Piattaforme Digitali per la Gestione del Territorio

Relazione del progetto d'esame

Studente: Elvi Bllano (271506)

Anno Accademico 2017–2018

Specifica del progetto

Motivazione e contesto applicativo

L'obiettivo del progetto è quello di creare una piattaforma digitale che permetta alle persone con disabilità o mobilità ristretta di accedere ai luoghi e servizi di una determinata città. Per questo ho deciso di utilizzare un bot di Telegram per fornire le informazioni dei vari luoghi.

Stato dell'arte

Per creare questa piattaforma mi sono affidato a un sito web esterno che gestisce dati della stessa tipologia del contesto del progetto, chiamato Wheelmap.org, che fornisce una mappa dei vari punti della città con le informazioni sull'accessibilità, basata su OpenStreetMap, una mappa online opensource. Le informazioni fornite dal sito provengono dalla segnalazione e l'aiuto dell'utenza.

Contributo

L'obbiettivo è quello di fornire un bot di Telegram che acceda alle informazioni fornite dal sito per poi mandarle all'utente così da semplificare la ricerca di luoghi accessibili da persone con disabilità o mobilità ristretta. Il progetto e il relativo codice è disponibile su GitHub al link github.com/elvibllano/Progetto_PDGT_2017-2018.

Nome

Il bot di Telegram si chiama Progetto_PDGT_ElviBllano e il suo username è @ElviBllanoBot.

Dati e servizi esterni di riferimento

I dati che vengono utilizzati nel progetto sono forniti dal sito Wheelmap.org che permette attraverso la visione di una mappa online di poter vedere i vari punti di interesse delle città che possono essere attività di ogni genere dall'Educazione alla Salute. Essendo la piattaforma gestita attraverso le informazioni date dagli utenti, che possono aggiungere punti che non sono segnati nella mappa o indicare se uno di questi punti è accessibile o meno. Con punto o nodo identifico le varie attività segnate nella mappa esempio un negozio o una scuola.

La mappa online mostra in modo visuale per ciascun punto un colore e le informazioni generali. I colori sono, verde per la completa accessibilità, giallo per l'accessibilità limitata, il rosso per la mancanza di accessibilità ed infine il colore grigio indica che per quel punto non è stata definita l'accessibilità. Oltre alla mappa online, fornisce anche un set di API per poter aver accesso ai loro dati ed elaborarli.

Per poter accedere ai dati, Wheelmap.org fornisce alla registrazione al sito una chiave di accesso alle API che deve essere usato nella richiesta al sito per poter accedere ai dati. La chiave che uso per la mia piattaforma è Wjqc8yX1XJAFGsUomsLu.

Per ricevere i dati si fanno richieste HTTP all'indirizzo URL specifico e i dati ricevuti possono essere di tipo JSON, JSONP e XML. Nel mio caso ho deciso di utilizzare il formato JSON.
Ecco la struttura base di un dato fornito in JSON:

```
{
  "conditions": {
    "format": "json"
  },
  "meta": {
    "item_count":2,
    "item_count_total":5930
  },
  "results": [
    { result object }
  ]
}
```

Le richieste HTTP possono essere di 3 tipi:
GET: Quando dobbiamo ricevere i dati dal sito.
POST: Quando vogliamo aggiungere un dato.
PUT: Quando vogliamo aggiornare un dato.
Nel caso del mio bot uso solo le richieste di tipo GET perché il mio bot prende soltanto le informazioni.

Il sito ci permette di richiedere 3 tipi di dati, i nodi che indicano i punti segnati nella mappa e le informazioni relative, le categorie che indicano le categorie di riferimento per ciascun punto (esempio Sport o Educazione) e i tipi di nodo che indicano il tipo di attività.

Per richiedere i vari dati bisogna inserire i parametri base che sono:

api_key: la chiave d'accesso fornita dal sito con cui poter accedere ai dati

page: la pagina corrente dei dati

per_page: quanti dati devono essere mostrati per pagina. Di default sono 200 e il massimo è 500.

Poi ci sono parametri più specifici per i dati. Nel caso delle categorie e tipi di nodo sono gli stessi.

