Sprawozdanie z Przygotowania Środowiska Deweloperskiego

Dominik Prabucki, Dominik Purgał

7 stycznia 2024

1 Wstęp

Projekt ten miał na celu przygotowanie środowiska developerskiego dla aplikacji opartej na frameworku Laravel przy użyciu technologii Docker. Główne składowe projektu obejmują obraz PHP, serwer baz danych (MySQL), serwer WWW (Nginx), oraz Composer do zarządzania zależnościami PHP. Dodatkowo w skład środowiska wchodzi również skonfigurowany obraz z nodejs i npm do pracy nad frontend aplikacji, oraz system kolejek Redis.

2 Instrukcja uruchomienia projektu

2.1 Kroki do uruchomienia projektu

Aby uruchomić projekt, postępuj zgodnie z poniższymi krokami:

1. Zainstaluj Docker na swoim systemie.
2. Sklonuj repozytorim git

   ```
   git clone https://github.com/AmonDeShir/CN-Project
   ```

3. Otwórz terminal i przejdź do katalogu, w którym znajduje się plik docker-compose.yml.
4. Zbuduj obrazy dockera

   ```
   docker compose build --no-cache --pull
   ```

5. Stwórz plik .env i dostosuj jego konfiguracje.

   ```
   cp .env.example .env
   ```

6. Uruchom projekt przy użyciu komendy:

```
docker compose up -d
```

7. Po zakończeniu procesu uruchomienia, projekt będzie dostępny pod adresem http://localhost.

2.2 Dostęp do konsoli PHP w kontenerze

Aby uzyskać dostęp do konsoli PHP wewnątrz kontenera, wykonaj następujące kroki:

1. Otwórz terminal.
2. Uruchom poniższą komendę:

```
docker compose exec -it php bash
```

3. Teraz jesteś wewnątrz kontenera PHP i możesz korzystać z narzędzi, takich jak Composer.

2.3 Dostęp do konsoli NodeJS w kontenerze

Aby uzyskać dostęp do konsoli NodeJS wewnątrz kontenera, wykonaj następujące kroki:

1. Otwórz terminal.
2. Uruchom poniższą komendę:

```
docker compose exec -it node bash
```

3. Teraz jesteś wewnątrz kontenera node i możesz korzystać z narzędzi, takich jak npm.

2.4 Uruchomienie serwera vite

Aby uruchomić serwer vite, wykonaj następujące kroki:

1. Otwórz terminal.
2. Uruchom poniższe komendy:

```
docker compose exec -it node bash
npm run dev
```

3. Po zakończeniu procesu uruchomienia, serwer będzie dostępny pod adresem http://localhost:5173/.

3 Opis pliku docker-compose.yml

Plik docker-compose składa się z pięciu usług (web, php, database, redis, node), oraz z konfiguracji woluminów przechowujących dane bazy i redis oraz jednej wspólnej sieci dla całego środowiska.

4 Opis poszczególnych usług

4.1 PHP (php)

Ta usługa korzysta z własnego obrazu opartego na obrazie php:8.2-fpm-alpine3.16 i jest odpowiedzialna za serwer PHP oraz instalacje composer'a. Pliki projektu są montowane do wewnętrznego katalogu /app kontenera. Poza instalacją composer'a dodatkowo obraz jest wzbogacony o PDO dla postgres sql oraz zależności wymagane przez Laravel'a. Obraz podczas startu uruchamia plik ./docker/php/entrypoint.sh, który jest odpowiedzialny za instalacje predis oraz uruchomienie serwera php. Usługa czeka z uruchomieniem na baze danych i serwer redis.

4.2 Nginx (web)

Usługa web korzysta z obrazu nginx:1.25.3-alpine i obsługuje serwer WWW. Port 80 jest mapowany na port 80 w kontenerze, a pliki konfiguracyjne Nginx są montowane z /docker/nginx/config.conf

4.3 Postgres (database)

Usługa database używa obrazu postgres:16.1-alpine3.19 i jest używana jako baza danych projektu. Konfiguracja bazy danych jest wczytywana ze zmiennych środowiskowych (plik .env). Dodatkowo baza danych zapisuje swoje dane na osobnych woluminie db_data. Usługa posada również healthcheck pozwalający sprawdzić, czy baza danych jest już gotowa do pracy.

4.4 System kolejek redis (redis)

Ta usługa wykorzystuje obraz redis:7.2.3-alpine jest odpowiedzialna za przygotowanie systemu kolejek. Konfiguracja redis jest wczytywana ze zmiennych środowiskowych. Dane redis są przechowywane na woluminie redis_data. Dostępny jest również healthcheck pozwalający sprawdzić, czy redis jest już gotowy do pracy.

4.5 NodeJS (node)

Ta usługa korzysta z własnego obrazu opartego na obrazie node:20-alpine i jest odpowiedzialna za przygotowanie środowiska nodejs do pracy nad frontend'em. Pliki projektu są montowane do wewnętrznego katalogu /app kontenera. Obraz

podczas startu uruchamia plik ./docker/node/entrypoint.sh, który jest odpowiedzialny za instalacje wszystkich potrzebnych bibliotek js (między innymi vite) oraz uruchomienie domyślnego entrypoint'a obrazu node:20-alpine.

5 Podsumowanie

Podczas pracy nad projektem udało nam się skonfigurować środowisko Docker, które pozwala na uruchomienie projektu z PHP, Nginx, PostgresSQL, Nodejs i Redis.

5.1 Napotkane problemy

Podczas pracy napotkaliśmy na problem z uruchomianiem entrypoint'ów, przez nasze obrazy, dodanie polecenia chmod +x rozwiązało problem. Na podobny problem napotkaliśmy podczas testów serwera php, który nie chciał wczytywać plików, okazało się że problem spowodowany był przez brak praw uzytkownika www-data do katalogu z plikami aplikacji "/app/storage". Problem rozwiązało dodanie komendy chown -R www-data:www-data /app/storage do entrypoint.sh obrazu php.

5.2 Co można było zrobić inaczej?

Można by zmienić baze danych na łatwiejszą w konfiguracji, na przykład na MariaDB. Nginx można zmienić na Apache. Dodatkowo można zastąpić plik entrypoint.sh odpowiednio przygotowanym plikiem Makefile. Same obrazy php i nodejs można by wysłać na DockerHub i wyeliminować w ten sposób czasochłonny proces budowania obrazów.