



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Diseño de Sistemas de Información 2025
Curso: K3101
Turno: Mañana

Trabajo Práctico Anual

Primera Entrega

GRUPO N°2	
Nombre y Apellido	Legajo
Dante Ezequiel Samudio	214.105-0
Genaro Melitón Clemente	213.006-3
José Leonardo Gutiérrez Zevallos	212.988-7
José Luis Becerra	204.675-1
Lucca Polastri	214.015-9

ENTREGA/REVISIÓN	1	2	3	4	5	6
FECHA ENTREGA	22/04/2025	27/05/2025	01/07/2025	09/09/2025	14/10/2025	25/11/2025
FIRMA DOCENTE						

Primera Entrega

Requerimientos de dominio

1. Como persona administradora, deseo crear una colección.
2. Como persona administradora, deseo poder importar hechos desde un archivo CSV.
3. Como persona visualizadora, deseo navegar todos los hechos disponibles de una colección.
4. Como persona visualizadora, deseo navegar los hechos disponibles de una colección, aplicando filtros.
5. Como persona contribuyente, deseo poder solicitar la eliminación de un hecho.
6. Como persona administradora, deseo poder aceptar o rechazar la solicitud de eliminación de un hecho.

Administración de Hechos

En esta etapa del diseño, se entiende a los hechos como unidades de información relevantes que deben ser almacenadas, buscadas y eventualmente compartidas. Cada hecho cuenta con un título, una descripción, una categoría, una ubicación geográfica, fechas asociadas (fecha del acontecimiento y fecha de carga), un origen que da cuenta de quién o qué lo reportó, y contenido multimedia vinculado. Además, se incluye un atributo booleano eliminado que permite representar la eliminación lógica del hecho sin quitarlo del sistema, conservando así trazabilidad.

Para modelar la procedencia de los hechos se definió la clase *Origen*, la cual contiene información básica sobre la fuente, como el nombre, apellido y edad del contribuyente (si aplicase) y el tipo de origen, representado por un enumerado *TipoOrigen*. Este enumerado permite distinguir entre hechos ingresados manualmente, importados desde datasets o reportados por contribuyentes.

El contenido multimedia se encapsula en la clase *Multimedia*, asociada a cada hecho, con un atributo formato representado por el enumerado *Formato* que contempla texto, imagen, audio o video. Esto asegura la extensibilidad para nuevos tipos de contenido en el futuro.

Administración de Colecciones y Filtros

Se decidió modelar la clase *Coleccion* para permitir a los usuarios administradores organizar hechos de acuerdo con un propósito determinado. Cada colección incluye un título, una descripción, un conjunto de hechos y un conjunto de filtros asociados que definen su criterio de selección.

La clase *Filtro* permite seleccionar hechos según tres criterios: la categoría, un rango de fechas y una zona geográfica. Para representar el área geográfica, se diseñó la clase *Zona*, que contiene un conjunto de ubicaciones y un método que permite determinar si un hecho pertenece a dicha zona. Este enfoque proporciona un criterio espacial flexible y extensible para búsquedas geográficas.

El método *cumpleFiltro* encapsula la lógica de validación para cada hecho respecto al filtro. Este diseño favorece la cohesión y la claridad, facilitando también futuras extensiones a nuevos tipos de criterios de filtrado.



Importación y Contribución de Hechos

Se reconoció la necesidad de contar con una fuente de datos externa que permita importar hechos de manera automatizada. Para ello, se introdujo la clase *ImportadorHechos*, que encapsula la lógica necesaria para la carga de hechos desde fuentes externas, utilizando el patrón Adapter a través de la interfaz *CSVReaderAdapter*. Esta decisión permite reducir el acoplamiento con los mecanismos concretos de lectura de archivos, facilitando su extensión y reutilización.

Solicitudes de Eliminación

Dado que los hechos pueden ser reportados por diferentes fuentes, se consideró necesario incluir un mecanismo formal de revisión para las solicitudes de eliminación. Se modeló la clase *Solicitud*, que contiene referencias al hecho en cuestión, un estado (pendiente, aceptada o rechazada), y datos del responsable y supervisor de la solicitud.

El método *estaFundado* evalúa si existe fundamento para proceder con la solicitud, promoviendo así una administración responsable del contenido. Esta estructura permite incorporar un flujo de revisión que podrá expandirse a otras operaciones como ediciones o validaciones de hechos en versiones futuras.

Correcciones 22/04/2025

Aplicación del patrón Strategy en los filtros

Se reemplazó la clase Filtro original por un conjunto de clases específicas que implementan la interfaz *FiltroStrategy*. Esta interfaz define el método *cumpleFiltro(hecho: Hecho)* y permite encapsular la lógica correspondiente a cada criterio de filtrado. Actualmente, se implementan estrategias de filtrado por zona, fecha, título y categoría. Esta decisión favorece la reutilización de componentes, facilita la incorporación de nuevos tipos de filtro sin afectar los existentes y permite componer dinámicamente distintos filtros según el contexto de uso.

Introducción de la fuente de datos FuenteEstaticaCsv

Para desacoplar la carga de hechos desde fuentes externas, se incorporó la clase *FuenteEstaticaCsv*, que implementa la interfaz *FuenteDeDatos*. Esta define el método *obtenerHechos(criterios: Set<FiltroStrategy>): Set<Hecho>*, lo cual estandariza el mecanismo de obtención de hechos y permite extender el sistema con nuevas fuentes (por ejemplo, servicios web o APIs externas) sin afectar el comportamiento de las colecciones. Esta abstracción promueve la apertura del diseño a cambios futuros y mantiene bajo acoplamiento con las clases consumidoras.

Delegación de la lectura de archivos al patrón Adapter

Se delegó en *FuenteEstaticaCsv* la responsabilidad de interactuar con el lector de archivos CSV, el cual se encuentra encapsulado mediante el patrón Adapter. Esto permite reemplazar fácilmente el componente lector sin necesidad de modificar otras partes del sistema, cumpliendo con los principios de inversión de dependencias y segregación de interfaces. El adaptador asegura además la compatibilidad con diferentes formatos de entrada o librerías externas.

Incorporación del historial de estados en Solicitud

Con el fin de mejorar la trazabilidad y el seguimiento de las decisiones tomadas sobre las solicitudes, se incorporó un historial de estados dentro de la clase *Solicitud*. Esto permite registrar cada transición del atributo estado (pendiente, aceptada o rechazada), preservando un registro completo de la evolución de cada solicitud. Esta mejora resulta clave en escenarios que requieren auditoría o análisis retrospectivo.

Asociación de múltiples contenidos multimedia en Hecho

La clase *Hecho* fue modificada para permitir la asociación con múltiples objetos de tipo Multimedia. Esta decisión responde a la necesidad de representar con mayor precisión los distintos tipos de contenido que pueden acompañar a un hecho, como imágenes, audios, videos y texto. Se mejora así la expresividad del modelo y se habilita un manejo más completo de las evidencias relacionadas con los hechos registrados.

Introducción de la clase Lugar

Con el objetivo de reflejar más adecuadamente las jerarquías geográficas, se creó la clase *Lugar*, la cual posee dos atributos: nombre y tipo. Este último se modela mediante el enumerado *TipoLugar*, que contempla los valores *LOCALIDAD*, *MUNICIPIO* y *PROVINCIA*. Esta abstracción permite representar de forma más estructurada los niveles territoriales involucrados en la ubicación de los hechos.

Refactorización de la clase Ubicación

Como complemento a la introducción de *Lugar*, se actualizó la clase *Ubicacion* para incluir los atributos *direccion: String* y *referenciaLugar: Lugar*. Esta modificación enriquece el modelo al permitir representar ubicaciones con mayor nivel de detalle, facilitando también la interpretación geográfica de los hechos registrados y su posterior procesamiento o agrupamiento.

Segunda Entrega

