Guía 2: Refactorizando la aplicación

Preparación

En esta guía vas a refactorizar algunas de las funcionalidades de tu aplicación. Vas a mejorar el código aplicando buenas prácticas de programación, pero el funcionamiento de la aplicación no se va a modificar. Para comprobar que todo siga funcionando correctamente luego de hacer los cambios, no olvides **correr los tests y verificar que sigan funcionando correctamente.**

Paso 1: Refactorizá la función reservarHorario(horario) utilizando la función filter.

En el archivo `restaurant.js`, se encuentra la función `reservarHorario(horario)`. Esta función recibe un horario y se encarga de eliminarlo del arreglo de horarios del restaurante. Para esto, recorre todos los horarios y cuando encuentra el que debe eliminar, utiliza la función `splice`para hacerlo.

- `splice(inicio, cantidad)`: recibe un inicio y la cantidad de elementos que debe eliminar a partir de ese inicio. En este caso, el inicio es la posición del elemento que debe eliminar y la cantidad es 1.

En este paso, modificarás la función para que elimine el elemento del arreglo utilizando la función `filter`. Tendrás que filtrar el arreglo de horarios para que solo se devuelvan los elementos que no son iguales al horario que se recibió por parámetro.

Recordá que `filter()` devuelve un nuevo arreglo con los elementos que cumplen con la condición pasada por parámetro. Tendrás que asignarle al restaurant este nuevo arreglo filtrado.

Paso 2: Modularizá la función obtenerPuntuacion().

En el archivo `restaurant.js` se encuentra la función `obtenerPuntuacion()`que se encarga de sumar todas las calificaciones del restaurant y sacar el promedio.

Esta función solo debería encargarse, como su nombre lo dice, de devolver la puntuación, no de sumar los elementos de un arreglo y además, sacar el promedio.

Para resolver este problema, creá dos funciones genéricas para resolver esas cuestiones:

- `sumatoria(numeros)`, que reciba un arreglo de numeros y devuelva su sumatoria.
- `promedio(numeros)`, que sume los elementos de un arreglo y luego calcule su promedio. Esta función debe utilizar la función `sumatoria(numeros)`

Una vez que tengas esas funciones creadas, utilizalas para modularizar la función `obtenerPuntuacion()`.

Paso 3: Modificá los nombre de las funciones obtC(), obtR() y obtH() y de la variables que utilizan por nombres más declarativos.

En el archivo `listado.js` se encuentran estas tres funciones. En los comentarios se explica que hace cada una de ellas y que significan las

variables que utilizan. Cómo seguramente pudiste notar, los nombre de las funciones y de las variables no son muy claros.

En este paso, vas a mejorar el nombre de las funciones y de las variables, para que representen mejor cuál es su objetivo.

Estas funciones son llamadas en `aplicacion.js`, desde `dibujarRubros()`, `dibujarHorarios()` y `dibujarCiudades()`. Cuando cambies el nombre de las funciones, recordá cambiarlo también cuándo se las llama.

Aclaración: en los siguientes pasos de la guía, vamos a referirnos a las funciones a las que vas a cambiarle el nombre como `obtenerRubros()`, `obtenerUbicaciones()` y `obtenerHorarios()`. De todas formas, podés darle el nombre que vos sientas que representa mejor lo que hace.

Paso 4: Eliminá la repetición de código en obtenerRubros(), obtenerUbicaciones() y obtenerHorarios().

Todas estas funciones realizan la acción de eliminar los elementos repetidos de un arreglo con este código:

```
var c2= c.filter(function(elem, index, self) {
        return index === self.indexOf(elem);
    });
```

Este código se repite en las 3 funciones. Creá una función que elimine los elementos repetidos de un arreglo y llamala desde las funciones `obtenerRubros()`, `obtenerUbicaciones()` y `obtenerHorarios()` para evitar la repetición de código.

Paso 5: Refactorizá las funciones obtenerRubros(), obtenerUbicaciones() y obtenerHorarios() aplicando la función map.

Estas funciones recorren todo el arreglo utilizando un ciclo `for` para obtener los atributos que necesitan de cada restaurante. Esta funcionalidad puede realizarse también utilizando la función `map()`.

Modificá esta parte de las funciones aplicando la función `map()`. Vas a notar que tu código queda más limpio y se comprende más cuál es su objetivo.

Paso 6: Modificá la función buscarRestaurant(id) aplicando la función find().

Para encontrar un elemento con un determinado id, se recorre todo el arreglo de restaurantes utilizando un ciclo `for`. Esto también puede resolverse utilizando la función `find()`.

Refactorizá la función `buscarRestaurante(id)` para que encuentre al restaurant con el id que recibe por parámetro utilizando la función `find()`.