CARLOS CHRISTIAN PÉREZ AZUAGA 2ºDAW2024-2025 13/12/2024

Configuración de un servidor Web con Node.js y Apache [RA3, {c), d)}]

Enunciado

Desarrollar una aplicación web básica con Node.js que muestre una página de inicio. Desplegar la aplicación utilizando Apache como servidor frontal para el contenido estático y Node.js con Express para manejar las peticiones dinámicas. Todo esto en contenedores Docker para simular un entorno de producción.

Objetivos

- **Usar Docker para simular un despliegue en un entorno productivo**
- **Integrar Apache y Node.js**

Instrucciones

A. Crear la aplicación con Node.js

1. **Crear el esqueleto de la aplicación**
    - **Crea una carpeta para tu proyecto, llámala `miapp`.**
    - **Inicia el proyecto con `npm init -y`.**

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> npm init -y
Wrote to C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp\package.json:

{
  "name": "miapp",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}


PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp>
```

Instala Express:

```
npm install express
```

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> npm install express

added 72 packages, and audited 73 packages in 2s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp>
```

2. **Estructura del proyecto:**

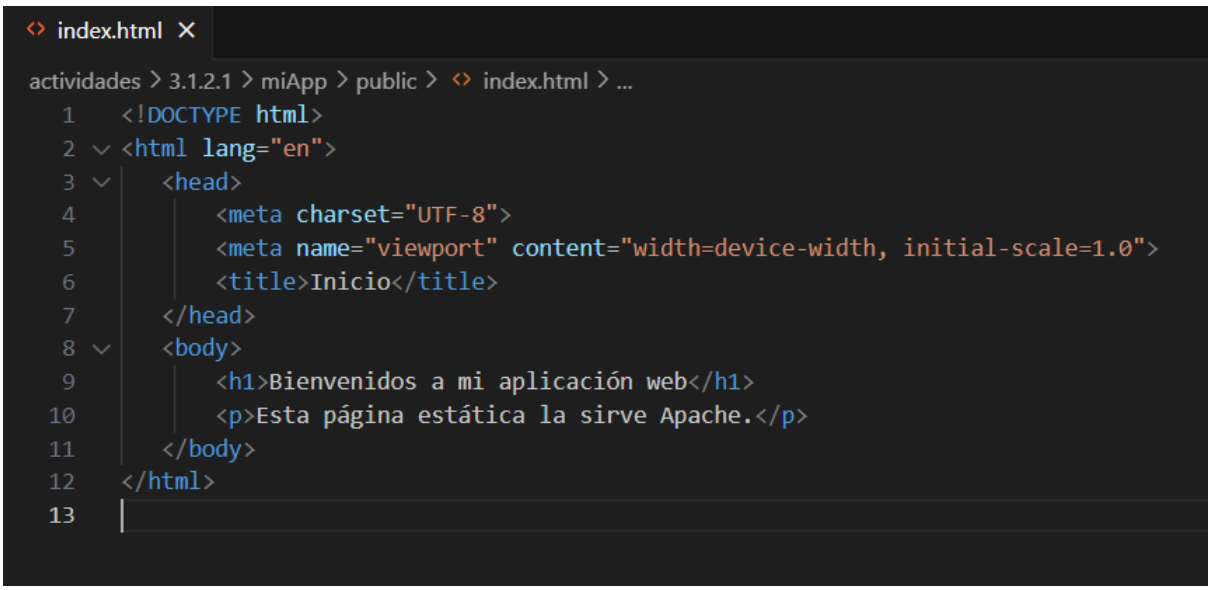
Dentro de `miapp`, creamos esta estructura de carpetas para organizar los archivos:

```
miapp/
├── apache-server/
│   ├── Dockerfile
│   └── httpd.conf
├── node-server/
│   ├── Dockerfile
├── public/          <-- Esta carpeta se copiará a
/usr/local/apache2/htdocs al crear el contenedor Apache Server.
│   └── index.html
├── src/             <-- Esta carpeta se montará como workdir en el
contenedor Node.
│   └── server.js
└── package.json
```

