Redes de computadoras

Desarrollo Web

ExpressJS

Express.js
Un web framework para node.js

Infraestructura web rápida, minimalista y flexible para Node.js

Express proporciona una delgada capa de características de aplicación web básicas, sin ocultar las características de nativas de Node.js.

¿que necesita la implementación de un servidor web?

- **Procesar HTTP:** Aceptar conexiones TCP, procesar solicitudes HTTP y enviar respuestas. (módulo http nativo de node)

- **Enrutamiento:** Proveer un mapeo de las URLs hacia funciones programadas en el servidor. Tabla de enrutamiento.

- **Soporte para intermediarios (Middleware):** Brindar un mecanismo para poder manejar la session, cookies, políticas de seguridad, compresión, etc...

Express.js

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

var app = express();

// El objeto app tiene métodos para
// - Enrutar HTTP request
// - Renderizar HTML (ej: mediante un preprocessor como pug)
// - Configurar middleware

app.get('/', function (httpRequest, httpResponse) {
    httpResponse.send('Hello world');
});

app.listen(3000);
```

Enrutamiento (routing)

Por métodos:

```
app.get(urlPath, requestProcessFunction);
app.post(urlPath, requestProcessFunction);
app.put(urlPath, requestProcessFunction);
app.delete(urlPath, requestProcessFunction);
app.all(urlPath, requestProcessFunction);
```

urlPath puede contener parámetros:

```
var urlPath = "/generos/:id";
```

httpRequest object

```
app.get('/generos/:id', function (httpRequest, httpResponse) {
```

request.params

Objeto que contiene los parámetros de la ruta, **:id**

request.query

Objeto que contiene los parámetros de la querystring **?id=20**

request.body

Objeto que contiene el cuerpo del mensaje

request.get(field**)**

Retorna el valor de una cabecera específica.

httpResponse object

```
app.get('/generos/:id', function (httpRequest, httpResponse) {
```

Response.write(content)

Crea el cuerpo de la respuesta con el contenido (content)

Response.status(code)

Setea el código de estatus de la respuesta.

Response.set(prop, value)

Setea una propiedad en la cabecera de la respuesta.

Response.end()

Finaliza la respuesta con un contenido opcional.

Response.send(mensaje)

Envía una respuesta.

Los métodos retornan el objeto response por lo que se pueden apilar:

```
httpResponse.status(200).write(contenido1).write(contenido2).end()
```

Intermediarios (Middleware)

Permite que un programa intermediario participe en la petición.

```
app.use(function (request, response, next) {
```

A su vez es posible interceptar todas las peticiones que coincidan con una ruta:

```
app.all(urlPath, function (request, response, next) {
```

Usos:

- Chequear si un usuario está logueado
- Parsear el cuerpo de la petición como JSON
- Manejo de sesión, cookies, encriptación, compresión, etc...

Enlaces:

Express.js

https://expressjs.com/

Taller (expressworks nodeschool):

https://github.com/azat-co/expressworks