# Tópicos Avançados em Cidades Inteligentes
# Aula 3 – Microserviços do Interscity – Parte 1

Pablo Teófilo Durans
Orientado por: Francisco José da Silva e Silva

Laboratório de Sistemas Distribuídos Inteligentes (LSDi)
Universidade Federal do Maranhão (UFMA)
http://www.lsdi.ufma.br

Agosto de 2018


# Sumário

1. Resource Catalog
2. Resource Discovery
3. Data Collector
4. Exemplo de Aplicações

# Sumário

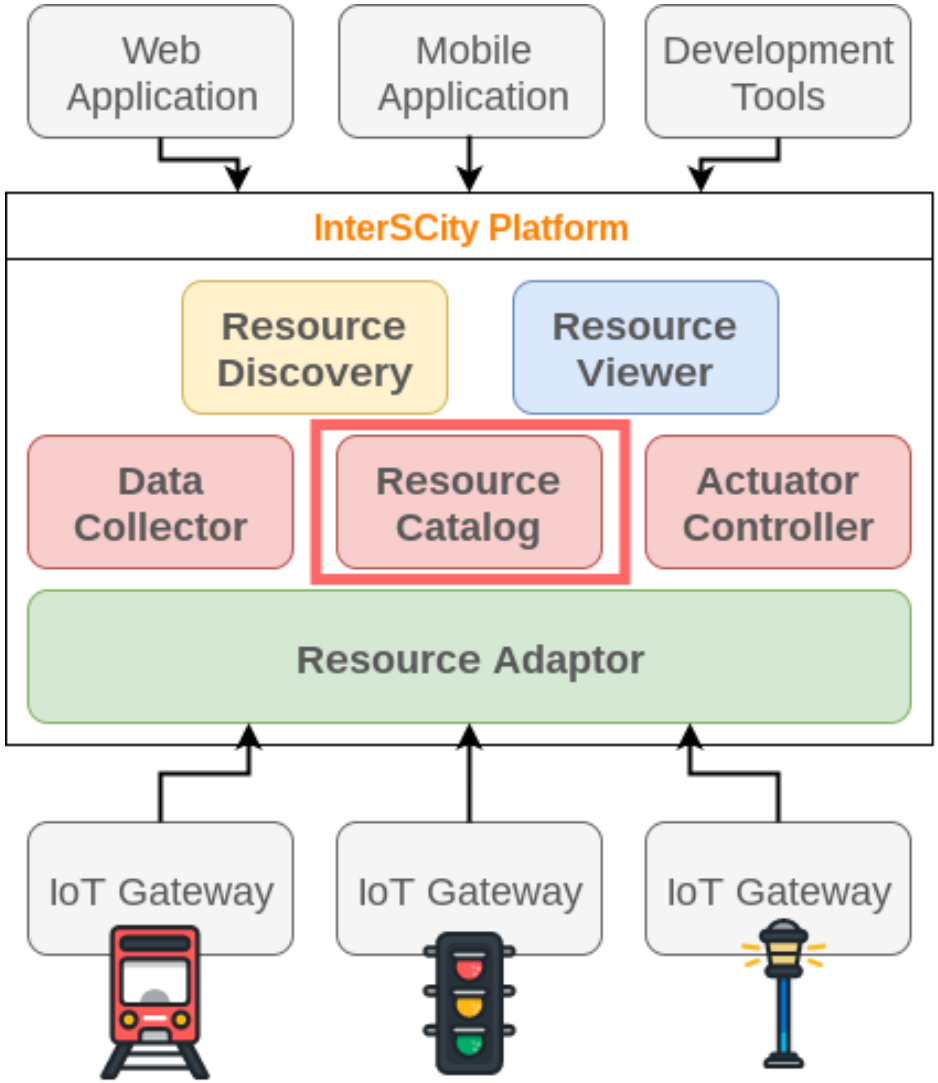
1 Resource Catalog

2 Resource Discovery

3 Data Collector

4 Exemplo de Aplicações

# Resource Catalog



# Resource Catalog

- Permite a Criação de **Recursos** e **Capacidades** da Cidade, armazenando dados estáticos como localização e descrição.

# Resource Catalog

- Permite a Criação de **Recursos** e **Capacidades** da Cidade, armazenando dados estáticos como localização e descrição.

