Wymagania projektowe
Serwisy internetowe .NET

Prowadzący: Stanisław Baranski stanislaw.baranski@pg.edu.pl

Wprowadzenie

Celem projektu jest utworzenie demonstracyjnej aplikacji odczytującej komunikaty z systemu kolejkowego, które po konwersji zostają zapisane w bazie NoSQL. Zgromadzone informacje można przeglądać w formie tabelarycznej (z filtrami), pobierać (w formie CSV, JSON) oraz wizualizować w formie wykresów. Dane w systemie kolejkowym mają demonstrować monitorowanie urządzeń pomiarowych IoT tj. temperatury, wilgotności, stężenia gazów, szybkości, natężenia UV itp. Każdy zespół wybiera rodzaj monitorowanej infrastruktury np. rolnictwo (uprawa określonego produktu), filtrowanie gazów, produkcja pieczywa, proces produkcji leków, proces produkcji samochodów itd. Wybrane przez zespół zastosowanie aplikacji determinuje rodzaj wykorzystanych czujników (prezentowanych w aplikacji). Każdy zespół powinien zademonstrować działanie aplikacji dla symulowanej infrastruktury składającej się z czterech, różnych typów czujników oraz 16 instancji czujników wysyłających dane w różnych odstępach czasu. Dane mogą być generowane dowolnymi metodami (własna aplikacja, wykorzystanie gotowych narzędzi).

Wymagania względem aplikacji

1. *Aplikacja jest napisana w technologii ASP.NET Core 6+,
2. *Kody źródłowe wraz z dokumentacją są dostępne w publicznym repozytorium GitHub,
3. *Licencja w projekcie zdefiniowana jest jako MIT X11,
4. *Wszystkie komponenty aplikacji są skonteneryzowane i w repozytorium istnieje odpowiedni Dockerfile,
5. *Przygotowano poprawny plik docker-compose.yml, który umożliwia uruchomienie aplikacji na lokalnym docker host'cie.
6. *Wszystkie komponenty aplikacje zostały wdrożone na lokalnym Docker host'cie (tylko taka aplikacja jest oceniana przez prowadzącego),
7. *Kompozycja opisuje:
    1. *System kolejkowy wraz z konfiguracją,
    2. *Bazę danych MongoDB,
    3. *Utworzoną aplikację (Backend),
    4. *Utworzoną aplikację (Frontend) w technologii Razor Pages, lub dowolnej innej React.js/Angular/Vue.JS, itd.
8. Utworzona aplikacja Backend oferuje:
    1. *Automatyczne pobieranie danych z kolejek, konwersję danych i ich zapis w bazie danych;
    2. *Odczyt danych z bazy danych,
    3. *Filtrowanie i sortowanie danych na podstawie podanych kryteriów,
    4. Eksport danych (przefiltrowanych, sortowanych) w formacie CSV, JSON,
9. Utworzona aplikacja Frontend oferuje:
    1. *przeglądanie zgromadzonych danych w formie tabelarycznej,
    2. *filtrowanie danych według daty, typu czujnika (grupa instancji), instancji czujnika,
    3. *sortowanie danych w tabeli;
    4. pobieranie danych w formacie CSV, JSON dla wybranych filtrów;
    5. prezentacja danych w formie wykresów dla wybranych filtrów.

6. pulpit, na którym widać ostatnią wartość i średnią wartość (dla ostatnich 100 komunikatów) dla każdego sensora - prezentowane dane nie wymagają odświeżania strony, wartości odświeżają się samodzielnie po zarejestrowaniu nowej wartości **(+0,5 oceny).**

10. *Przygotować skrypt lub program, który umożliwi przeprowadzenie symulacji generowania danych przez min. 16 sensów (4 sensory dla każdego typu).
11. *Aplikacja wykorzystuje bazę danych MongoDB do zapisu danych odczytanych z kolejek,
12. *Aplikacja wykorzystuje brokera MQTT (np. Eclipse Mosquitto) do przesyłania danych z sensorów do aplikacji

*** - zadania obligatoryjne na ocenę 3,0.**

Wdrożenie aplikacji do demonstracji/testów

Przygotowaną aplikację należy uruchomić na dockerze na swoim komputerze. Uruchomienie aplikacji jest obligatoryjne.

Podczas prezentacji zespół studencki uruchamia przygotowany generator danych, który wysyła je bezpośrednio do systemu kolejkowego (MQTT). Generator powinien umożliwić zdefiniowanie:

- zakresu liczbowego dla poszczególnych sensorów,
- określonej szybkości generowania danych na minutę dla poszczególnych sensorów.

Proces oceny

1. W terminie oddawania na zajęcia stawia się pełen zespół.
2. Zespół przed rozpoczęciem prezentacji upewnia się, że wszystkie komponenty systemu są prawidłowo uruchomione w Dockerze.
3. Baza danych mongoDB jest oczyszczona z danych historycznych.
4. Zespół przygotowuje generator do uruchomienia.
5. Zespół otwiera przeglądarkę internetową i wywołuje adres Frontendu.
6. Na polecenie prowadzącego zespół uruchamia generator danych.
7. Prowadzący sprawdza widok tabelaryczny do prezentacji danych - sprawdza możliwość założenia filtra na typ sensora, na instancję sensora, na zakres dat rejestracji danych.
8. Prowadzący sprawdza widok tabelaryczny do prezentacji danych - sprawdza możliwość sortowania danych w tabeli (z/bez włączonych filtrów).
9. Po zakończeniu pierwszego procesu generowania danych, prowadzący prosi o wygenerowanie określonej wartości liczbowej dla jednego ze wskazanych sensorów.
10. Prowadzący sprawdza widok tabelaryczny czy nowe dane są wyświetlane.
11. Jeśli zespół przygotował pulpit (wymaganie 9.6.) to prowadzący przechodzi do pulpitu i prosi o wygenerowanie określonych wartości przez wybrany sensor (sprawdza w ten sposób czy wyświetlane dane zostaną prawidłowo zaktualizowane) - zadanie jest uznane za zrealizowane jeśli czas opóźnienia nie przekracza 1 sekundy od momentu wygenerowania danych.

Projekty spełniające wszystkie wymagania oznaczone * otrzymują ocenę 3,0
Projekty spełniające wszystkie wymagania oznaczone * oraz 1 dodatkowe otrzymują ocenę 3,5
Projekty spełniające wszystkie wymagania oznaczone * oraz 2 dodatkowe otrzymują ocenę 4
Projekty spełniające wszystkie wymagania oznaczone * oraz 3 dodatkowe otrzymują ocenę 4,5
Projekty spełniające wszystkie wymagania otrzymują ocenę 5,0 (przy spełnieniu wymagania 9.6.)
Projekty niespełniające dowolnego wymagania oznaczonego * otrzymują ocenę 2,0.

Powodzenia! 🧑‍🔧

Opcjonalny moduł blockchainowy

Studenci mogą uzyskać zaliczenie przedmiotu bez przystępowania do egzaminu końcowego, jeśli zdobędą maksymalną liczbę punktów z poprzedniej części a dodatkowo zdecydują się na implementację poniższego rozszerzenia, obejmującego integrację z blockchainem oraz utworzenie smart contract'u typu ERC-20.

Cel dodatkowego modułu

Rozszerzenie funkcjonalności aplikacji o blockchain, gdzie każde urządzenie (sensor) jest nagradzane tokenami ERC-20 za każdą przesłaną wiadomość. Smart contract będzie zarządzał emisją i dystrybucją tokenów do portfeli sensorów, a administrator systemu będzie mógł przeglądać aktualny stan posiadania tokenów przez każdy sensor.

Wymagania:

1. **Implementacja smart contractu ERC-20**:
Każdy zespół musi zaprogramować i wdrożyć smart contract zgodny z ERC-20, który będzie pełnił funkcję tokenu nagradzającego sensory.
2. **Nagradzanie sensorów tokenami**:
Za każdym razem, gdy sensor wyśle komunikat do systemu, smart contract automatycznie przesyła określoną ilość tokenów na adres portfela przypisanego do tego sensora.
3. **Przegląd tokenów**:
Utworzyć interfejs (frontend), który umożliwia administratorowi systemu przegląd aktualnego stanu posiadania tokenów przez każdy sensor (jego adres portfela).
4. **Wybór blockchaina**:
Studenci mogą wykorzystać dowolny blockchain wspierający tworzenie tokenów, taki jak Ethereum, Solana, Base, czy Arbitrum. Aby uniknąć kosztów związanych z opłatami transakcyjnymi, zaleca się korzystanie z testowych sieci blockchainowych (np. Rinkeby dla Ethereum, Solana Devnet itp.), lub lokalnych instancji dewerloperskich (Foundry Anvil, Hardhat lub thirdweb dla Ethereum)
5. **Automatyczna integracja z istniejącą aplikacją**:
Aplikacja powinna zintegrować się z blockchainem, wysyłając odpowiednie transakcje do smart contractu w momencie, gdy sensory przesyłają dane do systemu.

Proces oceny:

1. **Poprawność smart contractu**:
Sprawdzenie, czy smart contract jest zgodny ze standardem ERC-20, poprawnie nagradza sensory za każdą przesłaną wiadomość oraz czy można przeglądać stan tokenów każdego sensora.

2. **Integracja z aplikacją**:
Sprawdzenie, czy aplikacja poprawnie integruje się z blockchainem i automatycznie przesyła tokeny do portfeli sensorów.

3. **Prezentacja funkcjonalności**:
Zespoły muszą zademonstrować działanie aplikacji, w tym przesyłanie tokenów i przegląd stanu portfeli sensorów przez administratora.

Open project (np. Social media platform)

Więcej informacji u prowadzącego stanislaw.baranski@pg.edu.pl

Dowolny projekt platformy społecznościowej wykorzystujący:

- Backend/Web API w ASP.NET (lub dowolnej innej technologii) implementujący logikę biznesową
- Dowolna baza danych do przechowywania danych użytkowników
- Dowolny zewnętrzny blockchain do rejestrowania transakcji. Najpopularniejszym jest Ethereum, jednak zachęcam do używania bardziej egzotycznych jak np. Base, Arbitrum, Solana, Starknet, Stellar, Unichain etc.

Minimalne wymagania:
- Wymagana jest minimalna implementacja backendu w technologii ASP.NET
- Wymagany jest webowy interfejs użytkownika w dowolnej technologii.
- Wymagane jest połączenie się z testową (darmową) siecią blockchain
- Aplikacja musi być oferować minimalną funkcjonalność w zdrowym rozsądku tego znaczenia.

Każdy blockchain oferuje darmową testową sieć do której dostęp uzyskujemy przez publiczne węzły, najpopularniejszym z nich jest
- https://www.infura.io/
- https://www.alchemy.com/
- https://cloud.google.com/application/web3/

Operacje na blockchanie wymagają uniszczania opłat transakcyjnych w kryptowalutach, w sieciach testowych kryptowaluty można uzyskać przez krany (ang. Faucet)

Możliwości zaliczenia przedmiotu: