System monitorujący

Tin – dokumentacja wstępna

GrzeGorz Aleksiuk

Robert Dudziński

Paweł Świątkowski

Michał Zadrożny

2020

Spis treści

[WSTEP 2](#_Toc41954433)

[WYMAGANIA 2](#_Toc41954434)

[SERWER 2](#_Toc41954435)

[KLIENT (CZUJNIK) 2](#_Toc41954436)

[KLIENT (ADMINISTRATOR) 2](#_Toc41954437)

[KLIENT (MONITORING) 3](#_Toc41954438)

[PODZIAŁ NA MODUŁY 3](#_Toc41954439)

[Klient czujnik 3](#_Toc41954440)

[Serwer 3](#_Toc41954441)

[Klient monitoring 4](#_Toc41954442)

[Panel administracyjny 4](#_Toc41954443)

[POŁĄCZENIA 6](#_Toc41954444)

[serwer-czujnik 6](#_Toc41954445)

[serwer-monitoring 6](#_Toc41954446)

[serwer-administrator 6](#_Toc41954447)

[INICJALIZACJA POŁĄCZENIA CZUJNIK-SERWER 6](#_Toc41954448)

[DODAWANIE NOWEGO CZUJNIKA 7](#_Toc41954449)

[DANE MONITOROWANE 7](#_Toc41954450)

[ZAWARTOŚĆ 7](#_Toc41954451)

[PRZEPŁYW DANYCH 7](#_Toc41954452)

[WIADOMOŚCI WYSYŁANE DO SERWERA 8](#_Toc41954453)

[Wiadomości przed wysłaniem są serializowane do ciągu bajtów a po odebraniu deserializowane do odpowiednich obiektów. 8](#_Toc41954454)

[SYTUACJE KRYTYCZNE 9](#_Toc41954455)

[MILESTONES 10](#_Toc41954456)

[PODZIAŁ PRACY 10](#_Toc41954457)

[WIDOKI Z APLIKACJI 10](#_Toc41954458)

# WSTEP

W ramach projektu zrealizowany zostanie system umożliwiający monitorowanie użycia zasobów komputera - 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 bazy danych Oracle)
* zwracanie danych dla zapytań od klientów (Monitoring i Administrator)
* uwierzytelnienie łączących się klientów poprzez zapisaną listę dozwolonych tokenów generowanych w panelu administracyjnym
* zostanie zrealizowany w języku C++

## KLIENT (CZUJNIK)

* monitorowanie parametrów komputera – procentowe zużycie RAMu
* buforowanie pomiarów, 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
* generowanie wykresów pokazujących zużycie RAMu
* 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 komunikacji z serwerem

## Serwer

* Każdy typ klienta (czujnik, administrator, monitoring) jest obsługiwany niezależnie, tzn. ma swój osobny moduł sieciowy oraz moduł wykonujący zapytania
* Obsługa sensorów:
  + Wątek obsługujący moduł sieciowy dla sensorów (podłączanie się i odbieranie wiadomości)
  + Wątek wykonujący zapytania od sensora – obsługa bazy danych (sprawdzenie poprawności tokenu, dodanie pomiaru)
* Obsługa monitoringu:
  + Wątek obsługujący moduł sieciowy dla monitoringu (podłączanie się klienta, odbieranie wiadomości)
  + Wątek wykonujący zapytania od monitoringu (obsługa bazy danych)
  + Wątek wysyłający odpowiedzi do klienta monitoringu
* Obsługa administratora:
  + Wątek obsługujący moduł sieciowy dla administratora (podłączanie się klienta, odbieranie wiadomości)
  + Wątek wykonujący zapytania od administratora (obsługa bazy danych)
  + Wątek wysyłający odpowiedzi do klienta administratora

## 

## Klient monitoring

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

## 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. Dedykowany 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() wyjmuje 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 request-response flow, wykorzystuje jednak dedykowany zestaw obiektów, metod i kolejek dla Administratora.

# POŁĄCZENIA

## serwer-czujnik

* każdy klient czujnik jest obsługiwany przez ten sam moduł sieciowy
* połączenie z wykorzystaniem protokołu TCP jest szyfrowane przy użyciu openSSL
* 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
* połączenie jest zestawiane na stałe w momencie uruchomienia aplikacji
* serwer, jest serwerem równoległym, tzn. może obsłużyć wiele połączeń z monitoringiem jednocześnie
* po stronie monitoringu wykorzystywany jest Netty – asynchroniczna obsługa odbierania/wysyłania wiadomości

## serwer-administrator

* połączenie z wykorzystaniem protokołu TCP
* połączenie jest zestawiane na stałe przy uruchomieniu aplikacji
* serwer, jest serwerem równoległym, tzn. może obsłużyć wiele połączeń z administratorem jednocześnie
* po stronie administratora wykorzystywany jest Netty – asynchroniczna obsługa odbierania/wysyłania wiadomości

## INICJALIZACJA POŁĄCZENIA CZUJNIK-SERWER

* Klient czujnik łączy się poprzez gniazdo TCP z serwerem (klient zna adres i port – podany jest w pliku konfiguracyjnym sensora)
* Serwer akceptuje połączenie
* Przy wykorzystaniu kluczy asymetrycznych, uzgadniany jest klucz symetryczny, wykorzystywany później do szyfrowania całej dalszej komunikacji - openSSL
* 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
    - sensor się wyłącza
  + 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 MONITOROWANE

## ZAWARTOŚĆ

* timestamp (określający czas wykonania pomiaru)
* wynik pomiaru – procentowe użycie RAMu

## 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 parametrami idSensora oraz okres czasu (last hour/last 24h/last month)
* Serwer sprawdza poprawność parametrów
* Serwer odpytuje bazę danych
* Serwer wysyła do klienta dostępne dane

# WIADOMOŚCI WYSYŁANE DO SERWERA

## Wiadomości przed wysłaniem są serializowane do ciągu bajtów a po odebraniu deserializowane do odpowiednich obiektów.

Budowa wiadomości:

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

Czujnik

* ADD\_MEASUREMENT
  + Rodzaj zapytania
  + timestamp pomiaru - czas POSIX
  + wartość zmierzona – procentowe zużycie RAMu
* AUTHENTICATE\_SENSOR/ CONNECTED\_SENSOR
  + Rodzaj zapytania
  + Token identyfikujacy

Klient monitorujący

* GET\_ALL\_SENSORS\_MONITORING
  + Rodzaj zapytania
* GET\_SET\_OF\_MEASUREMENTS
  + Rodzaj zapytania
  + Id sensora którego pomiary chcemy uzyskać
  + Zakres czasowy pomiarów (last hour/last 24h/last month)

Administrator

* GET\_ALL\_SENSORS
  + Rodzaj zapytania
* UPDATE\_SENSOR\_NAME
  + Rodzaj zapytania
  + Id sensora którego modyfikujemy
  + Nowa nazwa sensora
* REVOKE\_SENSOR
  + Rodzaj zapytania
  + Id sensora którego odwołujemy
* DISCONNECT\_SENSOR
  + Rodzaj zapytania
  + Id sensora którego odłączamy
* GENERATE\_TOKEN
  + Rodzaj zapytania

# SYTUACJE KRYTYCZNE

* Klient czujnik odczytuje pomiary niezależnie od połączenia z serwerem. W przypadku braku połączenia zapisuje odczyty do bufora. W przypadku przywrócenia połączenia (próba połączenia co ustalony interwał) klient wysyła 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 zamykane).
* W przypadku obciążenia bazy danych moduł wykonujący zapytania do bazy będzie się również opóźniał (czekając na wykonania polecenia na bazie), jednak same zapytania są buforowane w kolejkach (RequestQueue) dzięki czemu nie będziemy tracili zapytań.
* Klient monitoring oraz klient administrator w przypadku gdy zostanie utracone połączenie do serwera wyświetlają informację o tym i zamykają się
* Klient monitoring oraz klient administrator w przypadku gdy po uruchomieniu nie mogą nawiązać połączenia z serwerem wyświetlają informację o tym i zamykają się

# 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

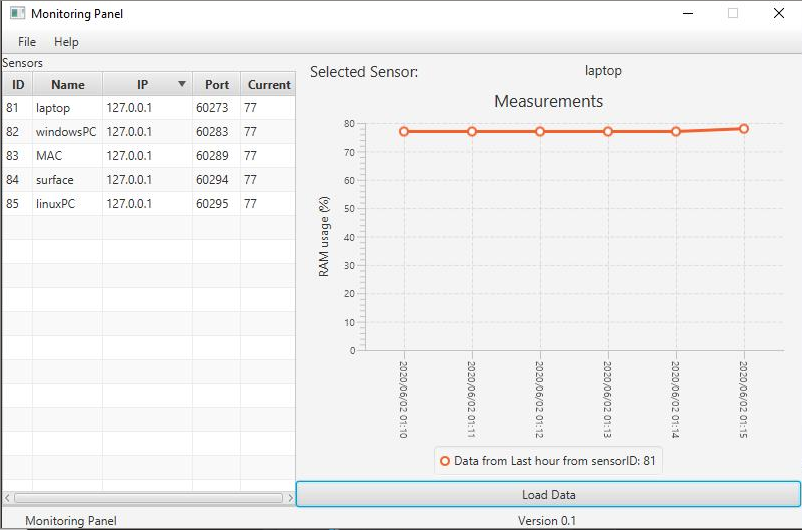
Podczas wykonywania każdego z kolejnych milestone’ów przydzielaliśmy dynamicznie zadania.

Główne odpowiedzialności:

* Aleksiuk Grzegorz – obsługa bazy danych, konfiguracja połączenia, wykonywanie requestów, integracja Windows-Mac
* Dudziński Robert – moduł komunikacyjny w C++, client sensor
* Świątkowski Paweł – moduł komunikacyjny w Javie (netty), client administrator
* Zadrożny Michał – client monitoring, bugfixing client administratora, testy integracyjne

# WIDOKI Z APLIKACJI

* Status monitoring:



* Administrator panel:

