Projekt_RIP - API

Jakub Engielski
Jan Kwiatkowski

Styczeń 2023

1 Opis funkcjonalny systemu

Celem "Projektu_RIP - API" było utworzenie aplikacji do pobrania danych z planu uczelni Collegium Witelona. Wraz z tym dochodzi także część formatująca, przystosowująca pobrane dane w ten sposób aby aplikacja web oraz aplikacja mobilna byłyby w stanie bez problemu się nimi obsługiwać.
Api będzie zgodnie z ustalonym harmonogramem, np. raz dziennie pobierało dane z planu uczelni. Wykonywało swoje funkcje, oraz wystawiało już sformatowane i posegregowane dane przy pomocy rest api.
Endpointy wykonane są zgodnie z standardami swaggera.

2 Streszczenie opisu technologicznego

Korzystamy z framework'u Symfony 6.1 wspierającego język PHP, dzięki któremy jesteśmy w stanie postawić ten projekt bez potrzeby pisania wielu rzeczy od nowa. Wybraliśmy Symfony gdyż jesteśmy z nim najbardziej zapoznani.
Docker umożliwia nam łatwe pakowanie, dostarczanie oraz uruchamianie aplikacji w formie lekkiego, przenośnego i samowystarczalnego kontenera. Co pozwala nam to uruchomić szybko i praktycznie wszędzie.
Api Platform jest paczką, która pozwala na wygodne i szybkie wystawienie rest api. Bez tego prace zostały by wydłużone bez żadnego dobrego powodu.
Goutte jest paczką pozwalającą nam na postawienie samego scrapera. Jest to niezbędna część tego projektu, bez niej postawienie scrapera w php byłoby o wiele bardziej obciażające jeśli chodzi o moc obliczeniową (Symfony Panther) albo wymagało by olbrzymiej pracy na około aby obejść prostotę systemu (Guzzle)
Doctrine jest paczką pozwalającą nam w wygodnu sposób pracować nad entity. Dzięki czemu jesteśmy w stanie wyprowadzić poprawne dane w formie api.
Bazą danych jaka jest wykorzystawna jest Postgres. Jest to zaawansowany system do zarządania relacyjnymi bazami danych, do tego jest to projekt open source.

3 Instrukcję lokalnego i zdalnego uruchomienia systemu

3.1 Postawienie systemu lokalnie

Wymagane oprogramowanie:
PHPStorm lub dowolny inny IDE
Docker Desktop
Terminal Windows (nie jest potrzebny osobno jeśli jest wbudowany w IDE jak w PHPStorm)
Github Desktop – aby móc wprowadzać zmiany, bądź pobierać aktualizacje jeśli są potrzebne. (nie jest wymagany jeśli jest wbudowany w IDE, bądź jeśli ktoś posiada zainstalowany pakiet GIT do użytku poprzez terminal)

Jak postawić środowisko testowe na dockerze?
Instrukcja dla Windowsa
Instrukcja dla Linuxa

Jeżeli wszystko zainstalowałeś, przejdź do instrukcji poniżej:

1. Pobranie projektu z repozytorium oraz przejście do jego folderu:

   git clone https://github.com/sbacanski0730/RIP-Rewak-and-PUM-API.git
   cd RIP-Rewak-and-PUM-API

2. Konfiguracja pliku .env

   cp .env .env.local

3. Uruchomienie kontenerów dockerowych:

   docker-compose up -d –build

4. Pobranie zależności Composer:

   docker-compose exec php composer install

5. Pobranie zależności npm:

   docker-compose exec php npm install

6. Budowanie Assetów:

docker-compose exec php npm run dev

Migracje:

1. Utworzenie migracji:

   php bin/console doctrine:migrations:diff
   php bin/console doctrine:migrations:migrate

Bez migracji:

1. Utworzenie schematu bazy danych:

   php bin/console doctrine:schema:update –force

2. Załadowanie zdefiniowanych wczesniej danych do bazy danych:

   php bin/console doctrine:fixtures:load

Usługi:

- api https://localhost:443/api
- postgres localhost:5432

Przydatne Komendy:

1. Docker - uruchomienie kontenerów:

   docker-compose up -d

2. Docker - zatrzymanie kontenerów:

   docker-compose stop

3. Docker - zatrzymanie i usunięcie kontenerów

   docker-compose down

4 Dokumentacja

Link do dokumentacji na naszych repozytoriach:
api: https://github.com/sbacanski0730/RIP-Rewak-and-PUM-API/tree/main/documentation

web: https://github.com/sbacanski0730/RIP-Rewak-and-PUM-Web/tree/main/documentation

mobile: https://github.com/sbacanski0730/RIP-Rewak-and-PUM-Mobile/tree/main/documentation

5 Wnioski projektowe

Pisanie pierwszego scrapera z api może być wyzwaniem. Tak było też w tym wypadku. Zwłaszcza, że z błędnych założeń chcieliśmy na początku napisać całość bez używania bazy danych. Spowodowało to wiele opóźnień i problemów. To oraz inne problemy związane z sprzętem komputerowym jednego członka zespołu opóźniły rozpoczęcie pisania tej części projektu.
Patrząc na to z obecnej perspektywy zamiast próbować tworzyć nowe rozwiązania samemu zamiast skorzystać z gotowej dokumentacji nie było najlepszym wyborem. Podeszliśmy do tego myśląc, że pozwoli to na optymalizację projektu. Pomimo tego projekt wymagał abyśmy szukali wiedzy poza standardową dokumentacją. Gdyż nie wszystko to co było potrzebne w naszym wypadku było w niej dostępne. Więc poza pisaniem kodu oraz korzystaniem z dokumentacji musieliśmy dokonać własnego zapoznania się z tematem.
Wiele w scraperze zależy od kodu na stronie, z której informacje są pobierane. Jak mogliśmy się przekonać z planem naszej uczelni, nie wszystkie strony posiadają klasy oraz kod zgodne z dobrymi praktykami. Więc wydobywanie informacji z planu wymagało trochę twórczego myślenia związanego z zapytaniami regex oraz odpowiednimi algorytmami.