AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Fizyki i Informatyki Stosowanej

KATEDRA INFORMATYKI STOSOWANEJ I FIZYKI KOMPUTEROWEJ

Praca dyplomowa

System do akwizycji danych z rozproszonych systemów pomiarowych
System for data acquisition from distributed measurement systems

Autor: Mateusz Barnacki
Kierunek studiów: Informatyka Stosowana
Opiekun pracy: dr inż. Antoni Dydejczyk

Kraków, 2023r.

Spis treści

1. Wstęp

2. Przegląd wzorców architektury aplikacji

3. Elementy charakterystyczne dla architektury mikrousługowej

4. Zapewnienie jakości w projekcie IT

5. Architektura aplikacji

6. Technologie wykorzystane w projekcie

7. Dokumentacja komponentów

7.1. Wdrożenie aplikacji na serwer wydziału

Jednym z problemów jakie należało rozwiązać w trakcie projektowania aplikacji było znalezienie sposobu na względnie szybkie zbudowanie i uruchomienie systemu na dowolnym środowisku. Istotnymi elementami przy wyborze odpowiedniego narzędzia było uwzględnienie popularności wykorzystania danej technologii przez środowisko programistów oraz przejrzyście napisana dokumentacja. Ponadto ze względów formalnych narzędzie musi być darmowe.

Uwzględniając wyżej wymienione kryteria autor pracy podjął decyzję o wykorzystaniu technologii konteneryzacji przy użyciu narzędzia Docker. Każdy kontener zawiera w sobie wszystkie potrzebne zależności do zbudowania oraz uruchomienia danego komponentu aplikacji. Kontenery są tworzone na podstawie obrazów, które w terminologii Dockera oznaczają niezmienne szablony definiujące reguły budowania kontenerów. Obrazy są definiowane za pomocą instrukcji zawartych w specjalnych plikach o nazwie Dockerfile.

W przypadku zastosowania architektury mikrousługowej uruchomienie programu wymaga stworzenia większej ilości kontenerów. Narzędziem dedykowanym do zarządzania rozbudowaną infrastrukturą aplikacji jest Docker Compose, który umożliwia zbudowanie i uruchomienie programu przy użyciu jednej komendy. Reguły budowania projektu są umieszczone w pliku o nazwie *docker-compose.yml*. W ramach budowania wielokontenerowej aplikacji można ustalić m.in. kolejność budowania oraz uruchamiania kontenerów, lokalizacje plików Dockerfile na podstawie których budowane są kontenery, lokalizację pliku zawierającego kopię zapasową dla baz danych.

Ważnym aspektem związanym z wykorzystaniem technologii konteneryzacji jest oddzielenie kontenera od środowiska zewnętrznego. Aplikacja działa w taki sam sposób na dowolnym systemie operacyjnym lub maszynie wirtualnej. Wobec tego można przetestować program na lokalnej maszynie nie wpływając na działanie systemu na środowisku produkcyjnym. W przypadku naprawy błędu wystarczy zreprodukować dane wejściowe, które spowodowały błąd i wprowadzić konieczne poprawki.

W celu uruchomienia aplikacji wymagana jest instalacja na docelowym urządzeniu systemu kontroli wersji Git oraz wyżej opisanego narzędzia Docker. System kontroli wersji Git służy do stworzenia lokalnej kopii zdalnego repozytorium. Jeżeli aplikacja jest uruchamiana na systemie operacyjnym Windows należy zainstalować dodatek o nazwie Git Bash. Uzasadnienie wykorzystania konsoli Git Bash nastąpi w kolejnej sekcji. Narzędzie Docker umożliwia automatyczne zbudowania oraz uruchomienia aplikacji.

W momencie pisania niniejszej pracy kod źródłowy programu jest umieszczony na platformie GitHub. W celu zabezpieczenia wrażliwych informacji dotyczących sposobu budowania API-Key oraz JWT repozytorium zostało oznaczone jako prywatne. Wydział udostępnił maszynę wirtualną wykorzystującą system operacyjny Rocky Linux. Maszyna znajduje się pod adresem *172.20.40.211*. W celu uruchomienia systemu zainstalowałem wszystkie potrzebne aplikacje wymienione powyżej. Działanie aplikacji można sprawdzić tylko i wyłącznie będąc zalogowanym do sieci wydziału. Strona logowania do systemu znajduje się pod adresem http://172.20.40.211:3000.

7.2. Pierwsze uruchomienie systemu

Poniższa instrukcja przedstawia wykorzystanie komend w dowolnym wierszu poleceń. Część niżej wymienionych operacji może zostać zastąpiona poprzez wykorzystanie narzędzi posiadających graficzny interfejs użytkownika takich jak IntelliJ IDEA lub Docker Desktop.

Pierwszym krokiem wymaganym do uruchomienia systemu na dowolnej maszynie jest utworzenie lokalnej kopii repozytorium za pomocą komendy *git clone*. Aplikacja do wygenerowania JWT wykorzystuje parę asymetrycznych kluczy służących do kodowania oraz dekodowania żetonów. W związku z tym do prawidłowego działania mechanizmu autoryzacji wymagane jest utworzenie klucza publicznego oraz klucza prywatnego. W tym celu trzeba przenieść się do lokalizacji *master-thesis/auth-service/src/main/resources*. Następnie administrator musi utworzyć nowy katalog o nazwie *certs*. W katalogu *certs* należy wykonać poniższą sekwencję komend.

```
1  openssl genrsa -out keypair.pem 2048
2  openssl rsa -in keypair.pem -pubout -out public.pem
3  openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in keypair.pem -out private.pem
```

Listing 1 Utworzenie klucza publicznego oraz klucza prywatnego

Biblioteka OpenSSL jest dostępna na systemach operacyjnych MacOS oraz Linux. W przypadku systemu operacyjnego Windows rekomendowane jest zainstalowanie konsoli Git Bash, która jest dodatkiem przy instalacji systemu kontroli wersji Git i zawiera wbudowaną wersję biblioteki OpenSSL. Pierwsza linia przedstawionego powyżej skryptu generuje klucz prywatny za pomocą algorytmu RSA o rozmiarze 2048 bitów, a następnie zapisuje go do pliku *keypair.pem*. Druga instrukcja procesuje wygenerowany klucz prywatny i generuje na jego podstawie klucz publiczny zapisując go do pliku *public.pem*. Ostatnia komenda tworzy klucz prywatny na podstawie danych z pliku *keypair.pem* w formacie PKCS8 i zapisuje go do pliku *private.pem*. Opcja *-nocrypt* została użyta ze względu na implementacje obsługi JWT we frameworku Spring Boot Security. Po utworzeniu plików *public.pem* oraz *private.pem* można usunąć plik *keypair.pem* oraz skopiować cały katalog do folderu *master-thesis/api-gateway/src/main/resources*.

Kolejnym krokiem jest wykorzystanie narzędzia Docker Compose. W tym celu należy przejść do folderu *master-thesis/* i wykonać poniższą komendę.

```
1  docker compose up -d
```

Listing 2 Zbudowanie oraz uruchomienie aplikacji

Przedstawiona komenda umożliwia wygenerowanie obrazów Dockera. Następnie na podstawie obrazów tworzone są kontenery zawierające poszczególne komponenty aplikacji. Opcja *-d* umożliwia wyłączenie śledzenia logów aplikacji po zakończeniu procesu budowania kontenerów.

Aplikacja po pierwszym uruchomieniu zawiera puste bazy danych. Dane logowania dla utworzonych użytkowników znajdują się w pliku *docker-compose.yml*. Utworzenie schematu bazy danych wymaga wykorzystania narzędzia konsolowego *psql*. Poniższa komenda umożliwi uruchomienie wiersza poleceń PostgreSQL.

```
docker exec -it master-thesis-postgres-1 psql -U postgres
```

Listing 3 Uruchomienie wiersza poleceń dla użytkownika postgres

Z perspektywy konsoli PostgreSQL wprowadzanie zapytań odbywa się w kontekście użytkownika *postgres*. Baza danych musi nazywać się *auth*. Kolejnym elementem jest przejście do konsoli PostgreSQL w kontekście nowoutworzonej bazy danych.

```
docker exec -it master-thesis-postgres-1 psql -U postgres -W auth
```

Listing 4 Uruchomienie wiersza poleceń dla bazy danych auth

Do utworzenia schematu bazy relacyjnej należy wykorzystać skrypt, który zawiera polecenia tworzące tabele oraz polecenia wstawiające podstawowe encje. Skrypt znajduje się w pliku *master-thesis/auth-service/src/main/resources/db-scripts/init_db.sql*.

Ostatnim elementem koniecznym do rozpoczęcia korzystania z aplikacji jest utworzenie pierwszego konta użytkownika o uprawnieniach administratora. Użytkownik ma przypisaną rolę administratora, ponieważ jest to jedyna rola, która daje uprawnienie do stworzenia nowego konta z poziomu klienta aplikacji. Serwis służący do autoryzacji podczas tworzenia nowego konta koduje hasło za pomocą funkcji haszującej o nazwie *bcrypt*. W konsekwencji nie można dodać nowego użytkownika z poziomu bazy danych, ponieważ podczas operacji logowania niezakodowane hasło znajdujące się w bazie będzie się różniło od hasła przetworzonego przez serwer aplikacji. Jedynym sposobem dodania nowego utworzenia z poziomu wiersza poleceń jest wykorzystanie biblioteki *cURL*.

```
curl -d
'{"username": "Admin",
  "email": "admin@agh.edu.pl",
  "description": "Admin user",
  "password": "admin",
  "roles": ["ADMIN"],
  "projects": []}'
-H "Content-Type: application/json"
-X POST http://localhost:8080/users
```

Listing 5 Utworzenie konta użytkownika przy użyciu biblioteki cURL

Powyższe polecenie wysyła żądanie HTTP typu POST. Do wysłanego żądania dołączony jest obiekt JSON (JavaScript Object Notation). W ciele obiektu JSON znajdują się dane potrzebne do stworzenia nowego konta. Alternatywnym sposobem wysłania powyższego zapytania jest wykorzystanie narzędzia o nazwie Postman. Postman udostępnia graficzny interfejs użytkownika, który znacząco ułatwia zmiany parametrów wykonywanych zapytań.

Po utworzeniu konta użytkownika można zalogować się do systemu działającego pod adresem *http://172.20.40.211:3000*.

7.3. Wdrażanie zmian w systemie

Jednym z elementów cyklu życia oprogramowania jest utrzymanie aplikacji. Przytoczony termin może być rozumiany jako naprawa błędów lub wdrożenie nowych funkcjonalności. Moment wprowadzania nowej wersji kodu źródłowego na serwerze wydziału musi wiązać się z zatrzymaniem dotychczas działającej aplikacji. W przypadku komponentu działającego w obrębie kontenera operacja wymaga przeprocesowania następującej sekwencji komend.

```
1  git pull origin main
2  docker stop auth-service
3  docker rm auth-service
4  docker rmi auth-service
5  docker compose up -d
```

Listing 6 Aktualizacja aplikacji działającej w kontenerze

Powyższy skrypt stanowi przykład aktualizacji serwisu o nazwie auth-service. Pierwsza z wymienionych komend pobiera do lokalnego repozytorium zmiany wprowadzone w kodzie źródłowym. Kolejna instrukcja zatrzymuje działanie kontenera. Trzecie polecenie usuwa starą wersję kontenera. Komenda w czwartej linii usuwa stary obraz na podstawie którego zbudowany był wcześniejszy kontener. Ostatnia instrukcja sprawdza status działania kontenerów, które są wymienione w pliku *docker-compose.yml*. Jeżeli któryś z kontenerów nie prezentuje statusu „Running”, Docker tworzy na nowo obraz i na podstawie tego obrazu buduje kontener i uruchamia aplikację. Analogiczny zestaw komend może być wykorzystany dla dowolnego komponentu tworzącego aplikację. W celu sprawdzenia, czy nowa wersja aplikacji uruchomiła się w sposób prawidłowy można wykorzystać komendę *docker logs*.

7.4. Obsługa systemu

Podczas działania aplikacji mogą wystąpić błędy lub nieoczekiwane zachowania. W takich sytuacjach zadaniem administrator systemu jest sprawdzenie przyczyny błędu w logach aplikacji. Każdy serwis generuje własne logi niezależnie od działania pozostałych serwisów. Architektura monolityczna charakteryzująca się działaniem wszystkich funkcjonalności w ramach pojedynczego serwera aplikacyjnego. W związku z tym archiwizuje każde zdarzenie niezależnie od złożoności operacji wykonywanej przez użytkownika. W przypadku zastosowania architektury mikrousługowej odpowiedzialność za realizację złożonych operacji może zostać podzielona na kilka niezależnych serwisów. Zatem w sytuacji wystąpienia błędu administrator jest zmuszony do przejrzenia logów zdarzeń w każdym komponencie uczestniczącym w realizacji żądania.

Innym aspektem charakterystycznym dla mikroserwisów jest sposób wykorzystania funkcjonalności udostępnianych przez serwer w ramach graficznego interfejsu użytkownika będącego klientem aplikacji. Użytkownik posiadający rolę administratora podczas przeprowadzania operacji zmiany uprawnień konta nie wie, że lista projektów wyświetlona w oknie dialogowym nie pochodzi z serwisu zarządzania uprawnieniami, lecz z serwisu zarządzania czujnikami.

Wśród najczęściej spotykanych przyczyn problemów można wskazać zatrzymanie działania serwisu biorącego udział w operacji lub błąd w implementacji funkcjonalności. W celu weryfikacji działania kontenerów tworzących aplikację należy wykorzystać poniższą komendę.

```
1 | docker compose ps
```

Listing 7 Komenda wyświetlająca listę działających kontenerów

Przedstawiona komenda musi zostać wykonana z poziomu katalogu, w którym znajduje się plik *docker-comopose.yml*. Fragment wyniku działania polecenia został przedstawiony na poniższym zdjęciu.

NAME	IMAGE	CREATED	STATUS
api-gateway	api-gateway:latest	47 hours ago	Up 47 hours
auth-service	auth-service:latest	47 hours ago	Up 47 hours
client	client:latest	47 hours ago	Up 47 hours
master-thesis-mongodb-1	mongo:6.0.5	7 days ago	Up 7 days
master-thesis-postgres-1	postgres:15.3	7 days ago	Up 7 days
sensor-service	sensor-service:latest	43 hours ago	Up 43 hours
service-registry	service-registry:latest	5 days ago	Up 5 days

Zdjęcie 1 Fragment wyniku działania polecenia docker compose ps dla serwisu do akwizycji danych

W celu zapewnienia czytelności Zdjęcia 1 autor usunął kolumny COMMAND, SERVICE oraz PORTS. // Opis znaczenia poszczególnych kolumn.

W celu zweryfikowania działania poszczególnych serwisów można wykorzystać komendę docker logs. Wskazane polecenie wyświetla informacje zbierane przez poszczególne serwisy w trakcie ich działania.

Alternatywnym sposobem analizy działania całego systemu jest wykorzystanie mechanizmu service discovery. Mechanizm udostępnia panel administratora w formie graficznego interfejsu użytkownika dostępnego z poziomu przeglądarki internetowej. Wspomniana usługa została zaimplementowana w ramach niniejszej pracy dyplomowej. Dokładny opis wykorzystania mechanizmu zostanie przedstawiony w następnym rozdziale.

8. Podsumowanie

Bibliografia