MENTOR

Mobilize Efforts for a harmonized diffusion of New smart and shared Mobility Technologies in the Overall program Regions

ID: 562850

Bike Sharing Merano Data Collector
Specifiche di integrazione nell'Open Data Hub

v1.2, 23.10.2019

Capofila IT / Lead Partner IT	Comune di Merano / Gemeinde Meran
Capofila CH / Lead Partner CH	Comune di Brig-Glis / Brig-Glis
Partner 1 IT	NOI Techpark
Partner 2 IT	SASA
Partner 1 CH	Postauto

Indice

Note preliminari

Alla luce della struttura dell'API del Data Provider (Digital Mobility Solutions GmbH, piattaforma MOQO), si è deciso in fase di avvio dei lavori di scegliere l'approccio di modellare i singoli veicoli (bici) come "station", secondo il concetto usato nell'Open Data Hub. Non saranno modellate nell'Open Data Hub le stazioni "virtuali" del servizio di bike sharing in quanto restituite solo come attributo statico nell'API esposta dalla piattaforma MOQO (return_requirements->areas).

End-point e meccanismo di autenticazione

L'end-point alla piattaforma MOQO è https://portal.moqo.de/api

Per effettuare le chiamate è necessario includere un token ed un Team ID.

Metadati stazioni

Attraverso il metodo "/cars" sono disponibili molteplici informazioni sul veicolo, la maggior parte dei quali è stato deciso di non integrare nell'Open Data Hub in quanto o non applicabili per il servizio di sharing in questione (molti si riferiscono a servizi di car sharing), o strettamente legati alla corretta esecuzione del servizio.

Le informazioni relative alle "station" bici da memorizzare nella tabella station sono quelle indicate in

Tabella 1.

Campi file webservice	Colonne tabella stations del database
id	**stationcode**
car_model->model_name_license	**name**
location	**pointprojection**

Campi file webservice	Colonne tabella stations del database
id	**stationcode**
car_model->model_name_license	**name**
location	**pointprojection**

Tabella 1: Mapping tra gli attributi del web-service della piattaforma MOQO e la tabella "station" del database.

Alcune osservazioni di dettaglio:

- i valori delle colonne **active** e **available** sono da settare a true.
- il valore della colonna **parent_id** non è da valorizzare.
- il valore della colonna **origin** è da settare con **BIKE_SHARING_MERANO**.
- il valore della colonna **stationtype** nel nostro database è da settare come **Bicycle**;

- i valori delle colonne **id** e **meta_data_id** (id corrispondente nella tabella metadata) vanno gestiti automaticamente in fase di scrittura del nuovo record nella tabella;
- il valore della colonna name è da settare concatenando le stringhe disponibili negli attributi .**car_model->model_name** e **license.**

Gli altri attributi sono da salvare come json nella tabella **metadata**, creando un opportuno riferimento con la stazione di riferimento (colonna station_id). Di tutti gli attributi disponibili, si è scelto di salvare solo il contenuto della struttura dati "**image**" che contiene diversi link all'immagine della bici.

Da notare che l'informazione legata alla posizione ("**location**") sarà estremamente variabile a causa degli utilizzi delle bici. L'informazione della posizione è da salvare anche nel JSON nella tabella **metadata**. In questo modo ad ogni variazione viene creato un nuovo record, ed il record nella tabella station viene agganciato al record nella tabella metadata più recente. In questo modo sono possibili in futuro analisi storiche sugli utilizzi delle singole bici.

Nota: se availabile = FALSE (vedi sotto), la location salvata è quella dell'ultima stazione in cui era available, per motivi di privacy. Ad ogni lettura del web-service il dato di posizione può essere quindi aggiornato.

Metadati tipi

I metadati dei tipi gestiti dal sistema vanno salvati nella tabella *type* del database. I tipi da salvare sono i seguenti:

- **availability** (indica se la bici è disponibile o meno per un noleggio). Va riutilizzato il tipo "availability" già gestito nell'Open Data Hub.
- **future_availability** (indica se la bici è disponibile o meno nell'immediato futuro). Va riutilizzato il tipo "future_availability" già gestito nell'Open Data Hub.
- **in_maintenance** (indica se la bici è in manutenzione, se non disponibile).

Alcune osservazioni di dettaglio:

- il valore del tipo "**availability**" è da prendere non dall'attributo "available" del web-service ma dallo **stato della prima availability slots**.
- il valore del tipo "**future_availability**" viene definito con la seguente logica:
    - se c'è solo uno slot nella struttura dati availability slots il record è da valorizzare con TRUE; è il caso di available = TRUE senza prenotazioni;
    - se c'è più di uno slot nella struttura dati availability slots il record è da generare, con questa logica: bisogna considerare l'attributo "until" della slot corrente (o l'attributo "from" di quella successiva). Se tale timestamp è inferiore al timestamp corrente sommato ad un valore di riferimento (definito ad **1 ora**) ora allora bisogna considerare il valore di availability della seconda finestra (che sarà il contrario della slot attuale); altrimenti si lascia il valore dell'attributo availability.

Con quest'ultima logica l'informazione future_availability diventa utile perché indica se ci sono "cambiamenti di stato" nella disponibilità della bici previsti entro la prossima ora. È da valutare in una futura release se riusciamo anche a gestire come tipo l'informazione esatta di quando la bici cambia stato.

Dati

Lo stato delle biciclette viene aggiornato ad ogni variazione di stato di una bici o in caso di un'operazione di un'utente tramite l'APP (es. richiesta prenotazione di una bici). Dal momento che i dati sono disponibili solo tramite GET (meccanismo pull) e non POST (meccanismo push), bisogna garantire una frequenza di aggiornamento nei dati piuttosto elevata in modo da fornire l'informazione più attualizzata possibile. Si propone di configurare inizialmente il sistema con chiamate all'API MOQO **ogni 5 minuti**.

I dati sono da salvare nelle tabelle measurement (in cui vengono salvati solo i valori più recenti) e measurementhistory. Alcune osservazioni di dettaglio:

- Il valore della colonna **period** nel nostro database va settata con **300** (vedi considerazioni sopra).
- Il valore della colonna **id** va gestito automaticamente in fase di scrittura del nuovo record nella tabella.
- Il valore della colonna **created_on** va gestito automaticamente in fase di scrittura del nuovo record nella tabella (è il timestamp relativo all'operazione di scrittura). Dal momento che l'API non fornisce alcuna indicazione sul timestamp in quanto i dati rappresentano lo stato attuale delle bici, tale valore è da memorizzare anche per la colonna **timestamp**.
- Il valore della colonna **double_value** è da valorizzare con il valore restituito dall'API MOQO.
- Il valore della colonna **provenance** è da valorizzare rispetto all'ID del record corrispondente nella tabella provenance.
- I valori delle colonne **station_id** e **type_id** sono da valorizzare in maniera appropriata in fase di scrittura dei record.