



Clase 9. Programación Backend

Motores de Plantillas



OBJETIVOS DE LA CLASE

- Entender qué es un motor de plantillas y su implementación en el backend.
- Conocer el motor de plantillas Handlebars: sintaxis y uso, e integrarlo a Express

CRONOGRAMA DEL CURSO

Clase 8



Router & Multer

Clase 9



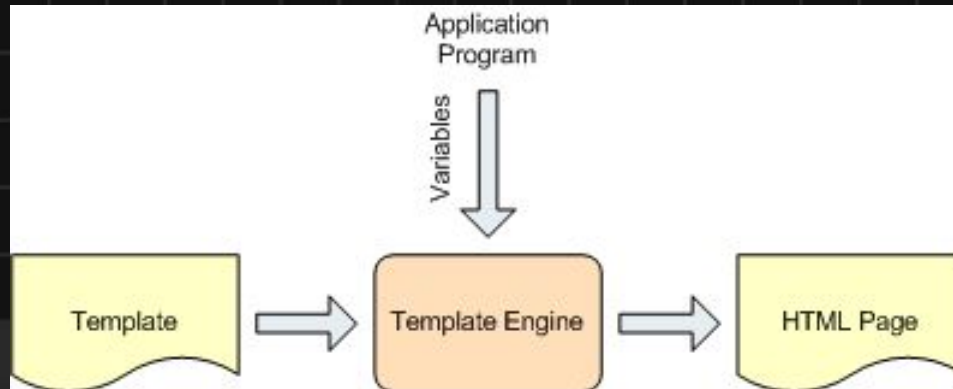
Motores de Plantillas

Clase 10



Pug & Ejs

Motores de plantillas (Template engines)

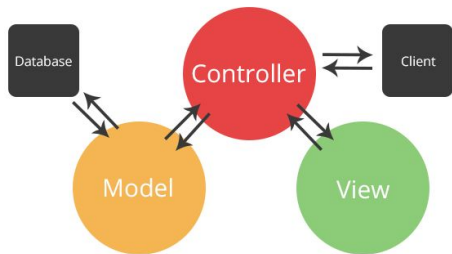




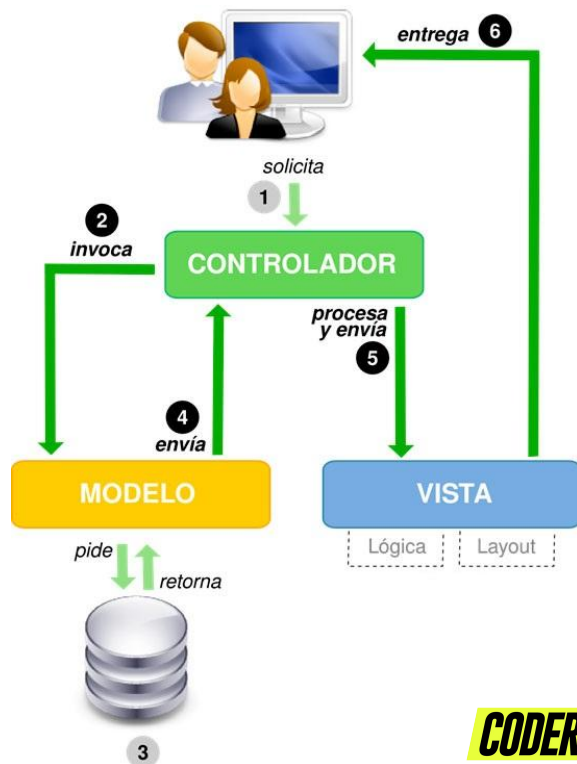
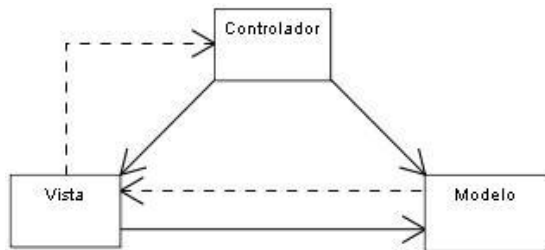
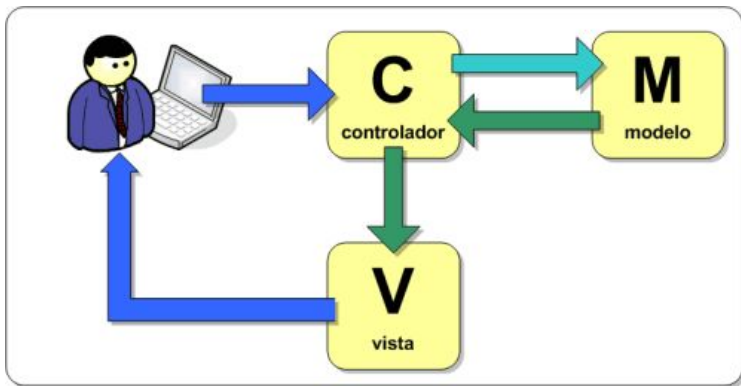
Concepto MVC

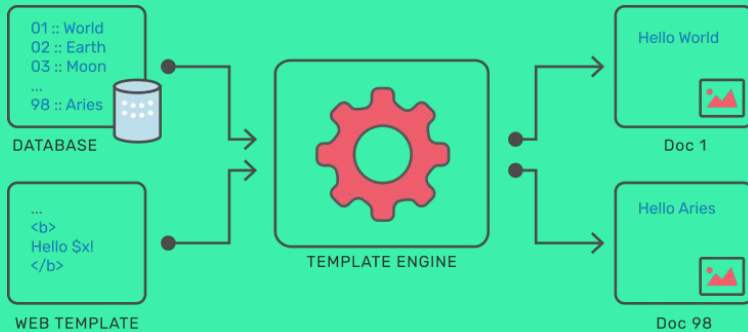
Modelo Vista Controlador

- En la programación en general y en la programación web en particular existe el denominado patrón **MVC** (*Modelo Vista Controlador*)
- Este patrón trata de **separar los datos de su presentación**. Por decirlo en términos web, separar el código del programador del código del diseñador web.
- Las plantillas (templates) son una aproximación más para resolver este problema.



Esquemas

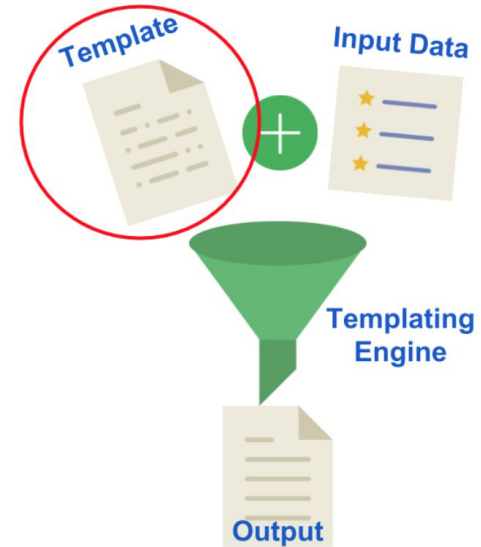
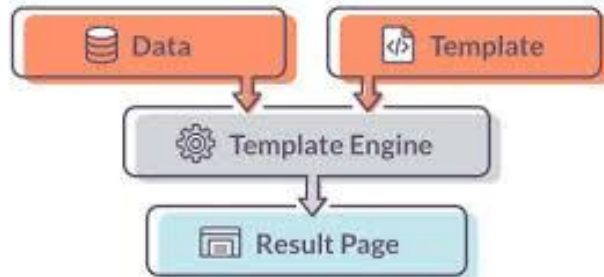
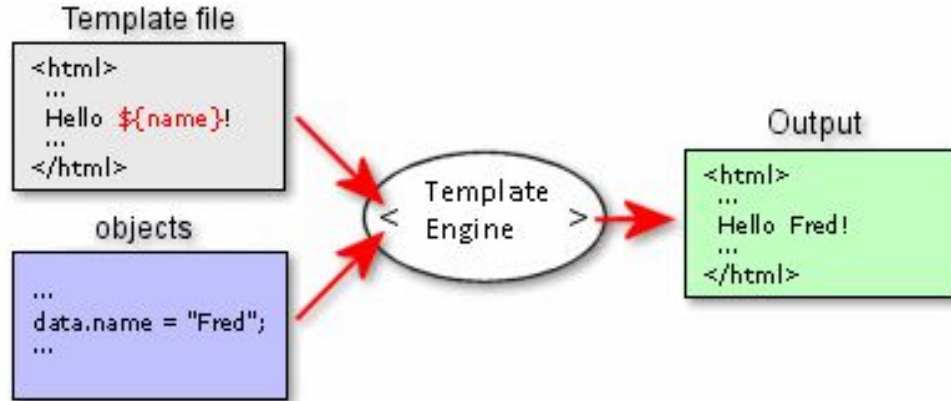




Motores de Plantillas

- Un motor de plantillas **lee** un **archivo de texto** (plantilla) que contiene la presentación ya preparada en un **lenguaje Pseudo HTML** e **inserta** en él la **información dinámica** que le ordena el "controlador" (la C de MVC) que representa la parte que **une la vista con la información**.
- La sintaxis a utilizar depende del motor de plantillas utilizado.
- Los motores de plantillas suelen tener un pequeño lenguaje de script que permite generar código dinámico.

Esquemas





Ventajas de utilizar un motor de plantillas

- El **código** resultante es **más organizado**, y tenemos garantía de que no habrá HTML mal formado.
- Podemos **separar nuestro equipo en dos**, al trabajar interfaces de usuario sin necesidad de desarrollar en Back-End.
- Los motores de plantilla nos permiten **reutilizar secciones de código** ayudando así a mantener nuestro proyecto optimizado.
- Existen muchas utilidades que nos ayudan a dotar de **mejor interacción** a la parte visual de nuestras aplicaciones.
- Con los motores de plantillas podemos separar, optimizar y organizar nuestro código




Desventajas de NO utilizar un motor de plantillas

- La no utilización de un motor de plantillas puede afectar la **velocidad** de nuestro desarrollo de aplicaciones.
- El riesgo de hacer **HTML mal formado** es mucho mayor, por lo que obtener certificaciones para nuestro código se puede hacer difícil.
- El código resultante puede resultar **difícil de documentar** y de compartir con otros desarrolladores.
- Si utilizamos código para generar HTML siempre será incómodo trabajar con **caracteres especiales**.
- Hay tendencia a no separar la lógica de aplicación de la presentación.

Ejemplo genérico



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Documento</title>
</head>
<body>
  Hola {{nombre}}
</body>
```

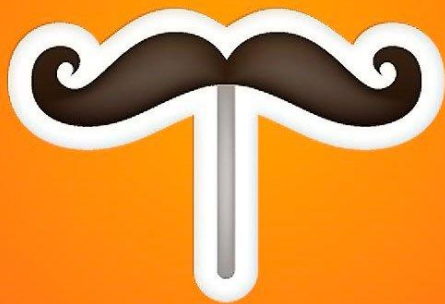


```
let datos = { nombre : 'Juan' }
```

Al procesar este fichero, el **motor de plantillas** lo recorrerá, parseará y sustituirá esa "etiqueta clave" **{{nombre}}** por el valor que le hayamos indicado (por ejemplo el nombre del visitante) de forma que tengamos una presentación personalizada.

Handlebars

handlebars





¿Qué es Handlebars?



- **Handlebars** es un **lenguaje de plantillas** simple.
- Utiliza una plantilla y un objeto de entrada para generar HTML u otros formatos de texto.
- Las plantillas de Handlebars tienen el aspecto de texto normal con **expresiones de Handlebars incrustadas**.
- Una expresión de Handlebars se compone de `{{ + algunos contenidos + }}`
- Cuando se **ejecuta** la plantilla, las **expresiones de Handlebars** se **reemplazan** con **valores** de un objeto de **entrada**.

Ejemplo Handlebars Online

 Handlebars 

Template

