

**Facultad: Ingeniería**  
**Escuela: Computación**  
**Asignatura: Programación Orientada a objetos**

**GUIA 6:**

## Arreglos de Objetos C#

**Materiales y Equipo**

<b>Nº</b>	<b>Cantidad</b>	<b>Descripción</b>
1	1	Guía de Laboratorio #6 de Programación Orientada a Objetos
2	1	Computadora con programa: ➤ Microsoft Visual C#
3	1	Dispositivo de memoria externo

**Introducción**

Arreglo de objetos.

La función básica de un arreglo es almacenar en una variable más de un valor de un mismo tipo de dato, por ejemplo la siguiente declaración `int[ ] numero= new int [5];` permite almacenar en la variable `numero`, 5 valores enteros.

En las clases el concepto de arreglos es el mismo, con la diferencia que ahora se almacenarán objetos de una clase o de diferentes clases.

Los objetos se pueden estructurar como un array. Los objetos son variables y tienen las mismas capacidades y atributos que cualquier tipo de variables, por tanto es posible disponer objetos en un array.

La sintaxis es exactamente igual a la utilizada para declarar y acceder al array. También disponemos de arrays bidimensionales.

Cuando se crea un array de objetos éstos se inicializan llamando al constructor sin argumentos. Por consiguiente, siempre que se prevea organizar los objetos en un array, la clase debe tener un constructor que pueda llamarse sin parámetros.

**Sintaxis para la definición del arreglo:**

```
nombre_clase [ ] nombrevector = new nombre_clase[tamaño]; /*creación del espacio de memoria para el vector*/  
nombrevector[x]= new clase( ); /*creación de las clases*/
```

Cuando necesitamos invocar algún elemento (propiedad o método) de la clase desde el Programa principal lo hacemos así:

```
nombrevector[x].elementoinvocado; //si es una propiedad  
nombrevector[x].elementoinvocado( ); //si es un método (si tiene parámetros no olvidarlos)
```

Recordemos que cada variación de x representa un nuevo objeto dentro del arreglo, con todos los atributos y métodos que implique.

**Arreglos sin dimensión fija:**

A su vez C# nos permite crear arreglos de dimensión dinámica (es decir crecen o disminuyen según necesidad) a este tipo de elementos se le conoce como List <T> y podemos crear listas de los objetos que necesitamos:

```
List<nombre_clase> nombre_lista = new List<nombre_clase>( );
```

Este tipo de arreglo en sí tiene la facultad de ocupar sus propias propiedades o métodos, por ejemplo:

Add (para agregar a la lista)

Remove (para borrar el último elemento ingresado en la lista)

RemoveAt(para borrar en un índice determinado de la lista)

Capacity (propiedad para ver la capacidad actual de la lista)

Clear (borra todos los elementos de la lista)

Count (propiedad que permite conocer cuántos elementos tiene la lista en ese momento)

Insert (ingresa en una posición específica)

**Procedimiento****EJEMPLO 1:**

Lo primero que debemos hacer es abrir un nuevo proyecto de consola en el cual haremos un programa crea una clase denominada alumno, la cual contiene arreglo para sus notas. Se piden los datos básicos del alumno y en el menú de opciones se ingresan a los estudiantes, se consultan todos los estudiantes inscritos y finalmente se puede ver el registro de todos.

**a. Crearemos la clase alumno**

```
class alumno
{
    string carnet;

    public string Carnet
    {
        get { return carnet; }
        set { carnet = value; }
    }
    string nombre;

    public string Nombre
    {
        get { return nombre; }
        set { nombre = value; }
    }
    string apellido;

    public string Apellido
    {
        get { return apellido; }
        set { apellido = value; }
    }
    string materia;

    public string Materia
```

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

float[] calificaciones = new float[3];

public float[] Calificaciones
{
    get { return calificaciones; }
    set { calificaciones = value; }
}

//Métodos
public void ingresardatos()
{
    Console.WriteLine("\n Ingrese el carnet del estudiante");
    carnet = Console.ReadLine();
    Console.WriteLine("\n Ingrese el nombre del estudiante");
    nombre = Console.ReadLine();
    Console.WriteLine("\n Ingrese el apellido del estudiante");
    apellido = Console.ReadLine();
    Console.WriteLine("\n Ingrese la materia del estudiante");
    materia = Console.ReadLine();

    int i;
    for (i = 0; i < 3; i++)
    {
        Console.WriteLine("\nIngrese la nota {0} de la materia {1} del estudiante {2}:
", i + 1, materia, nombre);
        calificaciones[i] = float.Parse(Console.ReadLine());
    }
}

public void mostrar()
{

```

```
float acumula = 0;
Console.WriteLine("\nEl alumno: " + Nombre + " " + Apellido + " con carnet " +
Carnet);
Console.WriteLine("\n Está cursando la materia " + Materia);
Console.WriteLine("\n Sus notas en esta asignatura son:");
for (int i = 0; i < 3; i++)
{
    Console.Write(calificaciones[i] + " ");
    acumula = calificaciones[i] + acumula;
}
float promedio = acumula / calificaciones.Length;
Console.WriteLine("\n\n Y su promedio es: "+promedio);
Console.WriteLine("\n\n -----");

}

}
```

- b. Ahora lo que haremos será un menú en Program para crear el arreglo, los objetos e invocar métodos

```
static void Main(string[] args)
{
    int tamvec;
    int op;

    Console.WriteLine("Ingrese número de estudiantes en su grupo");
    tamvec = int.Parse(Console.ReadLine());
    Console.Clear();
    alumno[] Estudiante = new alumno[tamvec];
    do
    {
        Console.WriteLine("\t*****MENÚ*****");
        Console.WriteLine("1. Ingresar datos de nuevo estudiante");
        Console.WriteLine("2. Ver listado de estudiantes inscritos");
        Console.WriteLine("3. Reporte de estudiantes");
```

```
Console.WriteLine("4. Salir");

op = int.Parse(Console.ReadLine());
Console.Clear();

switch (op)
{
    case 1:
        Console.WriteLine("SECCIÓN DE INGRESO");
        for (int i = 0; i < Estudiante.Length; i++)
        {
            Estudiante[i] = new alumno();
            Estudiante[i].ingresardatos();
            Console.Clear();
        }
        break;

    case 2:
        Console.WriteLine("\n-----");
        Console.WriteLine("\nLISTADO ALUMNOS");
        Console.WriteLine("\n-----");
        for (int i = 0; i < Estudiante.Length; i++)
        {
            Console.WriteLine("Estudiante número " + (i + 1) + ": ");
            Console.WriteLine(Estudiante[i].Nombre + " " + Estudiante[i].Apellido);
            Console.WriteLine("\n");
        }
        Console.WriteLine("\n");
        Console.ReadKey();
        Console.Clear();
        break;

    case 3:
        Console.WriteLine("\nREPORTE DE ESTUDIANTES");
        for (int i = 0; i < Estudiante.Length; i++)
        {
```

```
        Estudiante[i].mostrar();
    }
    Console.ReadKey();
    Console.Clear();
    break;

case 4:
    break;

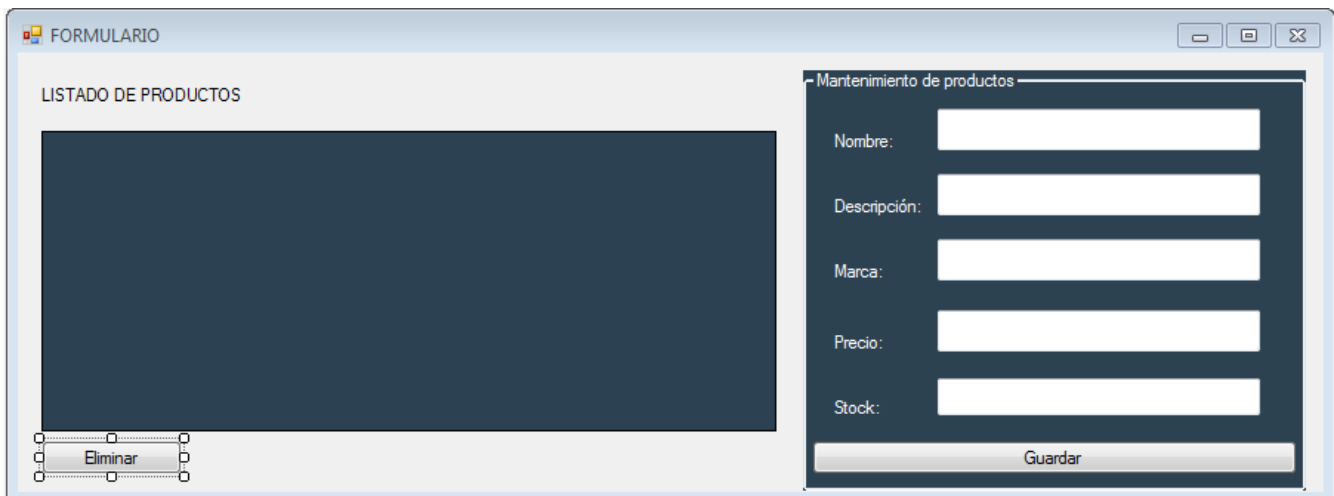
default:
    Console.WriteLine("\n Escriba opción válida");
    Console.ReadKey();
    break;

}
} while (op != 4);
Console.ReadKey();
}
```

## EJEMPLO 2:

Realizaremos un ejercicio para llevar un inventario de productos.

1. Crearemos un nuevo proyecto de Windows Form, el nombre es Inventario. El formulario lucirá como el de la imagen (o similar) usted tiene libertad en diseño (**el elemento de la izquierda es un DATAGRIDVIEW y la propiedad SelectionMode debe estar en FullRowSelect**)



2. Una vez creado el formulario agregaremos una nueva clase que llamaremos Producto. En la clase debe ir el siguiente código:

```
class Producto
{
    string nombre;

    public string Nombre
    {
        get { return nombre; }
        set { nombre = value; }
    }
    string descripcion;

    public string Descripcion
    {
        get { return descripcion; }
        set { descripcion = value; }
    }
    string marca;

    public string Marca
    {
        get { return marca; }
        set { marca = value; }
    }
    float precio;

    public float Precio
    {
        get { return precio; }
        set { precio = value; }
    }
    int stock;

    public int Stock
    {
        get { return stock; }
        set { stock = value; }
    }
}
```

3. Regresando a la clase del formulario incorporaremos dos líneas de código fuera de cualquier método:

```
/*listado que permite tener varios elementos de la clase Persona*/
private List<Producto> Productos = new List<Producto>();
private int edit_indice = -1; //el índice para editar comienza en -1, esto significa que
no hay ninguno seleccionado, esto servirá para el DataGridView.
```

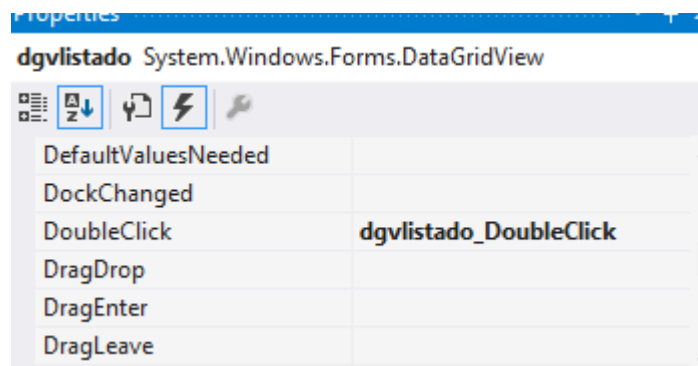


4. Antes de programar los botones, programaremos unos métodos que nos serán de utilidad. Estos dentro de la clase del formulario.

```
private void actualizarGrid()
{
    dgvlistado.DataSource = null;
    dgvlistado.DataSource = Productos; /*los nombres de columna que veremos son
los de las propiedades*/
}

private void reseteo()
{
    txtnombre.Clear();
    txtdescripcion.Clear();
    txtmarca.Clear();
    txtprecio.Clear();
    txtstock.Clear();
}
```

5. A continuación, debemos activar el evento doble click para el DataGridView, así que nos vamos a la opción de eventos y lo seleccionamos. (Cada vez que editemos fila vamos a tener que dar doble click)



Programaremos lo siguiente en el evento:

```
private void dgvlistado_DoubleClick(object sender, EventArgs e)
{
    DataGridViewRow selected = dgvlistado.SelectedRows[0];
    int posicion = dgvlistado.Rows.IndexOf(selected); //almacena en cual fila estoy
    edit_indice = posicion; //copio esa variable en índice editado

    Producto product = Productos[posicion]; /*esta variable de tipo persona, se carga
con los valores que le pasa el listado*/

    //lo que tiene el atributo se lo doy al textbox
    txtnombre.Text = product.Nombre;
```

```

txtdescripcion.Text = product.Descripcion;
txtmarca.Text = product.Marca;
txtprecio.Text = Convert.ToString(product.Precio);
txtstock.Text = Convert.ToString(product.Stock);
}

```

6. Ahora sí programaremos el botón para **guardar**, con el código:

```

private void btnguardar_Click(object sender, EventArgs e)
{
    //creo un objeto de la clase persona y guardo a través de las propiedades
    Producto product = new Producto();
    product.Nombre = txtnombre.Text;
    product.Descripcion = txtdescripcion.Text;
    product.Marca = txtmarca.Text;
    product.Precio = float.Parse(txtprecio.Text);
    product.Stock = int.Parse(txtstock.Text);

    if (edit_indice > -1) //verifica si hay un índice seleccionado
    {
        Productos[edit_indice] = product;
        edit_indice = -1;
    }
    else
    {
        Productos.Add(product); /*al arreglo de Productos le agrego el objeto creado con
        todos los datos que recolecté*/
    }

    actualizarGrid();//llamamos al procedimiento que guarda en datagrid
    reseteo(); //llamamos al método que resetea
}

```

7. Ahora es el turno del botón **Eliminar**

```

private void btneliminar_Click(object sender, EventArgs e)
{
    if (edit_indice > -1) //verifica si hay un índice seleccionado
    {
        Productos.RemoveAt(edit_indice);
        edit_indice = -1; //resetea variable a -1
        limpiar();
        actualizarGrid();
    }
    else
    {
        MessageBox.Show("Dar doble click sobre elemento para seleccionar y
borrar ");
    }
}

```

**EJEMPLO 3:**

Paso de listas entre formularios.

1. Basándose en el ejercicio de la Agenda de Contactos de la guía anterior, capturaremos información en un formulario y la almacenaremos en una lista, esta lista podrá ser enviada íntegramente a otro formulario en el cuál únicamente podrá ser consultada.
2. Lo primero que necesitamos para esta actividad es colocar la clase Persona como una clase pública

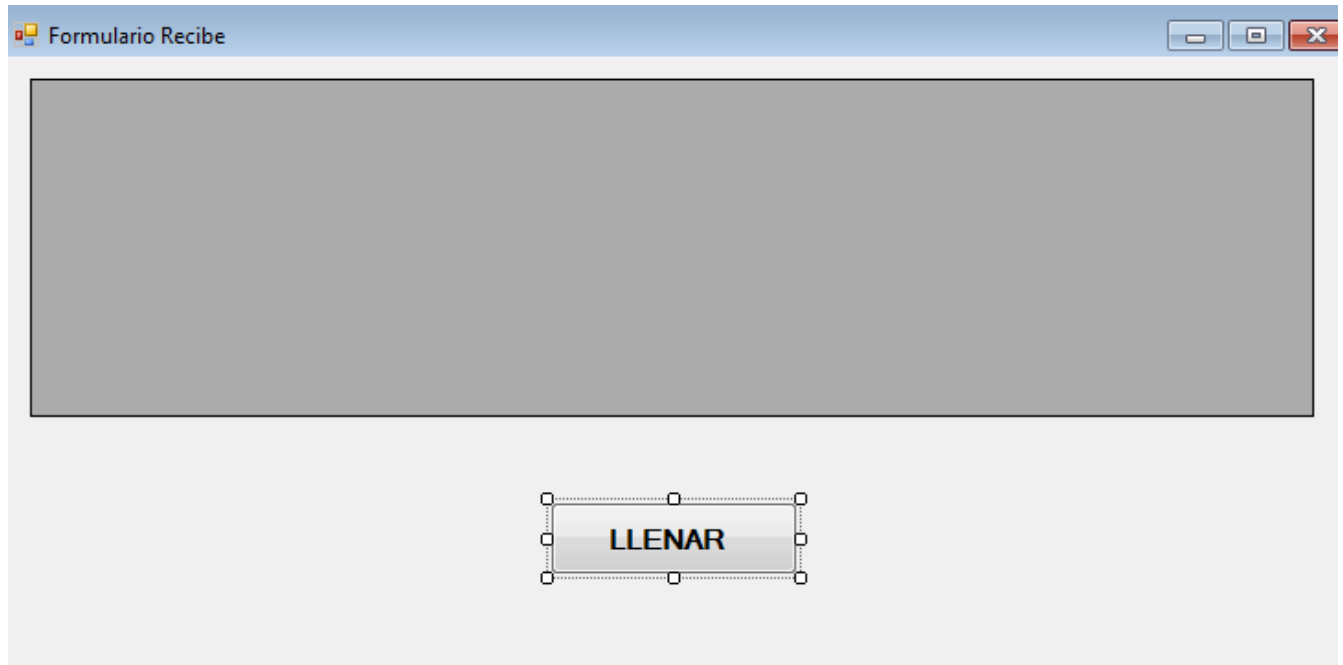
```
1 using System;  
2 using System.Collections.Generic;  
3 using System.Linq;  
4 using System.Text;  
5 using System.Threading.Tasks;  
6  
7 namespace AgendaDeContactos  
8 {  
9     public class Persona  
10    {  
11    }
```

3. En el formulario 1 donde capturamos los datos agregaremos un botón más para enviar la información que recogeremos.

The image shows a web form with four input fields on the left, each with a label: 'Nombre', 'Apellido', 'Teléfono', and 'Correo'. To the right of these fields are three buttons: 'Guardar', 'Eliminar', and 'Enviar Datos'. Below the input fields and buttons is a large, empty rectangular box with a gray background, likely intended for displaying a list of contacts or additional information.

Cree un nuevo formulario que luzca similar (o simplemente que tenga la misma

funcionalidad) al que se muestra en la imagen. Recuerde crearlo en el mismo proyecto.



El formulario 2 contiene:

- 1 DataGridView
- 1 botón

4. En el código del formulario 2 haremos varias acciones: 1. Crearemos una lista que reciba la información que se nos envía del formulario1. 2. Crearemos una función que actualice los datos recibidos y que permita cargarse en este DGV 3. Programamos el botón llenar de forma que se puedan actualizar estos datos recibidos y se llene el DGV.

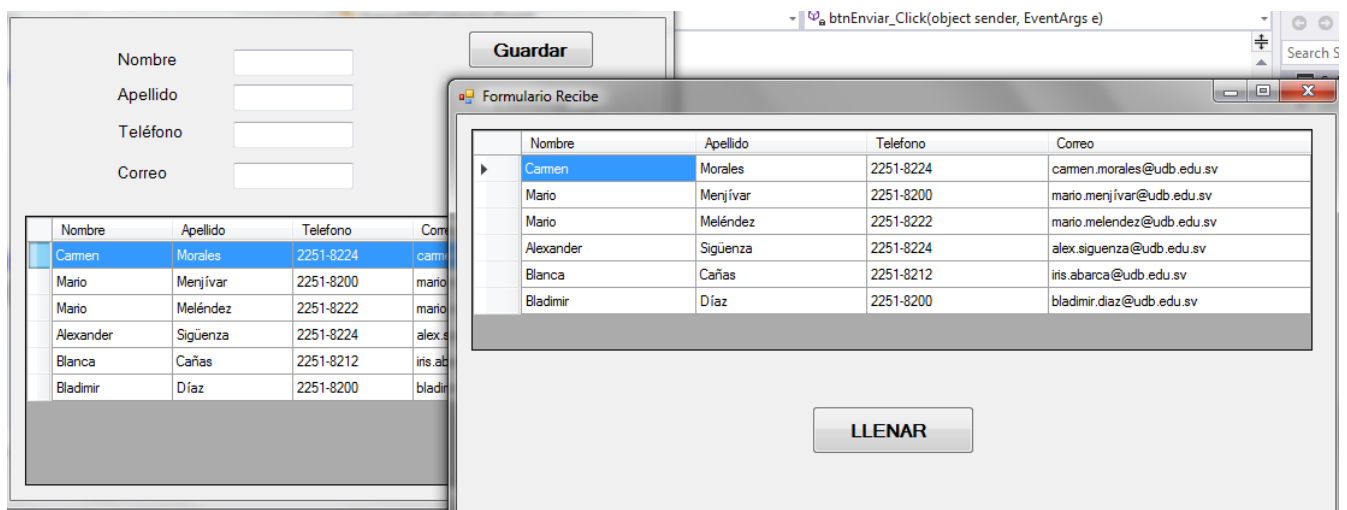
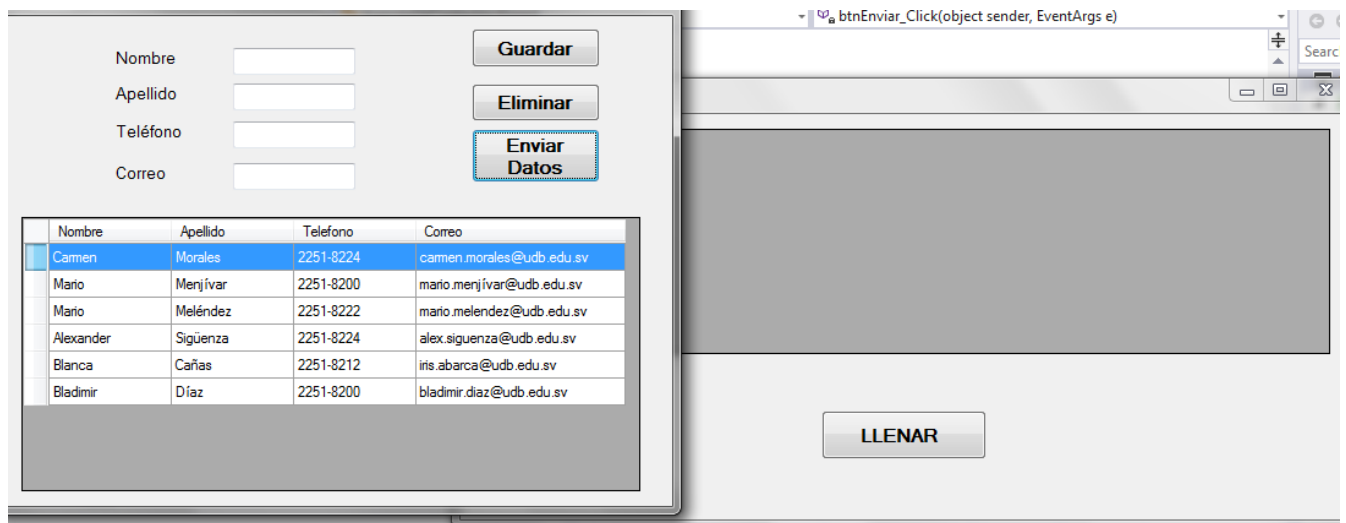
```
public partial class FrmRecibe : Form
{
    public List<Persona> PersonaRecibe = null; //creación de una lista que reciba
    public FrmRecibe()
    {
        InitializeComponent();
    }
    private void actualizarGrid() //función que llena el DGV del formulario 2
    {
        dataGridView1.DataSource = null;
        dataGridView1.DataSource = PersonaRecibe;
    }

    private void btnLlenar_Click(object sender, EventArgs e)
    {
        actualizarGrid(); } //actualiza DGV cada vez que se presione.
}
```

5. Regresemos a la programación del formulario principal y ahí realizaremos la siguiente modificación: programamos el botón que envía información al otro formulario
6. Incluya este código en la programación del botón Enviar Datos

```
private void btnEnviar_Click(object sender, EventArgs e)
{
    FrmRecibe formulario = new FrmRecibe(); //instancia de otro formulario
    formulario.PersonaRecibe = Personas; /*lista original Personas es enviada
    a la lista PersonaRecibe que está en el formulario 2 y que puede ser invocada por medio de
    la instancia del segundo formulario */
    formulario.Show(); //mostar el segundo formulario
}
```

7. Tendrá un resultado como el visto en la imagen.



**Desarrollo de habilidades****G6\_Ejercicio\_01:**

Modifique el ejemplo 1 realizado en Consola, de forma que tengamos posibilidad de manejarlo desde el entorno gráfico, sea lo más creativo posible.

**G6\_Ejercicio\_02:**

Modifique el ejemplo 2 realizado de forma que permita también cargar (mostrar) una imagen del producto cuando seleccionamos su fila.

**G6\_Ejercicio\_03:**

Complete el ejemplo 3 para que pueda verse información filtrada en el DGV2, ya sea porque se envía información seleccionada al segundo formulario o porque usted pueda realizar un filtro en la segunda ventana.