# Documento Breve – Descripción del Proyecto de Gestión Genérica en Java

Nombre del Proyecto: Sistema de Gestión Genérico con Java Swing

Lenguaje: Java SE

Paradigma: Orientado a Objetos

Arquitectura: MVC + Patrones de Diseño

Persistencia: Archivos JSON usando Gson

## 1. Estructura del proyecto y aplicación del patrón MVC

El proyecto se estructuró utilizando el patrón arquitectónico Modelo-Vista-Controlador (MVC) con paquetes separados:  
  
- MODELO: contiene clases que representan los datos (Entidad, DefinicionAtributo) y servicios de persistencia (PersistenciaService).  
- VISTA: contiene interfaces gráficas (VistaDefinicionAtributos, VistaEntidadFormulario, VistaTablaEntidades, etc.) que se adaptan dinámicamente a los atributos definidos.  
- CONTROLADOR: coordina la interacción entre modelo y vista (ControladorDefinicionAtributos, ControladorEntidadFormulario, ControladorVistaTablaEntidades).  
- SINGLETON: almacena atributos definidos por el usuario globalmente (AtributosSingleton).  
  
Cada vista se comunica con su respectivo controlador, el cual se encarga de validar, transformar y actualizar tanto la vista como el modelo según sea necesario.

## 2. Patrones de diseño utilizados y justificación

Se aplicaron los siguientes tres patrones de diseño:  
  
- Singleton: para compartir la definición dinámica de atributos (AtributosSingleton) en todo el sistema sin necesidad de pasar referencias entre clases.  
- Factory (implícita): el controlador de formulario actúa como fábrica al crear objetos Entidad en tiempo de ejecución basados en los atributos definidos.  
- Strategy (lógica): se implementó de forma lógica al validar dinámicamente valores según su tipo (Integer o String) durante la entrada de datos.  
  
Adicionalmente, podría considerarse el patrón Observer, ya que las vistas responden indirectamente a cambios en los datos persistidos, aunque no se implementó de forma explícita con interfaces de suscripción.

## 3. Manejo de definición dinámica de atributos

Desde el inicio, el sistema solicita al usuario definir los atributos que tendrá la entidad genérica (por ejemplo: nombre, edad, materia). Esta definición se guarda y es usada para:  
  
- Generar formularios de entrada dinámicamente.  
- Crear columnas de tablas.  
- Validar datos según el tipo de atributo.  
- Persistir y recuperar información con estructura dinámica.  
  
Esto se logró usando Map<String, Object> dentro de la clase Entidad, permitiendo manejar cualquier cantidad y tipo de atributo sin necesidad de redefinir clases.

## 4. Consideraciones de diseño y limitaciones

- Se decidió usar archivos JSON para evitar la complejidad de una base de datos externa, facilitando portabilidad y simplicidad.  
- La validación de tipos fue limitada a String e Integer por simplicidad, pero puede escalarse a tipos como boolean, double, Date, etc.  
- La edición y eliminación de entidades se hace desde la tabla de forma sencilla, pero podría mejorarse con confirmaciones o controles más detallados.  
- La búsqueda y filtrado aún no está implementada, pero la estructura está lista para integrarla.