- 3. **Crea una página estática:**

En `public/index.html`, añade una página sencilla:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>Inicio</title>
    </head>
    <body>
        <h1>Bienvenidos a mi aplicación web</h1>
        <p>Esta página estática la sirve Apache.</p>
    </body>
</html>
```

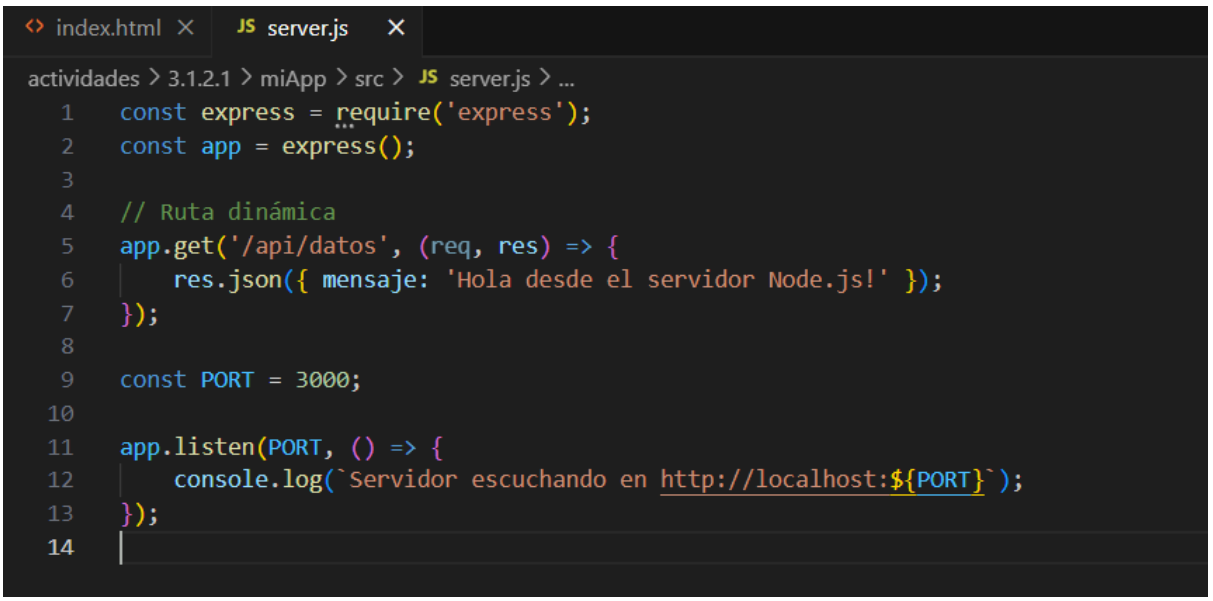


-

4. Configura el servidor con Express:

En `src/server.js`:

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

// Ruta dinámica
app.get('/api/datos', (req, res) => {
    res.json({ mensaje: 'Hola desde el servidor Node.js!' });
});

const PORT = 3000;

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



B. Configuración del servidor Apache

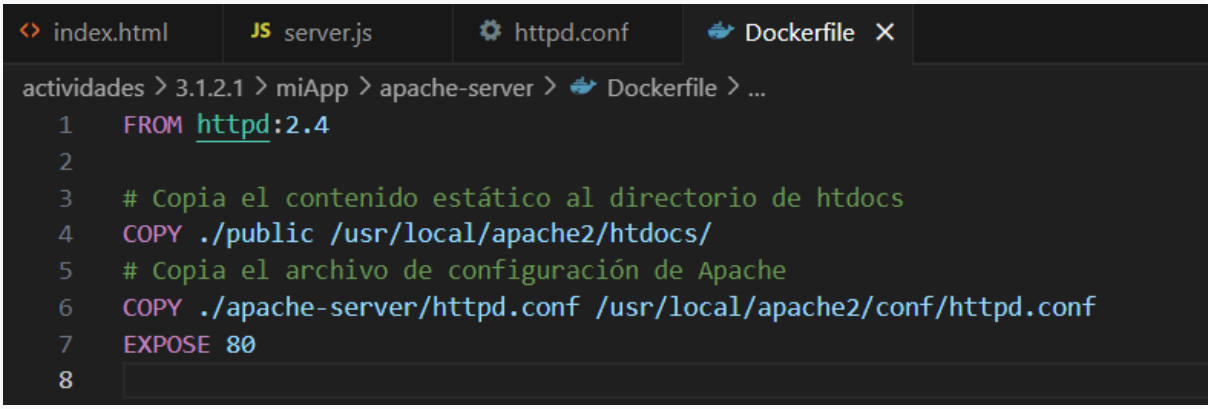
Descarga un archivo de configuración para Apache desde un contenedor httpd y utilízalo de partida añadiéndole las siguientes líneas:

```
# Módulos necesarios para el proxy
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

# Configuración principal
<VirtualHost *:80>
    ServerName localhost

    # Contenido estático
    DocumentRoot "/usr/local/apache2/htdocs"

    # Proxy dinámico hacia Node.js
    ProxyPreserveHost On
    ProxyPass /api http://node-container:3000/api
    ProxyPassReverse /api http://node-container:3000/api
```

```
</VirtualHost>

# Configura el directorio para el contenido estático
<Directory "/usr/local/apache2/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

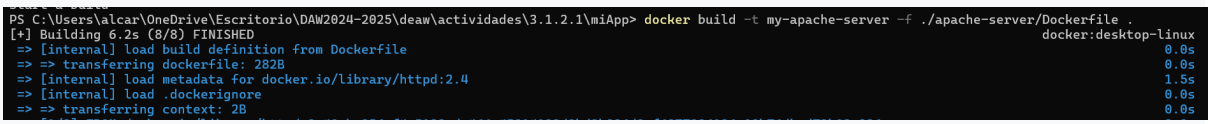
-
- **Configura un contenedor Docker con Apache**
    - **Crea un archivo `Dockerfile` en el directorio (`miapp/apache-server`) con este contenido:**
      `FROM httpd:2.4`

```
# Copia el contenido estático al directorio de htdocs
COPY ./public /usr/local/apache2/htdocs/

# Copia el archivo de configuración de Apache
COPY ./apache-server/httpd.conf /usr/local/apache2/conf/httpd.conf

EXPOSE 80
```



- **Construye el contenedor de Apache**
  `Ejecutar desde la carpeta miapp lo siguiente:`

docker build -t my-apache-server -f ./apache-server/Dockerfile .



C. Configuración del servidor de aplicaciones express.js

NOTA: En la versión original del ejercicio no se generaba un contenedor sino que se ejecutaba esta instrucción:

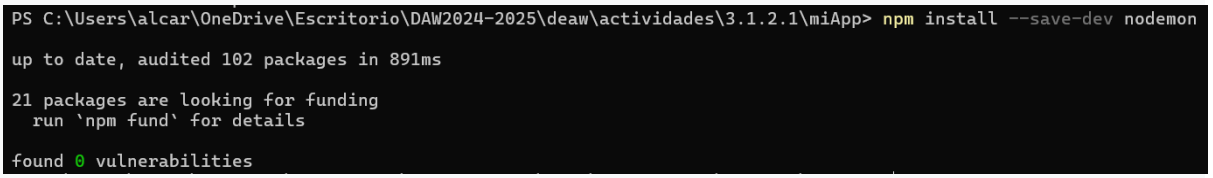
```
   docker run -d --name node-container --network mi-red -p 3000:3000 -v
$(pwd)/src:/usr/src/app node node src/server.js
```

Esto va a provocar un error ya que estamos importando express.js y por defecto no está instalado.

Para arreglar esto vamos a crear un contenedor y vamos a aprovechar para instalar `nodemon`, esta utilidad rearranca el servidor node si detecta cambios en el código, lo que junto al montaje de la carpeta de código en el contenedor desde la de Windows permitirá hacer cambios y desplegarlos en caliente.

- **Habilitar la recarga automática del servidor**
  Para que el servidor Node.js se recargue automáticamente al detectar cambios en el código, puedes instalar y usar una herramienta como `nodemon`. Instala `nodemon` como una dependencia de desarrollo en tu proyecto local:

```
npm install --save-dev nodemon
```



- **Configura un contenedor Docker con Node**

Crea un archivo `Dockerfile` en el directorio (`miapp/node-server`) con este contenido:

```
FROM node:16-alpine

# Establece el directorio de trabajo
WORKDIR /miapp

# Copia los archivos de dependencias
COPY package.json package-lock.json ./

# Instala las dependencias
RUN npm install

# Expone el puerto que usa el servidor
EXPOSE 3000

# Comando para ejecutar el servidor (con nodemon en lugar de node)
CMD ["npx", "nodemon", "src/server.js"]
```

httpd.conf | Dockerfile ...\apache-server | Dockerfile ...\node-server

actividades > 3.1.2.1 > miapp > node-server > Dockerfile > ...

```
1   FROM node:16-alpine
2   # Establece el directorio de trabajo
3   WORKDIR /miapp
4   # Copia los archivos de dependencias
5   COPY package.json package-lock.json ./
6   # Instala las dependencias
7   RUN npm install
8   # Expone el puerto que usa el servidor
9   EXPOSE 3000
10  # Comando para ejecutar el servidor (con nodemon en lugar de node)
11  CMD ["npx", "nodemon", "src/server.js"]
```

- **Construye el contenedor de Apache**
  `Ejecutar desde la carpeta miapp lo siguiente:`

docker build -t my-node-server -f ./node-server/Dockerfile .

```
found 0 vulnerabilities
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> docker build -t my-node-server -f ./node-server/Dockerfile .
[+] Building 1.1s (9/9) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.0s
 => => transferring dockerfile: 397B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/node:16-alpine                                                                 1.0s
 => [internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 2B                                                                                                   0.0s
 => [1/4] FROM docker.io/library/node:16-alpine@sha256:a1f9d027912b58a7c75be7716c97cfbc6d3099f3a97ed84aa490be9dee20e787            0.0s
 => [internal] load build context                                                                                                 0.0s
 => => transferring context: 44.65kB                                                                                              0.0s
 => CACHED [2/4] WORKDIR /miapp                                                                                                   0.0s
 => CACHED [3/4] COPY package.json package-lock.json ./                                                                           0.0s
 => CACHED [4/4] RUN npm install                                                                                                  0.0s
 => exporting to image                                                                                                            0.0s
```

D. Ejecutar los contenedores

- **Crea una red Docker:** `docker network create mi-red`

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> docker network create mi-red
cf1a58e649b4310a0be2c4d7814ea980f9e6ccc7824f3393a03f82fa328b70e0
```

- **Ejecuta el contenedor de Node.js**

docker run -d --rm --name node-container --network mi-red `-p 3000:3000` -v $(pwd)/src:/miapp/src my-node-server

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> docker run -d --rm --name node-container --network mi-red -p 3000:3000 -v
 $PWD/src:/miapp/src my-node-server
a948a63fd19c15ff81f1142be6659dacb5d77864872139a6f096ab6da3a3e35e
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp>
```

Como estoy ejecutando el comando desde powershell se usa $PWD en vez de ${pwd}

- **Ejecuta el contenedor de Apache**

docker run -d --rm --name apache-container --network mi-red -p 8080:80 my-apache-server

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> docker run -d --rm --name apache-container --network mi-red -p 8080:80 my
-apache-server
2ec6819e52080170b1cca114c150627df44c2966c667e1f94c03897744fdc43f
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp>
```

E: Prueba la aplicación

- **Verifica que el contenedor de Node.js está corriendo correctamente:**
- `docker ps`

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS                    NAMES
a948a63fd19c   my-node-server   "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   0.0.0.0:3000->3000/tcp   node-container
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp>
```

- **Revisa los logs del contenedor para confirmar que el servidor se inició sin errores:** `docker logs node-container`

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miapp>  docker logs node-container
[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/server.js`
Servidor escuchando en http://localhost:3000
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miapp>
```

Cambie el nombre de la carpeta a mi app desde el apartado C. porque me daban errores pero como no salí de powershell no se actualizo directamente pero aquí la prueba:

```
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miApp> cd ..
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1> cd .\miapp\
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miapp>  docker logs node-container
[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/server.js`
Servidor escuchando en http://localhost:3000
PS C:\Users\alcar\OneDrive\Escritorio\DAW2024-2025\deaw\actividades\3.1.2.1\miapp>
```

- **Accede a** `http://localhost:8080` **para ver la página estática servida por Apache.**
- **Accede a** `http://localhost:8080/api/datos` **para ver la respuesta dinámica del servidor Node.js.**

F: Reto adicional

En lugar de contenedores Docker configura un fichero docker-file que haga lo mismo.