Project work

ITSRunning

Alessandro Rizzi

6 Luglio 2018

Contesto

La corsa al giorno d’oggi è diventato uno degli esercizi preferiti da chi si vuole tenere in forma, infatti le piste ciclabili di tutte le città sono sempre più “invase” da corridori di ogni età.

Ogni runner vorrebbe tenere traccia dei propri allenamenti, così da poterli consultare e vedere i progressi fatti. Questa necessità è ancora più marcata quando tratta di corridori professionisti che partecipano a gare podistiche, e che hanno bisogno di analizzare la propria prestazione sportiva nel modo più dettagliato possibile.

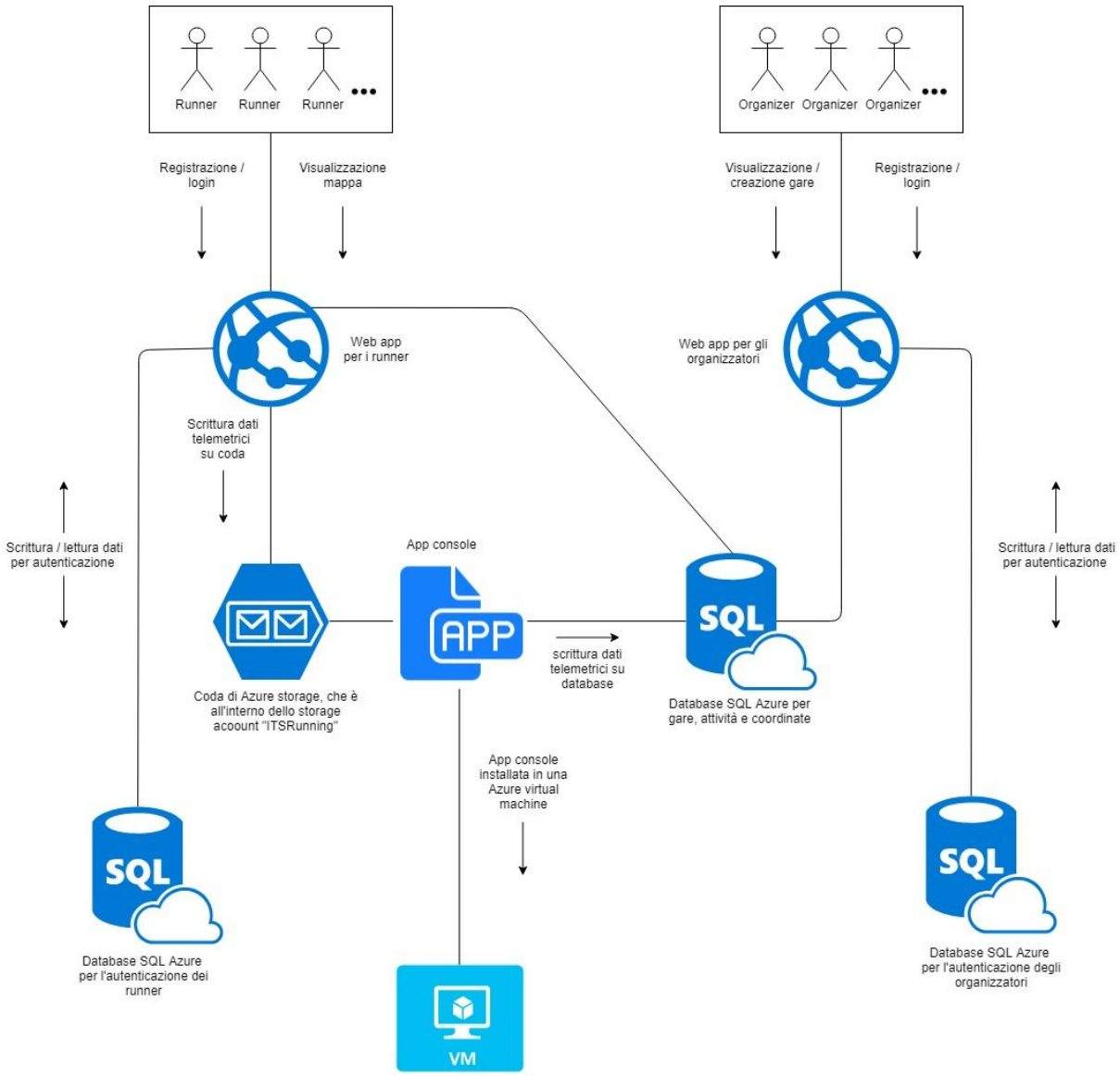
Introduzione

ITSRunning è una applicazione creata per le persone che praticano la corsa a livello sia amatoriale sia agonistico. È in grado di gestire in modo differenziato allenamenti e gare. La sua funzione principale è quella di visualizzare su una mappa il percorso effettuato dal corridore, aggiornando la posizione ogni 2 secondi. Nella base di dati viene salvata la telemetria della posizione ad ogni aggiornamento della posizione stessa.

ITSRunning permette, come anticipato in precedenza, di creare delle gare da parte di utenti organizzatori, oltre che visualizzare ogni gara creata. I runner possono partecipare alle gare semplicemente scegliendo l'opzione "gara", alla creazione dell'attività, e scegliendo la gara desiderata.

Per utilizzare l'applicazione, sia come runner sia come organizzatore, è necessaria la registrazione di un'account ed il login ad ogni accesso.

Schema architetturale



Politiche

Database: il database che ho scelto è **Azure SQL**, in quanto tra le specifiche a cui mi sono attenuto è menzionata la possibilità di fare statistiche, per esempio sui dati telemetrici di ogni runner, dovendo così raggruppare più dati che di norma sarebbero divisi, tutto ciò è possibile solamente con un database relazionale.

Numero di database: per questa applicazione ho deciso di utilizzare 3 database distinti, 2 di essi per l'autenticazione e uno per i dati telemetrici e le attività. Il motivo di questa scelta è soprattutto l'ordine, infatti mantenendo le tabelle dell'autenticazione dei runner e degli organizzatori in un database separato ciascuno, non ho dovuto cambiare nomi o schema alle tabelle, ma ho potuto mantenere quelli predefiniti da Asp.net core risparmiando molto tempo. Ho comunque salvato gli account dei runner e degli organizzatori anche nel terzo database, per averne un migliore controllo.

Coda: l'applicativo web che contiene la mappa ha necessità di salvare nel database una telemetria ogni 2 secondi, ipotizzando che tale applicativo venga utilizzato da centinaia di persone contemporaneamente, il database dovrebbe far fronte a centinaia di richieste di scrittura ogni 2 secondi, per questa ragione è necessario l'utilizzo di una coda su cui salvare i dati telemetrici prima del salvataggio vero e proprio a database. La tipologia di coda che ho scelto è quella dell'**Azure storage account**, in quanto può contenere fino a 80 GB di dati. Con tale capienza è possibile contenere i dati telemetrici di circa 56 000 che inviano un messaggio ogni 2 secondi per 3 ore, senza togliere mai nessun messaggio dalla coda.

Console application: ho scelto una console application come consumer per la coda in quanto, al contrario di altre tecnologie (es. function app), non ha problemi con le versioni dei pacchetti **NuGet**, ovvero non richiede versioni troppo recenti o inesistenti. La pubblicazione su Azure è stata effettuata tramite una **macchina virtuale**, in cui ho installato un Task che fa eseguire automaticamente la console application all'avvio della macchina virtuale.

Scalabilità: i due applicativi web (quello per i runner e quello per gli organizer), sono inseriti in due Azure application service, i quali si fanno automaticamente il deployment su server web fisici. Grazie alle application service è stato possibile abilitare l'**autoscaling**, ovvero la funzione che al crescere degli accessi al servizio fa crescere fa crescere le risorse a disposizione del servizio stesso.

Struttura implementativa

La struttura sotto il profilo tecnico è composta dalle seguenti parti:

- **Web App per i runner**: contiene la funzione principale del progetto, ovvero la mappa in cui il runner visualizza la sua posizione ed il percorso effettuato fino a quel momento. Tale funzione è stata creata usufruendo delle api messe a disposizione da Google Maps, dove tramite javascript è possibile acquisire ogni 2 secondi la posizione del dispositivo, disegnando ogni volta una linea sulla mappa tra gli ultimi due punti in cui il runner è passato così da visualizzare il percorso. Inoltre ad ogni aggiornamento della posizione viene inviato, tramite una chiamata Ajax di tipo POST, un messaggio con le coordinate della posizione e l'istante di tempo in cui viene "catturata", all'url api/Coordinates. Ad ogni chiamata POST l'api delle coordinate (CoordinatesController.cs) inserisce il dato arrivatogli dalla mappa, in formato Json, all'interno di una coda chiamata coordinates-queue. Quando l'utente smette di correre, tocca il tasto "Stop" e viene indirizzato alla pagina di riepilogo della corsa.
- **Coda**: non è creata a priori, viene creata alla prima esecuzione dell'applicazione.
- **Console application**: ha la funzione di leggere i dati dalla coda, trasformarli da Json a stringhe, inserirli nel database ed infine cancellarli dalla coda.
- **Databases autenticazioni**: ho creato due database per l'autenticazione degli utenti, uno per i runner e uno per gli organizzatori.
- **Database centrale**: serve per salvare le attività, le telemetrie e gare, con i rispettivi utenti.