**Documentación  
Proyecto: Sistema de Gestion de Bibliotecas**

**Sebastian Velasco Ocampo**

**Prueba Técnica**

**INTEGRASOFT.SAS  
Neiva - Huila  
2025**

**INDICE**

[**Introducción.**](#_c5dpdqi49o9y) **2**

[**Justificación.**](#_wovtg55h1vyx) **3**

[**Objetivo General.**](#_rdsplilxfiw3) **3**

[**Objetivos Específicos.**](#_5tihxhov4tp) **3**

[**Marco Teórico.**](#_jqfvgm2w4dd5) **4**

[Patrón MVC](#_bi6huoauhg6h) 5

[Arquitectura limpia](#_bcdd9ut0sam8) 5

[CRUD](#_vwjrtse771tt) 5

[Gestión de información](#_5adyq2z1xlpy) 5

[**Marco Conceptual.**](#_k8611voajw40) **5**

[Sistema de Gestión de Bibliotecas](#_9gka4vhh4hmh) 6

[CRUD (Create, Read, Update, Delete)](#_ne43x1zb8su6) 6

[Autor](#_unkyoqcht6ml) 6

[Libro](#_wlfwmg8h7h3s) 6

[Biblioteca](#_9qm47wve3vbg) 6

[API REST](#_5xst9lkcvkqb) 6

[Validaciones](#_whshcbmtfk6k) 7

[Full-Stack](#_juqzc6tdooqi) 7

[**Metodología de Desarrollo.**](#_spf47fqa5c8s) **7**

[Fases seguidas en el desarrollo](#_ukgwcdbngly6) 8

[**Ingeniería de Requisitos.**](#_1ilsjub89ylh) **9**

[Matriz de trazabilidad de requisitos para un sistema de gestión de bibliotecas](#_w1dsj954etjl) 9

[**Diseño del sistema.**](#_w0mdthrguyxm) **11**

[Diagrama UML](#_gny6x0xt039) 11

[**Cronograma.**](#_d7bnm1773d57) **11**

[**Repositorio.**](#_vmjn0elrpm5c) **12**

[**Referencias Bibliográficas.**](#_i922tof9pv6g) **12**

## 

## 

## 

## 

## 

## **Introducción.**

El Sistema de Gestión de Bibliotecas es una aplicación web full-stack que permite administrar autores, libros y bibliotecas mediante operaciones CRUD completas. El proyecto fue desarrollado como parte de una prueba técnica para desarrolladores, con el fin de demostrar la implementación de una solución estructurada bajo el patrón MVC limpio.

La arquitectura del sistema está organizada en capas que separan la presentación, la lógica de negocio y el acceso a datos, lo que facilita la mantenibilidad y escalabilidad del software. Asimismo, el sistema incorpora validaciones, endpoints REST y una interfaz responsiva que cumple con los criterios establecidos en las historias de usuario planteadas en los requerimientos.

## **Justificación.**

La gestión eficiente de bibliotecas requiere herramientas digitales modernas que permitan organizar y relacionar correctamente la información. Sistemas tradicionales suelen ser limitados en cuanto a escalabilidad y usabilidad, dificultando la administración de entidades y sus relaciones.

El desarrollo de este sistema responde a la necesidad de contar con una solución que asegure un manejo estructurado y confiable de los datos. Además, cumple con los requerimientos de la prueba técnica al implementar funcionalidades clave de administración, validaciones robustas y buenas prácticas de desarrollo de software.

## **Objetivo General.**

Desarrollar un Sistema de Gestión de Bibliotecas como aplicación web full-stack que implemente operaciones CRUD completas para autores, libros y bibliotecas, aplicando principios de arquitectura limpia y cumpliendo con los criterios de aceptación establecidos en la prueba técnica.

## **Objetivos Específicos.**

-Modelar las entidades Autor, Libro y Biblioteca, definiendo correctamente sus relaciones (1:N y N:M).

-Implementar validaciones en frontend y backend para garantizar la integridad de los datos.

-Construir una API REST con endpoints que permitan la creación, consulta y eliminación de entidades.

-Desarrollar una interfaz de usuario clara, responsiva y accesible para la gestión del sistema.

