Instituto Profesional DUOC UC

Escuela de Informática y Telecomunicaciones

Carrera: Ingeniería en Informática

Asignatura: Capstone

Informe Final de Proyecto

Módulo de Reportes Contraplano (MRC)

Integrado a WordPress mediante Plugin Propio

Integrantes del equipo:

- Marialejandra Méndez (Rol: Desarrollo / Frontend / Integración)
- Noel Saenz (Rol: QA / Pruebas / Control de Calidad)

Profesor: Daniel Montero

Empresa Colaboradora: Contraplano

Maipú, 21 de noviembre de 2025

Índice

1. Resumen Ejecutivo

El presente documento constituye el Informe Final del proyecto “Módulo de Reportes Contraplano (MRC)”, desarrollado como parte de la asignatura Capstone. El documento consolida la versión final del sistema, su alcance real, sus requisitos funcionales y no funcionales, la especificación actualizada de los casos de uso implementados, las funcionalidades descartadas por decisión del cliente, el análisis de trazabilidad y las conclusiones del trabajo realizado.

Durante el desarrollo del proyecto, se aplicó una metodología Scrumban, lo que permitió iterar, refinar el alcance y responder a cambios solicitados por la empresa Contraplano. Como resultado, el MRC se entregó como un módulo web integrado a WordPress mediante un plugin propio, que permite generar, validar, clasificar, listar y auditar reportes noticiosos con retención controlada de 90 días.

El proyecto final incluyó:

- ☐ Implementación completa de los casos de uso CU-01, CU-02, CU-04 y CU-05.
- ☐ Descartes justificados de CU-03, CU-06, CU-07 y CU-08, alineados al flujo editorial de la empresa.
- ☐ Despliegue operativo en Firebase Hosting y Firestore.
- ☐ Pruebas unitarias, validación en ambiente real y demostración a la empresa.

Este informe documenta de forma completa el estado final del sistema, sus decisiones técnicas y su coherencia con los objetivos de negocio y académicos del proyecto.

2. Introducción

2.1. Propósito

El presente documento constituye el Informe Final del proyecto “Módulo de Reportes Contraplano (MRC)”, desarrollado en el marco de la asignatura Capstone. Su propósito es consolidar el estado real y final del sistema, describiendo: los requisitos funcionales implementados, el conjunto final de casos de uso vigentes, las funcionalidades descartadas durante el desarrollo, la integración técnica con WordPress mediante un plugin propio, la arquitectura y decisiones de diseño, el proceso iterativo aplicado mediante Scrumban, y la trazabilidad entre requerimientos, pruebas y objetivos del proyecto.

El documento integra la Especificación de Requerimientos de Software (ERS) actualizada al cierre del proyecto, junto con los resultados obtenidos, asegurando una visión clara, verificable y alineada con las necesidades de la empresa Contraplano.

2.2. Alcance del producto

El MRC se integra al flujo editorial de WordPress y permite a los practicantes generar reportes noticiosos validados y clasificados. El alcance final del sistema —acotado de acuerdo con las prioridades reales de Contraplano— incluye:

● **Generación de reportes validada (CU-01):** campos obligatorios, normalización de URL, deduplicación y vista previa en HTML y texto plano.

● **Listado administrativo (CU-02):** consulta, filtros, vista previa, eliminación (soft y definitiva), restauración y paginación.

● **Gestión de bloques (CU-04):** catálogo oficial administrado por el rol Admin y opción "+Otro" para bloques temporales por reporte.

● **Bitácora automática (CU-05):** registro interno de todas las acciones críticas (crear, eliminar, restaurar, borrar definitivamente).

● **Integración con WordPress:** identificación del usuario, control de acceso y apertura del módulo mediante plugin propio.

● **Retención operativa de 90 días:** los reportes expiran automáticamente para mantener una base activa ligera.

● **Despliegue en Firebase Hosting + Firestore:** base de datos en la nube, seguridad, cifrado y operación sin servidores.

Funciones excluidas en el alcance final: dashboard analítico, exportes editoriales (newsletter, revista mensual), informe anual y procesos de archivado extendido.

2.3. Público objetivo

El sistema está dirigido a los siguientes perfiles de la organización:

- **Practicantes / Reporteros:** registran reportes diarios desde WordPress, sin iniciar sesión adicional.
- **Administradores:** gestionan reportes, aplican filtros, eliminan/restauran elementos y mantienen el catálogo de bloques.
- **Supervisión / Control de calidad:** revisan la coherencia clasificatoria y la auditoría registrada en la bitácora.
- **Dirección:** puede solicitar revisiones, auditorías o análisis sobre reportes, aunque el sistema final no incluye un dashboard visual.

2.4. Uso previsto

El MRC está diseñado para apoyar el flujo editorial diario de Contraplano, permitiendo:

- Registro eficiente y validado de reportes noticiosos.
- Deduplicación inmediata mediante hash de URL canónica.
- Clasificación por bloques oficiales o temporales.
- Administración y mantenimiento de reportes por el rol Admin.
- Recuperación o eliminación definitiva de reportes según ciclo editorial.
- Auditoría automática de todas las acciones relevantes.
- Integración fluida con WordPress mediante un plugin propio.

Se excluyen en la versión final funcionalidades de dashboard, newsletters, revistas mensuales y procesos anuales de archivado.

2.5. Referencias

El presente documento se alinea con:

- La Guía del Estudiante – Definición Proyecto APT (documento de inicio del proyecto).
- El manual de casos de uso de Ivar Jacobson (Use Case 2.0), como base metodológica para la definición de los casos de uso.
- Las mejores prácticas de gestión de proyectos recomendadas por el PMBOK (PMI), especialmente en lo referido a gestión del alcance, calidad y comunicación.
- Las metodologías ágiles y el enfoque Scrumban, que guían la organización del trabajo, la priorización iterativa de funcionalidades y la entrega incremental de valor.

3. Objetivos del Proyecto

3.1. Objetivo General

Diseñar e implementar un módulo de reportes para Contraplano que optimice la gestión, validación y control de reportes periodísticos, asegurando datos confiables, exportables y compatibles con los flujos actuales.

3.2. Objetivos específicos

1. Relevar y consolidar los requerimientos vigentes del módulo, estableciendo línea base y control de cambios.
2. Diseñar y desarrollar el formulario de generación de reportes con validaciones avanzadas y deduplicación por URL canónica (CU-01).
3. Implementar el listado administrativo de reportes con filtros, orden, paginación, selección masiva y gestión de eliminación/restauración (CU-02).
4. Configurar el campo de “bloques” desde una interfaz de administración (CU-04).

5. Establecer autenticación y roles mínimos para el acceso administrativo y resguardo de operaciones.
6. Asegurar la calidad mediante pruebas, bitácora de acciones, políticas de seguridad y retención de datos.
7. Desplegar y operar el módulo en Firebase Hosting, integrándose a WordPress mediante un plugin propio para su uso por practicantes y editores.

