Ejercicio 18: Sistema de Gestión de Biblioteca

PHP + MySQL + JavaScript - Proyecto en Equipo

📋 Índice

---

1. Introducción y Contexto

1.1 Descripción del Proyecto

Desarrollarán un Sistema de Gestión de Biblioteca que permita administrar el préstamo y devolución de libros. El sistema debe controlar el inventario de libros, los usuarios/socios de la biblioteca, y llevar un registro completo de los préstamos realizados.

Este proyecto es una evolución natural de los ejercicios anteriores (CRUD de Productos y Sistema de Estudiantes), agregando:

- Relaciones más complejas entre entidades
- Lógica de negocio con validaciones
- Manejo de estados y fechas
- Control de disponibilidad y restricciones

1.2 Objetivos Pedagógicos

Al finalizar este proyecto, los alumnos habrán practicado:

✅ Bases de Datos:

- Diseño de bases de datos relacionales
- Relaciones 1:N (uno a muchos)
- Consultas con JOINs
- Actualización de estados

✅ Backend (PHP):

- Validaciones complejas de negocio
- Manejo de sesiones y autenticación
- Cálculo y manipulación de fechas
- Transacciones de datos

✅ Frontend (JavaScript):

- Validaciones del lado del cliente
- Búsquedas dinámicas con AJAX
- Actualización de interfaz sin recargar
- Feedback visual al usuario

✅ Trabajo en Equipo:

- División de tareas
- Integración de módulos
- Uso de control de versiones (Git)
- Comunicación y resolución de problemas

1.3 Modalidad de Trabajo

- Equipos: 2 personas
- Duración: 2 semanas
- Enfoque: Monolítico híbrido (backend y frontend integrados)
- Frameworks: No se permite el uso de frameworks, pero pueden usar librerías específicas (ej: para QR, PDF, etc.)
- Opción avanzada: Los equipos que lo deseen pueden separar backend (API REST) y frontend (SPA), pero no es obligatorio

1.4 Cronograma Sugerido

Semana 1:

- Días 1-2: Diseño de BD, estructura de archivos, sistema de login

- Días 3-4: CRUD de libros y usuarios
- Día 5: Integración y pruebas básicas

Semana 2:

- Días 1-2: Módulo de préstamos y devoluciones
- Días 3-4: Dashboard, validaciones y funcionalidades opcionales
- Día 5: Testing, documentación y preparación de la presentación

---

2. Especificación Funcional

2.1 Funcionalidades OBLIGATORIAS (Core)

2.1.1 Sistema de Autenticación

Login:

- Formulario de inicio de sesión (usuario/email y contraseña)
- Validación de credenciales contra la base de datos
- Creación de sesión al autenticar correctamente
- Redirección al dashboard
- Mensaje de error si las credenciales son incorrectas

Logout:

- Destruir sesión activa
- Redirección al login

Protección de páginas:

- Todas las páginas del sistema (excepto login) deben verificar que existe una sesión activa
- Si no hay sesión, redirigir al login

2.1.2 Gestión de Libros

Listar Libros:

- Mostrar todos los libros en una tabla
- Columnas: ID, Título, Autor, ISBN, Editorial, Año, Categoría, Estado
- Indicador visual del estado (disponible/prestado)
- Buscador básico (por título o autor)
- Paginación (opcional pero recomendada)

Agregar Libro:

- Formulario con campos:
    - Título (obligatorio)
    - Autor (obligatorio)
    - ISBN (obligatorio, único)
    - Editorial
    - Año de publicación
    - Categoría (texto libre por ahora)
    - Descripción/sinopsis (opcional)
- Validaciones en cliente y servidor
- Al crear, el estado inicial es "disponible"

Editar Libro:

- Formulario pre-cargado con los datos actuales
- Permitir modificar todos los campos excepto el estado
- El estado solo se modifica mediante préstamos/devoluciones

Eliminar Libro:

- Confirmación antes de eliminar
- NO permitir eliminar si el libro está prestado actualmente
- Mostrar mensaje de error si se intenta eliminar un libro prestado

2.1.3 Gestión de Usuarios/Socios

Listar Usuarios:

- Tabla con: ID, Nombre completo, Email, Teléfono, Estado, Préstamos activos
- Buscador por nombre o email
- Indicador visual de estado (activo/suspendido)

Agregar Usuario:

- Formulario con:
    - Nombre completo (obligatorio)
    - Email (obligatorio, único)
    - Teléfono
    - Dirección
    - DNI/Documento (obligatorio, único)
    - Fecha de registro (automática)
- Estado inicial: "activo"

Editar Usuario:

- Modificar datos personales
- Cambiar estado (activo/suspendido)

Ver Detalle de Usuario:

- Datos personales
- Préstamos activos (con detalle)
- Historial de préstamos (últimos 10)

Eliminar Usuario:

- Confirmación previa
- NO permitir eliminar si tiene préstamos activos
- Considerar marcar como "inactivo" en lugar de eliminar

2.1.4 Gestión de Préstamos

Registrar Préstamo:

- Formulario/interfaz con:
    - Búsqueda y selección de libro (debe estar disponible)
    - Búsqueda y selección de usuario (debe estar activo)
    - Fecha de préstamo (por defecto: hoy)
    - Fecha de devolución esperada (automática: +14 días)
    - Observaciones (opcional)

Validaciones al prestar:

1. El libro debe estar disponible
2. El usuario debe estar activo (no suspendido)
3. El usuario NO debe tener préstamos vencidos
4. El usuario NO debe exceder el límite de préstamos simultáneos (máximo 3)
5. Si alguna validación falla, mostrar mensaje claro

Al confirmar el préstamo:

- Cambiar estado del libro a "prestado"
- Registrar en tabla de préstamos con estado "activo"
- Mostrar confirmación con los datos del préstamo

Listar Préstamos Activos:

- Tabla con: ID, Libro, Usuario, Fecha préstamo, Fecha devolución esperada, Días restantes
- Indicador visual si está vencido (rojo) o próximo a vencer (amarillo)

- Filtros: todos/solo vencidos/próximos a vencer
- Acción: botón "Registrar Devolución"

2.1.5 Gestión de Devoluciones

Registrar Devolución:

- Desde la lista de préstamos activos
- Al hacer clic en "Registrar Devolución":
    - Registrar fecha de devolución real (hoy)
    - Cambiar estado del préstamo a "devuelto"
    - Cambiar estado del libro a "disponible"
    - Si tiene atraso, calcular y mostrar días de demora
    - Confirmación de devolución exitosa

Historial de Préstamos:

- Ver todos los préstamos (activos y devueltos)
- Filtros por: usuario, libro, fecha, estado
- Exportar (opcional)

2.1.6 Dashboard Principal

Página inicial después del login con:

Estadísticas generales:

- Total de libros en biblioteca
- Libros disponibles
- Libros prestados
- Total de usuarios activos
- Total de préstamos activos
- Préstamos vencidos (destacado)

Secciones:

- Alertas de préstamos vencidos (tabla con acciones rápidas)
- Préstamos próximos a vencer (3 días o menos)
- Últimos préstamos realizados (5 más recientes)
- Libros más prestados (top 5)

Accesos rápidos:

- Botones para: Nuevo Préstamo, Ver Libros, Ver Usuarios

2.2 Funcionalidades OPCIONALES (Extras por Nivel)

Los equipos que completen el core obligatorio pueden implementar funcionalidades adicionales para aumentar su calificación:

Nivel 1 - Extensiones Simples (1-2 puntos extra c/u)

Categorías de Libros:

- Tabla adicional de categorías
- CRUD de categorías
- Asignar categoría al libro (relación)
- Filtrar libros por categoría

Búsqueda Avanzada:

- Múltiples criterios: título, autor, ISBN, categoría
- Búsqueda con AJAX sin recargar página
- Autocompletado en campos de búsqueda

Alertas Visuales:

- Dashboard con tarjetas de colores según estado
- Notificaciones badge con números (préstamos vencidos)
- Toast/alertas para acciones exitosas/fallidas

Historial Completo por Usuario:

- Ver todos los préstamos históricos de un usuario
- Estadísticas: total prestado, promedio de días, libros favoritos

Ranking de Libros:

- Top 10 libros más prestados
- Gráfico de barras simple (opcional)
- Por período: mes, año, histórico

Nivel 2 - Funciones Intermedias (3-4 puntos extra c/u)

Sistema de Reservas:

- Reservar un libro que está prestado
- Cola de reservas (FIFO)

- Al devolver, notificar al siguiente en la cola
- Usuario no puede pedir prestado el mismo libro si ya lo tiene reservado

Sistema de Multas:

- Calcular multa automática por día de atraso
- Configurar monto por día (ej: $50)
- Registro de multas pendientes
- Marcar multa como pagada
- Usuario no puede pedir prestado si tiene multas pendientes

Código QR para Libros:

- Generar QR único por libro (con ISBN o ID)
- Usar librería PHP QR Code
- Imprimir QR para pegar en libro físico
- Escanear QR para ver detalles del libro (opcional)

Exportar Reportes:

- Listado de préstamos a CSV
- Reporte de libro a PDF (con datos y estadísticas)
- Reporte de usuario a PDF (historial de préstamos)
- Usar librerías como TCPDF o FPDF

Notificaciones por Email:

- Recordatorio 2 días antes del vencimiento
- Notificación de préstamo vencido
- Confirmación de préstamo por email
- Usar PHPMailer o similar

Nivel 3 - Desafíos Avanzados (5-6 puntos extra c/u)

Sistema de Roles y Permisos:

- Rol "Admin": acceso completo
- Rol "Bibliotecario": no puede eliminar, solo gestionar préstamos
- Tabla de usuarios del sistema (diferente a socios)
- Permisos granulares por acción

Renovación de Préstamos:

- Extender plazo de devolución (+7 días)
- Solo si no hay reservas pendientes

- Límite de renovaciones (máximo 2 veces)
- Registro de renovaciones

Dashboard con Gráficos:

- Usar Chart.js
- Gráfico de préstamos por mes
- Gráfico de categorías más prestadas
- Estadísticas visuales en tiempo real

API REST:

- Separar completamente backend y frontend
- Endpoints JSON para todas las operaciones
- Autenticación JWT o similar
- Frontend consume API vía AJAX
- Documentación de API con ejemplos

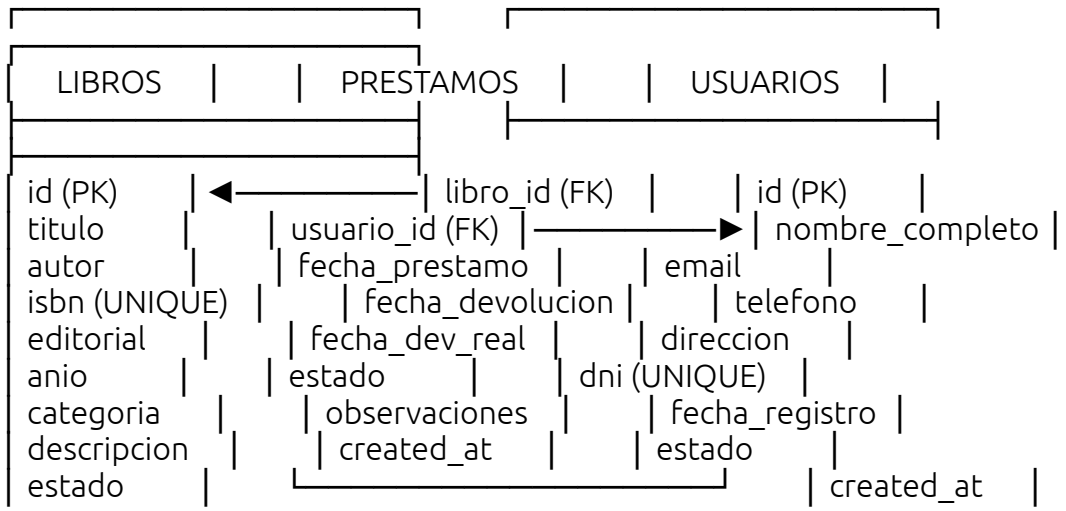
Gestión de Múltiples Copias:

