

# INTERFÍCIE GRÀFICA AMB WINDOWS FORMS

---

## DOCENCIA VIRTUAL



**Responsable del Tratamiento:** Universitat Politècnica de València (UPV)

**Finalidad:** Prestación del servicio público de educación superior en base al interés público de la UPV (Art. 6.1.e del RGPD).

**Ejercicio de derechos y segunda capa informativa:** Podrán ejercer los derechos reconocidos en el RGPD y la LOPDGDD de acceso, rectificación, oposición, supresión, etc., escribiendo al correo [dpd@upv.es](mailto:dpd@upv.es).

Para obtener más información sobre el tratamiento de sus datos puede visitar el siguiente enlace: <https://www.upv.es/contenidos/DPD>.

**Propiedad Intelectual:** Uso exclusivo en el entorno del aula virtual.

Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa y/o civil.

## Tema 7 – Seminari – Desenvolupament de Programari en VS

DSIC-UPV

Curs 2024-2025

# Objectius

- Comprendre els principis de les aplicacions visuals.
- Comprendre el disseny de la interfície gràfica d'usuari, ús de controls i esdeveniments.
- Comprendre la comunicació de la capa de presentació amb la capa de negoci tal com està concebuda en el projecte de pràctiques.
- Conèixer les característiques generals de la tecnologia WinForms de Microsoft

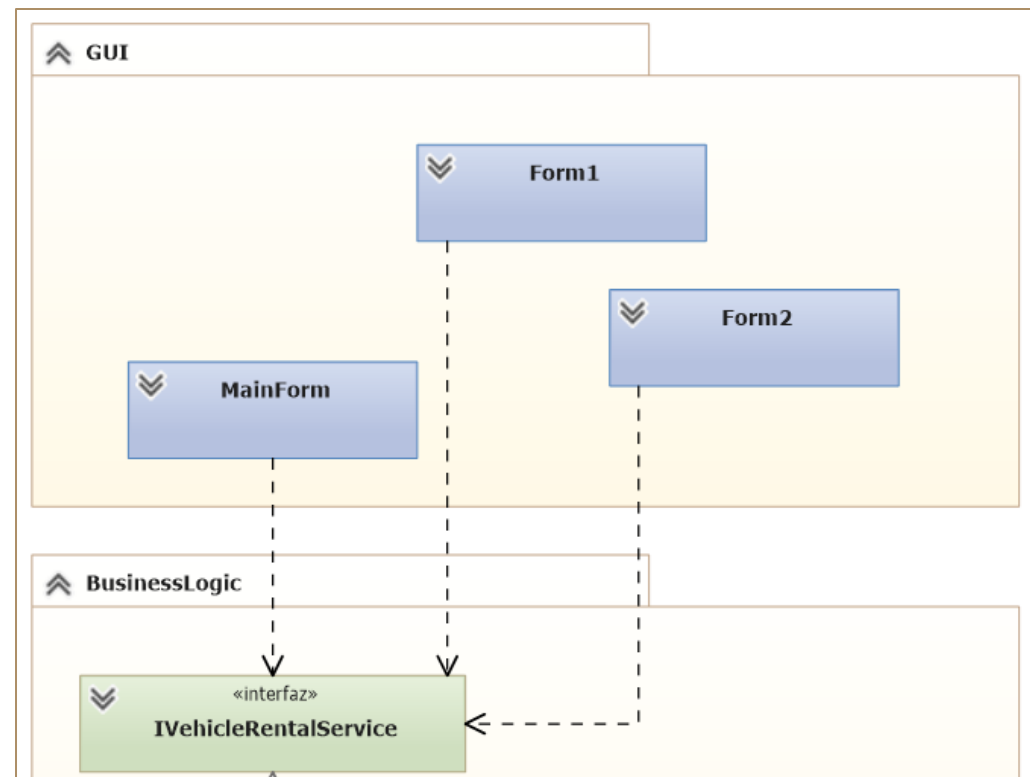
# Continguts

1. Creació d'una Aplicació WinForms Bàsica (1-14 pàg)
2. Controls en els formularis
3. Esdeveniments en els formularis
4. Disseny i ús de menús
5. Aplicacions amb mes d'un formulari
  1. Dissenyat per el programador
  2. Quadres de diàleg
6. Visualització de conjunt de dades
7. Herència Visual

# Disseny arquitectònic. Presentació

- Conjunt de formularis (un d'ells el MainForm)
- Tots els formularis accediran als serveis que ofereix la lògica de negoci a través d'una façana, la interfície IVehicleRentalService

- Per tant, **els formularis necessiten una referència a un objecte de tipus IVehicleRentalService, que es pot passar via constructor.**



# Introducció a WinForms

- Espai de noms `System.Windows.Forms` ofereix components per al desenvolupament d'aplicacions d'escriptori visuals (basades en finestres)
  - `Application`: Es el punt central d'una aplicació Windows. Utilitzant els seus mètodes es processen el missatges de Windows i es creen i destrueixen les aplicacions visuals.
  - `Form`: Representa a una finestra o quadre de diàleg d'una aplicació visual.
  - `Button`, `ListBox`, `TextBox`, `PictureBox`, `Label`,...: els controls habituals de Windows.
  - `StatusBar`, `ToolBar`,...: Barres d'estat, de tasques, etc.
  - `ColorDialog`, `FileDialog`,...: Quadres de diàleg estàndard.
  - `StripMenu`, `StripMenuItem`,...: Menús
  - `ToolTip`, `Timer`,...: Utilitats variades

# Creació d'una Aplicació Windows

- Afegeix un projecte de tipus **Aplicación de Windows Forms.NET Framework** (categoria C#, Windows, Escritori)

The image shows the Visual Studio interface for creating a new project. The process is divided into two main windows: 'Agregar un nuevo proyecto' and 'Configure su nuevo proyecto'.

**Step 1:** In the 'Agregar un nuevo proyecto' window, the 'C#' language is selected. The 'Windows' and 'Escritorio' categories are chosen from the filters. The 'Aplicación de Windows Forms (.NET Framework)' project type is highlighted.

**Step 2:** The 'Configure su nuevo proyecto' window shows the project name 'VehicleRental.GUI' entered in the 'Nombre del proyecto' field. The 'Ubicación' (Location) is set to 'C:\Users\Soledad\Source\VehicleRental\_ISW\_2019'. The 'Framework' is set to '.NET Framework 4.7.2'.

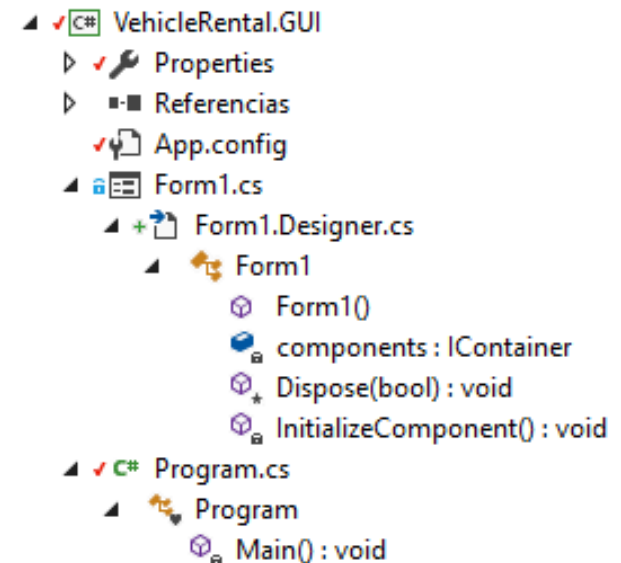
**Step 3:** The 'Siguiente' (Next) button is visible at the bottom right of the 'Configure su nuevo proyecto' window.

