



## PROYECTO ISI :SISTEMA DE GESTIÓN ESTUDIANTIL

### Análisis del código fuente y patrones de diseño:

- El patrón que se identificó es el singleton.
- El patrón Singleton se aplica en la clase DBConfigSingleton. Esta clase asegura que solo exista una única instancia de sí misma al mantener un atributo estático y privado instance del mismo tipo de la clase. Su constructor es privado y estático, y el método getInstance es el encargado de devolver siempre la misma instancia creada..
- El patrón de diseño identificado es Singleton.Se utiliza para asegurar una única conexión global a la base de datos para cada usuario.

### Implementación de Historias de Usuario (HU):

ID de HU	001
Título	Alta de profesor al sistema
Declaración	<b>Como</b> administrador del sistema, <b>quiero</b> registrar un nuevo profesor ingresando su información personal, <b>para</b> poder asignarlo a las asignaturas correspondientes dentro de una carrera.
Descripción Detallada	El sistema permitirá al administrador registrar nuevos profesores a través de un formulario digital. El formulario incluirá los campos obligatorios: nombre, apellido, DNI, correo electrónico, dirección y número de matrícula En caso de que falte un campo se avisará de que falta un

	<p>campo a completar.</p> <p>Una vez completado correctamente, el sistema almacenará los datos en la base de datos y confirmará el alta con un mensaje de éxito.</p> <p>El sistema mostrará un error si un dni o contacto ya fue registrado con anterioridad como también si el formato del contacto es incorrecto.</p> <p>También se incluirá la opción de volver al inicio que vuelve a la pantalla de inicio.</p>
<p>Criterios de Validación (Criterios de Aceptación)</p>	<ul style="list-style-type: none"> <li>● <b>Flujo exitoso:</b> Al completar todos los campos obligatorios (nombre, apellido, correo, DNI) con datos válidos y guardar, el sistema muestra un mensaje de éxito..</li> <li>● <b>Validaciones de Datos:</b> El sistema debe impedir el registro si: <ul style="list-style-type: none"> <li>○ Faltan campos obligatorios.</li> <li>○ Si el formato del correo electrónico no es válido.</li> <li>○ El correo electrónico o el DNI ya existen en la base de datos.</li> </ul> </li> <li>● <b>Manejo de Errores:</b> Si alguna validación falla, el sistema debe mostrar un mensaje de error claro, sin permitir que se guarde el formulario.</li> <li>● <b>Acción de Cancelar:</b> El formulario debe incluir un botón "Cancelar" que elimine todos los datos ingresados y devuelva al usuario a la pantalla anterior.</li> </ul>
<p>Tareas Asociadas a la Implementación</p>	<p><b>Creación de la base de datos:</b></p> <ul style="list-style-type: none"> <li>● Crear las tablas <b>persona</b> y <b>docente</b>.</li> </ul> <p><b>Implementación del modelo:</b></p> <ul style="list-style-type: none"> <li>● Crear las clases <b>Persona</b> y <b>Docente</b> en Java, con sus atributos, constructores, getters/setters</li> </ul> <p><b>Configuración de rutas en Spark Java:</b></p>

	<ul style="list-style-type: none"><li>● Definir la <b>ruta GET</b> que muestra el formulario de alta de docente.</li><li>● Definir la <b>ruta POST</b> que procesa el formulario, valida los datos y registra el nuevo docente.</li></ul> <p><b>Desarrollo del frontend con Mustache:</b></p> <ul style="list-style-type: none"><li>● Crear la plantilla <b>docente_form.mustache</b> con los campos de entrada y botones requeridos</li></ul>
--	--