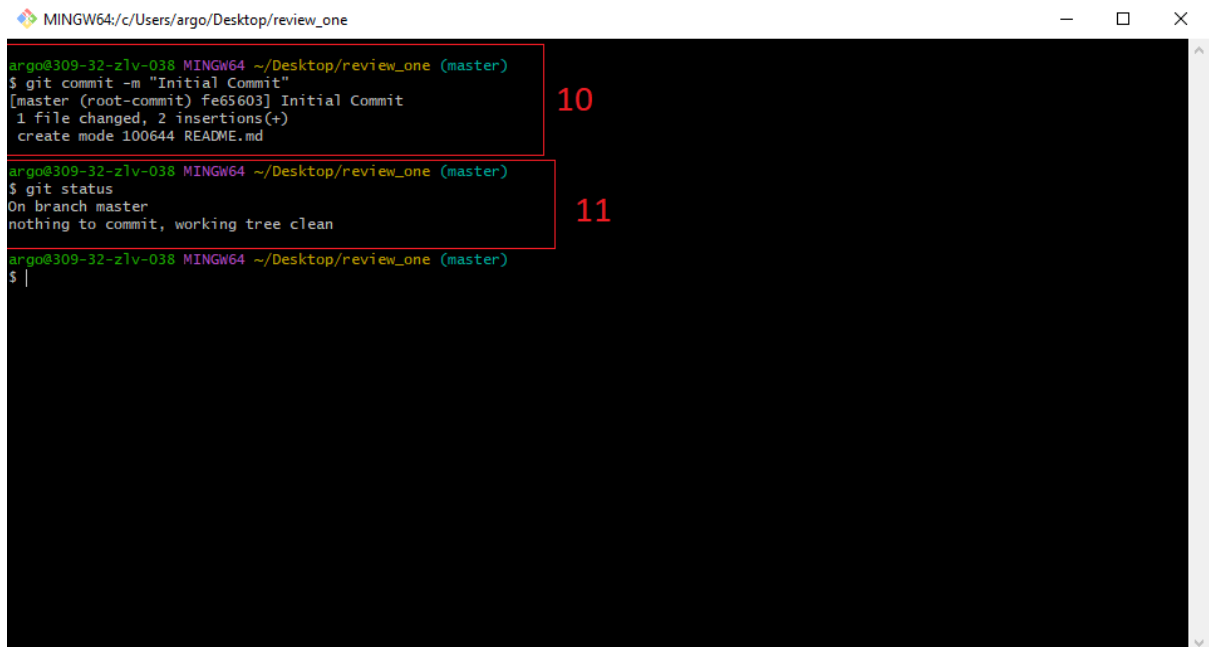
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$
```

4. Afegeix-lo a la commit area amb el comentari "Initial commit".

10. Para realizar los commit y “confirmar” los add’s, usamos el comando “git commit –m <Mensaje>”

11. Si usamos nuevamente git status veremos que no tenemos nada para realizar commit, este es un buen momento para realizar un push a nuestro repositorio remoto

A screenshot of a Windows terminal window titled 'MINGW64: c:/Users/argo/Desktop/review_one'. The terminal shows the execution of two git commands. The first command, 'git commit -m "Initial Commit"', is highlighted with a red box and labeled '10' in red text. Its output shows the commit was successful with hash 'fe65603'. The second command, 'git status', is highlighted with a red box and labeled '11' in red text. Its output shows the working tree is clean. The terminal text is as follows:

```
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git commit -m "Initial Commit"
[master (root-commit) fe65603] Initial Commit
1 file changed, 2 insertions(+)
create mode 100644 README.md

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git status
On branch master
nothing to commit, working tree clean

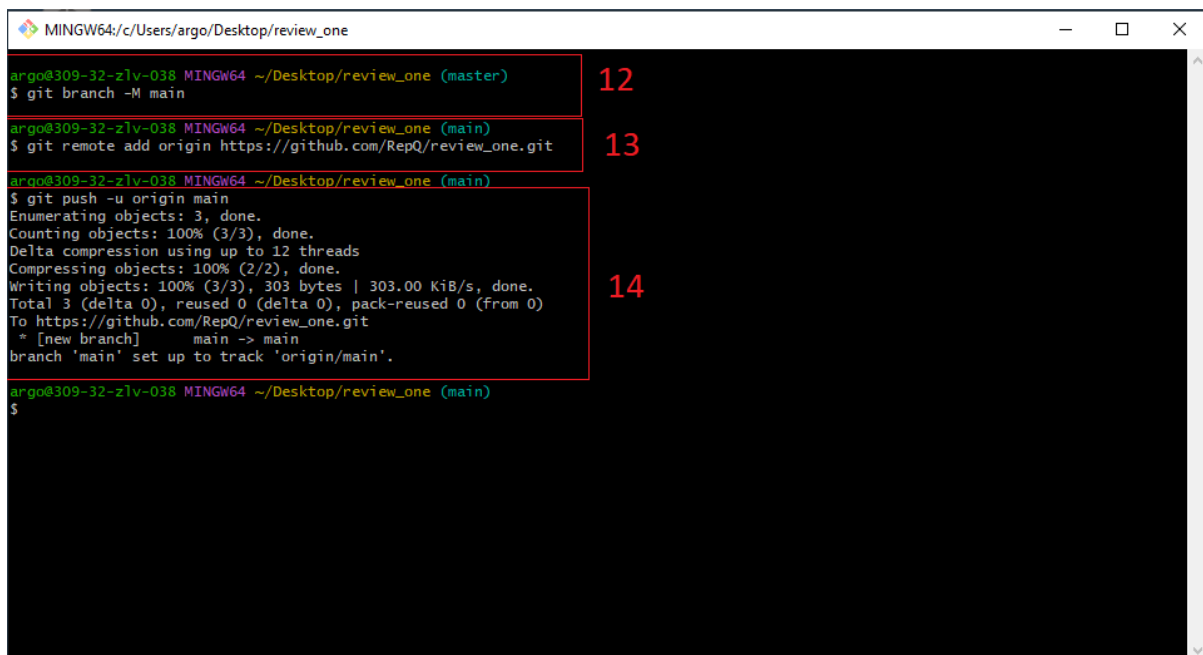
argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ |
```

5. Puja el teu repositori local a un repositori remot (ha de tenir el mateix nom).

12. Usamos “git branch -M main” basicamente para seguir la convencion mas reciente de Git para cambiar de “master” a “main” como rama principal

13. Vinculamos el repositorio local a uno remoto. Acá creamos una referencia llamada en este caso origin que apunta a nuestro repositorio remoto (la url)

14. Estamos subiendo a nuestro repositorio remoto, vinculado con origin, nuestra rama main creada previamente en el paso 12



```
MINGW64/c/Users/argo/Desktop/review_one

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (master)
$ git branch -M main

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (main)
$ git remote add origin https://github.com/RepQ/review_one.git

argo@309-32-z1v-038 MINGW64 ~/Desktop/review_one (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RepQ/review_one.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

6. Aplica les tècniques de refacció i optimització en el codi següent:

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int nota1 = 0;
        int nota2 = 0;
        int nota3 = 0;

        Console.WriteLine("Introdueix la primera qualificació");
        nota1 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Introdueix la segona qualificació: ");
        nota2 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Introdueix la tercera qualificació: ");
        nota3 = Convert.ToInt32(Console.ReadLine());

        int suma = nota1 + nota2 + nota3;

        float promig = suma / 3;

        Console.WriteLine("El promig de les notes es: " + promig );

        if (promig > 6)
        {
            Console.WriteLine("Aprovat");
        }
        else if (promig < 6)
        {
            Console.WriteLine("Suspès");
        }
    }
}
```

Crea el projecte dins del mateix repositori i puja'l al repositori remot.

7. Afegeix una funcionalitat al programa anterior que permeti emmagatzemar ciutats i els seus respectius codis postals. El programa ha de demanar el nom i el codi fins que l'usuari introdueixi un valor per finalitzar la introducció de dades i mostrar un llistat amb format dels valors introduïts.

