06 WEBSERVER

REDACCION DE APPS

**Objetivos:**

* **Crear un webserver y formar las bases para los siguientes temas**
* **Tomar una carpeta de nuestra apicacion de node y hacerla publica**

**CLASE 64.- Inicio del proyecto – Webserver**

Objetivo: Crear un primer Webserver

PUBLICAR UN TEXTO EN PAGINA WEB

En File Explorer

* Crear nuevo folder ’06-Webserver’
* Arrastrar este folder ’06-Webserver’ a VS Code para trabajarlo ahi

En consola

* Cd ’06-Webserver’
* Teclear ‘npm init’ (dar enter a todas las opciones)

Este proceso Inicializa ‘npm’ en este folder

En VS Code

En folder ’06-webserver’

* Crear file ‘app.js’

En navegador web

* Abrir pagina ‘nodejs.org’
* Seleccionar pestaña ‘Documentacion’
* Seleccionar la version API mas reciente
* Localizar ‘HTTP’ y solo verlo – es un pck que permite crear facilmente un webserver

En VS code

En ‘app.js’

* Poner al inicio

const http = require('http');

* Poner funcion de flecha

http.createServer((req, res) => {

res.write('Hola mundo');

res.end();

})

.listen(8080);

Esta funcion ‘levanta’ el servidor y se mantiene ‘escuchando’ el puerto 8080

* Poner la instrucción

console.log('Escuchando el puerto 8080');

En consola

* Teclear ‘node app’ – para levantar el servidor

debe aparecer el texto ‘Escuchando puerto 8080‘

En navegador web

* En recuadro URL teclear ‘localhost: 8080’
* Debe mostrarse ‘hola mundo’
* Click boton derecho, opcion ‘Inspeccionar’; debe abrirse ventana a la derecha y mostrarse el codigo html que se creo con la app

App-01-http.js

PUBLICAR UN TEXTO JSON EN PAGINA WEB

En VS code

* Comentar o remover la instrucción

res.write('Hola mundo');

* Poner la instruccion

res.writeHead(200, { 'Content-Type': 'application/json' });

Esta instrucción indica que se va a publicar un ‘json’

* Declarar el objeto ‘salida’

let salida = {

nombre: 'Ernesto',

edad: 58,

url: req.url

};

* Poner
* res.write(JSON.stringify(salida));
* Esta instrucción publica el objeto ‘salida’ en forma JSON

En consola

* Bajar el servidor (ctl-C)
* Teclear ‘node app.js’ - Levantar el servidor

Esto tiene que hacerse porque el servidor no detecta que hizo un cambio en ‘app.js’, mas adelante se vera como hacer que si detecte cualquier cambio

En navegador web

* Refrescar pantalla – debe aparece el objeto ‘salida {nombre, edad, url}’
* Seleccionar el URL ‘localhost:8080’ y copiarlo a Postman en la ventana ‘GET’ – para comprobar

En postman

* Dar click a ‘Send’
* En ventana de abajo debe aparecer el objeto ‘salida {nombre, edad, url}’
* En la pestaña ‘Headers’, en el content-type debe aparecer el ‘application-json’ solicitado a publicar

App-02-http-json.js

**CLASE 65.- Introduccion a EXPRESS**

Objetivo: usar el pck EXPRESS para crear un Webserver

ESCUCHAR EL PUERTO 3000; ‘/’; PUBLICAR ‘HOLA MUNDO’ CON EXPRESS

En consola

* Cd ’06-Webserver’

En navegador web

* Buscar ‘npm express’
* Abrir pagina ‘http://www.npmjs.com/package/express’
* Localizar a la derecha la seccion ‘install’; seleccionar ‘npm i express’ y copiarlo al prompt en la consola

En consola

* A la instrucción ‘npm i express’ añadirle ‘ -- save’ (porque es una dependencia que necesitaremos para correr el programa) y dar enter – se debe intalar el express

En VS Code

* renombrar el ‘app.js’ a ‘app-01-HTTP’ para no confundirlo posteriormente
* Crear un file nuevo ‘server.js’

En navegador web

En pagina ‘npm express’

* Localizar y selecciónar el codigo para usar ‘express’ (a la izquierda) y copiarlo al file ‘server.js’

const express **=** require('express')

const app **=** express()

app.get('/', function (req, res) {

  res.send('Hello World')

})

app.listen(3000)

En VS Code

En file ‘server.js’

* Si las declaraciones son con ‘var’, cambiarlas por ‘const’ porque seran constantes que nunca se van a modificar

const express = require('express')

const app = express()

* Reemplazar la funcion por una funcion de ‘flecha’ y cambiar ‘Hello world’ por ‘Hola mundo’

app.get('/', (req, res) => {

res.send('Hola mundo');

});

* Ver que ‘escucha’ el puerto ‘3000’

app.listen(3000, () => {

console.log('Escuchando el puerto 3000');

});