- Capacidades
  - **Sensores:** Dados que podem ser coletados dos recursos da cidade (Ex: temperatura)
  - **Atuadores:** Atributos dos recursos sobre os quais a plataforma pode atuar (Ex: estado do semáforo – verde, amarelo ou vermelho)

# Resource Catalog

- Permite a Criação de **Recursos** e **Capacidades** da Cidade, armazenando dados estáticos como localização e descrição.

- Capacidades
  - **Sensores:** Dados que podem ser coletados dos recursos da cidade (Ex: temperatura)
  - **Atuadores:** Atributos dos recursos sobre os quais a plataforma pode atuar (Ex: estado do semáforo – verde, amarelo ou vermelho)

- **Recursos:** possuem capacidades (sensores e atuadores)

# Resource Catalog

- Permite a Criação de **Recursos** e **Capacidades** da Cidade, armazenando dados estáticos como localização e descrição.
- Capacidades
  - **Sensores:** Dados que podem ser coletados dos recursos da cidade (Ex: temperatura)
  - **Atuadores:** Atributos dos recursos sobre os quais a plataforma pode atuar (Ex: estado do semáforo – verde, amarelo ou vermelho)
- **Recursos:** possuem capacidades (sensores e atuadores)
- Capacidade x Recurso – Relação muitos para muitos

# Resource Catalog API - Capacidades

| | | |
|---|---|---|
| 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 |

# Resource Catalog API - Capacidades

- url base: `http://<host>/catalog/capabilities`

# Resource Catalog API - Capacidades

- url base: `http://<host>/catalog/capabilities`

- Através dos métodos GET, é possível obter o id, nome, tipo(sensor ou atuador) e descrição de todas as capacidades ou de uma capacidade específica (neste caso informando o nome da capacidade na url).

# Resource Catalog API - Capacidades

- url base: `http://<host>/catalog/capabilities`
- Através dos métodos GET, é possível obter o id, nome, tipo(sensor ou atuador) e descrição de todas as capacidades ou de uma capacidade específica (neste caso informando o nome da capacidade na url).
- Através do método POST, é cadastrada uma nova capacidade.

# Resource Catalog API - Capacidades

- url base: `http://<host>/catalog/capabilities`
- Através dos métodos GET, é possível obter o id, nome, tipo(sensor ou atuador) e descrição de todas as capacidades ou de uma capacidade específica (neste caso informando o nome da capacidade na url).
- Através do método POST, é cadastrada uma nova capacidade.
- Através do método PUT, uma capacidade identificada na url é atualizada

# Resource Catalog API - Capacidades

- url base: `http://<host>/catalog/capabilities`
- Através dos métodos GET, é possível obter o id, nome, tipo(sensor ou atuador) e descrição de todas as capacidades ou de uma capacidade específica (neste caso informando o nome da capacidade na url).
- Através do método POST, é cadastrada uma nova capacidade.
- Através do método PUT, uma capacidade identificada na url é atualizada
- O método DELETE exclui uma capacidade

# Resource Catalog API - Json Capacidade

```
{
  "capabilities": [
    {
      "id": 1,
      "name": "bikecount",
      "function": 0, // "capability_type": "sensor",
      "description": "Number of bikes in a station"
    },
    {
      "id": 2,
      "name": "notifyuser",
      "function": 1, // "capability_type": "actuator",
      "description": "To notify users of city"
    }
  ]
}
```

# Resource Catalog API - Recurso

| | | |
|---|---|---|
| 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 |

# Resource Catalog API - Recursos

- url base: `http://<host>/catalog/resources`

# Resource Catalog API - Recursos

- url base: `http://<host>/catalog/resources`
- Através do método POST, é cadastrado um novo recurso.

# Resource Catalog API - Recursos

- url base: `http://<host>/catalog/resources`
- Através do método POST, é cadastrado um novo recurso.
- Através do método PUT, um recurso identificado na url é atualizado

# Resource Catalog API - Recursos

- url base: `http://<host>/catalog/resources`
- Através do método POST, é cadastrado um novo recurso.
- Através do método PUT, um recurso identificado na url é atualizado
- Não é possível excluir recursos

# Resource Catalog API - Recursos

- Através dos métodos GET na raiz, é possível obter diversas informações sobre os recursos da cidade (uuid, descrição, capacidades, localização etc) de todos os recursos;

# Resource Catalog API - Recursos

- Através dos métodos GET na raiz, é possível obter diversas informações sobre os recursos da cidade (uuid, descrição, capacidades, localização etc) de todos os recursos;
- Adicionando o uuid à url são mostradas as informações de um recurso da cidade específico;

# Resource Catalog API - Recursos

- Através dos métodos GET na raiz, é possível obter diversas informações sobre os recursos da cidade (uuid, descrição, capacidades, localização etc) de todos os recursos;
- Adicionando o uuid à url são mostradas as informações de um recurso da cidade específico;
- Através do GET na url interna search é possível filtrar os recursos da cidade por qualquer um de seus atributos **estáticos**;

# Resource Catalog API - Recursos

- Através dos métodos GET na raiz, é possível obter diversas informações sobre os recursos da cidade (uuid, descrição, capacidades, localização etc) de todos os recursos;
- Adicionando o uuid à url são mostradas as informações de um recurso da cidade específico;
- Através do GET na url interna search é possível filtrar os recursos da cidade por qualquer um de seus atributos **estáticos**;
- Através do GET na url interna sensors, é retornado apenas recursos da cidade que possuem alguma capacidade do tipo sensor;

# Resource Catalog API - Recursos

- Através dos métodos GET na raiz, é possível obter diversas informações sobre os recursos da cidade (uuid, descrição, capacidades, localização etc) de todos os recursos;
- Adicionando o uuid à url são mostradas as informações de um recurso da cidade específico;
- Através do GET na url interna search é possível filtrar os recursos da cidade por qualquer um de seus atributos **estáticos**;
- Através do GET na url interna sensors, é retornado apenas recursos da cidade que possuem alguma capacidade do tipo sensor;
- De forma analógica, a url interna actuators retorna os recursos da cidade que possuem alguma capacidade do tipo atuador.


# Resource Catalog API - Json Recurso

```
{
  "resources": [
    {
      "uuid": "45b7d363-86fd-4f81-8681-663140b318d4",
      "id": 10,
      "description": "McPherson Square Station",
      "capabilities": [
        "temperature",
        "bikecount"
      ],
      "status": "active",
      "lat": -77.032407,
      "lon": 38.900784,
// other attr: country, state, city, neighborhood,
//             postal_code, created_at, updated_at
    }   // {...}
  ]
}
```

