BeeToBit S.R.L.
via Sidney Sonnino, 32 - 09125 Cagliari (CA)
C.F./P.IVA: IT 03609460922
PEC: beetobitsrl@legalmail.it

API OPENDATA TRAFFICO DEL COMUNE DI CAGLIARI

OBIETTIVI E REQUISITI DEL PROGETTO

Lo scopo del progetto **OpenData Traffico** è di creare un'API Open Data, retta da infrastruttura Cloud, che erogasse i dati collezionati nel territorio comunale, e in quello dei comuni limitrofi, sotto forma di **API REST** pubbliche, utilizzabili sia dall'Amministrazione stessa che da terze parti per l'implementazione di applicazioni e, in genere, servizi digitali per i cittadini. Il progetto, inoltre, si pone degli obiettivi ***tecnologici*** e ***funzionali*** aggiuntivi:

1. **Scalabilità:** le API sviluppate dovranno; essere in grado di scalare orizzontalmente con semplicità: idealmente dovrebbe essere possibile far scalare l'intera infrastruttura delle API senza necessità di interventi di revisione manuale dell'infrastruttura;
2. **Gestione throttling e profili d'uso**: dovrà essere possibile impostare una limitazione *a priori* del numero di invocazioni per secondo consentite per le API, possibilmente con differenziazione a seconda di profili d'uso predefiniti; in caso di superamento del throttling l'API dovrà rispondere in modo specifico (as esempio `HTTP 429: Too Many Requests`), in modo che gli sviluppatori che fanno uso dell'API possano gestire opportuni meccanismi di retry;
3. **Applicazione semantica del modello REST:** la struttura delle API implementate dovrà rispecchiare le best practices delle architetture REST, individuando una serie di *risorse* e usando unicamente i verbi HTTP per indicare le operazioni da svolgere su queste risorse.
4. **Standardizzazione dei formati di dati:** i dati in input e output dovranno sempre seguire formati standard, quando presenti. In particolare, le date dovranno essere in formato ISO 8601 (https://it.wikipedia.org/wiki/ISO_8601);
5. **Documentazione automatica per gli sviluppatori:** le API dovranno supportare tecnologie per la generazione automatizzata della documentazione per gli sviluppatori, a partire dal codice delle API stesse o da file di definizione specifici (ad es. in formato YAML).

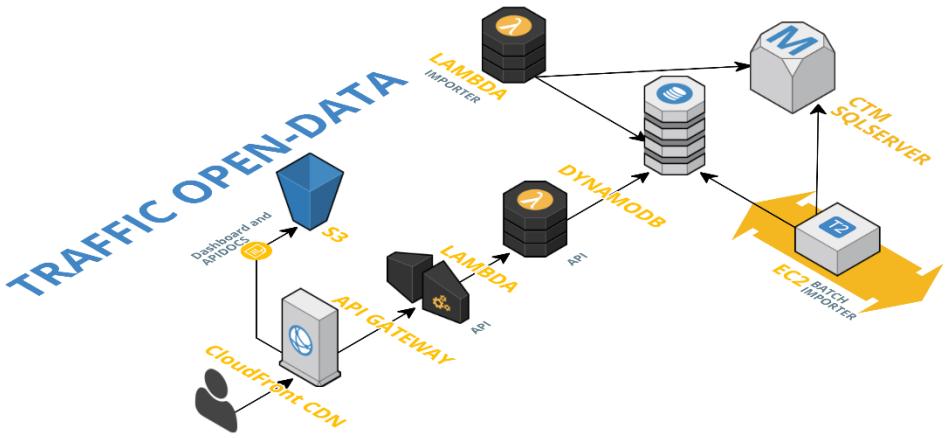
Il progetto si configura, quindi, come pilota per sperimentare metodi e tecnologie per la realizzazione in cloud di un'infrastruttura API Open Data più ampia rispondente ai requisiti descritti.

SOLUZIONE IMPLEMENTATA

ARCHITETTURA

La soluzione implementata su infrastruttura **Amazon Web Services** si basa principalmente sull'uso di microservizi managed serverless:

BeeToBit S.R.L.
via Sidney Sonnino, 32 - 09125 Cagliari (CA)
C.F./P.IVA: IT 03609460922
PEC: beetobitsrl@legalmail.it



Batch import: per questioni di performance l'import dei dati pregressi (circa 500 milioni di letture) è implementato tramite script Python su una batteria di macchina virtuali **Amazon EC2**; queste macchine verranno dismesse una volta completata l'importazione iniziale.

Data storage: la destinazione dei dati è il database managed nonsql **Amazon DynamoDB**. I dati sono importati *as-is* dal database Microsoft SQL Server e sottoposti a opportuna normalizzazione per il supporto dell'indicizzazione ottimizzata per DynamoDB.

Data import: il data storage Dynamo è sincronizzato alla sorgente SQL Server tramite uno script Python che gira su AWS Lambda; i dati vengono sincronizzati ogni minuto, con opportune politiche di retry.

API: la business logic delle API è implementata tramite un componente AWS Lambda in Python, collegato al servizio **API Gateway**, che si occupa dell'erogazione vera e propria degli endpoint REST, garantendo la crittazione tramite SSL e il supporto al throttling (sia per profili che per singolo metodo, con possibilità di white/black listing).

Deploy: Il sistema realizzato è strutturato e distribuito sull'infrastruttura AWS attraverso il servizio **Cloud Formation**. In questo modo è possibile modificarla e aggiornarla in modo controllato e prevedibile, applicando il controllo della versione all'infrastruttura AWS in modo analogo a un comune software applicativo (GIT).

Demo Dashboard: la demo della dashboard è stata realizzata come webapp JavaScript (tecnologia Bootstrap e jQuery) e viene ospitata su un bucket pubblico di **Amazon S3**, senza necessità di application server.

CARATTERISTICHE FUNZIONALI

- Le API supportano la **paginazione** dei risultati: se a una determinata chiamata corrispondono più di mille risultati, questi verranno restituiti a pagine contenenti 1000 risultati ciascuna, e un parametro chiamato `pagination_key` tramite il quale si potrà accedere, con chiamate successive, alle successive pagine di dati. Grazie a questa funzionalità, è possibile recuperare dataset di qualsiasi dimensione, senza limiti arbitrari imposti dal backend.

- Le API consentono di recuperare i dati in un intervallo temporale arbitrario, semplicemente specificando i parametri `from` e `to` in formato timestamp ISO 8601.

PERFORMANCE

Il tempo medio d'esecuzione dell'API con ritorno di 1000 risultati è di circa `450 msec`; sono al momento in corso ulteriori test di carico e performance, i cui risultati verranno compilati in un report separato

RIFERIMENTI

ENDOPOINT REST

GET - https://api.comune.cagliari.it/traffic/sensors

GET - https://api.comune.cagliari.it/traffic/sensor/{sensor_id}

GET - https://api.comune.cagliari.it/traffic/station/{station_id}

GET - https://api.comune.cagliari.it/traffic/stations

GET - https://api.comune.cagliari.it/traffic/station/{station_id}/readings

GET - https://api.comune.cagliari.it/traffic/sensor/{sensor_id}/readings

DEMO DASHBOARD

http://dashboard-api.comune.cagliari.it.s3-eu-west-1.amazonaws.com/traffic/pages/index.html

DOCUMENTAZIONE API PER SVILUPPATORI

http://docs-api.comune.cagliari.it/traffic/index.html