

# SARRERA

Visual Studio 2022 C# .NET

Idazmahairako interfaze garapena

# Sarrera. Idazmahairako interfaze garapena

## Edukiak:

- Izenen eremuak, aldagaiak, motak
- Klaseak
  - Objektu bat sortu eta modelatu:
    - Bere propietateak (aldagaiak, get, set)
    - Bere portaera (metodoen bidez)
    - STATIC erabilera
    - Herentzia eta Polimorfismoa
    - Klase eta metodo birtualak
    - Interfazeak
- Baldintzazko egitura: IF (baldintza){...}ELSE{...}
- Erroreak kontrolatu EXCEPTIONS: TRY ...CATCH ..FINALLY

## Sarrera. Idazmahairako interfaze garapena

- Egitura errepikakorra: FOR (hasiera; amaiera; gehiketa){...}  
FOREACH (var1 IN var2)  
DO {...} WHILE (baldintza)
- Aldagaien ikuspena: PUBLIC, PRIVATE, PROTECTED
- Teklak kontrolatzen: zenbakiak bakarrik onartu, koma bakarra onartu, lehenengo karakterra koma bada, “0,” jarri
- Ebentoak
- Parametroak pasatzen
- ARRAY eta ZERRENDAK

# Sarrera. Idazmahairako interfaze garapena

Instalatzeko (Visual Studio 2022 Community):

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

## Crear un proyecto

Plantillas de proyecto recientes

Aplicación de Windows Forms

C#

Aplicación de Windows Forms

Plantilla de proyecto para crear una aplicación de Windows Forms (WinForms) de .NET.

C#

Windows

Escritorio

Aplicación de Windows Forms (.NET Framework)

Proyecto para crear una aplicación con una interfaz de usuario de Windows Forms (WinForms)

C#

Windows

Escritorio

Biblioteca de controles de Windows Forms (.NET Framework)

Proyecto para crear controles que se van a utilizar en aplicaciones de Windows Forms (WinForms)

C#

Windows

Escritorio

Biblioteca

Biblioteca de clases de Windows Forms

Proyecto para crear una biblioteca de clases para Windows Forms

C#

Windows

Escritorio

Biblioteca

Atrás

Siguiente

## Información adicional

Aplicación de Windows Forms

C#

Windows

Escritorio

Framework ⓘ

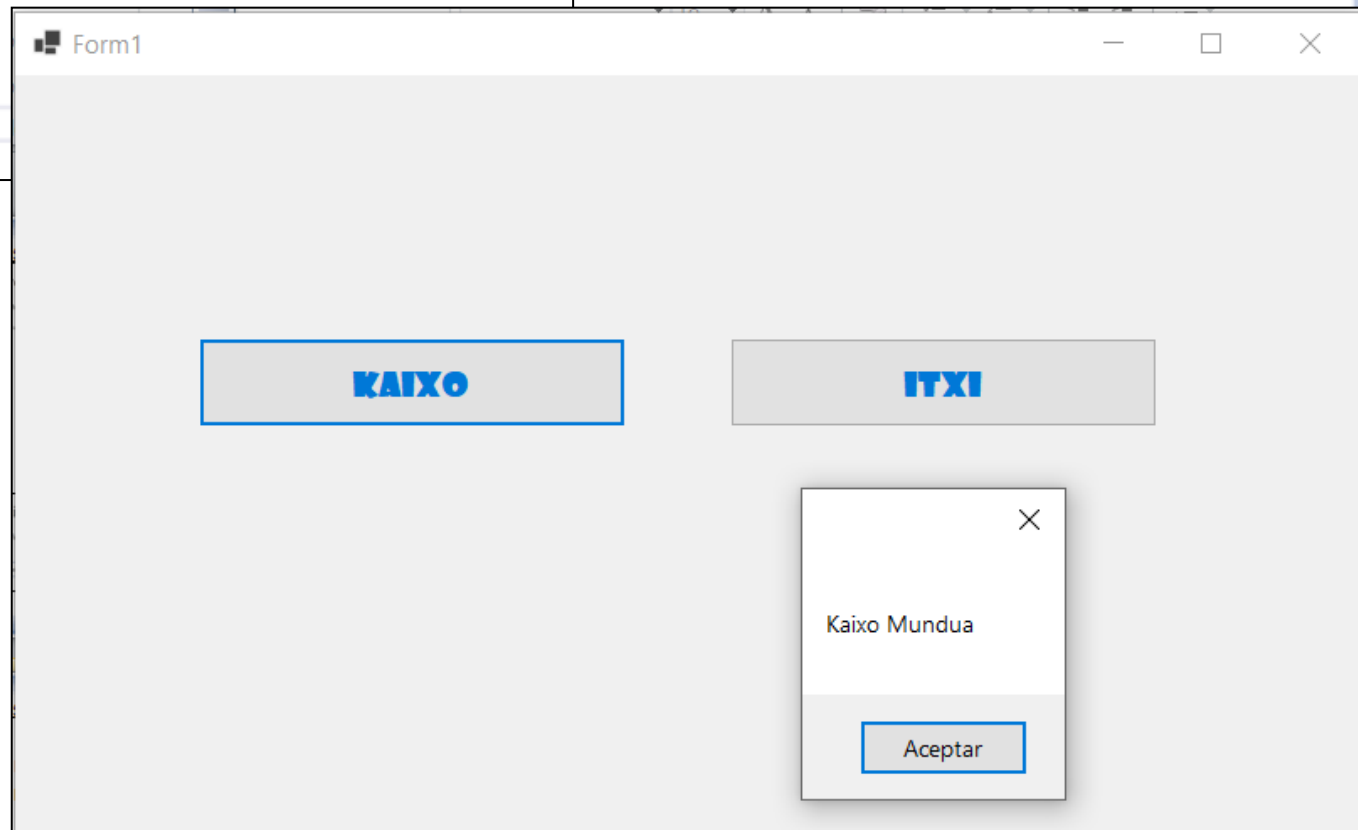
.NET 6.0 (Compatibilidad a largo plazo)

