# Sprawozdanie

Projekt Angular + FastAPI + postgresql + docker

Adam Zajler 131892

## Inicjalizacja docker-compose

W tym kroku został dodany plik **docker-compose.yml** którego zadaniem jest spięcie wszystkich pomniejszych kontenerów i usystematyzowanie ich pracy.

Zadeklarowane zostały 4 serwisy:

1. Web: jest to projekt w Angularze
2. Api: jest to FastAPI
3. Postgres: baza danych
4. Adminer: mały klient do obsługi bazy danych; na wzór trochę phpMyAdmin

Jest to standardowa konfiguracja zaczerpnięta z oficjalnych dokumentacji i gotowych przykładów ww. technologii.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Dodanie adminera

Adminer jest prostą usługą i nie wymagał żadnej konfiguracji; jedynie należało go zadeklarować w **docker-compose.yml** i poprawnie zalogować się do usługi.

Dane do logowania są takie same jak w pliku **docker-compose.yml**. Jedynie nazwa serwera jest tym samym co nazwa serwisu w **docker-compose.yml** czyli w tym przypadku **postgres**.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

## Dodanie fastapi

FastAPI w tej aplikacji służy jako pośrednik między klientem (Angularem) a bazą danych (postgres). Problemy jakie wystąpiły po drodze to m.in.: tworzenie tabeli gdy użytkownik pierwszy raz inicjuje kontenery, CORS z clienta, braki w dokumentacji (należało się substytuować poradnikami na Youtube oraz modelami językowymi).

Poniższy kod m.in.: dodaje CORS-a co pozwala na odpytywanie API z wszystkich klientów (niezalecane, jednakże na rzecz prostego projektu do przyjęcia), proste połącznie z postgresem (identyczne jak w adminer), funkcja która wywołuje się na evencie **@app.on\_enev(„startup”)** i warunkowo tworzy domyślną tabele, zapis do BD który wpierw czyści wszystkie rekordy i potem zapisuje podane dane (wszystko jest czyszczone ponieważ nigdy nie będzie nowszych rekordów niż te wprowadzane), finalnie na koniec za pomocą **@app.get** oraz **@app.post** wystawiane są endpointy **/data/** z możliwością wykonywania zapytań **POST** i **GET**.  
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

## Dodanie projektu w Angular

Projekt został domyślnie zainicjowany oraz to samo tyczy się serwisu **nbp.service**.

Serwis **nbp.service.ts**: służy do komunikacji z API. W ramach tego pliku powstały rzeczy jak

- avilableCurrencies: stała która przetrzymuje dostępne waluty. Zmiany w jej obrębie determinują to co znajduje się w BD oraz co będzie synchronizowane  
Obraz zawierający tekst, Oprogramowanie multimedialne, oprogramowanie, zrzut ekranu

Opis wygenerowany automatycznie  
- getRates: pobiera dane z API NBP, parametry jakie są wymagane to zakres czasu (start-koniec) oraz waluta np. EUR   
Obraz zawierający zrzut ekranu, tekst, Czcionka

Opis wygenerowany automatycznie  
- synchronizeAllRatesInDB: funkcja ta pobiera wszystkie dostępne waluty (zadeklarowane w **avilableCurrencies**) z API NBP, po czym ekstraktuje je na typ które obsłguiwany przez BD i finalnie dane są wysyłane do BD. Trudnością tutaj był **callback function** który wywołuje się zawsze, ale służy do obsługi przypadku braku danej waluty w BD, dzięki temu nowo zaciągnięte dane są od razu przekazywane do parsowania na front i nie trzeba czekać na pobranie danych z BD.  
Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie  
- getRatesFromDB: pobiera dany rekord waluty z BD  
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

- postRatesToDB: zapisuje dany rekord waluty do BD  
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie  
Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Komponent stron głównej. Tutaj przy uruchomieniu (**ngOnInit**)są pobierane waluty z BD. Dodatkowo jest funkcja **synchronizeRates** która wymusza wgranie do BD nowych danych z API NBP, po czym są one od razu przekazywane do **parseRates** które są odpowiedzialne za wyświetlenie ich na froncie. **ParseRates** zajmuje się również dzieleniem danych z BD na mniejsze fragmenty, jak ostatni dzień, ostatni miesiąc, ostatni kwartał.  
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Kod HTML odpowiedzialny za wyświetlanie jest bardzo prosty, i jedynie obsługuje wywołanie synchronizacji oraz wyświetlanie danych lub ewentualnie błędu w przypadku ich braku.  
Obraz zawierający tekst, elektronika, zrzut ekranu, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Po wybraniu USD ( brak danych w BD )

Obraz zawierający tekst, zrzut ekranu, Czcionka, algebra

Opis wygenerowany automatycznie

Po kliknięciu „synchronizuj dane”

Obraz zawierający tekst, zrzut ekranu, numer

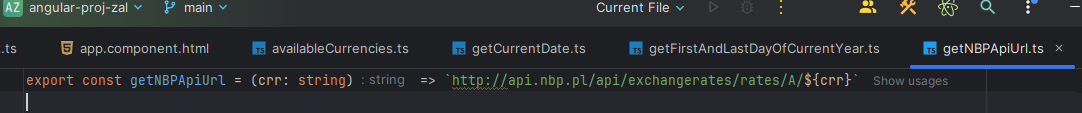
Opis wygenerowany automatycznie

W trakcie pracy nad aplikacją powstały funkcje pomocnicze jak **getCurrentDate.ts**, **getNBPApiUrl**, **getFirstAndLastDayOfCurrentYear**Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie



## Testy dla aplikacji