Simulazione prova di laboratorio di Sistemi e Applicazioni di Rete

Simulazione di esame del 4 Giugno 2018

Si richiede di sviluppare un'applicazione basata sulle tecnologie viste a lezione. Al fine di svolgere correttamente l'esame, è necessario avere account validi sulle seguenti piattaforme:

- Google App Engine[1]: si richiede il corretto funzionamento del relativo SDK locale per la verifica dell'esame;
- Mashape[2]: verificare il possesso della *API key* necessaria per l'utilizzo dei servizi indicati nella traccia.

Si raccomanda di verificare la soddisfazione dei vincoli indicati prima dell'inizio dell'esame. Il tempo massimo concesso per lo svolgimento della prova è di **2 ore**.

Consegna

Si realizzi un'applicazione Web per la gestione di un parcheggio in ambito smart city per il monitoraggio dei posti disponibili. L'applicazione deve mantenere lo stato di occupazione del parcheggio e deve implementare delle interfacce per la visualizzazione e per la manipolazione dello stato del parcheggio.

Si identificano tre tipi di interfacce di utilizzo:

1. visualizzazione e manipolazione dei posti disponibili tramite API REST
2. visualizzazione e prenotazione dei posti disponibili tramite pagine HTML
3. rilevazione dei posti tramite protocollo MQTT

Nell'implementazione si considerino i seguenti vincoli:

- si consideri un parcheggio multipiano composto da *3 piani*, in cui ogni piano ha capacità di *10 posti auto*;
- l'identificativo di ogni posto auto sia un numero di 3 cifre, in cui la prima cifra indica il numero del piano e le rimanenti il numero del posto auto univoco nel piano (esempio: 207 identifica il parcheggio numero 7 al secondo piano).

  Nota: oltre alle funzionalità richieste, si consiglia l'implementazione di una funzionalità per inizializzare il sistema (esempio: lo stato iniziale del del database del parcheggio).

API REST

L'applicazione deve esporre le seguenti funzionalità tramite opportune API:

- allo URI `/api/v1.0/availables` è possibile ottenere la lista dei posti disponibili;
- allo URI `/api/v1.0/slots/<slot-id>` è possibile ottenere o manipolare lo stato di un posto auto indicando il corrispondente codice identificativo.

L'interfaccia di utilizzo dell'API deve soddisfare **rigorosamente** il file di descrizione *OpenAPI* disponibile presso il sito del corso[3].

Interfaccia HTML

L'interfaccia HTML deve prevedere due pagine: la prima permette agli utenti di visualizzare lo stato del parcheggio; la seconda permette agli utenti di prenotare un posto auto.

La pagina di visualizzazione dello stato del parcheggio mostra l'elenco dei posti auto tramite i rispettivi identificativi, indicando per ognuno lo stato tramite un opportuno messaggio informativo o rappresentazione (ad esempio, colore verde per posto libero, colore rosso per posto occupato).

La pagina per prenotare il posto auto include una form per inserire la propria *email*. L'applicazione deve validare la validità sintattica della email fornita e selezionare autonomamente un posto auto fra quelli disponibili. L'applicazione

[1] https://cloud.google.com/appengine/
[2] https://market.mashape.com/
[3] https://weblab.ing.unimore.it/sar/1718/lab/parking-openapi.yaml

genera un *PIN* segreto di 5 cifre e memorizza la prenotazione associandola alla email e al PIN generato (si consideri un posto prenotato come occupato). Inoltre, ritorna all'utente l'immagine del codice QR (o una pagina HTML contenente l'immagine) che rappresenta l'email e il PIN. Il codice QR viene generato tramite un apposito servizio di terze parti[4]. Se non ci sono posti disponibili, l'applicazione mostra all'utente un opportuno messaggio informativo.

MQTT

Si richiede di implementare un sistema PubSub basato su MQTT per la gestione dei parcheggi che interagisce con le API dell'applicazione Web. Nell'ambito dell'esame il sistema deve essere emulato localmente. Si richiede di modellare degli opportuni topic per rappresentare al meglio lo stato di ciascun posto auto presente nei diversi piani del parcheggio rispettando i vincoli implementativi indicati.

Il sistema implementato è composto da due moduli python:

- uno script eseguibile `parking_publish.py` con interfaccia da linea di comando che permette di modificare lo stato di un posto auto e che pubblica questa informazione sulla rete MQTT. In dettaglio:
  - l'interfaccia dello script deve essere la seguente:

    ```
    parking_publish.py <slot-id> <state={free,occupied}>
    ```

    Ad esempio:

    ```
    parking_publish.py 303 occupied
    parking_publish.py 210 free
    ```

  - lo script modifica lo stato del posto auto tramite l'API `/api/v1.0/parkingslot`
  - lo script verifica l'esito della risposta e, in caso di stato modificato con successo, invia gli opportuni messaggi sulla rete MQTT;
- uno script eseguibile `parking_subscribe.py` che avvia un servizio di monitoraggio dello stato dei posti auto in un piano del parcheggio. Il servizio è implementato tramite un subscriber MQTT che rileva le modifiche allo stato di un piano del parcheggio e mantiene il numero di posti auto liberi ed occupati. Ad ogni aggiornamento dello stato, il servizio stampa l'informazione aggiornata su standard output a terminale. *NB*: questi servizi non comunicano con le API dell'applicazione Web, ma solo con la rete MQTT. In dettaglio:
  - l'interfaccia dello script deve essere la seguente:

    ```
    parking_subscribe.py <floor-number>
    ```

    Ad esempio:

    ```
    parking_subscribe.py 3
    ```

[4] https://market.mashape.com/neutrinoapi/qr-code