Piattaforma IoT MITT

Soluzione tecnologica su framework Open Source

Trentino Digitale SpA
Via G. Gilli 2, 38121 Trento | +39 0461 800111
tndigit@tndigit.it | tndigit@pec.tndigit.it
www.trentinodigitale.it

Indice dei contenuti

Via G. Gilli 2, 38121 Trento | +39 0461 800111
tndigit@tndigit.it | tndigit@pec.tndigit.it
www.trentinodigitale.it

1. Introduzione

Il documento descrive le seguenti componenti tecnologiche basate su framework OpenSource della piattaforma IoT del MITT:

- Layer di integrazione: Message Broker, Kafka Broker, Spring Boot Microservices e API Manager;
- Base Dati (SQL e NoSql)
- Complex Event Processor (Kafka Streams) e Identity Server
- Connettori per la ricezione dei dati dalle centraline
- Cluster DB NoSQL per la persistenza dei dati IoT
- Componenti di integrazione con i sistemi presenti sul MITT
- Connettori (API) esposti verso i sistemi verticali

Per la verticalizzazione di Fleet Management tale piattaforma consente l'implementazione delle seguenti funzionalità:

1) Visualizzazione in Real Time della flotta su Layer Cartografico
2) Visualizzazione dei percorsi effettuati dai veicoli (storico)
3) Inserimento del percorso da eseguire (vestizione)
4) Visualizzazione su mappa, in real time, le informazioni relative allo stato del mezzo
5) Creazione di allarmi mediante motore a regole per la manutenzione preventiva
6) Cruscotti web con fruizione di dati statistici sia su dati geo spaziali sia su dati di stato del mezzo

Per quanto riguarda i servizi al cittadino la piattaforma è a supporto dell'App Mobile Ibrida e del sito Web responsive che condivideranno la stessa base di codice HTML5/CSS/Javascript, con alcune funzioni disponibili solo nell'App Mobile.

Le funzionalità previste sono le seguenti:

1. Posizionamento utente su mappa con visualizzazione delle fermate dei trasporti pubblici presenti nel suo "intorno"
2. Orari programmati e orari reali con eventuali ritardi di linee autobus (paline virtuali)
3. Calcolo percorso intermodale con incluse tutte le linee di autobus attraverso le "Direction API" di Google
4. Informazioni sulla viabilità (code, incidenti, blocchi, lavori, ecc.) previa disponibilità delle informazioni

Per favorire la multi-modalità la piattaforma è predisposta per l'integrazione con realtà terze come parcheggi Car Sharing e stalli del Bike Sharing.

2. Architettura

2.1 Sicurezza

Per quanto riguarda i requisiti di sicurezza di livello applicativo, si descrivono nel seguito i meccanismi di sicurezza implementati. L'accesso al Message Broker è protetto da username e password inoltre, per motivi di sicurezza, il protocollo MQTT esposto dal Message Broker viene veicolato al di sopra di un canale crittografato di tipo TLS 1.2. Per l'instaurazione di una connessione sicura il server deve certificare la propria identità presentando un certificato digitale che contiene il nome del server, la Certification Authority che ne attesta l'autenticità, la data di scadenza e la chiave pubblica del server. Il client, prima di effettuare la connessione verifica la validità del certificato, quindi viene generata una chiave segreta di sessione che viene condivisa in maniera sicura tra il client e il server (attraverso il possesso da parte di quest'ultimo della chiave privata corrispondente al certificato digitale utilizzato). Concluso l'handshaking iniziale di connessione, tutti i successivi scambi di dati vengono criptati e decriptati utilizzando la chiave di sessione condivisa in precedenza.
Oltre al canale crittografato, la comunicazione verso il Message Broker prevede il meccanismo delle ACL; in questo modo è possibile garantire l'accesso controllato alle risorse evitando interferenze nella comunicazione degli attori della piattaforma.

Tutti i servizi SOAP e REST invece sono esposti tramite protocollo HTTPS invocabili esclusivamente attraverso il meccanismo di autenticazione e gestione dell'Api Manager;

Per quanto riguarda L'App Mobile Ibrida e le Web Application Responsive è utilizzato esclusivamente il protocollo HTTPS.

