System monitorujący

Tin – dokumentacja wstępna

GrzeGorz Aleksiuk

Robert Dudziński

Paweł Świątkowski

Michał Zadrożny

2020

Spis treści

[WSTEP 2](#_Toc37008967)

[WYMAGANIA 2](#_Toc37008968)

[PODZIAŁ NA MODUŁY 3](#_Toc37008973)

[POŁĄCZENIA 4](#_Toc37008978)

[INICJALIZACJA POŁĄCZENIA CZUJNIK-SERWER 5](#_Toc37008982)

[DODAWANIE NOWEGO CZUJNIKA 5](#_Toc37008983)

[DANE: 6](#_Toc37008984)

[ZAWARTOŚĆ 6](#_Toc37008985)

[PRZEPŁYW DANYCH 6](#_Toc37008986)

[BUDOWA WYSYŁANYCH DANYCH **Błąd! Nie zdefiniowano zakładki.**](#_Toc37008987)

[SYTUACJE KRYTYCZNE 7](#_Toc37008988)

[MILESTONES 7](#_Toc37008989)

[PODZIAŁ PRACY 8](#_Toc37008990)

[PRZYKŁADOWE WIDOKI 8](#_Toc37008991)

# WSTEP

W ramach projektu zrealizowany zostanie system umożliwiający monitorowanie użycia zasobów komputera (np. RAMu). Stworzony zostanie serwer agregujący dane zbierane z wielu klientów monitorujących. Dodatkowo powstanie aplikacja monitoringu, umożliwiająca podgląd danych aktualnych oraz historycznych. Do zarządzania tym systemem zostanie stworzona aplikacja administracyjna, umożliwiająca dodawanie nowych klientów-czujek oraz zarządzanie już dodanymi.

# WYMAGANIA

## SERWER

* agregowanie odczytów otrzymanych od klientów-czujników (wykorzystanie lokalnej bazy danych)
* zwracanie danych dla zapytań od klientów (Monitoring i Administrator)
* uwierzytelnienie łączących się klientów poprzez zapisaną listę dozwolonych tokenów ustalonych w panelu administracyjnym
* zostanie zrealizowany w języku C++

## KLIENT (CZUJNIK)

* monitorowanie parametrów komputera
* buforowanie pomiarów (bufor cykliczny), jeżeli nie ma możliwości wysłania ich do serwera
* wysyłanie danych do serwera
* zostanie zrealizowany w języku C++

## KLIENT (ADMINISTRATOR)

* aplikacja desktopowa
* wyświetlanie dodanych klientów-czujników
* wyświetlanie stanu połączenia z czujnikami
* generowanie tokenów pozwalających na dodawanie nowych czujników
* unieważnianie czujników
* zostanie zrealizowany w języku Java – do prezentacji graficznej została wykorzystana biblioteka JavaFX natomiast do połączenia wykorzystywany jest moduł Netty

## KLIENT (MONITORING)

* aplikacja desktopowa
* wyświetlanie danych z czujników
* tworzenie wykresów
* zostanie zrealizowany w języku Java - do prezentacji graficznej została wykorzystana biblioteka JavaFX natomiast do połączenia wykorzystywany jest moduł Netty

# PODZIAŁ NA MODUŁY

## Klient czujnik

* Moduł odczytujący pomiary
* Moduł zarządzający pomiarami
* Moduł komunikacji z serwerem

## Serwer

* Wątek akceptujący podłączanie się nowych klientów czujników
* Wątki odpowiedzialne za komunikacją z konkretnymi klientami czujnikami
* Wątek akceptujący podłączenia klientów monitoring
* Wątki obsługujące zapytania klientów monitoring
* Moduł obsługujący dostęp do bazy danych
* Wątek akceptujący połączenie klientów administracyjnych
* Wątki obsługujące zapytania klientów administracyjnych

## Klient monitoring

* Wątek odpowiedzialny za UI
* Wątek odpowiedzialny za połączenie z serwerem uruchamiany tylko w momencie gdy trzeba przesłać dane

## Panel administracyjny

* Wątek odpowiedzialny za UI
* Wątek odpowiedzialny za połączenie z serwerem

**ARCHITEKTURA**

Serwer

Obsługa każdego typu klienta jest niezależna.

Monitoring request-response flow:

1. Monitoring serializuje i wysyła wiadomość do serwera
2. Serwer odbiera wiadomość od monitoringu: MonitoringListener.onGotRequest()
3. Wykorzystanie SerializerMonitoringMessage do deserializacji otrzymanej wiadomości oraz zbudowania obiektu MonitoringRequest
4. Wrzucenie MonitoringRequest do kolejki monitoringRequestsQueue wewnątrz ServerModel
5. Detykowany wątek obsługujący metodę executeMonitoringRequests() wyjmuje kolejne MonitoringRequest z kolejki i wykonuje je
6. Po wykonaniu tworzony jest MonitoringResponse który zawiera informacje o które prosił Monitoring i wrzuca go do kolejki monitoringResponsesQueue
7. Dedykowany wątek obsługujący metodę sendMonitoringResponse() wyjmmuje kolejne MonitoringResponse z kolejki, następnie przy wykorzystaniu SerializerMonitoringMessage dokonuje serializacji danych i wysyła tak utworzoną wiadomość do Monitoringu
8. Monitoring odbiera wiadomość po czym ją deserializuje, a następnie wyświetla informacje z tej wiadomości

Obraz zawierający tekst

Opis wygenerowany automatycznie

Diagram from <https://app.diagrams.net/>

Administrator request-response flow:

Analogiczny do przedstawionego Monitoring flow, wykorzystuje jednak dedykowany zestaw obiektów, metod i kolejek dla Administratora.

# POŁĄCZENIA

## serwer-czujnik

* każdy klient czujnik jest obsługiwany przez osobny wątek
* połączenie z wykorzystaniem protokołu TCP jest szyfrowane symetrycznym kluczem ustalonym w czasie inicjalizacji
* w przypadku odebrania błędnych danych lub braku otrzymania danych w ustalonym czasie serwer zakończy połączenie
* wysyłanie wcześniej niewysłanych danych w momencie ponownego połączenia klienta z serwerem

## serwer-monitoring

* połączenie z wykorzystaniem protokołu TCP jest szyfrowane
* połączenie jest zestawiane tylko w momencie gdy jest ono potrzebne, tzn wtedy gdy monitoring chce uzyskać dane od serwera
* serwer, jest serwerem równoległym, tzn może obsłużyć wiele połączeń z monitoringiem jednocześnie

## serwer-administrator

* połączenie z wykorzystaniem protokołu TCP jest szyfrowane
* połączenie jest zestawiane na stałe
* serwer, jest serwerem równoległym, tzn może obsłużyć wiele połączeń z administratorem jednocześnie

## INICJALIZACJA POŁĄCZENIA CZUJNIK-SERWER

* Klient czujnik łączy się poprzez gniazdo TCP z serwerem (klient zna adres i port)
* Serwer akceptuje połączenie i tworzy oddzielny wątek do obsługi klienta
* Przy wykorzystaniu kluczy asymetrycznych, uzgadniany jest klucz symetryczny, wykorzystywany później do szyfrowania całej dalszej komunikacji
* Klient wysyła swój token (uzyskany z panelu administracyjnego i przekazany do klienta bezpiecznym kanałem)
* Serwer weryfikuje czy odebrany token znajduje się na liście akceptowalnych tokenów
  + Niepoprawny token
    - serwer informuje klienta o powodzie błędu i kończy połączenie
  + Poprawny token
    - serwer wysyła do klienta swój czas systemowy
    - klient synchronizuje swój czas systemowy z otrzymanym czasem serwerem

## DODAWANIE NOWEGO CZUJNIKA

* Klient Administracyjny wysyła do serwera prośbę o wygenerowanie nowego tokenu
* Serwer wysyła do klienta administracyjnego wygenerowany token
* Bezpiecznym kanałem przekazuje się token do klienta-czujnika
* Dalej jest wykonywany scenariusz ‘Inicjalizacja połączenia czujnik-serwer’

# DANE:

## ZAWARTOŚĆ

* timestamp (określający czas wykonania pomiaru)
* wynik pomiaru
* rodzaj mierzonego zjawiska.

## PRZEPŁYW DANYCH

Dodanie nowych pomiarów do systemu

* Klient czujnik wykonuje pomiary
* Klient czujnik przesyła do serwera dane (aktualne oraz dane z bufora, których nie udało się wcześniej wysłać)
* Serwer weryfikuje dane
* Serwer przesyła dane do bazy danych

Pobranie pomiarów z systemu

* Klient monitoring wysyła zapytanie do serwera z opcjonalnie wypełnionymi parametrami filtrującymi (okres czasu, częstotliwość pomiarów)
* Serwer sprawdza poprawność parametrów
* Serwer odpytuje bazę danych
* Serwer wysyła do klienta dostępne dane

## BUDOWA WIADOMOŚCI WYSYŁANYCH DO SERWERA

* nagłówek wiadomości:
  + wielkość przesyłanej wiadomości (liczba całkowita zapisana binarnie)
* treść wiadomości:
  + parametry zapytania (np. JSON) - rodzaj parametrów i ich interpretacja zależna od typu klienta

Czujnik

* + timestamp pomiaru - czas POSIX
  + wartości zmierzone
  + rodzaj pomiaru

Klient monitorujący

* + wybór czujki, zakres dat, interwał

Administrator

* + typ żądania (odwołanie czujki, prośba o wygenerowanie tokenu dla dodawanej czujki)

# SYTUACJE KRYTYCZNE

* Klient czujnik odczytuje pomiary niezależnie od połączenia z serwerem. W przypadku braku połączenia zapisuje odczyty do bufora cyklicznego, aby w przypadku przepełnienia bufora tracić starsze dane. W przypadku przywrócenia połączenia (próba połączenia co ustalony interwał) klient wysyła najpierw aktualny odczyt, a w międzyczasie wysyła partiami zbuforowane pomiary.
* Problem opóźnień sieci jest pomijany ze względu na istniejący znacznik czasowy dla każdego odczytu
* W przypadku aktywnego połączenia TCP i braku odpowiedzi od drugiej strony w ustalonym czasie połączenie jest przerywane (tym samym ew. wątki są zabijane).
* W przypadku obciążenia bazy danych moduł obsługujący bazę danych będzie buforował dane, aby nie zawieszać wątków otrzymujących dane od czujników

# MILESTONES

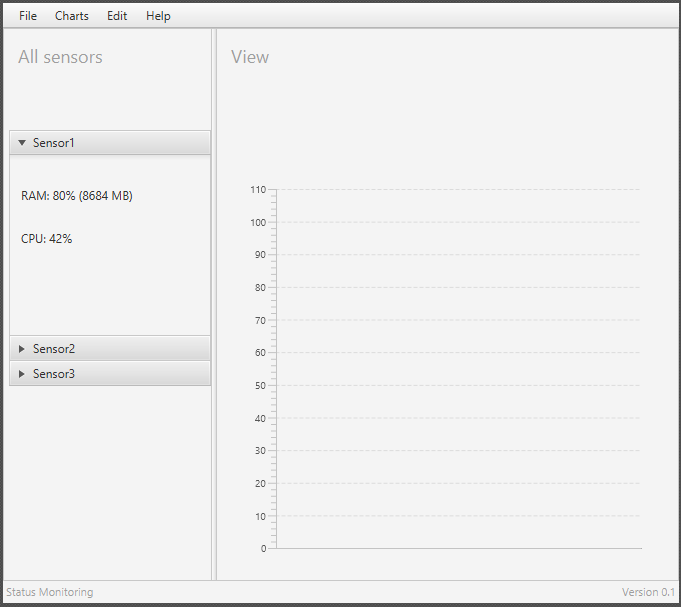
* stworzenie architektury systemy (zaprojektowanie, dokumentacja)
* połączenie serwera z bazą danych
* stworzenie modułów komunikacji w serwerze i kliencie czujniku
* odczytywanie i zapisywanie (buforowanie) pomiarów w kliencie czujniku
* połączenie między serwerem a klientem monitorującym (obsługa zapytań do bazy danych)
* obsługa UI klienta monitorującego
* połączenie między serwerem a klientem administracyjnym (generowanie tokenów, unieważnianie klientów)
* obsługa UI klienta administracyjnego
* testowanie

PODZIAŁ PRACY

Podczas wykonywania każdego z kolejnych milestone’ów będziemy dynamicznie przydzielać osoby odpowiedzialne za konkretne zadania.

# PRZYKŁADOWE WIDOKI

* Status monitoring:



* Administrator panel:

