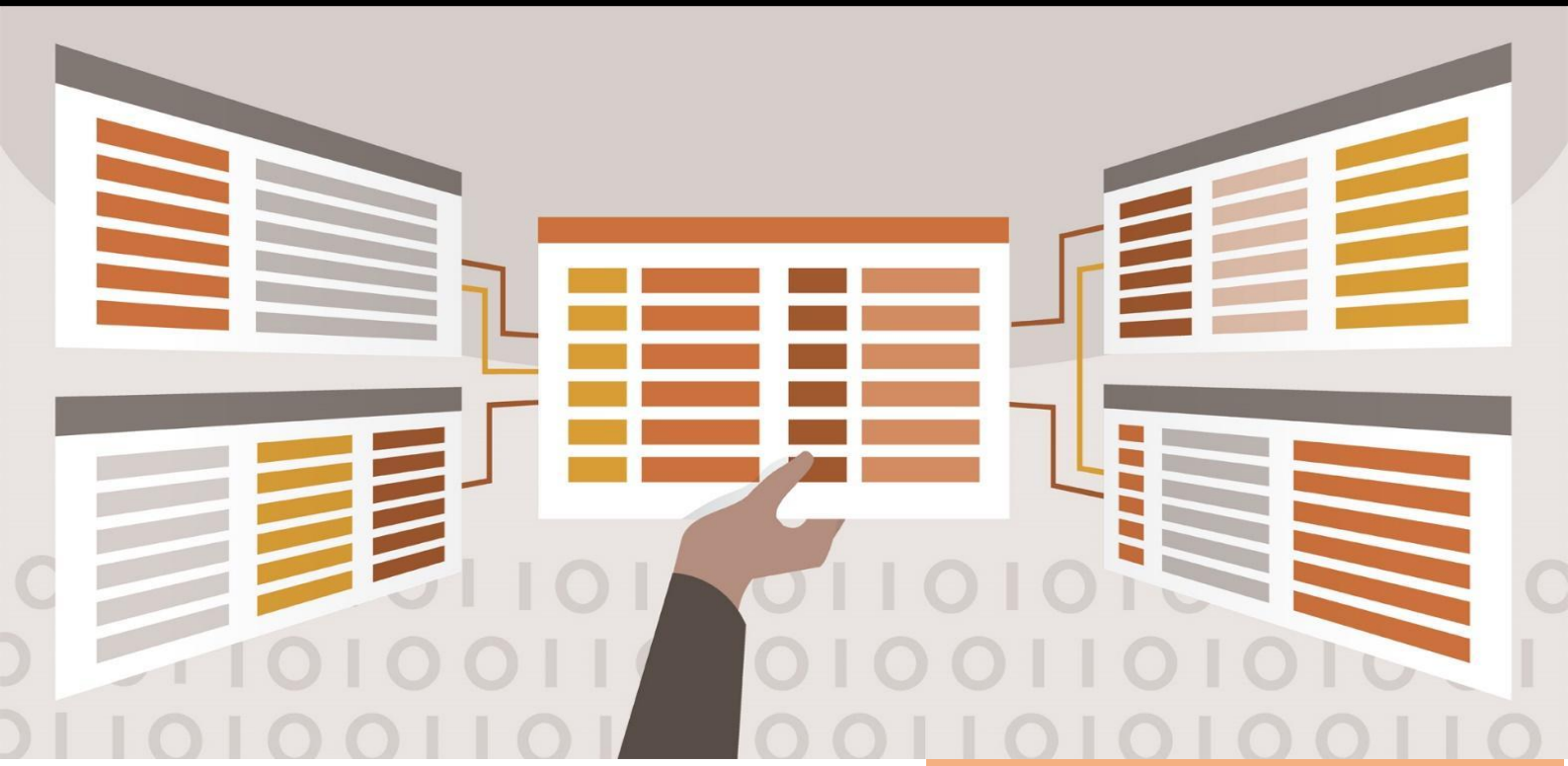
salesianos

2025

Trabajo de Digitilizacion



Jesus A. Gómez Rodríguez
Colegio Salesiano “San Ignacio”
5-2-2025

Índice

1. *Introducción al Proyecto de Digitalización* 3
2. *Alcance del Proyecto* 3
3. *Introducción a la Programación* 4
4. *Introducción a las Bases de Datos* 5
   a. *Relacionales (SQL):* 6
   b. *No Relacionales (NoSQL):* 6
5. *Introducción a Docker* 7
6. *Conclusión* 7
7. *Expectativas del proyecto a futuro* 8
   a. *Incorporar autenticación de usuarios:* 8
   b. *Añadir validaciones y control de errores:* 8
   c. *Integración con APIs externas:* 8
   d. *Mejora en la experiencia de usuario (UX/UI):* 8
   e. *Despliegue con Docker y MongoDB Atlas:* 8
   f. *Creación de una versión web o móvil:* 8
   g. *Incorporar funcionalidades avanzadas:* 9
8. *Bibliografía* 10

1. Introducción al Proyecto de Digitalización

El proyecto de digitalización de tareas cotidianas tiene como objetivo modernizar y optimizar los procesos tradicionales mediante la implementación de tecnologías actuales. En este contexto, se presenta la aplicación **Book Manager**, una herramienta diseñada para la gestión de libros. Este sistema busca reemplazar los métodos manuales de almacenamiento, búsqueda y organización de libros, mejorando la eficiencia, la accesibilidad y la seguridad de la información.

Desarrollada en Java, **Book Manager** utiliza MongoDB como base de datos NoSQL, lo que permite una mayor flexibilidad y escalabilidad en el manejo de datos. La elección de **MongoDB** responde a la necesidad de trabajar con grandes volúmenes de información que no requieren un modelo de datos fijo, facilitando la incorporación de diferentes tipos de libros y datos asociados sin complicaciones.

La implementación de **Java Swing** para la creación de la interfaz gráfica proporciona una experiencia de usuario amigable y moderna, permitiendo que los usuarios gestionen sus colecciones de libros de manera rápida e intuitiva. Este trabajo demuestra cómo las herramientas tecnológicas actuales pueden transformar procesos manuales en soluciones digitales eficientes, llevando al estudiante a una comprensión más profunda de cómo las aplicaciones modernas facilitan el trabajo y mejoran la productividad en un entorno profesional.

2. Alcance del Proyecto

Book Manager es una aplicación de escritorio que integra funcionalidades clave para la gestión eficiente de información bibliográfica. Su diseño está enfocado en la facilidad de uso y la capacidad de adaptarse a las necesidades cambiantes del usuario, lo que lo convierte en una solución práctica tanto para bibliotecas personales como para pequeñas colecciones de libros.

Las principales características de la aplicación incluyen:

- **Gestión de Libros**: Los usuarios pueden agregar libros con detalles como el título, autor, género, editorial, y una URL de la portada. Esta última se carga dinámicamente desde internet, lo que optimiza el almacenamiento local al evitar guardar archivos de imagen pesados.
- **Búsqueda Avanzada**: La aplicación permite buscar libros utilizando diversos filtros, como el título, el autor o el género. Esto proporciona flexibilidad al usuario para localizar cualquier libro en su base de datos sin importar la cantidad de títulos almacenados.
- **Operaciones CRUD**: La aplicación soporta operaciones completas de gestión de libros, permitiendo **crear**, **leer**, **actualizar** y **eliminar** entradas en la base de datos MongoDB. Esto facilita mantener actualizada la colección de libros y asegura que la información esté siempre disponible.

