Manual técnico para el desafío de Ciudad Inteligente 2018

Se entregará la imagen de una máquina virtual VirtualBox con sistema operativo linux CentOS.

El sistema tendrá tres servicios instalados en contenedores Docker. Uno de ellos tiene el servicio Context Broker (codename **Orion**) de la plataforma **FIWARE,** una base MongoDB sobre la que trabaja Orion y un proceso que simula la llegada de paquetes de datos de los buses e impacta ésta información en el componente Orion a través de su API abierta REST.

Éste proceso simulador lee los datos desde un archivo que contiene un histórico con posiciones de los buses de un periodo de varias horas.

El servicio de simulador tiene una interfase REST que permite pausar y reiniciar la simulación.

Además se proveerán archivos adicionales con juegos de datos.

Sobre la maquina virtual

La máquina virtual tendrá un usuario genérico “desafio” y con contraseña “root”

Uso del componente Orion

El contenedor del Orion tiene configurado el inicio automático por lo que siempre estará disponible.

Este componente Fiware tiene una API muy completa y extensa de la cual para nuestro caso solo será necesario utilizar el API de subscripciones y el de consultas.

El Orion trabaja en base a un mecanismo de tipo “publish / subscribe”, que es una característica especialmente diseñada para tiempo real.

De esta forma podemos enterarnos cuando hay datos nuevos de buses en forma fiable y efectiva sin tener que estar haciendo consultas aleatorias en ciertos períodos de tiempo con alto riesgo de perder información.

La suscripción se realiza sobre entidades existentes en el broker y se especifica una url en la que exista un cliente “escuchando” los eventos de esas entidades.

Consulta de los datos en Orion de la simulación

La consulta de los datos se puede realizar a través de la interfase REST de Orion.

Para consultar los buses que hay se debe realizar la siguiente invocación con un HTTP GET

con el header “fiware-service=default” y el header “fiware-servicepath=/”

http://<direcciondelamaquinavirtual>:1026/v2/entities

Esto devolverá todas las entidades (en este caso buses) disponibles en Orion en un archivo JSON.

Retornará algo similar a esto


```
[
    {
        "id": "241",
        "type": "Bus",
        "codigoBus": {
            "type": "Number",
            "value": 241,
            "metadata": {}
        },
        "linea": {
            "type": "Text",
            "value": "7516",
            "metadata": {}
        },
        "location": {
            "type": "geo:json",
            "value": {
                "type": "Point",
                "coordinates": [
                    -56.253967,
                    -34.8955
                ]
            },
            "metadata": {}
        },
        "timestamp": {
            "type": "DateTime",
            "value": "2018-10-02T15:50:40.00Z",
            "metadata": {}
        }
    }
]
```


Por ejemplo , la consulta con el comando CURL de UNIX será la siguientes

```
curl --request GET \
  --url http://<direcciondelamaquinavirtual>/v2/entities \
  --header 'fiware-service: default' \
  --header 'fiware-servicepath: /'
```

Quien quiera ampliar, puede verlo en la documentación oficial:

https://fiware-orion.readthedocs.io/en/latest/user/walkthrough_apiv2/#query-entity

Subscripciones

La url base es : http://<direccion de la maquina virtual>:1026/v2/subscriptions que devuelve todas las subscripciones que tiene el componente.

Con los métodos POST y DELETE podemos crear o eliminar una subscripción respectivamente.

Para crear la subscripción con el método POST se debe enviar como cuerpo un archivo JSON que tiene dos propiedades fundamentales, *subject* y *notification*. La propiedad *subject* refiere a las entidades de las que el cliente desea ser notificado y a cuales de las propiedades de esas entidades se les observa los cambios para generar los eventos. La propiedad *notification* refiere a la url del cliente y a los atributos de la entidad que interesa sean enviados en la notificación.

Quien quiera ampliar, puede verlo en la documentación oficial:

https://fiware-orion.readthedocs.io/en/latest/user/walkthrough_apiv2/#subscriptions

Uso del simulador

El simulador tiene una pequeña API rest que permite ejecutarlo desde el inicio, pausarlo y reiniciar desde la última pausa.

Al iniciarlo, el simulador comenzará a aplicar la historia definida y la reflejará en el servidor FIWARE Orion, la simulación es de aproximadamente tres horas.

Los datos serán impactados en el componente Orion con un formato predefinido para las entidades correspondientes a los buses.

Interfase REST del servicio de simulación

Se podrá interactuar con la simulación a través de los siguientes comandos REST

REINICIAR SIMULACIÓN

Arranca la simulación. Se escriben en el servidor FIWARE ORION los datos de los buses cada 15 segundos aproximadamente.

Realizar un GET al a dirección
http://<direccion ip de la maquina virtual>/api/simulador?op=reiniciar

PAUSA SIMULACIÓN

Pausa la simulación , y no se escriben datos en el servidor FIWARE.

Realizar un GET al a dirección
http://<direccion ip de la maquina virtual>/api/simulador?op=pausa

CONTINUAR SIMULACIÓN

Continua la simulación después haber realizado una pausa.
Realizar un GET al a dirección
http://<direccion ip de la maquina virtual>/api/simulador?op=continuar

Archivos de datos adicionales

Se podrá descargar el archivo JSON con la información de la ubicación de las paradas y el recorrido de las lineas

trayectosporlinea.json

Por ejemplo:

{"codigoParada":"4836","linea":"217","ordinal":"1","calle":"AV DRA MA L SALDUN DE RODRIGUEZ","esquina":"AV BOLIVIA","long":"-56.0833009323971","lat":"-34.8818168390103"}

En donde

- codigoParada: es un identificador unico de la parada
- linea: es la linea de bus
- ordinal: es un numero que ordena como una linea va visitando cada parada
- calle y esquina : contienen el texto de la esquina
- long: longitud
- lat: latitud

De la página http://sig.montevideo.gub.uy/ se podrá descargar cualquier otra información geográfica adicional.

Servicio TEA

El desafío implica la implementación de una solución TAE (Tiempo de arribo estimado) en un servicio REST / JSON. Éste servicio REST será consultado acerca de la predicción del arribo del siguiente bus de una línea a una parada, en donde la línea y la parada son parámetros obligatorios del servicio.

La firma de la llamada al servicio será la siguiente (invocación de un GET) :

http://localhost:8080/nextBus/<id_linea>/<id_parada>

La respuesta se hará en un archivo JSON con el siguiente formato

{ “id_linea”: xxxx, “id_parada”: xxxx, “id_bus”: xxxx, “location”: xxxx, “tea”: xxxx }

Las propiedades **id_linea** e **id_parada** son los identificadores de línea y parada que se pasaron como parámetros, la propiedad id_bus es el identificador del siguiente bus, location es la posición del bus en este momento (latitud / longitud en formato GEOJSON) y la propiedad **tea** especifica la predicción del tiempo de arribo del bus a la parada en segundos.

Ejemplo:

```
{
        “id_linea”: “1234”,
        “id_parada”: 1234,
        “id_bus”: 314,
        “location”: {
                "type": "Point",
                "coordinates":[-56.19539,-34.90608],
        },
        “tea”: 150
}
```

Además de este servicio que responde las consultas de tiempo de arribo se necesitará por lo menos de otro servicio que sea cliente de las subscipciones a Orion.

La implementación de los servicios REST necesarios se habilitan en Java (JAX-RS) sobre Tomcat 8 o javascript en NodeJS 8.

Solo se especifica la firma y el archivo JSON resultado del servicio de consulta TEA, para los otros servicios necesarios estos detalles quedan a criterio de cada grupo.

Se requiere que la solución utilice una variables de ambiente llamada ORION_URL para obtener el camino a la url base del componente Orion.

Evaluación

La evaluación se realizará tomando un caso de prueba definido por la mesa examinadora y se realizarán los siguientes pasos:

- Instalación de la solución entregada por el grupo a examinar en la máquina de prueba. Esto requiere la configuración de las variables de ambiente referidas en la sección anterior para que los servicios funcionen correctamente.
- Ejecución del simulador sobre un conjunto de datos históricos para la evaluación.
- Se invoca el servicio implementado por el grupo con las condiciones de la evaluación.

Con la información obtenida (archivos JSON devueltos por el servicio) se calculará el error cuadrático medio con respecto a los resultados correctos que son conocidos. Ese indicador final corresponde al 60% de la prueba y se completará con el 40% en base a la evaluación del código y el informe entregado por el grupo.