Universidad de San Carlos de Guatemala.
Centro Universitario de Occidente.
División de Ciencias de la Ingeniería.
Manejo e Implementación de Archivos.
Ing. Christian Lopéz

“DOCUMENTACIÓN TÉCNICA - GRAFILES”

Diego José Maldonado Monterroso.
Carné: 201931811.
Link Repositorio - Backend: https://github.com/DiegoMaldonado19/GraFiles-Backend
Link Repositorio - Frontend: https://github.com/DiegoMaldonado19/GraFiles-Frontend

Quetzaltenango, Guatemala.

31 de octubre de 2024.

INTRODUCCIÓN

Este proyecto consiste en desarrollar un gestor de archivos en la nube utilizando MongoDB, Node.js, Express, React y Docker. La aplicación permite a los usuarios almacenar, organizar y compartir archivos de manera segura y eficiente. La dockerización de la base de datos MongoDB asegura un entorno de desarrollo consistente y fácil de desplegar.

El objetivo es experimentar con tecnologías Javascript, y asi mismo, poder aplicar y ejecutar un entorno MERN (MongoDB, Express, React, NodeJS), para agilizar el proceso de desarrollo y ejecución.

MARCO TEÓRICO

Capítulo 1

1.1 Introducción a Docker

Docker es una plataforma de contenerización que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores. Estos contenedores son entornos ligeros y portátiles que garantizan que la aplicación se ejecute de manera consistente en cualquier entorno.

1.2 Arquitectura de Docker

Docker se compone de varios componentes clave:

- Docker Engine: El motor de contenedores que crea y ejecuta contenedores.
- Docker CLI: La interfaz de línea de comandos para interactuar con Docker.
- Docker Hub: Un registro de imágenes de contenedores que permite compartir y almacenar imágenes.

Capítulo 2.

2.1 Introducción a Docker Compose

Docker Compose es una herramienta para definir y ejecutar aplicaciones multi-contenedor. Utiliza un archivo YAML para configurar los servicios de la aplicación, lo que simplifica la gestión de entornos complejos. Con Docker Compose, se puede definir la configuración de los servicios en un archivo docker-compose.yml y luego iniciar todos los servicios con un solo comando: docker-compose up.

Capítulo 3.

3.1 Introducción a Docker Desktop

Docker Desktop es una aplicación para Windows y macOS que proporciona una interfaz gráfica para gestionar contenedores Docker. Incluye Docker Engine, Docker CLI, Docker Compose y otras herramientas necesarias para el desarrollo y la prueba de aplicaciones en contenedores.

3.2 Características de Docker Desktop

- Integración con IDEs: Facilita la integración con herramientas de desarrollo como VS Code.
- Kubernetes: Incluye soporte para Kubernetes, permitiendo la orquestación de contenedores.
- Gestión de Volúmenes: Permite la gestión eficiente de datos compartidos entre contenedores.

Capítulo 4.

4.1 Introducción a las Bases de Datos NoSQL

Las bases de datos no relacionales, o NoSQL, son sistemas de almacenamiento de datos que no utilizan el modelo de tablas y relaciones de las bases de datos SQL tradicionales. Son ideales para manejar grandes volúmenes de datos no estructurados o semi-estructurados.

4.2 Tipos de Bases de Datos NoSQL

- Clave-Valor: Almacenan datos como pares clave-valor.
- Documentales: Almacenan datos en documentos, como JSON o BSON.
- Gráficas: Utilizan grafos para representar y consultar datos.
- Memoria: Almacenan datos en la memoria principal para acceso rápido.

Capítulo 5.

5.1 Introducción a MongoDB

MongoDB es una base de datos NoSQL orientada a documentos que almacena datos en formato BSON. Es conocida por su flexibilidad y escalabilidad.

5.2 Características de MongoDB

Modelo de Datos Flexible: Permite almacenar documentos con estructuras variadas.
Escalabilidad Horizontal: Facilita la distribución de datos en múltiples servidores.
Consultas Ricas: Soporta consultas complejas y agregaciones.

Capítulo 6.

6.1 Introducción a Mongo Compass

MongoDB Compass es una interfaz gráfica para MongoDB que permite explorar, analizar y optimizar datos. Proporciona herramientas para la visualización de esquemas, la creación de consultas y la gestión de índices.

6.2 Funcionalidades de Mongo Compass

- Análisis de Esquemas: Permite entender la estructura de los datos.
- Consultas Interactivas: Facilita la creación y ejecución de consultas.
- Optimización de Índices: Ayuda a mejorar el rendimiento de las consultas.

Capítulo 7.

7.1 Introducción a JavaScript

JavaScript es un lenguaje de programación interpretado que se utiliza principalmente para el desarrollo web. Permite crear contenido dinámico e interactivo en las páginas web.

7.2 Características de JavaScript

- Lenguaje de Alto Nivel: Facilita la escritura de código legible y mantenible.
- Event-Driven: Soporta programación basada en eventos.
- Compatibilidad con Navegadores: Funciona en todos los navegadores modernos.

Capítulo 8

8.1 Introducción a NodeJS

Node.js es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript en el servidor. Está construido sobre el motor V8 de Chrome y es conocido por su capacidad para manejar aplicaciones de red escalables y de alto rendimiento.

8.2 Características de NodeJS

- Asincronía: Utiliza un modelo de E/S no bloqueante.
- Escalabilidad: Ideal para aplicaciones que requieren alta concurrencia.
- NPM: Incluye un gestor de paquetes que facilita la gestión de dependencias.

Capítulo 9.

9.1 Introducción a NPM

NPM (Node Package Manager) es el gestor de paquetes para Node.js. Permite a los desarrolladores instalar, compartir y gestionar dependencias de proyectos JavaScript.

9.2 Uso de NPM

- Instalación de Paquetes: npm install <paquete>
- Gestión de Dependencias: Facilita la actualización y eliminación de paquetes.
- Scripts: Permite definir y ejecutar scripts personalizados.

Capítulo 10.

10.1 Introducción a Express

Express es un framework web minimalista para Node.js que facilita la creación de aplicaciones web y APIs. Proporciona una serie de características robustas para el desarrollo de aplicaciones web.

10.2 Características de Express

- Manejo de Rutas: Define rutas para manejar diferentes solicitudes HTTP.
- Middleware: Permite añadir funcionalidades adicionales a las aplicaciones.
- Plantillas: Soporta motores de plantillas para generar HTML dinámico.

Capítulo 11.

11.1 Introducción a React JS

React es una biblioteca de JavaScript para construir interfaces de usuario. Desarrollada por Facebook, permite crear componentes reutilizables que gestionan su propio estado.

11.2 Características de React JS

- Componentes: Facilita la creación de interfaces modulares.
- Virtual DOM: Mejora el rendimiento al minimizar las actualizaciones del DOM real.
- Unidirectional Data Flow: Simplifica la gestión del estado de la aplicación.

Capítulo 12.

12.1 Introducción al Backend

El backend se refiere a la parte del desarrollo web que se encarga de la lógica del servidor, la gestión de bases de datos y la integración de APIs. Es responsable de procesar las solicitudes del cliente, ejecutar la lógica de negocio y devolver las respuestas adecuadas.

12.2 Tecnologías de Backend

- NodeJS: Para ejecutar JavaScript en el servidor.
- Express: Para crear aplicaciones web y APIs.
- Bases de Datos: Como MongoDB para el almacenamiento de datos.

Capítulo 13.

13.1 Introducción al Frontend

El frontend es la parte del desarrollo web que se ocupa de la interfaz de usuario y la experiencia del usuario. Incluye el diseño y la implementación de elementos visuales y la interacción con el usuario mediante HTML, CSS y JavaScript.

13.2 Tecnologías de Frontend

- HTML: Para estructurar el contenido.
- CSS: Para estilizar el contenido.
- JavaScript: Para añadir interactividad.

Capítulo 14.

14.1 Introducción a las API REST

Una API REST (Representational State Transfer) es un estilo de arquitectura para diseñar servicios web. Utiliza métodos HTTP estándar (GET, POST, PUT, DELETE) para interactuar con recursos y permite la comunicación entre diferentes sistemas de manera sencilla y escalable.

14.2 Principios de las API REST

- Stateless: Cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud.
- Cacheable: Las respuestas deben indicar si se pueden almacenar en caché.
- Uniform Interface: Debe haber una interfaz uniforme para interactuar con los recursos.

Capítulo 15.

15.1 Introducción al MERN Stack

El MERN Stack es un conjunto de tecnologías para el desarrollo de aplicaciones web que incluye MongoDB, Express, React y Node.js. Este stack permite a los desarrolladores trabajar con JavaScript en todas las capas de la aplicación, desde la base de datos hasta el frontend.

15.2 Componentes del MERN Stack

- MongoDB: Base de datos NoSQL para el almacenamiento de datos.
- Express: Framework para construir aplicaciones web y APIs.
- React: Biblioteca para construir interfaces de usuario.
- Node.js: Entorno de ejecución para JavaScript en el servidor, que permite construir aplicaciones de red escalables y de alto rendimiento.

15.3 Ventajas del MERN Stack

- JavaScript en todas partes: Permite a los desarrolladores utilizar un solo lenguaje de programación en el frontend y el backend.
- Desarrollo Rápido: Las herramientas y bibliotecas del MERN Stack facilitan el desarrollo rápido de aplicaciones web.
- Comunidad Activa: Amplia comunidad de desarrolladores que contribuyen con paquetes y soluciones.

Capítulo 16.

16.1 Comunicación entre Frontend y Backend

En una aplicación MERN, el frontend (React) se comunica con el backend (Express y Node.js) a través de API REST. Las solicitudes HTTP se envían desde el frontend y backend, donde se procesan y se interactúa con la base de datos (MongoDB) para obtener o almacenar datos.

16.2 Gestión del Estado

React utiliza bibliotecas como Redux o Context API para gestionar el estado de la aplicación. Esto permite mantener la consistencia de los datos en toda la interfaz de usuario y facilita la comunicación entre componentes.

Capítulo 17.

17.1 Desarrollo Local

Durante el desarrollo, Docker y Docker Compose se utilizan para crear entornos de desarrollo consistentes. Docker Desktop facilita la gestión de contenedores y la integración con herramientas de desarrollo.

17.2 Despliegue en Producción

Para el despliegue en producción, los contenedores Docker se pueden orquestar utilizando Kubernetes o servicios de orquestación en la nube. Esto asegura que la aplicación sea escalable y resiliente.

VERSIONES DE SOFTWARE

Sistema Operativo: Microsoft Windows 11.

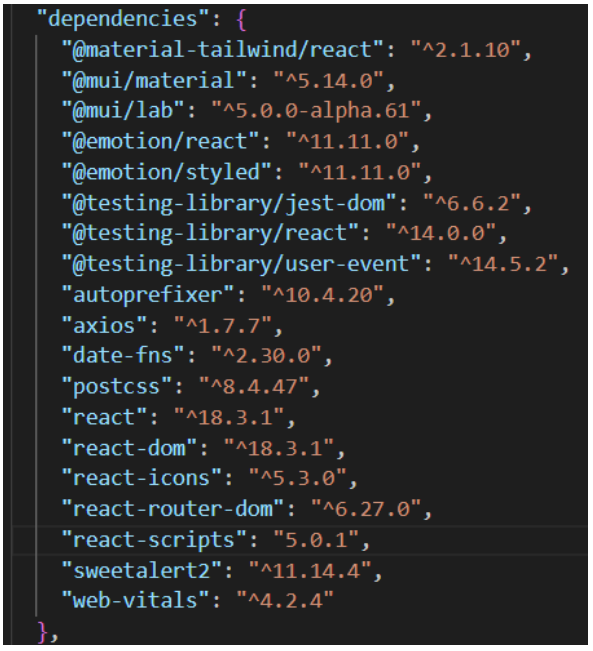
Backend (Docker, NodeJS, NPM, Mongo Compass)

- ***Docker Desktop:*** v4.34.3
- ***Mongo Compass:*** v1.44.5
- ***Imagen Mongo:*** mongo:latest
- ***NodeJS:*** v20.18.0
- ***NPM:*** v10.8.2
- ***Dependencias:***

```
"dependencies": {
  "bcrypt": "^5.1.1",
  "bcryptjs": "^2.4.3",
  "cors": "^2.8.5",
  "cors-anywhere": "^0.4.4",
  "express": "^4.21.1",
  "mongoose": "^7.6.3"
}
```

Frontend (NodeJS, NPM, ReactJS, TaildwindCSS)

- ***NodeJS:*** v20.18.0
- ***NPM:*** v10.8.2
- ***ReactJS:*** v18.3.1
- ***TaildwindCSS:*** v3.4.14
- ***Dependencias:***



```
"dependencies": {
  "@material-tailwind/react": "^2.1.10",
  "@mui/material": "^5.14.0",
  "@mui/lab": "^5.0.0-alpha.61",
  "@emotion/react": "^11.11.0",
  "@emotion/styled": "^11.11.0",
  "@testing-library/jest-dom": "^6.6.2",
  "@testing-library/react": "^14.0.0",
  "@testing-library/user-event": "^14.5.2",
  "autoprefixer": "^10.4.20",
  "axios": "^1.7.7",
  "date-fns": "^2.30.0",
  "postcss": "^8.4.47",
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-icons": "^5.3.0",
  "react-router-dom": "^6.27.0",
  "react-scripts": "5.0.1",
  "sweetalert2": "^11.14.4",
  "web-vitals": "^4.2.4"
},
```

CONFIGURACIÓN DE ENTORNO

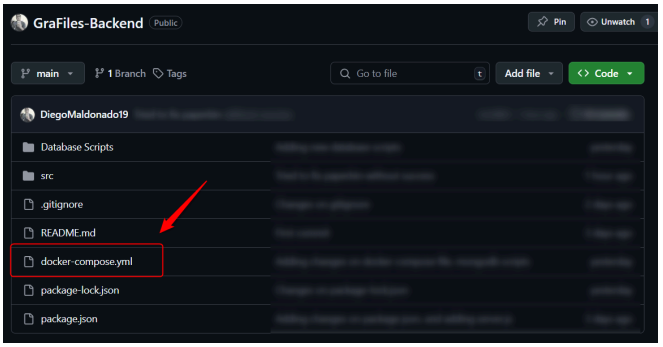
1. Debes instalar los siguientes programas en tu dispositivo (NodeJS, Docker Desktop, NPM, MongoDB Compass).

    - Node JS: https://nodejs.org/en/download/package-manager
    - NPM:https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager
    - Docker Desktop: https://www.docker.com/products/docker-desktop/
    - Mongo Compass: https://www.mongodb.com/es/products/tools/compass

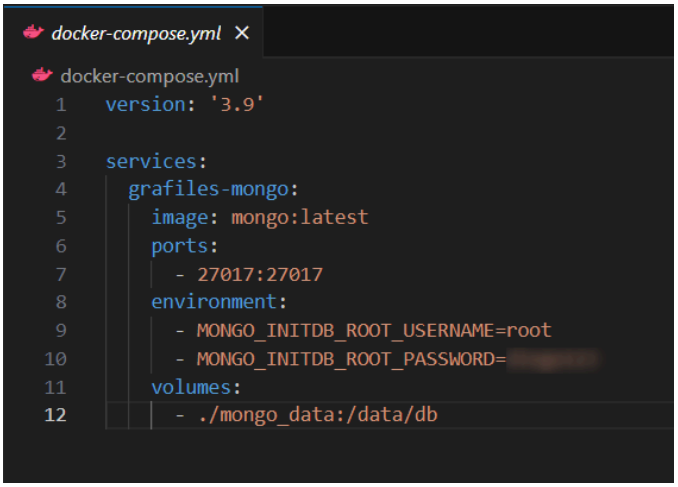
2. Como primer paso, debes clonar ambos proyectos en github a tu computadora local.

    - Link Repositorio - Backend: https://github.com/DiegoMaldonado19/GraFiles-Backend
    - Link Repositorio - Frontend: https://github.com/DiegoMaldonado19/GraFiles-Frontend

3. Posteriormente, debes abrir ambos proyectos en tu IDE de preferencia. Encontrarás el siguiente archivo “docker-compose.yml” dentro de la carpeta raíz del proyecto del Backend.



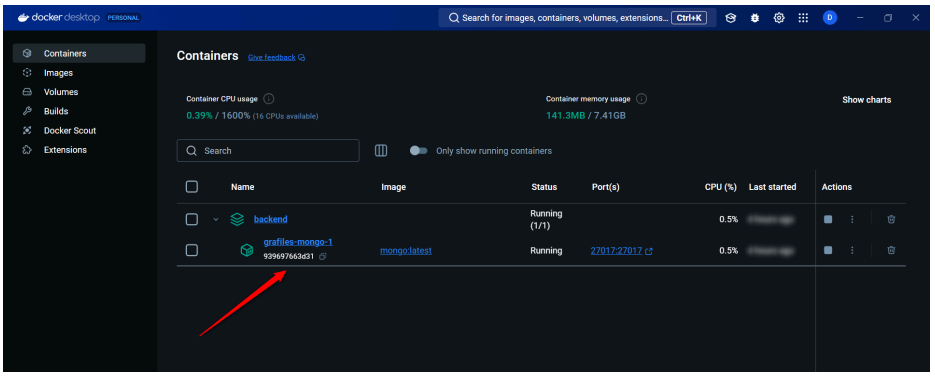
Este archivo contiene la configuración para crear nuestra imagen de MongoDB en un contenedor con Alpine Linux.



4. Debes iniciar una shell dentro de la carpeta, y escribir el siguiente comando:

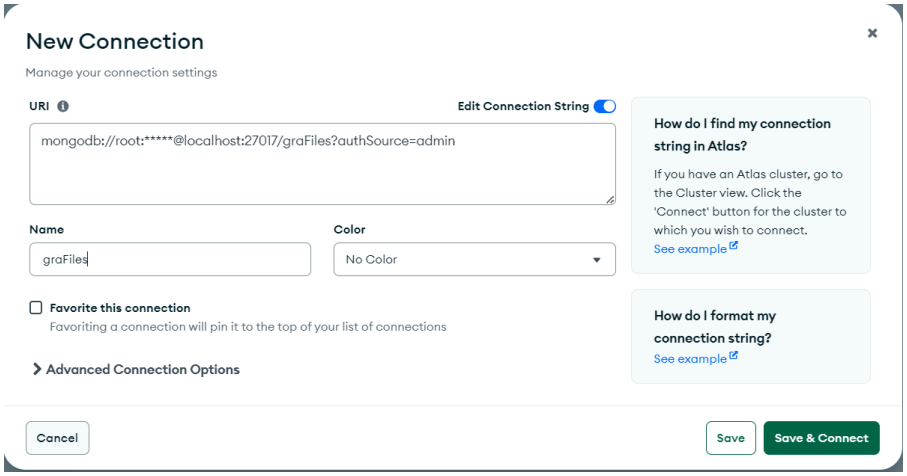
“docker compose up -d”

5. Luego de que docker haga un pull de la imagen y cree el contenedor con sus servicios respectivos, deberías ver tu contenedor corriendo de esta manera desde Docker Desktop, también puedes usar el comando ***“docker ps”.***



6. Una vez nuestro contenedor esté arriba, debemos abrir nuestra aplicación de Mongo Compass. Y crear una nueva conexión, para esto usaremos la siguiente conexión:

“mongodb://root:[contraseña]@localhost:27017”



También puedes conectarte al contenedor con el siguiente comando desde una shell local:

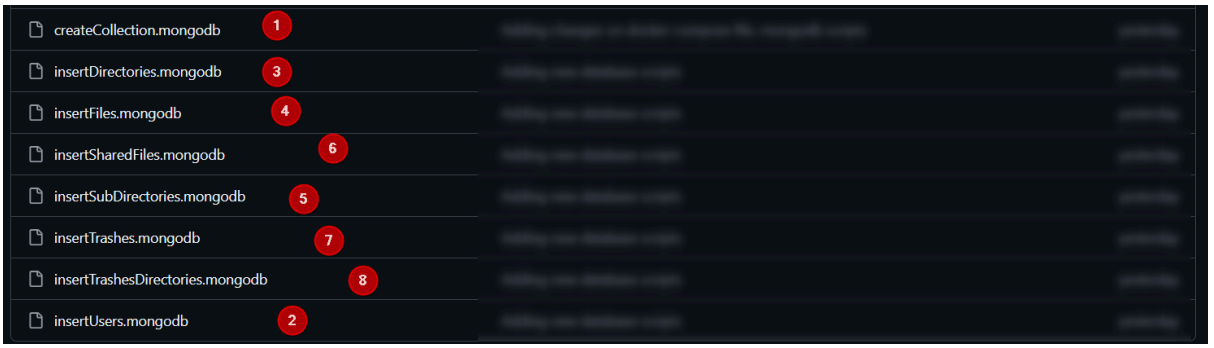
“docker exec -it backend-grafiles-mongo-1 bash”

Dentro del contenedor puedes correr este comando:

“mongosh mongodb://root:[contraseña]@localhost:27017”

7. Desde la GUI de Mongo Compass o la consola debes correr los siguientes scripts que se encuentran dentro del repositorio del Backend en el siguiente folder: “Database Scripts”.

   Importante: Debes guardar cada Objectid y reemplazarlo según sea necesario.



8. Debes abrir una consola para cada proyecto, eliminar el archivo “package-lock.json” y correr el siguiente comando para cada proyecto (Backend y Frontend):

   “npm install”

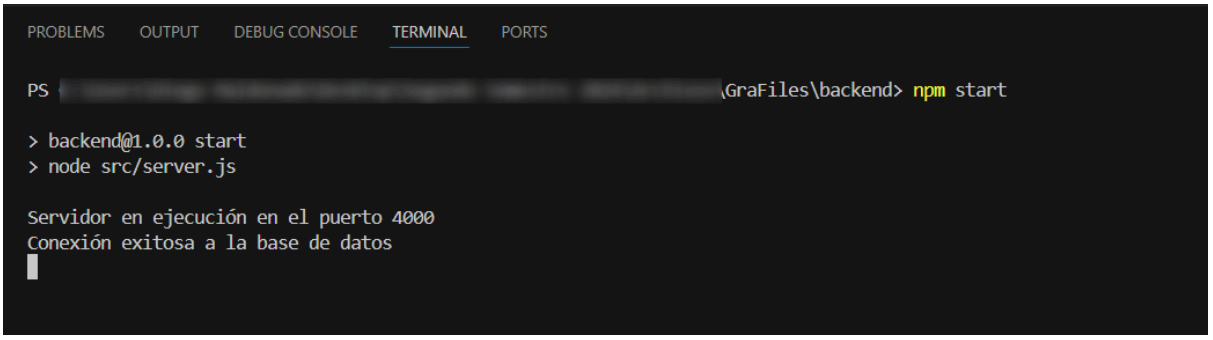
9. Una vez finalice la instalación de dependencias en tu local, deberías poder iniciar cada servidor usando ***“npm start”.***

10. El servidor para Backend corre en el puerto 4000.

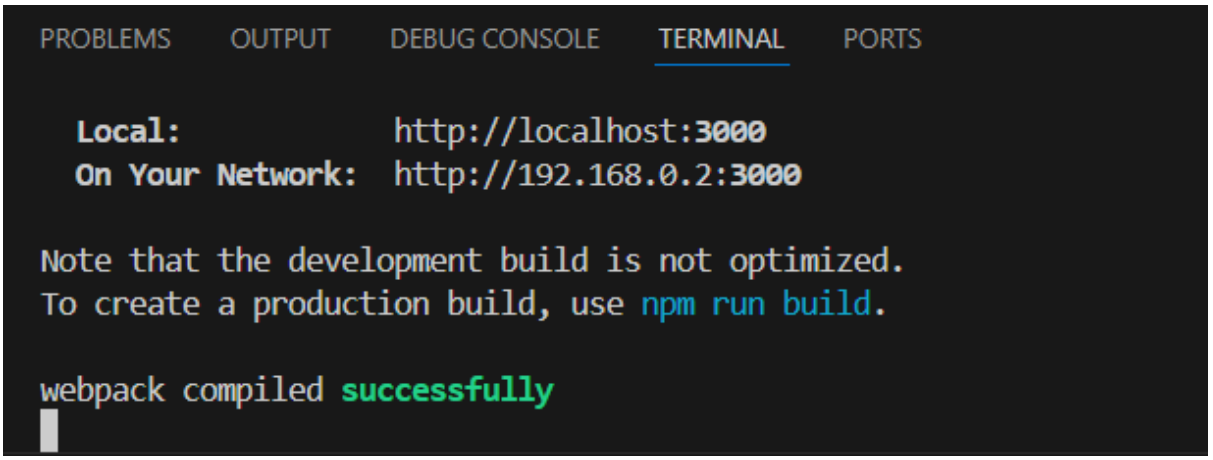
11. El servidor para Frontend corre en el puerto 3000.

12. Y listo:

Backend.



Frontend.



Aplicación.