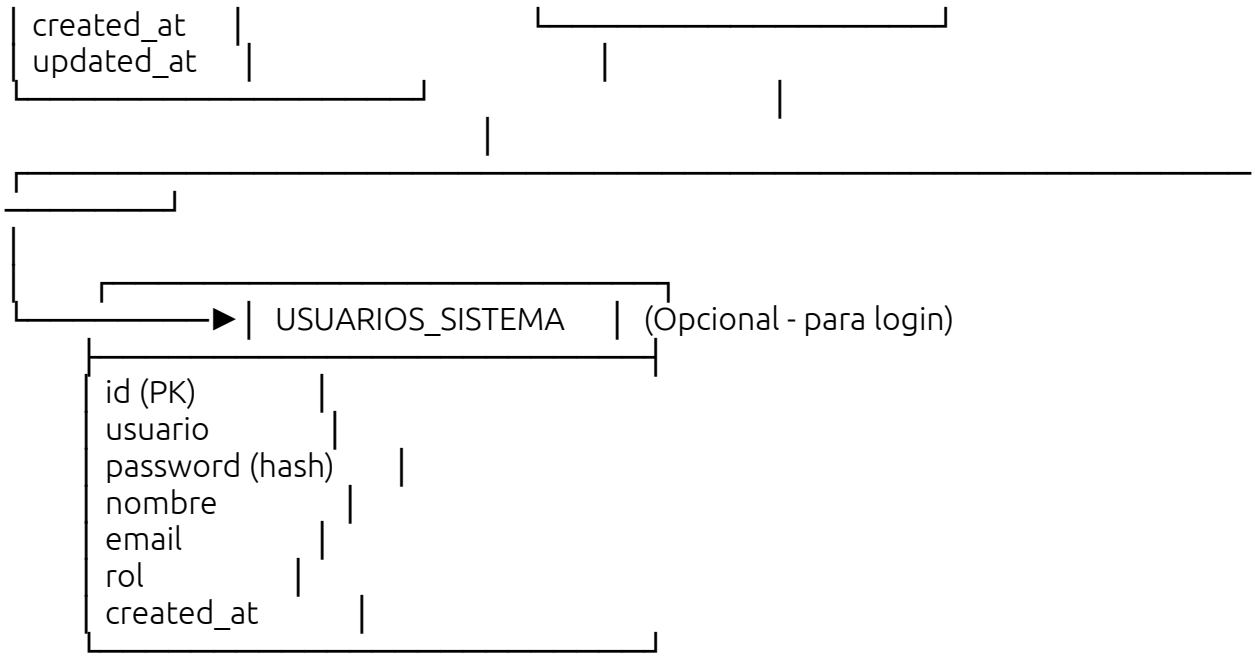
- Un libro puede tener múltiples ejemplares físicos
- Tabla adicional: ejemplares (con código único)
- Control de estado por ejemplar
- Préstamo de un ejemplar específico

---

3. Estructura de Base de Datos

3.1 Diagrama de Entidad-Relación (Modelo Conceptual)





3.2 Estructura de Tablas

Tabla: libros

Campo	Tipo	Restricciones
id	INT	PRIMARY KEY, AUTO_INCREMENT
titulo	VARCHAR(200)	NOT NULL
autor	VARCHAR(150)	NOT NULL
isbn	VARCHAR(20)	NOT NULL, UNIQUE
editorial	VARCHAR(100)	NULL
anio	YEAR	NULL
categoria	VARCHAR(50)	NULL
descripcion	TEXT	NULL
estado	ENUM	'disponible','prestado' DEFAULT 'disponible'
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Notas:

- estado solo tiene dos valores en este modelo simplificado
- categoria es texto libre (en opcionales pueden crear tabla de categorías)
- isbn debe ser único para evitar duplicados

Tabla: usuarios

Campo	Tipo	Restricciones
id	INT	PRIMARY KEY, AUTO_INCREMENT
nombre_completo	VARCHAR(150)	NOT NULL
email	VARCHAR(100)	NOT NULL, UNIQUE
telefono	VARCHAR(20)	NULL
direccion	VARCHAR(200)	NULL
dni	VARCHAR(20)	NOT NULL, UNIQUE
fecha_registro	DATE	DEFAULT CURRENT_DATE
estado	ENUM	'activo','suspendido' DEFAULT 'activo'
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Notas:

- estado permite suspender usuarios sin eliminarlos
- email y dni únicos para evitar duplicados
- No confundir con usuarios del sistema (login)

Tabla: prestamos

Campo	Tipo	Restricciones
id	INT	PRIMARY KEY, AUTO_INCREMENT
libro_id	INT	NOT NULL, FOREIGN KEY (libros.id)
usuario_id	INT	NOT NULL, FOREIGN KEY (usuarios.id)
fecha_prestamo	DATE	NOT NULL, DEFAULT CURRENT_DATE
fecha_devolucion	DATE	NOT NULL (calculada: +14 días)
fecha_dev_real	DATE	NULL (NULL mientras está activo)
estado	ENUM	'activo','devuelto','vencido' DEFAULT 'activo'
observaciones	TEXT	NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Notas:

- fecha_devolucion es la fecha esperada de devolución (calculada automáticamente)
- fecha_dev_real se completa cuando se devuelve el libro
- estado se actualiza automáticamente:
    - 'activo': préstamo en curso
    - 'devuelto': libro ya devuelto
    - 'vencido': fecha_devolucion < hoy y aún no devuelto
- Las claves foráneas aseguran integridad referencial

Tabla: usuarios_sistema (Para login - Obligatoria)

Campo	Tipo	Restricciones
id	INT	PRIMARY KEY, AUTO_INCREMENT
usuario	VARCHAR(50)	NOT NULL, UNIQUE
password	VARCHAR(255)	NOT NULL (almacenar hash)
nombre	VARCHAR(100)	NOT NULL
email	VARCHAR(100)	NOT NULL, UNIQUE
rol	ENUM	'admin','bibliotecario' DEFAULT 'bibliotecario'
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Notas:

- Esta tabla es para los usuarios que operan el sistema (bibliotecarios)
- NO confundir con la tabla usuarios (socios que piden libros prestados)
- La contraseña DEBE almacenarse como hash (usar password_hash() en PHP)
- Campo rol es opcional pero útil para futuras extensiones

3.3 Relaciones Explicadas

libros ← prestamos:

- Un libro puede tener muchos préstamos (historial)
- Un préstamo pertenece a un solo libro
- Relación: 1:N (uno a muchos)
- Restricción: No eliminar libro si tiene préstamos activos

usuarios ← prestamos:

- Un usuario puede tener muchos préstamos (historial)
- Un préstamo pertenece a un solo usuario
- Relación: 1:N (uno a muchos)
- Restricción: No eliminar usuario si tiene préstamos activos

3.4 Índices Recomendados

Para mejorar el rendimiento de las consultas:

```
-- Índices en libros
INDEX idx_isbn (isbn)
INDEX idx_estado (estado)
INDEX idx_titulo (titulo)

-- Índices en usuarios
```

INDEX idx_email (email)
INDEX idx_dni (dni)
INDEX idx_estado (estado)

-- Índices en prestamos
INDEX idx_libro_id (libro_id)
INDEX idx_usuario_id (usuario_id)
INDEX idx_estado (estado)
INDEX idx_fecha_devolucion (fecha_devolucion)

3.5 Datos de Prueba Sugeridos

Para facilitar el testing, creen al menos:

Usuarios del Sistema (login):

- admin / admin123 (rol: admin)
- biblio / biblio123 (rol: bibliotecario)

Libros (20-30 sugeridos):

- Mix de disponibles y prestados
- Diferentes categorías: ficción, educación, ciencia, historia, etc.
- Con datos reales (pueden usar libros conocidos)

Socios/Usuarios (10 sugeridos):

- Mix de activos y suspendidos
- Con diferentes cantidades de préstamos

Préstamos (15-20 sugeridos):

- Algunos activos (devueltos = NULL)
- Algunos devueltos (fecha_dev_real completada)
- Algunos vencidos (fecha_devolucion < hoy y no devueltos)
- Distribuidos entre usuarios

Ejemplo de libros para cargar:

1. Cien años de soledad - Gabriel García Márquez
2. 1984 - George Orwell
3. El principito - Antoine de Saint-Exupéry
4. Don Quijote de la Mancha - Miguel de Cervantes
5. Harry Potter y la piedra filosofal - J.K. Rowling ... (continuar con más)

---

4. Arquitectura y Estructura de Archivos

4.1 Enfoque: Monolítico Híbrido

El sistema será monolítico (backend y frontend integrados) pero con separación de responsabilidades para mantener el código organizado y facilitar el trabajo en equipo.

4.2 Estructura de Carpetas Sugerida

```
biblioteca/
│
├── index.php              # Redirige al login o dashboard
├── login.php              # Página de inicio de sesión
├── logout.php             # Cierre de sesión
├── dashboard.php          # Página principal (post-login)
│
├── config/
│   ├── database.php       # Conexión a BD
│   └── config.php         # Configuraciones generales
│
├── includes/
│   ├── header.php         # Header común (nav, meta)
│   ├── footer.php         # Footer común
│   └── auth.php           # Validación de sesión
│
├── libros/
│   ├── index.php          # Listar libros
│   ├── crear.php          # Formulario nuevo libro
│   ├── editar.php         # Formulario editar libro
│   ├── eliminar.php       # Eliminar libro (confirmación)
│   └── buscar.php         # Búsqueda AJAX (opcional)
│
├── usuarios/
│   ├── index.php          # Listar usuarios
│   ├── crear.php          # Formulario nuevo usuario
│   ├── editar.php         # Formulario editar usuario
│   ├── detalle.php        # Ver perfil y préstamos del usuario
│   └── eliminar.php       # Eliminar usuario
│
├── prestamos/
│   ├── index.php          # Listar préstamos activos
│   ├── nuevo.php          # Registrar nuevo préstamo
│   ├── devolver.php       # Registrar devolución
│   └── historial.php      # Ver historial completo
```

```
│   └── ajax_validar.php        # Validaciones AJAX
│
├── api/                    # (Opcional - para requests AJAX)
│   ├── libros.php              # Endpoints JSON de libros
│   ├── usuarios.php            # Endpoints JSON de usuarios
│   └── prestamos.php           # Endpoints JSON de préstamos
│
├── assets/
│   ├── css/
│   │   ├── style.css           # Estilos principales
│   │   └── dashboard.css       # Estilos del dashboard
│   ├── js/
│   │   ├── main.js             # Scripts generales
│   │   ├── validaciones.js     # Validaciones del cliente
│   │   └── busquedas.js        # Búsquedas con AJAX
│   └── img/
│       └── logo.png            # Logo de la biblioteca
│
├── sql/
│   ├── estructura.sql          # Estructura de tablas
│   └── datos_prueba.sql        # Datos de ejemplo
│
└── docs/
    ├── README.md               # Instrucciones de instalación
    ├── division_tareas.md      # Quién hizo qué
    └── presentacion.pdf        # Slides de la presentación
```

4.3 Separación de Responsabilidades

config/database.php - Conexión a BD

```
<?php
// Configuración de conexión (estructura sugerida)
define('DB_HOST', 'localhost');
define('DB_NAME', 'biblioteca');
define('DB_USER', 'root');
define('DB_PASS', '');

// Función de conexión que devuelve PDO o MySQLi
// Usar prepared statements siempre
```

includes/auth.php - Protección de páginas

```
<?php
// Verificar que existe sesión activa
// Si no existe, redirigir a login.php
// Incluir en todas las páginas protegidas
```

includes/header.php - Header común

```
<!DOCTYPE html>
<html>
<head>
   <!-- Meta tags, CSS, título -->
</head>
<body>
   <!-- Navbar con opciones según usuario logueado -->
```

4.4 Patrón Sugerido para Operaciones CRUD

Cada módulo (libros, usuarios, préstamos) sigue un patrón similar:

1. index.php: Lista/tabla con búsqueda y acciones
2. crear.php: Formulario para crear (POST procesa)
3. editar.php: Formulario para editar (GET carga, POST actualiza)
4. eliminar.php: Confirmación y eliminación (GET muestra, POST elimina)
5. detalle.php: (Opcional) Vista completa de un registro

4.5 Consideraciones de Seguridad Básica

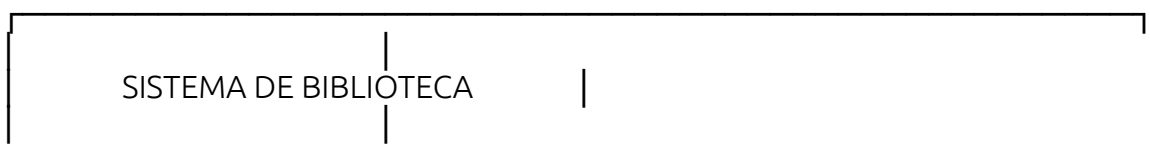
- ✅ Usar prepared statements (PDO o MySQLi) - SIEMPRE
- ✅ Validar y sanitizar todos los inputs (cliente y servidor)
- ✅ Hashear contraseñas con password_hash() y verificar con password_verify()
- ✅ Proteger todas las páginas con validación de sesión
- ✅ Validar tipos de datos (que un ID sea numérico, etc.)
- ✅ Escapar salidas HTML con htmlspecialchars() para prevenir XSS
- ✅ Usar tokens CSRF (opcional pero recomendado)

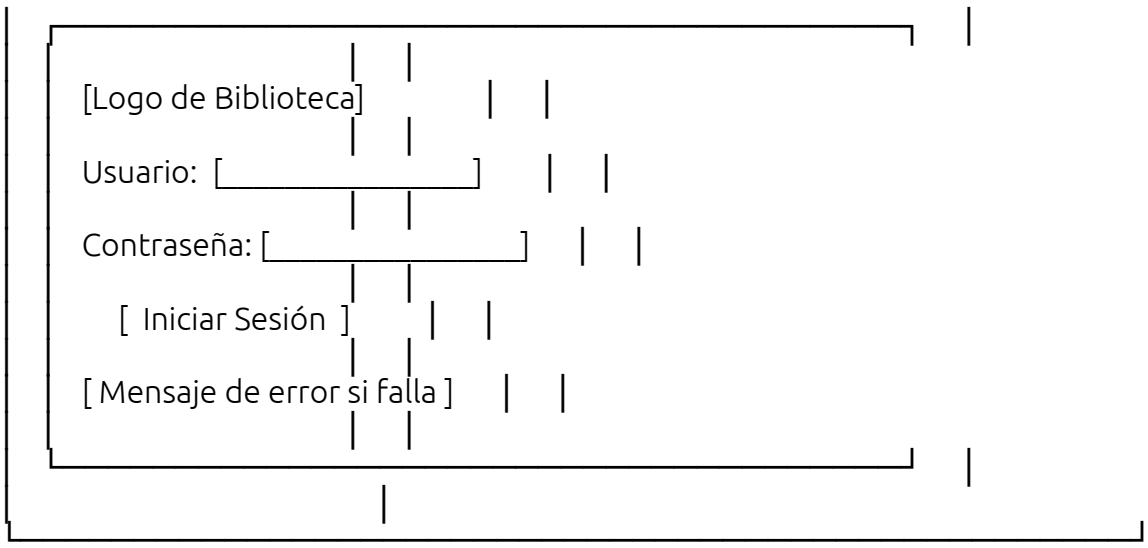
---

5. Wireframes y Mockups de Referencia

A continuación se describen visualmente las pantallas principales. Estos son wireframes de referencia, no diseños finales. Pueden adaptarlos según su creatividad.

