# Bienvenidos

Universidad
Provincial del Sudoeste
*Promoviendo el Desarrollo Armónico de la Región*

**Clase 09.**
Aplicaciones Móviles y Cloud Computing

# API RestFull

Universidad Provincial del Sudoeste
Promoviendo el Desarrollo Armónico de la Región

# Temario

| 08 | 09 | 10 |
| --- | --- | --- |
| **Desarrollo de aplicaciones Restfull** | **Desarrollo de aplicaciones Restfull** | **Introducción al lenguaje Java** |
| ✓ Arquitectura REST<br>✓ CRUD con NodeJS y Express<br>✓ Middlewares / Routers | ✓ Websockets<br>✓ Motores de Plantillas<br>✓ Dependencias socket.io / express-handlebars | ✓ Breve historia de Java<br>✓ Características de JAVA<br>✓ Conceptos básicos y primeros pasos |

Universidad Provincial del Sudoeste
Promoviendo el Desarrollo Armónico de la Región

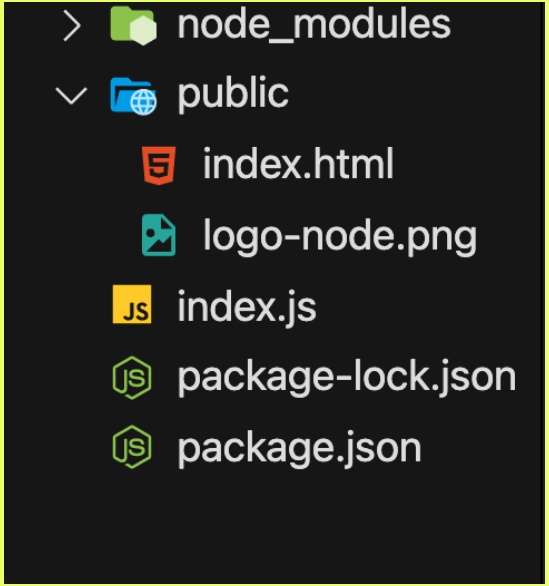
# Objetivos de la clase

- Comprender el funcionamiento de los motores de plantillas. Ejemplo de implementación con Express-Handlebars
- Websockets: descripción y uso de este protocolo
- Implementación de websockets en Node, mediante la librería Socket.io

# Archivos estáticos

**¿Cómo funciona?**

- ✓ Nuestro servidor tiene la posibilidad de alojar recursos que pueden ser visibles para el cliente de manera directa.
- ✓ Podemos configurar una carpeta para que el usuario pueda acceder y ver dichos recursos de manera directa sólo con acceder a la ruta donde se encuentra ubicada.
- ✓ En este curso y en proyectos profesionales podrás encontrar estos archivos en la carpeta “public”, haciendo referencia como dice el nombre, a recursos públicos de fácil acceso para el cliente.

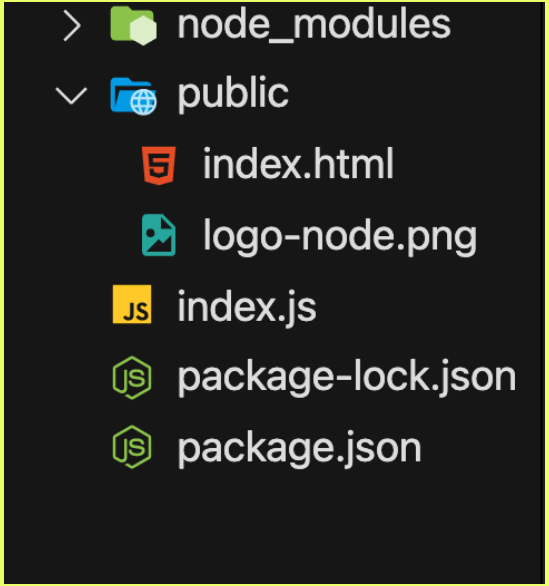


# Archivos estáticos

**¿Cuándo utilizarlos?**

Los dos casos principales para los cuales encontrarás un uso para esta carpeta “public” que guarda archivos estáticos son:

- ✓ Cuando necesitemos alojar imágenes y servirlas directamente al cliente.
- ✓ Cuando necesitemos alojar una página web en todos sus sentidos: html, css, js. En esta clase haremos una página sencilla para mostrar el alcance de public.



# ¿Cómo convertir una carpeta en un recurso estático?

Para poder utilizar los recursos de una carpeta de manera estática, basta con que en el servidor especifiquemos como “express.static” dicha carpeta con la siguiente sintaxis:

```
app.use(express.static('public'))
```

Indicamos que, todo lo que viva en la carpeta public, podrá ser accedido directamente desde la carpeta *public*.

# ¿Qué es un middleware?

Un middleware es simplemente una función.