-Aplicar principios de arquitectura MVC limpia para mantener separación de responsabilidades y orden en el código.

-Utilizar herramientas de control de versiones y documentar el proceso de instalación y ejecución del sistema.-Asegurar el cumplimiento de las historias de usuario y criterios de aceptación definidos en la prueba técnica.

## **Marco Teórico.**

El desarrollo de sistemas de información orientados a la gestión de bibliotecas se fundamenta en los principios de la **Ingeniería de Software**, la cual proporciona métodos, procesos y herramientas para construir aplicaciones confiables, mantenibles y escalables. Dentro de este contexto, la adopción de **arquitecturas limpias** y el **patrón Modelo-Vista-Controlador (MVC)** permiten separar responsabilidades, reduciendo la complejidad y facilitando la evolución del sistema.

Los sistemas de gestión de bibliotecas surgen como respuesta a la necesidad de digitalizar procesos que tradicionalmente se realizaban de forma manual, como el registro de autores, libros y bibliotecas. Estos sistemas han evolucionado para incluir relaciones más complejas, validaciones de integridad y consultas avanzadas, lo cual garantiza un acceso eficiente a la información.

Asimismo, el uso de **operaciones CRUD (Create, Read, Update, Delete)** constituye la base funcional de este tipo de aplicaciones, ya que proporcionan un mecanismo estándar para gestionar entidades. Adicionalmente, el desarrollo web moderno permite la implementación de estos sistemas bajo arquitecturas full-stack, combinando tecnologías de frontend y backend con el fin de ofrecer experiencias de usuario responsivas, accesibles y confiables.

La importancia de este marco teórico radica en establecer los fundamentos que sustentan el proyecto, destacando conceptos como:

### **Patrón MVC**

Separación de la lógica de presentación, negocio y datos.

### **Arquitectura limpia**

Diseño modular que promueve la mantenibilidad.

### **CRUD**

Operaciones básicas para la persistencia de datos.

### **Gestión de información**

Organización estructurada de entidades relacionadas.

## **Marco Conceptual.**

Para comprender el funcionamiento del Sistema de Gestión de Bibliotecas, es necesario definir los conceptos clave que lo estructuran:

### **Sistema de Gestión de Bibliotecas**

Aplicación informática diseñada para administrar y organizar los recursos de una biblioteca, incluyendo autores, libros y sus relaciones con las bibliotecas físicas o digitales.

### **CRUD (Create, Read, Update, Delete)**

Conjunto de operaciones fundamentales en la administración de datos que permiten crear, consultar, actualizar y eliminar registros.

### **Autor**

Entidad que representa a la persona responsable de la creación de una obra literaria, identificada por atributos como nombre y nacionalidad.

### **Libro**

Entidad que contiene información relacionada con el título, año de publicación, autor y biblioteca asociada.

### **Biblioteca**

Entidad que agrupa y organiza los libros, definida por un nombre y ubicación.

### **API REST**

Interfaz que permite la comunicación entre cliente y servidor mediante operaciones estándar como GET, POST, PUT y DELETE.

**Patrón MVC**

Modelo arquitectónico que divide el sistema en tres componentes principales: Modelo (datos), Vista (interfaz de usuario) y Controlador (lógica de negocio).

### **Validaciones**

Reglas implementadas en formularios y API para garantizar que los datos ingresados sean correctos, completos y consistentes.

### **Full-Stack**

Enfoque de desarrollo que integra tanto las tecnologías de frontend como las de backend en una misma aplicación.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **Metodología de Desarrollo.**

El desarrollo del Sistema de Gestión de Bibliotecas se basó en un enfoque **ágil iterativo-incremental**, aplicando prácticas de **Scrum** adaptadas al contexto de la prueba técnica. Este enfoque fue seleccionado debido a su compatibilidad con arquitecturas modulares como **MVC limpio**, lo que permitió dividir el sistema en capas independientes (presentación, lógica de negocio y datos) y avanzar de forma progresiva en la implementación.

### **Fases seguidas en el desarrollo**

1. **Levantamiento de Requisitos:** Identificación de las historias de usuario y criterios de aceptación proporcionados en la prueba técnica.
2. **Diseño Arquitectónico:** Aplicación del patrón **Modelo-Vista-Controlador (MVC)** para separar responsabilidades, junto con una **arquitectura por capas** para garantizar mantenibilidad y escalabilidad.
3. **Implementación:  
   Frontend:** Next.js 15 (React 18) + TypeScript + Tailwind CSS.

**Backend:** Next.js API Routes con servicios y validaciones.

**Base de datos simulada:** In-memory database (con proyección a integrar PostgreSQL o MySQL).

1. **Validación y Pruebas:** Desarrollo de pruebas manuales en cada iteración para validar funcionalidades CRUD, validaciones de formularios y consultas API.
2. **Entrega y Documentación:** Preparación de la documentación técnica siguiendo estándares de claridad, diseño arquitectónico y referencias APA.  
   Este enfoque permitió construir un sistema modular, escalable y fácil de mantener, cumpliendo con el **100% de las historias de usuario** planteadas.

## **Ingeniería de Requisitos.**

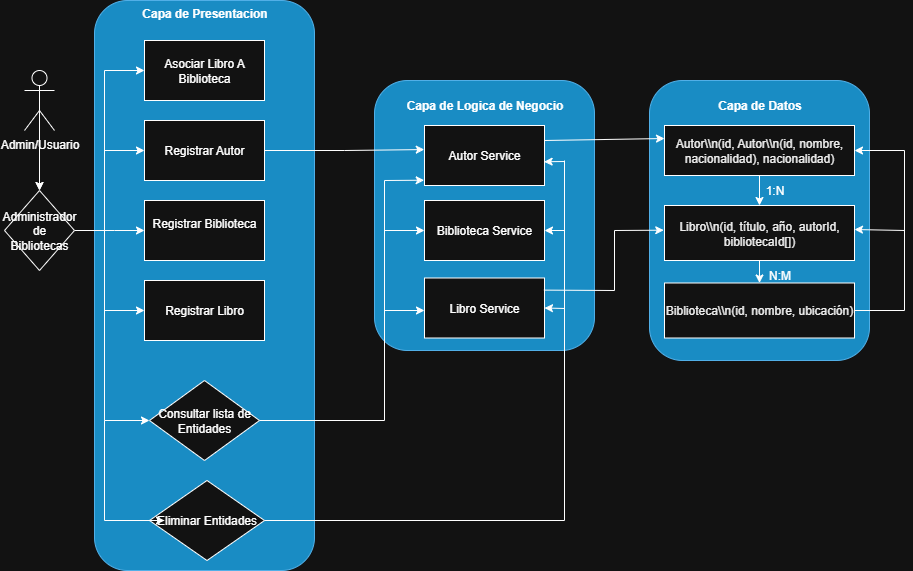
### Matriz de trazabilidad de requisitos para un sistema de gestión de bibliotecas

| ID- Req | Descripción | Tipo | Caso de uso asociado | Implementacion | Estado |
| --- | --- | --- | --- | --- | --- |
| RF-01 | | Registrar autores con nombre y nacionalidad | | --- | | Funcional | | Registrar Autor | | --- | | | Formulario + API /api/autors | | --- | | Activo |
| RF-02 | Registrar libros con título, año y autor asociado | Funcional | | Registrar Libro | | --- | | Formulario + API /api/libros | Activo |
| RF-03 | Registrar bibliotecas con nombre y ubicación | Funcional | | Registrar Biblioteca | | --- | | | Formulario + API /api/bibliotecas | | --- | | Activo |
| RF-04 | Asociar libros a bibliotecas | Funcional | | Asociar Libro a Biblioteca | | --- | | Selector en formulario de libros | Activo |
| RF-05 | Consultar listas de autores, libros y bibliotecas | Funcional | | Consultar Listas | | --- | | | Componentes List con SWR | | --- | | Activo |
| RF-06 | Eliminar entidades (autor, libro, biblioteca) | Funcional | | Eliminar Entidades | | --- | | | Botones de eliminación en cada entidad | | --- | | Activo |
| RNF-01 | |  | | --- |  | Interfaz responsiva y accesible | | --- | | No funcional | - | |  | | --- |  | Diseño con Tailwind CSS + WCAG | | --- | | Activo |
| RNF-02 | Arquitectura modular y mantenible | No funcional | - | | Patrón MVC Limpio implementado | | --- | | Activo |
| RNF-03 | | Escalabilidad del sistema | | --- | | No funcional | - | | Base In-Memory con posibilidad de escalar hacia PostgreSQL/MongoDB en entornos Node.js o integrar librerías React para persistencia distribuida | | --- | | Activo |
| RNF-04 | | Validación de datos en frontend y backend | | --- | | No funcional | - | | TypeScript + validaciones en formularios y API | | --- | | Activo |
| RNF-05 | | Compatibilidad con Node.js 18+ y navegadores modernos | | --- | | No funcional | - | | Probado en entorno local | | --- | | Activo |