5.1 Pantalla de Login

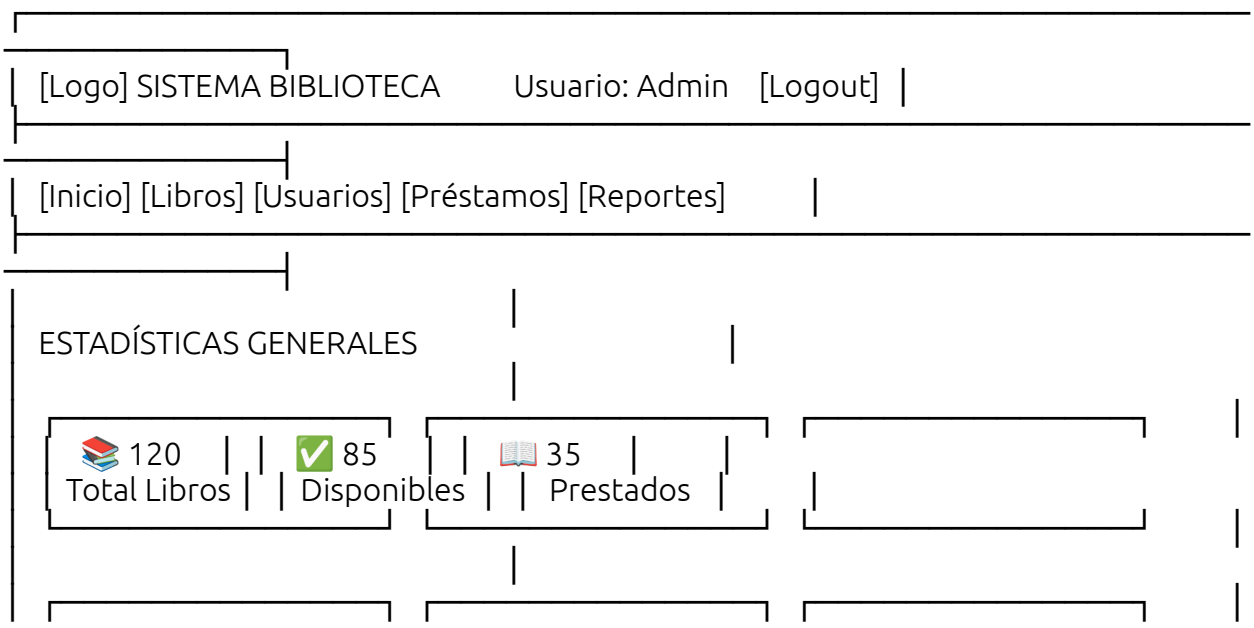


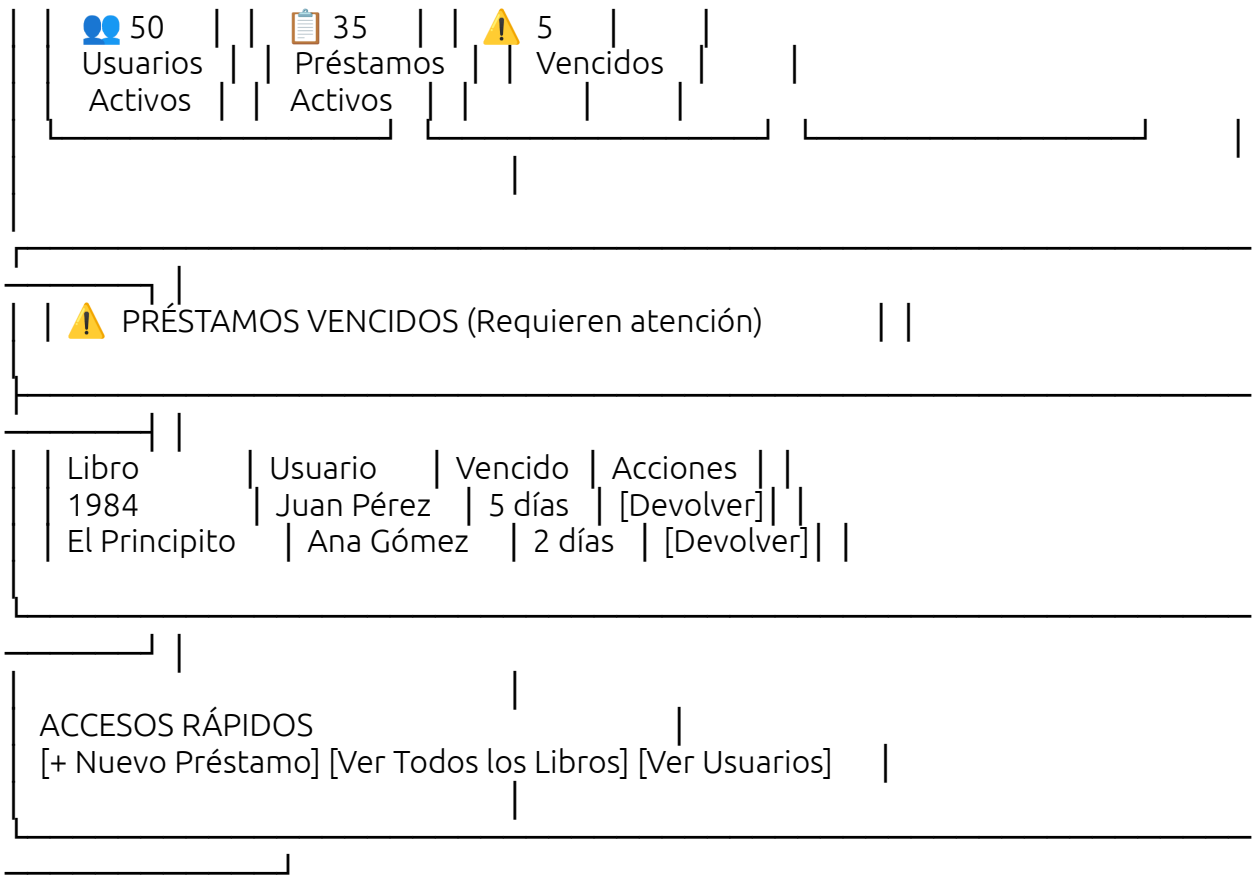


Elementos:

- Formulario centrado
- Logo o título de biblioteca
- Campos: usuario/email y contraseña
- Botón de login
- Mensaje de error/éxito (mostrar/ocultar dinámicamente)
- Fondo o imagen relacionada con libros (opcional)

5.2 Dashboard Principal

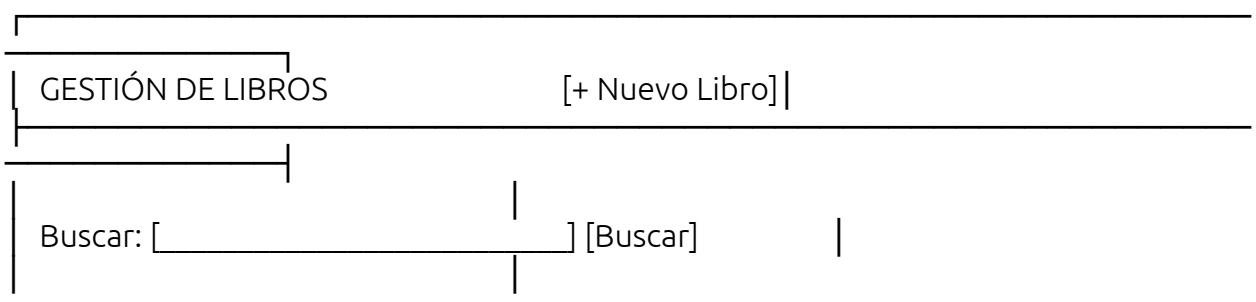


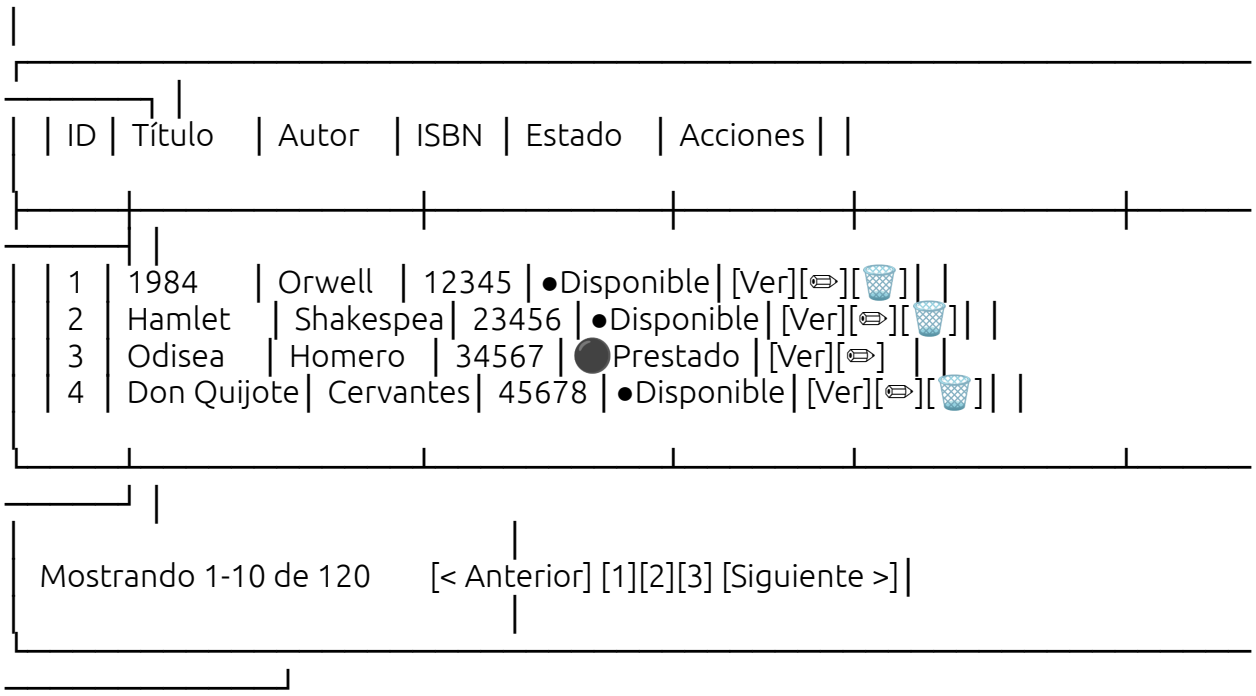


Elementos clave:

- Navbar superior con menú de navegación
- Cards/tarjetas con estadísticas numéricas
- Uso de iconos para mejor visualización
- Sección de alertas destacada (préstamos vencidos)
- Botones de acceso rápido a funciones principales
- Colores: verde=OK, amarillo=advertencia, rojo=urgente

