



Buenos
Aires
Ciudad

Agencia de Habilidades
para el Futuro

<Talento
Tech/>

Node.JS

Clase N° 9: Creando un Servidor Web

¡Les damos la bienvenida!

 Vamos a comenzar a grabar la clase.

Índice



1



2

Creando un Servidor Web

- Servidor Web Node Nativo
- Express JS
- Middlewares
- Subiendo el proyecto a la nube
 - GIT
 - GITHUB

Modelando una API Rest

- API Rest
- Estructura de archivos
- Capas de la aplicación
- División de responsabilidades

Objetivos de la Clase

Bienvenidos a nuestra presentación sobre la creación de servidores web. Exploraremos Node.js nativo, Express.js y middlewares para desarrollar aplicaciones web robustas y eficientes.

1 Servidor Node Nativo

Configurar y ejecutar un servidor básico en Node.js sin librerías externas.

2 Express.js

Instalar, configurar y utilizar Express.js para crear servidores web.

3 Middlewares

Introducir el concepto de middlewares en Express.js y su importancia.

Servidor Web Node Nativo

Introducción a Node.js

Servidor Web Node Nativo



1

Importar módulo http

Utilizar el módulo http nativo de Node.js para manejar solicitudes HTTP.

2

Crear el servidor

Usar `createServer` para configurar la respuesta del servidor.

3

Escuchar en un puerto

Configurar el servidor para escuchar en un puerto específico.

Código del Servidor Node Nativo

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

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola, mundo!');
});

server.listen(3000, () => {
  console.log(`Servidor corriendo en http://localhost:3000`);
});
```

Express JS

Introducción a Express.js



Express.js es un framework flexible para Node.js que simplifica la creación de servidores web.

Eficiencia

Manejo eficiente de rutas y middlewares para optimizar el desarrollo backend.

Instalación

Se instala fácilmente con **npm install express** en el proyecto.

Creando un Servidor con Express

Express 

Importar Express

```
import express from 'express';
```

1

2

3

Crear Instancia

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

Definir Rutas

```
app.get('/', (req, res) => {  
  res.send('Hola, mundo desde')
```

```
Express!');});
```

4

Escuchar Puerto

```
app.listen(3000, () => {  
  console.log('Servidor en  
  http://localhost:3000');});
```

Ventajas de Express.js





Simplicidad

API sencilla para manejar rutas y solicitudes HTTP.



Flexibilidad

Fácil integración de middlewares y otras librerías.



Ecosistema

Amplia comunidad y múltiples extensiones disponibles.

Express Generator

Express Generator

Express Generator es una herramienta que permite crear la estructura inicial de un proyecto de manera automática y estandarizada. Su **objetivo principal** es agilizar el proceso de configuración inicial al generar los archivos y directorios básicos necesarios para un proyecto de Express.

¿Para qué sirve?

Express Generator simplifica la creación de aplicaciones al proporcionar una base sólida con las configuraciones iniciales más comunes.

Incluye:

- **Estructura de carpetas predefinida:** Organiza los archivos del proyecto en carpetas como `routes`, `views` y `public`.
- **Plantillas integradas:** Ofrece compatibilidad con motores de plantillas como Pug, lo que facilita la generación de vistas dinámicas.
- **Archivos de configuración inicial:** Configura módulos y middlewares esenciales desde el inicio.

Instalación globalmente

```
npm install -g express-generator
```

Creación de un proyecto con Express Generator

- 1) Para generar un nuevo proyecto, ejecuta el siguiente comando:

```
express nombre-del-proyecto
```

- 2) Esto creará una nueva carpeta con la estructura inicial del proyecto.

```
cd nombre-del-proyecto
```

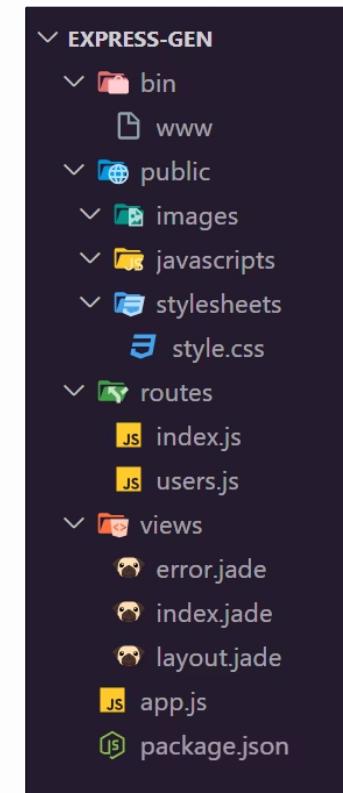
- 3) Una vez generado, navega al directorio del proyecto e instala las dependencias necesarias:

```
npm install
```

- 4) Finalmente, ejecuta el proyecto con:

```
npm start
```

Como se observa en la imagen, esta herramienta genera un entorno de trabajo con Express, listo para empezar a desarrollar.



Servidor de Archivos Estáticos

Configuración

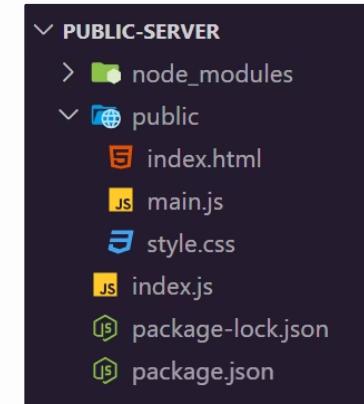
Usar express.static para servir archivos desde la carpeta 'public'.

Estructura

Organizar HTML, CSS, JavaScript e imágenes en la carpeta 'public'.

Paso a paso para crear un servidor de archivos estáticos

1. Crea la carpeta **public**: Dentro de tu proyecto, crea un directorio llamado public. Aquí es donde alojarás los archivos estáticos.



2. Una vez realizada la configuración básica, en el archivo index.js usamos el middleware **express.static**: que se utiliza para servir archivos estáticos. Es importante que apunte al directorio public por lo que nos ayudamos del módulo nativo path junto con __dirname para ubicar correctamente la referencia a la carpeta dentro del servidor.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Servidor de archivos estáticos</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Hola desde nuestro servidor de archivos estáticos</h1>
  </body>
  <script src="main.js"></script>
</html>
```

```
import express from 'express';
import { join, dirname } from 'path';
import { fileURLToPath } from 'url';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

const app = express();

// Configurar middleware para servir archivos estáticos
app.use(express.static(join(__dirname, 'public')));

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Servidor en http://localhost:${PORT}`);
});
```

3. Finalmente agregamos contenido a nuestros archivos estáticos

4. Ahora, si corremos nuestro servidor y nos dirigimos a la ruta <http://localhost:3000> desde el navegador, nos encontraremos con nuestro sitio web estático:



Hola desde nuestro servidor de archivos estáticos

*Tip: podés probar volcando todos los archivos de un sitio web estático que hayas creado en el pasado dentro de la carpeta **public** y deberías poder acceder y navegar localmente desde el navegador.



Middlewares

Middlewares en Express

¿Qué es un Middleware?

Un middleware es una función que se ejecuta entre el momento en que un cliente realiza una solicitud y el momento en que el servidor envía una respuesta. Cada middleware tiene la capacidad de modificar los objetos de solicitud y respuesta, finalizar el ciclo de solicitud/respondida, o pasar el control a otro middleware utilizando la función next().

Tipos de Middleware

Aplicación

Se ejecutan para todas las rutas.

Ruta

Manejan rutas específicas.

Terceros

Librerías externas que extienden Express, como `body-parser` o `cors`

Integrados

Incluidos en Express, como `express.static`.

Ejemplo de Middleware



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

app.get('/', (req, res) => {
  res.send('Hola desde Express con middlewares!');
});
```

Subiendo el proyecto a la nube

Subiendo el Proyecto a la Nube

Crear Repositorio Local

Inicializamos un repositorio Git en nuestro proyecto local.

Añadir Archivos

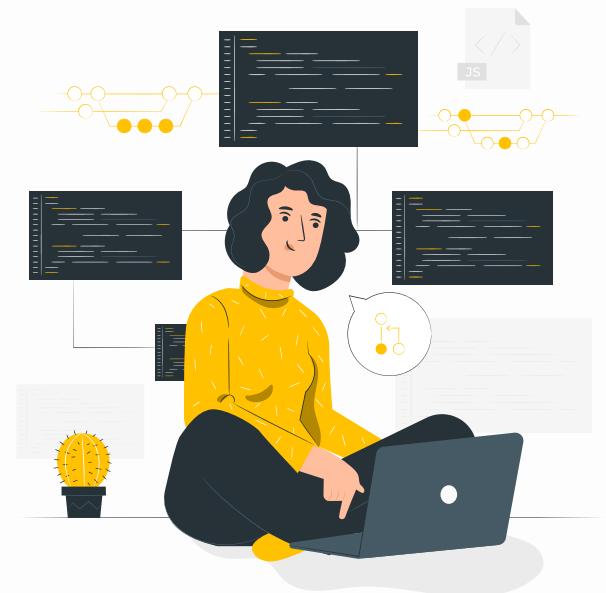
Agregamos los archivos del proyecto al staging de Git.

Realizar Commit

Hacemos un commit con un mensaje descriptivo.

Vincular con GitHub

Conectamos nuestro repositorio local con uno remoto en GitHub.



Git: Sistema de Control de Versiones



git

Control de Versiones

Git permite crear un historial del avance de nuestro código.



Gestión de Ramas

Facilita la gestión de diferentes versiones del proyecto.



Colaboración

Permite el trabajo en equipo de manera eficiente.

Crea un nuevo repositorio



1

Desde la terminal en nuestro proyecto, usamos el comando:

`git init`

2

El siguiente paso es añadir los archivos del proyecto al stage del GIT con:

`git add .`

3

Finalmente realizamos un commit junto a un mensaje para registrar todos nuestros archivos:

`git commit -m "[FEATURE] primera versión del proyecto completo"`

*Nota: en caso que hayas atravesado el desarrollo del proyecto mediante el uso de un repositorio de GIT puedes omitir este paso asegurándote que tengas la última versión de tu código registrada y lista para subir al repositorio remoto.



GitHub: Alojamiento de Repositorios

Alojamiento en la Nube

GitHub permite alojar nuestros proyectos Git de forma online.

Trabajo Colaborativo

Facilita el acceso al código desde cualquier parte del mundo.

Integración

Se integra fácilmente con servicios de hosting como Vercel.

Creando un Repositorio en GitHub

1 Iniciar Sesión

Accedemos a nuestra cuenta de GitHub.

2 Crear Repositorio

Damos clic en "New" para crear un nuevo repositorio.

3 Configurar

Elegimos un nombre y configuramos las opciones del repositorio.

4 Vincular

Conectamos nuestro repositorio local con el remoto.



Recursos Adicionales



[Documentación Node.js](#)

Profundizar en el uso del módulo path.



[Documentación Express.js](#)

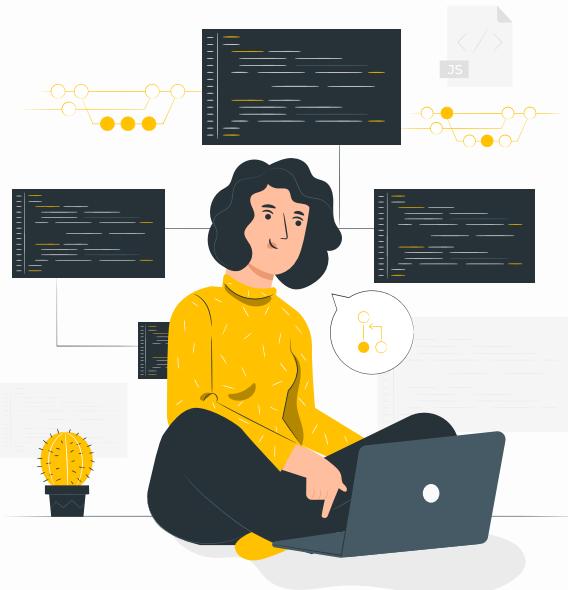
Explorar características avanzadas.



Herramientas API REST

[Postman](#) y [Insomnia](#) para probar APIs.

Próximos Pasos



1

Modelando una API Rest

Iniciar en la metodología para crear nuestra primera API Rest.

2

Request y Response

Aplicar conocimiento sobre la comunicación web.

3

Capa lógica

Controlar la respuesta de nuestra aplicación.



Ejercicios Prácticos

Ejercicio Práctico

Obligatorio

Ejercicio 1 - Configurando el Proyecto Base



“Bueno, esto está tomando forma, pero necesitamos establecer una base sólida para nuestro proyecto integrador”, comenta mientras señala tu laptop.

Sabrina asiente. “Crear un proyecto bien estructurado es como construir los cimientos de un edificio. Queremos que este proyecto sea el punto de partida para algo más grande.”



Ejercicio Práctico

Obligatorio

Misión:

1. Crea un nuevo proyecto con el comando `npm init -y` para generar un archivo `package.json` básico.
2. Inicializa un repositorio de Git en el proyecto con `git init`.
3. Crea un archivo `.gitignore` que excluya la carpeta `node_modules`.
4. Configura el archivo `package.json` para que soporte el estándar de imports ES Modules. Modifica el archivo añadiendo `"type": "module"`.
5. Define un script en `package.json` para ejecutar el proyecto con Node.js usando `"start": "node index.js"`.



Ejercicio Práctico

Obligatorio

Ejercicio 2 - Tu Primer Servidor Web

"Ahora es momento de que entres al mundo real del desarrollo web con Express," dice Matías, colocándose frente a la pizarra. "Queremos que configures un servidor básico que pueda manejar solicitudes."



Sabrina toma la palabra. "Recuerda, la clave está en lo simple pero funcional. Con una ruta bien configurada, demostraremos que nuestro servidor está listo para crecer."



Ejercicio Práctico

Obligatorio

Misión 2:

1. Usa la estructura creada en el ejercicio anterior para iniciar un servidor web.
2. Instala Express con **npm install express**.
3. Configura un servidor básico que corra en el puerto **3000** usando Express.
4. Agrega una ruta **/ping** que responda con el texto plano "/pong" cuando sea visitada desde un navegador.



Matías concluye: "Queremos ver un servidor funcional, código limpio y bien comentado. Este servidor será el corazón de nuestro proyecto integrador, así que pon tu mejor esfuerzo."