# Sumário

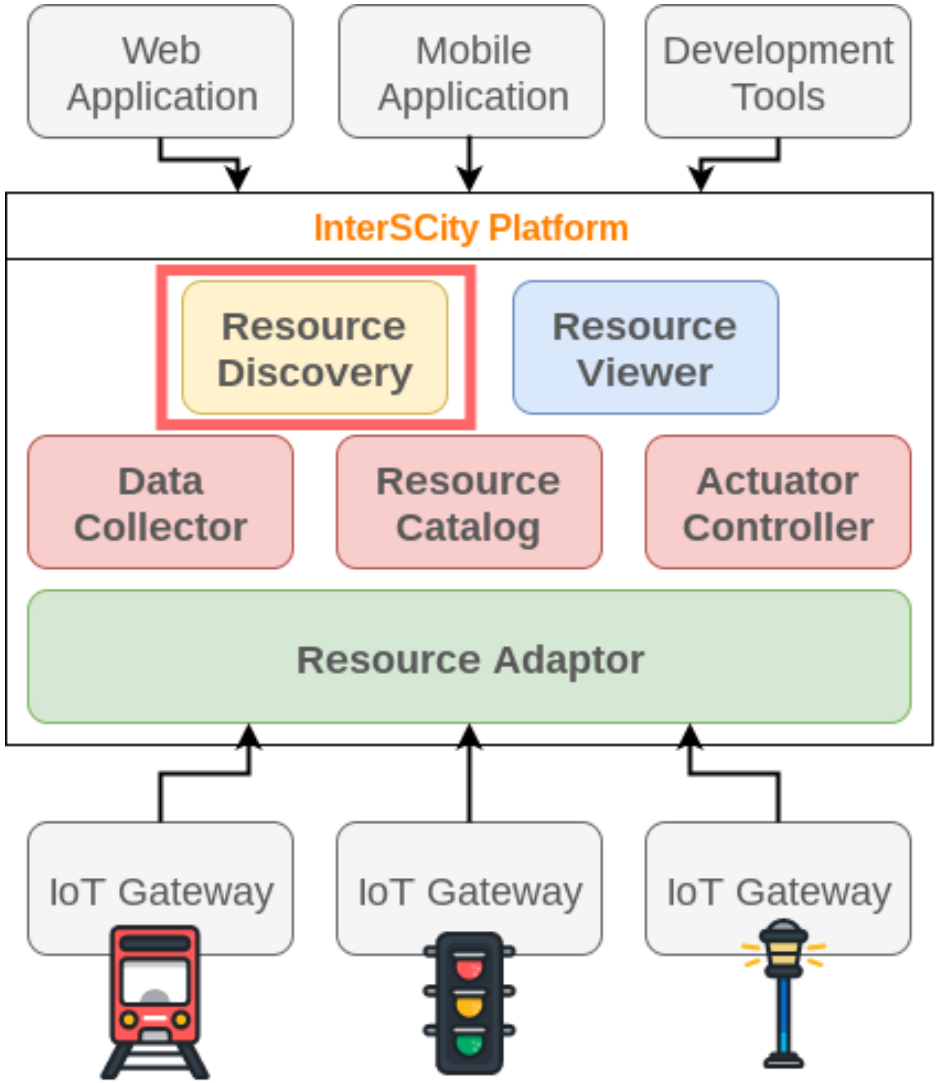
1 Resource Catalog

2 Resource Discovery

3 Data Collector

4 Exemplo de Aplicações

# Resource Discovery



# Resource Discovery

Permite que recursos sejam descobertos através filtros sobre:

- **Localização**: é dado um ponto em latitude e longitude e um raio de busca
- **Capacidade**: retorna os recursos que possuem uma capacidade
- **Dados do Contexto Atual**: Através de operadores: igual, diferente, maior que, menor que, igual ou maior que, igual ou menor que e presença em um conjunto.

# Exemplos de URLs

- `http://<host>/discovery/resources?capability=bikecount`
  Retorna todos os recursos que possuem a capacidade bikecount

- `http://<host>/discovery/resources?capability=bikecount;bikecount.gte=2`
  Retorna todos os recursos que possuem a capacidade bikecount cujo o valor de contexto é atualmente maior que 2

- `http://<host>/discovery/resources?lat=-77.032407;lon=38.900784;radius=500`
  Retorna os recursos que estão no raio de 500 metros da localização especificada

