Convenciones de Código - library-system

Este documento establece las convenciones de código para el proyecto 'library-system', asegurando consistencia y facilidad de mantenimiento. Se aplican al desarrollo inicial con JavaScript puro, manteniendo la posibilidad de migrar a JSX/React.

# 1. Estándares Generales

- Idioma: código y comentarios en español neutral o inglés, pero consistente en todo el proyecto.

- Formato: usar Prettier y ESLint (reglas recomendadas).

- Indentación: 2 espacios.

- Longitud máxima de línea: 100–120 caracteres.

- Comillas simples ('...') en JavaScript.

- Consistencia en el uso de punto y coma al final de las sentencias.

- Imports: rutas relativas o absolutas según convenga.

# 2. Nomenclatura

- Carpetas y archivos: kebab-case (ej: book-service.js).

- Componentes (si se usa JSX): PascalCase (ej: SearchBar.jsx).

- Funciones y variables: camelCase (ej: fetchByIsbn).

- Clases y tipos: PascalCase (ej: Book, OpenLibraryResponse).

- Constantes: UPPER\_SNAKE\_CASE (ej: DEFAULT\_PAGE\_SIZE).

- Ramas Git: tipo/alcance-descriptivo (ej: feat/books-import).

# 3. Estructura en Frontend

- Código JavaScript organizado en servicios, utils y componentes.

- Evitar código repetido: centralizar funciones comunes en utils.

- Servicios: todas las llamadas al backend deben estar centralizadas en la carpeta services.

- Estilos CSS en carpeta styles o css/, evitar estilos en línea.

# 4. Backend

- Arquitectura por capas: controller → service → repository.

- Validación de datos en el backend antes de procesar (p.ej., validar ISBN).

- Manejo de errores centralizado con middleware.

- Respuestas JSON con estructura consistente.

- Nombres de tablas en plural y snake\_case (ej: libros, categorias).

- Campos en snake\_case (ej: fecha\_publicacion).

# 5. Git y Control de Versiones

Formato de mensajes de commit (Conventional Commits):

<tipo>(alcance): resumen imperativo  
  
Ejemplos:  
- feat(books): importación por ISBN  
- fix(search): corregir paginación vacía

Tipos comunes: feat, fix, docs, style, refactor, test, chore.

Flujo de ramas: main (estable), dev (integración), feat/\* (nueva funcionalidad), fix/\* (correcciones).

# 6. Calidad y Pruebas

- Linter obligatorio antes de commits.

- Formateo automático con Prettier.

- Pruebas unitarias para funciones críticas.

- Documentación de API y base de datos en docs/.