Guía de Configuración y Puesta en Marcha

Proyecto de Ejemplo: API Node.js + Cliente React

Juan Felipe Orozco Cortés

Instructor - SENA

Agosto de 2025

Índice

1. Propósito

Este documento sirve como una guía técnica paso a paso para la instalación y ejecución del proyecto Full-Stack de ejemplo. El objetivo es que cada aprendiz pueda configurar y correr el entorno de desarrollo completo en su computador local, el cual consiste en un backend (API) de Node.js y un frontend (Cliente) de React.js.

2. Estructura del Proyecto

El repositorio está organizado como un "monorepo", conteniendo dos proyectos independientes en sus respectivas carpetas.

```
/
 gestor-tareas-backend/    # Contiene el código de la API de Node.js
 gestor-tareas-frontend/   # Contiene el código de la aplicación de React.js
```

Nota Importante: Para que la aplicación completa funcione, es necesario tener ambos proyectos corriendo al mismo tiempo en dos terminales separadas.

3. Prerrequisitos

Antes de empezar, asegúrese de tener instalado en su sistema:

- **Node.js:** Versión 18 o superior.
- **Git:** Para clonar el repositorio desde GitHub.

4. Guía de Instalación y Ejecución

Siga estos pasos en orden para poner en marcha el proyecto.

4.1. Paso 1: Clonar el Repositorio

Abra una terminal y clone el repositorio en la carpeta de su preferencia.

```
git clone https://github.com/topassky3/despliegue-api-nodejs-apache.git
```

Luego, navegue a la carpeta principal del proyecto:

```
cd despliegue-api-nodejs-apache
```

4.2. Paso 2: Poner en Marcha el Backend (El Servidor)

El backend es el "motor"que provee los datos.

1. **Navegue a la carpeta del backend:**

   ```
   cd gestor-tareas-backend
   ```

2. **Instale las dependencias:**

   ```
   npm install
   ```

3. **Inicie el servidor en modo de desarrollo:**

   ```
   npm run dev
   ```

4. **Verificación:** La terminal deberá mostrar un mensaje similar a:
   `[nodemon] starting ǹode src/app.js`
   `Servidor corriendo en http://localhost:4000`

¡Crítico! Deje esta terminal abierta. El servidor debe seguir corriendo.

4.3. Paso 3: Poner en Marcha el Frontend (La Interfaz de Usuario)

El frontend es la parte visual con la que interactuamos.

1. **¡ABRA UNA NUEVA TERMINAL!** No utilice la misma del backend.
2. **Desde la raíz del proyecto**, navegue a la carpeta del frontend:

   ```
   cd gestor-tareas-frontend
   ```

3. **Instale las dependencias:**

   ```
   npm install
   ```

4. **Inicie la aplicación de React:**

   ```
   npm run dev
   ```

5. **Verificación:** La terminal le mostrará la dirección para abrir la aplicación:
   `Local: http://localhost:5173/`

4.4. Paso 4: Prueba Final

Abra su navegador web y visite la dirección `http://localhost:5173/`.

Si ambos servidores están corriendo correctamente, deberá ver la aplicación de gestión de tareas cargando la información directamente desde su API local. ¡Felicidades, el entorno Full-Stack está funcionando!

5. Guía de Despliegue en Producción

Una vez que nuestra aplicación funciona en `localhost`, el siguiente paso es llevarla al mundo real. Este proceso implica mover nuestro código a un servidor en la nube y configurarlo para que sea accesible de forma segura desde internet.

5.1. Prerrequisitos para el Despliegue

- **Un Servidor Privado Virtual (VPS):** Es un servidor Linux en la nube. Puedes obtener uno de proveedores como **OVH, DigitalOcean, Google Cloud, Colombia Hosting,** etc. Para esta guía, asumimos un sistema operativo Debian o Ubuntu.

- **Un Nombre de Dominio:** Una dirección web (ej: `yarumaltech.com`) que apuntará a la IP de tu VPS.
- **Acceso SSH** a tu servidor.

5.2. Paso 1: Configuración del DNS

Debes crear un **registro tipo A** en el panel de control de tu proveedor de dominio para que un subdominio (ej: `api.yarumaltech.com`) apunte a la dirección IP pública de tu VPS.

5.3. Paso 2: Preparación del Servidor

Una vez conectado a tu servidor por SSH, instala el software necesario.

1. **Instalar Node.js y PM2** (Gestor de Procesos):

```
# Instalar una version LTS de Node.js
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar PM2 globalmente
sudo npm install -g pm2
```

2. **Instalar Apache2:**

```
sudo apt update && sudo apt install apache2
```

5.4. Paso 3: Ejecutar la API

Sube el código de tu backend al servidor (usando `git clone`) e inícialo con PM2 para que corra de forma persistente.

```
# Estando dentro de la carpeta del backend...
pm2 start src/app.js --name "mi-api"
```

5.5. Paso 4: Configurar Apache2 como Reverse Proxy

Creamos un archivo de "Virtual Host"para indicarle a Apache cómo manejar el tráfico para nuestro subdominio.

1. **Habilitar módulos de proxy:**

```
sudo a2enmod proxy proxy_http
sudo systemctl restart apache2
```

2. **Crear y editar el archivo de configuración:**

```
sudo nano /etc/apache2/sites-available/api.tudominio.com.conf
```

3. **Añadir la configuración del Reverse Proxy:**

```
<VirtualHost *:80>
    ServerName api.tudominio.com

    ProxyPass / http://localhost:4000/
    ProxyPassReverse / http://localhost:4000/
</VirtualHost>

```

Listing 1: Configuración del Virtual Host para el puerto 80.

4. **Habilitar el sitio y reiniciar Apache:**

   ```
   sudo a2ensite api.tudominio.com.conf
   sudo systemctl restart apache2
   ```

5.6. Paso 5: Asegurar con HTTPS (Let's Encrypt)

Finalmente, usamos `certbot` para obtener un certificado SSL gratuito y configurar HTTPS automáticamente.

```
# Instalar Certbot
sudo apt install certbot python3-certbot-apache

# Ejecutar Certbot para nuestro dominio
sudo certbot --apache -d api.tudominio.com
```

Sigue las instrucciones y elige la opción de redirigir todo el tráfico a HTTPS.

5.7. Paso Final: Conectando el Cliente React

El último paso es decirle a nuestra aplicación de React que ahora debe "hablarçon la nueva URL pública en lugar de con `localhost`.

1. Abre el código del **frontend** en tu computador local.
2. Navega al archivo principal que maneja el estado (ej: `src/App.jsx`).
3. Localiza la constante donde definiste la URL de la API.
4. **Modifica la URL** para que apunte a tu nuevo dominio seguro.

```
// --- ANTES (Para desarrollo local) ---
// const API_URL = 'http://localhost:4000/api';

// --- AHORA (Para producci n) ---
const API_URL = 'https://api.yarumaltech.com/api';

function App() {
  // ... el resto de tu c digo de React sigue igual
  useEffect(() => {
    const fetchTasks = async () => {
      // Fetch ahora usara la nueva URL de produccion
      const response = await fetch(`${API_URL}/tasks`);
      // ...
    };
    fetchTasks();
  }, []);

```

```
18    // ...
19 }
```

Listing 2: Modificación clave en el código de React.

Al hacer 'build' de tu aplicación de React con este cambio, ahora consumirá los datos desde tu servidor en producción.