

Proyecto de software basado en herramientas de integración continua

Integración continua

Entrega Semana 3

Integrantes:

Lisbeth Alvarez Zuluaga

Andres Felipe Triana Gallego

Politécnico Grancolombiano

Ingeniería de Software

2023

Contenido:.....	1
Proyecto a realizar y lenguaje	3
Justificación	4
Alcances.....	5
Enlace repositorio y Contexto del código	6

Proyecto a realizar y lenguaje

Realizaremos un ecommerce que nos permita poner en práctica todo lo aprendido, además de generar un entregable de alta confiabilidad, alta y rápida respuesta, fácil escalabilidad e integración. Para esto utilizaremos como base:

Front-end: JavaScript es esencial para el desarrollo del front-end de un sitio web de comercio electrónico. Aquí podemos utilizar bibliotecas y frameworks como React, Angular o Vue.js para crear interfaces de usuario interactivas y atractivas. Estas tecnologías nos permiten construir páginas de producto, carritos de compra, formularios de registro y otros componentes front-end.

Back-end: Para el desarrollo del back-end, Node.js nos parece una excelente opción. Node.js nos permite utilizar JavaScript en el lado del servidor. Puedes construir una API RESTful o GraphQL para gestionar la lógica empresarial, manejar la autenticación de usuarios, procesar pedidos y gestionar la base de datos.

Base de datos: JavaScript no es un lenguaje de base de datos, por lo que necesitaremos una base de datos para almacenar la información de productos, pedidos, usuarios, etc. Usaremos bases de datos relacionales MySQL.

Justificación

Crear un ecommerce con las características mencionadas en la respuesta anterior que permita emplear herramientas para la integración continua, utilizando JavaScript tanto en el front-end como en el back-end, nos da ventajas y aspectos positivos. Aquí hay algunos de ellos:

Ventajas:

Versatilidad tecnológica: El uso de JavaScript en el front-end y Node.js en el back-end nos brinda una plataforma versátil y unificada para el desarrollo.

Interactividad y experiencia de usuario mejorada: JavaScript nos permite crear interfaces de usuario altamente interactivas y atractivas. Los frameworks como React, Angular y Vue.js son ideales para proporcionar a los usuarios una experiencia de compra más agradable y personalizada.

Eficiencia en el desarrollo: Al utilizar JavaScript en todo el ciclo de desarrollo, nos permite optimizar la eficiencia y la velocidad de desarrollo.

Comunidad y recursos abundantes: JavaScript tiene una comunidad de desarrollo activa y una gran cantidad de recursos, tutoriales y bibliotecas disponibles para facilitar el desarrollo y resolver problemas comunes.

Escalabilidad y rendimiento: Node.js es conocido por su capacidad de manejar un alto volumen de conexiones simultáneas, lo que lo hace adecuado para sitios web de comercio electrónico con un tráfico creciente.

Seguridad: Con un enfoque adecuado en la seguridad, podemos garantizar que las transacciones en línea y los datos de los usuarios estén protegidos.

Flexibilidad en la elección de bases de datos: JavaScript nos permite elegir entre una variedad de bases de datos, ya sea una base de datos relacional o NoSQL, según nuestras necesidades y preferencias.

Alcances:

Gestión de productos: Poder administrar eficientemente una amplia variedad de productos, descripciones, precios y categorías en nuestro ecommerce.

Procesamiento de pedidos: Lograr automatizar el proceso de procesamiento de pedidos, desde la selección de productos hasta la confirmación y envío.

Gestión de usuarios y autenticación: Poder implementar un sistema de autenticación seguro para que los usuarios puedan crear cuentas, iniciar sesión y realizar un seguimiento de sus pedidos.

Pasarelas de pago: Integrar pasarelas de pago seguras y confiables para procesar transacciones en línea.

Seguimiento de inventario: Realizar seguimientos del inventario en tiempo real y gestiona el stock de productos.

Análisis y métricas: Implementar herramientas de análisis para rastrear el rendimiento de la tienda, las conversiones y el comportamiento de los usuarios.

Personalización y recomendaciones: Poder personalizar la experiencia de compra de los usuarios y ofrecer recomendaciones de productos relevantes.

- **Enlace del repositorio:** <https://github.com/lisalvarez18/proyecto-ecommerce>
- Estructura básica del proyecto node js la cual fue creada usando el administrador de paquetes NPM.

```

1 {
2   "name": "proyecto-ecommerce",
3   "version": "1.0.0",
4   "description": "Proyecto de comercio electrónico",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "repository": {
10    "type": "git",
11    "url": "git+https://github.com/lisalvarez18/proyecto-ecommerce.git"
12  },
13  "author": "",
14  "license": "ISC",
15  "bugs": {
16    "url": "https://github.com/lisalvarez18/proyecto-ecommerce/issues"
17  },
18  "homepage": "https://github.com/lisalvarez18/proyecto-ecommerce#readme"
19 }
20

```

- Hola mundo en node js usando un servidor http.

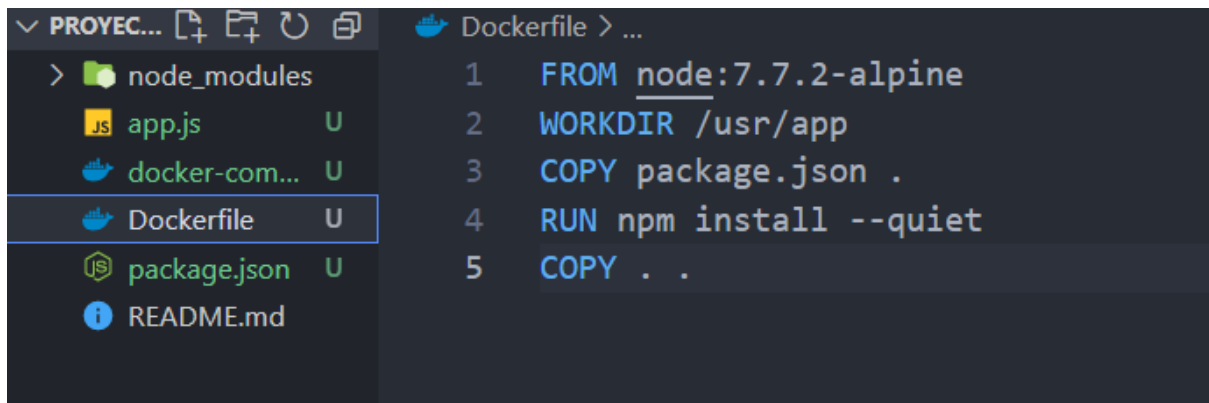
```

> node_modules
  app.js U
  docker-com... U
  Dockerfile U
  package.json U
  README.md

1
2 var http = require('http');
3 var server = http.createServer();
4
5 function mensaje(petic, resp) {
6   resp.writeHead(200, {'content-type': 'text/plain'});
7   resp.write('Hola Mundo');
8   resp.end();
9 }
10 server.on('request', mensaje);
11
12 server.listen(3000, function () {
13   console.log('La Aplicación está corriendo en el puerto 3000');
14 });

```

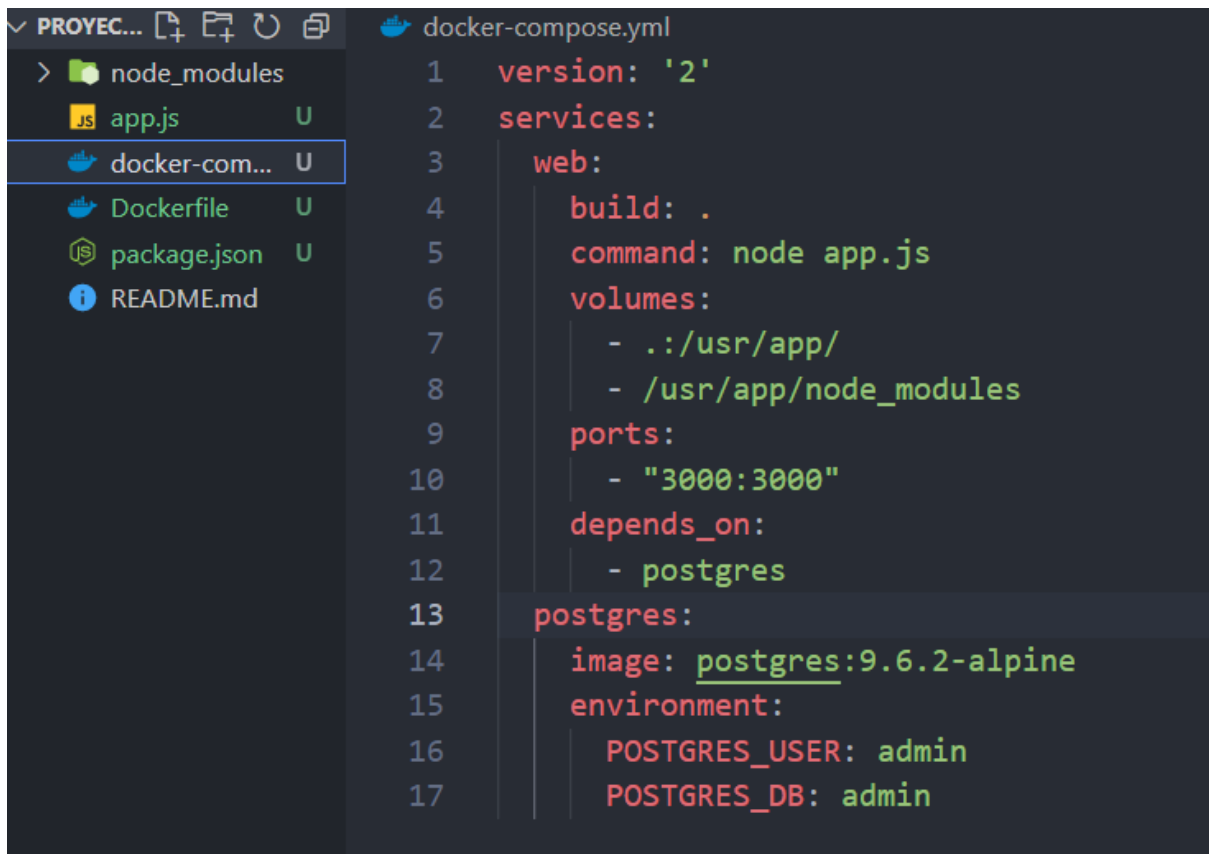
- Este código se usa para crear una imagen de la aplicación.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PROYEC...' with a folder 'node_modules' and files 'app.js', 'docker-com...', 'Dockerfile', 'package.json', and 'README.md'. The 'Dockerfile' file is selected. The code editor shows the content of the Dockerfile, which is a Dockerfile for a Node.js application. The code is as follows:

```
1 FROM node:7.7.2-alpine
2 WORKDIR /usr/app
3 COPY package.json .
4 RUN npm install --quiet
5 COPY . .
```

- Este código se usa para levantar dos contenedores, uno basado en la imagen de nuestra aplicación y el otro basado en la imagen de la base de datos de postgres.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PROYEC...' with a folder 'node_modules' and files 'app.js', 'docker-com...', 'Dockerfile', 'package.json', and 'README.md'. The 'docker-com...' file is selected. The code editor shows the content of the docker-compose.yml file, which is a docker-compose file for a Node.js application. The code is as follows:

```
1 version: '2'
2 services:
3   web:
4     build: .
5     command: node app.js
6     volumes:
7       - ../usr/app/
8       - /usr/app/node_modules
9     ports:
10      - "3000:3000"
11     depends_on:
12      - postgres
13   postgres:
14     image: postgres:9.6.2-alpine
15     environment:
16       POSTGRES_USER: admin
17       POSTGRES_DB: admin
```