- **Conexión con MongoDB**: **Book Manager** almacena los libros en formato JSON dentro de MongoDB, una base de datos NoSQL. Este enfoque permite almacenar información de manera más flexible y sin necesidad de definir un esquema fijo, lo que es especialmente útil al gestionar libros con características variadas y potencialmente cambiantes.
- **Carga Dinámica de Portadas**: A través de la URL almacenada en la base de datos, las imágenes de portada se cargan desde internet. Esta funcionalidad optimiza el espacio de almacenamiento en el dispositivo del usuario y garantiza que las portadas siempre sean actuales y accesibles.

En futuras versiones del proyecto, se considera la posibilidad de desplegar la aplicación en contenedores Docker, lo que simplificaría su implementación y escalabilidad, permitiendo que **Book Manager** pueda ser fácilmente distribuido y ejecutado en diferentes entornos, como servidores locales o en la nube.

Este proyecto no solo busca resolver un problema práctico de gestión de libros, sino también servir como una plataforma de aprendizaje que demuestra cómo la digitalización de procesos cotidianos a través de herramientas modernas como Java, MongoDB y Docker puede mejorar la eficiencia y organización en cualquier ámbito profesional.

3. Introducción a la Programación

La historia de la programación está estrechamente relacionada con la evolución de la computación. Desde tiempos antiguos, el ser humano ha buscado mecanismos para automatizar cálculos. Uno de los primeros ejemplos fue el ábaco, seguido siglos después por las primeras máquinas mecánicas de cálculo, como la Pascalina. En el siglo XIX, Charles Babbage diseñó la primera máquina analítica, un modelo conceptual de computadora mecánica, y Ada Lovelace escribió para ella el primer algoritmo, por lo que es considerada la primera programadora de la historia.

Con la llegada de los ordenadores electrónicos en el siglo XX, se introdujeron los lenguajes de programación de bajo nivel, como el lenguaje ensamblador, necesarios para interactuar directamente con el hardware. Posteriormente, surgieron lenguajes de alto nivel como FORTRAN (1957), diseñado para aplicaciones científicas, y COBOL (1959), enfocado en negocios. Estos avances hicieron que la programación fuera más accesible.

La programación informática ha pasado por diferentes etapas históricas que han marcado su evolución:

- Etapa Mecánica: Charles Babbage propuso la máquina analítica en el siglo XIX, y Ada Lovelace escribió el primer algoritmo conocido, convirtiéndose en la primera programadora.
- Lenguajes de Bajo Nivel: A mediados del siglo XX se comenzaron a usar lenguajes ensamblador que traducían instrucciones directamente al lenguaje de la máquina.
- Lenguajes de Alto Nivel: En los años 50 y 60 surgieron lenguajes como FORTRAN, COBOL y LISP, que simplificaban la programación para científicos y empresas.
- Paradigmas de Programación: Aparecieron nuevos enfoques como la programación estructurada (Pascal, C) y posteriormente la programación orientada a objetos con C++, Smalltalk y Java.
- Programación Moderna: Hoy se utilizan lenguajes como Python, JavaScript y Go que favorecen la productividad, junto con herramientas de colaboración, pruebas automatizadas y despliegue continuo.
- Ejemplos de Aplicaciones: Desde sistemas operativos, videojuegos y redes sociales, hasta inteligencia artificial y blockchain, la programación es el motor de la era digital.

4. Introducción a las Bases de Datos

Las bases de datos surgieron con la necesidad de gestionar eficientemente grandes volúmenes de información. En los años 60 y 70, con el auge de los sistemas informáticos en empresas, nacieron los primeros Sistemas de Gestión de Bases de Datos (SGBD). Edgar F. Codd propuso el modelo relacional en 1970, estableciendo las bases de lo que hoy conocemos como bases de datos SQL, que dominan en entornos donde la integridad de los datos es crucial.

Con el paso del tiempo y la llegada del Big Data, surgieron limitaciones en las bases de datos tradicionales para manejar datos masivos, distribuidos y no estructurados. Esto dio paso a las bases de datos NoSQL a partir de la década del 2000, impulsadas por empresas como Google, Amazon y Facebook. Estas bases ofrecen mayor flexibilidad y escalabilidad en la era de las aplicaciones web y móviles modernas.

Las bases de datos permiten almacenar, recuperar y manipular información de forma estructurada. Se pueden clasificar en:

a. ***Relacionales (SQL):***
   - Utilizan tablas y relaciones entre datos.
   - Requieren un esquema fijo (estructura predefinida).
   - Ejemplos: MySQL, PostgreSQL, Oracle.
   - Uso típico: sistemas bancarios, ERP, CRM.

b. ***No Relacionales (NoSQL):***
   - No utilizan el modelo de tablas rígidas.
   - Existen varios tipos:
     - Documentales (MongoDB): almacenan información en documentos JSON.
     - Clave-Valor (Redis): ideales para datos simples y rápidos.
     - De Grafos (Neo4j): modelan relaciones complejas como redes sociales.
     - Columnares (Cassandra): óptimas para análisis masivos de datos.

Ventajas de NoSQL (MongoDB en particular):

- Escalabilidad horizontal.
- Esquema flexible.
- Rápido acceso a grandes volúmenes de datos.

Ejemplo práctico en Book Manager:

```json
{
  "titulo": "1984",
  "autor": "George Orwell",
  "genero": "Distopía",
  "portada": "https://.../1984.jpg"
}
```

Este documento representa un libro almacenado en MongoDB sin necesidad de definir una tabla previa.

5. *Introducción a Docker*

Docker es una herramienta que permite empaquetar una aplicación junto a todas sus dependencias dentro de un contenedor. Los contenedores funcionan de manera similar a máquinas virtuales, pero son más ligeros y rápidos.

¿Por qué usar Docker?

- Garantiza que la aplicación funcione igual en cualquier ordenador.
- Facilita el despliegue en servidores o en la nube.
- Permite dividir la aplicación en partes independientes (microservicios).

Componentes básicos de Docker:

- Dockerfile: archivo de instrucciones para construir la imagen.
- Imagen: plantilla del contenedor (como una "foto" del sistema).
- Contenedor: instancia en ejecución de una imagen.

Uso con MongoDB:

Se puede usar docker-compose para levantar tanto la aplicación como la base de datos MongoDB en contenedores independientes pero conectados entre sí.

```yaml
version: '3.8'
services:
  mongo:
    image: mongo
    ports:
      - "27017:27017"
  bookmanager:
    build: .
    depends_on:
      - mongo
```

Con esta configuración, en segundos se puede tener el entorno de desarrollo listo en cualquier ordenador o servidor.

6. *Conclusión*

El desarrollo de Book Manager ejemplifica cómo los procesos tradicionales pueden transformarse mediante la digitalización. Gracias a Java, MongoDB y tecnologías modernas como Docker, se puede crear una herramienta funcional para gestionar bibliografía de manera simple y profesional. Este proyecto representa una introducción sólida al desarrollo de software real, fomentando habilidades clave como la lógica de programación, la gestión de datos y el pensamiento computacional.

7. Expectativas del proyecto a futuro

El desarrollo actual de Book Manager sienta las bases para una serie de mejoras y funcionalidades que pueden implementarse en el futuro, con el objetivo de convertir esta aplicación en una herramienta aún más completa, robusta y profesional. A continuación, se describen las expectativas del proyecto a corto, mediano y largo plazo:

a. Incorporar autenticación de usuarios:

Añadir un sistema de login que permita a distintos usuarios acceder con sus credenciales, diferenciando roles (usuario lector, administrador, editor). Esto permitirá personalizar el contenido y proteger la información según los permisos asignados.

b. Añadir validaciones y control de errores:

Implementar validación en los formularios de entrada de datos para asegurar que se introduzca información completa y válida (por ejemplo, no permitir campos vacíos o URLs incorrectas para imágenes). También se gestionarán errores de conexión con MongoDB y fallos en la carga de portadas.

c. Integración con APIs externas:

Conectar con servicios como OpenLibrary o Google Books API para obtener automáticamente información adicional del libro (sinopsis, ISBN, puntuación, etc.) al introducir el título o el autor. Esto mejorará la experiencia del usuario y enriquecerá la base de datos.

d. Mejora en la experiencia de usuario (UX/UI):

Rediseñar la interfaz gráfica utilizando librerías modernas para Java Swing o incluso migrar a JavaFX. Se busca una apariencia más profesional e intuitiva, incorporando paneles dinámicos, animaciones suaves y vistas por categorías o géneros.

e. Despliegue con Docker y MongoDB Atlas:

Empaquetar completamente la aplicación y la base de datos en contenedores Docker para facilitar su instalación en diferentes equipos o servidores. Además, se contempla el uso de MongoDB Atlas, una plataforma en la nube, para alojar la base de datos, lo cual permitiría acceder desde distintas ubicaciones y garantizar disponibilidad.

f. Creación de una versión web o móvil:

A futuro, se podría crear una versión web o móvil de la aplicación utilizando frameworks como Spring Boot (backend) y React o Flutter (frontend móvil). Esto permitiría a los usuarios consultar o registrar libros desde cualquier dispositivo.

g. Incorporar funcionalidades avanzadas:

Agregar opciones como exportar la base de datos a PDF o Excel, crear copias de seguridad automáticas, buscar libros por múltiples filtros combinados, y estadísticas de lectura (por ejemplo: autor más leído, género más consultado, etc.).

Estas futuras mejoras tienen como finalidad llevar a Book Manager desde un prototipo educativo a una aplicación completa, funcional y con potencial para usarse en bibliotecas escolares, personales o centros de documentación.

Incorporar autenticación de usuarios y roles de acceso.

- Añadir validación de formularios para mejorar la experiencia del usuario.
- Conectar con APIs externas de libros como OpenLibrary o Google Books.
- Mejorar la presentación de los libros con vistas dinámicas.
- Contar con un backend REST para futuras versiones web o móviles.
- Dockerizar la aplicación completamente para facilitar su despliegue.
- Utilizar MongoDB Atlas para centralizar la base de datos en la nube.

8. *Bibliografía*

- https://www.stackscale.com/es/blog/bases-de-datos
- https://thedataschools.com/que-es/nosql/
- https://www.ibm.com/es-es/topics/nosql-databases
- https://openwebinars.net/blog/sql-vs-nosql-comparativa-para-elegir-correctamente/
- https://www.digitalocean.com/community/tutorials/what-is-docker-es
- https://www.redhat.com/es/topics/containers/what-is-docker
- https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/A_brief_history
- https://es.wikipedia.org/wiki/Historia_de_la_programaci%C3%B3n
- https://www.mongodb.com/es
- https://neo4j.com/es/
- https://www.openlibrary.org
- https://developers.google.com/books/
- https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html (Java y JavaFX)
- https://spring.io/projects/spring-boot (Spring Boot)
- https://reactjs.org/ (React)
- https://flutter.dev/ (Flutter)
- https://www.baeldung.com/spring-boot-export-data-to-excel-pdf
- https://docs.mongodb.com/manual/administration/backups/ (MongoDB Backup)
- https://formvalidation.io/ (Validación de formularios en interfaces)
- https://thedataschools.com/que-es/nosql/
- https://www.ibm.com/es-es/topics/nosql-databases
- https://openwebinars.net/blog/sql-vs-nosql-comparativa-para-elegir-correctamente/
- https://www.digitalocean.com/community/tutorials/what-is-docker-es
- https://www.redhat.com/es/topics/containers/what-is-docker
- https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/A_brief_history
- https://es.wikipedia.org/wiki/Historia_de_la_programaci%C3%B3n
- https://www.mongodb.com/es
- https://neo4j.com/es/
- https://www.openlibrary.org
- https://developers.google.com/books/