Acerca de la API.

**API REST** que da acceso a cientos de personajes, imágenes, y episodios.

También está basada en **GraphQL**, que es un lenguaje de consulta de APIs y entorno de ejecución que permite al cliente obtener solo los datos que necesita de la API.

**Características.**

* **Consulta personalizada:** el cliente especifica los datos que quiere, reduciendo sobrecargas de datos no útiles.
* **Datos recuperados jerárquicamente**: La estructura jerárquica refleja la forma que se necesitan en el cliente.
* **Único Endpoint**: Toda interacción con la API usa un solo endpoint, simplificando la gestión.
* **Consulta y mutación**: No solo se consultan datos sino también se modifican a través de ‘mutations’.

**Consumo del API Rick & Morty.**

Se hace un consumo en REST seguida en la documentación del API. La URL base toda la información disponible, todas las solicitudes son **GET** y pasan por **HTTPS,** y las respuestas con en formato **JSON.**

Texto

Descripción generada automáticamente

Los recursos son Personaje, Ubicación y Episodio.

**Información y paginación.**

Se paginan automáticamente las respuestas, recibirá hasta 20 docs por página. Y cada respuesta contiene información en **objeto ‘{}’** acerca de la petición: **key, type and description**.

**Codificación.**

Llamamos la URL del api y creamos la función que pedirá al personaje.

Utilizamos **fetch** para hacer el consumo del api, con la URL como primer parámetro, y (opcional) el método, cabeceras, cuerpo, etc.

Texto

Descripción generada automáticamente

**Fetch devuelve una promesa**, por lo que no sabemos el tiempo que le tome realizar la consulta a la API. Por ello es bueno usarlo dentro de una función asíncrona ‘async’ al inicio de la función y un ‘await’ para estar en espera de que se resuelva o no.

Texto

Descripción generada automáticamente

La función contiene al metodo ‘try/catch’ para el manejo de errores al momento de realizar la petición con la API. Con **try** ejecuta la consulta hacia la API y **catch** se encarga de ejecutarse cuando ocurre un error y detiene la ejecución.

Respuesta del API en navegador:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Al manejarlo con una respuesta directa en JSON con ‘{ results }’ obtenemos únicamente de los ‘characters’ (personajes) solicitados.

Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Creación de interfaz visual con HTML**

Usamos elementos básicos como un titulo, un main, un div para las cartas de cada personaje.

Con ayuda del mismo script del api, creamos una función que contenga el control del DOM:

Utilizamos un quweySelector para el contenedor de las cartas

Creamos ‘createElement()’ un titulo en ‘h5’, y ese mismo titulo nos dará el nombre del personaje ( carácter.name )

Texto

Descripción generada automáticamente

Al solicitar el nombre, podemos hacerlo de esta forma o con un comando **destructurado**: que se refiere a la **destructuración de objetos** para procesar datos que da una API.

Extraemos los elementos que queremos de los personajes (character):