Relación de Problemas:
Orden Superior y Asincronía

Bases de Datos - Grado en Estadística y
Doble Grado en Matemáticas y Estadística

Abril 2026

1. Funciones de Orden Superior

Ejercicio 1. Dada una lista de temperaturas registradas en un laboratorio [22,5, 25,0, 19,8, 31,2, 15,4, 28,9], utiliza funciones de orden superior para obtener una nueva lista que contenga solo las temperaturas superiores a 25 grados convertidas a grados Fahrenheit ($F = C \times 1{,}8 + 32$).

Ejercicio 2. (Orden Superior) Crea una función llamada `procesarMedida(valor, transformacion)`. Esta función debe recibir un dato numérico y una función callback que aplique un ajuste. Utilízala para realizar las siguientes tareas de forma independiente:

- Convertir un peso de kilos a libras ($kg \times 2{,}204$).
- Calcular el IVA (21 %) de un coste sanitario.
- Redondear un valor decimal al entero más cercano.

Ejercicio 3. (Implementación) Crea una función llamada `detectarAnomalias(lecturas, testLogico)`. La función debe recibir un array de niveles de glucosa y un callback que devuelva `true` o `false`. La función debe devolver un array con los **índices** donde se cumple la anomalía. *No usar .filter()*, implementa la lógica con un bucle `for` o `forEach`.

2. Asincronía y Promesas

Ejercicio 4. Define una función `esperarAnalisis(muestraId)` que devuelva una Promesa. Tras 2 segundos, si el `muestraId` es par, se resuelve con el

texto “Análisis completado”; si es impar, se rechaza con “Muestra contaminada”.

Ejercicio 5. Explica la diferencia entre usar `.then()/.catch()` y `async/await`. Refactoriza el siguiente código para que use la sintaxis moderna con gestión de errores:

```
fetch('url')
  .then(r => r.json())
  .then(d => console.log(d))
  .catch(e => console.log(e));
```

3. Consumo de APIs (Fetch)

Ejercicio 6. Utilizando la API `https://jsonplaceholder.typicode.com/users`, obtén los datos del usuario con ID 5 e imprime en un solo `console.log` su nombre y las coordenadas de su localización (`lat` y `lng`).

Ejercicio 7. Utiliza la PokeAPI para crear una función `obtenerTodos` que obtenga los datos de nombres de los primeros 20 pokemon y los muestre con un elemento desplegable de HTML. Conectarla con la funcionalidad vista en clase que, dado un Pokemon, visualiza su información en una tarjeta.

Ejercicio 8. Crea un script que consulte la API de Star Wars (`https://swapi.dev/api/people/1/`). Si la respuesta no es correcta (ej. error 404), debe lanzar un error personalizado mediante `throw` que diga “Error: Personaje no encontrado”.

Ejercicio 9. (Filtrado de API) Realiza una petición para obtener los primeros 10 usuarios de JSONPlaceholder (`/users`). Recorre el array recibido y muestra por consola únicamente los nombres de aquellos usuarios cuya ciudad (`address.city`) comience por la letra “R” o “S”.

Ejercicio 10. Reto Final: Obtén los primeros 5 planetas de la SWAPI de forma secuencial. Para cada uno, imprime su nombre y el número total de residentes que tiene (la longitud del array `residents`). Asegúrate de que los resultados se muestren en orden de consulta.