Systemy Hipermedialne 2012

System monitorujący.

Grupa: poniedziałek 11:30 – 12:30

Architekt: Krzysztof Romanowski

Spis Treści

Części składowe i ogólny zarys komunikacji oraz podstawowe założenia

Podstawowe założenia

1. Formatem komunikacji w systemie będzie JSON
2. Autentykacja: Query Authentication over HTTPS – po zalogowaniu w odpowiedzi dostajemy cookies które potem dodajemy do każdego zapytania, co sprawdza serwer

Części składowe

1. **Monitor:** element mający za zadanie agregować, przyjmować oraz zarządzać wynikami pobranymi z sensorów. Udostępnia też tworzenie oraz zarządzanie subskrypcjami
2. **Sensor**: element pobierający dane oraz wysyłający je do monitora
3. **Catalog:** element zbierający dane od Monitorach oraz pomiarach przez nie udostępniane. Jest swoistą bazą danych
4. **Klient:** może przeglądać katalog monitorów oraz tworzyć oraz zarządzać subskrypcjami na danym monitorze. Czyta dane ze swoich subskrypcji. W ramach projektu nie będziemy rozróżniali klienta-GUI od klienta-Automatu jako że w punktu widzenia systemu będą komunikować się w identyczny sposób.

Resources

1) **catalog**/monitors
   a) GET – zwraca listę monitorów
   b) POST – dodaje/nadpisuję nowy monitor
   c) DELETE – usuwa monitor
2) **catalog**/feed_view
   a) GET – zwraca liste wszystkich feedów – cache z wywołań **monitor/feed** na wszystkich monitorach + wyszukiwanie
3) **monitor**/feed
   a) GET – zwraca listę feedów
   b) POST – tworzy nową feed
   c) DELETE – usuwa feed
4) **monitor/**sensors/{id}
   a) GET – zwraca dane z feeda (pooling)
   b) POST – służy do wysyłania nowych pomiarów
5) **(monitor | sensor)** / login
   a) POST – służy do logowania

Request:
{login: String /login użytkownika/
password: String /hasło użytkownika/}

Response: pusta odpowiedź z cookie'sem

Specyfikacja komunikatów

Ogólne zasady

1. Przy tworzeniu/edytowaniu nowego obiektu
   a) wypełniamy tylko pola oznaczone jako NEW_REQUIRE pozostałe pola wypełnia serwer
   b) w odpowiedzi dostajemy w pełny obiekt (serwer ustawia wszystkie pola, np. id)
   c) jeśli pole typu href jest oznaczone jako REFERENCE to przy tworzeniu podajemy zamiast URI id zasobu
2. Przy korzystaniu z API wykorzystujemy w pierwszej kolejności paramenty href nad statycznym linkowaniem (korzystamy z href wszędzie tam gdzie to możliwe)

Komunikaty

1) **catalog**/monitors
   a) GET – zwraca listę monitorów

      Request: brak

      Response: {monitors: [MonotorObjects*] /lista monitoriów na serwerze/ }

   b) POST – dodaje/nadpisuję nowy monitor

      Request: { monitor: MonitorObject }

      Response: { monitor: MonitorObject }

   c) DELETE – usuwa monitor

      Request: { monitor: MonitorObject }

      Response: { monitor: MonitorObject }

2) **catalog**/feed_view
   a) GET – zwraca listę feedów na wszystkich monitorach
      Request: lista paramentów do wyszukania (nazwa_parametru=wartość)
      Przykładowo: **catalog**/feed_view?mesure=ala&id=3
      Response: {sensors: SensorInfoObject* /lista obiektów na serwerze/}

3) **monitor/**sensors
   a) GET – zwróć listę sensorów

      Request: brak

      Response: {sensors: SensorInfoObject* /lista obiektów na serwerze/}

   b) POST – dodaje/napisuję nowy sensor

      Request: {sensor: SensorInfoObject}

      Request: {sensor: SensorInfoObject}

4) **monitor/**sensors/{id}
   a) POST – służy do wysyałania nowych pomiarów

      Request: {data: [Object*] /wyniki pomiarów/}

      Response: brak

   b) GET – zwraca dane z subskrypcji (pooling, od najświeższych danych)

      Request: parametry:

      - amount: Int – ilość pomiarów (domyślnie 0)
      - skip: Int – pomiń skip ostatnich pomiarów (domyślnie 0)

      Response:{measurements: MesuementObject* /wyniki pomiarów/}

   c) DELETE – usuwa sensor

5) **monitor**/functions
   a) GET – pobiera listę wspieranych funkcji

      Request: brak

      Response: {functions: [FunctionObject*]}

Błędy

W systemie wyróżniamy następujące błędy zwracane przez API:

1. BadData – zwracany gdy w zapytaniu typu POST dane były błędne/zasoby nie istniały

   kod HTTP: **400 Bad Request**
   body {mssage: messageString, messageDescription: Object}

2. NotFound – zwracany gdy urzytkownik odwołuję się do zasobu który nie istnieje na serwerze

   kod HTTP: **404 Not Found**
   body -

3. Unauthorized – zwracany gdy użytkownik nie ma uprawnień do zasobu / nie jest zalogowany

   kod HTTP: **401 Unauthorized**
   body: -

Obiekty

1. **Object** – dowolny obiekt
2. **MoniotorObjects** reprezentuje monitor

   { **id**: String */id monitora/*
   name: String */nazwa monitora NEW_REQUIRE/*
   href: URI */uri do monitora NEW_REQUIRE/*
   }
3. **MesuementObject** – reprezentuje winiki pomiarów

   { **id**: String */id zasobu/*
   sensor: URI */link do sensora/*
   measure: String */miara/*
   dataType: String */typ zwracanych danych/*
   data: [{ / *data feed/*
   date: Date */when/*
   what: ? */wynik pomiaru/*
   }]
   }
4. **SensorInfoObject** – reprezentuje sensor lub agregacje sensorów (agregacja to **AggregationSensorInfoObject**)

   { **id**: String */id zasobu/*
   name: String */nazwa NEW_REQUIRE/*
   ***href**: URI /uri do sensora/*
   measure: String */miara NEW_REQUIRE/*
   dataType: String */typ zwracanych danych NEW_REQUIRE/*
   frequency: Int */ilość sekund między kolejnymi pomiarami jeśli ujemna lub 0 pomiary nie są systematyczne (wartość ujemna oznacza szacowaną średnią ilość sekund między pomiarami) NEW_REQUIRE/*
   resource: String */co jest mierzone NEW_REQUIRE/*
   type: 'simple'|'aggregation' */typ sensora simple - „czyste” wyniki z sensora aggregation – agragacja wyników z sensora/sensorów NEW_REQUIRE/*
   }
5. **AggregationSensorInfoObject** extend **SensorInfoObject** – reprezentuje agregacje wyników

   { */pola z SensorInfoObject/*
   arguments: [URI*] */lista sensorów z których budowana jest agregacja REFERENCE NEW_REQUIRE/*
   function: URI */id funkcji agregującej REFERENCE NEW_REQUIRE/*
   }
6. **FunctionObject** – reprezentuje funkcje agregującą

   { **id**: String */id funkcji/*
   argumentCount: int */ilość argumentów/*
   name: String */nazwa funcji/*
   description: String /opis funkcji/
   }