Guía del Examen – C#

septiembre de 2019
by Felipe Ramírez, PhD.
PRELIMINAR

GitHub

La exposición de todos los programas del curso deberá realizarse a través de la plataforma Git.

1. Genera una cuenta para plataforma Git.
2. Descarga e instala **Github Desktop**, en tu equipo.

   https://github.com/

3. Toma el curso **Git y Github | Curso Práctico de Git y Github Desde Cero**, by Fazt

   https://www.youtube.com/watch?v=HiXLkL42tMU

4. Saber qué es un **Repositorio** y qué es un **Branch**.
5. Usando tu cuenta, genera un repositorio llamado `AprendiendoCSharp` Genera un `Readme` usando MD, donde expliques que es un repositorio de ejercicios para tu propio aprendizaje.
   a. Cuando encargue programas en C#, solicitaré la liga para acceder a su repositorio, clonaré sus archivos, y revisaré a partir de ahí.
6. Conocer los comandos básicos de Git:
   a. `Init`
   b. `Add`
   c. `Status`
   d. `Commit`
   e. `Push`
   f. `Pull`
   g. `Clone`

C#: Naturaleza del Lenguaje

En término generales, esto es lo que puede decirse de C#.

1. **Propósito General.** Permite desarrollar aplicaciones de cualquier tipo de rama del conocimiento y cualquier tipo de aplicación.
2. **Tercera Generación.** Porque está basado en un conjunto de inStringucciones que permiten detallar de forma procedural y secuencial la forma en cómo han de realizarse las tareas.
3. **Orientado a objetos.** Porque permite la creación e instanciación de clases, y permite implementar las tres características de POO: Encapsulmiento, Herencia y Polimorfismo.
4. **Compilado a código intermedio.** Porque el código fuente se compila, generándose un `.exe` o un `.dll`, que recibe el nombre de *assembly*; este código es pre-compilado, y no es realmente un binario ejecutable; pues le falta una segunda fase de compilación, que genera ahora sí el binario ejecutable para la plataforma operativa en donde se intente correr el programa. La plataforma destino deberá tener .NET Framework instalado, quien generará en tiempo de ejecución el binario ejecutable de la plataforma, la primera vez que se corra el programa.
5. **Sensible a mayúsculas y minúsculas (*case sensitive*).** Porque el uso de mayúsculas y minúsculas sí importa, tanto para las palabras reservadas, como para el nombre de los elementos creados por el desarrollador.
6. **No posicional.** Porque la posición en la que se encuentra el código en un programa no influye para nada en la manera en que el compilador genera el *assembly*.
7. **Requiere la especificación detallada de tipos (*Strong Type*).** Las variables requieren declararse antes de utilizarse, y si se declaran, deben utilizarse.
8. **Con terminador de línea.** C# requiere que todas las líneas concluyan con punto y coma (`;`).

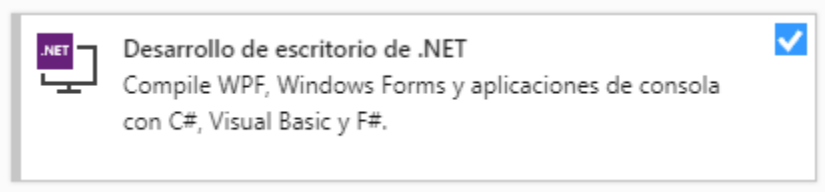
Ambientación

Para desarrollar aplicaciones C# en ambiente Windows, se requiere instalar varios componentes, en la secuencia que se sugiere.

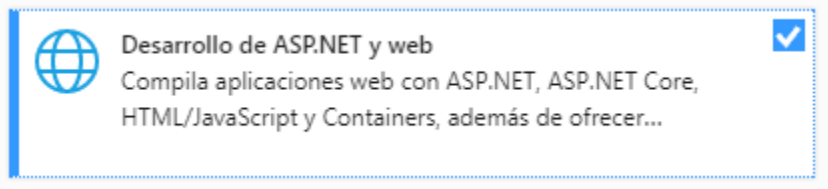
1. Instalar **Visual Studio 2019, Community Edition**. Incluye .NET Framework 4.8, compatible con anteriores hasta la 2.0. Instalar en idioma Inglés.

   https://visualstudio.microsoft.com/es/vs/

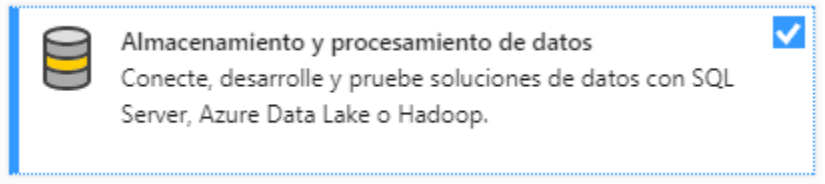
2. Al instalar, asegúrate de integrar los siguientes componentes:
   a. Desarrollo de escritorio. NET. Indispensable para el curso, para poder desarrollar aplicaciones de consola, Windows y WPF. **OBLIGATORIO**.



   b. Desarrollo ASP.NET y web. Si se van a desarrollar aplicaciones Web. **OPCIONAL**.



   c. Almacenamiento y procesamiento de datos. Si se desea integrar bases de datos robustas usando SQL Server y los servicios de Azure. **OPCIONAL**.



3. Instalar estas tres opciones pueden requerir 8.5 GB de espacio en disco, y como 2 horas de tiempo (dependiendo del equipo, puede ser más o menos).

Visual Studio

1. Entender el esquema de trabajo basado en **Soluciones – Proyectos – Programas**.
2. Entender el uso de **Solution Explorer**.
    a. Agregar o crear una solución.
    b. Agregar proyectos.
    c. Agregar elementos a los proyectos.
    d. Crear Folders para los proyectos.
3. Entender qué son los **Templates** de proyecto.
4. Creación de una solución en blanco, y agregarle un proyecto.
    a. Revisa el video **Cómo crear una solución en blanco usando Visual Studio 2019**, que preparé para entender esto:

        https://youtu.be/NFw5Zam1xVk

Lenguaje C#

Todos los temas contenidos en esta guía pueden ser consultados en el curso **Programación en C#: Curso para la certificación oficial {77-483 Exam}**

Consulta: http://www.aprendaenlinea.mx/p/programacioncsharp

5. Estudiar todos los temas del módulo 1 al 4.

MÓDULO 01 .NET Framework y Visual Studio

LECCIÓN .NET Framework y Visual Studio

LAB Crear la primera aplicación de consola

LAB Agregar Code Snippets

MÓDULO 02 Fundamentos de C#

LECCIÓN Fundamentos de C#

LAB Sintáxis básica de C#

LAB Tipos de datos

LAB Conversiones

LAB Pregunta texto

LAB Pregunta número

LAB Salida con formato

LAB Formatos predefinidos

LAB Manejo de arreglas

LAB Uso de Enumeradores

LAB Uso de estructuras de datos

MÓDULO 03 Estructuras de decisión y control

LECCIÓN Estructuras de decisión y control

LAB Manejo de operadores

LAB Concatenaciones

LAB Uso de condicional

LAB Uso de switch

LAB Uso de cíclos

LAB Estructuras anidadas

LAB Ciclo infinito

MÓDULO 04 Métodos y manejo de excepciones

LECCIÓN Métodos y manejo de excepciones

LAB Codificando Clases y Métodos

LAB Manejo de parámetros opcionales

LAB Manejo de parámetros de salida

LAB Parámetros de valor y de referencia

LAB Sobrecarga de métodos

LAB Manejo estructurado de excepciones

LAB Uso de expresiones regulares

Programas

Soluciones

Genera un folder llamado **AprendiendoCSharp**, en la unidad C:\, ahí, genera una solución nueva, que se llame **AprendiendoCSharp.**
En esa solución se agregarán todos los proyectos incluidos en esta guía.
Cada proyecto que se agregue deberá tener su propio Folder.
A menos que se indique otra cosa, todos los proyectos son de tipo **Console App (.NET Framework)**.

HolaMundo: Estructura del programa y salida por consola.

Elabore un proyecto de consola que muestre el mensaje **"Hola Mundo, ¡ahora en C#!"**. Al principio del programa, agregar comentarios que indiquen quién es el autor del programa, y la fecha de elaboración del programa.

Entradas y salidas:

```
Hola Mundo, ahora en C#!
```

HolaMundo | HolaMundo.Program

```
1  // Autor: Felipe Ramírez
2  // Fecha de creación: 13/09/2019
3
4  // Los comentarios se definen usando doble slash
5
6  // using sirve para invocar librerías utilizando nombres cortos.
7  // Las librerías deben estar referenciadas previamente, a nivel
8  // físico. Eso se puede comprobar extendiendo References en
9  // Solution Explorer.
10 using System;
11
12 // En Visual Studio los proyectos de consola se componen de un
13 // namespace, que contiene una clase, que contiene un método,
14 // que constituye lo que se ejecutará al correr el programa,
15 // por lo cual se le conoce como Entry Point Method.
16 namespace HolaMundo
17 {
18     // Note cómo los bloques de código se delimitan con
19     // curly brackets {} y que las líneas terminan en ;
       0 references
20     class Program
21     {
           0 references
22         static void Main(string[] args)
23         {
24             Console.WriteLine("Hola Mundo, ahora en C#!");
25             Console.ReadLine();
26         }
27     }
28 }
```

Conversiones: Convert y análisis de tipos.

Elabore un proyecto de consola llamado **Conversiones**, que declare una variable de tipo **string** con un valor de **"1234"**, y que muestre el data type de la variable: realizar la conversión del dato a **int**, y mostrar el nueva *data type*. Mostrar también el número que se convirtió.

Entradas y salidas:

```
System.String
System.Int32
El número es 1234
```


Conversiones | Conversiones.Program | Main(st

```
1   using System;
2
3   namespace Conversiones
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               // En C#, las variables se declaran enunciando el
10              // tipo de dato, seguido de nombre de la variable
11              // y opcionalmente, de un valor de inicio.
12              string numero = "1234";
13
14              // Se muestra la representción string (ToString())
15              // del tipo de dato (GetType())
16              // de la variable (numero).
17              Console.WriteLine(numero.GetType().ToString());
18              // Se convierte a int, y se muestra el tipo.
19              int intNumero = Convert.ToInt32(numero);
20              Console.WriteLine(intNumero.GetType().ToString());
21              // String.Format permite hacer macro substituciones,
22              Console.WriteLine(String.Format("El número es {0}",
23                  intNumero));
24              Console.ReadLine();
25          }
26      }
27  }
```

Aleatorio: Números aleatorios y conversiones.

Elabore un proyecto de consola llamado **`Aleatorio`**, que declare una variable a nivel clase, de tipo **`float`**, asignándole un valor cualquiera, explícitamente **`float`**; en el *entry point*, declare una variable local de tipo **`float`**, que adquiera un valor aleatorio entre 1 y 10, y que muestre en consola el resultado de la suma de las dos variables, usando el mensaje **"`La suma de x y y es z`"**.

Entradas y salidas:

```
La suma de 10.5 y 5.4 es 15.9
```


Aleatorio — Aleatorio.Program

```
1   using System;
2
3   namespace Aleatorio
4   {
        0 references
5       class Program
6       {
7           // La literal F, indica que se trata de
8           // un valor float. Si está fuera de un
9           // método, se considera variable de clase.
10          static public float numero1 = 24.5F;
            0 references
11          static void Main(string[] args)
12          {
13              // Declaración dentro de un método,
14              // hace que la variable sea local.
15              float numero2 = 0.0F;
16              // Se provee un valor aleatorio.
17              Random numAleatorio = new Random();
18              numero2 = (float)numAleatorio.Next(1, 11);
19              Console.WriteLine(string.Format(
20                  "La suma de {0} y {1} es {2}",
21                  numero1, numero2, numero1 + numero2));
22              Console.ReadLine();
23          }
24      }
25  }
```

Entrada: Revisando si un dato es de un tipo.

Elabore un proyecto llamado **Entrada**, que declare una variable que reciba un **valor**; si el valor puede ser transformado en **integer**, mostrar la leyenda **"Dato entero: x. ¡Muy bien!"** o de lo contrario, mostrar **"Dato no es entero. Intentar nuevamente."**
Ejecutar proporcionando un entero, un flotante, y una cadena.

Entradas y salidas:

a)
```
Escribe algo: hola
Dato no es entero. Intentar de nuevo.
```

b)
```
Escribe algo: 12.5
Dato no es entero. Intentar de nuevo.
```

c)
```
Escribe algo: 10
Dato entero 10. Muy bien!
```


Entrada | Entrada.Program

```
1   using System;
2
3   namespace Entrada
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               // Se declara una variable para preguntar la información
10              // y otra para recibir el valor entero equivalente, si es
11              // que la conversión es posible.
12              string valor;
13              int receptora = 0;
14              Console.Write("Escribe algo: ");
15              valor = Console.ReadLine();
16
17              // Se evalúa si el valor capturado puede convertirse a int
18              if (int.TryParse(valor, out receptora))
19              {
20                  // Si la conversión es posible, el valor convertido se
21                  // almacena en la variable int de trabajo, y se muestra.
22                  Console.WriteLine(
23                      String.Format("Dato entero {0}. Muy bien!",
24                      receptora));
25              }
26              else
27              {
28                  // Si no, se manda un mensaje de que la conversión no fue exitosa.
29                  Console.WriteLine("Dato no es entero. Intentar de nuevo.");
30              }
31
32              // Pausa.
33              Console.WriteLine("");
34              Console.WriteLine("Presiona INTRO para continuar");
35              Console.ReadKey();
36          }
37      }
38  }
```

Nombre: Métodos de tipo y concatenación

Elabore un proyecto llamado **Nombre**, que pregunte dos datos: nombre, y apellido. Los debe transformar a mayúsculas, y mostrar en forma de nombre completo (concatenación). La concatenación debe ser eficiente, por lo que no se puede usar **+**.

Entradas y salidas:

```
Captura un nombre: Felipe
Captura los apellidos: Ramírez
FELIPE RAMÍREZ
```


Nombre | Nombre.Program

```
1   using System;
2   using System.Text;
3
4   namespace Nombre
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main(string[] args)
9           {
10              string nombre;
11              string apellidos;
12              Console.Write("Captura un nombre: ");
13              nombre = Console.ReadLine();
14              Console.Write("Captura los apellidos: ");
15              apellidos = Console.ReadLine();
16              // Se asigna a las variables su versión en MAYÚSCULAS
17              nombre = nombre.ToUpper();
18              apellidos = apellidos.ToUpper();
19              // Los datos string son inmutables. Esto quiere decir que
20              // si los concatenas con +, realmente estás creando copias
21              // de lo que sumas, de tal manera que consumes más memoria.
22              // StringBuilder permite evitar ese fenómento.
23              // Se declara un objeto StringBuilder y se le asigna una nueva
24              // instancia de la clase (new). El constructor le aporta la primera
25              // parte, y posteriormente se añaden las demás.
26              StringBuilder nombreCompleto = new StringBuilder(nombre);
27              nombreCompleto.Append(" ");
28              nombreCompleto.Append(apellidos);
29
30              Console.WriteLine(nombreCompleto);
31
32              // Pausa.
33              Console.WriteLine("");
34              Console.WriteLine("Presiona INTRO para continuar");
35              Console.ReadKey();
36          }
37      }
38  }
```

Tabla: Uso de ciclos finitos (for)

Elabore un proyecto llamado **Tabla**, que pregunte un número entero del 1 al 9, y muestre la tabla de multiplicar del número proporcionado.

Entradas y salidas:

```
Dame un número del 1 al 9: 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
```

... así hasta terminar

Tabla | Tabla.Program

```
1   using System;
2
3   namespace Tabla
4   {
        0 references
5       class Program
6       {
            0 references
7           static void Main(string[] args)
8           {
9               // Cuando en una interfaz los datos se capturan como string
10              // cuando se ocupa que sean numéricos, se define una
11              // variable para el valor capturado, y otro para el valor
12              // en el tipo que se ocupa.
13              string _numero;
14              int numero;
15              // Se pregunta el dato como string, y se convierte a su equivalente
16              // numérico.
17              Console.Write("Dame un número del 1 al 9: ");
18              _numero = Console.ReadLine();
19              numero = Convert.ToInt32(_numero);
20
21              // Se genera un ciclo de número conocido de iteraciones.
22              for (int i = 1; i <= 10; i++)
23              {
24                  Console.WriteLine(
25                      String.Format("{0} x {1} = {2}",
26                       numero,i,numero*i));
27              }
28              // Pausa.
29              Console.WriteLine("");
30              Console.WriteLine("Presiona INTRO para continuar");
31              Console.ReadKey();
32          }
33      }
34  }
```

Tablas: Ciclo anidado

Elabore un proyecto llamado **Tablas**, que elabore las tablas de multiplicar del 1 al 10. Cada tabla deberá tener un encabezado **"Tabla del x"**. Entre una tabla y otra, debe haber un salto de línea.

Entradas y salidas:

```
Tabla del 1

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Tabla del 2

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
```

... así hasta terminar

Tablas | Tablas.Program

```
using System;

namespace Tablas
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            for (int i = 1; i <= 10; i++)
            {
                Console.WriteLine("");
                Console.WriteLine(String.Format("Tabla del {0}: ", i));
                Console.WriteLine("");
                // Un for dentro del otro for, permiten combinar sus
                // variables de secuencia
                for (int j = 1; j <= 10; j++)
                {
                    Console.WriteLine(
                        String.Format("{0} x {1} = {2}",
                        i,j,i*j));
                }
            }
            // Pausa.
            Console.WriteLine("");
            Console.WriteLine("Presiona INTRO para continuar");
            Console.ReadKey();
        }
    }
}
```

Compara: Condicional anidado

Elabore un proyecto llamado **Compara**, que pregunte dos números, y que muestre cuál de los dos es mayor, el primero o el segundo. También debe reportar si son iguales. El mensaje debe decir:
"Números proporcionados: x y y. El mayor es primero." (o el segundo, o son iguales, según sea el caso).

Entradas y salidas:

a)
```
Número 1:10
Número 2:10
Números proporcionados: 10 y 10. Los números son iguales.
```

b)
```
Número 1:10
Número 2:20
Números proporcionados: 10 y 20. El mayor es el segundo.
```

b)
```
Número 1:20
Número 2:10
Números proporcionados: 20 y 10. El mayor es el primero.
```



```
Compara                                                    Compara.Program
using System;

namespace Compara
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // Se preguntan los datos como string, se
            // convierten a int y se almacenan en variables
            // de trabajo. Se declaran 2 variables por línea.
            string _numero1, _numero2;
            int numero1, numero2;
            Console.Write("Número 1: ");
            _numero1 = Console.ReadLine();
            Console.Write("Número 2: ");
            _numero2 = Console.ReadLine();
            numero1 = Convert.ToInt32(_numero1);
            numero2 = Convert.ToInt32(_numero2);
            // un if dentro del otro, ambos con una salida por falso.
            if (numero1 == numero2)
            {
                Console.WriteLine(
                    String.Format(
                        "Números proporcionados {0} y {1}. {2}",
                        numero1, numero2, "Los números son iguales."));
            }
            else
            {
                if (numero1 > numero2)
                {
                    Console.WriteLine(
                        String.Format(
                            "Números proporcionados {0} y {1}. {2}",
                            numero1, numero2, "El mayor es el primero"));
                }
                else
                {
                    Console.WriteLine(
                        String.Format(
                            "Números proporcionados {0} y {1}. {2}",
                            numero1, numero2, "El mayor es el segundo."));
                }
            }
            // Pausa.
            Console.WriteLine("");
            Console.WriteLine("Presiona INTRO para continuar");
            Console.ReadKey();
        }
    }
}
```

Acumulado: Ciclo infinito y operadores incluyentes.

Elabore un proyecto llamado **Acumulado**, que pregunte números enteros indefinidamente. Sólo debe permitir números enteros, y notifica si no es así. Cada número que pregunte, deberá acumularlo, mostrando **"Acumulado hasta el momento: x"**. El programa no deja de preguntar números y acumularlos, hasta que se deje vacía la entrada.

Entradas y salidas:

```
Capture los enteros a acumular.
Dejar vacío y dar INTRO, para salir

Dame un número entero: hola
El dato proporcionado no es un número entero.
Intenta de nuevo.
Dame un número entero: 10
Monto acumulado: 10
Dame un número entero: 20
Monto acumulado: 30
Dame un número entero: 40
Monto acumulado: 70
Dame un número entero:

Presiona INTRO para continuar
```

Acumulado — Acumulado.Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Acumulado
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string _numero;
            int numero;
            int acumulado = 0;
            Console.WriteLine("Capture los enteros a acumular.");
            Console.WriteLine("Dejar vacío y dar INTRO, para salir");
            Console.WriteLine("");
            // Un ciclo while infinito, no concluye sino hasta que
            // se ejecuta de manera explícita un break.
            while (true)
            {

                Console.Write("Dame un número entero: ");
                _numero = Console.ReadLine();
                if (_numero == "")
                {
                    break;
                }
                else
                {
                    if (int.TryParse(_numero, out numero))
                    {
                        acumulado += numero;
                        Console.WriteLine(String.Format("Monto acumulado: {0}", acumulado));
                    }
                    else
                    {
                        Console.WriteLine("El dato proporcionado no es un número entero.");
                        Console.WriteLine("Intenta de nuevo.");
                    }
                }
            }
            // Pausa.
            Console.WriteLine("");
            Console.WriteLine("Presiona INTRO para continuar");
            Console.ReadKey();
        }
    }
}
```

Multiplo: Residuales, operadores lógicos y manejo de errores.

Elabore un proyecto llamado **Multiplo**, que pregunte un número entero. Si el número es múltiplo de 3 y múltiplo de 5, o múltiplo de 7, muestra el mensaje **"Correcto"**, de lo contrario, **"Incorrecto"**. Tip: Si un número es múltiplo de otro, residual es cero.

Entradas y salidas:

a)

```
Dame un número entero: 15
Correcto.
```

b)

```
Dame un número entero: 14
Correcto.
```

b)

```
Dame un número entero: 10
Incorrecto.
```


Multiplo | Multiplo.Program

```
using System;

namespace Multiplo
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            string _numero;
            int numero;
            bool esMultiplo3, esMultiplo5, esMultiplo7;

            try
            {

                Console.Write("Dame un número entero: ");
                _numero = Console.ReadLine();
                numero = Convert.ToInt32(_numero);
                // Si un número tiene un residual de cero con respecto
                // a un número, es que es su múltiplo.
                esMultiplo3 = ((numero % 3) == 0);
                esMultiplo5 = ((numero % 5) == 0);
                esMultiplo7 = ((numero % 7) == 0);
                // Si es múltiplo de 3 y de 5 al mismo tiempo,
                // o si es múltiplo de 7, correcto.
                if ((esMultiplo3 & esMultiplo5) | esMultiplo7)
                {
                    Console.WriteLine("Correcto.");
                }
                else
                {
                    Console.WriteLine("Incorrecto.");
                }
            }
            catch  (Exception e)
            {
                Console.WriteLine("El dato proporcionado causa errores.");
                Console.WriteLine(e.Message);
            }
            finally
            {
                // Pausa.
                Console.WriteLine("");
                Console.WriteLine("Presiona INTRO para continuar");
                Console.ReadKey();
            }
        }
    }
}
```

Fin de la guía.

Los programas contenidos en esta guía deberán estar elaborados y en su repositorio **Git**, donde serán revisados.

En el examen es teórico / práctico, y podrán utilizarse los siguientes recursos:

- Podrá utilizarse la computadora portátil, para desarrollar y ejecutar los programas solicitados.
- Podrán consultarse hasta 3 *cheat sheet* de C#, impresas.
- Podrán consultarse los programas que se hayan cargado en su repositorio Git. Sólo se puede consultar el repositorio personal.

REVISAR CONSTANTEMENTE ESTE ARCHIVO, PORQUE PUEDE CONTENER NUEVAS INDICACIONES. LAS MODIFICACIONES SE DETENDRÁN CUANDO SEA SEÑALADO COMO **"DEFINITIVO"**.