Sii-Mobility

Supporto di Interoperabilità Integrato per i Servizi al Cittadino e alla Pubblica Amministrazione

Trasporti e Mobilità Terrestre, SCN_00112
Deliverable ID: DE3.16
Titolo: Specifica di dettaglio dei moduli di integrazione con applicazioni fisse e mobili

Data corrente	14-04-2017
Versione (solo il responsabile può cambiare versione)	0.6
Stato (draft, final)	FINALE
Livello di accesso (solo consorzio, pubblico)	Pubblico quando completo
WP	WP1
Natura (report, report e software, report e HW)	Report
Data di consegna attesa	M12, Dicembre 2016
Data di consegna effettiva	14-04-2017
Referente primario, coordinatore del documento	Laura Coconea, SWARCO, laura.coconea@swarco.com
Contributor	P. Bellini, P. Nesi, D. Cenni, A. DiFino
Coordinatore responsabile del progetto	Paolo Nesi, UNIFI, paolo.nesi@unifi.it

Sommario

1 Executive Summary (SWARCO)

Questo deliverable describe le funzioni di integrazione fra i moduli del sistema Sii-Mobility. Le intergrazioni sono principalmente sviluppate tramite la definizione di API, Application Programme Interfaces attraverso le quali Applicazioni mobili o web possono effettuare delle chiamate ai vari servizi del sistema Sii-Mobility.
Le principali categoria di queste API sono relative alle comunicazioni fra:

- centrale di direzionamenot e sistema Sii, questi fa riferimento al sottosistema SSM che riceve le chiamate dai sistemi periferici e in certi casi comunica un dato o un valore come risposta. In qesto caso il sistema di direzionamento come i sistemi di segnalamento devono poter comunicare il loro stato e leggere da server i valori ai quali devono impostartsi
- sii-mobility e nuover sorgenti dati: per esempio i sistemi di IOT
- sistema di supervisione ed Sii-Mobility. Le API in questo caso sono relative a:
    - Informani di traffico, stato sensori
    - Informaizoni di eventi di traffico, incidenti, chusure, etc.
    - Informazioni relativi ai semafori per soluzioni di guida connessa.
    - Informaizoni relative allo stato del semaforo in termini di persone in attesa
- smart city API e applicazioni mobili o web
- PA e SME e Sii-mobility
- Varie smart city che hanno tutte un struttura conforme a Sii-mobility.

2 Architettura di Riferimento

Sii-Mobility intende creare una soluzione che possa abilitare un'ampia gamma di servizi al cittadino e commerciali in connessione e integrati con il sistema di mobilità: collezionando dati puntuali e attualizzati in tempo reale da varie fonti; analizzando i flussi di dati con varie tipologie di algoritmi producendo azioni e informazioni tramite applicazioni web e mobili, totem informativi, ecc.; mettendo a disposizione dati elaborati e puntuali, che potranno essere usati da PA, gestori, e imprese per produrre servizi più efficaci ed efficienti, e anche nuovi servizi integrati. Permettendo a PA e PMI di caricare ulteriori algoritmi sul sistema per erogare servizi verso gli utenti finali e verso le PA. Per esempio algoritmi di routing, di valutazione e predizione di condizioni critiche, di ottimizzazione delle risorse, di personalizzazione dei percorsi, di guida connessa, etc.

Nell'architettura del progetto **Sii-Mobility** si possono notare le interfacce per la connessione con altri sistemi di Smart city, con il sistema di mobilità nazionale, la rilevazione dati ambientali, le ordinanze, etc.

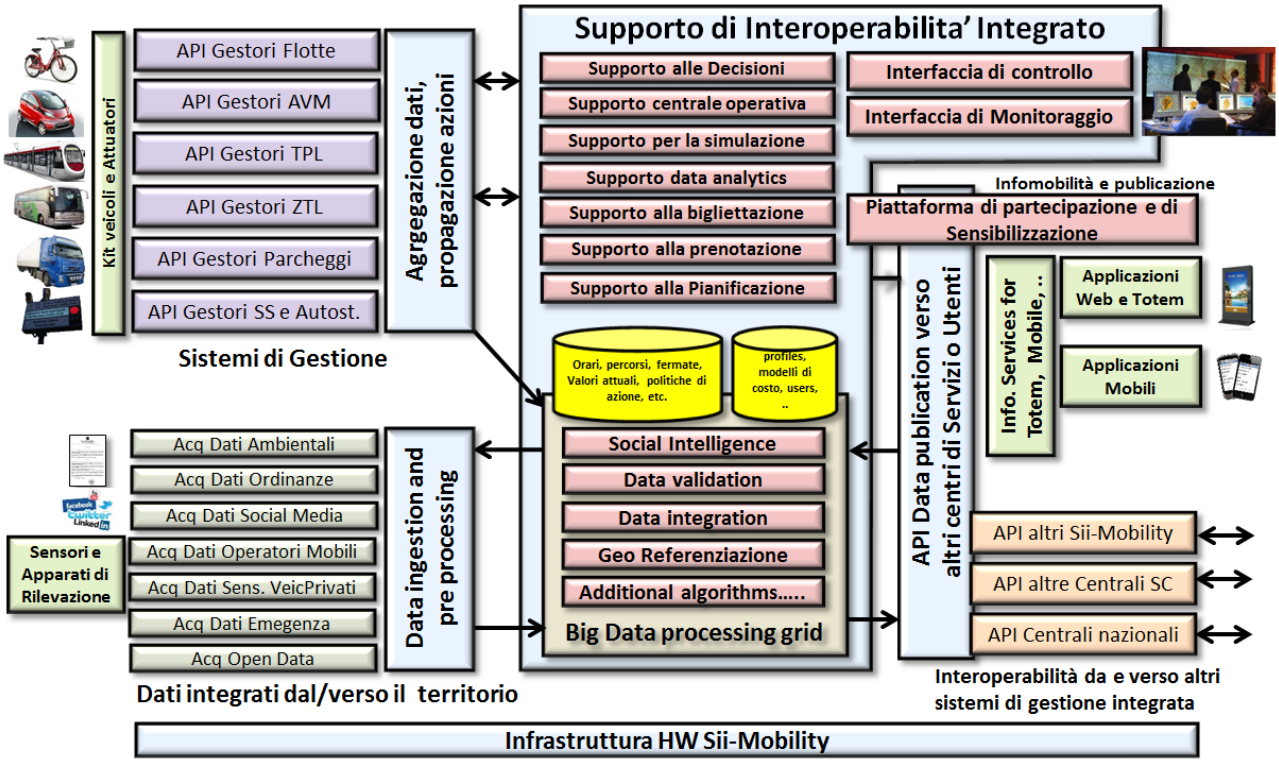


Figura 1: L'architettura di Sii-Mobility, con i dettagli interni di del SII, Supporto di Interoperabilità Integrato.

3 Specifica: Integrazione dei sensori e attuatori verso il SII, ottimizzazione

		ECM	Mizar	UNIFI	IN20	TIME	Negentis	EffKnow	liberologico	ataf	busitalia	cttnord	tiemme	argos	elfi	calamai	Midra	Project	GEOIN	QUESTIT	SOFTEC	EWINGS
3.4.1	Integrazione verso il SII, ottimizzazione: IN/OUT kit veicolari, sensori, attuatori		X				X			X	X	X	X									
3.4.2	Integrazione verso il SII, ottimizzazione: API per IN/OUT con mobile e Totem			X			X															
3.4.3	Integrazione verso il SII, ottimizzazione: API per dati PA e SME		X	X						X	X	X	X									
3.4.4	Integrazione verso il SII, ottimizzazione: API sii-mobility interop. e con altre centrali		X	X			X															

Attività	Sotto-sistema	Tool/Level	Descrizione sommaria
Integrazione verso il SII, ottimizzazione: IN/OUT kit veicolari, sensori, attuatori	Attuatori Sensor Server and Manager	Protocollo di comunicazione tra Centrale Direzionamento ed SII (API07)	Protocollo di comunicazione fra la Centrale Direzionamento ed il sistema SII. Il sistema di gestione deve presentare delle API per: accettare comandi dall'esterno programmando l'ora del cambio direzione oppure subito, fornire dati relativi allo stato ed ai sensori ed allo storico.
	Monitoring Supervisor	API per integrazione di nuove sorgenti dati (API08)	API per: integrare dati provenienti da nuovi sorgenti (e.g. FCD, Floating Cellular Data) e per fornire dati relativi allo stato ed ai sensori ed allo storico proveniente dal Gestore della mobilità.
Integrazione verso il SII, ottimizzazione: API per IN/OUT con mobile e Totem	Smart City API	API REST di Knowledge Base su base Geolocation (API09)	API per accesso ai dati della Knowledge Base tramite REST call, sulla base di una coordinata GPS, per ottenere: XML, JSON, HTML. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.
	Smart City API	API SPARQL di Knowledge Base (API10)	API per accesso ai dati di della Knowledge Base tramite SPQRQL, per ottenere: XML, JSON, HTML. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.
	Smart City API	API for rendering deductions on the basis of the whole ontological model o SmartDS (API11)	API con le quali è possibile fare query sulla knowledge base per servizi e per informazioni statistiche, ma anche sullo stato di processi decisionali dello SmartDS. Questo significa avere accesso ai dati del database del DashBoard manager. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.
	Smart City API	API Query ID di Knowledge Base (API12)	API per accesso ai dati di della Knowledge Base tramite Query ID, per ottenere: XML, JSON, HTML. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.

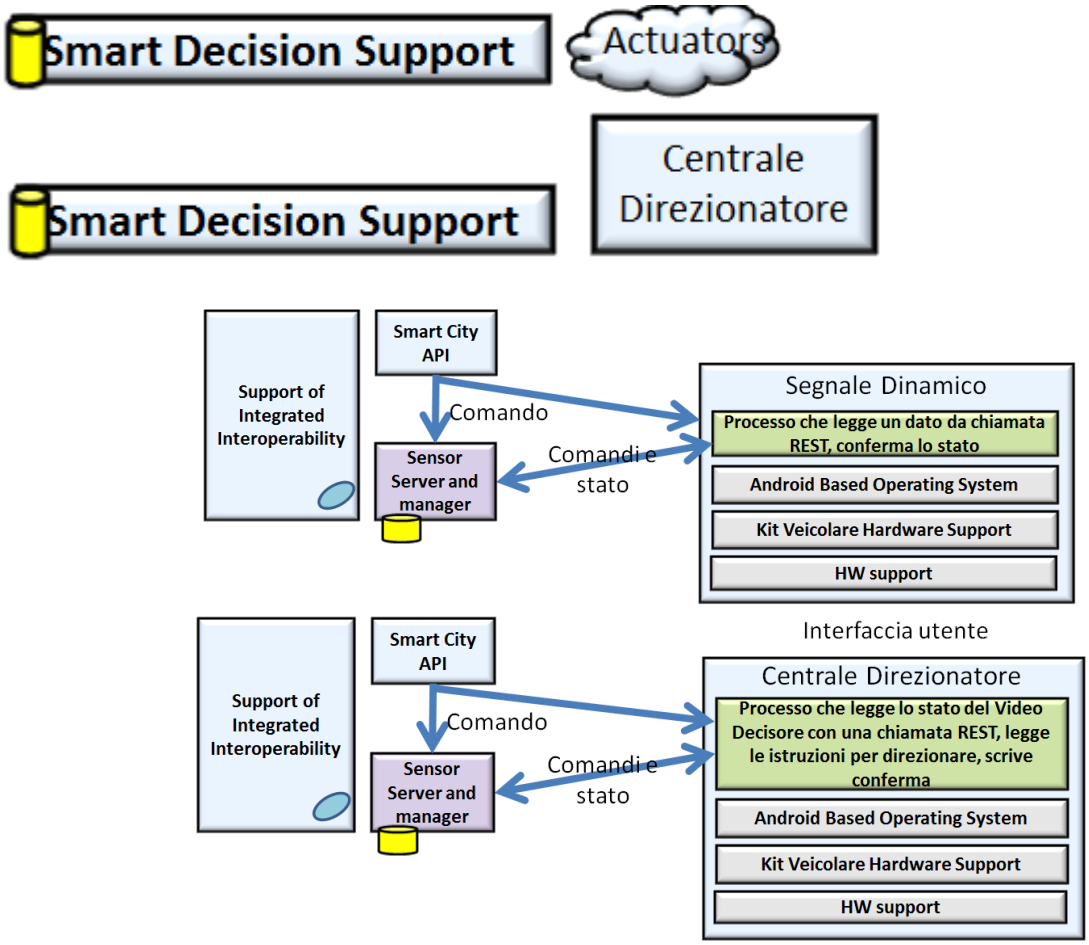
	Smart City API	API User Crowd Sourcing (API13)	API per fornire informazioni e dati al sistema partecipativo e comunque al Sii. Informazioni come: immagini, commenti, ranking/voti riguardo a servizi
	Smart City API	API User Engagement (API14)	API per inviare in push (da server verso le mobile APP e web) suggerimenti e stimoli a partecipare, compiti da svolgere, etc.
	Smart City API	API User Profiling and Recommendations (API15)	API per permettere alle App mobile e WEB di richiedere suggerimenti al server di user profiling and suggestion e comunicare user profile, come anche il suo menu, comportamento, etc..
Integrazione verso il SII, ottimizzazione: API per dati PA e SME	Smart City API	API request planning (API16)	API per richiedere la pianificazione del percorso fra due punti (o sequenza di punti) tramite: mezzi pubblici, auto, e a piedi. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.
	Smart City API	API pubblicazione dati (API17)	API per accedere a dati storici per PA e SME. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti per implementazione di nuovi scenari B2B o B2C
Integrazione verso il SII, ottimizzazione: API Sii-Mobility interop. e con altre centrali	Smart City API	API verso altre smart city (API18)	API con le quali altre centrali smart city possono richiedere e fornire informazioni. API con accesso condizionato in base all'utente, al profilo utente, ai dati richiesti.

4 IN/OUT kit veicolari, sensori, attuatori

Nei seguenti paragrafi saranno presentati i moduli di acquisizione ed integrazione di kit veicolari, sensori e attuatori con il sistema Sii-Mobility. Anche se tipicamente i kit veicolari, sensori e attuatori sono e possono essere connessi direttamente con altre centrali, una connessione diretta con Sii-Mobility permette il controllo diretto di certe situazioni di controllo e attuazione. Per esempio la rilevazione di dati specifici con sensori evoluti, e l'attuazione di politiche di controllo e segnalamento a più elevato livello di controllo.

4.1 Protocollo di comunicazione tra Centrale Direzionamento e SII (API07) (UNIFI)

La centrale di direzionamento è nella buona sostanza (come il sistema di messaggio variabile) una applicazione Android su un dispositivo Android gemellato con un Arduino che gestisce le parti IO.
La centrale di direzionamento va direttamente a leggere su certe API lo stato che dovrebbe acquisire. In questo modo è il server su *Sensors Server Manager* che espone il servizio, la App semplicemente va a leggere con modalità *polling* per vedere se vi sono dei cambi di stato programmati o da eseguire in immediatamente.
Si veda la soluzione fra *SmartDS*, *Dashboard* e *Sensor Server Manager*. Tale soluzione permette di scrivere dentro il SSM lo stato al quale un attuatore deve conformarsi. È l'attuatore stesso che chiama il SSM per leggere se vi sono informazioni/comandi per cambiare stato. Questo permette di avere attuatori anche in DHCP e anche su reti mobili. Il protocollo dovrebbe essere autenticato.



Descrizione Tool/algoritmo	Protocollo di comunicazione fra la Centrale

	Direzionamento ed il sistema SII. Il sistema di gestione Sensor Server Manager deve presentare delle API per: accettare comandi dall'esterno programmando l'ora del cambio direzione oppure subito, fornire dati relativi allo stato ed ai sensori ed allo storico.	
Dati primari in ingresso	Identificativi dei sensori, intervalli di tempo	
Dati prodotti in Uscita	Dati dei sensori di traffico, ZTL, parcheggi	
Principi (metodo, basi teoriche, etc.)	La centrale legge lo stato che deve acquisire o che altro dal Sensor Server Manager	
Casi di test (presenti/assenti)	nessuno	
Posizione casi di test	n.a.	
Principali problemi non risolti	nessuno	
Principali requisiti pendenti	nessuno	
Aspetti Tecnologici		
Stato (proposto/approvato)	proposto	
Implementato/non implementato	non implementato	
Stato implementazione, percentuale	0%	
Eseguibile/libreria/web app	Mobile app, lato server come PHP	
Single thread/Multithread	Multithread	
Linguaggio di sviluppo (java, Php, ETL)	Java su mobile, PhP su server	
Piattaforme supportate	Android, Linux	
Posizione del codice sorgente	repository di progetto	
Indirizzo/i web services (se presenti) con indicazione credenziali accesso (se necessarie)	da definire	
Indirizzo/i accesso via web (se presenti)	da definire	
Nomi tool/moduli usati (aggiungere una riga per ogni tool/modulo usato)	Interfacce API usate	Modello di comunicazione e formato
n.a.	n.a.	HTTP protocol
Formati usati (aggiungere una riga per ogni formato usato)	Condiviso con tool/modulo	Nome formato o riferimento a sezione definizione
JSON	n.a.	n.a.
Protocolli usati (aggiungere una riga per ogni protocollo usato)	Condiviso con tool/modulo	Nome protocollo o riferimento a sezione definizione
HTTP	n.a.	n.a.
Nomi database usati (aggiungere una riga per ogni db usato)	Descrizione	
	n.a.	
Tipo interfaccia utente (web/applicazione)	Modello sviluppo, linguaggio	Libreria usata per UI
n.a.	n.a.	n.a.
Libreria usata (aggiungere una riga per ogni libreria usata)	Nome e versione usata	Licenza: GPL, LGPL, PEK, proprietaria, commerciale, etc.
n.a.	n.a.	n.a.

Il Sensor Server and Manager esporrà delle API per il controllo degli attuatori. Le API saranno utilizzabili tramite protocollo REST su https autenticato. I singoli attuatori saranno identificati con un serviceUri e saranno ricercabili tramite API di ricerca servizi. Le API REST permetteranno di conoscere le caratteristiche di un sensore/attuatore e impostare un valore per una proprietà dell'attuatore eventualmente indicando l'ora alla quale la proprietà andrà impostata.

API REST:

- GET /api/v1/iot/state?serviceUri={service uri}
  dato un identificatore di un sensore/attuatore restituisce lo stato e le caratteristiche del sensore/attuatore descritto in formato JSON (nel caso ci siano impostazioni ritardate di una proprietà saranno elencate con il relativo id)
- POST /api/v1/iot/update
  prende in POST un oggetto JSON contenente: serviceURI dell'attuatore, proprietà da modificare, valore da impostare, eventuale data e ora a cui la proprietà andrà modificata.
- GET /api/v1/iot/delayed?id={id operazione delayed}
  operazione interna chiamata dal DISCES per effettuare la chiamata delayed

Il sistema verrà realizzato usando un context broker come Orion che permetta di comandare i singoli attuatori usando un protocollo standard come MQTT o CoAP. Per gestire le impostazioni ritardate di una proprietà si potrà usare lo scheduler DISCES per effettuare una chiamata REST per impostare effettivamente la proprietà.

4.2 API per l'integrazione di nuove sorgenti dati (API08) (MIZAR)

L'API descritta di seguito permette di integrare dati provenienti da nuovi sorgenti (e.g. FCD – Floating Car Data) e fornire dati relativi allo stato ed ai sensori ed allo storico provenienti dal Gestore della mobilità.
Queste API sono esportate dal Supervisore verso Sii-Mobility e includono dati del Supervisore attuale più i dati che saranno sviluppati dai punti 5.6.3.

4.2.1 Flusso di dati

I dati primari in ingresso saranno l'identificativo del sensore e misure quali, ad esempio, volumi, velocità, occupazione e classificazione veicoli. L'API riceve dal sensore tali misure e le inoltra al sistema SII.

4.2.2 Caratteristiche dell'API

Alcuni aspetti tecnologici dell'API sono espressi nella tabella qui di seguito:

Eseguibile/libreria/web app:	Web app
Single thread/Multithread:	Multithread
Linguaggio di sviluppo (java, Php, ETL):	C#
Piattaforme supportate:	Windows
Posizione del codice sorgente:	da definire
Indirizzo/i web services (se presenti):	http://IPserver/rest/api/v1/ per il progetto sarà configurato un nome utente (SII_test) e password (511_t35t)

4.2.3 Interfacce e standard

Il formato usato per la tipologia di dati scambiati in questione è JSON/XML.
Il modello di comunicazione e il protocollo usato sono il metodo HTTP, mentre non è presente nessun tipo di interfaccia utente

La libreria ha una licenza proprietaria, che è però ad uso gratuito per il progetto SII.

4.2.4 Protocollo Supervisore-SII, API: API08 (MIZAR)

Per ogni oggetto che produce misure (regolatori semaforici, stazioni di misura, sensori) è possibile impostare delle misure.
Per impostare le misure è necessario invocare il metodo `POST` al seguente URL:

```
http://domain/rest/v1/objecttypes/{GUID}/measures
```

con il valore appropriato nei campi *{GUID}*
In caso di successo, il metodo restituisce il codice HTTP `200 OK`.