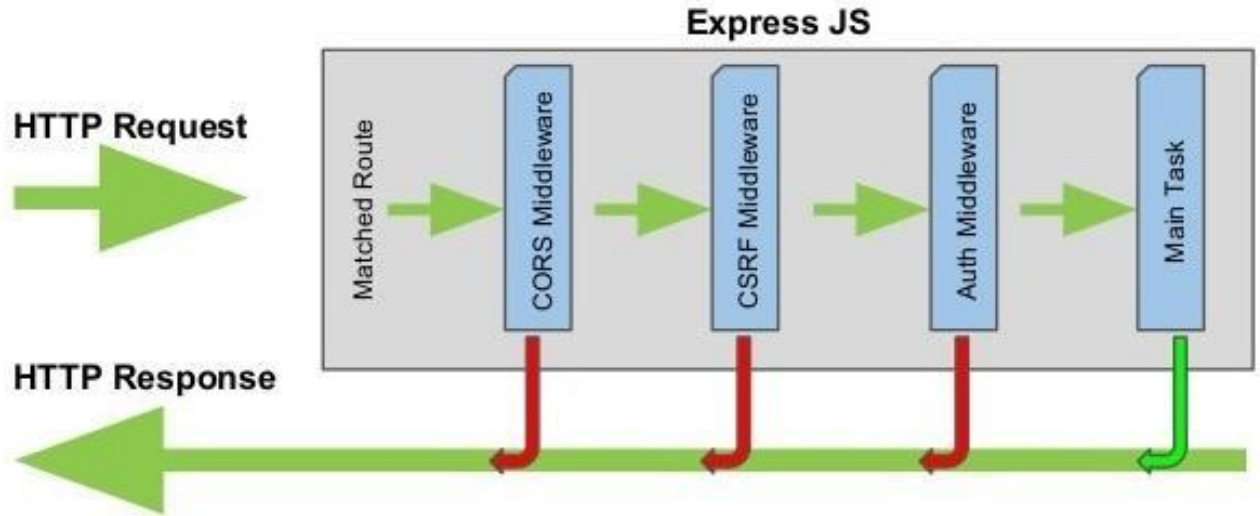
Cada vez que utilizamos **app.use** estamos utilizando un **middleware**. Son funciones que se ejecutan de manera intermedia entre la petición del cliente, y el servicio de nuestro servidor.

Como lo indica el nombre, “middleware” hace referencia a un intermediario. **Siempre se ejecuta antes de llegar al endpoint que corresponde**.

Podemos utilizar un middleware para:

- ✓ Dar información sobre las consultas que se están haciendo (logs)
- ✓ Autorizar o rechazar usuarios antes de que lleguen al endpoint (seguridad)
- ✓ Agregar o alterar información al objeto **req** antes de que llegue al endpoint (formato)
- ✓ Redireccionar según sea necesario (router)
- ✓ En ciertos casos, finalizar la petición sin que llegue al endpoint (seguridad)

# Flujo de múltiples middlewares a través de una petición




Este “main Task” sería el endpoint con tu servicio”.

Como lo ves en el diagrama anterior, los middlewares se ejecutan **EN ORDEN**, eso quiere decir que, si algún middleware depende de que se haya realizado otra operación ejecutada por un middleware previo, debemos colocarlos en cascada según prioridad.

# Middleware de nivel de aplicación

Este ejemplo muestra una función de middleware implementada vía **app.use**. La función se ejecuta cada vez que la aplicación recibe una solicitud.
Se denomina middleware “a nivel de aplicación”, dado que aplica a toda la app.

```
const app = express();

app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});
```

Podemos distinguir middlewares a nivel app, a nivel endpoint, y a nivel router. Existen también los middlewares de error. Por último, podemos distinguir middlewares nativos, y middlewares de terceros.

# ¿Qué es multer?

Multer es un **middleware de terceros**, pensado para poder realizar carga de archivos al servidor.

En ocasiones el cliente necesitará subir una imagen, un vídeo o un archivo, según sea nuestra aplicación. Ello nos lleva a configurar nuestro servidor para soportar estos archivos y poder almacenarlos en donde nosotros le indiquemos.

Al ser de terceros, necesitaremos instalarlo para poder utilizarlo.

# Motores de Plantillas

Un motor de plantillas es un recurso que nos permite dotar de dinamismo a las páginas web que genere nuestro backend. La idea básica es disponer de determinada sintáxis que permita interpolar en medio de nuestro código html, diferentes variables. Al momento de la renderización (representación en pantalla), las variables serán reemplazadas por el valor que están guardan.

Existen múltiples motores de plantillas tales como: Handlebars, ejs, Pug. Cada uno de estos tiene una sintaxis diferente, diferentes reglas de reemplazo de plantillas, aunque son muy similares en funcionamiento. En handlebars, las variables pueden interpolarse utilizando la sintáxis {{variable}}

```
9   <body>
10      <h1>¡Hola, {{nombre}}!</h1>
11      <p>Estuvimos procesando tu carrito de compra y el monto final será de {{precio}}</p>
12      <br>
13      <p>El pedido será entregado a {{dirección}}</p>
14      <a href="http://linkfalso.com">Cambiar dirección</a>
15      <p>Además el precio de envío será {{costo_envio}} ya que su membresia es {{tipo_membresia}}</p>
16  </body>
```

handlebars

<%= EJS %>

Effective JavaScript templating.


Universidad Provincial del Sudoeste

Promoviendo el Desarrollo Armónico de la Región

# ¿Cuándo utilizar motores de plantillas? nivel de dinamismo

**Nivel de dinamismo**

- ✓ El nivel de dinamismo refiere a qué tanta interacción tiene el usuario con la página, además de qué tan constantes son los cambios y renderizados de los elementos de la página.
- ✓ Tener definido el nivel de dinamismo que tendrá nuestro proyecto, nos permitirá focalizar la tecnología a utilizar.
- ✓ Que una herramienta sea poderosa, no significa que sea la solución para todo.


# ¿Como utilizar Express Handlebars?

Configuración básica de la librería:
https://www.npmjs.com/package/express-handlebars#basic-usage

Básicamente se define una **plantilla main**, conteniendo una estructura html completa, pero sin datos, más una instrucción especial ***{{{body}}}***. Y luego diferentes **vistas**, con html puntual, relativo a cada una. La **vista** se entrelazará con el **main** para conformar el resultado final.

Sintaxis para interpolación de variables dentro de una vista:
***{{variable}}***

Estructuras: es posible utilizar condicionales y bucles:

- {{#if variable}} ... {{/if}}
- {{#each variableArray{{ ... {{this}} o {{this.prop}} ... {{/each}}


Universidad Provincial del Sudoeste
Promoviendo el Desarrollo Armónico de la Región

# WebSockets

Websocket es un protocolo de comunicación basado en TCP para poder establecer esa conexión entre el cliente y el servidor. Esto, como sabemos, es el mismo objetivo que cubre HTTP.

La diferencia sustancial es que websockets permitirá establecer una **comunicación bidireccional** entre el cliente y el servidor. La comunicación bidireccional implica:

- ✓ Que el cliente puede obtener recursos del servidor cuando lo solicite (como en HTTP)
- ✓ Que el servidor pueda entregar información al cliente sin necesidad de que el cliente haga una petición.

Es de suma utilidad en aplicaciones como chats, portales de noticias (que impliquen actualizaciones en tiempo real), juegos en linea, etc.

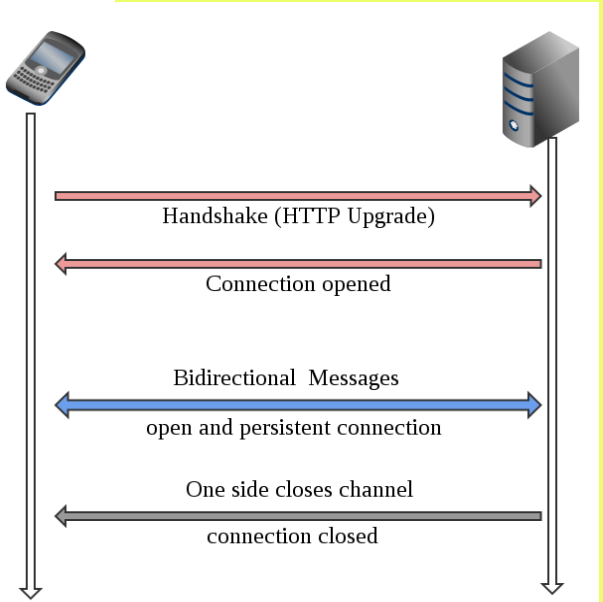
# Funcionamiento de un Websocket

Primero, el cliente tiene que enviar una solicitud HTTP llamada **Handshake** (apretón de manos). Este apretón de manos será un "acuerdo" o "contrato" de confianza para que el servidor pueda actualizar al cliente sin que éste se lo pida.

El servidor recibe la petición de Handshake y procede a "responderle el saludo". A esto se le llama "**Abrir conexión**".

A partir de este punto, el canal queda abierto **de manera bidireccional**, por lo que el cliente se puede comunicar con el servidor cuando quiera y viceversa.

La comunicación es "persistente" hasta que alguno de los dos lados decida cerrar el canal de comunicación.



# Socket.io

- ✓ Es una biblioteca de Javascript para poder implementar los sockets anteriormente mencionados.
- ✓ Debido al funcionamiento que hemos mencionado antes, **socket.io debe instanciarse tanto del lado del cliente, como del servidor**.
- ✓ Permite utilizar todo el potencial mencionado de los websockets, y cuenta con una API casi idéntica para cliente y para servidor.

# ¿Preguntas?

Universidad
Provincial del Sudoeste
*Promoviendo el Desarrollo Armónico de la Región*

# Muchas gracias.

Universidad
Provincial del Sudoeste
*Promoviendo el Desarrollo Armónico de la Región*