Ruteo

Una petición HTTP es un requerimiento que un cliente realiza a un servidor bajo el protocolo HTTP. El protocolo HTTP utiliza una diferentes de métodos estándar para ejecutar estas peticiones. Los métodos principales son:

- **GET**: con el cual solicitamos datos al servidor
- **POST**: se utiliza para enviar datos al servidor
- **PUT**: para la modificación de datos que ya se encuentran en el servidor
- **DELETE**: para eliminar datos.

En todos los métodos, los datos viajan desde el cliente hasta el servidor.

Para GET y el DELETE, los datos viajan en el encabezado, ya que se trata sólo de un dato de identificación para luego recibir información desde el servidor.

En el caso del POST la información viaja en el cuerpo del mensaje, del mismo modo suele suceder en el PUT.

Una ruta puede ser por ejemplo "/producto" o "/api/cliente", etc.

Las rutas son definidas en el backend, en etapa de desarrollo.

Una petición completa debe indicar la ruta y el método HTTP.

Método GET

- Los datos viajan de manera visible en la URL (limitado a 2000 caracteres).
- No es posible enviar datos binarios (imágenes, archivos, etc.)
- La URL y la información codificada se separan por un símbolo de **?** y los bloques de datos se separan con **&**:

```
www.ejemplo.com/index.htm?key1=value1&key2=value2&key3=value3...
```

Método POST

- Los datos viajan a través del **body del HTTP Request** (no aparece en la URL)
- No tiene límite en la cantidad de información a enviar.
- La información proporcionada no es visible, por lo que se puede enviar información sensible.
- Se puede usar para enviar datos binarios (archivos, imágenes...).

Ruteo en Express: Direccionamiento básico

Las rutas, en combinación con un método de solicitud de HTTP, definen los puntos finales End Point en los que pueden realizarse las solicitudes.

Tiene 2 componentes básicos:

- una vía de acceso (URI) o ruta
- un método de solicitud HTTP específico (GET, POST, etc.)

Estructura de una ruta

```
app.METHOD(PATH, HANDLER)
```

app: instancia de express
METHOD: método de solicitud HTTP (GET, POST, etc.)
PATH: vía de acceso al servidor
HANDLER: función que se ejecuta cuando se correlaciona la ruta

Ejemplo básicos de rutas

```
app.get('/producto', (req,res) =>{
    res.send('Responda a requerimiento por get en la ruta /producto')
})
```

```
app.post('/', (req, res) => {
    res.send('Responda a requerimiento por post en la ruta / ')
})
```

```
app.put('/usuario', (req, res) => {
    res.send('Responda a requerimiento por put en la ruta /usuario')
})
```

```
app.delete('/usuario/:id', (req, res) => {
    res.send('Responda a requerimiento por delete en la ruta /usuario/:id')
})
```

Métodos de respuesta

Método	Descripción
res.download()	Solicita un archivo para descargarlo.
res.end()	Finaliza el proceso de respuesta.
res.json()	Envía una respuesta JSON.
res.jsonp()	Envía una respuesta JSON con soporte JSONP.
res.redirect()	Redirecciona una solicitud.
res.render()	Representa una plantilla de vista.
res.send()	Envía una respuesta de varios tipos.
res.sendFile	Envía un archivo como una secuencia de octetos.
res.sendStatus()	Establece el código de estado de la respuesta y envía su representación de serie como el cuerpo de respuesta.

Solicitud con parámetros

Los parámetros de ruta son llamados segmentos URL que son usados para capturar los valores especificados en su posición en la URL. Los valores capturados se rellenan en el objeto *req.params*, con el nombre del parámetro de ruta especificado en el *path* así como sus respectivas claves.

- Con un solo parámetro
  Localhost:puerto/usuario/3

```
// devolviendo parámetros que paso por url
app.get('/usuario/:id', (req,res)=>{
    res.send(req.params.id)
})
```

- Con más de un parámetro
  Localhost:puerto/usuario/2022/05/12

```
//Capturar más de un parámetro
app.get("/usuario/:year/:month/:day", (req, res) => {
  res.send(req.params);
});
```

Recepción de formularios

Para poder recibir información desde formularios en Express, es necesario incluir el middleware:

```
app.use(express.urlencoded());
```

El método POST es utilizado para recibir la información que el usuario ingresó en un formulario. En el caso del ejemplo que sigue, el formulario es:

```
<!DOCTYPE html>
</html>
    <head>
        <title>Formulario</title>
    </head>
    <body>
        <h1>Formulario</h1>
        <form method="POST" action = "/form">
            <input type = "text" name = "nombre" placeholder = "Nombre"><br/>
            <input type = "text" name = "apellido" placeholder = "Apellido"><br/>
            <input type = "text" name = "mensaje" placeholder = "Mensaje"><br/>
            <input type = "submit" value="Enviar">
        </form>
    </body>
 </html>
```

Se recibe la información del formulario por método POST, es decir, que viaja en el cuerpo del mensaje (body) el nombre y contenido de cada campo del formulario.

En action se indica la ruta a dónde enviar la información del formulario una vez que el usuario oprime el botón. En este caso, se trata de una ruta del mismo servidor. La ruta es “/form”.

Ya en el servidor, la información del formulario se recibe en la variable req y como fue mediante método POST, se encuentra en el cuerpo (body) de la variable req.

En la variable res colocamos el mensaje que el servidor retorna al cliente.

En este ejemplo, mediante el método send, se le envía al cliente (el browser), una cadena (string).

```
app.post('/form', (req, res) => {
  res.send("Hola " + req.body.nombre + " " + req.body.apellido + " tu mensaje es: "
+
  req.body.mensaje);
});
```

Requerimientos

* req.query / devuelve un string literal que se pasó por la URL de la consulta
* req.params / devuelve los parámetros que se pasaron por la URL de la consulta
* req.body / devuelve los datos que viajaron por el cuerpo de la consulta

Ejemplo

En el siguiente ejemplo se puede ver una sencilla aplicación Express que escucha en el puerto 3000 y recibe peticiones HTML en la ruta ‘/hola’. En dicha ruta recibe 3 tipos de peticiones diferentes: get, post y put.

```
var express = require('express');
var app = express();


app.get('/hola',  (req, res) => {
```

```
  res.send('Hola mundo en GET');
});

app.post('/hola', (req, res) =>  {
  res.send('Hola mundo en POST');
});

app.put('/hola', (req, res) => {
  res.send('Hola mundo en PUT');
});

app.listen(3000, () => {
  console.log(`Servidor corriendo en puerto 3000...`);
});
```

Trabajo Práctico

Crear un formulario de registración con los siguientes datos: **nombre, apellido, edad, número de celular, país de nacimiento, país de residencia.**

Recibir los datos en el servidor y armar otra página de respuesta que incluya los datos del usuario y un enlace a la página de registración nuevamente.

Bibliografía utilizada y sugerida

- Express (n. d.) Recuperado de: https://expressjs.com/es/
- MDN - JavaScript (n.d.) Recuperado de: https://developer.mozilla.org/es/docs/Web/JavaScript
- NodeJs (n. d.) Recuperado de: https://nodejs.org/es/
- NodeJS Documentacion (n. d.) Recuperado de: https://nodejs.org/es/docs/