# Sumário

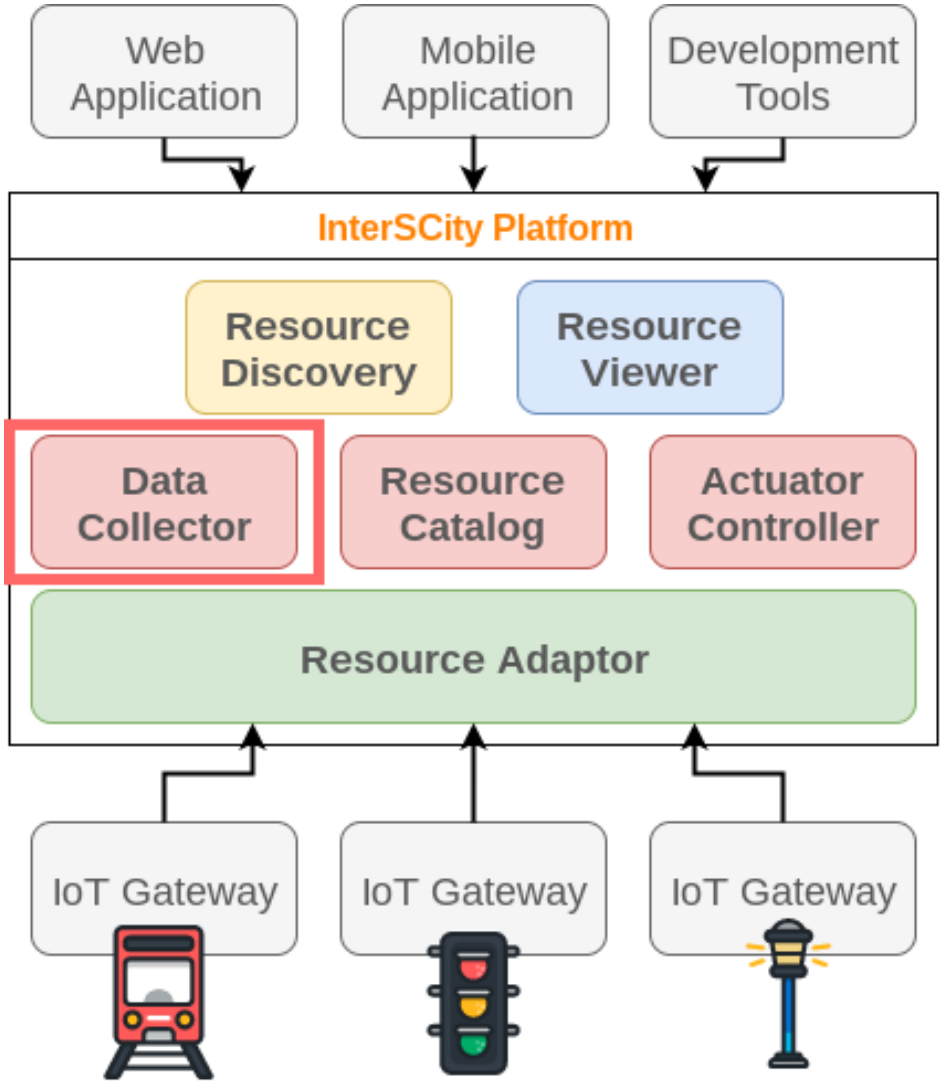
1 Resource Catalog

2 Resource Discovery

3 Data Collector

4 Exemplo de Aplicações

# Data Collector



# Data Collector

- Gerencia toda informação de contexto capturada pelos sensores de capacidades;

# Data Collector

- Gerencia toda informação de contexto capturada pelos sensores de capacidades;
- Utiliza o banco de dados NoSQL MongoDB;

# Data Collector

- Gerencia toda informação de contexto capturada pelos sensores de capacidades;
- Utiliza o banco de dados NoSQL MongoDB;
- As informações de contexto recentes ficam em memória;

# Data Collector

- Gerencia toda informação de contexto capturada pelos sensores de capacidades;
- Utiliza o banco de dados NoSQL MongoDB;
- As informações de contexto recentes ficam em memória;
- Filtros podem ser usados para retornar apenas dados relevantes

# Data Collector - API

| | | |
|---|---|---|
| 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 |

# Data Collector API - Json Request

```
{
  "uuids": [
  // Nao presente quando o uuid se encontra na url
    "45b7d363-86fd-4f81-8681-663140b318d4",
    "23ebc9f9-4efe-4a05-958b-0252b77cc700"
  ],
  "capabilities": [
    "temperature",
    "bikecount"
  ],
  "matchers": {
    "temperature.gte": 13.498,
    "temperature.lte": 18.091,
    "bikecount.gte": 2
  },
  "start_range": "2018-08-13T17:21:37",
  "end_range": "2018-08-13T18:21:37"
}
```

# Data Collector API - Json Response

```
{ "resources": [
    { "uuid": "45b7d363-86fd-4f81-8681-663140b318d4",
      "capabilities": {
        "temperature": [
          { "temperature": 38.313,
            "date": "2018-08-13T17:30:00"
          }// , ... Outras capturas de temperatura
        ]
        "bikecount": [
          { "bikecount": 4,
            "date": "2018-08-13T17:30:00"
          }//, ...
        ]
      }
    }//, ... Outros Recursos
  ]
}
```

# Exemplos de URLs

Método POST:

- `http://<host>/collector/resources/data`
  Retorna todos os valores de contexto que batem com as restrições passadas no conteúdo do http

# Exemplos de URLs

Método POST:

- `http://<host>/collector/resources/data`
  Retorna todos os valores de contexto que batem com as restrições passadas no conteúdo do http