Esempio di richiesta:

```
POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "Measures": [
    {
      "MeasureTimeUTC": "2016-06-08T15:50:00Z",
      "Sensors": [
        {
          "Data": [
            {
              "Key": "VOLUME",
              "Value": "20"
            },
            {
              "Key": "SPEED",
              "Value": "50"
            }
          ]
        },
        {
          "Data": [
            {
              "Key": "VOLUME",
              "Value": "30"
            },
            {
              "Key": "SPEED",
              "Value": "60"
            }
          ]
        },
        {
          "Data": [
            {
              "Key": "VOLUME",
              "Value": "20"
            },
            {
              "Key": "SPEED",
              "Value": "50"
            }
```

```
            ]
          },
          {
            "Data": [
              {
                "Key": "VOLUME",
                "Value": "30"
              },
              {
                "Key": "SPEED",
                "Value": "60"
              }
            ]
          }
        ]
      }
    ]
}' 'http://192.168.55.75:8088/rest/v1/detectionunits/6f796196-5ca7-4546-ad84-
17ae79224e4b/measures?token=1234'
```

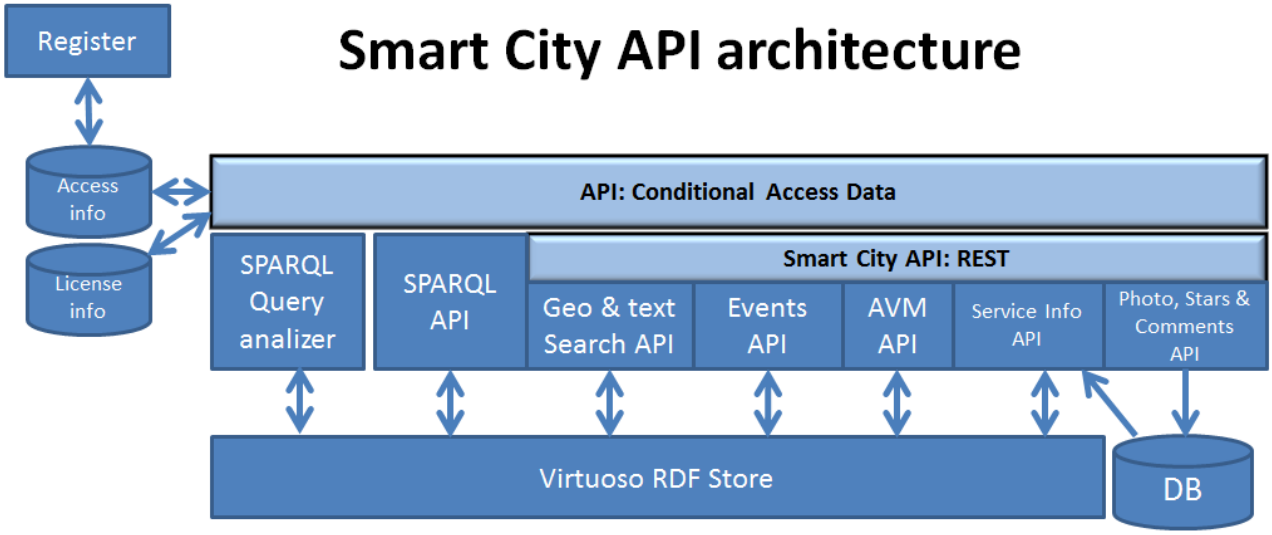
Esempio di risposta:

```
RESPONSE CODE: 200
RESPONSE HEADER:
{
  "access-control-allow-origin": "*",
  "date": "Wed, 08 Jun 2016 15:56:31 GMT",
  "server": "Microsoft-HTTPAPI/2.0",
  "content-length": "0",
  "content-type": null

}
```

5 API per IN/OUT con mobile e Totem

Nei seguenti paragrafi verranno descritte le specifiche delle API per l'accesso a dati ed informazioni, sia storici che in tempo reale, da applicazioni web, fisse e mobili verso la piattaforma Sii-Mobility.



5.1 API REST di Knowledge Base su base Geolocation (API09) (UNIFI)

5.1.1 Protocollo esposto

Il modulo espone una interfaccia REST su protocollo HTTP per l'accesso ai dati presenti nella KnowledgeBase.

The APIs are accessible mainly via HTTP GET requests at specific URLs with specific parameters provided in the query string. Query parameters are case sensitive (e.g. use maxDists and not maxdists). The "format" parameter in many cases can be equal to html or json (and json is assumed if it is not provided) to provide the result as machine readable JSON or as a human readable web page. Most APIs accept an optional user identifier (uid) that should be provided to identify the device (and indirectly the user) making the requests. The uid should be a unique identifier, currently the uid is generated as a SHA256 hash of the device uuid generated by cordova device plugin (see https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-device/). The history of user requests is used to produce suggestions and user engagements.
The requests to the API are CORS enabled thus APIs can be used cross domain from other sites.

The *multimedia* property provided by some APIs contain a URL to a multimedia file that is in many cases no more available, a caching service for images was setup because the images were too large for mobiles and now using this cache is currently the only way to retrieve these images. See the multimedia caching API to see how to use these images. Unfortunately the cache was realized only for images and thus pdf files and audio files are no more available.

Note: For APIs supporting format "html" the following additional optional parameters may be used:

- *map*: to set the type of map to be used ("satellite", "streets" or "grayscale");

- *controls*: to control the appearance of the controls on the left and right of the page, it can be "hidden" or "false" to be not visible or "collapsed" do be collapsed;
- *info*: to control the appearance of the info tab on the lower left of the page, it can be "hidden" or "false" to be not visible or "collapsed" to be collapsed;

Service search near GPS position

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are near a given GPS position. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field. It can also be used to find services that have a WKT spatial description that contains a specific GPS position.	
Parameters:	
selection	<latitude>;<longitude> of the GPS position
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.
text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxDists	maximum distance from the GPS position of the services to be retrieved, expressed in Km (0.1 is used if parameter is missing) if it is equal to "inside" it searches for services with a WKT geometry that contains the specified GPS position (e.g a park)
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. "it", "en", "fr", "de", "es") to be used for returned descriptions if available in multiple languages. Currently for languages other than "it" and "en" it returns "en" descriptions. (if parameter is missing "en" is assumed)
geometry	true/false, if true it returns a "hasGeometry" property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing "false" is assumed)
uid	optional user identifier
format	html or json
Results:	
when format = "html" it produces a web page showing the results of the query, like the following:	

when format = “json” it returns the services split in three sections (BusStops , SensorSites, Services). Each section is provided as GeoJSON “FeatureCollection”, the results are sorted by distance, additionally in each section the “fullCount” property reports the full number of results available matching the query, for example:


```
{
    "BusStops": {
        "fullCount": 26,
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.249078, 43.775326]
            },
            "type": "Feature",
            "properties": {
                "name": "Stazione Abside S.M.N.",
                "typeLabel": "Fermata",
                "tipo": "fermata",
                "serviceType": "TransferServiceAndRenting_BusStop",
                "busLines": "13 - 36 - 37",
                "serviceUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Stop_FM0328_5",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "photoThumbs": []
            },
            "id": 1
        }, … ]
    },
    "SensorSites": {
        "fullCount": 3,
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.24982, 43.77505]
            },
            "type": "Feature",
            "properties": {
                "name": "FI055ZTL00101",
                "tipo": "sensore",
                "typeLabel": "Sensore",
                "serviceType": "TransferServiceAndRenting_SensorSite",
                "serviceUri": "http://www.disit.org/km4city/resource/FI055ZTL00101",
                "photoThumbs": []
            },
            "id": 1
        }, … ]
    },
    "Services": {
```

```
        "fullCount": 84,
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                    "type": "Point",
                    "coordinates": [11.249473, 43.775867]
            },
            "type": "Feature",
            "properties": {
                    "name": "Parcheggio Stazione Firenze S.M.N.",
                    "tipo": "Parcheggio_auto",
                    "typeLabel": "Parcheggio auto",
                    "serviceType": "TransferServiceAndRenting_Car_park",
                    "serviceUri": "http://www.disit.org/km4city/resource/CarParkStazioneFirenzeS.M.N.",
                    "multimedia": ""
            },
            "id": 1
        }, … ]
    }
}
```


Examples:

- **Search for Accommodation, bus stop, sensor site or car park within 200m**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.7756;11.2490&categories=Accommodation;BusStop;SensorSite;Car_park&maxResults=10&maxDists=0.2&lang=it&format=json
- **Any entertainment service within 200m**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.7756;11.2490&categories=Entertainment&maxResults=20&maxDists=0.2&lang=it&format=json&geometry=true


```
{
  "Services": {
      "fullCount": 8,
      "type": "FeatureCollection",
      "features": [{
          "geometry": {
              "type": "Point",
              "coordinates": [11.24851, 43.77566]
          },
          "type": "Feature",
          "properties": {
              "name": "Giardino di piazza della Stazione",
              "tipo": "Aree_verdi",
              "typeLabel": "Aree verdi",
              "serviceType": "Entertainment_Green_areas",
              "hasGeometry": true,
              "serviceUri": "http://www.disit.org/km4city/resource/e62bc5f14bd412db00fcdcd6f9506857",
              "multimedia": ""
          },
          "id": 1
      }, {
          "geometry": {
              "type": "Point",
              "coordinates": [11.249722, 43.77561]
          },
          "type": "Feature",
          "properties": {
              "name": "Spartitraffico di piazza della Stazione",
              "tipo": "Aree_verdi",
              "typeLabel": "Aree verdi",
              "serviceType": "Entertainment_Green_areas",
              "hasGeometry": true,
              "serviceUri": "http://www.disit.org/km4city/resource/37a2cdb39f7c8e86c55990b4f3125256",
              "multimedia": ""
          },
          "id": 2
      }, {
          "geometry": {
              "type": "Point",
              "coordinates": [11.249624, 43.77658]
          },
```

```
            "type": "Feature",
            "properties": {
                "name": "SCUDERIA DEL BEJ DI SIVORI GIOVAN BATTISTA E C. - S.A.S.",
                "tipo": "Societa_sportive",
                "typeLabel": "Societa' sportive",
                "serviceType": "Entertainment_Sports_clubs",
                "hasGeometry": false,
                "serviceUri": "http://www.disit.org/km4city/resource/0a98b2ea221ba49356c20bed3c7b8f38",
                "multimedia": ""
            },
            "id": 3
        }]
    }
}
```


- **Any service whose geometry contains GPS position**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.7754;11.2494&categories=Service&maxResults=20&maxDists=inside&lang=it&format=json&geometry=true


```
{
  "Services": {
      "fullCount": 6,
      "type": "FeatureCollection",
      "features": [{
          "geometry": {
                  "type": "Point",
                  "coordinates": [11.249722, 43.77561]
          },
          "type": "Feature",
          "properties": {
                  "name": "Spartitraffico di piazza della Stazione",
                  "tipo": "Aree_verdi",
                  "typeLabel": "Aree verdi",
                  "serviceType": "Entertainment_Green_areas",
                  "hasGeometry": true,
                  "serviceUri": "http://www.disit.org/km4city/resource/37a2cdb39f7c8e86c55990b4f3125256",
                  "multimedia": ""
          },
          "id": 1
      }]
  }
}
```


- **Accomodation within 1Km with “casa di dante” in a textual description**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.7754;11.2494&categories=Accommodation&maxResults=2&maxDists=1&lang=it&format=json&text=casa%20di%20dante


```
{
    "Services": {
        "fullCount": 2,
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                    "type": "Point",
                    "coordinates": [11.256365, 43.771023]
            },
            "type": "Feature",
            "properties": {
                    "name": "CASA_DI_DANTE",
                    "tipo": "Affittacamere",
                    "typeLabel": "Affittacamere",
                    "serviceType": "Accommodation_Boarding_house",
                    "serviceUri": "http://www.disit.org/km4city/resource/c1cd4b12fabce2d9b3a1527fd5a7be79",
                    "multimedia": ""
            },
            "id": 1
        }, {
            "geometry": {
                    "type": "Point",
                    "coordinates": [11.256365, 43.771023]
            },
```

```
                "type": "Feature",
                "properties": {
                        "name": "CASA_DI_DANTE",
                        "tipo": "Affittacamere",
                        "typeLabel": "Affittacamere",
                        "serviceType": "Accommodation_Boarding_house",
                        "serviceUri": "http://www.disit.org/km4city/resource/8cb399e95b39475a9838eefa8ff5e683",
                        "multimedia": ""
                },
                "id": 2
            }]
        }
    }
```


Notes:

Service search near a service

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are near a given service identified by its serviceUri. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field. It can also be used to find services that have a WKT spatial description that contains a specific GPS position.	
Parameters:	
selection	serviceUri (http://...) of the service
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.
text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxDists	maximum distance from the GPS position of the services to be retrieved, expressed in Km (0.1 is used if parameter is missing) if it is equal to “inside” it searches for services with a WKT geometry that contains the specified GPS position (e.g a park)
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json
Results: The same format as Near a GPS position	
Examples: **Wine and food in 100m from Palazzo Vecchio** http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=http://www.disit.org/km4city/resource/7ad6d2d3be461b1f0514956279c00eab&categories=WineAndFood&maxResults=10&lang=it&for	

mat=json
Notes:

Service search within a GPS area

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are inside a rectangular area. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field.	
Parameters:	
selection	<lat1>;<lng1>;<lat2>;<lng2> are two GPS coordinates describing a rectangle where (lat1,lng1) is a south west point and (lat2, lng2) is a north east point.
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.
text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json
Results:	
the results format is the same as the previous API, reresults are sorted by distance from the center of the rectangle	
Examples:	
• **Search for an accommodation, bus stop, sensor site or car park in a GPS area** http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.7741;11.2453;43.7768;11.2515&categories=Accommodation;BusStop;SensorSite;Car_park&maxResults=10&lang=it&format=json	

Notes:

Service search within a WKT described area

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are inside a geographic region described using the Well Known Text (WKT) format. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field.	
Parameters:	
selection	wkt:<WKT string> describes the geographic region as WKT string.
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.
text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json
Results:	
the results format is the same as the previous API, in this case the sort order of results is undefined.	
Examples:	
• to write a WKT string the following service can be used https://arthur-e.github.io/Wicket/sandbox-gmaps3.html	

- **Search for any service in a WKT area**
  POLYGON((11.25539 43.77339,11.25608 43.77348,11.25706 43.77362,11.25759 43.77328,11.25755 43.77291,11.25675 43.77260,11.25536 43.77270,11.25539 43.77339))
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=wkt:POLYGON((11.25539%2043.77339,11.25608%2043.77348,11.25706%2043.77362,11.25759%2043.77328,11.25755%2043.77291,11.25675%2043.77260,11.25536%2043.77270,11.25539%2043.77339))&categories=Service&maxResults=0&lang=it&format=html

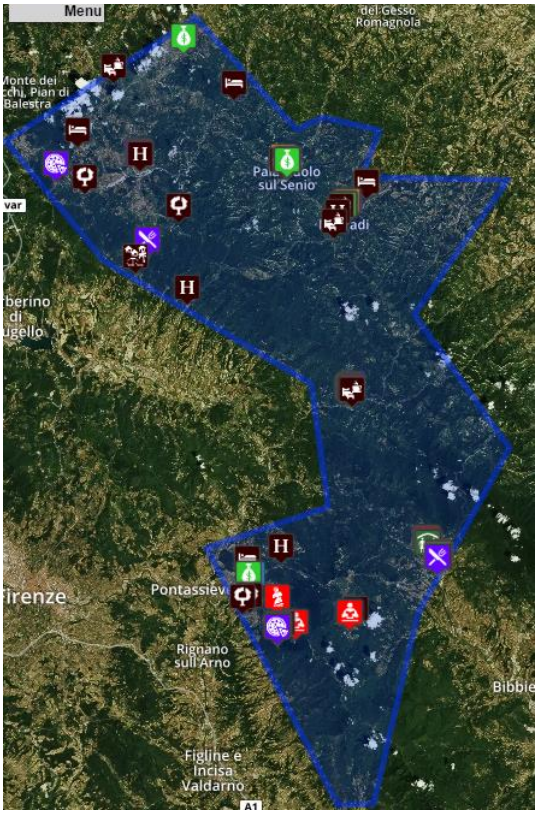
Bugs:
the html version may not consider all the parameters

Service search within a stored WKT described area

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are inside a geographic region described using the Well Known Text (WKT) format, by referring to the WKT with and identifier provided when the WKT is stored. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field. The list of available geometries can be retrived from servicemap in the "Search Area" selection box (with Search Range "specific area"). New geometries can be provided using the http://www.km4city.org/wkt web service which can store a WKT from a shp file or providing directly the WKT string.	
Parameters:	
selection	*geo:<geo_id>* where <geo_id> identifies a WKT string stored on the server.
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.
text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. "it", "en", "fr", "de", "es") to be used for returned descriptions if available in multiple languages. Currently for languages other than "it" and "en" it returns "en" descriptions. (if parameter is missing "en" is assumed)
geometry	true/false, if true it returns a "hasGeometry" property for each service stating if

	the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing "false" is assumed)
uid	optional user identifier
format	html or json
Results:	
the results format is the same as the previous API	
Examples:	

- **Search for any service in a WKT area**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=geo:ritmi_01&categories=Service&maxResults=100&lang=it&format=html



Bugs:
the html version may not consider all the parameters

Service search by municipality

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services that are in a specific municipality. The services can be filtered as belonging to specific categories (e.g. Accomodation, Hotel, Restaurant etc), or having specific words in any textual field.	
Parameters:	
selection	name of the municipality like FIRENZE, EMPOLI, PISA possibly with prefix "COMUNE di "
categories	the list of categories of the services to be retrieved, if omitted all kinds of services are returned. It can contain macro categories or categories, if a macro category is specified all categories in the macro category are used. The complete list of categories and macro categories can be retrieved on servicemap.disit.org.

text	words in this parameter are used to retrieve services that contain all these words in any textual description associated with the service.
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json
Results:	
the results format is the same as the previous API	
Examples:	

- **Search for any Entertainment service in the municipality of FIRENZE**
  http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=COMUNE%20di%20FIRENZE&categories=Entertainment&maxResults=100&lang=it&format=html

Bugs:
the html version accepts only a selection with prefix “COMUNE di ”

Full text search

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the geolocated entities (not only services) that match with a list of keywords. The results can be possibly filtered to be within a specified distance from a GPS position, or within a rectangular area or inside a WKT geolocated area.	
Parameters:	
search	the keywords separated with spaces that have to match with any textual

	description associated with an entity.
selection	optional “<lat>;<lng>” with a GPS position or “<lat1>;<lng1>;<lat2>;<lng2>” for a rectangular area or “wkt:<WKT_string>” or “geo:<geoid>” for a geographic area described as Well Known Text (see other APIs for more details)
maxDists	optional maximum distance from the GPS position of the entities to be retrieved, expressed in Km
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json

Results:

the results format is a GeoJSON “FeatureCollection” with the matching entities, additionally the “fullCount” property provides the full count of results available matching the query. For each “Feature” a minimal set of properties are provided.

