ACÁMICA

Agenda

Daily

Manejo de estados en selects

Programamos

Squad lead programa: filtros en colección de hoteles

Trainees programan: filtros en colección de hoteles

Daily

Daily

Sincronizando...

Toolbox

¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

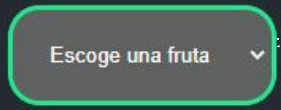
Challenge

¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

Manejo de estados en selects

EL objetivo es controlar este estado con React.

```
<select>
  <option>Escoge una fruta</option>
  <option>Manzana</option>
  <option>Pera</option>
  <option>Banana</option>
</select>
```



El select tiene un estado local, que es el valor seleccionado.

Programamos

¡todos y todas!

Paso 1: Manejadores de evento para fechas

En encuentros anteriores agregamos un evento a un select. El evento onChange, al dispararse, ejecutaba un alert con un mensaje.

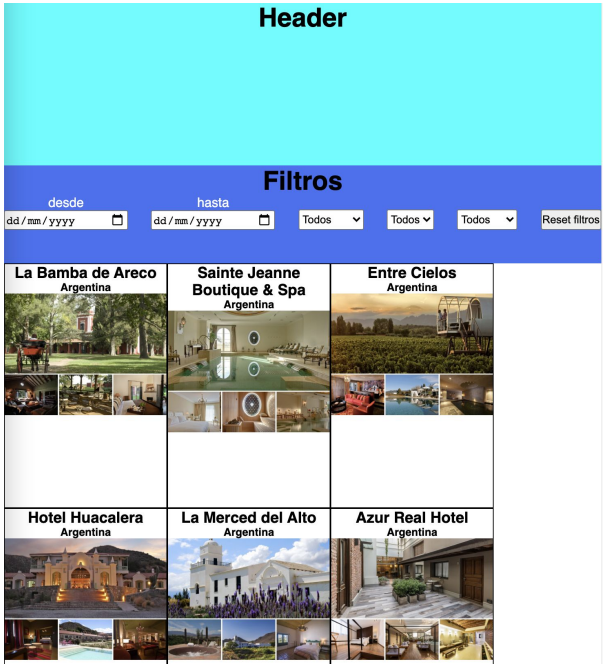
En ese mismo encuentro te pedimos que agregaras 2 select más, junto con los input de fecha. Empecemos desde este punto, que ya tiene esos elementos agregados. En Filtros.jsx, agrega dos manejadores de eventos para las fechas:

```
const manejarFechaDesde = (evento) => {
  alert(evento.target.value)
}
const manejarFechaHasta = (evento) => {
  alert(evento.target.value)
}
```

Agregalos a los inputs correspondiente:

```
<div className="fecha desde">
  <span>desde</span>
  <input type="date" onChange={manejarFechaDesde}/>
</div>
<div className="fecha hasta">
  <span>hasta</span>
  <input type="date" onChange={manejarFechaHasta}/>
</div>
```

Cambia la fecha y asegurate de que los alerts se muestran.



Paso 2: Controlar el estado para select pais

Para controlar el estado de los selects, vamos a importar el useState hook al principio de Filtros.jsx

```
import React, { useState } from "react";
```

Luego, al principio de la función Filtros, declarar un estado para el select país, el estado inicial, será el string “todos”.

```
const [pais, setPais] = useState("todos");
```

Luego, agrega el atributo “value” al select de país:

```
<select value={pais} onChange={manejarCambioPais}>
```

Y finalmente, invoca la función actualizadora del hook, con el valor proveniente de event.target.value (el país seleccionado)

```
const manejarCambioPais = (evento) => {
  setPais(evento.target.value);
};
```

Con estos cambios, ya React sabe qué país está seleccionado, al cambiar el select. Este valor lo almacena en la variable hook `pais`

Paso 3: Controlar el estado para tamaño

Replica los pasos anteriores para controlar el estado en el select de tamaño.

- Declara un hook `const [tamaño, setTamaño] = useState("todos");`
- Agrega el atributo “value” al select con el valor del hook `<select value={tamaño}`
- Modifica el manejador del evento para que actualice el tamaño

```
const manejarCambioTamaño = (evento) => {
  setTamaño(evento.target.value);
};
```

Paso 4: Controlar el estado para precio

Replica los pasos anteriores para controlar el estado en el select de tamaño.