- `http://<host>/collector/resources/data/last`
  Retorna o último valor de cada contexto (capability x resource) que batem com as restrições

# Exemplos de URLs

Método POST:

- `http://<host>/collector/resources/data`
  Retorna todos os valores de contexto que batem com as restrições passadas no conteúdo do http
- `http://<host>/collector/resources/data/last`
  Retorna o último valor de cada contexto (capability x resource) que batem com as restrições
- `http://<host>/collector/resources/016f3a5f-.../data`
  Retorna todos os valores de contexto do recurso 016f3a5f-... que batem com as restrições passadas no conteúdo do http

# Exemplos de URLs

Método POST:

- `http://<host>/collector/resources/data`
  Retorna todos os valores de contexto que batem com as restrições passadas no conteúdo do http
- `http://<host>/collector/resources/data/last`
  Retorna o último valor de cada contexto (capability x resource) que batem com as restrições
- `http://<host>/collector/resources/016f3a5f-.../data`
  Retorna todos os valores de contexto do recurso 016f3a5f-... que batem com as restrições passadas no conteúdo do http
- `http://<host>/collector/resources/016f3a5f-.../data/last`
  Retorna o último valor de cada contexto do recurso 016f3a5f-... que batem com as restrições

# Sumário

1 Resource Catalog

2 Resource Discovery

3 Data Collector

4 Exemplo de Aplicações

# Smart LSDI

- O Laboratório de Sistemas Distribuídos Inteligentes possui 5 salas ao total:
  - Sala de Servidores
  - Laboratório dos Alunos
  - Depósito de Computadores
  - Sala de Reunião
  - Sala dos Professores

# Smart LSDI

- O Laboratório de Sistemas Distribuídos Inteligentes possui 5 salas ao total:
  - Sala de Servidores
  - Laboratório dos Alunos
  - Depósito de Computadores
  - Sala de Reunião
  - Sala dos Professores
- Em cada uma dessas salas poderia-se coletar dados de temperatura, humidade, sensor de fumaça;

# Smart LSDI

- O Laboratório de Sistemas Distribuídos Inteligentes possui 5 salas ao total:
  - Sala de Servidores
  - Laboratório dos Alunos
  - Depósito de Computadores
  - Sala de Reunião
  - Sala dos Professores
- Em cada uma dessas salas poderia-se coletar dados de temperatura, humidade, sensor de fumaça;
- Também poderia ser possível executar alteração da temperatura do ar condicionado;

# Smart LSDI

- O Laboratório de Sistemas Distribuídos Inteligentes possui 5 salas ao total:
  - Sala de Servidores
  - Laboratório dos Alunos
  - Depósito de Computadores
  - Sala de Reunião
  - Sala dos Professores
- Em cada uma dessas salas poderia-se coletar dados de temperatura, humidade, sensor de fumaça;
- Também poderia ser possível executar alteração da temperatura do ar condicionado;
- Tais medições e atuações já são feitas na Sala de Servidores

# Smart LSDI - Modelagem

## Recursos

## Capacidades

### Sensores

### Atuador

# Aviso de Disponibilidade de Bicicletas - Washington DC

- Diversas cidades do mundo possuem sistemas de bicicletas compartilhadas;

# Aviso de Disponibilidade de Bicicletas - Washington DC

- Diversas cidades do mundo possuem sistemas de bicicletas compartilhadas;
- Usuário retira uma bicicleta próximo a sua origem e devolve-a perto do destino;

# Aviso de Disponibilidade de Bicicletas - Washington DC

- Diversas cidades do mundo possuem sistemas de bicicletas compartilhadas;
- Usuário retira uma bicicleta próximo a sua origem e devolve-a perto do destino;
- Estações possuem número limitado de bicicletas

# Aviso de Disponibilidade de Bicicletas - Washington DC

- Diversas cidades do mundo possuem sistemas de bicicletas compartilhadas;
- Usuário retira uma bicicleta próximo a sua origem e devolve-a perto do destino;
- Estações possuem número limitado de bicicletas
- Dentre outras cidades, Washington DC fornece uma API pública para verificar disponibilidade de bicicletas em estações

# DC Bikes - Modelagem

## Recursos

## Capacidades

### Sensor

Quantidade de bikes

### Atuador

Notificação ao Usuário