## 

## **Diseño del sistema.**

### Diagrama UML



## **Cronograma.**

## 

Enlace:  
<https://docs.google.com/spreadsheets/d/1Ino3RLtOE3mu_yLyHSady5AjfqSCH5tFiY0Sk1OwWS4/edit?usp=sharing>

## **Repositorio.**

## **Referencias Bibliográficas.**

Tecnologías Utilizadas

1. Next.js Documentation. (2024). *Next.js 15 - The React Framework for Production*. Vercel. [https://nextjs.org/docs](https://nextjs.org/docs?utm_source=chatgpt.com)
2. TypeScript Handbook. (2024). *TypeScript Documentation*. Microsoft. [https://www.typescriptlang.org/docs/](https://www.typescriptlang.org/docs/?utm_source=chatgpt.com)
3. React Documentation. (2024). *React - A JavaScript library for building user interfaces*. Meta. [https://react.dev/](https://react.dev/?utm_source=chatgpt.com)
4. Tailwind CSS Documentation. (2024). *Tailwind CSS - A utility-first CSS framework*. Tailwind Labs. [https://tailwindcss.com/docs](https://tailwindcss.com/docs?utm_source=chatgpt.com)
5. Radix UI Documentation. (2024). *Radix UI - Low-level UI primitives*. WorkOS. [https://www.radix-ui.com/](https://www.radix-ui.com/?utm_source=chatgpt.com)
6. Node.js Documentation. (2024). *Node.js v18 Documentation*. Node.js Foundation. [https://nodejs.org/docs/](https://nodejs.org/docs/?utm_source=chatgpt.com)
7. npm Documentation. (2024). *npm Docs*. npm, Inc. [https://docs.npmjs.com/](https://docs.npmjs.com/?utm_source=chatgpt.com)
8. Git Documentation. (2024). *Git Reference*. Software Freedom Conservancy. [https://git-scm.com/docs](https://git-scm.com/docs?utm_source=chatgpt.com)
9. Vercel Documentation. (2024). *Vercel Platform Documentation*. Vercel. [https://vercel.com/docs](https://vercel.com/docs?utm_source=chatgpt.com)

Arquitectura y Buenas Prácticas

1. Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
2. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
3. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.
4. Clean Code Blog. (2012). *The Clean Architecture*. Robert C. Martin. [https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html](https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html?utm_source=chatgpt.com)

Bases de Datos y Modelado

1. Date, C. J. (2003). *An Introduction to Database Systems*. Addison-Wesley.
2. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson.
3. Database Design Tutorial. (2024). *Entity Relationship Diagrams*. Lucidchart. [https://www.lucidchart.com/pages/er-diagrams](https://www.lucidchart.com/pages/er-diagrams?utm_source=chatgpt.com)

Desarrollo Web y Estándares

1. MDN Web Docs. (2024). *Web APIs*. Mozilla. [https://developer.mozilla.org/en-US/docs/Web/API](https://developer.mozilla.org/en-US/docs/Web/API?utm_source=chatgpt.com)
2. REST API Tutorial. (2024). *RESTful Web Services*. REST API Tutorial. [https://restfulapi.net/](https://restfulapi.net/?utm_source=chatgpt.com)
3. Web Content Accessibility Guidelines (WCAG) 2.1. (2018). W3C. [https://www.w3.org/WAI/WCAG21/Understanding/](https://www.w3.org/WAI/WCAG21/Understanding/?utm_source=chatgpt.com)

## 