**Step 4:** The 'Crear' (Create) button is visible at the bottom right of the 'Agregar un nuevo proyecto' window.

# Creació d'una Aplicació Windows

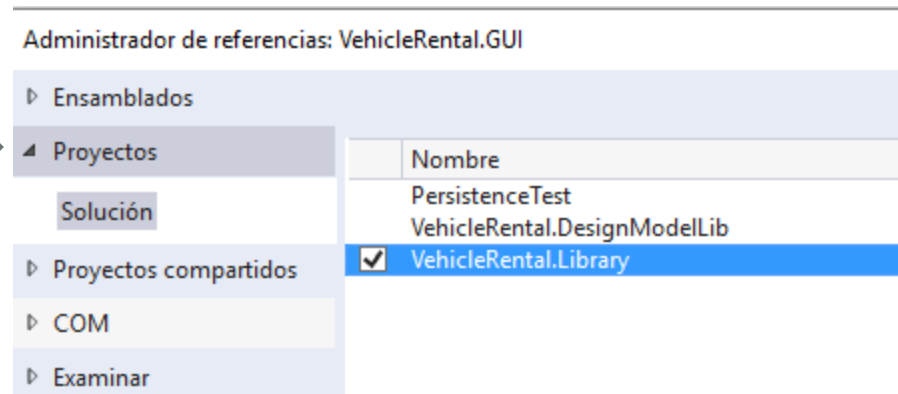
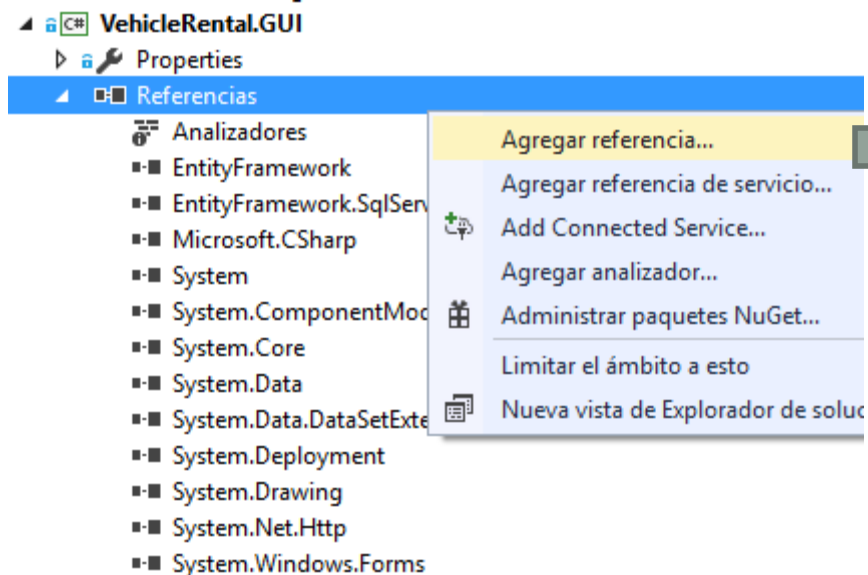
- Els fitxers que formen part d'aquest projecte són:
  - **Form1.cs**: disseny visual del formulari, permet afegir/esborrar/editar controls i canviar la seua apariència
    - Form1.Designer.cs conté la **definició parcial** de la classe Form1, generada automàticament pel dissenyador (conté mètodes Dispose e InitializeComponent). No ha de ser modificada manualment.
  - **Program.cs**: conté la definició del mètode Main().

```
static class Program
{
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```



# Gestió de dependències

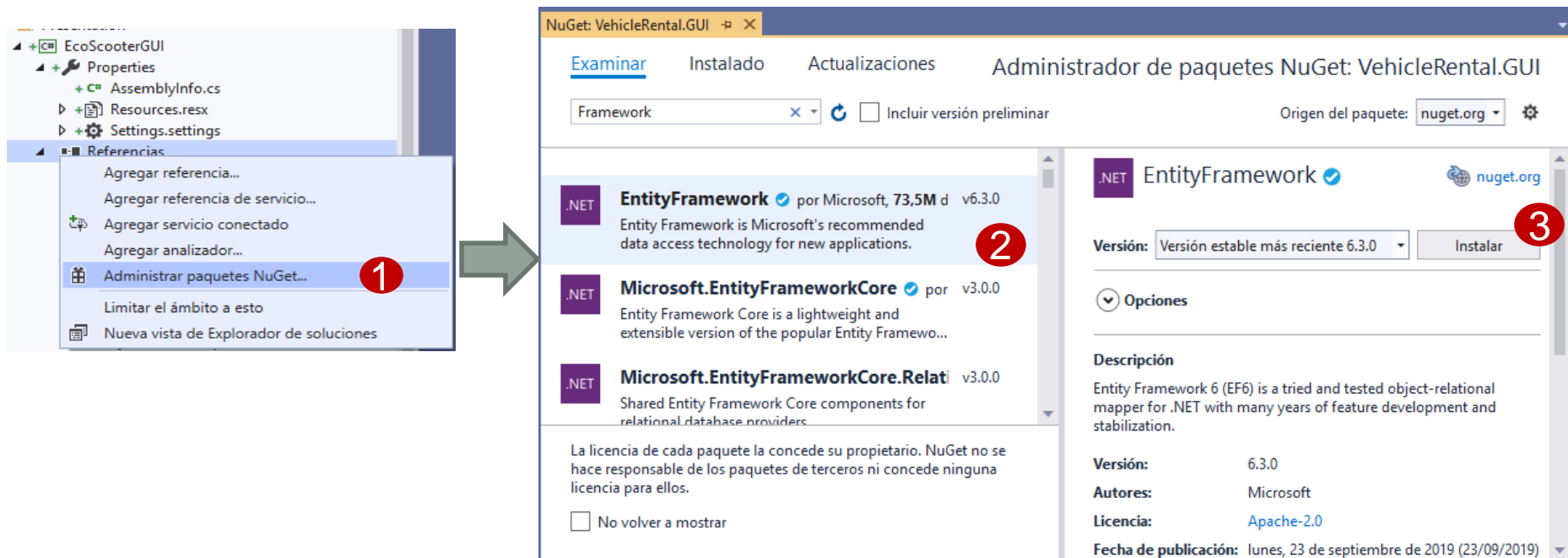
- Aquest projecte dependrà de `IVehicleRentalService` i les classes del domini, situades en el projecte `VehicleRental.Library`, per la qual cosa cal agregar una referència.
- Desplegar projecte i en la secció Referències, clic dret per a obrir menú contextual, prémer Agregar referència.
- En l'Administrador de referències marcar `VehicleRental.Library`





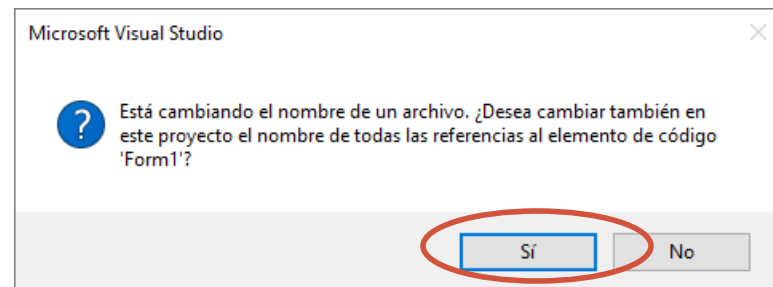
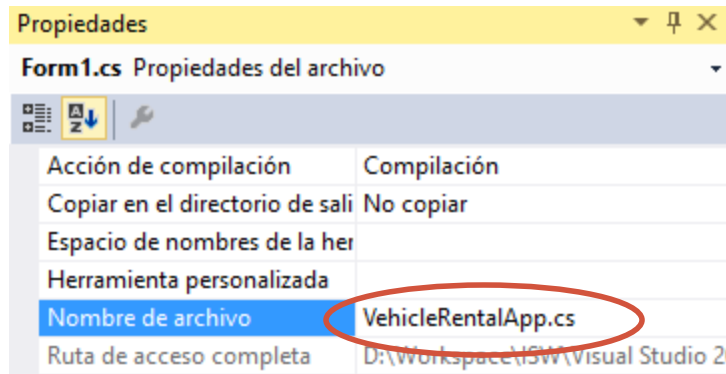
# Gestió de dependències

- A més, farà ús d'Entity Framework, per lo que hi ha que afegix el paquet **NuGet** corresponent:
  - Desplegar projecte i en la secció Referencias, botó dret per a obrir menú contextual i prémer sobre Administrar Paquetes NuGet
  - Buscar Entity Framework i prémer Instalar

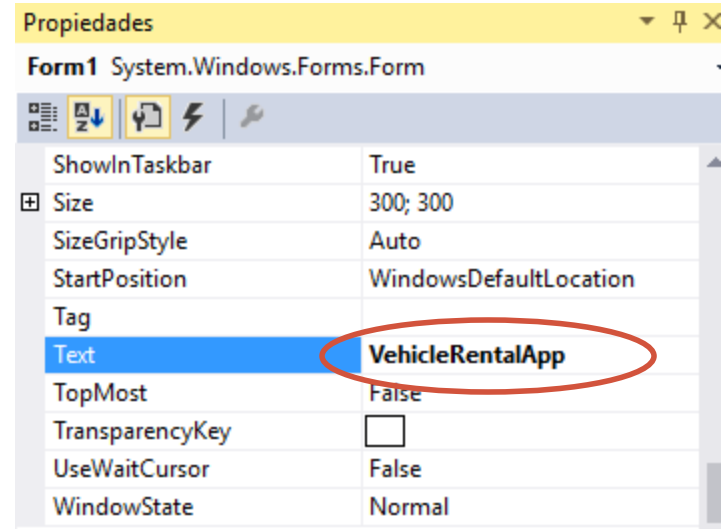


# Reanomenar formulari i donar-li títol


- En propietats de l'arxiu Form1.cs canviar el seu nom (serà el nom d'aqueixa classe formulari)

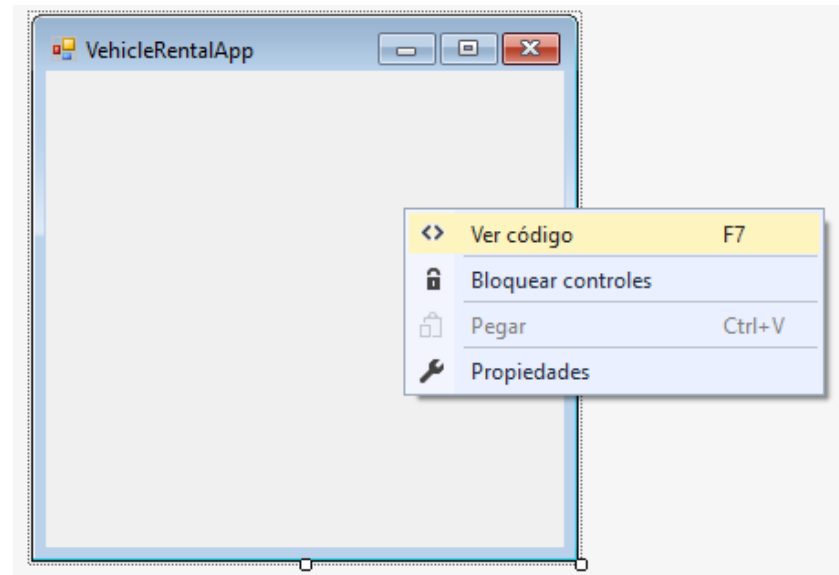


- En propiedades del formulari canviar la propietat Text, que serà el títol de la finestra



# Obrir codi

- Accedir al codi C# editable del formulari, hi ha dues formes:
  - Fer doble clic sobre  **Form1**
  - Seleccionar el formulari en el dissenyador o en l'explorador de solucions i fer *clic dret* > **Veure codi**, o prémer tecla **F7**



# Connectar amb capa de negoci

Modificar la classe **VehicleRentalApp** perquè tinga un atribut de tipus **IVehicleRentalService**, el qual caldrà passar com a paràmetre en el constructor, o mitjançant un mètode.

```
using VehicleRental.Services;

namespace VehicleRentalUI

public partial class VehicleRentalApp
{
    private IVehicleRentalService service; // també podria ser protected

    public VehicleRentalApp(IVehicleRentalService service)
    {
        InitializeComponent();
        this.service = service;
    }
}
```

# Connectar amb capa de negoci...

Modificar el mètode **Main (en la classe Program)** per a crear l'objecte de tipus **IVehicleRentalService** i passar-ho al formulari principal.

```
static void Main()
{
    IVehicleRentalService service = new VehicleRentalService(new
    EntityFrameworkDAL(new VehicleRentalDbContext()));

    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new VehicleRentalApp(service));
}
```

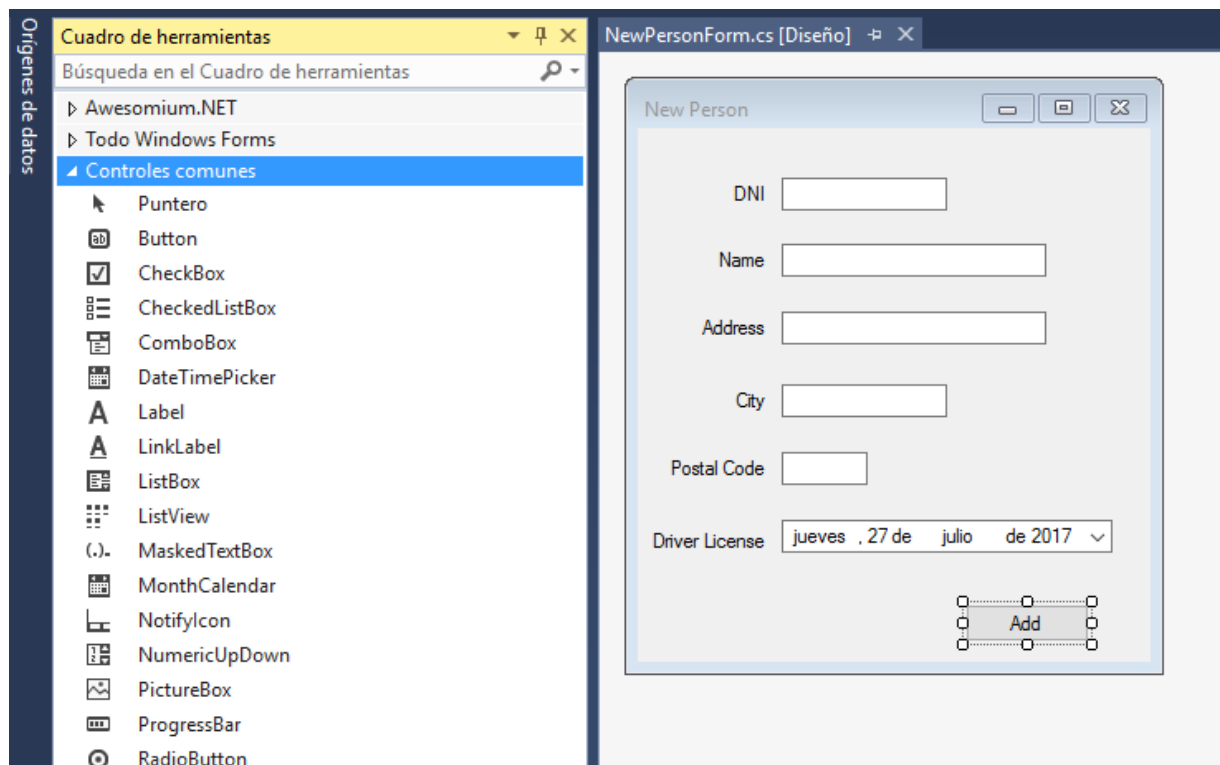
# Connectar amb la capa de persistència...

Modificar **App.config** per afegir la configuració de connexió a la base de dades que crea la capa de persistència:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
  </startup>
  <connectionStrings>
    <clear />
    <add name="VehicleRentalDbConnection"
          connectionString="Server=(localdb)\mssqllocaldb;Database=VehicleRentalDemo;Trusted_Connection=True;MultipleActiveResultSets=true"
          providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

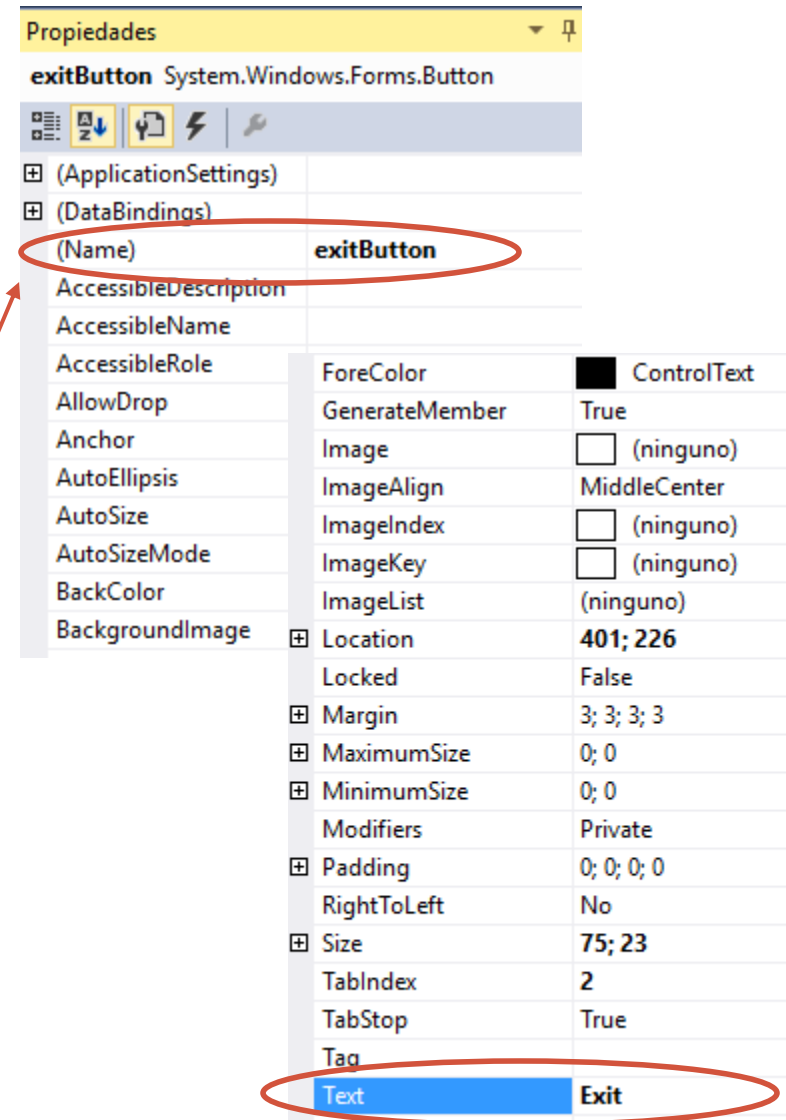
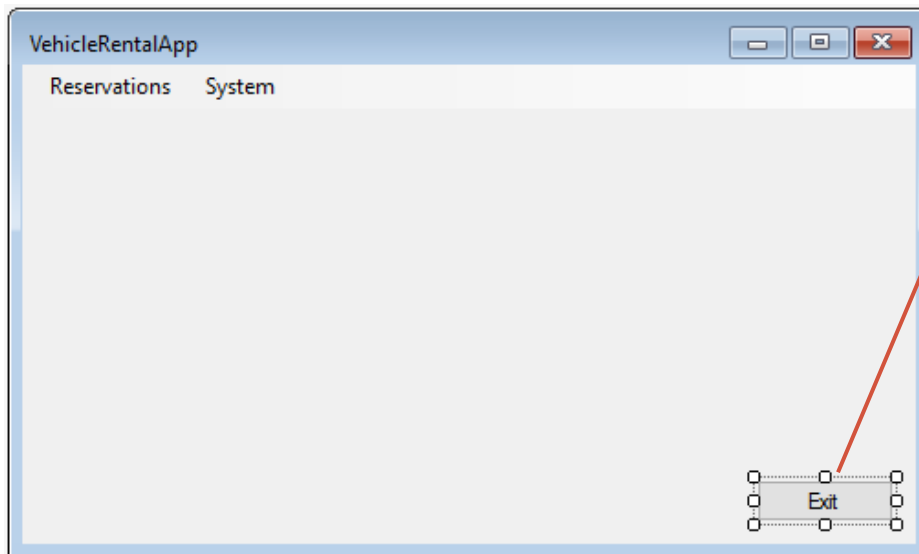
## 2. Controls en els formularis

- Són objectes de la classe Control: botons, quadres de text, botons de selecció, ...
- Es poden afegir en **temps de disseny** (mitjançant l'editor i la paleta de components) o en **temps d'execució**.



# Controls: Propietats

- **Name:** Representa el nom del control. És molt convenient usar noms significatius
- **Text:** Text en el control



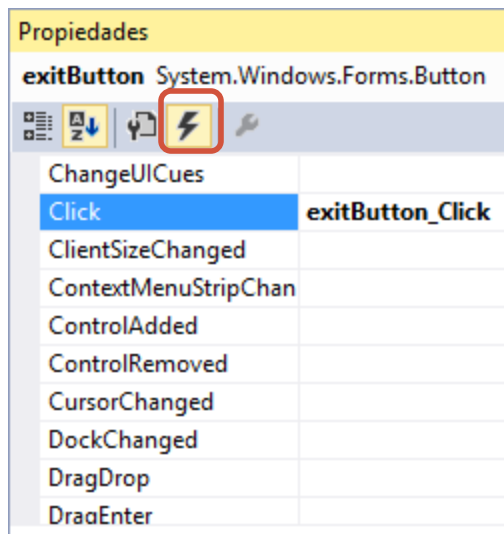


# Esdeveniments en els formularis

- Un **esdeveniment** és una acció a la que es pot respondre des de el codi.
- Els esdeveniments poden ser **generats** per:
  - Una acció de l'usuari (prémer una tecla, un botó del ratolí, etc.)
  - El codi del programa.
  - El sistema operatiu.
- Les aplicacions Windows solen ser aplicacions **controlades** per esdeveniments:
  - La ocurrència d'un esdeveniment provoca l'execució de codi, que es coneix com resposta a un esdeveniment.
  - El codi deu estar en mètodes "especials" denominats "**gestor d'esdeveniments**".
- Tot control proporciona un conjunt d'esdeveniments als que el programador pot associar un mètode.

# Esdeveniments: gestor d'esdeveniments

- Quan ocorris un **esdeveniment** i el seu **gestor** té codi, s'**executa** eixe codi. En cas contrari no ocorris res.
- Els esdeveniments publicats apareixen en la finestra de propietats.
- Podem associar un gestor a un esdeveniment de les següents formes:
  - Escriure el nombre del mètode gestor.
  - Seleccionar un mètode gestor del desplegable.
  - Fer doble *click*, amb lo que l'entorn ens crearà la declaració del mètode gestor per defecte.



Object generador  
de l'esdeveniment

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

Informació de  
l'esdeveniment

# Disseny i ús de menús

- La majoria de les aplicacions Windows tenen menús en el seus formularis.
- Hi ha dos tipus de menús:
  - `MenuStrip`: És un menú principal.
  - `ContextMenuStrip`: És un menú contextual.
- Tots els elements d'un menú s'emmagatzemen en la propietat `Item` que es una col·lecció d'objectes de la classe `ToolStripMenuItem`. Estos elements a la seua vegada poden contindre altres submenús.

# Disseny i ús de menús

**Cuadro de herramientas**

Búsqueda en el Cuadro de herramientas

- Awesomium.NET
- Todo Windows Forms
- Controles comunes
- Contenedores
- Menús y barras de herramientas**
  - Puntero
  - ContextMenuStrip
  - MenuStrip
  - StatusStrip
  - ToolStrip
  - ToolStripContainer
- Datos
- Componentes
- Impresión
- Cuadros de diálogo
- Interoperabilidad WPF
- General

No hay controles utilizables en este grupo. Arrastre un elemento a este texto y agréguelo al cuadro de herramientas.

**VehicleRentalApp.cs [Diseño]\***

**VehicleRentalApp**

Reservations System Escriba aquí

New List Escriba aquí

Assistent per a la creació del menú

Escriba el texto para ToolStripMenuItem

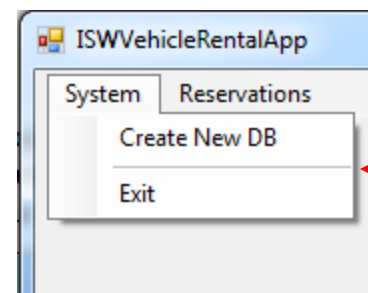
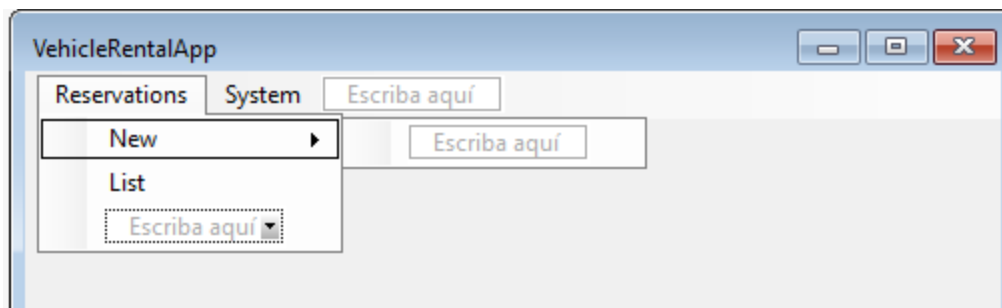
**menuStrip1**

Component no visual

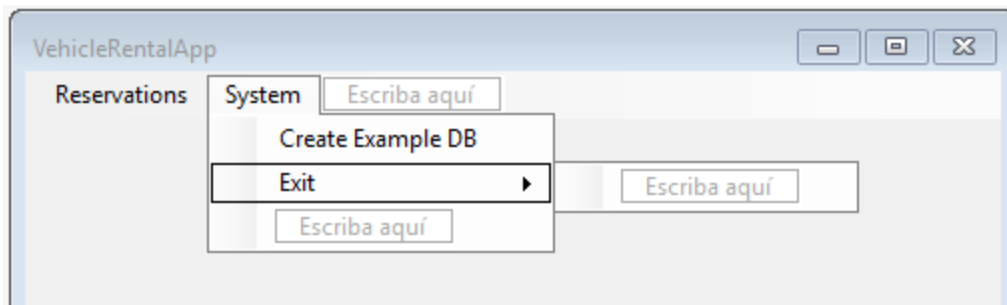
**menuStrip1** System.Windows.Forms.MenuStrip

Size	488; 24
Stretch	True
TabIndex	1
TabStop	False
Tag	
Text	menuStrip1

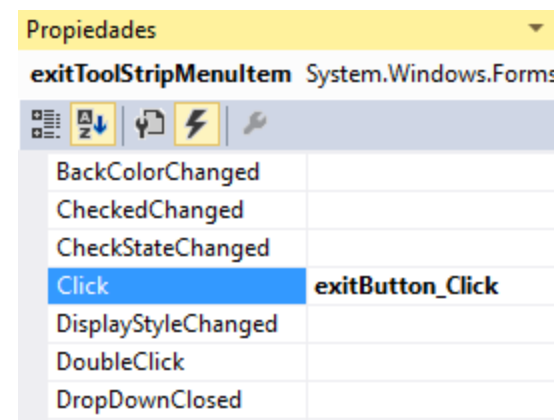
# Exemple menú



← Text: -



Nota: Podem reutilitzar els gestors d'esdeveniments (mètodes) amb diferents controls. Exemple: Item de menú Exit i botó Exit



# Aplicacions amb més d'un formulari

- Normalment en les aplicacions Windows es necessari utilitzar més d'un formulari.
- L'aspecte predefinit d'un formulari comú està determinat per la propietat `FormBorderStyle`.
- Podem trobar formularis:
  - Dissenyats pel programador: s'afegeixen amb *Proyecto/Agregar Windows Forms*.
  - Pre-definits per l'entorn: quadres de diàleg.

# Modalitat

- Un formulari es pot visualitzar de formes
  - **Modal:** Ha de ser tancat per a continuar treballant amb l'aplicació. Es mostra mitjançant el mètode `ShowDialog()`.

```
ExampleForm myForm = new ExampleForm();  
myForm.ShowDialog();
```

- **No Modals:** Permeten treballar amb diversos formularis de l'aplicació sense necessitat de tancar-los. Es mostren mitjançant el mètode `Show()`.

```
ExampleForm myForm = new ExampleForm();  
myForm.Show();
```

# Exemple de formulari principal

```
public partial class VehicleRentalApp : Form
{
    private IVehicleRentalService service;
    private NewReservationForm newReservationForm;    // podría ser var. local
    private ListReservationsForm listReservationForm; // podría ser var. local

    public VehicleRentalApp(IVehicleRentalService service)
    {
        InitializeComponent();
        listReservationForm = new ListReservationsForm(service);
        newReservationForm = new NewReservationForm(service);

    }
    private void newToolStripMenuItem_Click(object sender, EventArgs e)
    {
        newReservationForm.ShowDialog();
    }
    ...
}
```

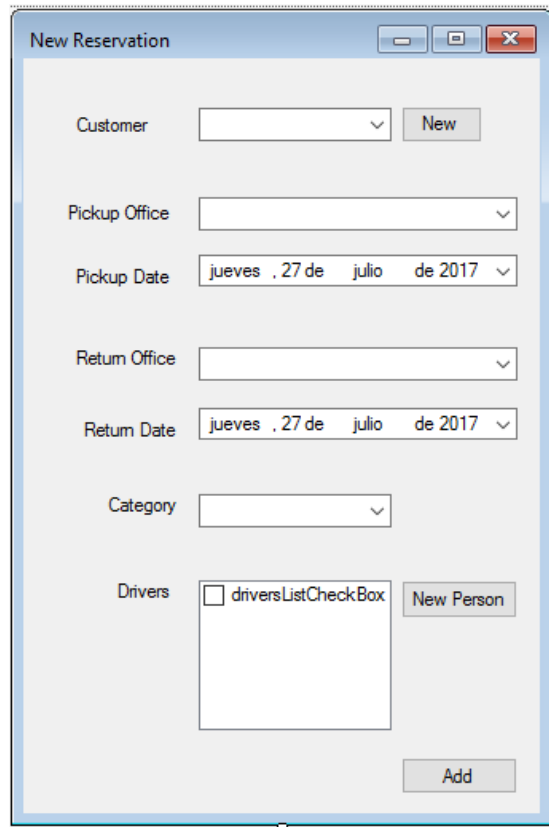
Pas de paràmetres en el constructor.  
Li passem el servici

Mostrarà el nou formulari de forma "Modal"



# Exemple de formulari *no principal*

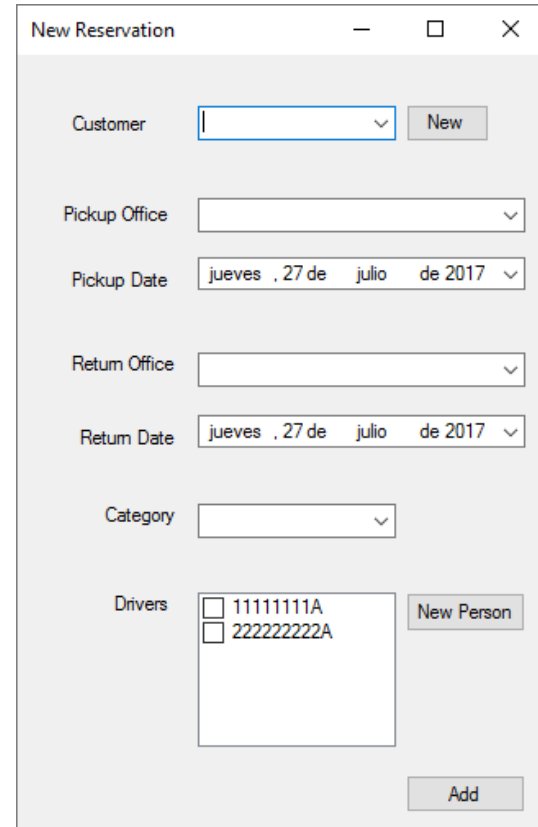
## Vista de disseny



The design view of the 'New Reservation' form shows a light blue title bar with standard window controls. The form layout includes:

- Customer:** A text box with a dropdown arrow and a 'New' button.
- Pickup Office:** A text box with a dropdown arrow.
- Pickup Date:** A date picker showing 'jueves , 27 de julio de 2017'.
- Return Office:** A text box with a dropdown arrow.
- Return Date:** A date picker showing 'jueves , 27 de julio de 2017'.
- Category:** A text box with a dropdown arrow.
- Drivers:** A section with a checkbox labeled 'driversListCheckBox' and a 'New Person' button.
- Add:** A button at the bottom right.

## Vista en execució



The execution view of the 'New Reservation' form shows the same layout as the design view, but with a light gray background and active controls:

- Customer:** The text box is active with a blue border, and the 'New' button is visible.
- Pickup Office:** A text box with a dropdown arrow.
- Pickup Date:** A date picker showing 'jueves , 27 de julio de 2017'.
- Return Office:** A text box with a dropdown arrow.
- Return Date:** A date picker showing 'jueves , 27 de julio de 2017'.
- Category:** A text box with a dropdown arrow.
- Drivers:** A list box containing two entries: '11111111A' and '22222222A', each with a checkbox. A 'New Person' button is to the right.
- Add:** A button at the bottom right.

# Exemple de formulari *no principal*

```
public partial class NewReservationForm : Form
{
    private IVehicleRentalService service;
    private NewPersonForm newPersonForm;
    private NewCustomerForm newCustomerForm;
    private Customer previousCustomerAdded;
    private string previousSelectedCustomerDNI;
```

```
    public NewReservationForm(IVehicleRentalService service)
    {
        InitializeComponent();
        this.service = service;
        newPersonForm = new NewPersonForm(service);
        newCustomerForm = new NewCustomerForm(service);
        LoadData();
    }...
```

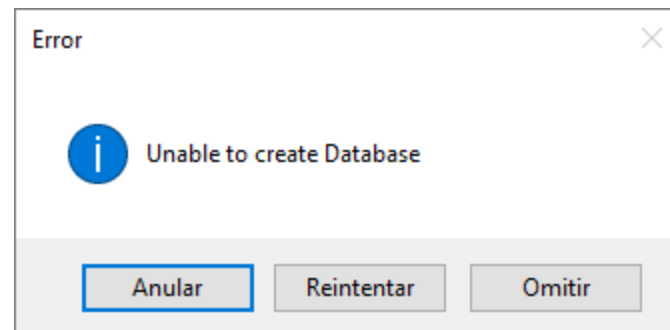
Rep paràmetres en el constructor

Mètode a implementar per carregar les dades en aquest formulari

# Quadres de Diàleg simples: MessageBox

- La classe `MessageBox` ens proporciona quadres de diàleg simples i de comportament modal (sòls permet la invocació de `Show()`).
- Se pot definir el títol de la finestra, el missatge a mostrar, els botons que apareixen i la icona del `MessageBox`, mitjançant els paràmetres que es passen en el mètode `Show()`.

```
DialogResult answer = MessageBox.Show(this, // Owner
    "Unable to create DB", // Message
    "Error", // Title
    MessageBoxButtons.AbortRetryIgnore, // Buttons included
    MessageBoxIcon.Exclamation); // Icon
if (answer == DialogResult.Retry)
{
    // Retry operation...
}
```

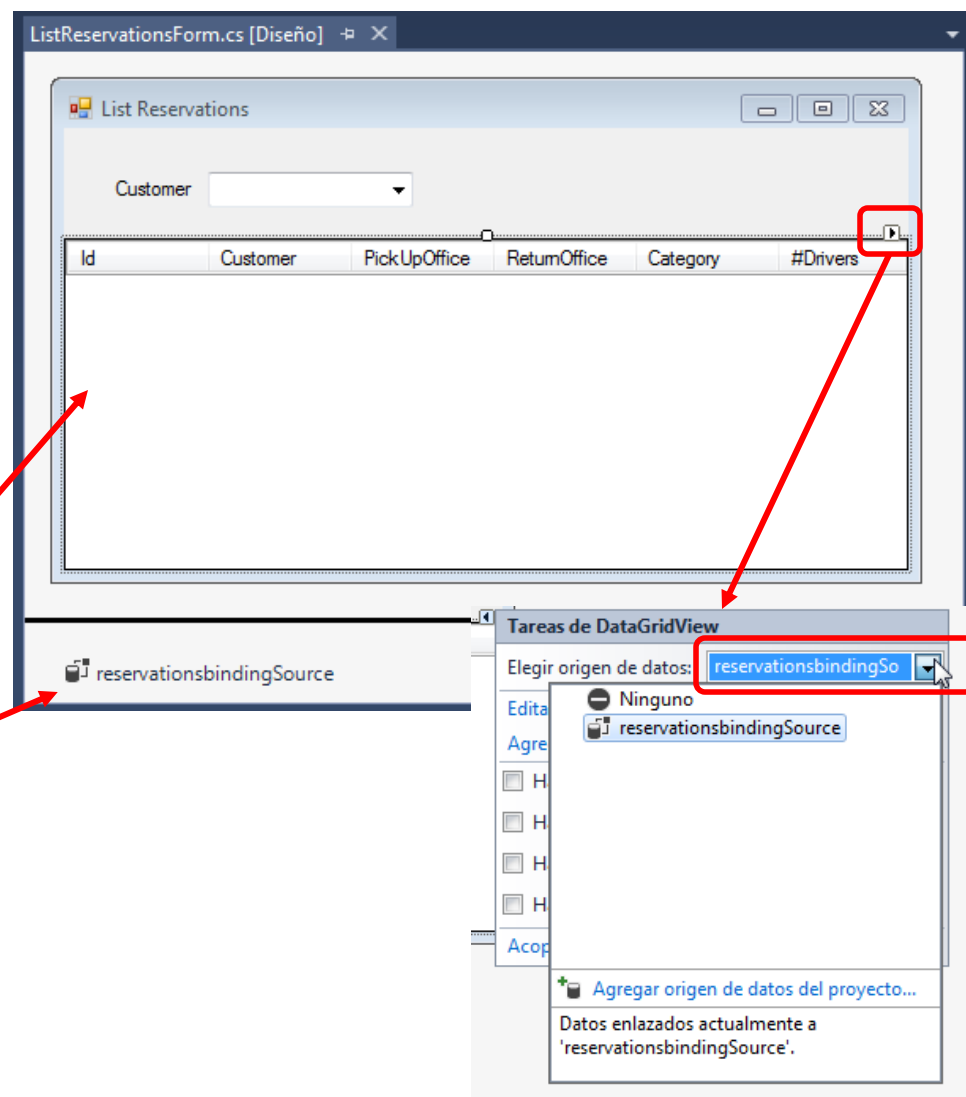
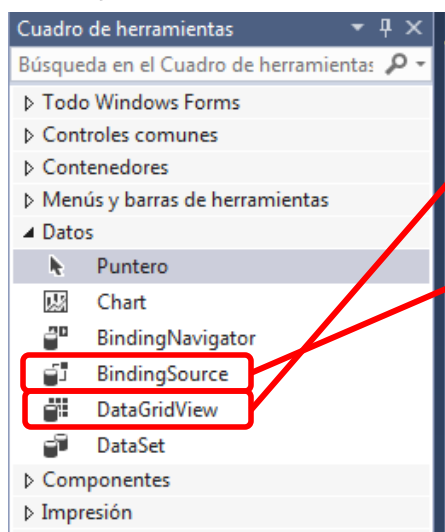


# Altres quadres de diàleg

- Operacions habituals com obrir i guardar fitxers, imprimir, seleccionar colors, etc.
  - `OpenFileDialog`, `SaveFileDialog`, `FolderBrowserDialog`, `ColorDialog`, `FontDialog`, `PageSetupDialog` y `PrintDialog`.
- Estenen a la `CommonDialog`, que defineix els mètodes i esdeveniments comuns a totes aquestes classes. Es mostren amb el mètode `ShowDialog()`, que torna un objecte de tipus `DialogResult` que pot prendre dos valors:
  - `DialogResult.OK` si l'usuari prem el botó OK en el quadre de diàleg, o
  - `DialogResult.CANCEL` en altre cas.

# Visualització del conjunt de dades

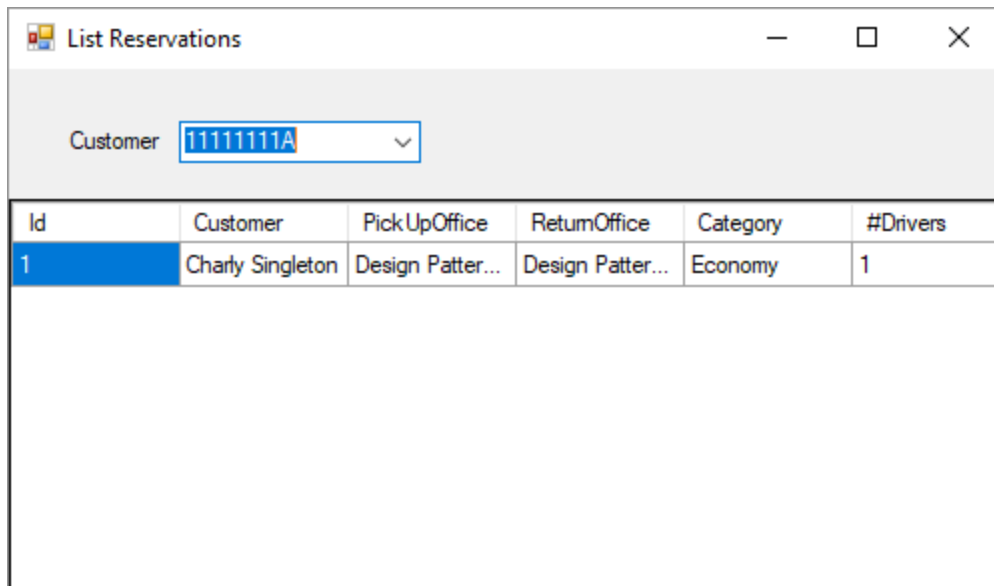
1. Afegeix el control BindingSource i assignar un nom.
2. Afegeix el control DataGridView
3. Assignar l'origen de dades del control
4. Afegeix columnes (següent diapositiva...)



# Visualització del conjunt de dades

Després de afegir les columnes, es deuen editar per assignar el nom de la propietat.

# Visualització del conjunt de dades



Id	Customer	Pick Up Office	Return Office	Category	#Drivers
1	Charly Singleton	Design Patter...	Design Patter...	Economy	1

## Funcionalitat desitjada:

- Quan es mostra el formulari, es podrà seleccionar un client (Customer) del ComboBox.
- Una vegada seleccionat un client es mostrarà les seues reserves en el control DataGridView.

# Visualización de colección de datos

1. Cuando se crea el formulario se deben añadir los clientes existentes al control de tipo ComboBox (customersComboBox). En el ejemplo, se hace en el método LoadData()

```
public ListReservationsForm(IVehicleRentalService service) : base(service)
{
    InitializeComponent();
    LoadData(); // se cargan los datos en el formulario
}

public void LoadData()
{
    ICollection<Customer> customers = service.findAllCustomers();
    customersComboBox.Items.Clear();
    if (customers!=null)
        foreach (Customer c in service.findAllCustomers())
            customersComboBox.Items.Add(c.Dni);
    customersComboBox.SelectedIndex = -1;
    customersComboBox.ResetText();
    reservationsbindingSource.DataSource = null;
}
```



# Visualización de colección de datos

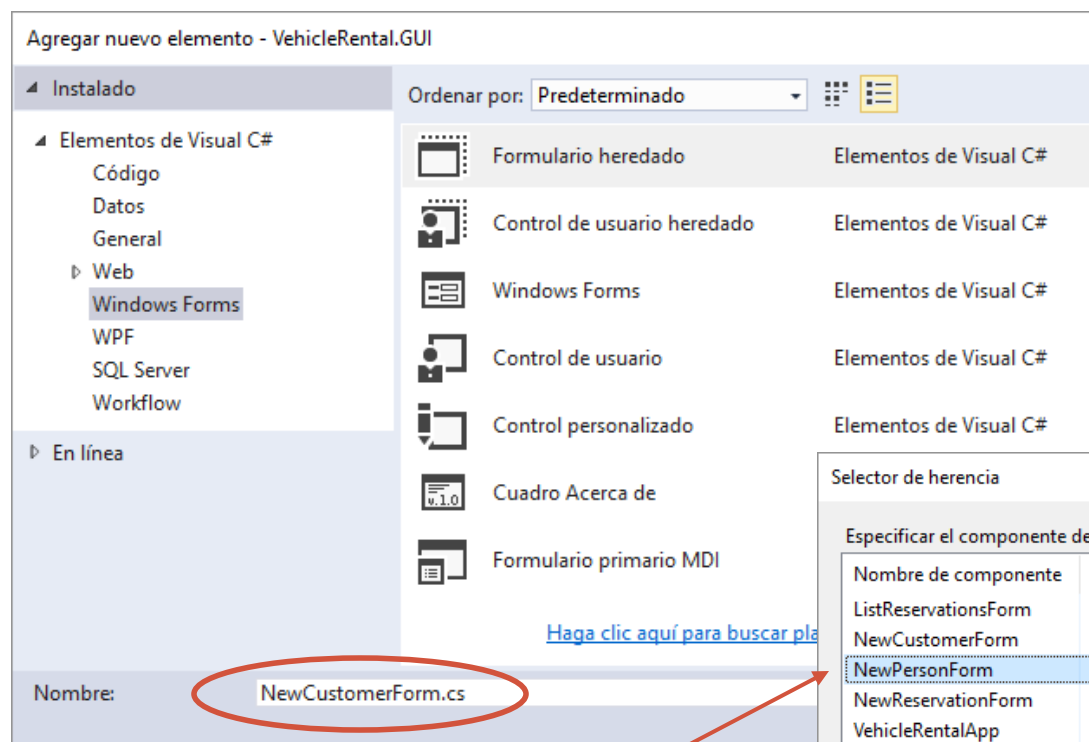
En seleccionar un element en el control ComboBox s'ha d'emplenar el control DataGridView. Es cridarà al gestor de l'esdeveniment SelectedIndexChanged del control ComboBox.

```
private void customersComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    string dni = (string) customersComboBox.SelectedItem;
    ICollection<Reservation> reservations = service.findReservationsbyCustomerID(dni);

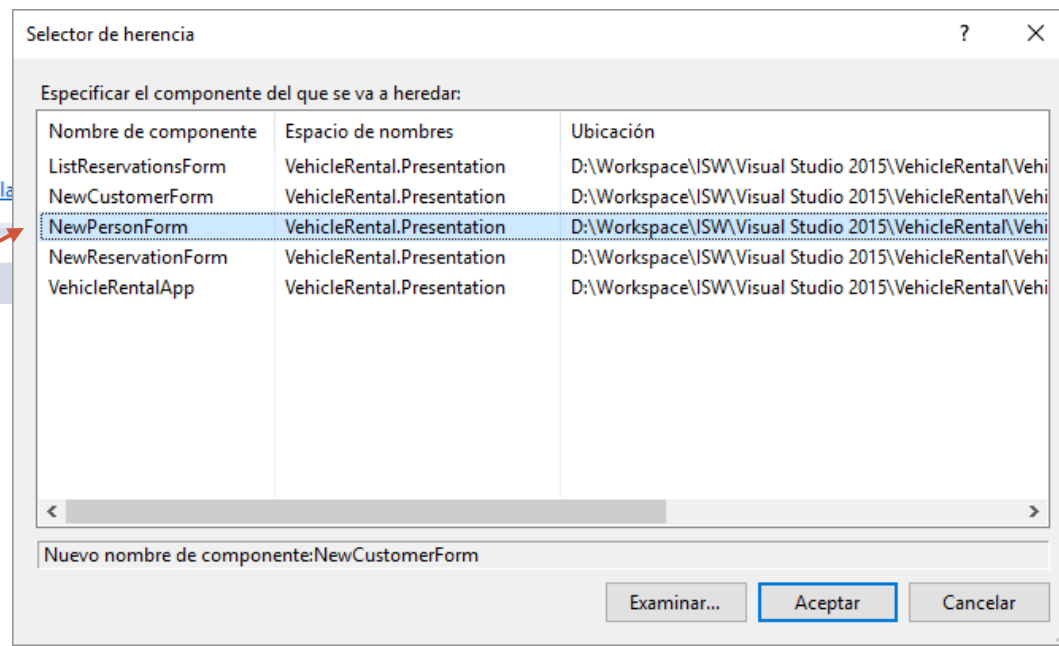
    //A BindingList of anonymous objects is used to provide the data model to the DataGrid

    BindingList<object> bindinglist = new BindingList<object>();
    foreach (Reservation r in reservations)
        //Adding one anonymous object for each reservation obtained
        bindinglist.Add(new
        {
            //ds_... are DataPropertyNames defined in the DataGridView object
            //see DataGridView column definitions in Visual Studio Designer
            ds_Id = r.Id,
            ds_Customer = r.Customer.Name,
            ds_PickUpOffice = r.PickUpOffice.Address,
            ds_ReturnOffice = r.ReturnOffice.Address,
            ds_Category = r.Category.Name,
            ds_NumDrivers = r.Drivers.Count
        });
    reservationsbindingSource.DataSource = bindinglist;
}
```

# Herència visual



La herència visual ens permet **reutilitzar** tant **comportament** com **aparença**



El nou formulari va a heretar de NewPersonForm

**NOTA:** es necessari compilar primer el formulari base NewPersonForm

# Herència visual. Reutilització de comportament

Tots els formularis utilitzen `IVehicleRentalService`. Per tant, crearem un formulari base `VehicleRentalFormBase` amb aquesta referència i **tots els formularis** heretaran d'ell

```
// Visual Studio no ho permet, però VehicleRentalFormBase seria una classe abstracta
public partial class VehicleRentalFormBase : Form
{
    private IVehicleRentalService service; // també podria ser atribut protected

    public VehicleRentalFormBase()
    {
        InitializeComponent();
    }

    public VehicleRentalFormBase(IVehicleRentalService service) : this()
    {
        this.service = service;
    }
}
```

# Herència visual. Reutilització de comportament

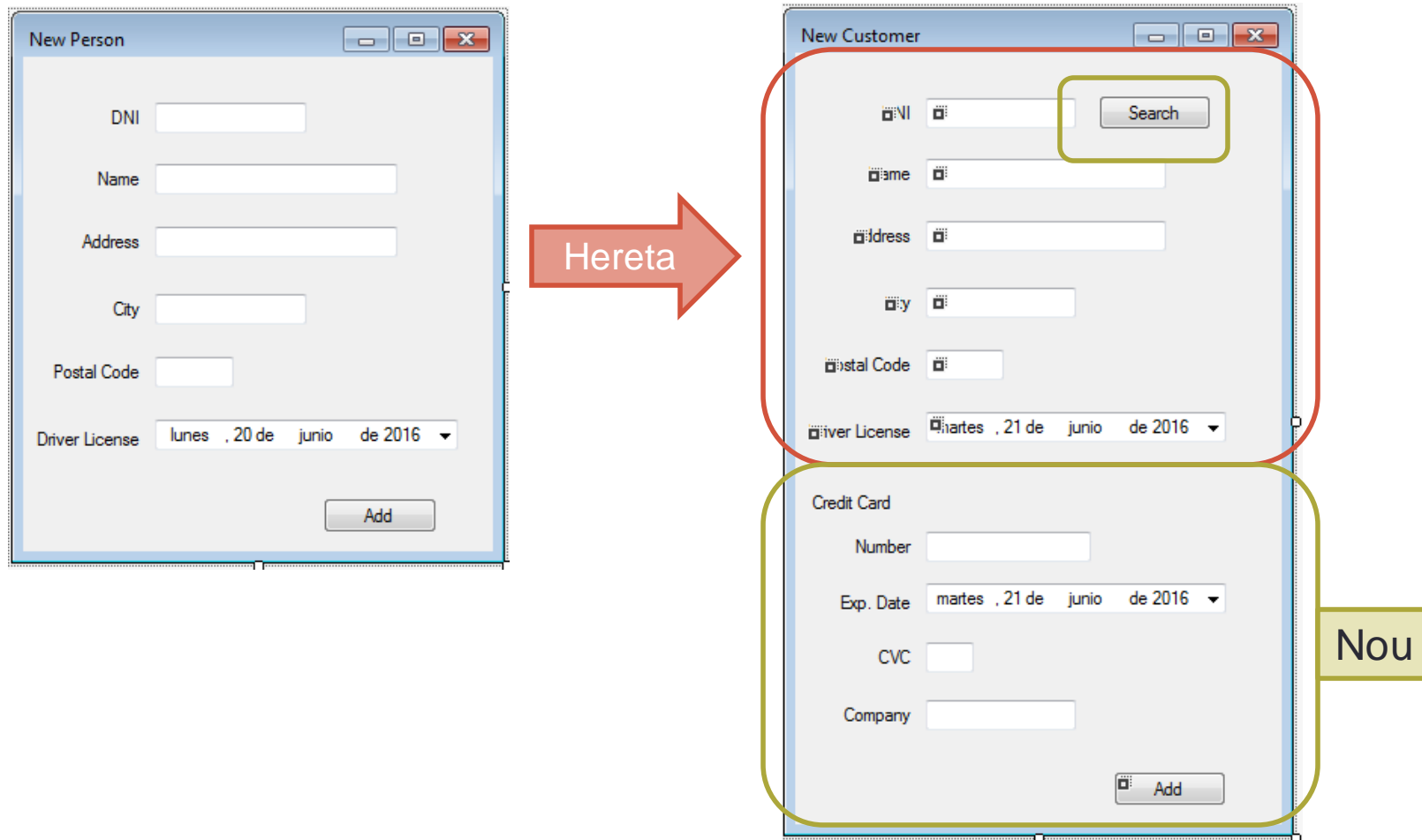
Per **exemple**, el formulari VehicleRentalApp...

```
public partial class VehicleRentalApp : VehicleRentalFormBase
{
    private ListReservationsForm listReservationForm;
    private NewReservationForm newReservationForm;

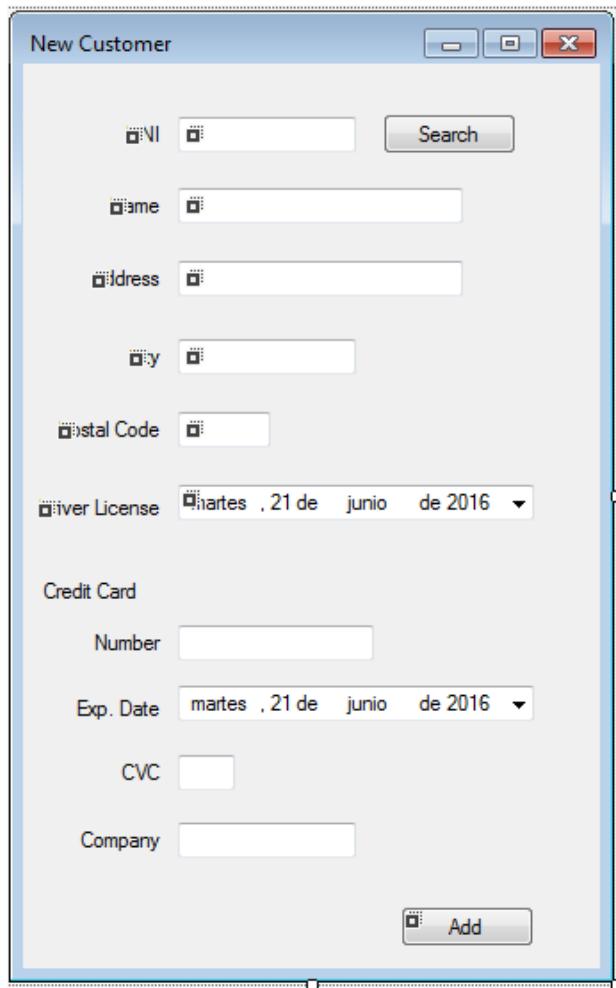
    public VehicleRentalApp(IVehicleRentalService service) : base(service)
    {
        InitializeComponent();
        listReservationForm = new ListReservationsForm(service);
        newReservationForm = new NewReservationForm(service);
    }
    ...

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

# Herència visual. Reutilització de l'aparença



# Herència visual. Reutilització de l'aparença



New Customer

NI  Search

Name

Address

City

Postal Code

Driver License  martes , 21 de junio de 2016

Credit Card

Number

Exp. Date  martes , 21 de junio de 2016

CVC

Company

Add

```
public partial class NewCustomerForm : NewPersonForm
{
    public NewCustomerForm() : base()
    {
        InitializeComponent();
    }

    public NewCustomerForm(IVehicleRentalService service)
    : base(service)
    {
        InitializeComponent();
    }
}
```

# Bibliografia bàsica

- D. Stone, C. Jarrett, M. Woodroffe. User Interface Design and Evaluation. Morgan Kaufmann, 2005
- S. Lauesen. User Interface Design. A Software Engineering Perspective. Addison Wesley, 2005
- Shneiderman, B. y Plaisant, C. Designing the User Interface. Pearson 5th ed., 2010

# Recursos

- Tutorials sobre els formularis Windows Forms

[https://msdn.microsoft.com/es-es/library/zftbwa2b\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/zftbwa2b(v=vs.110).aspx)

- Tutorial 1: Crear un visor d'imatge

<https://msdn.microsoft.com/es-es/library/dd492135.aspx>