

**Profesor: Victor Contreras**

**Alumno: Gonzalo Rosa**

**Trabajo Práctico**

**Ingenieria**

**de Software II**

**1) Capas del sistema y funciones principales**

**Capa de Presentación (UI)**

* Interacción con el usuario: mostrar catálogos, formularios de préstamo, devoluciones y registro de socios.
* Validación básica de entrada (campos obligatorios).
* Llamadas a la capa de lógica de negocio (servicios).

**Capa de Lógica de Negocio (Business)**

* Reglas del dominio: comprobar disponibilidad de libros, políticas de préstamo (duración máxima, penalización por no devolverlo a tiempo), validaciones complejas.
* Organización: coordinar entre repositorios (datos) y la UI.
* Transacciones de negocio (proceso de préstamo: verificar socio activo, libro disponible, registrar préstamo).

**Capa de Datos (DAL / Repositorio)**

* Acceso a la base de datos: consultas SQL, inserciones, actualizaciones.
* Mapear resultados a objetos del dominio.
* Encapsular detalles de persistencia (para que la lógica de negocio no dependa de SQL directo).

**2) Problema elegido y patrón de diseño sugerido**

**Problema:** *Acceso centralizado a la base de datos / manejo de la conexión*  
Si la aplicación abre conexiones sueltas a la base de datos, puede haber duplicación de código, errores de concurrencia y consumo excesivo de recursos.

**Patrón sugerido: Singleton** (para el gestor de la base de datos) combinado con **DAL (Capa de Acceso a Datos).**

* El Singleton asegura que solo exista un gestor de conexiones a la base de datos.
* La DAL encapsula todas las operaciones con la base de datos, exponiendo métodos claros (obtener\_por\_id, guardar, listar\_disponibles, etc.).
* Resultado: la lógica de negocio usa la DAL sin preocuparse por los detalles de conexión y hay control centralizado de las conexiones.

Diagrama

El contenido generado por IA puede ser incorrecto.**3) Diagrama UML**

**4) Lenguaje elegido para la aplicación:** Python.