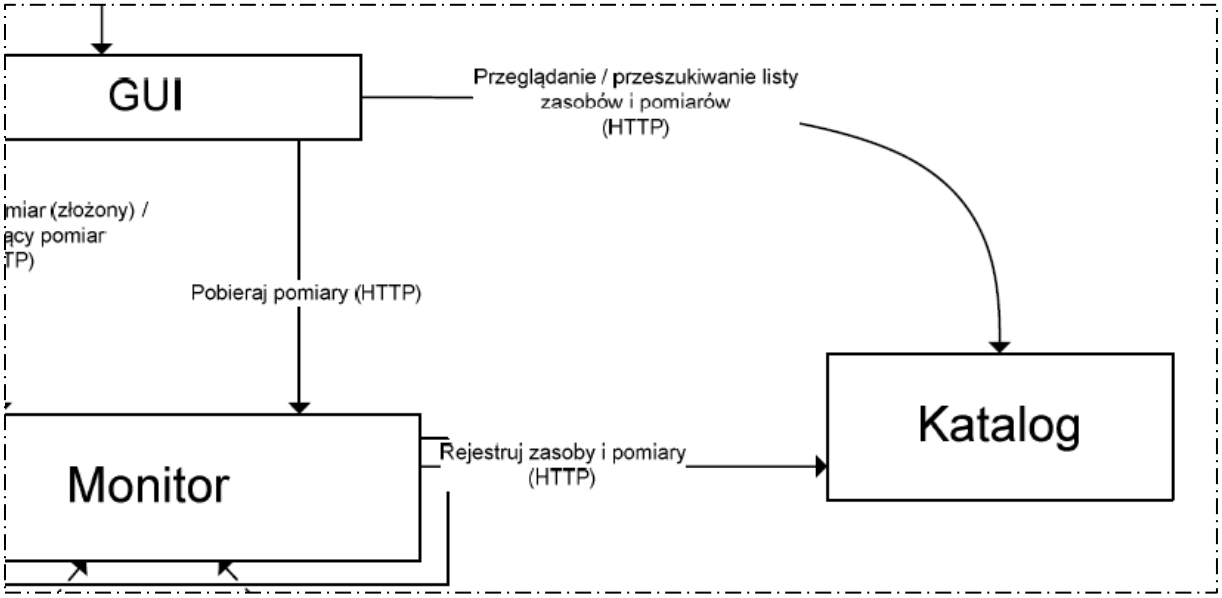
Katalog - dokumentacja komponentu

1. Opis

Celem i zadaniem realizowanym przez ten komponent jest umożliwienie przeglądania i przeszukiwania dostępnych monitorowanych zasobów i pomiarów.
Każdy ***Monitor*** odpowiedzialny jest za rejestrowanie i uaktualnianie w ***Katalogu*** swojej listy zasobów i pomiarów.



Rys 1. Fragment architektury systemu z uwzględnieniem roli Katalogu

2. Lokalizacja komponentu

Katalog został umieszczony w repozytorium projektu w dwóch postaciach:

1. *Kod źródłowy* – w postaci solucji projektu Visual Studio (kod źródłowy komponentu został napisany w języku C#, na platformie .NET, której środowisko uruchomieniowe jest konieczne do działania komponentu)
   https://github.com/pz-agh/pz-monitor/tree/master/pz-katalog
2. *Plik wykonywalny* – w postaci pliku CatalogueComponent.exe
   https://github.com/pz-agh/pz-monitor/tree/master/pz-katalog/Executable

3. Wymagania

Ze względu na obraną technologię (Microsoft .NET) i wersję frameworku .NET, uruchomienie aplikacji ***Katalogu*** wymaga spełnieniu kilku minimalnych wymagań software'owych:

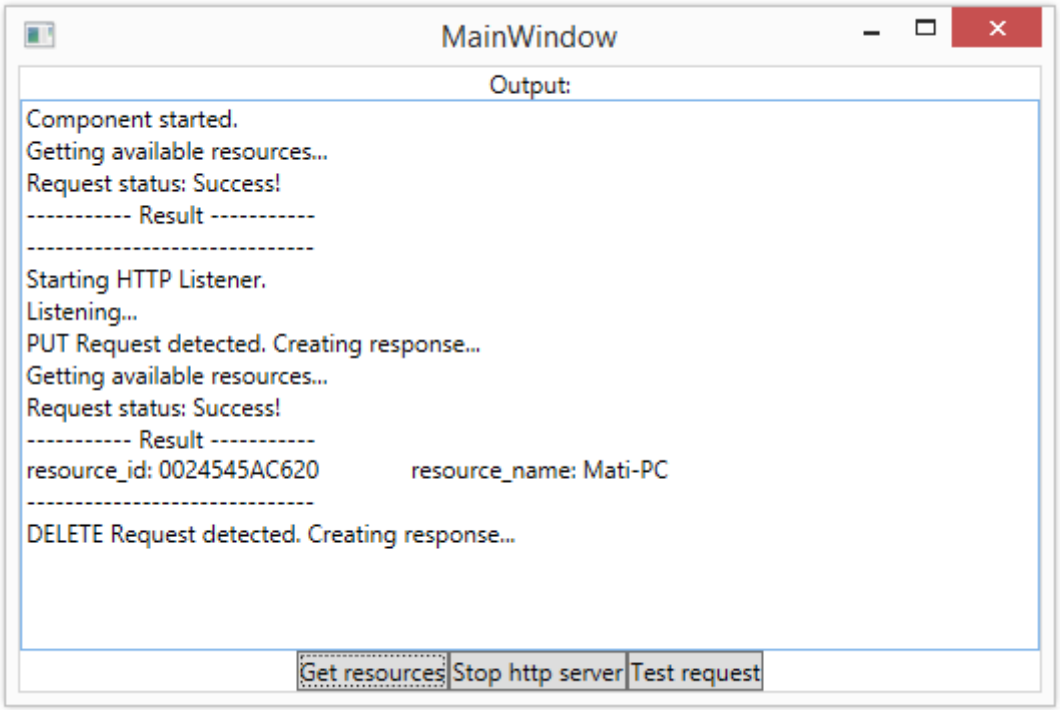
- Microsoft Windows 7+
- Microsoft .NET Framework 4.5+

4. Konfiguracja i uruchomienie komponentu

Aby użyć ***Katalog*** wystarczy pobrać plik wykonywalny .exe, z powyżej zamieszczonej lokalizacji i uruchomić. Komponent w tym momencie rozpocznie swoje działanie, tzn. zainicjuje pobranie z ***Monitorów*** informacji o dostępnych zasobach i pomiarach, a następnie przejdzie w tryb nasłuchiwania na zmiany sygnalizowane przez ***Monitory*** oraz na zapytania od strony klientów.
Z uwagi na to, że komponent komunikuje się i oczekuję informacji od Monitora już w momencie rozpoczęcia pracy dobrze Katalog uruchomić już po rozpoczęciu działania Monitora (choć nie jest to konieczne).

5. Interfejs graficzny komponentu

Komponent udostępnia prosty interfejs graficzny informujący o stanie komponentu i wykrytych operacjach/komunikacji.



Rys 2. Wygląd GUI Katalogu

6. Usługi REST dla Katalogu

Katalog udostępnia poniższe usługi poprzez interfejs REST:

- Usługa zwracająca informacje o dostępnych zasobach:
  - URL: {ip_address}:8081/resources/
  - Metoda HTTP: GET
  - Rezultat: "[{\"resource_id\":\"mac_address\",\"resource_name\":\"host_name\"},{...}"
- Usługa zwracająca informacje o dostępnych na danym zasobie pomiarach:
  - URL: {ip_address}:8081/resources/{resource_id}/measurements
  - Metoda HTTP: GET
  - Rezultat: [{\"measurement_name\":\"japieprze\"},{...}]
- Usługa dodająca zasób o danym identyfikatorze do bazy danych katalogu:
  - URL: {ip_address}/resources/{id}
  - Metoda HTTP: PUT
- Usługa usuwająca zasób o danym identyfikatorze z bazy danych katalogu:
  - URL: {ip_address}/resources/{id}
  - Metoda HTTP: DELETE
- Usługa dodająca pomiar określonego zasobu do bazy danych katalogu:
  - URL: { ip_address }/resources/{id}/measurements/{id}
  - Metoda HTTP: PUT
- Usługa usuwająca pomiar określonego zasobu z bazy danych katalogu:
  - URL: { ip_address }/resources/{id}/measurements/{id}
  - Metoda HTTP: DELETE

7. Skrócony opis działania

Po uruchomieniu komponent próbuje pobrać z Monitorów dane zasobów i dostepnych na tych zasobach pomiarów. Informacje te pobierane są za pomocą klasy WebRequest. Jeśli krok ten się powiedzie, w oddzielnym wątku uruchamiane są operacje odpowiedzialne za nasłuchiwanie zapytań HTTP.
Rejestrowanie zapytań HTTP realizowane jest za pomocą klasy HttpListener dostępnej standardowo w przestrzeni nazw System.Net. W momencie wykrycia zapytania klasa ta przechwyca wymagane dane i przekazuje je do metody odpowiedzialnej za sprawdzenie poprawności ścieżki i (jeśli jest to wymagane) wygenerowania odpowiedzi, a następnie wraca do trybu nasłuchiwania.
Przechwycony z ***Monitorów*** JSON po przemapowaniu na obiekty ***Katalogu*** służy następnie za podstawę do utworzenia bądź zaktualizowania stanu rejestru zasobów i pomiarów przechowywanych przez ***Katalog***.

Katalog przetwarza JSON i tworzy na jego podstawie obiekty/encje zasobów lub pomiarów. Tak przygotowana lista trafia następnie do mechanizmu odpowiedzialnego za tworzenie rejestru zasobów i pomiarów. ***Katalog*** przechowuje rejestr w postaci schematu .xml.

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<resources>
  <resource resource_id="0024545AC620" />
</resources>
```

Listing 1. Przykładowy rejestr zasobów

W wypadku przyjściu komunikatu z ***Monitora*** aktualizującego stan zasobów lub pomiarów, czyli dodania go bądź usunięcia (PUT albo DELETE) zostaje wyzwolony mechanizm aktualizujący rejestr (odpowiednio dodanie bądź usunięcie pozycji w rejestrze).
W wypadku przyjścia komunikatu od ***Klienta*** o chęci pobrania informacji o dostępnych zasobach i pomiarach, rejestr zostaje otwarty a następnie przetworzony na JSON i w tej postaci dostarczony klientowi. W ten sposób ***Klient*** zostaje poinformowany o dostępnych w systemie możliwościach o które następnie może zapytać już Monitor.