En consola

* Teclear ‘nodemon server.js’

con esta instrucción se levanta el servidor

* Aparece lineas con informacion y el msg ‘Escuchando el puerto 3000’

ver que no hay ningun feedback de este programa

En navegador web

* En recuadro URL teclear ‘localhost:3000’
* En ventana izquiera debe mostrar ‘Hola mundo’
* Boton derecho, opcion ‘inspeccionar’ para ver el codigo; en ventana derecha ver las instrucciones ‘html’ creadas

En postman

* En recuadro ‘GET’ teclear ‘http://localhost:3000’; dar click en boton ‘SEND’
* En ventana abajo,
  + en pestaña ‘Body’ debe mostrarse ‘Hola mundo’
  + en pestaña ‘Headers’ en ‘Content-type’ muestra que es un texto ‘html’ .. no ‘json’ ni ningun otro

Sever-01-express-app.get.js

ESCUCHAR EL PUERTO 3000; ‘/’; PUBLICAR EL OBJETO ‘SALIDA’ CON EXPRESS

En VS Code

En server.js

* Quitar o Comentar la instrucción

res.send('Hola mundo');

En el file ‘app-02-http-json.js’ seleccionar la declaracion del objeto ‘let salida = { …}’ y copiarlo al file ‘server.js’ dentro de la funcion de flecha ‘app.get’

app.get('/', (req, res) => {

let salida = {

nombre: 'Ernesto',

edad: 58,

url: req.url

};

res.send(salida);

});

En ‘server.js’

* Poner la instrucción

res.send(salida);

usando express, esta instrucción internamente hace la conversion a formato ‘JSON’

En postman

* En recuadro ‘GET’ teclear ‘http://localhost:3000’; dar click en boton ‘SEND’
* En ventana abajo,
  + en pestaña ‘Headers’ en ‘Content-type’ muestra que es un ‘application/json’; en X-powered-By: Express
  + en pestaña ‘Body’ debe mostrarse el contenido del objeto ‘salida’
* En ventana ‘GET’ teclear ‘http://localhost:3000/data’; dar click en boton ‘SEND’
* En ventana abajo,
  + en pestaña ‘Body’ debe mostrarse el mensaje ‘cannot GET /data’ – esto ocurre porque la instrucción ‘get’ en el file ‘server.js’ indica escuchar solo ‘/’

Server-02-express-app.get.js

ESCUCHAR EL PUERTO 3000; ‘/data’; PUBLICAR EL OBJETO ‘SALIDA’

Para escuchar ‘/data’

En VS Code

En ‘server.js’

* copiar toda la funcion ‘app.get(‘/’,…’ debajo de si misma
* en esta funcion de flecha copiada borrar la declaracion del objeto ‘salida’

let salida = {

nombre: 'Ernesto',

edad: 58,

url: req.url

};

* cambiar el ‘/’ por ‘/data’ y poner la instrucción ‘res.send(‘Hola data’);

app.get('/data', (req, res) => {

res.send('Hola data');

});

En postman

* En recuadro ‘GET’ teclear ‘http://localhost:3000/data’; dar click en boton ‘SEND’

En ventana abajo, en pestaña ‘Body’ debe mostrarse ‘Hola data’

* En recuadro ‘GET’ teclear ‘http://localhost:3000’; dar click en boton ‘SEND’

En ventana abajo, en pestaña ‘Body’ debe mostrarse el objeto ‘salida’ (nombre, edad, req.url)

En VS Code

En ‘server.js’

* modificar el ‘app.listen(3000)’ a funcion de flecha, con un console.log(‘Escuchando puerto 3000’)

En consola

* ver que nodemon muestra ese texto ‘Escuchando puerto 3000‘

Server-03-express-app.get-data.js

**CLASE 66.- Servir contenido estatico**

Objetivo: Publicar una pagina Web o aplicación Web utilizando un webserver montado en el pck EXPRESS

ESCUCHAR PUERTO 3000, PUBLICAR CONTENIDO DE ARCHIVO ‘INDEX-01’

En VS Code

* crear un nuevo folder ‘public’ (estara disponible para que cualquier persona lo pueda ver)
* en este folder ‘public’, crear un nuevo file ‘index.html’

En ‘index.html’

* Crear codigo template html con ‘! TAB’
* Poner
  + <title>Pagina Web Demo<title>
  + <h1> Hola mundo HTML<h1>
  + <p>Cualquier cosa<p>

En ‘server.js’

* Despues de la declaracion de ‘app = express()’ poner la instrucción:

app.use(express.static(\_\_dirname + '/public'));

Esta instruccion indica se busque en ‘cualquier’ directorio el folder ‘public’ y ahí el file que se le indique en el recuadro URL despues de ‘localhost:3000’

En navegador web

* En el recuadro URL teclear ‘localhost:3000/index-01.html’
* hay muchas probabilidades de que muestre:
  + En la pestaña de la pagina ‘Pagina web | demo’
  + En ventana izquierda ‘Hola mundo HTML’, ‘ Cualquier cosa’

es probable que no, porque no hemos quitado la instrucción ‘app.get()’ en el codigo de ‘server.js’ y puede confundir al interpretador y ‘chocar’ con la instrucción ‘app.use()’ del archivo ‘index-01’

* Con boton derecho, opcion ‘inspeccionar’ para ver el codigo; en ventana derecha ver las instrucciones html creadas por el interpretador

En VS Code

En ‘server.js’

* Quitar o comentar toda la instrucción ‘app.get()’

(para evitar que pueda confundir al interprete y ‘chocar’ con la instrucción ‘app.use()’ de ‘index-01’ al tiempo de ejecucion)

En navegador web

* En el recuadro URL teclear ‘localhost:3000/index-01’
* Debe mostrarse:
  + En la pestaña de la pagina ‘Pagina web | demo’
  + En ventana izquiera ‘Hola mundo HTML’, ‘ Cualquier cosa’
* En el recuadro URL teclear ‘localhost:3000/home’
* Se mostrara:
  + En la pestaña de la pagina ‘Cannot GET /home’

(porque no encontrara hay ningun folder o file llamado ‘/home’)

Server-04-express-app.use.js

Index-01.html

ESCUCHAR PUERTO 3000, PUBLICAR CONTENIDO DE ARCHIVO ‘HOME.HTML’

En VS Code

* En folder ‘public’, crear un nuevo file ‘home.html’

En ‘home.html’

* Crear codigo template html con ‘! TAB’
* Poner
  + <title>Home | Demo<title>
  + <h1> Home Page <h1>
  + <p>blank<p>

En navegador web

* En recuadro URL teclear ‘localhost:3000/home’
* Se mostrara:
  + En la pestaña de la pagina ‘Cannot GET /home’

porque no encontrara hay ningun folder o file llamado ‘/home’

* En el recuadro URL teclear ‘localhost:3000/home.html’
* Se mostrara:
* Debe mostrarse:
  + En la pestaña de la pagina ‘Home Page | demo’
  + En ventana izquiera ‘Home Page’

Server-04-express-app.use.js

home.html

Resumen:

Se puede crear una carpeta publica disponible para todo el publico donde se pueden publicar todo tipo de archivos que los pueda interpretar y publicar el navegador web usando la instrucción

app.use(express.static(\_\_dirname + '/public'));

**CLASE 67.- Crear una pequeña pagina web**

Objetivo: Hacer una pagina web un poco mas elaborada utilizando otros recursos como imágenes, css, etc.

En navegador web

* Abrir la pagina ‘getbootstrap.com’
* Click a boton ‘download’
* Ir a seccion ‘Source files’
* Click al boton ‘Download source’ – se baja el archivo ‘bootstrap-4.0.0’ comprimido
* Abrir archivo y descomprimirlo
* Entrar a folder ‘dist’ y seleccionar folders ‘css’ y ‘js’ (aquí hay muchos archivos)

En explorador de archivos

* Ir al path ’06-webserver/public’, crear un folder nuevo ‘assets’ y copiar en este los folders seleccionados ‘css’ y ‘js’

En consola

* Levantar el servidor – teclear ‘nodemon server’

En navegador web

* En recuadro URL teclear ‘localhost:3000/assets/’
* En ventana aparece el texto ‘Cannot GET /assets/’ ya que no hay un servicio llamado ‘assets’
* En recuadro URL teclear ‘localhost:3000/assets/css/bootstrap.min.css’
* En ventana aparece el codigo de este archvo lo que demuestra que es un URL valido

CAMBIAR FORMATO DE LETRA

En VS Code

En index.html

* Despues de los ‘<meta’ poner

<link rel="stylesheet" href="assets/css/bootstrap.min.css">

En navegador web

* En recuadro URL teclear ‘localhost:3000/index.html’

Ver que cambio el font de la letra del texto ‘Hola Mundo HTML’

PONER BARRA DE NAVEGACION CON MENU HORIZONTAL

En navegador web

En pagina ‘getbootstrap.com’

* Ir a seccion ‘Navbar’
* Localizar el primer ‘Navbar’ en esta seccion
* Localizar la seccion del codigo,

<nav class="navbar navbar-expand-lg navbar-light bg-light">

* Dar click a boton ‘copy’ en la parte superior derecha

En VS code

En index.html

* Pegar codigo de ‘navbar’ despues de la declaracion <body>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<a class="navbar-brand" href="#">Navbar</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="About.html">About</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

Dropdown

</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Something else here</a>

</div>

</li>

<li class="nav-item">

<a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>

</li>

</ul>

<form class="form-inline my-2 my-lg-0">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

</nav>

En navegador web

* En recuadro URL teclear ‘localhost:3000/index.html’
* Ver que en la ventana en la parte superior aparece la barra de navegacion como menu horizontal

CAMBIAR COLOR DEL FONDO DE LA BARRA DE NAVEGACION

En VS Code

En ‘index.html’

* Cambiar el color de la barra de navegacion - En la instrucción <nav class=…’ cambiar las dos palabras ‘light’ por ‘dark’

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

En navegador web

* Abrir o refrescar pagina ‘localhost:3000/index.html’
* Ver que en la ventana en la parte superior aparece la barra de navegacion como menu horizontal con fondo obscuro

PONER CUERPO A LA PAGINA

En navegador web

En pagina ‘getbootstrap’

* Ir a seccion ‘Jumbotron’
* Localizar la caja que dice ‘Hello, world’
* Localizar la seccion del codigo

<div class="jumbotron">

* click a boton ‘copy’ en la parte superior derecha

En VS Code

En ‘index.html’

* Borrar el codigo

<h1> Hola mundo HTML </h1>

<p> Cualquier cosa </p>

En esa zona pegar el codigo de ‘Jumbotron’ despues de la instrucción </nav>

</nav>

<div class="jumbotron">

<h1 class="display-4">Hello, world!</h1>

<p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>

<hr class="my-4">

<p>It uses utility classes for typography and spacing to space content out within the larger container.</p>

<a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>

</div>

En navegador web

* Abrir o refrescar pagina ‘localhost:3000/index.html’
* Ver que en la ventana debajo de la barra de navegacion aparece la caja que dice ‘Hello, world’ y todo el resto de texto contenido en ‘jumbotron’

PONER MENU QRY VERTICAL (PARA EN CASO DE QUE EL ESPACIO EN PANTALLA SE REDUZCA)

En VS Code

En ‘index.html’

* Antes de la instrucción que finaliza el cuerpo </body> poner

<script src="assets/js/bootstrap.min.js"></script>

Esta instrucción importa el qry del menu horizontal en caso de reduccion de espacio en pantalla. Recordar que el folder ‘js’ fue bajado de ‘bootstrap’

En navegador web

En pagina ‘getbootstrap’

* Ir a seccion ‘BootstrapCDN’
* Localizar el codigo de las dos instrucciónes para jQuery y Popper.js

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

* Click al boton ‘copy’ en la parte superior derecha

En VS Code

En ‘index.html’

* Antes de la instrucción

<script src="assets/js/bootstrap.min.js"></script>

La cual fue puesta para importar el qry vertical del menu horizontal

* Pegar las dos instrucciones

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

En navegador web

* Abrir o refrescar pagina ‘localhost:3000/index.html’
* Minimizar pantalla o abrir ventana de inspeccionar
* Ver que en la parte izquierda de la barra de navegacion aparece un boton con unas lineas que indican que se puede abrir el menu en forma de qry vertical
* Dar click a este boton ‘qry’
* Ver que se abren las opciones del menu en forma vertical
* Dar clcik a este boton ‘qry’ de nuevo
* Ver que se cierran las opciones del menu en forma vertical
* Ver que no haya ningun error en la pestaña ‘console’ de la ventana ‘inspeccionar’

LLAMAR AL ARCHIVO about.html DESDE EL ARCHIVO index.html

En VS Code

* Renombrar el archivo ‘home.html’ a ‘about.html’

En archivo ‘about.html’

* Cambiar esta linea a

<title>About | Demo</title>

En archivo ‘index.html’

* Localizar dos instrucciones ‘<a class=”nav-link” href=…’ .. la de ‘Home’ y la de ‘About’
* Cambiar los textos, los paths, y en el ‘about’ el nombre del archivo a invocar

<a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>

<a class="nav-link" href="About.html">About</a>

En navegador web

* Abrir o refrescar pagina ‘localhost:3000/index.html’
* Click en opcion ‘About’ en barra de navegacion
* Ver que accesa y muestra la pagina ‘about.html’
* Regresar con flecha izquierda
* Click en opcion ‘Home’ en barra de navegacion solo para saber que debe regresar a ‘Home’

PONER FOOTER

En VS Code

En ‘index.html’

* Antes de los ‘scripts’

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

* poner

<footer>

Ernesto Gonzalez - 2019

</footer>

DEFINIR UN ESTILO DE FORMATO

En folder ’06-webserver/assets/css’

* Crear nuevo archivo ‘style.css’

En archivo ‘style.css’

* Poner los estilos deseados para el ‘footer’

footer {

background-color: royalblue;

color: white;

width: 100%;

position: fixed;

bottom: 0px;

text-align: center;

padding: 10px;

}

En archivo ‘index.html’

* Debajo de

<link rel="stylesheet" href="assets/css/bootstrap.min.css">

* Poner esta linea

<link rel="stylesheet" href="assets/css/style.css">

Esta instrucción Importa el archivo ‘style.css’

En navegador web

* Abrir o refrescar pagina ‘localhost:3000/index.html’
* Ver que aparece el ‘footer’ con el estilo que se definio en ‘style.css’

Server-04-express-app.use.js

Index-02.html

about.html

**CLASE 68.- Handlebars**

Objetivo: Usar ‘engines’ para hacer un sitio web dinamico

En navegador web

* Abrir la pagina ‘handlebarsjs.com’ y solo verla
* Teclear ‘npm hbs’
* Abrir el paquete ‘hbs – npm’ <https://www.npmjs.com/package/hbs>
* Buscar la seccion ‘Install’
* Seleccionar la instrucción

npm install hbs

En consola

* Ir a path ’06-webserver’
* Pegar/teclear la instruccion ‘npm install hbs’ y añadir ‘–-save’ (dos guiones)

instalacion de hbs

En navegador web

En la pagina hbs <https://www.npmjs.com/package/hbs>

* Localizar la seccion ‘Use’
* Seleccionar la instrucción

app.set('view engine', 'hbs');

en VS Code

En server.js

* Debajo de la instruccion

app.use(express.static(\_\_dirname + '/public'));

