REPÚBLICA BOLIVARIANA DE VENEZUELA

MINISTERIO DEL PODER POPULAR PARA LA DEFENSA

UNIVERSIDAD NACIONAL EXPERIMENTAL POLITÉCNICA

DE LA FUERZA ARMADA NACIONAL BOLIVARIANA

UNEFA

NUCLEO CARABOBO-EXTENSION GUACARA.

SISTEMA BIBLIOTECA

Profesor	**Alumno:**	
Ing. Alonso	José Álvarez	C.I.V- 31.100.269
Materia: Lenguaje 2		
V Ing. Sistema		
Sección: 05S-2629 D1		

Guacara,14/11/2025

1. Análisis de Requerimientos Funcionales (ARF)

Módulo de Autenticación y Registro (Seguridad)

Requerimiento Funcional	Descripción	Actor Principal
Registro de Usuario	El sistema debe permitir a un nuevo usuario (rol user) registrarse proporcionando nombre, cédula, email y una contraseña segura.	Usuario
Inicio de Sesión	El sistema debe validar las credenciales (email y contraseña) de cualquier usuario (user o admin) y establecer una sesión.	Usuario, Administrador
Cierre de Sesión	El sistema debe permitir a cualquier usuario con sesión activa finalizarla, eliminando las variables de sesión.	Usuario, Administrador
Restricción de Acceso	El sistema debe aplicar restricciones por rol: Solo el admin puede acceder a vistas de gestión (CRUD de Libros, Autores, Solicitudes).	Sistema

Módulo de Lógica de Negocio (Biblioteca)

Requerimiento Funcional	Descripción	Actor Principal
Listado de Libros	El sistema debe mostrar un catálogo completo de libros disponibles a cualquier visitante.	Todos
Solicitud de Préstamo	El usuario (user) debe poder enviar una solicitud de préstamo para un libro específico desde el listado.	Usuario
Consulta de Préstamos	El usuario (user) debe poder ver el estado (En espera, Completado, Rechazado) de todas sus solicitudes en la vista "Mis Préstamos".	Usuario
Gestión de Préstamos	El administrador (admin) debe poder ver todas las solicitudes de préstamo y actualizar su estado (Completado, Rechazado, en espera, Sin disponibilidad).	Administrador
Gestión de Libros (CRUD)	El administrador (admin) debe poder crear, leer, actualizar y eliminar libros del catálogo.	Administrador
Gestión de Autores (CRUD)	El administrador (admin) debe poder crear, leer, actualizar y eliminar autores.	Administrador

2. Normalización de la Base de Datos

La normalización se basa en la estructura de las tablas creadas (users, book, author, loan).

Tablas: users, author, book, loan.

Primera Forma Normal (1FN)

- Todas las tablas cumplen la 1FN. Se utiliza una tabla de enlace implícita para la relación N:M entre book y author.

Segunda Forma Normal (2FN)

Regla: Debe estar en 1FN y todos los atributos no clave deben depender completamente de **toda** la clave primaria. Esto aplica principalmente a tablas con claves compuestas.

- **Tabla de Libros (book)**: Si book_id es la única clave primaria, 2FN se cumple.
- **Identificación de la relación N:M entre Libro y Autor**: Si un libro puede tener muchos autores y un autor muchos libros, la relación *debe* normalizarse a 2FN mediante una tabla de enlace:

Tabla	Clave Primaria Compuesta	Atributos No Clave	Dependencia Funcional
book_author (Nueva)	(book_id, author_id)	N/A	author_name dependería solo de author_id (Violación 2FN si estuviera aquí).

- **Conclusión:** Tu esquema **necesitaría una tabla de enlace** (book_author) para modelar correctamente la relación entre Libros y Autores, si fuera N:M. Si asumes que un libro tiene *un solo* autor (como lo sugiere book.author_id), 2FN se cumple en tu diseño actual.

Tercera Forma Normal (3FN)

Regla: Debe estar en 2FN y no debe contener dependencias transitivas

Tabla	Dependencia Transitiva (Ejemplo de Violación)	Mi base de datos cumple con lo siguiente :
users	city depende de zip_code, que no es la clave primaria.	**Cumplida** (Solo almacena datos de usuario directamente relacionados con user_id).
book	Si author_name estuviera aquí, dependería de author_id.	**Cumplida** (La información del autor está en author).
loan	Si book_title estuviera aquí, dependería de book_id.	**Cumplida** (Solo almacena book_id, user_id y status).