4. Requisitos funcionales

ID	Nombre	Descripción	Caso de uso asociado
RF-01	Generación de reportes	El sistema debe permitir la creación de reportes validando todos los campos obligatorios, normalizando la URL canónica y bloqueando la creación de reportes duplicados mediante hash.	CU-01
RF-02	Listar reportes	El sistema debe mostrar listados de reportes y permitir descargar.	CU-02
RF-04	Configurar bloque	El administrador debe poder agregar, editar o eliminar bloques oficiales.	CU-04
RF-05	Bitácora automática de acciones	Toda operación crítica (crear, eliminar, restaurar, hard delete) debe registrarse automáticamente en la colección `audit_logs`.	CU-05
RF-09	Control de acceso mediante modo WordPress	El sistema debe diferenciar los modos `admin` y `reporter` para habilitar o restringir funciones según rol.	Implícito
RF-10	Integración con WordPress	La app flutter web se integra a wordpress mediante un plugin que actúa de puente. Wordpress envía un mensaje firmado a firebase y este lo lee internamente para permitir el acceso, de lo contrario no permite acceso	Implícito / CU-01
RF-15	Vista previa del reporte	El sistema debe generar una vista previa del reporte en formato HTML y texto plano, visible antes de confirmar la creación.	CU-01 / CU-02
RF-16	Carga automática de autor desde WordPress	El sistema debe recibir el usuario WordPress autenticado mediante token firmado y bloquear el campo “Autor”.	CU-01
RF-17	Vista previa HTML desde el listado	El sistema debe mostrar la previsualización completa del reporte con un clic desde el listado administrativo.	CU-02
RF-18	Eliminación suave (soft delete)	El administrador debe poder enviar reportes a papelera, donde permanecerán disponibles para restauración.	CU-02
RF-19	Restauración de reportes	El administrador debe poder restaurar reportes anteriormente eliminados.	CU-02

RF-20	Eliminación definitiva (hard delete)	El administrador debe poder eliminar un reporte de forma permanente, eliminando su registro de Firestore.	CU-02
RF-21	Selección de bloque temporal por reporte (“+Otro”)	El sistema debe permitir que el reportero añada un bloque temporal solo para el reporte en curso si el bloque requerido no aparece en el listado oficial.	CU-01
RF-22	Búsqueda de bloques	El sistema debe incluir una barra de búsqueda dentro del selector de bloques para facilitar el uso en listas extensas.	CU-01 CU-04
RF-23	Búsqueda de reportes	El sistema debe permitir buscar reportes desde el listado mediante campo de texto (títulos o palabras clave relevantes).	CU-02
RF-24	Retención operativa de reportes	El sistema debe permitir la operación con reportes activos dentro de un máximo de 90 días.	CU-02
RF-25	Mostrar vistas de estado	El sistema debe mostrar claramente si un reporte está: • activo, • eliminado (papelera), • o eliminado definitivamente.	CU-02

Requisitos funcionales descartados

ID	Nombre	Descripción	Caso de uso asociado
RF-03	Consultar dashboard	El sistema debe mostrar métricas interactivas (totales, duplicados, evolución temporal, top autores, distribución por bloque) con posibilidad de exportar CSV/PNG.	CU-03
RF-06	Exportar newsletter semanal	El sistema debe generar un archivo HTML (y opcional CSV) con los reportes válidos de una semana, aplicando filtros y usando la plantilla activa o por defecto.	CU-06
RF-07	Exportar revista mensual	El sistema debe generar un archivo HTML (y opcional CSV) con los reportes válidos de un mes, agrupados por bloque y orden editorial simple.	CU-07
RF-08	Archivado anual e informe	El sistema debe generar un Informe Anual (gráficos y tablas), exportar los reportes del año anterior a almacenamiento económico, verificar integridad y limpiar base activa.	CU-08
RF-11	Bitácora obligatoria	Todas las operaciones críticas (crear, eliminar, exportar, configurar, archivar) deben registrarse en la bitácora con usuario, fecha, acción y parámetros clave.	Transversal
RF-12	Manejo de duplicados	El sistema debe impedir la creación de reportes con URL duplicada (hash coincidente) y permitir su eliminación	CU-01 / CU-02

		como duplicados, afectando métricas y listados.	
RF-13	Exportación CSV	Además de HTML, el sistema debe permitir exportar resultados filtrados a formato CSV para control y análisis externo.	CU-02
RF-14	Informe anual de métricas	El sistema debe generar automáticamente un informe consolidado del año anterior (totales, bloques, autores, duplicados, fuentes) antes del archivado.	CU-08

5. Requisitos de la interfaz externa

5.1. Interfaz de usuarios

El sistema MRC se ejecuta como una aplicación web (Flutter Web) embebida en WordPress mediante un plugin propio. Las interfaces expuestas al usuario final se dividen en dos modos principales: **Reportero** y **Administrador**, determinados por el token enviado desde WordPress.

5.1.1. Interfaz para Reporteros (mode=reporter)

El reportero accede al módulo desde WordPress sin autenticación adicional. La interfaz consiste en:

- **Formulario de generación de reportes (CU-01):**
  - Campos obligatorios y validados (Título, Fecha, Bloque, Fuente, URL noticia).
  - Campo Autor prellenado y bloqueado (desde WordPress).
  - Vista previa HTML y texto plano.
  - Selector de Bloques con barra de búsqueda.
  - Opción “+Otro” para crear un bloque temporal por reporte.
  - Botones: *Generar*, *Limpiar*, *Copiar HTML*, *Descargar TXT*.
  - Mensajes dinámicos de validación y deduplicación.
- **Alertas y mensajes:**
  - Confirmación de generación.
  - Advertencia de duplicados (hash coincidente).
  - Campos requeridos no completados.
  - Error de conectividad o validación.

5.1.2. Interfaz para Administradores (mode=admin)

El administrador accede al MRC desde WordPress con privilegios extendidos para gestión operativa.

La interfaz incluye:

- **Listado de reportes (CU-02):**
  - Tabla con: Título, Autor, Fecha, Bloque, Estado, Acciones.
  - Filtros por texto, bloque, estado y rango de fechas.
  - Paginación y ordenamiento dinámico.
  - Vista previa HTML con diseño mejorado.
  - Acciones: *Restaurar*, *Eliminar*, *Eliminar definitivamente*.
- **Gestión de Bloques (CU-04):**
  - Agregar bloques oficiales.
  - Renombrar y desactivar bloques.
  - Persistencia en config/categories.
  - Vista ligera, enfocada sólo en catálogo oficial.
- **Mensajes administrativos:**
  - “Reporte enviado a papelera.”
  - “Reporte restaurado.”
  - “Bloque actualizado.”
  - “Eliminación definitiva realizada.”

5.2. Interfaz de hardware

El MRC no depende de hardware especializado. El sistema opera completamente en la nube.

