

Desarrollo Backend

Bienvenid@s

Motores de plantillas
(*qué son - opciones disponibles - EJS*)

Clase 13

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO



Pon a grabar la clase



Temario

- Qué son los motores de plantillas
 - por qué utilizarlos
 - los motores más populares
- EJS
 - instalación y configuración
 - sintaxis
- Implementación
 - marcadores de posición
 - combinar HTML y JS
 - header y footer común a todas las plantillas
- Ventajas y desventajas



Qué son los motores de plantillas

Qué son los motores de plantillas

En esta unidad, nos alejaremos brevemente del trabajo con aplicaciones de backend netamente de servidores, para conocer lo que son motores de plantillas.

Estas herramientas están enfocadas en el desarrollo Frontend, pero implementando las mismas desde el backend. Veamos de qué se trata:



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Qué son los motores de plantillas

Los motores de plantillas son herramientas que permiten generar páginas web dinámicas mediante la combinación de plantillas predefinidas y datos que se obtienen de una base de datos o de otra fuente de datos.

En lugar de crear manualmente el HTML, los motores de plantillas permiten definir una plantilla con marcadores que indican dónde deben insertarse los datos y luego renderizar esa plantilla con los datos proporcionados.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Qué son los motores de plantillas

En Node.js, existen varios motores de plantillas que se pueden utilizar para generar HTML dinámico en el servidor.

Algunos de los más populares, son:

- EJS
- Handlebars
- Pug
- Mustache

Cada uno tiene sus propias ventajas y desventajas y, el mejor para utilizar, siempre dependerá de las necesidades puntuales del proyecto.



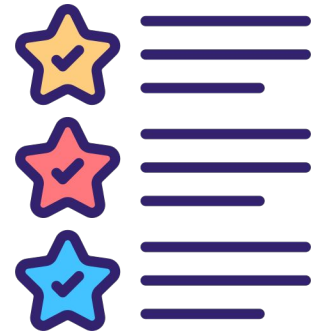
Por qué utilizarlos

Por qué utilizarlos

Algunas ventajas de implementar motores de plantillas, son:

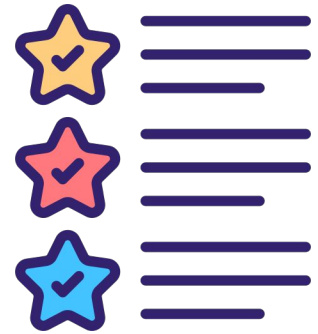
1. **Facilitan la creación de web dinámicas:** los motores de plantillas permiten combinar plantillas predefinidas con datos dinámicos para generar páginas web dinámicas. Esto hace que sea más fácil y rápido crear páginas web personalizadas y dinámicas.
2. **Separación de la lógica de presentación de la del negocio:** al utilizar un motor de plantillas, se puede separar la lógica de presentación (*el HTML y la estructura de la página*) de la lógica de negocio (*el código JavaScript que maneja los datos y las interacciones*). Esto hace que sea más fácil mantener y modificar el código en el futuro.

...



Por qué utilizarlos

3. **Reutilización de plantillas:** los motores de plantillas permiten definir plantillas reutilizables que se pueden utilizar en diferentes páginas web. Esto hace que sea más fácil y rápido crear páginas web consistentes y con un aspecto similar.
4. **Mayor eficiencia:** los motores de plantillas suelen tener un rendimiento mejor que la generación manual de HTML, ya que utilizan técnicas de caché y compilación para generar HTML de manera más rápida y eficiente.
5. **Compatibilidad con diferentes sistemas:** los motores de plantillas suelen ser compatibles con diferentes sistemas y frameworks, lo que permite utilizarlos en proyectos de Node.js, Express, Meteor, Ruby on Rails, Django, entre otros.



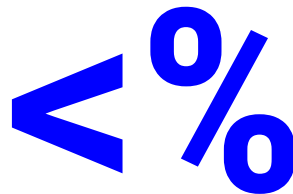
Los motores más populares

Los motores más populares

EJS

Embedded JavaScript: EJS es un motor de plantillas que permite utilizar JavaScript en las plantillas para crear HTML dinámico.

Las plantillas de EJS se parecen a las páginas HTML normales, pero incluyen código JavaScript entre etiquetas `<% %>` para acceder a los datos.



Los motores más populares

Handlebars

Handlebars es otro motor de plantillas que utiliza una sintaxis similar a la de EJS.

Sin embargo, a diferencia de EJS, Handlebars utiliza una sintaxis más sencilla y más fácil de leer, utilizando etiquetas `{{ }}` para acceder a los datos.



Los motores más populares

Pug

Pug es un motor de plantillas que utiliza una sintaxis especial que se parece más a un lenguaje de programación que a HTML.

Pug utiliza sangrías y bloques para definir la estructura de la página y accede a los datos mediante variables.



Los motores más populares

Mustache

Mustache es un motor de plantillas que es muy sencillo y fácil de usar.

Utiliza etiquetas dobles mustache brackets `{{ }}` para acceder a los datos, aunque no admite la inclusión de código JavaScript en las plantillas.



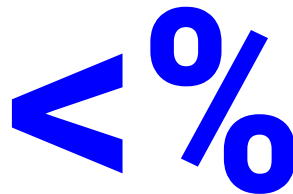
EJS

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Los motores más populares

EJS es el motor de plantillas elegido para entender cómo funciona esta tecnología.



La sintaxis de EJS es similar a la del HTML, lo que **hace que sea fácil de entender y utilizar**. EJS **permite definir plantillas** que contienen **marcadores de posición** para datos dinámicos, así como **lógica de programación en JavaScript**.

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Instalación y Configuración

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Instalación y configuración

Iniciaremos un proyecto de ejemplo, a partir de un nuevo proyecto **Node.js**.

Crearemos una carpeta llamada **Motor de plantillas**, abrimos la misma con Visual Studio Code, e iniciaremos el proyecto con `npm init -y` para crear `package.json`.



Instalación y configuración

Luego instalaremos Express JS más el resto de las dependencias necesarias, utilizando el comando **`npm install express path dotenv`**, y configuraremos **`package.json`** con los datos básicos de la aplicación más el archivo **`.env`**.

Recordemos modificar la sección **`scripts`** para definir el inicio de nuestro archivo JavaScript a través de la línea de comandos, incluyendo el parámetro **`--watch`**.

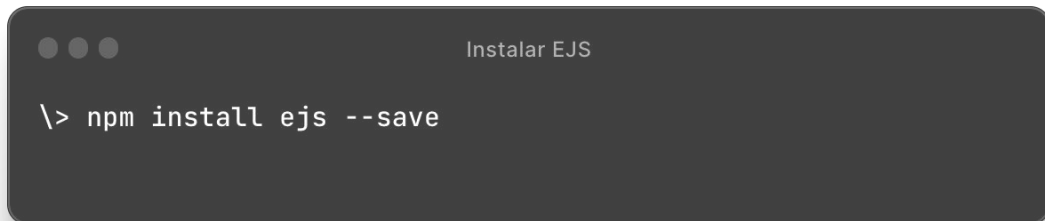


UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Instalación y configuración

Para utilizar EJS en un proyecto de Node.js, también debemos instalarlo a través de NPM. En la ventana o panel **Terminal**, utilizaremos el comando apropiado para instalarlo:

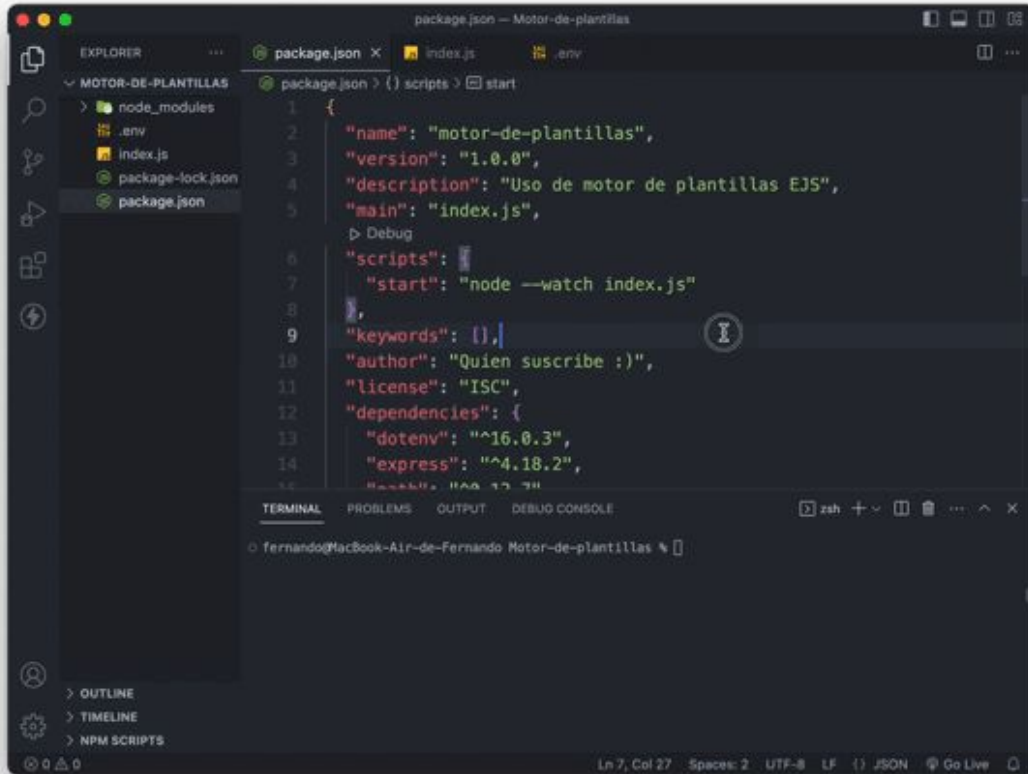
A dark-themed terminal window with three small circles in the top-left corner. The title bar in the top-right corner reads "Instalar EJS". The command prompt shows the command `\> npm install ejs --save` entered.

```
Instalar EJS

\> npm install ejs --save
```

La opción **--save** le indica a NPM que incluya el paquete en la sección de **dependencies** de **package.json**, ahorrándonos un paso adicional.

Instalación y configuración



```
package.json — Motor-de-plantillas
1 {
2   "name": "motor-de-plantillas",
3   "version": "1.0.0",
4   "description": "Uso de motor de plantillas EJS",
5   "main": "index.js",
6   "scripts": {
7     "start": "node --watch index.js"
8   },
9   "keywords": [],
10  "author": "Quien suscribe :)",
11  "license": "ISC",
12  "dependencies": {
13    "dotenv": "^16.0.3",
14    "express": "^4.18.2",
15    "nodemon": "^3.1.0"
16  }
17 }
```

Verifiquemos la correcta configuración de dependencias utilizadas, para ya poder configurar EJS en nuestro proyecto Node.js.

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Instalación y configuración

Con el archivo **index.js** ya creado y configurado, debemos inicializar el Motor de plantillas elegido. Para ello, debemos configurarlo utilizando el método **.set()** de nuestro servidor Express.

