Sii-Mobility

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

Trasporti e Mobilità Terrestre, SCN_00112

Deliverable ID: DE2.11a (versione a)

Titolo: Sistema SII Ottimizzato in varie versioni con completezza crescente

Data corrente	30-06-2017
Versione (solo il responsabile puo' cambiare versione	0.9
Stato (draft, final)	finale
Livello di accesso (solo consorzio, pubblico)	Consorzio
WP	WP2
Natura (report, report e software, report e HW..)	Report
Data di consegna attesa	30-06-2017
Data di consegna effettiva	30-06-2017
Referente primario, coordinatore del documento	Paolo Nesi, Paolo.nesi@unifi.it
Contributor	Nesi, Bellini, Cenni
Coordinatore responsabile del progetto	Paolo Nesi, UNIFI, paolo.nesi@unifi.it

Sommario

1 Executive Summary 3
2 Architettura generale funzionale 4
3 Componenti della Architettura 8
3.1 Big Data Warehouse 8
3.1.1 Processi ETL e/o Java di data ingestion 8
3.1.2 Data Agreement per l'accesso ai dati 9
3.1.3 Data Agreement per l'accesso ai dati gestori flotte 9
3.1.4 DataGate e CKAN 9
3.1.5 Sensor Server and Manager, SSM 10
3.1.6 IOT Management 11
3.1.7 Distributed Smart City Engine, DiSCES 13
3.2 Supporto alle decisioni 14
3.2.1 Dashboard Engine and Dashboard Builder 14
3.2.2 Notificator 15
3.2.3 Smart Decision Support, SmartDS 17
3.3 Social Monitoring 18
3.3.1 Participation Platform 18
3.3.2 Social Media Crawler and Manager 18
3.4 User Management, Engagement, Stimulation 20
3.4.1 User Crowd Sourcing Manager 20
3.4.2 User Engagement on Demand 21
3.4.3 Suggestion on Demand, Recommender 23
3.4.4 User Assistant, PAVAL, *Conversational Recommender System* 24
3.4.5 User Profiler 25
3.5 Tools Support and System Management 28
3.5.1 Support for Authentication and Services Access, LDAP 28
3.5.2 Monitoring Service Interface: monitoring smart city solution 29
3.6 Data Analytics 30
3.6.1 Traffic Flow Monitoring 31
3.6.2 Traffic Reconstruction 32
3.6.3 Parking Predictions 33
3.6.4 Origin Destination Matrix based on Mobile Data 33
3.6.5 Algorithm/Process Loader: 33
3.6.6 Routing: Path Planner 34
3.7 Knowledge base services 35
3.7.1 Km4City Ontology and Knowledge Base, version 1.6.4 35
3.7.2 ServiceMap based on Km4City Knowledge base 36
3.7.3 ServiceMap 3D 38
3.7.4 Loading Shapes and Paths for Search on ServiceMap/Km4City 39
3.7.5 SPARQL interface, Conditional Access System + Licensing 39
3.7.6 Linked Open Graph, visual browsing on Km4City 39
3.7.7 Statistic Data Map 40
3.8 Smart City API 40
3.8.1 Mobile e Web APP: 43
3.9 Ticketing and Booking 43

1 Executive Summary

Con il presente documento si intende dare evidenza della prima ottimizzazione del sistema Sii di Sii-Mobility.

Per il progetto Sii Mobility DISIT lab di UNIFI mettere a disposizione le risorse all'interno del proprio DataCenter. Viene inoltre utilizzato il datacenter di IN20 per il cloning di alcune funzioni in modo da avere delle soluzioni di hotspare, di balancing e gestione distribuita dei dati. Le due soluzioni: il master DISIT Datacenter e IN20 Datacenter sono basate su piattaforma VMware vSphere. Tutte le componenti nei due data center essenziali (logiche e fisiche) alla corretta fruizione del servizio sono monitorate costantemente da un sistema che segnala eventuali malfunzionamenti e/o disservizi ai vari referenti. Tutte le componenti funzionali e necessarie all'implementazione del della soluzione cloud Sii-Mobility (VM, storage, switch, firewall) sono progettate in modo ridondato così da garantire la massima affidabilità, sicurezza e disponibilità rendendo praticamente lo stesso resiliente al guasto. Nel deliverable DE3.22 è stata presentata una progettazione di regime del sistema mentre in questo deliverable viene presentata la situazione attuale e i risultati della prima validazione e sperimentazione.

Il presente documento presenta lo stato della soluzione Sii di Sii-Mobility e la prima ottimizzazione. Tale ottimizzazione ha determinato una ristrutturazione dell'architettura. Le motivazioni primarie per la ristrutturazione ed ottimizzazione sono le seguenti.

Nella prima versione del sistema SII il processo di acquisizione portava direttamente nell'RDF le triple con i dati relativi ai sensori e ai servizi real time. Questo ha determinato delle criticità in termini di (i) prestazioni di lettura dell'RDF store via Smart City API (se vi sono troppi processi che scrivono le query nello store semantico sono più lente), (ii) limitata scalabilità dell'RDF store per l'accesso a dati storici, (iii) necessità di effettuare e controllare l'accesso ai dati storici anche tramite soluzioni di navigazione tradizionale, accessibili facilmente da R e da altre soluzioni che sono performanti per l'accesso di tabelle e meno di per l'accesso a grafi.

Un altro punto di critico è sempre connesso all'RDF store che presenta delle limitazioni di scalabilità se si hanno territori molto ampi con molte informazioni e relazioni. La popolazione, il caricamento, del RDF store con il grafo strade puo' non essere semplice se il grafo di base completo non è accessibile. Una possibile soluzione e' partire da Open Street Map ma alcune informazioni possono non essere accessibili come per esempio: le svolte, i numeri civici, le preferenziali, le ordinanze, etc. Queste informazioni devono essere reperite in altro modo ed integrate, altrimenti se non sono accessibili elaborazioni come il routing, e il traffic reconstruction non sono fattibili, o lo sono con errori grossolani.

Non fanno parte di questo deliverable lo stato e l'evoluzione di componenti esterni al SII come:

- Kit Veicolari, Video Decisore, Direzionatore, Kit rastrelliera/bike (ECM e vari partner)
- Totem (TIME e vari partner)
- App Mobile di vario tipo (UNIFI e vari partner)
- Sistema di bigliettazione integrato (NEGENTIS principalmente)

2 Architettura generale funzionale

Sii-Mobility presenta una soluzione per abilitare un'ampia gamma di servizi al cittadino e commerciali in connessione e integrati con il sistema di mobilità. La soluzione in essere colleziona dati puntuali e attualizzati in tempo reale da varie fonti; analizza flussi di dati con varie tipologie di algoritmi (predittive, early warning, traiettorie, mappe OD, etc.) e produce azioni e informazioni tramite applicazioni web e mobili, totem informativi, ecc.; mette a disposizione dati elaborati e puntuali, che possono essere usati da PA, gestori, e imprese per produrre servizi più efficaci ed efficienti, e anche nuovi servizi integrati. In ottica Smart Lab o Living Lab, permette anche a PA e PMI di caricare nuovi 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, di business intelligence, early warning, prediction, 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.

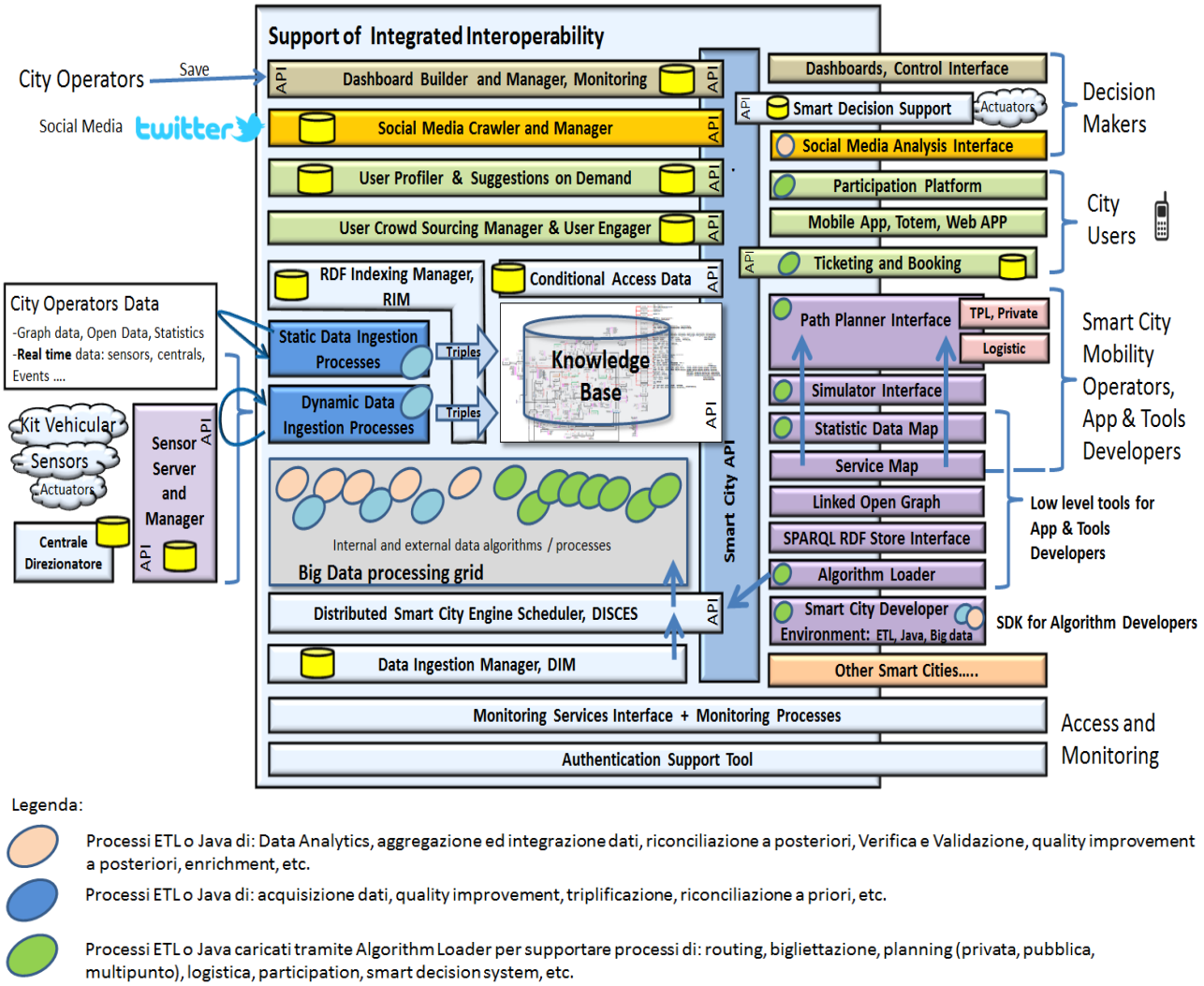


Figura 1: Architettura Sii-Mobility generica

Il sistema **Sii-Mobility** è composto da vari elementi. Le ellissi/cerchi colorate/i in verde, rosa e celeste sono processi ETL o Java che vengono messi in esecuzione nel back office e gestiti come

processi a cura del **Distributed Smart City Engine Scheduler, DISCES.** Lo stesso strumento è a servizio di vari tool anche di front end che ne governano il comportamento permettendo la messa in esecuzione di processi e la lettura del suo stato. Sono inoltre presenti elementi che non sono descritti in dettaglio in questo documento essendo elementi che non vengono allocati sui server di Sii-Mobility (cioè non fanno parte del SII) come: mobile APP, kit veicolari, sensori, pannelli a messaggio variabile, attuatori, etc.

Nel DE1.2a è stata riportata la specifica di integrazione del sistema Sii-mobility.
Nel DE2.12a è stata presentata la prima validazione della soluzione progettata in DE1.2a e sviluppata nel primo anno in riferimento al deploy corrispondente su cloud e la valutazione delle prestazioni.
Nel deliverable DE3.22 è stata presentata una progettazione di regime del sistema mentre in questo deliverable viene presentata la situazione attuale e i risultati della prima ottimizzazione a valle delle prima validazione e sperimentazione.

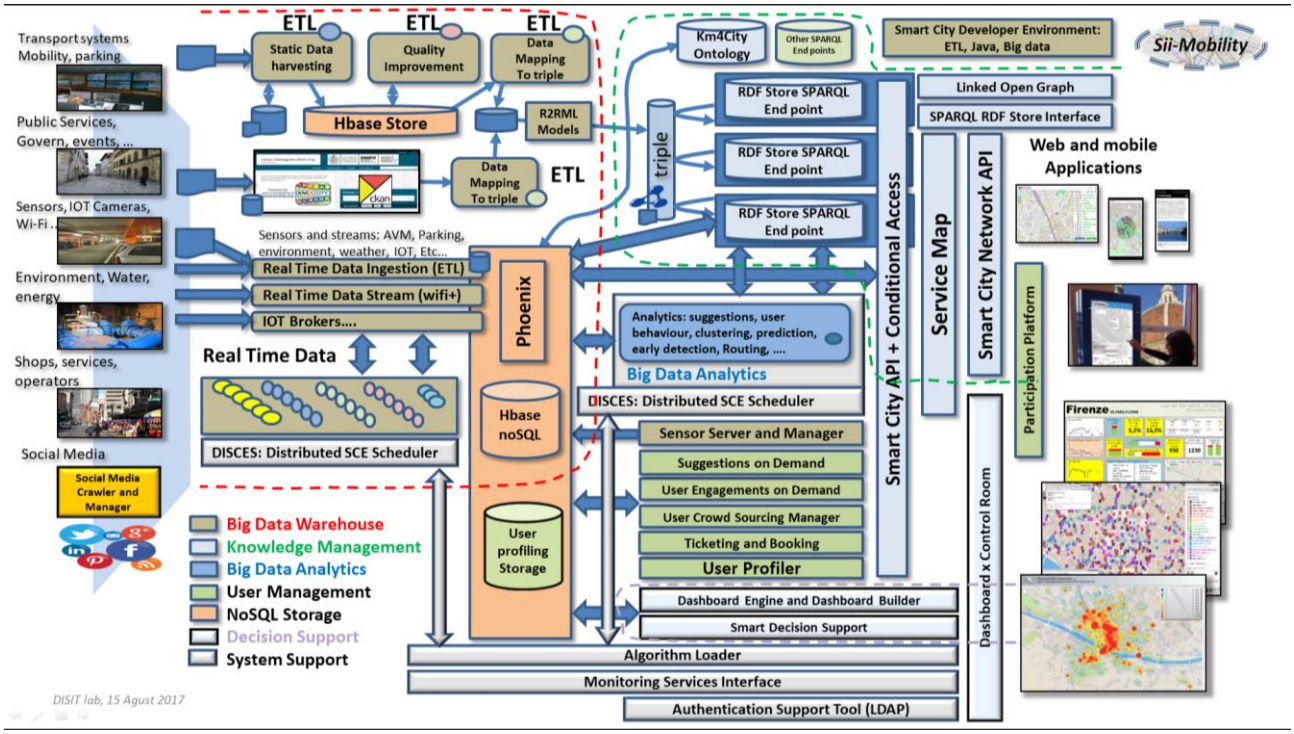
L'RDF store presenta delle limitazioni di scalabilità se si hanno territori molto ampi con molte informazioni e relazioni, e se i dati real time sono molti. Per esempio dopo un anno di acquisizioni in tempo reale, e' possibile che il volume dei dati statici sia circa il 50% del volume generale, mentre quello strettamente necessario rimane del 50%.

Nella prima versione del sistema a M12 il processo di acquisizione dati real time portava direttamente nell'RDF le triple con i dati relativi ai sensori e ai servizi. Questo ha determinato delle criticità in termini di

- prestazioni di accesso all'RDF store via SPARQL come Servicemap e/o Smart City API (se vi sono troppi processi che scrivono le query anche solo in lettura nello store semantico sono più lente). Si noti che tipicamente l'RDF store con la knowledge base Km4City viene utilizzato nel 99% dei casi per fare richieste di sola lettura con e senza inferenza ma che sfruttano le relazioni e la conoscenza del grafo della città.
- limitata scalabilità dell'RDF store per l'accesso a dati storici. Se vi sono molti dati storici, istanze di una stessa entità (per esempio un sensore), queste sono tipicamente solo utili ai fini del data analytics e non per la navigazione. Sono pertanto acceduti in modo sporadico e non dalla maggior parte degli utenti ma solo dai processi di back office. Pertanto avere milioni di triple che modellano dati storici puo' essere non utile, ma porta ad appesantire inutilmente i tempi di accesso al grafo.
- necessità di effettuare e controllare l'accesso ai dati storici anche tramite soluzioni di navigazione tradizionale, accessibili facilmente da R, Java, etc., come da altre soluzioni che sono performanti per l'accesso a tabelle e meno per l'accesso a grafi RDF.

Un altro punto di critico è collegato all'informazione contenuta nell'RDF store e al processo per acquisirla. Il caricamento del RDF store con il grafo strade corretto per il territorio puo' non essere semplice se il grafo di base completo non è accessibile o lo è parzialmente. In assenza di un grafo strade completo e dettagliato con il quale popolare l'RDF store, una possibile soluzione è partire da Open Street Map ma alcune informazioni possono non essere accessibili come per esempio: le svolte, i numeri civici, le preferenziali, le ordinanze, etc. In effetti, i dati di OSM non sono completi in ogni area, in molti casi i dati strutturali sono parziali (sono presenti nelle città principali ma non per le zone rurali e per le città minori). Queste informazioni devono poter essere reperite in altro modo ed integrate, altrimenti se non sono possibili elaborazioni come il routing, e il traffic reconstruction, o lo sono con errori grossolani.

Nella Figura seguente viene riportata la nuova architettura del Sii di Sii-Mobility, dove viene messo in evidenza: la nuova struttura della parte di data warehouse, Big Data Analytics, RDF store e tools, Decision Support, le nuove Smart City API, User Management, System Support, etc.



La nuova architettura presenta un sistema di Data Warehouse ed acquisizione dati completamente ristrutturato in cui:

- I **processi ETL relativi ai dati statici** (POI, posizione statica elementi in mappa anche con informazioni polyline e shape associate, in futuro) acquisiscono tali informazioni tramite la piattaforma precedentemente collaudata, con una serie di processi ETL: acquisizione, harvesting, quality improvement, data mapping. Dove il data mapping viene effettuato con un processo Karma che trasforma il dato tabella in Hbase in triple per lo RDF store, ServiceMap e Smart City API. Per l'acquisizione dei dati statici è possibile procedere in due modi, tramite
    - A) la realizzazione di una serie di ETL oppure
    - B) l'uso di un tool di accelerazione: il DataGate è un modulo di CKAN, sviluppato nel progetto. CKAN è un tool per il collezionamento di Open Data per le pubbliche amministrazioni. Molte PA lo utilizzano per la pubblicazione e il collezionamento dei loro Open Data. Il DataGate permette di caricare dataset in modo semplice tramite il caricamento degli stessi come file excel o in altri formati. Il DataGate passa a ripulirli dai campi non qualitativamente accettabili, e a caricarli nel sistema e nella Knowledge Base (RDF store) senza scrivere una linea di codice o un ETL. Il DataGate effettua alcune correzioni sui dati in automatico, per esempio, completamenti sui CAP, sui riferimenti alle province, regioni, etc. splitting, di indirizzi complessi in: via, numero civico e Cap, etc. Il DataGate utilizza nel suo back office un processo ETL automatizzato realizzato per gestire un'ampia gamma di casistiche.
- I **processi ETL relativi ai dati real time** sono tipicamente più semplici di quelli statici e vengono schedulati in modo periodico dal DISCES (alcuni di questi sono ogni pochi minuti e/o secondi, altri 1 volta ogni ora, 1 volta al giorno, etc.). Le informazioni statiche sono tipicamente i metadati, le posizioni GPS, mentre quelle dinamiche i valori dei sensori. Pertanto quelli real time sono più semplici e si riferiscono ad un insieme di elementi statici caricati in precedenza,

sempre tramite un processo ETL per dati statici. Per questa ragione i processi ETL per dati real time sono tipicamente snelli e diretti a una singola fase. In passato questi ETL per Real Time producevano triple per mandarle allo RDF store; nella nuova soluzione, questi processi scrivono dati acquisiti direttamente in Hbase e si fermano, i dati real time non arrivano nell'RDF store. Tali dati real time in Hbase/phoenix vengono direttamente acceduti da (i) Smart City API, (ii) ServiceMap (RDF Store), (iii) algoritmi di data Analytic, e (iv) dal Dashboard Builder tramite un adapter SQL realizzato in Apache Phoenix (JDBC, ODBC). Lo stesso adapter viene utilizzato per scrivere dentro HBase come dato tabellato Phoenix che permette di definire indici, etc.

  - Vi possono essere delle eccezioni.
    - Per esempio i dati Real Time che modificano la struttura della città come: (i) le ordinanze che cambiano i sensi di marcia o gli accessi, (ii) gli orari delle ZTL, (iii) i percorsi delle linee TPL e i loro orari pianificati, etc. Questi modificano la struttura del grafo e influenzano la struttura del grafo e pertanto l'RDF store.
    - Questi dati Real Time dovrebbero sovrascrivere i vecchi dati non più correnti, mentre lo storico dovrebbe comunque essere presente in HBase. In questo modo si limita la crescita dell'RDF che rimane nel sistema come indice reticolare fra le entità della smart city.

- I **processi di acquisizione dati in stream** sono online e quindi in esecuzione continua H24/7 e portano i dati direttamente in noSQL database per esempio Hbase tramite Phoenix oppure anche Mongo o anche HDFS direttamente. Portare i dati in Hbase tramite Phoenix permette di averli in un sistema accessibile per gli altri strumenti della soluzione via SQL.
- I **processi da e verso IOT.** Le soluzioni di IOT sono basate su dispositivi (sensori e/o attuatori) che fornisco dati in real time e permetto la ricezione di dati sempre in real time (per esempio un irrigatore che ti da l'umidità del suolo ed accetta comandi on/off di irrigazione). Queste operazioni sono effettuate tramite degli IOT Broker più o meno capaci di gestire protocolli multipli, formati multipli, sicurezza, la directory di IOT device connessi e registrati. I processi che leggono e agiscono sugli IOT broker possono essere direttamente connessi in lettura con lo storage HBase / Phoenix ed in scrittura per agire sugli attuatori direttamente dalle dashboard o dagli algoritmi che producono l'attuazione.

Di seguito sono riportate le funzionalità dei principali moduli e strumenti della soluzione facendo riferimento all'impatto che la nuova architettura ha avuto ed avrà sugli stessi strumenti e sull'avanzamento che hanno subito o che dovranno effettuare nel passare dalla versione 1.0 rilasciata a M12 alla versione 2.0 rilasciata a M18. Questa seconda versione e' al momento largamente operativa anche se non completamente.

3 Componenti della Architettura

3.1 Big Data Warehouse

Questo componente include tutti processi di acquisizione ed il big data storage per i dati acquisiti in modo da storicizzarli per il Data Analytics e renderli accessibili in real time per le Smart City API e Dashboard. La struttura del Big Data Warehouse ha subito grandi cambiamenti rispetto alla versione 1.0 della soluzione Sii di Sii-Mobility. In questa area vi sono i processi di acquisizione, gli strumenti di gestione dei data set, lo storage. Si veda in precedenza una descrizione dei processi di acquisizione.

Per la gestione dei data set in ingresso, le soluzioni DIM e RIM sviluppate in prima fase sono al momento poco utilizzate. Attualmente i processi per l'acquisizione dati sono sempre sviluppati in ELT e vengono direttamente allocati sullo scheduler DISCES senza per questo collezionare i metadati e licenze su DIM. Si ricorda che le informazioni di licenza sono su DIM come i metadati che potrebbero essere utili per la gestione. Anche il RIM non viene utilizzato poiché si è passati da una costruzione dell'indice RDF su GraphDB molto lenta ad una costruzione su Virtuoso molto veloce e pertanto facile da riprodurre con uno script.

- **DIM:** data ingestion manager: gestore dei processi di ingestion. Comprende i metadati relativi ai data set caricati e fra questi metadati le informazioni di licensing.
- **RIM:** RDF Indexing Manager: Tool per l'indicizzazione del nuovo store RDF. Lavora sia con OWLIM che con Virtuoso. Recentemente si lavora direttamente con Virtuoso per questioni di prestazioni.

E' stato inoltre sviluppato un nuovo strumento per accelerare i processi di caricamento detto DataGate, descritto in seguito.

Sono pertanto **moduli consigliati minimi** per questo processo di Big Data Warehouse:

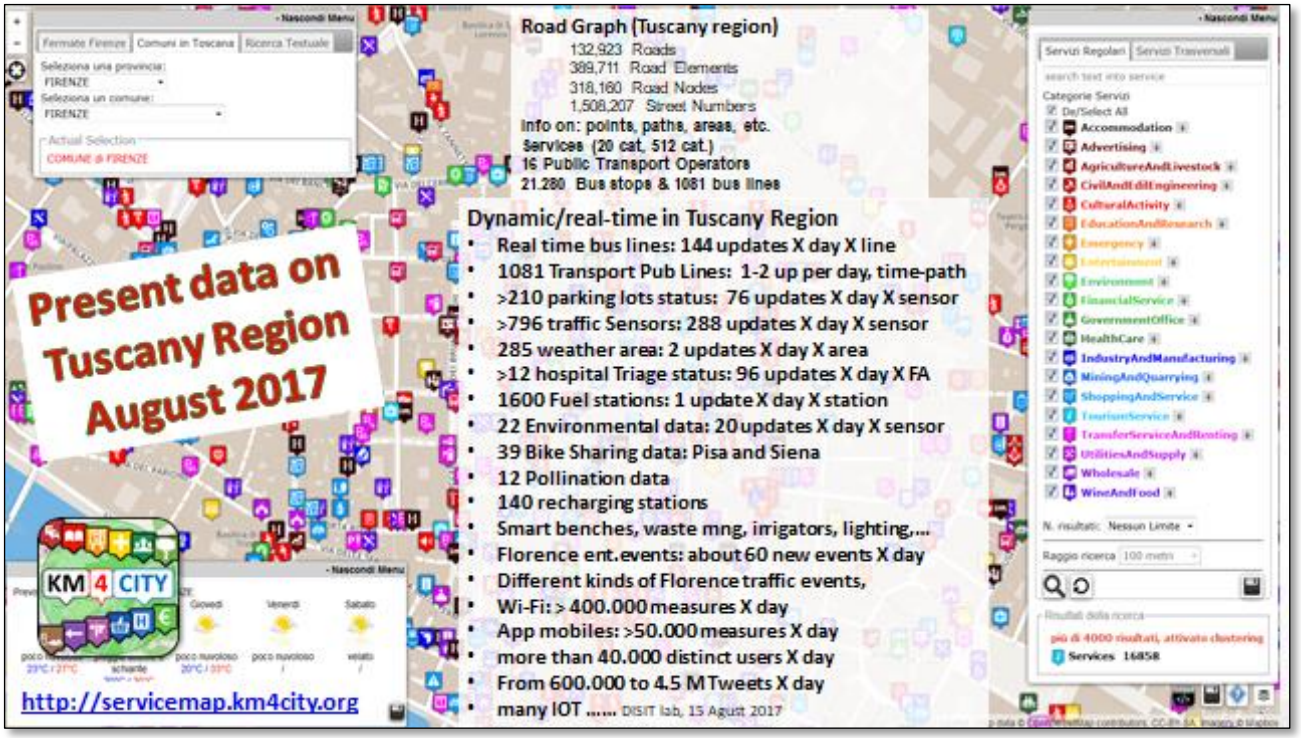
- **DataGate** con CKAN, utilizza MySQL e anche ETL process
- **ETL** development tool con Penthao Kettle e vari ETL già sviluppati
- **DISCES** per lo scheduling dei processi che a sua volta utilizza MySQL
- **Sensor Server Manager**, dipendente da MySQL oppure da Phoenix/Hbase
- **IOT broker** come Mosquitto oppure IOT Orion Broker di FiWare (versione modificata da Sii-Mobility per lavorare con Phoenix/Hbase)

3.1.1 Processi ETL e/o Java di data ingestion

Molti dei processi di caricamento dati sono già attivi altri devono essere sviluppati. I **Processi ETL e/o Java per il caricamento dei dati che** vengono eseguiti nei processing node sotto il controllo dello scheduler DISCES. Tali processi sono principalmente per il caricamento e la riconciliazione dei dati real time, fra questi come da figura:

- TPL, AVM
- Previsioni meteo
- Costi della benzina
- Sensori flusso traffico
- Previsioni meteo
- Stato del triage
- Stato dei parcheggi
- Stato del bike sharing
- Variabili ambiente e pollinazione
- Sensori / attuatori IOT

- Etc.



Sono pertanto **moduli consigliati minimi** per questo processo di Big Data Warehouse:

- MINIMO: **ETL** development tool con Penthao Kettle
- vari ETL già sviluppati accessibili da GitHub di DISIT lab

WHAT NEXT: *sviluppo di altri ETL per altri tipi di dati Real Time che arrivano in piattaforma, mentre i dati statici vengono caricati tramite DataGate/CKAN*

3.1.2 Data Agreement per l'accesso ai dati

Il documento di Data Agreement è stato sviluppato e distribuito ai partner in modo che questi possano procedere in parallelo a contattare i vari enti per definiti gli accordi di accesso ai dati.

3.1.3 Data Agreement per l'accesso ai dati gestori flotte

Il documento di Data Agreement è stato sviluppato e distribuito ai partner in modo che questi possano procedere in parallelo a contattare i vari enti per definiti gli accordi di accesso ai dati.

3.1.4 DataGate e CKAN

DataGate è un modulo di CKAN, sviluppato nel progetto (**http://DataGate.disit.org**). CKAN è un tool per il collezionamento e la pubblicazione di Open Data per le pubbliche amministrazioni, il più diffuso in assoluto al mondo. Il Servizio DataGate citato e sviluppato è in sostanza un'istanza di un CKAN con installato il modulo DataGate installato. Questo:

- permette di caricare in modo semplice dei data set tramite file contenenti fogli excel o altri formati.
- Permette di caricare anche i metadati relativi al data set in ingresso che possono essere utilizzati per cercarlo e ripubblicarlo sempre tramite il DataGate.
- e' in grado di analizzare ripulire i campi non qualitativamente accettabili, e caricarli nel sistema e nella Knowledge Base (RDF store) senza scrivere una linea di codice o un ETL.

- Accetta dati in un certo formato come suggerito dal sistema stesso, ma è molto flessibile, il formato richiesto non è estremamente rigido e complesso, sono forniti degli esempi.
- effettua correzioni sui dati in automatico, ma anche completamenti per esempio sui CAP, sui riferimenti alle province, regioni, etc.
- utilizza nel suo back office un processo ETL automatizzato realizzato per gestire un'ampia gamma di casistiche.
- Comunica con altri CKAN che hanno il DataGate per lo scambio di dati in automatico, o comunque con altri CKAN sempre per lo scambio di dati ma in formato CKAN.

Sono pertanto **moduli consigliati minimi** per questo processo di Big Data Warehouse:

- **DataGate** con CKAN, utilizza MySQL e anche ETL process
- **ETL** development tool con Penthao Kettle e vari ETL già sviluppati ed accessibili da Github di DISIT lab

WHAT NEXT: *DataGate deve poter migliorare alcuni aspetti rendendo più flessibile il caricamento di formati che contengono anche PolyLine aperte e/o chiuse, e poter accettare e trattare anche file di grosse dimensioni, con decine di migliaia di righe nelle tabelle, etc.*

3.1.5 Sensor Server and Manager, SSM

Sensor Server and Manager espone delle API che permettono alle applicazioni mobili e web di salvare il loro stato. Tali applicazioni in modo periodico, salvando il loro stato in PUSH, colgono anche l'occasione di ricevere dei comandi in PULL: di engagement, di suggestion, etc. Il sistema in questo modo è in grado di tracciare il comportamento utente: posizioni, movimenti, velocita', etc.
Nello stato salvato nel SSM vi sono informazioni complesse e complete quali per esempio:

- Posizione GPS, accelerazione, velocita', stato del telefono in misura (FG, BG, navigazione), etc..
- ID anonimo, tipo di telefono, sistema operativo,
- codice di APP, etc.
    - Codice di APP del tipo: Toscana, Firenze, Car Sharing APP, direzionatore, etc.
- Dati dai Kit Veicolari e relativi sensori (in progress)
- Dati da Direzionatori e controllori di ZL (in progress)
- Dati dai dongle ODB2 (in progress)
- Etc.

Il SSM colleziona questi dati in MySQL e/o su Hbase/Phoenix per successive elaborazioni di analytics, nel primo caso in modo tradizionale (ETL, Java, R), nel secondo caso in modo tradizionale (ETL, Java, R, etc.) e/o MapReduce/Spark.
I kit veicolari comunicano con il SII tramite il cellulare pertanto questo servizio permette di ricevere dati rilevati sul campo in modo periodico riguardo allo stato del veicolo, alle rilevazioni di sensori, etc. Queste informazioni possono essere utilizzate per completare il profiling del mezzo, il profiling della persona, il profiling collettivo, ed in ultimo per identificare condizioni di stato ed evoluzione di stato per scatenare processi di Allerta, Notifica, Engagement, Assistenza, Raccomandazione, etc. dai vari tool descritti in precedenza.

Quando il SSM deve inviare in risposta ad un PUSH (chiamata REST verso il SSM da parte di un mobile con IP non fisso) dei dati da parte dell'APP un comando lo deve chiedere al Servizio corrispondente. Per esempio, se il PUSH arriva da:

- APP mobile: possibile invio di engagement
- APP mobile con CAR/BIKE sharing: possibile invio di prenotazione/lock/unlock
- Direzionatore: possibile invio di comando di direzione o stato di direzione
- ZTL: possibile invio di dati, comando on/OFF, etc.
- Video Decisore: possibile invio di comando per controllo di stato

I requisiti di questi componenti sono relativi a come si acquisiscono i dati dai kit veicolari dai sistemi mobili e da eventuali sensori e attuatori.

Sono pertanto **moduli consigliati minimi** per questo processo di SSM:

- MINIMO: SSM con MySQL oppure Hbase/Phoenix

WHAT NEXT: *Su questo fronte e' necessario completare lo sviluppo delle parti estese:*

- *Acquisizione dati da:*
  - *Kit veicolari*
  - *Dongle ODBII*
  - *APP specializzate: direzionatore, X sharing, ZTL, video decisore, ….*
- *Protocollo avanzato verso altri server e gestori*
  - *APP mobile con CAR/BIKE sharing: possibile invio di prenotazione/lock/unlock*
  - *Direzionatore: possibile invio di comando di direzione o stato di direzione*
  - *ZTL: possibile invio di dati, comando on/OFF, etc.*
  - *Video Decisore: possibile invio di comando per controllo di stato*

3.1.6 IOT Management

Per la connessione con sistemi IOT è necessario inserire un nuovo componente nell'area DataWarehouse. Questa soluzione ha il compito di interfacciarsi con sensori ed attuatori IOT con le loro reti e portare dati dentro il SII come portare attuazioni dal SII verso gli Attuatori. I sensori/attuatori possono essere connessi tramite uno o più IOT Broker verso il sistema SII e pertanto dentro il data storage Hbase/Phoenix. Ogni sensore/attuatore deve poter essere registrato dentro la knowledge base Km4City tramite Smart City API del Servicemap. Sempre tramite Smart City API deve essere possibile fare delle ricerche di sensori/attuatori geolocalizzati (intorno ad un punto, lungo ad un traiettoria, dentro un'area, etc.).

Sono pertanto **moduli consigliati minimi** per questo processo di IOT Management:

- MINIMO: IOT broker modificato (IOT Orion Service broker + Cygnus Modificato), ServiceMap modificato, NodeRED modificato

- Si possono aggiungere svariti

WHAT NEXT: *Al momento e' stato sviluppato un progetto di integrazione che vede un servizio centralizzato per i dati dei sensori/attuatori con LOG su Hbase/Phoenix, ed un sistema per poter colloquiare con uno o più IOT broker. Ogni IOT broker dovrebbe poter:*

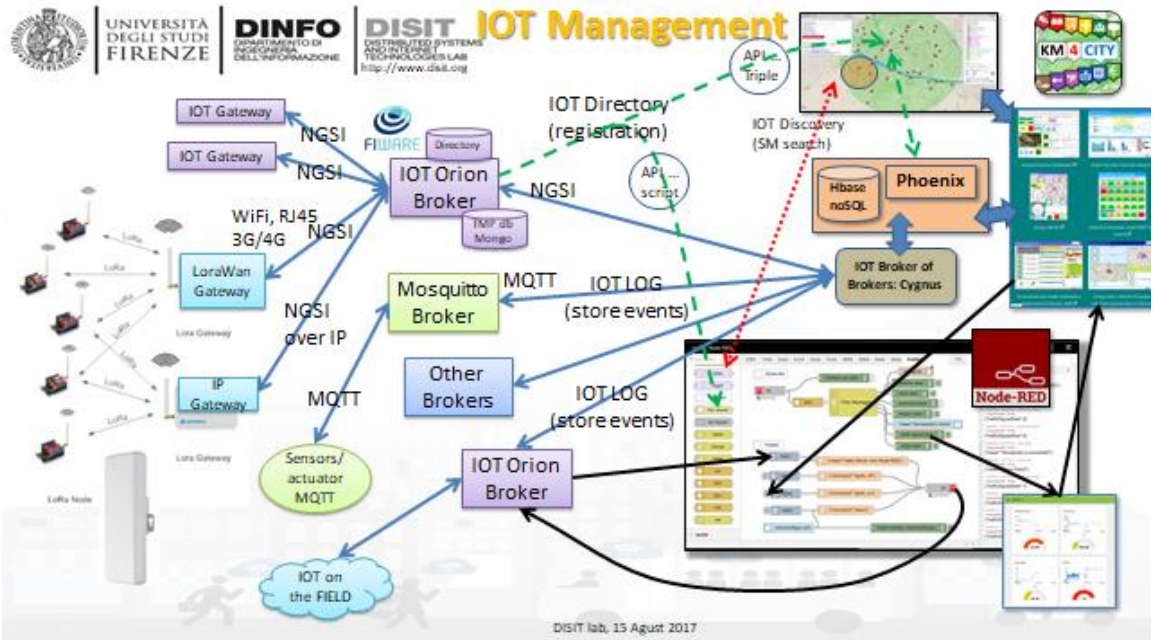
(1) registrare (connessioni a tratti verdi) i proprio IOT verso la Knowledge Base (servicemap, Smart City API, km4city: tramite opportune API esposte da ServiceMap, Smart City API; queste stesse API dovrebbero generare lo script per rendere accessibile e usabile il sensore/attuatore dentro NODERED con i suoi parametri); La registrazione di IOT device da parte di IOT broker semplici come Mosquitto deve essere definita a parte.

(2) LOGgare (connessioni con linea piena blu) i propri eventi (ingresso uscita) dentro il database noSQL Hbase/Phoenix per mezzo di un Cygnus adapter modificato che in base ad un sistema di publish subscription permette la connessione con IOT broker in NGSI; la modifica consiste nel permettere a Cygnus di salvare su Phoenix tutti i dati di SENSORI ed ATTUATORI che gli vengono registrati.

(3) connettere (connessioni con linea piena nere) processi NODERED in modo da effettuare in automatico registrazioni ad eventi e sottoscrizioni di attuazioni sul IOT corrispondente al simboli/blocchetto disponibile in NodeRed.

Inoltre:

- *Il Dashboard Manager deve presentare alcuni Widget per Attuatori in modo da poter azionare alcuni attuatori da interfaccia: selezione, scelta fine con manopola e/o cursore sliding, interruttore on/off, bottone/pulsante, inserimento testo, inserimento valori numerici, inserimento date, etc.*
- *Il Dashboard Manager deve poter avere dei widget per ospitare dash di Node-RED.*
- *I nuovi blocchetti NodeRED per sensori ed attuatori devono poter avere una connessione per fare discovery/search utilizzando i servizi di Servicemap, e smart city API in modo da avere indietro l'ID simbolico del sensore.*
- *Phoenix deve memorizzare tutti i dati in/out dei sensori/attuatori tracciando: date and time, IOT broker, IOT device ID, user requesting, process requesting, etc… lo IOT device ID permette di identificare nella KB il descrittore, la posizione, etc.*



3.1.7 Distributed Smart City Engine, DiSCES

Il DISCES è uno scheduler per la gestione di processi nel un back office big data. Il DISCES svolge la funzione di scheduler distribuito, attivando processi su varie macchine virtuali del Big Data Processing Grid, è ridondato e robusto, permette la gestione di processi periodici e sporadici, concatenati o meno. DISCES presenta un pannello di controllo e comando, e presenta delle API dalle quali si possono caricare e gestire i processi. Alcuni dati relativi all'andamento generale, ed allo stato dei processi sono accessibili via API per renderizzarli sulla Dashboard di Controllo della Smart City.
Nell'intero sistema vi sono in realtà svariate istanze del DISCES che presentano il loro database ed i loro rispettivi nodi di esecuzione. Questi DISCES sono al momento adibiti al:

- Gestione dei processi di acquisizione dati: ETL, Java, etc., qualsiasi tipo di processo eseguibile nel sistema operativo e compatibile da sistemi operativi come: Linux Based, ma anche Windows Server
- Gestione dei processi di Data Analytics: wifi analytics, engager, park prediction, wifi prediction, etc.
- Gestione dei processi di Twitter Vigilance, location nel Social Media Monitoring solution and area

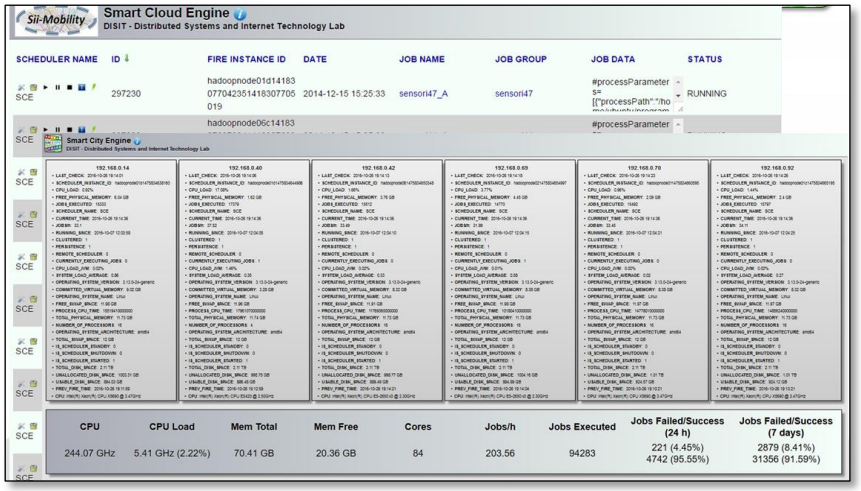


Figura: pannello di monitoraggio del sistema DISCES per la parte di Data Ingestion big data grid.

Sono pertanto **moduli consigliati minimi** per questo processo di Big Data Warehouse:

- MINIMO: **DISCES** per lo scheduling dei processi che a sua volta utilizza MySQL

WHAT NEXT: *Su questo fronte, il DISCES dovrebbe poter gestire in modo più accurato le risorse. Al momento lo scheduler adotta una gestione first come first served in base alle risorse, e al momento dei firing alloca il processo nel primo spazio libero che trova, con un'ottica di pseudo bilanciamento. Le risorse monitorate sono le VM dei nodi e non in modo estremamente fine. Una sua evoluzione dovrebbe poter:*

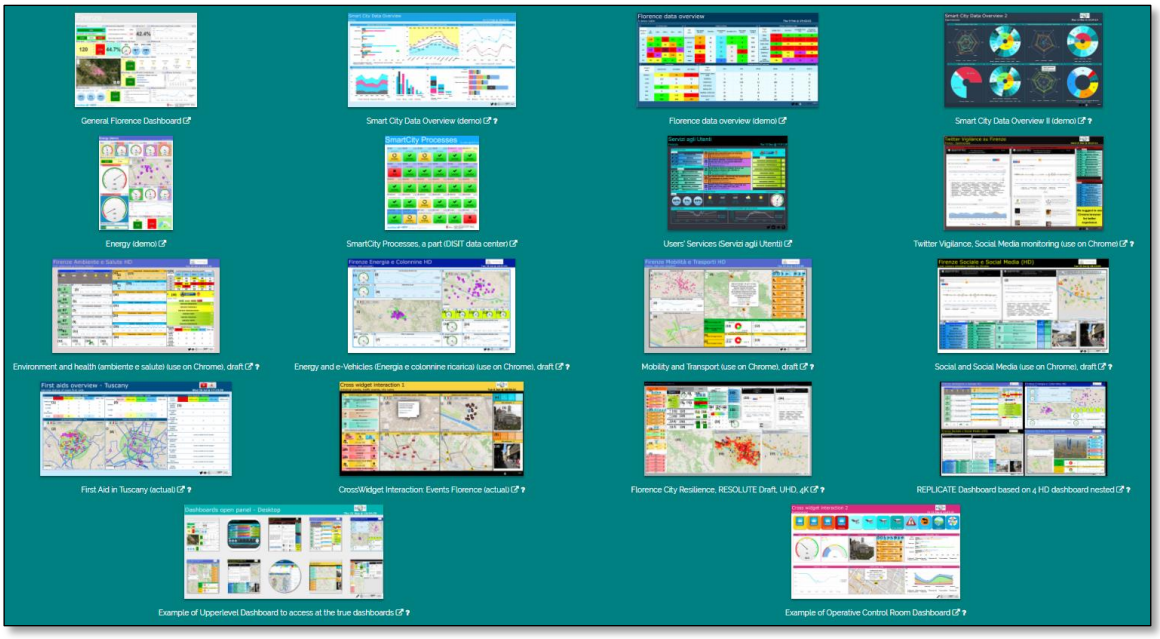
- *Monitorare in modo accurato: Host, VM e container, tramite Marathon, Mesos, e Docker*
  - *Resource monitor*
- *Accettare richieste asincrone ma anche periodiche,*
- *Schedulare le richieste ma anche fare una pianificazione di queste a fronte finito, per esempio una settimana, un mese.*
- *Decidere dove allocare i processi nelle risorse computazionali (allocator) e se necessario attivare/disattivare risorse nuove (elastic computing)*

Il DISCES dovrebbe essere conforme alla registrazione LDAP e al Notificator.

3.2 Supporto alle decisioni

3.2.1 Dashboard Engine and Dashboard Builder

Le Dashboard vengono utilizzate per la visualizzazione dei dati e degli eventi primari in termini di grafici e modelli sinottici che possono offrire una visione complessiva dello stato dell'area. Uso di soluzioni di grafica multischermo. In Sii-Mobility, vengono sviluppate viste specifiche che permettono di integrare una struttura a dashboard precedentemente sviluppata da DISIT in REPLICATE H2020. http://www.km4city.org/?controlRoom#realtimeData



Il Dashboard Manager è composto da due parti primarie:

- **Dashboard Builder and Manager (backend)**, Monitoring. Soluzione che permette la costruzione di DashBoard specifiche ad utenti come PA, PMI, operatori.
    - Questi sottosistemi permetto di accedere e collezionare dati di alto livello da vari database e API per mostrarli con sinottici consuntivi configurabili.
    - Accetta che alcuni operatori comunichino in push alcuni loro dati invece che esporre i loro dati su servizi API. Il Dashboard Manager presenta della API per poter ricevere tali dati in PUSH con chiamate REST anche autenticate.
    - **Accede a database come**: MySQL, Mongo, Phoenix/HBase, SPARQL like (Virtuoso, OWlim, GraphDb, etc.);
    - Nella definizione di una dashboard deve essere possibile identificare delle metriche e delle condizioni di allarme che possono terminare la produzione di notifiche. A questo riguardo viene utilizzato un Notificatore realizzato nel progetto REPLICATE.
- **Dashboard Control Interface (frontend)**. Modulo che permette l'accesso alle Dashboard prodotte dal Dashboard Builder e Manager.
    - la parte di front end viene letta da molti utenti se le dashboard prodotte sono accessibili anche ai cittadini.
    - Utilizza per le configurazioni MySQL

Le Dashboard possono mostrare dati puntuali, andamenti, dai su mappa, liste di eventi, etc. Possono essere costruite per la creazione di Control Room a parete oppure per pannelli di controllo applicativi per gli operatori.
In Sii-Mobility vengono sviluppate viste specifiche che permettono di integrare una struttura a dashboard sviluppata da DISIT. Un manuale utente è disponibile http://www.disit.org/6935

Il tool è accessibile da http://www.disit.org/dashboardSmartCity/management/
Si veda inoltre alcune dashboard tipo: http://www.km4city.org/?controlRoom#realtimeData

Il Dashboard Builder puo' avere delle metriche che possono determinare delle condizioni di allarme. In questo caso il Dashboard Builder utilizza il Notificator per inviare le notifiche. Mentre la rilevazione degli eventi di allarme viene effettuata a livello di Dashboard Engine.

Al momento sono stati sviluppati widget per il monitoraggio mezzi AVM, parcheggi, sensori del traffic, meteo, etc. Molti risultati del data analytics vengono resi evidenti tramite pannelli e dashboard, per esempio:

- Mappe e punti di interesse, percorsi TPL, posizione mezzi, stato paline, sensori, etc. tramite SERVICE MAP
- Traffic flow reconstruction
- Heatmap dei punti di interesse identificati tramite le App, traiettorie di maggio interesse, etc.
- Posizioni e Valori dei sensori di traffico
- Posizioni e Valori e predizioni dei sensori di parcheggio
- Posizioni e Valori dei dati di bike sharing
- Posizione e prezzi della benzina alle pompe
- Tracciati di routing
- Etc.

Sono pertanto **moduli consigliati minimi** per questo processo di Dashboarding

- MINIMO: **Dashboard manager** con MySQL
- **Notificator** per le notifiche ma non è indispensabile.

WHAT NEXT: *Per quanto riguarda questo strumento di Dashboard, puo' essere utile sviluppare:*

- *automazione della creazione di metriche relative a dati real time ed accesso a questi in modo diretto senza fare storage.*
- *widget evoluti per dati in uscita (attuatori), piker di entita', cross widget evoluto, monitoraggio stato di witi web, un heatmap generico, stato delle Smart City API, etc.*
- *sistema di gestione del dato in modo da poter gestire la crescita dei dati nello storage*
- *sistema di replay da un ceto date-Time, che implica poter avere replay via API anche da altri moduli che sono utilizzati via IFRAME: heatmap, wifi flow, traffic flow, etc.*
- *modellazione di KPI della Smart City e relativa visualizzazione con navigazione,*
- *le parti di acquisizione dati in PUSH con delle API,*

3.2.2 Notificator

Il Notificator e' un tool che permette di ricevere registrazioni di eventi da parte di altri tool, e su tali registrazioni conoscere come notificare eventuali eventi connessi ad un numero di certo utenti tramite un certo numero di canali. Le tipologie di notifiche attive sono al momento solo email, e possono essere configurate in modo molto flessibile anche con template e valori di default che permettono dal messaggio di risalire al tool, al widget, alla metrica, e all'instante di firing nonché alla condizione che lo ha provocato. Il Notificator, tiene traccia degli eventi, e ne fa un LOG accessibile per il monitoring. Lo stato e le evoluzioni del Notificator possono essere a loto volta monitorati tramite il Dashboard Manager.

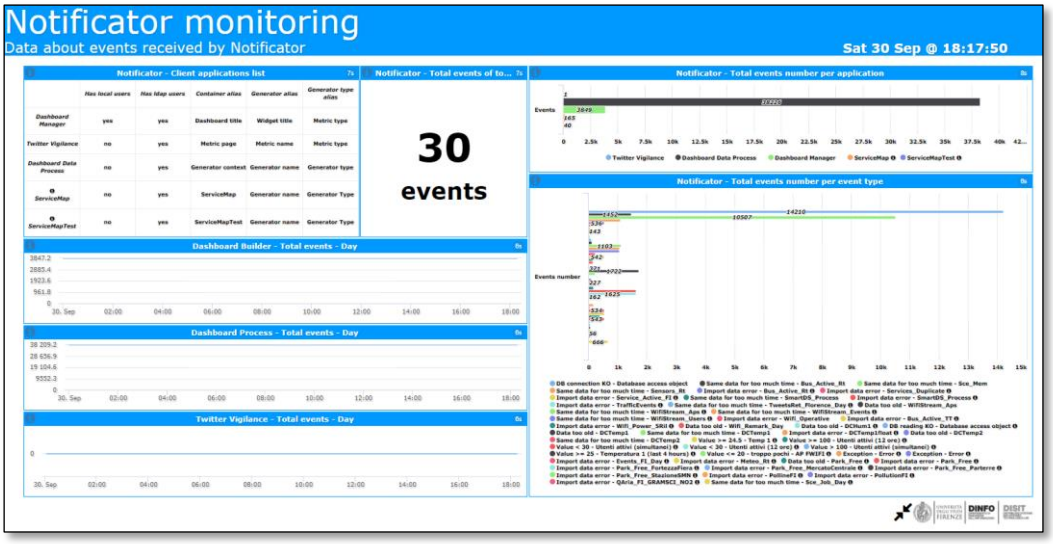
I **tool che usano il Notificator** sono al momento:

- Dashboard Manager

- Twitter Vigilance (daily) il real time deve essere connesso
- Dashboard Data process (engine)
- ServiceMap
- ServiceMap Test

Questi tool possono inviare notificare al Notificator un Allarme (la notifica dell'effettiva verifica per vero di una condizione di firing), ed e' quindi il Notificator che invia la notifica ad uno o più utenti registrati in locale o con LDAP. La definizione delle condizioni di Firing e la loro valutazione non è compito del Notificator. Tali condizioni possono essere definite solo da chi conosce i dati inerenti alle varie applicazioni, e sono sempre questi che possono conoscere come computare tali condizioni. Se cosi non fosse tutti i dati dovrebbero passare dentro il Notificator che dovrebbe conoscere troppo di molte applicazioni.

Notificator Monitoring:
http://www.disit.org/dashboardSmartCity/view/index.php?iddasboard=MTQ4



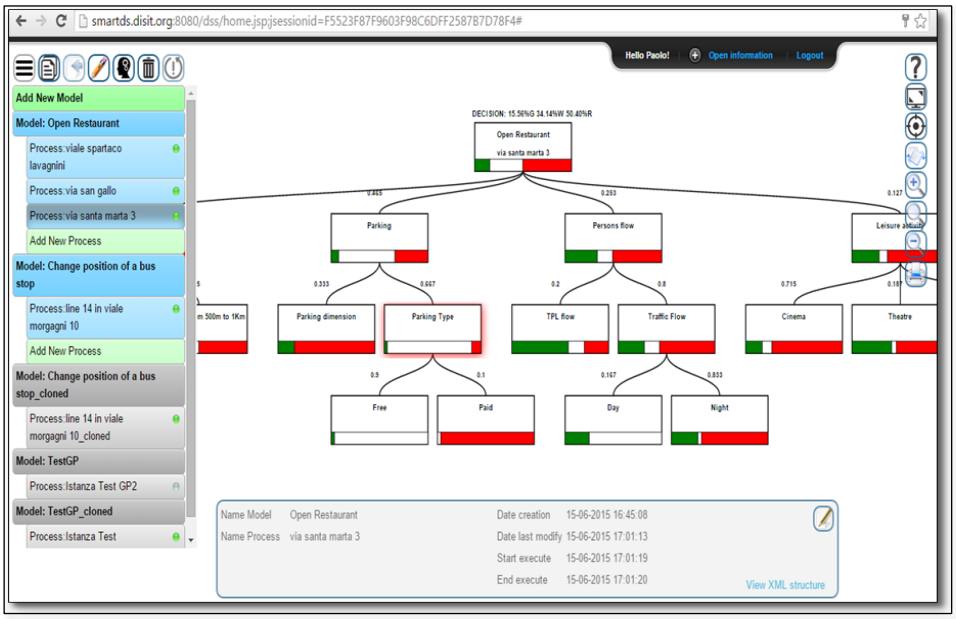
Sono pertanto **moduli consigliati minimi** per questo processo di Notification

- MINIMO: **Notificator** con MySQL

WHAT NEXT: *Per il Notificator i prossimi passi possono essere:*
- *Renderlo in grado di inviare notifiche effettuando chiamate REST in modo da interfacciarsi con sistemi di Ticketing, con sistemi di SMS, etc.*

3.2.3 Smart Decision Support, SmartDS

Lo SmartDS e' una soluzione per lo sviluppo di modelli di supporto alle decisioni per esempio per lo spostamento di fermate, etc. Questa soluzione e strumento è accessibile su http://smartds.disit.org, entrare come guest. Lo Smart Decision Support, SMART DS:
- Permette di formalizzazione di criteri di allarme e di alert che possono identificare condizioni critiche, correlazioni non attese, etc.;
- adotta un modello System Thinking per il supporto alle decisioni e accede ai dati della knowledge base (KB con le informazioni della città) come di altri database, per esempio quelli della dashboard, del sistema di social media, etc.;
- comunica con altri elementi dell'architettura tramite chiamate alle loro API o accedendo a data base MySQL o SPARQL, deve poter leggere anche dati da Phoenix;
- mette a disposizione delle API, per effettuare chiamate che vanno ad attualizzare i valori dei processi di supporto alle decisioni.



Sono pertanto **moduli consigliati minimi** per questo processo di SmartDS
- MINIMO: SmartDS con MySQL

WHAT NEXT: *Per lo SmartDS:*
- deve poter leggere anche dati da Phoenix;
- avere una maggior integrazione con Dashboard esponendo delle API per la verifica delle decisioni
- avere una maggior integrazione con ResilientDS esponendo delle API per la verifica delle decisioni (in connessione con RESOLUTE H2020)

3.3 Social Monitoring

3.3.1 Participation Platform

La Piattaforma di Partecipazione e Sensibilizzazione viene utilizzata per coinvolgere il cittadino a partecipare ma anche per informare e formare il cittadino, tramite totem, applicazioni mobili, web application, etc. dovranno essere sviluppati elementi singoli, widget, o pagine web da mettere in frame, per poter ricevere valutazioni da parte di utenti anche tramite touch. Inoltre i vari pannelli della dashboard dovrebbero poter interagire fra si loro.
La Piattaforma di Partecipazione e Sensibilizzazione espone pagine web per acquisire informazioni dai cittadini e per informarli tramite web e/o totem. Accede alle Smart City API per i dati. Le pagine della piattaforma possono essere prodotte con strumenti tradizionali ed avere degli inserimenti di Widget derivati da: ServiceMap, Traffic Flow reconstruction, ServiceMap 3D, Dashboard vari, etc. Si suggerisce lo sviluppo tramite HTML5 e l'uso di Widget del Dashboard Builder che già integra svariate funzioni base.

WHAT NEXT: *Questo tool non è ancora sviluppato ma si suppone si possa sviluppare basandosi ed espandendo elementi della (i) dashboard (Dashboard Builder Open source), (ii) Servicemap, (iii) ServiceMap 3D, (iv) dashboard builder, (v) Web App Cordova, (vi) altri tool creando dei pannelli di evidenza e partecipazione per il cittadino. Le applicazioni HTML5 sviluppate come piattaforma di partecipazione potranno essere utilizzate su:*

- *Pannelli di controllo nella Control Room*
- *Pannelli nei TOTEM posizionati in stazioni, paline, etc.*
- *Pagine WEB ad accesso pubblico*

3.3.2 Social Media Crawler and Manager

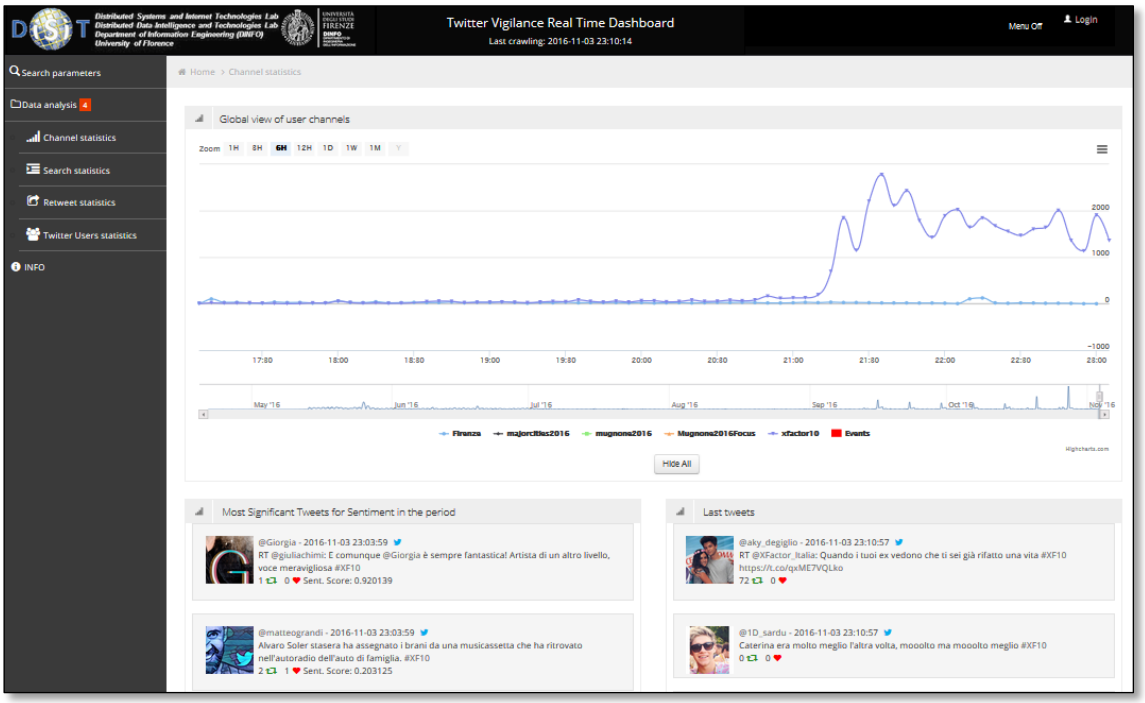
Il modulo di Social Media Crawler and Manager permette di collezionare e trarre vantaggio delle informazioni di dettaglio e commenti dai blog sul territorio come da social network e di comprenderle per trarre deduzioni e feedback nel sistema centrale di conoscenza. In Sii-Mobility, sono sviluppate estensioni rispetto al tool Twitter Vigilance già presente in DISIT lab prima dell'inizio del progetto Sii-Mobility. In particolare, in Sii-Mobility viene reso più usabile, e viene effettivamente utilizzato per monitore i servizi di mobilità e smart city in generale con cadenza giornaliera. Il tool Twitter Vigilance è composto da:

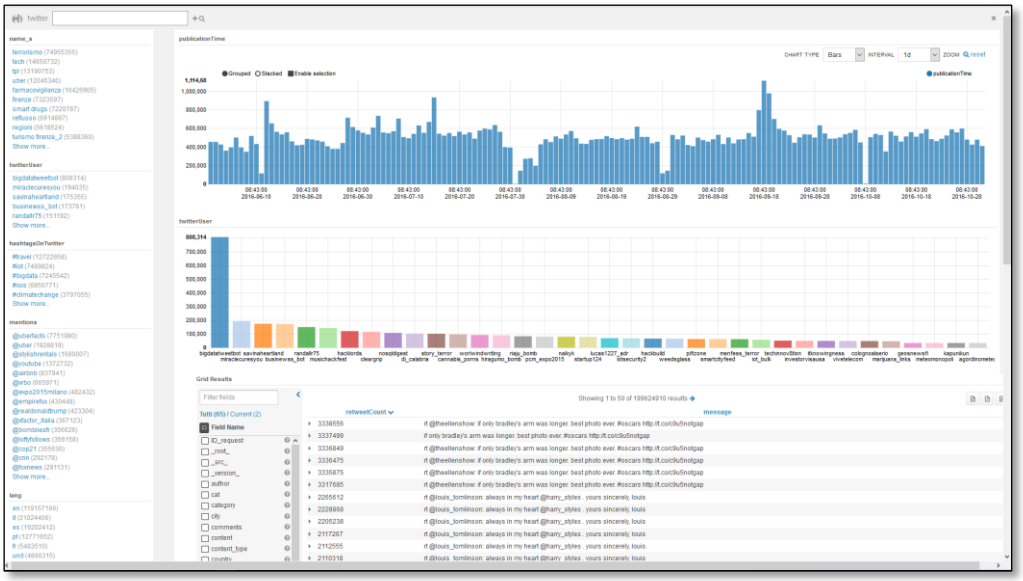
- **Social Media Crawler and Manager (backend)**: sistema di crawling massage da Tweet principalmente.
    - **Il Social Media Crawler and Manager** espone API rest per l'accesso ai dati come l'accesso al database per avere dati di sintesi e/o di dettaglio relativi ai Tweet, retweet, etc. In questo modo, i dati di sintesi possono essere accessibili sulle Dashboard delle control room.
- **Social Media Analysis interface (frontend), pe**rmette di rendere disponibili per la soluzione Sii-mobility delle informazioni di contesto importanti riguardo all'uso dei servizi, all'apprezzamento di questi, ad eventuali eventi burst, etc.

Sono iniziate le sperimentazioni per la valutazione dei servizi di TPL e UBER tramite canali di ascolto, si veda Deliverable specifici. Le valutazioni statistiche sono in corso. La parte funzionale è stata validata. Si veda http://www.disit.org/tv e http://www.disit.org/rttv (versione Real Time sviluppato in RESOLUTE project) Si veda deliverable: DE4.6a, e DE4.7a.

Lo strumento TV permette di attivare canali di ascolto su Twitter e di attivare varie metriche di monitoraggio e segnali allarme ed early warning su tali metriche. Queste metriche sono tipicamente

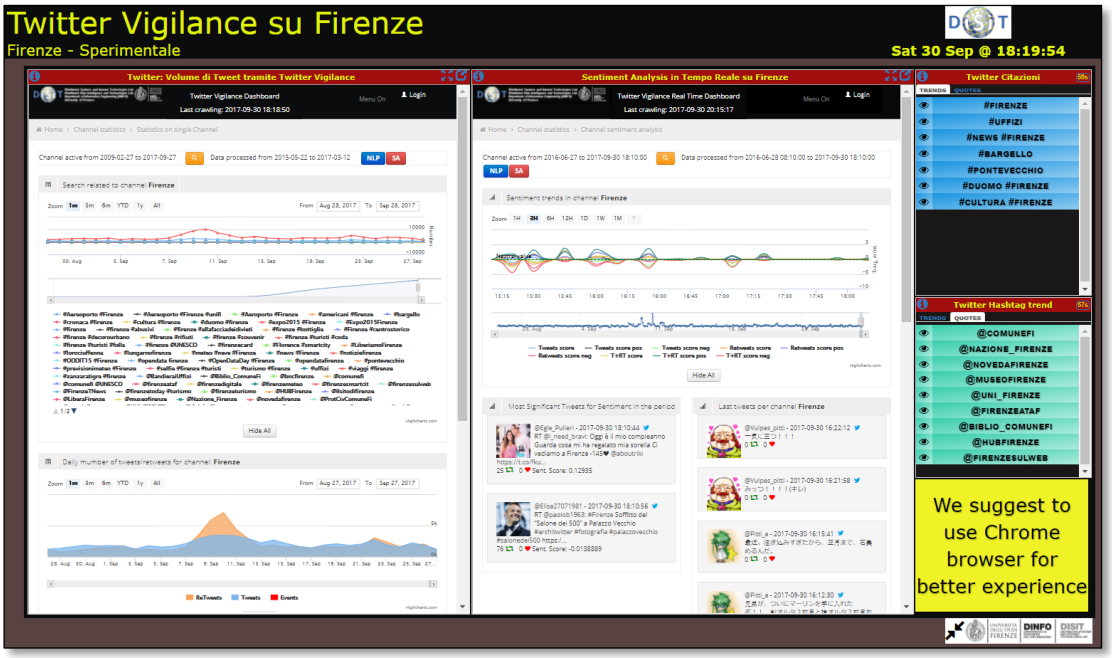
di volume, unicità utenti, relazione fra utenti, NLP, sentiment analysis, etc. Anche questi dati possono essere integrati in Dashboard con Widget che accedono a metriche specifiche oppure come IFRAME.





Il Twitter Vigilance puo' avere delle metriche che possono essere usate con strumenti per segnalare condizioni di allarme. In questo caso il Twitter Vigilance utilizza il Notificator per inviare le notifiche. Mentre la rilevazione degli eventi di allarme viene effettuata a livello di Dashboard Engine.

Twitter Vigilance Dashboard:
http://www.disit.org/dashboardSmartCity/view/index.php?iddasboard=Njc=



Sono pertanto **moduli consigliati minimi** per questo processo di Social Monitoring

- MINIMO: TwitterVigilance con MySQL
- La versione con Hadoop/HDFS e' in corso di sviluppo

WHAT NEXT: *Questo strumento di Twitter Vigilance puo' essere ulteriormente sviluppato permettendo di gestire questi dati su un'architettura parallela. In particolare si prevede di portare in modo progressivo alcuni dei processi verso uno storage HDFS e per poter permettere il computo di metriche progressive tramite Hadoop.*

3.4 User Management, Engagement, Stimulation

Questa area di sviluppo comprende un elevato numero di strumenti che dovranno subire alcune trasformazioni per poter gestire in modo centralizzato ed efficiente i dati relativi alle singole persone. Al momento i dati forniti dagli utenti sono distribuiti su più database e gestiti da più tool descritti in seguito.

3.4.1 User Crowd Sourcing Manager

Il tool **User Crowd Sourcing Manager** è (i) un canale di ascolto dei cittadini per le loro problematiche e notifiche di problemi riscontrati su servizi POI specifici (per esempio una palina che non va bene, un ristorante chiuso, un disservizio su una TPL, un cartello che non si legge, etc.), (ii) un canale per collezionare nuove informazioni su servizi attivi e anche nuovi. Questo e' composto da due parti:

- **User Crowd Sourcing Manager backoffice** che lavora in connessione alla piattaforma di partecipazione e sensibilizzazione rappresenta il loro database. Questo sottosistema permette di collezionare commenti, voti e contributi (immagini per esempio) dai singoli cittadini.
- **User Crowd Sourcing Manager frontend** espone le Smart City API per il caricamento di queste informazioni tramite le Smart City API. Vi sono inoltre delle pagine web di gestione in cui è possibile vedere la lista dei contributi inviati e procedere alla loro moderazione, cioè identificare quelli che possono essere ripubblicati o rigettati per la pubblicazione.

Il modulo per il collezionamento di voti, commenti, ed immagini è stato sviluppato nella prima versione e le API corrispondenti sono attive ed usate dalle APP mobili, web e Windows 10. Il tool di gestione di queste informazioni a livello di back office non è ancora molto funzionale ed usabile ma per il momento è sufficiente a poter gestire la moderazione dei contributi che arrivano dagli utenti. Solo quelli che vengono approvati dall'amministratore vengono caricati nella Knowledge base e pertanto sono resi visibili sul Servicemap, come dalle App. L'interfaccia non è accessibile dall'esterno ma solo come funzione interna di backoffice.

2016-12-11 16:49:52.0	Ponte Vecchio	Ponte vecchio	rejected
2016-12-02 10:04:56.0	Farmacia COMUNALE N. 13	E proprio una farmacia!	submitted
2016-11-30 20:16:18.0	EL BADAOUI ABDELFATTAH	Non esiste!	submitted
2016-11-28 17:54:17.0	SOTTO L'OMBRELLO DI MARY POPPINS SOCIETA' COOPERATIVA SOCIALE	Asilo nido 0-3 anni orario 7.30- Tel 055442653	validated

2017-01-05 11:06:09.0	Giardino dell Orticoltura		validated
2017-01-05 11:05:54.0	Giardino dell Orticoltura		validated
2016-12-31 01:22:42.0	PIAGGETTA		validated

Sono pertanto **moduli consigliati minimi** per questo processo di User Crowd Sourcing

- MINIMO: User Crowd Sourcing con MySQL

WHAT NEXT: *Questo strumento dovrebbe evolvere verso:*

- *una gestione armonizzata con i dati degli utenti. I contributi sono gestiti in modo collettivo ed anonimo ma potrebbero essere associati al profilo utente come suoi contributi personali.*
- *Compatibilità con i criteri e metodi GDPR*

3.4.2 User Engagement on Demand

Lo User Engagement viene utilizzato per informare e coinvolgere il city user assegnandogli compiti specifici o solo per informarlo. Per esempio:

- Hai visto questo POI potrebbe interessarti (raccomandazione)
- Hai visto che sei stato in questo POI, vuoi fare una recensione, caricare una foto, esprimere un voto, lasciare un commento, etc.
- Alle 14:50 eri in mobilità, con che mezzo: auto, bici, a piedi, bus, treno, etc.
- Ti è piaciuta Firenze?, puoi rispondere ad alcune domande ?
- Sei in coda a XXX, quante persone vedi?
- Hai parcheggiato in un'area a pagamento fuori dalla tua area di residenza.
- Ho visto che stai usando molto il mezzo privato, perché non provi ad usare il TPL… questo percorso sembra fatto per te ...
- non parcheggiare qui perché non è consentito

- parcheggia allo scambiatore che spenderesti meno
- usa i mezzi pubblici che spendi meno energia per lo stesso tratto
- ho visto che passi da …… ti suggerisco di fare una strada diversa
- Etc.

Questo approccio coinvolge i city users a fornire delle opinioni, partecipare e fornire informazioni tramite il loro mobile. Le informazioni sono ovviamente sul sistema di mobilità, sui servizi di mobilità ma anche sui servizi integrati di smart city e sul comportamento stesso dell'utente. Allo stesso tempo, e sullo stesso canale e mobile, il cittadino può ricevere informazioni utili.
Se il city user raccoglie il suggerimento inviato il sistema puo' verificare. Questa verifica viene fatta analizzando le tracce del comportamento dell'utente stesso tramite un consenso informato.
I componenti principali dell'User Engagment sono:

- **User Engager backoffice** presenta un back office per il calcolo delle condizioni e delle regole ed un front end per il deploy dei messaggi di engagement o assistenza agli utenti di mobile e totem.
- **User Engager frontend:** si interfaccia con il backoffice accedendo al database in modo diretto.

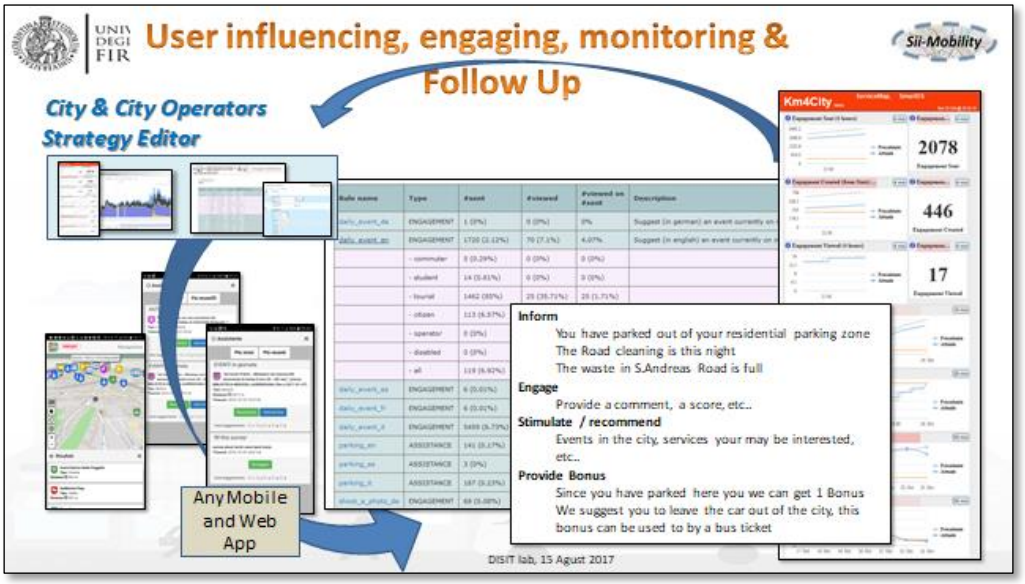


Figura: pannello di follow up delle regole di engagement, presenta per ogni tipo di regola e strategia il numero di emissioni, la percentuale di letture (esposizioni), e la percentuali di accettazioni.

L'**User Engager backoffice permette l**'inserimento delle strategie di engagement: informazioni, stimoli, richieste, sondaggi, richieste di verifica, etc. Queste regole vengono eseguite nel back office e per il loro computo hanno bisogno di conoscere il comportamento collettivo e puntuale (anche in tempo reale talvolta) di ogni singolo utente. A questo fine il modulo di engagement procede al calcolo delle condizioni e delle regole, e presenta un modulo di front end per il deploy dei messaggi di engagement o assistenza agli utenti di mobile e totem. L'interfaccia di definizione delle regole, come il tool di monitoraggio dei risultati (follow up) non sono accessibili dall'esterno ma solo dal backoffice.
In questo momento gli engagement vengono veicolati alle App al momento in cui la App chiama il Sensor Server manager per comunicare la sua posizione ed evoluzione.

Sono pertanto **moduli consigliati minimi** per questo processo di User Engagement

- MINIMO: User Engagement con MySQL, DROL

- MINIMO: Algoritmi di valutazione di metriche di base: velocita' media, velocita' storica, km e metri fatti, POI, traiettorie, etc. Tutti questi sono tipicamente messi in esecuzione su DISCES,
    - oppure su MapReduce in Hadoop in corso di sviluppo.
- MINIMO: Connessione con Sensor Server Manager
- MINIMO: Connessione con OAuth per social media mutuated registration
- MINIMO: Connessione con ServiceMap per la stima di alcune condizioni di contesto sul comportamento in città
- Algoritmi di machine learning per la valutazione del comportamento: mezzo con il quale l'utente si muove, etc.

WHAT NEXT: *In questo ambito*

- *Vanno sviluppati algoritmi su MapReduce/Spark per la stima di: velocita' medie, POI, traiettorie tipo, etc. etc.*
- *Vanno sviluppate funzioni specifiche che permettano di definire regole di engagement più mirate a comportamenti dei cittadini in mobilità.*
    - *Le regole di engagment vengono formalizzate in DROL, ma è in corso di sviluppo una interfaccia più semplice e coerente con la logica delle strategie che potrebbero essere definite dagli operatori come dalle pubbliche amministrazioni.*
- *Vanno attivati dei meccanismi di engagement con reward, che puo' essere fatto tramite:*
    - *sconti, o semplicemente ticket, etc. Questi possono essere prodotti come QR one shot e pertanto utilizzati solo una volta dall'utente che lo ha ricevuto*
    - *stelline o bollini che permettono di acquisire punti che poi possono essere spesi nel circuito oppure anche solo verso uno specifico operatore che gli ha rilasciati. Per esempio bollini/punti per lo sconto sui biglietti del TPL.*

3.4.3 Suggestion on Demand, Recommender

Il Recommender accede ai profili degli utenti sulle App e produce raccomandazioni sulla base del profilo attuale e cumulato nel tempo con modello ad apprendimento. Le raccomandazioni sono geolocalizzate e contestualizzate. Fra queste raccomandazioni anche le allerte della protezione civile, i tweet, il meteo, etc. Il backend del Recommender fornisce evidenze statistiche sul numero di raccomandazioni prodotte per categoria di utenti, quante raccomandazioni sono state viste (esposizioni) e approfondite (click), etc. Permette pertanto di completare il profilo di utente collettivo per categoria, ma anche di comprendere cosa apprezza l'utente, cosa cerca, dove si muove, come si muove, quali sono i posti che visita più frequentemente, etc.
Pertanto il recommender con il supporto dei dati provenienti dalle APP via il Sensor Server produce anche matrici di Origine Destinazione, heatmap, traiettorie, etc. che sono direttamente accessibili per gli amministratori. Il Recommender utilizza l'User Profiler.
Il front end del recommender presenta le API corrispondenti che fanno ovviamente parte delle Smart City API. Lo stesso strumento fornisce un'interfaccia per fare in modo che le APP mobili possano richiedere suggerimenti geolocalizzati.

- **Recommender backoffice:** i dati rilevati da mobile e altre sensori potranno servire per misurare i flussi di persone, flussi di mezzi, tracce e traiettorie, OD matrix, heatmap, etc.
- **Suggestion on Demand (frontend)** È in grado di produrre suggerimenti di vicinanza su: servizi utilizzando le Smart City API verso la Knowledge Base, tenendo conto anche del profilo del city user (cittadino, pendolare, turista, studente) e del comportamento specifico della persona. Fornisce anche informazioni di guida connessa, o per quanto riguarda ai bonus che possono derivare dal comportamento virtuoso dei cittadini.

Sono pertanto **moduli consigliati minimi** per questo Modulo di Recommendation

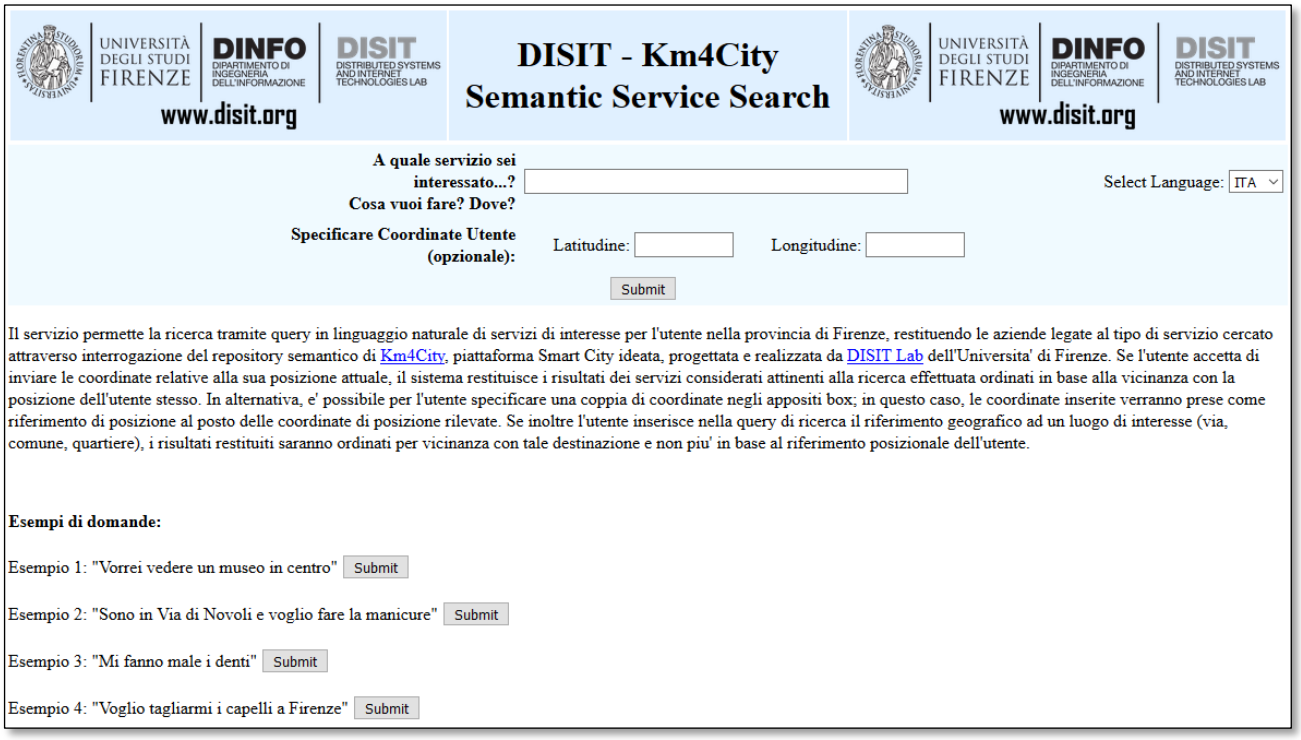
- MINIMO: User Recommender con MySQL
- MINIMO: Sensors Server Manager su MYSQL e/o Phoenix/HBase
- MINIMO: DISCES ed Algoritmi di analisi in Java

WHAT NEXT: *In questo ambito sono necessarie ottimizzazioni aggiuntive per:*

- *Riorganizzazione delle informazioni, il recommender deve cedere alcune informazioni di user profling per passarle all'User profiler.*
- *Il recommender deve poter accedere alle info dell'User profiler come (profilo storia delle raccomandazioni, query, facebook data, etc…) in forma anonima e via delle API.*

3.4.4 User Assistant, PAVAL, *Conversational Recommender System*

PAVAL User Assistant è uno strumento che a fronte di una domanda espressa in linguaggio naturale in varie lingue risponde con una lista di POI probabili che possono dare una risposta. http://paval.disit.org/Paval/



Sono pertanto **moduli consigliati minimi** per questo Modulo di Assistant PAVAL

- MINIMO: PAVAL con MySQL, GATE, NLP
- MINIMO: Accesso a ServiceMap per la conoscenza del territorio, via API
- MINIMO: Accesso ad alcune basi di dati locali

WHAT NEXT: *In questo ambito sono necessarie ottimizzazioni aggiuntive per:*

- *Rendere PAVAL capace di interloquire, rispondere quando non capisce in modo da stimolare nuove richieste più specifiche e comprensibili per lui stesso*
- *Rendere PAVAL capace di rispondere non solo alla collezione di strutture di frasi attuali (principalmente diretta ad identificare POI) ma anche capaci di*
    - *fornire indicazioni:*
        - *La strada per…*

- *Come arrivo a…….*
- *A che ora passa il prossimo tram*
- *Cosa mi consigli per arrivare a Capalle*

- *Rispondere in modo scherzoso in caso di offese*
- *Rispondere in Fiorentino stretto se richiesto, o se la domanda è in Fiorentino stretto*
  - *Icche mi consigli di fare…*
- *Comprendere il Fiorentino Stretto*
- *Comprendere anche altre lingue (processo già intrapreso)*
- *Indirizzare l'interlocutore verso i servizi in città:*
  - *Musei, ristoranti, attrazioni, poi delle digital location*
  - *Eventi in città*
  - *Acquisto di prodotti da una lista con descrizioni, etc.*
  - *Informazioni di mobilità*

3.4.5 User Profiler

Lo User Profiler colleziona le richieste che provengono dalle App Mobile e pertanto conosce il comportamento degli utenti, le loro preferenze, i punti in cui sostano, le velocita', le ricerche cha fanno, etc. Lo **User Profiler** colleziona le preferenze degli utenti sui suggerimenti, sulle ricerche che effettua, etc., tutto tramite le App.

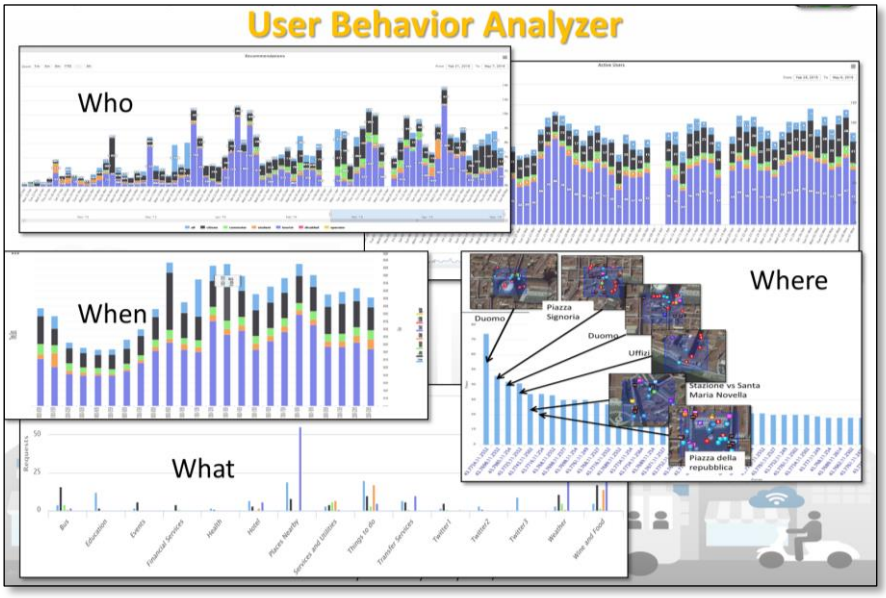


Figura: Alcuni dati statistici e diagrammi che sono accessibili tramite User Profiler

Lo User Profiler colleziona le preferenze sui suggerimenti, sulle ricerche che l'utente effettua sulle App. Questo profilo viene arricchito con informazioni sui tipi di POI degli utenti, sulle loro traiettorie tipo, sui loro tipici mezzi di trasporto, etc. (tramite Sensor Server Manager). Questi dati sono integrati tramite l'uso di algoritmi specifici che vanno a valutare i log e le tracce dei comportamenti utenti fornite dalle APP per produrre delle deduzioni con utilizzando soluzioni di data analytics tramite modelli e strumenti statistici e/o di machine learning. A questo riguardo si veda il ruolo del Sensor Server Manager che permette di collezionare dati che provengono dalle App mobility, ma anche dai Kit Veicolari, dai Dongle OBDII connessi via Bluetooth, etc.

Nella piattaforma Sii-Mobility Km4City vi sono svariate tipologie di informazioni che solo relative agli utenti e sono principalmente acquisite tramite il cellulare e la App:

- Profilo utente con credenziali: user name e email, oppure tramite OAuth come: facebook, G+, in questo ultimo caso una serie di informazioni possono essere ricevute ed utilizzate.
- Tracciamento dei movimenti delle persone via il loro cellulare (via SSM)
- Servizi richiesti preferiti sull'App (Recommender)
- Punti di maggior interesse: casa, lavoro, etc. e le loro coordinate GPS (Recommender, Engare, SSM)
- Traiettorie tipiche di movimento in città e anche oltre (Recommender, SSM)
- Contenuti caricati: immagini, commenti, rank (crawd service)
- LOG: dei menu cliccati, delle funzioni usate sul telefono, etc. (ServiceMap)
- LOG delle ricerche fatte e delle richieste effettuate dai dispositivi (service Map)
- Numero ed ID di dispositivi (SSM, Recommender)
- Tipo di guida (SSM, ODBII)
- Mezzi pubblici e privati utilizzato per i movimenti
- Volume di movimenti a piedi o con i mezzi (SSM, Engager)
- Etc.

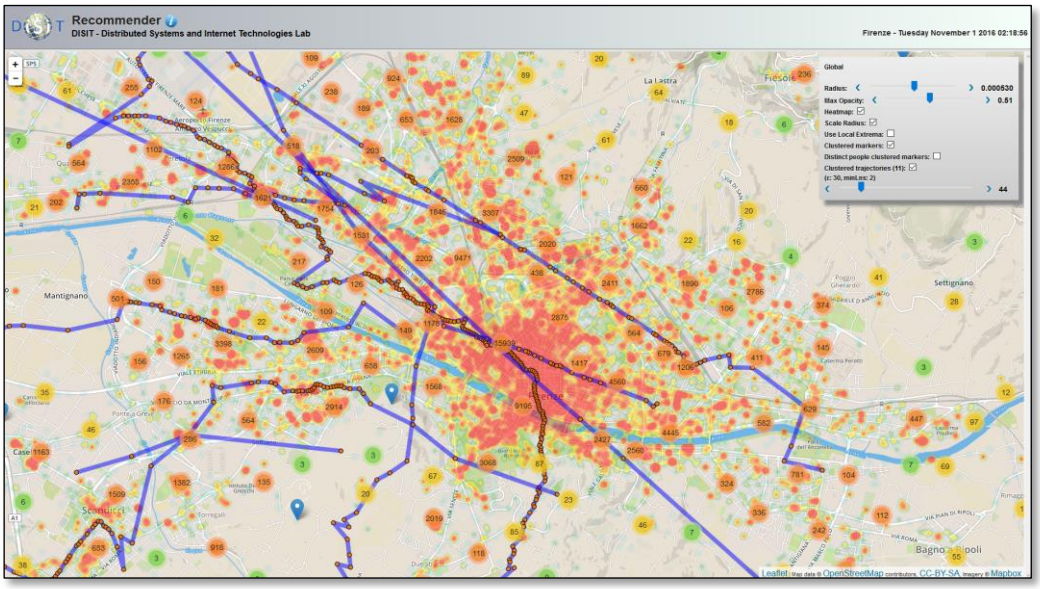


Figura: estrazione di traiettorie, heatmap e cluster sulla base dei movimenti rilevati dalle App

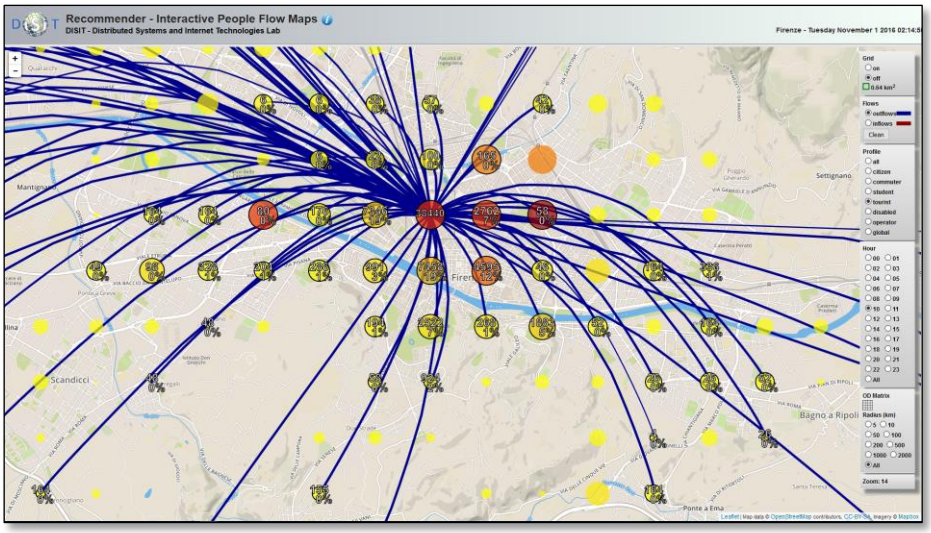


Figura: estrazione di mappe di origine destinazione in forma di SPIDER sulla base dei movimenti rilevati dalle App. Sono accessibili anche mappe OD tradizionali a matrice

Come risultato dei process di data analytics è possibile ottenere mappe che descrivono i movimenti di mezzi e persone tramite i sensori e le App. Le App di Sii-Mobility comunicano con il Sensor server and Manager lasciando i propri dati. Questi log di dati e posizioni sono utilizzate per tracciare in tempo reale gli spostamenti, derivare matrici di OD, e tenerne conto per gli algoritmi di raccomandazione e di engagement.

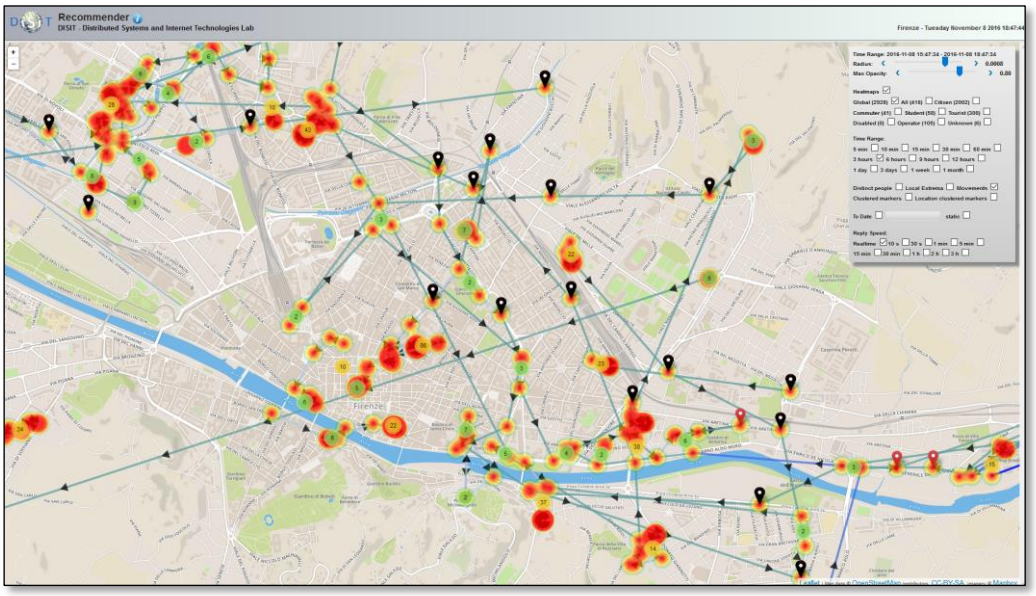


Figura: esempio di traiettorie in tempo reale sulla base dei dati dalle APP

Sono pertanto **moduli consigliati minimi** per questo Modulo di User Profiler

- MINIMO: User Profiler con MySQL
- MINIMO: Sensors Server Manager su MYSQL e/o Phoenix/HBase
- MINIMO: DISCES ed Algoritmi di analisi in Java

WHAT NEXT: *In questo ambito sono necessarie ottimizzazioni aggiuntive per:*

- *Concentrare sotto un'unica gestione dati di tipo diverso come elencati in precedenza*
- *essere compatibile in prospettiva con il GDPR che richiede che l'utente possa voler alienare i propri dati di account e lo storico in ogni momento, e che possa decidere quali tipologie di informazioni rimuovere e/o non concedere fra queste aree:*
    - *Profilo utente con credenziali: user name e email, oppure tramite OAuth come: facebook, G+, in questo ultimo caso una serie di informazioni possono essere ricevute ed utilizzate.*
        - *Numero ed ID di dispositivi (SSM, Recommender)*
    - *Servizi richiesti preferiti sull'App (Recommender)*
    - *Engagement:*
        - *Tracciamento dei movimenti delle persone via il loro cellulare (via SSM)*
        - *Punti di maggior interesse: casa, lavoro, etc. e le loro coordinate GPS (Recommender, Engage, SSM)*
        - *Traiettorie tipiche di movimento in città e anche oltre (Recommender, SSM)*
        - *Tipo di guida (SSM, ODBII)*
        - *Mezzi pubblici e privati utilizzato per i movimenti*
        - *Volume di movimenti a piedi o con i mezzi (SSM, Engager)*

  - *Contenuti caricati:*
    - *immagini, commenti, rank (crowd service)*
  - *LOG:*
    - *dei menu cliccati, delle funzioni usate sul telefono, etc. (ServiceMap)*
    - *delle ricerche fatte e delle richieste effettuate dai dispositivi (service Map)*
- *Sviluppo di algoritmi per il calcolo di informazioni di livello più elevato come POI, traiettorie, velocita', evoluzione dello stato che al momento sono demandate all'User Engager.*
- *Acquisizione e gestione dei dati da Kit Veicolari e da Dongle ODB2*
- *Comunicazione verso l'utente dei suoi dati come:*
  - *Km fatti a piedi in giornata e settimana*
  - *Km fatti con mezzi privati in giornata ed a piedi*
  - *Km fatti con mezzi pubblici in giornata ed a piedi*
  - *PPOI, Personal Punti di interesse: home, work, school, etc.*
  - *Ricerche fatte*
  - *Tipologie di POI più ricercati*
  - *Mezzi pubblici più utilizzati*
  - *Traiettorie tipiche*
  - *Contenuti caricati e contributi fatti*
  - *Preferiti sui POI*
  - *Comportamento alla guida tramite elaborazione dati da ODBII*
  - *Etc.*

3.5 Tools Support and System Management

3.5.1 Support for Authentication and Services Access, LDAP

Il Sistema di accesso autenticato si compone di:

- Una soluzione LDAP per l'autenticazione e il controllo accesso alle applicazioni di sviluppo e ai pannelli di controllo. Devono sottostare ad LDAP i seguenti tool:
  - DataGate (M)
  - Algorithm/Process Loader (M)
  - DISCES: distributed scheduler (M)
  - Dashboard Builder (ok)
  - Twitter Vigilance e TV RT (M)
  - Notificator (ok)
  - Accesso allo Storage Phoenix via: Zeppelin (M)
  - PAVAL (M)
- Una VM per la certificazione ed il supporto per HTTPS. Fanno uso di questo servizio fornendo pagine HTML con protocollo HTTPS:
  - DataGate (H)
  - Algorithm, and Process Loader (H)
  - DISCES (H)
  - Dashboard Builder (H)
  - ServiceMap (H)
  - Twitter Vigilance e TV RT (H)
  - Accesso allo Storage Phoenix via: Zeppelin (non ha senso)
  - Notificator (H)
  - PAVAL (ok)

Le applicazioni verso gli utenti finali prevedono la gestione via User Profiler ed il protocollo Open Authentication, OAuth.
Si veda inoltre il sistema di licensing per l'accesso condizionato ai dati. Gli utenti finali che accedono alle APP possono essere anche utenti registrati LDAP per esempio nel caso di utenti certificati come Polizia, Protezione Civile, etc. dovrebbe essere possibile fornire a questi gruppi di utenti un accesso controllato con maggior precisione e sicurezza. A questo riguardo si veda in seguito la sessione di accesso al RDF storage tramite licensing.

Sono pertanto **moduli consigliati minimi** per questo Modulo di Autenticazione
- MINIMO: VM servizio LDAP
- MINIMO: VM servizio OAuth, Open Authentication
- MINIMO: VM servizio HTTPS certification

WHAT NEXT: *In questo ambito:*
  - *i processi decritti in precedenza che presentano un (M) devono ancora essere portati a lavorare con LDAP.*
  - *i processi decritti in precedenza che presentano un (H) devono ancora essere portati a lavorare con https.*

3.5.2 Monitoring Service Interface: monitoring smart city solution

Il Monitoring Service Interface è il sistema di monitoraggio della smart city Sii-Mobility. Tale Sottosistema controlla l'esecuzione ed il corretto funzionamento della Smart City → tramite metriche che vengono monitorate su Dashboard:
- **degli host** con le loro risorse: CPU, Mem, storage, rete
- **delle macchine virtuali** con le loro risorse: CPU, Mem, storage, rete
- **dei processi di acquisizione dati** messi in esecuzione e schedulati tramite DISCES che fornisce alcuni parametri del tipo: stato del processo, numero di fallimenti nel giorno, numero di fallimenti alla settimana, nuova date and time di firing, numero di processi, numero di esecuzioni, etc.
- **dei processi di data analytic** messi in esecuzione e schedulati tramite DISCES che fornisce alcuni parametri del tipo: stato del processo, numero di fallimenti nel giorno, numero di fallimenti alla settimana, nuova date and time di firing, numero di processi, numero di esecuzioni, etc.
- **dei front end WEB** per esempio per controllare se i servizi sono attivi:
  - ServiceMap
  - ServiceMap3D
  - Dashboard Builder, e sue dashboard stesse
  - Traffic Flow Reconstruction
  - Web pages di Km4City.org, e di Sii-Mobility.org
  - Sensor Server Manager API, 1 MySQL e 2 Phoenix
  - Smart City API
  - Twitter Vigilance
  - Twitter Vigilance RT
  - Solr Twitter Vigilance
  - Engager Management
  - Crowd Sourcing services
  - Etc.
- **Delle Smart City API dal ServiceMap:**
  - Numero di chiamate, distribuzione dei TOP IP, numero di banned, etc

Questo tool, riporta alcune informazioni direttamente sulla Dashboard tecnica della smart city, per esempio il numero di colpi di clock usati al secondo, lo stato della memoria, il numeri di processi eseguiti, il numero di macchine virtuali attive rispetto a quello configurato, lo stato di salute del sistema, etc. Realizzato tramite Nagios o Zabbix o direttamente tramite il gestore delle Dashboard di Sii-mobility stesso.

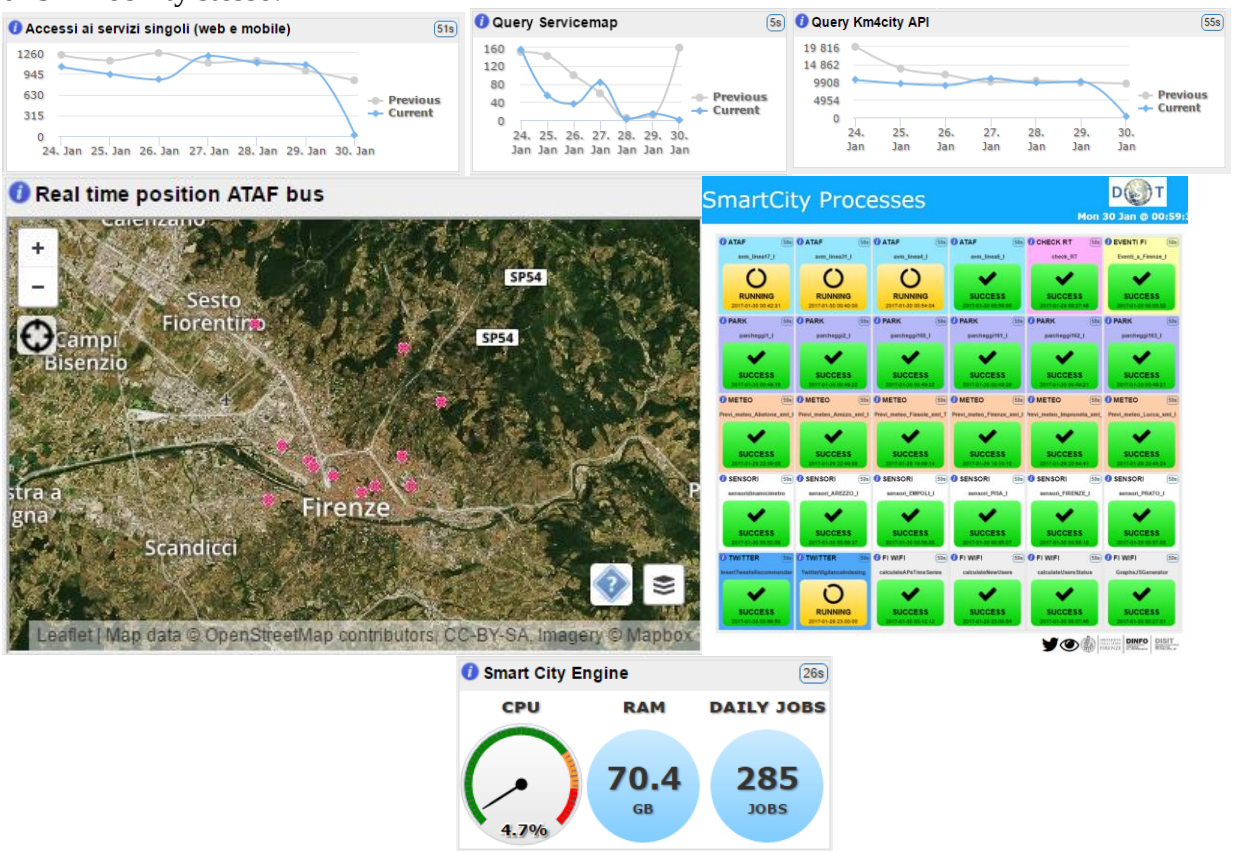


Figura: Visione di sintesi dei processi di smart city. Anche accessibile da http://www.disit.org/dashboardSmartCity/view/index.php?iddasboard=NTQ=&nome_dashboard=Stato%20processi%20-%20Small#

Sono pertanto **moduli consigliati minimi** per questo Modulo di Monitoraggio

- MINIMO: Dashboard Builder
- MINIMO: Notificator

WHAT NEXT: *In questo ambito:*

- *Realizzazione di metriche e di dashboard specifiche.*

3.6 Data Analytics

Algoritmi e processi di Data Analytic: calcolo dei comportamenti tipici dell'utenza su base APP (SI)

- Engager
  - calcolo dei comportamenti tipici dell'utenza su base APP
  - calcolo dei punti di interesse tipici degli utenti su base APP
  - calcolo dei punti di interesse tipici degli utenti (parziale)
  - calcolo delle condizioni premiali per i bonus (dall'engager)
  - etc.
- Suggestion

  - calcolo dei suggerimenti geolocalizzati (SI), direttamente dal suggestion on demand
  - calcolo dei suggerimenti in base al profilo utente su base APP
  - calcolo dei suggerimenti in base al profilo utente su base APP (SI), direttamente dal Suggestion on demand
  - calcolo delle traiettorie tipiche per persona su base APP
  - calcolo di mappe di origine destinazione su base APP (SI)
  - PAVAL assistant: *Conversational Recommender System*
  - etc.
- Data Analytics generico
  - calcolo delle predizioni sui parcheggi
  - calcolo delle predizioni sulle aree WiFI
  - calcolo di percorsi di routing DA-A TPL
  - ricostruzione del Traffico a partire dai sensor di traffico
  - etc.
- Routing
  - calcolo di percorsi di routing DA-A multimodali
  - calcolo di percorsi di routing DA-A pedonali
  - calcolo di percorsi di routing DA-A per il delivering
  - calcolo di percorsi di routing DA-A veicoli
  - etc.
- Ricerche via Servicemap
  - calcolo di ricerche indicizzate per POI e informazioni in generale (SI)
  - calcolo di ricerche indicizzate per POI e informazioni in generale
  - etc.

Questi processi di data analytics sono tipicamente sviluppati in R, Java, MapReduce, e anche in C. Con integrazioni in JavaScript e PhP.

WHAT NEXT: *Sono in corso di sviluppo svariati algoritmi predittivi, di routing, e di ricostruzione del traffico.*

3.6.1 Traffic Flow Monitoring

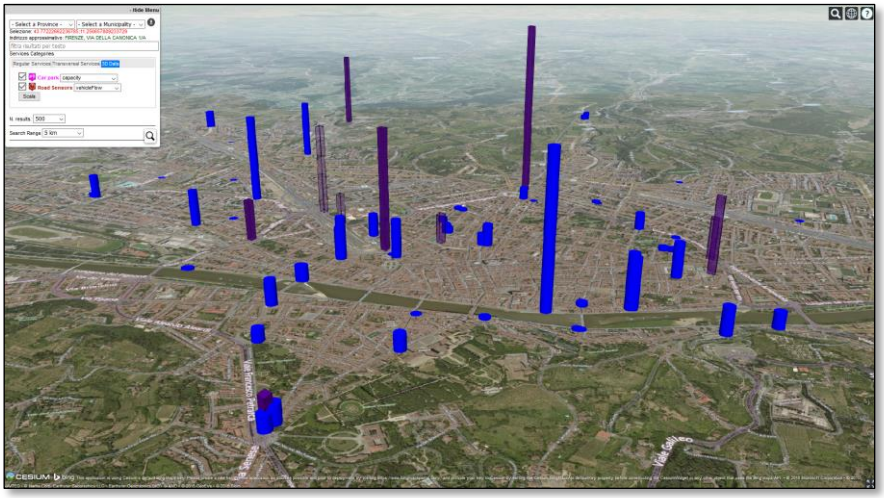


Figura: esempio di rappresentazione 3D dinamica dello stato dei sensori di flusso in Firenze. Sii-Mobility al momento acquisisce circa 780 sensori di traffico.
http://www.disit.org/servicemap3d/?c=provaabc

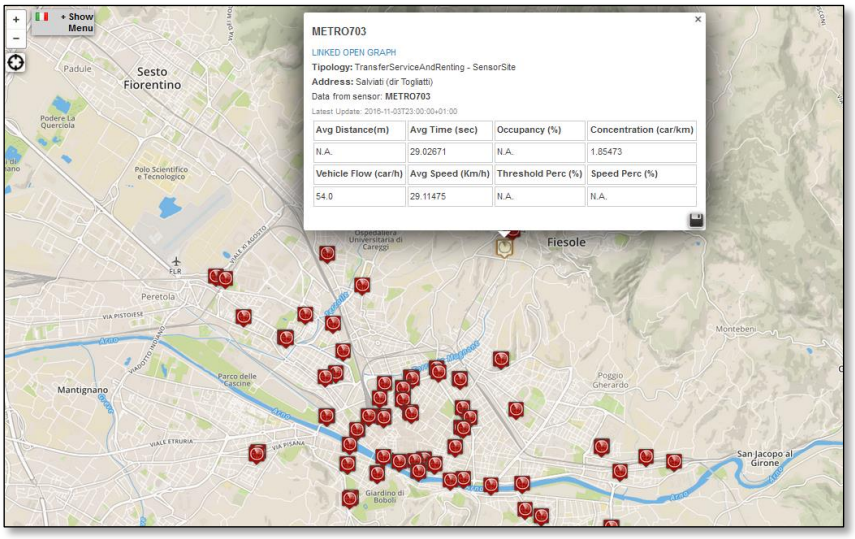
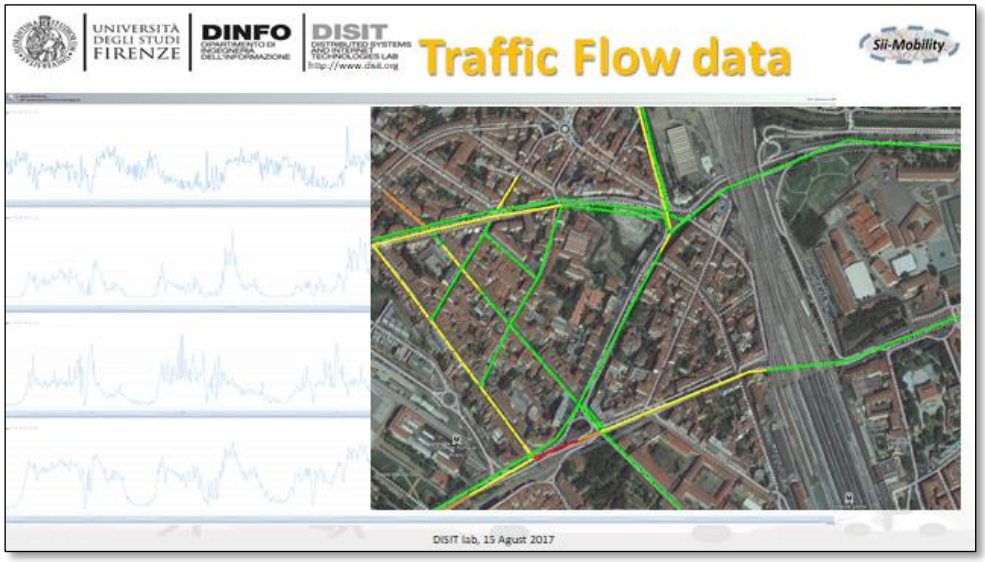


Figura: Posizione dei sensori di flusso in Firenze. Sii-Mobility al momento acquisisce circa 780 sensori di traffico. http://servicemap.disit.org , http://servicemap.disit.org/WebAppGrafo/api/v1/?queryId=e3e5029e1e8e1bafd1df80ba28d66b24&format=html

3.6.2 Traffic Reconstruction

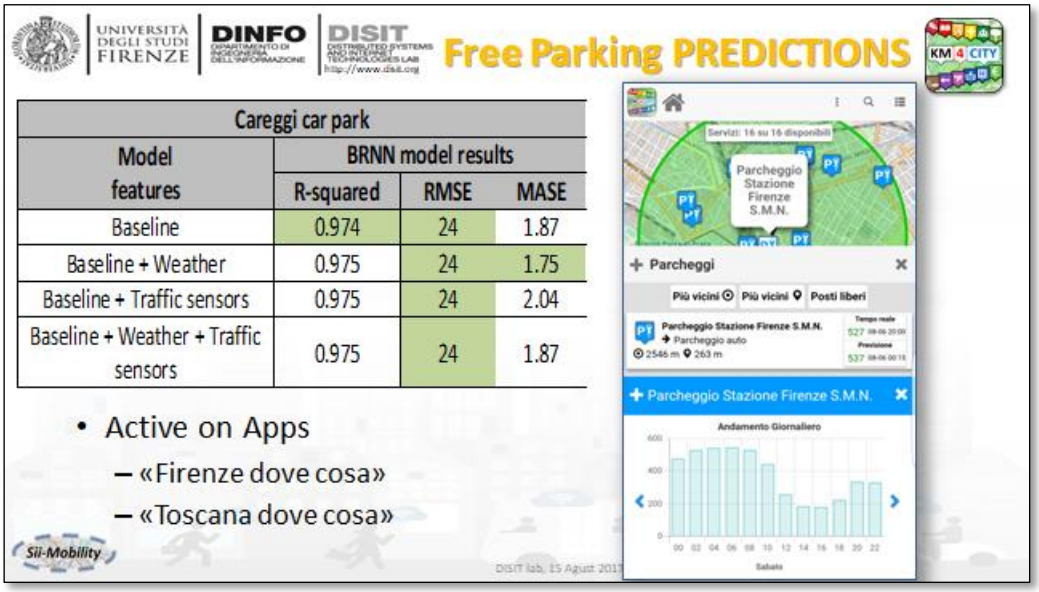
Per esempio: http://www.disit.org/siimobilitytrafficflow2
Questo tipo di elaborazione parte dalle misure di traffico flow effettuate con i sensori in punti strategici della città per ricostruire il traffico anche negli altri punti. A questo fine tiene conto della struttura della città, delle informazioni stradali, delle ZTL, dei varchi, delle preferenziali, delle svolte, etc.



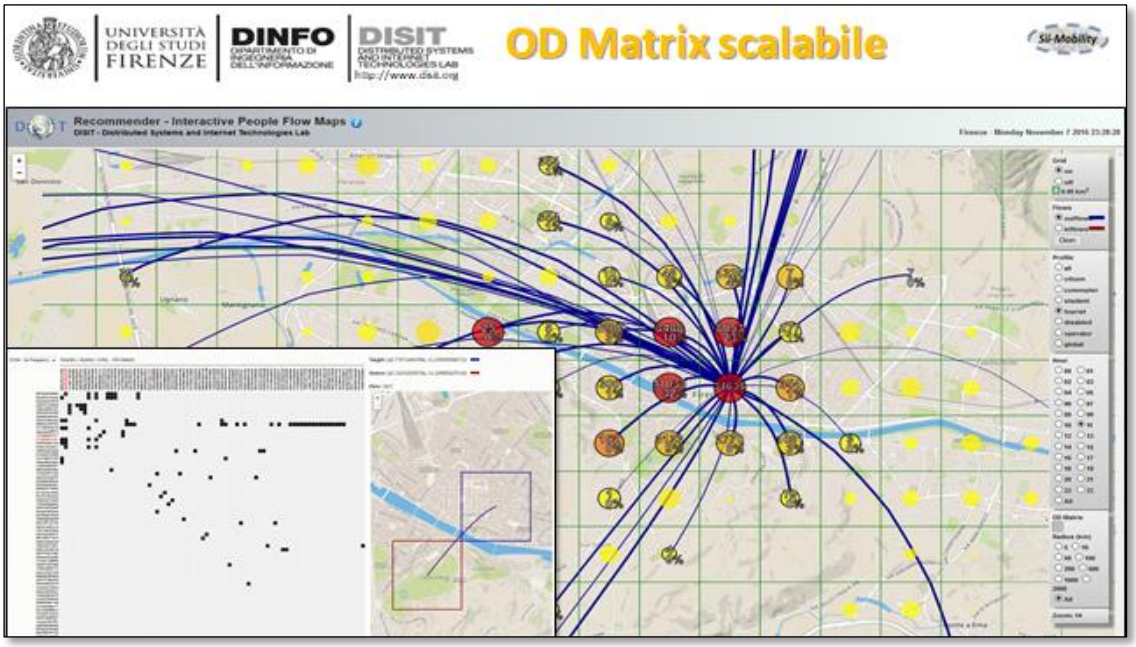
Questo strumento di ricostruzione è la base per verificare gli effetti di cambi nei sistemi di controllo traffico, negli orari dei servizi. Etc. La simulazione viene intesa come simulazione del comportamento di traffico o del sistema in certe condizioni. La stessa simulazione può consistere nel replay di situazioni di flusso e nella rivalutazione di strategie diverse da quelle attive al momento della raccolta dati.

3.6.3 Parking Predictions

Questo tipo di elaborazione permette di calcolare delle predizioni sugli slot liberi nei principali parcheggi coperti in Firenze e Toscana. La predizione viene ottenuta con soluzioni di machine learning neural network.



3.6.4 Origin Destination Matrix based on Mobile Data



3.6.5 Algorithm/Process Loader:

L'Algorithm/Process Loader è componente che permette di rendere semplice il caricamento di nuovi processi nel back office del SII di Sii-mobility Smart City. Questi algoritmi e processi possono:

- essere scritti in ETL/Java.
- fornire dati per la KB e/o fornire un'interfaccia nelle Smart City API.
- produrre risultati validi per le smart City API.

Il Algorithm/Process Loader acquisisce un pacchetto ZIP dove il processo è incluso. Lo spacchetta allocando il processo in opportune directory per il DISCES e lo registra sul DISCES stesso.
L'utente che carica il Processo puo' monitorare la sua esecuzione da remoto tramite l'interfaccia web del Algorithm/Process Loader.

3.6.6 Routing: Path Planner

Questo modulo sfrutta algoritmi di routing messi in esecuzione nel back office. In particolare è predisposta una VM specifica per il computo del routing. Il Routing viene esposto nelle smart city API per le Web e Mobile App. Sono al momento presenti algoritmi e processi per il

- calcolo di percorsi di routing DA-A pedonali (SI, presente in forma draft), distanza minima e percorso piu' quito.
- calcolo di percorsi di routing DA-A veicoli
- calcolo di Di percorsi di routing DA-A TPL (da sviluppare)
- calcolo di percorsi di routing DA-A multimodali (da sviluppare)
- calcolo di percorsi di routing DA-A per il delivering, logistica (da sviluppare)

E' già stata impostata tutta la struttura e pertanto aggiungere altri algoritmi non sarà molto oneroso.

Il path planner interface è direttamente integrate dentro il Service Map e dentro le Web e mobile App.

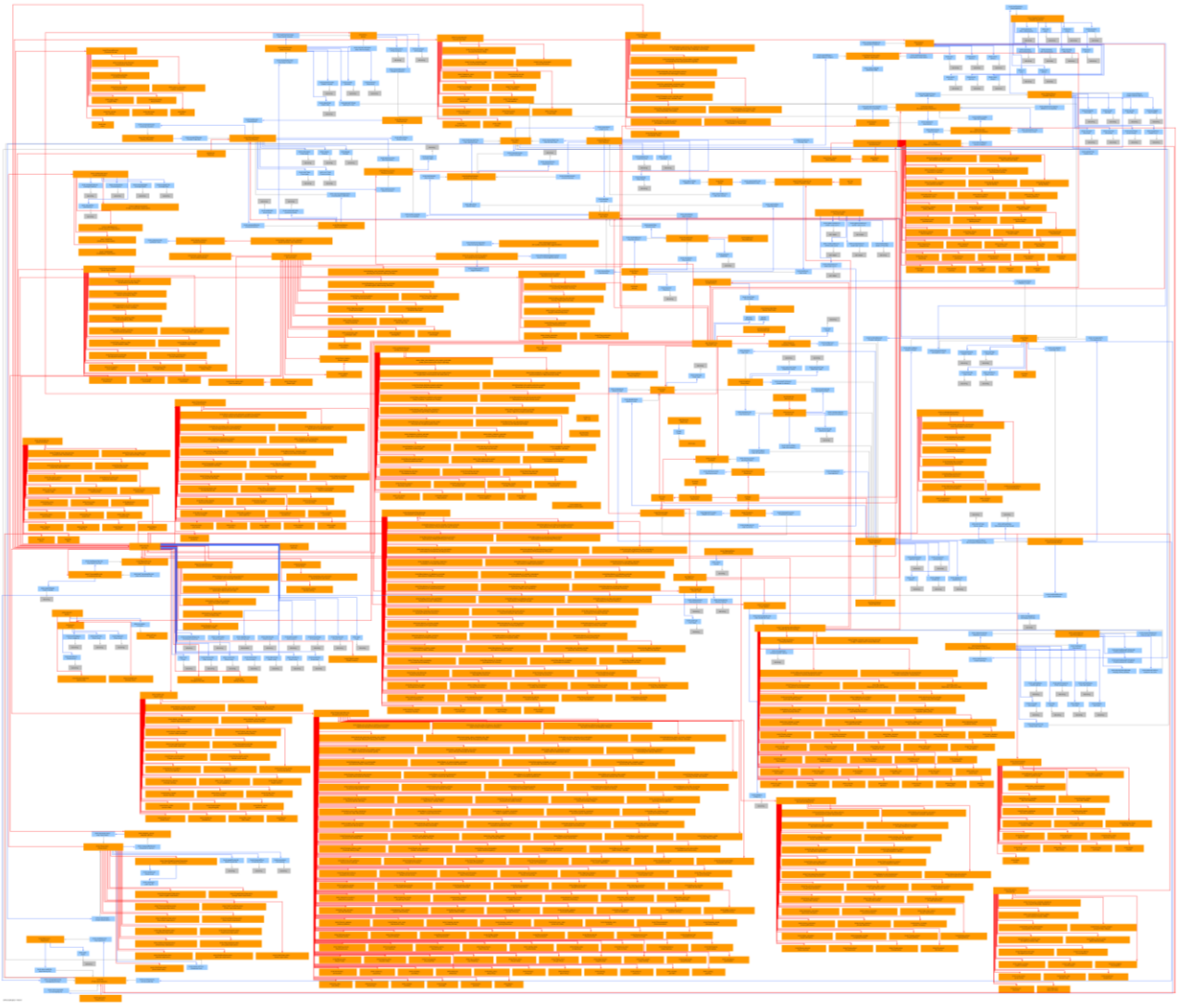
3.7 Knowledge base services

Questa area rappresenta il modello dati della Smart City con storage di tipo RDF basto su modello Km4City.

3.7.1 Km4City Ontology and Knowledge Base, version 1.6.4

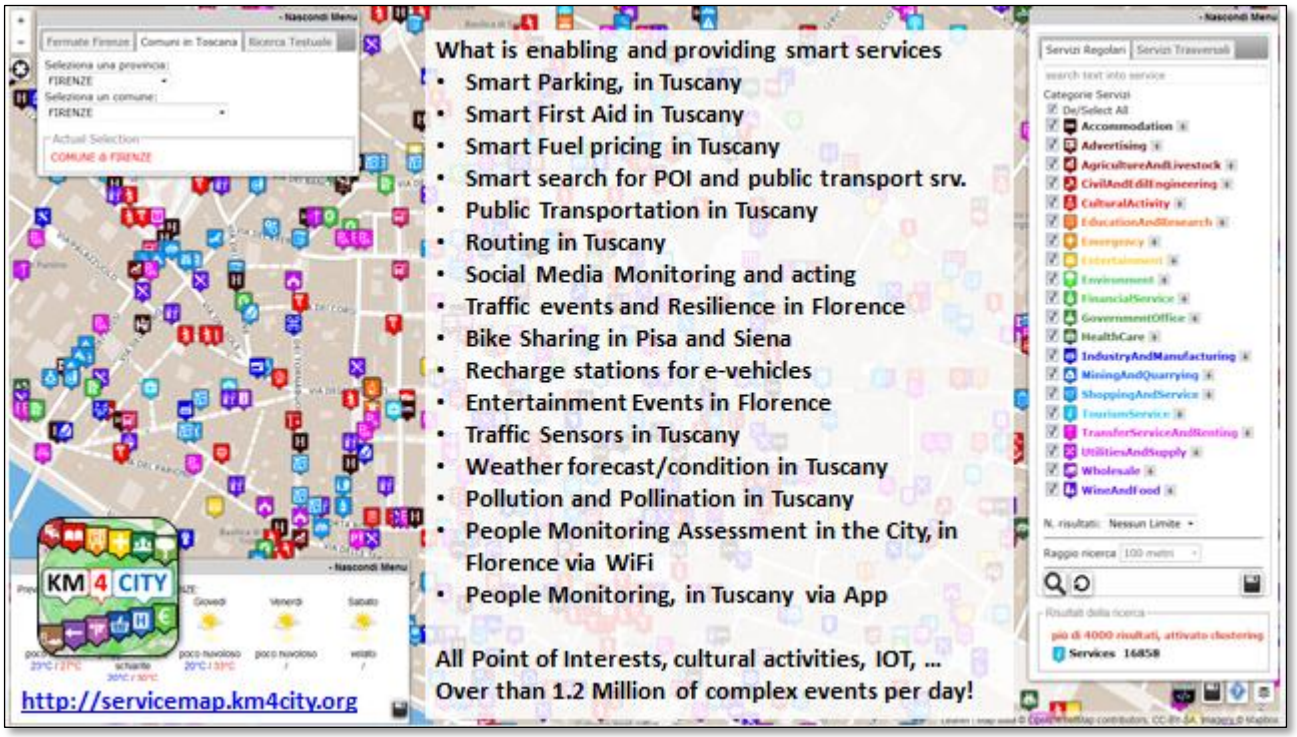
Km4City è una ontologia (Knowledge Model for the City), accessibile e libera da licenze di ogni sorta. La documentazione puo' essere acceduta da:

- SVG: http://www.km4city.org/img/Km4City-v1-6-4.svg
- ZIP con vari formati e doc: http://www.disit.org/6506
- Online documentation:
  - http://wlode.disit.org/WLODE/extract?url=http://www.disit.org/km4city.rdf
  - http://wlode.disit.org/WLODE/extract?url=http://www.disit.org/km4city.rdf&lang=it



La knowledge base for the smart city can be accessed via Smart City API, or directly in SPARQL. Lo strumento più semplice che permette di fare query visuali e navigare dentro il knowledge base è il Servicemap descritto in seguito ma vi sono molti altri tool più tecnici.

La versione 1.6.4 è molto migliorata rispetto alla versione 1.6.2. Sono state aggiunti molti dettagli per rendere efficiente il Routing, la ricostruzione di traffic flow, l'import di dati da Open Street Map, la gestione di molti sensori, la gestione di svariati tipi innovativi di POI, il bike sharing, il car sharing, predizioni sui parcheggi, etc. etc.



Sono pertanto **moduli consigliati minimi** per questo Modulo di Km4City

- MINIMO: Semplicemente documentazione
- Esistono strumenti specifici e come primo strumento il ServiceMap
- Esiste una versione di Km4City con il modello per il calcolo di dati di Rischio
- Esiste una versione di Km4City con il modello per la gestione di dati statistici

WHAT NEXT: *In questo ambito, estensione del modello ontologico Km4City per gestire:*

- *altre tipologie di sensori ed attuatori IOT/IOE*
- *KPI indicator delle smart city*
- *Altre tipologie di dataset*
- *Integrazione con la versione Km4City per dati di rischio*
- *Integrazione con la versione Km4City per dati statistici*
- *Modellazione dei meter, contatori di misurazione e gestione energia*
- *Modellazione dei sistemi di recupero energia e accumulo eneergia*

3.7.2 ServiceMap based on Km4City Knowledge base

La Knowledge Base di Sii-Mobility si basa sul modello Km4City descritto in precedenza, ed integra aspetti innovativi non presenti in Km4City come dettagli sugli aspetti di mobilità per esempio: dettagli stradali per il routing (direzioni, svolte, velocità, lunghezze, tipologia strada, tipologia mezzi consentiti), orari TPL, orari treni, licensing, sensori evoluti, etc. http://servicemap.disit.org

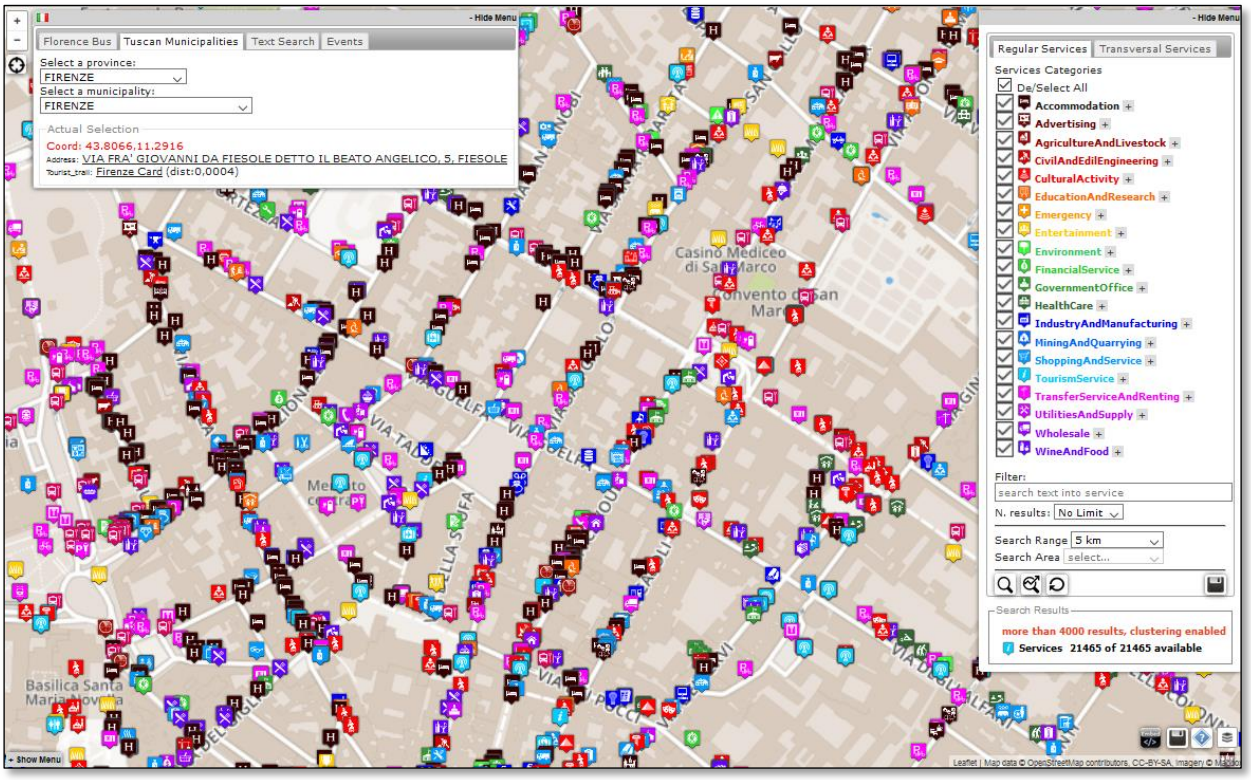


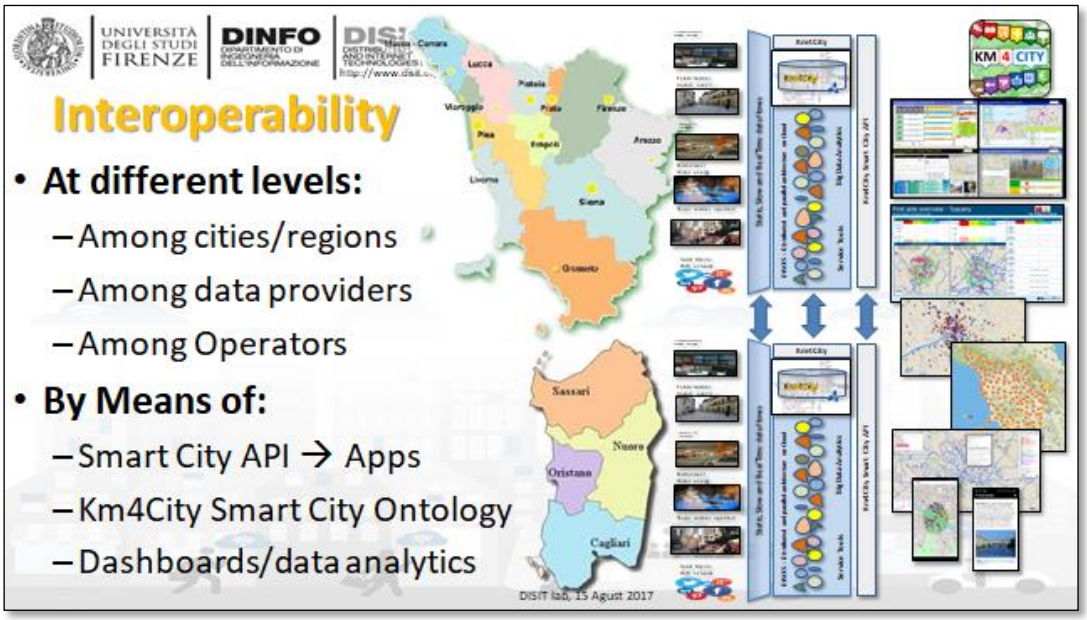
Figura: Strumento di Servicemap per lo sviluppo di query visuali sui dati della KB, per la ricerca, la generazione automatica di chiamate Smart City API, la comprensione del modello Sii-Mobility e Km4City.

Il Servicemap viene utilizzato come sistema di sviluppo per le APP come descritto nel meeting sii-mobility del 24 gennaio 2017. http://www.disit.org/6852
In questa area di sviluppo sono in corso di studio e sviluppo soluzioni per:

- Accelerare i tempi di creazione di nuove istanze di un RDF store centrato in una nuova città o regione, per esempio tramite OSM e arricchimento di altre informazioni. Si veda per esempio la soluzione centrata su Sardegna
- Abbandonare il modello che vede il grafo strade prelevato dal grafo della regione Toscana a favore di un grafo generato anche in Toscana da OSM che permette avendo i dettagli dovuto poter far routing, etc. ma anche traffic flow recostruction
- Scalare in termini di multi ServiceMap che collaborano alla copertura nazionale tramite un'interfaccia che permetta di distribuire le chiamate alle Smart City API sulla base delle competenze e della distribuzione dei grafi su un numero anche elevato di ServiceMap allocati in regioni multiple, anche con dati ridondati, etc.

Sono pertanto **moduli consigliati minimi** per questo Modulo di ServiceMap

- MINIMO: Servicemap basato su Virtuoso, con MySQL per la configurazione e log
- Si possono in seguito utilizzare strumenti specifici e come primo strumento il ServiceMap
- Esistono diverse versioni del Servicemap con modelli diversi in sperimentazione. In particolare: con i dati della Sardegna, con i dati derivati da Open Street Map, etc.
- Esiste una versione del Servicemap per il test di accesso
- Esiste una versione del ServiceMap Federato per poter gestire federazioni di ServiceMap e di Smart City API, come descritto nella figura seguente

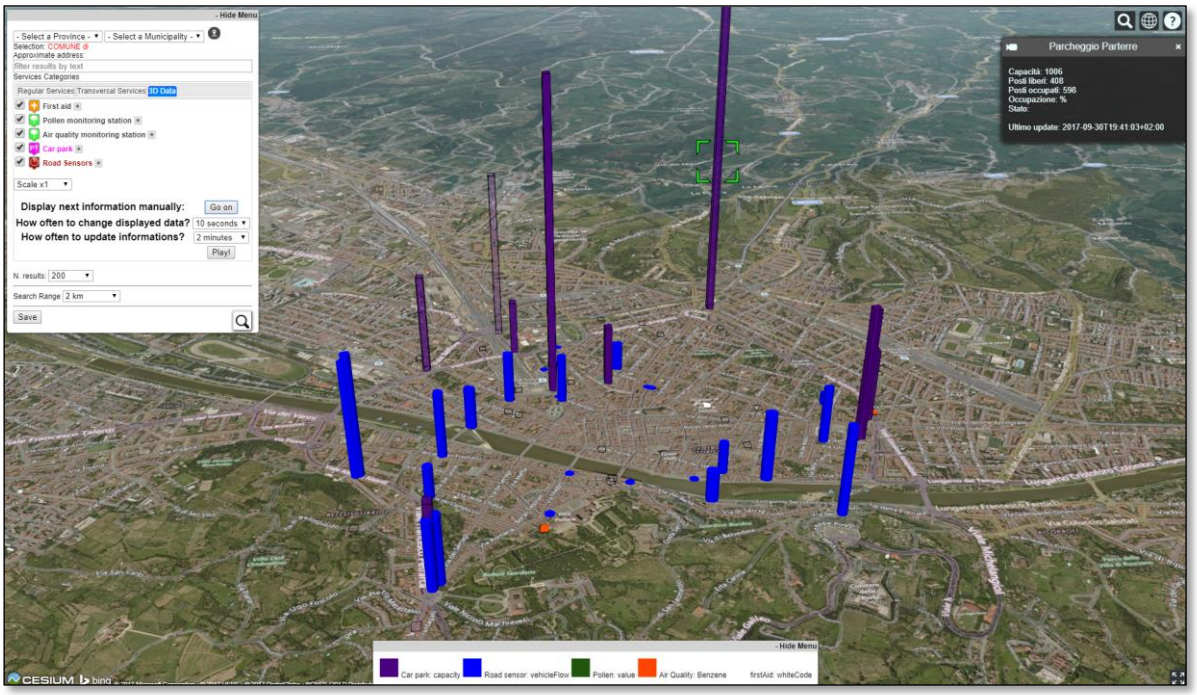


WHAT NEXT: *In questo ambito, l'estensione di Servicemap va nella direzione di:*

- *Gestione delle entità modellate con Km4City.*
- *Integrazione con aspetti di allocazione spazi in città*
- *Integrazione con la versione Km4City per dati di rischio*
- *Integrazione con la versione Km4City per dati statistici*
- *Accesso ai KPI direttamente collocati in città*

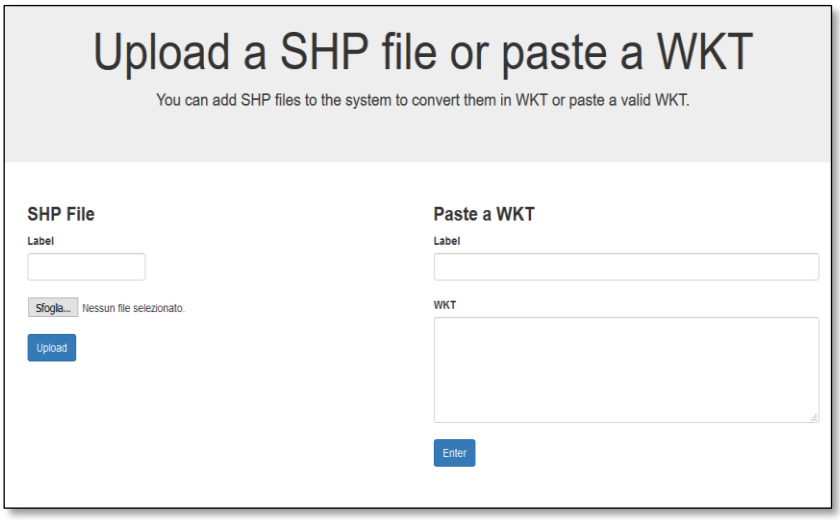
3.7.3 ServiceMap 3D

Versione del Servicemap per la rappresentazione animata 3D di dati real time.
Questa soluzione permette di essere programmata per l'animazione con la selezione delle sorgenti e e degli intervalli di presentazione: http://www.disit.org/servicemap3d/?c=provaabc



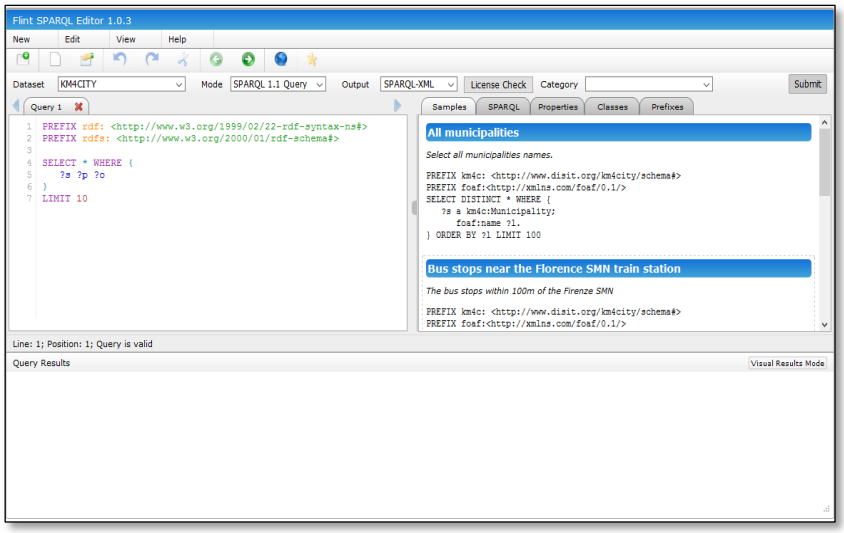
3.7.4 Loading Shapes and Paths for Search on ServiceMap/Km4City

Se il sistema ServiceMap deve poter fare delle query lungo linea o dentro un'area particolari (che non sono già elementi presenti nella knowledge base), è possibile caricare questi dati tramite un sevizio. **http://www.km4city.org/wkt/** Una volta caricate queste informazioni sono subito utilizzabili per le ricerche tramite l'interfaccia "Search on Specific Area" del ServiceMap.



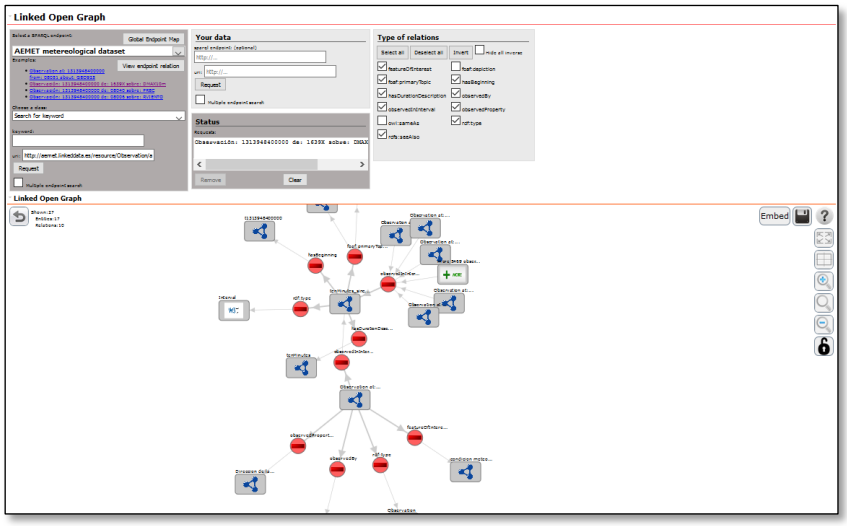
3.7.5 SPARQL interface, Conditional Access System + Licensing

SPARQL interface + Licensing: interfaccia per sviluppare query SPARQL e poter verificare il licensing sull'accesso ai grafi tramite license in relazione al tipo di utenza. Ogni data set che viene caricato potrebbe avere condizioni di licenza diverse e restrittive rispetto all'uso corrente per Km4City. Pertanto si rende necessario poter controllare l'accesso sulla base di regole e license assegnate alle singole persone. È stato sviluppata una soluzione per l'accesso controllato ai grafi del RDF store sulla base di un user profile. Da finalizzare priamo sviluppo completo. Accessibile direttamente su: http://log.disit.org/sparql_query_frontend/



3.7.6 Linked Open Graph, visual browsing on Km4City

Linked Open Graph RDF store navigation and query. Questo tool era già presente in DISIT lab viene esteso in Sii-mobility con funzioni di navigazione estesa e per servire il sistema di sviluppo App ServiceMap. Accessibile su http://log.disit.org



3.7.7 Statistic Data Map

Statistic Data Map: è un NoSQL database come RDF store che colleziona dati statistici riguardo alla città, alla mobilità etc. Ad oggi parzialmente sviluppato con tecnologia DataCube. La soluzione presente è parziale e non è ancora integrata con gli altri tool.

3.8 Smart City API

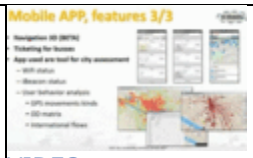
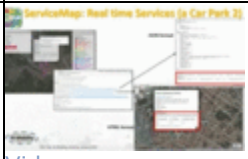
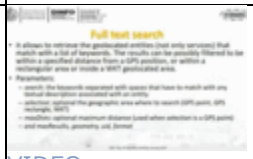
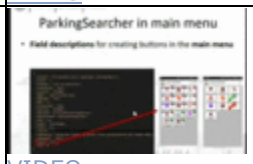
Smart City API Knowledge base: accesso a tutti i servizi delle Smart City API, compresi quelli descritti in precedenza, **si veda il dettaglio dello stato attuale nel deliverable DE3.16 e nella documentazione, slide, e video relativi al meeting di training del 24 Gennaio 2017 e all'Hackathon di primavera del 2017.**

- Search data: by text, near, along, etc...
  - Resolving text to GPS and formal city nodes model
- Empowering the city users
- Access to event information
- Supporting City Users in using Public Mobility
- Supporting City Users in using Private Mobility
- New Experience to access at Cultural and Touristic info
- New way to access at health services
- Access at Environmental information
- Profiled Suggestions to City Users
- Personal Assistant
- Sharing knowledge among cities (si veda sessione su ServiceMap)

Smart City API authentication and licensing Sviluppato in versione draft non completamente integrata con il Servicemap e nemmeno con le APP. Al momento si ha una parziale integrazione con DIM e SPARQL interface + Licensing.

Social Media Mutuated registration via OAuth: Sviluppato in versione draft non completamente integrata con il Servicemap. Al momento si ha una parziale integrazione con Engager e con User Profiler.

Documentazione sui API: http://www.disit.org/6852

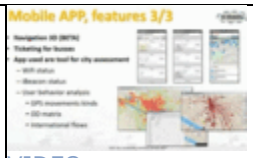
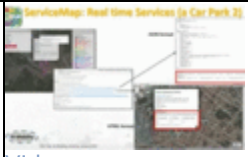
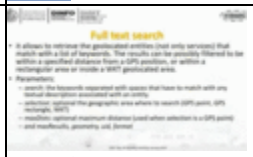
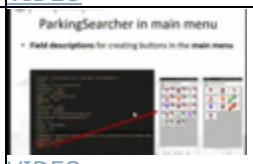
 SLIDE	 VIDEO	Sii-Mobility Workshop on APP development via Smart City API, Km4City
 SLIDE	 Video	Service map, development tool for APP. Generator for Smart City APP calls.
 SLIDE	 VIDEO	APP development via Smart City API, Km4City
 SLIDE	 VIDEO	How to develop Modules for Sii-mobility App via Smart City API
		Documentation of smart city API

Front end Smart City API domains to provide services to management smart city applications, and to web and mobile applications.	**CitySDK**	**ECIM**	**Transport .API**	**Navitia.io**	**Km4City**
API: Service Search					
(API GPS) Search Full Text	X2	X2		X	Xi
(API GPS) Search around a GPS point	X	X	X	X	Xi
(API GPS) Get location from GPS					Xi
(API LOCATION) Search along a line, a polyline	X				Xi
(API LOCATION) Search in an area, a closed shape	X			X	Xi
(API LOCATION) Search for street, region, municipality, etc.				X	Xi
API: Mobility					
Get Public Transport, bus-stops, lines, and schedule			X		X
Get Real time delay of Public Busses	X		X	X	X
Get Traffic Flows Status	X	X	X	X	(X)
Get Parking Status	X	X		X	X
Get fuel station status-prices					X
Get ped and vehicle Routing (Latitude, Longitude | POI; Latitude, Longitude | POI)	(X)	(X)	(X)		X
Get an Intermodal Routing	X	X	X	X	(X)
Get an Integrated Ticketing		X	X	X	(X)
Get a Routing for Good Delivering (multi stop planning)	X		X		(X)
API: Environment, Sensors and Actuators, IOT, health					
Get health structures (hospital, doctors, etc.)					X
Get First Aid Status					X
Get Weather Forecast	X			X	X
Get Sensor/Actuator Value/Status	X	X		X	X
Get pollution, temperature, pollination, etc.					X
API: User Participation and Awareness					
Get Social Media Monitoring Info			X		X
Save Crowd Sourcing Comments per service	X1		X		X
Save Crowd Sourcing Votes and Media per service	X	X	X		X
Get/Set variable message panel status per location					X
(API EVENT) Get Events in the city/area (today, week, and month)	X				X
API: Personal Assistant (engagement + recommender)					
Save User Profile	(x)	(x)		(x)	X
Get Suggestions on Demand					X
Get soundages, information, engagements					X
Get Civil Protection news in Push					X
Save Mobile Sensors Status	X	X		X	X
API: Domains of Geo Located Services					
(API INFO) Culture and Tourism	X	X			X
(API INFO) Point of Interest	X	X	X	X	X
(API INFO) Mobility and transport, parking, flow	X	X	X	X	X
(API INFO) Education and training	X	X			X
(API INFO) Government and Pub Services	X	X			X
(API INFO) Commerce and Industry	X	X			X
(API INFO) Health and personal	X	X			X
(API INFO) Public Energy, Energy and home	X	X			X
(API INFO) Energy and Mobility (fuel, recharging)	X	X	X		X
API kind of Call (JSON and/or HTML)					
SPARQL Query					X
SPARQL Query with Inference					Xi
REST	X	X	X	X	Xi
Query ID					Xi
API for Non Functional features					
Direct API Authentication	X	X	X	X	X
API Authentication via Social Media		X			(X)
Data Licensing Control	X		X	X	(X)

3.8.1 Mobile e Web APP:

Si veda deliverable DE3.16 e le informazioni e la documentazione distribuita il giorno 24 Gennaio 2017 durante il quale sono stati presentati i tool per lo sviluppo delle App, la versione dell'App come kit di sviluppo in formato sorgente, le modalità di sviluppo, le Smart City API accessibili e attivate, etc.

Documentazione sullo sviluppo di APP via API: http://www.disit.org/6852

 SLIDE	 VIDEO	Sii-Mobility Workshop on APP development via Smart City API, Km4City
 SLIDE	 Video	Service map, development tool for APP. Generator for Smart City APP calls.
 SLIDE	 VIDEO	APP development via Smart City API, Km4City
 SLIDE	 VIDEO	How to develop Modules for Sii-mobility App via Smart City API
		Documentation of smart city API

3.9 Ticketing and Booking

Ticketing and Booking backoffice Non sviluppato
Ticketing and Booking Frontend Non sviluppato

Ticketing and Booking: Bigliettazione integrata, etc., integrazione dei sistemi di bigliettazione integrata della regione e delle TPL.

Ticketing and Booking backoffice. Questo sottosistema permette di produrre biglietti in modo intermodale e multivendor. Queste funzioni sono anche esportate come servizi che possono essere usati da Web e Mobile App. Gli algoritmi di computo intermodale come quelli che calcolano i bonus dovrebbero poter essere messi in esecuzione su Big Data processing Grid. Il sistema di bigliettazione integrata dovrebbe poter integrare sistemi di pagamento diversi, e non solo per il trasporto pubblico, ma anche per il parcheggio, pedaggi, entrate in ZTL, etc.

Supporto alla prenotazione: per fornire informazioni di previsione su certi eventi pianificati e inattesi. Per esempio per la prenotazione dei prelievi, per la prenotazione delle consegne.

Ticketing and Booking Frontend. Che espone le Smart City API per le APP e le WEB App.