Per richiedere i dati l'indirizzo url è strutturato in questo modo:

http://wheelmap.org/api/categories?api_key&locale per la categoria
http://wheelmap.org/api/node_types?api_key&locale per i tipi di nodo

Oltre ai parametri base c'è anche il parametro locale che mostra la traduzione dei dati perché i dati vengono forniti di default in inglese.

Nel caso del nodo la richiesta è più strutturata ed è rappresentata come segue:

http://wheelmap.org/api/nodes?api_key&bbox&wheelchair

Oltre ai parametri base qui ci sono anche i parametri
bbox: indica le coordinate del luogo da cercare
wheelchair: indica lo stato di accessibilità ai punti
q: serve per poter cercare i punti utilizzando il nome del luogo da cercare in alternativa o in aggiunta alle coordinate.
La richiesta dei nodi può essere ulteriormente filtrata in base alla categoria di riferimento in questo modo:
http://wheelmap.org/api/categories/1/nodes?... : dove il numero indica una delle 12 categorie che sono presenti in questo momento della stesura della relazione.
Per maggiori informazioni sull'uso delle API fornite dal sito si può consultare la documentazione fornita al sito https://wheelmap.org/en/api/docs.
Un altro servizio che ho utilizzato è Altervista.org, una piattaforma web italiana di webhosting, per utilizzare gli script per il funzionamento del bot e le richieste dei dati a wheelmap.org.

Casi d’uso

Caso d’uso 1

Il caso d’uso principale della piattaforma è quella di fornire con l’app di Telegram un accesso ai dati riguardo al luogo d’interesse utilizzando smartphone o tablet data la loro mobilità maggiore e dato il fatto che sono diventati molto più numerosi e indispensabili dei pc fuori casa. Ma può essere utilizzato tranquillamente anche da pc dato che Telegram fornisce il software anche per questi dispositivi.

Architettura del sistema

Bot Telegram

Il nome del bot è Progetto_PDGT_ElviBllano e il suo username è @ElviBllanoBot. Per ricevere le varie informazioni riguardo ad un luogo, il bot mette a disposizione i seguenti comandi:

/start Comando che avvia il bot e invia un messaggio all'utente dove fornisce informazioni sull'obbiettivo del bot, informazioni dello sviluppatore e un link al repository GitHub.

/help Comando che mostra la lista dei comandi che l'utente può usare.

/cercaluogo <luogo> Comando che dato il nome della città inserita mostra all'utente le informazioni di tutti i punti registrati su wheelmap.org e la posizione dei punti.

/cercaluogoy Comando che dato il luogo inserito mostra le informazioni e la posizione di tutti i punti che hanno accesso completo (cioè wheelchair = yes)
/cercaluogol Comando che dato il luogo inserito mostra le informazioni e la posizione di tutti i punti che hanno accesso limitato (cioè wheelchair = limited)
/cercaluogo<numero categoria> <luogo> Comando che dato il nome della città fornisce informazioni sui punti che fanno parte della categoria definita dal numero indicato nel corpo del comando.
Lista delle categorie:
1. Trasporto pubblico
2. Alimenti
3. Tempo libero
4. Banca/Posta
5. Educazione
6. Shopping
7. Sport
8. Turismo
9. Alloggi
10. Vari
11. Governo
12. Salute

Implementazione

Telegram fornisce due funzioni per poter ricevere i messaggi ricevuti dal bot. La prima funzione è il getUpdates che consiste nel richiedere continuamente a Telegram se ci sono messaggi, per fare ciò bisogna creare uno script con un ciclo infinito che invia la richiesta. La seconda funzione invece è il Webhook che, diversamente dal precedente resta in ascolto senza far nulla finché Telegram non ci avvisa del messaggio ricevuto con richiesta POST HTTPS, evitando di creare un ciclo infinito. I vantaggi del secondo sono la leggerezza nel codice e stabilità dovuta dal fatto che se nel primo metodo nel ciclo ci fosse un'eccezione non gestita, il bot si sarebbe bloccato interamente invece nel secondo caso si blocca solo nel caso che crea l'eccezione ma continua a funzionare per gli altri messaggi. Nel mio caso ho deciso di utilizzare il Webhook perché, il servizio di web hosting Altervista che utilizzo per gli script non permette di far girare script che creano cicli infiniti di richieste perché li identifica come possibilmente malevoli e li chiude. Webhook lavora sui singoli messaggi diversamente dal getUpdates che può lavorare con al massimo 100 messaggi alla volta.

Per poter utilizzare le funzioni Telegram dobbiamo usare il token del bot che ci viene fornito alla sua creazione, nel caso di questa piattaforma il token utilizzato è 387262509:AAFfGgvwpm9Wb1GSvIqxDV6HfsMPbZ9GHL0.

Ho deciso di dividere gli script in 3 diversi file che hanno i loro compiti specifici.

Il primo è lo script telegram.php che gestisce il riconoscimento dei comandi inviati dall'utente e il compito che devono svolgere. I comandi e il loro compito sono stati già spiegati precedentemente.

Il secondo è telegramManager.php che lavora sempre su telegram ma gestisce le varie funzioni che non sono collegate direttamente al comportamento di vari comandi. Qui sono contenuti le funzioni che gestiscono l'invio del messaggio di risposta del bot, con le informazioni sui nodi e l'invio delle coordinate gps dei luoghi ricevuti da wheelmap.org.

L'ultimo script e wheelmapManager.php, gestisce tutte le richieste HTTP da inviare ai server di Wheelmap.org per poter ricevere i dati e poi elaborarli. Tutte le richieste per i dati sono gestite da un'unica funzione che in base all'operazione data all'atto di invocazione del metodo seleziona l'indirizzo URL a cui fare la richiesta http. Poi ci sono le funzioni che gestiscono l'ordinamento testuale dei dati da mostrare nel messaggio di risposta all'utente del bot.

Le ultime funzioni dello script che ho creato sono dovute a un problema della lingua di riferimento. I dati che wheelmap.org fornisce con le API per le informazioni, che sono identificati come nodi, sono nella lingua predefinita dal sito che è l'inglese. Quindi le informazioni riguardanti lo stato di accessibilità, la categoria e il tipo di attività ci vengono fornite in inglese, che non è possibile tradurre direttamente alla richiesta http al sito ed avendo creato il bot principalmente per un contesto italiano mi serviva gestire le traduzioni. Nel caso dello stato dell'accessibilità avendo solo 4 stati è bastato una selezione che in base allo stato fornito restituisce la traduzione. Invece negli altri casi non è possibile usare la soluzione precedente perché sono più numerosi e possono esserne aggiunti nel tempo. Quindi

ho deciso di richiedere al sito le traduzioni usando gli id delle categorie e tipi di nodo contenuti nel corpo delle informazioni riguardanti i nodi.

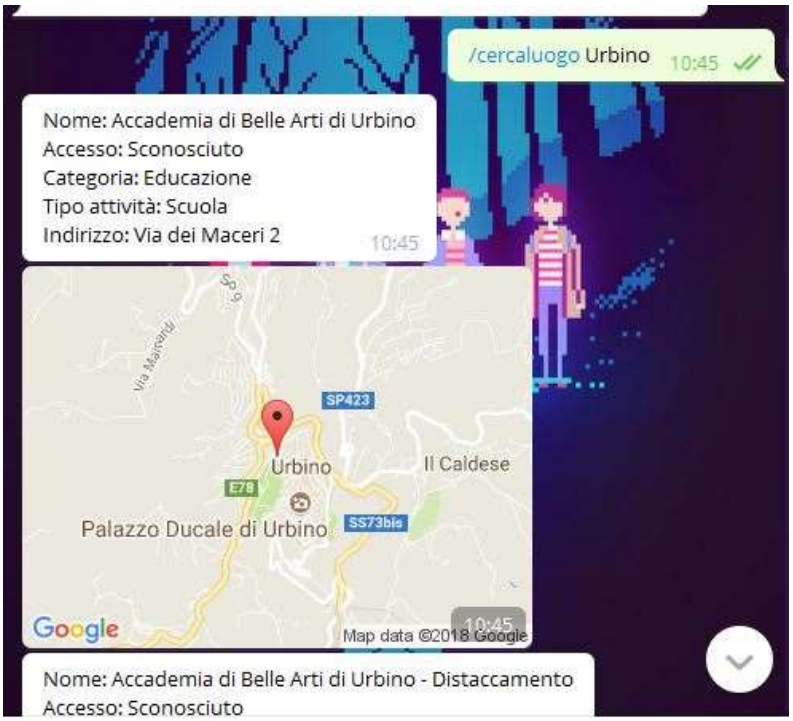
Un'implementazione futura può essere la possibilità di aggiungere o modificare i nodi nella mappa.

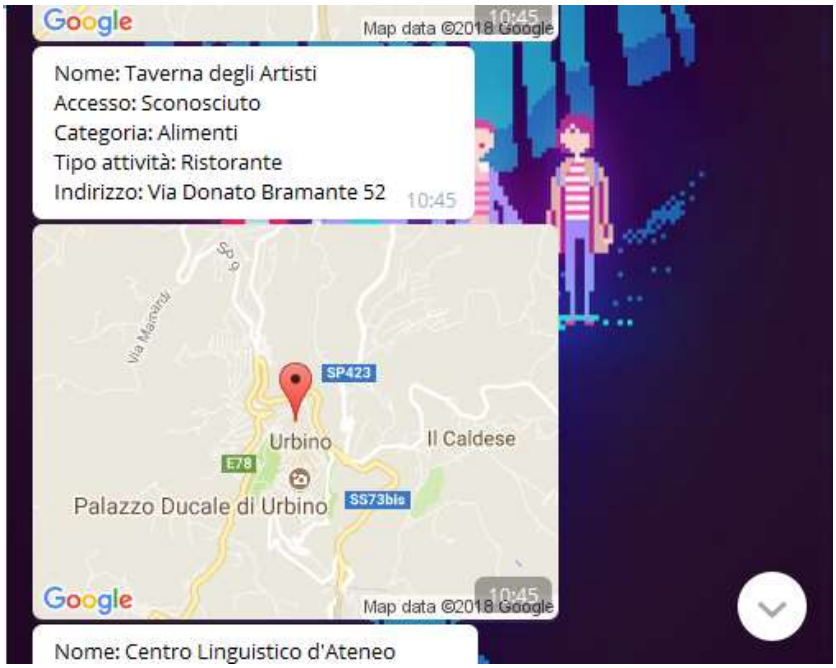
Tracce di conversazioni

Comando /start:

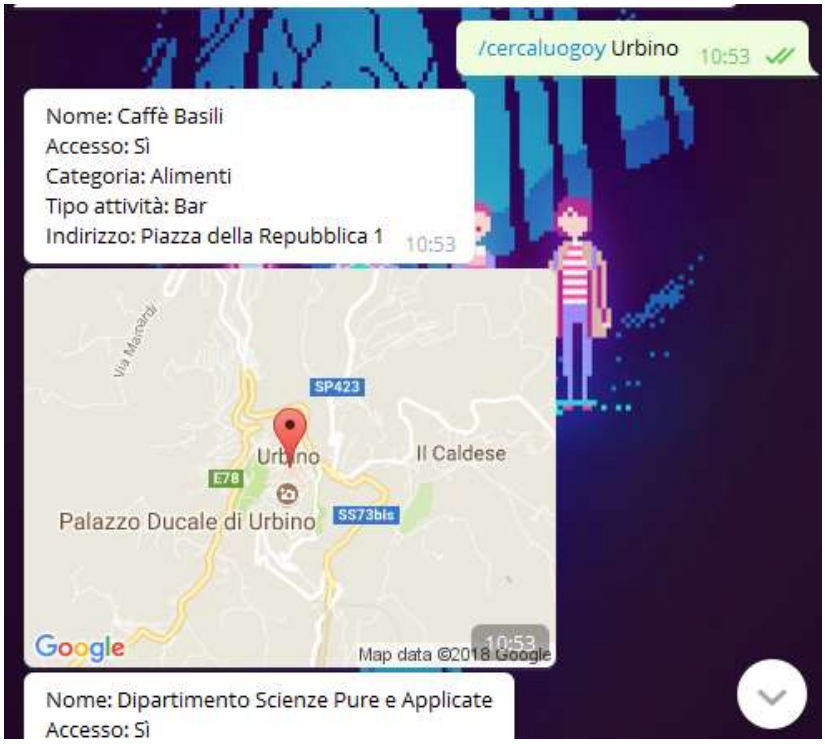
Comando /help:

Comando /cercaluogo Urbino:





/cercaluogoy Urbino



/cercaluogo6 Urbino: