

EVALUACIÓN ORDINARIA

Desarrollo Web en entorno cliente
CFGS DAW

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:230302.0946

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

CONTENIDO

1. Ejercicio 1 (5 puntos)	3
1.1 Apartado 1	3
1.1.1 Ejemplos	3
<i>Ejemplo 1</i>	3
<i>Ejemplo 2</i>	4
1.2 Apartado 2	4
1.2.1 Ejemplos	4
<i>Ejemplo 1</i>	4
1.3 Apartado 3	5
1.3.1 Ejemplos	5
<i>Ejemplo 1</i>	5
<i>Ejemplo 2</i>	5
1.4 Gestión de errores	5
2. Ejercicio 2 (5 puntos)	6
2.1 Apartado 1	6
2.2 Apartado 2	7
2.3 Apartado 3	7
2.4 Entorno de desarrollo	9
3. Linter	9
4. Entrega	9
5. Criterios de evaluación	9

EXAMEN EVALUACIÓN ORDINARIA

Revisa los criterios de evaluación para saber que es lo que debes tener en cuenta a la hora de realizar el ejercicio: **no es suficiente que el programa cumpla la función**, debe cumplir además otros criterios para ser considerado un buen código JavaScript.

Recuerda leer bien lo que pide la práctica: no se trata de que lo hagas como pienses que debe funcionar sino como se especifica en la misma.

Puedes encontrar las plantillas de código en <https://github.com/CEED-2022/examen-ordinaria>

1. EJERCICIO 1 (5 PUNTOS)

1.1 Apartado 1

Crea un módulo llamado `search_drinks.js` que exporte por defecto una función `searchDrinks` que admita como parámetro una cadena. La función utilizará el API de <https://www.thecocktaildb.com> para obtener una lista de todos los cócteles que contenga esa cadena.

La función hará lo siguiente:

1. Enviar una petición `GET` a la URL (la cadena se debe transformar con `encodeURIComponent(cadena)`):
`https://www.thecocktaildb.com/api/json/v1/1/search.php?s=<cadena>`
2. La petición devuelve datos en formato JSON. Para cada una de las bebidas del campo `drink`, la función devolverá el nombre, la imagen y un array con los ingredientes. En los ejemplos tienes el detalle de lo que debe devolver la función.
3. Si no se han devuelto bebidas (en la respuesta, el campo es nulo) devolverá un array vacío.

Puedes ver ejemplos de llamadas y respuestas del API en el repositorio con las plantillas de código.

1.1.1 Ejemplos

Ejemplo 1

```
searchDrinks('tur')

[
  {
    name: 'Turkeyball',
    image: 'https://www.thecocktaildb.com/images...',
    ingredients: [ 'Wild Turkey', 'Amaretto', 'Pineapple juice' ]
  },
  {
    name: 'Turf Cocktail',
    image: 'https://www.thecocktaildb.com/images...',
    ingredients: [ 'Dry Vermouth', 'Gin', 'Anis', 'Bitters', 'Orange peel' ]
  }
]
```

Ejemplo 2

```
searchDrinks('patata')  
[]
```

1.2 Apartado 2

Crea un módulo llamado `search_ingredients.js` que exporte por defecto una función llamada `searchIngredients` que admita como parámetro un array de cadenas con ingredientes de cócteles. La función debe hacer lo siguiente:

- Para cada uno de los ingredientes lanzará una petición `GET` al API de TheCocktailDB para obtener los datos de ese ingrediente. La URL será la siguiente (el ingrediente se debe transformar con `encodeURIComponent(ingrediente)`, y es indiferente para el API si la cadena del ingrediente contiene mayúsculas):
`https://www.thecocktaildb.com/api/json/v1/1/search.php?i=<ingrediente>`
- Las peticiones de todos los ingredientes se deben lanzar en paralelo.
- No se deben realizar peticiones duplicadas. Dos ingredientes que tengan las mismas letras pero diferente capitalización (por ejemplo, `Lemon` y `lemon`) se consideran el mismo ingrediente.
- Devolverá un array de objetos, cada uno con el nombre que haya devuelto el API y la descripción de cada ingrediente. No hace falta que esté ordenado.
- Puedes suponer que el API siempre va a devolver datos para todos los ingredientes que le pases a la función (aunque puede devolver una descripción nula)
- En algunos casos, el API puede devolver una descripción nula.

1.2.1 Ejemplos

Ejemplo 1

```
const ingredients = ['Gin', 'Tonic Water', 'Tonic Water', 'Anis']  
searchIngredients(ingredients)  
  
[  
  {  
    name: 'Gin',  
    description: 'Gin is a distilled alcoholic drink...'  
  },  
  { name: 'Anis', description: null },  
  {  
    name: 'Tonic Water',  
    description: 'Tonic water (or Indian tonic water) is ...'  
  }  
]
```

1.3 Apartado 3

Crea un módulo llamado `search_cocktails.js` que exporte por defecto una función llamada `searchCocktails`. La función debe admitir un único parámetro y devolverá un objeto con dos campos: el primero contendrá la lista de cócteles con ese nombre (idéntica a la del apartado 1) y el segundo, una lista con los ingredientes de dichos cócteles (idéntica a la del apartado 2)

Debes utilizar los módulos desarrollados en los apartados anteriores.

1.3.1 Ejemplos

Ejemplo 1

```
searchCocktails()

{
  drinks: [
    {
      name: 'Turkeyball',
      image: 'https://www.thecocktaildb.com/images/...jpg',
      ingredients: [ 'Wild Turkey', 'Amaretto', 'Pineapple juice' ]
    },
    {
      name: 'Turf Cocktail',
      image: 'https://www.thecocktaildb.com/images/...jpg',
      ingredients: [ 'Dry Vermouth', 'Gin', 'Anis', 'Bitters', 'Orange peel' ]
    }
  ],
  ingredients: [
    {
      name: 'Wild Turkey',
      description: 'Wild Tu...'
    },
    {
      name: 'Amaretto',
      description: 'Amar...'
    },
    { name: 'Anis', description: null },
    { name: 'Bitters', description: null }
    ...
  ]
}
```

Ejemplo 2

```
searchCocktails('patata')

{ drinks: [], ingredients: [] }
```

1.4 Gestión de errores

En cada uno de los apartados, la función debe lanzar una excepción si alguna de las peticiones falla. Si el fallo está al hacer el `fetch` devolverá el error que haya dado `fetch` y si el `status` de una petición no es 200 devolverá el error `"Error: <status>"` siendo `<status>` el status HTTP devuelto al realizar la petición.

2. EJERCICIO 2 (5 PUNTOS)

2.1 Apartado 1

Crea un componente React para obtener y mostrar una lista de cócteles. El componente admitirá como propiedades un nombre y un tema (**claro** u **oscuro**).

Para obtener la lista el componente llamará a la función `apiCall` de `api.js`, proporcionada en las plantillas de código, con el nombre pasado como propiedad. Esta función devuelve una de dos listas de forma aleatoria. Un ejemplo de los datos que devuelve esta función sería:

```
{
  drinks: [
    {
      name: 'Turkeyball',
      image: 'https://www.thecocktaildb.com/images/...jpg',
      ingredients: [ 'Wild Turkey', 'Amaretto', 'Pineapple juice' ]
    },
    {
      name: 'Turf Cocktail',
      image: 'https://www.thecocktaildb.com/images/...jpg',
      ingredients: [ 'Dry Vermouth', 'Gin', 'Anis', 'Bitters', 'Orange peel' ]
    }
  ],
  ingredients: [
    {
      name: 'Wild Turkey',
      description: 'Wild Tu...'
    },
    {
      name: 'Amaretto',
      description: 'Amar...'
    },
    { name: 'Anis', description: null },
    { name: 'Bitters', description: null }
  ],
  ...
}
```

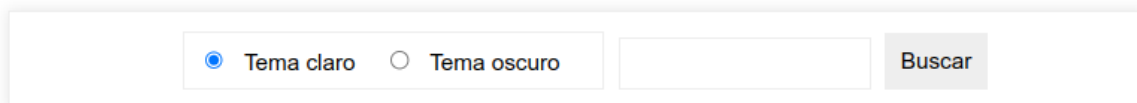
El HTML que debe generar es el siguiente, añadiendo tantos `li` como cócteles diferentes se reciban y cambiando `tema-claro` por `tema-oscuro` según el valor del tema pasado :

```
<div class="cocktail-list">
  <h2>Cocktail List</h2>
  <ul>
    <li>
      <div class="cocktail tema-claro">
        <h3>Turkeyball</h3>
        <div class="image-container"></div>
        <div class="ingredients">
          <h4>Ingredients:</h4>
          <ul>
            <li>Wild Turkey</li>
            <li>Amaretto</li>
            <li>Pineapple juice</li>
          </ul>
        </div>
      </div>
    </li>
  </ul>
</div>
```

No debe cargar la lista de cócteles a menos que se le pase un nombre diferente al que ya tenía. Si se pasa un nombre vacío debe generar un `ul` vacío en la lista de cócteles.

2.2 Apartado 2

Crea un componente que tenga dos radio buttons con los valores Tema claro y Tema oscuro, y un formulario con un campo de texto y un botón buscar:

El formulario está contenido en un div con la clase 'form-container' y 'tema-claro'. Dentro, hay un div con la clase 'theme-selector' que contiene dos radio buttons: 'Tema claro' (seleccionado) y 'Tema oscuro'. A la derecha de estos, hay un campo de texto vacío y un botón 'Buscar'.

El HTML generado debe ser el siguiente:

```
<div class="form-container tema-claro">
  <div class="theme-selector">
    <label>
      <input type="radio" name="tema" value="claro" checked="">
      Tema claro
    </label>
    <label>
      <input type="radio" name="tema" value="oscuro">
      Tema oscuro
    </label>
  </div>
  <form>
    <input type="text" value="">
    <button>Buscar</button>
  </form>
</div>
```

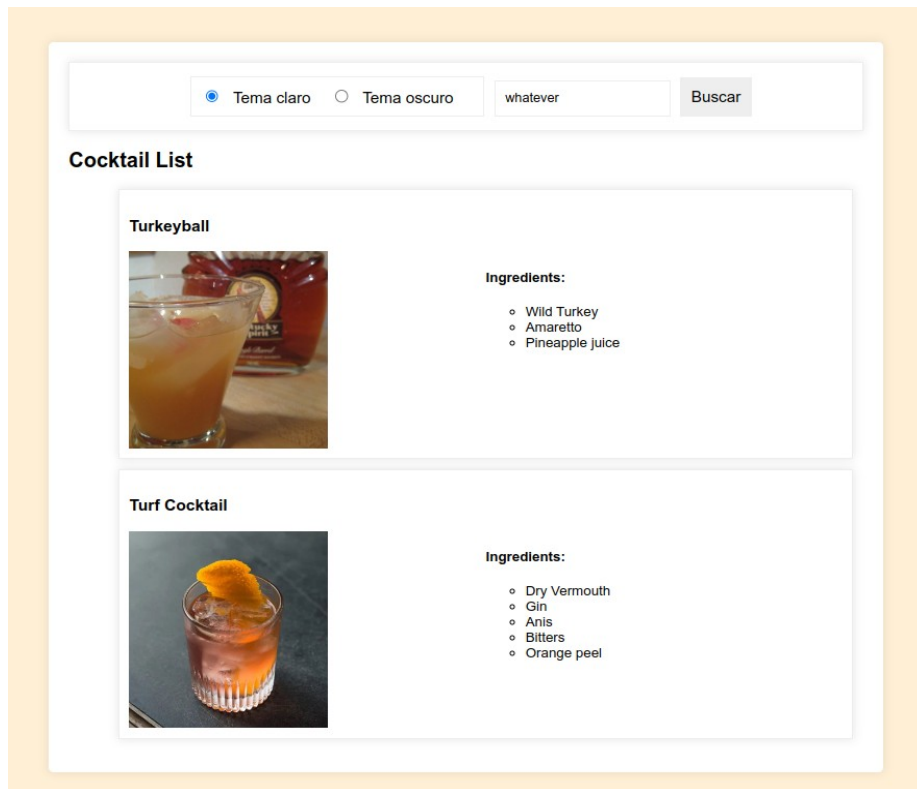
2.3 Apartado 3

Con los dos componentes anteriores, crea una aplicación utilizando React para obtener una lista de cócteles a partir de un nombre. Debe generar el mismo HTML que en las plantillas de código del repositorio del examen.

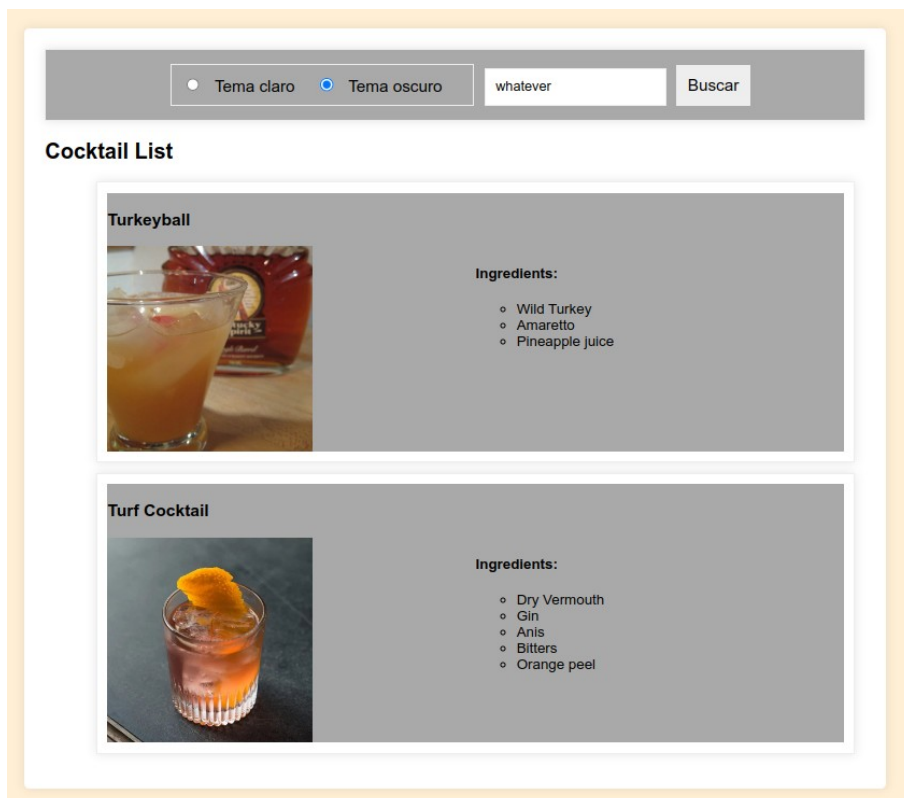
El funcionamiento será el siguiente:

- Cuando se pulse buscar se cargará la lista con el resultado de llamar a `apiCall` con ese nombre
- Si el nombre está vacío, se borrará la lista
- Cuando se pulse en un radio button se cambiará inmediatamente al tema correspondiente añadiendo los estilos adecuados.

La aplicación tendrá este aspecto con el tema claro:



Con el tema oscuro:



2.4 Entorno de desarrollo

Además de programar el código, deberás montar un entorno de desarrollo con Webpack. Debe admitir tres scripts:

- `yarn start` lanzara `webpack-dev-serve` con autoreloading
- `yarn build` generará los ficheros distribuibles de la aplicación en el directorio `dist`
- `yarn eslint` ejecutará el linter sobre los ficheros JavaScript del directorios `src`

Todo el código HTML, CSS y js, así como las imágenes, estarán en el directorio `src`.

Se debe poder lanzar la aplicación y funcionar correctamente con el código generado en `dist`. No incluyas ese código en tu entrega, debe poder generarse automáticamente con `yarn build`

3. LINTER

Para que los ejercicios se corrijan el linter no debe devolver ningún error. En caso de que el linter devuelva un error, la máxima nota del ejercicio será de 1.

Las reglas del linter son las que están especificadas en `.eslintrc.json` en el repositorio con las plantillas del código. No puede modificarse este fichero. Asimismo no se admitirá deshabilitar ninguna de las reglas de eslint por ningún medio: en ese caso se procederá como si el programa hubiese fallado el linter.

4. ENTREGA

Para la entrega debes eliminar los directorios `node_modules` y `dist` y comprimir el directorio de cada ejercicio en un único fichero `.zip` o `.gz`. Cada ejercicio se entregará en la tarea del curso habilitada a tal efecto.

5. CRITERIOS DE EVALUACIÓN

- El programa es correcto, realiza la función que se solicita en el enunciado
- Se han utilizado estructuras del lenguaje adecuadas: bucles, condicionales, operadores, etc.
- Se ha estructurado correctamente el código utilizando módulos
- Se han utilizado variables y constantes de forma adecuada
- Se utilizan correctamente y cuando corresponda los tipos de datos y objetos predefinidos del lenguaje (Arrays, objetos planos, Map, Set, etc.)
- Se han utilizado funciones para estructurar el código, definiendo y utilizando parámetros y valores de respuesta de forma adecuada
- El programa es lo más sencillo posible para realizar su función.
- No existe código repetido: se han extraído los comportamientos comunes a funciones y se

ha intentado hacer el código genérico.

- El programa cumple todas las reglas definidas para el linter.
- Se han utilizado correctamente las funciones de React