Examples:

Search for any geolocated entity matching “via nave”

http://servicemap.disit.org/WebAppGrafo/api/v1/?search=via%20nave&maxResults=10&lang=en&format=json


```
{
   "fullCount": 558,
   "type": "FeatureCollection",
   "features": [{
        "geometry": {
             "type": "Point",
             "coordinates": [11.315443, 43.756367]
        },
        "type": "Feature",
        "properties": {
             "serviceUri": "http://www.disit.org/km4city/resource/e96076db6e4e2b8b43fb660579eb4de8",
             "name": "PICCIOLI DANIELE",
             "tipo": "servizio",
             "photoThumbs": [],
             "multimedia": "",
             "civic": "",
             "serviceType": "CulturalActivity_Theatre",
             "typeLabel": "Theatre"
        },
        "id": 1
   }, … {
        "geometry": {
             "type": "Point",
             "coordinates": [10.898357, 43.729973]
        },
        "type": "Feature",
        "properties": {
             "serviceUri": "http://www.disit.org/km4city/resource/RT04801406596TO",
             "name": "VIA NAVE DI VITIANA",
             "tipo": "servizio",
             "photoThumbs": [],
             "multimedia": "",
             "civic": "1",
             "serviceType": "",
             "typeLabel": "Road"
        },
        "id": 8
```

```
        }, … ]
    }
```


Bugs:
the html version may not consider all the parameters

Address/POI search by text

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/location
it allows to retrieve a list of street addresses and service names based on a text search. The search may be filteres	
Parameters:	
search	a text with the words to be found in the names of the streets, civic number, municipality names and service names
searchMode	optional can be AND or ANDOR (default ANDOR), indicates if all or any word of the query need to match
position	optional "<lat>;<lng>" with a GPS position.
maxDists	optional maximum distance in km from *position* for searching the text (if omitted 5 km is assumed)
excludePOI	optional true or false (assumed false if missing), if true the search is performed only on street names, civic numbers and municipalities
maxResults	optional maximum number of results provided (default 10)
uid	optional user identifier
format	optional format of results, only json
Results:	
A GeoJSON FeatureCollection object with the matching objects	
Examples:	

http://servicemap.disit.org/WebAppGrafo/api/v1/location/?search=via%20calzaioli&format=json


```
{
    "type": "FeatureCollection",
    "count": 1263003,
    "features": [{
        "geometry": {
            "type": "Point",
            "coordinates": [11.255358, 43.77244]
        },
        "type": "Feature",
        "properties": {
            "serviceUri": "http://www.disit.org/km4city/resource/817ccce02f5aa0ef8c34744c4c25bcc6",
            "serviceType": "CulturalActivity Monument_location",
            "name": "Via dei Calzaiuoli",
            "city": "FIRENZE",
            "id": 1
        }
    }, {
        "geometry": {
            "type": "Point",
            "coordinates": [11.309124, 43.835896]
        },
        "type": "Feature",
        "properties": {
            "serviceUri": "http://www.disit.org/km4city/resource/f9fbe7453f63d3063cd4e33c43c1f5eb",
            "serviceType": "CulturalActivity Printing_and_services",
            "name": "FUTURE GRAPHIC DI CAMAIOLI GIUSTI VERONICA",
            "city": "FIESOLE - CALDINE",
            "id": 2
        }
    }, … {
        "geometry": {
            "type": "Point",
            "coordinates": [11.255217, 43.77035]
        },
        "type": "Feature",
```

```
        "properties": {
            "serviceUri": "http://www.disit.org/km4city/resource/RT048017002601CV",
            "serviceType": "StreetNumber",
            "address": "VIA DEI CALZAIOLI",
            "civic": "15 R",
            "city": "FIRENZE",
            "id": 5
        }
    }, …]
}
```


http://servicemap.disit.org/WebAppGrafo/api/v1/location/?search=via%20calzaioli&excludePOI=true&format=json


```
{
    "type": "FeatureCollection",
    "count": 1261873,
    "features": [{
        "geometry": {
            "type": "Point",
            "coordinates": [11.255217, 43.77035]
        },
        "type": "Feature",
        "properties": {
            "serviceUri": "http://www.disit.org/km4city/resource/RT048017002601CV",
            "serviceType": "StreetNumber",
            "address": "VIA DEI CALZAIOLI",
            "civic": "15 R",
            "city": "FIRENZE",
            "id": 1
        }
    }, …]
}
```


Bugs:

Event search

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/events/
it allows to retrieve the geolocated events in a given temporal range (day, week or month). The results can be possibly filtered to be within a specified distance from a GPS position, or within a rectangular area or inside a WKT described geographic area.	
Parameters:	
range	time range for the events to be retrieved, it can be ‘day’ for the events of the day of the request, ‘week’ for the events in the next 7 days or ‘month’ for the events in the next 30 days (if omitted ‘day’ is assumed).
selection	optional “<lat>;<lng>” with a GPS position or “<lat1>;<lng1>;<lat2>;<lng2>” for a rectangular area or “wkt:<WKT_string>” or “geo:<geoid>” for a geographic area described as Well Known Text (see other APIs for more details).
maxDists	optional maximum distance from the GPS position of the events to be retrieved, expressed in Km.
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
uid	optional user identifier
format	only json
Results:	
the results format is a GeoJSON “FeatureCollection” with the matching events. For each “Feature” a set of properties is provided.	
Examples:	

Search for events of today

http://servicemap.disit.org/WebAppGrafo/api/v1/events/?range=day&format=json


```
{
   "Event": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.251058, 43.769848]
            },
            "type": "Feature",
            "properties": {
                "serviceUri": "http://www.disit.org/km4city/resource/Event_18794_973b96efaf3f99f1b70af19cda4e3bf4",
                "name": "Tra arte e moda",
                "tipo": "event",
                "place": "MUSEO SALVATORE FERRAGAMO ",
                "startDate": "2016-05-19",
                "startTime": "10.00 -19.30; chiuso 1/1, 01/05, 15/08 e 25/12",
                "endDate": "2017-04-07",
                "freeEvent": "NO",
                "address": "PIAZZA DI SANTA TRINITA",
                "civic": "2",
                "categoryIT": "Mostre",
                "price": "6 (incluso museo/including museum)",
                "phone": "055 3562466",
                "descriptionIT": "La mostra riflette il complesso rapporto fra arte e moda prendendo spunto dalla storia di
Salvatore Ferragamo che si ispirò alle avanguardie artistiche del '900 per realizzare le sue creazioni. ",
                "website": "www.ferragamomuseo.com/museo",
                "serviceType": "Event"
            },
            "id": 1
        }, ... ]
}}
```


Bugs:
problems with duplicated events and with accented chars (solved for new events, still present for old events).

Address and geometry search by GPS

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/location
it allows to retrieve the complete address (municipality, street and civic number) given the GPS position. It may also provide a list of services or public transport lines intersecting with the provided GPS position.	
Parameters:	
position	"<lat>;<lng>" with a GPS position.
intersectGeom	true or false (assumed false if missing), if true it reports all the services and public transportation lines that have a geometry intersecting with the provided GPS position.
uid	optional user identifier
format	only json
Results:	
A JSON object with properties: • *address*: the street name. • *number*: the civic number. • *addressUri*: the URI identifying the civic number in the road graph. • *municipality*: the estimated municipality (it may not work properly on the municipalities borders) • *municipalityUri*: the URI identifying the municipality in the road graph.	

- *intersect*: array of objects with properties:
  - *name*: name of the intersecting service or public transport line.
  - *uri*: URI of the intersecting service or public transport line.
  - *class*: URI representing the class
  - *type*: type of geometry intersecting the GPS position, can be *lineString* or *Polygon*
  - *routeType:* type of route can be Bus, LightRail, Ferry, Train
  - *agency*: name of the agency providing the service
  - *direction*: direction of the line
  - *distance*: distance of the GPS position with the intersecting geometry

note: address, number and addressUri may be not present if the GPS position is outside a populated place.

Examples:

http://servicemap.disit.org/WebAppGrafo/api/v1/location/?position=43.7741;11.2505&format=json


```
{
    "address": "VIA PANZANI",
    "municipality": "FIRENZE",
    "number": "17\/A",
    "addressUri": "http:\/\/www.disit.org\/km4city\/resource\/RT048017023351CV",
    "municipalityUri": "http:\/\/www.disit.org\/km4city\/resource\/048017"
}
```


http://servicemap.disit.org/WebAppGrafo/api/v1/location/?position=43.7741;11.2505&intersectGeom=true&format=json


```
{
    "address": "VIA PANZANI",
    "municipality": "FIRENZE",
    "number": "17\/A",
    "addressUri": "http:\/\/www.disit.org\/km4city\/resource\/RT048017023351CV",
    "municipalityUri": "http:\/\/www.disit.org\/km4city\/resource\/048017",
    "intersect": [{
        "distance": 1.2392468323025842E-4,
        "name": "Firenze Card",
        "class": "http:\/\/www.disit.org\/km4city\/schema#Tourist_trail",
        "type": "LineString",
        "uri": "http:\/\/www.disit.org\/km4city\/resource\/2a93692aa1eb7d680d9b4e0da668b408"
    }, {
        "distance": 3.1448272583131523E-4,
        "routeType": "Bus",
        "direction": "Salviatino",
        "name": "11",
        "agency": "Ataf&Linea",
        "class": "http:\/\/vocab.gtfs.org\/terms#Route",
        "type": "LineString",
        "uri": "http:\/\/www.disit.org\/km4city\/resource\/Bus_ataflinea_Trip_1923_3337883"
    }, ... ]
}
```


Bugs:

Service info

The Service info API allows getting information about a specific service or entity identified by a serviceURI property returned from the search APIs. Information can be get using the following REST API but also using the Linked Data paradigm using the serviceURI itself.

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve information about a service using its serviceUri. It can return an html representation (format=”html”) or a machine readable representation (format=”json”)	
Parameters:	
serviceUri	the serviceUri of the service

lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
realtime	true or false (if omitted true is implied) indicates if the last value of the time varying properties should be provided in the result or not.
uid	optional user identifier
format	html or json

Results:

if format is “html” provides a visual representation of the service on the map. If format is json the API provides a GeoJSON description of the service with the main properties (name, address, city, type, etc.) and possibly some time varying properties for some kinds of services (traffic sensors, car park sensors, etc.).
The following is an example for a SensorSite (traffic sensor)


```
{
    "Sensor": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.2702, 43.77467]
            },
            "type": "Feature",
            "properties": {
                "name": "FI055ZTL02001",
                "typeLabel": "Sensor",
                "serviceType": "TransferServiceAndRenting_SensorSite",
                "serviceUri": "http://www.disit.org/km4city/resource/FI055ZTL02001",
                "municipality": "FIRENZE",
                "address": "VIA DELLA MATTONAIA",
                "photos": [],
                "photoThumbs": [],
                "photoOrigs": [],
                "avgStars": 0.0,
                "starsCount": 0,
                "comments": []
            },
            "id": 1
        }]
    },
    "realtime": {
        "head": {
            "sensor": ["FI055ZTL02001"],
            "vars": ["avgDistance", "avgTime", "occupancy", "concentration", "vehicleFlow", "averageSpeed", "thresholdPerc",
"speedPercentile", "instantTime"]
        },
        "results": {
            "bindings": [{
                "avgDistance": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "avgTime": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "occupancy": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "concentration": {
                    "type": "literal",
                    "value": "0.0"
                },
                "vehicleFlow": {
                    "type": "literal",
                    "value": "42.0"
                },
```

```
                    "averageSpeed": {
                        "type": "literal",
                        "value": "0.0"
                    },
                    "thresholdPerc": {
                        "type": "literal",
                        "value": "Not Available"
                    },
                    "speedPercentile": {
                        "type": "literal",
                        "value": "Not Available"
                    },
                    "instantTime": {
                        "type": "literal",
                        "value": "2017-01-17T16:32:00+01:00"
                    }
                }]
            }
        }
}
```


Examples:

- see the following sections for details on the various kinds of services

Bugs:

Generic service

For generic services (e.g. Accommodations, Restaurants, etc.) the following properties are provided in the GeoJSON properties:

- *serviceUri:* an URI identifying the service globally
- *name*: name of the service
- *typeLabel*: label associated with the type of service in the language provided with the lang parameter
- *serviceType:* a string containing "<MacroClass>_<ServiceType>"
- *city*, *address*, *civic*; municipality, address and civic number of the service
- *phone, fax, website, email:* phone, fax, website, email of the service
- *note:* notes associated with the service
- *description, description2:* two descriptions of the service, one in Italian and the other in English if available.
- *multimedia:* an url to a multimedia resource
- *linkDBpedia:* array of urls to dbpedia resources
- *photo, photoThumbs, photoOrigs:* array of urls to photos, thumbnails and original photos provided using the photo API.
- *wktGeometry:* a Well Known Text geometry associated with the service
- *avgStars:* average number of stars provided with the stars API
- *starsCount:* number of ratings provided by users.
- *comments:* array of comments on the service provided by users using the comments API

the following is an example:


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.361144, 44.00213]
                },
```

```
                "type": "Feature",
                "properties": {
                    "name": "IL_BRONCO",
                    "typeLabel": "Boarding house",
                    "serviceType": "Accommodation_Boarding_house",
                    "phone": "0558430207",
                    "fax": "",
                    "website": "www.ristoranteilbronco.it",
                    "province": "FI",
                    "city": "SCARPERIA",
                    "cap": "50038",
                    "email": "info@ristoranteilbronco.it",
                    "linkDBpedia": [],
                    "note": "",
                    "description": "",
                    "description2": "",
                    "multimedia": "",
                    "serviceUri": "http://www.disit.org/km4city/resource/9fc542b468509b922aeb833273dd40d0",
                    "address": "VIA DANTE",
                    "civic": "95",
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                },
                "id": 1
            }
        ]
    }
}
```


Event

http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/Event_18794_973b96efaf3f99f1b70af19cda4e3bf4


```
{
    "Event": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.251058, 43.769848]
            },
            "type": "Feature",
            "properties": {
                "serviceUri": "http://www.disit.org/km4city/resource/Event_18794_973b96efaf3f99f1b70af19cda4e3bf4",
                "name": "Tra arte e moda",
                "name2": "Accross art and fashion",
                "website": "www.ferragamomuseo.com/museo",
                "address": "PIAZZA DI SANTA TRINITA",
                "number": "2",
                "province": "FI",
                "city": "Firenze",
                "note": "",
                "description": "La mostra riflette il complesso rapporto fra arte e moda prendendo spunto dalla storia di Salvatore Ferragamo che si ispirò alle avanguardie artistiche del '900 per realizzare le sue creazioni. ",
                "description2": "The exhibition reflects the complex relationship between art and fashion starting from the the story of Salvatore Ferragamo who realized his creations inspired by the avant-garde art of the ' 900.  ",
                "startDate": "2016-05-19T00:00:00+02:00",
                "startTime": "10.00 -19.30; chiuso 1/1, 01/05, 15/08 e 25/12",
                "endDate": "2017-04-07T00:00:00+02:00",
                "eventCategory": "Mostre",
                "eventCategory2": "Exhibitions",
                "photos": [],
                "photoThumbs": [],
                "photoOrigs": [],
                "avgStars": 0.0,
                "starsCount": 0,
                "comments": []
            },
            "id": 1
        }]
```

```
    }
}
```


Parking service

http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/RT04801702315PO


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.24947, 43.77587]
                },
                "type": "Feature",
                "properties": {
                    "name": "Garage La Stazione Spa",
                    "typeLabel": "Car park",
                    "serviceType": "TransferServiceAndRenting_Car_park",
                    "phone": "055284784",
                    "fax": "",
                    "website": "",
                    "province": "FI",
                    "city": "FIRENZE",
                    "cap": "50123",
                    "email": "",
                    "linkDBpedia": [],
                    "note": "",
                    "description": "",
                    "description2": "",
                    "multimedia": "",
                    "serviceUri": "http://www.disit.org/km4city/resource/RT04801702315PO",
                    "address": "PIAZZA DELLA STAZIONE",
                    "civic": "3A",
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                },
                "id": 1
            }
        ]
    },
    "realtime": {
        "head": {
            "parkingArea": ["Garage La Stazione Spa"],
            "vars": ["capacity", "freeParkingLots", "occupiedParkingLots", "occupancy", "updating"]
        },
        "results": {
            "bindings": [{
                "capacity": {
                    "value": "617"
                },
                "freeParkingLots": {
                    "value": "322"
                },
                "occupiedParkingLots": {
                    "value": "579"
                },
                "occupancy": {
                    "value": "0.0"
                },
                "status": {
                    "value": "enoughSpacesAvailable"
                },
                "updating": {
                    "value": "2017-01-18T14:25:00+01:00"
                }
            }]
        }
```

```
    }
}
```


Traffic sensor

http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/METRO487


```
{
    "Sensor": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.25003, 43.7747]
            },
            "type": "Feature",
            "properties": {
                "name": "METRO487",
                "typeLabel": "Sensor",
                "serviceType": "TransferServiceAndRenting_SensorSite",
                "serviceUri": "http://www.disit.org/km4city/resource/METRO487",
                "municipality": "FIRENZE",
                "address": "ZTL02 - Preferenziale P.zza Unità-Panzani",
                "photos": [],
                "photoThumbs": [],
                "photoOrigs": [],
                "avgStars": 0.0,
                "starsCount": 0,
                "comments": []
            },
            "id": 1
        }]
    },
    "realtime": {
        "head": {
            "sensor": ["METRO487"],
            "vars": ["avgDistance", "avgTime", "occupancy", "concentration", "vehicleFlow", "averageSpeed", "thresholdPerc", "speedPercentile",
"instantTime"]
        },
        "results": {
            "bindings": [{
                "avgDistance": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "avgTime": {
                    "type": "literal",
                    "value": "2.49806"
                },
                "occupancy": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "concentration": {
                    "type": "literal",
                    "value": "3.522905"
                },
                "vehicleFlow": {
                    "type": "literal",
                    "value": "330.0"
                },
                "averageSpeed": {
                    "type": "literal",
                    "value": "93.6727"
                },
                "thresholdPerc": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "speedPercentile": {
                    "type": "literal",
                    "value": "Not Available"
                },
                "instantTime": {
                    "type": "literal",
```

```
                    "value": "2017-01-18T09:41:00+01:00"
                }
            }]
        }
    }
}
```


Weather Forecast

http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/048017


```
{
    "head": {
        "location": "FIRENZE",
        "vars": ["day", "description", "minTemp", "maxTemp", "instantDateTime"]
    },
    "results": {
        "bindings": [{
            "day": {
                "type": "literal",
                "value": "Mercoledi"
            },
            "description": {
                "type": "literal",
                "value": "nuvoloso"
            },
            "minTemp": {
                "type": "literal",
                "value": "4"
            },
            "maxTemp": {
                "type": "literal",
                "value": "6"
            },
            "instantDateTime": {
                "type": "literal",
                "value": "2017-01-18T09:39:00+01:00"
            }
        }, {
            "day": {
                "type": "literal",
                "value": "Giovedi"
            },
            "description": {
                "type": "literal",
                "value": "coperto"
            },
            "minTemp": {
                "type": "literal",
                "value": "3"
            },
            "maxTemp": {
                "type": "literal",
                "value": "7"
            },
            "instantDateTime": {
                "type": "literal",
                "value": "2017-01-18T09:39:00+01:00"
            }
        }, {
            "day": {
                "type": "literal",
                "value": "Venerdi"
            },
            "description": {
                "type": "literal",
                "value": "poco nuvoloso"
            },
            "minTemp": {
                "type": "literal",
                "value": "1"
            },
            "maxTemp": {
                "type": "literal",
                "value": "7"
```

```
            },
            "instantDateTime": {
                "type": "literal",
                "value": "2017-01-18T09:39:00+01:00"
            }
        }, {
            "day": {
                "type": "literal",
                "value": "Sabato"
            },
            "description": {
                "type": "literal",
                "value": "poco nuvoloso"
            },
            "minTemp": {
                "type": "literal",
                "value": ""
            },
            "maxTemp": {
                "type": "literal",
                "value": ""
            },
            "instantDateTime": {
                "type": "literal",
                "value": "2017-01-18T09:39:00+01:00"
            }
        }, {
            "day": {
                "type": "literal",
                "value": "Domenica"
            },
            "description": {
                "type": "literal",
                "value": "nuvoloso"
            },
            "minTemp": {
                "type": "literal",
                "value": ""
            },
            "maxTemp": {
                "type": "literal",
                "value": ""
            },
            "instantDateTime": {
                "type": "literal",
                "value": "2017-01-18T09:39:00+01:00"
            }
        }]
    }
}
```


Bus stop

http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/Bus_ataflinea_Stop_FM0022_5


```
{
    "BusStop": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.249069, 43.776485]
            },
            "type": "Feature",
            "properties": {
                "name": "Stazione Pensilina",
                "serviceUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Stop_FM0022_5",
                "typeLabel": "BusStop",
                "address": "",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "serviceType": "TransferServiceAndRenting_BusStop",
                "photos": [],
                "photoThumbs": [],
                "photoOrigs": [],
```

```
                "avgStars": 0.0,
                "starsCount": 0,
                "comments": []
            },
            "id": 1
        }]
    },
    "busLines": {
        "head": {
            "busStop": "Stazione Pensilina",
            "vars": ["busLine", "lineUri", "lineDesc"]
        },
        "results": {
            "bindings": [{
                "busLine": {
                    "type": "literal",
                    "value": "1"
                },
                "lineUri": {
                    "type": "literal",
                    "value": "http://www.disit.org/km4city/resource/Bus_ataflinea_Route_122797549"
                },
                "lineDesc": {
                    "type": "literal",
                    "value": "Lapo\/Boccaccio - S.Maria Novella Fs"
                }
            }, {
                "busLine": {
                    "type": "literal",
                    "value": "11"
                },
                "lineUri": {
                    "type": "literal",
                    "value": "http://www.disit.org/km4city/resource/Bus_ataflinea_Route_1073492795"
                },
                "lineDesc": {
                    "type": "literal",
                    "value": "Salviatino-Le Gore"
                }
            }, {
                "busLine": {
                    "type": "literal",
                    "value": "17"
                },
                "lineUri": {
                    "type": "literal",
                    "value": "http://www.disit.org/km4city/resource/Bus_ataflinea_Route_1208385503"
                },
                "lineDesc": {
                    "type": "literal",
                    "value": "Viale Verga-Via Boito\/Cascine"
                }
            }, …
            ]
        }
    },
    "timetable": {
        "head": {
            "vars": ["date", "arrivalTime", "lineName", "lineDesc", "routeName", "trip"]
        },
        "results": {
            "bindings": [{
                "date": {
                    "type": "literal", "value": "2017-01-18"
                },
                "arrivalTime": {
                    "type": "literal", "value": "14:52:00"
                },
                "departureTime": {
                    "type": "literal", "value": "14:52:00"
                },
                "lineName": {
                    "type": "literal", "value": "6"
                },
                "lineDesc": {
                    "type": "literal", "value": "Novelli-Smn-Torregalli"
```

```
                    },
                    "routeName": {
                        "type": "literal", "value": "Ospedale Torre Galli"
                    },
                    "trip": {
                        "type": "uri", "value": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3364525"
                    }
                }, {
                    "date": {
                        "type": "literal", "value": "2017-01-18"
                    },
                    "arrivalTime": {
                        "type": "literal", "value": "14:56:00"
                    },
                    "departureTime": {
                        "type": "literal", "value": "14:56:00"
                    },
                    "lineName": {
                        "type": "literal", "value": "11"
                    },
                    "lineDesc": {
                        "type": "literal", "value": "Salviatino-Le Gore"
                    },
                    "routeName": {
                        "type": "literal", "value": "La Gora"
                    },
                    "trip": {
                        "type": "uri", "value": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3344062"
                    }
                }, …
                ]
            }
        },
        "realtime": {
        }
}
```


Fuel Station

http://www.disit.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/Fuel_station_01a234db6235dd55448a5044d9d26a52


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.279211, 43.78041]
                },
                "type": "Feature",
                "properties": {
                    "serviceUri": "http://www.disit.org/km4city/resource/Fuel_station_01a234db6235dd55448a5044d9d26a52",
                    "serviceType": "TransferServiceAndRenting_Fuel_station",
                    "name": "PINI E SETTESOLDI SNC",
                    "typeLabel": "Fuel station",
                    "phone": "",
                    "fax": "",
                    "website": "",
                    "province": "FI",
                    "city": "FIRENZE",
                    "cap": "50131",
                    "email": "",
                    "note": "",
                    "description": "",
                    "description2": "",
                    "multimedia": "",
                    "address": "VIALE DEI MILLE",
                    "civic": "",
                    "brand": "AgipEni",
                    "linkDBpedia": [],
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
```

```
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                }
            }
        ]
    },
    "realtime": {
        "head": {
            "vars": ["measuredTime", "fuel", "price", "currency", "self"]
        },
        "results": {
            "bindings": [{
                "measuredTime": {
                    "value": "2017-01-13 16:01:52"
                },
                "fuel": {
                    "value": "Benzina"
                },
                "price": {
                    "value": "1.579"
                },
                "currency": {
                    "value": "EUR"
                },
                "self": {
                    "value": "true"
                }
            }, {
                "measuredTime": {
                    "value": "2017-01-13 16:01:52"
                },
                "fuel": {
                    "value": "Blue Diesel"
                },
                "price": {
                    "value": "1.539"
                },
                "currency": {
                    "value": "EUR"
                },
                "self": {
                    "value": "true"
                }
            }, {
                "measuredTime": {
                    "value": "2017-01-13 16:01:52"
                },
                "fuel": {
                    "value": "Blue Super"
                },
                "price": {
                    "value": "1.729"
                },
                "currency": {
                    "value": "EUR"
                },
                "self": {
                    "value": "true"
                }
            }, {
                "measuredTime": {
                    "value": "2017-01-13 16:01:52"
                },
                "fuel": {
                    "value": "Gasolio"
                },
                "price": {
                    "value": "1.439"
                },
                "currency": {
                    "value": "EUR"
                },
                "self": {
                    "value": "true"
```

```
                    }
                }]
            }
        }
}
```


First aid (added with RESOLUTE project)

http://www.disit.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/dde440c760ef578da41599feb2396631


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.260015, 43.773457]
                },
                "type": "Feature",
                "properties": {
                    "name": "PRONTO SOCCORSO OSPEDALE SANTA MARIA NUOVA",
                    "typeLabel": "First aid",
                    "serviceType": "Emergency_First_aid",
                    "phone": "0552758844",
                    "fax": "0552758844",
                    "website": "",
                    "province": "FI",
                    "city": "FIRENZE",
                    "cap": "50100",
                    "email": "",
                    "linkDBpedia": [],
                    "note": "",
                    "description": "",
                    "description2": "",
                    "multimedia": "",
                    "serviceUri": "http://www.disit.org/km4city/resource/dde440c760ef578da41599feb2396631",
                    "address": "PIAZZA SANTA MARIA NUOVA",
                    "civic": "1",
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                },
                "id": 1
            }
        ]
    },
    "realtime": {
        "head": {
            "vars": ["measuredTime", "state", "redCode", "yellowCode", "greenCode", "blueCode", "whiteCode"]
        },
        "results": {
            "bindings": [{
                "measuredTime": {
                    "value": "2017/01/19T15:25:00.000"
                },
                "state": {
                    "value": "Con Destinazione"
                },
                "redCode": {
                    "value": "0"
                },
                "yellowCode": {
                    "value": "5"
                },
                "greenCode": {
                    "value": "5"
                },
                "blueCode": {
                    "value": "1"
                },
```

```
            "whiteCode": {
                "value": "0"
            }
        }, {
            "measuredTime": {
                "value": "2017/01/19T15:25:00.000"
            },
            "state": {
                "value": "In Attesa"
            },
            "redCode": {
                "value": "0"
            },
            "yellowCode": {
                "value": "2"
            },
            "greenCode": {
                "value": "5"
            },
            "blueCode": {
                "value": "1"
            },
            "whiteCode": {
                "value": "0"
            }
        }, {
            "measuredTime": {
                "value": "2017/01/19T15:25:00.000"
            },
            "state": {
                "value": "In Visita"
            },
            "redCode": {
                "value": "0"
            },
            "yellowCode": {
                "value": "4"
            },
            "greenCode": {
                "value": "5"
            },
            "blueCode": {
                "value": "1"
            },
            "whiteCode": {
                "value": "0"
            }
        }, {
            "measuredTime": {
                "value": "2017/01/19T15:25:00.000"
            },
            "state": {
                "value": "Oss. Temporanea"
            },
            "redCode": {
                "value": "0"
            },
            "yellowCode": {
                "value": "1"
            },
            "greenCode": {
                "value": "2"
            },
            "blueCode": {
                "value": "1"
            },
            "whiteCode": {
                "value": "0"
            }
        }, {
            "measuredTime": {
                "value": "2017/01/19T15:25:00.000"
            },
            "state": {
                "value": "Totali"
            },
```

```
                "redCode": {
                    "value": "0"
                },
                "yellowCode": {
                    "value": "12"
                },
                "greenCode": {
                    "value": "17"
                },
                "blueCode": {
                    "value": "4"
                },
                "whiteCode": {
                    "value": "0"
                }
            }]
        }
    }
}
```


Smart waste container (added with REPLICATE project)

http://www.disit.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/cassonetto01


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.2557, 43.7745]
                },
                "type": "Feature",
                "properties": {
                    "serviceUri": "http://www.disit.org/km4city/resource/cassonetto01",
                    "serviceType": "Environment_Smart_waste_container",
                    "name": "Cassonetto via martelli",
                    "typeLabel": "Smart waste comtainer",
                    "phone": "055232323",
                    "province": "FI",
                    "city": "Firenze",
                    "cap": "",
                    "address": "via martelli",
                    "civic": "2",
                    "wasteType": "http://www.disit.org/km4city/schema#anyWaste",
                    "capacity": "200",
                    "collectionTime": "alle 13:00 tutti I giorni",
                    "physicalShape": "campana",
                    "linkDBpedia": [],
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                }
            }
        ]
    },
    "realtime": {
        "head": {
            "vars": ["measuredTime", "wasteLevel", "batteryLevel"]
        },
        "results": {
            "bindings": [{
                "measuredTime": {
                    "value": "2017-01-19T15:46:31+01:00"
                },
                "wasteLevel": {
                    "value": "0.53592324"
                },
                "batteryLevel": {
                    "value": "261.33566"
```

```
                    }
                }]
            }
        }
}
```


Smart bench (added with REPLICATE project)

http://www.disit.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/bench001


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.2554, 43.7737]
                },
                "type": "Feature",
                "properties": {
                    "serviceUri": "http://www.disit.org/km4city/resource/bench001",
                    "serviceType": "Entertainment_Smart_bench",
                    "name": "Panchina via martelli",
                    "typeLabel": "Smart bench",
                    "phone": "055232323",
                    "fax": "",
                    "website": "",
                    "province": "FI",
                    "city": "Firenze",
                    "cap": "",
                    "email": "",
                    "note": "",
                    "description": "",
                    "description2": "",
                    "multimedia": "",
                    "address": "via martelli",
                    "civic": "2",
                    "seats": "4",
                    "withWifi": "true",
                    "withUsb": "true",
                    "withAudio": "true",
                    "linkDBpedia": [],
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                }
            }
        ]
    },
    "realtime": {
        "head": {
            "vars": ["measuredTime", "temperature", "humidity", "pressure", "airQualityCO2", "light", "sittingsInRefPeriod", "totalSittings",
"passagesInRefPeriod", "totalPassages"]
        },
        "results": {
            "bindings": [{
                "measuredTime": {
                    "value": "2017-01-19T15:42:52+01:00"
                },
                "temperature": {
                    "value": "36.675144"
                },
                "humidity": {
                    "value": "83.60987"
                },
                "pressure": {
                    "value": "24.017311"
                },
                "airQualityCO2": {
                    "value": "85.21111"
```

```
                    },
                    "light": {
                        "value": "0.51458716"
                    },
                    "sittingsInRefPeriod": {
                        "value": "0"
                    },
                    "totalSittings": {
                        "value": "656"
                    },
                    "passagesInRefPeriod": {
                        "value": "3"
                    },
                    "totalPassages": {
                        "value": "3313"
                    }
                }]
            }
        }
}
```


Smart irrigator (added with REPLICATE project)

http://www.disit.org/ServiceMap/api/v1/?serviceUri=http://www.disit.org/km4city/resource/irrigatore01


```
{
    "Service": {
        "type": "FeatureCollection",
        "features": [
            {
                "geometry": {
                    "type": "Point",
                    "coordinates": [11.2496, 43.7736]
                },
                "type": "Feature",
                "properties": {
                    "serviceUri": "http://www.disit.org/km4city/resource/irrigatore01",
                    "serviceType": "Environment_Smart_irrigator",
                    "name": "Irrigatore p.zza S. Maria Novella",
                    "typeLabel": "Smart irrigator",
                    "phone": "0552556677",
                    "province": "FI",
                    "city": "Firenze",
                    "note": "",
                    "description": "",
                    "description2": "",
                    "address": "p.zza Santa Maria Novella",
                    "civic": "23",
                    "linkDBpedia": [],
                    "wktGeometry": "",
                    "photos": [],
                    "photoThumbs": [],
                    "photoOrigs": [],
                    "avgStars": 0.0,
                    "starsCount": 0,
                    "comments": []
                }
            }
        ]
    },
    "realtime": {
        "head": {
            "vars": ["measuredTime", "currentlyActive", "temperature", "internalTemperature", "humidity", "soilWaterPotential", "leafWetness"]
        },
        "results": {
            "bindings": [{
                "measuredTime": {
                    "value": "2017-01-19T15:46:31+01:00"
                },
                "currentlyActive": {
                    "value": "true"
                },
                "temperature": {
                    "value": "14.397217"
                },
```

```
                "internalTemperature": {
                    "value": "26.770363"
                },
                "humidity": {
                    "value": "26.808607"
                },
                "soilWaterPotential": {
                    "value": "329.1715"
                },
                "leafWetness": {
                    "value": "23.110199"
                }
            }]
        }
    }
}
```


Energy meter (added with REPLICATE project)
Under development

Recharge station (added with REPLICATE project)
Under development

Smart street light (added with REPLICATE project)
Under development

Air quality monitoring station
Under development

Public transport API
In the following the API that are related with public transports are reported.
Note: The information regarding timetable is acquired in GTFS format. Due to different names used in the previous version of the API that was only for buses, the names used in the API are not aligned with GTFS nomenclature in particular bus lines are mapped to GTFS routes and bus routes are mapped to GTFS trips. In the next version of the API names used may change to be aligned with GTFS.

Agency list

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/agencies
the API provide a list of the public transport agencies available	
Parameters:	
uid	optional user identifier
format	only json
Results:	
the API provides an array of JSON objects of the agencies available, for each agency is provided the agency name and the agency URI used to identify the agency in other APIs	
Examples:	


```
{
    "Agencies": [{
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_ataflinea_Agency_172",
        "name": "Ataf&Linea"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_acvbus_Agency_173",
        "name": "Autolinee Chianti Valdarno"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_amvbus_Agency_171",
        "name": "Autolinee Mugello Valdisieve"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_blubus_Agency_175",
```

```
        "name": "BluBus"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_cap_Agency_169",
        "name": "C.A.P. Consorzio Autolinee Pratesi"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_ctt_Agency_500",
        "name": "CTT NORD"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_cpt_Agency_176",
        "name": "Consorzio Pisano Trasporti"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_etruriamobilita_Agency_168",
        "name": "Etruria Mobilità"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Tram_gest_Agency_303",
        "name": "GEST S.p.A."
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_piubus_Agency_170",
        "name": "Piùbus"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_sienamobilita_Agency_167",
        "name": "Siena Mobilità"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Train_tft_Agency_196",
        "name": "T.F.T. S.p.A."
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_tiemme_Agency_400",
        "name": "TIEMME SPA"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Train_trenitalia_Agency_163",
        "name": "TRENITALIA S.p.A."
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Ferry_toremar_Agency_205",
        "name": "Toremar Toscana Regionale Marittima Spa"
    }, {
        "agency": "http:\/\/www.disit.org\/km4city\/resource\/Bus_vaibus_Agency_174",
        "name": "Vaibus"
    }]
```


Bugs:

(Bus) Lines list

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-lines
the API provide a list of the public transport lines available for a given agency.	
Parameters:	
agency	URI of the agency whose lines are to be retrieved
uid	optional user identifier
format	only json
Results:	
the API provides an array of JSON objects of the lines available, for each line is provided the line long and short name, the uri identifying the line.	
Examples:	
http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-lines/?agency=http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172	


```
{
    "BusLines": [{
        "agency": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
        "shortName": "C1",
        "longName": "Parterre-Ponte Alle Grazie",
        "uri": "http:\/\/www.disit.org\/km4city\/resource\/Bus_ataflinea_Route_1380827827"
    }, {
        "agency": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
        "shortName": "S3",
        "longName": "Scuola Marconi-L'Olmo",
        "uri": "http:\/\/www.disit.org\/km4city\/resource\/Bus_ataflinea_Route_1858266107"
```

}, …]
}

Note:
The API can be used on any kind of public transport (Tram, Train, etc.) not only Bus.

(Bus) Routes list

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-routes
API provide a list of the public transport routes available for a given agency, line or passing by a specific stop.	
Parameters:	
agency	URI of the agency whose lines are to be retrieved
line	URI or shortName of a line (if URI is provided the agency is not needed)
busStopName	URI or name of a stop (if URI is provided the agency is not needed)
geometry	if true the WKT geometry of the route is returned (false is assumed if not provided)
uid	optional user identifier
format	only json
Results:	

the API provides an array of JSON objects of the routes available, for each route is provided:

- *line*: line shot name
- *route*: the route URI
- *routeName*: optional route name
- *wktGeometry*: the WKT geometry of the route
- *firstBusStop*: name of the first bus stop
- *lastBusStop*: name of the last bus stop

Examples:

http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-routes/?agency=http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172&line=11&geometry=true


```
{
    "BusRoutes": [{
        "line": "11",
        "route": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3337883",
        "routeName": "",
        "wktGeometry": "LINESTRING(11.2172537345524 43.7326316393217, 11.2173853491045 43.7325390476232, …)",
        "firstBusStop": "La Gora",
        "lastBusStop": "Salviatino"
    }, {
        "line": "11",
        "route": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3338595",
        "routeName": "",
        "wktGeometry": "LINESTRING(11.2939833018846 43.7848045962375, 11.2939931338599 43.7848236867993,…)",
        "firstBusStop": "Salviatino",
        "lastBusStop": "La Gora"
    }]
}
```


http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-routes/?agency=http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172&busStopName=Stazione%20Pensilina


```
{
    "BusRoutes": [{
        "line": "1",
        "route": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3337874",
        "routeName": "",
```

```
        "firstBusStop": "Boccaccio",
        "lastBusStop": "Stazione Palazzo Congressi"
    }, {
        "line": "2",
        "route": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3322861",
        "routeName": "",
        "firstBusStop": "Calenzano",
        "lastBusStop": "Stazione Palazzo Congressi"
    }, {
        "line": "4",
        "route": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3323029",
        "routeName": "",
        "firstBusStop": "Cappuccini",
        "lastBusStop": "Stazione Mercato Centrale"
    }, … ]
}
```


Note:
The API can be used on any kind of public transport (Tram, Train, etc.) not only Bus.

(Bus) Stop list

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-stops
API provide a list of the public transport stops available for a given route.	
Parameters:	
route	URI of the route whose bus stops are to be retrieved
geometry	if true the WKT geometry of the route is returned
uid	optional user identifier
format	only json
Results:	
the API provides an JSON Object with line number (aka line short name) and line name (aka line long name) and a GeoJSON FeatureCollection with the stops. The stops are provided in stop order, from the first to the last.	
Examples:	

http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-stops/?route=http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3337883&geometry=true


```
{
    "Route": {
        "lineNumber": "11",
        "lineName": "Salviatino-Le Gore",
        "wktGeometry": "LINESTRING(11.2172537345524 43.7326316393217, 11.2173853491045 43.7325390476232, …)"
    },
    "BusStops": {
        "type": "FeatureCollection",
        "features": [{
            "geometry": {
                "type": "Point",
                "coordinates": [11.217254, 43.73263]
            },
            "type": "Feature",
            "properties": {
                "popupContent": "La Gora",
                "name": "La Gora",
                "serviceUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Stop_FM1208_5",
                "tipo": "fermata",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "serviceType": "TransferServiceAndRenting_BusStop"
            },
            "id": 1
        }, {
            "geometry": {
                "type": "Point",
                "coordinates": [11.220704, 43.73418]
            },
```

```
            "type": "Feature",
            "properties": {
                "popupContent": "Volterrana 02",
                "name": "Volterrana 02",
                "serviceUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Stop_FM1209_5",
                "tipo": "fermata",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "serviceType": "TransferServiceAndRenting_BusStop"
            },
            "id": 2
        }, … ]
    }
}
```


Note:
The API can be used on any kind of public transport (Tram, Train, etc.) not only Bus.

Search (Bus) Routes in a geographic area

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/
API provides a list of the public transport routes that have a stop in a specified area.	
Parameters:	
selection	optional “<lat>;<lng>” with a GPS position or “<lat1>;<lng1>;<lat2>;<lng2>” for a rectangular area or “wkt:<WKT_string>” or “geo:<geoid>” for a geographic area described as Well Known Text (see other APIs for more details)
maxDists	optional maximum distance from the GPS position of the entities to be retrieved, expressed in Km (0.1 is assumed if not present)
maxResults	maximum number of results to be returned (if parameter is missing 100 is assumed), if it is 0 all results are returned.
agency	optional URI of an agency to restrict the search to a specified agency
geometry	if true the WKT geometry of each route is returned (considered false if not provided)
uid	optional user identifier
format	only json
Results:	

the API provides a JSON Object with all the routes that have stops on the specified area. For each route the following properties are provided:

- lineNumber: the line short name
- lineName: the line long name
- route: the route name
- routeUri: an URI identifying the route (it can be used to retrieve all the stops of the route)
- direction: with first and last stop
- agency: with agency name
- agencyUri: with agency URI
- polyline: with the WKT geometry of the route

Examples:

http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/?selection=43.7755;11.2495&maxDists=0.1&maxResults=5&geometry=true


```
{
    "PublicTransportLine": {
        "type": "FeatureCollection",
        "features": [{
            "type": "Feature",
            "properties": {
                "lineNumber": "12",
                "lineName": "",
                "route": "",
                "routeUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3137547",
                "direction": "Campo Marte Fs ➔ Stazione Parcheggio",
```

```
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "polyline": "LINESTRING(11.2762059770919 43.7774442270155, 11.2761623454295 43.777427353435, …)",
                "serviceType": "PublicTransportLine"
            },
            "id": 1
        }, {
            "type": "Feature",
            "properties": {
                "lineNumber": "36",
                "lineName": "",
                "route": "",
                "routeUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3317289",
                "direction": "Cascine Del Riccio ➔ Stazione Abside S.M.N.",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "polyline": "LINESTRING(11.2551477298522 43.7339067055819, 11.2550069037315 43.7335043206344, …)",
                "serviceType": "PublicTransportLine"
            },
            "id": 2
        }, {
            "type": "Feature",
            "properties": {
                "lineNumber": "13",
                "lineName": "",
                "route": "",
                "routeUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3300218",
                "direction": "Il David ➔ Stazione Palazzo Congressi",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "polyline": "LINESTRING(11.2648824363224 43.7625434190618, 11.2648878248007 43.7625306663665, …)",
                "serviceType": "PublicTransportLine"
            },
            "id": 3
        }, {
            "type": "Feature",
            "properties": {
                "lineNumber": "11",
                "lineName": "",
                "route": "",
                "routeUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3337883",
                "direction": "La Gora ➔ Salviatino",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "polyline": "LINESTRING(11.2172537345524 43.7326316393217, 11.2173853491045 43.7325390476232, …)",
                "serviceType": "PublicTransportLine"
            },
            "id": 4
        }, {
            "type": "Feature",
            "properties": {
                "lineNumber": "C2",
                "lineName": "",
                "route": "",
                "routeUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Trip_1923_3365643",
                "direction": "Leopolda ➔ Piazza Beccaria",
                "agency": "Ataf&Linea",
                "agencyUri": "http://www.disit.org/km4city/resource/Bus_ataflinea_Agency_172",
                "polyline": "LINESTRING(11.2389601794313 43.7773069544217, 11.2389099511421 43.777364365556, …)",
                "serviceType": "PublicTransportLine"
            },
            "id": 5
        }]
    }
}
```


Note:
The API can be used on any kind of public transport (Tram, Train, etc.) not only Bus.

Estimated Bus position

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-position
API provides the estimated current position of buses	

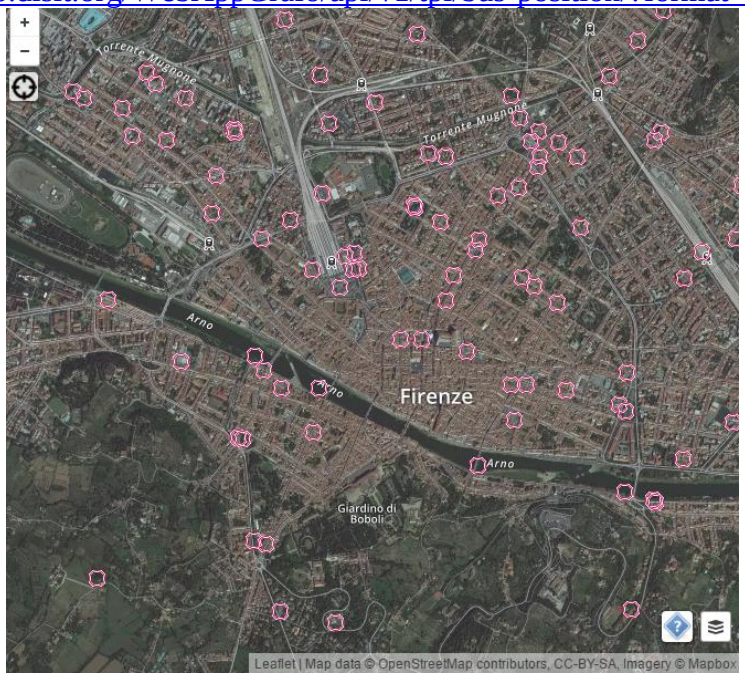
Parameters:	
uid	optional user identifier
format	json or html
Results:	

when format is html the API provides web visualization of the current bus positions while if format is json it provides a GeoJSON "FeatureCollection" with the data of each bus that is currently active. For each bus the following properties are provided:

- vehicleNum: the number of vehicle
- line: the line short name
- direction: with first and last stop
- detectionTime: the delay in minutes from the current time and the time the position was acquired.

