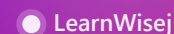


Auditoría, secretos y configuración

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

Registros de auditoría para comandos sensibles, ID de correlación, gestión de secretos, configuración por entorno y logging sin filtrar datos. Qué significa esto para un desarrollador que protege una aplicación Wisej.NET en producción, y cómo leer este módulo.



Enfoque en producción

Registros de auditoría para comandos sensibles, ID de correlación, gestión de secretos, configuración por entorno y logging sin filtrar datos.

1. El problema en producción

AccessOps ya autentica a las personas, resuelve sus permisos y protege sus documentos, y nada de eso sirve la mañana en que alguien pregunta quién aprobó la solicitud 4471, desde qué dirección y si antes se rechazó a alguien. Los cinco módulos anteriores decidieron lo que está permitido; este decide lo que se puede reconstruir después. También cierra el hueco que convierte un incidente pequeño en uno grande: una contraseña de base de datos o un secreto de cliente OIDC en un archivo confirmado en el repositorio hace dos años.

2. Conceptos clave

Concepto	Por qué importa
registro de auditoría	Un registro de negocio de cada comando sensible —quién actuó, sobre qué, cuándo y desde dónde— conservado durante un periodo de retención establecido, porque alguien tendrá que responder de ello.
almacenamiento de solo anexo	Una tabla de auditoría en la que la cuenta de la aplicación solo puede hacer <code>INSERT</code> . Si el código que escribe el historial también puede reescribirlo, el historial no demuestra nada.
evidencia de manipulación	Una secuencia sin huecos más una cadena de hashes, en la que el hash de cada fila cubre sus propios campos y el hash anterior, de modo que una fila editada rompe la cadena.
ID de correlación	Un identificador generado cuando empieza un comando y repetido en cada línea de log y cada fila de auditoría que provoca, para poder reconstruir después un único clic.
ámbito de logging	Un bloque de propiedades ambiental — <code>ILogger.BeginScope</code> o un enricher de Serilog — que añade el ID de correlación y el actor a todo lo que se registra dentro de él.
almacén de secretos	El lugar donde vive una credencial fuera del build: user secrets en local, variables de entorno en un contenedor, un almacén gestionado en producción, o una identidad administrada, sin nada almacenado en absoluto.
enmascaramiento	Una regla sobre lo que nunca llega a un destino de logs —contraseñas, tokens, cookies, cadenas de conexión y datos personales— aplicada mediante logging estructurado, no mediante la memoria.

3. Desarrollo de la lección

El registro que sobrevive a la discusión

Hay dos cosas distintas a las que se llama «logging». Una línea de log es de diagnóstico y puede mostrarse o rotarse y desaparecer al cabo de una semana. Una fila de auditoría es una prueba: tiene un periodo de retención, un responsable y un lector que no es desarrollador. AccessOps necesita un `AuditService` como único punto de entrada para el segundo tipo, al que llama cada comando sensible: aprobar una solicitud, conceder o revocar un permiso, descargar un documento confidencial y cada intento de inicio de sesión del Módulo 2. Nada más escribe en la tabla de auditoría.

Una fila responde a seis preguntas: el identificador estable del actor, el nombre de la acción (`Approval.Approve` , `Document.Download`), el tipo y el ID del destino, la marca de tiempo UTC del reloj del servidor, la dirección y el user agent del cliente, y el ID de correlación. Registra también el resultado, porque una denegación suele ser más interesante que un éxito. En Wisej.NET, la dirección de red y el user agent del navegador del cliente proceden del contexto de aplicación por sesión y no de nada que envíe la página, así que captúralos una vez al crear el `SessionContext` tras el inicio de sesión. Toma también el actor de ese contexto verificado, nunca de un control ni de un parámetro proporcionado por el llamante.

Después haz que la fila sea difícil de borrar. El login de base de datos de la aplicación recibe `INSERT` y nada más, para que un error o una sentencia inyectada no puedan borrar en silencio un martes entero. Da a cada fila una secuencia sin huecos y un hash sobre sus propios campos más el hash de la fila anterior; un proceso programado vuelve a calcular la cadena y nombra la primera fila que ya no coincide. Eso es *evidencia* de manipulación, no prevención: hace visible un cambio en lugar de imposible. Por último, escribe la fila de auditoría en la misma transacción que el cambio de negocio, para que una aprobación sin su registro sea imposible.

```
using (_log.BeginScope(new Dictionary<string, object>
    { ["CorrelationId"] = ctx.CorrelationId, ["UserId"] = ctx.UserId }))
using (var tx = _db.BeginTransaction())
{
    _approvals.Approve(requestId, ctx);           // the business change
    _audit.Record(AuditEntry.For(ctx,             // actor, address, correlation
        "Approval.Approve", "ApprovalRequest", requestId));
    tx.Commit();                                 // both, or neither
}
```

Un ID, una historia

El ID de correlación se crea donde empieza un comando del usuario: al principio de un controlador de eventos delgado, antes de llamar a ningún servicio. Basta con un `Guid`; si ya trabajas con `System.Diagnostics.Activity`, usa el identificador de traza de la actividad actual para que el ID coincida con todo lo demás que hable el formato de trazas del W3C. Ponlo en el contexto por comando que también lleva el actor y la dirección del cliente, y pasa ese contexto a cada servicio. No vuelvas a la sesión desde dentro de un servicio: el trabajo en segundo plano iniciado con `Application.StartTask` puede sobrevivir a la sesión que lo inició.

