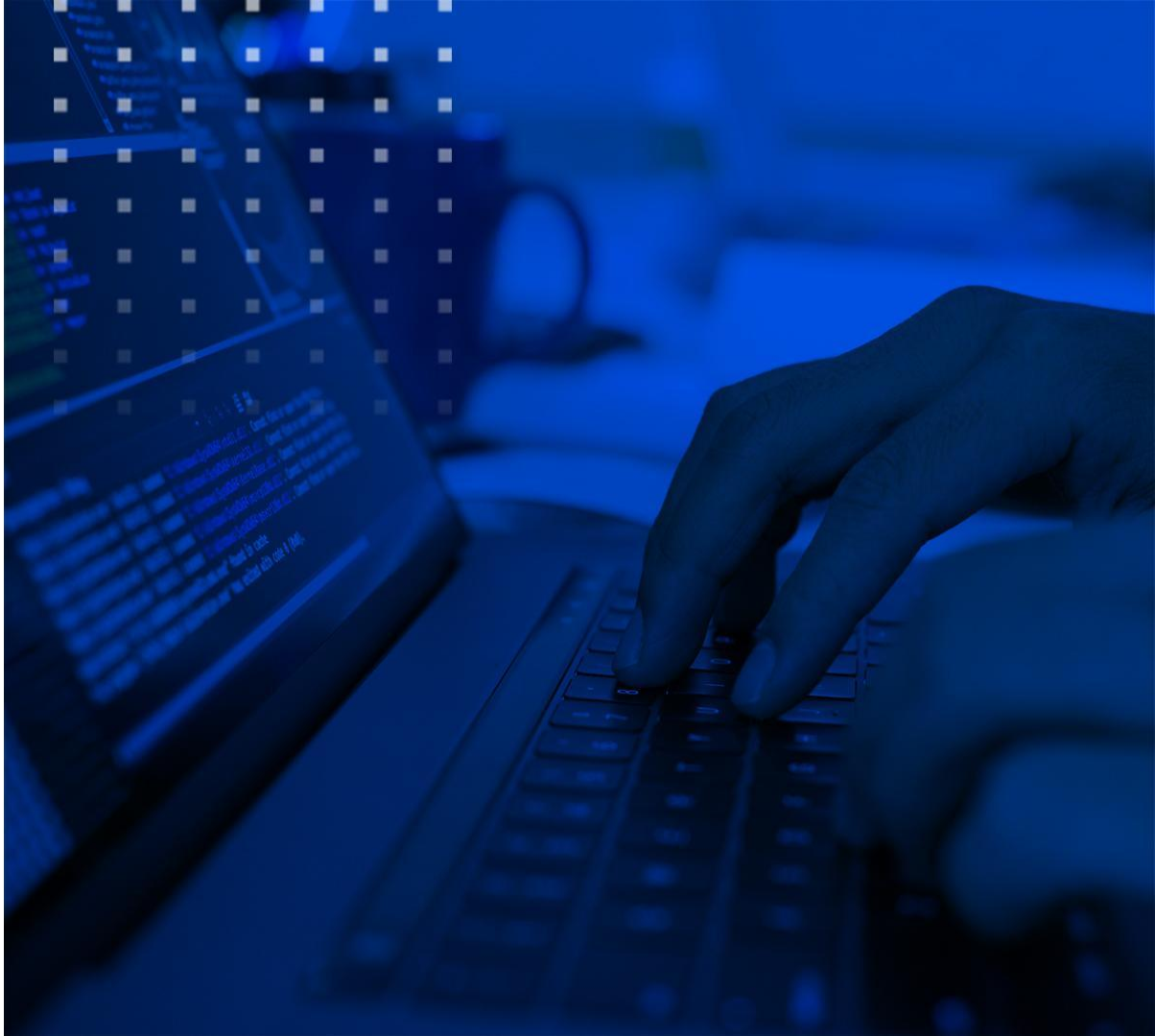
[i]ncluyeme.com

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

CURSO

FULL STACK DEVELOPER
NIVEL INICIAL



UNIDAD 14
Node.js: Express y Handlebars

EXPRESS - INTRODUCCIÓN

EXPRESS

Es el framework web más popular de Node, y es la librería

Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la ejecución de una petición.

LIBRERÍAS

A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado librerías para trabajar con:

- Envio de mail: nodemailer
- Sesiones: express-session
- Base de datos: mysql

NODEMON

- Es una herramienta que monitorea los cambios en el código fuente que se está desarrollando y automáticamente reinicia el servidor.