```
<p>Hola {{nombre}} {{apellido}}</p>
```

Input

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

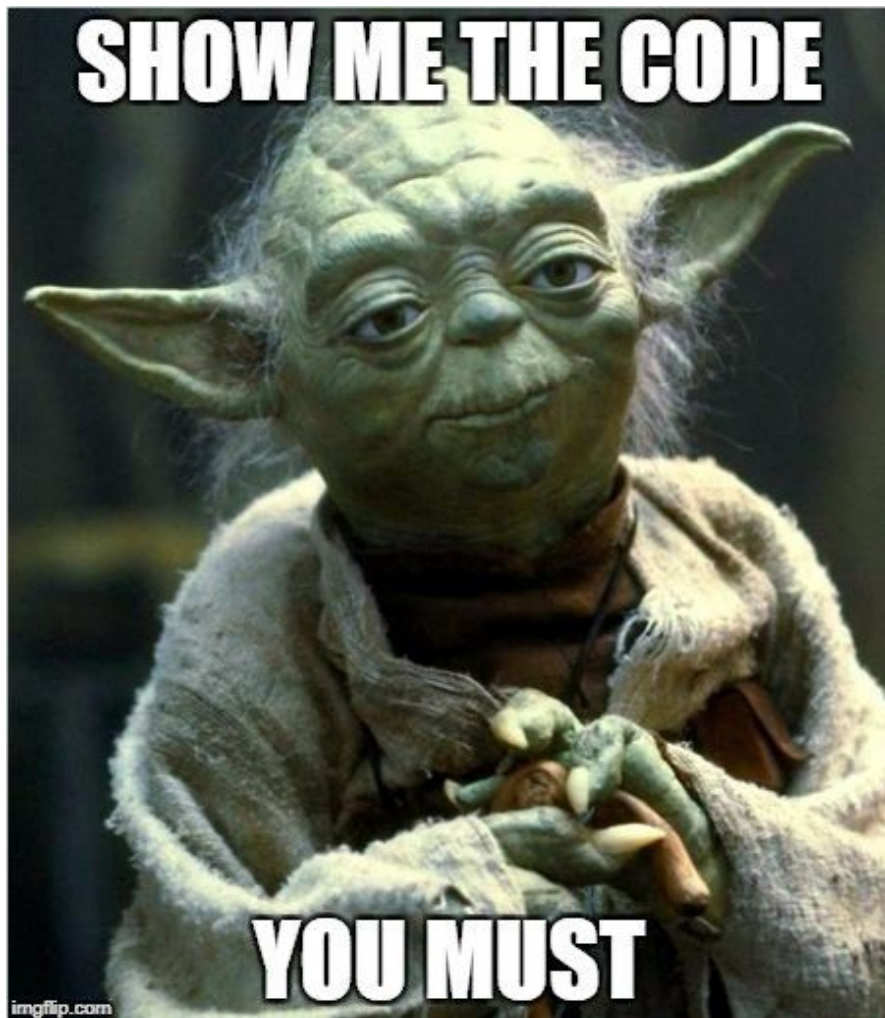
Output

```
<p>Hola Juan Perez</p>
```

<https://handlebarsjs.com/examples/simple-expressions.html>

CODER HOUSE

SHOW ME THE CODE



imgflip.com

CODER HOUSE

Implementación de Handlebars desde el CDN

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <span></span> <!-- para inyectar el resultado -->
  <!-- incluir handlebars desde el CDN -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/4.7.7/handlebars.min.js"/></script>
  <script>
    const template = Handlebars.compile('<h1>{{nombre}}</h1>'); // compila la plantilla
    const html = template({ nombre: 'coder' }); // genera el html
    document.querySelector('span').innerHTML = html; // inyecta el resultado en la vista
  </script>
</body>
</html>
```




Datos personales

Vamos a practicar lo aprendido hasta ahora

Tiempo: 10 minutos

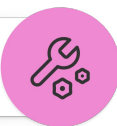


Realizar una página web que permita mostrar datos personales de la siguiente forma:

```
<h1>Datos Personales</h1>
<ul>
  <li>(nombre)</li>
  <li>(apellido)</li>
  <li>(edad)</li>
  <li>(email)</li>
  <li>(teléfono)</li>
</ul>
```

Con los datos que provienen desde un objeto:

```
{
  nombre: '...',
  apellido: '...',
  edad: ...,
  email: '...',
  telefono: '...'
}
```



Importar Handlebars vía CDN en el frontend para crear dicha vista en forma dinámica. Esta página será servida desde el espacio público de un servidor basado en node.js y express.



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

Motores de plantillas para Express

Fuente: <https://expressjs.com/>

CODER HOUSE

Creando un motor de plantillas custom para express



- Utilizamos el método `app.engine(ext, callback)` para crear nuestro propio motor de plantilla. **ext** hace referencia a la extensión de archivo y **callback** es la función de motor de plantilla, que acepta como parámetros la ubicación del archivo, el objeto options y la función callback.
- El método `app.set('views', path)` especifica la carpeta de plantillas.
- El método `app.set('view engine', name)` registra el motor de plantillas.

Ejemplo

El siguiente código es un ejemplo de implementación de un motor de plantilla muy simple para la representación de archivos **.ntl**.

```
const fs = require('fs');
// defino el motor de plantilla
app.engine('ntl', function (filePath, options, callback) {
  fs.readFile(filePath, function (err, content) {
    if (err) {
      return callback(new Error(err));
    }
    const rendered = content.toString()
      .replace('#title#', '' + options.title + '')
      .replace('#message#', '' + options.message + '');
    return callback(null, rendered);
  });
});
app.set('views', './views'); // especifica el directorio de vistas
app.set('view engine', 'ntl'); // registra el motor de plantillas
```

Ejemplo

- La aplicación ahora podrá representar archivos **.ntl**. Creamos un archivo denominado **index.ntl** en el directorio views con el siguiente contenido:

```
#title#  
#message#
```

- A continuación, creamos la siguiente ruta en la aplicación:

```
app.get('/', function (req, res) {  
  res.render('index', { title: 'Hey', message: 'Hello there!' });  
});
```




Motor de plantillas custom

Crearemos nuestro propio motor de plantillas.

Tiempo: 10 minutos



Desarrollar un motor de plantillas custom para un servidor basado en express, que permita representar en la ruta '/cte1' el siguiente archivo de plantilla 'plantilla1.cte':

```
<h1>^^titulo$$</h1>
<p>^^mensaje$$</p>
<b>^^autor$$</b>
<hr>
<i><b>Versión: ^^version$$</b></i>
```

Con los datos que provienen desde un objeto:

```
{
  titulo: (algún título en string),
  mensaje: (algún mensaje en string),
  autor: (algún autor en string),
  version: (numerica)
}
```



Este motor personalizado debe permitir parsear objetos de datos con claves dinámicas y volcar sus valores en la plantilla seleccionada.

Crear otra ruta '/cte2' que represente otro archivo de plantilla: 'plantilla2.cte' con los datos nombre, apellido y la fecha/hora provenientes de un objeto.

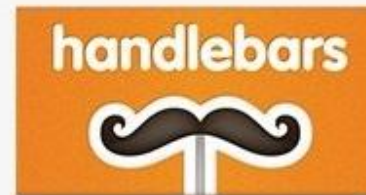
Handlebars en express



+

express

+



Introducción

- Handlebars puede funcionar de dos formas
 - desde el **lado del servidor**
 - desde el **lado del cliente**.



Esta **versatilidad** hace que podamos decidir mejor cómo queremos realizar nuestras aplicaciones, ya que si es una **SPA** tal vez el enfoque del **lado del cliente** sea más sencillo y útil, pero si queremos un **website** tal vez generar todo en el **servidor** sea más útil.

Instalar Handlebars del lado del Servidor

Para utilizar **Handlebars** del **lado del servidor** y así integrarlo con express ejecutamos el siguiente comando:

```
npm install express-handlebars
```

Cmder

```
C:\nodeHandlebars
```

```
λ npm install express-handlebars
```

```
[██████████].....] | idealTree: timing idealTree Completed in 71ms
```

Así npm va a descargar todos los componentes necesarios para que podamos incorporar este motor en nuestro proyecto.

Ejemplo handlebars en Node.js (código de servidor)

```
const express = require("express");
const app = express();
//cargamos el módulo handlebars
const handlebars = require("express-handlebars");

//establecemos la configuración de handlebars
app.engine(
  "hbs", //nombre referencia a la plantilla (se usa luego en set)
  handlebars({ //función de configuración handlebars
    extname: ".hbs", //extensión a utilizar (en lugar de .handlebars por defecto)
    defaultLayout: 'index.hbs', //plantilla principal
    layoutsDir: __dirname + "/views/layouts", //ruta a la plantilla principal
    partialsDir: __dirname + '/views/partials/' //ruta a las plantillas parciales
  })
);

//establecemos el motor de plantilla que se utiliza
app.set("view engine", "hbs");
//establecemos directorio donde se encuentran los archivos de plantilla
app.set("views", "./views");
//espacio público del servidor
app.use(express.static("public"));
```

```
▼ HB
  > node_modules
  ▼ public
    # style.css
  ▼ views
    ▼ layouts
      📄 index.hbs
    ▼ partials
      📄 hatedChamps.hbs
      📄 lovedChamps.hbs
      📄 main.hbs
  {} package-lock.json
  {} package.json
  JS server.js
```

