**Proyecto: Gestor de Tareas**

**1. Clases y Estructura**

* **Tarea (Task)**: Esta clase debe tener las propiedades necesarias como:
  + Título, Descripción, Fecha de vencimiento, Categoría, Prioridad.
  + Propiedades autoimplementadas.
  + Métodos como MostrarTarea(), ActualizarTarea(), EliminarTarea().
  + Validaciones con **expresiones regulares** (por ejemplo, para la fecha de vencimiento).
* **Categoría (Category)**: Esta clase debe contener:
  + Nombre de la categoría (puede ser un enum con categorías predefinidas).
* **Gestor de Tareas (TaskManager)**: Esta clase debe usar una **colección** (como List<Task>) para almacenar las tareas.
  + Métodos para **agregar**, **editar**, **eliminar**, **buscar** (por fecha, prioridad, etc.).
  + Métodos que permitan visualizar todas las tareas.

**2. Funcionalidades a Implementar**

* **Condicionales**:
  + Usar condicionales para verificar la prioridad de una tarea y asignar diferentes colores o mensajes según la prioridad.
* **Bucles**:
  + Utilizar bucles foreach, for o while para mostrar las tareas almacenadas.
* **Colecciones**:
  + Usar un Dictionary para almacenar las tareas por ID, una List para tareas sin prioridad, y un Queue o Stack para tareas urgentes (dependiendo de lo que prefieras).
* **Delegados**:
  + Usar un delegado para ordenar las tareas por fecha, prioridad, etc., utilizando Func o Predicate.
* **Tipos Genéricos**:
  + Utiliza un tipo genérico en alguna parte del proyecto (por ejemplo, un método que maneje diferentes tipos de colecciones).
* **Enums**:
  + Utiliza un enum para definir prioridades (Alta, Media, Baja).
* **Interfaces**:
  + Define una interfaz IGestorTareas que implemente las operaciones de agregar, editar y eliminar tareas.
* **Clases Abstractas**:
  + Define una clase abstracta TareaBase que sea extendida por la clase Tarea.
* **Excepciones**:
  + Crea excepciones personalizadas para manejar errores (por ejemplo, una excepción si se intenta agregar una tarea sin fecha de vencimiento válida).
* **Getters y Setters**:
  + Usa getters y setters donde sea necesario.
* **Métodos estáticos**:
  + Un método estático para contar cuántas tareas se han agregado.
* **Sobreescritura de Métodos**:
  + Sobrescribe el método ToString() para que las tareas se impriman de una forma legible.

**3. Interfaz de Usuario (Consola)**

* Menú con opciones para agregar, editar, eliminar, ver tareas.
* Opción para filtrar tareas por prioridad o fecha.
* Opción para mostrar todas las tareas almacenadas.

**Requisitos:**

* Usa **excepciones** para manejar casos como fechas mal formateadas.
* Implementa **delegados** para ordenación de tareas por fecha o prioridad.
* Usa **generics** para crear métodos que trabajen con diferentes colecciones.

**Bonus:**

* Implementa la opción de **guardar las tareas** en un archivo (por ejemplo, usando StreamWriter y StreamReader).