Resumen

En este ejemplo veremos el funcionamiento de las peticiones HTTP y usaremos una aplicación llamada Postman para enviar estas peticiones.

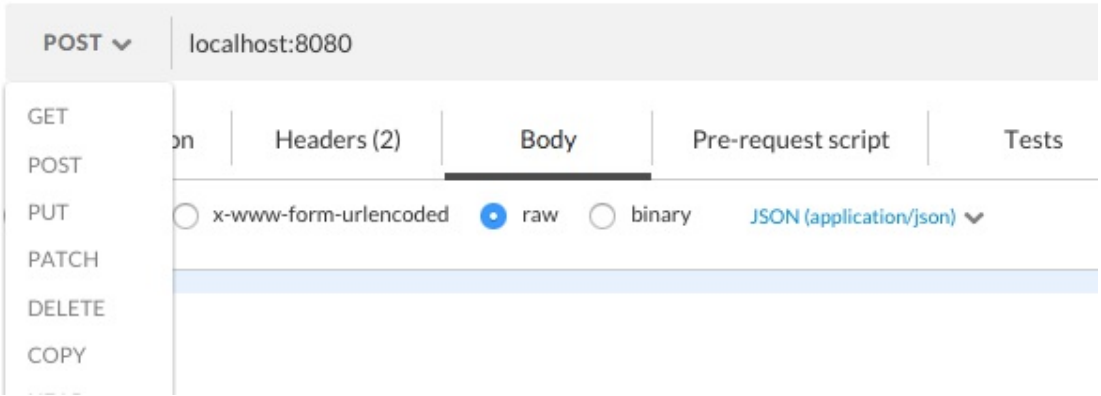
Postman

Postman es una aplicación para crear y enviar peticiones HTTP. En este ejemplo la usaremos para entender como node.js trata las peticiones HTTP y la seguiremos usando durante todo el curso.

Descarga

Para descargar Postman simplemente debemos tener instalado Google Chrome y descargar la [aplicación (https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop)](https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop).

Uso

El 

funcionamento es muy simple, en el menú desplegable seleccionamos el tipo de peticion HTTP y en la barra horizontal escribimos la peticion HTTP.

En campo que hay más abajo, tenemos un menú con 5 elementos (Authorization, Headers, Body, Pre-request script, Tests). Para este ejemplo no los necesitaremos. Pero normalmente solo usaremos el campo BODY (en el que escribiremos el cuerpo de nuestra petición), el formato 'raw' i 'JSON(application/json)'. Más adelante lo veremos.

Creando el servidor con Node y MongoDB

En este ejemplo usaremos lo que aprendimos sobre MongoDB e introduciremos las peticiones HTTP. El objetivo es que mediante estas peticiones HTTP podamos utilizar las operaciones CRUD en nuestra base de datos.

Parsear URL

Primero de todo vamos a parsear la url que tenemos con la finalidad de obtener los datos de ella. Para eso utilizaremos el modulo `url`.

```
var url = require('url');
```

Con la funcion `parse` del módulo podemos obtener todos los datos de nuestra url.

```
http.createServer(function(req, res) {
    var parsedUrl = url.parse(req.url, true);
}
```

La estructura que sigue una url es la siguiente.

```
[user:password@]host[:port]][/]path[?query][#fragment]
```

El módulo url nos permite seleccionar la parte de la url que queramos. En este ejemplo simplemente usaremos las propiedades `pathname` y `query` pero existen más, los podéis encontrar [aquí (https://nodejs.org/api/url.html#url_url_format_urlobj)](https://nodejs.org/api/url.html#url_url_format_urlobj).

```
parsedUrl.pathname   // si queremos el path
parsedUrl.query      // si queremos la consulta
```

Peticiones HTTP

> Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

Una vez tenemos el url parseado y podemos adquirir los datos que nos da el usuario, es hora de enviarle peticiones al servidor para que pueda realizar las operaciones CRUD en la base de datos.

Las peticiones pueden ser de varios tipos, los que utilizaremos en este curso son: `GET`, `POST`, `PATCH` y `DELETE`.

- GET: Para hacer consultas. Por ejemplo si queremos los estudiantes con nombre

'Joaquin'

```
method 'GET':
localhost:8080/student?nombre=Joaquin
```

- POST: Para insertar. Por ejemplo si quermos añadir un estudiante:

```
method 'POST':
localhost:8080/student?nombre=Joaquin&carrera=informatica&universidad=upc
```

- PATCH: Para hacer actualizaciones. (Similar a POST)
- DELETE: Para eliminar. (Similar a GET)

Con Postman podemos elegir el tipo de peticion desde el menú desplegable.

Conectar e interactuar con la base de datos

Para establecer conexión con la BD simplemente haremos lo siguiente:

```
MongoClient.connect("mongodb://localhost:27017/students", function(err,
database) {
    if (err) log.error("Error connecting to database with uri: " + DB_URI);
    else {
        log.info("Successfully connected to database");
        db = database;
    }
});
```

Ahora ya podemos realizar las operaciones que vimos de MongoDB en nuestro Javascript.

Por ejemplo, podemos hacer una busqueda pasandole por parametro la query que hemos obtenido de la URL. En este caso ejecutariamos esta operación cuando recibieramos una petición GET. La DB nos devuelve la consulta en formato JSON, para pasarlo a String utilizamos JSON.stringify(result)

```
collection.find(parsedUrl.query).toArray(function (err, result) {
    if (err) {
        res.end("Error retrieveing student/s: " + err);
        log.error("Error retrieveing student/s: " + err);
    }
    else res.end(JSON.stringify(result));
});
```

Si queremos insertar un elemento lo haríamos de la siguiente manera:

```
collection.insert(parsedUrl.query, function(err, result) {
    if (err) {
        res.end("Couldn't add that student:" + err);
        log.error("Couldn't add that student:" + err);
    }
    else res.end("true");
});
```

En el ejemplo podréis ver de manera más ordenada este procedimiento.