2.2 Architettura logica

Per la componente di App Mobile Ibrida e per il sito Web Responsive verranno implementate delle componenti di backend in grado di interfacciarsi con gli attuali sottosistemi del MITT.
La comunicazione tra l'App Mobile/sito Web ed i sistemi di backend avverrà attraverso l'invocazione di servizi REST basati su codifica JSON

L'immagine che segue rappresenta l'architettura logica delle componenti applicative a supporto dei servizi al cittadino.

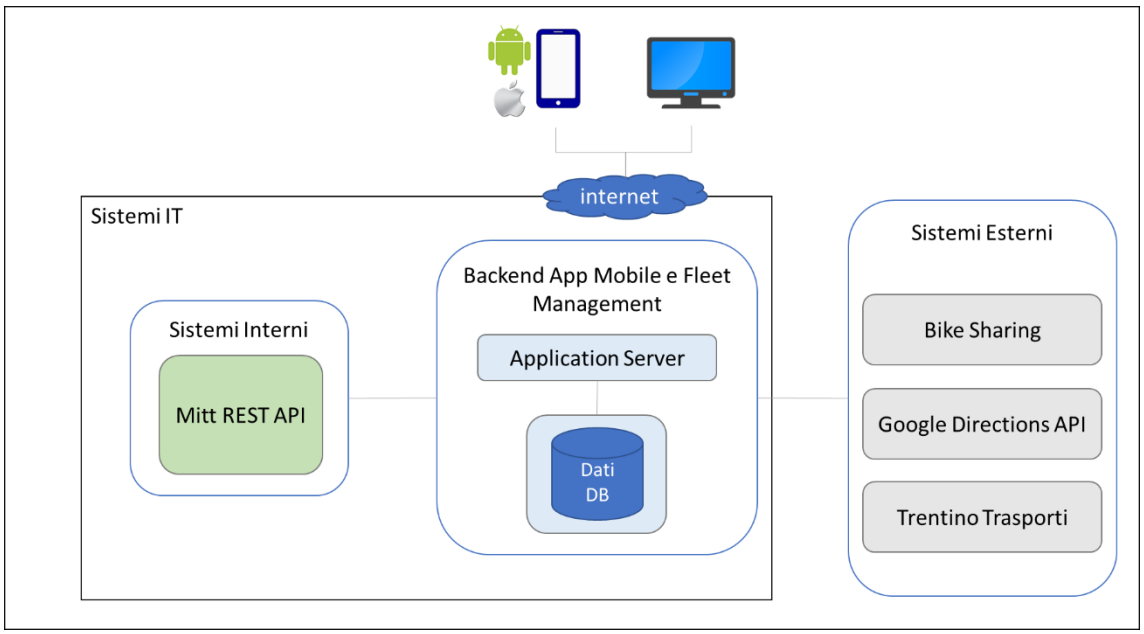


Figura 1 – Architettura Applicativa

Le build dell'App Mobile Ibrida sono realizzate per le piattaforme mobile iOS e Android.

Le notifiche push sono realizzate attraverso il servizio Cloud di Google FCM (Firebase Cloud Messaging), non rendendo quindi necessario l'installazione "on premise" di un server di push dedicato.

2.3 Architettura software

La realizzazione di quanto descritto si basa sulle seguenti componenti software Open Source qui elencate:

Per la componente di Integration Layer:

- Apache ActiveMQ: MQTT Message Broker;
- Apache Kafka: distributed message broker
- Spring Boot Microservices: acquisizione dati centraline + real time distributed processing
- WSO2 Identity Server: Identity & Access Management;
- WSO2 Api Manager

Per la componente di Persistence Layer:

- Apache Cassandra;
- PostgreSQL

Per la componente di Application Server Layer

- Spring Boot Microservices;
- Jetty Web Server

Verticalizzazioni

- Fleet Management
- Mobile App

2.4 Apache ActiveMQ

Utilizzato come componente di gestione eventi il Message Broker abilita le applicazioni allo scambio di informazioni asincrone e a pubblicare messaggi verso vari client con un modello publish/subscriber basato su MQTT. All'interno della soluzione viene utilizzato come punto centralizzato della gestione degli eventi, e smista tra le componenti i messaggi più rilevanti.

