

Análisis técnico de la aplicación ComicVault

Arquitectura general

Vista Capa presentación.

- Interfaz de usuario: HTML y CSS y JS.
- **public** / index.php, estilos.css y funciones.js.

Controlador y modelo. Capa lógica de negocio y controlador.

- Lógica de negocio y flujo.
- Acceso y manipulación de datos. CRUD
- **app** / funciones.php

Persistencia. Capa de datos.

- Almacenamiento de datos.
- **data** / datos.json.

1. Capa de persistencia o datos. data

- **Formato:** json
- **Estructura:** Array sencillo.
- **ID:** Autoincremental con generarComi() y obtenerElementoMaximo().
- Desventajas: Múltiples usuarios, riesgo de sobreescritura.

2. Capa de modelo y controlador o acceso a datos y modelo de negocio. app

- **cargaJSON():** Leer archivos.
- **guardaJSON():** Escribir archivos.
- **generarIdComic():** Con ID unico, buscando huecos.
- **listarComics():** Filtro en backend.
- **anadir, modificar y eliminar:** CRUD.
- **Redirección POST:** Dentro del header para evitar reenvío al recargar.
- **Mensajes:** \$_GET['msg] para tener un feedback por si las funciones se han ejecutado correctamente.
- **Acción por POST:** Desacopla acción de URL, me pareció más sencillo y más seguro.

3. Capa de vista o interfaz. public

- **Formulario:.** Metodo="post" y action="../app/funciones.php"
- **Edición:** "?editar=ID". Para saber si editar o añadir.
- **Filtro:** Puro Js, hace que sea rápido.
- **Responsive:** básico.

NO SPA

- Tiene que volver a cargar con cada acción(eliminar, editar y añadir).
- Usa el header para dirección del servidor.
- Al usar filtro Js solo oculta filas.

Si MPA

- index.php. Única página.
- Todas las acciones pasan por el servidor.

No AJAX

- Añadir Comic: Post + recarga. fetch() para añadir.
- Editar: Get + recarga. Carga formulario en modal.
- Eliminar: Post + recarga. fetch(). para borrar.
- Filtrar: totalmente en cliente.

Conclusión:

- ComicVault es una MPA con filtros en cliente.
- No es SPA.
- No usa AJAX.