Examples:

http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-position/?format=html



http://servicemap.disit.org/WebAppGrafo/api/v1/tpl/bus-position/?format=json


```
{
    "type": "FeatureCollection",
    "features": [{
        "geometry": {
            "type": "Point",
            "coordinates": [11.340633, 43.735943]
        },
        "type": "Feature",
        "properties": {
            "vehicleNum": "3133579",
            "line": "24",
            "direction": "Sorgane Piazza Rodolico &#10132; Grassina",
            "tipo": "RealTimeInfo",
            "serviceUri": "busCode3133579",
            "detectionTime": "0",
            "serviceType": "bus_real_time"
        },
        "id": 1
    }, {
        "geometry": {
            "type": "Point",
            "coordinates": [11.272773, 43.774574]
        },
        "type": "Feature",
        "properties": {
            "vehicleNum": "3134531",
            "line": "12",
```

```
            "direction": "Piazzale Michelangelo &#10132; Stazione Parcheggio",
            "tipo": "RealTimeInfo",
            "serviceUri": "busCode3134531",
            "detectionTime": "0",
            "serviceType": "bus_real_time"
        },
        "id": 2
    }, {
        "geometry": {
            "type": "Point",
            "coordinates": [11.253791, 43.78007]
        },
        "type": "Feature",
        "properties": {
            "vehicleNum": "3137538",
            "line": "12",
            "direction": "Piazzale Michelangelo &#10132; Stazione Parcheggio",
            "tipo": "RealTimeInfo",
            "serviceUri": "busCode3137538",
            "detectionTime": "2",
            "serviceType": "bus_real_time"
        },
        "id": 3
    }, … ]
}
```


Note:
Currently it provides the position of ATAF&Linea buses based on the timetable.

Shortest path finder API

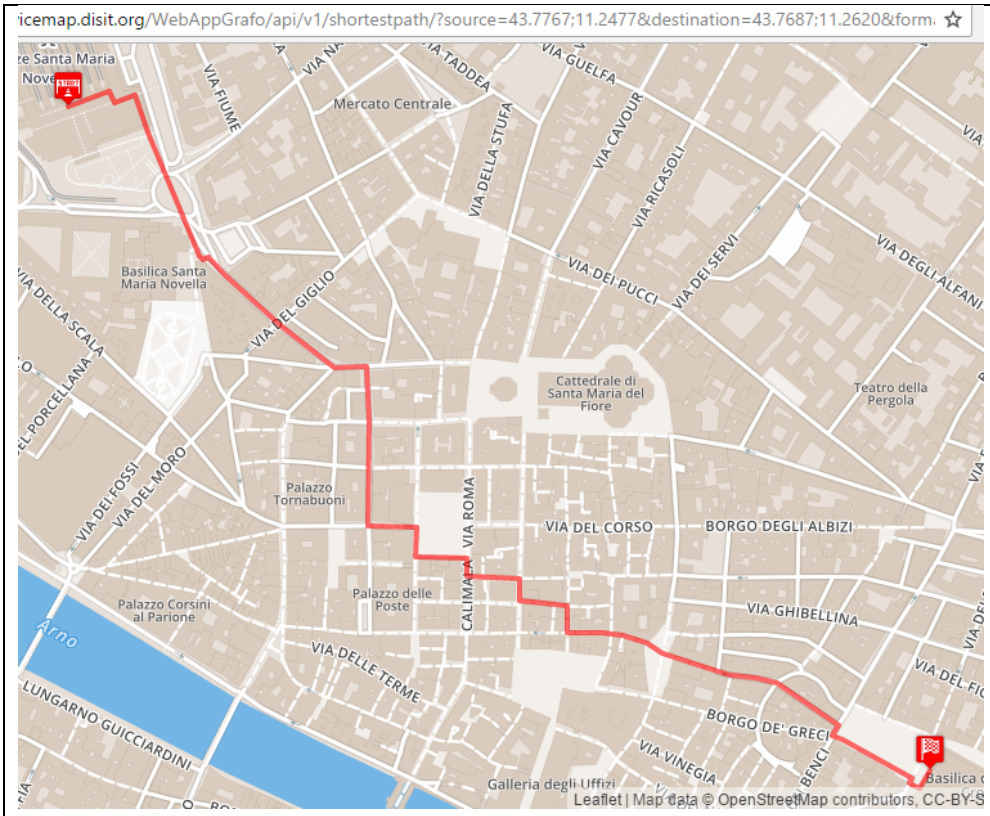
URL	http://servicemap.disit.org/WebAppGrafo/api/v1/shortestpath
This API allows to get a path from a source point to a destination point. The points can be specified as latitude;longitude coordinates or using the serviceUri of a service. The path is provided as WKT geometry and as a sequence of arcs between nodes (the service uses the OpenStreetMap road graph). The type of route can be specified as using public transport, feet, car or bike (using cycle paths whenever possible). The start datetime is used to select the options for public_transport and to evaluate the time needed to make the path.	
Parameters:	
source	"<lat>;<lng>" or service URI of the starting point
destination	"<lat>;<lng>" or service URI of the destination
routeType	can be "public_transport", "foot_shortest", "foot_quiet", "car", "bike_security" (foot_shortest is assumed if missing).
maxFeetKM	maximum distance in km by feet for routeType=public_transport (default 0.1)
startDatetime	datetime of start (current datetime if omitted) (e.g. "2017-01-13T12:34:00")
format	json or html
uid	optional user identifier
Results:	
the API provides a JSON object with the path from the source to the destination.The following is an example:	


```
{
    "journey": {
        "source_node": {
            "node": "2531656503",
            "lon": 11.2477,
            "lat": 43.7767
        },
        "destination_node": {
            "node": "4006368396",
            "lon": 11.262,
            "lat": 43.7687
        },
        "search_route_type": "shortest_foot_optimization"
        "search_max_feet_km": 0.1,
```

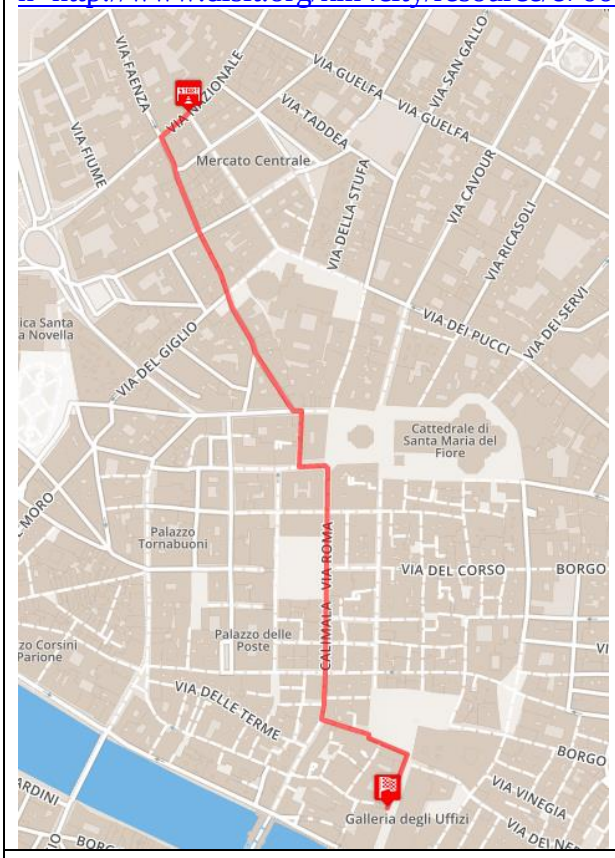
```
        "start_datetime": "2017-01-12T12:34:00",
        "routes": [{
            "wkt": "LINESTRING(11.247773299999992 43.7765506,11.24788979999999 43.7765815,...)",
            "arc": [{
                "distance": 0.009997643869982628,
                "start_datetime": "12:34:00",
                "end_datetime": "12:34:07",
                "destination_node": {
                    "lon": 11.24788979999999,
                    "lat": 43.7765815,
                    "node_id": "2531656509"
                },
                "source_node": {
                    "lon": 11.247773299999992,
                    "lat": 43.7765506,
                    "node_id": "2531656503"
                },
                "transport_provider": "",
                "transport": "foot",
                "transport_service_type": "",
                "desc": "nd"
            }, ... {
                "distance": 0.013927524955311556,
                "start_datetime": "12:38:05",
                "end_datetime": "12:38:15",
                "destination_node": {
                    "lon": 11.250239099999982,
                    "lat": 43.77465860000007,
                    "node_id": "271149487"
                },
                "source_node": {
                    "lon": 11.250108900000013,
                    "lat": 43.77474099999993,
                    "node_id": "1754184405"
                },
                "transport_provider": "",
                "transport": "foot",
                "transport_service_type": "",
                "desc": "Via Panzani"
            }, ... ]
    },
    "response": {
        "error_message": "successful",
        "current_operation": "route optimization",
        "error_code": "0"
    },
    "elapsed_ms": 3943,
    "message_version": "1.0"
}
```


Examples:

http://servicemap.disit.org/WebAppGrafo/api/v1/shortestpath/?source=43.7767;11.2477&destination=43.7687;11.2620&format=html



http://servicemap.disit.org/WebAppGrafo/api/v1/shortestpath/?source=43.7772;11.2522&destination=http://www.disit.org/km4city/resource/e76655ae0ae0a956df3a60500b2861dd&format=html



Notes:
This API is under development and currently supports only foot_shortest, foot_quite and car

routes.

Image caching API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/imgcache
The API provides a cache for the given image url, it downloads the image, scales it to the thumbnail or medium size depending on the size requested and save it for future requests.	
Parameters:	
imageUrl	url to the image
size	the size of the image to be produced, it can be equal to “thumb”, “medium” or a number between 1 and 2000 pixels.
Results:	
It provides the scaled image produced in the same format as the original to be fit in a square of *size* x *size*, if the url is not an image it redirects to the original url.	
Examples:	
http://servicemap.disit.org/WebAppGrafo/api/v1/imgcache?imageUrl=http://www.florenceheritage.it/mobileApp/immagini/zocchi/148.jpg&size=thumb	
Notes:	

5.2 API SPARQL di Knowledge Base (API10) (UNIFI)

5.2.1 Procotollo, API: SPARQL

Il protocollo usato per effettuare query SPARQL sulla base di conoscenza sarà compatibile con il protocollo standard di accesso agli endpoint SPARQL 1.1 come specificato in https://www.w3.org/TR/sparql11-protocol/#query-operation (il protocollo di update non sarà accessibile dall’esterno). Per limitare l’accesso ad alcuni dati sensibili alcune parti della base di conoscenza sarà accessibile solo tramite autenticazione effettuata tramite certificato (sia lato client che lato server) quindi usando protocollo HTTPS. L’accesso a parti della base di conoscenza sarà possibile anche utilizzando connessione non autenticata ma i dati utilizzabili saranno limitati ai soli dataset senza restrizioni di uso.

The data is currently available also using the standard W3C linked data protocol. It allows getting a machine readable representation of a resource like http://www.disit.org/km4city/resource/048017 as RDF/XML format. In the HTTP request protocol the header parameters Accept with “application/rdf+xml” should be specified while if the resource url is open in a web browser a (quite) human readable html version is generated.
Details on the RDF/XML format can be found at https://www.w3.org/TR/rdf-syntax-grammar/

Data can be also accessed using the standard W3C SPARQL 1.1 language and SPARQL query protocol at http://servicemap.disit.org/WebAppGrafo/sparql
Details on SPARQL 1.1 can be found at https://www.w3.org/TR/sparql11-overview/
At http://log.disit.org/sparql_query_frontend/ a query user interface can be found to play with SPARQL queries with some examples. Moreover the knowledge graph can be navigated using the Linked Open Graph viewer available at http://log.disit.org

5.3 API for rendering deductions on the basis of the whole ontological model of SmartDS (API11) (UNIFI)

Queste API permetteranno di rivalutare un'albero associato ad una istanza di processo definita in SmartDS (rivalutando le query SPARQL associate ai nodi foglia) per ottenere l'italian flag associato al nodo radice.

La API prevede invio in POST con Content-type: application/xml su URL
http://server/dss/rest/modelinstanceoperations/
di un xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<operationinstance>
        <modelInstanceId>14</modelInstanceId>
        <description>computeDecision</description>
</operationinstance>
```

dove il modelInstanceId indica l'id dell'istanza del modello da ricalcolare. Il risultato è l'istanza aggiornata con i valori dell'italian flag dei nodi:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<modelinstance>
   <modelId>10</modelId>
   <objective>TestGP_cloned</objective>
   <url>http://www.disit.org/</url>
   <size>5</size>
   <modelUserId>1</modelUserId>
   <modelInstanceId>14</modelInstanceId>
   <specific_objective>Istanza Test GP2_cloned</specific_objective>
   <date_create_instance>17-06-2015 17:32:13.000</date_create_instance>
   <date_last_modify_instance>24-03-2017 19:15:34.045</date_last_modify_instance>
   <status>3</status>
   <start_exec>2017-02-01 19:17:04.232</start_exec>
   <end_exec>2017-02-01 19:17:43.000</end_exec>
   <instanceUserId>1</instanceUserId>
   <children>
      <position>C0</position>
      <description>TestGP</description>
      <isLeaf>false</isLeaf>
      <modelId>10</modelId>
      <url>null</url>
      <comment>null</comment>
      <IF_calculated>
         <IF_green>0.41049491981882724</IF_green>
         <IF_white>0.2853878075651854 5</IF_white>
         <IF_red>0.30411727261598726</IF_red>
      </IF_calculated>
     …..
   </children>
</modelinstance>
```

L'elenco delle istanze disponibili è reperibile in GET alla URL http://server/dss/rest/modelinstances

mentre l'elenco dei modelli da http://server/dss/rest/models

5.4 API Query ID di Knowledge Base (API12) (UNIFI)

This API provides results based on a query saved on ServiceMap. Different kinds of queries can be saved from the ServiceMap widget when you find the disk icon. Clicking on it, and filling a form, the system will perform two steps (1) saving the query performed in a database for your further

reuse, (2) send to the user a set of links via email. Among them, two links are: one for read only (rendering only if HTML, or getting the JSON of results), the second, allows you to open again the editing web page on the ServiceMap to overwrite the previously saved query performed.

5.4.1 Procotollo, API: Query ID sulla KB

Service search by query id

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/
it allows to retrieve the set of services associated with a query stored using the servicemap user interface.	
Parameters:	
queryId	identifier of the query stored on servicemap
lang	ISO 2 chars language code (e.g. “it”, “en”, “fr”, “de”, “es”) to be used for returned descriptions if available in multiple languages. Currently for languages other than “it” and “en” it returns “en” descriptions. (if parameter is missing “en” is assumed)
geometry	true/false, if true it returns a “hasGeometry” property for each service stating if the service has a complex WKT geometries (linestring, polygon) associated with it (if parameter is missing “false” is assumed)
uid	optional user identifier
format	html or json
Results:	
the results format is the same as the previous API	
Examples:	
Search for any BusStop or CulturalActivity service in 100m near Santa Maria del Fiore http://servicemap.disit.org/WebAppGrafo/api/v1/?queryId=e02db54355fea40808300473c3537ff&format=json&lang=it	
Bugs:	

5.5 API User Crowd Sourcing (API13) (UNIFI)

5.5.1 Protocollo esposto, API: Crowd Sourcing

Il modulo espone una interfaccia REST su protocollo HHTP per l’accesso ai dati presenti nella piattaforma di Crowd Sourcing. Nel caso che non riconosca un insieme valido di parametri restituisce HTTP response 400.

Nome: Tweets per channel
URL: http://www.disit.org/tv/query
Descrizione
L’API restituisce i tweet del canale selezionato in un intervallo specifico di tempo. I campi restituiti sono: message, publicationTime, twitterUser, twitterId, lang, retweetCount, favoriteCount, geo_lat, geo_long, hashtagsOnTwitter, links
Modalità: GET

Parametri	
Canale	Obligatorio
[start_date, end_date]	Se start_dat è null vengono presi tutti fino ad end_date; viceversa se end_date è null vengono presi I twee a partire da start._date
format	L'output può essere o csv o json.

Esempi

Restituisce tutti I tweet relativi canale "apretoscana" a parte dal 17 dicembre 2015 fino al 19 dicembre 2015.

http://disit.org/tv/query/query.php?channel=apretoscana&start_date=2015-12-17&end_date=2015-12-19&format=json

I risultati JSON sono:

{

"message":"Neuer Blog zum #medtech #matchmaking #event [INSIGHT] Tools, Support and Capital for Inn ovators ist online: https:\/\/t.co\/O3403aToaD #SIP",
  "publicationTime":"2015-12-17 09:23:12",
  "twitterUser":"anitajoerg1",
  "twitterId":"677403660582825984",
  "lang":"de",
  "retweetCount":"0",
  "favoriteCount":"0",
  "geo_lat":"0.00000",
  "geo_long":"0.00000",
  "hashtagsOnTwitter":"#medtech #matchmaking #event #SIP",
  "links":"https:\/\/t.co\/O3403aToaD http:\/\/bit.ly\/1YY55X8"
},
{
  "message":"Thank you @traveldavide!\n#matchmaking #love #happiness #dating #luxurylifestyle https:\ /\/t.co\/W8G2NwZTvb",
  "publicationTime":"2015-12-17 09:32:54",
  "twitterUser":"IvyIntMM",
  "twitterId":"677406101827133440",
  "lang":"en",
  "retweetCount":"1",
  "favoriteCount":"2",
  "geo_lat":"0.00000",
  "geo_long":"0.00000",
  "hashtagsOnTwitter":"#matchmaking #love #happiness #dating #luxurylifestyle",
  "links":"https:\/\/t.co\/W8G2NwZTvb https:\/\/twitter.com\/anonymagence\/status\/677180450985648 128"
},

}

(I risultati sono stati troncati)

Nome: Tweet ID per channel

URL: http://www.disit.org/tv/query	
Descrizione	
L'API restituisce tutti i tweet ID dei tweets del canale selezionato in un intervallo specifico di tempo.	
Modalità: GET	
Parametri	
Canale	Obligatorio
[start_date, end_date]	Se start_date è null vengono presi tutti fino ad end_date; viceversa se end_date è null vengono presi I twee a partire da start._date
onlyID	True
Esempi	

Restituisce tutti i tweet ID relativi canale "apretoscana" a partire dal 17 dicembre 2015 fino al 19 dicembre 2015.

http://disit.org/tv/ query/query.php?channel=apretoscana&start_date=2015-12-17&end_date=2015-12-19&onlyID=true

I risultati JSON sono:

{
{ **"twitterId"**:"677627227895431168"},
{ **"twitterId"**:"677628560383811585"},
{ **"twitterId"**:"677629191706251266"},
{ **"twitterId"**:"677630463884836867"}
}

(These results have been truncated.)

Nome: List of active search	
URL: http://www.disit.org/tv/query	
Descrizione	
L'API restituisce tutte le ricerche attive	
Modalità: GET	
Parametri	
activeSearch	activeSearch=true
Esempi	

L'API restituisce tutte le ricerche attive

http://disit.org/tv/query/query.php?activeSearch=true

I risultati JSON sono:

{
{
"ID_request":"1",
"text_req":"#meteo",
"lan_req":"it"
},
{
"ID_request":"6",
"text_req":"#ODDIT15 #Firenze",
"lan_req":"it"
},
{

```
    "ID_request":"7",
    "text_req":"#fodd",
    "lan_req":"it"
  },
  {
    "ID_request":"8",
    "text_req":"#OpenDataDay #Firenze",
    "lan_req":""
  },
  {
    "ID_request":"9",
    "text_req":"@flash_meteo",
    "lan_req":"it"
  },
}
```

(I risultati sono stati troncati)

Nome: Retweet number per tweet	
URL: http://www.disit.org/tv/query	
Descrizione	
L'API restituisce il numero di retweet associati ad un tweet padre indicato tramite il twitterId.	
Modalità: GET	
Parametri	
TweetID	Obbligatorio
Esempi	

Restituisce il numero di retweet associati a 605237373820006400
http://disit.org/tv/query/query.php?originalTweet=605237373820006400
I risultati JSON sono:
{
" MAX(retweetCount)":"9",
}	

Nome: Tweets per search		
URL: http://www.disit.org/tv/query		
Descrizione		
L'API restituisce i tweet della ricerca selezionata in un intervallo specifico di tempo. I campi restituiti sono: twitterId, message, publicationTime, lang, retweet, originalTweet**.**		
Modalità: GET		
Parametri		
Canale	Obbligatorio	
[start_date, end_date]	Se start_date è null vengono presi tutti fino ad end_date; viceversa se end_date è null vengono presi I twee a partire da start._date	
format	L'output può essere o csv o json.	
lang		
Esempi		
Restituisce tutti I tweet "italiani" relativi alla ricerca "Fedez" a parte dal 17 dicembre 2015 fino al 18 dicembre 2015.		

http://disit.org/tv/query/query.php?search=%23fedez&start_date=2015-12-17&end_date=2015-12-18&lang=it
I risultati JSON sono:

{
  {
    "message":"Mercoled\u00ec 23 Dicembre #ChristmasParty #PinetaGarden\nSpecialGuest #FEDEZ\nPull man & Prevendite da ogni zona!... https:\/\/t.co\/eWM0WUWvpT",
    "twitterId":"677470005567557632",
    "publicationTime":"2015-12-17 13:46:50",
    "lang":"it",
    "originalTweet":"",
    "retweet":"0"
  },
  {
    "message":"Ho sempre pensato che questa canzone fosse intensa e molto forte.. Davvero, davvero bell a! \n#Fedez @Fedez https:\/\/t.co\/lQGwT4WRax",
    "twitterId":"677486476414869504",
    "publicationTime":"2015-12-17 14:52:17",
    "lang":"it",
    "originalTweet":"",
    "retweet":"0"
  },
  {
    "message":"Ho sempre pensato che questa canzone fosse intensa e molto forte.. Davvero, davvero bell a! \n#Fedez https:\/\/t.co\/lQGwT4WRax",
    "twitterId":"677488322462240768",
    "publicationTime":"2015-12-17 14:59:37",
    "lang":"it",
    "originalTweet":"",
    "retweet":"0"
  },

(I risultati sono stati troncati)

Nome: Number of Tweets per search	
URL: http://www.disit.org/tv/query	
Descrizione	
L'API restituisce il numero di tweet della ricerca selezionata ordinata per lingua.	
Modalità: GET	
Parametri	
ID_request	Obbligatorio
lang	
Esempi	
Restituisce il numero di tweet della ricerca identificata da 101 ordinati per lingua **http://disit.org/tv/query/query.php?ID_request=101** **I risultati JSON sono:** { **"lang"**:"it", **"count"**:"605"	

```
    },
    {       "lang":"en",     "count":"130"
    },
    {       "lang":"und",     "count":"82"
    },
    {       "lang":"es",     "count":"26"
    }
}
```

(I risultati sono stati troncati)

Nome: First 10 hot hashtags	
URL: http://www.disit.org/tv/query	
Descrizione	
L'API restituisce i primi 10 hashtag per numero di tweet associati al canale specificato.	
Modalità: GET	
Parametri	
Trends	Obbligatorio
Esempi	

restituisce i primi 10 hashtag per numero di tweet associati al canale Firenze

http://disit.org/tv/query/query.php?trends=Firenze
I risultati JSON sono:


```
{
        {       'value' : '#Expo2015',          'tweet_count' : 3746
        },
        {       'value' : '#expo2015',          'tweet_count' : 3321
        },
        {       'value' : '#expo2015',          'tweet_count' : 2992
        },
        {       'value' : '#Expo2015',          'tweet_count' : 2989
        },
        {       'value' : '#Expo2015',          'tweet_count' : 2180
        },
        {       'value' : '#noexpo',            'tweet_count' : 1891
        },
        {       'value' : '#Expo2015',          'tweet_count' : 1881
        },
        {       'value' : '#Expo2015',          'tweet_count' : 1521
        },
        {       'value' : '#Expo2015',          'tweet_count' : 1386
        },
        {       'value' : '#Expo2015Milano',            'tweet_count' : 1124
        }
}
```


Nome: First 10 hot mentions
URL: http://www.disit.org/tv/query
Descrizione

L'API restituisce le prime 10 mentions per numero di tweet associate al canale specificato.	
Modalità: GET	
Parametri	
Mentions	Obbligatorio
Esempi	

Restituisce le prime 10 mention per numero di tweet associate al canale EXPO2015
http://disit.org/tv/query/query.php?trends=EXPO2015
I risultati JSON sono:


```
{
  {        'value' : '@Expo2015Milano',        'tweet_count' : 1423
        },
        {        'value' : '@matteorenzi',        'tweet_count' : 257
        },
        {        'value' : '@Pad_Ita2015',        'tweet_count' : 122
        },
        {        'value' : '@AstroSamantha',        'tweet_count' : 100
        },
        {        'value' : '@camilleriwrites',        'tweet_count' : 88
        },
        {        'value' : '@FedericaMog',        'tweet_count' : 88
        },
       {        'value' : '@beppesevergnini',        'tweet_count' : 77
        },
        {        'value' : '@EUExpo2015',        'tweet_count' : 77
        },
        {        'value' : '@corriereit',        'tweet_count' : 76
        },
        {        'value' : '@SkyTG24',        'tweet_count' : 74
        },
}
```


Rating and comment API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/feedback
API accepts a star rating (1-5) and/or a comment on a specific service. Comments are not automatically associated with the service, a moderator has to validate the comment provided.	
Parameters:	
serviceUri	URI identifying a service
stars	value 1 to 5 (if omitted no ratings is provided)
comment	comment provided by the user
lang	the language used in the comment
uid	a user identifier associated with the user providing the data
Results:	
the API fails using HTTP error code 404 if the serviceURI is not valid, stars or comment is not provided or user id is not provided.	
Examples:	
http://servicemap.disit.org/WebAppGrafo/api/v1/feedback?service=...&stars=2&comment=a%20comment&uid=...	
Notes:	

Service Photo API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/photo/
API accepts in POST as a multipart form the serviceUri, the user id and a photo in jpeg or png format. The photo provided is not automatically associated with the serviceUri a moderator will check it and decide.	
Parameters:	
serviceUri	URI identifying a service
uid	a user identifier associated with the user providing the data
file	a part named "file" with the photo to be uploaded, the part should contain the mimetype or the filename
Results:	
the API fails using HTTP error code 404 if the serviceURI is not valid, user id is not provided or a part named "file" is not present and the mimetype of this file cannot be found or if it's not valid.	
Examples:	
NA	
Notes:	

Last contributions API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/feedback/last
API reports a list of the last photos, comments and starred services from the users.	
Parameters:	
uid	a user identifier
lang	ISO 2 chars language code (e.g. "it", "en", "fr", "de", "es") to be used for returned descriptions if available in multiple languages. Currently for languages other than "it" and "en" it returns "en" descriptions. (if parameter is missing "en" is assumed)
format	only json
Results:	
the API reports a JSON object with informations on the last contributions..	
Examples:	

http://servicemap.disit.org/WebAppGrafo/api/v1/feedback/last


```
{
    "LastPhotos": [{
        "serviceUri": "http://www.disit.org/km4city/resource/af388d64a33b2624456a9a268ab01b54",
        "typeLabel": "Free WiFi point",
        "serviceType": "TourismService_Wifi",
        "long": "11.25355",
        "lat": "43.77682",
        "serviceName": "Firenze WIFI",
        "photo": "http://servicemap.disit.org/WebAppGrafo/api/v1/photo/file-5690474034488739316.jpg",
        "photoThumb": "http://servicemap.disit.org/WebAppGrafo/api/v1/photo/thumbs/file-5690474034488739316.jpg",
        "photoOrig": "http://servicemap.disit.org/WebAppGrafo/api/v1/photo/originals/file-5690474034488739316.jpg",
        "timestamp": "2017-01-22 16:38:20.0"
    }, ...],
    "LastComments": [{
        "serviceUri": "http://www.disit.org/km4city/resource/cd9fa722072d84aa47d5bc6a74932c46",
        "typeLabel": "Museum",
        "serviceType": "CulturalActivity_Museum",
        "long": "11.263607",
        "lat": "43.769848",
        "serviceName": "MUSEO DI CASA BUONARROTI",
        "comment": "Palazzo del seicento comprato da Michelangelo nel quale si trovano diverse sculture e disegni di Michelangelo",
        "timestamp": "2016-12-17 09:04:14.0"
```

```
    }, ...],
    "LastStars": [{
        "serviceUri": "http://www.disit.org/km4city/resource/20950a98d5fc0d1d69115d2b531b7793",
        "typeLabel": "Museum",
        "serviceType": "CulturalActivity_Museum",
        "long": "11.263603",
        "lat": "43.769836",
        "serviceName": "CASA_BUONARROTI",
        "stars": 5,
        "timestamp": "2016-12-17 10:14:37.0"
    }, ...]
}
```


Notes:

Submit annotation API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/annotation/

API accepts in POST a JSON object with information on an annotation.

```
{
  "uid": "...",
  "id" : "<unique id identifier>,
  "position" : "<lat>;<long>", # ma anche con un serviceUri
  "type" : "<tipo definito dall'applicazione>",
  "visibility" : "private| public", # private=visibile solo a utente in una app, public= visibile a tutti gli utenti che usano una app
  ["timestamp" : "2017-02-13T12:34:15",] #se omesso prende data ora sottomissione
  "properties" : { ... un qualsiasi oggetto json, è l'applicazione che sa come interpretare i dati... }
}
```

JSON object properties:

uid	a user identifier associated with the user providing the data
id	unique identifier of the annotation, if already present the annotation is updated, we suggest to use your application name as a prefix when generating the id.
position	GPS position as “<lat>;<long>” or a serviceURI of the annotation
type	type of the annotation, expressed as names separted by dots, it implies a hierarchy “myapp.parking” or “myapp.sensor.traffic” to avoid name clashes the root type should identify the application providing the annotation.
visibility	should be *private* or *public*, private means that the annotation is visible only to the user identified by the uid and public it can be retrived by all users
timestamp	optional timestamp associated with the annotation, if omitted the current time is used.
properties	a JSON object with any properties as needed by the application

Results:

the API fails using HTTP error code 400 if uid, position, type visibility are not correct

Examples:

NA

Notes: Under developemnt

Delete annotation API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/annotation/

API accepts in DELETE a request to delete an annotation given the uid and the annotation id. Only the creator can delete an annotation.

Paramters:	
uid	a user identifier associated with the user providing the data
id	unique identifier of the annotation, if already present the annotation is updated
Results:	
the API fails using HTTP error code 400 if uid, or id are not valid or the user has no rights to delete the annotation	
Examples:	
Notes: Under developemnt	

Retrieve annotations API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/annotation/
API accepts in GET a request for a list of annotations, around a point, of a specific type or super type, with a specific visibility	
Paramters:	
uid	a user identifier (optional if requesting public annotations)
type	type or super type of the annotations to be retrieved, if the type ends with ".*" all the sub types are considered (e.g. myapp.*, myapp.sensor.*)
visibility	optional visibility of the annotations to be retrieved, it can be private, public or any (default any)
position	optional position expressed with <lat>;<long> or via serviceUri
maxDists	optional maximum distance in Km from the position specified (default 0.1)
maxResults	optional maimum number of results (default 100)
minTimestamp	optional filter on the minimum timestamp
order	optional order of results it can be timestamp_desc, timestamp_asc, distance_asc, distance_desc
Results:	
the API provides a GeoJSON FeatureCollection object with information of the annotations matching the requested data.	
Examples:	
http://servicemap.disit.org/WebAppGrafo/api/v1/annotation/?type=myapp.*&visibility=public	
Notes: Under developemnt	

5.6 API User Engagement (API14) (UNIFI)

Profilo Tool/Algoritmo	
API User Engagement	
Nome responsabile	
Partner responsabile	UNIFI
Descrizione Tool/algoritmo	API per inviare in push (da server verso le mobile APP e web) suggerimenti e stimoli a partecipare, compiti da svolgere, etc.
Dati primari in ingresso	Identificativo dell'utente, mediante il quale in sistema

	recupera la sua posizione, le sue preferenze, lo storico delle assistance e engagement forniti in precedenza	
Dati prodotti in Uscita	Produzione di suggerimenti di assistenza e coinvolgimento personalizzati	
Principi (metodo, basi teoriche, etc.)	Il modulo ciclicamente controlla se le regole di engagement sono valide ed eventualmente ritorna le azioni relative	
Casi di test (presenti/assenti)	Corrispondenti a informazioni di assistenza per evitare multe, ridurre i tempi di parcheggio, usare lo scambiatore, consumare meno, andare in bike, etc.	
Posizione casi di test		
Principali problemi non risolti	nessuno	
Principali requisiti pendenti	nessuno	
Aspetti Tecnologici	Utilizzo di piattafoma a regole	
Stato (proposto/approvato)	Proposto	
Implementato/non implementato	in fase di implementazione	
Stato implementazione, percentuale	30%	
Eseguibile/libreria/web app	web app	
Single thread/Multithread	multi thread	
Linguaggio di sviluppo (java, Php, ETL)	Java, PHP	
Piattaforme supportate	Windows, Linux, Unix	
Posizione del codice sorgente	svn	
Indirizzo/i web services (se presenti) con indicazione credenziali accesso (se necessarie)		
Indirizzo/i accesso via web (se presenti)		
Nomi tool/moduli usati (aggiungere una riga per ogni tool/modulo usato)	Interfacce API usate	Modello di comunicazione e formato
Assistance, engagement		HTTP REST
ServiceMap	EventS, Services	HTTP REST
Formati usati (aggiungere una riga per ogni formato usato)	Condiviso con tool/modulo	Nome formato o riferimento a sezione definizione
JSON	ServiceMap	JavaScript Object Notation
Protocolli usati (aggiungere una riga per ogni protocollo usato)	Condiviso con tool/modulo	Nome protocollo o riferimento a sezione definizione
REST HTTP	ServiceMap	
Nomi database usati (aggiungere una riga per ogni db usato)	Descrizione	
AccessLog Recommender Sensor	Log degli accessi ai servizi da parte degli utenti presi dal ServiceMap Dati del raccomandatore Dati relativi alla posizione e allo stato dell'utente	
Tipo interfaccia utente (web/applicazione)	Modello sviluppo, linguaggio	Libreria usata per UI
web		
Libreria usata	Nome e versione	Licenza: GPL, LGPL, PEK,

(aggiungere una riga per ogni libreria usata)	usata	proprietaria, commerciale, etc.
spring	org.springframework 1.3.3.RELEASE	Apache License 2.0
hibernate	org.hibernate 5.1.0.Final	GNU Lesser General Public License
drools	6.4.0.Final	Apache License 2.0

5.6.1 Procotollo: User Engagement

Note di assistance o di engagement sono forniti direttamente quando l'applicazione mobile invia i dati come un evento sul Sensor Server and Manager. Il modulo di engagement, valuta in continuo l'evoluzione di stato degli utenti con i loro mobile e precalcola note di assistance e engagement che vengono mandate in risposta alla successiva chiamata di salvataggio dei dati di stato del telefono. Questo tipo di approccio (inviare le engagment nello stesso scambio dati di salvataggio di stato) evita di avere processi multiple sul telefono che lo caricano troppo e potrebbe produrre problemi di consumo e di carico del telefono stesso, cosa per altro poco accettabile per l'utente.

Il modulo espone una interfaccia REST su protocollo HTTP per l'accesso agli engagement preparati per l'utente specificato.

Nome: Engagement (internal API)	
URL: http://192.168.0.17:8080/engager-api/engager	
Descrizione	
API per richiedere engagement per un utente.	
Modalità: GET	
Parametri Obbligatori	
uid	Hash dell'id utente calcolato dalla mobile app
Return	Assessor: true/false Engagement: una lista di engagement, Assistance: una lista di assistance Un engagement / assistance e' descritto da: id: identificatore univoco type: tipologia (SURVEY, SHOW, REQUEST) classe:ASSISTANCE/ENGAGEMENT title: titolo del messaggio msg: messaggio dell'egangement/assistance (testo o sondaggio) uri (opzionale): uri del servizio relativo all'engagement gps lat/long (opzionale): posizione relativa all'engagement serviceType (opzionale): tipo del servizio serviceName: nome del servizio, specifica il titolo nel caso di un sondaggio serviceLabel: label del servizio time_elapse: tempo di scadenza del messaggio, in millisecond dal 1 gennaio 1970 action_bannedfor: tempo per il quale il messaggio verra' bannato action_howmany: quanti messaggi dello stesso tipo possono essere inviati contemporaneamente action_sendrate: ogni quanti minuti messaggi dello stesso tipo possono essere inviati

	action_rulename: nome della regola che ha generato questo engagement/assistance

Esempio

http://192.168.0.17:8080/engager-api/engager?uid=0938d0d0941b654663b4224ec0f53059bab29e14dc97670c68513d3940fe0149

JSON Result:

{"assessor":false,
"engagement":[
{"type":"REQUEST_PHOTO",
"msg":"\"Arte e Fotografie negli archivi di Giorgio Castelfranco e Rodolfo Siviero\" at MUSEO DI CASA SIVIER (until 2017-02-27)",
"classe":"ENGAGEMENT",
"id":769490,
"time_elapse":84801558,
"title":"EVENT today",
"uri":"http://www.disit.org/km4city/resource/Event_20624_497e2ca3c8526f8b1e9cb0ee3",
"gps_lat":43.764866,"gps_long":11.263975,
"time_created":1487672421000,
"serviceType":"Event",
"serviceLabel":"Mostre",
"serviceName":"Arte e Fotografie negli archivi di Giorgio Castelfranco e Rodolfo Siviero",
"action_bannedfor":0,
"action_howmany":1,
"action_rulename":"daily_event_en",
"action_sendrate":0}
],
"assistance":[]}

Nome: feedback rimuovi Engagement	
URL: http://engager.km4city.org/engine/api/cancel-engagement	
Descrizione	
API per notificare che un utente ha richiesto la rimozione di un engagement dal suo terminale	
Modalità: GET	
Parametri Obbligatori	
Id	Id dell'engagement che l'utente ha rimosso
Return	OK se la rimozione e' andata a buon fine KO se la rimozione non e' andata a buon fine + message che specifica il tipo di problema

Esempio

http:// engager.km4city.org/engine/api /cancel-engagement?id=16401

JSON Result:

{"result":"OK","message":null}

Nome: Survey

URL: http://engager.km4city.org/api/survey-collector	
Descrizione	
API per collezionare i risultati dei sondaggi proposti all'utente	
Modalità: POST	
Parametri Obbligatori	
user_id	Hash dell'id utente calcolato dalla mobile app
survey_id	Nome del sondaggio
engagement_id	Id dell'engagement che contiene il sondaggio
completed_time	Millisecondi passati dal 1 gennaio 1970 all'invio del sondaggio
survey_response	Testo in JSON della risposta al sondaggio (dipende dal tipo di sondaggio)
Return	OK se la rimozione e' andata a buon fine KO se la rimozione non e' andata a buon fine + message che specifica il tipo di problema
Esempio	

```
function send_post() {
   var url = " http://engager.km4city.org/api/survey-collector
  var method = "POST";
  var data="{\"user_id\":
\"0149d66674bf6af68f061e3a3557179a40ea0cd01cb04176c3f50b6189269a20\",\"survey_id\":
\"survey_turist_en\",\"completed_time\": 1474894199801,\"survey_response\": {\"question1\":
[\"one\"]}}";
  var async = true;
  var request = new XMLHttpRequest();
  request.open(method, url, async);
  request.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
  request.setRequestHeader("Content-length", data.length);
  request.setRequestHeader("Connection", "close");
  request.send(data);
}
```

JSON Result:
{"result":"OK","message":null}

5.7 API User Profiling and Recommendations (API15) (UNIFI)

Smart City API

Profilo Tool/Algoritmo	
API User Profiling and Recommendations	
Nome responsabile	
Partner responsabile	DISIT-UNIFI
Descrizione Tool/algoritmo	Il Recommender fornisce raccomandazioni all'utente sulla base del proprio profilo (i.e., student, commuter, tourist, citizen, all) a cui corrispondono le categorie di interesse, delle coordinate geografiche e del raggio di prossimità entro cui effettuare le raccomandazioni. È

	uno strumento che utilizza la Single Value Decomposition per la produzione delle raccomandazioni API per permettere alle App mobile e WEB di richiedere suggerimenti al server di user profiling and suggestion e comunicare user profile, come anche il suo menu, comportamento, etc..	
Dati primari in ingresso	Azione, UserId	
Dati prodotti in Uscita	Raccomandazioni in formato JSON	
Principi (metodo, basi teoriche, etc.)	La rilevanza dei servizi raccomandati dipende dall'attività dell'utente e dalla prossimità ai servizi	
Casi di test (presenti/assenti)	assenti	
Posizione casi di test		
Principali problemi non risolti		
Principali requisiti pendenti		
Aspetti Tecnologici		
Stato (proposto/approvato)	proposto	
Implementato/non implementato	Non implementato	
Stato implementazione, percentuale	0%	
Eseguibile/libreria/web app	Web app	
Single thread/Multithread	Single thread	
Linguaggio di sviluppo (java, Php, ETL)	Java, PHP	
Piattaforme supportate	Linux, Mac, Windows, Unix	
Posizione del codice sorgente		
Indirizzo/i web services (se presenti) con indicazione credenziali accesso (se necessarie)	http://www.disit.org/SmartCityRecommender/	
Indirizzo/i accesso via web (se presenti)	http://www.disit.org/SmartCityRecommender/	
Nomi tool/moduli usati (aggiungere una riga per ogni tool/modulo usato)	Interfacce API usate	Modello di comunicazione e formato
ServiceMap	Weather, EventS, Services	HTTP REST
RDF Store	SPARQL Query	HTTP REST
Formati usati (aggiungere una riga per ogni formato usato)	Condiviso con tool/modulo	Nome formato o riferimento a sezione definizione
JSON	ServiceMap	
JSON	RDF Store	
Protocolli usati (aggiungere una riga per ogni protocollo usato)	Condiviso con tool/modulo	Nome protocollo o riferimento a sezione definizione
HTTP REST	ServiceMap	
HTTP REST	RDF Store	
Nomi database usati (aggiungere una riga per ogni db usato)	Descrizione	
AccessLog Recommender	Log degli accessi ai servizi da parte degli utenti presi dal ServiceMap Dati del raccomandatore	
Tipo interfaccia utente (web/applicazione)	Modello sviluppo, linguaggio	Libreria usata per UI
web	PHP	HighCharts

Libreria usata (aggiungere una riga per ogni libreria usata)	Nome e versione usata	Licenza: GPL, LGPL, PEK, proprietaria, commerciale, etc.
Apache Mahout	0.11.0	Apache License, Version 2.0

5.7.1 Protocollo esposto: Recommendations

Il modulo espone una interfaccia REST su protocollo HTTP per l'accesso alle raccomandazioni per l'utente.

Nome: Recommend	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per richiedere raccomandazioni per un utente.	
Modalità: GET, POST	
Parametri Obbligatori	
action	Azione richiesta (in questo caso *recommend*)
user	Hash dell'id utente calcolato dalla mobile app
profile	Il profilo dell'utente (i.e., student, commuter, tourist, citizen, all)
language	Lingua dell'utente (i.e., en, it)
latitude	Latitudine in formato decimale
longitude	Longitudine in formato decimale
distance	Raggio di ricerca in km
Parametri Opzionali	
mode	Modalità delle coordinate (i.e., gps, manual)
version	Versione dell'app mobile; questo campo è necessario se si vogliono ottenere tweet nei suggerimenti
aroundme	Specifica se si vogliono ricevere i suggerimenti più vicini (e.g., true) L'effetto di aroundme = true è quello di impostare svd = false e accettare suggerimenti già suggeriti, con il risultato di ottenere sempre le raccomandazioni dei servizi più vicini. Il raccomandatore svd = true (default) fa sì che l'algoritmo scelga di volta in volta quali categorie utilizzare per costruire la query SPARQL, e quindi i risultati possono variare (possono essere forniti prima raccomandazioni di servizi lontani, e successivamente più vicini).
svd	se utilizzare il raccomandatore svd (i.e., true)
Esempio	

http://disit.org/SmartCityRecommender/?action=recommend&user=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b&profile=student&language=en&latitude=43.7727&longitude=11.2532&distance=1

JSON Result:

[{"suggestions":[{"Service":{"features":[{"geometry":{"coordinates":[11.2544,43.7732],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"ShoppingAndService_Pharmacy","note":"farmacie.100","website":"","address":"PIAZZA DI SAN GIOVANNI","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/50b6312db054b3dab55edf1c0aac0004","description":"","description2":"","photos":[],"linkDBpedia":[],"civic":"17R","multimedia":"","cap":"50144","avgStars":0.0,"province":"FI","starsCount":0,"phone":"","name":"Farmacia S. ANTONINO","typeLabel":"Pharmacy","fax":"","email":""}}],"type":"FeatureCollection"}},{"Service":{"f

eatures":[{"geometry":{"coordinates":[11.2535,43.7716],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"ShoppingAndService_Pharmacy","note":"farmacie.94","website":"","address":"PIAZZA DELLA
REPUBBLICA","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/4846b073f8032213bf0630d747e93272","description":"","description2":"","photos":[],"linkDBpedia":[],"civic":"23R","multimedia":"","cap":"50123","avgStars":0.0,"province":"FI","starsCount":0,"phone":"","name":"Farmacia
INTERNAZIONALE","typeLabel":"Pharmacy","fax":"","email":""}}],"type":"FeatureCollection"}},{"Service":{"features":[{"geometry":{"coordinates":[11.2546,43.7735],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"ShoppingAndService_Pharmacy","note":"farmacie.49","website":"","address":"PIAZZA DI SAN
GIOVANNI","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/d71a90da83e4a9acdc40e58fe947d6cd","description":"","description2":"","photos":[],"linkDBpedia":[],"civic":"20R","multimedia":"","cap":"50129","avgStars":0.0,"province":"FI","starsCount":0,"phone":"","name":"Farmacia ALL INSEGNA DEL
MORO","typeLabel":"Pharmacy","fax":"","email":""}}],"type":"FeatureCollection"}}],"label":"Things to do","priority":1,"group":"Things to
do"},{"suggestions":[{"Service":{"features":[{"geometry":{"coordinates":[11.2532,43.7732],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"WineAndFood_Restaurant","note":"","website":"","address":"VICOLO DI SANTA MARIA
MAGGIORE","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/c4f1ebc8d44d373823173ffeb3695214","description":"IMPRESA INDIVIDUALE – Individuali o assimilabili o non iscritti al
RI","description2":"","photos":[],"linkDBpedia":[],"civic":"1","multimedia":"","cap":"50123","avgStars":0.0,"province":"FI","starsCount":0,"phone":"","name":"BERNACCHIONI
SIMONE","typeLabel":"Restaurant","fax":"","email":""}}],"type":"FeatureCollection"}},{"Service":{"features":[{"geometry":{"coordinates":[11.2535,43.772],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"WineAndFood_Restaurant","note":"","website":"","address":"PIAZZA DELLA
REPUBBLICA","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/87486751af941d9a1d5be5297f90e3fd","description":"SOCIETA' A RESPONSABILITA' LIMITATA – Società di
capitale","description2":"","photos":[],"linkDBpedia":[],"civic":"5","multimedia":"","cap":"50100","avgStars":0.0,"province":"FI","starsCount":0,"phone":"","name":"PERSEUS FIESOLE
SRL","typeLabel":"Restaurant","fax":"","email":""}}],"type":"FeatureCollection"}},{"Service":{"features":[{"geometry":{"coordinates":[11.2522,43.7716],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"WineAndFood_Restaurant","note":"","website":"","address":"VIA DEI
PESCIONI","comments":[],"city":"FIRENZE","serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/e6a5976759e6ade6714ff0ce8de29876","description":"","description2":"","photos":[],"linkDBpedia":[],"civic":"2","multimedia":"","cap":"50100","avgStars":0.0,"province":"FI","starsCount":0,"phone":"05526651","name":"Hotel Helvetia & Bristol e Ristorante Hosteria
Bibendum","typeLabel":"Restaurant","fax":"0552399897","email":"information.hbf@royaldemeure.com"}}],"type":"FeatureCollection"}}],"label":"Wine and Food","priority":2,"group":"Wine and Food"},
…
}]
(These results have been truncated)

aroundme	alreadyRecommended	svd	action
true	true	true	restituisce sempre e soltanto i suggerimenti più vicini (alreadyRecommended e svd sono ignorati)
true	true	false/null	restituisce sempre e soltanto i suggerimenti più vicini (alreadyRecommended e svd sono ignorati)
true	false/null	true	restituisce sempre e soltanto i suggerimenti più vicini (alreadyRecommended e svd sono ignorati)
true	false/null	false/null	restituisce sempre e soltanto i suggerimenti più vicini (alreadyRecommended e svd sono ignorati)

false/null	true	true/null	può restituire i suggerimenti già suggeriti negli n giorni precedenti (attualmente n=7), ma non necessariamente quelli più vicini
false/null	true	false	restituisce sempre e soltanto i suggerimenti più vicini
false/null	false/null	true/null	restituisce soltanto i suggerimenti non suggeriti negli n giorni precedenti (attualmente n=7), ma non necessariamente quelli più vicini
false/null	false/null	false	restituisce soltanto i suggerimenti più vicini fra quelli non suggeriti negli n giorni precedenti (attualmente n=7)

Nome: Recommend for Group	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per richiedere raccomandazioni per un utente per un gruppo.	
Modalità: GET, POST	
Parametri Obbligatori	
action	l'azione richiesta (in questo caso *recommendForGroup*)
user	l'hash dell'id utente calcolato dalla mobile app
profile	il profilo dell'utente (i.e., student, commuter, tourist, citizen, all)
group	il nome del gruppo
language	la lingua dell'utente (i.e., en, it)
latitude	la latitudine in formato decimale
longitude	la longitudine in formato decimale
distance	il raggio di ricerca in km
Parametri Opzionali	
mode	la modalità delle coordinate (i.e., gps, manual)
version	la versione dell'app mobile; questo campo è necessario se si vogliono ottenere tweet nei suggerimenti
svd	se utilizzare il raccomandatore svd (i.e., true)
Esempio	

Search services in the category of Accommodation and Bus Stops around a point, giving the coordinates Latitude and Longitude, within 200m for services and 100m for bus stops.

http://192.168.0.207:8080/SmartCityRecommender/?action=recommendForGroup&user=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b&profile=student&group=Bus&language=en&latitude=43.7727&longitude=11.2532&distance=1

JSON Result:

[{"suggestions":[{"realtime":{},"busLines":{"head":{"busStop":"PECORI","vars":"busLine"},"results":{"bindings":[{"busLine":{"type":"literal","value":"C1"}},{"busLine":{"type":"literal","value":"C2"}}]}},"BusStop":{"features":[{"geometry":{"coordinates":[11.2533,43.7727],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"TransferServiceAndRenting_BusStop","address":"","avgStars":0.0,"comments":[],"starsCount":0,"serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/FM0758","name":"PECORI","typeLabel":"BusStop","photos":[]}}],"type":"FeatureCollection"}},{"realtime":{},"busLines":{"head":{"busStop":"PECORI","vars":"busLine"},"results":{"bindings":[{"busLine":{"type":"literal","value":"C1"}},{"busLine":{"type":"literal","value":"C2"}}]}},"BusStop":{"features":[{"geometry":{"coordinates":[11.2533,43.7727],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"TransferServiceAndRenting_BusStop","address":"","avgStars":0.0,"comments":[],"starsCount":0,"serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/FM0758","name":"PECORI","typeLabel":"BusStop","photos":[]}}],"type":"FeatureCollection"}},{"realtime":{},"busLines":{"head":{"busStop":"VECCHIETTI","vars":"busLine"},"results":{"bindings":[{"busLine":{"type":"literal","value":"11"}},{"busLine":{"type":"literal","value":"6"}}]}},"BusStop":{"features":[{"geometry":{"coordinates":[11.2528,43.7725],"type":"Point"},"id":1,"type":"Feature","properties":{"serviceType":"TransferServiceAndRenting_BusStop","address":"VIA DEI VECCHIETTI","avgStars":0.0,"comments":[],"starsCount":0,"serviceUri":"http:\/\/www.disit.org\/km4city\/resource\/FM3004","name":"VECCHIETTI","typeLabel":"BusStop","photos":[]}}],"type":"FeatureCollectio

n"}}],"label":"Bus","priority":2,"group":"Bus"}]

Nome: Dislike	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per mettere il dislike per un gruppo.	
Modalità: GET, POST	
Parametri Obbligatori	
action	l'azione richiesta (in questo caso *dislike*)
user	l'hash dell'id utente calcolato dalla mobile app
group	il nome del gruppo da rimuovere dalle raccomandazioni
Esempio	
http://disit.org/SmartCityRecommender/?action=dislike&user=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b&group=Hotel **Result:** true, false	

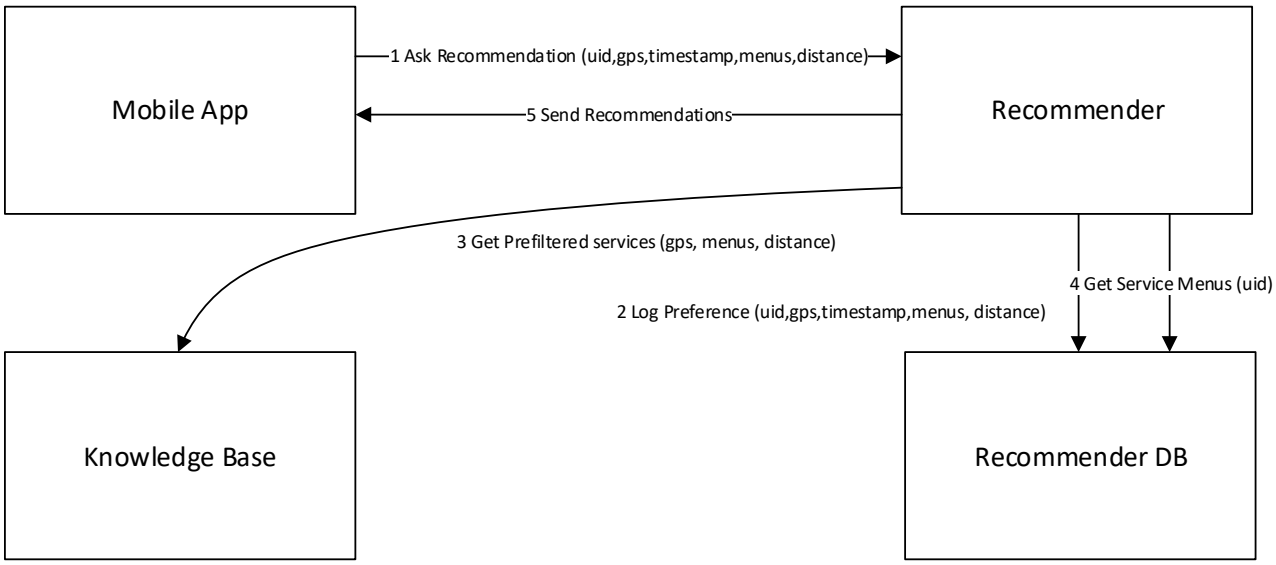
Nome: Remove Dislike	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per rimuovere il dislike per tutti i gruppi.	
Modalità: GET, POST	
Parametri Obbligatori	
action	l'azione richiesta (in questo caso *removeDislike*)
user	l'hash dell'id utente calcolato dalla mobile app
Esempio	
http://disit.org/SmartCityRecommender/?action=removeDislike&user=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b **Result:** true, false	

Nome: Log Tweet	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per fare il log di un tweet visualizzato sul database del recommender	
Modalità: GET, POST	
Parametri Obbligatori	
action	l'azione richiesta (in questo caso *logViewedTweet*)
user	l'hash dell'id utente calcolato dalla mobile app
twitterId	l'id del tweet
Esempio	
http://disit.org/SmartCityRecommender/?action=logViewedTweet&user=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b&twitterId=694606765888016385&group=Twitter1 **Result:** true, false	

Nome: Assess	
URL: http://disit.org/SmartCityRecommender/	
Descrizione	
API per fare l'assessment di un servizio	
Modalità: GET, POST	
Parametri Obbligatori	
action	l'azione richiesta (in questo caso *assess*)
uid	l'hash dell'id utente calcolato dalla mobile app
serviceUri o genID	l'id del servizio da valutare
vote	il voto (1-5)
suggType	il tipo di assess (assis, recom, engage)
Esempio	
http://disit.org/SmartCityRecommender/?action=assess&uid=3043b85d23d6f4879e1765c2c2e431cbc71d393065af06b03486ba4a04642b5b&serviceUri=prova&vote=1&suggType=recom **Result:** true	

1) Il dispositivo mobile invia al recommender una notifica di presenza, nel caso in cui l'utente si sia spostato almeno di N/2 metri (con N pari alla distanza impostata sul dispositivo), oppure quando viene cliccata una raccomandazione, con i seguenti parametri:
    a. userID;
    b. gps location: latitude, longitude;
    c. timestamp;
    d. menus, le categorie impostate sul dispositivo mobile, se le utente le ha personalizzate, oppure il profilo al quale l'utente appartiene (e.g., studente);
2) Il recommender scrive sul database MySQL l'evento registrato inviatogli dal dispositivo mobile come al punto 1;
3) Il recommender fa una query sull'RDF store per chiedere la lista dei servizi, considerando l'orario di apertura dei servizi e con i vincoli definiti dall'utente:
    a. gps location: latitude, longitude;
    b. timestamp;
    c. menus, le categorie impostate sul dispositivo mobile, se l'utente le ha personalizzate, oppure quelle associate al profilo al quale l'utente appartiene (e.g., studente);
    d. categorie dei servizi (Accommodation, CulturalActivity, Education, Emergency, Entertainment, EnvironmentAndAgriculture, FinancialService, GovernmentOffice, HealthCare, Shopping, TourismService, TransferService, WineAndFood ecc.), servizi trasversali (DL, LF, BUS, Meteo, Eventi in città);
4) Chiede con una query su database la lista dei menu associati al profilo utente;

Invia le raccomandazioni al dispositivo mobile secondo le regole definite nelle impostazioni del recommender e i vincoli trovati con i punti 3 e 4.



5.7.2 Protocollo esposto: User Mobility Info

Il modulo espone una interfaccia REST su protocollo HTTP per l'accesso alle informazioni sulla mobilità si un utente.

User information API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/userinfo
API reports a information about a user identified by a uid	
Parameters:	
uid	a user identifier
format	optional format of results (only json)
Results:	
the API reports a JSON object with informations about a user: kind of profile selected (tourist, citizen, student, commuter), home and work inferred positions, date of first use, comments provided, starred services and uploaded phots	
Examples:	

http://servicemap.disit.org/WebAppGrafo/api/v1/userinfo?uid=....
 	
{
"uid": "abce123…",
"profile": "tourist",
"homePosition": "43.123;10.123",
"workPosition": "",
"firstUseDate": "2017-01-12 08:12:01",
"comments": […],
"stars": […],
"photos": […],
}	

Notes:	

User mobility information API

URL	http://servicemap.disit.org/WebAppGrafo/api/v1/userinfo/mobility
API reports mobility information about a user identified by a uid, the data is provided on a dates range, it provides an estimation of the types of vehicles used for mobility: foot, car, bus, train with the distance and time.It can optionally include an estimation of the car driving style in the date	

range provided. The driving style information include an evaluation from 1 to 5 (1 poor, 5 excellent) and the distance travelled evaluated in the urban area and in the extra urban area.	
Parameters:	
uid	a user identifier
fromDate	optional a date to start from (included) e.g. “2017-02-12”, today is assumed if omitted
toDate	optional a date to arrive to (included) e.g “2017-02-16”, today is assumed if omitted
includeDrivingStyle	optional true/false (assumed false if omitted) states if details on the driving style of the user should be provided
includeTransports	optional true/false (assumed true if omitted)
format	optional format of results (only json)
Results:	
the API reports a JSON object with informations on mobility of user for each day in the range of dates provided (some dates can be skipped if no data is available for a date) and	
Examples:	

http://servicemap.disit.org/WebAppGrafo/api/v1/userinfo/mobility?uid=....&includeDrivingStyle=true
 	
{
"uid": "abce123…",
"mobility": [{
"date": "2017-02-12",
"transports" : [{transport: "foot", distance: 300, time: "00:15:00” },
{transport: "car", distance: 25000, time: "01:15:00” },
{transport: "bus", distance: 5000, time: "01:15:00” }],
"drivingStyle" : { "urban" : { points: 4, distance: 11000}, "extraurban" : { points: 2, distance :14000 } }
}]
}	

Notes:	

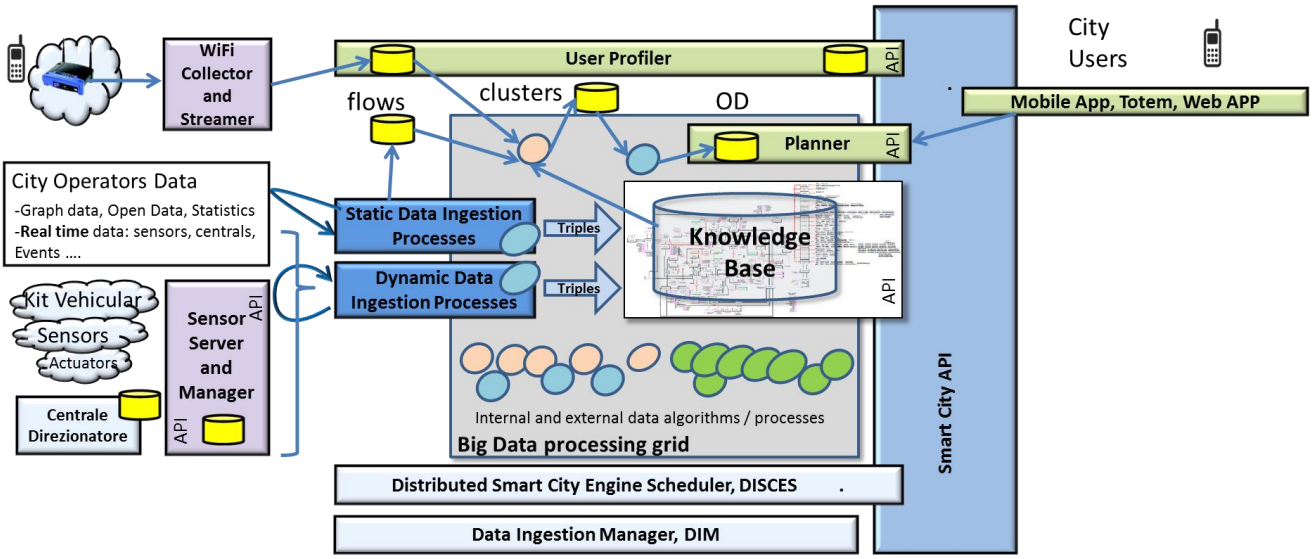
6 API per dati PA e SME

Nei seguenti paragrafi sono descritte le interfacce API per l’accesso ai dati ed informazioni: sia storici che puntuali real-time, per poter:

(i) abilitare lo sviluppo di applicazioni mobili e web sviluppate dalle PMI,

(ii) pubblicare dati/risultati provenienti da algoritmi di data analytics caricati sulla piattaforma da parte di PA e PMI.

6.1 API request planning (TPL e Logistic) (API16) (MIZAR)



Queste API permettono di richiedere la pianificazione del percorso fra due punti (o sequenza di punti) tramite: mezzi pubblici, auto, e/o a piedi. Queste API hanno un accesso condizionato in base all'utente, al profilo utente e ai dati richiesti.