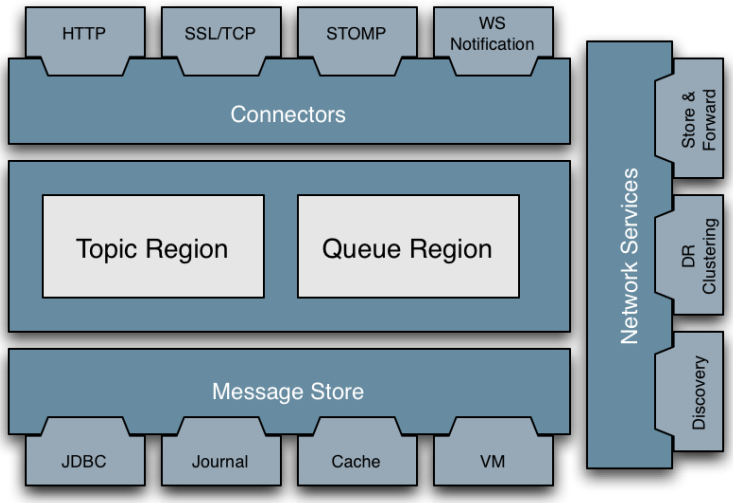


Figura 2 – Architettura Applicativa

2.5 Apache Kafka

Apache Kafka è una piattaforma open source che consente la costruzione di pipeline di ingestion ed elaborazione dei dati in tempo reale. Ogni record in Kafka viene "appeso" al record precedente, proprio come si fa con un file di log. I record sono raggruppati all'interno dei topic, ed i topic possono essere divisi in partizioni; ogni partizione contiene una porzione dei dati del topic, e il partizionamento avviene o in modo uniforme, o in base ad una chiave. I topic hanno una durata configurabile, e a seconda del caso d'uso possono anche essere usati come persistenza durabile. La scelta di Kafka è stata dettata dai requisiti di scalabilità, in quanto attualmente è l'unico prodotto capace di essere scalato in maniera orizzontale.

2.6 Spring Boot Microservices

Spring Boot abilita all'implementazione di applicazioni Java in modo rapido e semplice, tramite un server incorporato (per impostazione predefinita si utilizza una versione integrata di Tomcat) eliminando così la necessità di contenitori Java EE. Con Spring Boot è possibile esporre componenti, come i servizi REST, in modo indipendente e compliant alle architetture a microservizi.
Spring Boot può agevolmente integrare diverse tipologie di librerie con diversi scopi; nel progetto corrente sono state integrate le seguenti:

- Kafka Streams
- Spring REST
- Spring Data JPA
- Spring Data Cassandra

Kafka Streams è una libreria di stream processing utilizzata per aggregare ed elaborare i dati. Questa libreria si appoggia su Kafka e rappresenta il layer di stream processing della piattaforma. È possibile sviluppare pipeline di trasformazione ed aggregazione dei dati sia in Java che in Scala, tramite una ricca suite di operatori che gestiscono sia trasformazioni stateless che stateful, garantendo ove possibile un processamento exactly-once. È possibile eseguire aggregazioni, join, finestre e scrivere trasformazioni custom.
Nel caso si utilizzino operatori stateful, è possibile utilizzare le API di query interattive di Kafka Streams per poter interrogare lo stato dell'operatore in tempo reale. Le query interattive non rappresentano un rimpiazzo delle classiche query su database, ma un rafforzamento di queste ultime: esse permettono di poter monitorare in tempo reale le operazioni che vengono eseguite sullo stream dei dati, ma non sono adatte per interrogazioni su larga scala, che sono affidate a database appositi.
Spring REST abilita l'esposizione di servizi restful.
Spring Data JPA e Spring Data Cassandra abilitano il layer di applicazione alla memorizzazione e alla lettura dei dati rispettivamente da PostgreSQL e Cassandra.

2.7 WSO2 Identity Server

Utilizzato come componente di Security, fornisce un sistema sofisticato di gestione della sicurezza e delle identità per le Web Application e i Servizi. Supporta il pieno controllo di sicurezza delle interazioni online, il **single sign-on** tra applicazioni e servizi, la gestione semplificata del processo di provisioning e gestione delle identità. Si basa su un sistema di controllo degli accessi a ruoli (RBAC), un sistema di controllo degli accessi definito da policy a grana fine, utilizzabile sia per il SSO degli utenti alle applicazioni e servizi sia di applicazioni a servizi.
Come tutti i prodotti WSO2, l'Identity Server è il risultato della composizione di diversi moduli funzionali auto-consistenti e garantisce gli stessi elevati livelli di standardizzazione, interoperabilità, parametricità, portabilità, riusabilità, integrazione e sicurezza del framework WSO2 Carbon sul quale è costruito.

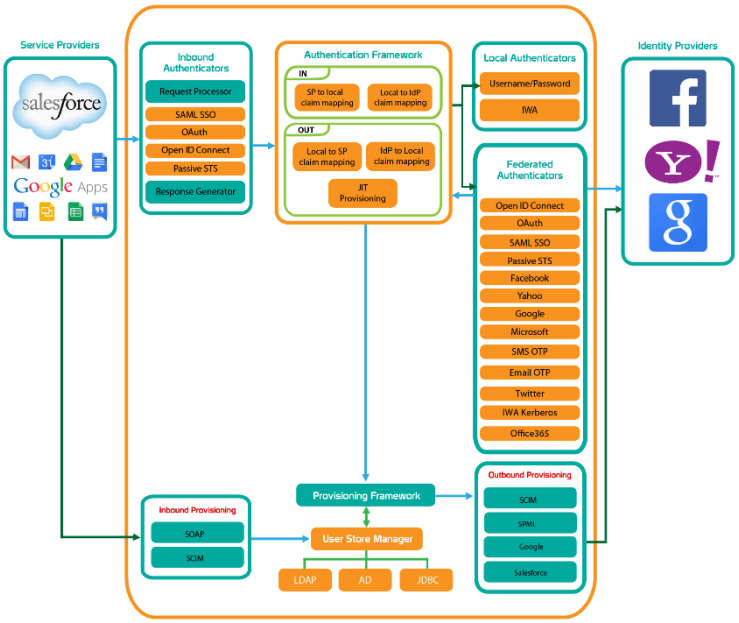


Figura 3 - Architettura WSO2 Identity Server

2.8 WSO2 API Manager

L'API Manager protegge l'accesso ai dati esposti dai servizi web. Tramite l'utilizzo dell'API Manager è possibile esporre gli endpoint in maniera tale che solo gli utenti autorizzati possano leggere i dati. Inoltre, è possibile fissare un limite alle letture che verranno effettuate, in modo da creare una sorta di monetizzazione dei dati esposti, che di base potrebbe non essere prevista dalla specifica base dati. In generale, utilizzando questo approccio è molto facile integrare qualsiasi servizio esterno e renderlo parte della piattaforma mantenendo gli stessi utenti e ruoli utilizzati anche per gli altri servizi, a patto che quest'ultimo esponga degli endpoint http. Usato in accoppiata con l'Identity Server, l'API Manager garantisce in maniera semplificata l'implementazione del meccanismo di sicurezza OAuth2.

