UT7.- EJERCICIOS DE PRÁCTICA — FETCH, PROMESAS Y ASYNC/AWAIT

EJERCICIO 1 — FETCH BÁSICO (LECTURA DE JSON)

OBJETIVO

Entender cómo funciona `fetch` y el flujo básico de promesas.

ENUNCIADO

Dado un archivo `movimientos.json` con este formato:

```
{
  "movimientos": [
    { "pieza": "Peón", "from": "e2", "to": "e4" },
    { "pieza": "Caballo", "from": "g8", "to": "f6" }
  ]
}
```

Tareas:

1. Usa `fetch` con `.then()`
2. Convierte la respuesta a JSON
3. Muestra **solo los movimientos** por consola en este formato:

`Peón: e2 → e4`

No modificar el HTML

EJERCICIO 2 — FETCH CON ASYNC / AWAIT

OBJETIVO

Reescribir el ejercicio anterior usando `async / await`.

ENUNCIADO

Partiendo del mismo archivo `movimientos.json`:

1. Crea una función `async cargarMovimientos()`
2. Usa `await fetch(...)`
3. Usa `await response.json()`
4. Muestra los movimientos en una lista `<ul>` del HTML

Debe verse algo así:

```
Peón: e2 → e4
Caballo: g8 → f6
```

EJERCICIO 3 — MANEJO DE ERRORES CON TRY / CATCH

OBJETIVO

Entender cómo manejar errores reales en peticiones asíncronas.

ENUNCIADO

1. Intenta cargar un archivo que **no existe** (`datos.json`)
2. Usa `try / catch`
3. Si hay error:
   - Muestra el mensaje:
     "No se han podido cargar los datos"
   - No debe romperse la página

El mensaje debe mostrarse en un `<p id="error">`

EJERCICIO 4 — BUSCADOR CON FETCH (NIVEL MEDIO)

OBJETIVO

Simular un buscador real como el del sprint.

ENUNCIADO

Dado un archivo `servicios.json`:

```
{
  "servicios": [
    { "nombre": "Clases de ajedrez", "categoria": "Educación" },
    { "nombre": "Torneo online", "categoria": "Competición" },
    { "nombre": "Análisis de partidas", "categoria": "Entrenamiento" }
  ]
}
```

Tareas:

1. Cargar los servicios con `fetch`
2. Al escribir en un `<input>`, filtrar por nombre
3. Mostrar resultados dinámicamente sin recargar

Usa `async / await`

EJERCICIO 5 — HISTORIAL DE MOVIMIENTOS DINÁMICO (AJEDREZ)

OBJETIVO

Aplicar asincronía + DOM en un contexto real de proyecto.

ENUNCIADO

1. Usa `fetch` para cargar `movimientos.json`
2. Cada vez que se pulse un botón **"Cargar historial"**:
   - Limpia el historial anterior
   - Vuelve a cargar los movimientos
3. Los movimientos deben mostrarse numerados:

```
1. Peón: e2 → e4
2. Caballo: g8 → f6
```

Usa:

- `async / await`
- DOM dinámico
- Código claro y comentado

EJERCICIO 6 — AMPLIACIÓN EJERCICIO 3.- FETCH DESDE INTERNET

EJERCICIO A

Muestra 10 aperturas en lugar de 5.

EJERCICIO B

Muestra solo las aperturas con más de 1000 partidas.

EJERCICIO C

Añade un <input> para filtrar por nombre del movimiento.