* Poner la linea

// Express hbs engine

* Pegar la linea

app.set('view engine', 'hbs');

en VS Code

En editor

* En folder ‘public’, renombrar el archivo ‘index.html’ a ‘index-02.html’ (para no confundirlo)
* En folder ’06-webserver’, crear folder ‘views’
* En folder ‘views’, crear archivo ‘home.hbs’
* En ‘index-02.html’ seleccionar todo el codigo y copiarlo a ‘home.hbs’ (son mas de 70 lineas ya que es el codigo que publica la pagina)

En consola

* Checar que se instalo correctamente ‘hbs’
* Teclear ‘nodemon server.js’ (para levantar el servidor)
* Debe mostrarse el mensaje ‘escuchando peticiones en el puerto 3000’

En navegador web

* En recuadro URL, teclear ‘localhost:3000’
* En ventana debe mostrarse el texto ‘Cannot GET /’ (porque no esta recibiendo instrucción de ningun programa)
* En recuadro URL, teclear ‘localhost:3000/index-02.html’
* En ventana debe mostrarse la pagina creada con este programa
* En recuadro URL, teclear ‘localhost:3000’ y dejarlo asi

En VS Code

En ‘server.js’

* Despues de la instrucción

app.set('view engine', 'hbs');

* Poner esta funcion de flecha

app.get('/', (req, res) => {

res.render('home', {

nombre: 'Ernesto',

anio: new Date().getFullYear()

});

});

Con la instrucción ‘res.render’ se enviara a la pagina web el output del archivo ‘home.hbs’, o sea la ‘pagina web | demo’ con el contenido de las variables ‘nombre’ y ‘anio’ declaradas en el objeto de esta instrucción ‘res.render’

En ‘home.hbs’

* En la seccion ‘<div class=…’, modificar esta instrucción

<h1 class="display-4">Hello, {{ nombre }}!</h1>

* En la seccion ‘footer’, modificar esta instrucción asi

Ernesto Gonzalez - {{ anio }}

Para que en la pagina web muestre las variables ‘nombre’ y ‘anio’ tal como se definieron en el file ‘server.hbs’ en la instrucción ‘app.get(..)’

En navegador web

* En recuadro URL, teclear ‘localhost:3000’
* En ventana debe mostrarse la pagina ‘Pagina Web | Demo’ creada por el file ‘home.hbs’ con las variables ‘nombre’ y ‘anio’ definidas en el file ‘server.hbs’ en la instrucción ‘app.get(..)’

Server-05-express-render.js

Home-01.hbs

about.html

**CLASE 69.- Usando parciales con HBS**

Objetivos:

* Crear los files ‘head.hbs’, ‘navbar.hbs’, y ‘footer.hbs’, y quitar ese codigo de los files ‘home.hbs’, y ‘about.hbs’, para que puedan ser importados desde estos mismo flies o desde cualquier file sin tener que repetirlas en cada uno de ellos
* Crear el file ‘About.hbs’ para que sea llamado desde el file ‘Home.hbs’ con el boton correspondiente en el ‘Navbar’

En navegador web

* Abir la pagina ‘hbs - npm’ <https://www.npmjs.com/package/hbs>
* Ir a seccion ‘Helpers and Partials’
* Seleccionar la instruccion

var hbs **=** require('hbs');

* Copiarla a file ‘server.js’ debajo de la declaracion de la constante ‘app = express();’ y cambiar ‘var’ por ‘const’

const app = express();

const hbs = require('hbs');

* Ir a la pagina ‘hbs - npm’ <https://www.npmjs.com/package/hbs>
* Ir a seccion ‘Helpers and Partials’
* Ver la instruccion

hbs.registerPartials(\_\_dirname **+** '/views/partials' [, callback]);

solo verla ya que sera usada en este ejercicio con su repectiva customizacion

En VS code

* En folder ‘views’, crear el folder ‘parciales’ quedando ‘views/parciales’

En file ‘server.js’

* Teclear la sig instrucción debajo de el comentario ‘// express hbs engine’

// Express hbs engine

hbs.registerPartials(\_\_dirname + '/views/parciales');

esta instruccion le indica al ‘hbs’ que en este path estaran funciones que se podran invocar desde cualquier otro file

CREAR EL PARCIAL ‘NAVBAR’

En VS code

* En folder ‘views/parciales’ crear el file ‘navbar.hbs’
* En el file ‘home.hbs’, cortar todo el codigo de ‘navbar’ desde ‘<nav ..’ hasta ‘</nav’, y pegarlo en el file ‘navbar.hbs’ recientemente creado
* <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
* <a class="navbar-brand" href="#">Navbar</a>
* <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
* <span class="navbar-toggler-icon"></span>
* </button>
* <div class="collapse navbar-collapse" id="navbarSupportedContent">
* <ul class="navbar-nav mr-auto">
* <li class="nav-item active">
* <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
* </li>
* <li class="nav-item">
* <a class="nav-link" href="/about">About</a>
* </li>
* <li class="nav-item dropdown">
* <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
* Dropdown
* </a>
* <div class="dropdown-menu" aria-labelledby="navbarDropdown">
* <a class="dropdown-item" href="#">Action</a>
* <a class="dropdown-item" href="#">Another action</a>
* <div class="dropdown-divider"></div>
* <a class="dropdown-item" href="#">Something else here</a>
* </div>
* </li>
* <li class="nav-item">
* <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
* </li>
* </ul>
* <form class="form-inline my-2 my-lg-0">
* <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
* <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
* </form>
* </div>
* </nav>

En consola

* Ir al path ’06-webserver’
* Levantar el servidor con la instrucción ‘nodemon server’

En navegador web

* En recuadro URL teclear ‘localhost:3000’

debe verse la pagina ‘home.hbs’ SIN el ‘navbar’ ya que el codigo fue ‘quitado’ de este file y no se le ha indicado que lo invoque del ‘navbar.hbs’

En VS code

En file ‘home.hbs’

* Despues de la instrucción <body>, teclear la instrucción

{{> navbar }}

Nota: Esta instrucción importa o invoca la funcion ‘navbar.hbs’

En navegador web

* En recuadro URL teclear ‘localhost:3000’ o refrescar la pagina

debe verse la pagina ‘home.hbs’ CON el ‘navbar’

CREAR EL PARCIAL ‘HEAD’

En VS code

* En folder ‘views/parciales’ crear el file ‘head.hbs’
* En file ‘home.hbs’, cortar todo el codigo de ‘head’ desde ‘<!DOCTYPE html>’ hasta ‘<body>’, y pegarlo en ‘head.hbs’

En VS code

En file ‘home.hbs’

* Al inicio, poner la instrucción

{{> head }}

Nota: Esta instrucción importa o invoca la funcion ‘head.hbs’

En navegador web

* En recuadro URL teclear ‘localhost:3000’ o refrescar la pagina

Debe aparecer error indicando que no encuentra la pagina ‘head’; esto se debe a que el ‘nodemon’ no detecta nuevos archivos ‘hbs’ creados despues de que se levanto el servidor cuando se tecleo el nodemon la ultima vez.

En consola

* Bajar y levantar el servidor nuevamente tecleando la instrucción:

‘nodemon server -e js,hbs,html,css’

Nota: Esta instrucción levanta el servidor (ejecutando el file ‘server.js’) y el argumento ‘-e’ le indica que este pendiente permantemente de la creacion o cambios a todos los archivos que tengan esas extensiones de manera que en lo subsecuente se estaran detectando nuevos archivos o cambios a los archivos con esas extensiones y deberan ser reflejados al refrescar la pagina que escucha el puerto 3000

En navegador web

* En recuadro URL teclear ‘localhost:3000’ o refrescar la pagina

Debe aparecer la pagina ‘home’ ya mostrando el ‘head’

CREAR EL PARCIAL ‘FOOTER’

En VS code

* En folder ‘views/parciales’ crear el file ‘footer.hbs’
* En file ‘home.hbs’, cortar todo el codigo de ‘footer’ desde ‘<footer>’ hasta ‘</html>’, y pegarlo en ‘footer.hbs’

En VS code

En file ‘home.hbs’

* Al final, teclear la instrucción

{{> footer }}

Nota: Esta instrucción importa o invoca la funcion ‘footer.hbs’

En navegador web

* En recuadro URL teclear ‘localhost:3000’ o refrescar la pagina

Debe aparecer la pagina ‘home’ ya mostrando el ‘footer’

CREAR LA PAGINA ‘ABOUT’

En VS code

* En folder ‘views’ crear el file ‘about.hbs’
* Teclear las instrucciones:

{{> head }}

{{> navbar }}

{{> footer }}

En VS code

* En el path ‘public/assets’ crear el folder ‘img’
* Localizar un archivo con alguna imagen y copiarlo al path ‘public/assets/img’

Nota: para este ejercicio usamos el archivo ‘En-cual-vas.jpg’

En el file ‘about.hbs’

* Despues de la instrucción que invoca al ‘navbar’, teclear las siguientes instrucciones:

<div class="row">

<div class="col-8">

<h1>En que etapa vas?</h1>

<p>

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Accusamus, eos incidunt. Rem voluptatem unde aspernatur exercitationem sed laudantium vel at quo natus ut, voluptate minus cumque explicabo impedit illo doloremque?

</p>

</div>

<div class="col-4">

<img src="assets/img/en-cual-vas.jpg" class="img-thumbnail">

</div>

</div>

Nota: estas instrucciones son para mostrar un texto y una imagen al llamarse la pagina ‘about.hbs’. Notese que en la instrucción <img src=…> se le esta indicando el path y el nombre del archivo ‘en cual vas.jpg’ que contiene la imagen que se quiere mostrar en dicha pagina ‘about.hbs’

En el file ‘server.js’

* Despues del bloque de instrucciónes ‘app.get(..) para el render de ‘home’, Poner las siguientes instrucciones para hacer el render de ‘about’

app.get('/about', (req, res) => {

res.render('about', {

anio: new Date().getFullYear()

});

En el file ‘navbar.hbs’

* Modificar las siguiente instruccion

<a class="nav-link" href="/about">About</a>

Nota: esta instrucción le indica al programa ‘home.hbs’ que al dar click en la opcion ‘about’del ‘navbar’ debe abrir el archivo ‘about.hbs’ del cual según la anterior instrucción debe hacerse el ‘render’ a la consola

En navegador web

* En recuadro URL teclear ‘localhost:3000’ o refrescar la pagina
* Dar click en la pestaña ‘about’ del ‘navbar’

Debe aparecer la pagina ‘about’ mostrando el ‘head’, el ‘navbar’, el ‘footer’ invocados y el contenido de texto e imagen que se indico en el body del archivo ‘about.hbs’

Server-06-hbs-parciales.js

Footer.hbs

Head.hbs

navbar.hbs

About.hbs

Home.hbs

**CLASE 70.- Usando Helpers**

Objetivo:

* Crear un archivo separado (helpers) donde se graben funciones comunes como la publicacion de un texto o la obtencion de una fecha, para que puedan ser disparadas desde diferentes programas o desde diferentes secciones de un mismo programa cuando se requieran

En VS code

En file ‘server.js’

* En las dos sentencias ‘app.get’ en la instrucción ‘res.render()’ quitar el argumento ‘anio:..’ quedando asi
* app.get('/', (req, res) => {
* res.render('home', {
* nombre: 'erNesto gLez bermUdez',
* });
* });
* app.get('/about', (req, res) => {
* res.render('about', {
* });

En consola

* Levantar el servidor con la instrucción

‘nodemon server -e js,hbs,html,css’

En navegador web

* En recuadro URL Abrir / refrescar la pagina ‘localhost:3000’

En el footer no debe aparecer el año

En VS code

En file ‘server.js’

* Debajo de esta sentencia

app.set('view engine', 'hbs');

* Poner las siguientes instrucciones

// helpers

hbs.registerHelper('getAnio', () => {

return new Date().getFullYear();

});

En el file ‘footer.hbs’

* Cambiar a ‘getAnio’ el contenido entre los corchetes { }

Ernesto Gonzalez - {{ getAnio }}

Esto indica que al ejecutarse el ‘footer.hbs’ se invoque el helper ‘getAnio’ que en este momento se encuentra en el file ‘server.js’

En navegador web

* En recuadro URL Abrir / refrescar la pagina ‘localhost:3000’

En el footer ya debe aparecer el año que fue obtenido en la instrucción rgister.helper

En VS code

En file ‘server.js’

* Debajo del primer helper poner las instrucciones de este otro helper

hbs.registerHelper('capitalizar', (texto) => {

let palabras = texto.split(' ');

palabras.forEach((palabra, idx) => {

palabras[idx] = palabra.charAt(0).toUpperCase() palabra.slice(1).toLowerCase();

});

return palabras.join(' ');

});

Este helper hara las siguientes acciones:

* con el ‘split’ encuentra todas las palabras separadas por espacio y las asigna al arreglo ‘palabras’ siendo cada palabra un elemento del arreglo
* con el ‘forEach’ hace un ciclo del arreglo ‘palabras’ y a cada uno de los elementos les pone mayuscula a la primera letra y minusculas al resto de las letras
* con el join junta todos elementos del arreglo, ya modificados, separandolos por un espacio en blanco

En file ‘home.hbs’

* Modificar esta sentencia como se indica dentro de los corchetes { }

<h1 class="display-4">Hello, {{ capitalizar ‘erNesto gonzaleZ’ }}!</h1>

En navegador web

* En recuadro URL Abrir / refrescar la pagina ‘localhost:3000’

Debe aparecer el nombre con la primera letra mayuscula y el resto minusculas asi ‘Ernesto Gonzalez’

En file ‘home.hbs’

* Modificar esta sentencia como se indica dentro de los corchetes { }

<h1 class="display-4">Hello, {{ capitalizar nombre }}!</h1>

La variable ‘nombre’ que tiene el contenido ‘erNesto gLez bermUdez ’ se declara en el file ‘server.js’, en la funcion ‘res.render(‘home’…’, en el objeto que se envia al file ‘home.hbs’

En navegador web

* En recuadro URL Abrir / refrescar la pagina ‘localhost:3000’

Debe aparecer el nombre con la primera letra de cada palabra en mayuscula y el resto en minusculas asi: ‘Ernesto Gonzalez Bermudez’

CREAR HELPERS

En VS Code

* En el directorio ’06-webserver’ crear el folder ‘hbs’
* En el folder ‘hbs’ crear el file ‘helpers.js’
* Cortar todo el bloque de helpers del file ‘server.js’ y pegarlo en el file ‘helpers.js’

// helpers

hbs.registerHelper('getAnio', () => {

return new Date().getFullYear();

});

hbs.registerHelper('capitalizar', (texto) => {

let palabras = texto.split(' ');

palabras.forEach((palabra, idx) => {

palabras[idx] = palabra.charAt(0).toUpperCase() palabra.slice(1).toLowerCase();

});

return palabras.join(' ');

});

* Copiar la instrucción del file ‘server.js’ al file ‘helpers.js’ en la primera linea

const hbs = require('hbs');

Esta instrucción define el ‘hbs’ tambien en el file ‘helpers.js’

* En el file ‘server.js’ despues de la instruccion

const hbs = require('hbs');

* poner la instrucción para requerir el file ‘helpers.js’

require('./hbs/helpers');

En navegador web

* En recuadro URL Abrir / refrescar la pagina ‘localhost:3000’

Debe aparecer el nombre con la primera letra mayuscula y el resto minusculas asi: ‘Ernesto Gonzalez Bermudez’; observese que los ‘helpers’ ya fueron ubicados en el file ‘helpers.js’ y estan siendo importados por el ‘home.hbs’ al invocar el helper ‘capitalizar nombre’ y por el ‘footer.hbs’ al invocar el helper ‘getAnio’.

Server-07-hbs-helpers.js

Home.hbs

En-cual-vas.jpg

About.hbs

/Views/

/Public/Assets/img/

Helpers.hbs

Head.hbs

navbar.hbs

/Views/parciales/

/hbs/

Footer.hbs

**CLASE 71.- Heroku – Subiendo nuestra aplicación a produccion**

Objetivo:

* Subir nuestra aplicación a la nube y que pueda ser accesada en todo el mundo a traves de internet simplemente tecleando un URL

En navegador web

* Buscar ‘heroku’
* Accesar la pagina ‘heroku: cloud application platform’ – <https://www.heroku.com/>
* Crear cuenta con e-mail y contraseña (recordarlos porque se requeriran posteriormente)
* Log in
* Si hay aplicaciones creadas se muestran en la primer pantalla
  + Si se quiere borrar alguna:
    - dar click en ella,
    - ir a ‘settings’,
    - ir a la parte inferior,
    - dar click a boton ‘delete app’,
    - teclear nombre de la aplicación,
    - dar click en boton ‘delete app’
* Dar click en boton ‘New’ (parte superior derecha)
* Dar click en ‘create new app’
* Teclear el ‘app-name’; ( eg-lobo-webpage )
* Seleccionar la region mas cercana al pais
* Dar click a boton ‘Create app’

Pasa a la siguiente pantalla

* Ver los botones de opciones
* Localizar la seccion ‘Install the Heroku CLI’
* Click al link ‘Downlowad and install the heroku CLI’

Pasa a la siguiente pantalla

* En la parte inferior dar click al sistema operativo correspondiente
* Se descarga un archivo, abrirlo, instalarlo, dar ‘next’ a todas las opciones, y finish

Nota: cerrar la pantalla de comandos en la consola (si este abierta)

En consola

* Teclear ‘heroku -v’

Para ver la version de heroku

En VS code

En file ‘server.js’

* Despues de la linea

require('./hbs/helpers');

* poner la linea

const port = process.env.PORT || 3000;

* modificar estas lineas

app.listen(port, () => {

console.log(`Escuchando el puerto ${ port }`);

});

En file ‘package.json’

* Despues de la linea

"scripts": {

* Poner las lineas

"start": "node server.js",

"nodemon": "nodemon server.js",

En ‘administrador de archivos’ de VS code

* Bajo el folder ’06-webserver’, crear el file ‘.gitignore’

En file ‘.gitignore’

* Teclear ‘node\_modules/’

Recordar que los files que se indiquen en este file seran excuidos de cualquier proceso ‘git’

Cerrar todos los archvios de VS code (para no confundirse)

PREPARACION DE ARCHIVOS CON GIT

En consola

* Teclear ‘git init’

Para inicializar el repositorio

* Teclear ‘git status’

Ver que se muestren todos los files excepto el ‘node\_modules/’

* Teclear ‘git add .’
* Teclear ‘git commit -m “Primer commit’

Deben mostrarse todos los archivos del folder ’06-webserver’

* Teclear ‘git status’

Debe mostrarse el mensaje ‘On branch master nothing to commit, working tree clean’

SUBIR EL RESPOSITORIO A HEROKU

En consola

* Teclear ‘heroku login’
* Teclear credenciales heroku: email, contraseña (las que se dieron cuando se creo la cuenta)

Debe aparecer el msg de que el ‘Log in’ en heroku se hizo correctamente

En navegador web

En pagina ‘heroku.com’

* Ir a seccion ‘create a new Git repository’
* Seleccion instruccion ‘heroku git:remote -a eg-lobo-webpage’ y copiarla a la consola

En consola

* Dar enter para ejecutar el comando copiado – ver que no marque error

En navegador web

En pagina ‘heroku.com’

* Ir a seccion ‘deploy your application’
* Seleccionar instruccion ‘git push heroku master’ y copiarla a la consola

En consola

* Dar enter para ejecutar la instrucción copiada

Heroku inicia el proceso, detecta que es una aplicación de node, instala los paquetes de node, dispara el proceso de ‘npm start’

* Al terminar la instalacion entrega un URL https

<https://eg-lobo-webpage.herokuapp.com/>

* Para ejecutar la pagina se puede llamar este link o bien teclear ‘heroku open’
* En ambos casos, en el navegador debe abrirse la ‘pagina web | demo’