



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	Como administrador del sistema, quiero registrar un nuevo profesor ingresando su información personal, para 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, se incluirá la opción de volver al inicio que vuelve a la pantalla de inicio. El formulario incluirá los campos obligatorios: nombre, apellido, DNI, correo electrónico, dirección y número de matrícula

	<p>En caso de que falte un campo se avisará de que falta un campo a completar.</p> <p>Una vez que todos los datos se ingresen correctamente, el sistema almacenará la información en la base de datos y mostrará un mensaje de éxito sobre el formulario, ofreciendo opciones para registrar otro docente o volver al inicio.</p> <p>Si el DNI o el contacto ya fueron registrados previamente, o si el formato del contacto es incorrecto, el sistema mostrará un mensaje de error, permitiendo al administrador reintentar el alta o regresar al inicio.</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"> ● Flujo exitoso: Al completar todos los campos obligatorios (nombre, apellido, correo, DNI) con datos válidos y guardar, el sistema muestra un mensaje de éxito.. ● Validaciones de Datos: El sistema debe impedir el registro si: <ul style="list-style-type: none"> ○ Faltan campos obligatorios. ○ Si el formato del correo electrónico no es válido. ○ El correo electrónico o el DNI ya existen en la base de datos. ● Manejo de Errores: Si alguna validación falla, el sistema debe mostrar un mensaje de error claro, sin permitir que se guarde el formulario. ● Acción de Cancelar: El formulario debe incluir un botón "Cancelar" que elimine todos los datos ingresados y devuelva al usuario a la pantalla anterior.
<p>Tareas Asociadas a la Implementación</p>	<p>Creación de la base de datos:</p> <ul style="list-style-type: none"> ● Crear las tablas persona y docente. <p>Implementación del modelo:</p> <ul style="list-style-type: none"> ● Crear las clases Persona y Docente en Java, con sus atributos, constructores, getters/setters

Configuración de rutas en Spark Java:

- Definir la **ruta GET** que muestra el formulario de alta de docente.
- Definir la **ruta POST** que procesa el formulario, valida los datos y registra el nuevo docente.

Desarrollo del frontend con Mustache:

- Se creó la plantilla **docente_form.mustache**, que contiene un formulario de ingreso de docentes, mensajes de feedback para informar al administrador si el alta fue exitosa o si hubo errores, y los botones necesarios para enviar el formulario o navegar según corresponda.