

Alexandre Meslin (Bolsista CAPES/BRASIL)

Noemi Rodriguez

Markus Endler

(meslin, noemi, endler)@inf.puc-rio.br

MUSANet – Mobile Urban Sensor and Actuator Network



Programação

SEMANA 1

Dia 22

- REST e JSON
- Introdução ao MUSANet

Dia 23

- Instalando e conhecendo a plataforma InterSCity

Dia 24

- Instalação, configuração do ContextNet.
- Desenvolvendo primeiras aplicações.

Dia 25

- Desenvolvendo aplicações com o ContextNet

Dia 26

- Utilizando o Mobile-Hub para captura de dados – inclui transmissão da informação para o ContextNet

SEMANA 2

Dia 29

- Implementação de uma versão distribuída do ContextNet.

Dia 30

- Projetos usando o MUSANet

Dia 31

- Primeiros testes dos protótipos das aplicações para locais inteligentes

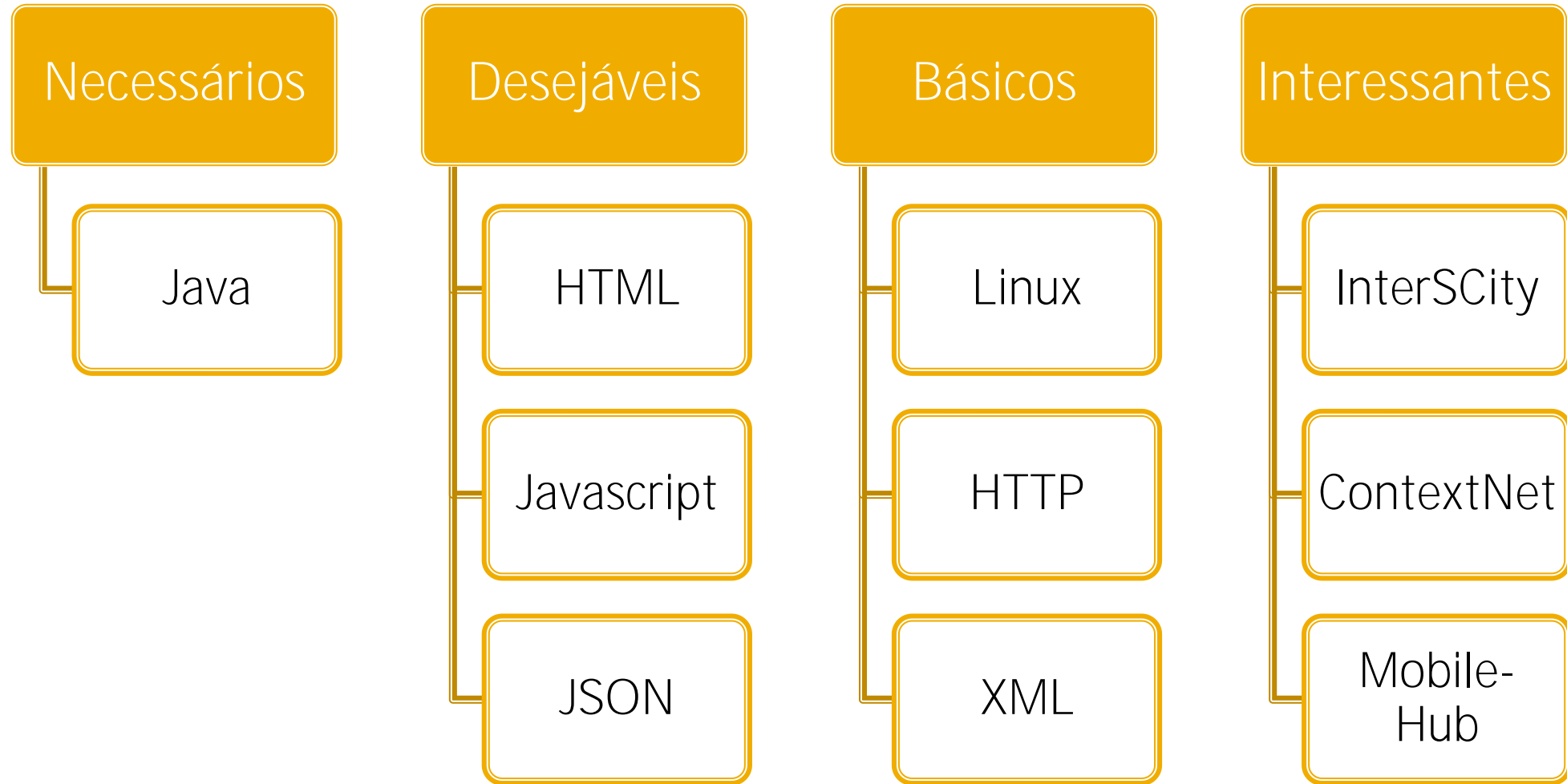
Dia 01

- Continuação dos testes

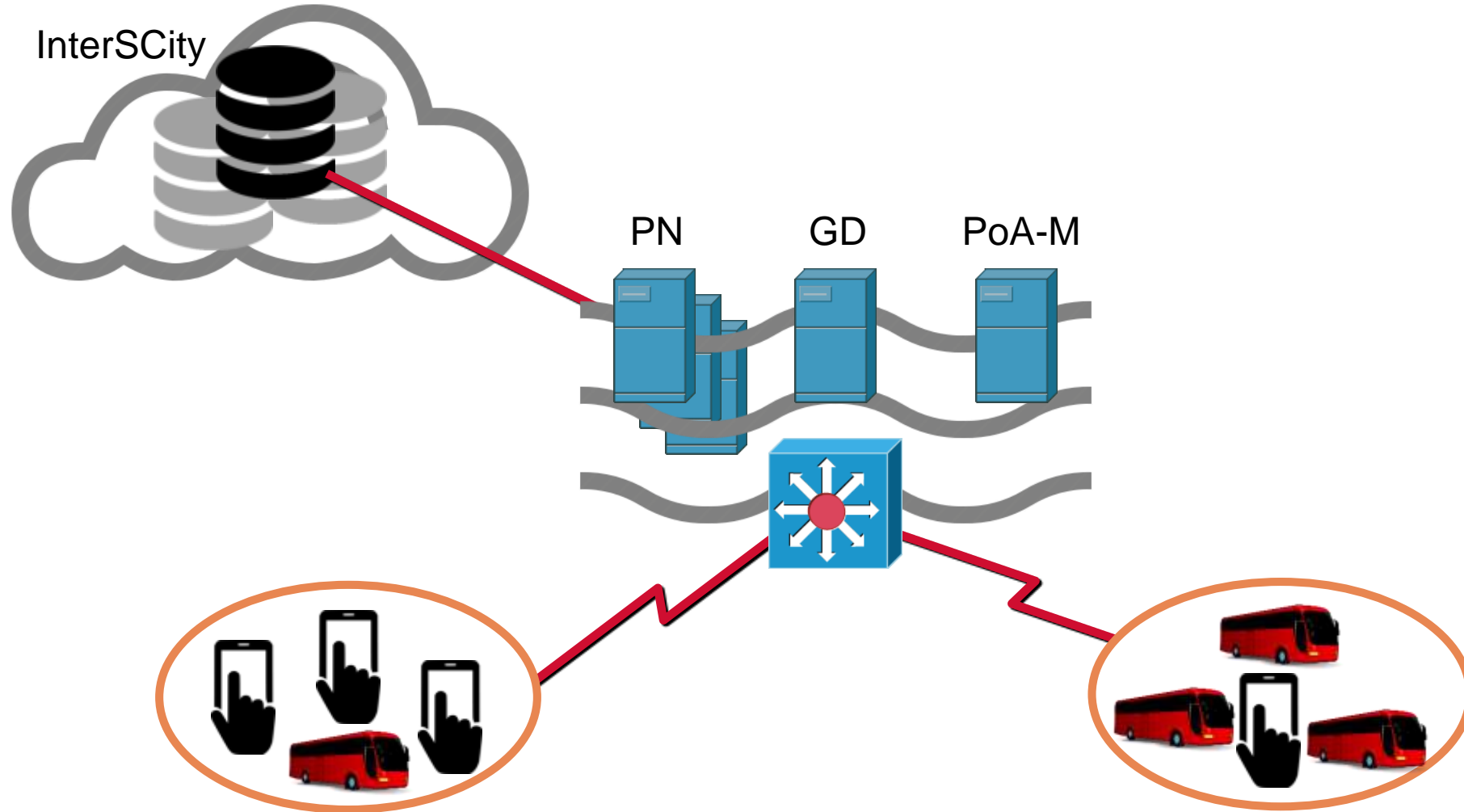
Dia 02

- Validação e avaliação das aplicações desenvolvidas

Conhecimentos Prévios



Mini Overview



Ambiente

APLICATIVOS

- Eclipse
(ou equivalente genérico)
- Ubuntu
(ou equivalente genérico)
- Java 8
- Tomcat (qualquer versão)
- ContextNet
- OpenSplice
- Browser
- VMWare ou VirtualBox



MAVEN

```
<repositories>
  <repository>
    <id>LAC PUC-Rio</id>
    <url>https://bitbucket.org/endler/contextnet-dependencies/raw/master</url>
  </repository>
</repositories>

<dependencies>
  <dependency>
    <groupId>br.pucrio.inf.lac</groupId>
    <artifactId>contextnet</artifactId>
    <version>2.7</version>
  </dependency>
  <dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20180813</version>
  </dependency>
  <dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1</version>
  </dependency>
</dependencies>
```

Slides

- Alguns slides foram baseados no material preparado por:
 - Markus Endler
 - Felipe Carvalho
 - Arthur Del Esposte

Perguntas?



JSON

JavaScript Object Notation



O Que é JSON

Javascript Object Notation

Uma sintaxe para armazenar e transmitir informações em formato texto – assim como é feito com XML

Independente de linguagem de programação

Auto-descritivo e fácil de entender

Alguns dizem que JSON é menor, mais rápido e mais fácil do que XML – você decide

- <https://www.youtube.com/watch?v=kc8BAR7SHJI>

Exemplo de uma estrutura em JSON

```
{  
  "aircraft": "A320",  
  "pilot": {  
    "firstName": "John",  
    "lastName": "Adams"  
  },  
  "passenger": [  
    "George Washington",  
    "Thomas Jefferson"  
  ]  
}
```

Refere-se a um objeto com a seguinte estrutura e valores:

- objeto.aircraft = "A320"
objeto.pilot.firstName = "John"
objeto.pilot.lastName = "Adams"
objeto.passenger[0] = "George Washington"
objeto.passenger[1] = "Thomas Jefferson"

Sintaxe

Baseado em Javascript

- Dados aparecem em pares de nome/valor
- Dados são separados por vírgula
- Chaves {} separam objetos
- Colchetes [] separam vetores
- Sinal de igual = é substituído por dois pontos :

Sintaxe

Os valores podem ser do tipo

Numérico

String

Booleano

Vetor (array)

Objeto

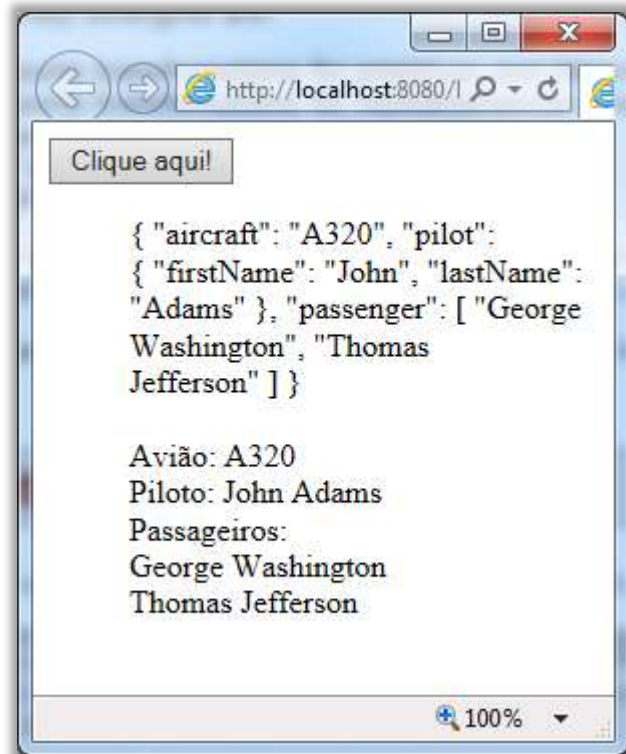
Null

Exemplo: código do lado do servidor

```
public class JSON extends HttpServlet
{
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        JSONObject json = new JSONObject();
        // Put a simple element
        json.put("aircraft", "A320");
        // Add a JSON Object
        JSONObject pilot = new JSONObject();
        pilot.put("firstName", "John");
        pilot.put("lastName", "Adams");
        json.put("pilot", pilot);
        // Accumulate values in an array
        json.accumulate("passenger", "George Washington");
        json.accumulate("passenger", "Thomas Jefferson");
        PrintWriter out = response.getWriter();
        response.setContentType("text/plain");
        out.println(json.toString());
    }
}
```

Resultados:

NAVEGADOR



CONSOLE

```
JSON: {  
  "aircraft": "A320",  
  "pilot": {  
    "firstName": "John",  
    "lastName": "Adams"  
  },  
  "passenger": [  
    "George Washington",  
    "Thomas Jefferson"  
  ]  
}
```

Exemplo

Obtendo posição dos ônibus no Rio de Janeiro

```
package br.com.meslin.ufma.main;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import org.json.JSONArray;
import org.json.JSONObject;
```


```
public class GetBuses {
    // constants
    private static final String USER_AGENT =
    "Mozilla/5.0";
    private static final int DATAHORA = 0;
    private static final int ORDEM = 1;
    private static final int LINHA = 2;
    private static final int LATITUDE = 3;
    private static final int LONGITUDE = 4;
    private static final int VELOCIDADE = 5;
```

Exemplo

Obtendo posição dos ônibus no Rio de Janeiro

```
public class GetBuses {
    ...
    public static void main(String[] args) {
        GetBuses getBuses = new GetBuses();
        JSONArray jsonBuses = getBuses.production();

        for(int i=0; i<jsonBuses.length(); i++) {
            JSONArray jsonBus = jsonBuses.getJSONArray(i);
            // Houston, we have a problem...LINHA sometime is string, sometimes int
            try {
                System.out.println(jsonBus.getInt(LINHA) + ": " + jsonBus.getString(ORDEM) + " " +
                    jsonBus.getString(DATAHORA) + "(" + jsonBus.getDouble(LATITUDE) + "," +
                    jsonBus.getDouble(LONGITUDE) + ") @" + jsonBus.getDouble(VELOCIDADE) + "km/h");
            } catch(Exception e) {
                System.out.println(jsonBus.getString(LINHA) + ": " + jsonBus.getString(ORDEM) + " " +
                    jsonBus.getString(DATAHORA) + " (" + jsonBus.getDouble(LATITUDE) + "," +
                    jsonBus.getDouble(LONGITUDE) + ") @" + jsonBus.getDouble(VELOCIDADE) + "km/h");
            }
        }
    }
    ...
}
```



Retorna um Array JSON

Exemplo

Obtendo posição dos ônibus no Rio de Janeiro

```
public class GetBuses {
    ...
    public JSONArray production() {
        // First step: get data from city hall website
        try {
            url = new
URL("http://dadosabertos.rio.rj.gov.br/apiTransporte/apresentacao/rest/index.cfm/obterTodasPosicoes");
            connection = (URLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("User-Agent", USER_AGENT);
            responseCode = connection.getResponseCode();
        } catch (IOException e) {
            System.err.println("Fatal error");
        }
        ...
    }
}
```

Exemplo

Obtendo posição dos ônibus no Rio de Janeiro

```
public class GetBuses {
    ...
    public JSONArray production() {
        ...
        // read bus data
        if(responseCode == 200) {
            try {
                BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                String inputLine;
                buffer = new StringBuffer();
                while((inputLine=reader.readLine())!=null) {
                    buffer.append(inputLine);
                }
                reader.close();
            } catch (IOException e) {
                System.err.println("Fatal error...");
            }
        } else {
            System.err.println("Responde code...");
            return null;
        }
        ...
    }
}
```



Deveria haver tratamento de erro aqui.
Pelo menos ler a mensagem de erro!

Exemplo

Obtendo posição dos ônibus no Rio de Janeiro

```
public class GetBuses {
    ...
    public JSONArray production() {
        ...
        // Second step: create the buses structure
        // create a JSON object based on bus data
        if(buffer != null) {
            JSONObject jsonObject = new JSONObject(buffer.toString());
            JSONArray jsonData = jsonObject.getJSONArray("DATA");
            return jsonData;
        }
        else {
            System.err.println("Error code #" + responseCode);
        }
        return null;
    }
}
```

Exercícios

Exercício

- Crie uma estrutura JSON para representar uma turma de alunos de determinada disciplina
 - Nome da disciplina
 - Nome do professor
 - Nome e matrícula dos alunos
 - 3 notas para cada aluno

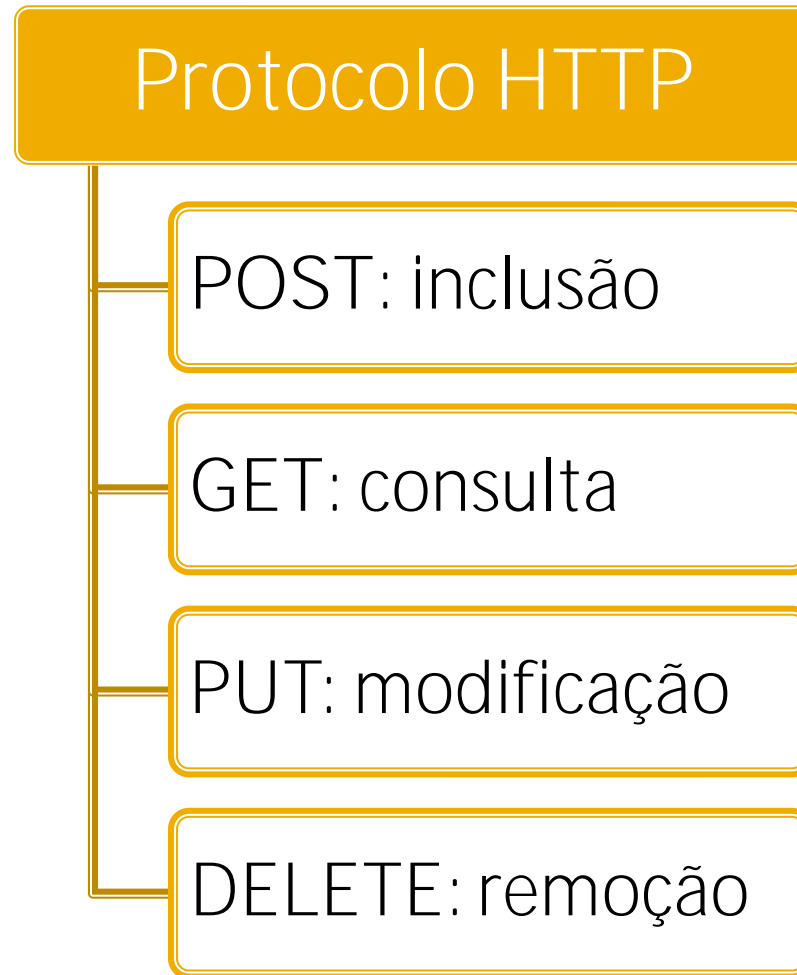
Perguntas?



REST

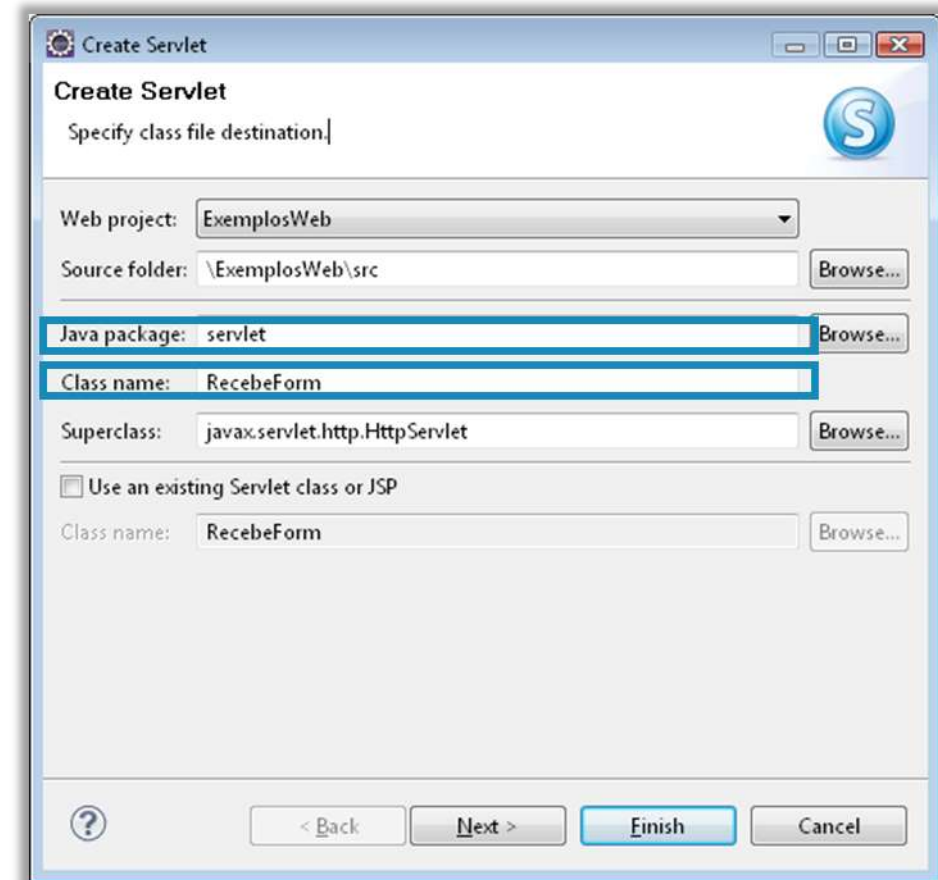
Representational State Transfer

REST: Representational State Transfer



Criando nosso primeiro servlet

- Selecione File → New → Servlet
- Informe o nome do pacote onde o servlet será criado
- Informe o nome da classe (será o nome do arquivo .java também!)
- (normalmente não é necessário modificar a informação de superclasse)
- Clique Next



Criando nosso primeiro servlet

- Informe o nome pelo qual o servlet será conhecido no mundo (normalmente, o mesmo nome da classe)

- Opcionalmente, dê uma descrição ao seu servlet (é apenas um comentário, para seres humanos)

- Clique Next

Create Servlet

Enter servlet deployment descriptor specific information.

Name: RecebeForm

Description: Servlet para receber um formulário post ou get

Initialization Parameters:

Name	Value	Description
------	-------	-------------

URL Mappings:

/RecebeForm

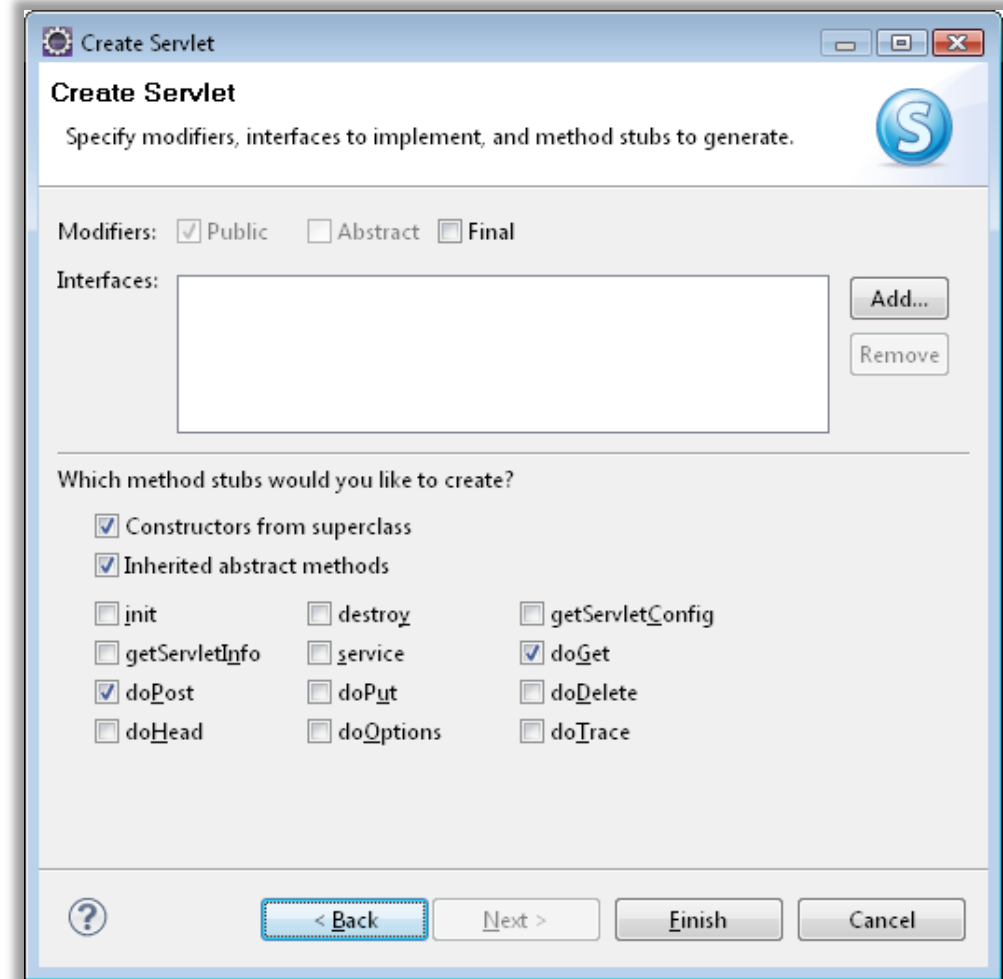
Endereço do servlet criado: /Projeto/RecebeForm

< Back Next > Finish Cancel

Criando nosso primeiro servlet

- Informe algumas opções necessárias para o seu funcionamento
- No nosso caso, agora, não preciso de mais nada...

- Clique Finish



Consulta linha de ônibus

Pacotes

```
package br.com.meslin.ufma.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.json.JSONArray;

import br.com.meslin.ufma.main.GetBuses;
```

```
/**
 * Servlet implementation class GetBusesByLine
 */
@WebServlet("/GetBusesByLine")
public class GetBusesByLine extends HttpServlet {
    private static final long serialVersionUID=1L;

    // constants
    private static final int LINHA = 2;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public GetBusesByLine() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```


Consulta linha de ônibus

Métodos de POST e GET – obtenção dos dados

```
/**
 * @see HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest
request, HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub
}
```

```
/**
 * @see HttpServlet#doGet(HttpServletRequest
request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException {
    GetBuses buses = new GetBuses();
    JSONArray busesJSON = buses.production();
```



Consulta linha de ônibus

Prepara os dados para enviar para o callback em AJAX

```
JSONArray busesByLineJSON = new JSONArray();
for(int i=0; i<busesJSON.length(); i++) {
    JSONArray aBus = busesJSON.getJSONArray(i);
    String linha;
    try {
        linha = Integer.toString(aBus.getInt(LINHA));
    } catch (Exception e) {
        linha = aBus.getString(LINHA);
    }
    if(linha.equals(request.getParameter("linha"))) {
        busesByLineJSON.put(aBus);
    }
}
```

```
ServletOutputStream out = response.getOutputStream();
out.print(busesByLineJSON.toString(2));
out.close();
}
```

Instalando o Postman

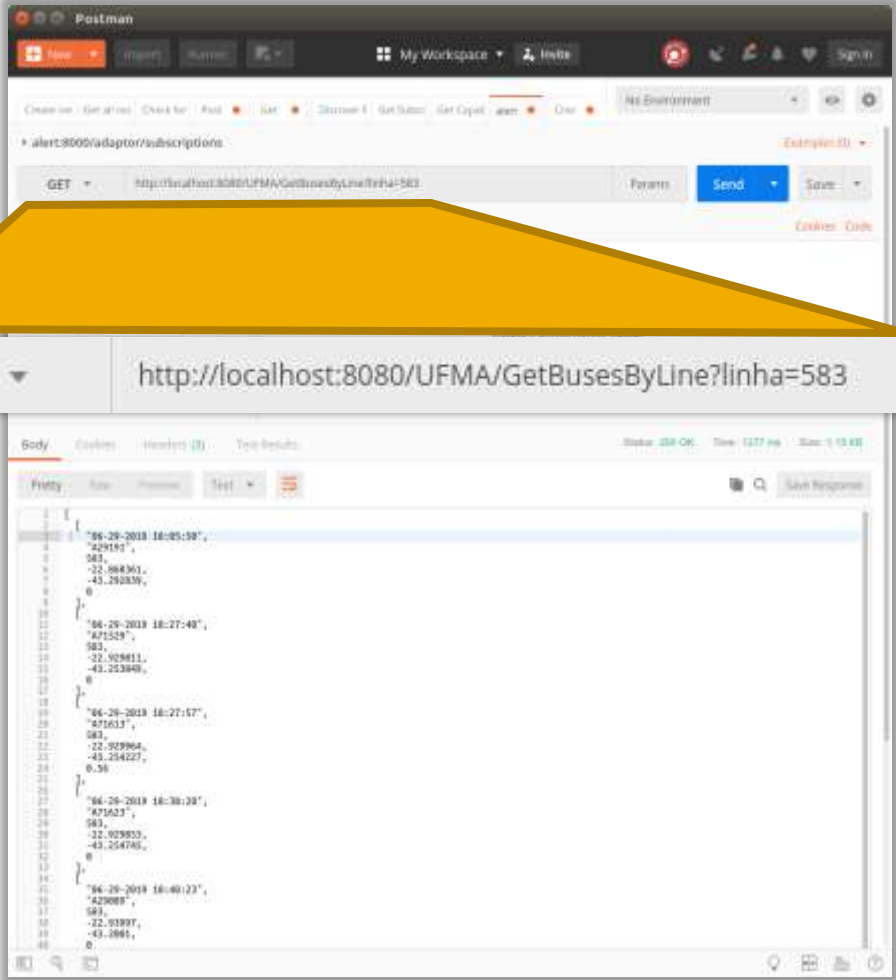
(se você ainda não tem)

- Fazer download do Postman a partir de <https://www.getpostman.com> para o diretório de Downloads.
- Descompactar o arquivo, copiar para o diretório /opt e criar o link simbólico com os comandos a seguir (atenção para a versão correta):

```
$ tar -xvzf Postman-linux-x64-6.3.0.tar.gz
$ sudo mv Downloads/Postman /opt/
$ sudo ln -s /opt/Postman/Postman /usr/bin/postman
```

Usando o Postman (Linux ou Windows)

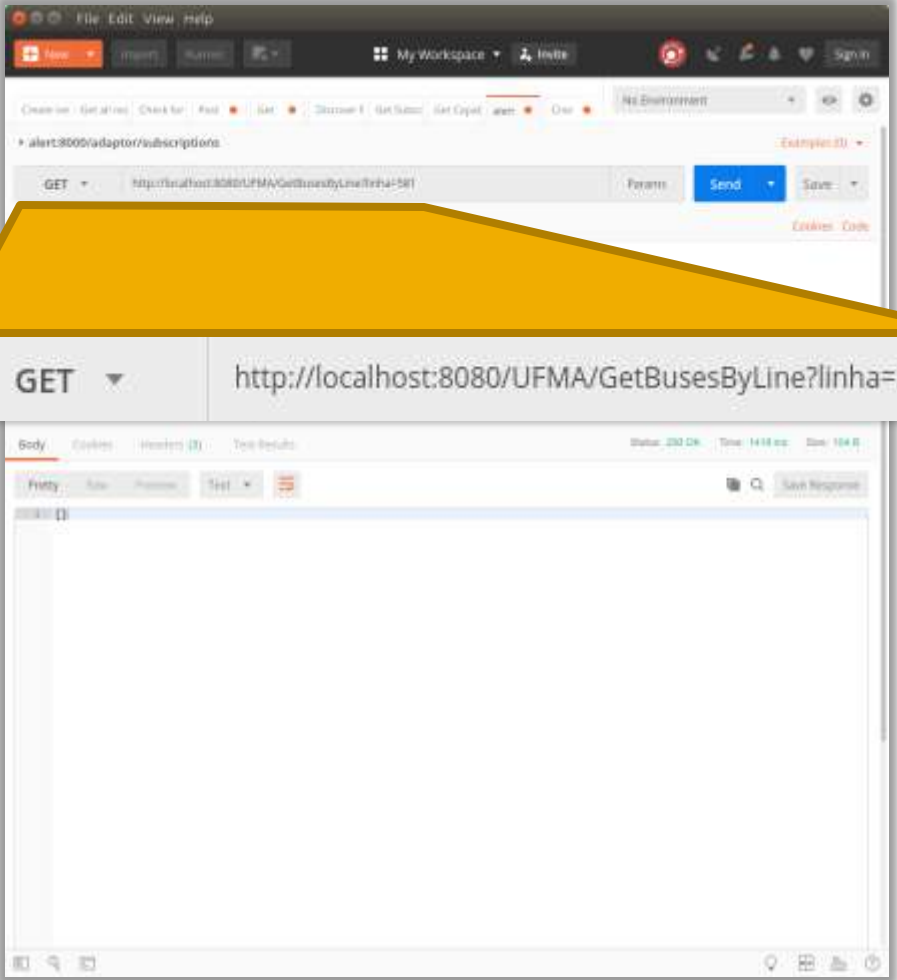
Testando...



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/UFMA/GetBusesByLine?linha=583`. The response body is displayed in a JSON array format, containing four bus entries with their respective IDs, line numbers, and coordinates.

Request: GET `http://localhost:8080/UFMA/GetBusesByLine?linha=583`

```
1 {
2   "06-29-2019 18:05:50",
3   "429189",
4   "583",
5   "-22.884361",
6   "-43.292839",
7   0
8 }
9
10 {
11   "06-29-2019 18:27:49",
12   "471529",
13   "583",
14   "-22.928611",
15   "-43.253849",
16   0
17 }
18
19 {
20   "06-29-2019 18:27:57",
21   "471637",
22   "583",
23   "-22.929964",
24   "-43.254227",
25   0.56
26 }
27
28 {
29   "06-29-2019 18:30:29",
30   "471623",
31   "583",
32   "-22.928853",
33   "-43.254745",
34   0
35 }
36
37 {
38   "06-29-2019 18:40:23",
39   "429089",
40   "583",
41   "-22.93997",
42   "-43.2861",
43   0
44 }
```



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/UFMA/GetBusesByLine?linha=581`. The response body is empty, indicating a failed or non-existent request.

Request: GET `http://localhost:8080/UFMA/GetBusesByLine?linha=581`

