

EVALUACIÓN	PARCIAL 1	GRUPOS	TODOS	FECHA	XX/XX/XXXX
MATERIA	Desarrollo full stack				
CARRERA	Analista Programador/Analista en Tecnologías de la Información				
CONDICIONES	- Puntos: 50 - Duración: 2 horas - SIN MATERIAL - Consultas sólo sobre interpretación de la letra				

Sistema de Gestión de Bebederos públicos

En todo el parcial se asumen las importaciones y exportaciones como dadas, no es necesario incluirlas en ningún caso, ni para archivos propios ni paquetes externos.

También se asumen hechos los procesos de login o registro y que hay un token válido disponible en `localStorage` para utilizar, debe enviarse token en las llamadas a la API

La API se asume publicada en la ruta base <https://api.bebederos.uy>.

Node.js

1. Validador con Joi (5 pts.)

Implemente un esquema de validación con **Joi** para el modelo **Bebederos** con las siguientes propiedades:

- **nombre:** string, requerido, mínimo 3 y máximo 50 caracteres.
- **ubicacion:** string, requerido, mínimo 10 y máximo 100 caracteres.
- **estado:** string, requerido, con valores "operativo", "mantenimiento", "fuera de servicio"
- **caudal:** numero, requerido, litros por minuto minimo 0,5 máximo 50.
- **fechaRegistro:** fecha, con fecha por defecto la actual

Además puede crear un validador Joi solo para estado que seria usado en el punto 4.

2. Modelo de Mongoose (5 pts.)

Cree un modelo de **Mongoose** correspondiente al modelo **Bebedero** definido en Joi.

Utilice el nombre de colección **bebederos**.

En este esquema no se esperan validaciones (se asume validado en Joi).

3. Capa de consultas a base de datos (6 pts.)

Implemente la capa de consultas a la base de datos que permita:

- Obtener todos los bebederos (getAllBebederos).
- Crear un nuevo bebedero (createBebedero(data)).
- Actualizar estado por id (updateEstado(id, nuevoEstado)).

4. Controller y Routes (11 pts.)

Cree la capa de **controllers** y **rutas** de Express que conecta con los servicios anteriores:

- **GET /bebederos** → lista todos los bebederos.
- **POST /bebederos** → crea un bebedero, validando con Joi.
- **PATCH /bebederos/:id/estado** → elimina un bebedero por id.

La ruta **POST** deberá utilizar la validación según el esquema de Joi. En el caso del **PATCH** puede validar que el estado que viene en el body, este incluido entre los estados de bebederos o usar un validador Joi.

React

5. Redux Slice (8 pts.)

Cree un **slice de Redux Toolkit** para la entidad **bebederos**, que incluya:

- Estado inicial con un array de bebederos {items:[]}
- Acción **setItems** para guardar la lista completa.
- Acción **addItem** para agregar un nuevo bebedero.
- Acción **updateEstado** para eliminar un bebedero por id.

6. Componente ListaBebederos (9 pts.)

Cree un componente **ListaBebederos** que:

- Use **useEffect** para hacer una petición **GET** a /bebederos.
- Al recibir los datos, dispere la acción **setItems**.
- Renderice los bebederos en **div (p, article etc)** mostrando:
 - o Nombre
 - o Ubicación
 - o Estado
 - o Caudal
- El componente debera recibir por parametro una lista de bebederos, en caso de estar definida se listaran sus datos, y en caso contrario se listaran los datos del store

7. BotonFiltrarOperativos (6 pts.)

Agregar al componente que contenga al anterior que:

- Al hacer clic:
 - o Con un state del componente, que filtre solo los bebederos cuyo estado sea "operativo".
 - o Actualice la vista **mostrando únicamente los bebederos activos** (filtro frontend, sin llamada a la API). El listado general de la parte anterior se mantiene.
 - o Al presionar el botón un número par de veces, se vuelve a mostrar todos los bebederos en cualquier estado, y un numero impar solo los activos (boton tipo toggle)

Se valorará la utilización de nombres y arquitectura que sigan las convenciones vistas en clase, la optimización de la solución y que se respeten las estructuras solicitadas.

En caso de no poder resolver alguna de las partes, se puede suponer como completada cuando se solicite.

Identificar claramente qué parte del ejercicio se está resolviendo.