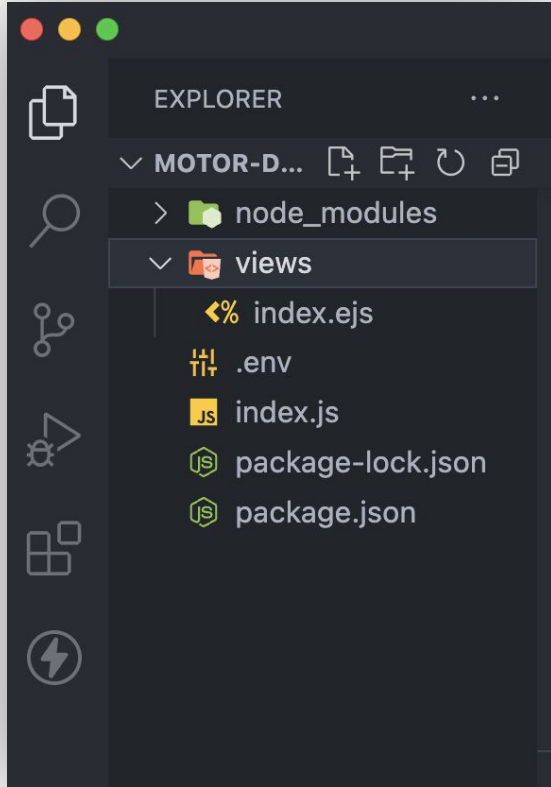
Configurar EJS

```
app.set('view engine', 'ejs');  
app.use(express.static('views'));
```

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

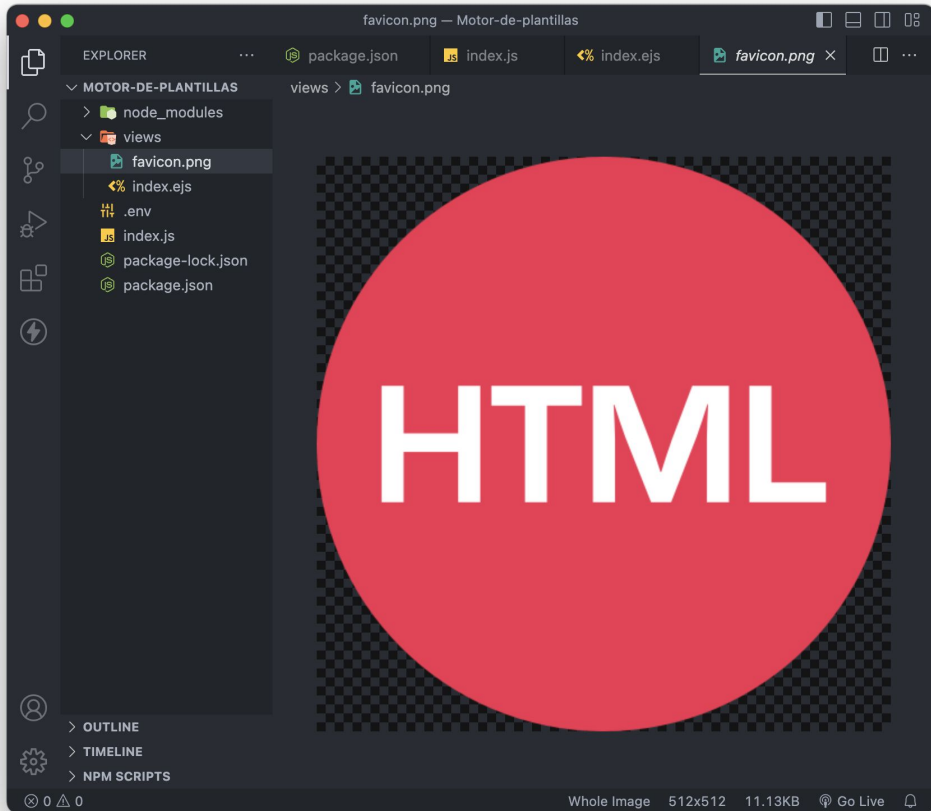
Instalación y configuración



Vamos a servir un sitio estático, tal como lo indica el método **`express.static()`**. Debemos crear una subcarpeta en nuestro proyecto llamada **views**.

Dentro de esta crearemos el archivo HTML, pero con extensión EJS (**`index.ejs`**).

Instalación y configuración



Para tener una mejor experiencia en el renderizado, más cercana al desarrollo web frontend, sumemos un ícono que cumpla el rol de **favicon**.

Con esto, estamos listos para comenzar a armar nuestro **documento HTML - EJS**.

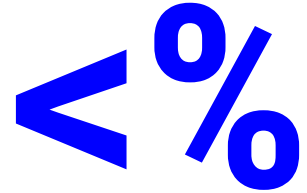
Sintaxis

Sintaxis

Veamos entonces, cómo es la estructura de un archivo .EJS.

Tal como dijimos anteriormente, las plantillas de EJS se parecen a las páginas HTML normales, pero incluyen código JavaScript entre etiquetas `<%%>` para acceder a los datos.

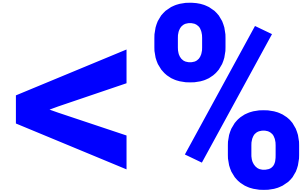
Para tener un punto de comparación, estas etiquetas especiales se comportan de forma similar al uso de **Template Literals** en JavaScript moderno: `${variable}`.



Sintaxis

La sintaxis de EJS, se divide en dos partes:

- Uso de marcadores de posición para la inserción de datos
- Combinación de HTML y JavaScript en las plantillas de EJS



En sí, EJS toma en parte la filosofía que podemos encontrar en Librerías o Frameworks Frontend, pero de una forma más ligera. Además, nos ayudan a simplificar el HTML en gran medida, logrando así una performance efectiva para cargar documentos HTML resultantes en muy buen tiempo de respuesta.

