

Experiencia avanzada de flujos de trabajo: asistentes, orquestación modal y compensación

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Flujos de negocio complejos, resultados tipados, deshacer/compensación, flujos de trabajo reanudables. Lo que esto significa para un equipo avanzado de Wisej.NET — y cómo leer este módulo.



Enfoque de producción

Flujos de negocio complejos, resultados tipados, deshacer y compensación, flujos de trabajo reanudables.

1. El problema en producción

Los flujos de trabajo empresariales rara vez caben en un único botón Save. Las pantallas avanzadas coordinan aprobaciones, asignaciones, validaciones, cargas de archivos y comprobaciones externas sin que el usuario pierda la confianza.

2. Conceptos clave

Concepto	Por qué importa
orquestación del flujo de trabajo	Un servicio que es dueño de la secuencia, las reglas y el estado de un proceso de negocio de varios pasos, para que el flujo no quede enterrado entre varios formularios.
estado del asistente	Las respuestas acumuladas y el estado de validación de un asistente de varios pasos, guardados en un único objeto tipado en lugar de dispersos entre los controles.
resultado tipado	Un valor de retorno que modela explícitamente el éxito, el fallo de validación y los errores, de modo que quien llama debe gestionar cada ruta en lugar de depender de excepciones.
comando de compensación	Una operación que deshace los efectos de un paso anterior cuando falla un paso posterior, manteniendo el sistema coherente sin una única transacción gigantesca.
flujo de trabajo reanudable	Un proceso que persiste su progreso para que el usuario pueda cerrar el navegador y continuar más tarde, o para recuperarse de un fallo desde el último paso correcto.
matriz de rutas de fallo	Una tabla que enumera cada paso, lo que puede salir mal en él y cómo responde el flujo, para que la gestión de fallos se diseñe en lugar de improvisarse.

3. Desarrollo de la lección

Un flujo de trabajo no es una secuencia de formularios

Un asistente es solo la parte visible de un proceso de negocio. La pregunta de diseño avanzada es dónde vive el estado del flujo, qué servicio es dueño de las transiciones y cómo se reanuda la UI tras una interrupción.

Los resultados tipados mantienen el código honesto

Los diálogos y las páginas de un asistente deberían devolver una intención tipada, no indicadores mágicos dispersos entre los controles. Así las transiciones del flujo se pueden probar y son más fáciles de revisar.

Comandos tipados de entrada, resultados tipados de salida

El Escalation Wizard recoge un motivo, archivos adjuntos, un aprobador, una fecha límite y opciones de notificación a lo largo de seis páginas. Ninguna de esas páginas ejecuta lógica de negocio. Rellenan un único `EscalationCommand` y, cuando el usuario confirma, un servicio de flujo de trabajo lo ejecuta y devuelve un `WorkflowResult`. El comando es un objeto simple con todos los campos tipados y validados, así que se puede construir en una prueba unitaria tan fácilmente como en el asistente.

El resultado también está tipado. Indica si se ha creado el escalado, qué validaciones han fallado y en qué campo, si un paso externo como la notificación ha tenido éxito y qué debe hacer el usuario a continuación. Quien llama evalúa esos desenlaces en lugar de capturar excepciones o inspeccionar cadenas. Una página del asistente que recibe un fallo de validación puede saltar al paso correcto y resaltar el campo; un llamador por lotes puede registrar el mismo resultado sin ninguna UI.

La primera pregunta de repaso del laboratorio es si el flujo de trabajo se puede probar sin el asistente. Con un comando y un resultado tipados, la respuesta es sí, y esa prueba existe antes que el asistente.

Asistentes y ventanas modales que siguen siendo ligeros

Un diálogo modal o una página de asistente es un formulario, y en este curso los formularios son ligeros. La página recoge valores, realiza la validación de entrada necesaria para habilitar el botón siguiente y escribe los valores en el objeto de estado del asistente. No decide si el aprobador está permitido, si la fecha límite es aceptable para esa prioridad ni si se requiere un adjunto. Esas son reglas de negocio, y viven en el servicio de flujo de trabajo, donde se aplican a todos los llamadores.

El objeto de estado del asistente es la parte visible de esa disciplina. Es un único objeto tipado que contiene todo lo recogido hasta el momento y qué pasos están completos. Las páginas leen y escriben en él, la última página lo convierte en el comando y, como es un solo objeto, se puede persistir, que es lo que hace que el flujo sea reanudable tras actualizar el navegador. La segunda pregunta del laboratorio, qué estado sobrevive a una actualización, se responde con lo que haya en ese objeto y esté guardado en el servidor.

Una ventana modal que llama a un repositorio, o una página de asistente que contiene un `DbContext`, esconde lógica de negocio en un Form y acabará duplicada la primera vez que se necesite el mismo flujo sin el asistente.

Compensación donde las transacciones no llegan

Crear un escalado persiste un registro, envía una notificación a través de un sistema externo y escribe una entrada de auditoría. Una transacción de base de datos cubre el primero y el último, pero no la notificación, que ya se ha enviado cuando falla la escritura de auditoría. Las transacciones distribuidas entre sistemas externos rara vez están disponibles y rara vez compensan su coste, así que el flujo usa compensación: para cada paso que no se puede revertir, define una acción compensatoria que lo deshaga o lo neutralice.

La ruta de fallo del laboratorio es que la notificación falle después de que la persistencia haya tenido éxito. El escalado se conserva, porque es válido, y un comando de compensación registra que la notificación está pendiente y programa un reintento o un seguimiento manual. El caso inverso, que la auditoría falle después de la notificación, se compensa registrando explícitamente el hueco de auditoría y lanzando una alerta, porque una notificación enviada no se puede anular.

La compensación se diseña paso a paso y se pone por escrito antes que el código. La pregunta para cada paso es qué aspecto debería tener el sistema si este paso falla después de que los anteriores hayan tenido éxito.

Una matriz para cada ruta de fallo

La matriz de rutas de fallo es una tabla con una fila por cada paso del flujo y una columna por cada forma en que puede terminar: confirmación, cancelación, tiempo de espera agotado, fallo de validación y fallo externo. Cada celda indica qué hace el sistema, qué ve el usuario, qué se persiste y qué se audita. Rellenar la matriz saca a la luz las rutas que nadie había diseñado, como qué ocurre cuando la selección del aprobador agota el tiempo de espera o cuando el usuario cancela después de haber cargado los adjuntos.

La matriz es también el plan de pruebas. Cada celda se convierte en al menos una prueba del servicio de flujo de trabajo, y las celdas que implican la UI se convierten en comprobaciones manuales en el guion de la demo. Cuando un revisor pregunta qué ocurre si la comprobación externa no está disponible, la respuesta es una referencia a una celda, no una suposición.

- Confirmación: el comando se ejecuta, se muestra el resultado y se escribe la entrada de auditoría.
- Cancelación: el estado recogido se descarta o se guarda como borrador, y se eliminan los archivos cargados.
- Tiempo de espera agotado: el flujo se puede reanudar desde su estado persistido y nada queda aplicado a medias.
- Fallo de validación: el usuario vuelve al paso que falla con el campo resaltado.
- Fallo externo: se ejecuta la compensación de ese paso y se registra el desenlace.



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Diseñar flujos de trabajo de varios pasos con comandos y resultados tipados.
- ✓ Usar flujos modales y de asistente sin esconder lógica de negocio en los Forms.
- ✓ Implementar compensación o deshacer cuando no es posible usar transacciones reales.
- ✓ Documentar las rutas de confirmación, cancelación, tiempo de espera agotado, fallo de validación y fallo externo.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Crea un Escalation Wizard que recoja el motivo, los adjuntos, el aprobador, la fecha límite y las opciones de notificación. Implementa la orquestación en un servicio, no en las páginas del asistente, y añade compensación para un fallo de notificación simulado.

Enfoque sugerido

- ✓ Lee el problema en producción e identifica el riesgo de arquitectura antes de tocar la UI.
- ✓ Crea o actualiza primero el límite correspondiente de servicio, comando, modelo o componente.
- ✓ Construye la pantalla o el componente Wisej.NET siguiendo los estándares de proyecto establecidos.
- ✓ Añade una demostración de la ruta de fallo, no solo del caso ideal.
- ✓ Anota lo que ha cambiado en el diario de aprendizaje y actualiza la lista de comprobación correspondiente.

Entregables requeridos

- ✓ Flujo de pantallas del asistente
- ✓ Servicio de flujo de trabajo
- ✓ Objetos tipados de comando y resultado
- ✓ Matriz de rutas de fallo
- ✓ Ejemplo de compensación

Preguntas de repaso

- ✓ ¿Se puede probar el flujo de trabajo sin el asistente?
- ✓ ¿Qué estado sobrevive a una actualización del navegador?
- ✓ ¿Qué ocurre si la notificación tiene éxito pero falla el registro de auditoría?



Regla avanzada de producción

No te detengas cuando la pantalla parezca correcta. Explica a quién pertenece el estado, las implicaciones de seguridad, las rutas de fallo y el comportamiento en producción antes de dar el módulo por terminado.