- Declara un hook `const [precio, setPrecio] = useState("todos");`
- Modifica el manejador del evento para que actualice el tamaño `<select value={precio}`

```
const manejarCambioPrecio = (evento) => {
  setPrecio(evento.target.value);
};
```

Paso 5: Controlar el estado para fechas

Para las fechas, debemos aplicar el mismo patrón de los selects. Lo primero es declarar dos hooks para las fechas desde y hasta. Iniciamos el estado para ambas fechas con un string vacio, por que no queremos ninguna fecha preseleccionada.

```
const [desde, setDesde] = useState("");
const [hasta, setHasta] = useState("");
```

Luego, tendremos que especificar el atributo “value” con desde/hasta donde corresponda.

```
<div className="fecha desde">
  <span>desde</span>
  <input value={desde} type="date" onChange={manejarFechaDesde} />
</div>
<div className="fecha hasta">
  <span>hasta</span>
  <input value={hasta} type="date" onChange={manejarFechaHasta} />
</div>
```

Finalmente, completamos los manejadores de eventos para las fechas, invocando la función actualizadora del hook, donde corresponda.

```
const manejarFechaDesde = (evento) => {
  setDesde(evento.target.value);
};
const manejarFechaHasta = (evento) => {
  setHasta(evento.target.value);
};
```

Checkpoint

Si todo ha salido bien hasta ahora, podremos cambiar los valores de los selects y fechas sin problemas. Gracias a los pasos anteriores ahora React sabe de todos los valores que los selects y fechas tienen seleccionados en cada momento.

Paso 6: Resetear filtros

Esta acción consiste en llamar a todas las funciones actualizadoras de cada hook, y setear los valores a sus estados iniciales. Para esto, modifica la función manejarClick, a esto:

```
const manejarClick = () => {
  setPais("todos");
  setTamaño("todos");
  setPrecio("todos");
  setDesde("");
  setHasta("");
};
```

Prueba cambiar los filtros y luego hacer click en el botón de “reset”. ¡Observa como los filtros se devuelven a sus estados iniciales! Te dejamos el ejercicio terminado.

¡BREAK!

Ph. Credit: Drew Coffmann

Programo

Squad lead - Filtrar hoteles, con Javascript (sin React)

1-Filtros en hoteles

Veamos un ejemplo de como filtrar hoteles, solo con Javascript. Para empezar, tendremos que trabajar con la colección de hoteles del proyecto. Asumamos que tenemos los siguientes valores, y queremos filtrar la colección por ellos.

```
import { hotelsData } from "./hotelsData.js";

const pais = "argentina";
const tamaño = "pequeño";
const precio = "$$";
```

2-Estrategia

Para filtrar los hoteles por 3 valores, vamos a tener empezar con la data original “hotelsData” y empezar a aplicar los filtros un valor a la vez (*más adelante veremos una forma resumida de hacer esto*), por ejemplo, el de país:

```
const hotelesFiltradosPorPais = hotelsData.filter((hotel) => {
  // ... filtro pais
});
```

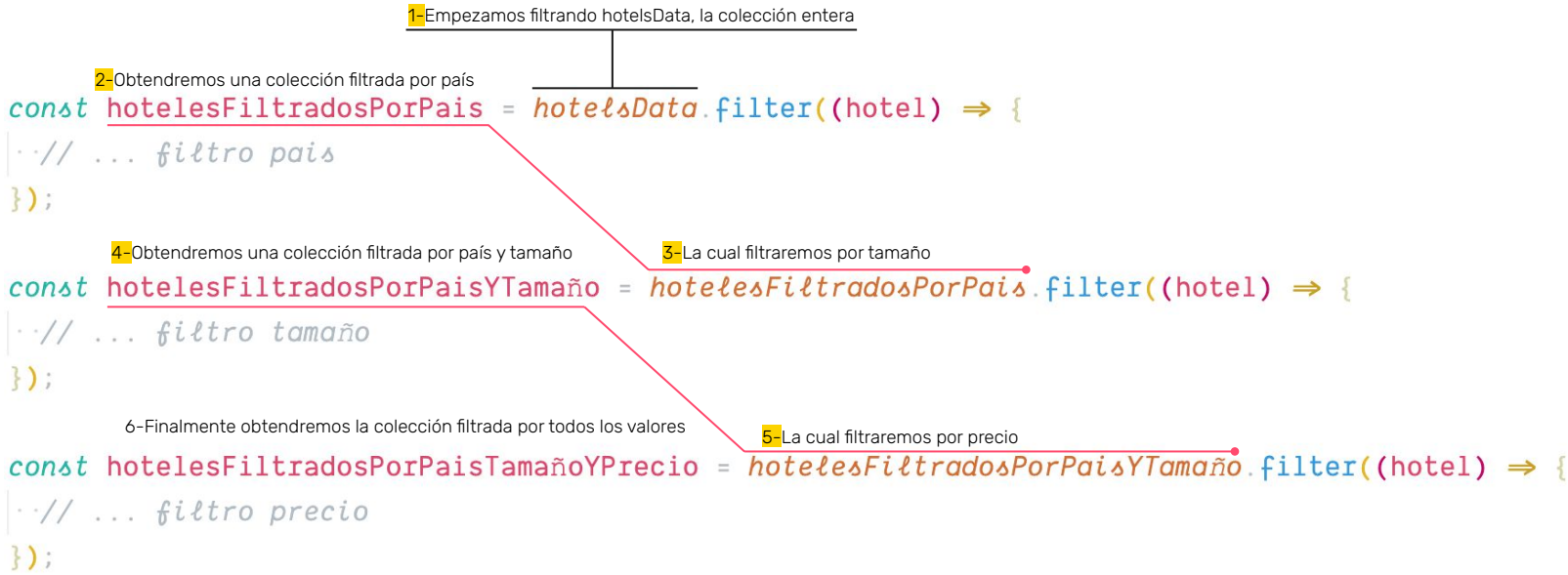
En “hotelesFiltradosPorPais” tendremos un array de objetos de hoteles que ya estan, como su nombre lo indica, filtrados por país. Ahora, podemos crear una colección nueva donde además de filtrados por país, esten filtrados por tamaño.

```
const hotelesFiltradosPorPaisYTamaño = hotelesFiltradosPorPais.filter((hotel) => {
  // ... filtro tamaño
});
```

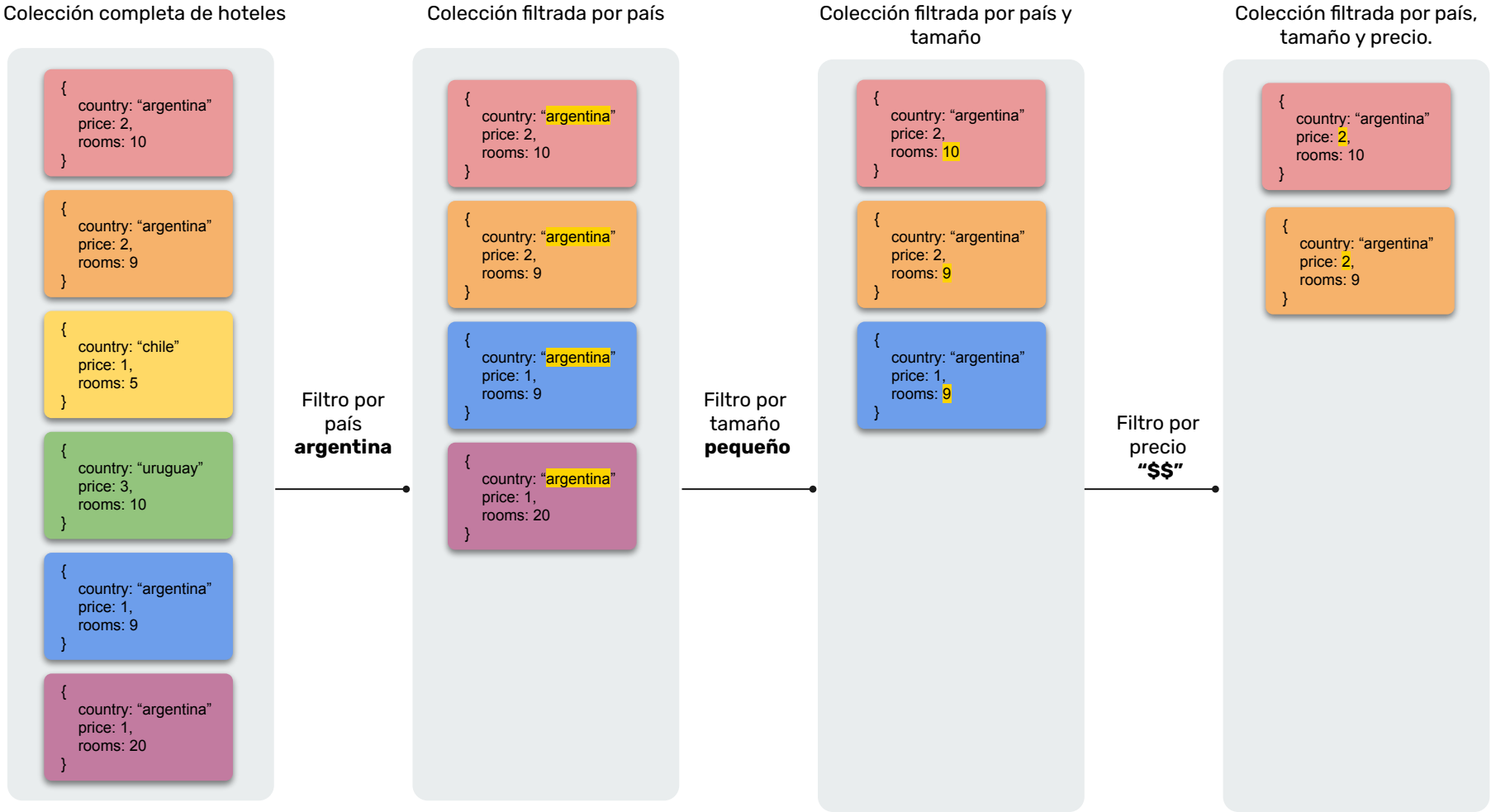
En “hotelesFiltradosPorPaisYTamño” tendremos un array de objetos de hoteles que ya estan, como su nombre lo indica, filtrados por país y tamaño. Ahora, podemos crear una colección nueva donde además de filtrados por país y tamaño, esten filtrados por precio.

```
const hotelesFiltradosPorPaisTamañoYPrecio = hotelesFiltradosPorPaisYTamaño.filter((hotel) => {
  // ... filtro precio
});
```

2-Estrategia



Este patrón lo podremos seguir repitiendo para filtrar por fechas, o cualquier otro valor.


Colección completa de hoteles
{ country: “argentina” price: 2, rooms: 10 }
{ country: “argentina” price: 2, rooms: 9 }
{ country: “chile” price: 1, rooms: 5 }
{ country: “uruguay” price: 3, rooms: 10 }
{ country: “argentina” price: 1, rooms: 9 }
{ country: “argentina” price: 1, rooms: 20 }
Filtro por país argentina
Colección filtrada por país
{ country: “argentina” price: 2, rooms: 10 }
{ country: “argentina” price: 2, rooms: 9 }
{ country: “argentina” price: 1, rooms: 9 }
{ country: “argentina” price: 1, rooms: 20 }
Filtro por tamaño pequeño
Colección filtrada por país y tamaño
{ country: “argentina” price: 2, rooms: 10 }
{ country: “argentina” price: 2, rooms: 9 }
{ country: “argentina” price: 1, rooms: 9 }
Filtro por precio “$$”
Colección filtrada por país, tamaño y precio.
{ country: “argentina” price: 2, rooms: 10 }
{ country: “argentina” price: 2, rooms: 9 }

3-Filtrar por país

Para filtrar por país, podemos hacer algo como esto:

```
const hotelesFiltradosPorPais = hotelsData.filter((hotel) => {
  if (pais === "todos") {
    return true;
  } else {
    return hotel.country.toLowerCase() === pais.toLowerCase();
  }
});
```

Primero, verificamos si el valor del filtro es “todos” y de ser así, retornamos true, lo cual devolvería a “hotelesFiltradosPorPais” cada objeto, es decir, no se aplica un filtro. Pero, si no es “todos”, entonces retornamos el objeto cuyo país, sea igual a la variable país.

Utilizamos .toLowerCase al momento de leer ambos valores (país del filtro y pais del hotel) para evitar inconsistencias de mayusculas y minusculas.

La variable “hotelesFiltradosPorPais” tendra un array de objetos hoteles, todos en Argentina.

Punto de partida

4-Filtrar por tamaño

Si quisiéramos filtrar los hoteles por tamaño y país, podemos hacer algo como esto:

```
const hotelesFiltradosPorTamañoYPais = hotelesFiltradosPorPais.filter((hotel) => {
    if (tamaño === "todos") {
        return true;
    } else {
        if (tamaño === "pequeño") {
            return hotel.rooms <= 10;
        } else if (tamaño === "mediano") {
            return hotel.rooms > 10 && hotel.rooms < 20;
        } else {
            return hotel.rooms >= 20;
        }
    }
}
);
```

Creamos un nuevo array que va a iterar sobre los hoteles previamente filtrados por país, y luego aplicamos la lógica correspondiente para devolver el objeto hotel que tenga el tamaño que se pide.

5-Filtrar por precio

Finalmente, si quisiéramos filtrar los hoteles por precio, país y tamaño, podemos hacer lo siguiente:

```
const hotelesFiltradosPorTamañoPaisYPrecio = hotelesFiltradosPorTamañoYPais.filter((hotel) => {
    if (precio === "todos") {
        return true;
    } else {
        let nivelPrecioNumero = precio.length;
        return hotel.price === nivelPrecioNumero;
    }
}
);
```

Pasamos el string a un número con .length. Esta propiedad nos dirá cuántos "$" hay en precio, y así podremos contrastar contra la propiedad "price", la cual es númerica.

En "hotelesFiltradosPorTamañoPaisYPrecio" obtendremos un array con hoteles filtrados por las 3 condiciones

Programan

Filtrar hoteles, con Javascript (sin React)

Programan

En grupos de 3 (redondear a 2 en caso de que no existan suficientes personas) deberán replicar lo explicado anteriormente. Partiendo de la data de hoteles deberán crear una colección filtrada en base a país, tamaño, y precio.

Este ejercicio busca poner en práctica la atención al detalle, manejo de sintaxis y capacidad lógica.
Acá te damos una guia para empezar.

1. Definan el entorno, Code Sandbox, VS Code, u otro editor de texto donde se sientan en comodidad
2. Importen la data de hoteles y guardenla en un archivo. Asegurense de exportar la colección
3. En otro archivo (preferiblemente index.js), importen la data de hoteles
4. Definan 3 variables, para país, tamaño y precio, como en el punto 1 del "programo". Asignen valores correspondientes. Estas son las opciones posibles para cada variable:

Para país:	**Para precio:**	**Para tamaño:**
"Argentina"	"$"	"pequeño"
"Chile"	"$$"	"mediano"
"Uruguay"	"$$$"	"grande"
"Brazil"	"$$$$"	

5. Una persona por grupo deberá codear solo un filtro, si el grupo es de 2, entre los 2 deberán construir el tercer filtro. Deberán estar de acuerdo con los nombres de las colecciones, ya que dependen entre filtros, como se ve en el punto 2 del "programo"
6. Al terminar, coloquen los filtros en un solo archivo. Deberán mantener el orden de cada uno ya que algunos filtros usan colecciones que fueron creadas previamente, como se vio en la estrategia.

Casos de prueba

Una vez ensamblados los filtros, deberán poner a prueba su creación y ver si resulta como lo esperado. En caso de que fallen, tendrán que ir ajustando y leyendo los errores que la consola arroja. **¡No se preocupen si hay fallos!** En programación debemos ir esculpiendo poco a poco nuestro código hasta que resulte como queremos.

Para los casos de prueba, considera las siguientes estadísticas:

Hay 8 hoteles en Argentina.
- 2 tienen precio en “$”, 4 tienen precio en “$$”, y 2 tienen precio en “$$$$”.
- 1 es pequeño, 5 son medianos y 2 son grandes.

Hay 4 hoteles en Brasil.
- 2 tienen precio “$$$” y 2 tienen precio “$$$$”
- 1 es pequeño, 2 son medianos y 1 es grande.

Hay 5 hoteles en Chile.
- Todos son grandes
- Todos tienen precio “$$$$”

Hay 1 hotel en Uruguay.
- Es pequeño
- Su precio es “$”

Casos de prueba

Considerando los datos anteriores, cambia los valores de tus variables de filtrado (país, precio y tamaño) y analiza si los filtros están produciendo las colecciones de forma correcta. Acá te damos algunas referencias.

```
const pais = "todos";
const tamaño = "todos";
const precio = "todos";
```

Mostrará por consola una colección de 18 objetos.

```
const pais = "argentina";
const tamaño = "todos";
const precio = "todos";
```

Mostrará por consola una colección de 8 objetos.

```
const pais = "chile";
const tamaño = "todos";
const precio = "$";
```

Mostrará por consola un array vacío.

```
const pais = "brasil";
const tamaño = "mediano";
const precio = "$$";
```

Mostrará por consola un array vacío.

```
const pais = "uruguay";
const tamaño = "grande";
const precio = "todos";
```

Mostrará por consola un array vacío.

```
const pais = "todos";
const tamaño = "todos";
const precio = "$$$$";
```

Mostrará por consola una colección de 9 objetos.

```
const pais = "brasil";
const tamaño = "todos";
const precio = "$$$";
```

Mostrará por consola una colección de 2 objetos.

```
const pais = "chile";
const tamaño = "grande";
const precio = "$$$$";
```

Mostrará por consola una colección de 5 objetos.

Necesitas una ayuda? Te dejamos el ejercicio terminado aquí.

Para la próxima

1) Termina el ejercicio de la meeting de hoy.
2) Lee la toolbox 31.
3) Resuelve el challenge.

ACÁMICA