# Sarrera. Idazmahairako interfaze garapena

Adibidea: Kaixo Mundua (Windows Forms motako aplikazioa)

```
1 referencia
private void btnKaixo_Click(object sender, EventArgs e)
{
    MessageBox.Show("Kaixo Mundua");
}

1 referencia
private void btnItxi_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```



# Sarrera. Idazmahairako interfaze garapena

Adibidea: Oinarrizko kalkulagailua

The image shows a screenshot of a basic calculator application window titled "Form1". The window has a light gray background and contains two input fields at the top. The left input field contains the number "3" and the right input field contains the number "2". Below the input fields are four buttons arranged in a 2x2 grid: a plus sign (+), a minus sign (-), a multiplication sign (x), and a division sign (/). The plus sign button is highlighted with a blue border. To the right of the calculator interface, there is a small dialog box with a close button (X) in the top right corner. The dialog box displays the result "5,00" and has an "Aceptar" button at the bottom, which is also highlighted with a blue border.

# Sarrera. Idazmahairako interfaze garapena

Objektuetara zuzendutako programazioa erabiliz: klasea definitzen da

```
namespace WinForms_Kalkulagailua
{
    internal class Kalkulagailua
    {
        //propietateak
        public float Zenbaki1 { get; set; }
        public float Zenbaki2 { get; set; }

        //eraikitzailea
        public Kalkulagailua(float zenbaki1, float zenbaki2)
        {
            this.Zenbaki1 = zenbaki1;
            this.Zenbaki2 = zenbaki2;
        }

        //metodoak
        public float Gehiketa()
        { return Zenbaki1+ Zenbaki2; }
    }
}
```

## Sarrera. Idazmahairako interfaze garapena

Objektuetara zuzendutako programazioa erabiliz: botoiaren ebentoa programatzen da:

```
private void btnGehiketa_Click(object sender, EventArgs e)
{
    //Objektua sortzen

    Kalkulagailua kalkulagailua = new Kalkulagailua(float.Parse(txtZenbaki1.Text),
float.Parse(txtZenbaki2.Text));

    //Objektuaren metodoa erabiltzen

    MessageBox.Show(kalkulagailua.Gehiketa().ToString("0.00"));
}
```



## Sarrera. Idazmahairako interfaze garapena

**Ariketa:** Beste botoiak programatu eta erroreak kontrolatu:

-**Try .....Catch .....Finally** egitura erabili: testu kaxan ez dela ezer idazten edota ez dela zenbakia. Erroreak kontrolatu

## Sarrera. Idazmahairako interfaze garapena

- Zuzenean ez utzi zenbakia ez den ezer idazten

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
```

```
{ if (!Char.IsDigit(e.KeyChar)           // ez bada zenbakia
  && e.KeyChar != Convert.ToChar(Keys.Back) // ez bada atzera
  && e.KeyChar != Convert.ToChar(Keys.Delete) // ez bada ezabatu
  && e.KeyChar != Convert.ToChar(",")    // ez bada koma
  {
    e.Handled = true;                  // ebentoa kontrolatu
    return;                           // bueltatu
  }
  else
  {
    if (e.KeyChar == Convert.ToChar(",")) // koma bada
    {
      if (txtZenbaki1.Text.IndexOf(",") >= 0) // jada badago koma bat
      {
        e.Handled = true;          // ebentoa kontrolatu
        return;                   // bueltatu
      }
      else
      {
        if (txtZenbaki1.Text.Length == 0) // lehenengo koma aurretik 0 bat jartzeko
        {
          e.Handled = true;
          SendKeys.Send("0,");
        }
      }
    }
  }
}
```

## Sarrera. Idazmahairako interfaze garapena

- Eta ez da utzi behar beste inondik kopiatzen ere:  
Horretarako propietate bat erabili daiteke:
  - **ShortcutsEnabled** (Testu kaxa bakoitzean) **false** jarrita.
- Egin ditugun balidazio guztiak erabili behar dira, lehenengoa, testu kaxak hutsik ez direla ixten bermatzeko eta bigarrena, letrarik idazten ez uzteko

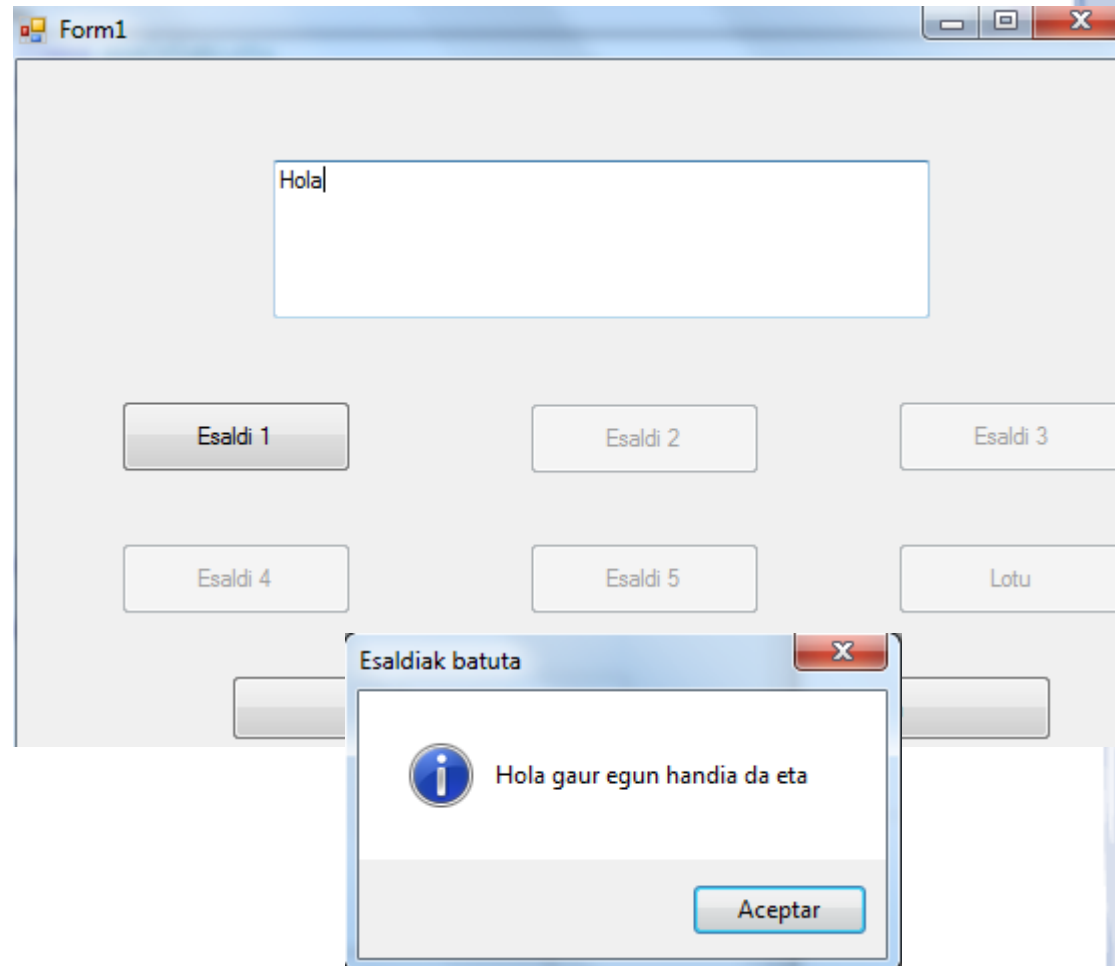
# Sarrera. Idazmahairako interfaze garapena

## Ariketak

**Ariketa 1:** Sortu esaldiak lotzen dituen aplikazioa. Hasieran “Esaldi1” botoia bakarrik aktibatuta dago. Testu kutxan lehenengo esaldia idatzi eta esaldi1 botoia sakatzen dugu, hau desaktibatua geratzen da eta bigarrena aktibatuta. Bostgarren botoia sakatu ondoren “lotu” botoia aktibatzen da. Hau zapaltzerakoan, esaldi guztiak agertuko dira, euren artean tarte zuri batekin. “Garbitu” zapaldu ondoren “Esaldi1” botoia aktibatu eta fokia testu kutxara doa.

**Klase bat egin, Esaldia:** 2 propietate eta metodo bat eduki behar ditu

Propietate bat **EsaldiaBatuta:**  
**irakurtzeko soilik**, orduan **get** bakarrik dauka.



## Sarrera. Idazmahairako interfaze garapena

**Ariketa 2:** Hasieran 1. irudia ikusten da. Zenbaki bat sartu ondoren “Hurrengoa” botoia sakatzen dugu. “2.zenbakia” agertzen da etiketan eta beste zenbaki bat itxaroten geratzen da. Hiru eta laugarren zenbakiekin berdin egiten dugu. Azken zenbakia sartu eta “Hurrengoa” sakatuz gero 2. irudia agertuko da.

Formula:  $(a+2b+3c+4d)/4$

**\*\*egin den eragiketa agertzen da\*\***

“Garbitu” sakatuta hasierako modura itzultzen gara.

-Klase bat sortu zenbakiak gordetzeko eta eragiketa egiteko

-Switch case .... egitura erabili

The image shows three sequential screenshots of a Windows application window titled "Form1".

- Top Screenshot:** The window contains a label "1. zenbakia" in blue text next to a text box containing the number "2". Below the text box are three buttons: "Hurrengoa", "Garbitu", and "Iten".
- Middle Screenshot:** The window contains a label "2. zenbakia" in blue text next to a text box containing the number "5". Below the text box are three buttons: "Hurrengoa", "Garbitu", and "Iten".
- Bottom Screenshot:** The window contains a label "Emaítza" in blue text next to a text box displaying the calculation result:  $(2 + (5 \times 2) + (6 \times 3) + (5 \times 4)) / 4 = 12,5$ . Below the text box are three buttons: "Hurrengoa", "Garbitu", and "Iten".

# Sarrera. Idazmahairako interfaze garapena

## ARRAY

Array-ak balio asko gordetzeko erabiltzen dira. Dimentsio batekoak edo askokoak izan daitezke

```
char[] bokalak = new char[5] {'a', 'e', 'i', 'o', 'u'};
for (int i=0; i<5; i++)          // bokalak.Length erabili ahal da dimentsio bakarrarekin
{
    Console.WriteLine("Elemento {0}: {1}", i, bokalak[i]);
    a= Console.ReadLine();
}

int[,] matriz1 = new int[3, 2] {{1, 2}, {3, 8}, {4, 6}};    //3 zutabe, 2 lerro
for (int i=0; i<matriz1.GetLength(0); i++)                  //GetLength(0) lehenengo dimentsioko luzera
{
    Console.Write("Zutabe {0}: ", i+1);
    for (int j=0; j<matriz1.GetLength(1); j++) //GetLength(1) bigarren dimentsioko luzera
    {
        Console.Write("{0} ", matriz1[i,j]);
    }
    Console.WriteLine();
    Console.WriteLine();
}
a=Console.ReadLine();
```

# Sarrera. Idazmahairako interfaze garapena

**Ariketa 2 - array:** Ariketa aldatu array bat erabiliz:

Formularioan:

```
float[] zenbakiak = new float[4];
```

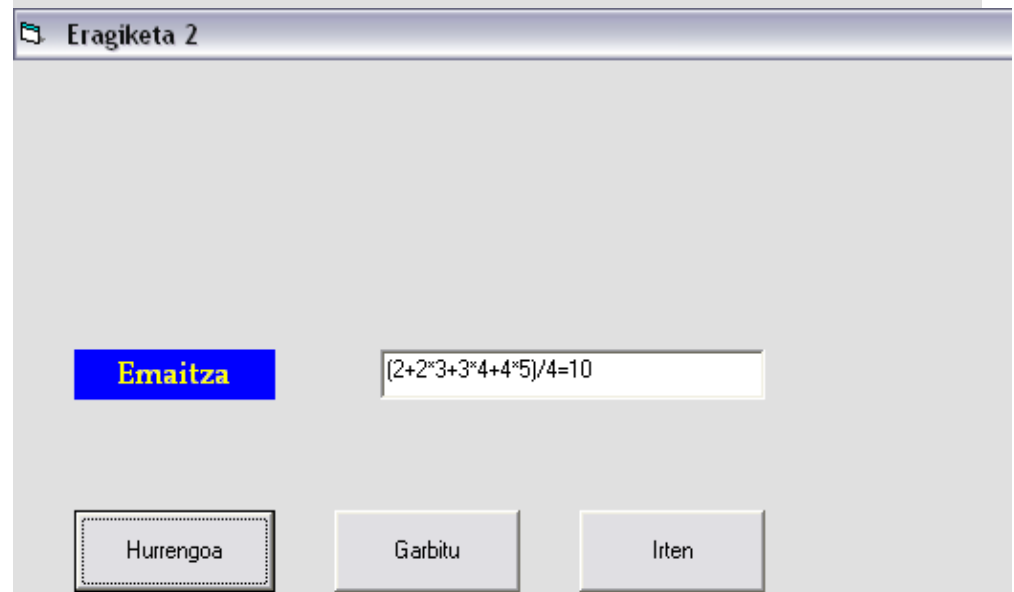
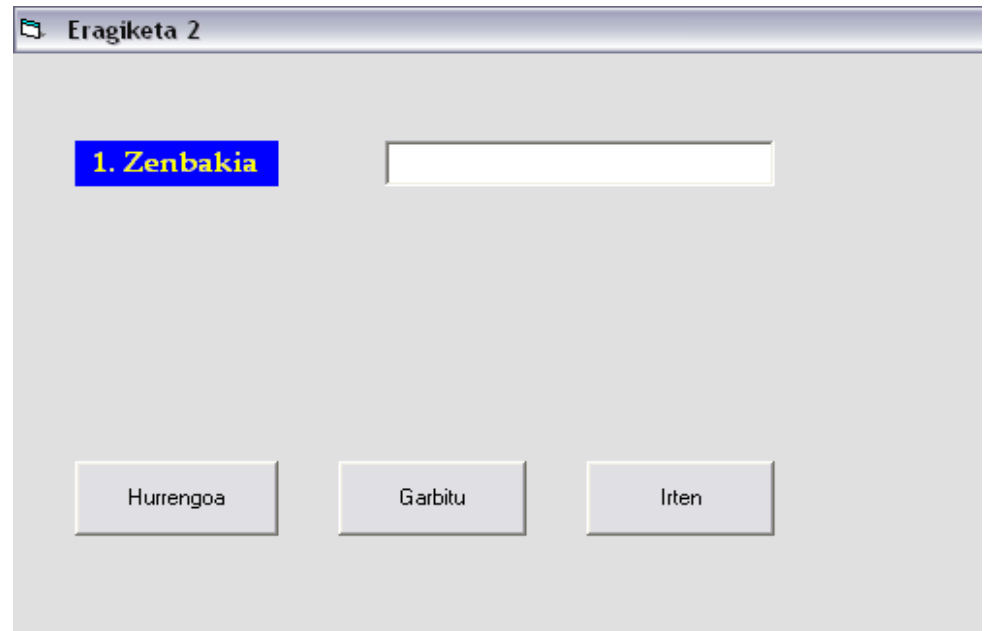
Klasean:

Propietatea array bat izango da:

```
float[] zenbakiak = new float[4];  
public float[] Zenbakiak
```

```
{  
    get  
    {  
        return zenbakiak;  
    }  
  
    set  
    {  
        zenbakiak = value;  
    }  
}
```

+ Metodoa formula aplikatzeko



# Sarrera. Idazmahairako interfaze garapena

## LIST – ZERREDA GENERIKOA

Zerrenda generikoa edozein motako objektuen zerrenda bat gordetzeko erabiltzen da:

Adibidea:

Klasea bat sortzen dugu:

```
Class Datuak
{
    public string Izena;

    public Datuak(string izena)
    {this.Izena = izena;}
}
```

Programa nagusian, bi objektu gordetzen ditugu “zerrenda” batean

```
Static void Main()
{
    List<Datuak> datuak = new List<Datuak>();
    datuak.Add(new Datuak("Aulkia"));
    datuak.Add(new Datuak("Laranja"));
}
```

Consolan ikusten ditugu, “for” egitura erabiliz:

```
for (int i = 0; i < datuak.Count; i++)
{
    Console.WriteLine(datuak[i].Izena);
}
Console.ReadLine();
```

Consolan ikusten ditugu, “foreach” egitura erabiliz:

```
foreach (Datuak datua in datuak)
{
    Console.WriteLine(datua.Izena);
}
Console.ReadLine();
```



# Sarrera. Idazmahairako interfaze garapena

## STATIC

Klase bat estatikoa izan daiteke(ezin da instantziatu). Klase ez estatikoetan propietate estatikoak edo metodo estatikoak topatu ditzakegu:

- Ez dira objetuen osagaiak eta beraz ez dira desberdinak objektu bakoitzentzat.
- Klasearen kideak dira eta beraz kopia bakarra existitzen da. Klasearen izena aurretik jarrita erabiltzen dira.

Adibidea:

```
Class Automobile
{ static int sizeOfGasTank;
public static int SizeOfGasTank
{
get { return sizeOfGasTank; }
}
public static void Drive()
{ }
```

```
// Other non-static fields and properties... }
```

Programa nagusian:

```
.....
Automobile.Drive();

int i = Automobile.SizeOfGasTank;

.....
```

# Sarrera. Idazmahairako interfaze garapena

**Ariketa 2 - zerrenda:** Ariketa aldatu “zerrenda”kin Formularioan Ariketa4 klaseko objektuak sortu eta gorde zerrenda batean eta azkenean emaitza eman.

**List<Ariketa4> zenbakiak = New list<Ariketa4>();**

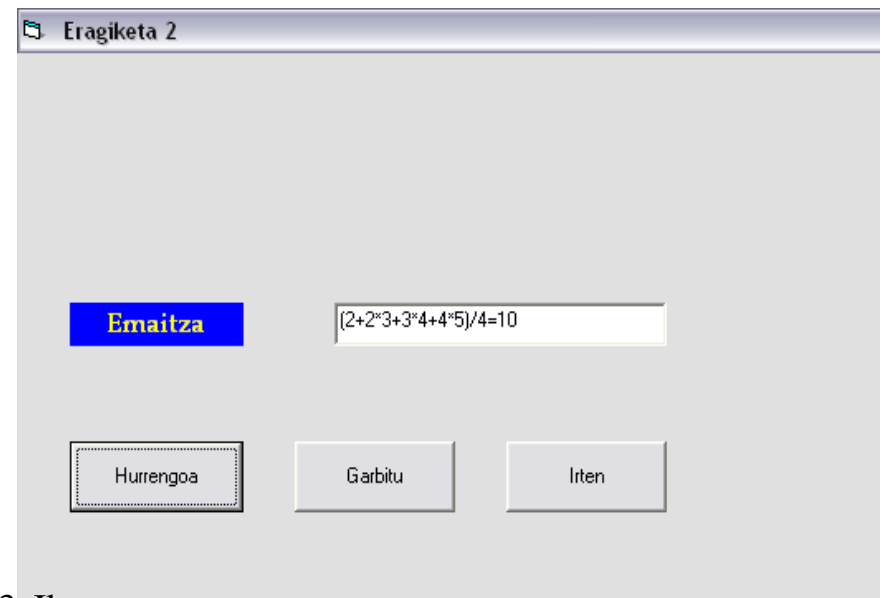
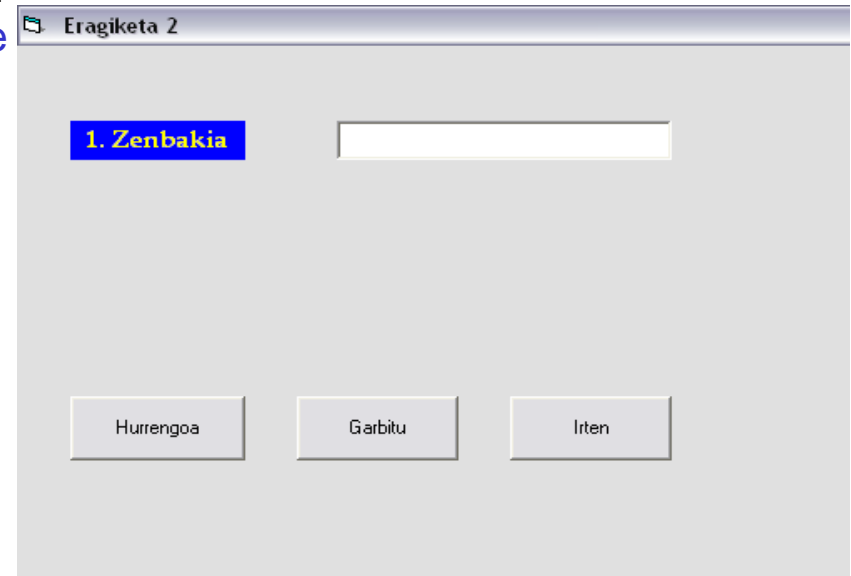
Klaseak propietate bi ditu:

**Class Ariketa4**

```
{ string label;  
  double zenbakia;  
  Public string Label  
  {Get  
    {return label;}  
  Set  
    {label = value;}  
  }  
  Public double Zenbakia  
  {Get  
    {return zenbakia;}  
  Set  
    {zenbakia = value;}  
  }  
}
```

**eta metodo estatikoa**

```
public static double eragiketa(List<ariketa4> objektuak)  
{  
}
```



## Sarrera. Idazmahairako interfaze garapena

### HERENTZIA eta POLIMORFISMOA

**Ariketa 3:** Sortu klase bat “Kontaktua”

propietate hauekin: **Izena**, **Abizena**, **Emaila** (“virtual”), **izenOsoa** (irakurtzeko soilik)

Eta metodo hau: **Gorde** (“virtual”) mezu bat ateratzen duena: “Kontaktua ondo gorde da”

Beste klase bat sortu “**Bezeroa**”, aurreko klasetik heredatzen duena.

Gehi propietate bat: **Kategoria**

Eta metodo **Gorde** berriro definitzen du (override), mezua aldatuta “Bezeroa ondo gorde da”

Beste klase bat sortu “**Langile**”, aurreko klasetik heredatzen duena.

Gehi propietate bi: **Soldata**, **SegurtasunSoziala**. Eta propietate **Emaila** berriro definitzen du (override), dominioa eskolakoa den kontrolatzeko: **value.Substring(value.Length - 14, 14) != “@iesunibhi.com”**. Ez badago ondo, orduan errorea sortu:

**Exception ex = new Exception(“Emaila ez duzu ondo jarri”); throw ex;** (try, catch, exception egitura erabili errorea kontrolatzeko)

Eta metodo **Gorde** berriro definitzen du (override), mezua aldatuta “Langilea ondo gorde da”

## Sarrera. Idazmahairako interfaze garapena

### HERENTZIA eta POLIMORFISMOA

#### Ariketa 3:

The screenshot shows a Windows-style application window titled "Kontaktuak gehitu". The window contains a form for adding a contact. On the left, there are four text input fields labeled "Nan:", "Izena:", "Abizena:", and "Email:". On the right, there is a section titled "Mota" (Type) with three radio button options: "Kontaktua" (selected), "Bezeroa" (Group), and "Langilea" (Employee). Below the "Mota" section, there are two dashed boxes. The left dashed box is titled "Bezeroa" and contains a "Kategoria" (Category) dropdown menu. The right dashed box is titled "Langilea" and contains two text input fields labeled "Soldata:" and "Segurtasun soziala:". At the bottom of the window, there are two buttons: "Gorde" (Save) and "Irten" (Exit).

# Sarrera. Idazmahairako interfaze garapena

## GitHub-era konektatu:



«Crear un nuevo repositorio de GitHub» [aukeratu](#)

Propietario:  
GitHub-en dudan «Organización»  
bat aukeratu, adibidez:  
**UNI-DAM2-IG**

### Crear un repositorio GIT

Enviar cambios a un nuevo repositorio remoto

GitHub

Azure DevOps

Otros

Repositorio remoto existente

Solo local

**Inicializar un repositorio GIT local**

Ruta de acceso local

Plantilla .gitignore

☐ Add a README

**Crear un nuevo repositorio de GitHub**

Cuenta

Propietario

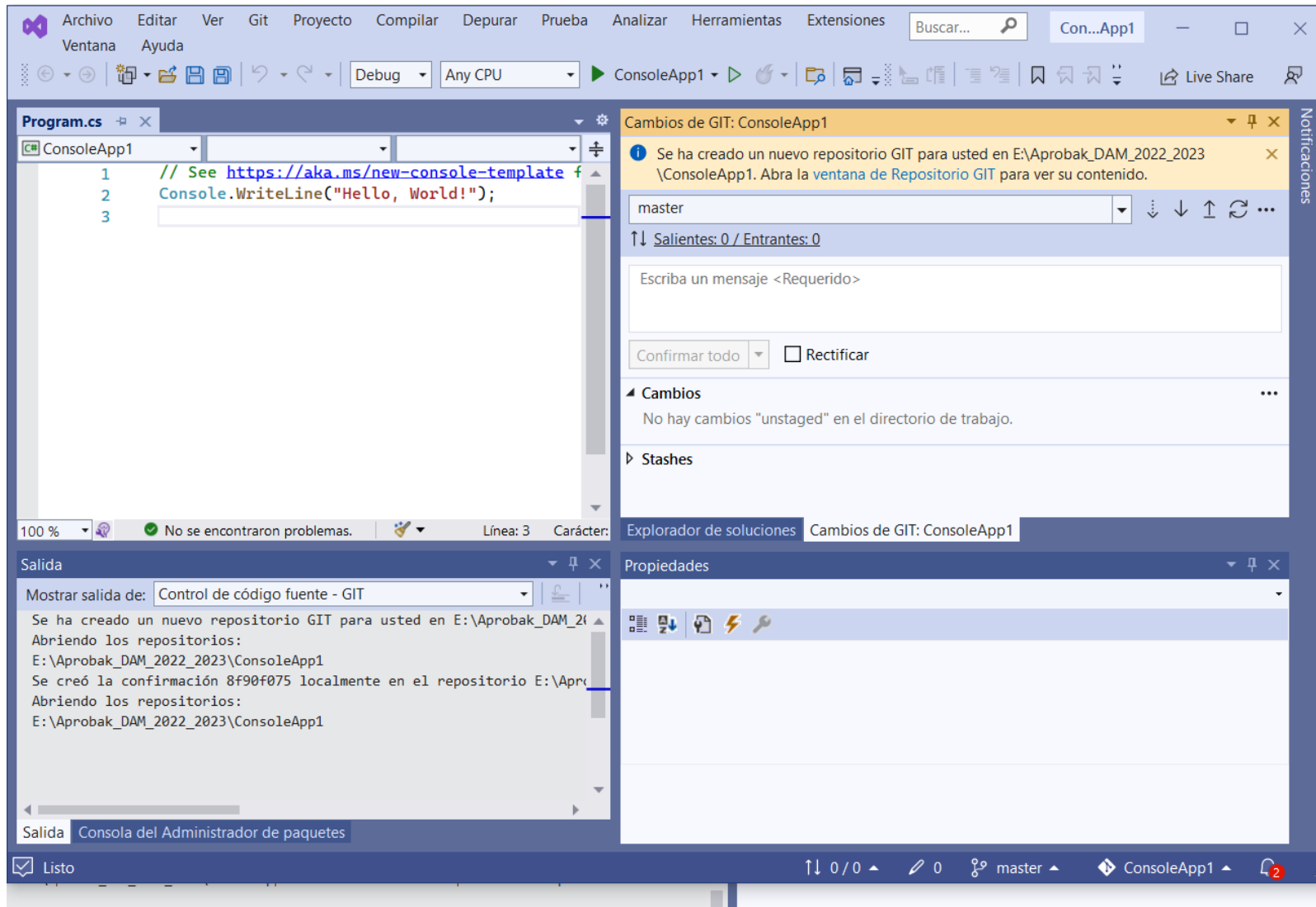
Nombre del repositorio

Descripción

☒ Repositorio privado

# Sarrera. Idazmahairako interfaze garapena

## GitHub-era konektatu:



# Sarrera. Idazmahairako interfaze garapena

**Estekak:**