```
{}
[]
```

Exercícios

Exercício

- Crie um servlet para retornar todos os ônibus que estão se locomovendo entre uma determinada faixa de velocidade passada como parâmetro

Perguntas?

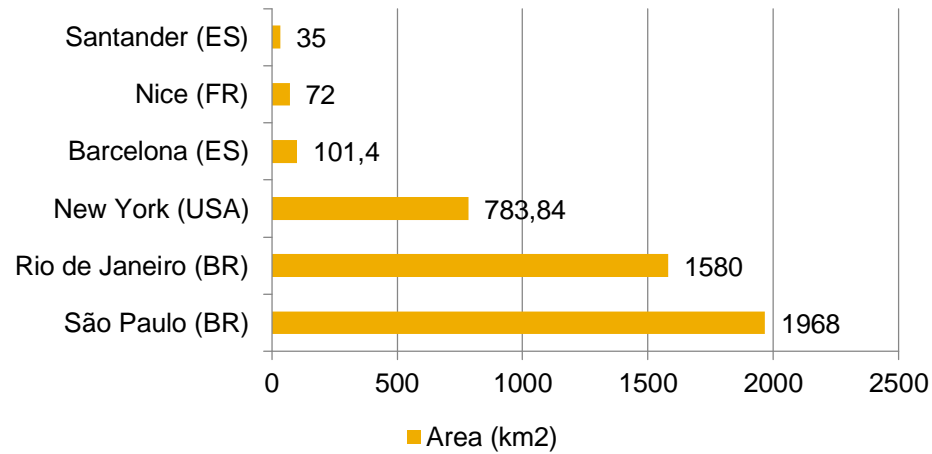


MUSANet

Mobile Urban Sensor and Actuator Network

What's the size of the problem?

Some Cities



Nice - France



Barcelona - Spain



Santander - Spain

Sources:
•Google Maps
•Wikipedia

Sensors used in Smart Cities

- Santander
 - ~ 1,500 sensors
- Barcelona
 - ~ 2,000 sensors
- Rio de Janeiro
 - ~ 8,000 sensors only in buses
- São Paulo
 - ~ 14,000 sensors only in buses
- Other problems:
 - Inhabitants can be sensors and/or actuators
 - Many sensors may be mobile
 - Data must be delivered in a timely manner

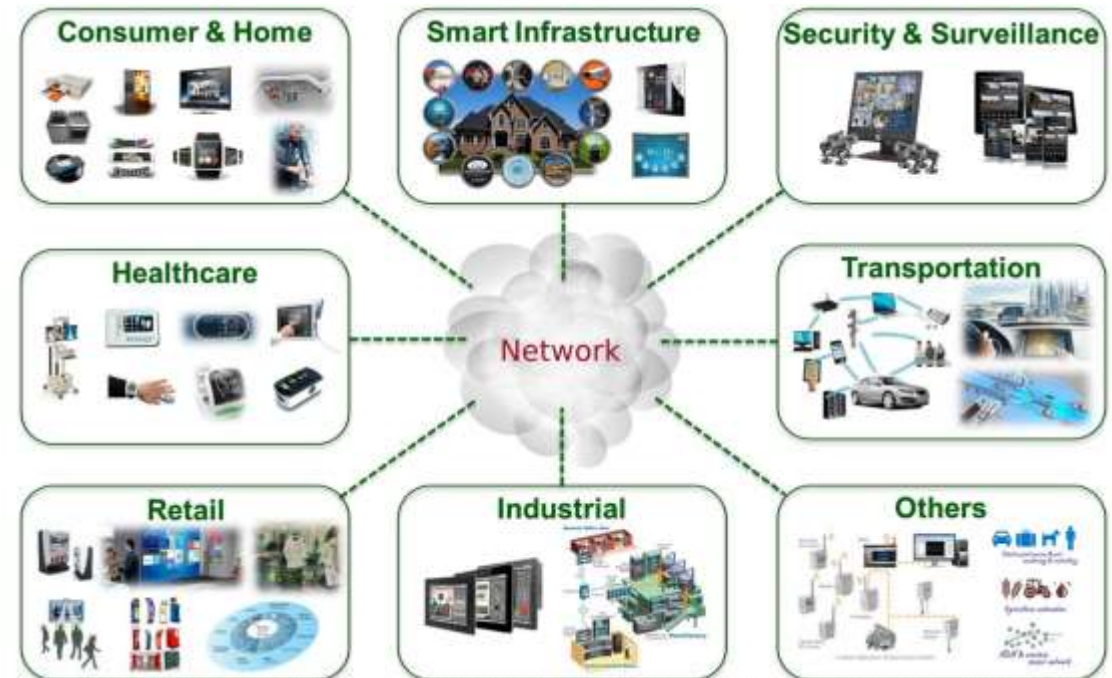


Our Goals

- Target
 - Very large city
 - Lots of sensors and actuators
 - Large population
 - Create an middleware to support (**large-scale**) WSN for smart cities
 - Hierarchical
 - Distributed
 - Cloud-based
 - Context-aware
 - Mobile nodes
- } Mobile!

Internet of Things (IoT)

- Things = devices with embedded processor, and sensors and/or actuators, able to connect wirelessly



Internet of Things (IoT)

Ou melhor Internet of Mobile Things (IoMT)

- Objects that can move or be moved
- ... possibly with few resources and only short/medium range connectivity (Bluetooth, ZigBee)
- ... but they should be connected to the Internet whenever possible



Parcel + SmartTags



Domestic Robot



Keychain



Car



Quadcopter/ Drones



Health Sensors



Beacons and Sensor Tags

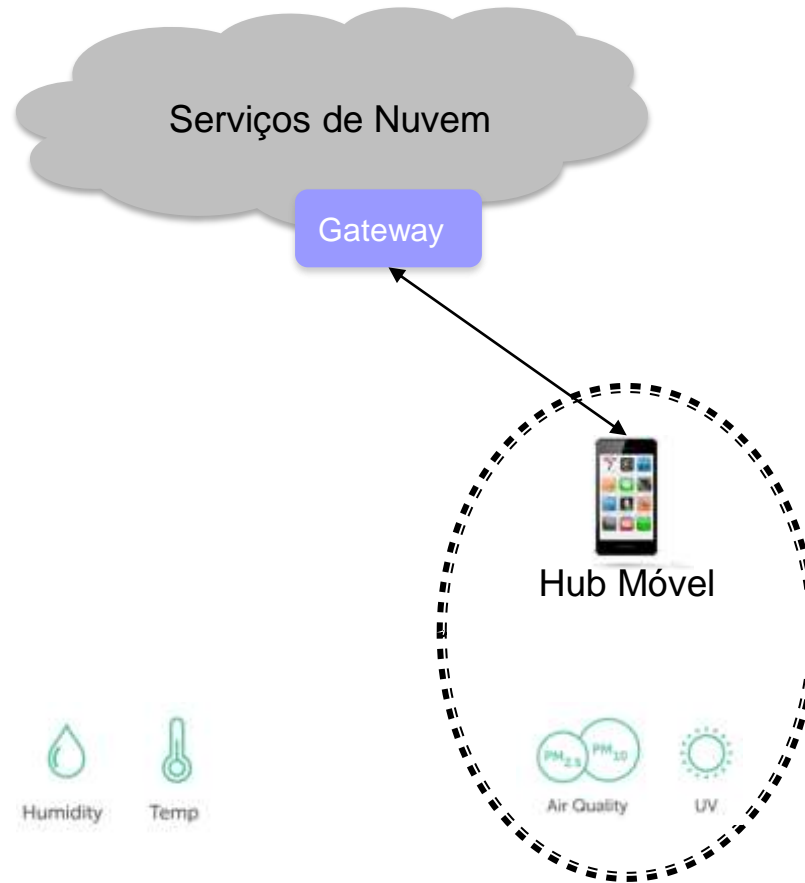


Toys

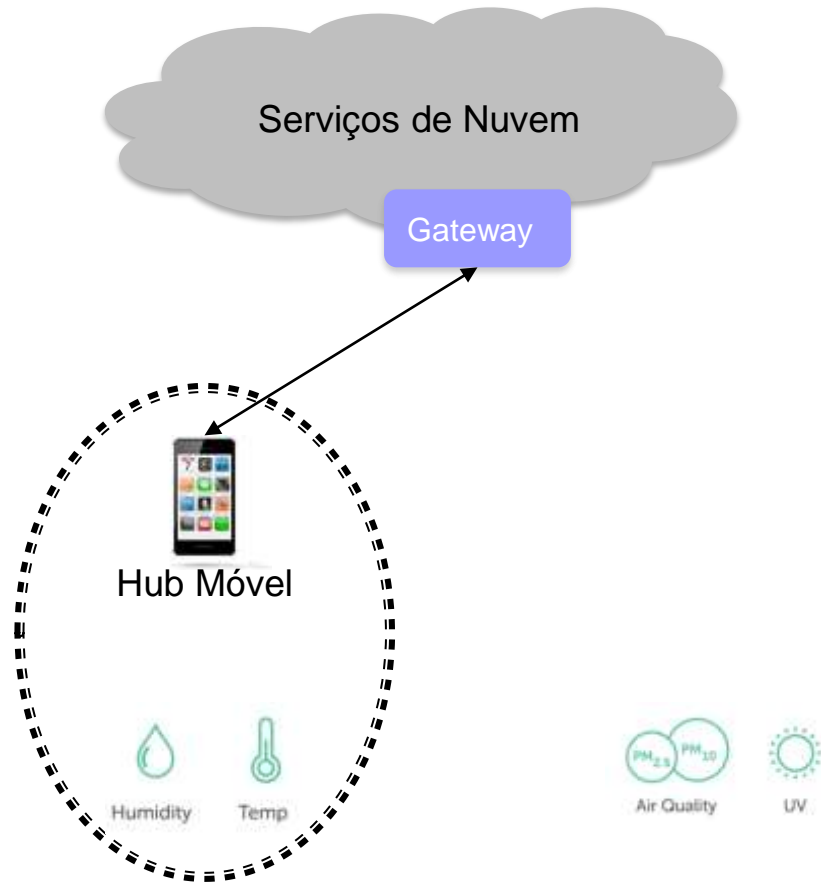
Internet of Mobile Things (IoMT)

- Abordagem: Usar smartphones como intermediários entre coisas inteligentes e serviços na nuvem.
- Grande presença de smartphones:
 - Mais de 2 bilhões de pessoas possuem pelo menos um smartphone. Até 2020 esse número deve alcançar 7 bilhões.
 - Brasil (2016): 38,3 milhões de usuários móveis.
 - Mais de 100 smartphones vendidos por hora no Brasil.
- Possuem:
 - Conectividade Internet (3G/4G/5G & Wi-Fi).
 - Razoável capacidade de processamento e armazenamento.
 - Vários sensores (GPS, acelerômetro, magnetômetro, etc.).
 - Crescente suporte à tecnologias WPAN:
 - Bluetooth 4.0/ Low Energy.
 - NFC.

Cenário em IoMT



Cenário em IoMT



Exemplo de Aplicação IoT/loMT

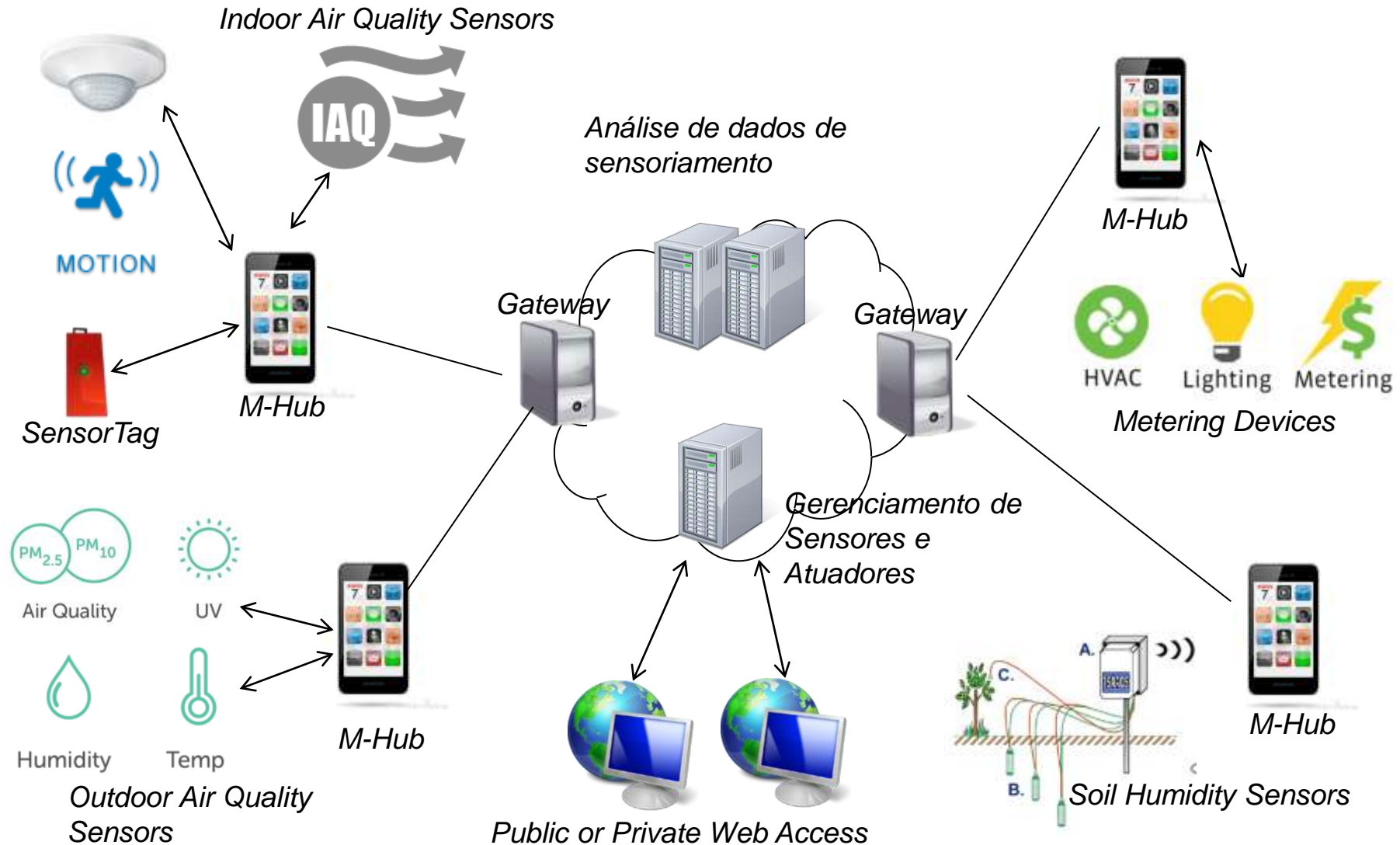
- Detecção de focos de incêndio florestal.
- Incêndios elevam ligeiramente a temperatura e reduzem significativamente a umidade relativa do ar.
- Sensores de temperatura e umidade (com BLE) podem ser fixados em placas de sinalização nas estradas que passam pela floresta.
- Carros que passam pela região poderiam enviar os dados coletados para uma central de monitoramento.
- Assim, seria possível detectar o foco do incêndio e por onde está se alastrando.



Aplicações de IoMT

- Sensoriamento ambiental participativo (*crowd sensing*).
- Smart Home / Smart Office.
- Cidades Inteligentes: Intelligent Transportation Systems, Smart Parking, etc.
- Healthcare.
- Logística.
- Varejo.
- Lazer.
- Smart Manufacturing com robôs móveis.

IoMT : Sensoriamento Participativo e Oportunístico

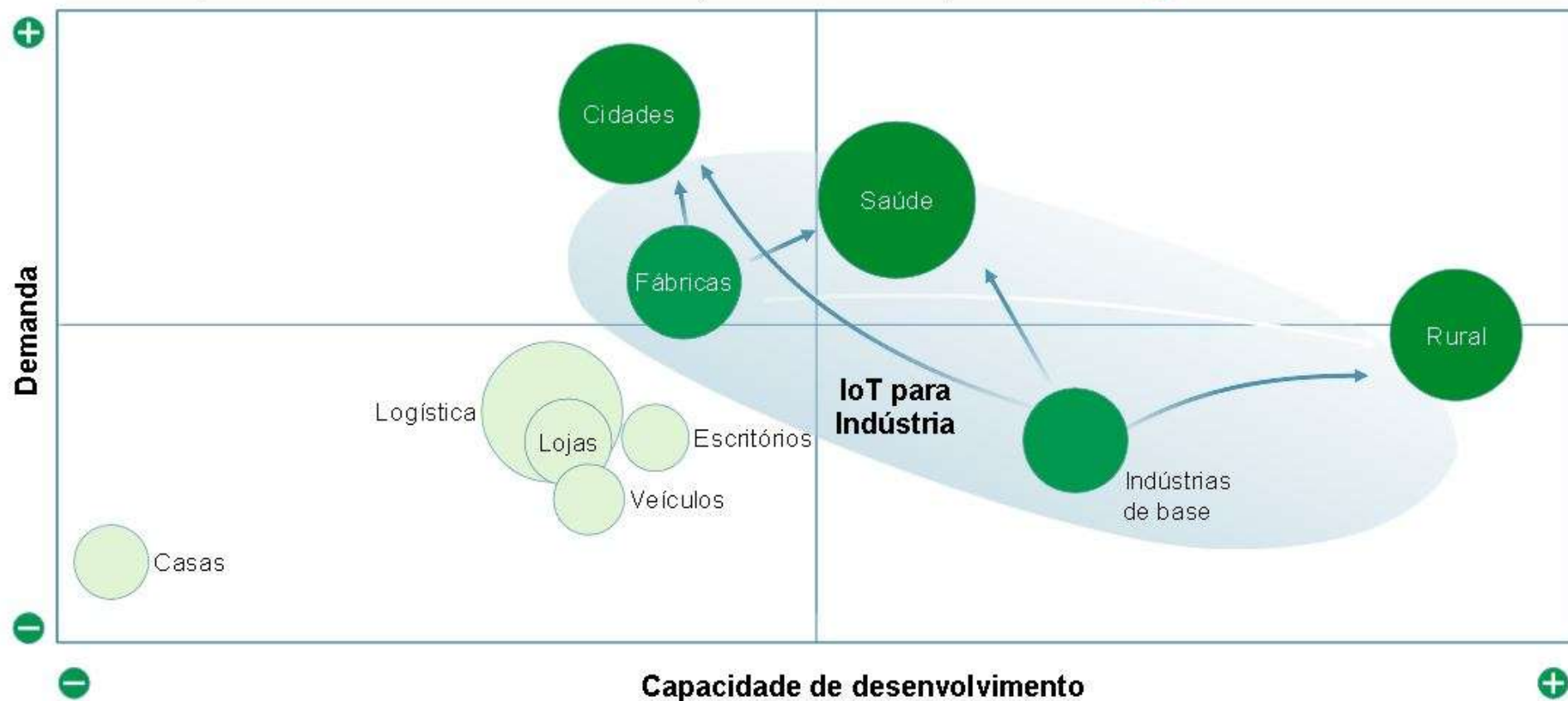


Plano Nacional de IoT para o Brasil (2018-2020)

A matriz de priorização destacou quatro Frentes Prioritárias de IoT para o país

Demanda x Capacidade de desenvolvimento x Oferta (tamanho do círculo)

● Frente Prioritária



MUSANet Layers

MUSANet – Mobile
Urban Sensor and
Actuator Network

Three-layer
architecture

InterCity	Storage Data Visualization Structured Queries Resource Catalog
ContextNet	Gateway Group Definer Processing Node CEP
Mobile-Hub	Bluetooth WiFi 3G/4G CEP

Lower Layer: Mobile Hub

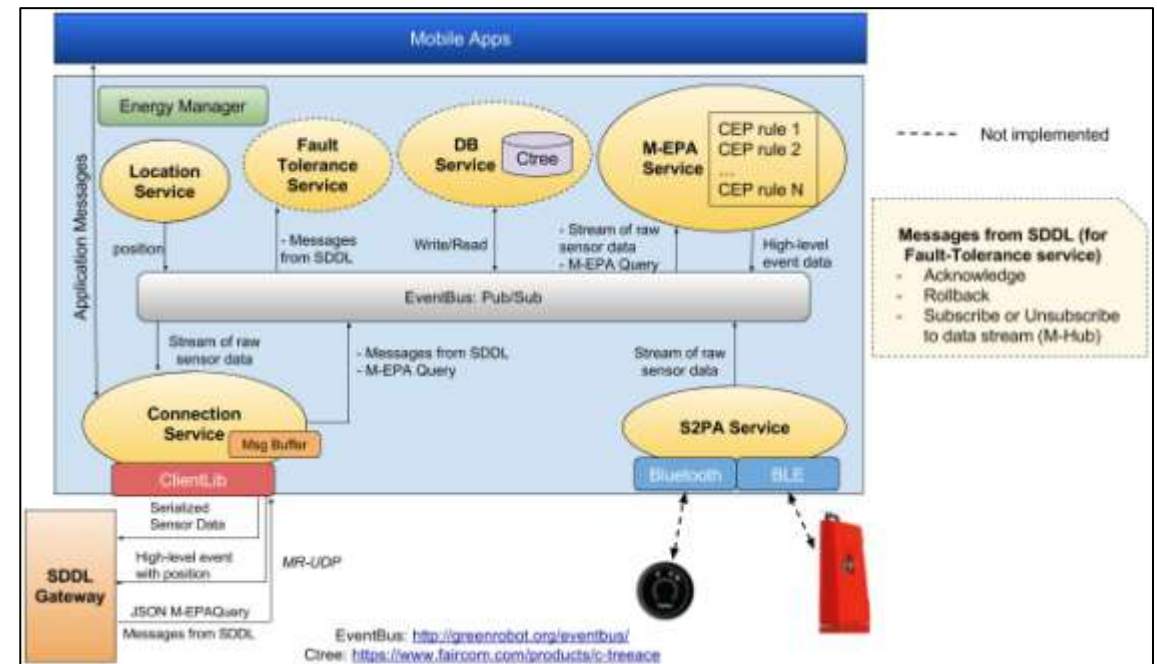
Componente do middleware que executa em Android e provê:

- **Comunicação** com Gateways do SDDL e outros M-Hubs.
- **Descoberta de smart objects** nas proximidades (usando Bluetooth Classic e BLE).
- Serviço de **data stream processing** (para filtragem & sumarização, agregação de dados de sensores) e tomada local de decisão.
- Possibilita adição de **informação contextual** aos dados coletados (tempo, posição, etc.) provida pelos sensores embutidos no smartphone.
- Serviço de **gerenciamento de energia** parametriza a execução dos demais serviços (frequência de *scan* para descoberta, comunicação com Internet, etc.).
- Serviço de persistência de dados (DBS) e serviço de gerenciamento de atuação (M-ACT) sobre smart objects (**em desenvolvimento**).

Lower Layer: Mobile Hub

Sensor discovery and data collection

- Sensor gateway
 - Bluetooth
 - BLE
- CEP (Complex Event Processing)
- Context-aware



Mobile Hub

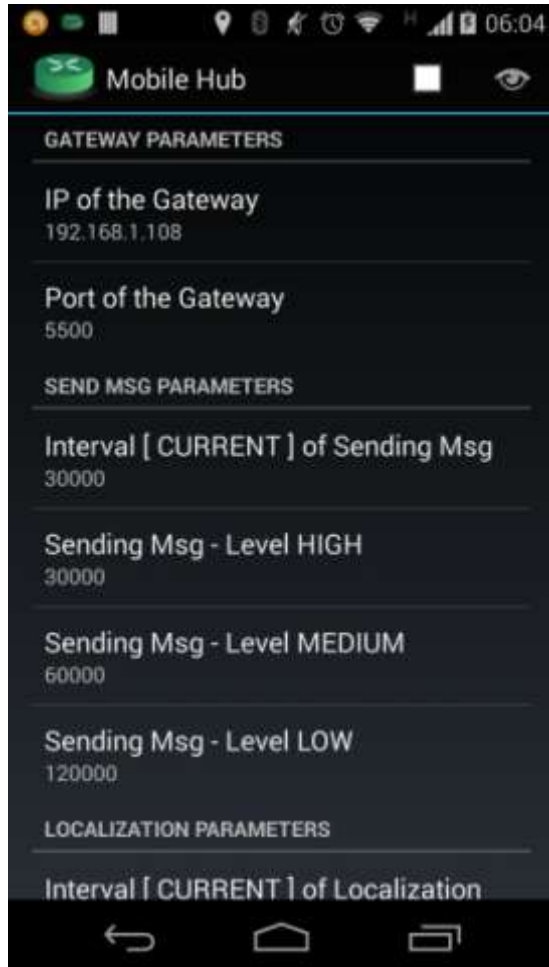


CC2541 SensorTags (Texas Instruments)

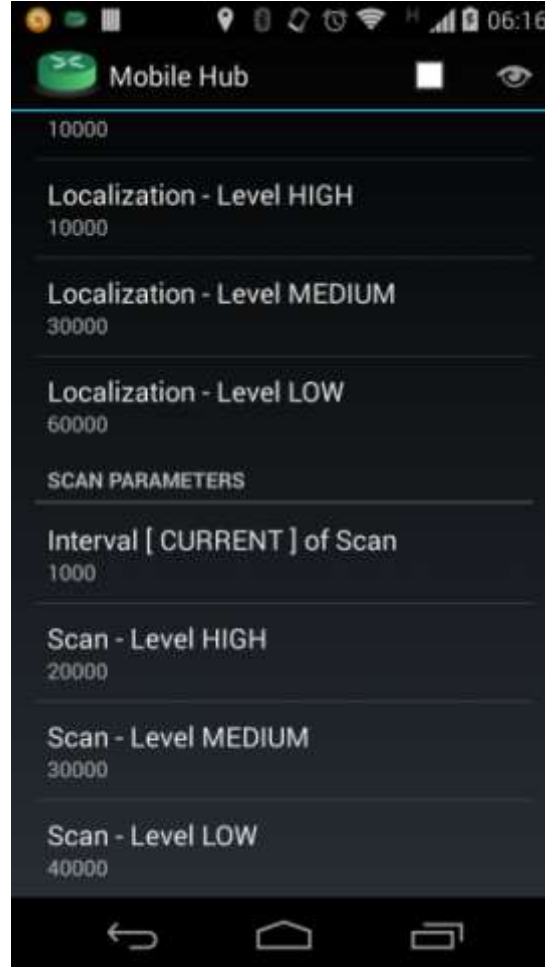


M-Hub conectado ao Zephyr BioHarness-3

Interface do Mobile Hub



Parâmetros de Conexão

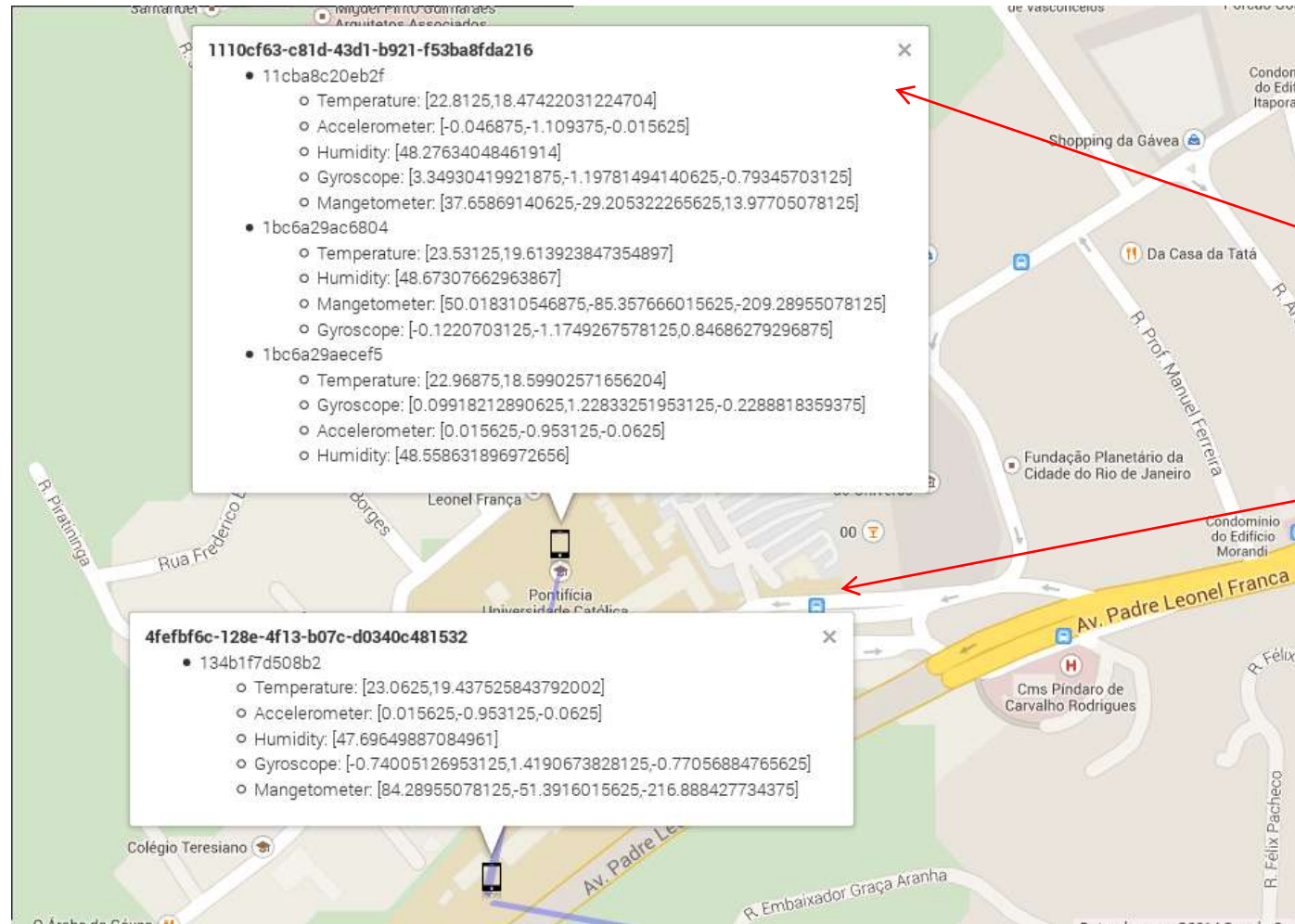


Parâmetros de Scan



Objetos Conectados

Exemplo de Aplicação do Mobile-Hub



3 M-OBs próximos ao M-Hub-1

1 M-OB próximo ao M-Hub-2

Middle Layer: ContextNet

Objetivos:

- Arquitetura de software escalável para comunicação móvel e processamento com baixa latência.
- Processamento de mobile data streams.
- Complex Event Processing (CEP) distribuído.
- Ciência de Contexto e adaptação.
- Suporte a colaboração e coordenação mobile-mobile.

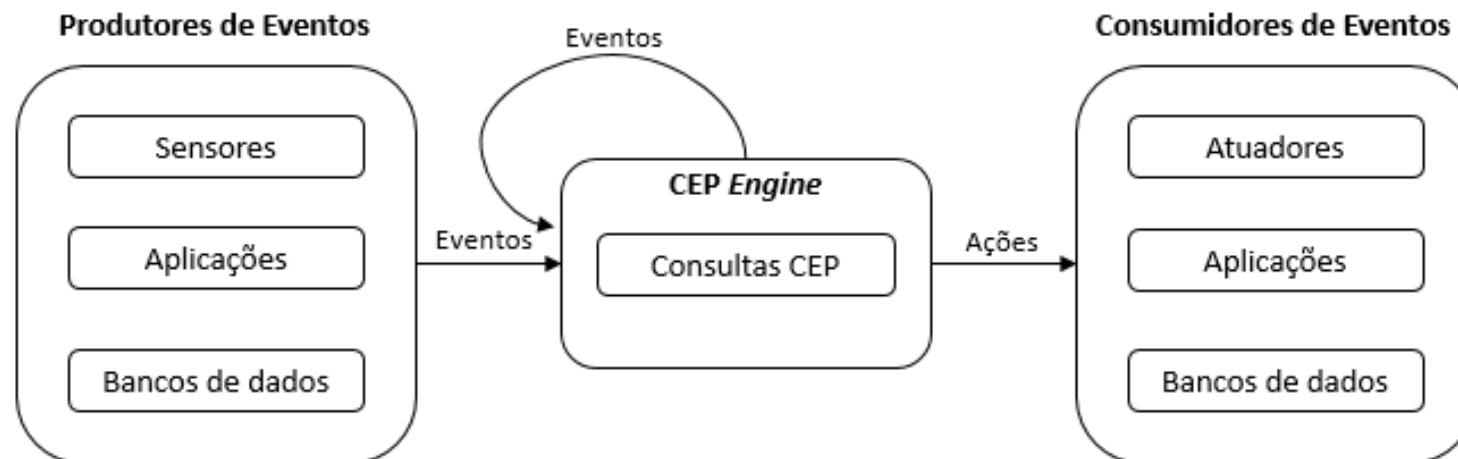
O middleware mobile-nuvem (Scalable Data Distribution Layer – SDDL) provê:

- comunicação escalável mobile-mobile, multi-ponto e com confiabilidade.
- balanceamento de conexões e data streams, handover transparente.

Usado em vários protótipos de pesquisa.

Complex Event Processing

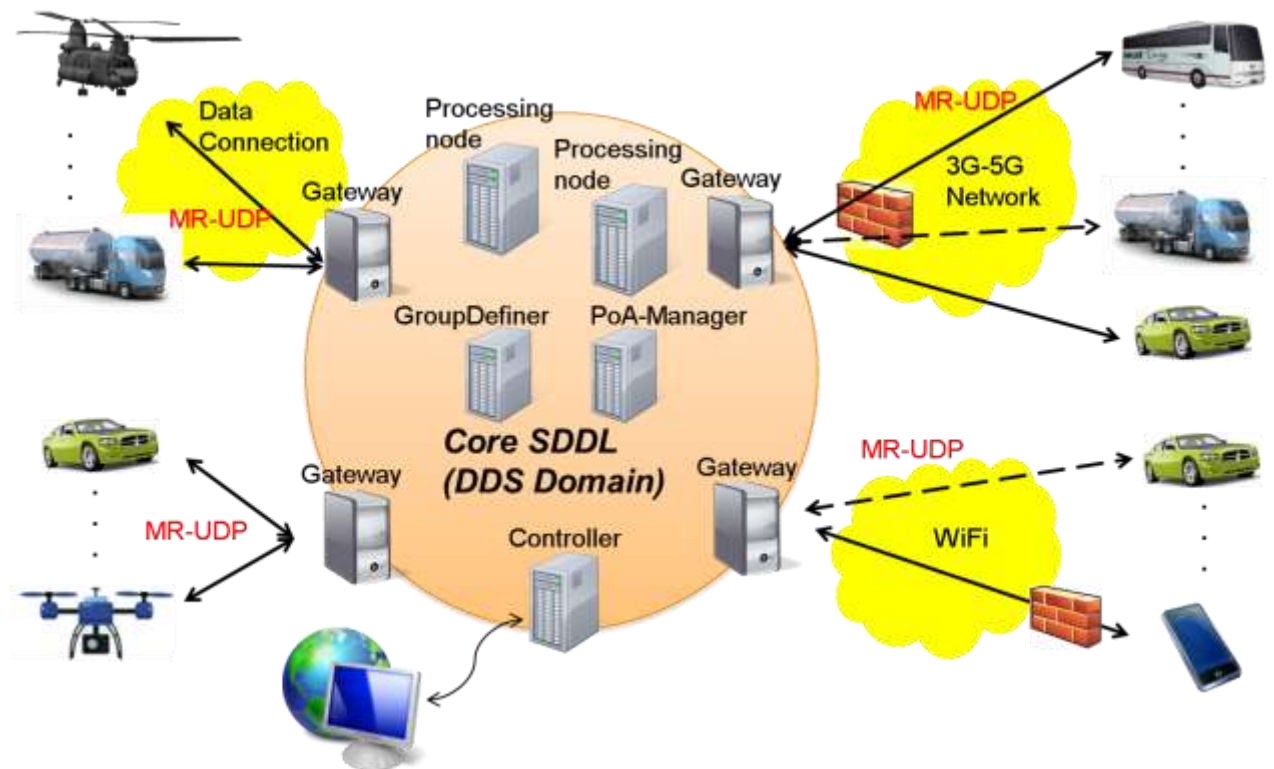
- Paradigma (e Tecnologia) para analisar e manipular **fluxos de eventos** utilizando consultas contínuas (on-line).
- Linguagem CQL (*Continuous Query Language*):
 - **SELECT * FROM SensorData.win:time(5 s) WHERE sensorName LIKE '%temperature%' AND latitude > -22 AND latitude < -21 AND longitude > -43 AND longitude < -42**
 - Expressividade para elaboração de consultas.



ContextNet

Reliable wireless communication
between mobile and cloud

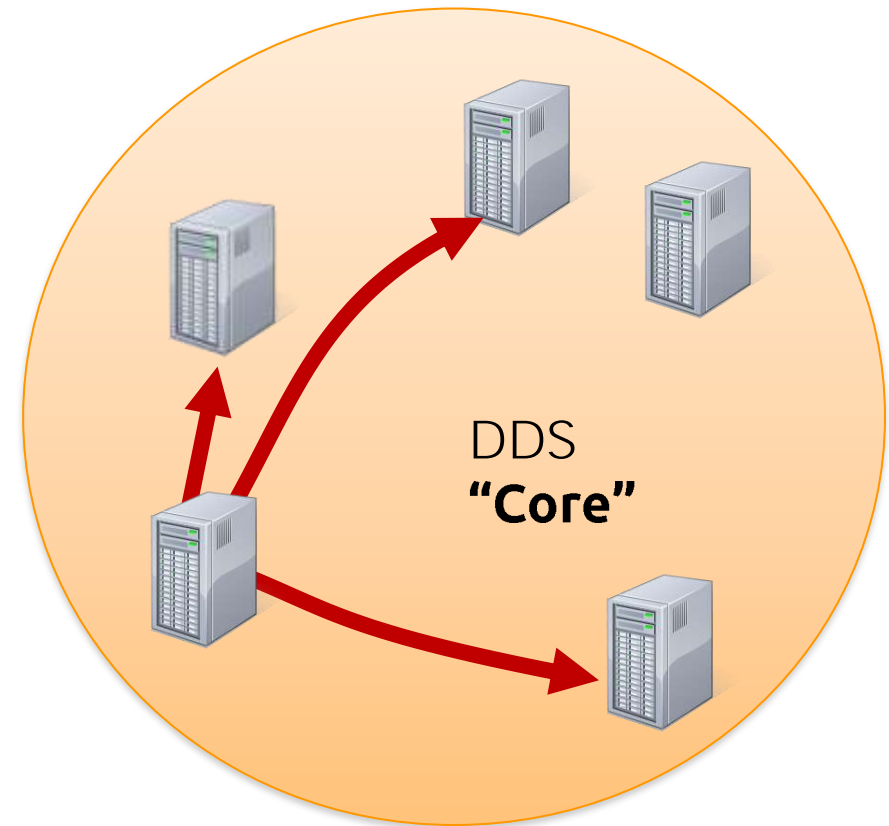
- SDDL Core
- Mobile nodes
 - MR-UDP over IP



M. Endler et al, "ContextNet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking," in Proceedings of the Workshop on Posters and Demos Track. ACM, 2011, p. 2.

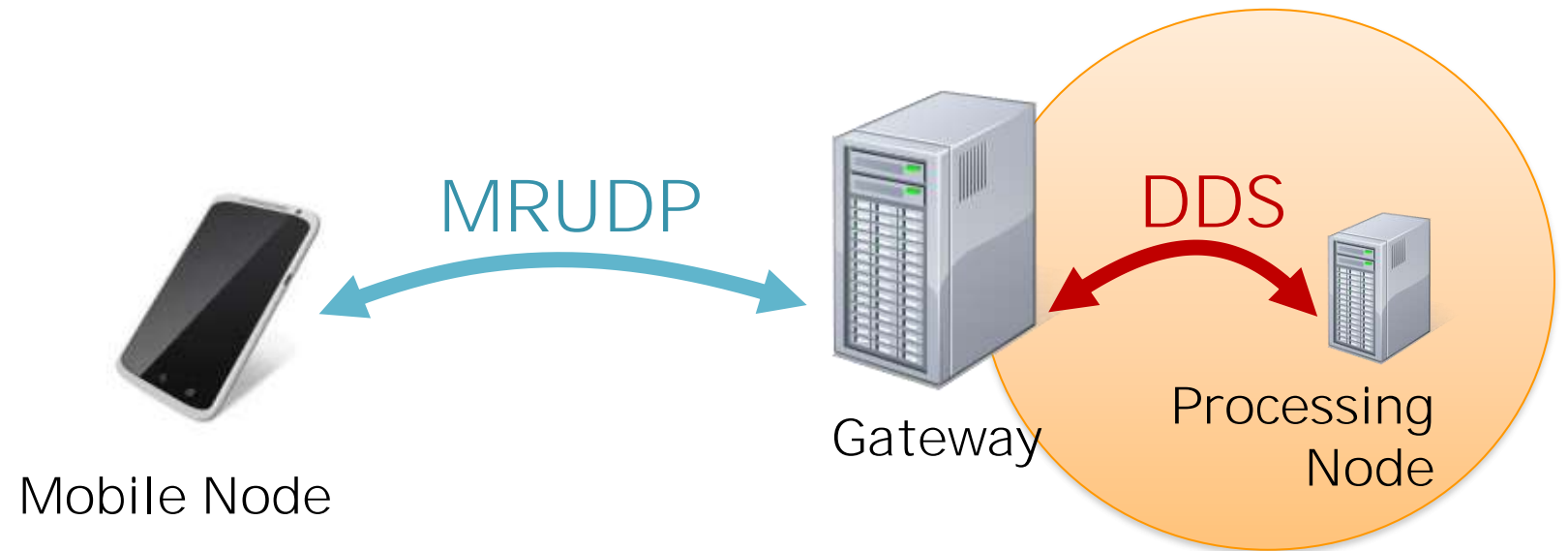
Data Distribution Service™

- Especificação OMG de um middleware Publish-Subscribe para comunicação em tempo real e de alta vazão em sistemas de larga escala.
- DDS define uma arquitetura P2P completamente descentralizada.

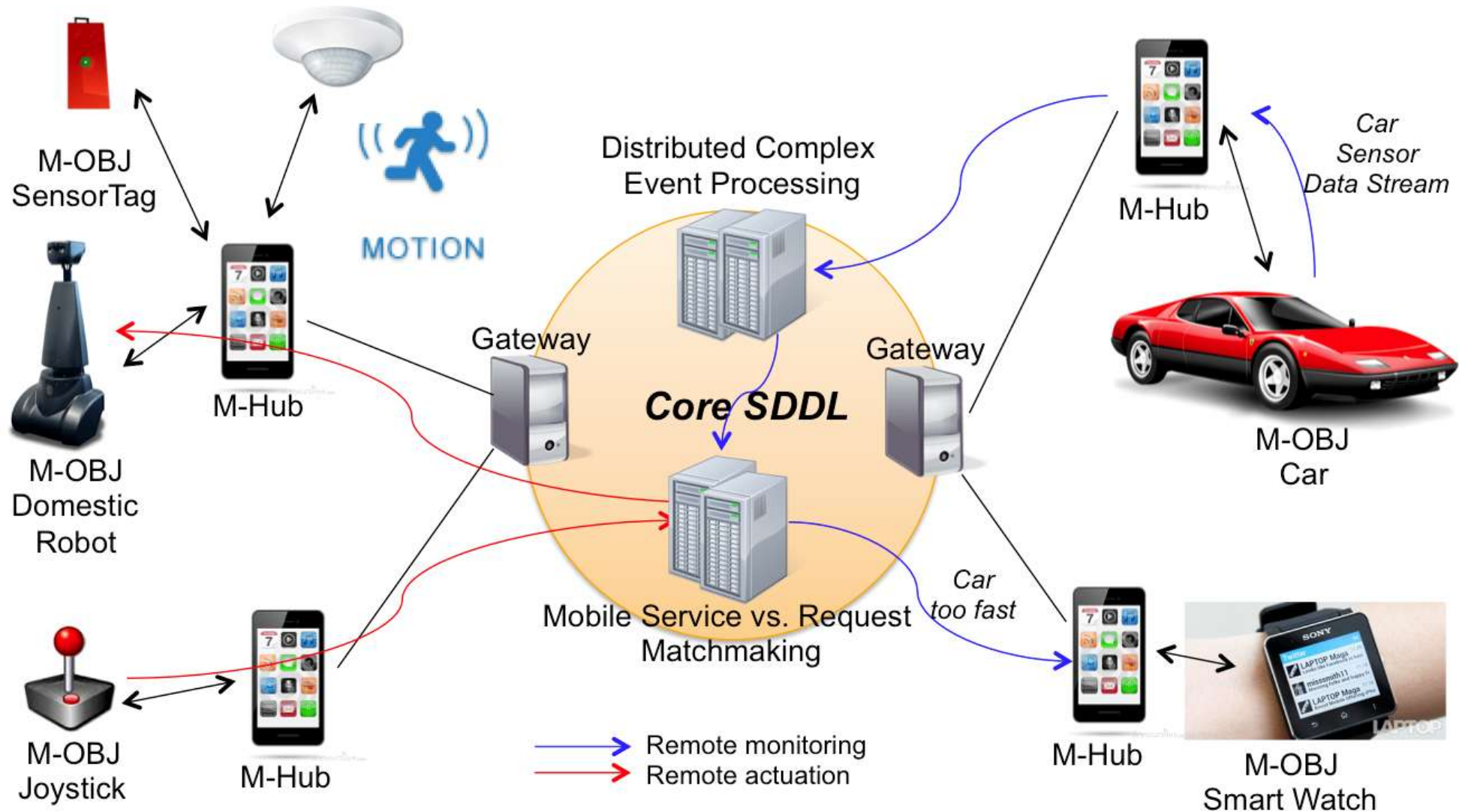


Protocols

- MRUDP – Mobile Reliable UDP.
- DDS – Data Distribution Service.



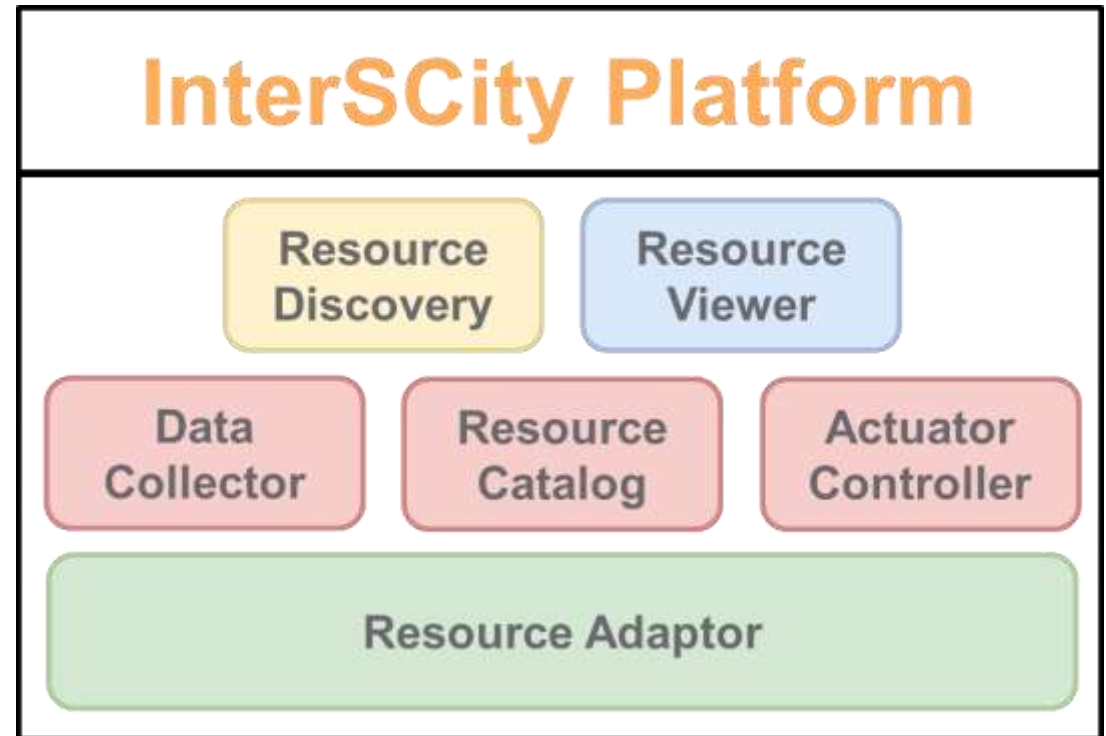
ContextNet: Arquitetura



Upper Layer: InterSCity

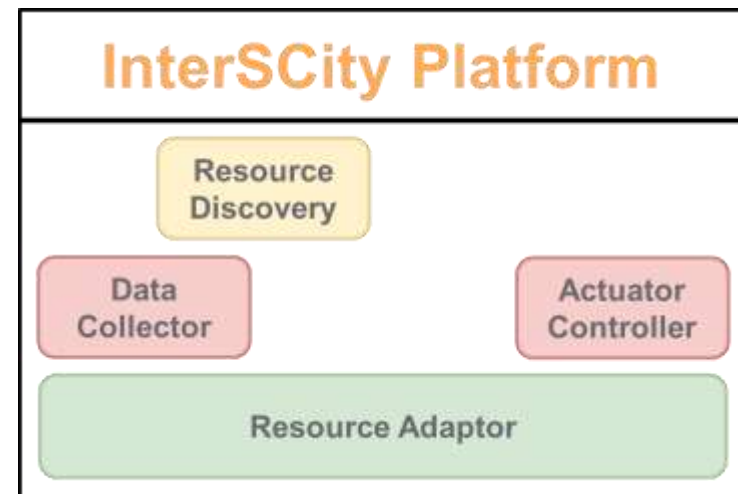
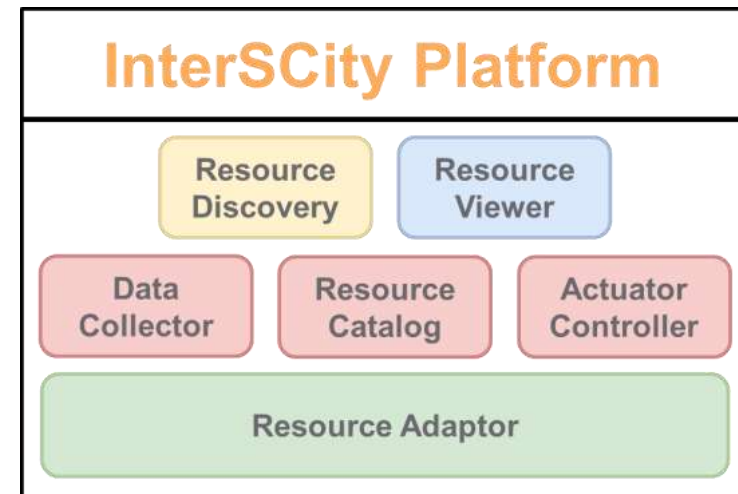
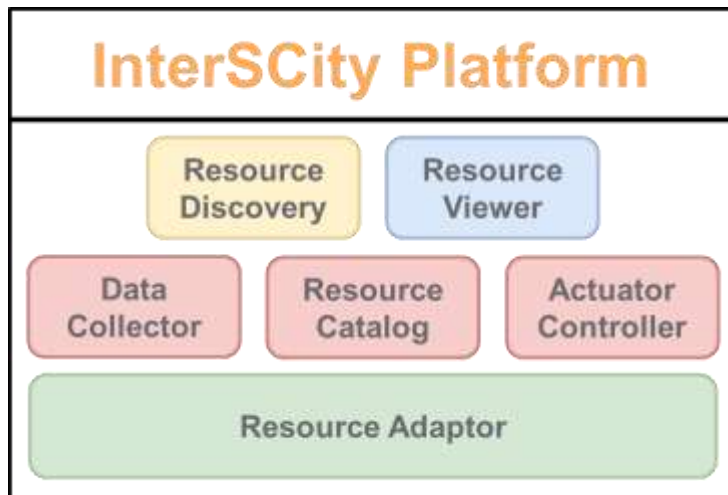
Model based storage and retrieval of sensor data

- Microservices
- Access over HTTP (Intranet)
- Structured queries
- Parallel execution of the platform in clusters

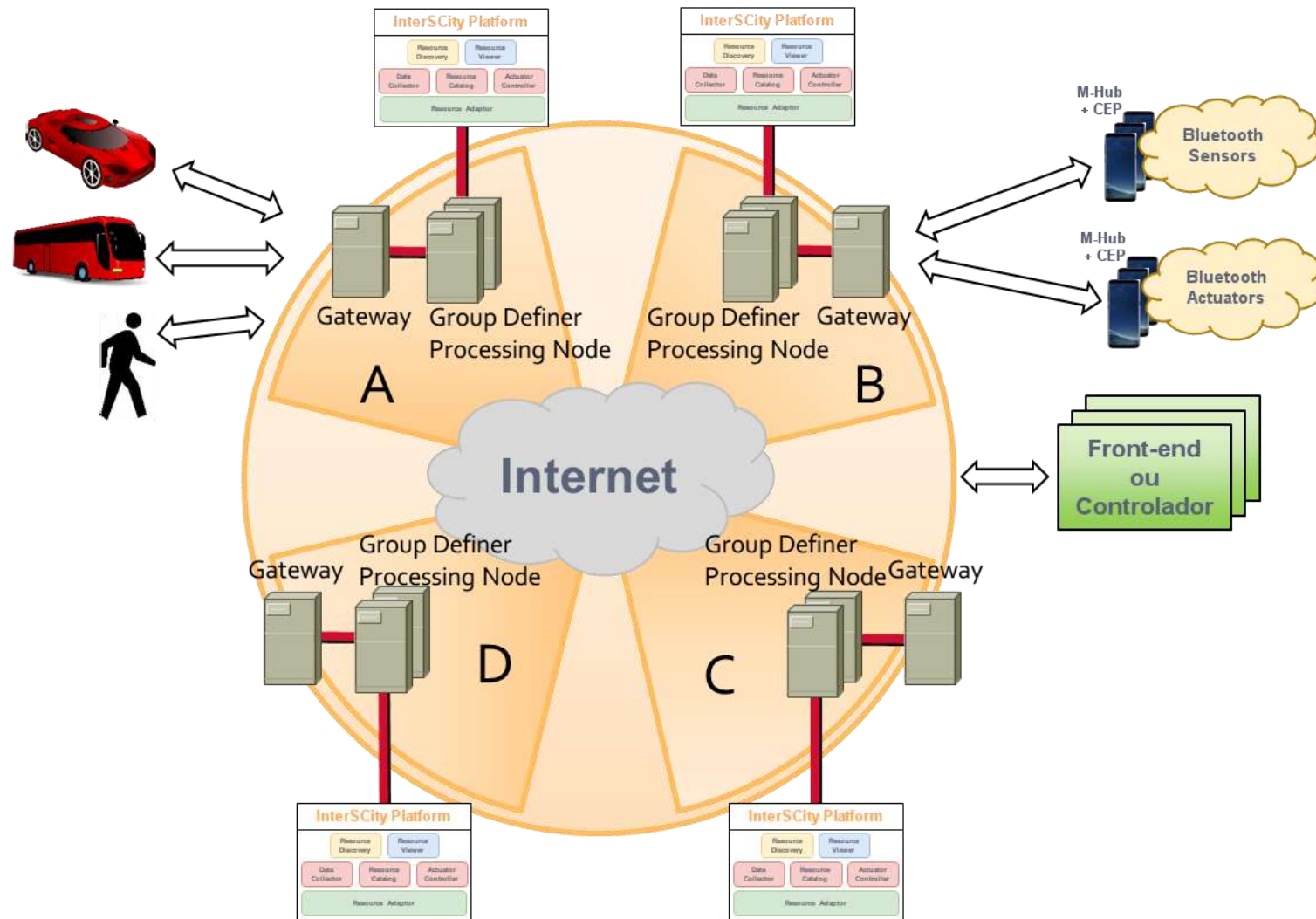


InterSCity

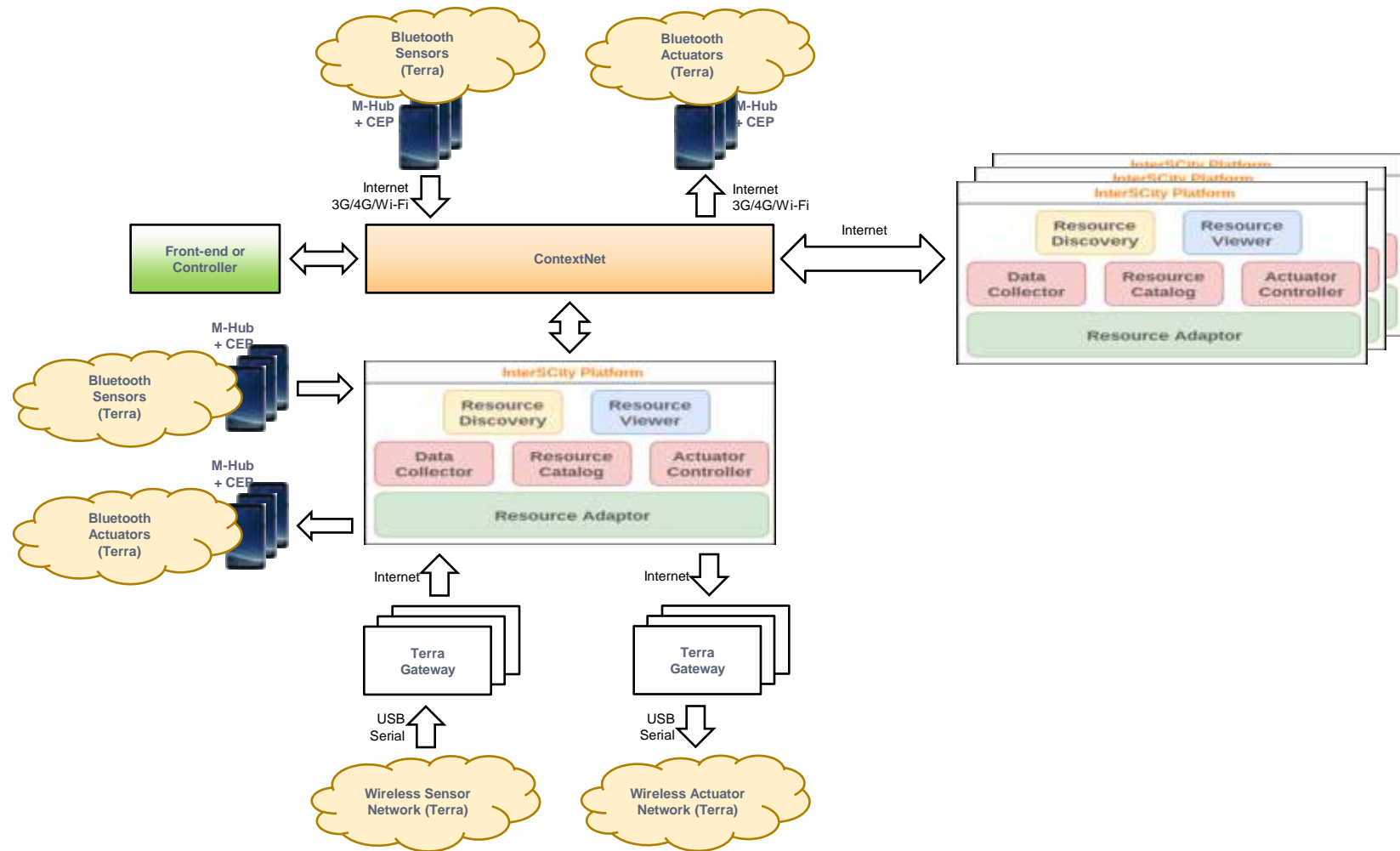
- Microservice replication



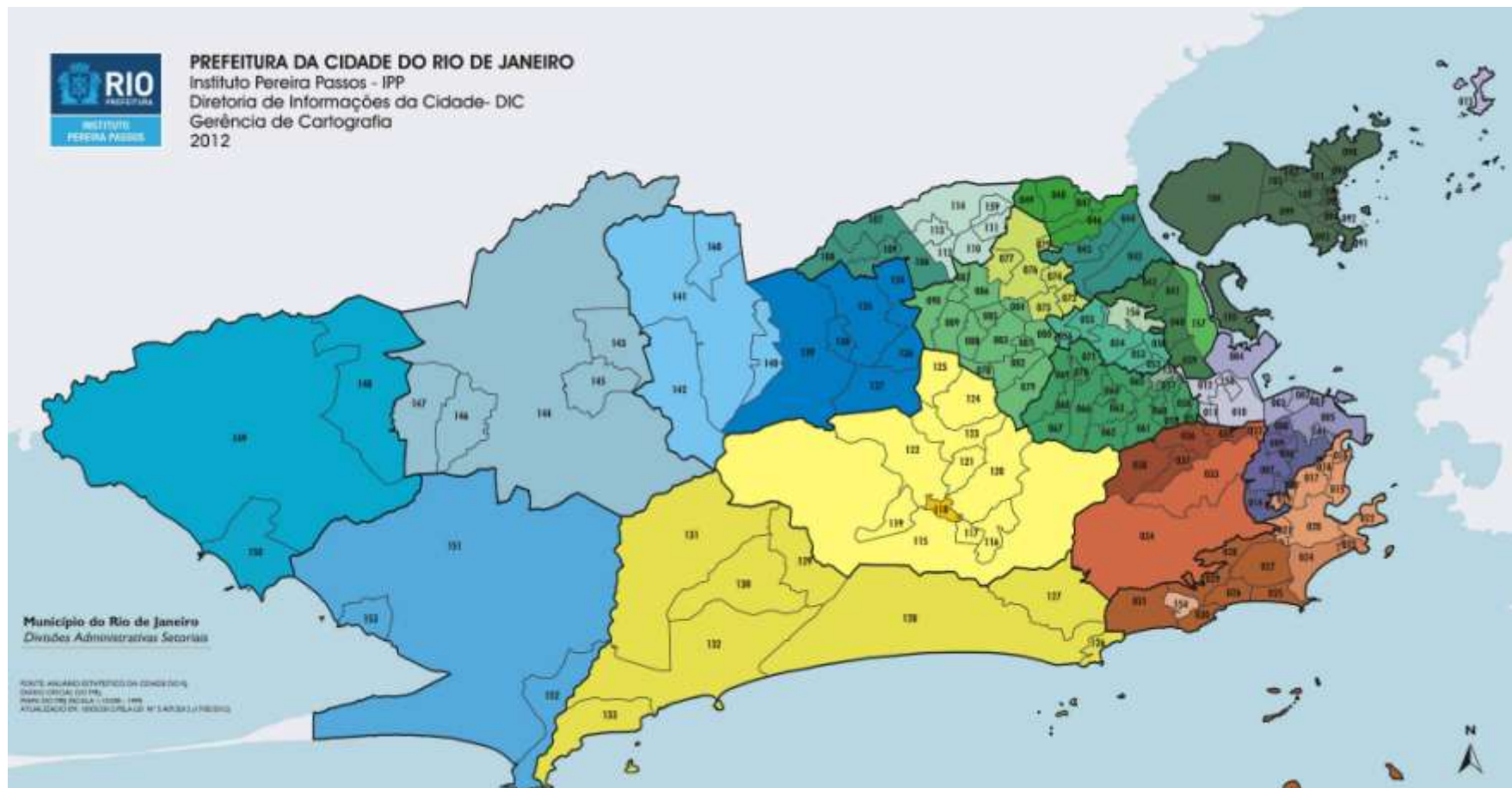
MUSANet



MUSANet



Rio de Janeiro e seus bairros



Perguntas?



PERGUNTAR NÃO OFENDE

Instalando e conhecendo a plataforma InterSCity

<http://interscity.org/>

Slides InterSCity

- Os slides a seguir foram extraídos da apresentação de Arthur Del Esposte, desenvolvedor do InterSCity
 - <https://www.youtube.com/watch?v=kLxIUbVFyOA&t=420s>
 - https://docs.google.com/presentation/d/1QGjurOAoCO0vdEc0ZuBJoAy8Vi8TkpmczvzoKFmWMPc/edit#slide=id.g228574380c_0_0

Horizontal Solutions

Transportation Systems

Health Care Systems

Traffic Management Systems

City Management Systems

Smart City Platform



* Icons from [Flaticon](#)

Definition of Smart City Platform

“An integrated middleware environment that supports software developers in designing, implementing, deploying, and managing applications for Smart Cities.” - (Santana et al, 2016)

Health Care
Systems

Traffic
Management
Systems

Smart City Platforms



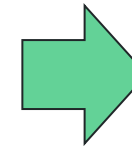
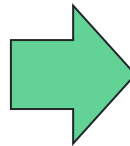
Opportunities

**Proprietary
Solutions**

**Lack of
Validations**

**Poor
Reusability**

**Ad-hoc
Solutions**



**Reusable Open
Software & Data**

**Practical and
Scientific
Validations**

**Flexible and
Evolvable
Architecture**

**Multidisciplinary
Research**

The InterSCity Platform

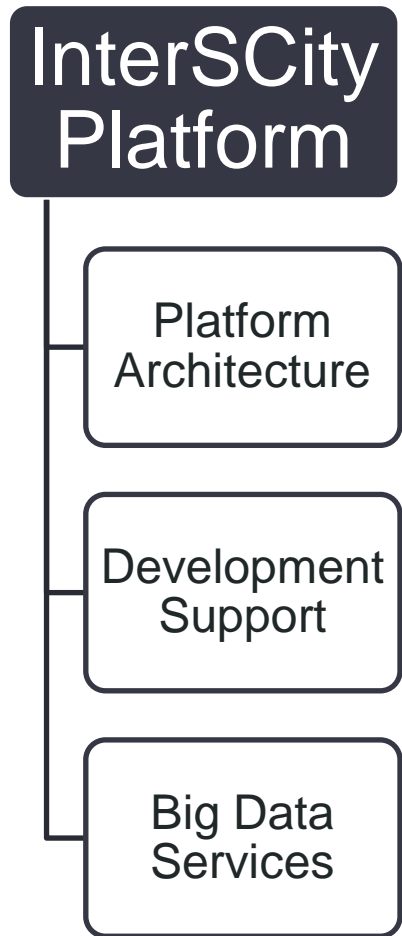
- The InterSCity platform is an **open-source microservice-based** project to enable novel smart city research, development, and deployment initiatives
- It provides a set of high-level **cloud-based services** that mediate the communication between the applications and the city devices



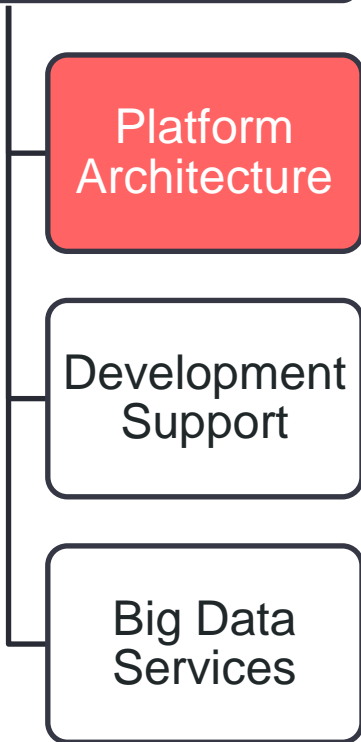
<https://gitlab.com/groups/smart-city-software-platform>

Licensed under **MPL 2.0**

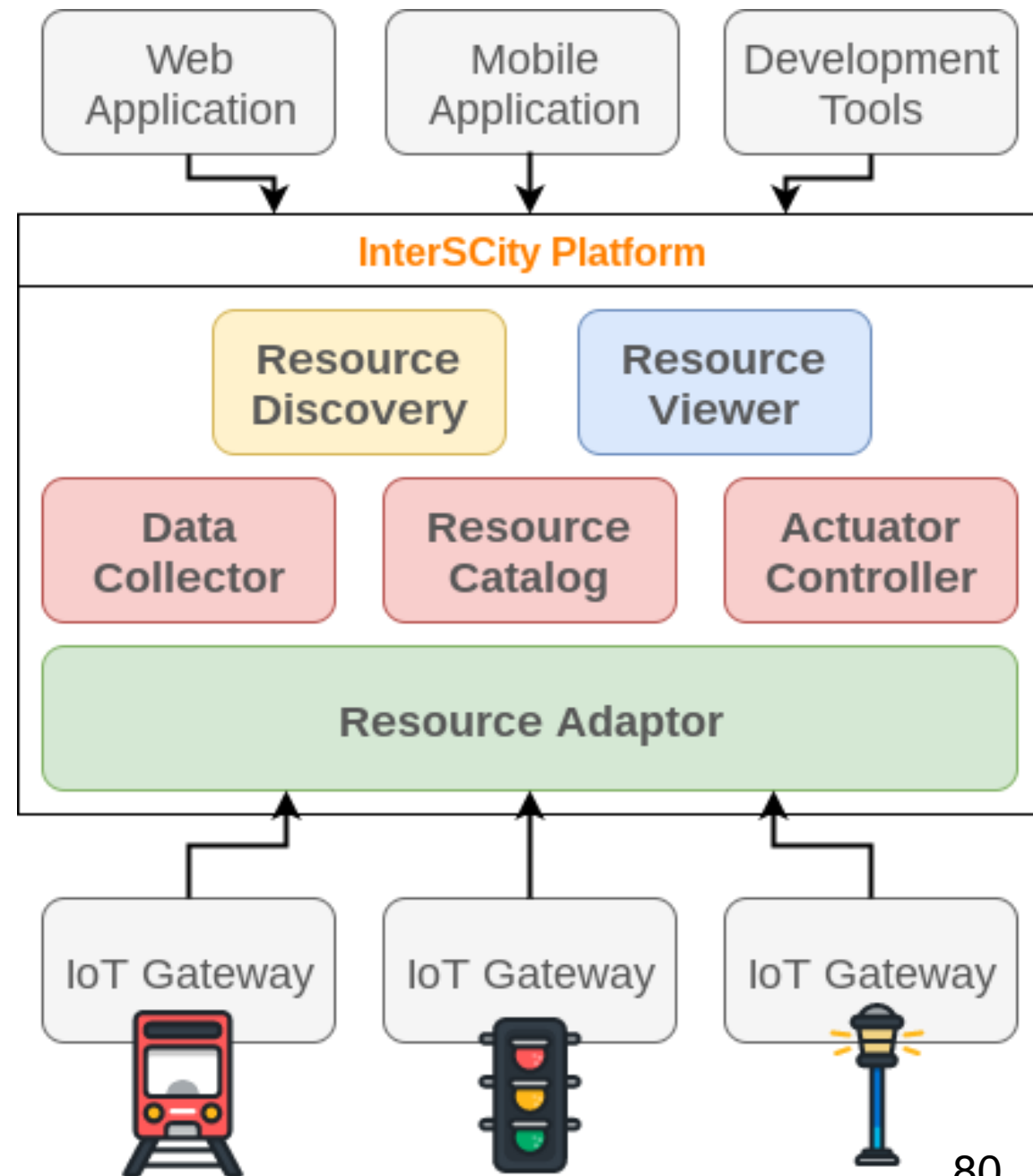
Research Lines



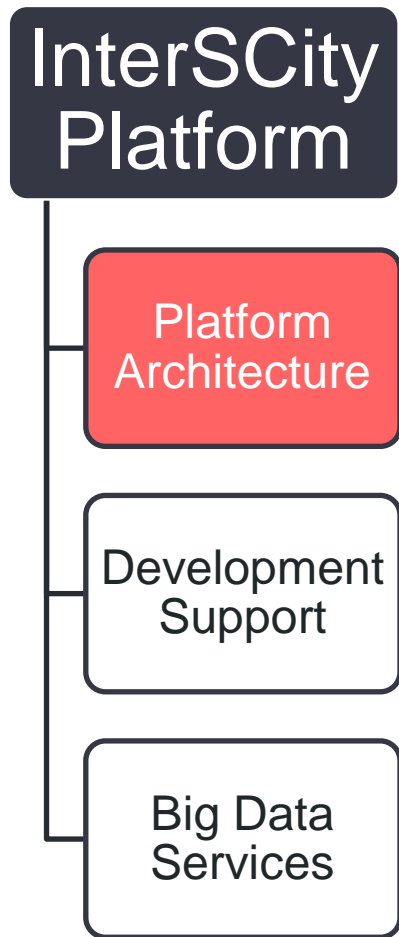
InterSCity Platform



InterSCity Platform overview, boundaries, and microservices



Why Microservices for Smart Cities?



- Microservices is a software architecture style composed of a collection of loosely coupled, fine-grained services
- Refining traditional SOA guidelines
- Microservices communicate with lightweight mechanisms
- **Support several non-functional requirements (mainly scalability and evolvability)**
- **Horizontal scalability through functional decomposition**

InterSCity Platform

Platform Architecture

Development Support

Big Data Services

Manage heterogeneous city resources

Integrate with IoT Systems

Data storage and processing

Context-aware resource discovery

Web Application

Mobile Application

Development Tools

InterSCity Platform

Resource Discovery

Resource Viewer

Data Collector

Resource Catalog

Actuator Controller

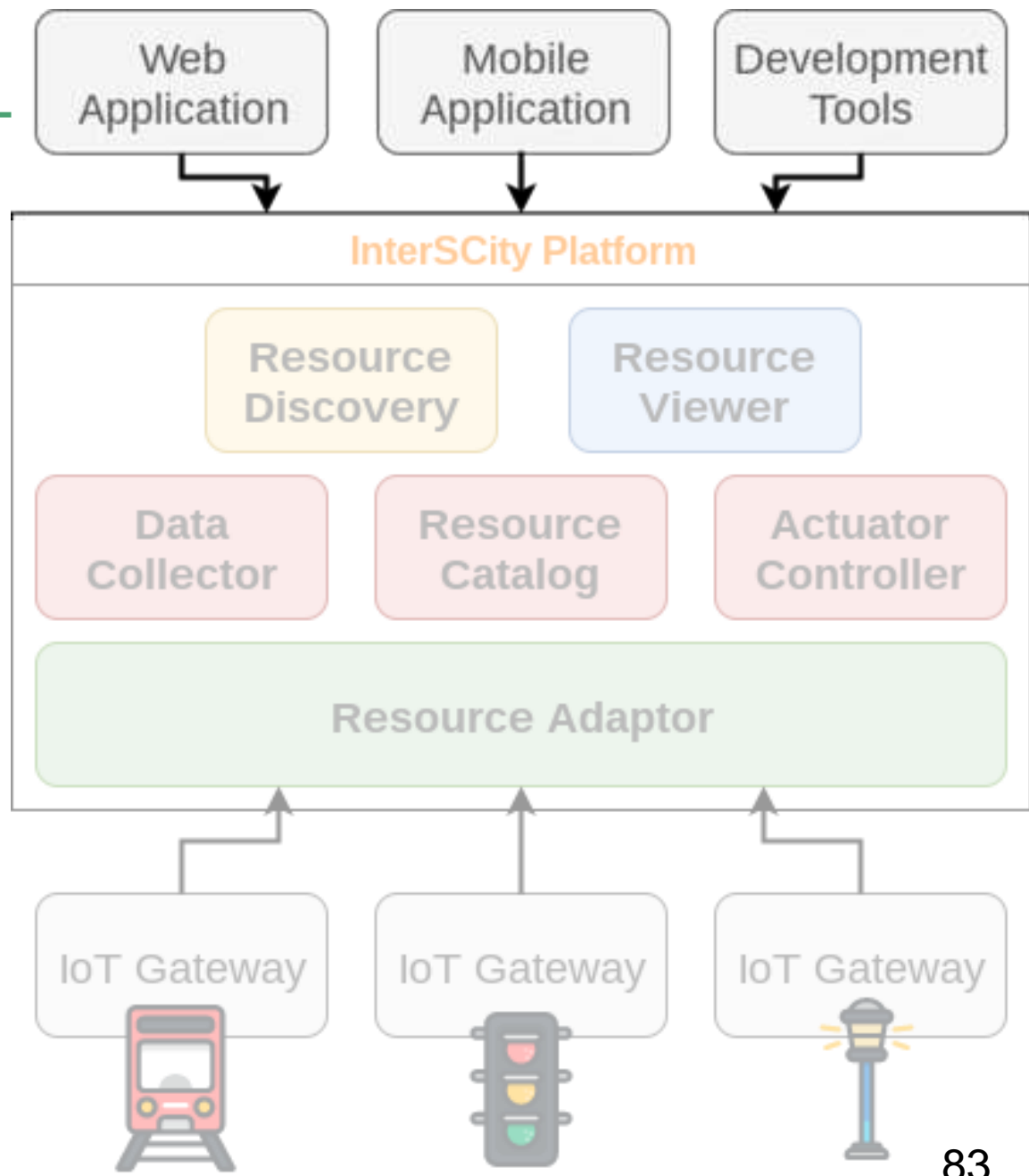
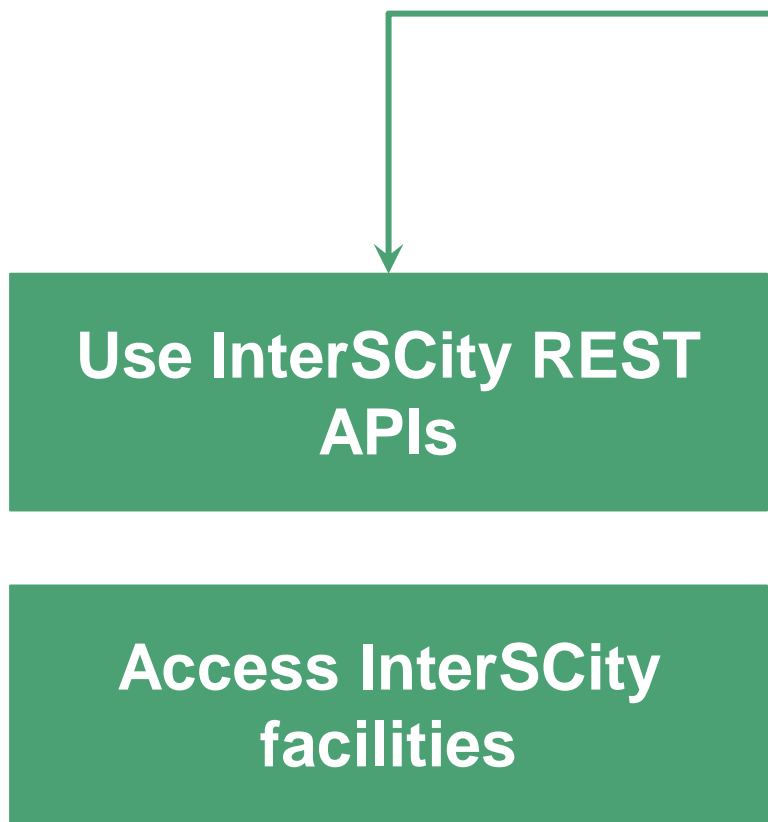
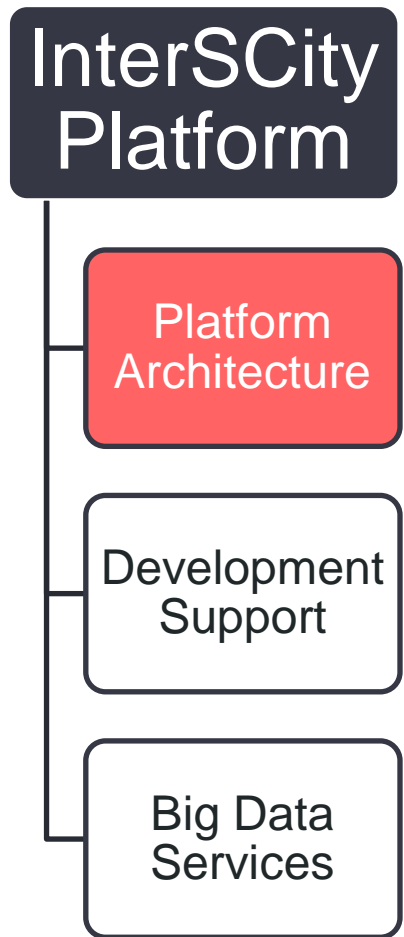
Resource Adaptor

IoT Gateway

IoT Gateway

IoT Gateway





InterSCity Platform

Platform Architecture

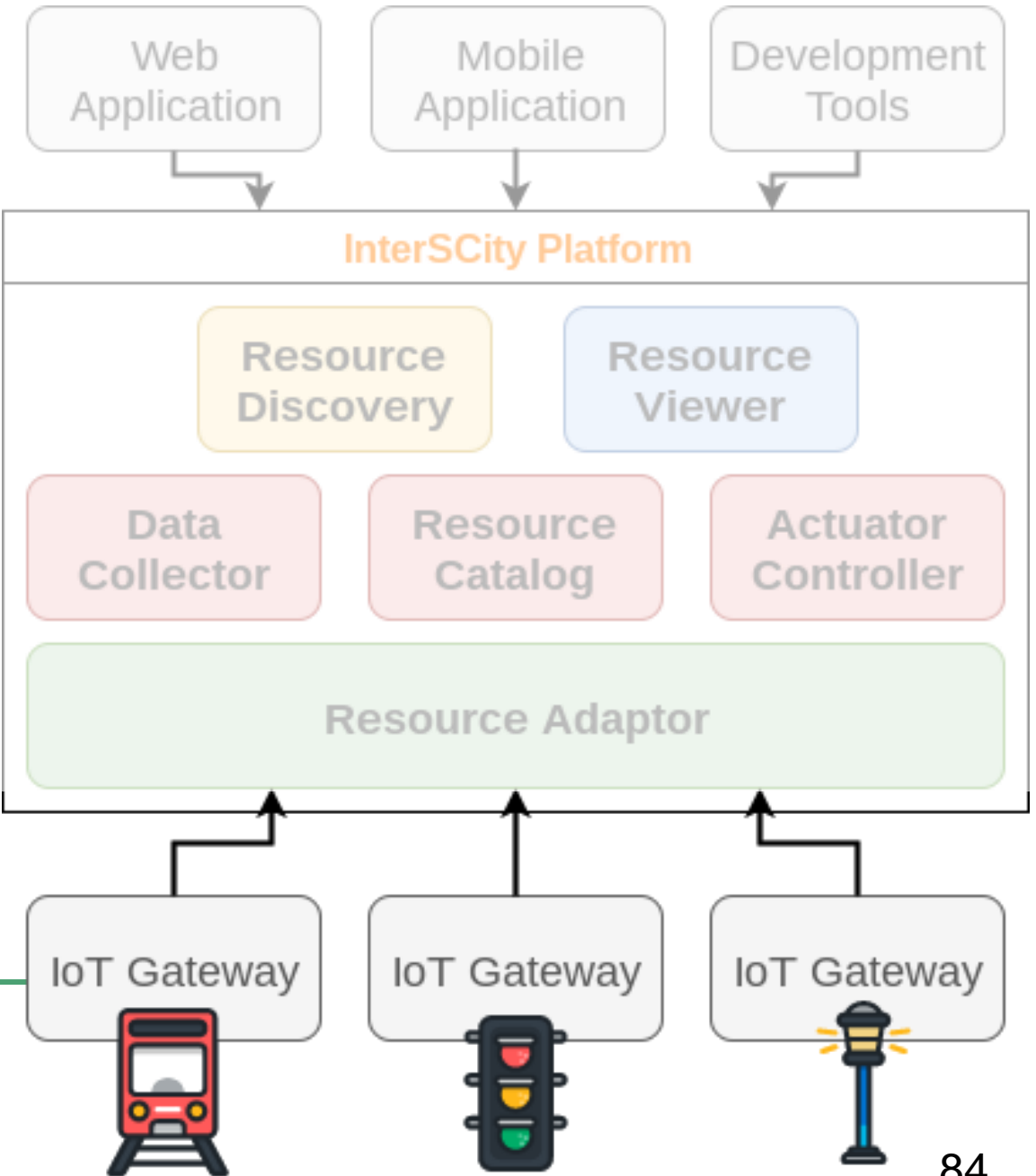
Development Support

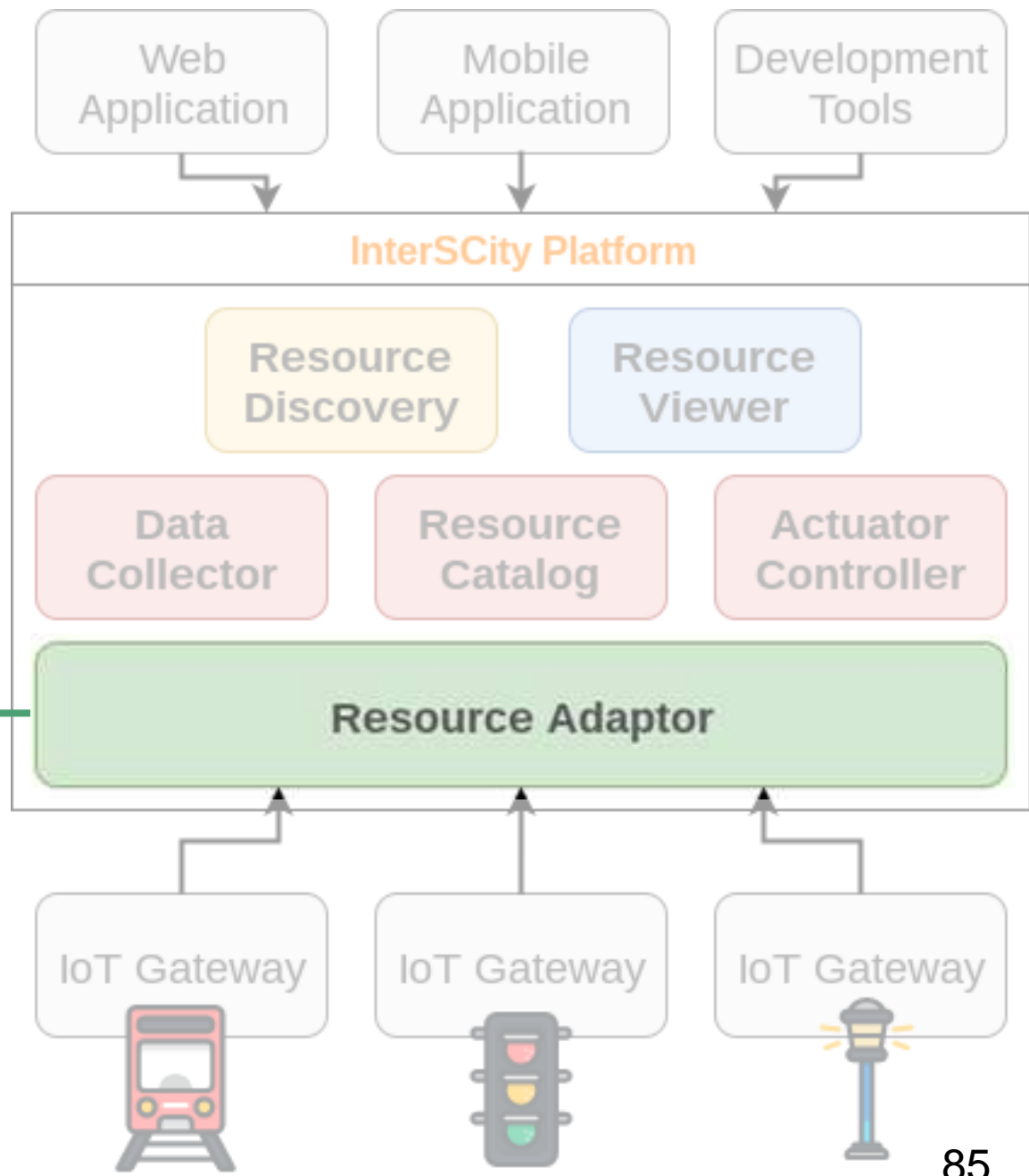
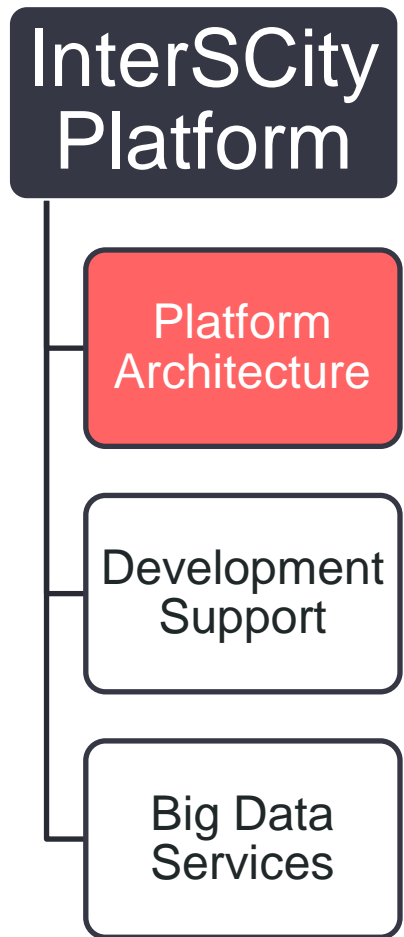
Big Data Services

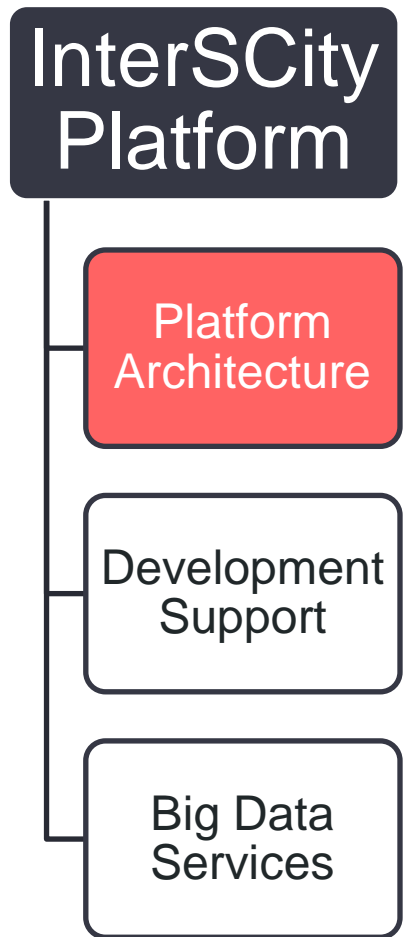
Register new resources

Post observed data from resources

Subscribe to receive actuator commands

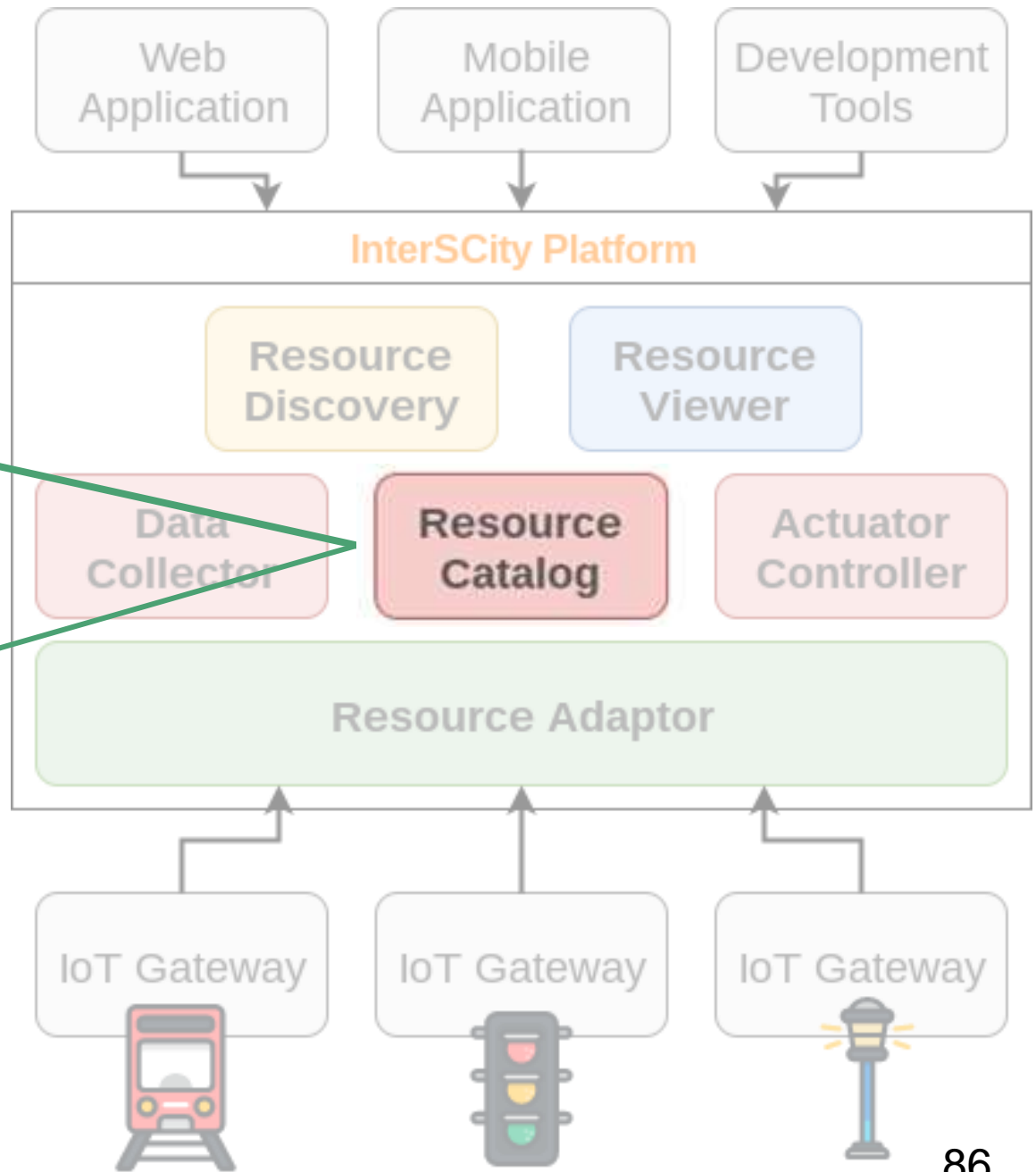






Store city resources static data

Notify services on resource creation and update



InterSCity Platform

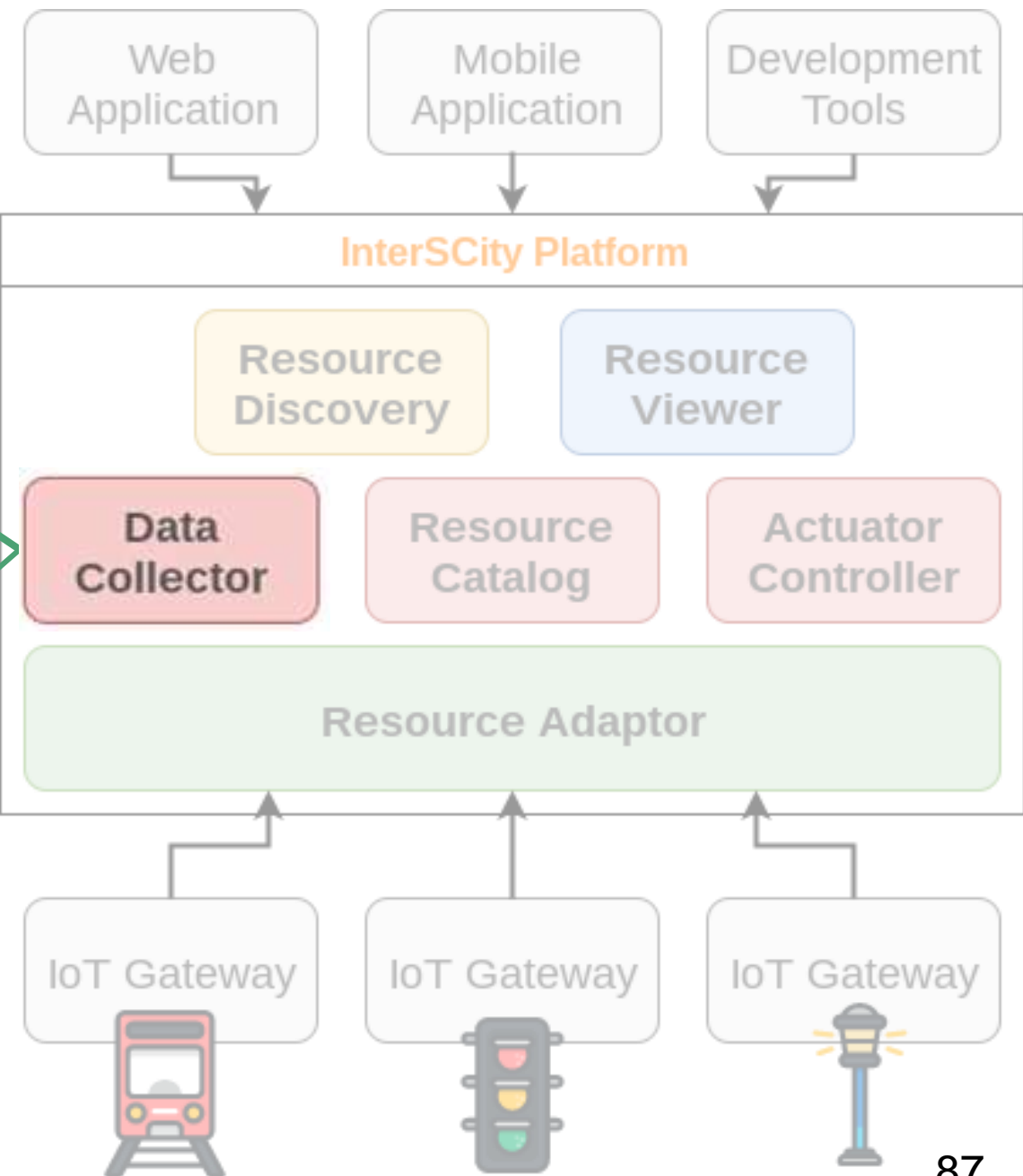
Platform Architecture

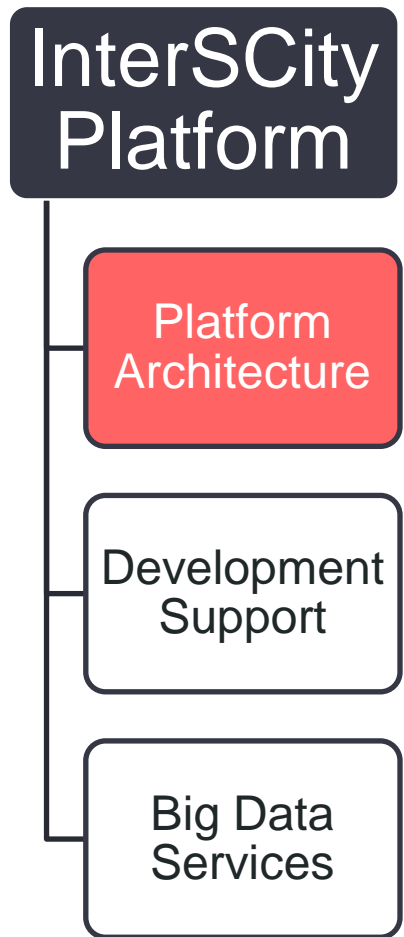
Development Support

Big Data Services

Store sensor data

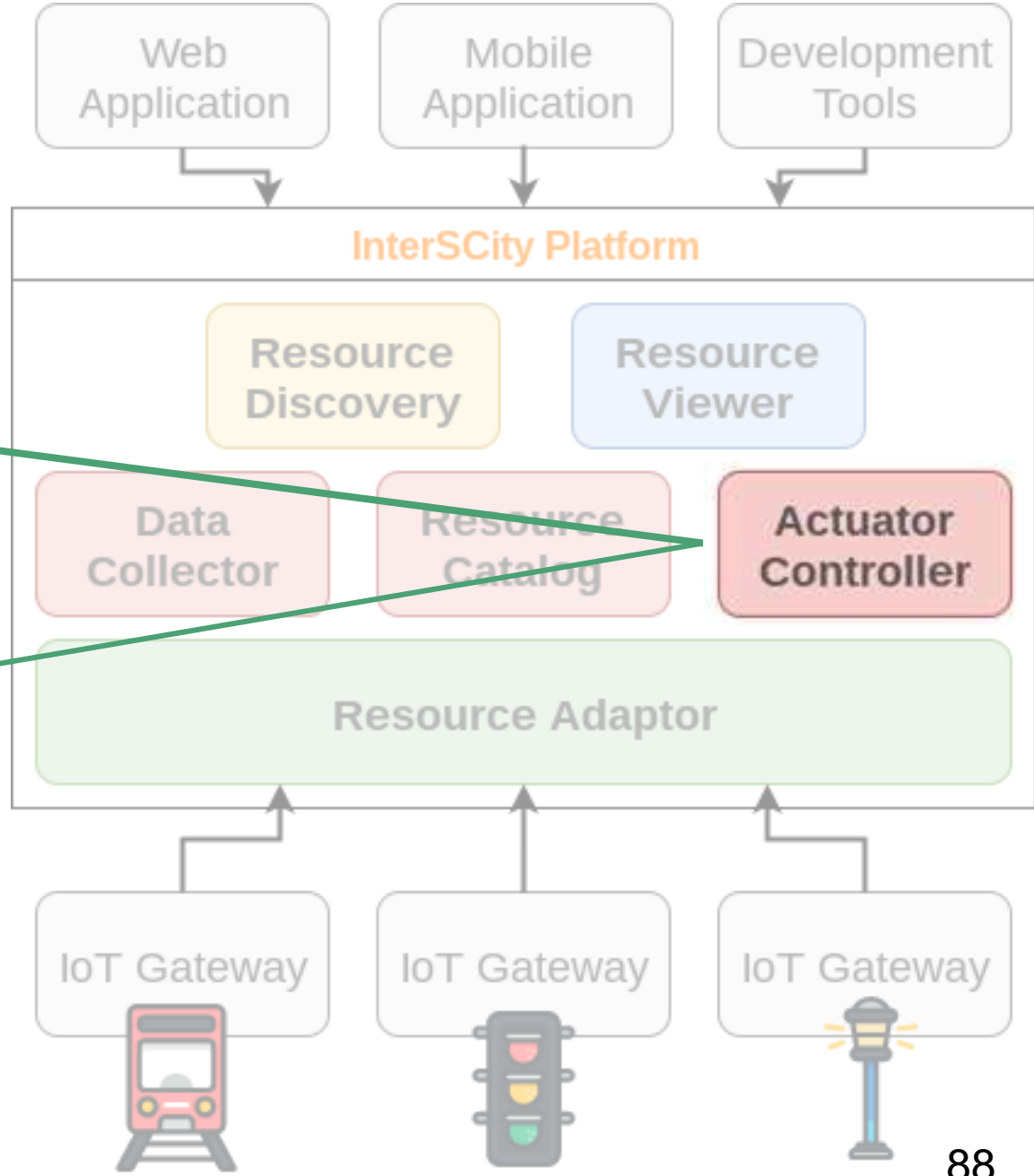
API to access historical and real-time data





Receive actuation commands from applications

Notify on actuation requests



InterSCity Platform

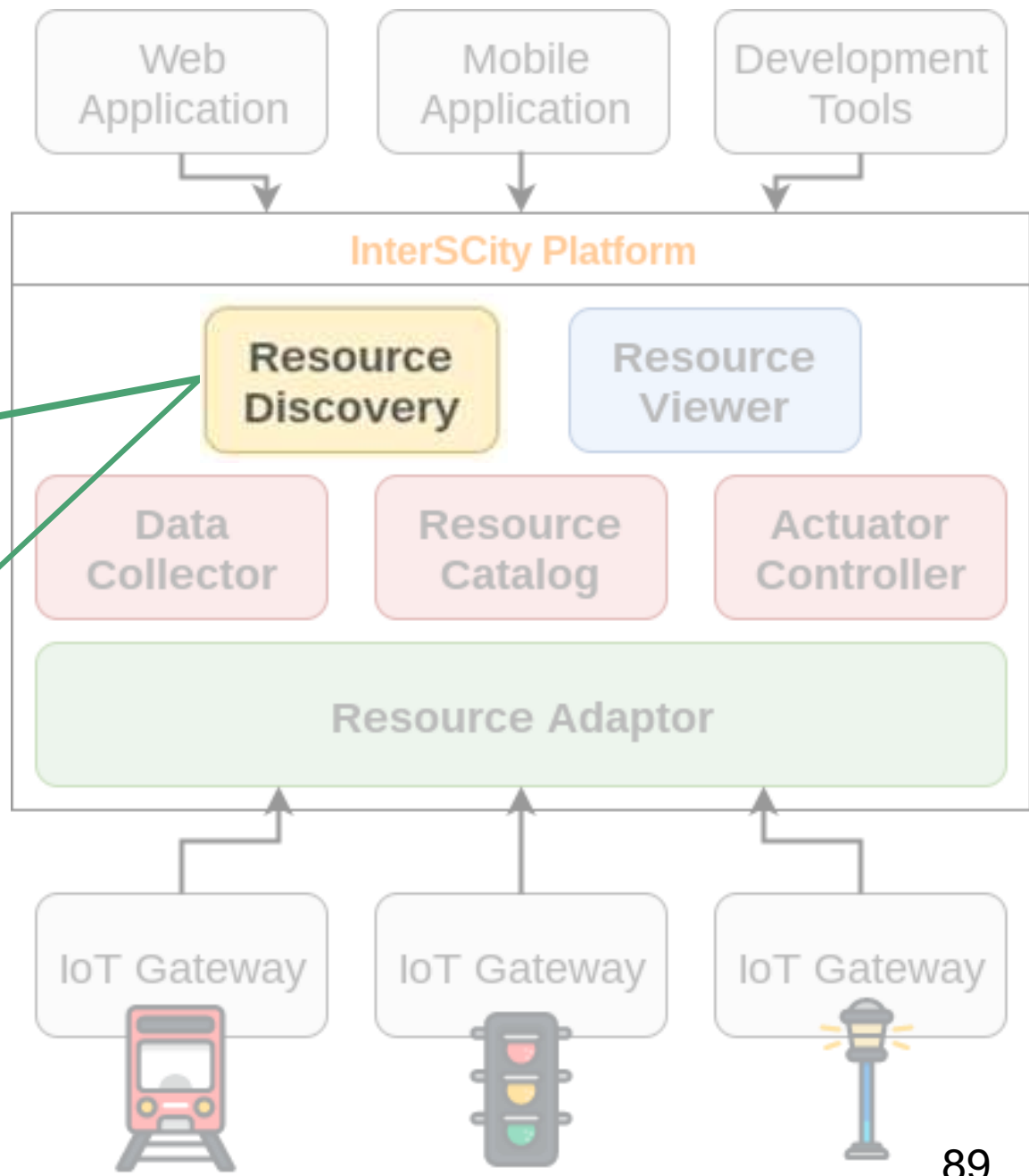
Platform Architecture

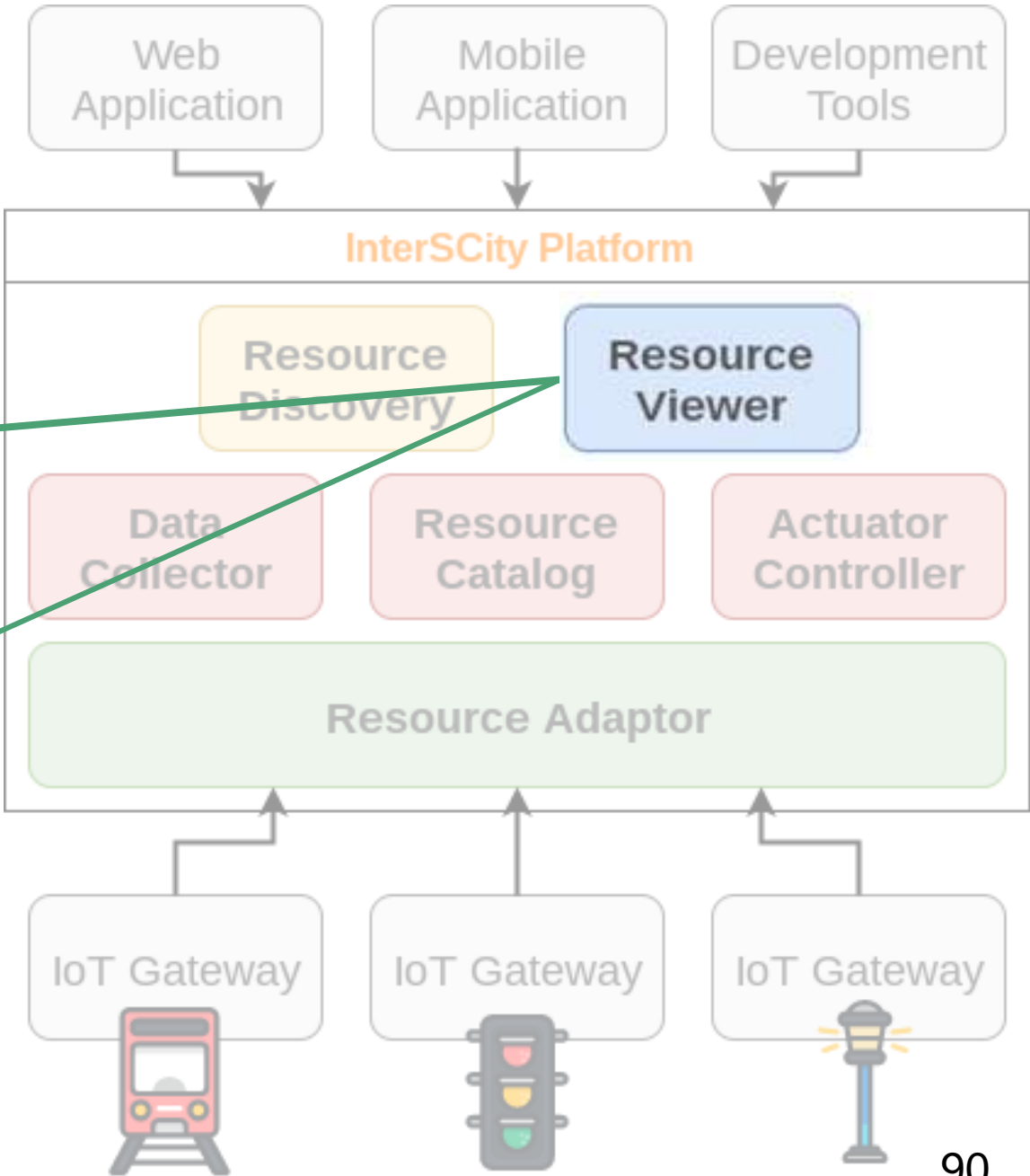
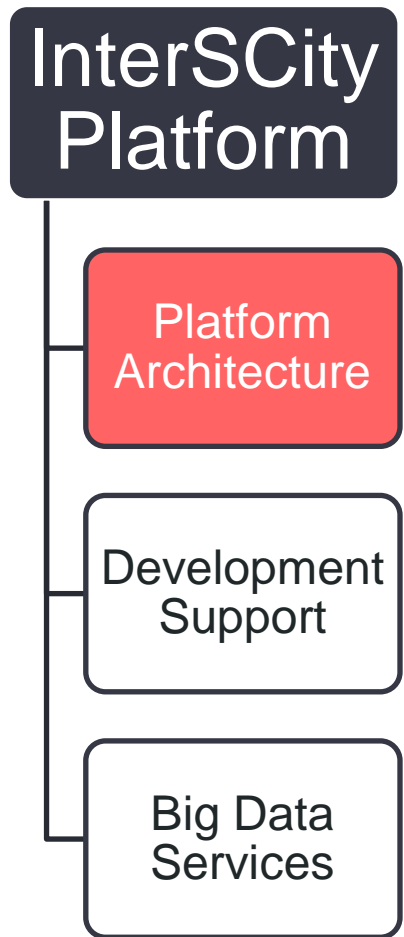
Development Support

Big Data Services

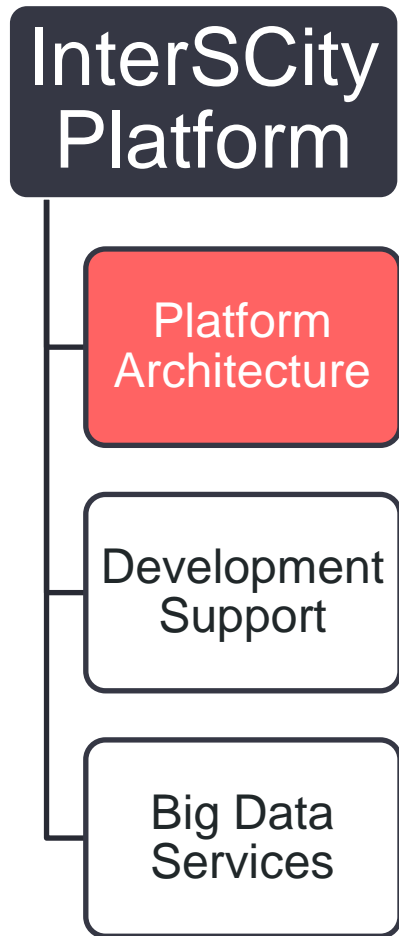
Service to discovery city resources

Search filters based on location, range sensor values, static data





Design Principles



- Modularity via Microservices
- Distributed Models and Data
- Decentralized Evolution
 - Heterogeneity of technologies
- Reuse of Open Source Projects
- Asynchronous vs Synchronous
 - AMQP and HTTP
- Stateless Services

How to achieve scalability?

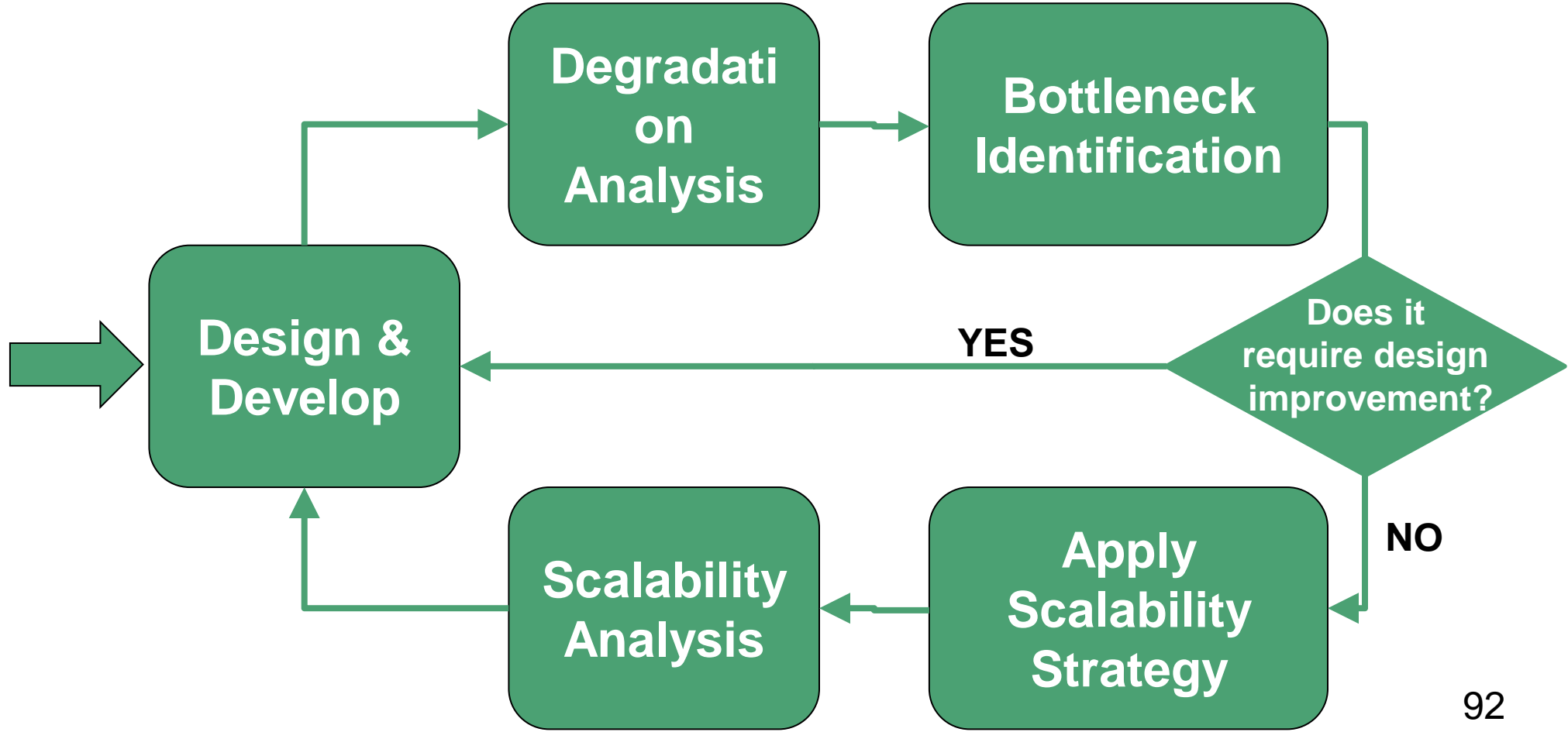
Scalability-seeking Experimental Method

InterSCity Platform

Platform Architecture

Development Support

Big Data Services



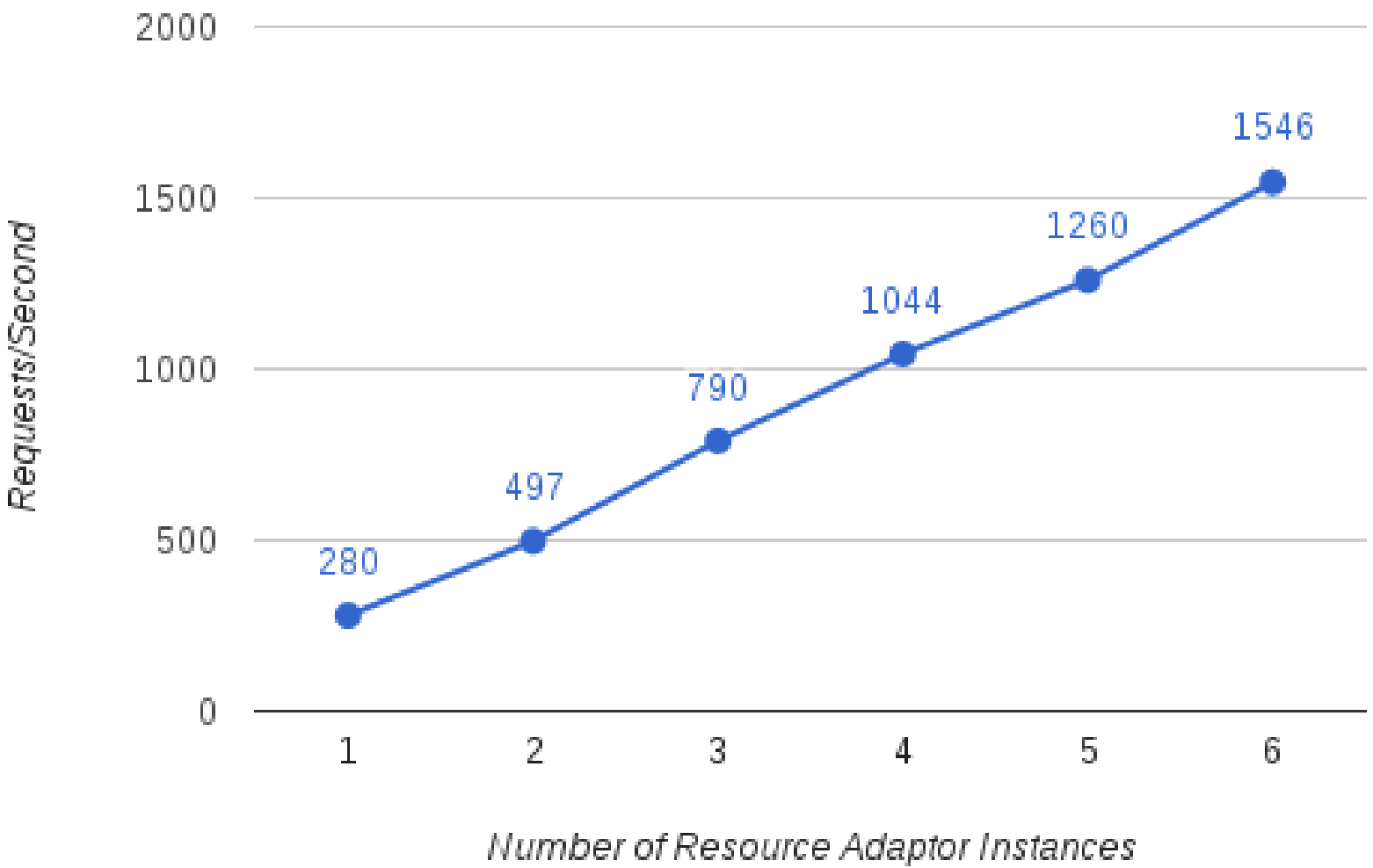
Scalability Analysis - Scale-up

**InterSCity
Platform**

Platform
Architecture

Development
Support

Big Data
Services



Scalability Analysis - Speedup

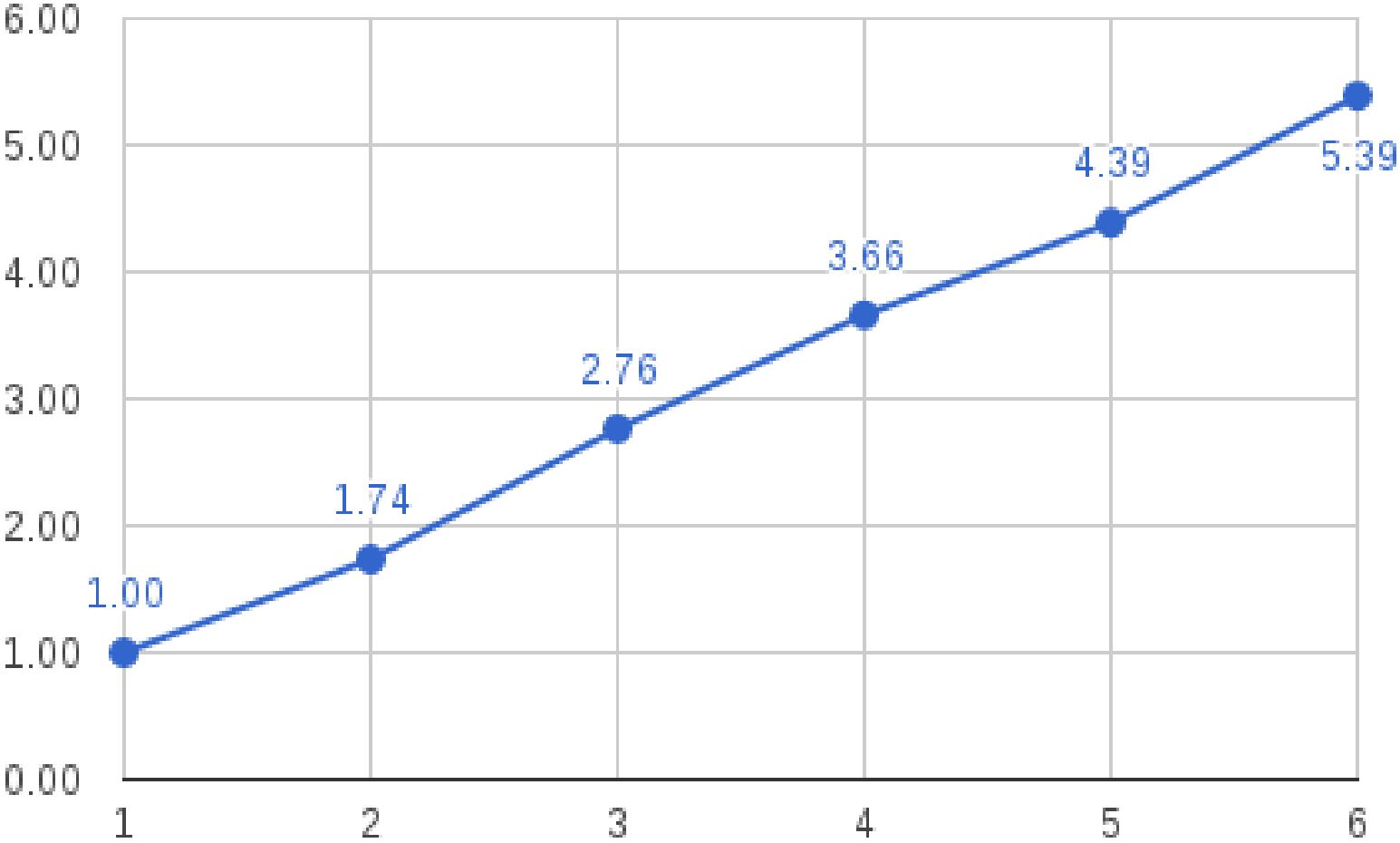
**InterSCity
Platform**

Platform
Architecture

Development
Support

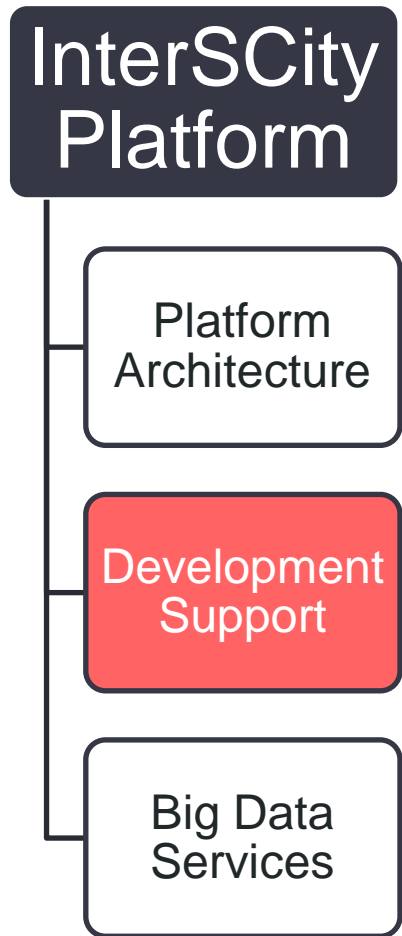
Big Data
Services

Speedup



Number of Resource Adaptor Instances

Development Support



Development Support

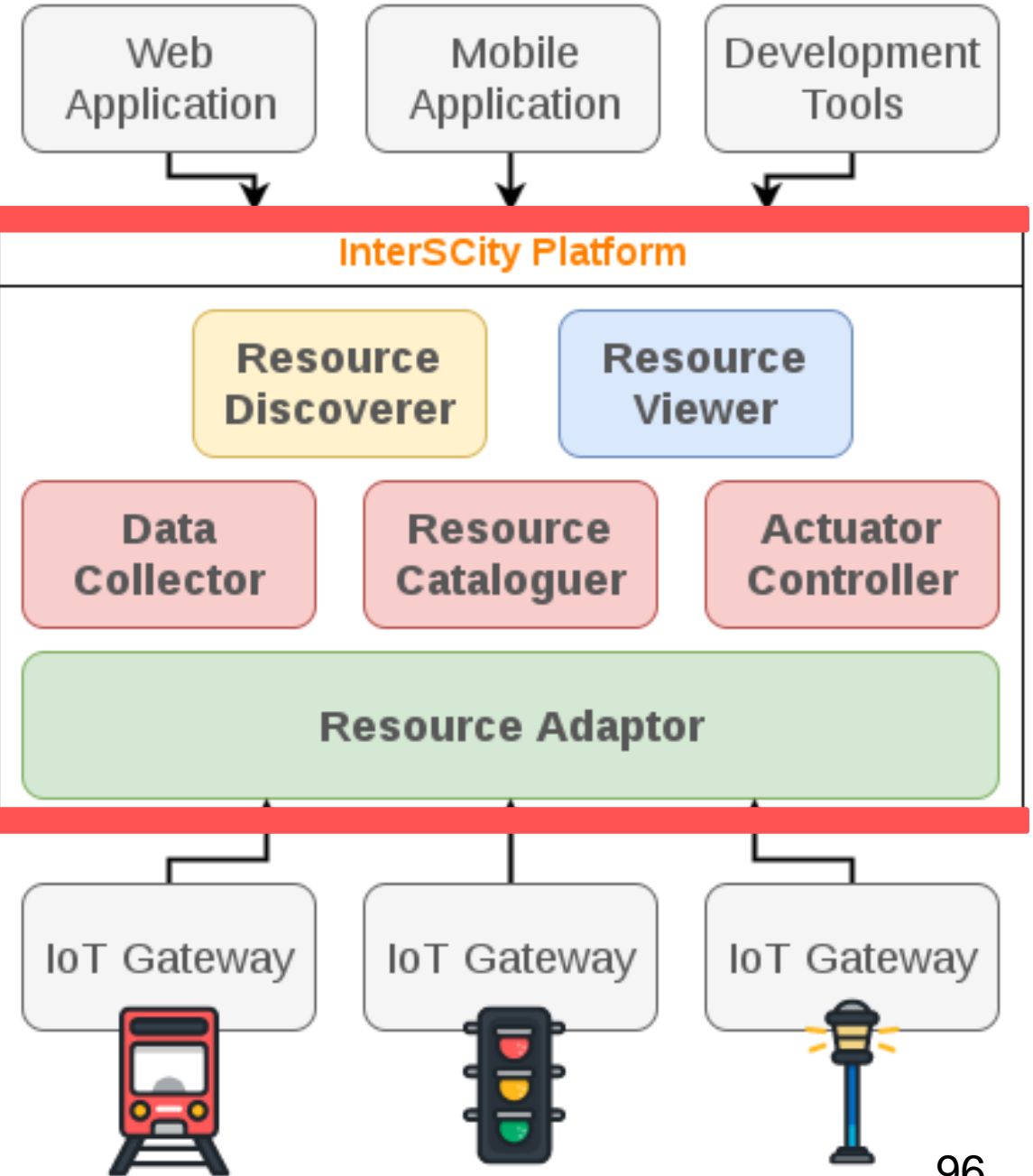
InterSCity Platform

Platform Architecture

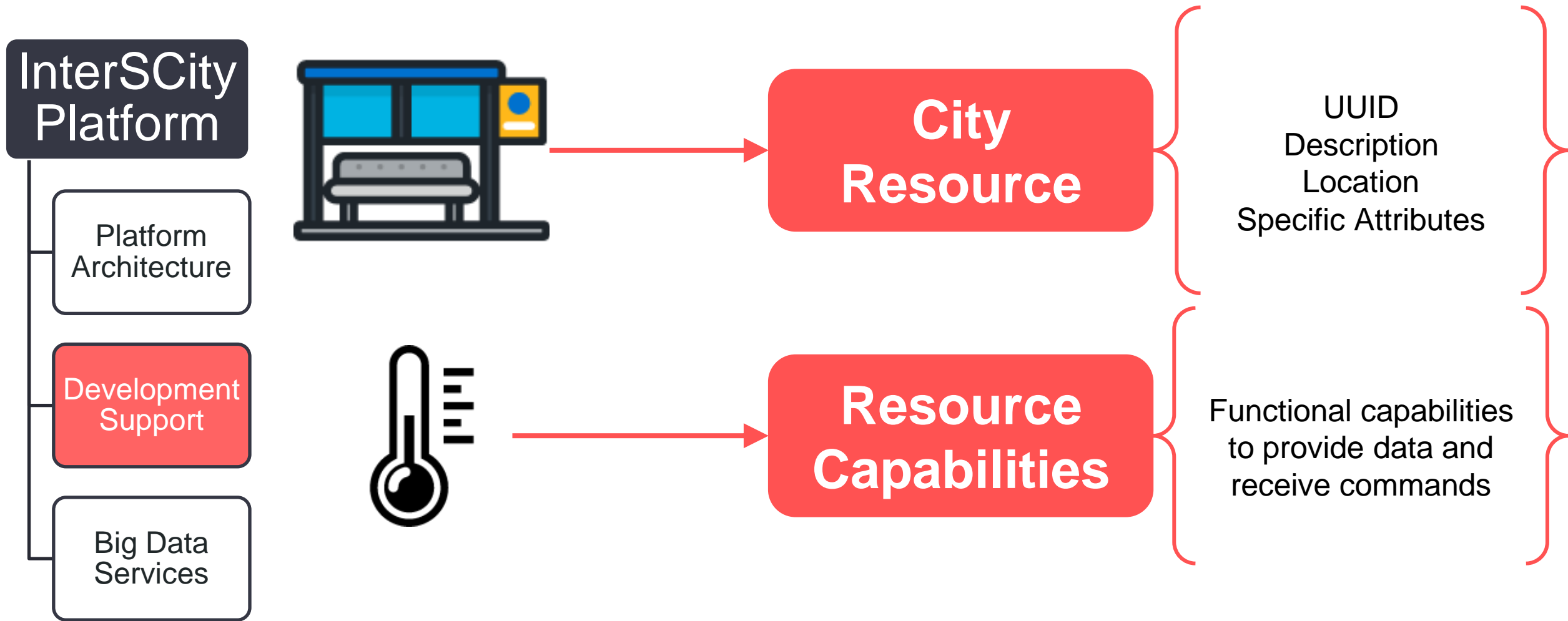
Development Support

Big Data Services

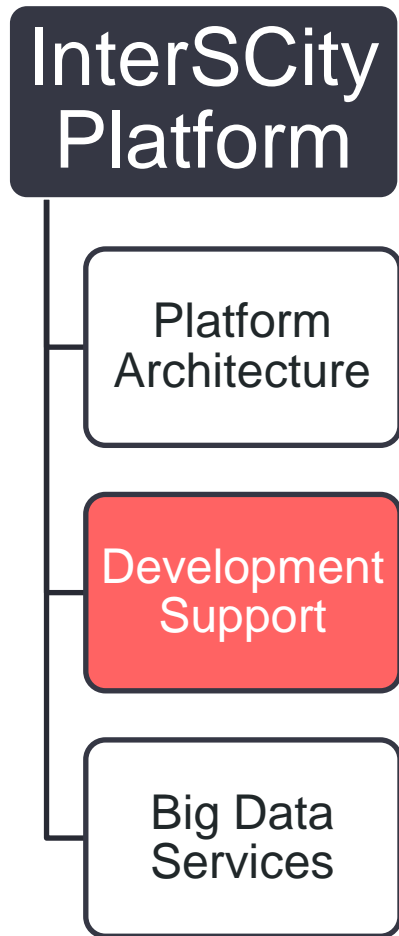
All external interfaces of the platform are **REST** APIs based on the **HTTP** protocol and **JSON** messages



Main Abstractions

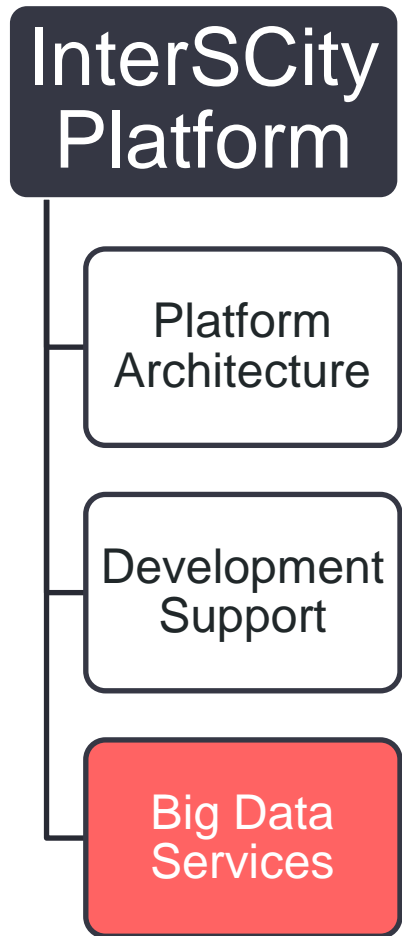


What else could be done?

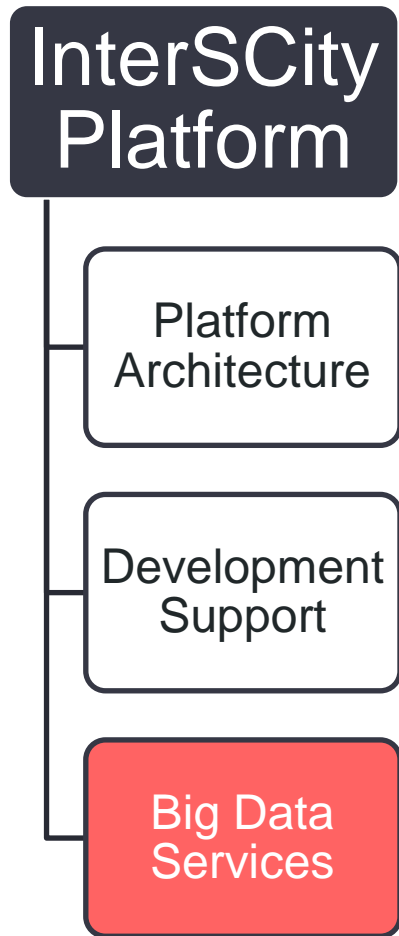


- Interactive API documentation
- Native libraries to abstract API calls
- DSL/Framework to facilitate the integration of low-level IoT Platforms

Big Data Services

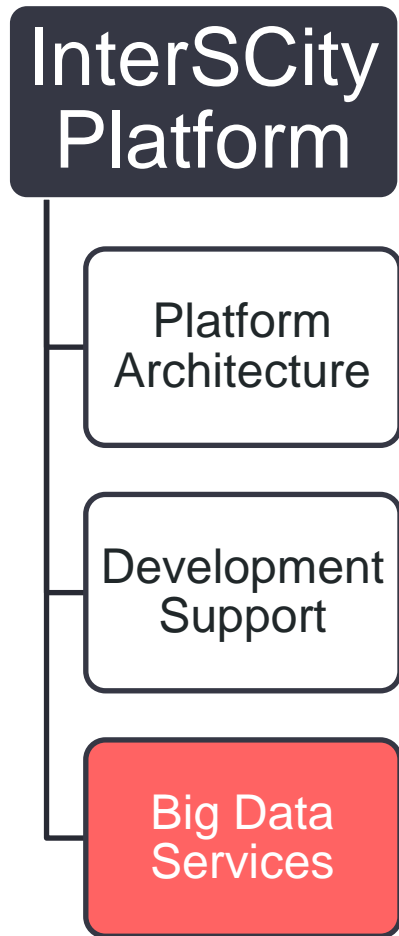


Big Data Research Topics



- Big Data generic services to improve InterSCity's development support
- Stream processing
- Big Data Visualization and Analytics
- Complex Event Processing - CEP
- Historical data processing
- Machine Learning
- Applications

Big Data Research Topics



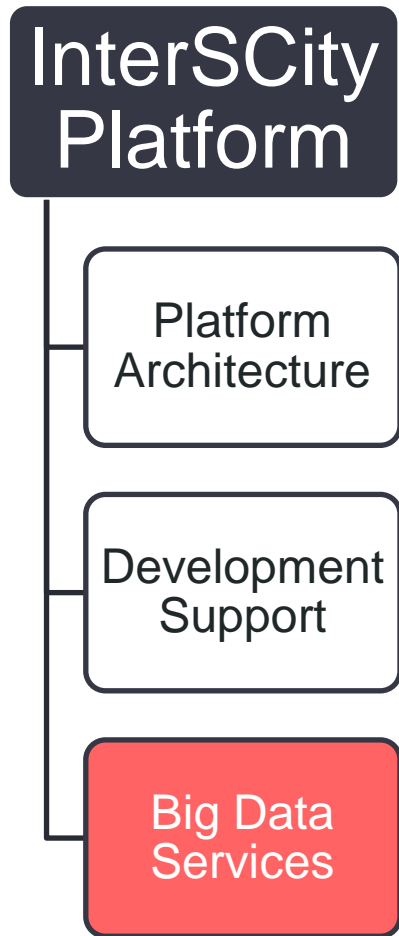
- Big Data generic services to improve InterSCity's development support
- Stream processing
- Big Data Visualization and Analytics
- Complex Event Processing - CEP
- Historical data processing
- Machine Learning
- Applications

Ongoing Effort

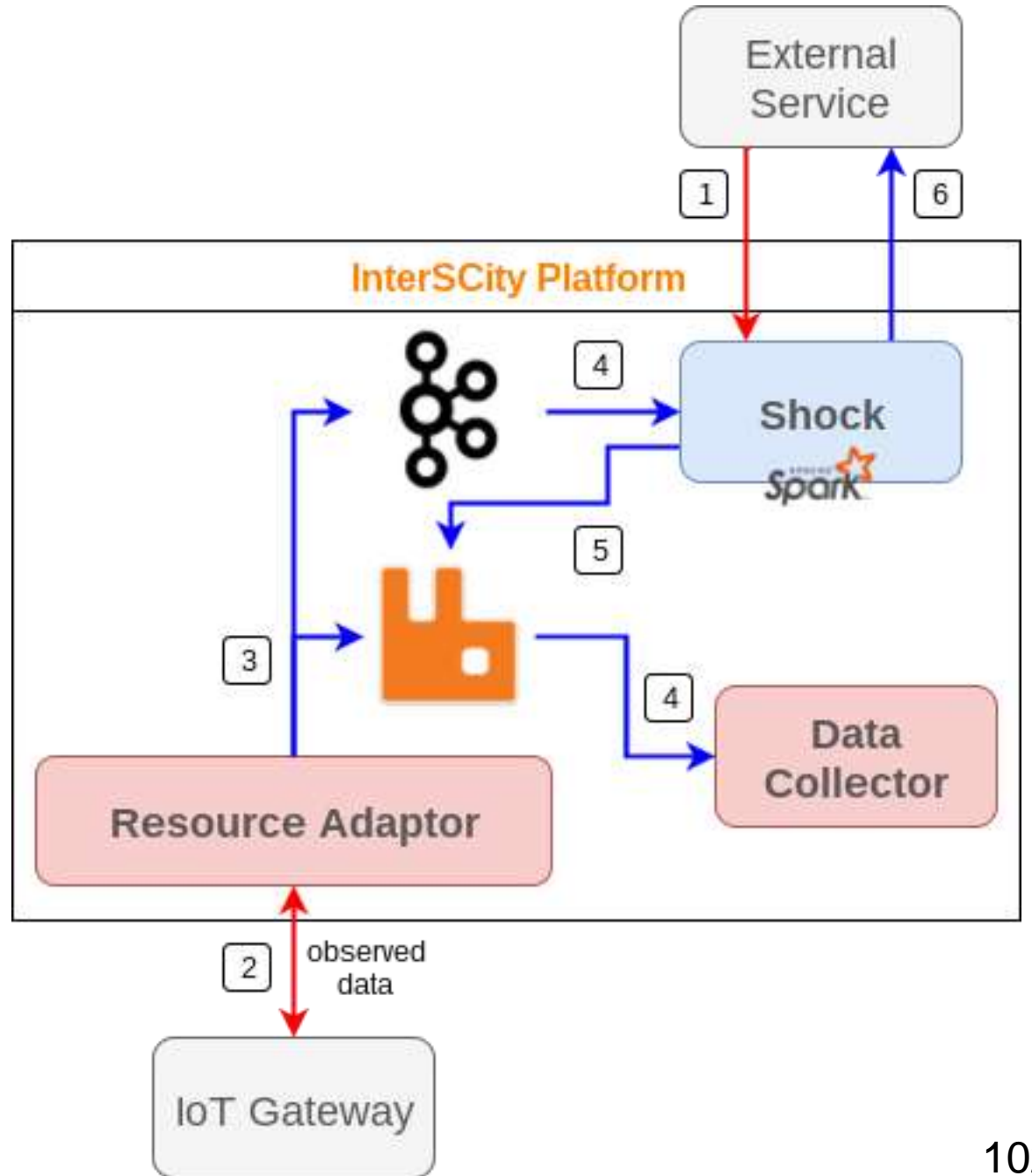
Mapped or Starting

Not initialized

Big Data Service



- Creation of Composite Resources
- Aggregation of Resource Observations
- Resource Observations Filtering
- Automatic actuation



Perguntas?



Instalando o InterSCity

Instalando o InterSCity

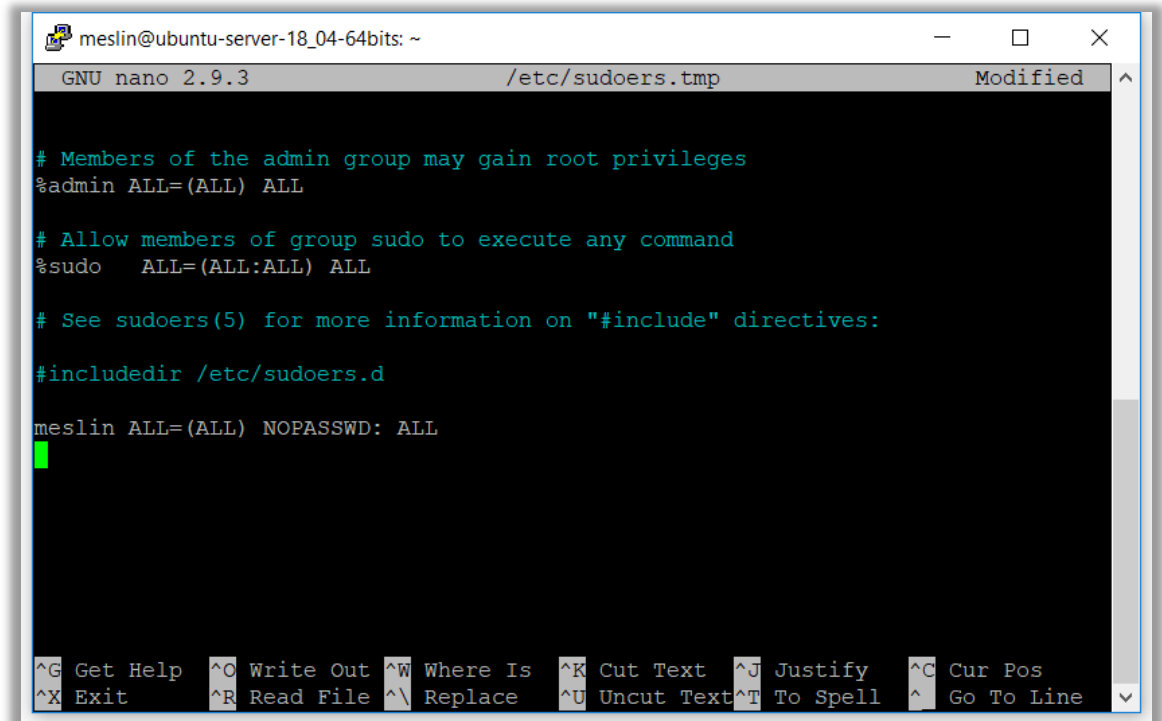
- Antes de começar:
 - Para facilitar a vida, habilitar o uso de sudo sem precisar informar a senha
 - Edite o arquivo de configuração do sudo com o comando:

\$ **sudo visudo**

- Acrescente a seguinte linha no final do arquivo:

usuário ALL=(ALL) NOPASSWD: ALL

- Substituindo *usuário* pelo seu username



```
meslin@ubuntu-server-18_04-64bits: ~
GNU nano 2.9.3 /etc/sudoers.tmp Modified
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

meslin ALL=(ALL) NOPASSWD: ALL
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Instalando o InterSCity

- Antes de começar:

- Atualize o sistema

```
$ sudo apt update
```

```
$ sudo apt -y upgrade
```

```
$ sudo apt -y autoremove
```

- Para o seu conforto

- Instale um servidor SSH

- Para instalar o OpenSSH Server, use o seguinte comando:

```
$ sudo apt -y install openssh-server
```

Instalando o InterSCity

Se for necessário IP fixo

UBUNTU 18.04

- Editar o arquivo
/etc/netplan/50-cloud-init.yaml

```
network:
  ethernets:
    ens3:
      addresses: [172.16.3.202/24]
      gateway4: 172.16.3.1
      dhcp4: no
      nameservers:
        addresses: [139.82.16.3, 10.8.0.1]
  version: 2
```

- E depois usar o comando:

```
$ sudo netplan apply
```

UBUNTU 16.04

- Editar o arquivo
/etc/network/interfaces

```
auto ens160
iface ens160 inet static
    address 172.16.1.202
    netmask 255.255.255.0
    network 172.16.1.0
    broadcast 172.16.1.255
    gateway 172.16.1.1
    dns-nameservers 139.82.16.3
    dns-search inf.puc-rio.br
```

Instalando o InterSCity

- Na falta de DNS, você pode usar o arquivo `/etc/hosts` para resolver nomes:

```
# Endereços do MUSANet
172.16.1.202      ContextNet1
172.16.2.202      ContextNet2
172.16.3.202      ContextNet3
172.16.4.202      ContextNet4

172.16.0.211     cliente1
172.16.0.212     cliente2
172.16.0.213     cliente3
172.16.0.214     cliente4
```


Instalando o InterSCity

- Instalação descrita em:

```
https://gitlab.com/intercity/intercity-  
platform/intercity-  
platform/tree/master/src
```

Instalando o InterSCity

Git

Ruby 2.3

Docker
(≥ 1.13)

Docker
Compose
(≥ 1.10)

InterSCity

Instalando o InterSCity

Preliminares

git

- Normalmente pré-instalado na maior parte das distribuições Linux
- `$ sudo apt install git`

gpg2

- Você vai precisar para instalar o Ruby
- `$ sudo apt install gnupg2`

Instalando o InterSCity

Ruby 2.3

- Descrito em <https://rvm.io/>
- Instalar chave GPG:
- ```
$ gpg2 --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
7D2BAF1CF37B13E2069D6956105BD0E739499BDB
```

- RVM:
- ```
$ curl -sSL https://get.rvm.io | bash -s stable --rails
```

Instalando o InterSCity

Docker

- Descrito em

<https://docs.docker.com/install/linux/docker-ce/ubuntu/>

```
$ sudo apt -y install apt-transport-https ca-certificates curl gnupg-agent  
software-properties-common  
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
$ sudo apt-key fingerprint 0EBFCD88  
$ sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"  
$ sudo apt update  
$ sudo apt -y install docker-ce docker-ce-cli containerd.io
```

Instalando o InterSCity

Docker

- Para testar a instalação do Docker:

```
$ sudo docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
```

```
latest: Pulling from library/hello-world
```

```
ca4f61b1923c: Pull complete
```

```
Digest:
```

```
sha256:083de497cff944f969d8499ab94f07134c50bcf5e6b9559b27182d3fa80ce3f7
```

```
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the

Instalando o InterSCity

Docker Compose

- Descrito em <https://docs.docker.com/compose/install/>

```
$ sudo curl -L  
"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-  
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
$ sudo chmod +x /usr/local/bin/docker-compose
```


Instalando o InterSCity

Docker

- Para testar a instalação do Docker Compose:

```
$ docker-compose --version  
docker-compose version 1.16.1, build 1719ceb
```

Instalando o InterSCity (finalmente!!!)

- Descrito em <https://gitlab.com/smart-city-software-platform/dev-env>

```
$ sudo mkdir /opt/InterSCity
$ cd /opt/InterSCity
$ sudo git clone https://gitlab.com/smart-city-software-platform/dev-env
$ cd dev-env
$ sudo git submodule init
$ sudo git submodule update
$ ./project setup
$ sudo docker network create platform
```

- Iniciar o projeto, dentro do diretório /opt/InterSCity/dev-env:

```
$ ./project start
```

Instalando o InterSCity

Testando...

- Para testar o projeto, execute o comando a seguir. A resposta deverá ser uma lista de capabilities previamente configurada no InterSCity:

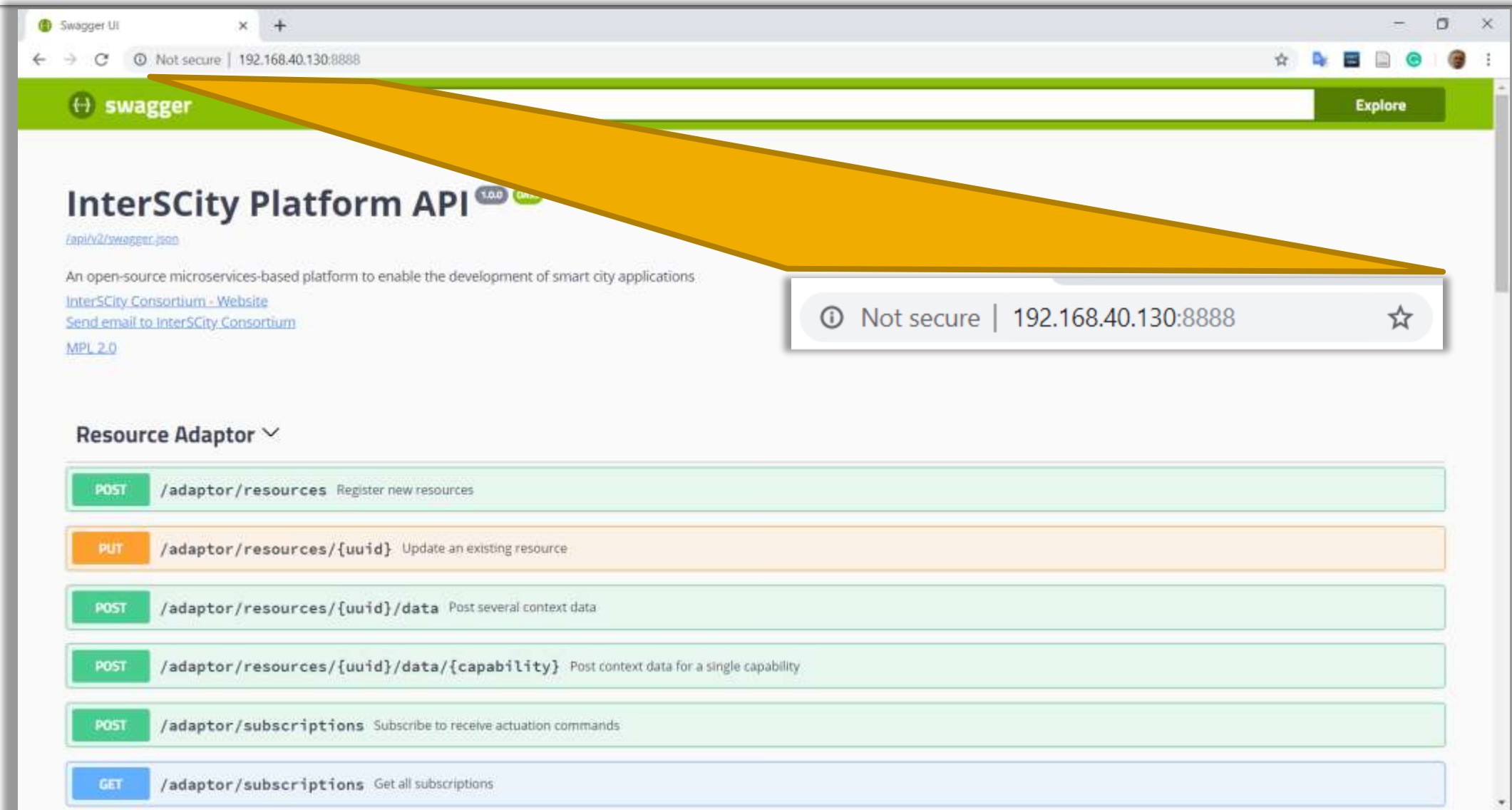
```
$ curl -X GET "http://localhost:8000/catalog/capabilities"  
{  
  "capabilities": [  
    {"id": 1, "name": "air-  
quality", "function": 0, "description": null}, {"id": 2, "name": "air-  
humidity", "function": 0, "description": null}, {"id": 3, "name": "parking-  
occupancy", "function": 0, "description": null}, {"id": 4, "name": "video", "function":  
0, "description": null}, {"id": 5, "name": "queue-  
length", "function": 0, "description": null}, {"id": 6, "name": "semaphore", "function":  
1, "description": null}, {"id": 7, "name": "illuminate", "function": 1, "description":  
null}, {"id": 8, "name": "parking-  
type", "function": 2, "description": null}, {"id": 9, "name": "current_users", "functio  
n": 0, "description": null}, {"id": 10, "name": "info_facility_type", "function": 2, "de  
scription": null}, {"id": 11, "name": "info_total_capacity", "function": 2, "descripti
```

Perguntas?



Conhecendo o InterSCity

Conhecendo o InterSCity



Swagger UI

Not secure | 192.168.40.130:8888

swagger Explore

InterSCity Platform API 1.0.0

[api/v2/swagger.json](#)

An open-source microservices-based platform to enable the development of smart city applications.

[InterSCity Consortium - Website](#)
[Send email to InterSCity Consortium](#)
[MPL 2.0](#)

Not secure | 192.168.40.130:8888

Resource Adaptor

- POST** `/adaptor/resources` Register new resources
- PUT** `/adaptor/resources/{uuid}` Update an existing resource
- POST** `/adaptor/resources/{uuid}/data` Post several context data
- POST** `/adaptor/resources/{uuid}/data/{capability}` Post context data for a single capability
- POST** `/adaptor/subscriptions` Subscribe to receive actuation commands
- GET** `/adaptor/subscriptions` Get all subscriptions

InterSCity Resource Catalog

Resource Catalog ∨	
GET	<code>/catalog/resources</code> Get all resources registered on the platform
POST	<code>/catalog/resources</code> Register new resources
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource
GET	<code>/catalog/capabilities</code> Get all capabilities
POST	<code>/catalog/capabilities</code> Create a new capability
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability

InterSCity Resource Catalog

Resource Catalog	
GET	<code>/catalog/resources</code> Get all resources registered on the platform.
POST	<code>/catalog/resources</code> Register new resources.
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities.
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities.
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters.
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource.
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource.
GET	<code>/catalog/capabilities</code> Get all capabilities.
POST	<code>/catalog/capabilities</code> Create a new capability.
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability.
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability.
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability.

The screenshot shows the Swagger UI for the `GET /catalog/capabilities` endpoint. The interface includes a description, a parameter table, and a response table.

Parameters

Name	Description
<code>capability_type</code> string (query)	filter by capability's types in [sensor, actuator]

Responses

Code	Description	Links
200	The list of capabilities	No links

Example Value | Model

```
{
  "capabilities": [
    {
      "id": 1,
      "name": "Semaphore",
      "function": 1,
      "description": "Manipulate the semaphore status"
    },
    {
      "id": 2,
      "name": "Temperature",
      "function": 0,
      "description": "Measure the temperature of the environment"
    }
  ]
}
```


InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform.

POST /catalog/resources Register new resources.

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.

GET /catalog/resources/search Search for registered resources using different filters.

PUT /catalog/resources/{uuid} Update an existing resource.

GET /catalog/resources/{uuid} Get data about a specific resource.

GET /catalog/capabilities Get all capabilities.

POST /catalog/capabilities Create a new capability.

GET /catalog/capabilities/{name} Get data about a specific capability.

PUT /catalog/capabilities/{name} Update an existing capability.

DELETE /catalog/capabilities/{name} Delete a specific capability.

The screenshot shows the Postman interface for a GET request to `http://172.16.10.1:8000/catalog/capabilities`. The response body is displayed in the Pretty view, showing a JSON array of capabilities:

```
{
  "capabilities": [
    {
      "id": 1,
      "name": "air-quality",
      "function": 0,
      "description": null
    },
    {
      "id": 2,
      "name": "air-humidity",
      "function": 0,
      "description": null
    },
    {
      "id": 3,
      "name": "parking-occupancy",
      "function": 0,
      "description": null
    },
    {
      "id": 4,
      "name": "video",
      "function": 0,
      "description": null
    },
    {
      "id": 5,
      "name": "game-length",
      "function": 0,
      "description": null
    },
    {
      "id": 6,
      "name": "semaphore",
      "function": 1,
      "description": null
    },
    {
      "id": 7
    }
  ]
}
```

InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform.

POST /catalog/resources Register new resources.

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.

GET /catalog/resources/search Search for registered resources using different filters.

PUT /catalog/resources/{uuid} Update an existing resource.

GET /catalog/resources/{uuid} Get data about a specific resource.

GET /catalog/capabilities Get all capabilities.

POST /catalog/capabilities Create a new capability.

GET /catalog/capabilities/{name} Get data about a specific capability.

PUT /catalog/capabilities/{name} Update an existing capability.

DELETE /catalog/capabilities/{name} Delete a specific capability.

The screenshot shows a Postman interface with a GET request to `http://172.16.10.130:8000/catalog/capabilities`. The response is a JSON array of capabilities:

```
{
  "capabilities": [
    {
      "id": 1,
      "name": "air-quality",
      "function": 0,
      "description": null
    },
    {
      "id": 2,
      "name": "air-humidity",
      "function": 0,
      "description": null
    },
    {
      "id": 3,
      "name": "parking-occupancy",
      "function": 0,
      "description": null
    },
    {
      "id": 4,
      "name": "video",
      "function": 0,
      "description": null
    },
    {
      "id": 5,
      "name": "game-length",
      "function": 0,
      "description": null
    },
    {
      "id": 6,
      "name": "semaphore",
      "function": 1,
      "description": null
    },
    {
      "id": 7
    }
  ]
}
```

InterSCity Resource Catalog

Resource Catalog	
GET	<code>/catalog/resources</code> Get all resources registered on the platform.
POST	<code>/catalog/resources</code> Register new resources.
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities.
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities.
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters.
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource.
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource.
GET	<code>/catalog/capabilities</code> Get all capabilities.
POST	<code>/catalog/capabilities</code> Create a new capability.
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability.
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability.
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability.

The screenshot shows a Postman interface with a GET request to `http://172.16.10.130:8000/catalog/capabilities`. The response is a JSON array of capabilities:

```
{
  "capabilities": [
    {
      "id": 1,
      "name": "air-quality",
      "function": 0,
      "description": null
    },
    {
      "id": 2,
      "name": "air-humidity",
      "function": 0,
      "description": null
    },
    {
      "id": 3,
      "name": "parking-occupancy",
      "function": 0,
      "description": null
    },
    {
      "id": 4,
      "name": "video",
      "function": 0,
      "description": null
    },
    {
      "id": 5,
      "name": "queue-length",
      "function": 0,
      "description": null
    }
  ]
}
```

InterSCity Resource Catalog

Resource Catalog ∨	
GET	<code>/catalog/resources</code> Get all resources registered on the platform
POST	<code>/catalog/resources</code> Register new resources
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource
GET	<code>/catalog/capabilities</code> Get all capabilities
POST	<code>/catalog/capabilities</code> Create a new capability
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability

InterSCity Resource Catalog

Resource Catalog	
GET	/catalog/resources Get all resources registered on the platform.
POST	/catalog/resources Register new resources.
GET	/catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search Search for registered resources using different filters.
PUT	/catalog/resources/{uuid} Update an existing resource.
GET	/catalog/resources/{uuid} Get data about a specific resource.
GET	/catalog/capabilities Get all capabilities.
POST	/catalog/capabilities Create a new capability.
GET	/catalog/capabilities/{name} Get data about a specific capability.
PUT	/catalog/capabilities/{name} Update an existing capability.
DELETE	/catalog/capabilities/{name} Delete a specific capability.

The screenshot shows the Swagger UI for the POST endpoint `/catalog/capabilities`. The interface includes a title bar with the Swagger logo and browser tabs. The main content area is titled "POST /catalog/capabilities Create a new capability". Below the title, there is a description "Create a new capability". The "Parameters" section is empty, indicating no parameters are required. The "Request body" section is set to "application/json" and contains an example JSON object:

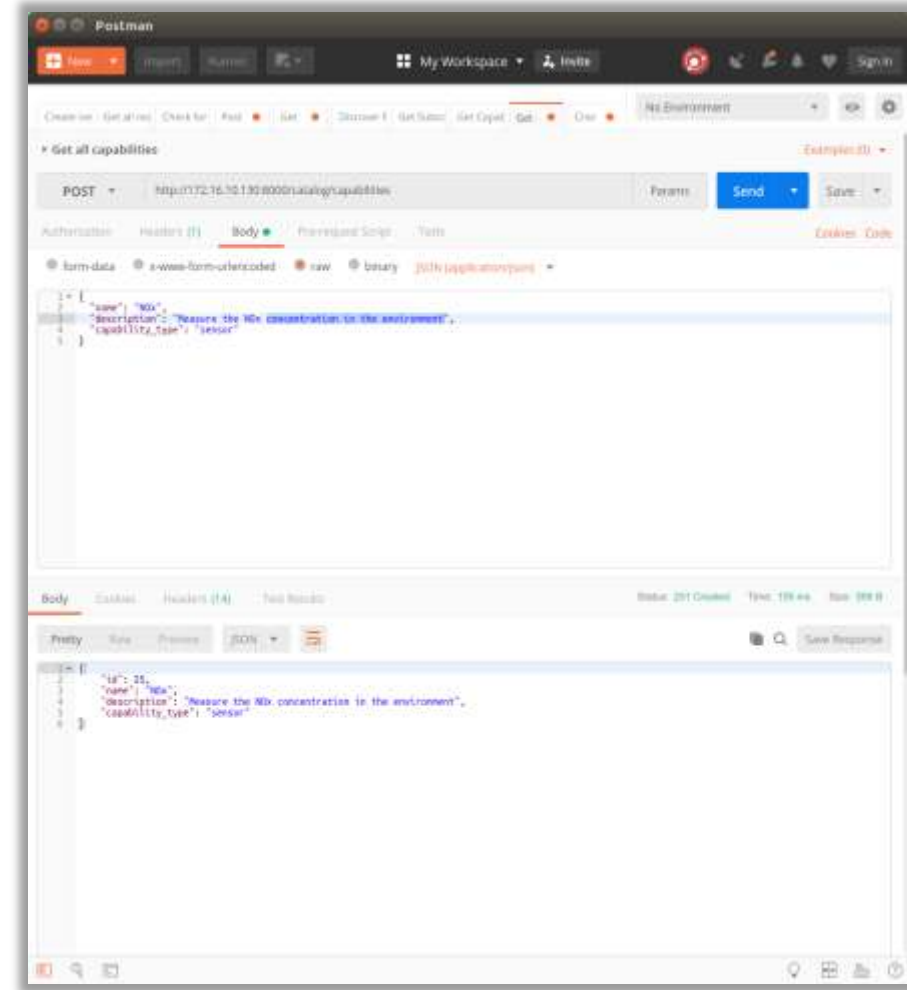
```
{  "name": "temperature",  "description": "Measure the temperature of the environment",  "capability_type": "sensor"}
```

. The "Responses" section is a table with columns for "Code", "Description", and "Links". It lists two responses: a 201 status code for "The created capability" and a 422 status code for "Unprocessable entity error". Both responses have a dropdown menu set to "application/json" and "No links" listed.

Code	Description	Links
201	The created capability	No links
422	Unprocessable entity error.	No links

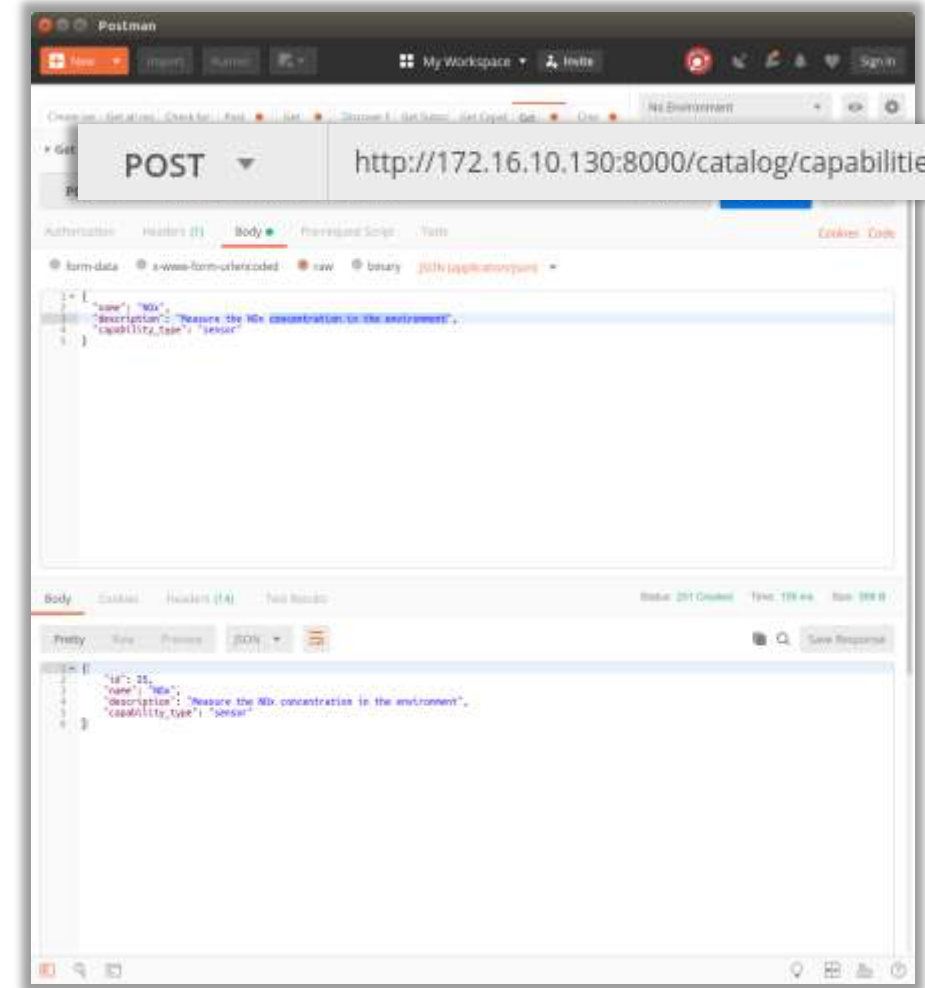
InterSCity Resource Catalog

Resource Catalog	
GET	/catalog/resources Get all resources registered on the platform.
POST	/catalog/resources Register new resources.
GET	/catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search Search for registered resources using different filters.
PUT	/catalog/resources/{uuid} Update an existing resource.
GET	/catalog/resources/{uuid} Get data about a specific resource.
GET	/catalog/capabilities Get all capabilities.
POST	/catalog/capabilities Create a new capability.
GET	/catalog/capabilities/{name} Get data about a specific capability.
PUT	/catalog/capabilities/{name} Update an existing capability.
DELETE	/catalog/capabilities/{name} Delete a specific capability.



InterSCity Resource Catalog

Resource Catalog	
GET	/catalog/resources Get all resources registered on the platform.
POST	/catalog/resources Register new resources.
GET	/catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search Search for registered resources using different filters.
PUT	/catalog/resources/{uuid} Update an existing resource.
GET	/catalog/resources/{uuid} Get data about a specific resource.
GET	/catalog/capabilities Get all capabilities.
POST	/catalog/capabilities Create a new capability.
GET	/catalog/capabilities/{name} Get data about a specific capability.
PUT	/catalog/capabilities/{name} Update an existing capability.
DELETE	/catalog/capabilities/{name} Delete a specific capability.



InterSCity Resource Catalog

Resource Catalog	
GET	/catalog/resources Get all resources registered on the platform.
POST	/catalog/resources Register new resources.
GET	/catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search Search for registered resources using different filters.
PUT	/catalog/resources/{uuid} Update an existing resource.
GET	/catalog/resources/{uuid} Get data about a specific resource.
GET	/catalog/capabilities Get all capabilities.
POST	/catalog/capabilities Create a new capability.
GET	/catalog/capabilities/{name} Get data about a specific capability.
PUT	/catalog/capabilities/{name} Update an existing capability.
DELETE	/catalog/capabilities/{name} Delete a specific capability.

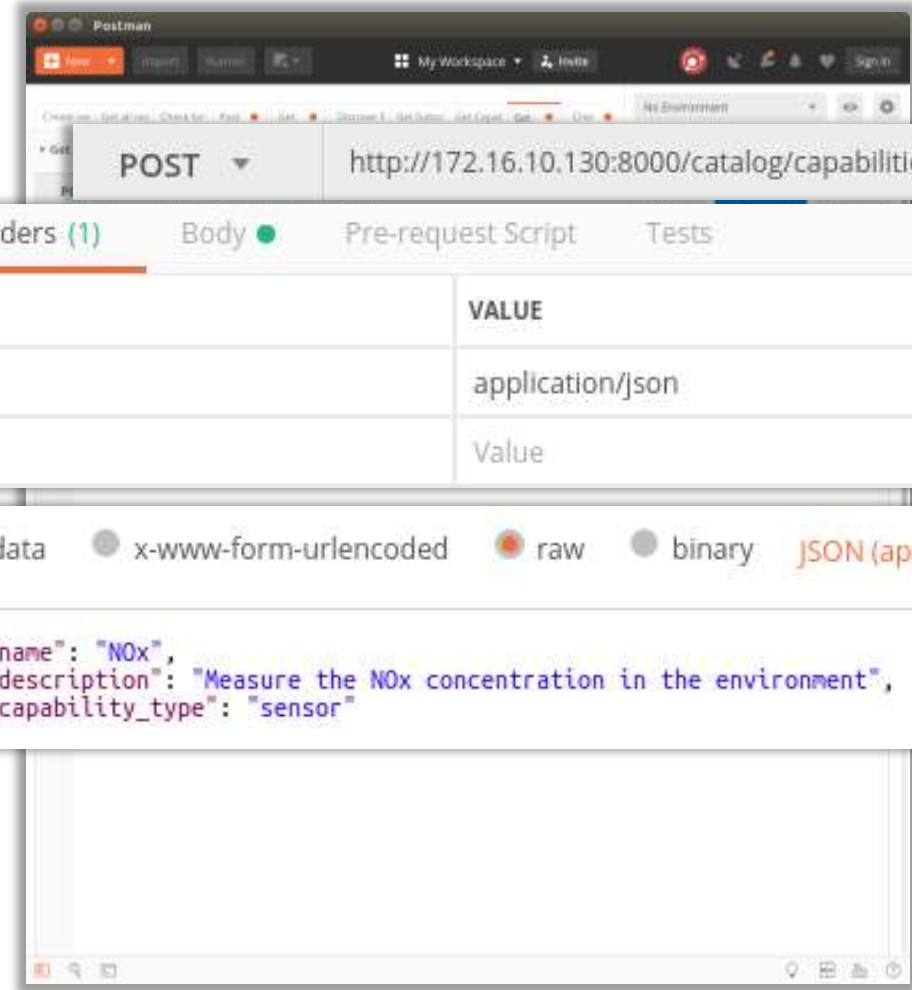
The screenshot shows a Postman interface for a POST request to `http://172.16.10.130:8000/catalog/capabilities`. The request headers are set to `Content-Type: application/json`. The request body is a JSON object:

```
{  "id": 15,  "name": "Mia",  "description": "Measure the Mlx concentration in the environment",  "capability_type": "sensor"}
```

The response status is 201 Created, with a time of 124ms and a size of 598 B.

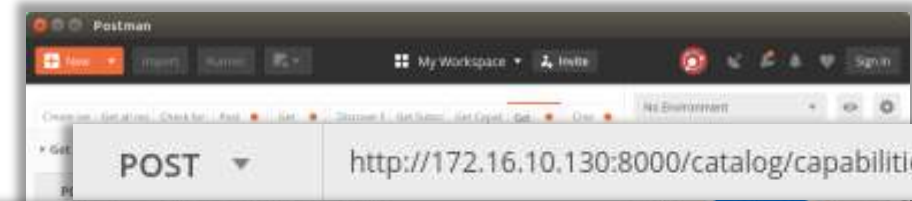
InterSCity Resource Catalog

Resource Catalog	
GET	/catalog/resources Get all resources registered on the platform.
POST	/catalog/resources Register new resources.
GET	/catalog/resources/sensors Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search Search for registered resources using different filters.
PUT	/catalog/resources/{uuid} Update an existing resource.
GET	/catalog/resources/{uuid} Get data about a specific resource.
GET	/catalog/capabilities Get all capabilities.
POST	/catalog/capabilities Create a new capability.
GET	/catalog/capabilities/{name} Get data about a specific capability.
PUT	/catalog/capabilities/{name} Update an existing capability.
DELETE	/catalog/capabilities/{name} Delete a specific capability.



InterSCity Resource Catalog

Resource Catalog	
GET	<code>/catalog/resources</code> Get all resources registered on the platform.
POST	<code>/catalog/resources</code> Register new resources.
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities.
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities.
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters.
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource.
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource.
GET	<code>/catalog/capabilities</code> Get all capabilities.
POST	<code>/catalog/capabilities</code> Create a new capability.
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability.
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability.
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability.



KEY	VALUE
Content-Type	application/json
Key	Value

```
form-data x-www-form-urlencoded raw binary JSON (application/json)
1 {
2   "name": "NOx",
3   "description": "Measure the NOx concentration in the environment",
4   "capability_type": "sensor"
5 }
```

```
Pretty Raw Preview JSON
1 {
2   "id": 25,
3   "name": "NOx",
4   "description": "Measure the NOx concentration in the environment",
5   "capability_type": "sensor"
6 }
```

InterSCity Resource Catalog

Resource Catalog ▾	
GET	<code>/catalog/resources</code> Get all resources registered on the platform
POST	<code>/catalog/resources</code> Register new resources
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource
GET	<code>/catalog/capabilities</code> Get all capabilities
POST	<code>/catalog/capabilities</code> Create a new capability
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability

InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform

POST /catalog/resources Register new resources

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities

GET /catalog/resources/search Search for registered resources using different filters

PUT /catalog/resources/{uuid} Update an existing resource

GET /catalog/resources/{uuid} Get data about a specific resource

GET /catalog/capabilities Get all capabilities

POST /catalog/capabilities Create a new capability

GET /catalog/capabilities/{name} Get data about a specific capability

PUT /catalog/capabilities/{name} Update an existing capability

DELETE /catalog/capabilities/{name} Delete a specific capability

The screenshot shows the Swagger UI for the POST endpoint `/catalog/resources`. The interface includes a title bar with the browser address `172.16.10.130:3000/Resource%20Catalog/swagger-ui`. The main content area is titled "POST /catalog/resources Register new resources" and contains the following information:

- Description:** Register a new city resource. This action will return a new UUID to the registered resource, which must be used in future requests.
- Parameters:** No parameters.
- Request body:** A dropdown menu is set to `application/json`.
- Example Value:** A JSON object representing a city resource:

```
{  "data": {    "description": "A public bus",    "capabilities": {      "temperature": "illustate"    },    "status": "active",    "lat": -23.559616,    "lon": -46.731296  }  }
```
- Responses:** A table with columns for Code, Description, and Links. The response `201` is listed with the description "The created resource" and "No links". A dropdown menu is set to `application/json`.
- Example Value (Response):** A JSON object representing the response:

```
{  "data": {    "uuid": "45b7a303-86fd-4f91-8681-66314961180f",    "description": "A public bus",    "capabilities": {      "temperature": "illustate"    }  }  }
```


InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform

POST /catalog/resources Register new resources

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities

GET /catalog/resources/search Search for registered resources using different filters

PUT /catalog/resources/{uuid} Update an existing resource

GET /catalog/resources/{uuid} Get data about a specific resource

GET /catalog/capabilities Get all capabilities

POST /catalog/capabilities Create a new capability

GET /catalog/capabilities/{name} Get data about a specific capability

PUT /catalog/capabilities/{name} Update an existing capability

DELETE /catalog/capabilities/{name} Delete a specific capability

The screenshot shows a REST client interface with a POST request to `http://172.16.10.1:8080/catalog/resources`. The request body is a JSON object representing a resource:

```
{
  "data": {
    "description": "A mobile environment sensor",
    "capabilities": {
      "temperature": "humidity",
      "humidity": "NO"
    },
    "status": "active",
    "lat": -2.524291,
    "lon": -44.294779
  }
}
```

The response is also shown in a pretty-printed JSON format:

```
{
  "data": {
    "id": "a",
    "url": null,
    "created_at": "2019-07-21T08:52:25.220Z",
    "updated_at": "2019-07-21T08:52:25.220Z",
    "lat": -2.524291,
    "lon": -44.294779,
    "status": "active",
    "collect_interval": null,
    "description": "A mobile environment sensor",
    "uuid": "528115c9-1306-460c-9f73-6aef12af1912",
    "city": null,
    "neighborhood": null,
    "state": null,
    "postal_code": null,
    "country": null,
    "capabilities": [
      "NO",
      "temperature",
      "humidity"
    ]
  }
}
```

InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform

POST /catalog/resources Register new resources

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities

GET /catalog/resources/search Search for registered resources using different filters

PUT /catalog/resources/{uuid} Update an existing resource

GET /catalog/resources/{uuid} Get data about a specific resource

GET /catalog/capabilities Get all capabilities

POST /catalog/capabilities Create a new capability

GET /catalog/capabilities/{name} Get data about a specific capability

PUT /catalog/capabilities/{name} Update an existing capability

DELETE /catalog/capabilities/{name} Delete a specific capability

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://172.16.10.130:8000/catalog/resources
- Body (Request):**

```
{
  "description": "A mobile environment sensor",
  "capabilities": {
    "temperature": "NO",
    "humidity": "NO"
  },
  "status": "active",
  "lat": -2.524291,
  "lon": -44.294779
}
```
- Body (Response):**

```
{
  "data": {
    "id": "a",
    "url": null,
    "created_at": "2019-07-21T08:52:25.228Z",
    "updated_at": "2019-07-21T08:52:25.228Z",
    "lat": -2.524291,
    "lon": -44.294779,
    "status": "active",
    "collect_interval": null,
    "description": "A mobile environment sensor",
    "uuid": "528115cb-130e-460c-9f73-6aef18af1912",
    "city": null,
    "neighborhood": null,
    "state": null,
    "postal_code": null,
    "country": null,
    "capabilities": {
      "NO": {
        "temperature": "NO",
        "humidity": "NO"
      }
    }
  }
}
```

InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform

POST /catalog/resources Register new resources

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities

GET /catalog/resources/search Search for registered resources using different filters

PUT /catalog/resources/{uuid} Update an existing resource

GET /catalog/resources/{uuid} Get data about a specific resource

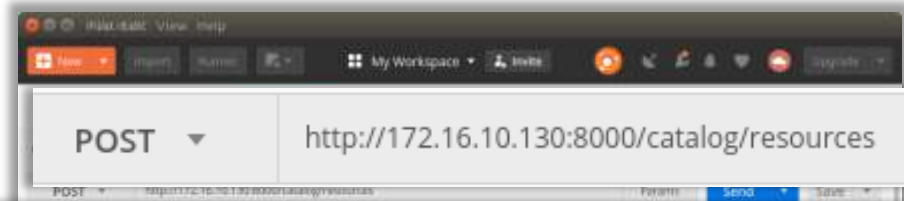
GET /catalog/capabilities Get all capabilities

POST /catalog/capabilities Create a new capability

GET /catalog/capabilities/{name} Get data about a specific capability

PUT /catalog/capabilities/{name} Update an existing capability

DELETE /catalog/capabilities/{name} Delete a specific capability



KEY	VALUE
Content-Type	application/json
Key	Value



InterSCity Resource Catalog

Resource Catalog

GET /catalog/resources Get all resources registered on the platform

POST /catalog/resources Register new resources

GET /catalog/resources/sensors Get all resources registered on the platform with sensor capabilities

GET /catalog/resources/actuators Get all resources registered on the platform with actuator capabilities

GET /catalog/resources/search Search for registered resources using different filters

PUT /catalog/resources/{uuid} Update an existing resource

GET /catalog/resources/{uuid} Get data about a specific resource

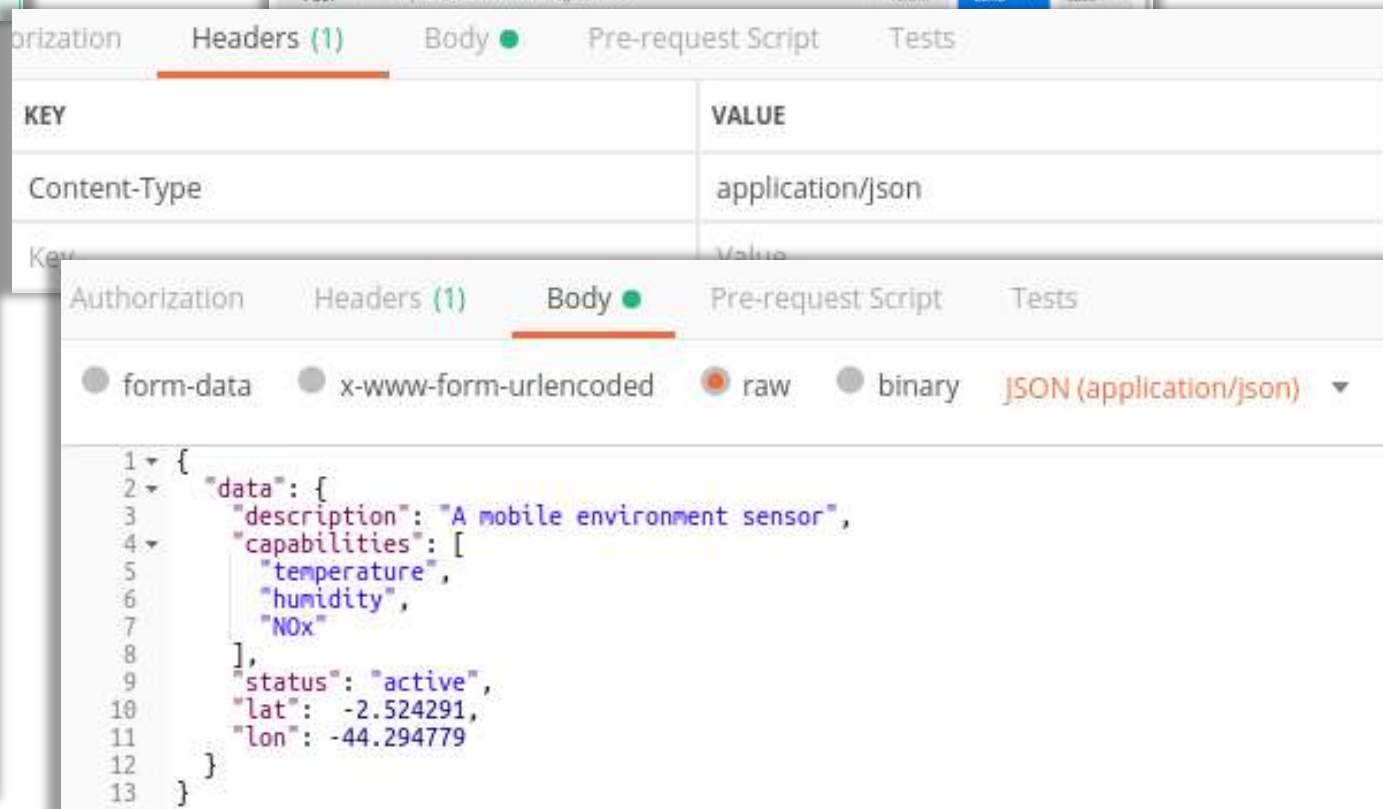
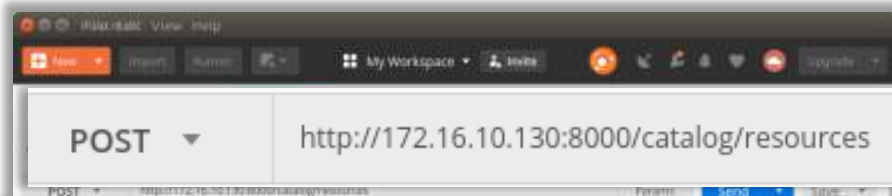
GET /catalog/capabilities Get all capabilities

POST /catalog/capabilities Create a new capability

GET /catalog/capabilities/{name} Get data about a specific capability

PUT /catalog/capabilities/{name} Update an existing capability

DELETE /catalog/capabilities/{name} Delete a specific capability



InterSCity Resource Catalog

Resource Catalog ▾	
GET	<code>/catalog/resources</code> Get all resources registered on the platform
POST	<code>/catalog/resources</code> Register new resources
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource
GET	<code>/catalog/capabilities</code> Get all capabilities
POST	<code>/catalog/capabilities</code> Create a new capability
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability

InterSCity Resource Catalog

Resource Catalog

GET	/catalog/resources	Get all resources registered on the platform.
POST	/catalog/resources	Register new resources.
GET	/catalog/resources/sensors	Get all resources registered on the platform with sensor capabilities.
GET	/catalog/resources/actuators	Get all resources registered on the platform with actuator capabilities.
GET	/catalog/resources/search	Search for registered resources using different filters.
PUT	/catalog/resources/{uuid}	Update an existing resource.
GET	/catalog/resources/{uuid}	Get data about a specific resource.
GET	/catalog/capabilities	Get all capabilities.
POST	/catalog/capabilities	Create a new capability.
GET	/catalog/capabilities/{name}	Get data about a specific capability.
PUT	/catalog/capabilities/{name}	Update an existing capability.
DELETE	/catalog/capabilities/{name}	Delete a specific capability.

Swagger UI interface showing the DELETE endpoint for /catalog/capabilities/{name}. The interface includes a description of the endpoint, a parameter table, and a response table.

DELETE /catalog/capabilities/{name} Delete a specific capability

This endpoint can be used to delete an existing capability. As a consequence, both new and updated resources will not be able to provide this capability on the platform.

Name	Description
name <small>required</small>	Capability's name
string	
(path)	

Code	Description	Links
204	The capability was destroyed	No links
404	Capability not found	No links

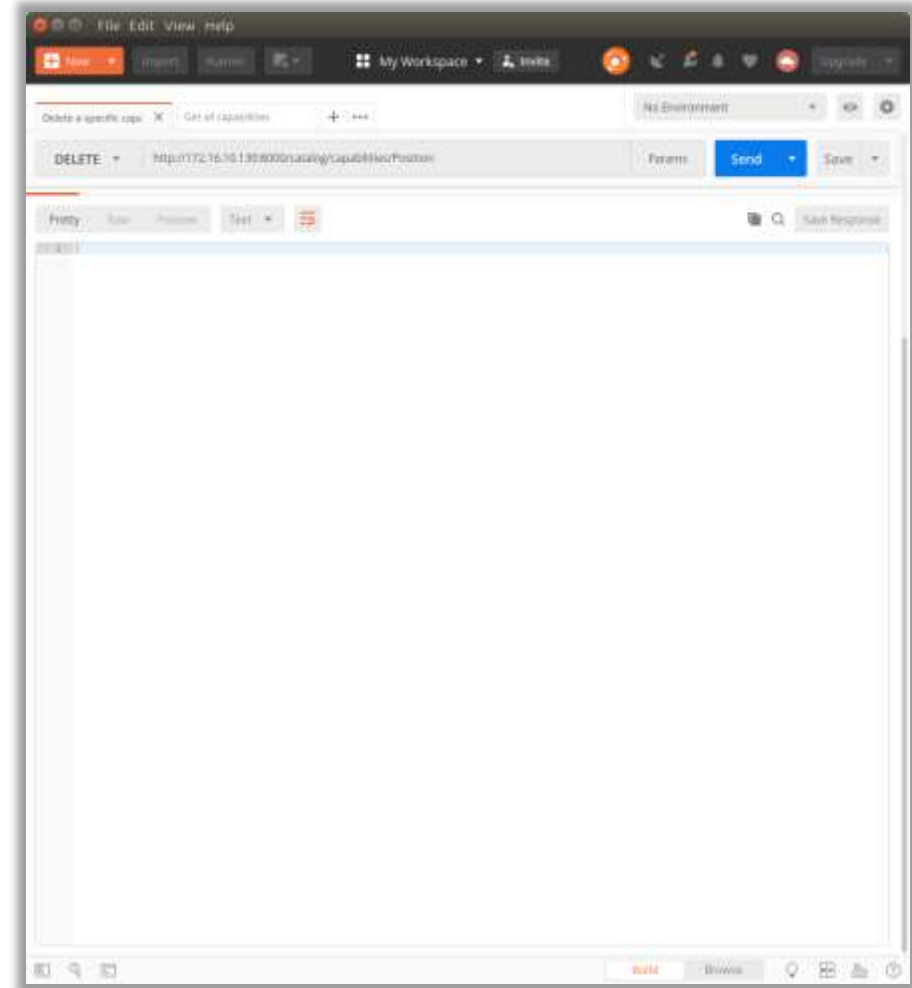
application/json

Example Value / Model

```
{ "error": "Capability not found" }
```

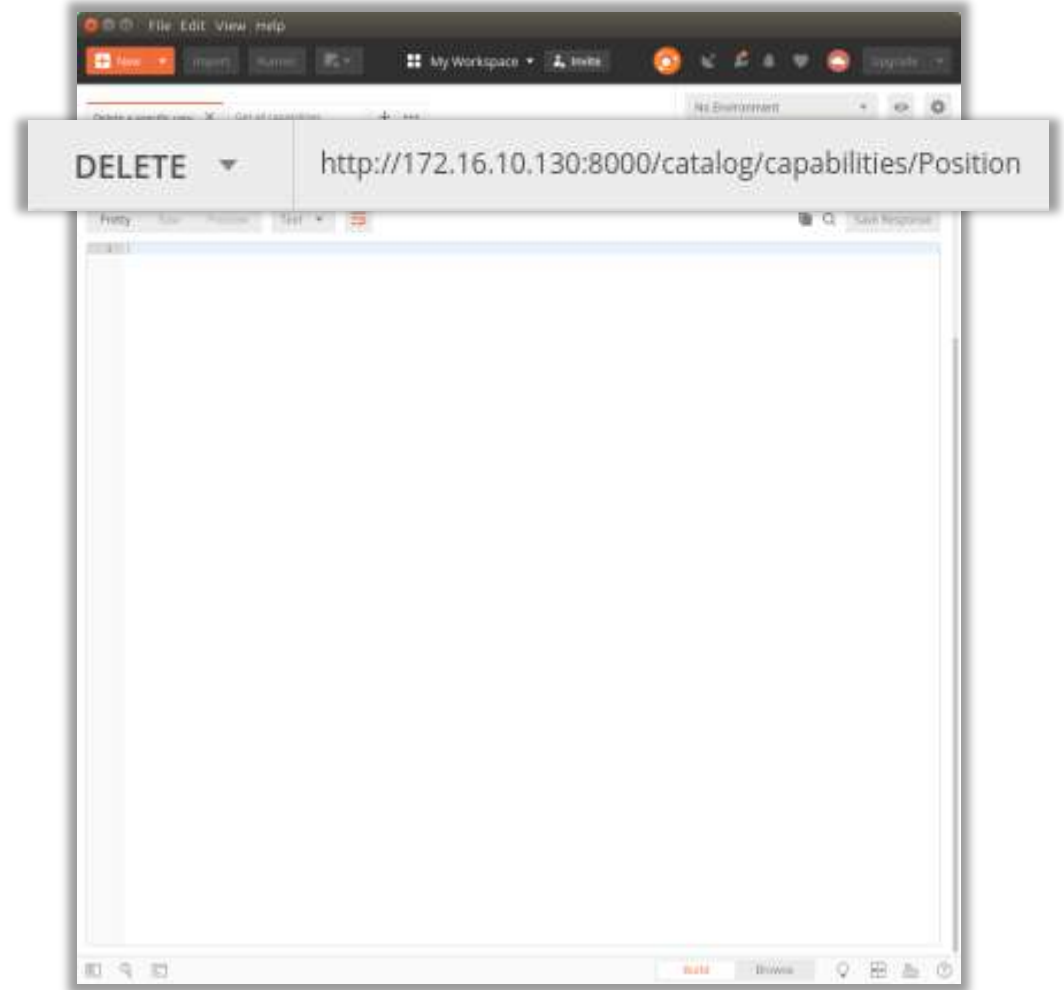
InterSCity Resource Catalog

Resource Catalog	
GET	<code>/catalog/resources</code> Get all resources registered on the platform.
POST	<code>/catalog/resources</code> Register new resources.
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities.
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities.
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters.
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource.
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource.
GET	<code>/catalog/capabilities</code> Get all capabilities.
POST	<code>/catalog/capabilities</code> Create a new capability.
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability.
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability.
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability.



InterSCity Resource Catalog

Resource Catalog	
GET	<code>/catalog/resources</code> Get all resources registered on the platform.
POST	<code>/catalog/resources</code> Register new resources.
GET	<code>/catalog/resources/sensors</code> Get all resources registered on the platform with sensor capabilities.
GET	<code>/catalog/resources/actuators</code> Get all resources registered on the platform with actuator capabilities.
GET	<code>/catalog/resources/search</code> Search for registered resources using different filters.
PUT	<code>/catalog/resources/{uuid}</code> Update an existing resource.
GET	<code>/catalog/resources/{uuid}</code> Get data about a specific resource.
GET	<code>/catalog/capabilities</code> Get all capabilities.
POST	<code>/catalog/capabilities</code> Create a new capability.
GET	<code>/catalog/capabilities/{name}</code> Get data about a specific capability.
PUT	<code>/catalog/capabilities/{name}</code> Update an existing capability.
DELETE	<code>/catalog/capabilities/{name}</code> Delete a specific capability.



InterSCity Resource Adaptor

Resource Adaptor ▾

POST /adaptor/resources Register new resources

PUT /adaptor/resources/{uuid} Update an existing resource

POST /adaptor/resources/{uuid}/data Post several context data

POST /adaptor/resources/{uuid}/data/{capability} Post context data for a single capability

POST /adaptor/subscriptions Subscribe to receive actuation commands

GET /adaptor/subscriptions Get all subscriptions

PUT /adaptor/subscriptions/{id} Update a specific subscription

GET /adaptor/subscriptions/{id} Get data about a specific subscription

InterSCity Resource Adaptor

Resource Adaptor ▾

POST /adaptor/resources Register new resources

PUT /adaptor/resources/{uuid} Update an existing resource

POST /adaptor/resources/{uuid}/data Post several context data

POST /adaptor/resources/{uuid}/data/{capability} Post context data for a single capability

POST /adaptor/subscriptions Subscribe to receive actuation commands

GET /adaptor/subscriptions Get all subscriptions

PUT /adaptor/subscriptions/{id} Update a specific subscription

GET /adaptor/subscriptions/{id} Get data about a specific subscription

The screenshot shows a Swagger UI interface for the endpoint `POST /adaptor/resources/{uuid}/data`. The description states: "Post several context data. Post a set of data collected by city resources. Resources may collect observations about the city environment through sensor devices, which provide its sensor capabilities, such as temperature, humidity, to cite a few. It is possible to post several observed data by time. Thus, the body must specify the correspondent capability for each observation." The parameters table lists `uuid` as a required string path parameter with the description "Resource's uuid". The request body is set to `application/json`. An example value is provided as a JSON object:

```
{
  "data": [
    "environment_monitoring": [
      {
        "temperature": 30,
        "humidity": 40,
        "pressure": 10,
        "timestamp": "2017-06-14T17:52:25.428Z"
      },
      {
        "temperature": 30,
        "humidity": 44,
        "pressure": 10,
        "timestamp": "2017-06-14T17:57:25.428Z"
      }
    ],
    "bus_monitoring": [
      "location": [
        "lat": -10.00000,
        "lon": -23.20000
      ],
      "speed": 54,
      "new_line": "RTSC-20-1",
      "timestamp": "2017-06-14T17:52:25.428Z"
    }
  ]
}
```

InterSCity Resource Adaptor

Resource Adaptor

POST /adaptor/resources Register new resources

PUT /adaptor/resources/{uuid} Update an existing resource

POST /adaptor/resources/{uuid}/data Post several context data

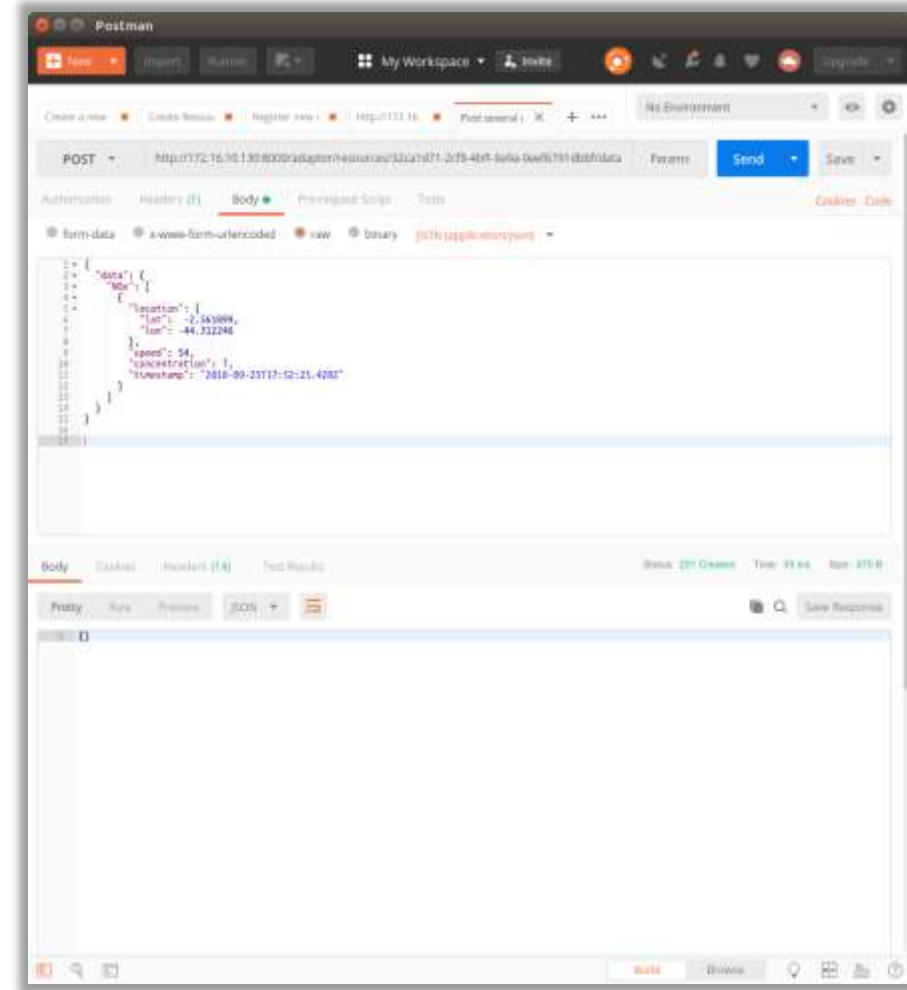
POST /adaptor/resources/{uuid}/data/{capability} Post context data for a single capability

POST /adaptor/subscriptions Subscribe to receive actuation commands

GET /adaptor/subscriptions Get all subscriptions

PUT /adaptor/subscriptions/{id} Update a specific subscription

GET /adaptor/subscriptions/{id} Get data about a specific subscription



InterSCity Resource Adaptor

Resource Adaptor

POST /adaptor/resources Register new resources

PUT /adaptor/resources/{uuid} Update an existing resource

POST /adaptor/resources/{uuid}/data Post several context data

POST /adaptor/resources/{uuid}/data/{capability} Post context data for a single capability

POST /adaptor/subscriptions Subscribe to receive actuation commands

GET /adaptor/subscriptions Get all subscriptions

PUT /adaptor/subscriptions/{id} Update a specific subscription

GET /adaptor/subscriptions/{id} Get data about a specific subscription

POST

http://172.16.10.130:8000/adaptor/resources/32ca1d71-2cf9-4bff-9a9a-0eef6791dbbf/data

The screenshot shows a Postman interface with a successful POST request. The URL is `http://172.16.10.130:8000/adaptor/resources/32ca1d71-2cf9-4bff-9a9a-0eef6791dbbf/data`. The request body is a JSON object:

```
{  "data": {    "id": "32ca1d71-2cf9-4bff-9a9a-0eef6791dbbf",    "location": {      "lat": -2.563088,      "lon": -44.322246    },    "speed": 54,    "spicestrafim": 1,    "timestamp": "2018-09-23T17:12:25.428Z"  }}
```

The response is empty, and the status is 201 Created. The interface also shows the 'Body' tab selected, and the response is displayed in the 'Pretty' view.

InterSCity Resource Adaptor

The image displays a REST API client interface for the "Resource Adaptor". On the left, a list of endpoints is shown with their respective HTTP methods and descriptions:

- POST** /adaptor/resources Register new resources
- PUT** /adaptor/resources/{uuid} Update an existing resource
- POST** /adaptor/resources/{uuid}/data Post several context data
- POST** /adaptor/resources/{uuid}/data/{capability} Post context data for a capability
- POST** /adaptor/subscriptions Subscribe to receive actuation commands
- GET** /adaptor/subscriptions Get all subscriptions
- PUT** /adaptor/subscriptions/{id} Update a specific subscription
- GET** /adaptor/subscriptions/{id} Get data about a specific subscription

The main area shows a selected **POST** request to `http://172.16.10.130:8000/adaptor/resources/32ca1d71-2cf9-4bff-9a9a-0eef6791dbbf/data`. The "Headers (1)" tab is active, displaying the following header:

KEY	VALUE
Content-Type	application/json
Key	Value

The "Body" tab is also visible, showing a JSON body structure:

```
{}  
}
```

InterSCity Resource Adaptor

The image displays a REST API interface for the 'Resource Adaptor' and a detailed view of a POST request in Postman.

Resource Adaptor API Endpoints:

- POST** /adaptor/resources Register new resources
- PUT** /adaptor/resources/{uuid} Update an existing resource
- POST** /adaptor/resources/{uuid}/data Post several context data
- POST** /adaptor/resources/{uuid}/data/{capability} Post context data for a capability
- POST** /adaptor/subscriptions Subscribe to receive actuation commands
- GET** /adaptor/subscriptions Get all subscriptions
- PUT** /adaptor/subscriptions/{id} Update a specific subscription
- GET** /adaptor/subscriptions/{id} Get data about a specific subscription

Postman Request Details:

- Method:** POST
- URL:** http://172.16.10.130:8000/adaptor/resources/32ca1d71-2cf9-4bff-9a9a-0eef6791dbbf/data
- Headers (1):**

KEY	VALUE
Content-Type	application/json
- Body:** raw (JSON (application/json))

```
1 {
2   "data": {
3     "NOx": [
4       {
5         "location": {
6           "lat": -2.561899,
7           "lon": -44.312246
8         },
9         "speed": 54,
10        "concentration": 7,
11        "timestamp": "2018-09-25T17:52:25.428Z"
12      }
13    ]
14  }
15 }
```

InterSCity

Actuator Controller

Actuator Controller ▾

GET

`/actuator/commands`

Get all actuator commands requested to the platform by client applications

POST

`/actuator/commands`

Send commands to city resources

InterSCity Data Collector

Data Collector ▾

POST

`/collector/resources/data`

Get historical data of sensor capabilities of all city resources

POST

`/collector/resources/{uuid}/data`

Get historical data of sensor capabilities of a specific city resource

POST

`/collector/resources/data/last`

Get the most recent data of sensor capabilities from all city resources

POST

`/collector/resources/{uuid}/data/last`

Get the most recent data of sensor capabilities from a specific city resource

InterSCity

Resource Discovery

Resource Discovery

GET

`/discovery/resources` Context-aware search endpoint to discovery city resources

InterSCity Resource Discovery

Resource Discovery

GET /discovery/resources Context-aware search endpoint to discovery city resources

The screenshot shows the Swagger UI for the endpoint `GET /discovery/resources`. The title is "Context-aware search endpoint to discovery city resources". The description states: "This endpoint can be used to search for city resources based context data. One may combine static and dynamic filters to discovery existing resources. One may use the following context-based filters for resource discovery:"

- capability** - The system will filter the results by resources with the given capability.
- matchers of value** - Filter the data collected based on the current context data of resources through value matchers. This filter must be used to perform dynamic data-based queries by specifying the attribute of context data and their matching rules. You can combine several matchers in a single request. The following operators are available:
 - eq** - Specifies equality condition.
 - gt** - Selects those data where the value of the specified attribute is greater than a specified value.
 - gte** - Selects those data where the value of the specified attribute is greater than or equal a specified value.
 - lt** - Selects those data where the value of the specified attribute is less than a specified value.
 - lte** - Selects those data where the value of the specified attribute is less than or equal a specified value.
 - in** - Selects those data where the value of the specified attribute equals any value in the specified array.
 - ne** - Selects those data where the value of the specified attribute is not equal to a specified value. It's a good idea to use this filter combined with the **capabilities** to avoid returning data that do not even have this attribute.
- location** - To discovery resources based on location data, you must always provide both **lat** and **lon** parameters. With these two parameters, the system will discovery resources which are located exactly at the point informed. However, you may also combine them with the following parameter:
 - radius** - Tell the system to include resources near to the location formed by **lat** and **lon** considering the radius (in meters) informed.

See the following examples using curl:

- Search for resources which have the ability to monitor the environment:

```
curl http://localhost:8080/discovery/resources?capability=environment_monitoring
```
- Search for resources that provides context data related to medical procedures and whose last procedure was carried out in the specialty of either cardiology or psychiatry, in male patients and over the age of 18 years:

```
curl --globoff -L "http://localhost:8080/discovery/resources?specialty.in[]=cardiology&specialty.in[]=psychiatry&patient.age.gte=18&patient.gender.eq=male"
```
- Search for resources with the capability of environment monitoring and whose last data regarding the temperature is between 18 and 30 degrees celsius:

```
curl --globoff -L "http://localhost:8080/discovery/resources?capability=environment_monitoring&temperature.gte=18&temperature.lte=30"
```
- Search for resources near to the Galéria do Rock (located at -23.543793, -46.638736), considering a radius of 500 meters:

```
curl --globoff -L "http://localhost:8080/discovery/resources?lat=-23.543794&lon=-46.638736&radius=500"
```

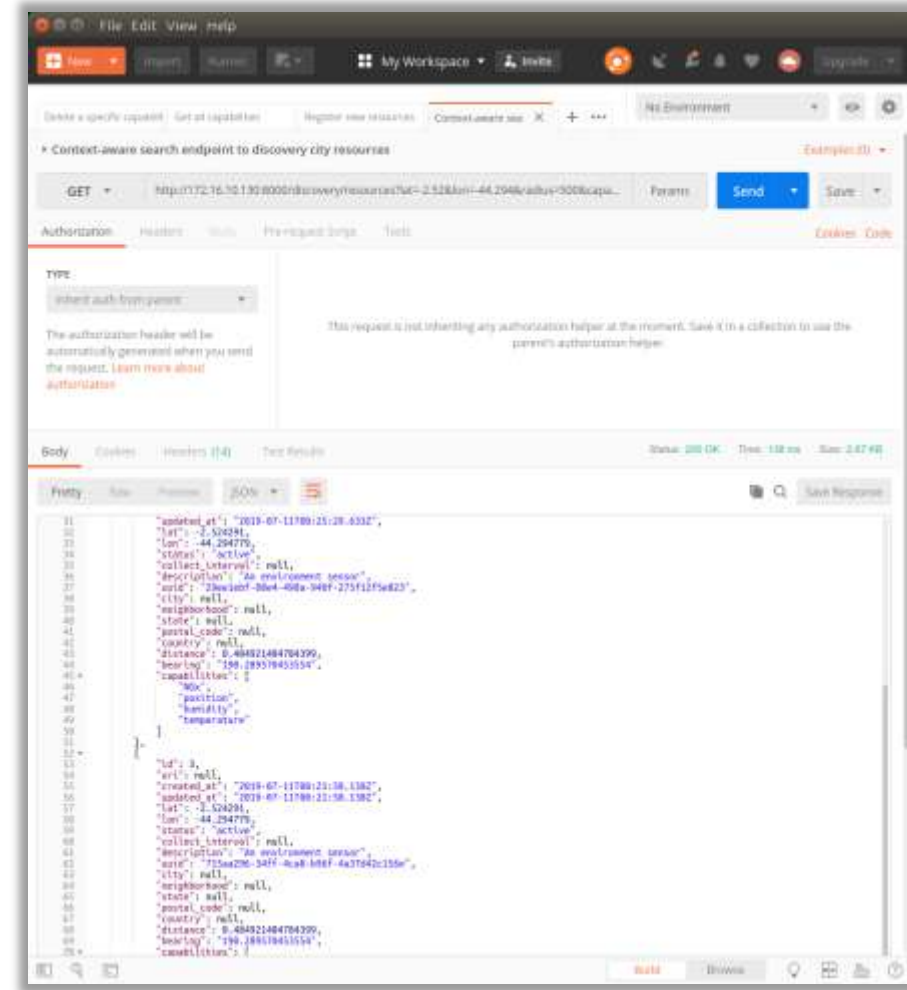
Parameters

Name	Description
capability	Filter by resources with a given capability

InterSCity Resource Discovery

Resource Discovery

GET /discovery/resources Context-aware search endpoint to discovery city resources



InterSCity Resource Discovery

GET ▾

`http://172.16.10.130:8000/discovery/resources?lat=-2.52&lon=-44.294&radius=500&capability=NOx`

Resource Discovery ▾

GET `/discovery/resources` Context-aware search endpoint to discovery city resources

The screenshot shows a REST client interface with a GET request and its JSON response. The request is `http://172.16.10.130:8000/discovery/resources?lat=-2.52&lon=-44.294&radius=500&capability=NOx`. The response is a JSON array of two resource objects. The first object is:

```
{
  "updated_at": "2019-07-11T00:21:28.632Z",
  "lat": -2.524291,
  "lon": -44.294779,
  "status": "active",
  "collect_interval": null,
  "description": "an environment sensor",
  "uid": "2ba1e0f-80e4-40da-940f-275f12f5e823",
  "city": null,
  "neighborhood": null,
  "state": null,
  "postal_code": null,
  "country": null,
  "distance": 0.484921484704399,
  "bearing": "190.185978455554",
  "capabilities": {
    "Nox": {
      "position": {
        "bearing": {
          "temperature"
        }
      }
    }
  }
}
```

The second object is:

```
{
  "uid": 0,
  "lat": null,
  "lon": null,
  "created_at": "2019-07-11T00:21:58.138Z",
  "updated_at": "2019-07-11T00:21:58.138Z",
  "lat": -2.524291,
  "lon": -44.294779,
  "status": "active",
  "collect_interval": null,
  "description": "an environment sensor",
  "uid": "735ae296-34ff-4a4b-b1ef-4a37042c1556",
  "city": null,
  "neighborhood": null,
  "state": null,
  "postal_code": null,
  "country": null,
  "distance": 0.484921484704399,
  "bearing": "190.186194615554",
  "capabilities": {
  }
}
```

InterSCity

Resource Discovery

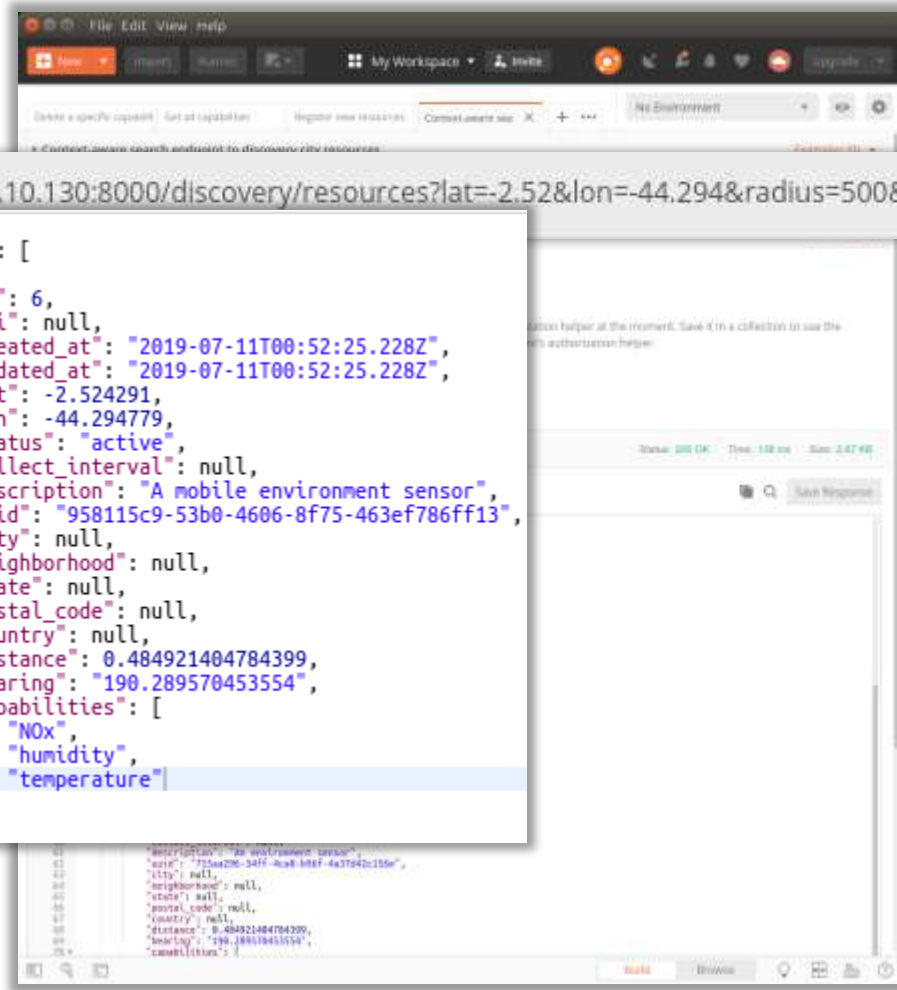
GET

http://172.16.10.130:8000/discovery/resources?lat=-2.52&lon=-44.294&radius=500&capability=NOx

Resource Discovery

GET /discovery/resources Context-aware search endpoint to discovery city resources

```
{
  "resources": [
    {
      "id": 6,
      "uri": null,
      "created_at": "2019-07-11T00:52:25.228Z",
      "updated_at": "2019-07-11T00:52:25.228Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "A mobile environment sensor",
      "uuid": "958115c9-53b0-4606-8f75-463ef786ff13",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "humidity",
        "temperature"
      ]
    }
  ]
}
```



InterSCity

Resource Discovery

GET ▾

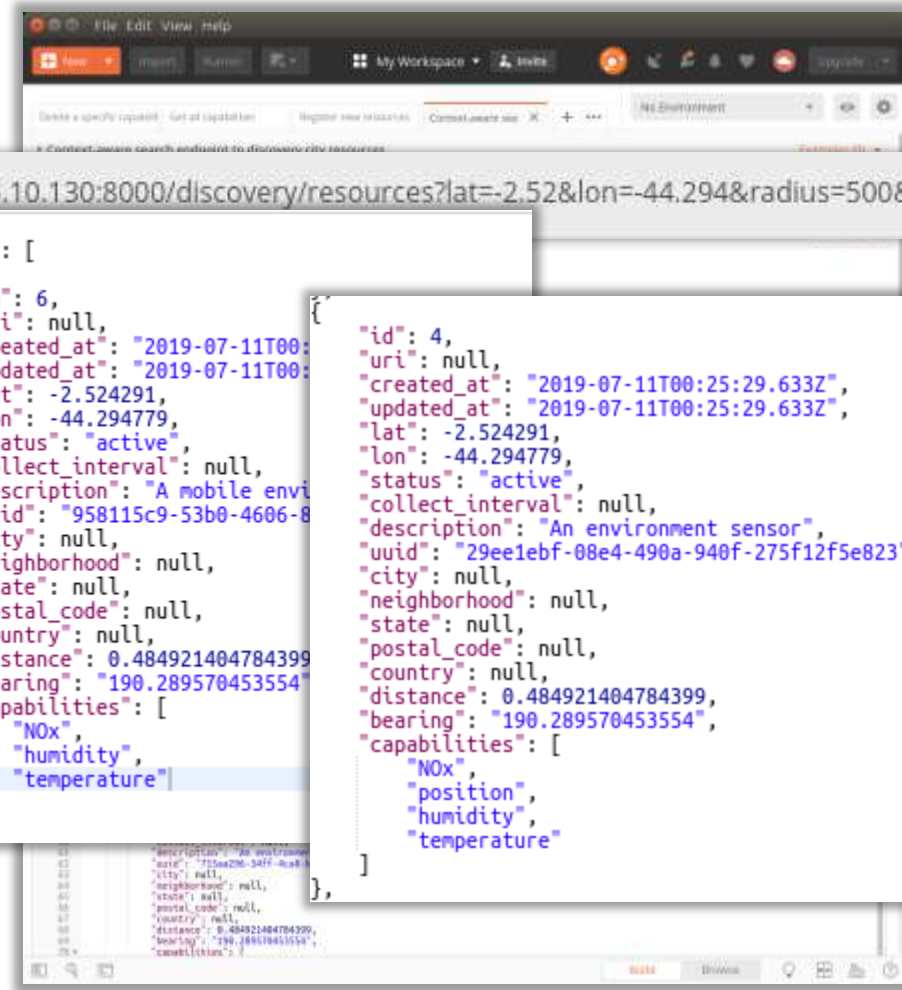
http://172.16.10.130:8000/discovery/resources?lat=-2.52&lon=-44.294&radius=500&capability=NOx

Resource Discovery ▾

GET /discovery/resources Context-aware search endpoint to discovery city resources

```
{
  "resources": [
    {
      "id": 6,
      "uri": null,
      "created_at": "2019-07-11T00:00:00.000Z",
      "updated_at": "2019-07-11T00:00:00.000Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "A mobile environment sensor",
      "uuid": "958115c9-53b0-4606-8000-000000000000",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "humidity",
        "temperature"
      ]
    }
  ]
}
```

```
{
  "resources": [
    {
      "id": 4,
      "uri": null,
      "created_at": "2019-07-11T00:25:29.633Z",
      "updated_at": "2019-07-11T00:25:29.633Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "An environment sensor",
      "uuid": "29ee1ebf-08e4-490a-940f-275f12f5e823",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "position",
        "humidity",
        "temperature"
      ]
    }
  ]
}
```



InterSCity Resource Discovery

GET

http://172.16.10.130:8000/discovery/resources?lat=-2.52&lon=-44.294&radius=500&capability=NOx

Resource Discovery

GET /discovery/resources Context-aware search endpoint to discovery city resources

```
{
  "resources": [
    {
      "id": 6,
      "uri": null,
      "created_at": "2019-07-11T00:25:29.633Z",
      "updated_at": "2019-07-11T00:25:29.633Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "An environment sensor",
      "uuid": "958...",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "humidity",
        "temperature"
      ]
    },
    {
      "id": 4,
      "uri": null,
      "created_at": "2019-07-11T00:25:29.633Z",
      "updated_at": "2019-07-11T00:25:29.633Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "An environment sensor",
      "uuid": "715aa296-34ff-4ca8-b96f-4a37d42c156e",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "position",
        "humidity",
        "temperature"
      ]
    },
    {
      "id": 3,
      "uri": null,
      "created_at": "2019-07-11T00:21:50.138Z",
      "updated_at": "2019-07-11T00:21:50.138Z",
      "lat": -2.524291,
      "lon": -44.294779,
      "status": "active",
      "collect_interval": null,
      "description": "An environment sensor",
      "uuid": "715aa296-34ff-4ca8-b96f-4a37d42c156e",
      "city": null,
      "neighborhood": null,
      "state": null,
      "postal_code": null,
      "country": null,
      "distance": 0.484921404784399,
      "bearing": "190.289570453554",
      "capabilities": [
        "NOx",
        "position",
        "humidity",
        "temperature"
      ]
    }
  ]
}
```

Perguntas?



Exercícios

Exercício

- Crie uma “capacidade” para monitorar a qualidade do ar
- Crie 3 recursos que possuam capacidade de monitorar a qualidade do ar
- Entre com os dados nos recursos
 - Suponha que os recursos são móveis

InterSCity

Via Java

Classe MyURLConnection

Pacotes

```
package br.com.meslin.alert.connection;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import br.com.meslin.alert.util.Debug;
```

Classe MyHTTPConnection

Construtor e método auxiliar

```
public class MyHTTPConnection {
    private static final String USER_AGENT = "Mozilla/5.0";
    private String interSCityURI;

    public MyHTTPConnection() {
        interSCityURI = Constants.INTERSCITY_URL;
    }

    public MyHTTPConnection(String interSCityIPAddress) {
        if(interSCityIPAddress == null) {
            interSCityURI = Constants.INTERSCITY_URL;
        } else {
            interSCityURI = "http://" + interSCityIPAddress + ":8000";
        }
    }
    ...
}

public String getIpAddress() {
    return interSCityURI;
}
```

Classe MyURLConnection

Delete

```
public String sendDelete(String directory, String data) throws Exception {
    final String url = intersCityURI + "/" + directory + "/" + data;
    final String method = "DELETE";

    HttpURLConnection con = (HttpURLConnection) (new URL(url)).openConnection();
    con.setRequestMethod(method);
    con.setRequestProperty("User-agent", USER_AGENT);
    con.setDoOutput(true);

    int responseCode = con.getResponseCode();
    if(responseCode /100 != 2) { throw new HTTPException(responseCode); }
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String inputLine;
    StringBuffer answer = new StringBuffer();
    while((inputLine = in.readLine()) != null) {
        answer.append(inputLine) ;
    }
    in.close();
    con.disconnect();
    return answer.toString();
}
```

Classe MyURLConnection

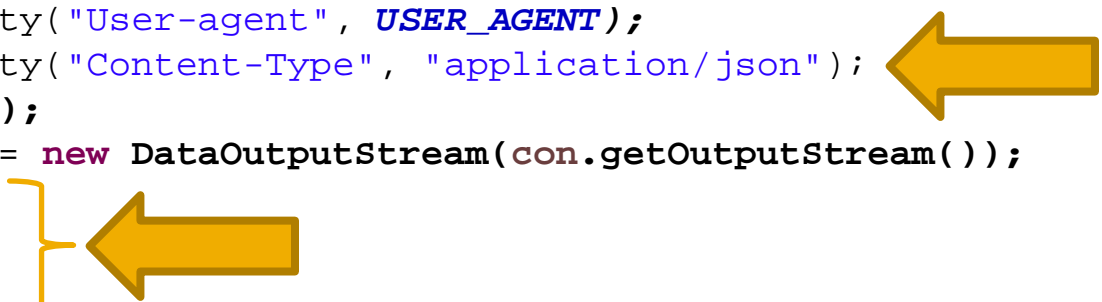
Get

```
public String sendGet(String directory, String data) throws Exception {
    final String url = intersCityURI + "/" + directory + "?" + data;
    final String method = "GET";
    HttpURLConnection con = (HttpURLConnection) (new URL(url)).openConnection();
    con.setRequestMethod(method);
    con.setRequestProperty("User-agent", USER_AGENT);
    int responseCode;
    try {
        responseCode = con.getResponseCode();
    } catch (Exception e) {
        throw new Exception(e.getMessage());
    }
    if(responseCode /100 != 2) { throw new HTTPException(responseCode); }
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String inputLine;
    StringBuffer answer = new StringBuffer();
    while ((inputLine = in.readLine()) != null) { answer.append(inputLine) ; }
    in.close();
    con.disconnect();
    return answer.toString();
}
```

Classe MyHTTPConnection

Put

```
public String sendPut(String directory, String data) throws Exception {
    final String url = intersCityURI + "/" + directory;
    final String method = "PUT";
    HttpURLConnection con = (HttpURLConnection) (new URL(url)).openConnection();
    con.setRequestMethod(method);
    con.setRequestProperty("User-agent", USER_AGENT);
    con.setRequestProperty("Content-Type", "application/json");
    con.setDoOutput(true);
    DataOutputStream wr = new DataOutputStream(con.getOutputStream());
    wr.writeBytes(data);
    wr.flush();
    wr.close();
    int responseCode = con.getResponseCode();
    if(responseCode /100 != 2) { throw new HTTPException(responseCode); }
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String inputLine;
    StringBuffer answer = new StringBuffer();
    while ((inputLine = in.readLine()) != null) { answer.append(inputLine) ; }
    in.close();
    con.disconnect();
    return answer.toString();
}
```



Classe MyURLConnection

Post

```
public String sendPost(String directory, String jsonString) throws Exception {
    final String url = intersCityURI+"/"+directory; final String method = "POST"; String inputLine;
    HttpURLConnection con = (HttpURLConnection) (new URL(url)).openConnection();
    con.setRequestMethod(method);
    con.setRequestProperty("User-agent", USER_AGENT);
    con.setRequestProperty("Content-Type", "application/json");
    con.setDoOutput(true);
    DataOutputStream wr = new DataOutputStream(con.getOutputStream());
    wr.writeBytes(jsonString); wr.flush(); wr.close();
    int responseCode = con.getResponseCode();
    if(responseCode /100 != 2) {
        BufferedReader in = new BufferedReader(new InputStreamReader(con.getErrorStream()));
        StringBuffer answer = new StringBuffer();
        while ((inputLine = in.readLine()) != null) { answer.append(inputLine) ; }
        in.close(); con.disconnect();
        throw new HTTPException("Code: " + responseCode + ": " + answer.toString());
    }
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    StringBuffer answer = new StringBuffer();
    while ((inputLine = in.readLine()) != null) { answer.append(inputLine) ; }
    in.close(); con.disconnect();
    return answer.toString();
}
```

Finalmente um código com leitura de msg de erro!!!

Perguntas



Instalação do ContextNet

Instalação do ContextNet

Java 8

OpenSplice
6.9

Hyperic
SIGAR

ContextNet
2.7

Instalação do ContextNet Java

- Java
 - No momento que esse documento foi gerado, o ContextNet somente era compatível com Java 8.
 - Para instalar o Java 8, utilize os seguintes comandos:

```
$ sudo apt install openjdk-8-jdk-headless
```

Instalação do ContextNet OpenSplice

- Instale o OpenSplice.
 - Instruções mais detalhadas em <http://www.lac.inf.puc-rio.br/dokuwiki/doku.php?id=installingdds>
- Faça download (ou upload) do arquivo `OpenSpliceDDSV6.4.140407OSS-HDE-x86_64.linux-gcc4.6-glibc2.15-installer.tar.gz`
- Descompacte o arquivo, crie o diretório `/opt/OpenSplice` e copie o diretório HDE para `/opt/OpenSplice`

```
$ tar -xvf OpenSpliceDDSV6.4.140407OSS-HDE-x86_64.linux-gcc4.6-glibc2.15-  
installer.tar.gz  
$ sudo mkdir /opt/OpenSplice  
$ sudo mv HDE /opt/OpenSplice/
```

- O resultado será:

```
$ ls -l /opt/OpenSplice/  
total 4  
drwxrwxr-x 3 contextnet contextnet 4096 Apr 14 2014 HDE
```

Instalação do ContextNet OpenSplice

Para facilitar, podemos fazer download a partir do Google Drive ou do GitHub.

OpenSplice v6.9:


- https://drive.google.com/file/d/1iwcjNxzdPK5zoqvTpg7w9mCmiN8b_Lbv/view?usp=sharing
- <https://github.com/ADLINK-IST/opensplice/releases>

Instalação do ContextNet OpenSplice

Crie um arquivo vazio chamado `opensplice.sh` e torne o arquivo executável:

- `$ sudo vi /etc/profile.d/opensplice.sh`
- `$ sudo chmod +x /etc/profile.d/opensplice.sh`

Copie o seguinte conteúdo para dentro do arquivo:

- `OSPL_HOME=/opt/OpenSplice/HDE/x86_64.linux`
- `PATH=$OSPL_HOME/bin:$PATH`
- `LD_LIBRARY_PATH=$OSPL_HOME/lib:$LD_LIBRARY_PATH` 
- `CPATH=$OSPL_HOME/include:$OSPL_HOME/include/sys:${CPATH:=}`
- `OSPL_URI=file://$OSPL_HOME/etc/config/ospl.xml`
- `export OSPL_HOME PATH LD_LIBRARY_PATH CPATH OSPL_TMPL_PATH OSPL_URI`

Instalação do ContextNet OpenSplice

Edite o arquivo

`/opt/OpenSplice/HDE/x86_64.linux/etc/config/ospl.xml:`

- `$ sudo vi /opt/OpenSplice/HDE/x86_64.linux/etc/config/ospl.xml`

Substitua o seu conteúdo pelo texto disponível em

- http://wiki.lac.inf.puc-rio.br/doku.php?do=export_code&id=installingdds&codeblock=1

Instalação do ContextNet

Hyperic SIGAR

Opcionalmente, para evitar a mensagem de erro ao iniciar o Gateway do ContextNet, instale a biblioteca SIGAR.

Faça upload (ou download) da biblioteca hyperic-sigar-1.6.4.zip ou equivalente para o host a partir de

- <https://sourceforge.net/projects/sigar/files/>

Descompacte, mova e muda as permissões com os comandos a seguir (pode ser necessário instalar o unzip):

- `$ unzip hyperic-sigar-1.6.4.zip`
- `$ sudo mv hyperic-sigar-1.6.4/sigar-bin/lib/*.so /opt/OpenSplice/HDE/x86_64.linux/lib/`
- `$ sudo chmod 775 /opt/OpenSplice/HDE/x86_64.linux/lib/libsigar-*`

Instalação do ContextNet (Finalmente!!!)

Download do ContextNet:

- `http://wiki.lac.inf.puc-rio.br/doku.php?id=download`

Javadoc

- `http://wiki.lac.inf.puc-rio.br/api/`

Crie o diretório `/opt/ContextNet` e copie o ContextNet (a ClientLib e a UDILib) para lá.

- `$ sudo mkdir /opt/ContextNet`
- `$ sudo cp contextnet-2.7.jar /opt/ContextNet`

Perguntas?



Desenvolvendo primeiras aplicações

ContextNet

Desenvolvendo aplicações com o ContextNet

Tutorial

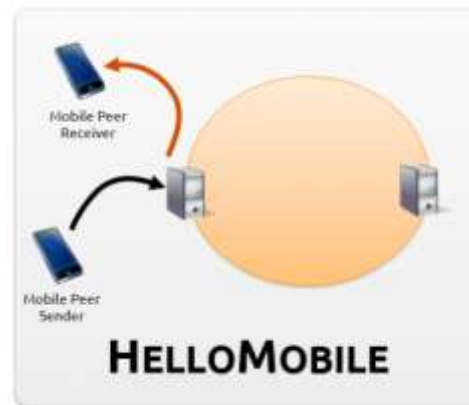
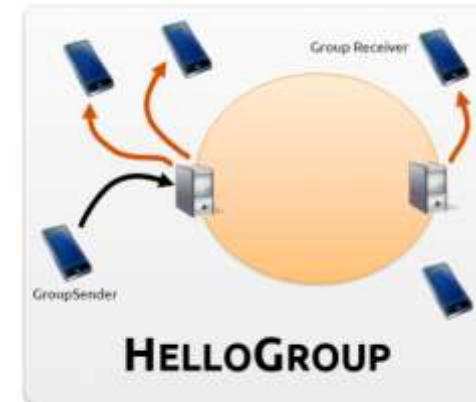
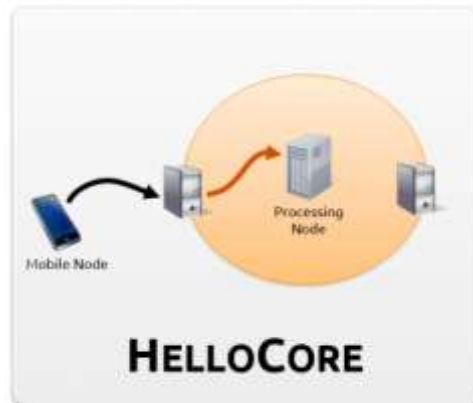
HelloCore

HelloMobile

HelloGroup

HelloAndroid

- Tutorial em <http://wiki.lac.inf.puc-rio.br/doku.php?id=tutorial>



Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

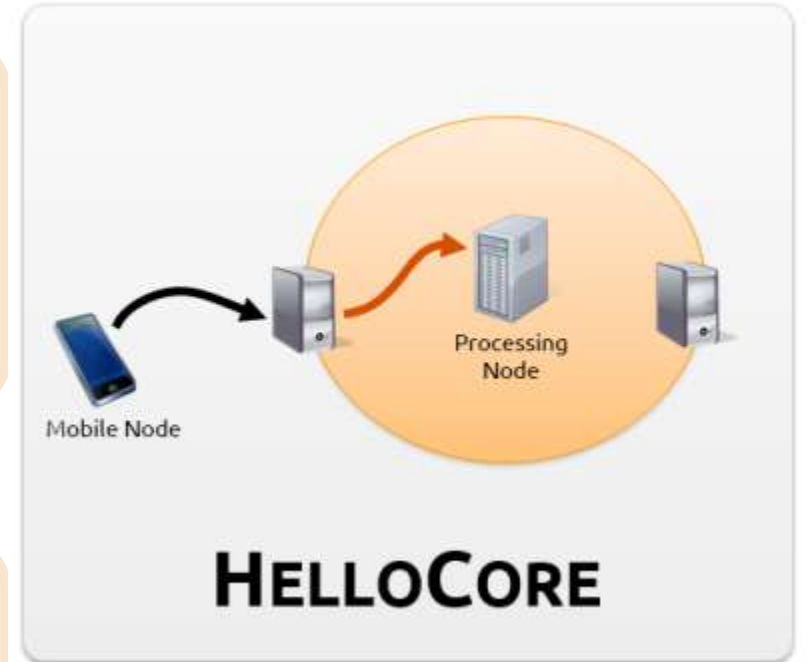
HelloAndroid

Dois componentes:

- Cliente móvel
- Nó fixo

Objetivo

- Enviar uma mensagem do cliente móvel para o nó fixo



Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

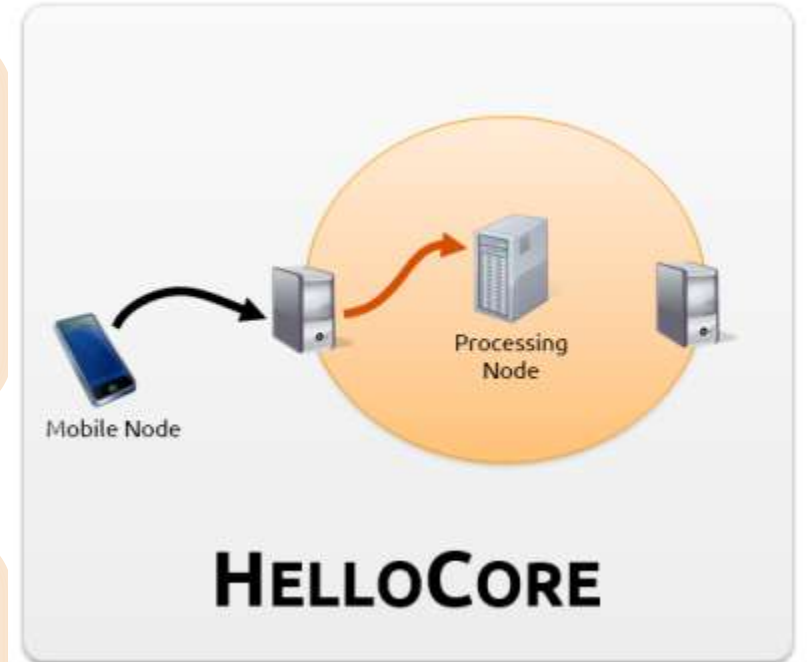
HelloAndroid

Dois componentes:

- Cliente móvel
- Nó fixo

Objetivo

- Enviar uma mensagem do cliente móvel para o nó fixo



Desenvolvendo aplicações com o ContextNet

HelloCoreClient (1/4) – pacotes

Tutorial

HelloCore

```
package br.com.meslin.lac;  
  
import java.io.IOException;  
import java.net.InetSocketAddress;  
import java.net.SocketAddress;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;
```

HelloMobile

```
import lac.cnclib.net.NodeConnection;  
import lac.cnclib.net.NodeConnectionListener;  
import lac.cnclib.net.mrudp.MrUdpNodeConnection;  
import lac.cnclib.sddl.message.ApplicationMessage;  
import lac.cnclib.sddl.message.Message;
```

HelloGroup

HelloAndroid

Desenvolvendo aplicações com o ContextNet HelloCoreClient (2/4) – construtor e main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloCoreClient implements NodeConnectionListener {
    private static String gatewayIP = "127.0.0.1";
    private static int gatewayPort = 5500;
    private MrUdpNodeConnection connection;

    public static void main(String[] args) {
        Logger.getLogger("").setLevel(Level.OFF);
        new HelloCoreClient();
    }

    public HelloCoreClient() {
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        try {
            connection = new MrUdpNodeConnection();
            connection.addNodeConnectionListener(this);
            connection.connect(address);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet HelloCoreClient (3/4) – envio da mensagem

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
@Override
public void connected(NodeConnection remoteCon) {
    ApplicationMessage message = new ApplicationMessage();
    String serializableContent = "Hello World";
    message.setContentObject(serializableContent);

    try {
        connection.sendMessage(message);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override
public void newMessageReceived(NodeConnection remoteCon, Message message) {
    System.out.println(message.getContentObject());
}
```

Envio de mensagem
para o núcleo
(Processing Node)

Desenvolvendo aplicações com o ContextNet HelloCoreClient (4/4) – métodos auxiliares

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
// other methods

@Override
public void reconnected(NodeConnection remoteCon, SocketAddress endPoint, boolean wasHandover,
boolean wasMandatory) {}

@Override
public void disconnected(NodeConnection remoteCon) {}

@Override
public void unsentMessages(NodeConnection remoteCon, List<Message> unsentMessages) {}

@Override
public void internalException(NodeConnection remoteCon, Exception e) {}
}
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

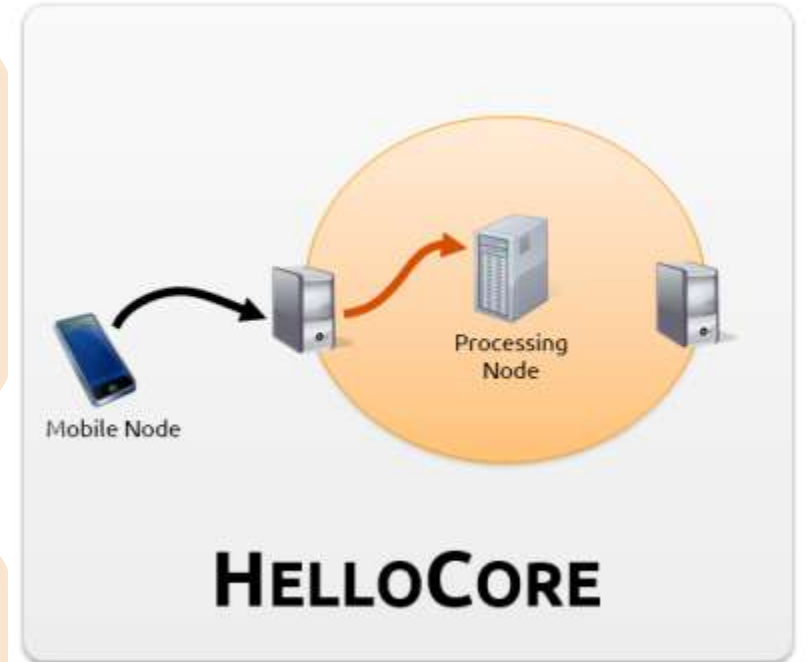
HelloAndroid

Dois componentes:

- Cliente móvel
- Nó fixo

Objetivo

- Enviar uma mensagem do cliente móvel para o nó fixo



Desenvolvendo aplicações com o ContextNet HelloCoreServer (1/5) – pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
```

```
import lac.cnclib.sddl.message.ApplicationMessage;
```

```
import lac.cnclib.sddl.serialization.Serialization;
```

```
import lac.cnet.sddl.objects.ApplicationObject;
```

```
import lac.cnet.sddl.objects.Message;
```

```
import lac.cnet.sddl.objects.PrivateMessage;
```

```
import lac.cnet.sddl.udi.core.SddlLayer;
```

```
import lac.cnet.sddl.udi.core.UniversalDDSLayerFactory;
```

```
import lac.cnet.sddl.udi.core.listener.UdiDataReaderListener;
```

Desenvolvendo aplicações com o ContextNet HelloCoreServer (2/5) – main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloCoreServer implements UDIDataReaderListener<ApplicationObject> {  
    SddlLayer core;  
    int counter;  
  
    public static void main(String[] args) {  
        Logger.getLogger( "" ).setLevel(Level.OFF);  
  
        new HelloCoreServer();  
    }  
}
```

Desenvolvendo aplicações com o ContextNet HelloCoreServer (3/5) – registro

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloCoreServer implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    public HelloCoreServer() {  
        core = UniversalDDSLayerFactory.getInstance();  
        core.createParticipant(UniversalDDSLayerFactory.CNET_DOMAIN);  
  
        core.createPublisher();  
        core.createSubscriber();  
  
        Object receiveMessageTopic = core.createTopic(Message.class, Message.class.getSimpleName());  
        core.createDataReader(this, receiveMessageTopic);  
  
        Object toMobileNodeTopic = core.createTopic(  
            PrivateMessage.class,  
            PrivateMessage.class.getSimpleName());  
        core.createDataWriter(toMobileNodeTopic);  
    }  
    ...  
}
```

Desenvolvendo aplicações com o ContextNet HelloCoreServer (4/5) – finalizações

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloCoreServer implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    public HelloCoreServer() {  
        ...  
        counter = 0;  
        System.out.println("=== Server Started (Listening) ===");  
        synchronized (this) {  
            try {  
                this.wait();  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
    ...  
}
```

Desenvolvendo aplicações com o ContextNet HelloCoreServer (5/5) – recebendo dados

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloCoreServer implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    @Override  
    public void onNewData(ApplicationObject topicSample) {  
        Message message = (Message) topicSample;  
        System.out.println(Serialization.fromJavaByteStream(message.getContent()));  
  
        PrivateMessage privateMessage = new PrivateMessage();  
        privateMessage.setGatewayId(message.getGatewayId());  
        privateMessage.setNodeId(message.getSenderId());  
  
        synchronized (core) {  
            counter++;  
        }  
  
        ApplicationMessage appMsg = new ApplicationMessage();  
        appMsg.setContentObject(counter);  
        privateMessage.setMessage(Serialization.toProtocolMessage(appMsg));  
  
        core.writeTopic(PrivateMessage.class.getSimpleName(), privateMessage);  
    }  
}
```

Alerta de spoiler: cenas dos próximos capítulos

Decodificando tipos de mensagens

```
public void onNewData(ApplicationObject topicSample) {
    if(topicSample instanceof Message) {
        Message msg = (Message) topicSample;
        String content = new String(msg.getContent());
        JSONParser parser = new JSONParser();
        try {
            JSONObject object = (JSONObject) parser.parse(content);
            String tag = (String) object.get("tag");
            switch (tag) {
                case "SensorData":
                    handleSensorData(object);
                    break;
                case "EventData":
                    final String label = (String) object.get("label");
                    final String data = (String) object.get("data");
                    handleEvent(label, data);
                    break;
                case "ReplyData":
                case "ErrorData":
                    handleMessage(tag, object);
                    break;
            }
        } catch (Exception e) {
            System.out.println("Mensagem sem tag (pode ter sido um keep-alive): " + e.getMessage());
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet

Executando

Tutorial

HelloCore

Precisaremos de três “máquinas” (consoles):

Gateway

- `$ java -jar /opt/ContextNet/contextnet-2.7.jar 127.0.0.1 5500 OpenSplice`

HelloMobile

Nó fixo

- `$ java -jar HelloCoreServer.jar`

HelloGroup

Cliente móvel

- `$ java -jar HelloCoreClient.jar`

HelloAndroid

Desenvolvendo aplicações com o ContextNet

Resultado

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar /opt/ContextNet/contextnet-2.7.jar  
r 127.0.0.1 5500 OpenSplice  
Working Directory = /home/meslin  
Gateway started...  
Gateway MR-UDP IP: 127.0.0.1:5500  
ReliableSocket: new utils pool size = 3  
█
```

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloCoreServer.jar  
=== Server Started (Listening) ===  
Hello World  
█
```

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloCoreClient.jar  
ReliableSocket: new utils pool size = 3  
1  
█
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

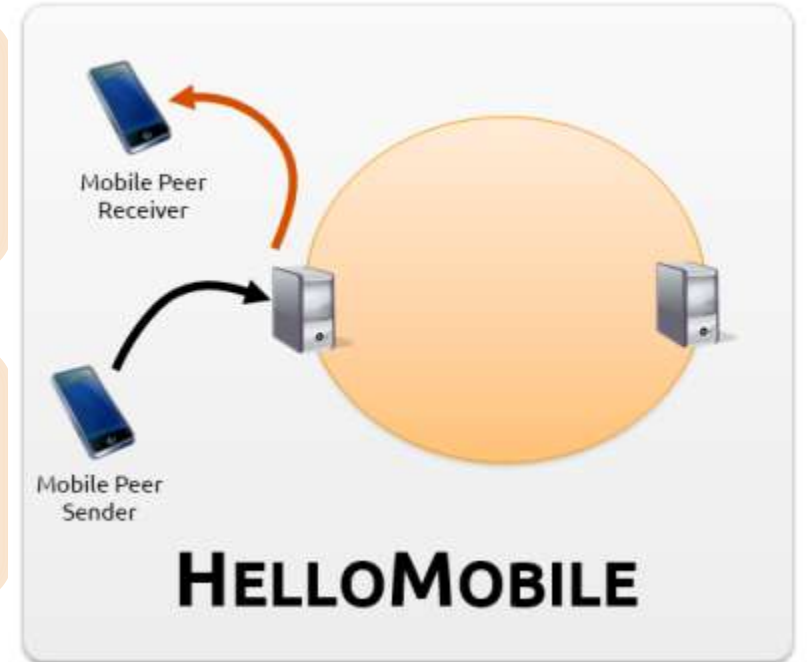
- Cliente móvel transmissor
- Cliente móvel receptor

Classe auxiliar

- CustomData

Objetivo

- Enviar uma mensagem de um cliente móvel para outro cliente móvel



Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

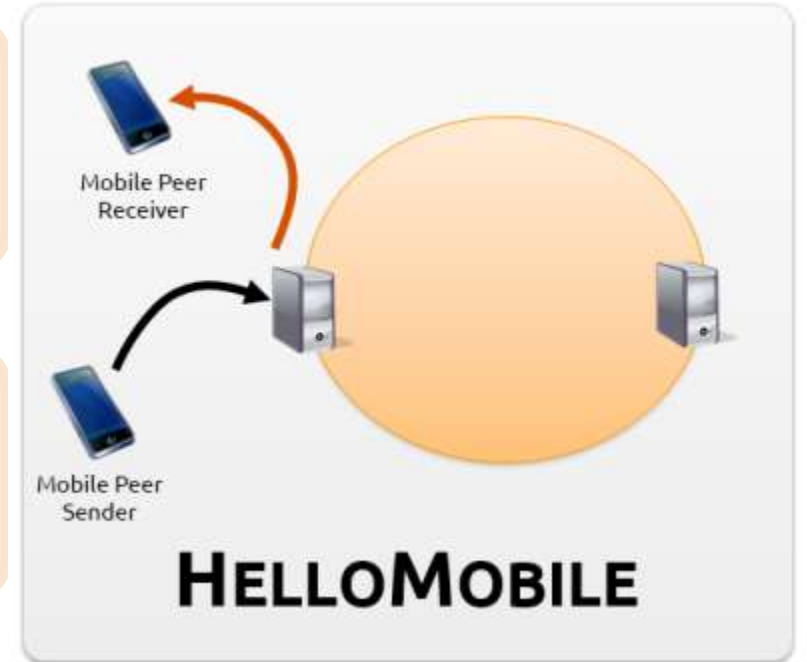
- Cliente móvel transmissor
- Cliente móvel receptor

Classe auxiliar

- CustomData

Objetivo

- Enviar uma mensagem de um cliente móvel para outro cliente móvel



Desenvolvendo aplicações com o ContextNet CustomData (1/2) – pacotes e construtor

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.io.Serializable;
```

```
import javax.swing.ImageIcon;
```

```
public class CustomData implements Serializable {  
    private static final long serialVersionUID = 6093226637618022646L;  
    private String caption;  
    private ImageIcon icon;
```

```
    public CustomData() {  
    }
```

```
    public CustomData(String caption, String icon) {  
        this.caption = caption;
```

```
        this.icon = new ImageIcon(Thread.currentThread().getContextClassLoader().getResource(icon));  
    }  
}
```

Desenvolvendo aplicações com o ContextNet CustomData (2/2) – métodos de acesso

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public String getCaption() { return caption; }

public ImageIcon getIcon() { return icon; }

public void setCaption(String caption) { this.caption = caption; }

public void setIcon(ImageIcon icon) { this.icon = icon; }

@Override
public String toString() {
    String firstPart = "Caption: " + caption;
    String secondPart = "\nIcon Height: " + icon.getIconHeight();
    String thirdPart = "\nIcon Width: " + icon.getIconWidth();

    return firstPart + secondPart + thirdPart;
}
}
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

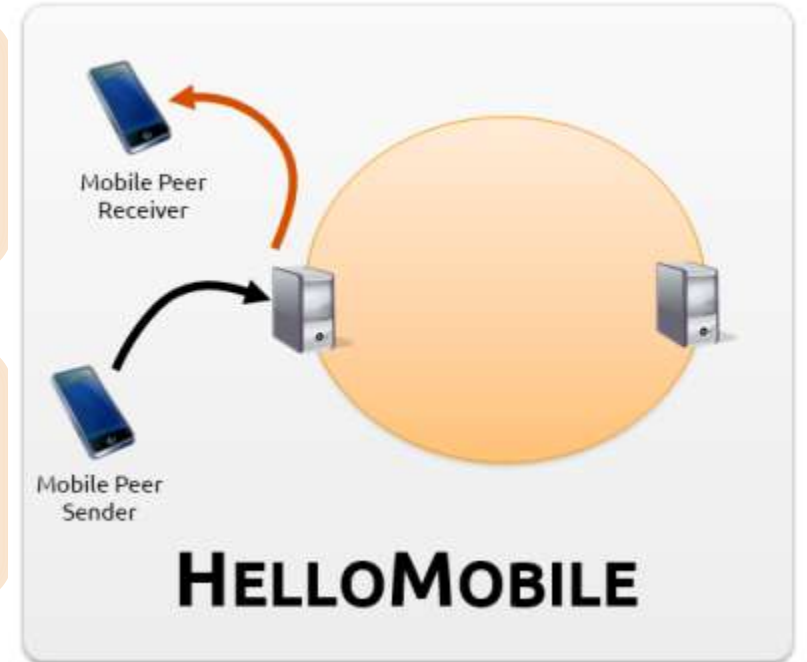
- Cliente móvel transmissor
- Cliente móvel receptor

Classe auxiliar

- CustomData

Objetivo

- Enviar uma mensagem de um cliente móvel para outro cliente móvel



Desenvolvendo aplicações com o ContextNet HelloMobileSender (1/6) – pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.io.IOException;
```

```
import java.net.InetSocketAddress;
```

```
import java.net.SocketAddress;
```

```
import java.util.List;
```

```
import java.util.UUID;
```

```
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
```

```
import lac.cnclib.net.NodeConnection;
```

```
import lac.cnclib.net.NodeConnectionListener;
```

```
import lac.cnclib.net.mrudp.MrUdpNodeConnection;
```

```
import lac.cnclib.sddl.message.ApplicationMessage;
```

```
import lac.cnclib.sddl.message.Message;
```


Desenvolvendo aplicações com o ContextNet HelloMobileSender (2/6) – construtor

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileSender implements NodeConnectionListener {
    private static String gatewayIP = "127.0.0.1";
    private static int gatewayPort = 5500;
    private MrUdpNodeConnection connection;
    private UUID myUUID;

    public HelloMobileSender() {
        myUUID = UUID.fromString("bb103877-8335-444a-be5f-db8d916f6754");
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        try {
            connection = new MrUdpNodeConnection(myUUID);
            connection.addNodeConnectionListener(this);
            connection.connect(address);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet HelloMobileSender (3/6) – main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileSender implements NodeConnectionListener {  
    ...  
    public static void main(String[] args) {  
        Logger.getLogger( "" ).setLevel( Level.ALL );  
  
        HelloMobileSender sender = new HelloMobileSender();  
        sender.sendPicture( "Rio de Janeiro", "/home/meslin/rio.jpg" );  
    }  
    ...  
}
```

Desenvolvendo aplicações com o ContextNet HelloMobileSender (4/6) – método auxiliar

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileSender implements NodeConnectionListener {
...
    public void sendPicture(String caption, String imageName) {
        CustomData serializableContent = new CustomData(caption, imageName);
        ApplicationMessage message = new ApplicationMessage();
        message.setContentObject(serializableContent);
        message.setRecipientID(UUID.fromString("788b2b22-baa6-4c61-b1bb-01cff1f5f878"));

        try {
            System.out.println("Sending image + caption");
            connection.sendMessage(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
...
}
```

Desenvolvendo aplicações com o ContextNet HelloMobileSender (5/6) – mensagens

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileSender implements NodeConnectionListener {
...
    public void connected(NodeConnection remoteCon) {
        ApplicationMessage message = new ApplicationMessage();
        message.setContentObject("Registering");

        try {
            connection.sendMessage(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void newMessageReceived(NodeConnection remoteCon, Message message) {
        System.out.println("Sender received the message!!");
        System.out.println(message.getContentObject());
    }
    ...
}
```

Desenvolvendo aplicações com o ContextNet HelloMobileSender (6/6) – outros métodos

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileSender implements NodeConnectionListener {  
    ...  
    public void reconnected(  
        NodeConnection remoteCon,  
        SocketAddress endPoint,  
        boolean wasHandover,  
        boolean wasMandatory) {}  
    public void disconnected(NodeConnection remoteCon) {}  
    public void unsentMessages(NodeConnection remoteCon, List<Message> unsentMessages) {}  
    public void internalException(NodeConnection remoteCon, Exception e) {}  
}  
...  
}
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

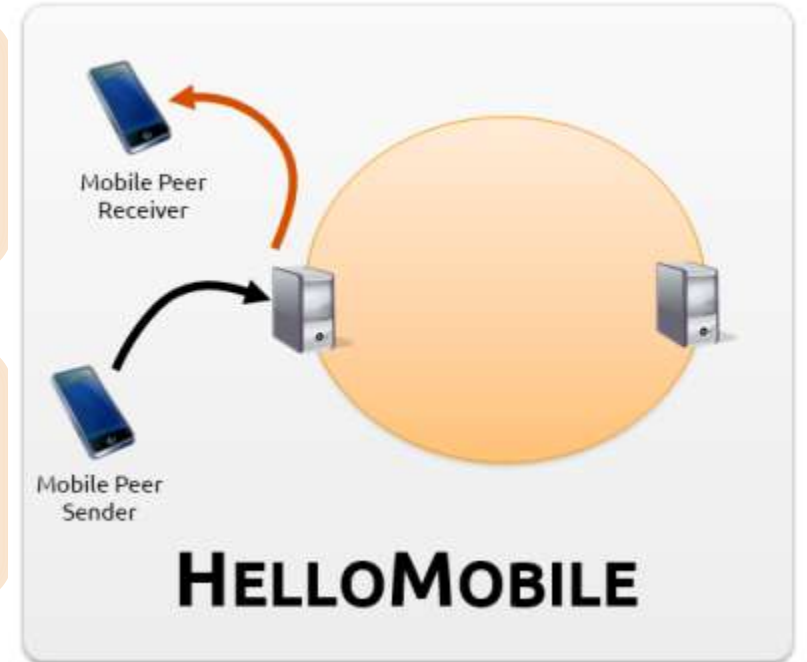
- Cliente móvel transmissor
- Cliente móvel receptor

Classe auxiliar

- CustomData

Objetivo

- Enviar uma mensagem de um cliente móvel para outro cliente móvel



Desenvolvendo aplicações com o ContextNet HelloMobileReceiver (1/4) – pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.io.IOException;  
import java.net.InetSocketAddress;  
import java.net.SocketAddress;  
import java.util.List;  
import java.util.UUID;  
import java.util.logging.Level;  
import java.util.logging.Logger;
```

```
import lac.cnclib.net.NodeConnection;  
import lac.cnclib.net.NodeConnectionListener;  
import lac.cnclib.net.mrudp.MrUdpNodeConnection;  
import lac.cnclib.sddl.message.ApplicationMessage;  
import lac.cnclib.sddl.message.Message;
```


Desenvolvendo aplicações com o ContextNet HelloMobileReceiver (2/4) – construtor e main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileReceiver implements NodeConnectionListener {
    private static String      gatewayIP = "127.0.0.1";
    private static int        gatewayPort = 5500;
    private MrUdpNodeConnection connection;
    private UUID              myUUID;

    public HelloMobileReceiver() {
        myUUID = UUID.fromString("788b2b22-baa6-4c61-b1bb-01cff1f5f878");
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        try {
            connection = new MrUdpNodeConnection(myUUID);
            connection.addNodeConnectionListener(this);
            connection.connect(address);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Logger.getLogger("").setLevel(Level.INFO);
        new HelloMobileReceiver();
    }
}
```

Desenvolvendo aplicações com o ContextNet

HelloMobileReceiver (3/4) – métodos de conexão

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileReceiver implements NodeConnectionListener {
...
    public void connected(NodeConnection remoteCon) {
        ApplicationMessage message = new ApplicationMessage();
        message.setContentObject("Registering");

        try {
            connection.sendMessage(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void reconnected(NodeConnection remoteCon, SocketAddress endPoint, boolean wasHandover,
boolean wasMandatory) {}
    public void disconnected(NodeConnection remoteCon) {}
    public void unsentMessages(NodeConnection remoteCon, List<Message> unsentMessages) {}
    public void internalException(NodeConnection remoteCon, Exception e) {}
}
```

Desenvolvendo aplicações com o ContextNet HelloMobileReceiver (4/4) – recebe mensagem

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloMobileReceiver implements NodeConnectionListener {
...
    public void newMessageReceived(NodeConnection remoteCon, Message message) {
        System.out.println("Receiver Node!");
        System.out.println("Message: " + message.getContentObject());

        ApplicationMessage appMessage = new ApplicationMessage();
        appMessage.setContentObject("Rio de Janeiro continua lindo!");
        appMessage.setRecipientID(message.getSenderID());

        try {
            connection.sendMessage(appMessage);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet

Executando

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Precisaremos de três "máquinas" (consoles):

Gateway

- `$ java -jar /opt/ContextNet/contextnet-2.7.jar 127.0.0.1 5500 OpenSplice`

Cliente que irá receber a mensagem

- `$ java -jar HelloMobileReceiver.jar`

Cliente que irá enviar a mensagem

- `$ java -jar HelloMobileSender.jar`

Desenvolvendo aplicações com o ContextNet

Resultado

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar /opt/ContextNet/contextnet-2.7.jar  
r 127.0.0.1 5500 OpenSplice  
Working Directory = /home/meslin  
Gateway started...  
Gateway MR-UDP IP: 127.0.0.1:5500  
ReliableSocket: new utils pool size = 3  
█
```

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloMobileReceiver.jar  
ReliableSocket: new utils pool size = 3  
Receiver Node!  
Message: Caption: Rio de Janeiro  
Icon Height: 3072  
Icon Width: 2304  
█
```

```
meslin@ubuntu-server-18_04-64bits: ~  
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloMobileSender.jar  
ReliableSocket: new utils pool size = 3  
Sending /home/meslin/rio.jpg  
Sending image + caption  
Sender received the message!!  
Rio de Janeiro continua lindo!  
█
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

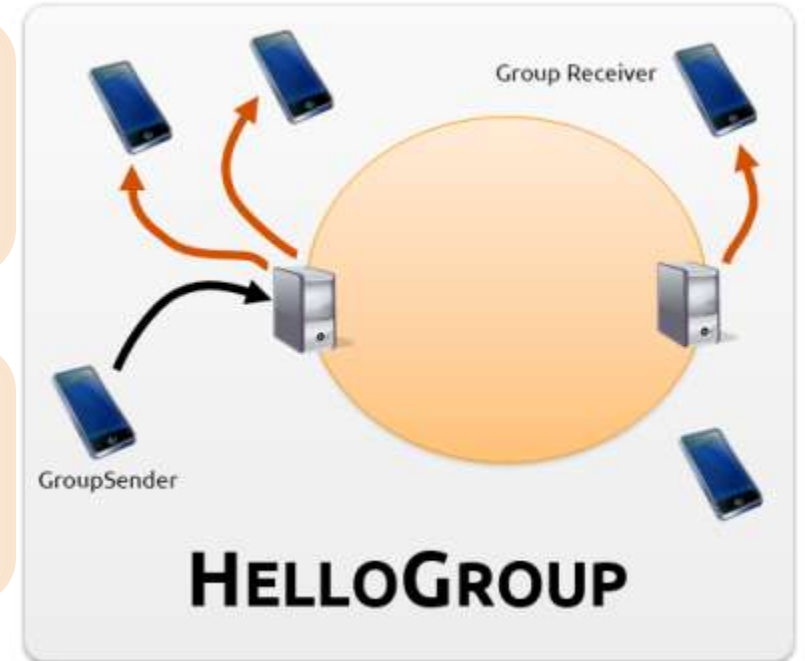
- Cliente móvel transmissor
- Cliente móvel receptor (2)

Classe auxiliar

- GroupDefiner

Objetivo

- Enviar uma mensagem do cliente móvel transmissor para os clientes móveis receptores



Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

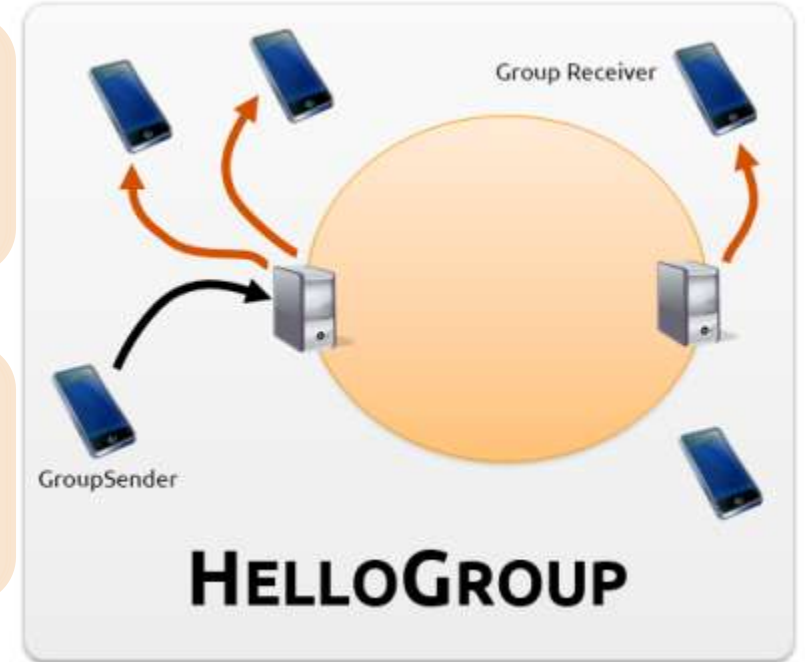
- Cliente móvel transmissor
- Cliente móvel receptor (2)

Classe auxiliar

- **GroupDefiner**

Objetivo

- Enviar uma mensagem do cliente móvel transmissor para os clientes móveis receptores



Desenvolvendo aplicações com o ContextNet HelloGroupDefiner (1/1) – pacotes e main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;

import lac.cnet.groupdefiner.components.GroupDefiner;
import lac.cnet.groupdefiner.components.groupselector.GroupSelector;

public class HelloGroupDefiner {

    public static void main(String[] args) {

        GroupSelector groupSelector = new HelloGroupSelector();
        new GroupDefiner(groupSelector);

        try {
            Thread.sleep(Long.MAX_VALUE);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet HelloGroupSelector (1/2) - pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.util.HashSet;
```

```
import java.util.Set;
```

```
import lac.cnet.groupdefiner.components.groupselector.GroupSelector;
```

```
import lac.cnet.sddl.objects.GroupRegion;
```

```
import lac.cnet.sddl.objects.Message;
```

Desenvolvendo aplicações com o ContextNet

HelloGroupSelector (2/2) – definição de grupos

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupSelector implements GroupSelector {
    public int getGroupType() { return 3; }
    public Set<Integer> processGroups(Message nodeMessage) {
        System.out.println("STARTED CLASSIFYING GROUP MESSAGE");
        // Add the node to two groups
        HashSet<Integer> groupCollection = new HashSet<Integer>(2, 1);
        // Some default group (ID = 1)
        groupCollection.add(1);
        // Even (2) or Odd Group (3)
        if (nodeMessage.getSenderId().getLeastSignificantBits() % 2 == 0) {
            groupCollection.add(2);
        } else {
            groupCollection.add(3);
        }
        System.out.println("ENDED CLASSIFYING GROUP MESSAGE\n");
        return groupCollection;
    }
}
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

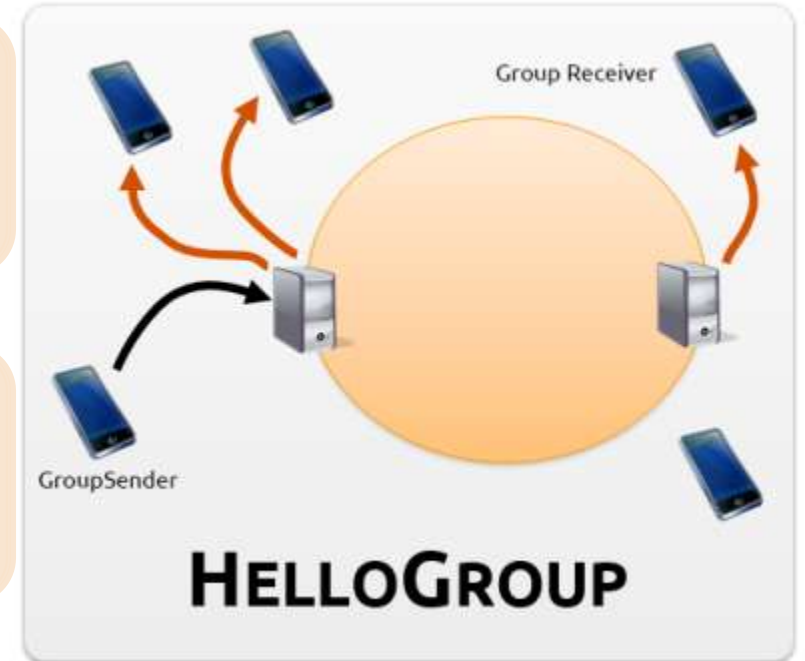
- **Cliente móvel transmissor**
- Cliente móvel receptor (2)

Classe auxiliar

- GroupDefiner

Objetivo

- Enviar uma mensagem do cliente móvel transmissor para os clientes móveis receptores



Desenvolvendo aplicações com o ContextNet HelloGroupClient – pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;
```

```
import java.io.IOException;  
import java.net.InetSocketAddress;  
import java.net.SocketAddress;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;
```

```
import lac.cnclib.net.NodeConnection;  
import lac.cnclib.net.NodeConnectionListener;  
import lac.cnclib.net.groups.Group;  
import lac.cnclib.net.groups.GroupCommunicationManager;  
import lac.cnclib.net.groups.GroupMembershipListener;  
import lac.cnclib.net.mrudp.MrUdpNodeConnection;  
import lac.cnclib.sddl.message.ApplicationMessage;  
import lac.cnclib.sddl.message.Message;
```

Desenvolvendo aplicações com o ContextNet HelloGroupClient – construtor

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupClient implements NodeConnectionListener, GroupMembershipListener {
    private final static String gatewayIP = "172.0.0.1";
    private final static int gatewayPort = 5500;
    private GroupCommunicationManager groupManager;

    public HelloGroupClient() {
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        try {
            MrUdpNodeConnection connection = new MrUdpNodeConnection();
            connection.connect(address);
            connection.addNodeConnectionListener(this);
            try {
                Thread.sleep(30000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet

HelloGroupClient – main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupClient implements NodeConnectionListener, GroupMembershipListener {  
    ...  
    public static void main(String[] args) {  
        System.err.println("main");  
        Logger.getLogger("").setLevel(Level.OFF);  
  
        new HelloGroupClient();  
    }  
    ...  
}
```


Desenvolvendo aplicações com o ContextNet HelloGroupClient – envio de mensagem

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupClient implements NodeConnectionListener, GroupMembershipListener {
...
    @Override
    public void connected(NodeConnection remoteCon) {
        groupManager = new GroupCommunicationManager(remoteCon);
        groupManager.addMembershipListener(this);
        try {
            // i = 1 Default
            // i = 2 EVEN
            // i = 3 ODD
            for (int i = 1; i < 4; i++) {
                Group group = new Group(3, i);
                ApplicationMessage appMsg = new ApplicationMessage();
                System.err.println("Message GroupID: " + i);
                appMsg.setContentObject("Message GroupID: " + i);
                groupManager.sendGroupcastMessage(appMsg, group);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet HelloGroupClient – outros métodos

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupClient implements NodeConnectionListener, GroupMembershipListener {
    ...
    @Override
    public void reconnected(NodeConnection remoteCon, SocketAddress endPoint, boolean
wasHandover, boolean wasMandatory) {}
    @Override
    public void disconnected(NodeConnection remoteCon) {}
    @Override
    public void newMessageReceived(NodeConnection remoteCon, Message message) {
        System.out.println(message.getContentObject());
    }
    @Override
    public void unsentMessages(NodeConnection remoteCon, List<Message> unsentMessages) {}
    @Override
    public void internalException(NodeConnection remoteCon, Exception e) {}
    @Override
    public void enteringGroups(List<Group> groups) {}
    @Override
    public void leavingGroups(List<Group> groups) {}
}
```

Desenvolvendo aplicações com o ContextNet

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Dois componentes:

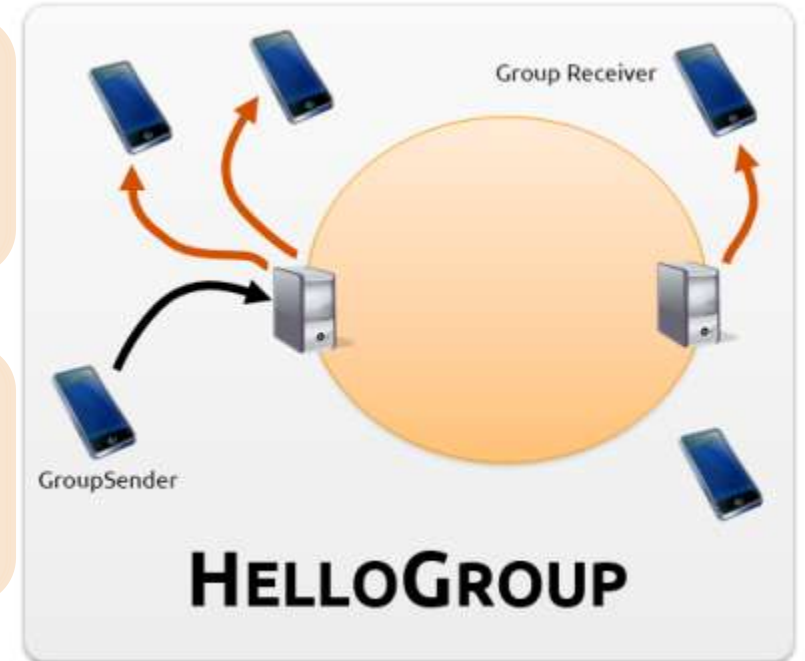
- Cliente móvel transmissor
- **Cliente móvel receptor (2)**

Classe auxiliar

- GroupDefiner

Objetivo

- Enviar uma mensagem do cliente móvel transmissor para os clientes móveis receptores



Desenvolvendo aplicações com o ContextNet HelloGroupDefinerEvenReceiver – pacotes

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
package br.com.meslin.lac;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.SocketAddress;
import java.util.List;
import java.util.UUID;
import java.util.logging.Level;
import java.util.logging.Logger;

import lac.cnclib.net.NodeConnection;
import lac.cnclib.net.NodeConnectionListener;
import lac.cnclib.net.groups.Group;
import lac.cnclib.net.groups.GroupCommunicationManager;
import lac.cnclib.net.groups.GroupMembershipListener;
import lac.cnclib.net.mrudp.MrUdpNodeConnection;
import lac.cnclib.sddl.message.ApplicationMessage;
import lac.cnclib.sddl.message.Message;
```

Desenvolvendo aplicações com o ContextNet HelloGroupDefinerEvenReceiver – construtor

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupDefinerEvenReceiver implements NodeConnectionListener,
GroupMembershipListener {
    private final static String      gatewayIP      = "127.0.0.1";
    private final static int         gatewayPort    = 5500;
    private GroupCommunicationManager groupManager;

    public HelloGroupDefinerEvenReceiver() {
        UUID uuid = UUID.randomUUID();
        while (uuid.getLeastSignificantBits() % 2 != 0) {
            uuid = UUID.randomUUID();
        }
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        try {
            MrUdpNodeConnection connection = new MrUdpNodeConnection(uuid);
            connection.connect(address);
            connection.addNodeConnectionListener(this);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet HelloGroupDefinerEvenReceiver – main

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
public class HelloGroupDefinerEvenReceiver implements NodeConnectionListener,
GroupMembershipListener {
...
    public static void main(String[] args) {
        Logger.getLogger("").setLevel(Level.OFF);
        new HelloGroupDefinerEvenReceiver();
    }

    @Override
    public void connected(NodeConnection remoteCon) {
        groupManager = new GroupCommunicationManager(remoteCon);
        groupManager.addMembershipListener(this);

        try {
            ApplicationMessage message = new ApplicationMessage();
            String serializableContent = "Bogus Message for Group Definer";
            message.setContentObject(serializableContent);

            remoteCon.sendMessage(message);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


Desenvolvendo aplicações com o ContextNet HelloGroupDefinerEvenReceiver – outros...

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
@Override
    public void reconnected(NodeConnection remoteCon, SocketAddress endPoint, boolean
wasHandover, boolean wasMandatory) {}
    @Override
    public void disconnected(NodeConnection remoteCon) {}
    @Override
    public void newMessageReceived(NodeConnection remoteCon, Message message) {
        System.out.println(message.getContentObject());
    }
    @Override
    public void unsentMessages(NodeConnection remoteCon, List<Message> unsentMessages) {}
    @Override
    public void internalException(NodeConnection remoteCon, Exception e) {}
    @Override
    public void enteringGroups(List<Group> groups) {}
    @Override
    public void leavingGroups(List<Group> groups) {}
}
```


Desenvolvendo aplicações com o ContextNet HelloGroupDefinerOddReceiver

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

- Identico ao HelloGroupDefinerEvenReceiver
- Modificar a linha:
while (uuid.getLeastSignificantBits() % 2 != 0) {
- Para:
while (uuid.getLeastSignificantBits() % 2 == 0) {

Desenvolvendo aplicações com o ContextNet

Executando

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

Precisaremos de cinco "máquinas" (consoles):

Gateway

- `$ java -jar /opt/ContextNet/contextnet-2.7.jar 127.0.0.1 5500 OpenSplice`

Group Definer

- `$ java -jar HelloGroupDefiner.jar`

Clientes que irão receber a mensagem

- `$ java -jar HelloGroupDefinerEvenReceiver.jar`
- `$ java -jar HelloGroupDefinerOddReceiver.jar`

Cliente que irá enviar a mensagem

- `$ java -jar HelloGroupClient.jar`

Desenvolvendo aplicações com o ContextNet

Resultado

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

```
Terminal File Edit View Search Terminal Help
meslin@ubuntu-server-18_04-64bits:~$ java -jar /opt/ContextNet/contextnet-2.7.jar
r 172.16.10.128 5500 OpenSplice
Working Directory = /home/meslin
Gateway started...
```

```
meslin@ubuntu-server-18_04-64bits:~$
```

```
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloGroupDefiner.jar
STARTED CLASSIFYING GROUP MESSAGE
Sender id: -6204754154168151913
ENDED CLASSIFYING GROUP MESSAGE
```

```
STARTED CLASSIFYING GROUP MESSAGE
Sender id: -7900936872813709582
ENDED CLASSIFYING GROUP MESSAGE
```

```
meslin@ubuntu-server-18_04-64bits:~$
```

```
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloGroupDefinerOddReceiver.jar
ReliableSocket: new utils pool size = 3
Message GroupID: 3
Message GroupID: 1
```

```
meslin@ubuntu-server-18_04-64bits:~$
```

```
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloGroupDefinerEvenReceiver.jar
ReliableSocket: new utils pool size = 3
Message GroupID: 1
Message GroupID: 2
```

```
meslin@ubuntu-server-18_04-64bits:~$
```

```
meslin@ubuntu-server-18_04-64bits:~$ java -jar HelloGroupClient.jar
ReliableSocket: new utils pool size = 3
```

Desenvolvendo aplicações com o ContextNet Versão Android

Tutorial

HelloCore

HelloMobile

HelloGroup

HelloAndroid

- Não dessa vez! Programação Android não é a minha praia e não deu tempo de preparar o material, mas tem um excelente tutorial em:
<http://wiki.lac.inf.puc-rio.br/doku.php?id=helloandroid>
- Dispositivos móveis reais em:
<https://aws.amazon.com/device-farm/>



Desenvolvendo aplicações com o ContextNet

O que é Complex Event Processing?

- Um Paradigma e tecnologia para analisar e manipular **fluxos de eventos** (data stream) em consultas contínuas (on-line)
- CEP inclui:
 - Padrões que verificam relações causais e temporais entre eventos de diferentes tipos
 - Manipulação e geração de eventos complexos, derivados de eventos mais simples
 - Vários tipos de janelas deslizantes para análise contínua e batch de eventos (Ex. win(5 events) ou win(1 sec))
 - Event Processing Network (EPN) define a lógica completa do processamento em vários passos

Desenvolvendo aplicações com o ContextNet Esper Engine

- **Esper** é uma engine CEP open-source:
 - Regras definidas em linguagem SQL-like
 - Regra consiste de Filtros, Maps, Padrões, Janelas e Listeners
 - Para Java e C#

Desenvolvendo aplicações com o ContextNet

Implementando CEP – pacotes

```
package br.com.meslin.cep;  
  
import java.util.HashSet;  
  
import br.com.meslin.onibus.aux.event.EventBusArrivingListener;  
import br.com.meslin.onibus.aux.event.EventRegion;  
import br.com.meslin.onibus.aux.model.Bus;  
  
import com.espertech.esper.client.Configuration;  
import com.espertech.esper.client.EPServiceProvider;  
import com.espertech.esper.client.EPServiceProviderManager;  
import com.espertech.esper.client.EPStatement;
```


Desenvolvendo aplicações com o ContextNet

Implementando CEP – métodos

```
public class MyCEP {
    ...
    public MyCEP() {
        // configuration
        Configuration config = new Configuration();
        config.addEventTypeAutoName("br.com.meslin.onibus.aux.event");
        config.addEventType("EventRegion", EventRegion.class.getName());
        // creating a statement
        epService = EPServiceProviderManager.getProvider("myCEPEngine", config);
        String expression = "select * from EventRegion having EventRegion.region = 1";
        EPStatement statement = epService.getEPAdministrator().createEPL(expression);
        statement.addListener(new EventBusArrivingListener());
    }

    public void createEvent(Bus bus, HashSet<Integer> newGroups) {
        for(int group : newGroups) {
            if(!bus.getGroups().contains(group)) {
                EventRegion eventRegion = new EventRegion(bus, group, System.currentTimeMillis());
                epService.getEPRuntime().sendEvent(eventRegion);
            }
        }
    }
}
```

Desenvolvendo aplicações com o ContextNet

Implementando CEP – classe de evento

```
package br.com.meslin.onibus.aux.event;

import java.util.Date;

import br.com.meslin.onibus.aux.model.Bus;

public class EventRegion {
    private Bus bus;
    private int region;
    private Date timeStamp;

    public EventRegion(Bus bus, int region, long time) {
        this.bus = bus;
        this.region = region;
        this.timeStamp = new Date(time);
    }
}
```

Desenvolvendo aplicações com o ContextNet

Implementando CEP – listener

```
package br.com.meslin.onibus.aux.event;  
  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;  
  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import br.com.meslin.onibus.aux.Debug;  
import br.com.meslin.onibus.aux.StaticLibrary;  
import br.com.meslin.onibus.aux.contextnet.MessageSender;  
import br.com.meslin.onibus.aux.model.Bus;  
  
import com.espertech.esper.client.EventBean;  
import com.espertech.esper.client.PropertyAccessException;  
import com.espertech.esper.client.UpdateListener;
```

Desenvolvendo aplicações com o ContextNet

Implementando CEP – listener

```
public class EventBusArrivingListener implements UpdateListener {
    private Map<Integer, Integer> nextRegion;

    public EventBusArrivingListener() {
        nextRegion = new HashMap<Integer, Integer>();
        nextRegion.put(1, 10002);
        nextRegion.put(2, 10001);
    }

    @Override
    public void update(EventBean[] newEvents, EventBean[] oldEvents) {
        try {
            MessageSender messageSender = MessageSender.getInstance();
            String linha = ((Bus)newEvents[0].get("bus")).getLinha();
            String ordem = ((Bus)newEvents[0].get("bus")).getOrdem();
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("message", "ATENÇÃO!!! Seu ônibus " + linha + "(" + ordem + ") está chegando");
            messageSender.sendMessageToGroup(targetRegion, jsonObject);
        } catch (PropertyAccessException | JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Perguntas?



Exercícios

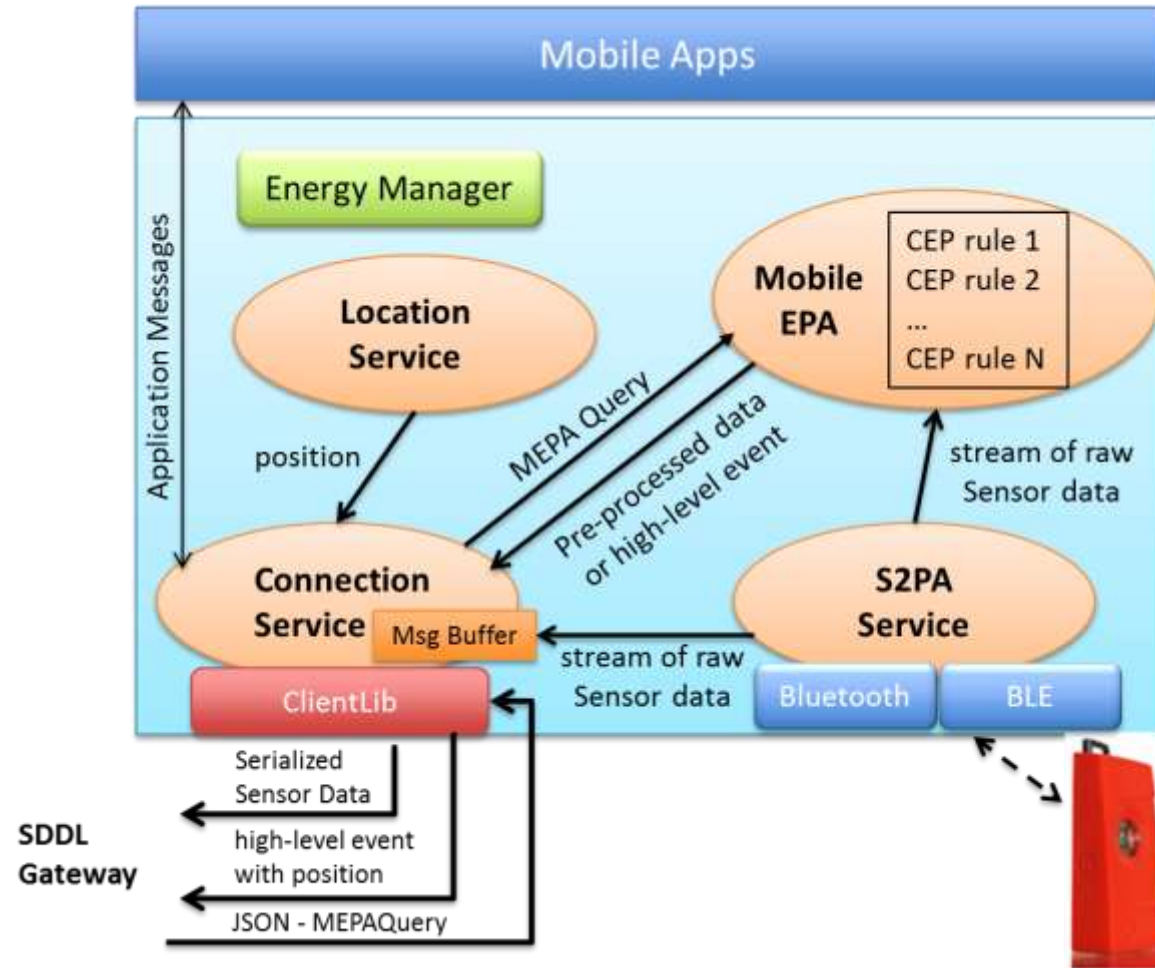
Exercício

- Faça um sistema de mensagens para grupos
 - Crie os grupos baseados em tipos de ocupação funcional do usuário

Utilizando o Mobile-Hub para captura de dados

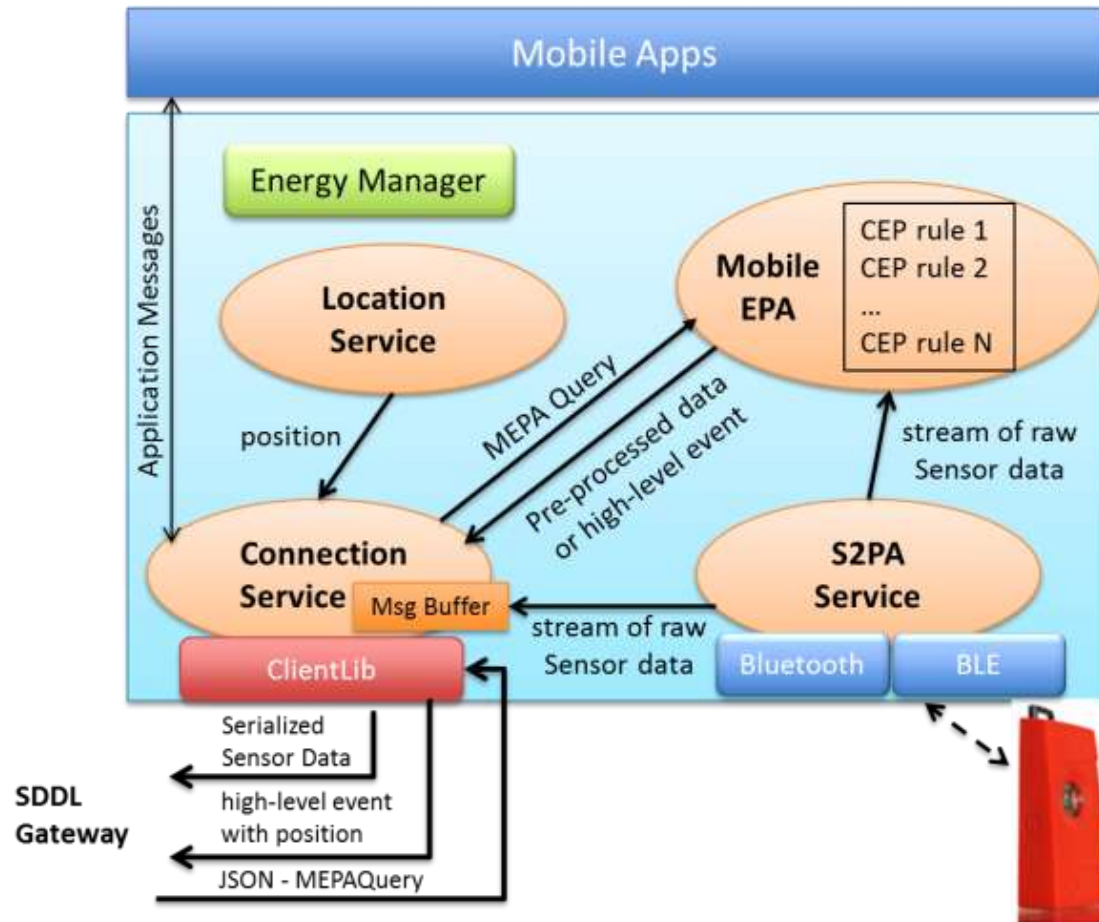
Transmitindo e recebendo dados para/de o ContextNet

Mobile-Hub Arquitetura



Mobile-Hub

Arquitetura



S2PA (Short-range Sensor, Presence, and Actuation): descoberta e monitoramento de M-OBJs

Connection Service: gerência de mensagens

Location Service: obtenção da localização através de GPS, rede ou manual

Energy Manager: monitor de bateria

MEPA (Mobile Event Processing Agente) Service: CEP

Mobile-Hub

Requerimentos

The M-Hub application (apk) requires:

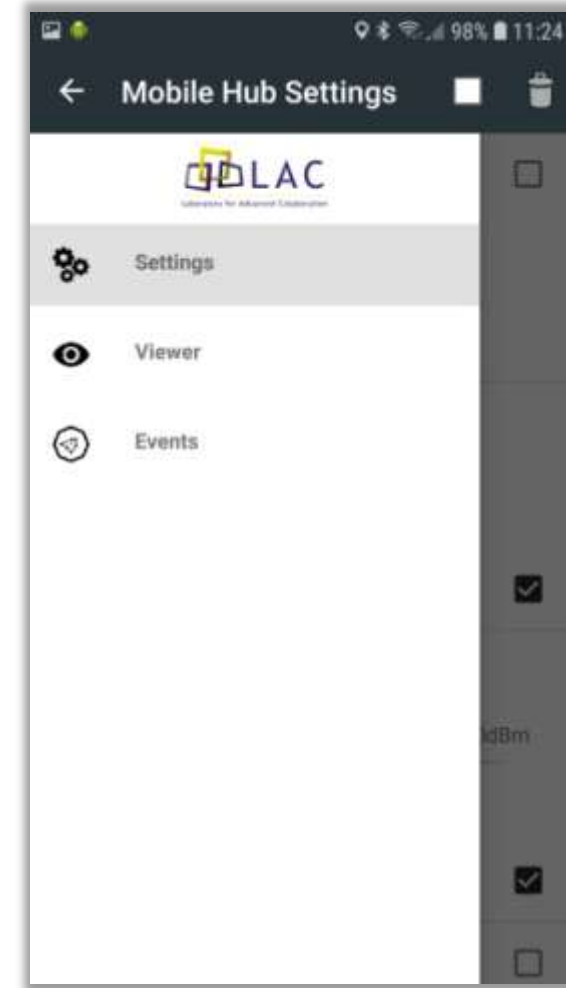
- Android ≥ 4.3
- a Gateway.

The current implementation has support for Bluetooth Low Energy as a WPAN technology

On the settings activity of the M-Hub, you can uncheck some services/manager

Mobile-Hub

Configurações

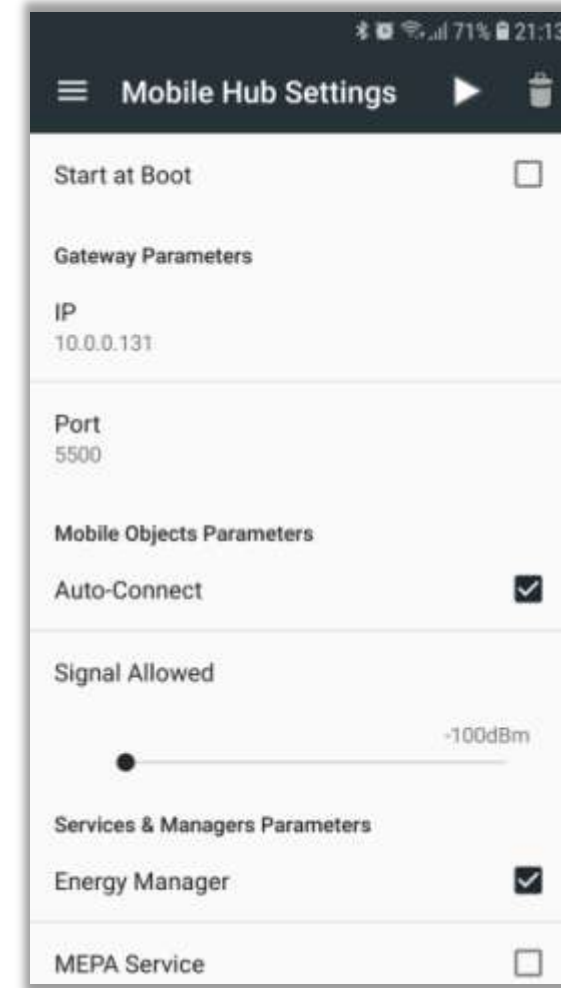


Mobile-Hub

Configurações

Gateway

- Endereço IP
- Porta

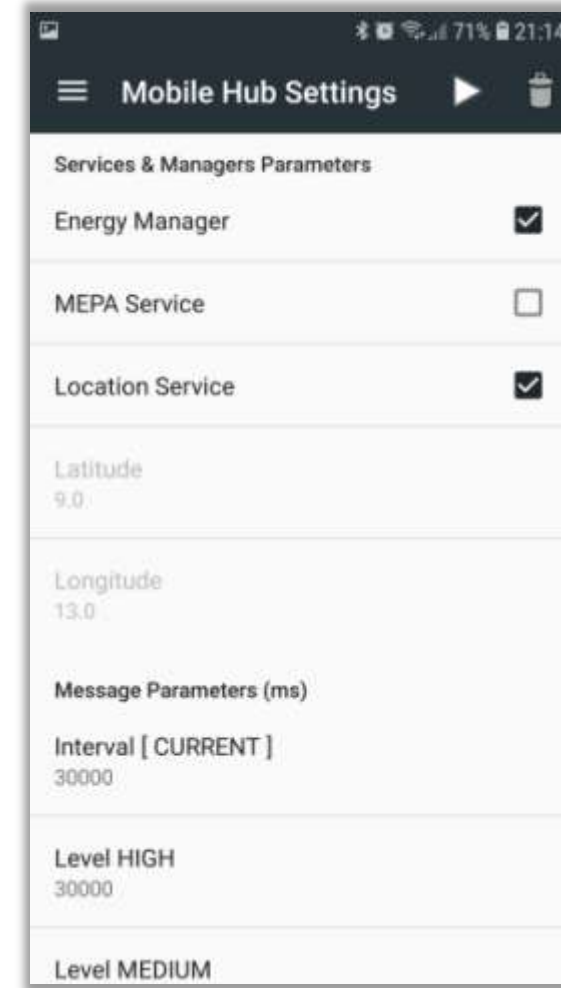


Mobile-Hub

Configurações

Parâmetros

- Gerenciamento
- Serviços

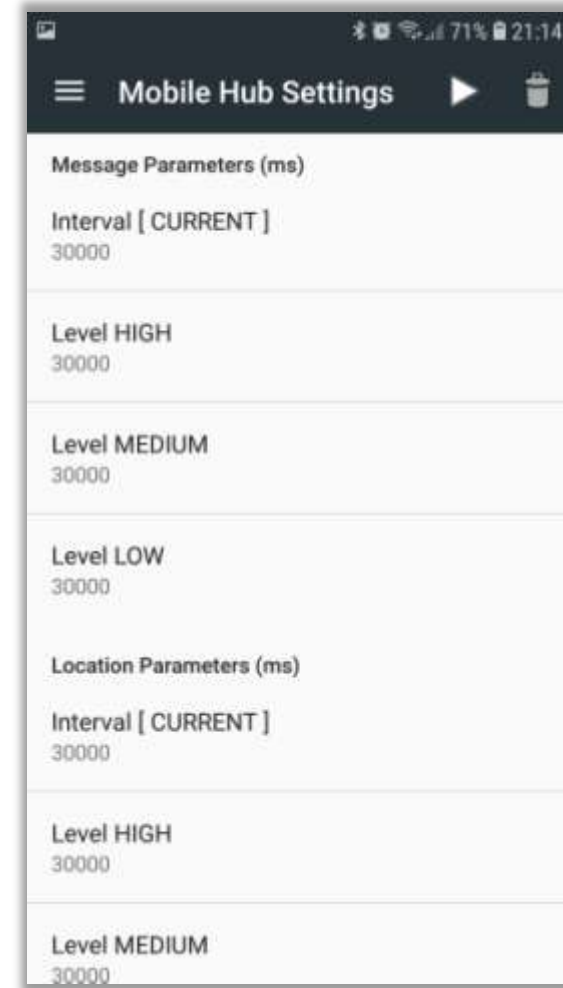


Mobile-Hub

Configurações

Intervalo de mensagens de acordo com o nível de bateria

- Atual
- Nível alto
- Nível médio
- Nível baixo

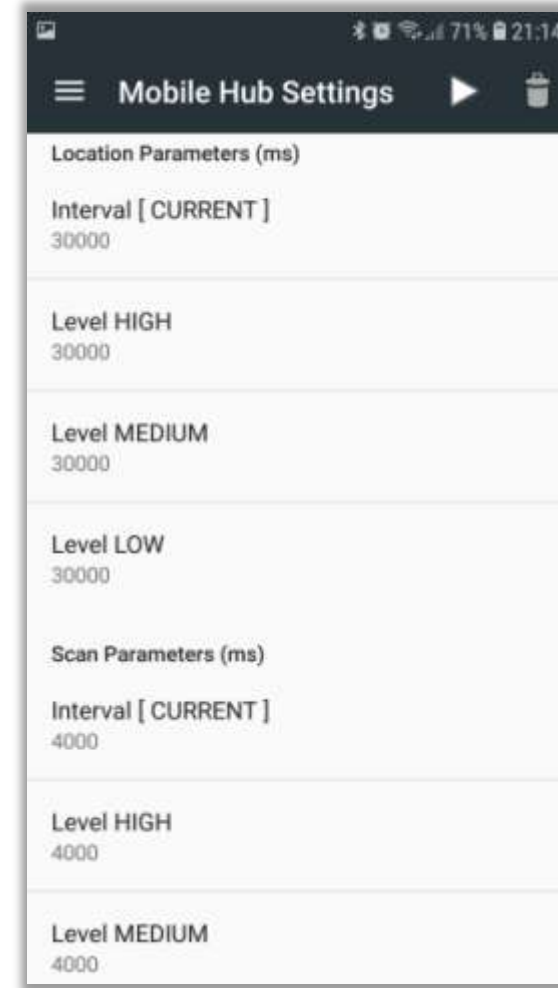


Mobile-Hub

Configurações

Intervalo de mensagens de localização de acordo com o nível de bateria

- Atual
- Nível alto
- Nível médio
- Nível baixo

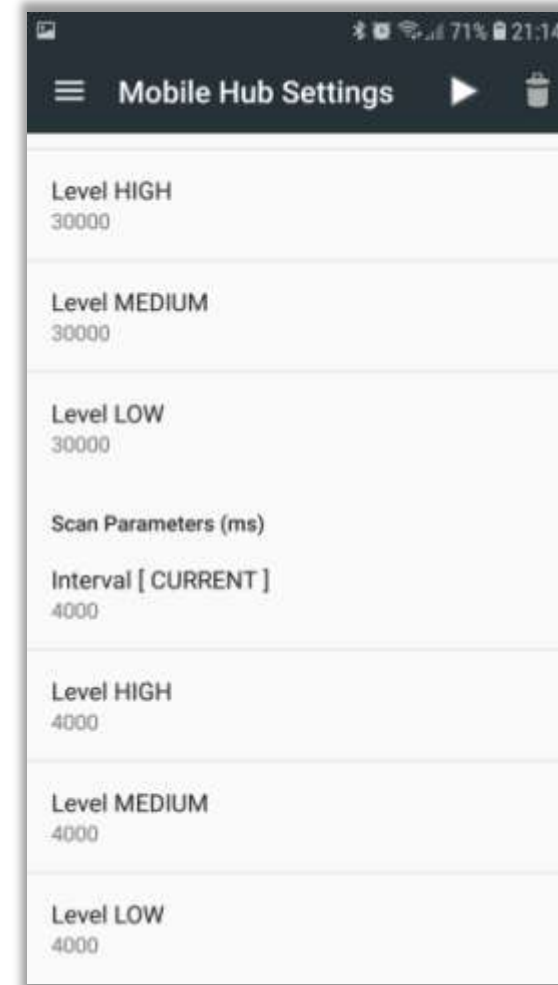


Mobile-Hub

Configurações

Intervalo de procura por dispositivos de acordo com o nível de bateria

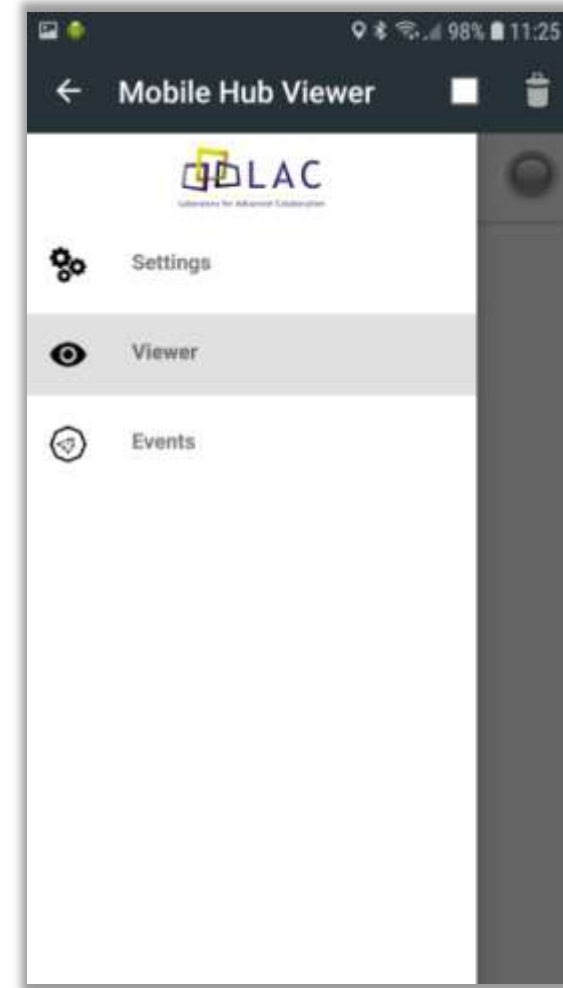
- Atual
- Nível alto
- Nível médio
- Nível baixo



Mobile-Hub

Visualização

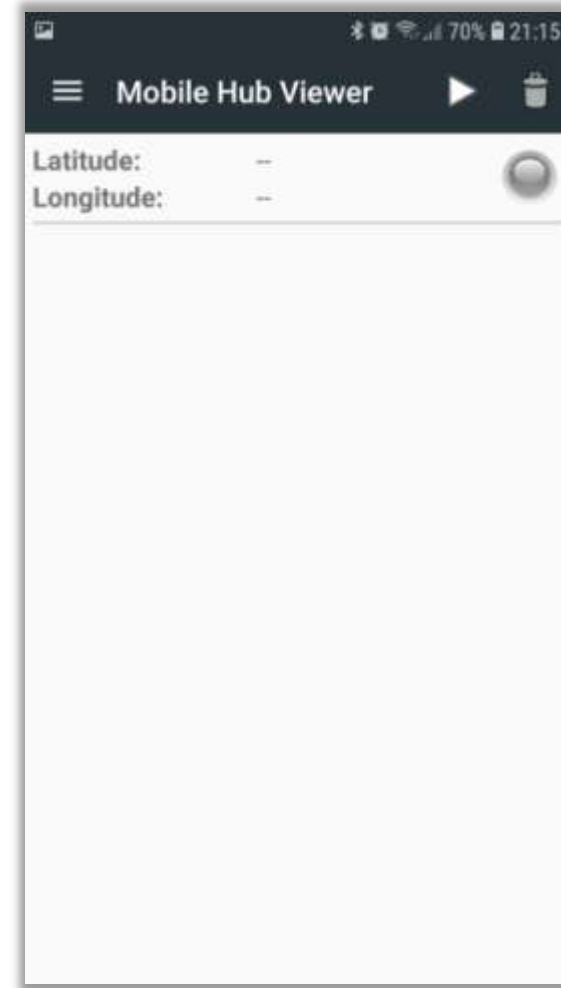
- Dispositivos conectados



Mobile-Hub

Visualização

Nenhum dispositivo
conectado

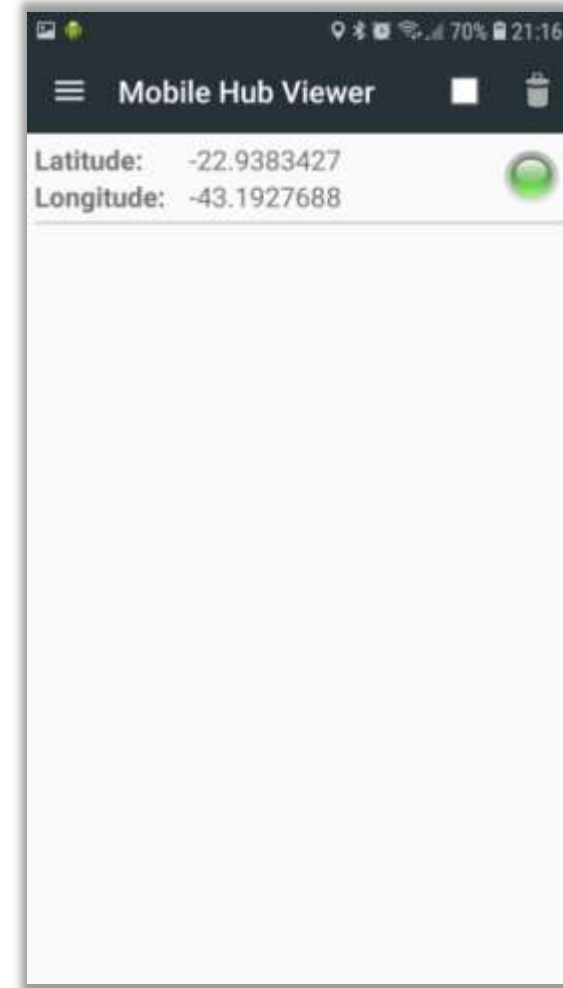


Mobile-Hub

Visualização

Transmitindo posição

- Latitude: -22,9383427
- Longitude: -43,1927688



Mobile-Hub

Visualização

Conectado a um sensor tag

- Acelerômetro
- Barômetro
- Giroscópio
- Umidade
- Magnetômetro
- Temperatura

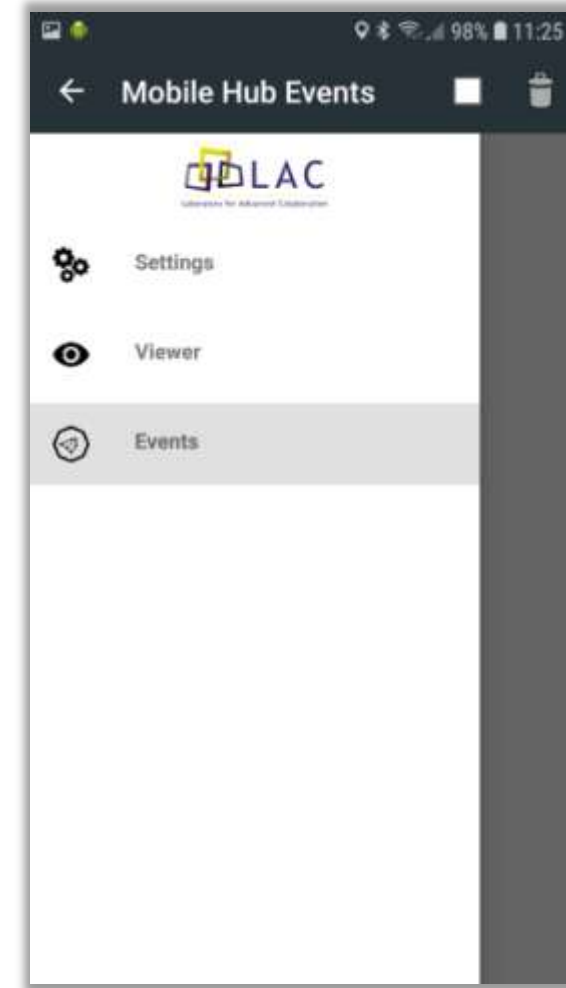


The screenshot shows the 'Mobile Hub Viewer' application interface. At the top, there is a status bar with icons for location, Wi-Fi, cellular signal, 100% battery, and the time 11:20. Below the status bar is a dark header with a hamburger menu icon, the text 'Mobile Hub Viewer', a square icon, and a trash icon. The main content area displays the following information:

Latitude:	-22.9383297	
Longitude:	-43.192775	
<hr/>		
	1-78A5048CEA0A	
	RSSI: -63.0	
Accelerometer:	0.00 -0.97 -0.09	
Barometer:	101511.72	
Gyroscope:	1.56 -2.68 0.13	
Humidity:	79.24	
Magnetometer:	-7.14 -10.56 28.96	
Temperature:	22.62 20.25	

Mobile-Hub

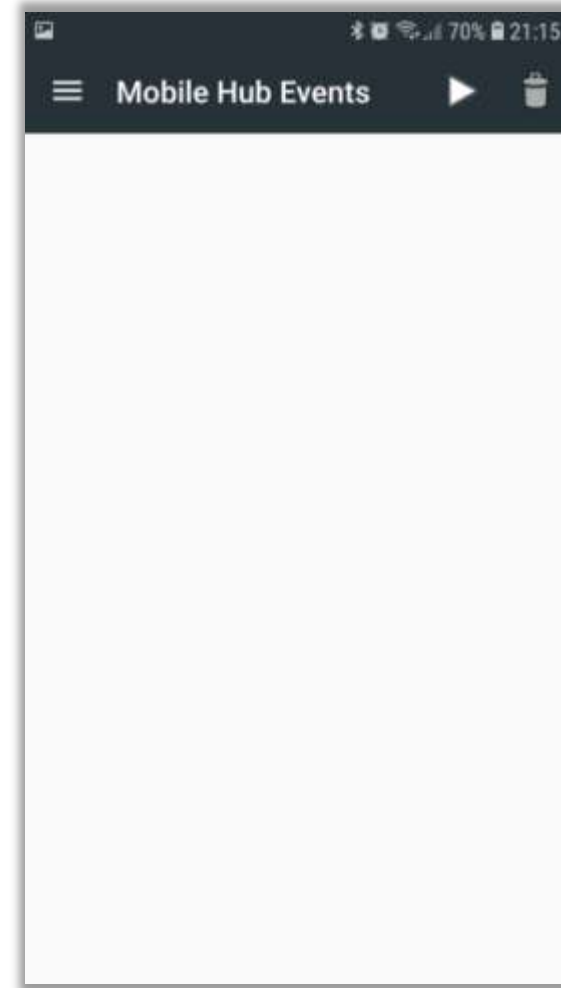
Visualização de eventos



Mobile-Hub

Eventos

Nenhum evento a ser exibido

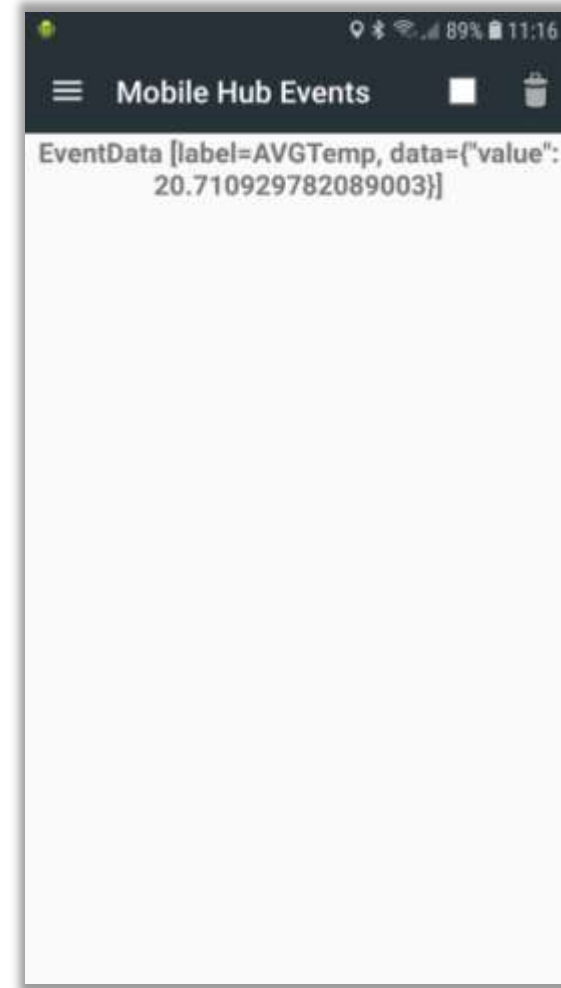


Mobile-Hub

Eventos

Evento de temperatura

- Nome: AVGTemp
- Dado: 20,710929782089003



Mobile-Hub

Mensagens

CONEXÃO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-2CB6E5B2DF07",  
  "action":"found",  
  "signal":-80,  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288348  
}
```

LOCALIZAÇÃO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "date":"Tue Jul 16 11:45:23 GMT-03:00 2019",  
  "accuracy":19.6560001373291,  
  "provider":"network",  
  "speed":0,  
  "bearing":0,  
  "altitude":46.89999771118164,  
  "connection":"java.lang.InfoConnectivityWifi",  
  "battery":96,  
  "charging":false,  
  "tag":"LocationData",  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "timestamp":1563288323  
}
```

Mobile-Hub

Mensagens

CONEXÃO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-2CB6E5B2DF07",  
  "action":"found",  
  "signal":-80,  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288348  
}
```

M-Obj UUID

LOCALIZAÇÃO

```
{  
  "date":"Tue Jul 16 11:45:23 GMT-03:00 2019",  
  "accuracy":19.6560001373291,  
  "provider":"network",  
  "speed":0,  
  "bearing":0,  
  "altitude":46.89999771118164,  
  "connection":"java.lang.InfoConnectivityWifi",  
  "battery":96,  
  "charging":false,  
  "tag":"LocationData",  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "timestamp":1563288323  
}
```

M-Hub UUID

Mobile-Hub

Mensagens

BARÔMETRO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Barometer",  
  "sensor_value":[101508.01809210484],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288350  
}
```

UMIDADE

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Humidity",  
  "sensor_value":[69.40245056152344],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288351  
}
```

Mobile-Hub

Mensagens

GIROSCÓPIO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Gyroscope",  
  "sensor_value":[1.29699707,-2.357482910,0.0915527],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288351  
}
```

TEMPERATURA

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Temperature",  
  "sensor_value":[26.34375,22.941261956570088],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288351  
}
```


Mobile-Hub

Mensagens

ACELERÔMETRO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Accelerometer",  
  "sensor_value":[0,-0.984375,-0.09375],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288351  
}
```

MAGNETÔMETRO

```
{  
  "uuid":"9509494b-b270-4cd7-a5a2-08cc6bb998d1",  
  "source":"1-78A5048CEA0A",  
  "action":"read",  
  "signal":-53,  
  "sensor_name":"Magnetometer",  
  "sensor_value":[-8.39233398,-8.1176757,29.4799804],  
  "latitude":-22.9383307,  
  "longitude":-43.1927734,  
  "tag":"SensorData",  
  "timestamp":1563288350  
}
```

Mobile-Hub

Mensagens

EVENTO

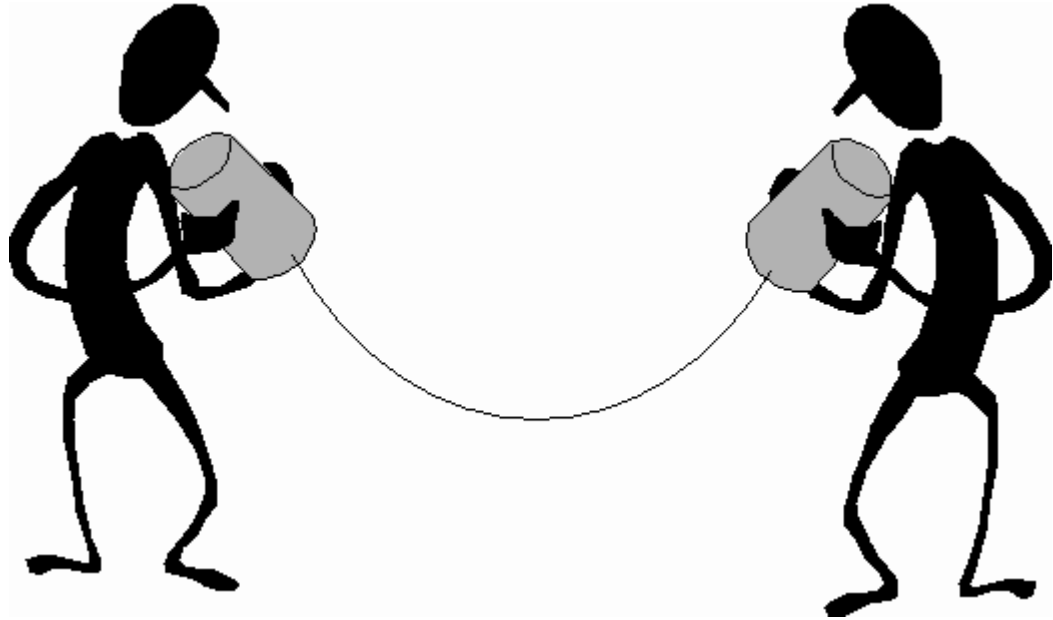
```
{  
  "tag": "EventData",  
  "uuid": "b06de58d-6a20-44f9-8cd4-83f074c2edd6",  
  "label": "AVGTemp",  
  "data": {"value": 22.30},  
  "latitude": -22.98137128,  
  "longitude": -43.23421961,  
  "timestamp": 1442169467  
}
```

STATUS

```
{  
  "tag": "ErrorData|ReplyData",  
  "uuid": "b06de58d-6a20-44f9-8cd4-83f074c2edd6",  
  "component": "MEPASservice|S2PASservice",  
  "message": "Action not supported.",  
  "latitude": -22.98137128,  
  "longitude": -43.23421961,  
  "timestamp": 1442169467  
}
```

Mobile-Hub

Recebendo mensagens



Tipo de Mensagem	Nome da Tag
Dados	SensorData
Eventos	EventData
Respostas	ReplyData
Erros	ErrorData

Mobile-Hub

Recebendo mensagens (1/6)

```
package br.com.meslin.lac;

import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

import lac.cnet.sddl.objects.ApplicationObject;
import lac.cnet.sddl.objects.Message;
import lac.cnet.sddl.objects.PrivateMessage;
import lac.cnet.sddl.udi.core.SddlLayer;
import lac.cnet.sddl.udi.core.UniversalDDSLayerFactory;
import lac.cnet.sddl.udi.core.listener.UIDDataReaderListener;
```

Mobile-Hub

Recebendo mensagens (2/6)

```
public class MyProcessingNode implements UDIDataReaderListener<ApplicationObject> {
    private SddlLayer core;
    public MyProcessingNode() {
        core = UniversalDDSLayerFactory.getInstance();
        core.createParticipant(UniversalDDSLayerFactory.CNET_DOMAIN);
        core.createPublisher();
        core.createSubscriber();
        Object receiveMessageTopic = core.createTopic(Message.class, Message.class.getSimpleName());
        core.createDataReader(this, receiveMessageTopic);
        Object toMobileNodeTopic = core.createTopic(PrivateMessage.class, PrivateMessage.class.getSimpleName());
        core.createDataWriter(toMobileNodeTopic);
        System.out.println("=== Server Started (Listening) ===");
        synchronized (this) {
            try {
                this.wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    ...
}
```

Mobile-Hub

Recebendo mensagens (3/6)

```
public class MyProcessingNode implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    public static void main(String[] args) {  
        new MyProcessingNode();  
        while (true);  
    }  
}
```

Mobile-Hub

Recebendo mensagens (4/6)

```
public class MyProcessingNode implements UDIDataReaderListener<ApplicationObject> {
    ...
    @Override
    public void onNewData(ApplicationObject topicSample) {
        if(topicSample instanceof Message) {
            Message msg = (Message) topicSample;
            String content = new String(msg.getContent());
            JSONParser parser = new JSONParser();

            try {
                JSONObject object = (JSONObject) parser.parse(content);
                String tag = (String) object.get("tag");

                switch (tag) {
                    ...
                }
            } catch (Exception e) {
                System.out.println("Mensagem sem tag (pode ter sido um keep-alive): " + e.getMessage());
            }
        }
    }
}
```


Mobile-Hub

Recebendo mensagens (5/6)

```
public class MyProcessingNode implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    switch (tag) {  
        case "SensorData":  
            handleSensorData(object);  
            break;  
  
        case "EventData":  
            final String label = (String) object.get("label");  
            final String data = (String) object.get("data");  
            handleEvent(label, data);  
            break;  
  
        case "ReplyData":  
        case "ErrorData":  
            handleMessage(tag, object);  
            break;  
    }  
    ...  
}
```

Mobile-Hub

Recebendo mensagens (6/6)

```
public class MyProcessingNode implements UDIDataReaderListener<ApplicationObject> {  
    ...  
    private void handleMessage(String tag, JSONObject object) {  
        System.out.println("handleMessage: " + tag);  
    }  
  
    private void handleEvent(String label, String data) {  
        System.out.println("handleEvent: " + data);  
    }  
  
    private void handleSensorData(JSONObject object) {  
        System.out.println("handleSensorData: " + object.toString());  
    }  
}
```

Sugestão: implemente usando alguma interface gráfica!

Mobile-Hub

Enviando regras CEP

- Linguagem EPS da EsperTech

http://esper.espertech.com/release-5.0.0/esper-reference/html_single/index.html

- Formato:

```
[
  {
    "MEPAQuery": {
      "type": "add|remove|start|stop|clear|get",
      "label": "AVGTemp",
      "object": "rule|event",
      "rule": "SELECT avg(sensorValue[1]) as value FROM
SensorData(sensorName='Temperature').win:time_batch(10 sec)",
      "target": "local|global"
    }
  }
]
```

Mobile-Hub

Enviando regras CEP – pacotes (1/5)

```
package br.com.meslin.lac;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.SocketAddress;
import java.util.List;
import java.util.UUID;

import lac.cnclib.net.NodeConnection;
import lac.cnclib.net.NodeConnectionListener;
import lac.cnclib.net.mrudp.MrUdpNodeConnection;
import lac.cnclib.sddl.message.ApplicationMessage;
import lac.cnclib.sddl.message.ClientLibProtocol.PayloadSerialization;
import lac.cnclib.sddl.message.Message;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
```

Mobile-Hub

Enviando regras CEP - construtor (2/5)

```
public class SendMEPAQuery implements NodeConnectionListener {
    private MrUdpNodeConnection connection;
    private String gatewayIP = "127.0.0.1";
    private int gatewayPort = 5500;

    public SendMEPAQuery() {
        InetAddress address = new InetAddress(gatewayIP, gatewayPort);
        UUID myUUID = UUID.randomUUID();
        try {
            connection = new MrUdpNodeConnection(myUUID);
            connection.addNodeConnectionListener(this);
            connection.connect(address);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Mobile-Hub

Enviando regras CEP – main e métodos auxiliares (3/5)

```
public class SendMEPAQuery implements NodeConnectionListener {
...
    public static void main(String[] args) {
        SendMEPAQuery mepa = new SendMEPAQuery();
        mepa.doAll();
    }
...
    @Override
    public void connected(NodeConnection arg0) { }
    @Override
    public void disconnected(NodeConnection arg0) { }
    @Override
    public void internalException(NodeConnection arg0, Exception arg1) { }
    @Override
    public void newMessageReceived(NodeConnection arg0, Message topicSample) { }
    @Override
    public void reconnected(NodeConnection arg0, SocketAddress arg1, boolean arg2, boolean arg3) { }
    @Override
    public void unsentMessages(NodeConnection arg0, List<Message> arg1) { }
}
```

Mobile-Hub

Enviando regras CEP – criando a mensagem (4/5)

```
public class SendMEPAQuery implements NodeConnectionListener {
    ...
    private void doAll() {
        String eplString = "select avg(sensorValue[1]) as value "
            + "from SensorData(sensorName='Temperature').win:time_batch(10 sec) "
            + "where sensorValue[1] > 20";
        JSONObject mepaQuery = new JSONObject();
        mepaQuery.put("type", "add"); // add|remove|start|stop|clear|get
        mepaQuery.put("label", "AVGTemp"); // rule|event
        mepaQuery.put("object", "rule");
        mepaQuery.put("rule", eplString);
        mepaQuery.put("target", "local");

        JSONArray jsonMsg = new JSONArray();
        JSONObject oneQuery = new JSONObject();
        oneQuery.put("MEPAQuery", mepaQuery);
        jsonMsg.add(oneQuery);
    }
    ...
}
```

Mobile-Hub

Enviando regras CEP – enviando a mensagem (5/5)

```
public class SendMEPAQuery implements NodeConnectionListener {
    ...
    private void doAll() {
        ...
        ApplicationMessage msg = new ApplicationMessage();
        msg.setRecipientID(UUID.fromString("63c38558-8f1e-4096-bd43-876872072f50"));
        msg.setContentObject(jsonMsg.toString());
        msg.setPayloadType(PayloadSerialization.JSON);
        try {
            connection.sendMessage(msg);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    ...
}
```


Perguntas?



Exercícios

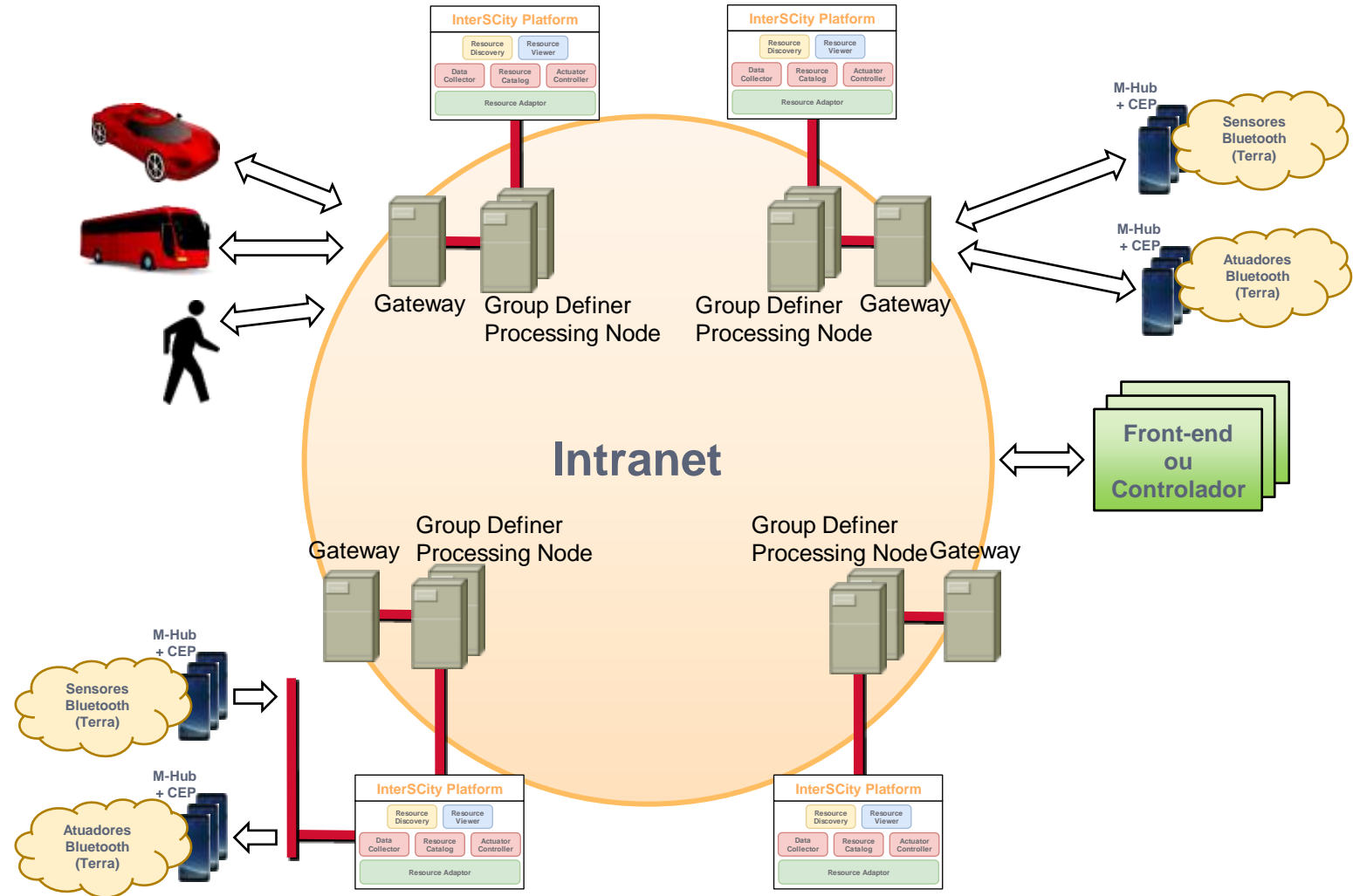
Exercício

- Exiba as informações do sensor tag.
- DESAFIO!!! Crie uma bússola.

ContextNet Distribuído

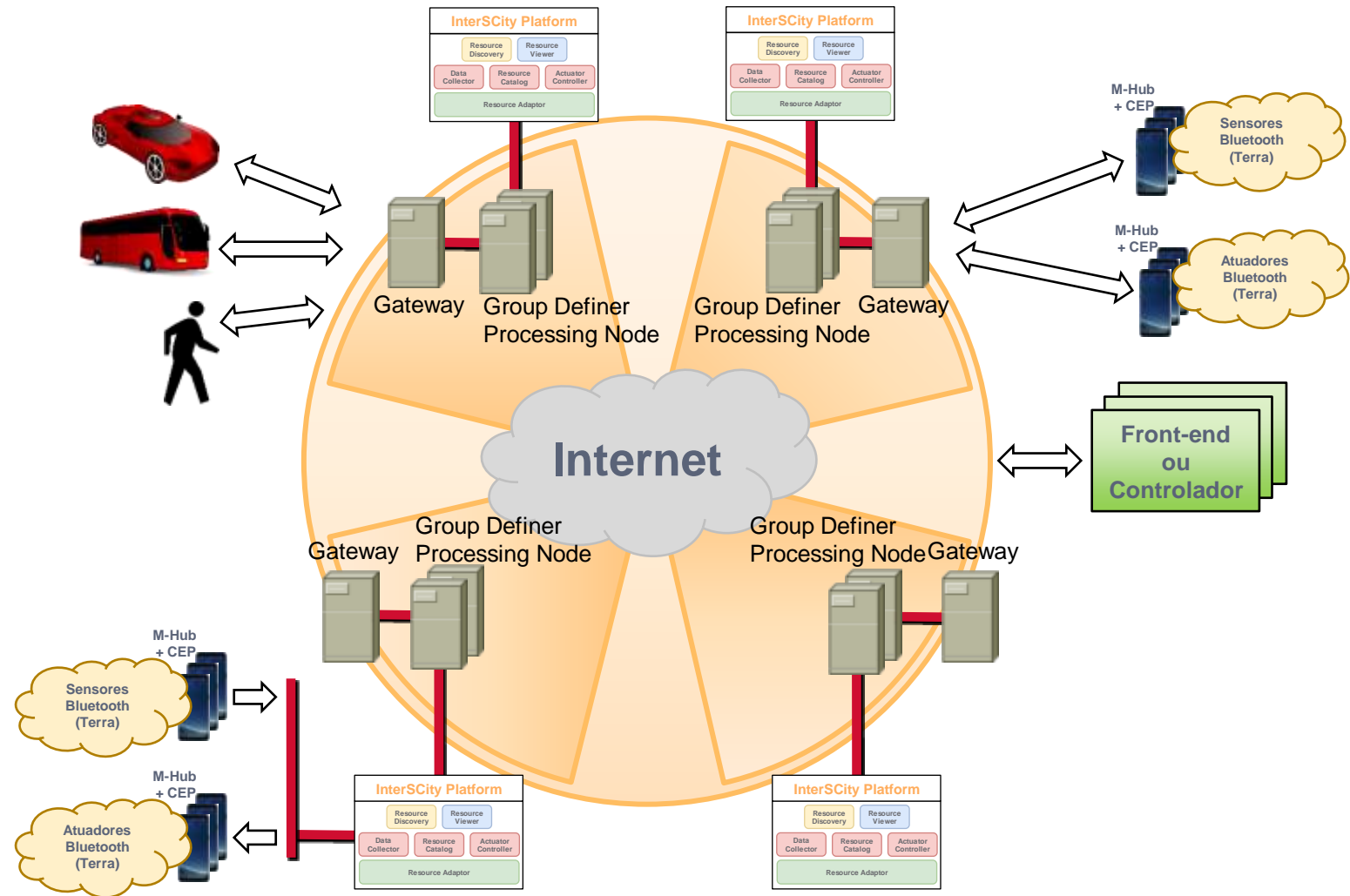
Até agora foi assim...

- Um único domínio de broadcast
- Mesma rede local (LAN)
- Descoberta automática de peers



Agora vai ser assim...

- Vários domínios de broadcast
- Rede de longa distância (WAN)
- Peers pré-configurados



O que muda para o programador?

- Nada
- Precisamos apenas reconfigurar o OpenSplice
 - Editar o arquivo ospl.xml



Editar o arquivo:


/opt/OpenSplice/HDE/x86_64.linux/etc/config/ospl.xml

```
<OpenSplice>
. . .
<DDSI2Service name="ddsi2">
  <General>
    <NetworkInterfaceAddress>
      172.16.1.202
    </NetworkInterfaceAddress>
    <AllowMulticast>false</AllowMulticast>
    <EnableMulticastLoopback>
      false
    </EnableMulticastLoopback>
    <CoexistWithNativeNetworking>
      false
    </CoexistWithNativeNetworking>
  </General>
. . .
```



Endereço IP da interface local

```
<OpenSplice>
. . .
<DDSI2Service name="ddsi2">
. . .
  <Discovery>
    <Peers>
      <Peer address="172.16.1.202"/>
      <Peer address="172.16.2.202"/>
      <Peer address="172.16.3.202"/>
      <Peer address="172.16.4.202"/>
    </Peers>
  </Discovery>
</DDSI2Service>
</OpenSplice>
```



Endereço dos peers

Perguntas?



Projetos usando o MUSANet

Projetos

Mobilidade

Comunicação

Sensores – Atuadores

Usuários

Grupos

Algumas sugestões

Comunicação por grupos

- Médicos/enfermeiros/técnicos/administrativos
- Professores/alunos graduação/alunos pós-graduação/por turma

Gerência de viajantes

- <http://wiki.lac.inf.puc-rio.br/doku.php?id=travelersgroup>

Ônibus chegando

- Aviso de que há algum ônibus chegando no ponto
- <http://dadosabertos.rio.rj.gov.br/apiTransporte/apresentacao/rest/index.cfm/obterTodasPosicoes>
- <http://www.sptrans.com.br/desenvolvedores/api-do-olho-vivo-guia-de-referencia/documentacao-api/>

Realidade aumentada (mas não muito...)

- Informações de contexto de acordo com a posição

Mais exemplos em:

- <http://wiki.lac.inf.puc-rio.br/doku.php?id=classes>

Perguntas?



Primeiros testes dos protótipos



Continuação dos testes



Validação e avaliação das aplicações desenvolvidas



Alexandre Meslin (Bolsista CAPES/BRASIL)

Noemi Rodriguez

Markus Endler

(meslin, noemi, endler)@inf.puc-rio.br

MUSANet – Mobile Urban Sensor and Actuator Network

