Boletín de 30 Ejercicios: Node.js, MySQL y Express

Caso de Estudio: Taxonomía y Fauna Mundial

1. Bloque 1: Configuración y Operaciones CRUD Básicas

1. Configura una conexión básica usando el driver `mysql2` y verifica que el servidor esté activo.
2. Inserta una nueva clase denominada *Amphibia*.
3. Recupera todos los campos de la tabla *habitats*.
4. Inserta tres familias nuevas asociadas a la clase *Mammalia* en una sola instrucción.
5. Actualiza el nombre científico del *Tigre* a *Panthera tigris tigris*.
6. Elimina cualquier hábitat cuyo clima sea *Árido*.
7. Obtén el *nombre_comun* de las especies con población mayor a 10,000.
8. Busca especies cuyo nombre común contenga la letra *a*.
9. Selecciona las familias cuyo ID sea mayor que 2, ordenadas alfabéticamente.
10. Cuenta el número total de especies registradas en la base de datos.

2. Bloque 2: Consultas Relacionales (JOINS) y Agregación

11. Realiza un `INNER JOIN` para mostrar cada especie junto al nombre de su familia.
12. Muestra el nombre común de la especie y el nombre de su clase (requiere doble `JOIN`).
13. Obtén el promedio de población estimada de todos los mamíferos.
14. Lista todos los hábitats de una especie concreta usando la tabla intermedia *especie_habitat*.
15. Encuentra la población máxima y mínima registrada en la tabla de especies.
16. Muestra el nombre de las familias y cuántas especies tiene cada una (`GROUP BY`).
17. Lista las clases que tienen más de 2 familias asociadas (`HAVING`).
18. Obtén las especies que no tienen ningún hábitat asignado (`LEFT JOIN` y `NULL`).
19. Muestra los nombres científicos en mayúsculas y la longitud de su nombre.
20. Crea una consulta que concatene el nombre común y el científico en una sola columna.

3. Bloque 3: Promesas y Seguridad

21. Implementa un bloque `try/catch` para manejar errores de conexión.
22. Usa un `Pool` de conexiones para realizar 3 consultas simultáneas.
23. Realiza una consulta filtrando por ID usando *Prepared Statements* para evitar inyección SQL.

4. Bloque 4: Full-Stack (Express + Cliente Web)

24. **API REST**: Crea un endpoint `GET /api/clases` que devuelva todas las clases en JSON.

25. **Filtro Dinámico**: Crea una ruta `GET /api/especies/:familiaId` que devuelva los animales de dicha familia.

26. **Interfaz de Usuario**: Crea un archivo HTML con un `<select>` que se llene dinámicamente con las familias de la BD al cargar la página.

27. **Evento Cambio**: Implementa la lógica para que, al elegir una familia en el `select`, se dibuje una tabla con sus especies.

28. **Buscador en Tiempo Real**: Crea un input de texto que, al escribir, solicite al servidor especies cuyo nombre coincida y las muestre.

29. Crea una ruta `POST /api/especies` que permita registrar un nuevo animal. La ruta debe extraer los datos (nombre, familia, población, etc.) directamente del `req.body`. Tras una inserción exitosa, el servidor debe responder con el ID del nuevo registro y un código de estado 201.

30. Implementa una ruta `PUT /api/especies/:id` para actualizar la información de una especie existente. El ID se recibirá por la URL (*params*) y los nuevos datos de población o estado se recibirán en el cuerpo de la petición (*body*). Utiliza una sentencia SQL de tipo `UPDATE`.