

Cuadros de diálogo, validación y flujos de trabajo

Cuadros de diálogo y flujos modales · Validación y el patrón

Guardar/Cancelar

 LearnWisej

Cuadros de diálogo y flujos modales

Qué es un diálogo, cuándo usarlo en lugar de una página, el flujo modal en cinco pasos y en qué ayudan los cuadros de mensaje.



Objetivo de la lección

Entiende los **diálogos** y el flujo modal de alta y edición en el que se apoyan las aplicaciones de gestión: abrir una ventana centrada en una tarea, dejar que el usuario la complete, confirmar las acciones arriesgadas y devolver un resultado claro.

1. ¿Qué es un diálogo?

Un **diálogo** es una pantalla más pequeña que aparece encima de la página principal para que el usuario complete una tarea concreta. En este curso los diálogos más habituales son **Create Ticket** y **Edit Ticket**. Un diálogo no debe intentar ser toda la aplicación: recoge los valores necesarios, los valida y devuelve un resultado claro.



Idea clave

Un diálogo es *modal*: el usuario lo completa o lo cancela antes de volver a la página que hay detrás. Esa concentración en una sola tarea es lo que hace predecibles los flujos de alta y edición.

2. Diálogo frente a página

Tipo de pantalla	Mejor uso	Ejemplo
Página / área de contenido principal	Pantallas grandes de la aplicación donde los usuarios pasan tiempo y navegan entre funcionalidades.	Página Tickets, página Customers, página Settings.
Diálogo / ventana modal	Un flujo pequeño y concreto que debe completarse o cancelarse antes de volver.	Diálogo Create Ticket, diálogo Edit Ticket, mensaje Confirm Delete.

3. El patrón habitual del flujo modal

Casi todas las funcionalidades de alta y edición siguen el mismo flujo de cinco pasos. Respetarlo hace que el flujo sea legible y predecible.

Clic en botón	→ el usuario elige New o Edit
Abrir el diálogo	→ se muestra el formulario / ventana modal
Validar	→ se comprueban los campos obligatorios
Guardar	→ se llama a la capa de servicio
Actualizar la UI	→ se actualiza la cuadrícula y se informa al usuario

El usuario hace clic en **New** o **Edit**, la aplicación abre un diálogo, el diálogo valida los campos, el botón **Save** llama a un servicio y la página principal se actualiza *después* de que el diálogo se cierre correctamente.

4. Cuadros de mensaje y confirmaciones

Usa cuadros de mensaje cuando el usuario necesite una respuesta inmediata o deba confirmar una acción arriesgada.

Situación	Respuesta recomendada
Guardado completado	Muestra un breve mensaje de éxito y actualiza la lista de tickets.
Falta un campo obligatorio	Muestra un mensaje de validación claro junto al campo o en la parte superior del diálogo.
Eliminar el ticket seleccionado	Pide confirmación antes de eliminar.
Cancelar después de cambiar valores	Opcional: avisa de que se perderán los cambios no guardados.



Ya estás listo para la siguiente lectura

Sabes describir qué es un diálogo, cuándo usar un diálogo en lugar de una página, el flujo modal de cinco pasos y en qué ayudan los cuadros de mensaje. A continuación: validar la entrada y el patrón Save/Cancel coherente.

Validación y el patrón Guardar/Cancelar

Valida cerca de la interfaz sin encerrar ahí las reglas de negocio, escribe un manejador Save legible y aplica un patrón Save/Cancel coherente.



Objetivo de la lección

Valida la entrada cerca de la interfaz sin encerrar ahí las reglas de negocio, escribe un manejador Save legible y aplica un patrón Save/Cancel coherente para que los usuarios siempre sepan qué ha pasado.

1. Validación: cerca de la interfaz, pero sin quedar atrapada en ella

La validación debe estar lo bastante cerca de la interfaz para que los usuarios reciban una respuesta inmediata, pero las reglas de negocio importantes deben poder probarse fuera del clic del botón. El diálogo comprueba los problemas simples de los campos y después llama a una clase de servicio para las reglas reutilizables.

Tipo de validación	Ejemplo	Buen lugar
Validación simple de la interfaz	El cuadro de texto del título está vacío; no se ha seleccionado Priority.	Un método del diálogo como <code>ValidateForm()</code> .
Regla de negocio	Un Support Agent no puede editar un ticket cerrado.	<code>TicketService</code> o un servicio de validación.
Regla de persistencia	El ID del ticket ya existe en el almacenamiento.	La capa de repositorio / servicio.

2. Gestión segura de eventos

Los eventos de clic de los botones deben ser legibles. No escondas llamadas lentas a la base de datos ni lógica de negocio complicada dentro de un evento trivial de la interfaz, salvo que sea intencionado y tenga un nombre claro. Un buen evento Save llama a pequeños helpers como `ValidateForm()`, `BuildTicketFromFields()` y `ticketService.SaveTicket(ticket)`.

```
private void btnSave_Click(object sender, EventArgs e)
{
    if (!ValidateForm())
        return;

    Ticket ticket = BuildTicketFromFields();
    ticketService.SaveTicket(ticket);

    this.DialogResult = DialogResult.OK;
    this.Close();
}
```



Valida antes de guardar

El manejador sale antes de tiempo cuando `ValidateForm()` falla: no se guarda nada y el diálogo sigue abierto. Solo la entrada válida llega al servicio.

3. Un patrón Save / Cancel coherente

- **Save** valida primero, después llama al servicio y cierra el diálogo *solo cuando todo ha ido bien*.
- **Cancel** cierra el diálogo sin guardar.
- Los mensajes de error deben decirle al usuario exactamente qué debe corregir.
- La página padre actualiza la cuadrícula solo cuando el diálogo devuelve `OK`.
- Mantén claros los nombres de los botones: `btnSave`, `btnCancel`, `btnNew`, `btnEdit`.

4. Hábitos de desarrollo para este módulo

- Nombra los controles con claridad para que el código de los eventos sea legible.
- Mantén el diseño del diálogo sencillo y centrado en su tarea.
- Separa el estado de la interfaz, las reglas de negocio y los datos guardados.
- Usa servicios para las operaciones de tickets reutilizables.
- Usa los cuadros de mensaje y los mensajes de validación de forma coherente.



Ya estás listo para el laboratorio

A continuación: mira el recorrido guiado de los diálogos, supera la evaluación de conocimientos y después construye diálogos Create y Edit Ticket reutilizables: valida primero, guarda a través del servicio, devuelve `DialogResult.OK` y actualiza la cuadrícula con una respuesta clara.