Sintaxis

Aquí tenemos un ejemplo básico de la sintaxis de EJS.

Esta estructura va definida en el archivo EJS, y podemos ver que en la misma se intercalan los tags HTML, y dentro de las etiquetas EJS definimos una variable o propiedad JS, la cual **desplegará dentro del nodo texto del documento HTML, el valor que tenga** asignado.



Sintaxis

```
<h1><%= title %></h1>  
<p><%= message %></p>
```

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Sintaxis

Si contamos con un array de objetos el cual, por ejemplo, tiene un listado de productos y queremos visualizar el mismo en el documento HTML, EJS nos permite agregar la estructura de etiquetas necesarias para poder iterar dicho array mediante un ciclo **for convencional**.

```
Sintaxis

<ul>
  <% for (let i = 0; i < products.length; i++) { %>
    <li><%= products[i].name %></li>
  <% } %>
</ul>
```

Sintaxis

Todos los parámetros que le indicaremos a la plantilla EJS, deben estar generados dentro de una constante denominada **data**, en formato objeto literal. Desde allí el Motor de plantillas se ocupará de armar la estructura renderizando correctamente el contenido.

```
Sintaxis

const data = {
  title: 'Mi sitio web con EJS',
  message: 'Bienvenido a mi sitio web generado a partir de
un motor de plantillas.',
  products: [{ name: 'Notebook Lenovo', price: 720 },
              { name: 'Macbook Air 13', price: 1250 },
              { name: 'Tablet Droid 10.1', price: 350 }]
};
```

Implementación

Implementación

Veamos entonces, cómo es la estructura de un archivo **.EJS** implementando los marcadores de posición. Abrimos el archivo en cuestión y definimos los tags HTML básicos correspondientes al **<head>** del documento HTML.

```
index.ejs x
views > index.ejs > ul
1 <title>Mi motor de plantillas EJS</title>
2 <link rel="shortcut icon" href="/favicon.png" />
3 <meta charset="utf-8">
4
```

Implementación

Seguido a dicha estructura, definimos los tags HTML correspondientes a **<body>**. En cada tag, intercalamos los marcadores de posición de acuerdo a lo que hayamos definido en el objeto **data**.

```
index.ejs x
views > index.ejs > link
5   <div class="container">
6     <h1><%= title %></h1>
7     <p><%= message %></p>
8     <button class="button">ver productos</button>
9   </div>
10  <footer>
11    <div class="footer">
12      <p><b>Copyright 2023 - Quien lo programa</b> 🤖 </p>
13    </div>
14  </footer>
```

Implementación

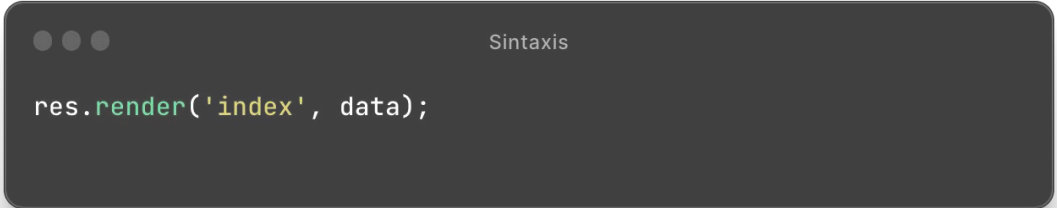
En el archivo **index.js**, armamos la estructura de código para escuchar una petición en el raíz del sitio web. Allí definimos la estructura de la constante **data**, y definimos cada propiedad con su valor, correspondiente a cada marcador de posición a usar en la plantilla EJS.

```
Sintaxis

app.get('/', (req, res) => {
  const data = {
    title: 'Mi sitio web con EJS',
    message: 'Bienvenido a mi sitio web generado a partir de
un motor de plantillas.',
    products: [{ name: 'Notebook Lenovo', price: 720 },
               { name: 'Macbook Air 13', price: 1250 },
               { name: 'Tablet Droid 10.1', price: 350 }]
  };
});
```

Implementación

Antes del cierre del método `.get()`, agregamos el método `render()` contenido dentro del objeto `response (res)`. Este método espera como primer parámetro, el archivo EJS que debe renderizar, y como segundo parámetro, el objeto data que creamos con la estructura correspondiente a los marcadores de posición.



```
res.render('index', data);
```

Implementación

```
Sintaxis

const express = require('express');
const app = express();
const path = require('path');
const dotenv = require('dotenv');
    dotenv.config();
const PORT = process.env.PORT || 3000

app.set('view engine', 'ejs');
app.use(express.static('views'));

app.get('/', (req, res) => {
    const data = {
        title: 'Mi sitio web con EJS',
        message: 'Bienvenido a mi sitio web generado a partir de
un motor de plantillas.',
        products: [{ name: 'Notebook Lenovo', price: 720 },
                    { name: 'Macbook Air 13', price: 1250 },
                    { name: 'Tablet Droid 10.1', price: 350 }]
    };
    res.render('index', data);
});
```

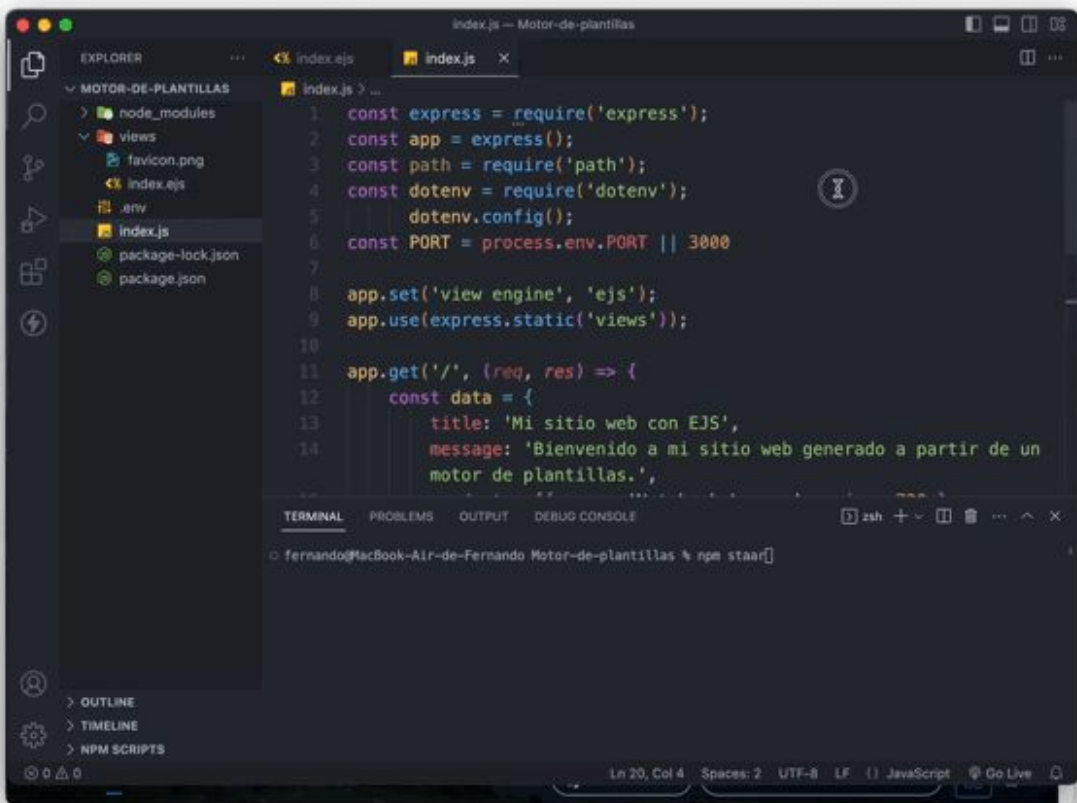
Aquí, el código completo del ejemplo que venimos trabajando.

Solo nos queda ejecutar el proyecto para que se inicie el servidor web, y probar el mismo en una pestaña de un web browser.

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Implementación



```
1 const express = require('express');
2 const app = express();
3 const path = require('path');
4 const dotenv = require('dotenv');
5   dotenv.config();
6 const PORT = process.env.PORT || 3000
7
8 app.set('view engine', 'ejs');
9 app.use(express.static('views'));
10
11 app.get('/', (req, res) => {
12   const data = {
13     title: 'Mi sitio web con EJS',
14     message: 'Bienvenido a mi sitio web generado a partir de un
    motor de plantillas.'
```

Verificamos que todo esté OK, y ejecutamos **npm start** en la Terminal, para poner a correr el proyecto.

Debemos encontrarnos con un documento HTML y los tags correctamente referenciados, además de los tags **** dinámicos.

Estilizar el HTML

Estilizar el HTML

La plantilla EJS puede ser totalmente estilizada con CSS puro e incluso con framework CSS como Bootstrap o derivados.

Veamos un pequeño ejemplo de cómo mejorar la estética de la plantilla integrando de forma rápida el microframework CSS [Milligram](#). Junto a este, agregamos también la **fente web Roboto**, para cambiar casi sin esfuerzo la estructura HTML.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Estilizar el HTML

La plantilla EJS puede ser totalmente estilizada con CSS puro e incluso con una librería CSS como Bootstrap o derivados.

Sintaxis

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css?  
family=Roboto:300,300italic,700,700italic">  
<link rel="stylesheet"  
href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/milligram.min.css">
```

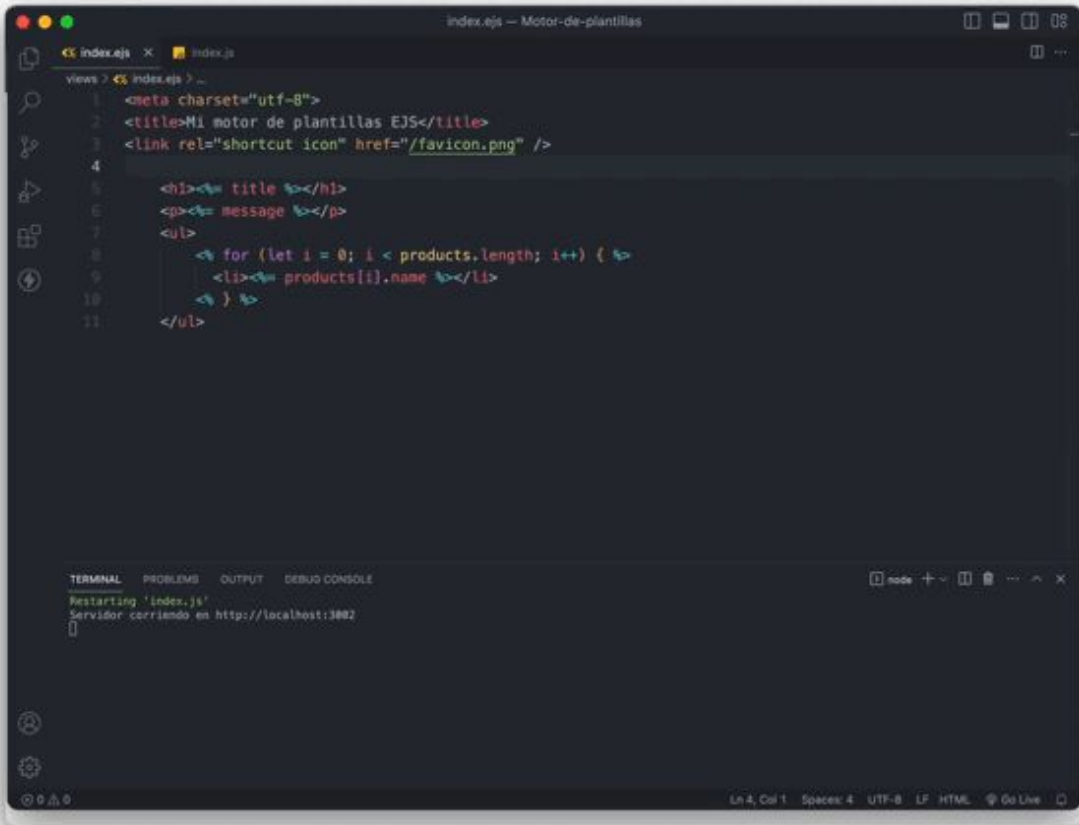
Estilizar el HTML

Referenciamos los **<link>** en el apartado correspondiente el encabezado del documento EJS, tal como lo haríamos en un documento HTML convencional.

A screenshot of a code editor with a dark theme. The top bar shows two tabs: 'index.ejs' (active) and 'index.js'. Below the tabs, the breadcrumb 'views > index.ejs > ...' is visible. The code is as follows:

```
1 <meta charset="utf-8">
2 <title>Mi motor de plantillas EJS</title>
3 <link rel="shortcut icon" href="/favicon.png" />
4 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,300italic,700,700italic">
5 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/milligram.min.css">
6
```

Estilizar el HTML



```
index.ejs — Motor-de-plantillas
index.ejs
1 <meta charset="utf-8">
2 <title>Mi motor de plantillas EJS</title>
3 <link rel="shortcut icon" href="/favicon.png" />
4
5 <h1><%= title %></h1>
6 <p><%= message %></p>
7 <ul>
8   <%= for (let i = 0; i < products.length; i++) { %>
9     <li><%= products[i].name %></li>
10    <%= } %>
11 </ul>

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Restarting 'index.js'
Servidor corriendo en http://localhost:3002
```

Así de fácil podemos cambiar la estética de un documento EJS. Incluso podemos referenciar un archivo CSS externo, de la misma forma que referenciamos este microframework CSS.

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Estilizar el HTML

```
7 <div class="container">
8   <h1><%= title %></h1>
9   <p><%= message %></p>
10  <ul>
11    <% for (let i = 0; i < products.length; i++) { %>
12      <li><%= products[i].name %></li>
13    <% } %>
14  </ul>
15 </div>
```

También, cualquier clase CSS que contenga el framework homónimo, puede ser referenciada dentro de la estructura **<body>** del archivo EJS, por supuesto que luego de referenciar la hoja de estilos.

Marcadores de posición y comandos específicos

Marcadores de posición y comandos específicos

Veamos a continuación los diferentes marcadores de posición con los que cuenta EJS.

Marcador	Descripción
<code><%= %></code>	Este marcador de posición se utiliza para insertar contenido en la plantilla. Cualquier expresión JavaScript válida se puede utilizar dentro de los marcadores, y su resultado se mostrará en la plantilla. Por ejemplo, si tienes una variable <code>title</code> con el valor <code>"Mi sitio web"</code> , puedes insertarla en la plantilla con <code><%= title %></code> .
<code><%- %></code>	Este marcador de posición se utiliza para insertar contenido sin escapar en la plantilla. Esto significa que cualquier HTML o JavaScript incluido en los marcadores se mostrará tal cual, sin ser procesado por el navegador. Este marcador de posición es útil si necesitas insertar código HTML o JavaScript generado dinámicamente en la plantilla.
<code><% %></code>	Este marcador de posición se utiliza para incluir código JavaScript en la plantilla. El código dentro de los marcadores se ejecutará cuando se procese la plantilla, pero no se mostrará en la salida final. Esto te permite realizar operaciones o lógica más complejas en la plantilla, como ciclos, condiciones, etc.

Marcadores de posición y comandos específicos

Para controlar el flujo de ejecución y generar contenido dinámicamente, EJS contiene una serie de comandos específicos que nos facilitan estas tareas.

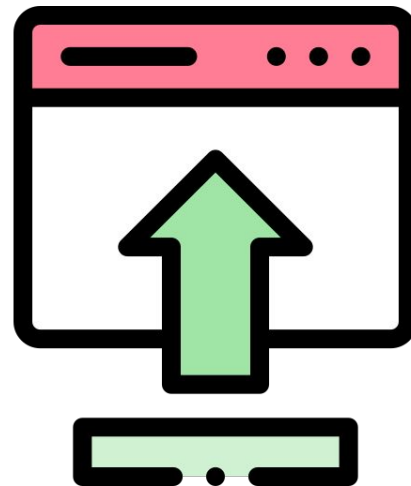
Descripción	
<code><% if (condition) { %> ... <% } %></code>	Permite ejecutar un bloque de código si se cumple una condición determinada. Puedes utilizar cualquier expresión JS como condición.
<code><% for (var i = 0; i < items.length; i++) { %> ... <% } %></code>	Permite ejecutar un bloque de código varias veces, en función del número de elementos en un arreglo o variable.
<code><% while (condition) { %> ... <% } %></code>	Permite ejecutar un bloque de código mientras se cumpla una condición determinada, usando cualquier expresión JavaScript como condición.
<code><% switch (variable) { case 'value': %> ... <% break; } %></code>	Permite ejecutar un bloque de código en función del valor de una variable, sobre una estructura switch similar a la de JS.
<code><%- include('filename') %></code>	Permite incluir otro archivo EJS dentro de la plantilla actual. Puedes utilizar esto para reutilizar bloques de código en varias plantillas.

header y footer, común a
todas las plantillas

header y footer común a todas las plantillas

Nos quedamos con la explicación de la última diapositiva, tomando como referencia la función **include()**.

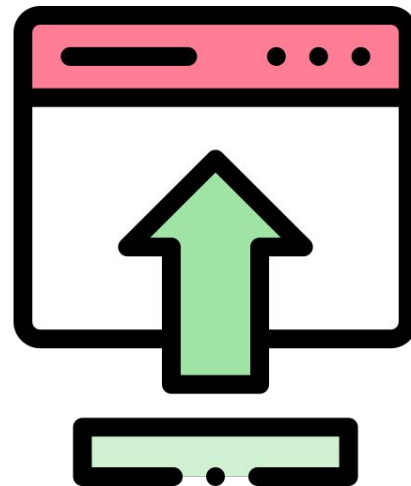
La misma tiene un papel importantísimo en EJS, ya que nos permite generar un encabezado **<head>** y un pie de página **<footer>**, común a todas las plantillas EJS.



header y footer común a todas las plantillas

De esta forma, definimos un archivo **head** y un archivo **footer**, ambos con el contenido HTML predeterminado, común a todas las páginas del sitio o aplicación web.

Luego, nos queda crear todas las plantillas EJS con la estructura correspondiente al **<body>** de cada documento, y agregar la función **include()**.



header y footer común a todas las plantillas

Definimos una plantilla EJS con el nombre **head.ejs**. Dentro de la misma, agregamos el contenido HTML correspondiente al encabezado del documento homónimo.

```
head EJS

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mi sitio web</title>
    <link rel="stylesheet" href="/ruta/a/tu/hoja/de/estilos.css">
    <script src="/ruta/a/tu/archivo/de/scripts.js"></script>
  </head>
  <body>
```

header y footer común a todas las plantillas

Luego hacemos lo propio con **footer.ejs**. En este caso, solo insertamos el contenido del pie de página, ya que el cierre del tag **<body>** podemos definirlo por cada documento **'cuerpo'** creado con EJS.

```
footer EJS

<div class="container">
  <p><b>Copyright 2023 - Quien suscribe programando</b> 🧐 </p>
</div>
```

header y footer común a todas las plantillas

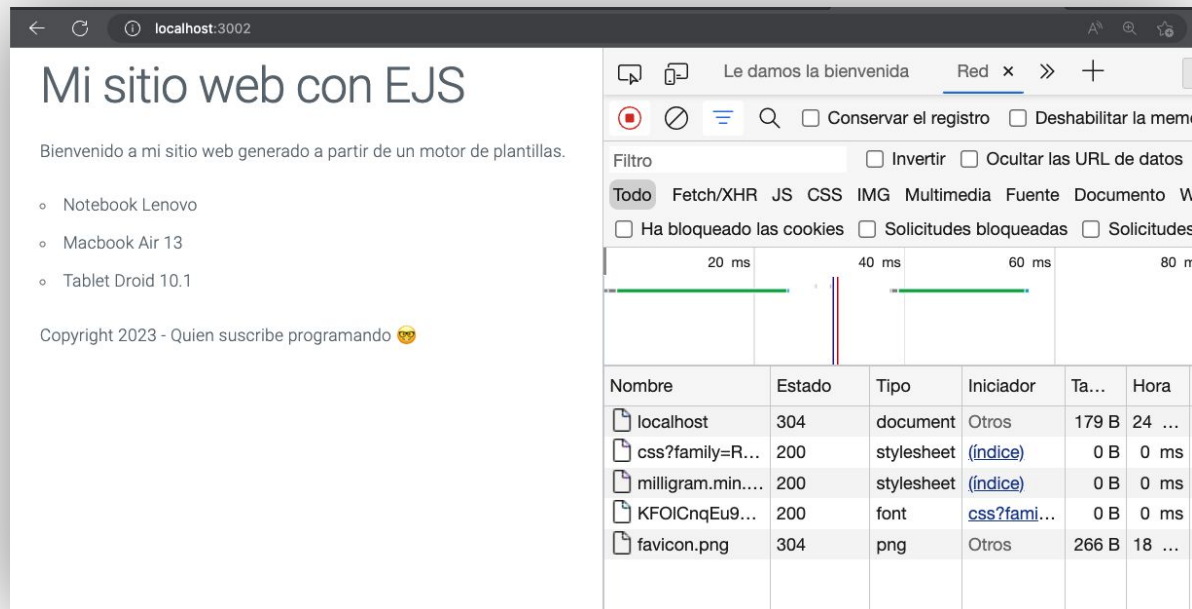
```
file EJS