Añadir el ID a las líneas de log no es tarea del llamante. Abre un ámbito de logging alrededor del comando — `ILogger.BeginScope` con un diccionario de propiedades, o el `LogContext` de Serilog con un enricher— y cada línea escrita dentro de él llevará automáticamente el ID de correlación y el actor, incluidas las líneas de las bibliotecas que están por debajo. Usa logging estructurado para que el ID sea una propiedad con nombre en el destino, lo que convierte «muéstrame todo lo de la correlación 7f3c» en una consulta y no en un grep. Después hazlo visible: cuando un comando falle, termina la frase amable con el ID como referencia.

Los secretos pertenecen al entorno, no al repositorio

AccessOps tiene ya tres credenciales: la cadena de conexión de la base de datos, el secreto de cliente OIDC del Módulo 4 y la clave de firma de los enlaces compartidos con caducidad del Módulo 5. Ninguna debe estar en un archivo con control de versiones. En desarrollo usa los user secrets de .NET: el proyecto recibe un `UserSecretsId` y los valores viven fuera de la carpeta de la solución, así que un clon del repositorio no contiene credenciales. En el despliegue, proporcióñalas como variables de entorno o léelas de un almacén gestionado mediante un proveedor de configuración. Mejor aún, usa una identidad administrada: la aplicación se autentica como ella misma y recibe un token de vida corta, así que no hay ningún secreto almacenado que pueda filtrarse.

Esto funciona porque la configuración de ASP.NET Core está en capas y las fuentes posteriores prevalecen: `appsettings.json`, después la capa del entorno, después user secrets en desarrollo, después las variables de entorno y después un proveedor de almacén. Los archivos confirmados contienen la estructura y valores predeterminados inofensivos; los valores con forma de secreto llegan por entorno. El propio `Default.json` de Wisej.NET configura el marco de la aplicación —tema, expiración de la sesión, ventana de inicio— y es tan poco un almacén de secretos como `Web.config`.

Un secreto que no puedes rotar con calma es uno que rotarás mal a las 3 de la madrugada, así que el procedimiento de rotación es aquí un entregable. Anota, por cada secreto: el responsable, el almacén, la caducidad y los pasos de sustitución. Diseña para que haya dos credenciales válidas a la vez —un segundo secreto de cliente, un segundo login de base de datos— para que la rotación sea un reinicio escalonado y no una caída. Y una vez que un valor se ha confirmado en el repositorio, borrar la línea no sirve: róvalo y revoca el antiguo.

Registra el incidente, no a la persona

Los logs se copian a índices de búsqueda, se envían a servicios de terceros, se abren en portátiles de soporte y se conservan mucho más tiempo del que nadie pretende. Deben contener lo suficiente para reconstruir un incidente y nada que sea dañino por sí mismo. Nunca registres una contraseña (ni siquiera una fallida), un token, una cabecera de autorización, una cookie, una cadena de conexión ni el contenido de un documento confidencial.

Los datos personales exigen la misma disciplina por otra razón: una fila de auditoría es un registro legítimo con una política de retención, mientras que una línea de depuración con la dirección particular de un empleado es una copia descontrolada de ese dato. Registra el ID estable del usuario en lugar del nombre y el correo, y el ID del documento en lugar de un nombre de archivo que puede ser sensible en sí mismo. El logging estructurado ayuda, porque registras propiedades con nombre a partir de una lista de permitidas en lugar de serializar un objeto de solicitud entero y cruzar los dedos.

- Nunca registres: contraseñas, tokens, cookies, cadenas de conexión, contenido de documentos.
- Registra por referencia: ID de usuario, ID de documento, ID de solicitud; no nombres ni datos personales en texto libre.
- Hazlo cumplir en el destino: una política de enmascaramiento, más una prueba que compruebe que un marcador de secreto conocido nunca llega a la salida.

Cierra el ciclo con una revisión y no con una regla: recorre de principio a fin la salida de log de una aprobación real y de una descarga real, y marca cada campo como necesario, enmascarado o eliminado.



Regla de seguridad

Nunca se confía en el navegador. Toda decisión importante —quién es el usuario, qué puede hacer, qué registro puede ver— se toma en el servidor, dentro de un servicio, y la interfaz solo la refleja. Si una regla solo existe como un botón oculto, no existe.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Registrar quién hizo qué, cuándo y desde dónde en cada comando sensible con un registro de auditoría que evidencie las manipulaciones.
- ✓ Llevar un ID de correlación a través de los logs para poder seguir una acción del usuario entre servicios.
- ✓ Mantener los secretos fuera del control de código fuente mediante configuración por entorno, almacenes de secretos e identidades administradas.
- ✓ Registrar lo suficiente para investigar un incidente sin registrar contraseñas, tokens ni datos personales.

Guía del laboratorio / examen

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



Laboratorio práctico

Añade auditoría y gestión de secretos a AccessOps: un `AuditService` que registra aprobaciones, cambios de permisos y descargas de documentos con el usuario, la hora, la dirección y el ID de correlación; una tabla de auditoría de solo anexo; un ID de correlación creado por comando y escrito en cada línea de log; las cadenas de conexión y el secreto de cliente OIDC trasladados a la configuración del entorno con un procedimiento de rotación documentado; y una revisión de los logs que enmascara los campos sensibles.

Punto de partida

Abres la solución AccessOps tal como está al final del Módulo 5: un `SessionContext` verificado creado al iniciar sesión, un `PermissionService` y una protección de autorización dentro de `DocumentService` y `ApprovalService`, el inicio de sesión OIDC y la función de documentos con su canal de subida y sus enlaces compartidos firmados. La pantalla de aprobaciones, el editor de permisos de administración y la lista de documentos funcionan; simplemente no dejan rastro de lo que hacen, y la cadena de conexión de la base de datos y el secreto de cliente OIDC siguen en un archivo de configuración bajo control de versiones.

Enfoque sugerido

- ✓ Crea la tabla de auditoría con una secuencia sin huecos, el actor, la acción, el tipo y el ID del destino, la marca de tiempo UTC, la dirección del cliente, el user agent, el resultado, el ID de correlación y una columna de hash, y concede después al login de base de datos de la aplicación solo `INSERT`: ni `UPDATE` ni `DELETE`.
- ✓ Escribe `AuditService` como único escritor de esa tabla, tomando el actor y los datos del cliente del `SessionContext` verificado, calculando el hash de cada fila sobre sus propios campos más el hash de la fila anterior e incluyendo la inserción en la misma transacción que el cambio de negocio.
- ✓ Añade un contexto por comando que genere un ID de correlación al principio de cada controlador de eventos, pásalo a `ApprovalService`, `DocumentService` y al editor de permisos, y abre un ámbito de logging para que cada línea escrita durante el comando lleve el ID y el actor.
- ✓ Saca la cadena de conexión, el secreto de cliente OIDC y la clave de firma de los enlaces compartidos de los archivos de configuración versionados y lléalos a `user secrets` en desarrollo y a variables de entorno en el despliegue; después redacta el procedimiento de rotación con responsables, fechas de caducidad y el solapamiento de dos credenciales válidas.
- ✓ Llama a `AuditService` desde la aprobación, la concesión y revocación de permisos y la descarga de documentos —registrando tanto las denegaciones como los éxitos—; después ejecuta una aprobación y una descarga y revisa cada línea de log resultante en busca de contraseñas, tokens, cabeceras, cadenas de conexión y datos personales.

Entregables requeridos

- ✓ AuditService y tabla de auditoría de solo anexo
- ✓ ID de correlación en cada comando y cada línea de log
- ✓ Secretos trasladados a la configuración del entorno
- ✓ Procedimiento de rotación de secretos
- ✓ Revisión del enmascaramiento de los logs

Preguntas de repaso

- ✓ Tu tabla de auditoría es de solo inserción para la aplicación y cada fila está encadenada por hash. Nombra algo que un atacante con el propio login de base de datos de la aplicación aún puede hacerle al registro y explica exactamente cómo lo pondría de manifiesto el proceso de verificación.
- ✓ Una aprobación se inicia en la pantalla de aprobaciones y termina en una tarea en segundo plano iniciada con `Application.StartTask` después de que el usuario haya cerrado sesión. ¿De dónde obtiene esa tarea su ID de correlación y su actor, y qué se rompe si los lee de la sesión?
- ✓ Un desarrollador elimina el secreto de cliente OIDC del archivo de configuración versionado y sube el cambio. ¿Por qué el secreto sigue comprometido y qué tiene que ocurrir antes de que el incidente quede realmente cerrado?



Regla de seguridad

Nunca se confía en el navegador. Toda decisión importante —quién es el usuario, qué puede hacer, qué registro puede ver— se toma en el servidor, dentro de un servicio, y la interfaz solo la refleja. Si una regla solo existe como un botón oculto, no existe.