6.1.1 Flusso di dati

I dati primari in ingresso sono il punto di inizio e il punto di fine del percorso. Dopo aver ricevuto i dati in ingresso, il sistema calcola il percorso migliore e restituisce una lista di archi da percorrere, rappresentanti le possibili opzioni.

6.1.2 Caratteristiche dell'API

Alcuni aspetti tecnologici dell'API sono espressi nella tabella qui di seguito:

Eseguibile/libreria/web app	Web API
Single thread/Multithread	Single thread
Linguaggio di sviluppo (java, Php, ETL)	PHP
Piattaforme supportate	Linux
Posizione del codice sorgente	Repository di progetto

6.1.3 Interfacce e standard

Il formato usato per la tipologia di dati scambiati è Data Objects.
Il modello di comunicazione e il protocollo usati sono SOAP (Simple Object Access Protocol), mentre non è presente nessun tipo di interfaccia utente (web o applicazione).

6.1.4 Procotollo SII-PathPlanner, API: PathPlannerRequest

Per inviare una richiesta al servizio di calcolo percorso, è necessario inviare una richiesta autenticata da un token con la seguente chiamata:

```
http://domain/rest/v1/objects/supportedtypes?token={token};startPoint={WGS84};stopPoint={WGS84};tran
sportationMode=PrivateVehicle
```

Il parametro transportationMode è di default PublicTransport; sono ammessi PrivateVehicle e Pedestrian.
La risposta ottenuta sarà una lista di archi di OpenStreetMap, in formato json, del tipo:

```
[
  {
    "ArcCode": "string",
    "Description": "string",
    "StartGeoInfo": {
      "Latitude": 0,
      "Longitude": 0
    },
    "StopGeoInfo": {
      "Latitude": 0,
      "Longitude": 0
    }
  },
  {
    "ArcCode": "string",
    "Description": "string",
    "StartGeoInfo": {
      "Latitude": 0,
      "Longitude": 0
    },
    "StopGeoInfo": {
      "Latitude": 0,
      "Longitude": 0
    }
  }

]
```

6.2 API Pubblicazione Dati Supervisore (API17) (MIZAR)

Le seguenti API permettono a PA e SME di accedere a dati storici, rilevati dal sistema Supervisore. Hanno un accesso condizionato in base all'utente, al profilo utente, ai dati richiesti per implementazione di nuovi scenari B2B o B2C.

6.2.1 Flusso di dati

L'API espone i dati del Supervisore, in formato OpenData, mettendoli a disposizione della piattaforma SII. In particolare, i dati esposti includono:

- Storico di Livelli di Servizio,
- O/D,
- volumi e conteggi dei sensori,
- trasporto pubblico,
- eventi (esposti in formato DENM),
- previsioni semaforiche (esposte in formato SPaT)

6.2.2 Caratteristiche dell'API

Alcuni aspetti tecnologici dell'API sono espressi nella tabella qui di seguito:

Eseguibile/libreria/web app	Web app
Single thread/Multithread	Multithread
Linguaggio di sviluppo (java, Php, ETL)	C#
Piattaforme supportate	Windows
Posizione del codice sorgente	Repository di progetto
Indirizzo/i web services (se presenti) con indicazione credenziali accesso (se necessarie)	http://IPserver/rest/api/v1 (per misure e dati) http://IPserver/api/vehicleservice/denm (per DENM) http://IPserver/api/vehicleservice/spat (per spat)

6.2.3 Interfacce e standard

L'interfaccia API usata è REST API, usando come modello di comunicazione il modello HTTP.
Il formato usato per la tipologia di dati scambiati in questione è JSON.
Inoltre, il protocollo è anche REST API

La libreria usata, REST API, ha una licenza proprietaria, ad uso gratuito per il progetto SII.

6.2.4 Procotollo Supervisore-SII (API17) (MIZAR)

Per ottenere le misure dal supervisore, è prima necessario ottenere l'elenco degli oggetti presenti con la seguente chiamata:

```
http://domain/rest/v1/objects/supportedtypes?token={token}
```

```
[
  {
    "Type": "string",
    "Uri": "string"
  }
]
```

In seguito si deve effettuare una chiamata per ottenere la lista degli oggetti di una determinata tipologia:

```
http://domain/rest/v1/objecttypes?token={token}
```

```
[
  {
    "GUID": "string",
    "Code": "string",
    "Description": "string",
    "Type": "string",
    "Uri": "string",
    "Status": "string",
    "GeoInfo": {
      "Projection": "string",
      "Latitude": 0,
      "Longitude": 0,
      "Direction": 0
```

```
    },
    "LastUpdateTimeUTC": "string"
  }
]
```

Per ottenere le misure di un determinato oggetto è infine necessario invocare il seguente URL:

```
http://domain/rest/v1/objecttypes/{GUID}/measures?token={token}&fromDate={datefrom}&toDate={dateto}
```

```
{
  "Measures": [
    {
      "MeasureTimeUTC": "2016-06-08T15:45:00Z",
      "Sensors": [
        {
          "Data": [
            {
              "Group": "string",
              "Key": "string",
              "Value": "string"
            }
          ]
        }
      ]
    }
  ]
}
```

Per ottenere invece gli eventi in formato DENM è necessario invocare il seguente URL:

http://serveraddress/api/vehicleservice/denms/?applicationkey=21372328277482247&latitude=20.233333&longitude=20.4444555&altitude=240&heading=3.3&speed=50.4

Ottenendo la seguente risposta:

```
<ItsPduMessageList xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XM
LSchema-instance">
<ItsPduMessage>
<Base64Binary>
AQEAAAABgAAAAACAAInZ4k10AnagOFY7Wk6ShrSdIWAAAAAAGGoMAAAEBA==
</Base64Binary>
</ItsPduMessage>
<ItsPduMessage>
<Base64Binary>
AQEAAAABgAAAAACAAQnafzN4AnagOFY7Wk6SlrSdIXAAAAAAGGoMAAAEBA==
</Base64Binary>
</ItsPduMessage>
<ItsPduMessage>
<Base64Binary>
```

```
AQEAAAABgAAAAACAAYnafzN4AnagOFY7Wk6S1rSdIKAAAAAAGGoMAAAEBA==
</Base64Binary>
</ItsPduMessage>
<ItsPduMessage>
<Base64Binary>
AQEAAAABgAAAAACAAgnafzN4AnagOFY7Wk6SxrSdILAAAAAAGGoMAAAEBA==
</Base64Binary>
</ItsPduMessage>
</ItsPduMessageList>
```

Il messaggio DENM è codificato Base64Binary con codifica PER unaligned.

Per ottenere invece gli le previsioni semaforiche in formatio SPaT:

```
http://serveraddress/api/vehicleservice/spats?intersectionid=10001;10002?token=25dfweaf4566
```

Ottenendo in risposta:

```
<PredictionList xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSc
hema-instance">
<PredictionCollection>
  <Prediction>
    <IntersectionId>10001</IntersectionId>
    <TimeStamp>2014-05-20T11:09:14.6+02:00</TimeStamp>
    <Value>
        <States>
            <MovementState>
                <LaneNumbers>
                    <LaneNumber>1</LaneNumber>
                </LaneNumbers>
            <CurrentState>
                <NumberOfBits>4</NumberOfBits>
                <TrimZeroBits>false</TrimZeroBits>
                <State>0010xxxx</State>
            </CurrentState>
            <NextChanges>
            <Change>
                <MinTimeToChange>0</MinTimeToChange>
                <MaxTimeToChange>0</MaxTimeToChange>
                <LikelyTimeToChange>36</LikelyTimeToChange>
                <Confidence>0</Confidence>
                <PassState>false</PassState>
```

```
                <PredCnt>0</PredCnt>
            </Change>
        </States>
    </Value>
  </Prediction>
  <Prediction>
    <IntersectionId>10002</IntersectionId>
    <TimeStamp>2014-05-20T11:09:14.6+02:00</TimeStamp>
    <Value>
        <States>
            <MovementState>
                <LaneNumbers>
                    <LaneNumber>1</LaneNumber>
                </LaneNumbers>
            <CurrentState>
                <NumberOfBits>4</NumberOfBits>
                <TrimZeroBits>false</TrimZeroBits>
                <State>0010xxxx</State>
            </CurrentState>
            <NextChanges>
            <Change>
                <MinTimeToChange>0</MinTimeToChange>
                <MaxTimeToChange>0</MaxTimeToChange>
                <LikelyTimeToChange>36</LikelyTimeToChange>
                <Confidence>0</Confidence>
                <PassState>false</PassState>
                <PredCnt>0</PredCnt>
            </Change>
        </States>
    </Value>
  </Prediction>
</PredictionCollection>
```

7 API Sii-Mobility interop. e con altre centrali

7.1 API verso altre smart city (API18) (UNIFI)

Le API esportate verso altre Smart City sono una visione ridotta rispetto alle Smart City API complessive del sistema.

Si intende realizzare una rete di SM-NODE, ServiceMap Node, dove:

- Ogni **SM-NODE** prende in carico una zona/area territoriale in via primaria ma che contengono anche delle repliche di aree concettualmente lontane e limitrofe.
- Il sistema ha un tasso di replica X definito nella rete per ogni GRAFO che viene caricato (per ogni dataset).
- Ogni SM-NODE permette di caricare triple tramite il solito meccanismo del ServiceMap. Ad ogni dataset caricato viene attributo un:
    - ID Grafo.
    - License model;
    - Una valenza territoriale (anche ricavata automaticamente)
    - Un grado di replica: con eventuale lista degli altri SM-NODE autorizzati alla replica oppure semplicemente il numero MINIMO di repliche. Il grado di replica puo' essere anche 0 intendendo che non si autorizzano repliche.
- si ha un meccanismo di cache che accumula in locale richieste che sconfinano rispetto all'area geografica di competenza, su base GPSed

Scenari di riferimento:

- Aggiunta SM-NODE:
    - Un nuovo SM-NODE deve negoziare/dichiarare una responsabilità geografica che gli viene concessa o meno sulla base delle precedenti attribuzioni.
    - La mappa delle attribuzioni è scritta nel Server centrale di Km4City. Gli SM-NODE leggono tale tabella di attribuzioni sia per
        - capire dove propagare le richieste che per
        - concedere le richieste che potrebbero non essere giustificate.
    - Senza il doppio controllo si potrebbero chiedere cose anche non essendo autorizzati.
    - Magri tramite protocollo HTTPS o migrazione di triple file per la migrazione dei GRAFI
    - Su tale base si prende in carico una copia di tale parte più altre info e parti per ridondare il tutto (X repliche) con Tabella SM-NODE tipo…

ID NODE	SM-NODE	IP	Area	Lista Grafi: grafo:repliche:dove
SMN1	https://sm1.km4city.org	150.217.XX.YY	{descrizione area}	G1-SMN1:3:ANY; Gwifi-SMN1:0:none; G4-SMN1:2:SMN3,SMN2 G5-SMN1:ALL;ANY;
SMN2	https://sm2.km4city.org			
SMN3				

Dove G1-SMN1, Gwifi-SMN1, G4-SMN1 sono nomi di grafi univoci per la rete. ALL significa che non si hanno upperbound sul numero di repliche.

- Verifica dello stato della rete di SM-NODE:
    - Ogni SM-NODE accede ad un file di Tabella SM-NODE fornita da server Km4City.org all'installazione ma che viene aggiornata ogni 10 minuti scaricandola come lista nodi e aree di competenza.
        - SM-NODE: identificativo di domain,
        - Aree di competenza:…….
        - Data cached
    - Il SM-NODE non accetta connessioni se non da quelli che appartengono a tale lista scaricata da Km4City. Ed accetta richieste di accesso dati per migrazione solo dagli SM-NODE autorizzati ad accedere a tali dati sulla base di tale tabella.
    - Chiede agli altri SM-NODE elencati il loro stato e lo riporta in un front di stato e monitoraggio: SM-NODE stato (on/off alive, numero richieste per day, data ultima lettura tabella nodi, numero di triple, lista aree di competenza ricevuta, lista di elementi cached, etc..)
- Aggiunta dati da qualsiasi SM-NODE della rete dovrebbe essere indifferente anche se
    - il Sistema dovrebbe ricordarsi chi gli ha aggiunti in modo da poterli anche togliere o moderare (non accettare)
    - al momento in cui un SM-NODE A aggiunge dati fa un'operazione di Announce verso il server centrale. Questo delega altri X SM-NODE a fare delle richieste al SM-NODE A per avere delle copie di tali dati. Oppure il Server centrale semplicemente riceve dal SM-NODE A il pacco dei dati in termini di triple che SM-NODE A ha caricato e decide come fare per replicarlo lui su altri SM-NODE ?????
- Al momento di dismiss / abbandono di un SM-NODE si deve, a livello centrale o in modo distribuito, re-distribuire il carico sugli altri SM-NODE sulla base della coverage complessiva territoriale. Si deve pertanto a livello centrale definire la coverage territoriale alla quale si tende come lista di aree… anche disgiunte.
    - L'abbandono / dismiss puo' essere (i) notificato oppure (ii) identificato in base alla verifica di stato della rete di SM-NODE dopo un certo time out se l'SM-NODE in questione non risponde.
- Accesso alle API e richiesta dati da qualunque SM-NODE, che dovrebbe ridirigere la richiesta su quello che ha il dato vero e proprio in base alla sua competenza primaria di zona.
    - Ogni client dovrebbe avere una lista di possibili SM-NODE con descrizione di zone di loro competenza… pertanto direttamente nel client far partire la richiesta in modo diretto al SM-NODE di competenza.. Oppure fare la richiesta ad un unico punto e poi delegare… ma questa soluzione e' meno paritaria… piu' facile da controllare anche come volume dati e flussi.
    - Ogni SM-NODE la lista e la mappa delle attribuzione delle zone di come i SM-NODE si dividono il territorio e lo ridondano
    - Richieste GPSed possono facilmente essere instradate verso il o i SM-NODE giusti
    - Richieste per ricerca testuale sono più difficili da eseguire perché potenzialmente potrebbero richiedere il coinvolgimento di tutta la rete di SM-NODE
        - vengono comunque gestite con un raggio GPS limitato dalle coordinate della posizione del chiamante che comunque deve essere accessibile.
        - solo se l'utente esplicita la regione vengono accettate anche non vicine, in quel caso viene delegato al nodo opportuno. Per esempio: via Rossi a Roma….. viene tradotta nella richiesta al server SM-NODE di Roma di via Rossi.

Visione tecnica:

- definizione e realizzazione di API generali per la richiesta di questi servizi
- definizione di API di integrazione e scambio fra SM-NODE

- realizzazione di una WebApp per Tomcat/Java che possa fare questo servizio di P2P sharing e gestione degli announce, etc. su un server dotato di MySQL anche con meccanismo di autenticazione, etc.. fra nodi in modo da evitare che altri non SM-NODE (non registrati come IP) possano accedervi.

Ogni nodo ha uno stato:

- **NEW**, se il nodo è appena stato aggiunto nella tabella dei nodi e deve acquisire i dati di sua competenza
- **LOADING**, se il nodo sta acquisendo dagli altri nodi i grafi di sua competenza ma ancora non è operativo
- **WAITING_CHECK**, se il nodo è in attesa di check che lo stato sia corretto e funzionante
- **RUNNING**, se il nodo ha acquisito tutti grafi di sua competenza e può essere usato dagli altri nodi
- **OFFLINE**, se il nodo non è più raggiungibile
- **OUT**, se il nodo è stato dismesso o sarà OFFLINE per un lungo periodo di tempo

API definite sono:

- **SuperCityAPI**, sono API REST per l'accesso ai dati presenti nella tabella di competenza dei nodi
    - **GET /api/v1/graphs**
        - fornisce l'elenco dei grafi presenti nel sistema con indicazioni statistiche sulla loro consistenza e contenuto e con elenco dei nodi che lo possiedono e quale è il nodo di competenza primaria
    - **POST /api/v1/graphs/{graph_id}**
        - un nodo può aggiornare le statistiche dei grafi di sua competenza primaria.
    - **GET /api/v1/sm-nodes**
        - fornisce l'elenco dei nodi presenti con il loro stato, sulla base dell'IP della richiesta viene identificato quale è il nodo richiedente
    - **POST /api/v1/sm-nodes/{node_id}/state**
        - accetta la notifica di stato fornita dal nodo (verificato tramite IP) per passare da NEW a LOADING a WAITING_CHECK a RUNNING, la notifica di LOADING/RUNNING deve essere inviata regolarmente (ogni 10 min) altrimenti passa a stato OFFLINE. Oltre allo stato nuovo deve anche essere inviato l'elenco dei grafi presenti ed il loro stato se OK, LOADING con percentuale di caricamento, WAIT per indicare che il grafo è in attesa di essere caricato o FAIL (con messaggio di fallimento) per indicare che per qualche motivo il grafo non può essere caricato.
- **PeerCityAPI** divise in:
    - **FrontCityAPI** sono una replica delle SmartCityAPI, che chiamano le SmartCityAPI dei SM-NODE opportuni
    - **InterCityAPI** permettono lo scambio dati tra SM-NODE diversi
        - **GET /api/v1/graphs/{graph_id}**
            - richiede il download di un grafo, viene fornito solo se richiesto da un IP che può avere una copia del grafo
        - **GET /api/v1/graphs/{graph_id}/stats**
            - restituisce le statistiche sul grafo richiesto, se passato il parametro force, evita qualisasi caching e ricalcola tutte le statistiche.
        - **GET /api/v1/triples?query=**
            - restituisce a un qualsiasi nodo le triple di una query di tipo CONSTRUCT, viene usata per acquisire parti di uno o più grafi da

tenere in cache. La richiesta viene accettata solo se da un IP di un qualsiasi nodo

Funzionamento:
SM-NODE:

- acquisisce regolarmente lo stato della tabella dei nodi dal supercity server.
- controlla il suo stato e se NEW inizia a scaricare i grafi che deve gestire richiedendoli dai nodi RUNNING che li hanno e inizia a notificare al super city server lo stato LOADING
- quando ha finito di scaricare i dataset li carica sulla KB e passa allo stato WAITING_CHECK
- quando è nello stato WAITING CHECK e arriva da supercity server indicazione che il suo stato è RUNNING cambia di stato e notifica regolarmente il suo stato RUNNING (lo stato waiting_check serve ad evitare che altri nodi chiamino le API prima che sia verificato il comportamento corretto del nodo)
- comunque deve sempre controllare se nella tabella acquisita son indicati per il nodo altri dataset che devono essere scaricati ed aggiunti.
- quando viene richiesta una FrontCityAPI stabilisce sulla base delle informazioni nella tabella o in base a una cache a quale nodo/nodi inviare la chiamata, se il nodo non è quello locale fa un forward oppure fa da proxy. Se la query coinvolge il nodo stesso e uno o più nodi fa la chiamata agli altri nodi della stessa api e fa merge dei risultati, in caso di fallimento di una o più chiamate ad altri nodi invia quello che ha ottenuto. In alcuni casi può anche decidere di richiedere parte dei grafi da altri nodi facendo una chiamata come query SPARQL CONSTRUCT per prelevare alcuni dati e salvarli in locale.

8 Acronimi

- API: Application Program Interface
- AVL: Automatic vehicle location
- AVM: Automatic Vehicle Monitoring
- BDaaS: Big Data as a Service
- CAP principle: Consistency Availability Partition Tolerance principle
- CBB: Content Based Billing
- CBB: Content Based Billing
- CEN: European Committee for Standardization
- DBMS: database management system
- FCD: Floating Cellular Data
- GPRS: General packet radio service
- GPS: Global positioning System
- GSM: Global System for Mobile
- ICT: Information and Communication Technologies
- ITS: Intelligent Transport Systems
- LCD: liquid-crystal display
- LOD: linked open data
- MC: Mobile Collector
- MMS: Multimedia Messaging Service
- NLP: Natural Language Processing
- NoSQL: no SQL database
- OD: open data
- OD: Open Data
- OGC: Open Geospatial Consortium
- OWL: Web Ontology Language
- PA: Pubblica Amministrazione

- PMI: Piccola e Media Impresa
- PMS: Private Mobile Systems
- PMV: Pannelli a Messaggio Variabile
- POS: part-of-speech
- RDF: Resource Description Framework
- RFID: Radio Frequency IDentification o Identificazione a radio frequenza
- RTTI: Real-time Travel & Traffic Information
- SDI: Spatial Data Infrastructures
- SII: sistema di interoperabilità integrato
- SIMONE: progetto Simone
- SMS: Short Message Service
- SN: social networking, oppure sensor network
- SOA: Service Oriente Architecture
- SOAP: Simple Object Access Protocol
- SSAMM: Agenzia per la Mobilità Metropolitana strumenti di supporto, TOSCANA
- TMC: Traffic Message Channel
- TPEG: Transport Protocol Experts Group
- TPL: gestore trasporto pubblico locale
- UML: Unified Modeling Language
- UMTS: Universal Mobile Telecommunications System
- UTC: Urban Traffic Control
- UUDI: Universal Description Discovery and Integration
- V2I: Vehicle-to-Infrastructure
- V2V: vehicle-to-vehicle
- VMS: Variable Message Sign
- VWSN: Vehicular Wireless Sensor Networks
- W3C: World Wide Web Consortium
- WSD: Word Sense Disambiguation
- WSDL: Web Services Description Language
- WSN: Wireless Sensor Networks
- XMI: XML Metadata Interchange standard di OMG
- XML: Extensible Markup Language
- ZTL: Zona a Traffico Limitato