5.3 Listado de Libros

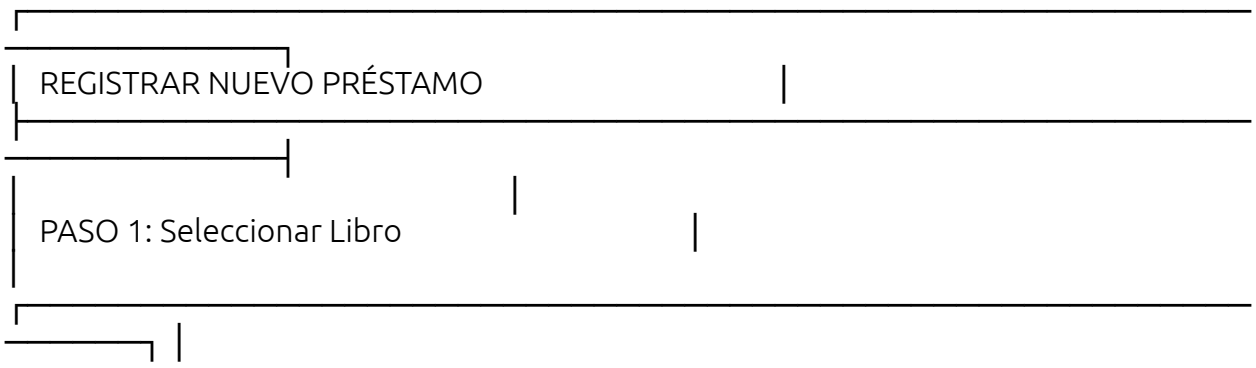


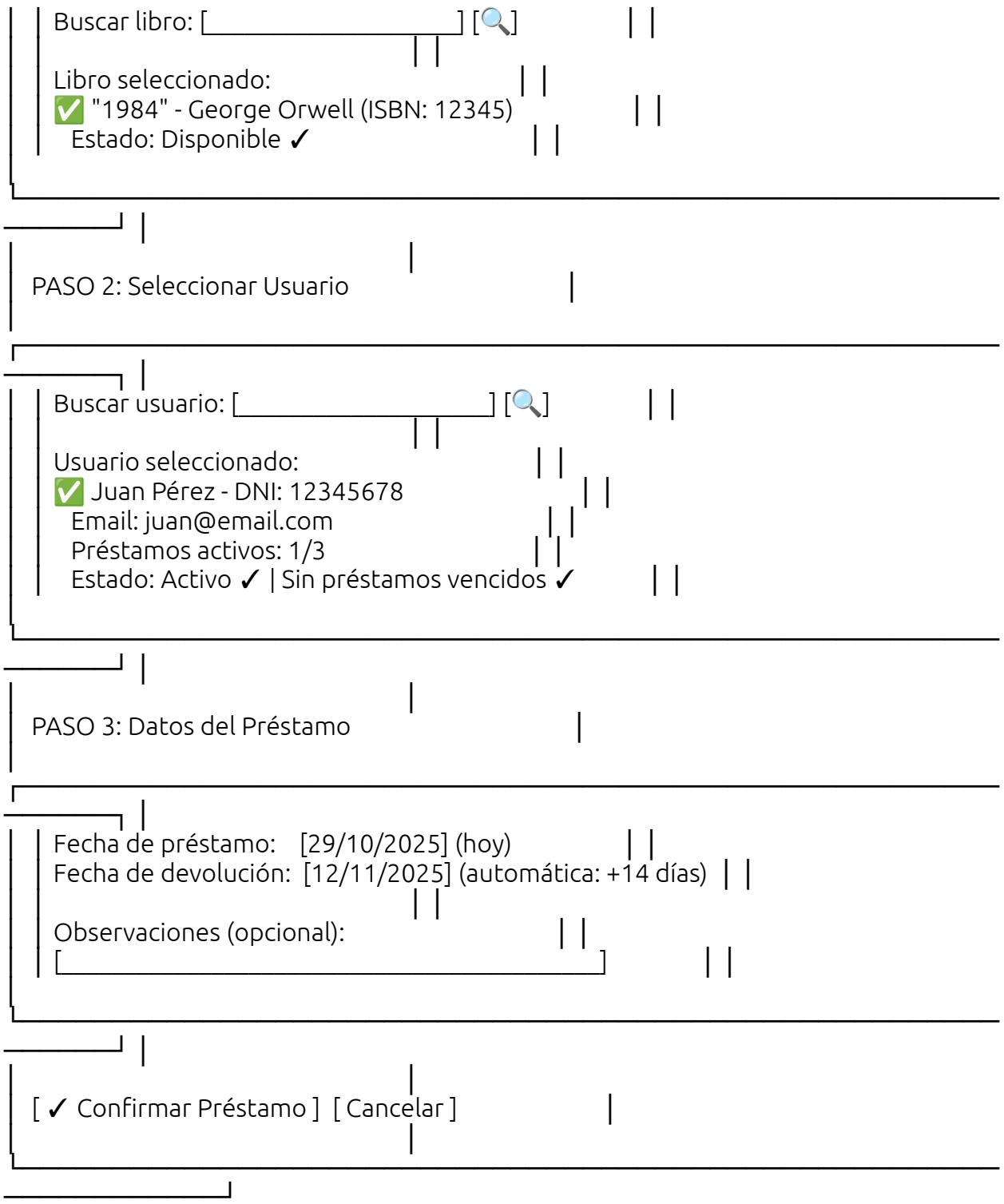


Elementos:

- Buscador en tiempo real (AJAX opcional)
- Tabla responsive con datos clave
- Badge/indicador visual de estado (disponible=verde, prestado=gris)
- Botones de acción:
    - Ver: ver detalles completos
    - Editar (✏): modificar datos
    - Eliminar (🗑): solo si está disponible
- Paginación si hay muchos registros
- Botón "Nuevo Libro" destacado

5.4 Formulario de Nuevo Préstamo



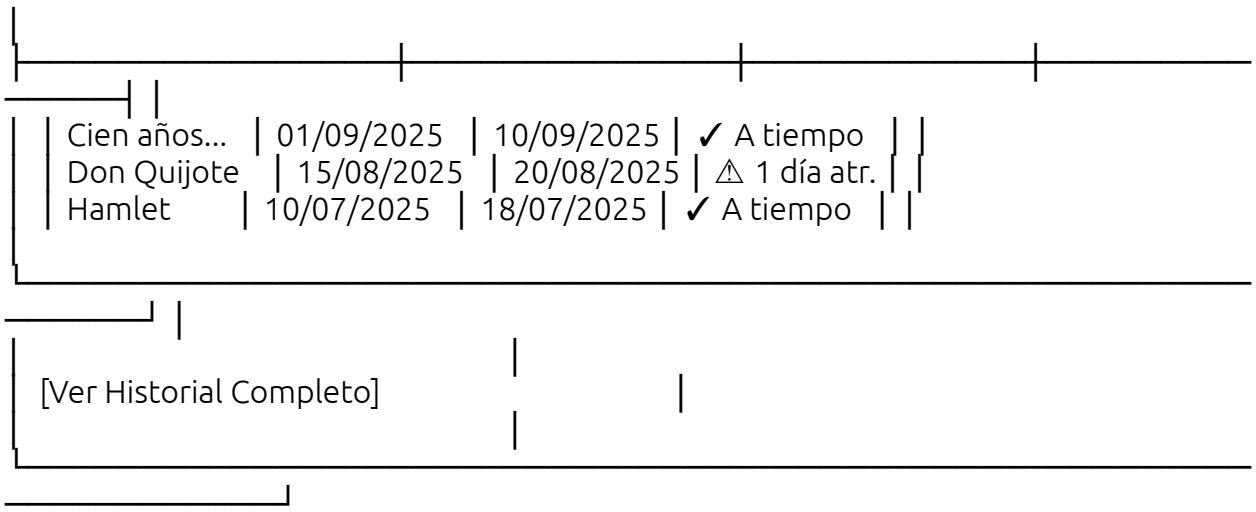


Elementos clave:

- Formulario en pasos/secciones claras

- Búsqueda dinámica con autocompletado (AJAX)
- Validaciones visuales en tiempo real:
  - ✅ = validación OK
  - ❌ = validación fallida
- Mostrar información relevante para la toma de decisión
- Fechas calculadas automáticamente
- Confirmación destacada

5.5 Perfil de Usuario (Socio)



Elementos:

- Datos personales del socio
- Lista de préstamos activos con indicadores
- Historial reciente con estado de devolución
- Acciones rápidas desde el perfil
- Indicador visual de estado (activo/suspendido)

5.6 Paleta de Colores Sugerida

Códigos de color recomendados:

Primario (Azul biblioteca): #2C3E50
Secundario (Verde éxito): #27AE60
Advertencia (Amarillo): #F39C12
Peligro (Rojo): #E74C3C
Información (Azul claro): #3498DB
Fondo claro: #ECF0F1
Texto oscuro: #2C3E50
Texto claro: #7F8C8D

Uso:

- Verde: disponible, operaciones exitosas
- Rojo: vencido, errores, eliminar
- Amarillo: advertencias, próximo a vencer
- Azul: información, enlaces, acciones neutras

---

6. Guía de Implementación

6.1 Cronograma Detallado (2 Semanas)

SEMANA 1: Fundamentos y CRUDs

Día 1-2: Setup y Autenticación

Persona A:

- Crear estructura de carpetas del proyecto
- Configurar base de datos (crear BD, tablas)
- Implementar config/database.php
- Crear archivo config/config.php con constantes
- Cargar datos de prueba iniciales

Persona B:

- Diseñar y desarrollar login.php
- Implementar logout.php
- Crear includes/auth.php (validación de sesión)
- Implementar sistema de login con password_hash
- Crear tabla usuarios_sistema y usuarios de prueba

Checkpoint Día 2:

- ✅ BD creada y poblada con datos de prueba
- ✅ Login funcional con validación
- ✅ Sesiones funcionando correctamente
- ✅ Protección básica de páginas implementada

Día 3-4: CRUD de Libros y Usuarios

Persona A: CRUD de Libros

- libros/index.php - Listar todos los libros
- libros/crear.php - Formulario y procesamiento
- libros/editar.php - Formulario y actualización
- libros/eliminar.php - Confirmación y eliminación
- Validaciones servidor (campos obligatorios, ISBN único)
- Búsqueda básica por título o autor

Persona B: CRUD de Usuarios

- usuarios/index.php - Listar todos los usuarios
- usuarios/crear.php - Formulario y procesamiento
- usuarios/editar.php - Formulario y actualización
- usuarios/detalle.php - Ver perfil del usuario
- Validaciones servidor (email único, DNI único)
- Estado activo/suspendido

Checkpoint Día 4:

- ✅ CRUD completo de libros funcional
- ✅ CRUD completo de usuarios funcional
- ✅ Validaciones básicas implementadas
- ✅ Mensajes de éxito/error funcionando

Día 5: Integración y Dashboard

Ambos (pair programming recomendado):

- Crear dashboard.php con estadísticas básicas
- Implementar includes/header.php con navbar
- Implementar includes/footer.php
- Aplicar estilos CSS básicos consistentes
- Integrar todos los módulos con navegación
- Testing conjunto de lo desarrollado hasta ahora
- Corrección de bugs encontrados

Checkpoint Día 5:

- ✅ Dashboard con estadísticas básicas
- ✅ Navegación funcionando entre todos los módulos
- ✅ Diseño consistente en todas las páginas
- ✅ No hay errores críticos

SEMANA 2: Préstamos, Validaciones y Finalización

Día 1-2: Módulo de Préstamos

Persona A: Registro de Préstamos

- prestamos/nuevo.php - Formulario de nuevo préstamo
- Búsqueda dinámica de libro (AJAX recomendado)
- Búsqueda dinámica de usuario (AJAX recomendado)
- Validaciones de negocio:
    - Libro disponible
    - Usuario activo
    - Límite de préstamos (máx 3)
    - Usuario sin préstamos vencidos
- Cálculo automático de fecha de devolución (+14 días)
- Inserción en BD y actualización de estado del libro

Persona B: Devoluciones e Historial

- prestamos/index.php - Listar préstamos activos
- Filtros: todos/vencidos/próximos a vencer
- prestamos/devolver.php - Registrar devolución
- Actualizar fecha_dev_real y estado
- Cambiar estado del libro a disponible
- prestamos/historial.php - Ver todos los préstamos
- Cálculo de días de atraso (si aplica)

Checkpoint Día 2:

- ✅ Se pueden registrar préstamos con todas las validaciones
- ✅ Se pueden registrar devoluciones
- ✅ Estados se actualizan correctamente
- ✅ Historial visible y funcional

Día 3: Dashboard Completo y Validaciones Avanzadas

Persona A: Dashboard Mejorado

- Implementar todas las estadísticas:
    - Total libros, disponibles, prestados
    - Total usuarios activos
    - Préstamos activos y vencidos
- Sección de alertas: préstamos vencidos
- Sección: próximos a vencer (3 días o menos)
- Top 5 libros más prestados
- Últimos préstamos realizados
- Accesos rápidos con botones

Persona B: Validaciones y Mejoras

- Validaciones JavaScript del lado del cliente
- Confirmaciones antes de eliminar (modal o confirm)
- Mensajes de error/éxito mejorados (toasts o alerts)
- Implementar indicadores visuales de estado (badges)
- Validación de que no se elimine libro prestado
- Validación de que no se elimine usuario con préstamos

Checkpoint Día 3:

- ✅ Dashboard completo y funcional
- ✅ Todas las validaciones críticas implementadas
- ✅ UX mejorada con feedback visual

Día 4: Funcionalidades Opcionales y Pulido

Ambos (según elección):

Si tienen tiempo, implementar 1-3 funcionalidades opcionales:

Nivel 1 (más accesibles):

- Sistema de categorías con filtros
- Búsqueda avanzada con múltiples criterios
- Alertas visuales mejoradas en dashboard
- Historial completo por usuario con estadísticas
- Ranking de libros más prestados con gráfico simple

Nivel 2 (intermedias):

- Sistema de reservas básico
- Cálculo de multas por atraso
- Generación de código QR para libros
- Exportar listados a CSV

O bien, si prefieren enfocarse en calidad:

- Pulir estilos CSS y responsive design
- Mejorar animaciones y transiciones
- Optimizar consultas SQL
- Refactorizar código duplicado
- Agregar comentarios y documentación

Checkpoint Día 4:

- ✅ Al menos 1 funcionalidad opcional implementada (si eligieron hacerlo)
- ✅ Código limpio y comentado
- ✅ Sin warnings ni errores en consola
- ✅ Testing exhaustivo de todas las funciones

Día 5: Testing, Documentación y Presentación

Persona A:

- Testing completo de todas las funcionalidades
- Crear sql/estructura.sql con export de BD
- Crear sql/datos_prueba.sql con inserts de ejemplo
- Escribir docs/README.md con:
    - Descripción del proyecto
    - Requisitos (PHP, MySQL, etc.)
    - Instrucciones de instalación
    - Credenciales de acceso de prueba
    - Funcionalidades implementadas

Persona B:

- Escribir docs/division_tareas.md con:
    - Qué hizo cada integrante
    - Cómo se dividieron el trabajo
    - Dificultades encontradas y soluciones
- Preparar presentación (slides o documento):
    - Introducción al proyecto
    - Funcionalidades principales (con capturas)
    - Aspectos técnicos destacados
    - Demo en vivo (guion preparado)
    - Conclusiones y aprendizajes
- Grabar un video corto del sistema funcionando (backup para la demo)

Checkpoint Final:

- ✅ Todos los archivos entregables listos
- ✅ Documentación completa
- ✅ Presentación preparada
- ✅ Sistema testeado exhaustivamente
- ✅ Código en repositorio Git (opcional pero recomendado)

6.2 División de Trabajo: Alternativas

Opción A - Por Módulo Funcional (Recomendada)

- Persona 1: Libros + Dashboard
- Persona 2: Usuarios + Autenticación
- Ambos: Préstamos (pair programming)

Opción B - Por Capa

- Persona 1: Backend (PHP, BD, lógica)
- Persona 2: Frontend (HTML, CSS, JS, validaciones)
- Requiere más comunicación y coordinación

Opción C - Por Feature Completa

- Persona 1: Autenticación + CRUD Libros + Préstamos
- Persona 2: CRUD Usuarios + Devoluciones + Dashboard
- Cada uno hace features de punta a punta

Recomendación: Usen la Opción A por módulo funcional, ya que permite avanzar en paralelo con menos conflictos de integración.