- Es una herramienta muy útil para desarrollo de aplicaciones

Instalación: https://www.npmjs.com/package/nodemon

INICIAR UN PROYECTO

1. Ejecutar: npm init -y

2. Instalar: npm i express
   - https://www.npmjs.com/package/express

3. Creamos punto de acceso *app.js*

4. Ir a : localhost:3000

RUTAS

En sitios web o aplicaciones web dinámicas, que accedan a bases de datos, el servidor espera recibir peticiones HTTP del navegador (o cliente).

Cuando se recibe una petición, la aplicación determina cuál es la acción adecuada correspondiente, de acuerdo a la estructura de la URL y a la información (opcional) indicada en la petición con los métodos POST o GET.

Dependiendo de la acción a realizar, puede que se necesite leer o escribir en la base de datos, o realizar otras acciones necesarias para atender la petición correctamente.

MÉTODOS

Express posee métodos para especificar qué función ha de ser llamada dependiendo del verbo HTTP usado en la petición (GET, POST, DELETE, etc.) y la estructura de la URL ("ruta").

También tiene los métodos para especificar qué plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y cómo generar la visualización adecuada para cada caso.

DEFINICIÓN

La definición de ruta tiene la siguiente estructura:

app.METHOD(PATH, HANDLER)

Donde:

- **app** es una instancia de express.
- **METHOD** es un método de solicitud HTTP.
- **PATH** es una vía de acceso en el servidor.
- **HANDLER** es la función que se ejecuta cuando se correlaciona la ruta.

RUTAS

Los siguientes ejemplos ilustran las definiciones de rutas simples.

```
const express = require("express");
const app = express();

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

app.get("/contacto", (req, res) => {
 res.send("Contacto");
});
```

Las primeras dos líneas incluyen (mediante la orden **requiere()**) el módulo de Express y crean una aplicación de Express.

Este elemento se denomina comúnmente **app**, y posee métodos para el enrutamiento de las peticiones HTTP, configuración del 'middleware', y visualización de las vistas de HTML, uso del motores de 'templates', y gestión de las configuraciones de las aplicaciones que controlan la aplicación.

Las líneas que siguen en el código (las tres líneas que comienzan con app.get) muestran una definición de ruta que se llamará cuando se reciba una petición HTTP GET con una dirección ('/') relativa al directorio raíz.

La función 'callback' toma una petición y una respuesta como argumentos, y ejecuta un send() en la respuesta, para enviar la cadena de caracteres: "Hola Mundo!".

¡A Practicar!

1. Iniciá un proyecto con express.
2. Definí 4 rutas:

- una para la home (/),
- 2 a elección y
- una 404 (*).

MOSTRAR CONTENIDO ESTÁTICO

La función express.static () es una función de middleware incorporada en Express para entregar archivos estáticos.

```
app.use(express.static("public"));
```

¡A Practicar!

1. Creá una carpeta *public*.
2. Creá 2 páginas html, una de ellas index.html con su correspondiente contenido css.
3. Definí las rutas correspondientes. Recordá utilizar.

```
app.use(express.static("public"));
```

HANDLEBARS

HANDLEBARS

- Es un sistema de plantillas Javascript basado en Mustache Templates.
- Mantiene separados el código y la vista.
- Permite generar HTML a partir de objetos con datos en formato JSON.

INSTALACIÓN

Adaptador para express:

https://github.com/pillarjs/hbs

$ npm install hbs

USO

El uso de hbs como motor de visualización predeterminado requiere solo una línea de código en la configuración de la aplicación. Esto generará archivos .hbs cuando se llame a **res.render.**

```
const hbs = require('hbs');
```

res.render ()

La función res.render () se usa para renderizar una vista y envía la cadena HTML renderizada al cliente.

```
app.get('/', (req, res)=>{
 res.render('home');
})
```

ARGUMENTOS DESDE EL CONTROLADOR

En el controlador:

```
app.get('/', (req, res)=>{
 res.render('home',{
   nombre: 'Cosme Fulanito',
   titulo:'Travel Coffe'
 });
})
```

En la vista:

```
        <title>{{titulo}}</title>
```

PARTIALS

Crear parciales nos permite reutilizar código.

```
//Handlebars
app.set('view engine', 'hbs');
hbs.registerPartials( __dirname + '/views/partials');
```

En las vistas:

```
  <!-- Header -->
        {{> header }}
```

¡A Practicar!

1. Establecé dos páginas con sus respectivas vistas utilizando **res.render()** y con los mismos parámetros.
   -
2. Creá un parcial llamado footer.
   -
3. Luego, implementá los parciales en las dos páginas. Recordá utilizar **{{ }}**


[i]ncluyeme
.com


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES