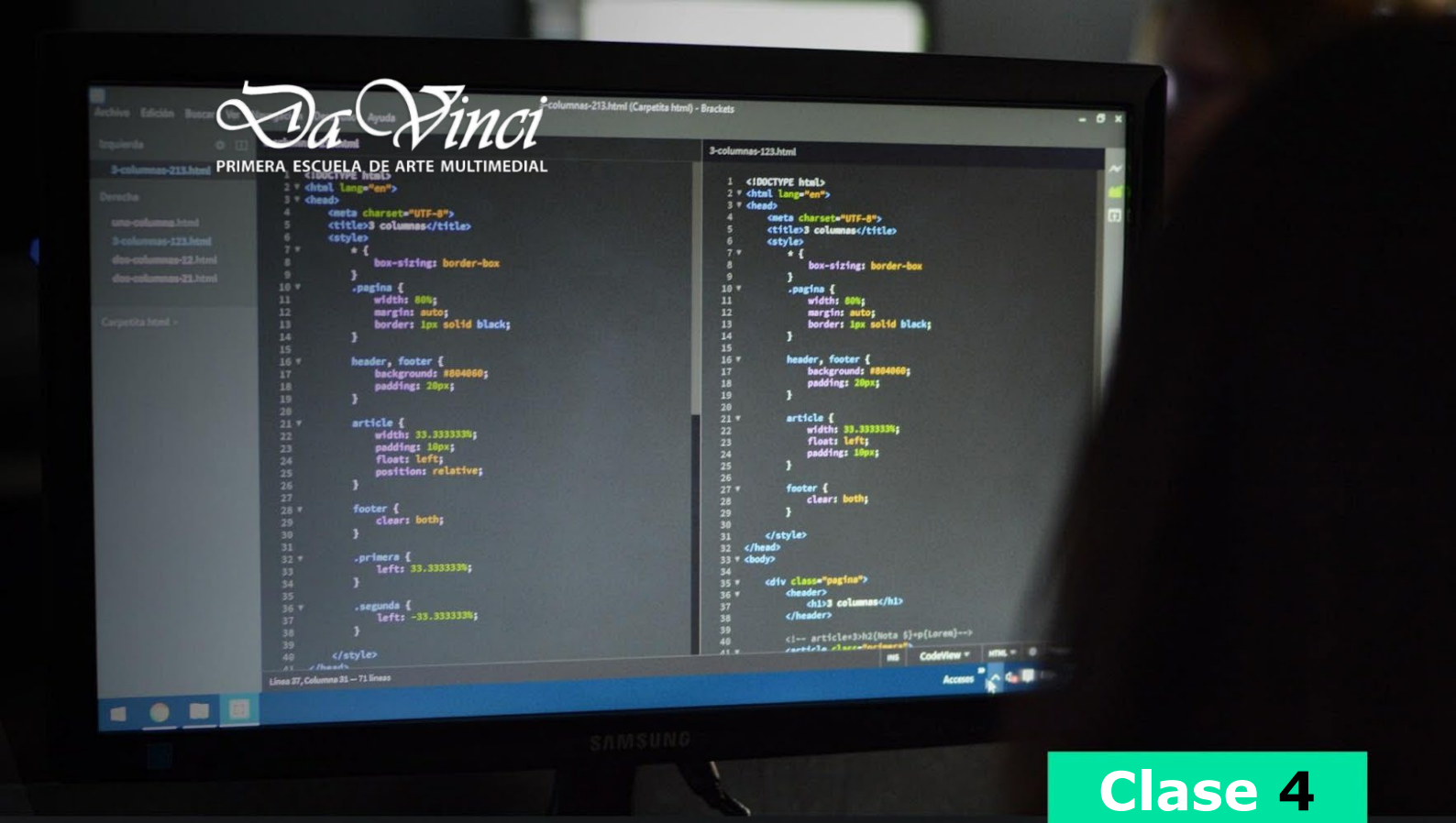

Clase 4

Diseño y Programación Web

Materia:
Aplicaciones Híbridas

Docente contenidista: MARCOS GALBÁN, Camila Belén

Express.js

Peticiones HTTP

HTTP (HyperText Transfer Protocol) es el protocolo que se utiliza para la comunicación entre clientes (como navegadores web) y servidores en la web. Las peticiones HTTP permiten a los clientes solicitar recursos del servidor y enviar datos a él. Cada petición HTTP tiene una estructura específica que incluye varios componentes:

Línea de solicitud:

- Método HTTP (GET, POST, PUT, DELETE, etc.)
- Ruta (URL) del recurso solicitado
- Versión del protocolo HTTP

Encabezados de la petición:

- Proveen información adicional sobre la solicitud, como el tipo de contenido, el agente de usuario, las cookies, etc.

Cuerpo de la petición (opcional):

- Contiene datos enviados al servidor (usualmente en peticiones POST y PUT).

Métodos HTTP Comunes

GET:

Solicita datos de un recurso específico.

No tiene cuerpo de la solicitud.

Ejemplo: Obtener una lista de usuarios.

```
GET /usuarios HTTP/1.1
Host: example.com
```

POST:

Envía datos al servidor para crear un nuevo recurso.

Tiene un cuerpo de la solicitud.

Ejemplo: Crear un nuevo usuario.

```
POST /usuarios HTTP/1.1
Host: example.com
Content-Type: application/json

{
"nombre": "Juan",
"edad": 30
}
```

PUT:

Actualiza un recurso existente con datos nuevos.

Tiene un cuerpo de la solicitud.

Ejemplo: Actualizar la información de un usuario.

```
PUT /usuarios/1 HTTP/1.1
Host: example.com
Content-Type: application/json

{
"nombre": "Juan",
"edad": 31
```

```
}
```

DELETE:

Elimina un recurso específico.

No tiene cuerpo de la solicitud.

Ejemplo: Eliminar un usuario.

```
DELETE /usuarios/1 HTTP/1.1
Host: example.com
```

Respuesta HTTP

La respuesta HTTP enviada por el servidor también tiene una estructura específica, que incluye:

Línea de estado:

- Protocolo HTTP y versión.
- Código de estado HTTP (200 OK, 404 Not Found, etc.)
- Mensaje de estado.

Encabezados de respuesta:

- Proveen información adicional sobre la respuesta, como el tipo de contenido, la longitud del contenido, etc.

Cuerpo de la respuesta (opcional):

- Contiene los datos solicitados o el resultado de una operación.

Ejemplo de Respuesta:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 60

{
"nombre": "Juan",
"apellido": "Perez",
"edad": 30
}
```

¿Qué es Express?

Express es un framework web para Node.js, diseñado para simplificar la creación y el manejo de aplicaciones web y APIs. Es ligero, rápido y flexible, proporcionando un robusto conjunto de características para desarrollar aplicaciones web y móviles.

Características Principales de Express

Manejo de Rutas:

Express facilita la definición y el manejo de rutas para las peticiones HTTP. Permite definir rutas con diferentes métodos HTTP como GET, POST, PUT, DELETE, etc.

GET:

```
app.get('/get-endpoint', (req, res) => {
     res.send(' GET request');
   });
```

POST:

```
app.post('/post-endpoint', (req, res) => {
     res.send('POST request');
   });
```

PUT:

```
app.put('/put-endpoint', (req, res) => {
     res.send('PUT request');
   });
```

DELETE:

```
app.delete('/delete-endpoint', (req, res) => {
     res.send('DELETE request');
   });
```

Middleware:

Los middlewares en Express son funciones que se ejecutan durante el ciclo de vida de una petición. Pueden modificar la solicitud y la respuesta, y pueden terminar el ciclo de solicitud-respuesta o llamar al siguiente middleware en la pila.

```
app.use(express.json()); // Para manejar JSON
app.use(express.urlencoded({ extended: true })); // Para manejar datos
codificados en URL
```

Generación de Respuestas:

Express proporciona métodos para enviar respuestas HTTP, incluyendo `res.send()`, `res.json()`, `res.status()`, entre otros.

send():

El método `send()` se utiliza para enviar una respuesta al cliente. Puede enviar texto, HTML o un objeto JavaScript.

```
app.get('/texto', (req, res) => {
     res.send('Respuesta con texto plano);
   });

   app.get('/html', (req, res) => {
     res.send('<h1>Respuesta con HTML</h1>');
   });

   app.get('/objeto', (req, res) => {
     res.send({ message: 'Respuesta como object response' });
   });
```

json():

El método `json()` se utiliza para enviar una respuesta JSON al cliente.

```
app.get('/json', (req, res) => {
     res.json({ message: 'Respuesta JSON' });
   });
```

status()

El método status() se utiliza para establecer el código de estado HTTP de la respuesta. Este método puede ser encadenado con send() o json() para enviar la respuesta con un código de estado específico.

```
app.get('/status-texto', (req, res) => {
  res.status(200).send('Respuesta texto plano con status 200');
});

app.get('/status-html', (req, res) => {
  res.status(201).send('<h1>Respuesta HTML con status 201</h1>');
});
```

```
app.get('/status-objeto', (req, res) => {
  res.status(202).send({ message: Respuesta en objeto 202' });
});

app.get('/status-json', (req, res) => {
  res.status(203).json({ message: Respuesta JSON con status 203' });
});
```

Configurabilidad:

Express es altamente configurable, permitiendo ajustar su comportamiento a través de variables de entorno y archivos de configuración.

Integración con Plantillas:

Puede integrarse fácilmente con motores de plantillas como Pug, EJS, Handlebars, etc., para generar HTML dinámico.

Obtener parametros

```
app.get('/user/:id', (req, res) => {
     const userId = req.params.id;
     res.send(`User ID es ${userId}`);
   });
```

Cuerpo de las peticiones:

Para manejar el cuerpo de las peticiones, es necesario utilizar middleware como `express.json()` y `express.urlencoded()`.

```
app.use(express.json());
   app.use(express.urlencoded({ extended: true }));

   app.post('/user', (req, res) => {
     const userData = req.body;
     res.send(`User data received: ${JSON.stringify(userData)}`);
   });
```

Ejemplo de servidor

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});

app.get('/', (req, res) => {
  res.send('¡Hola Mundo!');
});

app.get('/usuarios', (req, res) => {
  res.send('Lista de usuarios');
});

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