```
Program.cs
T0_Project_SantiagoVergara
Program
RecolectarDatos<T>(Action<int> M

1  /* ***** */
2  /* ***** */
3  /* ***** */
4  /* Program.cs          ##      #      #      #      #      */
5  /*                      ##      #      #      #      #      */
6  /* By: santi <santi@itb>      ##      #      #      #      #      */
7  /*                      ##      #      #      #      #      */
8  /* Created: 2024/09/19 22:32:40 by santi      ##      #      #      #      */
9  /* Updated: 2024/09/19 22:32:52 by santi      ##      #      #      #      */
10 /* ***** */
11 /* ***** */
12
13 using System;
14 using System.Collections.Generic;
15
16 // Clase encargada de administrar las constantes del programa
17 public class Constantes
18 {
19     public const String APROBADO = "Aprobado";
20     public const String SUSPENDIDO = "Suspendido";
21     public const String PROMEDIO = "El promedio de las notas es: ";
22     public const String CALIFICACION = "Introduzca la calificacion # ";
23     public const String CONTINUE_MENSAJE = "Presione 0 para continuar, cualquier otra para terminar: ";
24     public const String INGRE_CIUADAD = "Ingrese el nombre de la ciudad: ";
25     public const String INGRE_COD_POSTAL = "Ingrese el codigo postal de la Ciudad";
26     public const String SEPARADOR = " - ";
27     public const String FORMATO_CIUADAD = "CIUDAD - CODIGO POSTAL";
28     public const String CONTINUAR = "Ingrese cualquier valor para continuar";
29     public const String UNKNOW = "UNKNOW City";
30     public const String NUMERO_NO_VALIDO = "Número no válido, intente de nuevo: ";
31     public const float MINIMO_PROMEDIO = 6;
32     public const int PRESIONA_CONTINUAR = 0;
33 }
```

```
Program.cs
T0_Project_SantiagoVergara
Program
RecolectarDatos<T>(Action<int> Mensaje, Func<T> ObtenerDatos, Action<T> ProcesarDatos)

33
34 public class Program
35 {
36     // Función genérica encargada de recolectar datos. Esta función evita la repetición de código.
37     public static void RecolectarDatos<T>(Action<int> Mensaje, Func<T> ObtenerDatos, Action<T> ProcesarDatos)
38     {
39         // Variable que lleva el conteo de iteraciones, utilizada para mostrar al usuario cuántos datos ha ingresado
40         int i;
41         // Variable booleana para controlar cuándo detener el bucle
42         bool stop;
43
44         // Inicializamos el índice de iteración a 1 y stop a falso
45         i = 1;
46         stop = false;
47         // Bucle que continuará ejecutándose hasta que el usuario decida detenerse
48         while (!stop)
49         {
50             // Se ejecuta 'Mensaje' para mostrar un mensaje personalizado al usuario. Se le pasa el índice 'i'
51             Mensaje(i);
52             // Se obtiene el dato mediante la función 'ObtenerDatos', el cual retorna un valor del tipo genérico 'T'
53             T dato = ObtenerDatos();
54             // El dato obtenido se procesa utilizando el delegado 'ProcesarDatos', que recibe como parámetro el dato 'T'
55             ProcesarDatos(dato);
56             // Se muestra el mensaje para continuar o terminar
57             Console.WriteLine(Constantes.CONTINUE_MENSAJE);
58             // Se verifica si el usuario desea detenerse. Si el valor ingresado no es igual a 'PRESIONA_CONTINUAR', se rompe el bucle
59             if (NumeroSeguro() != Constantes.PRESIONA_CONTINUAR)
60                 stop = true;
61             // Incrementamos el índice 'i' para la próxima iteración
62             i++;
63             // Limpiamos la consola después de cada iteración
64             Console.Clear();
65         }
66     }
67 }
```



```

Program.cs
T0_Project_SantiagoVergara
Program
RecolectarDatos<T>(Action<int> Mensaje, Func<T> ObtenerDatos,

68 //Funcion encargada de validar la String antes de ser usada
69 1 referencia
70 public static String StringSegura()
71 {
72     //Se valida que la String no sea NULL y se retorna, en caso de serlo se retorna la constante UNKNOWN
73     return (Console.ReadLine() ?? Constantes.UNKNOWN);
74 }
75 //Funcion encargada de validar el numero ingresado por el usuario
76 3 referencias
77 public static int NumeroSeguro()
78 {
79     int Calificacion;
80
81     //Bucle que controla la validacion del input, si no se logra convertir a un int se sigue pidiendo el numero
82     while (!int.TryParse(Console.ReadLine(), out Calificacion))
83         Console.WriteLine(Constantes.NUMERO_NO_VALIDO);
84     return (Calificacion);
85 }
86 //Funcion encargada de mostrar si se aprobó o suspendió dependiendo del promedio
87 1 referencia
88 public static void MostrarResultadoPromedio(float promedio)
89 {
90     //Condicional para verificar si Aprueba o Suspende, dependiendo del valor de la constante MINIMO_PROMEDIO
91     if (promedio >= Constantes.MINIMO_PROMEDIO)
92         Console.WriteLine(Constantes.APROBADO);
93     else
94         Console.WriteLine(Constantes.SUSPENDIDO);
95 }
96 //Funcion encargada de calcular el promedio de las notas
97 1 referencia
98 public static void CalculoPromedio(List<int> notas)
99 {
100     float Promedio;
101
102     //Se usa la funcion Average de las listas para obtener el promedio del total de elementos de la lista
103     Promedio = (float)notas.Average();
104     Console.WriteLine(Constantes.PROMEDIO + Promedio);
105     MostrarResultadoPromedio(Promedio);
106 }

```

```

Program.cs
T0_Project_SantiagoVergara
Program
RecolectarDatos<T>(Action<int> Mensaje, Func<T> ObtenerDatos,

104 //Funcion encargada de obtener las calificaciones
105 1 referencia
106 public static void Calificaciones(List<int> notas, Func<int>ObtenerCalificacion)
107 {
108     RecolectarDatos
109     (
110         //Le indicamos al usuario cuantas calificaciones lleva
111         i => Console.WriteLine(Constantes.CALIFICACION + i),
112         //Se obtiene la calificacion por input del usuario
113         () => ObtenerCalificacion(),
114         //Se añade la calificacion obtenida a la lista notas
115         (nota) => notas.Add(nota)
116     );
117 }
118 //Funcion encargada de Obtener las ciudades y sus respectivos codigos postales
119 1 referencia
120 public static void ListCities(Dictionary<String, int> Cities, Func<String>ObtenerCiudad, Func<int>ObtenerCodigoPostal)
121 {
122     RecolectarDatos
123     (
124         i => Console.WriteLine(Constantes.INGRE_CIUADAD),
125         () => ObtenerCiudad(),
126         (City) =>
127         {
128             Console.WriteLine($"{Constantes.INGRE_COD_POSTAL} {City}");
129             int CityPostal = ObtenerCodigoPostal();
130             Cities.Add(City, CityPostal);
131         }
132     );
133     ShowCities(Cities);
134 }

```

```
Program.cs
T0_Project_SantiagoVergara
Program
RecolectarDatos<T>(Action<int> Mensaje, Func<

133
134
135 //Funcion encargada de mostrar por pantalla las ciudades y codigos postales obtenidos en un formato Ciudad -Codigo Postal
136 //1 referencia
137 public static void ShowCities(Dictionary<String, int> Cities)
138 {
139     Console.WriteLine(Constants.FORMATO_CIUADAD);
140     foreach (var entry in Cities)
141     {
142         Console.WriteLine($"{entry.Key}{Constants.SEPARADOR}{entry.Value}");
143     }
144     Console.WriteLine(Constants.CONTINUAR);
145     Console.ReadLine();
146     Console.Clear();
147 }
148 0 referencias
149 static void Main(string[] args)
150 {
151     //Para guardar las ciudades junto a sus codigos postales usamos una estructura de Diccionarios
152     Dictionary<String, int> Cities;
153     //Para guardar las notas ingresadas por el usuario usamos una lista de enteros
154     List<int> notas;
155
156     //Creamos la lista y guardamos su referencia en notas
157     notas = new List<int>();
158     //Creamos el diccionario y guardamos su referencia en Cities
159     Cities = new Dictionary<string, int>();
160     ListCities(Cities, ()=>StringSegura(), ()=>NumeroSeguro());
161     Calificaciones(notas, ()=>NumeroSeguro());
162     CalculoPromedio(notas);
163 }
```