**Proyecto Final: Requisitos de entrega y requerimientos del software**

Este documento tiene la función de sintetizar los requerimientos funcionales y no funcionales del proyecto final de modelos de programación. (documento sujeto a cambios)

Para entregar el proyecto se requiere lo siguiente.

1. **Acta de constitución:**
   * Descripción de la metodología ágil seleccionada.
   * Cronograma detallado (fases, reuniones, objetivos y metas alcanzables).
   * Justificación de roles asignados y descripción de reuniones por fase.
2. **Sistema funcional:**
   * Desarrollo en Java con GUI.
   * Cumplimiento de los requerimientos establecidos.
3. **Diagrama de clases UML:**
   * Identificar patrones de diseño utilizados.
4. **Manuales del sistema:**
   * Manual técnico.
   * Manual de usuario.
5. **Presentación del proyecto:**
   * Video (máximo 15 minutos) cargado en YouTube mostrando resultados.
6. **Informe de ingeniería:**
   * Decisiones tomadas, justificación de patrones de diseño y reflexión crítica del proyecto.

**Requerimientos del Proyecto**

**Requerimientos Funcionales**

1. **Gestión de documentos:**
   * Permitir que usuarios introduzcan, consulten y modifiquen documentos. (CRUD)
2. **Datos de los documentos:**
   * Título.
   * Fecha de publicación (año, mes, día).
   * Autores.
   * ISBN o SSN.
   * Editorial.
3. **Tipos de documentos:**
   * **Libros:** Registrar el número total de páginas.
   * **Ponencias:** Registrar el nombre del congreso donde se presentaron.
   * **Artículos científicos:** Utilizar SSN en lugar de ISBN.
4. **Información de contacto:**
   * Correo electrónico.
   * Dirección física.
   * Número de teléfono.
5. **Historial de documentos:**
   * Registrar quién creó el documento y la fecha de creación.
   * Registrar modificaciones (usuario y fecha).
   * Registrar eliminación (usuario y fecha, solo por el creador).
6. **Estado de los documentos:**
   * Crear documentos inicialmente con datos en blanco.
   * Gestionar reservas online para consulta física.
   * Permitir devoluciones de documentos.
   * Baja de documentos (no se pueden eliminar si están reservados).
   * Reactivación de documentos dados de baja.
7. **Autenticación de usuarios:**
   * Utilizar usuario y contraseña para gestionar permisos de creación, modificación y eliminación de documentos.

**Requerimientos No Funcionales**

1. **Interfaz gráfica de usuario (GUI):**
   * Desarrollo en Java.
   * Interfaz intuitiva y accesible.
2. **Seguridad:**
   * Manejo adecuado de credenciales y datos sensibles.
3. **Diseño mantenible:**
   * Cumplir con estándares de diseño y patrones de programación. (SOLID)
4. **Calidad del software:**
   * Enfocarse en la escalabilidad y mantenibilidad del sistema.(SOLID)

**Algunos posibles patrones de diseño que se pueden utilizar**

1. **DAO (Data Access Object)**
2. **DTO**

* **Propósito:**
  + Separar la lógica de acceso a la base de datos del resto de la aplicación.
* **Aplicación:**
  + CRUD de documentos y usuarios.

### Factory Method

* **Propósito:**
  + Crear diferentes tipos de documentos (libros, ponencias, artículos) mediante una interfaz común.
* **Aplicación:**
  + Simplifica la creación y extensión de nuevas clases de documentos.

**3. Singleton**

* **Propósito:**
  + Asegurar que solo exista una instancia del sistema de autenticación o conexión a la base de datos.

\* PROXY

**4. Observer 🡪(STATE)**

* **Propósito:**
  + Notificar cambios en el estado de los documentos (por ejemplo, reservas o devoluciones).

**6. Template Method (No estoy muy seguro toca analizarlo mejor)**

* **Propósito:**
  + Definir la estructura de una operación dejando que las subclases implementen pasos específicos.
* **Aplicación:**
  + Estandarizar operaciones comunes entre los tipos de documentos.

**7. Builder**

* **Propósito:**
  + Construir objetos complejos (documentos) de manera controlada.
* **Aplicación:**
  + Garantiza que todos los campos requeridos estén completados correctamente.