3. Documentación del Código (PHPDoc Style)

Documentación de Clases y Propiedades

Archivo/Clase	Elemento	Descripción
models/Loan.php	class Loan	Modela la entidad de la solicitud de préstamo. Interactúa con la tabla loan para crear, leer y actualizar el estado de las solicitudes.
models/Loan.php	$conn	Objeto de conexión a la base de datos (PDO).
models/Loan.php	$loan_id, $book_id, $user_id, $status	Propiedades que mapean las columnas de la tabla loan.
controllers/LoanController.php	class LoanController	Maneja la lógica de negocio relacionada con las solicitudes de préstamo (creación de solicitudes, gestión de estado y visualización de préstamos).
index.php	$base_path	Ruta base de la aplicación (e.g., /biblioteca), utilizada para generar URLs absolutas.
index.php	$route_parts	Array resultante de dividir la URI, usado para manejar rutas con parámetros (/loan/request/3).

Documentación de Métodos y Funciones

Clase	Método	Descripción	Parámetros	Retorno
Loan	__construct()	Constructor que inyecta la conexión a la base de datos (PDO).	PDO $db: Objeto de conexión.	N/A
Loan	createRequest()	Registra una nueva solicitud de préstamo en estado 'en_espera'.	N/A (Usa $this->book_id y $this->user_id).	bool: true si se ejecuta correctamente, false si falla (e.g., solicitud duplicada).
Loan	readUserRequests()	Obtiene todas las solicitudes de préstamo realizadas por un usuario específico.	int $user_id: ID del usuario logueado.	PDOStatement: Resultado de la consulta.
Loan	readAllRequests()	Obtiene todas las solicitudes de préstamo con información de libro y usuario para la	N/A	PDOStatement: Resultado de la consulta.

		vista de gestión del Administrador.		
Loan	updateStatus()	Actualiza el estado de una solicitud (en_espera, completado, etc.).	int $loan_id, string $status: Nuevo estado.	bool: true si la actualización fue exitosa.
LoanController	index()	**[ADMIN]** Muestra la lista de gestión de préstamos. Procesa la actualización de estado vía POST.	string $method: Método HTTP ('GET' o 'POST').	void
LoanController	myLoans()	**[USER]** Muestra la vista con todas las solicitudes de préstamo realizadas por el usuario actual.	N/A	void
LoanController	request()	**[USER]** Procesa la solicitud de préstamo de un libro específico y redirige al listado.	int $book_id: ID del libro a solicitar.	void (Redirección)
index.php	**Route /loan**	Ruta de gestión de solicitudes (solo para admin). Llama a LoanController->index().	N/A	N/A
index.php	**Route /myloans**	Ruta de visualización de solicitudes (solo para user). Llama a LoanController->myLoans().	N/A	N/A
index.php	**Route /loan/request/{id}**	Ruta para ejecutar la creación de una solicitud de préstamo por parte del usuario. Llama a LoanController->request().	N/A	N/A

Documentación General del Sistema

Elemento	Descripción
Arquitectura	El sistema utiliza el patrón **Modelo-Vista-Controlador (MVC)**, implementado a través de un **Front Controller** (index.php) que maneja todas las solicitudes (routing).
Controlador	El LoanController y otros controladores gestionan la lógica de la aplicación y la interacción entre el modelo y la vista.
Modelo	Los modelos (Loan, Book, Author, etc.) interactúan directamente con la base de datos a través de PDO, manejando la lógica de persistencia de datos.
Vista	Las vistas (archivos en views/) contienen la estructura HTML y PHP mínima para mostrar los datos proporcionados por el controlador.
Seguridad	La autenticación y la autorización por rol (isAdmin(), isLoggedIn()) son manejadas por el AuthController y verificadas en el *Router* y los controladores para restringir el acceso a funcionalidades sensibles.
Flujo de Préstamos	Un usuario normal (user) solicita un libro. La solicitud se registra como 'en_espera'. El Administrador (admin) gestiona y actualiza el estado de la solicitud.