Ejemplo handlebars en Node.js (código de servidor y vista)

```
fakeApi = () => [  
  { name: 'Katarina', lane: 'midlaner' },  
  { name: 'Jayce', lane: 'toplaner' },  
  { name: 'Heimerdinger', lane: 'toplaner' },  
  { name: 'Jayce', lane: 'midlaner' },  
  { name: 'Azir', lane: 'midlaner' }  
];  
  
app.get("/", (req, res) => {  
  //Sirve el cuerpo de la página "main.hbs" en el contenedor "index.hbs"  
  res.render("main", { suggestedChamps: fakeApi(), listExists: true });  
});  
  
const PORT = 8080  
  
app.listen(PORT, err => {  
  if(err) throw new Error(`Error en servidor ${err}`)  
  console.log(`El servidor express escuchando en el puerto ${PORT}`)  
})
```

```
{{!-- index.hbs --}}  
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="utf-8" />  
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />  
    <title>My Awesome App</title>  
    <meta name="viewport" content="width=device-width, initial-scale=1.0, viewport-fit=cover" />  
    <link rel="stylesheet" type="text/css" href="./style.css" />  
  </head>  
  <body>  
    {{{body}}}  
  </body>  
</html>
```

```
{{!-- main.hbs --}}  
{{> lovedChamps}}  
<br>  
{{> hatedChamps}}  
<br>  
  
<h2>Liste of my most suggested League champions</h2>  
{{!-- dependiendo del valor de la variable listExists --}}  
{{!-- si está es true entonces la lista será renderizada --}}  
{{!-- si está es false entonces la lista NO será renderizada --}}  
{{#if listExists}}  
  <ul>  
    {{#each suggestedChamps}}  
      <li class={{this.lane}}>{{this.name}}</li>  
    {{/each}}  
  </ul>  
{{/if}}
```


Ejemplo handlebars en Node.js (código de vista y salida)

```
{{!-- hatedChamps --}}  
<h2>Liste of my most hated League champions</h2>  
<ul>  
  <li class="midlaner">Yasuo</li>  
  <li class="midlaner">Zoe</li>  
  <li class="toplaner">Mundo</li>  
  <li class="toplaner">Darius</li>  
  <li class="midlaner">Fizz</li>  
</ul>
```

```
{{!-- lovedChamps --}}  
<h2>Liste of my most loved League champions</h2>  
<ul>  
  <li class="midlaner">Leblanc</li>  
  <li class="midlaner">Lux</li>  
  <li class="toplaner">Teemo</li>  
  <li class="midlaner">Kassadin</li>  
  <li class="toplaner">Jarvan IV</li>  
</ul>
```

← → ↻ localhost:8080

Liste of my most loved League champions

Leblanc

Lux

Teemo

Kassadin

Jarvan IV

Liste of my most hated League champions

Yasuo

Zoe

Mundo

Darius

Fizz

Liste of my most suggested League champions

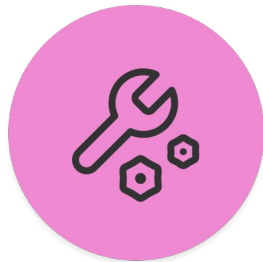
Katarina

Jayce

Heimerdinger

Zed

Azir



Handlebars con express

Tiempo: 10 minutos



- Transformar el primer desafío, pero esta vez la página dinámica la creará el servidor desde handlebars instalado y configurado para trabajar con express.
- Utilizar la misma estructura de plantilla HTML dentro de una pagina web con encabezado y el mismo objeto de datos.
- El servidor escuchará en el puerto 8080 y el resultado lo ofrecerá en su ruta root.

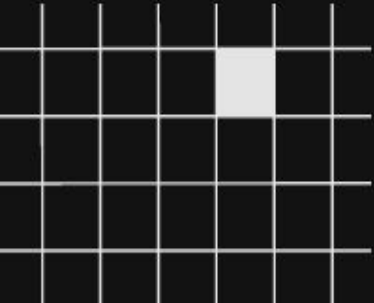
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Introducción a los motores de plantillas y su uso con Express
 - Motor de plantillas Handlebars
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE