Los ejercicios son para que practiquéis vosotros. **Solo se entregarán las prácticas** indicando en la tarea la URL donde encontrar el desarrollo en GitHub. Es obligatorio realizarlos con el entorno de desarrollo que se ha definido al inicio del curso.

Para organizarlo mejor, vamos a crear una página home (index.html) y ahí añadiremos enlaces a cada una de las prácticas de forma que se pueda navegar desde esa página a cada una de las prácticas que hacemos.

Ejercicios sencillos:

Ej1: Usa fetch para obtener datos de una API pública y mostrarlos en la consola. Usa la API pública https://reqres.in/api/users?page=1. Haz una solicitud GET para obtener la lista de usuarios. Finalmente muestra el resultado en la consola (el objeto)

Ej2: Usa la misma API: https://reqres.in/api/users?page=1. Recupera la lista de usuarios. Muestra el first_name y email de cada usuario en un elemento <ul>.

Ej3: Sobre el código anterior añade un manejo de errores utilizando catch y muestra un mensaje como "Error al cargar los datos" en la página si falla. Puedes verificar la propiedad response.ok para saber si la solicitud fue exitosa. Para probarlo modifica la url por una que no exista.

Ej4: Usa AJAX para enviar datos a un servidor. Usa el endpoint https://reqres.in/api/users. Envía un objeto JSON con un name y un job usando el método POST. Muestra la respuesta del servidor en la consola. Usa el método fetch con las opciones method y body.

Práctica 1 (Obligatoria)

La página https://randomuser.me/ permite obtener datos aleatorios de personas pensando en que los desarrolladores y otros profesionales puedan utilizarlos en sus pruebas y test.

Las instrucciones de la API de este servicio gratuito están en la URL: https://randomuser.me/documentation. En todo caso la idea es hacer peticiones vía GET a la URL: https://randomuser.me/api/

Se pueden pasar parámetros para indicar cuántos usuarios aleatorios deseamos, el sexo, política de contraseñas, páginas, formato de respuesta, etc. En la página de documentación viene un ejemplo de la estructura]SON de las respuestas.

Como resumen indicamos que es un objeto formado por dos propiedades: *results* e *info*.

- La primera es un array donde cada elemento lo forma un objeto con los datos del usuario aleatorio.
- La propiedad *info* contiene otros detalles entre los que destaca una semilla que permite repetir una petición con los mismos datos y datos de paginación (cuando deseamos dividir en páginas los resultados se debe usar la misma semilla).

La aplicación mostrará la foto, nombre, apellido, email, dirección y estado al que pertenece el usuario. Cada vez que actualicemos la página, se pedirá otro usuario. Ejemplo de resultado:

laerke sorensen
Email: lserke.s0rensen@example.com
201 msllevaenget
nykobing sj. (MIDTJYLLAND)

Práctica 2 (Obligatoria)

Se proporciona el siguiente código HTML y parte del código Javascript, para utilizarlo si se desea (cogerlo de aules, fuera de esta tarea).

Para esta práctica se van a utilizar dos APIs (se recomienda utilizar la herramienta Postman para ver qué pasa cuando se hace un GET o un POST a las urls que se detallan a continuación):

-https://reqres.in/api/users/{number}, método GET, para obtener un listado de usuarios

-https://httpbin.org/post, método POST, para enviar un usuario para que se cree en la base de datos fictícia.

En la pantalla se mostrarán dos campos que debe introducir el usuario:

- El número de segundos de retraso para la petición de obtención de datos.

- El número de usuario que se recuperará de la base de datos ficticia: Números válidos del 1 al 12.

No se debe hacer ningún tipo de validación sobre esos campos.

Habrá un botón de enviar. Al clicarse deben pasar las siguientes cosas:

1) Llamar a la primera URL para obtener el usuario que se ha introducido en pantalla, con el retraso en segundos que se ha introducido en pantalla. Esto sería un ejemplo de URL a invocar en caso de haberse introducido 2 segundos de retraso, usuario número 5:

https://reqres.in/api/users/5?delay=5

Hay que controlar el caso de que el usuario introduzca un número superior a 12 o una letra. En ese caso, la llamada va a devolver un HTTP 404 (El response.ok será false y el

response.status tendrá el 404). En ese caso, hay que lanzar una excepción que capturará nuestro catch correspondiente.

Si todo ha ido bien, se rellenarán los campos de pantalla ID y EMAIL, con los valores de ID, EMAIL que devuelve el API.

2) Llamar a la segunda URL (https://httpbin.org/post), para hacerle un POST del usuario que se ha obtenido en la primera llamada (json dentro del atributo "data" de la primera llamada). En el body del post debe ir el usuario (en formato JSON). Aquí estamos emulando la grabación de un usuario en la base de datos. Esa llamada post devuelve un JSON, que dentro del campo "json" tiene lo que se ha enviado en el body. Hay que recuperarlo y rellenar el campo name de pantalla, con el valor del atributo "first_name" que viene en la respuesta.

3) Para acabar hay que rellenar el campo status, con un 200 en caso de que haya ido todo bien o el response.status en caso de que hayamos introducido un número superior a 12 en el "número de usuario".

Para resolver esta práctica, se puede hacer mediante promesas o mediante async/await.

Se muestra en el siguiente vídeo el resultado de ambos casos: una ejecución correcta y una incorrecta (mirar en aules, fuera de este archivo).

Práctica 3 (Opcional)

La agencia meteorológica española **AEMET** dispone de una API para datos abiertos en la que proporciona numerosas posibilidades de obtener información meteorológica.

Para poder obtener los datos de esta agencia, necesitamos darnos de alta como desarrolladores en la dirección:
https://opendata.aemet.es/centrodedescargas/altaUsuario

Las instrucciones de funcionamiento de la API de AEMET están en: https://opendata.aemet.es/dist/index.html

En el alta necesitamos indicar un email que tendremos que confirmar para que se nos conceda una API Key, una clave única que se otorga a cada desarrollador. Esa API la deberemos enviar cada vez que hagamos peticiones a la AEMET.

Nuestra aplicación, inicialmente, muestra un cuadro de texto en el que introducir nuestra API y un botón de cargar mapa:

Imagen del día de la AEMET

Pega tu API Key

Cargar mapa

Tras pegar nuestra API y pulsar el botón ***Cargar mapa,*** solicitaremos el mapa meteorológico del día en la dirección: https://opendata.aemet.es/opendata/api/mapasygraficos/analisis

Se quitarán los controles de formulario y se mostrará (en horizontal) el mapa. Hay que recordar que debemos enviar vía GET un parámetro llamado apikey, con nuestra clave (la cual se obtiene del formulario).

Los datos que llegan nos ofrecen un objeto JSON donde la propiedad llamada **datos,** es una URL a la imagen del mapa. La petición http nos proporciona ese enlace, pero no el mapa en sí. Por ello, hay una segunda petición (a esa URL devuelta por la propiedad datos) que es la que nos proporciona el mapa.