6.3 Puntos de Integración Críticos

Momento de integrar (requiere coordinación):

1. Día 2: Conectar login con dashboard
2. Día 4: Conectar CRUDs con header/footer común
3. Día 5: Integración completa pre-semana 2
4. Día 2 (S2): Conectar préstamos con libros y usuarios
5. Día 3 (S2): Integrar todo con dashboard

Tips para la integración:

- Usen nombres de variables y funciones consistentes
- Acuerden estructura de respuestas AJAX (JSON)
- Prueben juntos después de cada integración
- Usen Git con branches para evitar conflictos

---

7. Validaciones y Reglas de Negocio

7.1 Validaciones Obligatorias del Sistema

Al Registrar un Préstamo

Validaciones del Libro:

1. ✅ El libro existe en la base de datos
2. ✅ El libro está en estado "disponible"
3. ❌ Si el libro está prestado, mostrar: "El libro no está disponible actualmente"

Validaciones del Usuario:

1. ✅ El usuario existe en la base de datos
2. ✅ El usuario está en estado "activo" (no suspendido)
3. ✅ El usuario NO tiene préstamos vencidos
4. ✅ El usuario NO excede el límite de préstamos simultáneos (máximo 3)
5. ❌ Si está suspendido: "Usuario suspendido, no puede realizar préstamos"
6. ❌ Si tiene vencidos: "Usuario tiene préstamos vencidos pendientes"
7. ❌ Si excede límite: "Usuario ya tiene 3 préstamos activos (límite alcanzado)"

Cálculos Automáticos:

- Fecha de préstamo: por defecto HOY (editable si lo desean)
- Fecha de devolución: fecha_prestamo + 14 días (configurable)
- Estado inicial del préstamo: "activo"

Acciones al Confirmar Préstamo:

1. Insertar registro en tabla prestamos
2. Actualizar estado del libro a "prestado"
3. Mostrar confirmación con resumen del préstamo
4. Opción: enviar email de confirmación (funcionalidad opcional)

Al Registrar una Devolución

Validaciones:

1. ✅ El préstamo existe y está en estado "activo"
2. ✅ El préstamo pertenece al usuario indicado

Cálculos:

- Fecha de devolución real: HOY
- Calcular días de atraso: fecha_dev_real - fecha_devolucion
    - Si atraso > 0: mostrar advertencia y calcular multa (opcional)
    - Si atraso <= 0: devolución a tiempo

Acciones al Confirmar Devolución:

1. Actualizar fecha_dev_real con la fecha actual
2. Cambiar estado del préstamo a "devuelto"
3. Cambiar estado del libro a "disponible"
4. Mostrar resumen con días de atraso (si aplica)
5. Si hay reservas pendientes: notificar al siguiente usuario (funcionalidad opcional)

Al Eliminar un Libro

Validaciones:

1. ✅ El libro NO debe tener préstamos activos
2. ❌ Si tiene préstamos activos: "No se puede eliminar: libro actualmente prestado"
3. ⚠️ Confirmación: "¿Está seguro? Esta acción no se puede deshacer"

Alternativa (recomendada):

- En lugar de eliminar, marcar como "inactivo" o "dado de baja"
- Preservar historial de préstamos

Al Eliminar un Usuario

Validaciones:

1. ✅ El usuario NO debe tener préstamos activos
2. ❌ Si tiene préstamos activos: "No se puede eliminar: usuario tiene préstamos pendientes"
3. ⚠️ Confirmación: "¿Está seguro? Esta acción no se puede deshacer"

Alternativa (recomendada):

- Cambiar estado a "inactivo" en lugar de eliminar físicamente
- Preservar historial del usuario

Validaciones de Campos en Formularios

Libros:

- Título: requerido, máx 200 caracteres
- Autor: requerido, máx 150 caracteres
- ISBN: requerido, único, formato válido (números y guiones)
- Año: numérico, entre 1000 y año actual
- Editorial: opcional, máx 100 caracteres

Usuarios:

- Nombre completo: requerido, máx 150 caracteres
- Email: requerido, único, formato válido
- DNI: requerido, único, solo números
- Teléfono: opcional, formato validado
- Dirección: opcional

Préstamos:

- Libro ID: requerido, debe existir
- Usuario ID: requerido, debe existir
- Fecha préstamo: requerida, no futura
- Fecha devolución: requerida, posterior a fecha préstamo

7.2 Reglas de Negocio Configurables

Estas reglas pueden ajustarse según necesidades:

```
// config/config.php
define('DIAS_PRESTAMO', 14);          // Días de préstamo por defecto
define('MAX_PRESTAMOS_USUARIO', 3);    // Máximo préstamos simultáneos
define('MULTA_DIA_ATRASO', 50);       // Pesos por día de atraso (opcional)
define('DIAS_ALERTA_VENCIMIENTO', 3);  // Días antes de vencer para alertar
```

7.3 Estados del Sistema

Estados de Libro:

- disponible: Puede ser prestado
- prestado: Actualmente prestado a un usuario

(Opcional) Si implementan múltiples copias:

- en_reparacion: No disponible temporalmente
- dado_de_baja: Ya no está en el catálogo

Estados de Usuario:

- activo: Puede solicitar préstamos
- suspendido: No puede solicitar préstamos (por multas, etc.)

(Opcional) Si implementan sistema más completo:

- inactivo: Usuario dado de baja pero con historial

Estados de Préstamo:

- activo: Préstamo en curso (libro no devuelto)
- devuelto: Préstamo finalizado correctamente
- vencido: Préstamo activo que superó la fecha de devolución

Nota: El estado "vencido" puede calcularse dinámicamente:

```
-- Calcular vencidos en una consulta:
SELECT * FROM prestamos
WHERE estado = 'activo'
AND fecha_devolucion < CURDATE();
```

O bien, pueden actualizar periódicamente con un script:

```
// Actualizar estados vencidos diariamente
UPDATE prestamos
SET estado = 'vencido'
WHERE estado = 'activo'
AND fecha_devolucion < CURDATE();
```

7.4 Ejemplo de Validación Completa (Pseudocódigo)

```
// Ejemplo: Validar antes de registrar préstamo

function validarPrestamo($libro_id, $usuario_id) {
    $errores = [];

    // 1. Validar libro
    $libro = obtenerLibro($libro_id);
    if (!$libro) {
        $errores[] = "Libro no encontrado";
    } elseif ($libro['estado'] != 'disponible') {
        $errores[] = "El libro no está disponible";
    }

    // 2. Validar usuario
    $usuario = obtenerUsuario($usuario_id);
    if (!$usuario) {
        $errores[] = "Usuario no encontrado";
    } elseif ($usuario['estado'] == 'suspendido') {
        $errores[] = "Usuario suspendido";
    }
```

```
    // 3. Verificar préstamos vencidos
    if (tienePrestamosVencidos($usuario_id)) {
        $errores[] = "Usuario tiene préstamos vencidos";
    }

    // 4. Verificar límite de préstamos
    $prestamos_activos = contarPrestamosActivos($usuario_id);
    if ($prestamos_activos >= MAX_PRESTAMOS_USUARIO) {
        $errores[] = "Usuario alcanzó el límite de préstamos simultáneos";
    }

    return $errores; // Array vacío si todo OK
}

// Uso:
$errores = validarPrestamo($_POST['libro_id'], $_POST['usuario_id']);
if (empty($errores)) {
    // Proceder con el préstamo
    registrarPrestamo(...);
} else {
    // Mostrar errores
    foreach ($errores as $error) {
        echo "<div class='alert alert-danger'>$error</div>";
    }
}
```

7.5 Manejo de Fechas en PHP

Calcular fecha de devolución (+14 días):

```
$fecha_prestamo = date('Y-m-d'); // Hoy
$fecha_devolucion = date('Y-m-d', strtotime($fecha_prestamo . ' +14 days'));
```

Calcular días de atraso:

```
$fecha_esperada = '2025-10-15';
$fecha_real = date('Y-m-d');

$datetime1 = new DateTime($fecha_esperada);
$datetime2 = new DateTime($fecha_real);
$dias_atraso = $datetime1->diff($datetime2)->days;

if ($fecha_real > $fecha_esperada) {
```

```
    echo "Atrasado por $dias_atraso días";
}
```

Calcular días restantes:

```
$fecha_devolucion = '2025-11-12';
$hoy = date('Y-m-d');

$datetime1 = new DateTime($hoy);
$datetime2 = new DateTime($fecha_devolucion);
$dias_restantes = $datetime1->diff($datetime2)->days;

if ($hoy < $fecha_devolucion) {
    echo "Vence en $dias_restantes días";
} else {
    echo "Vencido";
}
```

---

8. Criterios de Evaluación

8.1 Distribución de Puntos (Total: 100 puntos)

1. Funcionalidad (40 puntos)

Funcionalidades Core Obligatorias (35 puntos):

Funcionalidad	Puntos
Sistema de autenticación (login/logout)	3
CRUD Libros completo	6
CRUD Usuarios completo	6
Registro de préstamos con validaciones	8
Registro de devoluciones	5
Dashboard con estadísticas	4
Listados y búsquedas	3

Funcionalidades Opcionales (hasta 5 puntos extra):

- Cada funcionalidad Nivel 1: +1-2 puntos
- Cada funcionalidad Nivel 2: +3-4 puntos
- Cada funcionalidad Nivel 3: +5-6 puntos

---

2. Código y Estructura (25 puntos)

Aspecto	Puntos	Criterios
Estructura y organización	5	- Estructura de carpetas lógica - Separación de responsabilidades - Archivos con nombres descriptivos
Calidad del código PHP	8	- Código limpio y legible - Uso correcto de prepared statements - Manejo de errores - Comentarios donde es necesario
Base de Datos	6	- Diseño normalizado de tablas - Relaciones correctas (FK) - Tipos de datos apropiados - Índices en campos clave
Validaciones	6	- Validaciones en servidor (PHP) - Validaciones en cliente (JS) - Validaciones de negocio correctas - Mensajes de error claros

Penalizaciones:

- -5 puntos: No usar prepared statements (SQL injection vulnerable)
- -3 puntos: Contraseñas en texto plano (no usar password_hash)
- -2 puntos: Código extremadamente desorganizado o duplicado

---

3. Diseño y Experiencia de Usuario (15 puntos)

Aspecto	Puntos	Criterios
Diseño visual	5	- Interfaz consistente y prolija - Uso apropiado de colores - Tipografía legible

Usabilidad	5	- Navegación intuitiva - Formularios claros y organizados - Feedback visual de acciones
Responsive	3	- Funciona en diferentes tamaños de pantalla - Tablas adaptativas
Detalles UX	2	- Indicadores de estado visuales - Confirmaciones antes de acciones críticas - Mensajes de éxito/error claros

---

4. Trabajo en Equipo (10 puntos)

Aspecto	Puntos	Criterios
División de tareas	4	- División equitativa y clara del trabajo - Documentación de quién hizo qué - Evidencia de colaboración
Integración	3	- Módulos integrados correctamente - Sin inconsistencias entre partes - Código cohesivo
Documentación	3	- README con instrucciones claras - Documento de división de tareas - Comentarios en código complejo

Evaluación individual: Cada integrante recibirá la nota grupal, pero puede haber ajustes (+/-10%) según la contribución individual evidenciada en la documentación.

---

5. Presentación (10 puntos)

Aspecto	Puntos	Criterios
Claridad de exposición	3	- Explicación clara del proyecto - Ambos integrantes participan
Demo en vivo	4	- Demostración fluida de funcionalidades - Manejan bien los errores (si ocurren) - Muestran casos de uso relevantes

Conocimiento técnico	3	- Responden preguntas correctamente - Explican decisiones técnicas - Demuestran comprensión del código

---

8.2 Escala de Calificación

Puntos	Calificación	Descripción
90-100	Excelente (9-10)	Cumple todo el core + funcionalidades opcionales de calidad. Código ejemplar.
80-89	Muy Bueno (8-9)	Cumple todo el core correctamente + algunas opcionales. Buen código.
70-79	Bueno (7-8)	Cumple el core con algunos detalles menores. Código aceptable.
60-69	Aprobado (6-7)	Cumple el core básico pero con falencias. Código mejorable.
50-59	Insuficiente (4-5)	Core incompleto o con errores significativos.
0-49	No aprobado (1-3)	Funcionalidades críticas faltantes o no funcionan.

8.3 Requisitos Mínimos para Aprobar (60 puntos)

✅ Obligatorios:

- Login/logout funcionando
- CRUD de libros (al menos listar, crear y editar)
- CRUD de usuarios (al menos listar y crear)
- Poder registrar un préstamo
- Poder registrar una devolución
- Al menos 2-3 validaciones de negocio implementadas
- Base de datos con estructura correcta
- Código sin vulnerabilidades críticas (SQL injection, XSS básico)

8.4 Bonificaciones Especiales

+5 puntos extra (acumulables):

- ⭐ Uso de Git con commits descriptivos y frecuentes
- ⭐ Tests automatizados (PHPUnit o similar)
- ⭐ Implementación completa de API REST
- ⭐ Sistema particularmente innovador o creativo
- ⭐ Código excepcionalmente limpio y documentado

Nota: Las bonificaciones no pueden hacer que la nota supere 100 puntos totales.

---

9. Presentación del Proyecto

9.1 Formato de la Presentación

Duración: 10-15 minutos por equipo

- 5-7 minutos: Presentación con slides o documento
- 5-7 minutos: Demo en vivo del sistema
- 3-5 minutos: Preguntas y respuestas

Modalidad:

- Ambos integrantes deben participar equitativamente
- Pueden dividirse las secciones a presentar
- La demo la pueden hacer entre ambos

9.2 Contenido de la Presentación

1. Introducción (1 minuto)

- Nombre del equipo e integrantes
- Título del proyecto: "Sistema de Gestión de Biblioteca"
- Contexto breve: qué problema resuelve

2. Funcionalidades Principales (2-3 minutos)

Presentar con capturas de pantalla o diagramas:

- ✅ Gestión de libros (CRUD)
- ✅ Gestión de usuarios/socios

- ✅ Préstamos y devoluciones
- ✅ Dashboard con estadísticas
- ✅ Funcionalidades opcionales implementadas (si aplica)

Tip: No explicar todos los detalles, solo lo más destacado.

3. Aspectos Técnicos (2 minutos)

- Tecnologías utilizadas (PHP, MySQL, JavaScript)
- Arquitectura: monolítico híbrido o API REST
- Estructura de base de datos (mostrar diagrama ER)
- Desafíos técnicos enfrentados y cómo los resolvieron

Ejemplos:

- "Tuvimos que implementar validaciones complejas para..."
- "Usamos AJAX para mejorar la experiencia en búsquedas..."
- "Implementamos un sistema de cálculo de multas automático..."

4. División de Trabajo (1 minuto)

- Cómo se organizaron
- Qué hizo cada integrante
- Cómo integraron el trabajo

Tip: Mostrar respeto y reconocimiento del trabajo del compañero.

5. Demo en Vivo (5-7 minutos)

Guion sugerido para la demo:

1. Login al sistema (30 seg)
    - Mostrar pantalla de login
    - Ingresar credenciales
    - Acceder al dashboard
2. Dashboard (1 min)
    - Recorrer las estadísticas principales
    - Destacar alertas de préstamos vencidos
3. Gestión de Libros (1-2 min)
    - Mostrar listado de libros
    - Buscar un libro
    - Agregar un nuevo libro (formulario y confirmación)

    - (Opcional) Editar un libro existente
4. Gestión de Usuarios (1 min)
    - Mostrar listado de usuarios
    - Ver detalle de un usuario (préstamos activos)
5. Préstamo de Libro (2-3 min) [核心 - lo más importante]
    - Ir a "Nuevo Préstamo"
    - Buscar y seleccionar un libro disponible
    - Buscar y seleccionar un usuario
    - Mostrar validaciones en tiempo real:
        - ✅ Libro disponible
        - ✅ Usuario sin préstamos vencidos
        - ✅ No excede límite
    - Confirmar préstamo
    - Volver a listado y mostrar que el libro cambió a "prestado"
6. Devolución de Libro (1-2 min)
    - Ir a "Préstamos Activos"
    - Seleccionar un préstamo
    - Registrar devolución
    - Mostrar cálculo de días (a tiempo o atrasado)
    - Verificar que el libro volvió a "disponible"
7. Funcionalidad Opcional (1 min - si aplica)
    - Mostrar la funcionalidad extra que implementaron
    - Ejemplo: sistema de multas, QR, reportes, etc.

Tips para la demo:

- 🎯 Ensayen antes para que sea fluida
- 🎯 Tengan datos de prueba listos (no crear todo desde cero)
- 🎯 Si algo falla, mantengan la calma y expliquen qué debería ocurrir
- 🎯 Preparen un video de backup por si hay problemas técnicos
- 🎯 Destaquen las validaciones funcionando

6. Conclusiones y Aprendizajes (1 minuto)

- Qué aprendieron con este proyecto
- Qué fue lo más desafiante
- Qué mejorarían o agregarían con más tiempo
- Agradecimientos

9.3 Preguntas Frecuentes que Pueden Hacer

Prepárense para responder:

Sobre funcionalidades:

- ¿Qué pasa si un usuario intenta pedir prestado un libro que ya está prestado?
- ¿Cómo calculan la fecha de devolución?
- ¿Qué validaciones implementaron?
- ¿Por qué eligieron no usar frameworks?

Sobre implementación técnica:

- ¿Usaron prepared statements? ¿Por qué es importante?
- ¿Cómo almacenan las contraseñas?
- ¿Cómo manejan las sesiones?
- ¿Qué pasaría si dos bibliotecarios intentan prestar el mismo libro al mismo tiempo?

Sobre trabajo en equipo:

- ¿Cómo se dividieron el trabajo?
- ¿Usaron Git? ¿Cómo manejaron conflictos?
- ¿Qué fue lo más difícil de la integración?

Sobre decisiones de diseño:

- ¿Por qué eligieron esta estructura de base de datos?
- ¿Por qué no permitir eliminar libros prestados?
- ¿Cómo decidieron la paleta de colores?

9.4 Checklist Pre-Presentación

24 horas antes:

- [ ] Slides o documento de presentación completo
- [ ] Guion de demo preparado y ensayado
- [ ] Video de backup grabado (por si hay problemas técnicos)
- [ ] Sistema testeado exhaustivamente
- [ ] Datos de prueba cargados y listos
- [ ] Ambos integrantes repasaron su parte

1 hora antes:

- [ ] Sistema corriendo localmente (probado)
- [ ] Navegador limpio (sin tabs innecesarias)
- [ ] Zoom apropiado para que se vea bien en proyector

- [ ] Consola sin errores
- [ ] Usuario de prueba logueado (para ahorrar tiempo)

Durante la presentación:

- [ ] Hablar claro y pausado
- [ ] Mirar a la audiencia, no solo a la pantalla
- [ ] Ambos participan equitativamente
- [ ] Responder con confianza (incluso si no saben algo)
- [ ] Gestionar el tiempo (no pasarse ni quedarse cortos)

---

10. Entregables

10.1 Qué Deben Entregar

1. Código Fuente Completo

Carpeta del proyecto con toda la estructura:

```
biblioteca/
├── (todos los archivos .php)
├── config/
├── includes/
├── libros/
├── usuarios/
├── prestamos/
├── assets/
├── sql/
└── docs/
```

Formato de entrega:

- Archivo .zip con el nombre: biblioteca_apellido1_apellido2.zip
- O repositorio Git (GitHub, GitLab, Bitbucket) con acceso compartido
- Si usan Git: incluir el .git para ver el historial de commits

---

2. Base de Datos

Archivo SQL con:

sql/estructura.sql:

```
-- Creación de base de datos
CREATE DATABASE IF NOT EXISTS biblioteca;
USE biblioteca;

-- Estructura de todas las tablas
CREATE TABLE libros (...);
CREATE TABLE usuarios (...);
CREATE TABLE prestamos (...);
CREATE TABLE usuarios_sistema (...);

-- Índices
CREATE INDEX ...;
```

sql/datos_prueba.sql:

```
-- Datos de ejemplo para testing
INSERT INTO usuarios_sistema ...;
INSERT INTO libros ...;
INSERT INTO usuarios ...;
INSERT INTO prestamos ...;
```

Nota: Separar estructura y datos facilita la evaluación.

---

3. Documentación

A. README.md (archivo principal)

Debe incluir:

```
# Sistema de Gestión de Biblioteca

## Integrantes
- Juan Pérez (juan@email.com)
- María González (maria@email.com)

## Descripción
Sistema web para gestionar préstamos de libros...

## Requisitos
- PHP 7.4 o superior
```

- MySQL 5.7 o superior
- Servidor web (Apache/Nginx) o usar `php -S`
- Extensiones PHP: mysqli/PDO, mbstring

Instalación

1. Clonar o descomprimir el proyecto en la carpeta del servidor web
2. Importar la base de datos:
   - Ejecutar `sql/estructura.sql`
   - Ejecutar `sql/datos_prueba.sql`
3. Configurar conexión en `config/database.php`:
   ```php
   define('DB_HOST', 'localhost');
   define('DB_NAME', 'biblioteca');
   define('DB_USER', 'tu_usuario');
   define('DB_PASS', 'tu_contraseña');
   ```

4. Acceder a http://localhost/biblioteca en el navegador

Credenciales de Prueba

Usuarios del Sistema (login):

- Admin: usuario admin / contraseña admin123
- Bibliotecario: usuario biblio / contraseña biblio123

Usuarios de Prueba (socios):

- Juan Pérez (DNI: 12345678)
- Ana García (DNI: 87654321)
- ... (listar 3-5 más)

Funcionalidades Implementadas

Core (Obligatorias)

- ✅ Sistema de autenticación
- ✅ CRUD de libros
- ✅ CRUD de usuarios/socios
- ✅ Registro de préstamos con validaciones
- ✅ Registro de devoluciones
- ✅ Dashboard con estadísticas

Opcionales

- ✅ Sistema de categorías
- ✅ Búsqueda avanzada con AJAX
- ✅ Generación de código QR para libros
- (listar lo que implementaron)

Estructura del Proyecto

```
biblioteca/
├── index.php
├── login.php
├── config/
│   └── database.php
├── libros/
│   ├── index.php
│   └── ...
...
```

Tecnologías Utilizadas

- Backend: PHP 8.1
- Base de Datos: MySQL 8.0
- Frontend: HTML5, CSS3, JavaScript vanilla
- Librerías: (listar si usaron alguna)

Capturas de Pantalla

(Opcional pero recomendado: 3-5 capturas clave)

Problemas Conocidos

(Si hay algún bug menor conocido, mencionarlo)

Mejoras Futuras

- Implementar sistema de notificaciones por email
- Agregar reportes en PDF
- ...

Licencia

Proyecto académico - Curso de PHP/MySQL

B. division_tareas.md

```markdown
# División de Tareas - Sistema de Biblioteca

## Equipo
- **Integrante A:** Juan Pérez
- **Integrante B:** María González

## División de Trabajo

### Semana 1

**Juan Pérez:**
- Setup inicial del proyecto
- Configuración de base de datos
- CRUD completo de libros
- Búsquedas y filtros de libros
- Validaciones de libros

**María González:**
- Sistema de login/logout
- Protección de páginas (auth)
- CRUD completo de usuarios
- Perfil de usuario
- Validaciones de usuarios

**Ambos (pair programming):**
- Header y footer comunes
- Dashboard básico
- Integración de módulos

### Semana 2

**Juan Pérez:**
- Módulo de registro de préstamos
- Búsquedas dinámicas con AJAX
- Validaciones de negocio de préstamos
- Dashboard completo con estadísticas

**María González:**
```

- Módulo de devoluciones
- Historial de préstamos
- Cálculo de fechas y atrasos
- Sistema de alertas visuales

Ambos:
- Testing exhaustivo
- Generación de código QR (funcionalidad opcional)
- Documentación
- Presentación

Herramientas de Colaboración
- Control de versiones: Git (GitHub)
- Comunicación: WhatsApp + reuniones presenciales
- Coordinación: Trello / lista compartida

Desafíos y Soluciones

Desafío 1: Integrar búsquedas AJAX
Solución: Creamos un archivo separado para los endpoints JSON y usamos fetch() desde el frontend.

Desafío 2: Validaciones complejas de préstamo
Solución: Centralizamos todas las validaciones en una función que devuelve array de errores.

Desafío 3: Conflictos en Git al trabajar en paralelo
Solución: Creamos branches separadas y hacíamos merge después de probar localmente.

Tiempo Estimado por Módulo
- Setup y configuración: 4 horas
- Login/autenticación: 3 horas
- CRUD Libros: 5 horas
- CRUD Usuarios: 5 horas
- Módulo Préstamos: 8 horas
- Dashboard y estadísticas: 4 horas
- Testing y bugfixing: 6 horas
- Documentación y presentación: 5 horas
Total: ~40 horas

Aprendizajes

Juan:
- Mejoré significativamente en el manejo de AJAX y respuestas JSON
- Aprendí a estructurar validaciones complejas

- Comprendí mejor el uso de JOIN en consultas SQL

María:
- Profundicé en el manejo de sesiones en PHP
- Aprendí a trabajar con password_hash y autenticación segura
- Mejoré mis habilidades de trabajo en equipo y Git

Conclusión
El proyecto nos permitió aplicar todos los conceptos vistos en clase y nos enfrentó a desafíos reales de desarrollo web. La división de trabajo fue equilibrada y logramos integrar exitosamente todos los módulos.

---

4. Presentación

Formato: PDF, PowerPoint, Google Slides, o similar

Nombre del archivo: presentacion_apellido1_apellido2.pdf

Contenido mínimo:

- Slide 1: Título, integrantes
- Slide 2-3: Funcionalidades principales
- Slide 4: Arquitectura técnica
- Slide 5: División de trabajo
- Slide 6: Demo (puede ser solo título, ya que la demo es en vivo)
- Slide 7: Conclusiones

Opcional:

- Video de 3-5 minutos mostrando el sistema funcionando (backup de la demo)

10.2 Formato de Entrega

Opción A: Archivo ZIP

```
biblioteca_perez_gonzalez.zip
├── src/  (código fuente completo)
├── sql/
│   ├── estructura.sql
│   └── datos_prueba.sql
├── docs/
│   ├── README.md
```

```
│   └── division_tareas.md
└── presentacion.pdf
```

Opción B: Repositorio Git (Recomendado)

- Crear repositorio en GitHub/GitLab
- Incluir todos los archivos
- README.md en la raíz
- Compartir link con permisos de lectura
- Bonus: historial de commits visible

10.3 Checklist Final de Entrega

Antes de entregar, verificar:

Código:

- [ ] Todo el código funciona sin errores
- [ ] No hay warnings en consola del navegador
- [ ] No hay errores de PHP (display_errors verificado)
- [ ] Todas las funcionalidades core implementadas
- [ ] Código comentado en partes complejas
- [ ] Variables y funciones con nombres descriptivos
- [ ] Sin código comentado innecesario o archivos de prueba

Base de Datos:

- [ ] Script de estructura ejecuta sin errores
- [ ] Script de datos ejecuta sin errores
- [ ] Datos de prueba variados y realistas
- [ ] Credenciales de login incluidas en datos

Documentación:

- [ ] README.md completo y claro
- [ ] Instrucciones de instalación probadas
- [ ] Division_tareas.md detallado
- [ ] Credenciales de prueba documentadas
- [ ] Funcionalidades implementadas listadas

Presentación:

- [ ] Slides completas y revisadas
- [ ] Guion de demo preparado
- [ ] Sistema testeado para la demo
- [ ] Video de backup (opcional)

General:

- [ ] Nombres de archivos correctos
- [ ] No incluir carpetas innecesarias (node_modules, .idea, etc.)
- [ ] Tamaño del .zip razonable (<10MB sin contar videos)
- [ ] Entregado en tiempo y forma

---

11. Tips y Buenas Prácticas

11.1 Manejo de Fechas en PHP

Configurar zona horaria al inicio:

```
// config/config.php
date_default_timezone_set('America/Argentina/Buenos_Aires');
```

Obtener fecha actual:

```
// Para insertar en BD (formato SQL)
$fecha_hoy = date('Y-m-d'); // 2025-10-29

// Para mostrar al usuario (formato legible)
$fecha_mostrar = date('d/m/Y'); // 29/10/2025
```

Sumar días a una fecha:

```
$fecha_prestamo = '2025-10-29';
$fecha_devolucion = date('Y-m-d', strtotime($fecha_prestamo . ' +14 days'));
// Resultado: 2025-11-12
```

Comparar fechas:

```
$fecha1 = '2025-10-29';
$fecha2 = '2025-11-15';
```

```
if (strtotime($fecha1) < strtotime($fecha2)) {
   echo "fecha1 es anterior";
}
```

Calcular diferencia entre fechas:

```
$fecha_esperada = new DateTime('2025-10-15');
$fecha_real = new DateTime('2025-10-20');
$diferencia = $fecha_esperada->diff($fecha_real);

echo $diferencia->days; // 5 días
```

Formatear fecha de la BD para mostrar:

```
// Desde BD: 2025-10-29
$fecha_bd = '2025-10-29';

// Convertir a formato amigable
$fecha_obj = new DateTime($fecha_bd);
echo $fecha_obj->format('d/m/Y'); // 29/10/2025
echo $fecha_obj->format('d M Y'); // 29 Oct 2025
```

11.2 Validaciones Importantes

Validar que un ID sea numérico:

```
if (!is_numeric($_GET['id']) || $_GET['id'] <= 0) {
   die("ID inválido");
}
$id = (int)$_GET['id']; // Cast a entero
```

Validar email:

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
   $errores[] = "Email inválido";
}
```

Sanitizar inputs:

```
$titulo = trim($_POST['titulo']); // Eliminar espacios
```

```
$titulo = htmlspecialchars($titulo); // Prevenir XSS básico
```

Validar campos requeridos:

```
$errores = [];

if (empty($_POST['titulo'])) {
    $errores[] = "El título es obligatorio";
}

if (empty($_POST['autor'])) {
    $errores[] = "El autor es obligatorio";
}

if (!empty($errores)) {
    // Mostrar errores
    foreach ($errores as $error) {
        echo "<p class='error'>$error</p>";
    }
    exit;
}
```

11.3 Seguridad Básica

Usar Prepared Statements SIEMPRE:

```
// ❌ MAL - Vulnerable a SQL Injection
$sql = "SELECT * FROM libros WHERE id = " . $_GET['id'];
$result = $conn->query($sql);

// ✅ BIEN - Con MySQLi
$stmt = $conn->prepare("SELECT * FROM libros WHERE id = ?");
$stmt->bind_param("i", $_GET['id']);
$stmt->execute();
$result = $stmt->get_result();

// ✅ BIEN - Con PDO
$stmt = $pdo->prepare("SELECT * FROM libros WHERE id = :id");
$stmt->execute(['id' => $_GET['id']]);
$libro = $stmt->fetch();
```

Hashear contraseñas:

```
// Al registrar usuario
$password = $_POST['password'];
$password_hash = password_hash($password, PASSWORD_DEFAULT);

// Guardar $password_hash en la BD, NO el password en texto plano

// Al validar login
$password_ingresado = $_POST['password'];
$password_bd = $row['password']; // Hash desde BD

if (password_verify($password_ingresado, $password_bd)) {
    // Contraseña correcta
} else {
    // Contraseña incorrecta
}
```

Proteger páginas con sesión:

```
// includes/auth.php
session_start();

if (!isset($_SESSION['usuario_id'])) {
    header('Location: login.php');
    exit;
}

// Incluir al inicio de cada página protegida:
// require_once 'includes/auth.php';
```

Prevenir XSS al mostrar datos:

```
// Al mostrar datos del usuario en HTML
$nombre = htmlspecialchars($row['nombre'], ENT_QUOTES, 'UTF-8');
echo "<h1>Bienvenido, $nombre</h1>";
```

11.4 Organización del Código

Evitar código duplicado - Usar funciones:

```
// funciones.php
function obtenerLibroPorId($conn, $id) {
    $stmt = $conn->prepare("SELECT * FROM libros WHERE id = ?");
```

```
    $stmt->bind_param("i", $id);
    $stmt->execute();
    return $stmt->get_result()->fetch_assoc();
}

// Usar en múltiples archivos:
$libro = obtenerLibroPorId($conn, $_GET['id']);
```

Centralizar mensajes:

```
// includes/mensajes.php
function mostrarExito($mensaje) {
    echo "<div class='alert alert-success'>$mensaje</div>";
}

function mostrarError($mensaje) {
    echo "<div class='alert alert-danger'>$mensaje</div>";
}

// Uso:
mostrarExito("Libro creado exitosamente");
```

Separar lógica de presentación:

```
// libros/crear.php

// PARTE 1: Lógica PHP (al inicio)
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Procesar formulario
    // Validar
    // Insertar en BD
    // Redirigir
}

// PARTE 2: HTML (después)
?>
<!DOCTYPE html>
<html>
...
```

11.5 Debugging y Testing

Mostrar errores en desarrollo:

```
// config/config.php (solo en desarrollo, NO en producción)
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

Debug de consultas SQL:

```
// Ver qué query se está ejecutando
$sql = "SELECT * FROM prestamos WHERE usuario_id = ?";
echo $sql; // Para debug

$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $usuario_id);
$stmt->execute();

// Ver errores de MySQL
if ($stmt->error) {
    echo "Error: " . $stmt->error;
}
```

Usar var_dump para inspeccionar variables:

```
// Ver contenido de un array
var_dump($_POST);

// Ver contenido de objeto
var_dump($libro);

// Versión más legible
echo "<pre>";
print_r($libro);
echo "</pre>";
```

Testing manual - Casos a probar:

✅ Préstamo de libro:

- Usuario normal con préstamos disponibles → ✓ Debe funcionar
- Usuario con 3 préstamos activos → ✗ Debe rechazar
- Usuario con préstamo vencido → ✗ Debe rechazar
- Usuario suspendido → ✗ Debe rechazar

- Libro no disponible → ✗ Debe rechazar

✅ Eliminación de libro:

- Libro disponible → ✓ Debe eliminar
- Libro prestado → ✗ Debe rechazar

✅ Validación de formularios:

- Enviar formulario vacío → ✗ Debe mostrar errores
- Campos con valores válidos → ✓ Debe procesar
- ISBN duplicado → ✗ Debe rechazar

11.6 Performance y Optimización

Evitar consultas dentro de loops:

```
// ❌ MAL - N+1 queries problem
$prestamos = $conn->query("SELECT * FROM prestamos");
while ($prestamo = $prestamos->fetch_assoc()) {
   // Por cada préstamo, otra consulta
   $libro = $conn->query("SELECT * FROM libros WHERE id = " . $prestamo['libro_id']);
}

// ✅ BIEN - Una sola consulta con JOIN
$sql = "SELECT p.*, l.titulo, l.autor, u.nombre_completo
      FROM prestamos p
      JOIN libros l ON p.libro_id = l.id
      JOIN usuarios u ON p.usuario_id = u.id";
$prestamos = $conn->query($sql);
```

Usar índices en columnas frecuentemente buscadas:

```
-- Mejorar búsquedas
CREATE INDEX idx_libro_titulo ON libros(titulo);
CREATE INDEX idx_usuario_email ON usuarios(email);
CREATE INDEX idx_prestamo_estado ON prestamos(estado);
```

Limitar resultados con LIMIT:

```
// No traer todos los registros si solo necesitas algunos
```

$sql = "SELECT * FROM prestamos ORDER BY fecha_prestamo DESC LIMIT 10";

11.7 Git y Control de Versiones

Comandos básicos que necesitarán:

```
# Clonar repositorio
git clone https://github.com/usuario/biblioteca.git

# Crear branch para trabajar
git checkout -b feature/modulo-prestamos

# Ver cambios
git status

# Agregar archivos modificados
git add .

# Hacer commit
git commit -m "Implementar registro de préstamos con validaciones"

# Subir cambios
git push origin feature/modulo-prestamos

# Cambiar a rama principal
git checkout main

# Traer últimos cambios
git pull origin main

# Mergear tu rama a main
git merge feature/modulo-prestamos
```

Mensajes de commit descriptivos:

✅ BIEN:
- "Agregar CRUD completo de libros"
- "Implementar validaciones de préstamo"
- "Fix: corregir cálculo de días de atraso"
- "Mejorar diseño del dashboard"

❌ MAL:
- "cambios"

- "fix"
- "asdfasdf"
- "probando"

11.8 Responsive Design Básico

Media queries simples:

```
/* Estilos para desktop (por defecto) */
.dashboard-cards {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}

/* Tablets */
@media (max-width: 768px) {
    .dashboard-cards {
        grid-template-columns: repeat(2, 1fr);
    }

    table {
        font-size: 14px;
    }
}

/* Móviles */
@media (max-width: 480px) {
    .dashboard-cards {
        grid-template-columns: 1fr;
    }

    /* Hacer tabla scrolleable horizontalmente */
    .table-responsive {
        overflow-x: auto;
    }
}
```

---

12. Recursos y Referencias

12.1 Documentación Oficial

PHP:

- Manual de PHP: https://www.php.net/manual/es/
- PDO (PHP Data Objects): https://www.php.net/manual/es/book.pdo.php
- MySQLi: https://www.php.net/manual/es/book.mysqli.php
- Funciones de fecha: https://www.php.net/manual/es/function.date.php
- password_hash: https://www.php.net/manual/es/function.password-hash.php

MySQL:

- Referencia SQL: https://dev.mysql.com/doc/refman/8.0/en/
- JOINs explicados: https://www.mysqltutorial.org/mysql-join/
- Tipos de datos: https://dev.mysql.com/doc/refman/8.0/en/data-types.html

JavaScript:

- MDN Web Docs: https://developer.mozilla.org/es/docs/Web/JavaScript
- Fetch API: https://developer.mozilla.org/es/docs/Web/API/Fetch_API
- Eventos DOM: https://developer.mozilla.org/es/docs/Web/Events

CSS:

- MDN CSS: https://developer.mozilla.org/es/docs/Web/CSS
- Flexbox Guide: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Grid Guide: https://css-tricks.com/snippets/css/complete-guide-grid/

12.2 Librerías Útiles (Opcionales)

Para funcionalidades opcionales:

Generación de QR:

- PHP QR Code: https://github.com/chillerlan/php-qrcode
- Ejemplo de uso:

```
use chillerlan\QRCode\QRCode;
$qrcode = new QRCode();
echo '<img src="'.$qrcode->render($data).'" />';
```

Generación de PDF:

- TCPDF: https://tcpdf.org/

- FPDF: http://www.fpdf.org/
- mPDF: https://mpdf.github.io/

Envío de Emails:

- PHPMailer: https://github.com/PHPMailer/PHPMailer
- Alternativa: SwiftMailer

Gráficos:

- Chart.js (JavaScript): https://www.chartjs.org/
- Ejemplo simple:

```
<canvas id="myChart"></canvas>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
   new Chart(document.getElementById('myChart'), {
      type: 'bar',
      data: {...}
   });
</script>
```

12.3 Herramientas de Desarrollo

Editores de Código:

- Visual Studio Code (recomendado): https://code.visualstudio.com/
    - Extensiones útiles: PHP Intelephense, MySQL, GitLens
- PHPStorm (más avanzado): https://www.jetbrains.com/phpstorm/
- Sublime Text: https://www.sublimetext.com/

Servidores Locales:

- XAMPP: https://www.apachefriends.org/
- WAMP (Windows): https://www.wampserver.com/
- MAMP (Mac): https://www.mamp.info/
- Laragon (Windows): https://laragon.org/
- O usar PHP built-in server: `php -S localhost:8000`

Git:

- Git: https://git-scm.com/
- GitHub Desktop: https://desktop.github.com/

- Tutorial interactivo: https://learngitbranching.js.org/

Base de Datos:

- phpMyAdmin (viene con XAMPP/WAMP)
- MySQL Workbench: https://www.mysql.com/products/workbench/
- DBeaver (universal): https://dbeaver.io/

Testing/Debug:

- Browser DevTools (F12 en navegador)
- Xdebug (PHP): https://xdebug.org/
- Postman (para APIs): https://www.postman.com/

12.4 Tutoriales y Recursos de Aprendizaje

PHP y MySQL:

- W3Schools PHP: https://www.w3schools.com/php/
- PHP: The Right Way: https://phptherightway.com/
- MySQL Tutorial: https://www.mysqltutorial.org/

Seguridad:

- OWASP Top 10: https://owasp.org/www-project-top-ten/
- SQL Injection Prevention: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Git:

- Git Book: https://git-scm.com/book/es/v2
- GitHub Guides: https://guides.github.com/

Diseño:

- Color Hunt (paletas): https://colorhunt.co/
- Google Fonts: https://fonts.google.com/
- Flaticon (iconos): https://www.flaticon.com/
- Unsplash (imágenes): https://unsplash.com/

12.5 Ejemplos de Código Clave

Estructura básica de conexión PDO:

```
// config/database.php
try {
    $pdo = new PDO(
        "mysql:host=localhost;dbname=biblioteca;charset=utf8mb4",
        "root",
        "",
        [
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
            PDO::ATTR_EMULATE_PREPARES => false
        ]
    );
} catch (PDOException $e) {
    die("Error de conexión: " . $e->getMessage());
}
```

Paginación básica:

```
// Configuración
$por_pagina = 10;
$pagina = isset($_GET['pagina']) ? (int)$_GET['pagina'] : 1;
$offset = ($pagina - 1) * $por_pagina;

// Contar total de registros
$total = $conn->query("SELECT COUNT(*) as total FROM libros")->fetch_assoc()['total'];
$total_paginas = ceil($total / $por_pagina);

// Consulta con LIMIT
$sql = "SELECT * FROM libros LIMIT $offset, $por_pagina";
$libros = $conn->query($sql);

// Mostrar links de paginación
for ($i = 1; $i <= $total_paginas; $i++) {
    echo "<a href='?pagina=$i'>$i</a> ";
}
```

Búsqueda con AJAX (búsqueda.js):

```
// Frontend
document.getElementById('buscar').addEventListener('input', function() {
```

```
    const query = this.value;

    fetch(`api/buscar_libros.php?q=${encodeURIComponent(query)}`)
        .then(response => response.json())
        .then(data => {
            const resultados = document.getElementById('resultados');
            resultados.innerHTML = '';

            data.forEach(libro => {
                resultados.innerHTML += `
                    <div class="resultado">
                        <strong>${libro.titulo}</strong> - ${libro.autor}
                    </div>
                `;
            });
        });
});

// Backend: api/buscar_libros.php
header('Content-Type: application/json');
$query = $_GET['q'] ?? '';

$stmt = $pdo->prepare("SELECT * FROM libros WHERE titulo LIKE ? OR autor LIKE ? LIMIT 10");
$busqueda = "%$query%";
$stmt->execute([$busqueda, $busqueda]);

echo json_encode($stmt->fetchAll());
```

---

13. FAQ y Troubleshooting

13.1 Preguntas Frecuentes

P: ¿Podemos usar Bootstrap o Tailwind para los estilos? R: Sí, pueden usar frameworks CSS. La restricción de "sin frameworks" aplica a frameworks backend (Laravel, CodeIgniter) y frontend pesados (React, Vue). CSS frameworks están permitidos.

P: ¿Tenemos que usar Git obligatoriamente? R: No es obligatorio, pero es muy recomendado y suma puntos extra. Facilita mucho el trabajo en equipo.

P: Si implementamos una API REST, ¿contamos doble puntaje? R: Sí, implementar API REST suma puntos extra significativos (Nivel 3), pero asegúrense de que el core funcione perfectamente primero.

P: ¿Podemos cambiar los 14 días de préstamo? R: Sí, es configurable. Pueden usar 7, 14, 21 días, lo que les parezca razonable. Solo documéntenlo.

P: ¿Qué pasa si no llegamos a implementar funcionalidades opcionales? R: No es problema. Las opcionales son para destacar. Con el core completo y bien hecho pueden sacar 80-85 puntos.

P: ¿Podemos hacer el proyecto de otra temática similar? R: Consulten con el profesor primero, pero en general, la idea es que todos trabajen sobre el mismo proyecto para facilitar la evaluación.

P: ¿Usamos MySQLi o PDO? R: Cualquiera de los dos está bien, pero PDO es más moderno y portable. Si ya usan uno en clase, mantengan ese.

P: ¿Cómo manejamos si uno del equipo no colabora? R: Documenten individualmente quién hizo qué en division_tareas.md. La calificación puede ajustarse individualmente si hay evidencia clara de desbalance.

13.2 Problemas Comunes y Soluciones

Problema: "Call to undefined function mysqli_connect()"

Causa: Extensión MySQLi no habilitada en PHP

Solución:

```
# En php.ini, descomentar:
extension=mysqli

# O en Arch/Manjaro:
sudo pacman -S php-mysqli
# Reiniciar servidor web
```

Problema: "Access denied for user 'root'@'localhost'"

Causa: Credenciales incorrectas de MySQL

Solución:

1. Verificar usuario y contraseña en config/database.php
2. Verificar que MySQL esté corriendo:

```
sudo systemctl status mysql
# o
sudo systemctl start mysql
```

3. Resetear contraseña de root si es necesario

Problema: Sesión no persiste entre páginas

Causa: session_start() no llamado o llamado después de output

Solución:

```
<?php
// SIEMPRE antes de cualquier HTML
session_start();
?>
<!DOCTYPE html>
...
```

Problema: SQL Injection vulnerable

Causa: Concatenación directa de variables en queries

Solución:

```
// ❌ MAL
$sql = "SELECT * FROM usuarios WHERE id = " . $_GET['id'];

// ✅ BIEN
$stmt = $conn->prepare("SELECT * FROM usuarios WHERE id = ?");
$stmt->bind_param("i", $_GET['id']);
```

Problema: "Headers already sent"

Causa: Intentar usar header() después de enviar HTML

Solución:

```
// Asegurarse de que header() esté antes de cualquier echo o HTML
```

```
if ($login_exitoso) {
   header('Location: dashboard.php');
   exit; // IMPORTANTE: siempre usar exit después de header
}
```

Problema: Fechas con día de diferencia

Causa: Zona horaria no configurada

Solución:

```
// Al inicio del proyecto (config.php)
date_default_timezone_set('America/Argentina/Buenos_Aires');
```

Problema: Caracteres extraños (Ã±, Ã©, etc.)

Causa: Encoding incorrecto

Solución:

```
// En conexión
$conn->set_charset("utf8mb4");

// En HTML
<meta charset="UTF-8">

// En MySQL
CREATE DATABASE biblioteca CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Problema: AJAX no funciona, no pasa nada

Causa: Error en JavaScript o URL incorrecta

Solución:

```
// Agregar debugging
fetch('api/buscar.php')
   .then(response => {
      console.log('Status:', response.status); // Ver status
      return response.json();
```

```
    })
    .then(data => {
        console.log('Data:', data); // Ver qué llega
        // Procesar data
    })
    .catch(error => {
        console.error('Error:', error); // Ver errores
    });

// Revisar consola del navegador (F12)
// Revisar Network tab para ver qué se envía/recibe
```

Problema: No se puede eliminar libro prestado pero el código parece correcto

Causa: La validación no se está ejecutando o la consulta es incorrecta

Solución:

```
// Verificar que el libro no tenga préstamos activos
$stmt = $conn->prepare("SELECT COUNT(*) as total FROM prestamos WHERE libro_id =
? AND estado = 'activo'");
$stmt->bind_param("i", $libro_id);
$stmt->execute();
$result = $stmt->get_result()->fetch_assoc();

if ($result['total'] > 0) {
    die("No se puede eliminar: libro actualmente prestado");
}

// Proceder con eliminación
```

13.3 Cómo Pedir Ayuda

Si se encuentran bloqueados, sigan este proceso:

1. Identificar el error exacto:
    - Leer el mensaje de error completo
    - Verificar consola del navegador (F12)
    - Verificar logs de PHP/Apache
2. Intentar solucionarlo:

    - Buscar el error en Google
    - Revisar documentación oficial
    - Revisar ejemplos de este documento
3. Preparar información para consultar:
    - Descripción clara del problema
    - Qué se espera que ocurra
    - Qué ocurre actualmente
    - Código relevante
    - Mensaje de error completo
    - Qué se intentó hacer para solucionarlo
4. Dónde pedir ayuda:
    - Profesor en clase
    - Grupo de WhatsApp del curso
    - StackOverflow (en inglés)
    - Foros de PHP/MySQL

Tip: Muchas veces, el simple hecho de escribir el problema detalladamente ayuda a encontrar la solución.

13.4 Qué Hacer Si Se Atrasan

Si están en Semana 1 y van atrasados:

- Enfóquense primero en login y BD funcionando
- Dividan mejor las tareas
- Hagan pair programming en lo que traba
- Pidan ayuda temprano

Si están en Semana 2 y falta mucho:

- Prioricen el core: préstamos y devoluciones
- Simplifiquen el dashboard (solo estadísticas básicas)
- Olviden las funcionalidades opcionales
- Aseguren que lo básico funcione bien

Si están a 2 días de la entrega:

- Focus 100% en que el core funcione
- Testing exhaustivo
- Documentación mínima pero clara
- Ensayar la demo

Recuerden: Es mejor entregar un sistema simple que funcione bien, que uno complejo que esté lleno de bugs.

🎯 Mensaje Final

Este proyecto es una oportunidad excelente para consolidar todo lo aprendido sobre PHP, MySQL y desarrollo web. No se trata solo de aprobar, sino de construir algo funcional y del que puedan estar orgullosos.

Consejos finales:

✨ Trabajen en equipo de verdad: Comuníquense frecuentemente, ayúdense mutuamente, integren seguido.

✨ Comiencen temprano: No dejen todo para última semana. El cronograma sugerido funciona si lo siguen.

✨ Testing constante: Prueben cada funcionalidad apenas la terminan. No esperen al final para testear.

✨ Código limpio: Código legible hoy es tiempo ahorrado mañana. Comenten, usen nombres descriptivos.

✨ Pidan ayuda: No se queden trabados. Consulten temprano si algo no funciona.

✨ Disfruten: Están construyendo algo real. Diviértanse en el proceso.

¡Éxitos con el proyecto! 🚀📚

Anexo: Checklist Rápido de Entrega

📋 Checklist Final - Sistema de Biblioteca

Funcionalidades Core (Obligatorias):

- [ ] Login funcional con validación
- [ ] Logout funcional
- [ ] Protección de páginas (sesión)

- [ ] CRUD Libros: Listar, Crear, Editar, Eliminar
- [ ] CRUD Usuarios: Listar, Crear, Editar, Ver Detalle
- [ ] Registrar préstamo con todas las validaciones
- [ ] Registrar devolución
- [ ] Dashboard con estadísticas
- [ ] Listado de préstamos activos
- [ ] Búsqueda básica de libros

Base de Datos:

- [ ] Estructura correcta (4 tablas)
- [ ] Relaciones con FK
- [ ] Datos de prueba cargados
- [ ] Scripts SQL funcionando

Validaciones Críticas:

- [ ] Libro disponible para prestar
- [ ] Usuario activo
- [ ] Usuario sin préstamos vencidos
- [ ] Límite de 3 préstamos
- [ ] No eliminar libro prestado
- [ ] No eliminar usuario con préstamos

Seguridad:

- [ ] Prepared statements en todas las consultas
- [ ] Contraseñas hasheadas
- [ ] Validaciones servidor y cliente
- [ ] Sesiones protegidas

Documentación:

- [ ] README.md completo
- [ ] division_tareas.md detallado
- [ ] Credenciales de prueba documentadas
- [ ] Instrucciones de instalación claras

Presentación:

- [ ] Slides preparadas
- [ ] Guion de demo ensayado
- [ ] Ambos integrantes preparados
- [ ] Video backup (opcional)

General:

- [ ] Sin errores de PHP
- [ ] Sin errores en consola
- [ ] Diseño consistente
- [ ] Código comentado
- [ ] Proyecto comprimido correctamente

Fecha de entrega: [A completar por el profesor] Fecha de presentación: [A completar por el profesor]

Documento creado para el curso de PHP + MySQL + JavaScript Versión 1.0 - Octubre 2025