Requisitos mínimos:

- **Cliente (usuario):**
  - Cualquier equipo capaz de ejecutar un navegador moderno (Chromium/Chrome/Edge/Firefox).
  - Conectividad estable a Internet.
- **Servidor / Plataforma:**
  - Google Firebase Hosting (infraestructura administrada).
  - Firestore (base de datos NoSQL).
  - No requiere servidores propios ni hardware de Contraplano.

5.3. Interfaz de software

El sistema interactúa con los siguientes componentes:

5.3.1. WordPress (plugin MRC)

- Envía token firmado con: uid, display_name, role, mode.
- Controla acceso al módulo según rol.
- Incrusta el módulo MRC mediante iframe o página dedicada.

5.3.2. Firebase Firestore

Colecciones utilizadas:

- reports (reportes)
- audit_logs (bitácora)
- config/categories (bloques oficiales)
- sessions (registro ilustrativo de acceso por modo)

5.3.3. Flutter Web

- UI compilada a JavaScript/HTML.
- Manejo del estado del formulario, vistas, validación y comunicación con Firestore.

5.3.4. APIs internas del módulo

- Módulo de normalización de URL.
- Generador de hash.
- Renderizador de vista previa HTML.
- Controlador de deduplicación.

5.4. Interfaz de comunicación

5.4.1. Comunicación WordPress → MRC

- Método: Querystring seguro o postMessage.
- Contenido: token firmado + rol + modo.
- Protocolo: HTTPS obligatorio.

5.4.2. Comunicación MRC → Firebase

- **Protocolo:** HTTPS (REST implícito y SDK oficial de Firebase).
- **Operaciones:**
  - get, set, update, delete en Firestore.
  - No se utilizan websockets ni triggers de Cloud Functions.

5.4.3. Comunicación interna

- La app no expone endpoints propios.
- No interactúa con servidores de Contraplano.

- Toda la comunicación con Firestore usa credenciales del cliente configuradas en el SDK.

6. Requisitos no funcionales

Los siguientes requisitos no funcionales aseguran que el Módulo de Reportes Contraplano (MRC) opere con niveles adecuados de rendimiento, seguridad, mantenibilidad y confiabilidad. Todos los requisitos están alineados con el alcance final del sistema y se basan en la operación real sobre Firebase Hosting + Firestore e integración con WordPress.

6.1. Rendimiento y disponibilidad

- **RNF-01 – Tiempo de carga inicial:** El módulo debe cargar completamente en ≤ 3 segundos en una conexión promedio (30–50 Mbps) al ejecutarse desde WordPress.
- **RNF-02 – Validación del formulario (CU-01):** Las validaciones locales del formulario, incluida la normalización y cálculo de hash, deben ejecutarse en ≤ 0,5 segundos.
- **RNF-03 – Consultas a Firestore (CU-02):** Los listados y filtros deben responder en ≤ 1,5 segundos, considerando paginación nativa de Firestore.
- **RNF-04 – Operación continua:** El sistema debe estar disponible al menos 99,5% del tiempo, sujeto a la disponibilidad de Firebase Hosting.
- **RNF-05 – Vista previa HTML:** La generación de vista previa debe completarse en ≤ 0,8 segundos.

6.2. Seguridad

- **RNF-06 – Acceso mediante WordPress:** El MRC debe operar únicamente si recibe un token firmado emitido por WordPress identificando: uid, nombre, rol, modo.
- **RNF-07 – Roles diferenciados:** El modo *reporter* no puede acceder a funciones de administración (CU-02, CU-04, CU-05).
- **RNF-08 – Comunicaciones seguras:** Toda comunicación debe realizarse por HTTPS. No se permite tráfico plano o HTTP.
- **RNF-09 – Datos mínimos:** Solo se almacenan datos mínimos necesarios: nombre del autor, bloque, URL, contenido del reporte y timestamps.

- **RNF-10 – Auditoría (CU-05):** Toda acción crítica debe ser registrada en audit_logs con: usuario, acción, fecha, id de reporte.

6.3. Usabilidad y estética

- **RNF-11 – Interfaz consistente:** La UI debe mantener un diseño coherente, responsivo y minimalista, con estilos uniformes entre modos *admin* y *reporter*.
- **RNF-12 – Mensajes claros:** El sistema debe mostrar mensajes explícitos para errores, validaciones, duplicados, permisos y confirmaciones.
- **RNF-13 – Búsqueda eficiente:** El buscador de bloques y el buscador del listado administrativo deben ser accesibles y con retroalimentación inmediata.
- **RNF-14 – Flujo sin fricción:** El usuario debe poder generar un reporte con un máximo de 3 interacciones principales: completar campos → vista previa → generar.

6.4. Escalabilidad y capacidad

- **RNF-15 – Escalabilidad horizontal:** Firestore debe soportar crecimiento en cantidad de reportes sin necesidad de reconfiguración de infraestructura.
- **RNF-16 – Operación con cientos de reportes activos:** El sistema debe operar fluidamente con al menos 300–500 reportes activos dentro del rango de 90 días.
- **RNF-17 – Paginación obligatoria:** Las listas deben paginarse automáticamente para evitar descargas excesivas de documentos.

6.5. Mantenibilidad y extensibilidad

- **RNF-18 – Código modular:** La aplicación Flutter debe mantenerse modular (controladores, widgets, servicios, utilidades).
- **RNF-19 – Independencia del plugin:** El plugin de WordPress debe ser lo más simple posible y no contener lógica propia; la lógica debe residir en el MRC.
- **RNF-20 – Extensibilidad en categorías**: El diseño debe permitir agregar o desactivar bloques sin afectar reportes históricos.

6.6. Portabilidad y compatibilidad

- **RNF-21 – Navegadores compatibles:** Compatible con: Chrome, Firefox, Edge y Safari (versiones modernas).

- **RNF-22 – Portabilidad web:** Debe ejecutarse sin instalar software adicional, solo mediante navegador.
- **RNF-23 – Independencia de sistema operativo:** Compatible con Windows, macOS, Linux y dispositivos móviles (visualización básica).

6.7. Operación y respaldo

- **RNF-24 – Retención de 90 días:** Los reportes deben mantenerse en Firestore hasta expirar naturalmente según las políticas de la empresa.
- **RNF-25 – Recuperación:** El administrador debe poder restaurar reportes eliminados sin pérdida de información.
- **RNF-26 – Respaldos automáticos de Firestore:** Se utilizarán los respaldos automáticos nativos de Firebase (controlados por Google Cloud).

6.8. Restricciones y lineamientos

- **RNF-27 – No se permiten accesos directos:** El MRC no puede abrirse sin venir desde WordPress con token válido.
- **RNF-28 – Limitaciones del navegador:** La app no tiene acceso a archivos locales salvo descarga de .txt.
- **RNF-29 – Límite de escritura Firestore:** Las operaciones deben cumplir las cuotas de lectura/escritura por minuto de Firestore.

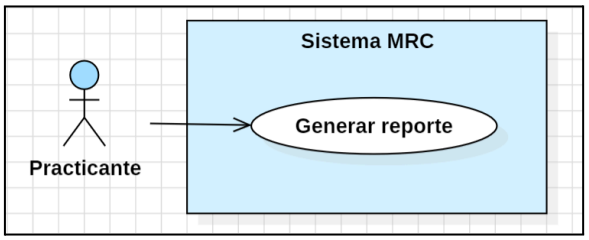
Requisitos No Funcionales Descargados (informativos)

- RNF-D01 – Generación de datos agregados para dashboard: Descartado porque CU-03 fue removido y Contraplano mueve estos análisis a WordPress.
- RNF-D02 – Exportación de newsletter semanal (CU-06): Requería plantillas y motor de exportación no compatibles con la arquitectura final.
- RNF-D03 – Exportación de revista mensual (CU-07): Descartado al no aportar valor dentro del flujo del MRC.
- RNF-D04 – Archivado anual y limpieza avanzada: Incompatible con la retención real de 90 días.
- RNF-D05 – Versionado completo del formulario: Se simplificó la configuración a solo gestión de bloques (CU-04).

7. Casos de uso

CU-01 Generar reporte

Este caso de uso sufrió algunos ajustes en su alcance funcional durante las fases 2 y 3, pero persiste como el eje central del sistema y se encuentra completamente implementado en la versión final. La principal actualización es que el acceso ya no se realiza mediante un token JWT firmado —como se planteó en el ERS inicial— sino mediante el plugin de WordPress y un mecanismo de sesión validado desde Firestore. Todas las demás funciones esenciales (validaciones, deduplicación, vista previa y bitácora) se mantienen vigentes.



Actor principal: Reportero (Practicante con sesión activa en WordPress)

Interesados: Administración/Editor (define bloques y supervisa uso), Supervisión (revisión general)

Disparador: El usuario ingresa al panel de WordPress y hace clic en el elemento del menú “MRC Reportes”.

- Si es **Admin**, ingresa al modo administrador.
- Si es **Editor/Reportero**, ingresa al modo reportero.

Meta/Descripción: Registrar un reporte válido utilizando la configuración activa del formulario, validando datos, generando la vista previa y entregando HTML y texto plano listos para ser copiados o exportados.

Alcance: Aplicación web MRC (Flutter Web) desplegada sobre Firebase, lanzada exclusivamente desde WordPress mediante un plugin que provee el modo de acceso y la sesión activa del usuario.

Supuestos de acceso

- El MRC solo puede abrirse desde WordPress mediante el plugin.
- La aplicación recibe parámetros (mode, name, session) para identificar al usuario y registrar la sesión en Firestore.
- El campo Autor se prellena y bloquea según el usuario identificado por WordPress.

- No existe acceso directo público a la URL del módulo.

Precondiciones

1) Usuario autenticado en WordPress con un rol permitido (admin o reporter).
2) Existe una sesión activa o se genera una nueva al cargar el módulo.
3) Existe una configuración activa de bloques y campos.
4) Conectividad a Firestore disponible.

Postcondiciones (éxito)

1. El reporte se almacena como documento válido en Firestore, con:
    - estado "valid",
    - canonical_url y canonical_hash,
    - created_by,
    - mode del usuario,
    - expires_at (TTL aprox. 90 días).
2. Se genera bitácora (audit_logs) con la acción report.create.
3. HTML y texto plano quedan disponibles para copiar o descargar como .txt.
4. El formulario queda limpio tras el guardado.

Postcondiciones (fallo)

- No se persiste información.
- Se muestra un mensaje claro al usuario.
- Se registra un evento de error en bitácora si corresponde.

Reglas de negocio (RB)

- ☐ **RB1. Validaciones estrictas:**
    - Título, Bloque, URL de noticia, Fecha y Fuente son obligatorios.
    - La URL de noticia debe ser HTTPS y validarse contra dominio permitido.
- ☐ **RB2. Deduplicación real:**
    - Se normaliza la URL (canonicalización).

- Se calcula canonical_hash.
- Si ya existe un reporte con el mismo hash → bloqueo.

☐ **RB3. Bitácora obligatoria:**
Cada operación de creación genera un evento con actor, modo, ID del reporte y resultado.

☐ **RB4. Campos preestablecidos por WP:**
El campo Autor no puede modificarse manualmente por el reportero.

☐ **RB5. Generación de contenido:**
Se construyen versiones HTML y texto del reporte, listas para uso editorial.

Flujo básico

1. El usuario hace clic en "MRC Reportes" dentro de WordPress.
2. El módulo valida la sesión recibida y determina si el usuario es admin o reportero.
3. Se carga el formulario con:
   - Autor prellenado
   - Bloques disponibles
   - Configuración activa
4. El usuario ingresa los campos requeridos.
5. El sistema valida cada campo según las reglas editoriales.
6. El sistema canonicaliza la URL y calcula el hash.
7. El sistema detecta duplicados (si el hash existe → error).
8. Se muestra la vista previa en HTML y texto.
9. El usuario confirma "Generar".
10. El sistema guarda el documento en Firestore.
11. Se registra la acción en audit_logs.
12. Se limpia el formulario.

Flujos alternativos

- **A1 – Sesión inválida:**
  Rechazo inmediato y mensaje "No pudimos verificar tu sesión en WordPress".

- **A2 – Validación fallida:**
  Resaltar el campo con el mensaje correspondiente.
- **A3 – URL duplicada:**
  Mensaje “La URL ya fue reportada”.
- **A4 – Error de red / Firestore:**
  Mensaje “Intenta nuevamente”.

Mensajes de interfaz (reales del sistema)

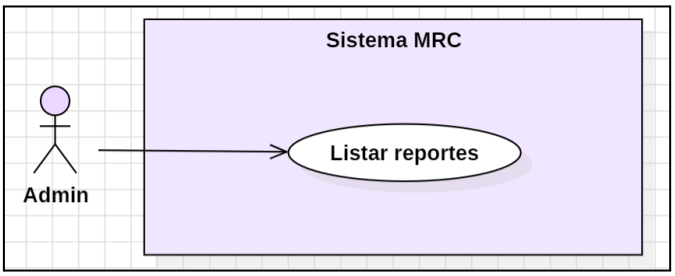
- “Reporte generado.”
- “Formato de URL no válido.”
- “La URL ya fue reportada.”
- “Autor cargado desde WordPress: {display_name}.”
- “No pudimos verificar tu sesión.”

Criterios de aceptación

- Campos requeridos validan correctamente.
- URL canonicalizada y deduplicada.
- Se genera HTML y texto plano coherente.
- Registro en audit_logs con acción y usuario.
- Reporte disponible en listado con estado “valid”.

CU-02 Listar y administrar reportes

Este caso de uso sufrió varios ajustes durante las fases 2 y 3 del proyecto. En la versión inicial del ERS incluía acciones avanzadas como “marcar duplicados”, “exportaciones editoriales (newsletter/revista)”, “CSV”, y “sin restauración”. Sin embargo, por cambios de alcance solicitados por la empresa y alineación con el flujo real de trabajo, el alcance final se consolidó en un módulo de listado con filtros, paginación, vista previa, papelera, restauración, eliminación definitiva y bitácora. Este caso de uso persistió hasta el final y fue implementado completamente.



Actor principal: Admin

Interesados: Supervisión (control de calidad), Practicantes/Reporteros (cuyos reportes pueden verse afectados)

Disparador: El Admin o editor (reportero) accede al módulo desde WordPress, haciendo clic en “MRC Reportes” y eligiendo el ícono “Ver reportes” dentro del formulario en el sistema MRC.

Meta/Descripción: Permitir que el administrador consulte el historial de reportes, aplique filtros, revise detalles, restaure reportes eliminados, elimine definitivamente registros, y consulte acciones previas, todo con registro automático en la bitácora. O también, que el editor acceda y pueda realizar busquedas en los filtros disponibles pero sin el permiso de eliminar o restaurar reportes.

Alcance: Aplicación web MRC (Flutter Web) integrada vía plugin de WordPress.

Supuestos de acceso:

- El usuario accede con rol Admin, determinado por el plugin de WordPress mediante el parámetro mode=admin.
- La sesión está validada por Firestore.
- Se muestran todos los reportes disponibles dentro del TTL (90 días antes de eliminación automática).

Precondiciones

1. El usuario es Admin y tiene permiso para ver y administrar reportes.
2. Existen reportes almacenados en la colección reports.
3. Conexión activa con Firestore.
4. Bitácora operativa (audit_logs).

Postcondiciones (éxito)

1. Los reportes se visualizan según los filtros aplicados.

2. Acciones administrativas se reflejan en Firestore:
    - **soft delete** → el reporte cambia su estado a "removed".
    - **restore** → el reporte vuelve a "valid".
    - **hard delete** → el documento se elimina definitivamente.
3. Toda acción queda registrada con detalles en audit_logs.
4. La vista se actualiza para reflejar el cambio.

Postcondiciones (fallo)

- No se modifica el estado de los reportes.
- Se muestra un mensaje de error acorde.
- Se registra un evento de error en audit_logs cuando corresponde.

Reglas de negocio (RB)

- RB1. Permisos: Solo Admin puede realizar acciones administrativas (eliminar, restaurar, eliminar definitivamente).
- RB2. Soft delete (papelera):
    - Cambia state a "removed".
    - Oculta el reporte del listado estándar.
    - Puede restaurarse mientras no venza su TTL.
- RB3. Restauración: Solo posible para reportes con state="removed" y aún dentro del TTL.
- RB4. Hard delete:
    - Elimina el documento permanentemente.
    - Acción irreversible, registrada en bitácora.
- RB5. Filtros:
    - Bloque
    - Fechas (rango)
    - Autor
    - Estado (valid, removed)

    - Texto libre (buscador)
    - Rank math

- RB6. Paginación y rendimiento: Las consultas deben mantenerse dentro de performance razonable según Firestore.
- RB7. Bitácora obligatoria: Toda acción administrativa debe producir una entrada en audit_logs con: actor, acción, id del reporte, timestamp, motivo opcional.
- RB8. Vista previa: El Admin puede abrir el detalle del reporte desde la lista sin modificarlo.

Flujo básico

1. El Admin abre la sección Listar reportes mediante el icono de Ver reportes dentro del MRC.
2. El sistema recupera los reportes válidos (state="valid") ordenados por fecha.
3. El Admin aplica filtros opcionales (bloque, fecha, autor, estado, texto, rank math).
4. El sistema actualiza la lista según filtros vigentes.
5. El Admin selecciona un reporte o varios.
6. El Admin elige una acción:
    - Eliminar (enviar a papelera soft delete)
    - Descargar
7. El sistema ejecuta la acción seleccionada.
8. El sistema registra la acción en audit_logs.
9. El listado se actualiza.

Flujos alternativos / Excepciones

1. Admin aplica filtro de estado a eliminados
2. Admin selecciona uno o más reportes de la papelera y elige una acción entre:
    - Restaurar reportes, cambiando de estado a válido
    - Eliminar definitivamente desde la base de datos
3. La vista se actualiza automáticamente según la acción elegida

A1 – Sin resultados

- El sistema muestra “No se encontraron reportes con estos filtros”.

A2 – Falta de permisos

- Si el usuario no es Admin (modo reporter), las acciones administrativas se ocultan.

A3 – Restauración inválida

- Si el reporte ya expiró (TTL), no puede restaurarse y el sistema notifica.

A4 – Error en Firestore

- Si ocurre un error al actualizar/borrar:
  - se notifica al usuario,
  - no se cambia el estado del reporte,
  - se registra en bitácora un evento report.error.

Mensajes de interfaz (reales del sistema)

- “Reporte eliminado (enviado a papelera).”
- “Reporte restaurado.”
- “Reporte eliminado definitivamente.”
- “No se encontraron reportes.”
- “Ocurrió un error. Intenta nuevamente.”

Criterios de aceptación

- Los filtros se aplican correctamente.
- Vista previa muestra datos reales del reporte.
- Soft delete cambia el estado a "removed".
- Restaurar vuelve el estado a "valid".
- Hard delete borra el documento.
- Cada acción genera un registro correcto en audit_logs.

Porciones (slices)

- P1: Listado + filtros básicos + paginación.
- Soft delete + restauración + hard delete.
- P3: Integración con bitácora.
- P4: Vista previa desde la tabla.
- P5: Filtros avanzados (texto, estado, etc.).

CU-03 Consultar dashboard

Este caso de uso fue finalmente descartado del alcance del proyecto.

Aunque en el ERS inicial se planteó la creación de un dashboard analítico con métricas (totales, duplicados, top practicantes, reportes por bloque, evolución temporal, exportaciones, etc.), durante las fases 2 y 3 la empresa Contraplano decidió concentrar la analítica y la visualización de estadísticas directamente en su entorno de WordPress, donde otro practicante estaba construyendo un módulo adicional de reporting con datos editoriales más amplios. Para evitar duplicidad de esfuerzos, asegurar alineación con el flujo editorial real y cumplir los plazos de desarrollo, este caso de uso fue removido del alcance del MRC. En consecuencia, el módulo final se enfocó en fortalecer la generación, administración, deduplicación y auditoría de reportes, manteniendo trazabilidad completa pero sin una vista de dashboard interno.

CU-04 Configurar bloque

Este caso de uso persistió hasta el final del proyecto, aunque con un alcance acotado respecto a la versión original del ERS.

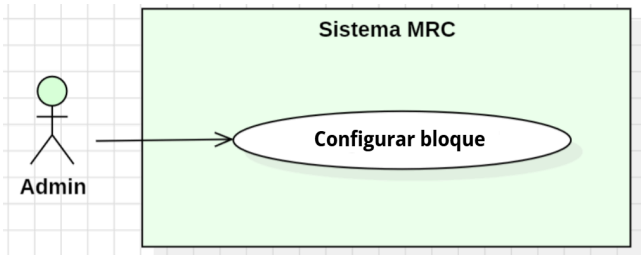
En un inicio, CU-04 contemplaba un sistema avanzado de “versionado de formulario”, con borradores, publicación, rollback, previsualización completa, reglas complejas, validaciones, y edición detallada de todos los campos del formulario. Durante el desarrollo del proyecto (Fase 2 y Fase 3), Contraplano solicitó simplificar este módulo y concentrarlo únicamente en gestionar los Bloques que alimentan el formulario de reportes.

Además, durante la implementación se incorporó una funcionalidad adicional —no contemplada originalmente— que permite a los reporteros editar temporalmente el Bloque por reporte, mediante un mecanismo de “+Otro” que garantiza continuidad operativa cuando el Administrador no está disponible para actualizar el catálogo oficial.

El resultado final es un CU estable, útil y totalmente implementado:

- El Administrador gestiona la lista oficial de Bloques desde el Modo Admin.

- El Reportero puede seleccionar un bloque existente o crear un bloque temporal solo para el reporte actual (+Otro).
- Los cambios administrativos se almacenan en config/categories.
- Los bloques temporales solo se guardan dentro del documento del reporte (colección reports).



Actor principal: Admin (gestiona la lista oficial de bloques)

Actor secundario: Reportero, cuando usa la opción "+Otro" para ingresar un bloque temporal por reporte.

Interesados: Administración / Editor (lineamientos editoriales), Practicantes/Reporteros (consumen la lista de bloques), Supervisión (consistencia de clasificación por bloque)

Disparador:

- El Administrador ingresa al modo Admin del módulo MRC y selecciona Configurar Bloques.
- El Reportero, durante la generación de un reporte, selecciona la opción "+Otro" en el selector de Bloque.

Meta/Descripción: Permitir la mantención del catálogo editorial de Bloques y, adicionalmente, permitir que el reportero defina un bloque temporal en caso de que el que necesita no esté disponible en el catálogo oficial.

Alcance: Aplicación web MRC (Flutter Web), integrada vía plugin de WordPress y respaldada por Firestore.

Supuestos de acceso

- El Administrador accede con mode=admin y tiene permisos para modificar configuraciones.
- El Reportero accede con mode=reporter, puede seleccionar bloques oficiales y, si corresponde, crear un bloque temporal solo para el reporte en curso.
- El catálogo permanente se encuentra en el documento config/categories.

- Los bloques temporales se guardan únicamente dentro del documento del reporte.

Precondiciones

1. El usuario Admin está correctamente autenticado y en modo de configuración.
2. Existe un documento de configuración activo (config/categories).
3. El sistema tiene conectividad con Firestore.

Postcondiciones (éxito)

Para el Administrador

1. Los cambios en el catálogo oficial de Bloques se persisten en config/categories.
2. Otros módulos (especialmente CU-01) utilizan inmediatamente la lista actualizada.

Para el Reportero

1. El bloque temporal ingresado se guarda dentro del reporte actual.
2. No altera el catálogo oficial.
3. Es visible únicamente en el historial de reportes y solo dentro del reporte donde se utilizó.

Postcondiciones (fallo)

- No se actualiza el catálogo oficial si Firestore falla.
- Se muestra un mensaje de error.
- No se modifica el reporte si el usuario cancela el flujo de bloque temporal.

Reglas de negocio (RB)

RB1 — Catalogación oficial (Admin)

- Sólo el Administrador puede crear, renombrar, eliminar o desactivar Bloques permanentes.
- Estos bloques alimentan siempre el formulario de CU-01.

RB2 — Bloque temporal por reporte (Reportero)

- El Reportero puede crear un Bloque temporal usando “+Otro” cuando el bloque que necesita no existe.
- Este Bloque temporal:

- emerge de un cuadro flotante,
- requiere nombre válido,
- se usa solo en el reporte actual,
- se almacena en el documento del reporte,
- no modifica la lista oficial.

RB3 — Integridad con CU-01

- Los bloques oficiales siempre aparecen en el selector de CU-01.
- Los bloques temporales no aparecen en el listado futuro para otros reportes.

RB4 — Búsqueda

- El selector de Bloques permite búsqueda por texto para facilitar la selección en listas extensas.

RB5 — Persistencia

- Los cambios administrativos se guardan en config/categories.
- Los bloques temporales se guardan en el documento reports/{id}.

Flujo básico (Administrador)

1. El Admin abre el módulo MRC desde WordPress.
2. El sistema detecta mode=admin y despliega el menú de Configuración.
3. El Admin ingresa a Configurar Bloques.
4. El sistema muestra el listado oficial de Bloques.
5. El Admin agrega, edita o elimina uno o más bloques, y el sistema actualiza los cambios en tiempo real y Firestore actualiza config/categories.

Flujo básico (Reportero — Bloque temporal)

1. El Reportero abre el módulo MRC desde WordPress.
2. En el selector de Bloques, busca una categoría.
3. Si no existe o no aparece, el Reportero selecciona +Otro.
4. Se abre una ventana flotante solicitando el nombre del bloque temporal.
5. El usuario ingresa el nombre y confirma “Usar”.

6. El bloque temporal se asigna al reporte.
7. El reporte se guarda con ese bloque dentro del documento del reporte.
8. El sistema continúa el flujo normal de CU-01.

Flujos alternativos / Excepciones

A1 — Nombre inválido en Bloque temporal

- La ventana no registra campos vacíos, signos, números sin letras, y la única forma de salir es a través de la opción cancelar.

A2 — Cancelación del Bloque temporal

- Al presionar “Cancelar”, no se registra el bloque temporal.
- Se mantiene la selección anterior.

Criterios de aceptación

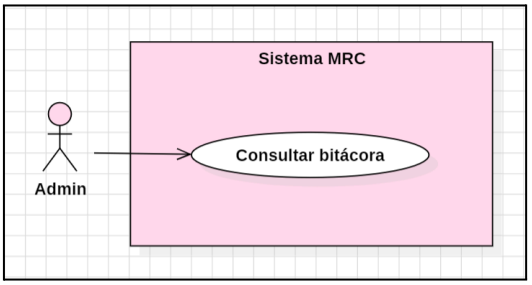
- Los cambios administrativos actualizan config/categories.
- El selector de Bloques refleja la lista oficial.
- La búsqueda por texto funciona correctamente.
- El bloque temporal se aplica sólo al reporte actual.
- El bloque temporal se almacena en Firestore dentro del documento del reporte y no aparece como categoría oficial.
- El flujo “+Otro” permite cancelar sin efectos secundarios.

CU-05 Consultar bitácora

Este caso de uso persistió hasta el final del proyecto, aunque con un alcance reducido respecto al ERS inicial.

En la primera versión del ERS, CU-05 contemplaba una interfaz completa de consulta, filtros avanzados, paginación, exportación CSV y navegación detallada sobre eventos. Durante el desarrollo (Fase 2 y Fase 3) se priorizó la operación crítica del sistema, por lo que la bitácora se implementó totalmente en el backend, almacenando cada evento en Firestore, garantizando trazabilidad, auditoría y registro de acciones relevantes.

Aunque no se desarrolló una interfaz visual para consultar la bitácora, el sistema registra correctamente todos los eventos necesarios (creación, eliminaciones, restauraciones), permitiendo cumplir el propósito fundamental de gobierno, trazabilidad y seguridad.



Actor principal: Admin

Interesados: Dirección / Editor (gobernanza documental), Supervisión / QA (seguimiento de acciones y eventos), TI / Operaciones (diagnóstico, trazabilidad)

Disparador: Una acción genera un evento en la bitácora (p. ej., crear reporte, eliminar, restaurar, eliminación definitiva). En una futura iteración, el Admin podría abrir un módulo visible de auditoría, pero en la versión actual la bitácora funciona como registro interno.

Meta/Descripción: Registrar de forma automática, consistente y permanente cada acción relevante realizada en el sistema MRC, con metadatos suficientes para garantizar trazabilidad completa.

Alcance: Sistema MRC (Flutter Web) con backend Firestore.

Supuestos de acceso

- Todas las operaciones críticas deben generar un evento en la colección audit_logs.
- El sistema conoce el actor responsable (usuario de WordPress).
- Los eventos son inmutables.

Precondiciones

1. Existe conectividad con Firestore.
2. La acción a registrar (crear, eliminar, restaurar, eliminar definitivo) está correctamente autenticada.
3. El usuario ejecuta la operación desde el MRC con sesión válida.

Postcondiciones (éxito)

1. Un nuevo documento se registra en la colección audit_logs.
2. Contiene:
    - timestamp (ts)
    - usuario (id y nombre)

- acción (report.create, report.delete, report.restore, report.hardDelete, etc.)
- id del reporte afectado
- parámetros relevantes
- resultado (ok)

3. El sistema queda en estado consistente.

Postcondiciones (fallo)

- La operación original puede fallar, pero la bitácora intentará registrar un evento de error.
- El sistema muestra un mensaje claro al usuario.

Modelo de evento almacenado (simplificado y real)

- ts: Fecha/hora del evento
- user_name: Nombre del usuario de WordPress (admin o reportero)
- user_id: Identificador del usuario (si aplica)
- action: Tipo de acción (create, delete, restore, hardDelete, error)
- entity_type: Normalmente “report”
- entity_id: Id del reporte afectado
- params: Datos adicionales (motivo, bloque, nombre, etc.)
- result: ok o error

Reglas de negocio (RB)

RB1 — Inmutabilidad: Una vez registrado, un evento en audit_logs no puede modificarse ni eliminarse.

RB2 — Integralidad: Toda operación que cambia el estado de un reporte debe generar un evento.

RB3 — Usuario responsable: Debe registrarse el nombre del usuario que ejecuta la acción.

RB4 — Synchronización con CU-01 y CU-02

- CU-01 registra report.create
- CU-02 registra report.delete, report.restore, report.hardDelete

RB5 — Datos mínimos: Cada evento debe contener al menos: acción, timestamp, usuario, id de entidad.

Flujo básico

1. El usuario realiza una acción sobre un reporte (crear, eliminar, restaurar o eliminar definitivamente).
2. El módulo correspondiente construye la estructura del evento.
3. El sistema envía el evento a audit_logs.
4. Firestore confirma el registro.
5. El sistema continúa el flujo principal (por ejemplo, actualiza la lista de reportes).

Flujos alternativos

A1 — Error al guardar el evento

- El sistema captura el error.
- Registra un segundo evento report.error si es posible.
- Notifica al usuario que hubo un problema, sin alterar la consistencia del sistema.

A2 — Usuario desconocido

- Si no se pudo determinar el usuario (caso excepcional), se registra user_name: "unknown".
- Esto será eliminado al aplicar la corrección final prevista.

Mensajes de interfaz (relacionados indirectamente)

La bitácora no tiene UI propia, pero se asocia a mensajes como:

- “Reporte generado.”
- “Reporte eliminado.”
- “Reporte restaurado.”
- “Reporte eliminado definitivamente.”
- “Error: intenta nuevamente.”

Criterios de aceptación / Casos de prueba

- La creación de un reporte genera un evento correcto con report.create.

- Eliminar un reporte genera report.delete.
- Restaurar genera report.restore.
- Hard delete genera report.hardDelete.
- El evento contiene el usuario, fecha y contexto.
- Los eventos son visibles en la colección audit_logs sin alterarse.
- Cada evento se registra **antes** de actualizar la UI del usuario.

Porciones (slices)

- **P1** — Registro de creación (report.create)
- **P2** — Registro de acciones administrativas (soft delete, restore, hard delete)
- **P3** — Manejo de errores
- **P4** — Validación de consistencia de datos
- **P5 (futuro)** — UI para consulta de auditoría

CU-06 Exportar newsletter semanal

Este caso de uso fue descartado del alcance final del proyecto.

Aunque originalmente se planteó generar un HTML semanal para envío editorial, dicha funcionalidad no encajaba de forma natural en la arquitectura del MRC, ya que requería construir un motor adicional de exportación, plantillas, UI y lógica editorial ajena al flujo principal de reportes. Además, la empresa indicó que este tipo de exportes se gestionaría directamente en plataformas externas (como Mailrelay) o desde WordPress, por lo que duplicar lógica dentro del MRC no aportaba valor ni mantenibilidad. Por motivos de foco, coherencia y tiempo, este CU fue eliminado.

CU-07 Exportar revista mensual

Este caso de uso también fue descartado del alcance final del proyecto.

Su implementación implicaba crear módulos adicionales para selección masiva, agrupación por bloque, plantillas mensuales, vista previa y mecanismos complejos de exportación, lo cual excedía las responsabilidades y naturaleza del MRC, un sistema orientado

principalmente a la generación y administración de reportes individuales. Además, no existía una necesidad real por parte de Contraplano para integrar esta exportación dentro del MRC, ya que estas tareas editoriales se realizan en otros sistemas. Por ello, se descartó para mantener un diseño claro y sostenible.

CU-08 Generar informe anual

Este caso de uso fue descartado debido a limitaciones estructurales y coherencia operativa.

El MRC utiliza una base de datos con retención máxima de 90 días, orientada exclusivamente a operaciones de corto plazo. Dado que el Informe Anual requería acceso a históricos completos (12 meses), almacenamiento de grandes agregados, métricas y archivado, la función no era viable en la arquitectura final. Además, este CU estaba estrechamente vinculado al dashboard (CU-03), también descartado, y no existía un requisito real de la empresa para que el MRC asumiera este tipo de procesos de larga duración. Por ello, se eliminó del alcance.

8. Definiciones y acrónimos

Esta sección presenta los términos y abreviaturas utilizados en el Módulo de Reportes Contraplano (MRC), con el fin de asegurar una comprensión común entre todos los actores involucrados: administradores, practicantes, docentes evaluadores y la empresa Contraplano.

Definiciones

- **Módulo de Reportes Contraplano (MRC):** Aplicación web desarrollada en Flutter Web, integrada a WordPress mediante un plugin propio, que permite generar, validar, administrar y auditar reportes noticiosos.
- **Reporte:** Registro generado por un practicante desde WordPress, compuesto por título, fecha, bloque, fuente, descripción y una URL normalizada de la noticia. Puede encontrarse en estado: *activo*, *eliminado (soft delete)* o *eliminado definitivamente*.
- **Practicante/Reportero:** Usuario que ingresa al MRC desde WordPress mediante token firmado. Solo puede generar reportes (CU-01) y utilizar bloques oficiales o temporales.
- **Administrador (Admin):** Usuario con permisos para gestionar reportes (CU-02), administrar bloques oficiales (CU-04), y consultar la bitácora operativa (CU-05).
- **Supervisor/Editor:** Actor con rol de control de calidad, que revisa métricas, auditorías y coherencia de los reportes.

- **Bloque oficial:** Categoría administrada desde el panel del MRC y almacenada en `config/categories`. Ej.: Política, Economía, Deportes.
- **Bloque temporal (+Otro):** Categoría creada por el reportero solo para un reporte específico cuando no existe en el catálogo oficial. No se almacena como categoría global.
- **Soft delete (Eliminado):** Estado donde el reporte se oculta del listado principal pero puede restaurarse.
- **Hard delete (Eliminación definitiva):** Eliminación permanente del reporte desde Firestore. No puede recuperarse.
- **Bitácora:** Registro de auditoría que almacena acciones críticas: creación, eliminación, restauración y eliminación definitiva, junto al actor y fecha.
- **URL canónica:** Versión normalizada de la URL de la noticia, utilizada para evitar duplicados mediante hash.

Acrónimos

- **ATP:** Anteproyecto de Título
- **ERS:** Especificación de Requerimientos de Software
- **CU:** Caso de Uso
- **RNF:** Requisito No Funcional
- **WP:** WordPress
- **JWT:** JSON Web Token
- **CSV:** Comma-Separated Values
- **HTML:** HyperText Markup Language
- **PMBOK:** Project Management Body of Knowledge
- **PMI:** Project Management Institute
- **TTL:** Time To Live (no se usa activamente en esta versión final, pero puede mantenerse porque describe eliminaciones en sistemas modernos)
- **MRC:** Módulo de Reportes Contraplano

9. Trazabilidad y Scrumban

9.1. Propósito de la trazabilidad

La trazabilidad asegura que cada requisito, objetivo y funcionalidad implementada en el MRC tenga una correspondencia clara con sus casos de uso, requisitos funcionales y criterios de validación. Esta trazabilidad permite verificar el cumplimiento del alcance solicitado por Contraplano y documentar la evolución del proyecto a través de iteraciones Scrumban.

9.2. Matriz de trazabilidad (extracto)

Objetivo del proyecto	Requisito funcional	Caso de uso	Validación / Prueba
Estandarizar y validar la generación de reportes	RF-01, RF-02, RF-15, RF-16	CU-01	Validación de campos, hash, vista previa, creación correcta
Reducir duplicidad de noticias	RF-01, RF-23	CU-01 / CU-02	Validación de hash duplicado en CU-01 y eliminación/restauración en CU-02
Administrar reportes y estados	RF-02, RF-17, RF-18, RF-19, RF-20, RF-24, RF-25	CU-02	Soft delete, restauración, hard delete
Asegurar control editorial mediante bloques	RF-04, RF-21, RF-22	CU-04	Gestión de bloques + “+Otro”
Garantizar registros de auditoría	RF-05	CU-05	Bitácora visible, registros consistentes
Preservar seguridad e integridad del acceso	RF-09, RF-10	CU-01–CU-05	Token WordPress y bloqueo de accesos
Soporte operativo y despliegue web	RF-15–RF-26	-	Pruebas de ambiente, despliegue en Firebase

Nota: los RNF (rendimiento, seguridad, usabilidad, mantenibilidad) también se trazan transversalmente a todos los CU y pruebas.

9.3. Planificación iterativa con Scrumban

El enfoque Scrumban permitió gestionar el proyecto con un equipo reducido (2 integrantes), iterando mediante flujo continuo y pequeños incrementos funcionales:

- Backlog visual con tareas clasificadas en *Pendiente* → *En progreso* → *Terminado*.

- WIP limitado (máx. 2–3 tareas simultáneas) para evitar bloqueos.
- Revisiones semanales con Contraplano para validar avances y ajustar prioridades.
- Retrospectivas cortas para optimizar tiempos, mejorar comunicación y ajustar estimaciones.
- Entrega incremental, donde cada CU se completó como una “porción” del sistema.

9.4. Beneficios del enfoque Scrumban

- Iteraciones cortas y adaptables permitieron ajustar el alcance cuando Contraplano descartó dashboard, exportes e informe anual.
- Visibilidad constante del avance mediante tablero Kanban y bitácora.
- Reducción de riesgos gracias al desarrollo incremental y pruebas continuas.
- Mejor alineamiento con usuarios reales (practicantes y editores), quienes validaron las funciones en uso real desde WordPress.

10. Anexos

10.1. Video demostrativo del sistema MRC

10.2. Documento de pruebas del sistema

10.3. Evidencia adicional del sistema

11. Conclusiones

El desarrollo del Módulo de Reportes Contraplano (MRC) permitió implementar una solución efectiva, ligera y alineada a las necesidades operativas reales de la empresa. A través de un enfoque iterativo Scrumban y un trabajo colaborativo con Contraplano, se logró entregar un sistema funcional, seguro y totalmente integrado al flujo editorial existente en WordPress.

El proyecto cumplió su objetivo general: diseñar e implementar un módulo que optimiza la generación, validación y administración de reportes periodísticos, asegurando consistencia, deduplicación, trazabilidad y un uso intuitivo para practicantes y administradores. Asimismo, se cumplieron los objetivos específicos, desarrollando formularios validados, un listado administrativo robusto, gestión flexible de bloques, bitácora automática y un despliegue estable en Firebase.

Durante el avance del proyecto, se produjo una evolución natural del alcance. Algunos casos de uso originalmente considerados —como dashboard, newsletters, revista mensual e informe anual— fueron descartados en coordinación con la empresa, priorizando un sistema

liviano, práctico y centrado en el flujo operacional diario. Esta capacidad de adaptación fue posible gracias al uso de Scrumban, que permitió responder rápidamente a cambios sin comprometer la calidad.

El resultado final es una aplicación estable, segura y mantenible, actualmente integrada y operativa en el entorno real de Contraplano. El MRC contribuye directamente a la eficiencia editorial, mejorando la calidad de los reportes y reduciendo la duplicidad, al mismo tiempo que facilita la labor de los practicantes y proporciona trazabilidad completa para los administradores. El proyecto demuestra no solo competencias técnicas, sino también la capacidad de comprender, adaptar y entregar soluciones acordes a un entorno profesional real.