Laboratorio 4 Polimorfismo: CMS

Competencias por desarrollar:

- Identificar las funcionalidades necesarias para crear una solución viable y útil.
- Correcto manejo de clases y sus relaciones.
- Modelar su programa en UML.
- Identificar escenarios donde puede aprovecharse el polimorfismo vía interfaces.

Instrucciones

El mayor reto como desarrolladores es poder concretar las necesidades de un usuario para no crear funcionalidades adicionales o que falten partes importantes a implementar.

La **Estudio de Grabación Audiovisual (EGA)** necesita una herramienta que les permita gestionar y publicar diferentes tipos de contenidos en línea. Dado que cada vez más catedráticos y estudiantes generan materiales académicos digitales, se requiere un sistema que facilite la administración, publicación y organización de artículos, videos, e imágenes desde una sola plataforma. Esto contribuirá a que sea más dinámico poder ver y compartir el contenido que generan y que llegue a más posibles estudiantes interesados en estudiar o ser parte.

(ver video de referencia: https://www.youtube.com/watch?v=36U_JOJQYUw)

Para esto, usted ha sido elegido para presentar una propuesta de software que implemente un **Sistema de Gestión de Contenidos (CMS)**, aplicando los principios de **Programación Orientada a Objetos (POO)** y el patrón de arquitectura **MVC (Modelo–Vista–Controlador)**.

El sistema debe permitir **crear, editar y eliminar contenidos** como artículos, videos o imágenes.

Cada tipo de contenido debe tener su propio comportamiento al momento de **publicarse o visualizarse**.

Los contenidos deben poder **clasificarse en categorías o etiquetas**.

Se debe poder **buscar y filtrar** contenidos según su tipo o categoría.

Existen diferentes **tipos de usuarios**: administradores (pueden publicar y eliminar) y editores (solo pueden crear y editar).

El sistema debe permitir **generar reportes** o resúmenes de los contenidos publicados.

En el futuro podrían agregarse **nuevos tipos de contenidos** (por ejemplo, podcasts o infografías).

PARTE 1

Determine cuales clases considera necesarias para realizar su solución. Recuerde trabajar con MVC. Para esto plantéese las siguientes interrogantes:

- ¿Como debe verse nuestra jerarquía de clases?
- ¿Cuál clase podría heredar para evitar repetir código?
- ¿Cuáles clases podríamos relacionar con una interface para que al agregar una nueva clase sepamos cuales métodos son necesarios implementar al momento que crezca nuestro desarrollo?

Genere su diagrama de clases usando UML que permita ver y comprender su solución.

(Adjunte una imagen muy simple de cómo se vería su "Vista" (un prototipo de GUI) para identificar todas las funcionalidades que su "Controlador" debe contemplar)

¿Qué otra funcionalidad podría incluir su CMS que no tenemos contemplada aun?

Preparación del ambiente de trabajo:

- Cree un repositorio nuevo.
- Asegúrese de que los integrantes pueden colaborar en el proyecto.
- Defina la estructura de carpetas del proyecto para organizar las clases a usar.
- Cree sus clases con sus atributos y los métodos pendientes a implementar (solo cree los métodos aun no funcionales).
- Suba la estructura y los archivos generados al repositorio.

Entregables

- Parte 1
  - Documento .pdf que contenga:
    - Explicación del propósito de las clases
    - Explicación de la organización de la jerarquía de clases
    - Explicación del propósito de las interfaces a utilizar
    - Prototipo de GUI que le permitió identificar requisitos funcionales.
  - Diagrama de clases.
    - Imagen (.jpeg o .png) con buena resolución que permita ver todos los detalles del diagrama de clases diseñado
  - Link del repositorio de github donde trabajará el grupo.

Evaluación

[50 pts.] Parte 1 - Análisis y diseño.

- Sintaxis y correcto uso de relaciones y modificadores de visibilidad en el diagrama de clases [**10 pts.]**
- Correcta separación de responsabilidades y aplicación de MVC y los principios de diseño para el cumplimiento de los requisitos funcionales. **[10 pts.]**
- Correcto uso de herencia en las relaciones entre clases. Evidencia de polimorfismo en controladores **[10 pts.]**
- Correcto uso de interfaces en las relaciones entre clases. Evidencia de polimorfismo en controladores **[10 pts.]**
- Repositorio de Github con estructura de carpetas y de clases diseñadas. **[10 pts.]**

PARTE 2

Desarrollar un producto de calidad requiere organización para que no sea difícil al momento de agregar más funcionalidades o agregar personas al equipo. Una gran ventaja es que ya tenemos los conocimientos para poder trabajar de forma colaborativa.

Con su grupo de trabajo, implemente la funcionalidad que identificó y diseñó en la Parte 1. Debe programar en .java. Debe haber evidencia de código entregado por todos los integrantes del grupo en el repositorio.

Entregables

- Enlace al repositorio de github usado para la implementación.

Evaluación

[50 pts.] Parte 2 - Programa que implemente su diseño.

- **[15 pts.]** Correcto uso de *polimorfismo vía herencia e interfaces*. Las clases implementadas, incluyendo herencia e interfaces son suficientes para resolver el problema planteado.
- **[20 pts.]** Cumplimiento de requisitos funcionales.
- **[10 pts.]** Usabilidad; interfaz amigable con el o los usuarios.
- **[05 pts.]** Comentarios y encabezados.

Nota: La evaluación será individual, si no hay evidencia de contribuciones significativas al repositorio no tendrá nota.