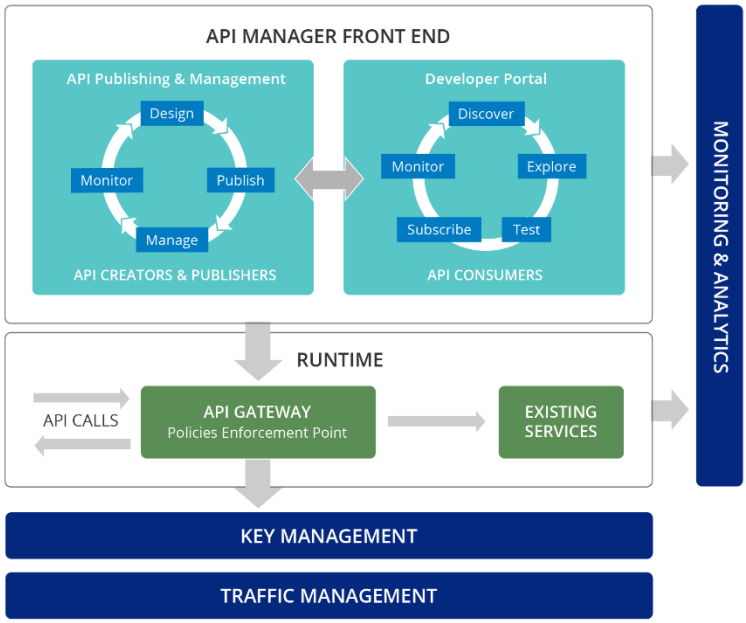


Figura 4 - Architettura WSO2 API Manager

2.9 Fleet Management

Fleet Management è composta da una componente web cartografica e da una sezione di dashboard. Mediante la stessa sarà possibile accedere a tutti i dati storici e real-time della flotta, sia in modalità mappa che in modalità analitica. Fleet Management consente inoltre di gestire la flotta veicoli e del relativo equipaggiamento di bordo al fine di programmare, analizzare e ottimizzare le attività di manutenzione.

Nello specifico mediante Fleet Management è possibile:

- Tenere sotto controllo tutti i veicoli sia in movimento che fermi;
- Effettuare delle analisi specifiche visualizzando dashboard che presentano sia dati raw sia dati opportunamente elaborati dalla componente di piattaforma;
- Impostare regole di geofencing selezionando su mappa le aree e classificandole;
- Inserire le istruzioni di vestizione per il percorso assegnato;
- Visualizzare i percorsi di tutti i mezzi;
- Monitorare in real-time lo stato di funzionamento del veicolo;
- Monitorare in real-time lo stato degli apparati di bordo;
- Pianificare gli interventi di manutenzione;
- Impostare allarmi al fine di evitare rotture improvvise di componenti e di intervenire in maniera rapida in caso di segnalazione;
- Rilevare in tempo reale le attività ed i consumi della flotta e individuare le aree di risparmio;

2.10 Mobile APP

Muoversi in Trentino è una mobile app ibrida, disponibile per IOS e Android, che agganciandosi agli Open Data della mobilità offerti da Trentino Trasporti permette di usufruire di tutta una serie di informazioni legate alla mobilità nella regione Trentino.

Nello specifico mediante Muoversi in Trentino è possibile:

- Visualizzare il programmato delle corse dei mezzi
- Visualizzare il posizionamento in tempo reale dei veicoli sulla linearizzata della corsa
- Calcolare del ritardo del mezzo in tempo reale
- Calcolare il percorso tramite le API Directions di Google
- Ricercare per linee e stazioni
- Visualizzare i dati del Bike Sharing
- Visualizzare i dati delle news sulle linee

3. Dimensionamento

3.1 Architetura tecnica

Si riporta di seguito il diagramma di architettura tecnica del progetto MITT con il relativo dimensionamento delle singole componenti. Il capacity delle singole componenti si basa sulle assunzioni descritte nell'elenco sottostante e fa riferimento ad un'architettura di partenza che presenta la peculiarità della scalabilità sia orizzontale sia verticale. Pertanto, il numero di VM e/o delle risorse elaborative può/deve essere variato in funzione del volume e del throughput dei dati immessi nel sistema.

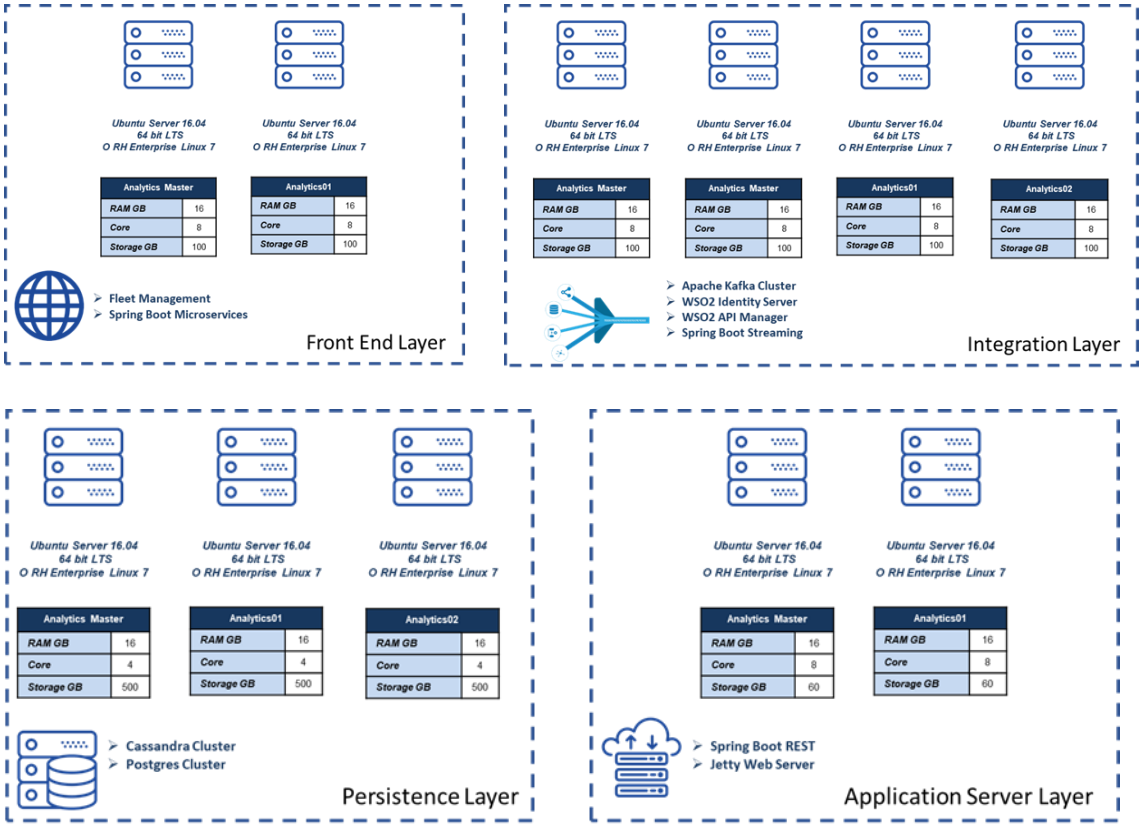


Figura 5 - Architettura Tecnica

3.2 Dimensionamento Piattaforma e Verticalizzazioni WEB:

Per una soluzione in Alta Affidabilità si necessita della giusta ridondanza delle macchine e dei componenti applicativi e pertanto si richiedono un totale di 16 VM con le seguenti caratteristiche per l'ambiente di Produzione:

Virtual Machine	N° VM	V.CPU (GB)	RAM (GB)	Storage (GB)	S.O.	Moduli SW
Persistence Layer	3	4	16	384	RH Enterprise Linux 7	Apache Cassandra
Persistence Layer	2	4	10	94	RH Enterprise Linux 7	PostgreSQL
Integration Layer	3	6	12	94	RH Enterprise Linux 7	Apache Kafka
Integration Layer	2	6	12	64	RH Enterprise Linux 7	Spring Boot REST + Microservices
Integration Layer	2	4	8	64	RH Enterprise Linux 7	API Manager
Front End Layer	4	4	8	64	RH Enterprise Linux 7	Spring Boot UDP + API Gateway

Per quanto riguarda invece l'ambiente di certificazione, questo risulterà speculare a quello di produzione ma con un sizing diverso (come elencato di sotto):

Virtual Machine	N° VM	V.CPU (GB)	RAM (GB)	Storage (GB)	S.O.	Moduli SW
Persistence Layer	3	4	8	110	RH Enterprise Linux 7	Apache Cassandra
Persistence Layer	2	2	6	65	RH Enterprise Linux 7	PostgreSQL
Integration Layer	3	4	8	50	RH Enterprise Linux 7	Apache Kafka
Integration Layer	2	4	8	40	RH Enterprise Linux 7	Spring Boot REST + Microservices
Integration Layer	2	2	4	40	RH Enterprise Linux 7	API Manager
Front End Layer	2	2	6	40	RH Enterprise Linux 7	API Gateway
Front End Layer	2	4	4	40	RH Enterprise Linux 7	Spring Boot UDP

4. Schemi Infrastrutturali

Di seguito uno schema che mostra la suddivisone delle componenti sulle specifiche LAN. Le risorse sono state mappate sulle risorse dichiarate nelle tabelle del capitolo 2.

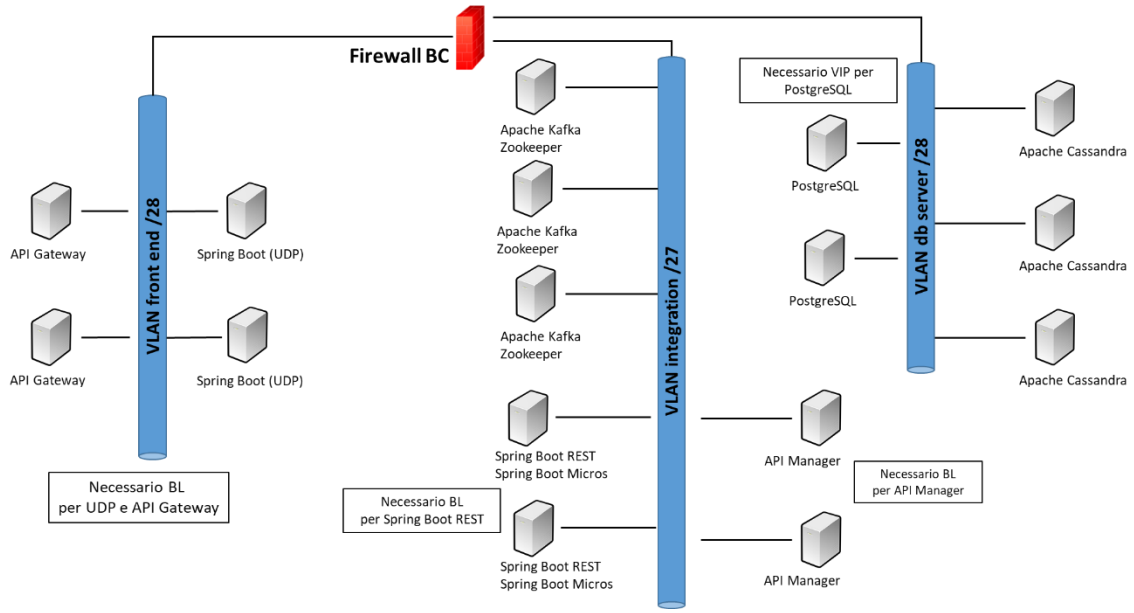


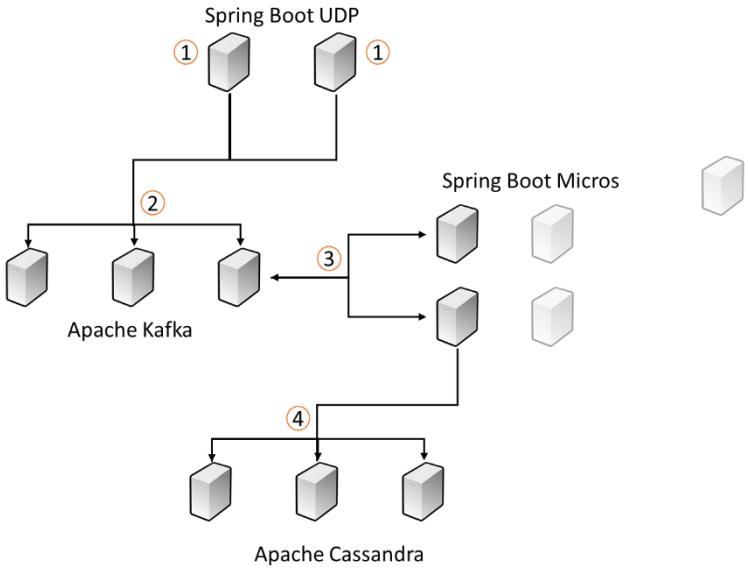
Figura 7 - Architettura di rete 1

5. Flussi di Piattaforma

Di seguito vengono riassunti i principali flussi di piattaforma

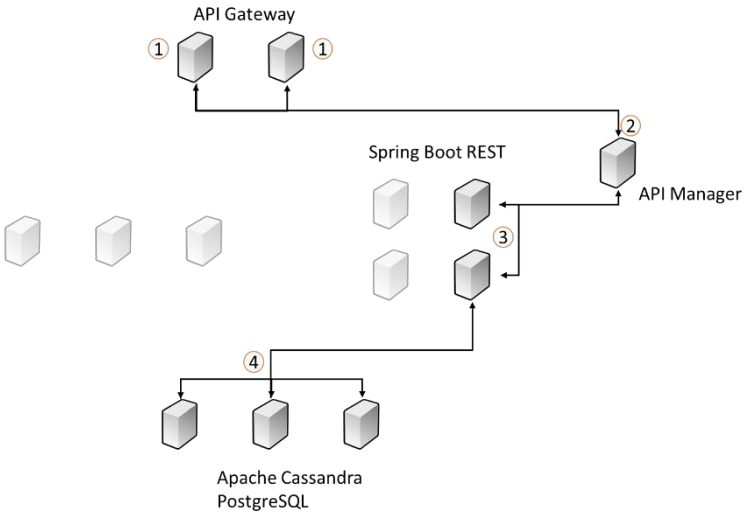
5.1 Flusso UDP – Elaborazione messaggi centralina

I messaggi inviati dalle centraline vengono acquisiti tramite protocollo UDP ed inviati sul broker distribuito Kafka. I microservizi di streaming processing elaborano i dati, li trasformano, li aggregano ed infine li persistono.



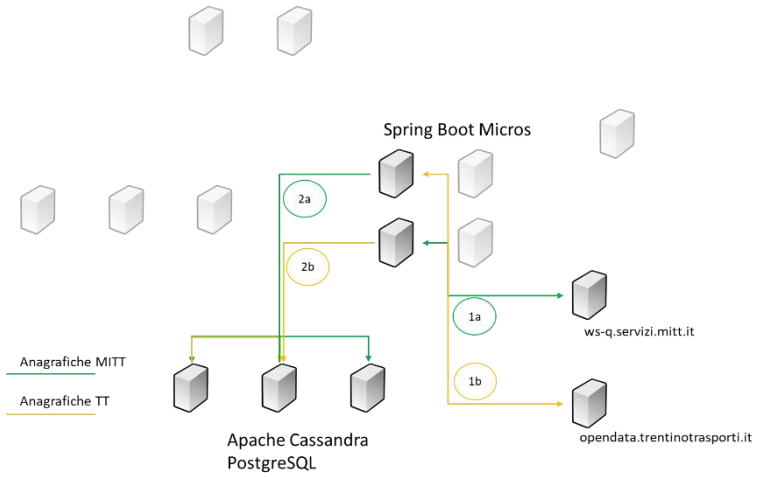
5.2 Flusso HTTP – REST API

Le chiamate in ingresso dalle verticalizzazioni Fleet Management e Mobile App vengono ribaltate tramite Reverse Proxy su un API Manager responsabile di verificare i criteri di sicurezza. Se abilitate, le stesse vengono reindirizzate ai microservizi Spring Boot REST i quali interrogano lo strato di persistenza e ritornano i dati al chiamante.



5.3 Flusso HTTP – Ingestion Anagrafiche

Ad intervalli schedulati (di notte) vengono attivati i microservizi di ingestion i quali interrogano i servizi REST messi a disposizione sia dal MITT che da Trentino Trasporti. I dati acquisiti vengono elaborati prima di essere persistiti all'interno dei diversi database.



5.4 Google Directions API

Quando dalla Mobile APP si richiede il calcolo di un percorso la piattaforma effettua una chiamata alle Google Directions API. La chiamata avviene su internet attraverso un token di sicurezza rilasciato da Google.