

Sistema de Gestión de Contenidos (CMS) - Polimorfismo

1. Análisis del Sistema (30 puntos)

1.1 Requisitos funcionales del sistema (5 pts)

Escribir los puntos exactos que debe cumplir tu sistema.

- El usuario puede crear, editar y eliminar contenidos (Artículos, Videos, Imágenes).
- El sistema permite a cada contenido tener un comportamiento único al publicarse, aprovechando el polimorfismo.
- El usuario puede clasificar contenidos en categorías o etiquetas.
- El usuario puede buscar y filtrar contenidos según su tipo o categoría.
- El sistema debe gestionar dos roles: Administradores (publicar y eliminar) y Editores (crear y editar).
- El sistema puede generar reportes o resúmenes de los contenidos publicados.

1.2 Clases necesarias y su propósito (5 pts)

Clase	Propósito
Contenido (Abstracta)	Clase base que define atributos comunes (título, autor, fecha) y actúa como superclase para la jerarquía de contenidos. (Herencia)
IPublicable (Interfaz)	Define el método 'publicar()'. Permite el Polimorfismo al tratar diferentes tipos de Contenido de manera uniforme.
Articulo	Tipo de contenido específico para texto. Hereda de Contenido e implementa IPublicable.
Video	Tipo de contenido audiovisual. Hereda de Contenido e implementa IPublicable.
Imagen	Tipo de contenido visual. Hereda de Contenido e implementa IPublicable.
Usuario (Abstracta)	Clase base para todos los usuarios. Contiene atributos comunes (nombre, id).
Administrador	Usuario con permisos para publicar y eliminar contenidos. Hereda de Usuario.

Editor	Usuario limitado a crear y editar contenido. Hereda de Usuario.
Categoria	Representa la clasificación temática (categorías) de los contenidos.
CMSModel	Clase del Modelo. Gestiona colecciones de datos (contenidos, usuarios) y la lógica de negocio (CRUD, reportes, búsqueda).
CMSController	Clase del Controlador. Maneja la entrada de la View y coordina las actualizaciones en el Model.
CMSView	Clase de la Vista. Maneja la interacción con el usuario (ej. menú de consola) y la salida de información.

1.3 Atributos de cada clase (10 pts)

Clase: Contenido (Abstracta)

Atributo	Tipo de dato	Visibilidad	Propósito
id	int	- private	Identificador único del contenido.
titulo	String	- private	Título del contenido.
autor	Usuario	- private	Usuario que creó el contenido.
fechaCreacion	Date	- private	Fecha y hora de la creación del objeto.
categoria	Categoria	- private	Objeto Categoria para clasificación.
estado	String	- private	Estado del contenido: "Creado", "Editado", "Publicado".

Clase: Artículo (Hereda de Contenido)

Atributo	Tipo de dato	Visibilidad	Propósito
cuerpo	String	- private	El texto completo que conforma el artículo.
palabrasClave	List<String>	- private	Etiquetas o keywords adicionales.
longitud	int	- private	Número de palabras del artículo.

Clase: Usuario (Abstracta)

Atributo	Tipo de dato	Visibilidad	Propósito
idUsuario	int	- private	Identificador único del usuario.
nombre	String	- private	Nombre completo del usuario.
email	String	- private	Correo electrónico (puede servir para login).

Clase: CMSModel

Atributo	Tipo de dato	Visibilidad	Propósito
contenidos	List<Contenido>	- private	Colección para gestionar todos los contenidos (Articulos, Videos, Imagenes).
usuarios	List<Usuario>	- private	Colección para gestionar los usuarios del sistema.

Agregar las necesarias

1.4 Métodos de cada clase (10 pts)

Clase: Contenido (Abstracta)

Clase	Método	Parámetros : Tipo de dato	Visibilidad	Propósito
Contenido	Contenido()	id, titulo, autor, categoria	+ public	Constructor.
Contenido	getTipo()	: String	+ public	Devuelve el tipo de contenido ("Articulo", "Video", etc.) para filtros.
Contenido	setEstado()	estado: String	+ public	Permite actualizar el estado del contenido (ej. Publicado).

Contenido	abstract visualizar()	: void	+ public	Define el método base de visualización.
-----------	--------------------------	--------	----------	-----------------------------------------------

Interfaz: IPublicable

Clase	Método	Parámetros : Tipo de dato	Visibilidad	Propósito
IPublicable	publicar()	: void	+ public	Define la acción de publicación única para cada contenido. (Polimorfismo por Interfaz)

Clase: Artículo (Implementa IPublicable)

Clase	Método	Parámetros : Tipo de dato	Visibilidad	Propósito
Articulo	publicar()	: void	+ public	Implementación de IPublicable: Muestra el artículo en formato web.
Articulo	calcularLongitud()	: int	+ public	Calcula la longitud en palabras del cuerpo del artículo.

Clase: CMSModel

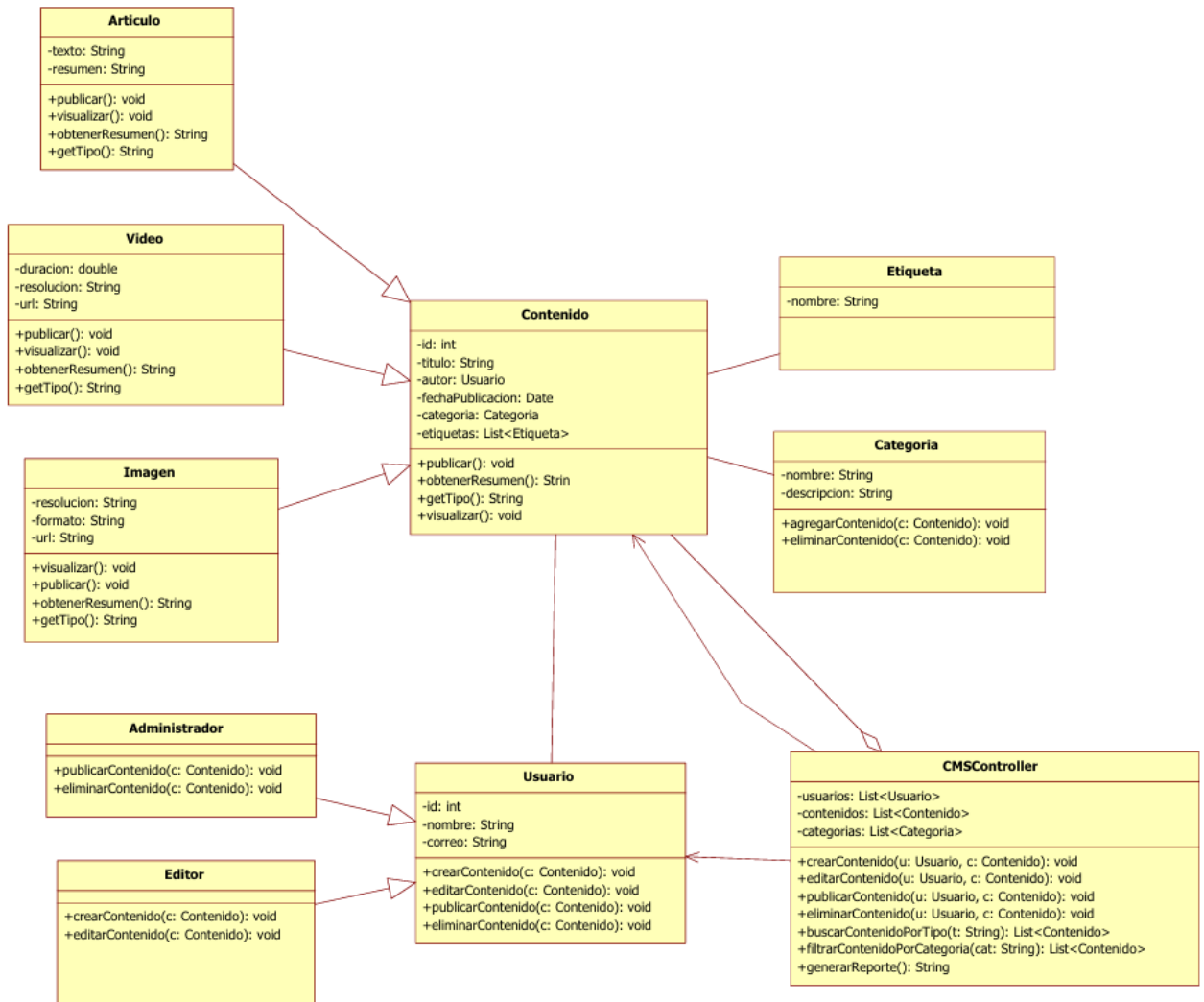
Clase	Método	Parámetros : Tipo de dato	Visibilidad	Propósito
CMSModel	crearContenido()	c: Contenido, u: Usuario	+ public	Agrega un nuevo contenido a la lista.
CMSModel	editarContenido()	id: int, c: Contenido, u: Usuario	+ public	Actualiza un contenido existente (permisos de Editor/Admin).

CMSModel	eliminarContenido()	id: int, u: Usuario	+ public	Elimina un contenido (permisos de Administrador).
CMSModel	publicarContenido()	id: int, u: Usuario	+ public	Busca contenido, verifica permisos, y ejecuta su método publicar().
CMSModel	buscarContenido()	tipo: String, categoria: String	: List<Contenido>	Filtra contenidos por tipo o categoría.
CMSModel	generarReporte()	: String	+ public	Genera un resumen de los contenidos publicados (títulos, autores, fecha).

2. Diseño: Diagrama de Clases (30 puntos)

- Asegúrate de mostrar atributos y métodos con visibilidad (+, -). [cite: 105]
- Indica relaciones entre clases (asociación, agregación, herencia, implementación). [cite: 106]
- Incluye el driver program (Main). [cite: 107]

Diagrama de clases aquí o adjunto en un archivo aparte. [cite: 108]



3. Programa (40 puntos)

En cada archivo `*.java`, asegurarse de incluir: [cite: 110]

- Las clases necesarias. [cite: 111]

- Uso adecuado de objetos. [cite: 112]

Menú que debe implementar el driver program: [cite: 113]

- Login (como Administrador o Editor)

- Nuevo Contenido (Artículo, Video, Imagen)
- Editar Contenido
- Publicar Contenido
- Eliminar Contenido
- Buscar Contenido por Tipo
- Filtrar Contenido por Categoría
- Generar Reporte
- Salir

GitHub: colocar aquí la URL: [cite: 119]

<https://github.com/0o0-ct/Polimorfismo.git>

Checklist antes de entregar [cite: 121]

- ✓ ¿Está claro el análisis? [cite: 122]
- ✓ ¿El diagrama tiene los elementos UML correctamente? [cite: 123]
- ✓ ¿Subiste tu código a GitHub con todo lo necesario? [cite: 124]