Análisis del código fuente y patrones de diseño:

Patrón Singleton.

```
public static synchronized DBConfigSingleton getInstance() {
        if (instance == null) {
            instance = new DBConfigSingleton();
        }
        return instance;
    }
```

public para que el método sea global.
static permite llamar al método desde la clase, sin la necesidad de crear un objeto primero.
synchronized evita el uso concurrente del método, lo que podría romper el patrón.

Chequea si existe previamente una instancia de la clase, si existe la devuelve, si no, la crea y devuelve.

```
//Obtener la instancia única del singleton de configuración de la base de datos.
DBConfigSingleton dbConfig = DBConfigSingleton.getInstance();
```

Se obtiene la única instancia de la BD, ahora dbConfig hace referencia a dicha instancia.

Este patrón creacional evita posibles errores relacionados a múltiples conexiones a la base de datos y duplicación de recursos, permitiendo un acceso centralizado y eficiente.

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	Una vez ingresado al sistema se debe mostrar una ventana

Detallada	(panel de control) con la opción de cargar un docente, al seleccionar la opción, se deberá desplegar una pestaña con los campos de entrada de datos correspondientes al docente. Si se ingresa un DNI que ya se encuentra cargado en la BD, se deberá mostrar un mensaje mencionando que el docente ya se encuentra cargado, caso contrario, los datos se almacenan en la correspondiente BD. Los datos se cargarán en la BD solo si se ingresan datos en TODOS los campos.
Criterios de Validación (Criterios de Aceptación)	● 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: ○ 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.
Tareas Asociadas a la Implementación	☐ Crear la clase Teacher ☐ Crear la tabla Teacher en la base de datos ☐ Importar las dependencias correspondientes ☐ Implementar setters y getters ☐ Implementar la carga de docentes en la clase App