Sesión 13 – Semana 4

Asincronías y Verbos HTTPS

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia

Contenido

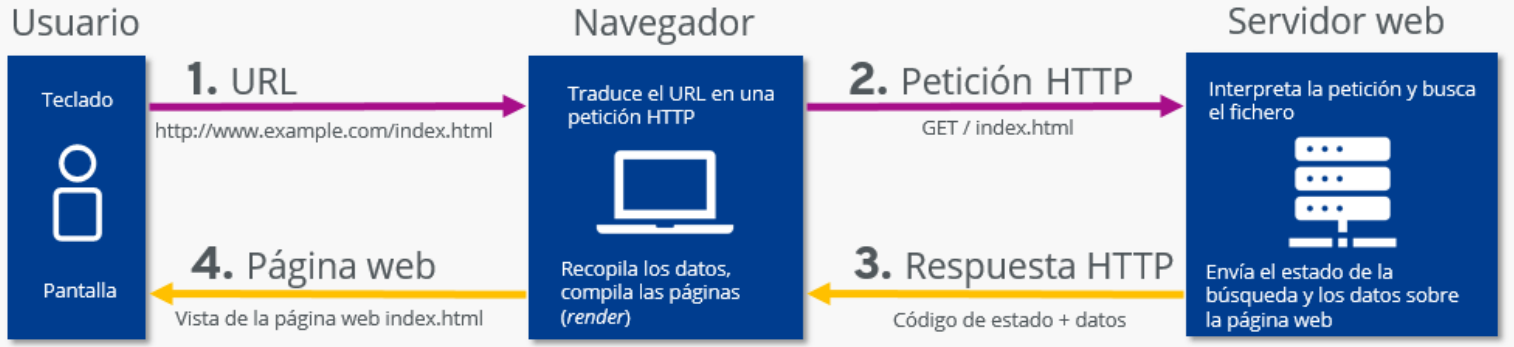
Verbos HTTPS

1. Definición de HTTP
2. Verbos HTTP
3. Fetch
    1. GET
    2. POST
    3. PUT
    4. PATCH
    5. DELETE

Verbos HTTPs

¿Qué es HTTP?

(Hyper Text Transfer Protocol- Protocolo de transferencia de hipertexto) Es un protocolo que especifica las reglas de la comunicación entre los navegadores y servidores. Sigue el clásico **modelo cliente-servidor**, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo.



Verbos HTTPs

¿Cuales son?

Una de las especificaciones de este protocolo son sus verbos, estos nos ayudan a indicar acciones:

1. **GET:** Lo utilizamos para solicitar datos o recursos específicos. Por ejemplo, obtener un usuario de Twitter en función de su nombre de usuario.
2. **HEAD:** Es similar a una petición GET pero sin contenido, sólo traer los encabezados. En ejemplo de su uso sería cuando vamos a utilizar APIs, para comprobar si lo que vamos a enviar es correcto y puede ser procesado.
3. **POST:** Envía datos a un recurso para su creación. Por ejemplo, crear un nuevo registro de usuario con nombre, edad y dirección de correo electrónico.
4. **PUT:** Es utilizado para actualizar un recurso. Por ejemplo, actualizar la dirección de correo electrónico de un usuario.
5. **PATCH:** Actualiza un sección especifica de un recurso.
6. **DELETE:** Elimina por completo un recurso. Por ejemplo, eliminar un usuario de la base de datos.

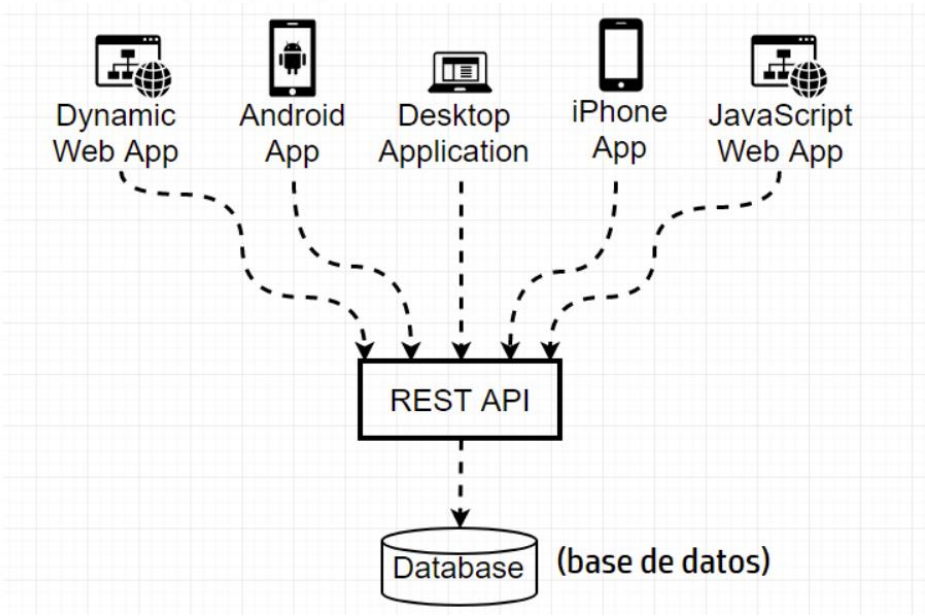
Fetch (URL, OBJECT)

¿Que es?

Es una función nativa de JavaScript para consumir recursos de forma asíncrona de forma simple y fácil, que devolverá una `PROMISE` que será aceptada cuando reciba una respuesta y sólo será rechazada si hay un fallo de red o si por alguna razón no se pudo completar la petición. `Fetch` te permite trabajar con **REST APIs** y con opciones adicionales, como almacenar datos en caché, leer respuestas de transmisión y más.

una **REST API:**

1. permite enviar y extraer datos de un almacén de datos.
2. Tiene 3 elementos: La solicitud (request), la respuesta (response) y los encabezados (header).
    1. **Request**: los datos que envías a la API, como una identificación de pedido (id) para obtener los detalles del pedido.
    2. **Response**: Los datos que obtengas del servidor después de una solicitud exitosa o fallida.
    3. **Header**: Metadatos adicionales que se mandan a la API para ayudar al servidor a comprender qué tipo de solicitud se está mandando, por ejemplo, “content-type” (tipo de contenido).



Fetch (URL, OBJECT)

```
/**
 * Fetch API en JavaScript para hacer peticiones HTTP
 *
 * @author parzibyte
 */
const URL = "https://httpbin.org/post";
const OPCIONES_PETICION = {
    method: "POST",//GET, PUT, HEAD, POST, DELETE...
    headers: {
        "X-Hola": "Valor",
        "Content-Type": "application/json",
    },
    body: "El cuerpo",
};

fetch(URL, OPCIONES_PETICION)
    .then(resultadoRaw => {
        // Lo decodificamos como texto plano
        return resultadoRaw.text();
        // Se pueden usar los siguientes métodos
        resultadoRaw.arrayBuffer();
        resultadoRaw.blob();
        resultadoRaw.formData();
        resultadoRaw.json();
    })
    .then(resultadoDecodificado => {
        // Hacer algo con el resultado
    })
    .catch(error => {
        console.log("Ocurrió un error en la petición: ", error);
    });
```

Estructura

```
const options = {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify(jsonData)
};
```

Objeto que especifica el método de la petición a realizar

Métodos de procesamiento

Método	
STRING .text()	Devuelve una promesa con el texto plano de la respuesta.
OBJECT .json()	Idem, pero con un objeto **json**. Equivalente a usar **JSON.parse()**.
OBJECT .blob()	Idem, pero con un objeto **Blob** (binary large object).
OBJECT .arrayBuffer()	Idem, pero con un objeto **ArrayBuffer** (buffer binario puro).
OBJECT .formData()	Idem, pero con un objeto **FormData** (datos de formulario).
OBJECT .clone()	Crea y devuelve un clon de la instancia en cuestión.
OBJECT Response.error()	Devuelve un nuevo objeto **Response** con un error de red asociado.
OBJECT Response.redirect(url, code)	Redirige a una **url**, opcionalmente con un **code** de error.

Fetch

GET

Método HTTP por defecto es GET

```
// Solicitud GET (Request).
fetch('https://api.github.com/users/manishmshiva')
    // Exito
    .then(response => response.json())  // convertir a json
    .then(json => console.log(json))    //imprimir los datos en la consola
    .catch(err => console.log('Solicitud fallida', err)); // Capturar errores
```

```
fetch('https://api.github.com/users/manishmshiva', {
  method: "GET",
  headers: {"Content-type": "application/json;charset=UTF-8"}
})
.then(response => response.json())
.then(json => console.log(json));
.catch(err => console.log(err));
```

Empleando los HEADERS.

Fetch

POST

```
// datos mandados con la solicutud POST
let _datos = {
  titulo: "foo",
  principal: "bar",
  Id:1
}

fetch('https://jsonplaceholder.typicode.com/posts', {
  method: "POST",
  body: JSON.stringify(_datos),
  headers: {"Content-type": "application/json; charset=UTF-8"}
})
.then(response => response.json())
.then(json => console.log(json));
.catch(err => console.log(err));
```

La propiedad “body” pasa una cadena JSON como input en cadena JSON, y los encabezados (headers) deben ser un objecto JSON.

Fetch

PUT

```
fetch('http://localhost:3005/users/1', {
        method: 'PUT',
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify({ mail: 'ñññññ', password: 'ñññññ' })
 })
.then(res => res.json())
.then(res=> {
      console.log(res);
});
```

Para actualizar el recurso específico se requiere incluir el id del elemento en la url.

Fetch

DELETE

```
fetch('http://localhost:3005/users/2', {
      method: 'DELETE',
})
.then(res => res.json())
.then(res=> {
      console.log(res);
});
```

Para actualizar el recurso específico se requiere incluir el id del elemento en la url.

Fuentes

1. https://developer.mozilla.org/es/docs/Web/HTTP
2. https://www.freecodecamp.org/espanol/news/tutorial-de-fetch-api-en-javascript-con-ejemplos-de-js-fetch-post-y-header/
3. https://lenguajejs.com/javascript/peticiones-http/fetch/
4. https://pablomonteserin.com/curso/javascript/ejemplos-api-fetch/

MAKAIA

WWW.MAKAIA.ORG

Info: comunicaciones@makaia.org

Corporación MAKAIA
Medellín, Colombia
Carrera 43A – 34-155. Almacentro
Torre Norte, Oficina 701
Teléfono: (+574) 448 03 74
Móvil: (+57) 320 761 01 76

@makaiaorg