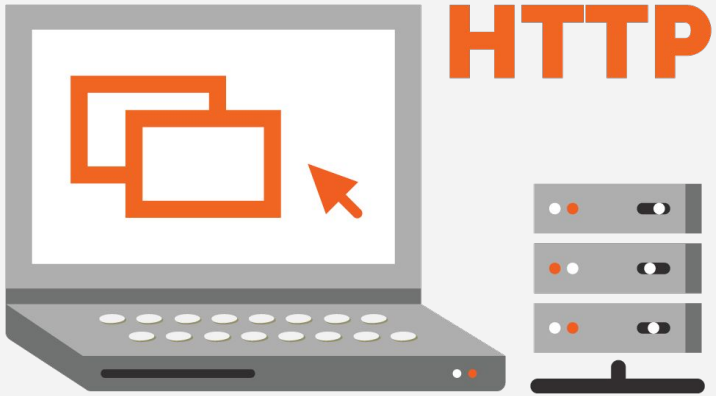
Node.js

Cordero Hernández,
Marco R.

En sesiones pasadas

- HTTP
  - Conceptos
  - Solicitudes
  - Respuestas
  - AJAX / XHR



CONTENIDOS

01	02	03
Node.js	NPM	Servicios WEB / REST API
	04	05
	Express	Middlewares

ADVERTENCIA

Antes de conocer acerca de Node.js, debe tenerse en cuenta que lo relacionado a esta herramienta introduce posibles riesgos (filtración de usuarios, contraseñas) y complicaciones innecesarias si no se hace buen uso de la misma.

Entre lo más común, es cuando se trabaja con repositorios remotos (usualmente con Git y GitHub) y no se ignora el directorio `node_modules`, lo cual resultará en repositorios de gran tamaño. Esto puede resolverse usando un archivo `.gitignore` (a nivel de repositorio) y agregando la siguiente regla:

```
node_modules/
```

01

Node.JS

Node.js

Según la referencia oficial, Node.js es “un entorno de ejecución de JavaScript que permite a los desarrolladores crear servidores, aplicaciones web, herramientas de línea de comando y scripts”

… en otras palabras, permite ejecutar **JavaScript** del lado del servidor.

Node.js

Es código abierto, multiplataforma y también orientado a eventos asíncronos (no crea nuevos hilos por cada petición).

Tiene muchas librerías que se pueden descargar con ***npm*** que facilitan el desarrollo por su gran variedad de soluciones.

Ejercicio opcionalmente *obligatorio*

DESCARGAR NODE.JS

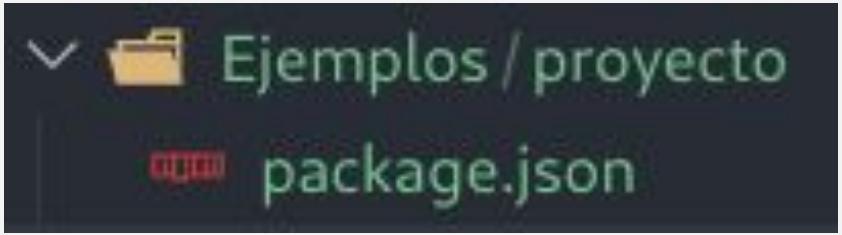
Si aún no lo tienes…

Inicialización de proyectos

`npm init` para crear las bases del proyecto

- **`-y`** para aceptar las configuraciones por default

Después de esto, un archivo llamado **package.json** se encontrará en el directorio donde se ejecutó el comando inicial



Frameworks y herramientas comunes

- **Express**: Para crear y administrar servidores web
- **Socket.io**: Comunicación en tiempo real (chat en línea, juegos, notificaciones, etc.)
- **Multer**: Middleware enfocado principalmente a la carga de archivos desde el front-end hacía el back-end
- **Mongoose**: Utilería para modelado y almacenamiento de datos en MongoDB

Ejercicio

1. Crea un nuevo proyecto de node (**npm init**)
2. En la misma ruta, crea un archivo **server.js**
3. Copia el siguiente código dentro

```
const http = require('http'), hostname = '127.0.0.1'; // localhost
const port = 3000;

const server = http.createServer((req, res) => {
   res.statusCode = 200;
   res.setHeader('Content-Type', 'text-plain'); res.end('Hola mundo\n');
});

server.listen(port, hostname, () => {
   console.log(`Servidor corriendo en http://${hostname}:${port}`);
});
```

4. Ejecútalo con **node server.js**

02

NPM

Instalación de paquetes

Node **P**ackage **M**anager permite instalar *paquetes* desde su repositorio oficial, en donde una librería nace cada nanosegundo.

Para hacerlo, se usa:
`npm i[nstall] <nombre del paquete> <n2> <n…>`

- `-g` para guardar de manera global
- `-D | –save-dev` para guardar como dependencia de desarrollo

Mini ejercicio
Instala la librería `chalk@4.1.2` de forma local. Se usa con:

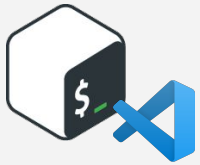
- `const chalk = require('chalk');`
- `console.log(chalk.blue.bold('Hola mundo'));`

Listado y eliminación de paquetes

- **npm** `list`
  - Lista los paquetes instalados en el proyecto actual
  - `-g` para ver los paquetes globales

- **npm** `uninstall <nombre del paquete> <n2> <n…>`
  - Desinstala uno o varios paquetes
  - `-D` para paquetes de desarrollo
  - `-g` para paquetes globales

Ejercicio *obligatoriamente opcional*



Instalar **nodemon** como dependencia de [-D]esarrollo

- Este paquete sirve para actualizar el servidor cada que se guarda algún archivo del proyecto

Modificar la sección de de `scripts` dentro de **package.json**

- Crea un nuevo script para ejecutar el proyecto con node
- Crea otro para ejecutarlo con **nodemon**

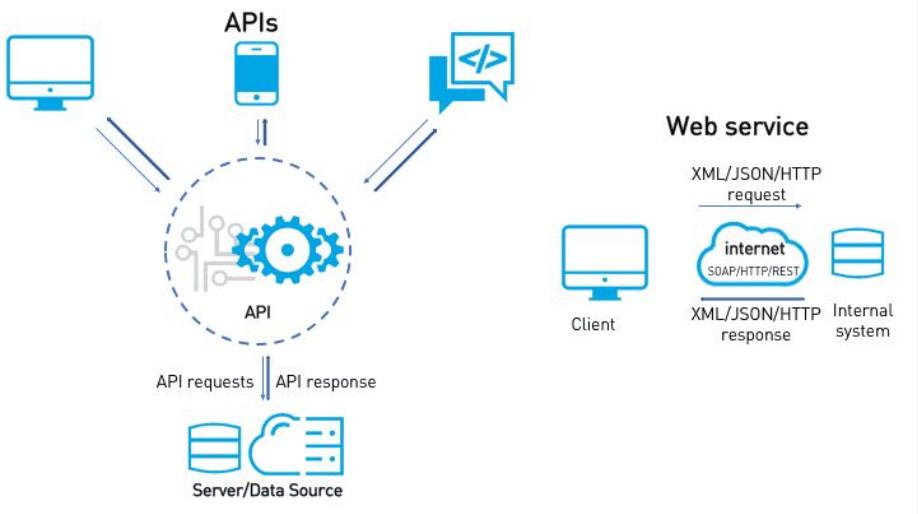
Ejecuta el proyecto con **nodemon** y verifica que los cambios se muestran automáticamente

03

SERVICIOS WEB / REST API

Servicios WEB

Definidos como *programas diseñados para intercambiar información entre al menos 2 máquinas sobre una red*, permiten a un cliente la posibilidad de solicitar información a un servidor.



REST API

- **API** (**A**pplication **P**rogramming **I**nterface): Características y reglas que definen cómo una aplicación puede interactuar con otra aplicación.

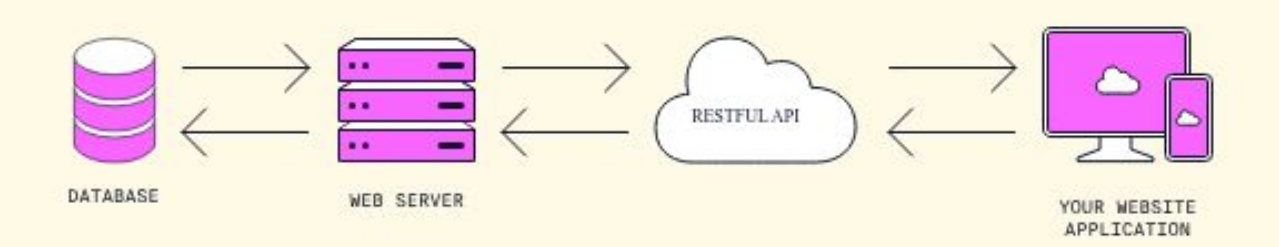
- **REST** (**RE**presentational **S**tate **T**ransfer): Arquitectura de software para sistemas distribuidos (como la web). Se enfoca en **HTTP** en conjunto de XML o **JSON**. Las operaciones se solicitan mediante los métodos de HTTP como *GET, POST, PATCH, PUT, DELETE,* etc.

Servicios WEB RESTful

- Basados en ***recursos*** usualmente contenidos dentro de entidades/objetos (usuarios, alumnos, productos, etc.).

- Los recursos son identificados por **URIs**

- La manera en que los recursos son descritos (atributos y formato) se conoce como **representación** (XML o JSON)



Servicios WEB RESTful

Otras características de estos incluyen:

- **Interfaz uniforme**
  - Verbos de HTTP - GET, POST, PUT, ...
  - URIs - Nombre de los recursos (Endpoints)
  - Respuestas de HTTP - Status, Body
- "**Cacheable**" - Información que puede ser almacenable.
- **Sin estado** - El servidor no guarda el estado del cliente. Cada solicitud no depende de las anteriores; cada mensaje debe dar la suficiente información sobre el contexto para ser procesada acorde.

Servicios WEB RESTful

- **Cliente-Servidor**: Asume sistema desconectado en todo momento hasta que una petición realizada por un cliente se hace a un servidor (este no necesariamente está atendiendo solicitudes todo el tiempo)

- **Sistema de capas**: Asume que podría ser una conexión no directa con el servidor (hay intermediarios); Esto evita “tumbar” la máquina que lo hospeda.

Ejemplo de endpoints y métodos

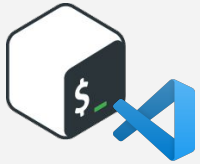
Recurso	Endpoint	Métodos	Descripción
alumno	/alumno	GET, POST	Lista de alumnos, Nuevo alumno
	/alumno/{id}	GET, PUT, PATCH, DELETE	Obtener alumno, Actualizar todo el alumno, Actualizar al alumno, Borrar alumno
	/alumno/{id}/updateImage	PUT	Actualizar url de imagen

Ejemplo más en avanzado https://petstore.swagger.io/

04

EXPRESS

Conceptos iniciales



Express es un *framework* que permite la creación rápida y fácil de un servidor web (usualmente preferido sobre *http*).

Instalación: `npm i express`

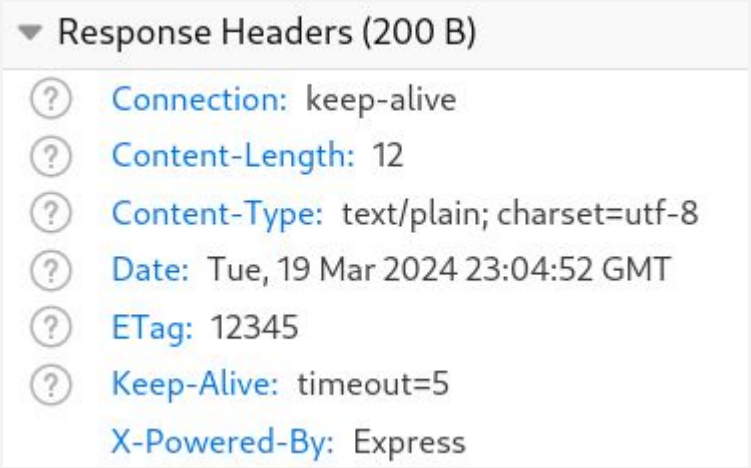
```
// Librerías externas
const express = require('express');

// Configuración de Express
const app = express();
const port = 3000;

// Rutas del servidor
app.get('/', (req, res) => res.send('Hola mundo!'));
app.route('/home').get((req, res) => res.send('Hola casita!'));

// Ejecución del servidor
app.listen(port, () => console.log(`Servidor corriendo en el puerto ${port}`));
```

Envío de headers

Desde `res` se tiene:

- `header`(<tipo de header>, <valor del header>)
- `set`({“header1”: “valor”, “header2”: “valor”})
- `set`(“header”, “valor”)

```
res.set('Content-Type', 'text-plain');
res.set({
   'Content-Type': 'text/plain',
   'Content-Length': '123',
   'ETag': '12345'
});
```



Lectura de headers

Desde `req` se tiene:

- `header`(‘nombre del header’)
- `get`(‘nombre del header’)

```
console.log(req.header('User-Agent'));
```

```
Mozilla/5.0 (X11; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0
```

05

MIDDLEWARES

Concepto

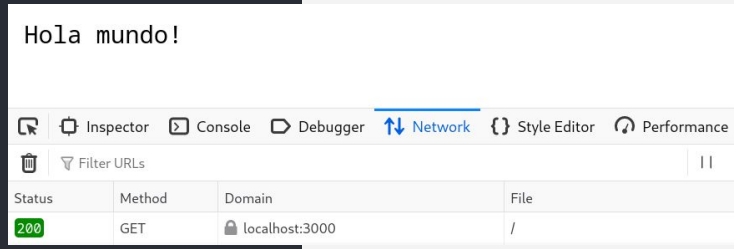
En **Express** es posible definir funciones intermedias entre el origen de la petición y el destino, las cuales se encargarían de realizar algún ***preprocesamiento*** o ***validación*** antes de seguir con la petición; a esto se le conoce como ***middlewares***.

Se encargan de:

- Ejecutar cualquier código (ej. Guardar un *log* con todas las peticiones)
- Realizar cambios a la solicitud y los objetos de la respuesta (body-parser)
- Finalizar el ciclo de solicitud/respuestas (ej. Si un usuario no está autenticado, redirigir a login)
- Invocar la siguiente función de middleware en la pila (es posible tener más de un middleware a la vez)

Uso

```
// Desde log.js
function log(req, res, next) {
   console.log(`Método: ${req.method}`);
   console.log(`URL: ${req.originalUrl}`);
   console.log(`Fecha: ${new Date(Date.now()).toString()}`);

   // Extraer header
   console.log(`Accept: ${req.get('Accept')}`);

   // Siguiente paso de la cadena
   next();
}

// Desde index.js
app.use(log); // Middleware para todas las rutas
app.use('/alumnos', log); // Middleware exclusivo de alumnos

// Ruta con middleware
app.get('/alumnos', log, (req, res) => {
   res.send('Mostrando alumnos...');
});
```




```
Método: GET
URL: /
Fecha: Tue Mar 19 2024 17:27:39 GMT-0600 (Central Standard Time)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
```