<%- include('head') %>
  <div class="container">
    <h1><%= title %></h1>
    <p><%= message %></p>
    <ul>
      <% for (let i = 0; i < products.length; i++) { %>
        <li><%= products[i].name %></li>
      <% } %>
    </ul>
  </div>
<%- include('footer') %>
</body>
</html>
```

Finalmente, agregamos la función **include()** donde corresponde, para referenciar en esta los archivos del **encabezado** y del **pié de página**.

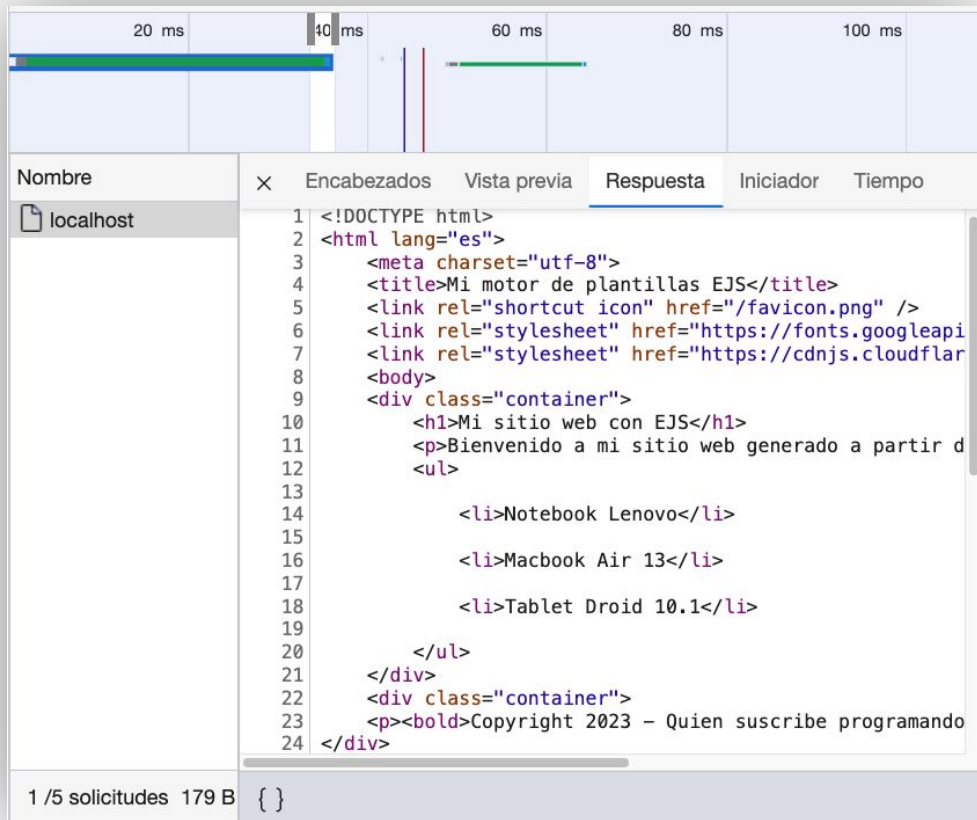
Con esto realizado, podemos volver a probar nuestro proyecto.

header y footer común a todas las plantillas



Analizamos el resultado con **DevTools > Network** (red), y así podemos comprobar que, todo recurso (*local* y *remoto*), se carga correctamente.

header y footer común a todas las plantillas



Y, si también verificamos con DevTools documento descargado en el navegador web, comprobaremos que su estructura HTML está generada como si fuese un documento único.

Ventajas y desventajas

Ventajas de uso de un motor de plantillas

Como podemos apreciar, EJS es una herramienta poderosa para implementar en el desarrollo de aplicaciones backend, y, si debemos enumerar algunas de sus ventajas, destacamos:

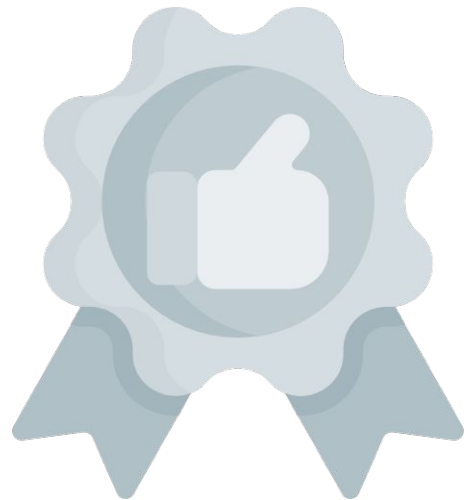
- Separación de responsabilidades
- Reutilización de código
- Flexibilidad
- Mayor velocidad
- Facilidad de mantenimiento



Desventajas de uso de un motor de plantillas

Aunque los motores de plantillas tienen muchas ventajas, también hay algunas desventajas a tener en cuenta:

- Curva de aprendizaje
- Sobrecarga
- Tamaño del paquete
- Seguridad

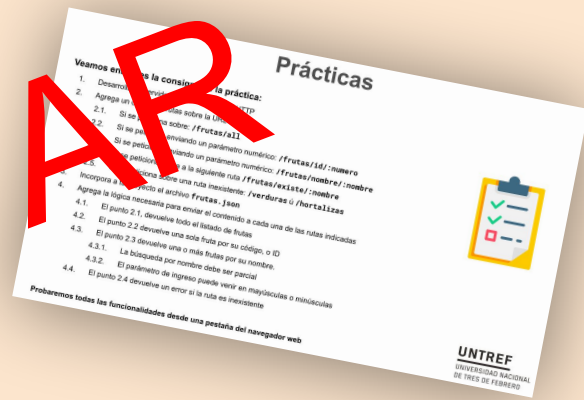


Espacio de trabajo

Espacio de trabajo

Recrea el TP realizado en la clase 10, migrando su estructura de servidor web a Express JS. La funcionalidad deberá ser la misma que la solicitada en el trabajo práctico.

Actualiza el control de errores de este servidor web para las rutas inexistentes, de acuerdo a las opciones que vimos en esta unidad. Implementa la que más te parezca acorde.



Tiempo estimado: 30 minutos.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

```
const questions = [ 'dudas', 'consultas', '🤔' ]
```



```
> node gracias.js
```

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO