ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INFORMATICA - SCIENZA e INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Analisi, progettazione e distribuzione in cloud di applicativo per l'organizzazione di eventi condivisi

Relatore:
Chiar.mo Prof.
Michele Colajanni

Presentata da:
Giacomo Romanini

Sessione Luglio 2025

Anno Accademico 2025/2026

Abstract

Lo sviluppo di un applicativo multipiattaforma diretto all'organizzazione di eventi condivisi, caratterizzato in particolare dalla condivisione multimediale in tempo reale, richiede opportune capacità di scalabilità, atte a garantire una risposta efficace anche con alti volumi di richieste, offrendo prestazioni ottimali. Le tecnologie cloud, con la loro disponibilità pressoché illimitata di risorse e la completa e continua garanzia di manutenzione, offrono l'architettura ideale per il supporto di simili progetti, anche con fondi limitati.

Tuttavia, l'integrazione tra la logica applicativa ed i molteplici servizi cloud, insieme alla gestione delle loro interazioni reciproche, comporta sfide specifiche, in particolare legate all'ottimizzazione di tutte le risorse. L'individuazione e la selezione delle soluzioni tecnologiche più adatte per ogni obiettivo, così come l'adozione delle migliori pratiche progettuali, devono procedere parallelamente con lo sviluppo del codice, al fine di sfruttare efficacemente le potenzialità offerte.

In tale prospettiva, questa tesi illustra le scelte progettuali ed implementative adottate nello sviluppo dell'applicativo in questione, evidenziando l'impatto dell'integrazione delle risorse cloud sul risultato finale.

Indice

Introduzione

In un contesto sociale sempre più connesso, la crescente quantità di contatti, la rapidità delle comunicazioni e l'accesso universale alle informazioni rendono la ricerca, l'organizzazione e la partecipazione ad eventi estremamente facile, ma al contempo generano un ambiente frenetico e spesso dispersivo.

Risulta infatti difficile seguire tutte le opportunità a cui si potrebbe partecipare, considerando le numerose occasioni che si presentano quotidianamente. Basti pensare, ad esempio, alle riunioni di lavoro, alle serate con amici, agli appuntamenti informali per un caffè, ma anche a eventi più strutturati come fiere, convention aziendali, concerti, partite sportive o mostre di artisti che visitano occasionalmente la città.

Questi eventi possono sovrapporsi, causando dimenticanze o conflitti di pianificazione, con il rischio di delusione o frustrazione. Quando si è invitati a un evento, può capitare di essere già impegnati, o di trovarsi in attesa di una conferma da parte di altri contatti. In questi casi, la gestione degli impegni diventa complessa: spesso si conferma la partecipazione senza considerare possibili sovrapposizioni, o dimenticandosi, per poi dover scegliere e disdire all'ultimo momento.

D'altra parte, anche quando si desidera proporre un evento, la ricerca di un'attività interessante può diventare un compito arduo, con la necessità di consultare numerosi profili social di locali e attività, senza avere inoltre la certezza che gli altri siano disponibili. Tali problemi si acuiscono ulteriormente quando si tratta di organizzare eventi di gruppo, dove bisogna allineare gli impegni di più persone.

In questo contesto, emergono la necessità e l'opportunità di sviluppare uno strumento che semplifichi la proposta e la gestione degli eventi, separando il momento della proposta da quello della conferma di partecipazione. In tal modo, gli utenti possono valutare la disponibilità degli altri prima di impegnarsi definitivamente, facilitando in conteporanea sia l'invito sia la partecipazione.

In risposta a tali richieste è stata creata Wyd, un'applicazione che permette agli utenti di organizzare i propri impegni, siano essi confermati oppure proposti. Essa permette anche di rendere più intuitiva la ricerca di eventi attraverso la creazione di uno spazio virtuale centralizzato dove gli utenti possano pubblicare e consultare tutti gli eventi disponibili, diminuendo l'eventualità di perderne qualcuno. La funzionalità chiave di questo progetto si fonda sull'idea di affiancare alla tradizionale agenda degli impegni confermati un calendario separato, che mostri tutti gli eventi a cui si potrebbe partecipare.

Una volta confermata la partecipazione a un evento, questo verrà spostato automaticamente nell'agenda personale dell'utente. Gli eventi creati potranno essere condivisi con persone o gruppi, permettendo di visualizzare le conferme di partecipazione. Considerando l'importanza della condivisione di contenuti multimediali, questo progetto prevede la possibilità di condividere foto e video con tutti i partecipanti all'evento, attraverso la generazione di link per applicazioni esterne o grazie all'ausilio di gruppi di profili. Al termine dell'evento, l'applicazione carica automaticamente le foto scattate durante l'evento, per allegarle a seguito della conferma dell'utente.

Figura 1: Il logo di Wyd

La realizzazione di un progetto come Wyd implica la risoluzione e la gestione di diverse problematiche tecniche. In primo luogo, la stabilità del programma deve essere garantita da un'infrastruttura affidabile e scalabile. La persistenza deve essere modellata per fornire alte prestazioni sia in lettura che in scrittura indipendentemente dalla quantità delle richieste, rimanendo però aggiornata e coerente. La funzionalità di condivisione degli eventi richiede inoltre l'aggiornamento in tempo reale verso tutti gli utenti coinvolti. Infine, il caricamento ed il salvataggio delle foto aggiungono la necessità di gestire richieste di archiviazione di dimensioni significative.

Organizzazione dei capitoli

Il seguente elaborato è suddiviso in cinque capitoli.
Nel primo capitolo si affronta la fase di analisi delle funzionalità, durante la quale, partendo dall'idea astratta iniziale, si definiscono i requisiti e le necessità del sistema, per poi creare la struttura generale ad alto livello dell'applicazione.
Nel secondo capitolo si affrontano le principali scelte architetturali e di sviluppo che hanno portato a definire la struttura centrale dell'applicazione.
Il terzo capitolo osserva lo studio effettuato per gestire la memoria, in quanto fattore che più incide sulle prestazioni. Particolare attenzione è stata dedicata, infatti, a determinare le tecnologie e i metodi che meglio corrispondono alle esigenze derivate dal salvataggio e dall'interazione logica degli elementi.
Il quarto capitolo si concentra sulle scelte implementative adottate per l'inserimento le funzionalità legate alla gestione delle immagini, che, oltre ad introdurre problematiche impattanti sia sulle dimensioni delle richieste sia sull'integrazione con la persistenza, richiedono l'automatizzazione del recupero delle immagini.
Infine, nel quinto capitolo, verranno analizzati e discussi i risultati ottenuti testando il sistema.

0.1 Costo previsto del sistema

Sebbene lo scopo di questa tesi sia di evidenziare l'effetto delle scelte implementative sulla scalabilità del sistema, l'aspetto economico non può essere ignorato durante la realizzazione di un progetto. Si cerca quindi di prevedere la spesa effettiva delle risorse utilizzate in base a un utilizzo possibile, soffermandosi maggiormente sulla dinamica di definizione dei parametri richiesti rispetto all'effettiva necessità, che può variare in base al mercato dell'applicazione o alla sua maturità.

Essendo l'applicazione in fase prototipale, si può solamente provare a dedurre l'effettivo utilizzo delle risorse proporzionalmente al carico previsto. Per il calcolo del costo del sistema si è deciso di considerare un carico di cinquemila utenti attivi giornalmente, che indica un utilizzo consistente e maturo dell'applicazione.

Per stimare il costo del sistema viene usato uno strumento offerto da Azure chiamato Pricing Simulator. Questo strumento, accessibile online, permette di calcolare il costo dell'utilizzo delle risorse in base all'utilizzo che si prevede di farne. Per ogni servizio, in base alle sue peculiarità, viene chiesto l'inserimento di dati specifici per il suo utilizzo. Per alcuni servizi, in cui era stato adottato il piano di utilizzo più economico, è stato necessario assumere l'adozione di un piano che permetta effettivamente di gestire il carico previsto. I costi vengono calcolati mensilmente.

Azure Static Web App prevede due tipologie di piani, gratuito o standard. Il piano gratuito fornisce 100 gigabyte di bandwidth e un massimo di 0.5 gigabyte di memoria. La dimensione attuale dell'applicazione risulta di circa 20 megabyte, che viene però compressa dal sistema. Il limite della memoria non risulta quindi un problema, ma bisogna verificare che non si superi la quantità fornita di bandwidth.

La dimensione dei dati ricevuti dal browser in fase di caricamento del sito ammonta a un massimo di 10 megabyte. Questa quantità si riduce notevolmente se il sito è già stato caricato in cache, a un massimo stimato di 250 kilobyte. Considerando che il metodo di fruizione principale del progetto avvenga tramite applicazione, si stimano 2500 utenti mensili da browser. Stimando un utilizzo quotidiano dell'applicazione, il primo giorno

sarà necessario scaricare l'intero sito, mentre le volte successive solo aggiornare la cache, che porta il calcolo della quantità di dati effettivamente trasferita al seguente:
2500 x (10 Mb + 250 Kb x 29) = 43 Gb.
I quarantatrè gigabyte così previsti rientrano ampiamente nei cento offerti, permettendo l'utilizzo del piano gratuito.

Per stimare il costo delle Azure Functions è necessario calcolare le interazioni totali mensili con gli utenti. Queste vengono previste per 200 richieste giornaliere, che includono sia richieste in scrittura che in lettura. Si calcolano così un milione di invocazioni giornaliere, per un totale di trenta milioni di chiamate mensili.

Dai test si evince una durata media delle funzioni che si aggira sui 200 millisecondi. Per stare sul sicuro, la durata prevista per richiesta viene arrotondata a 250 millisecondi. Allo stesso modo, si prevede un utilizzo medio di 0.5 gigabyte di memoria per ogni invocazione. Nel calcolo bisogna considerare le risorse gratuite e un costo di €0.000015 per GB/s e €0.117 per milione di richieste, portando la spesa totale prevista attorno ai € 60.

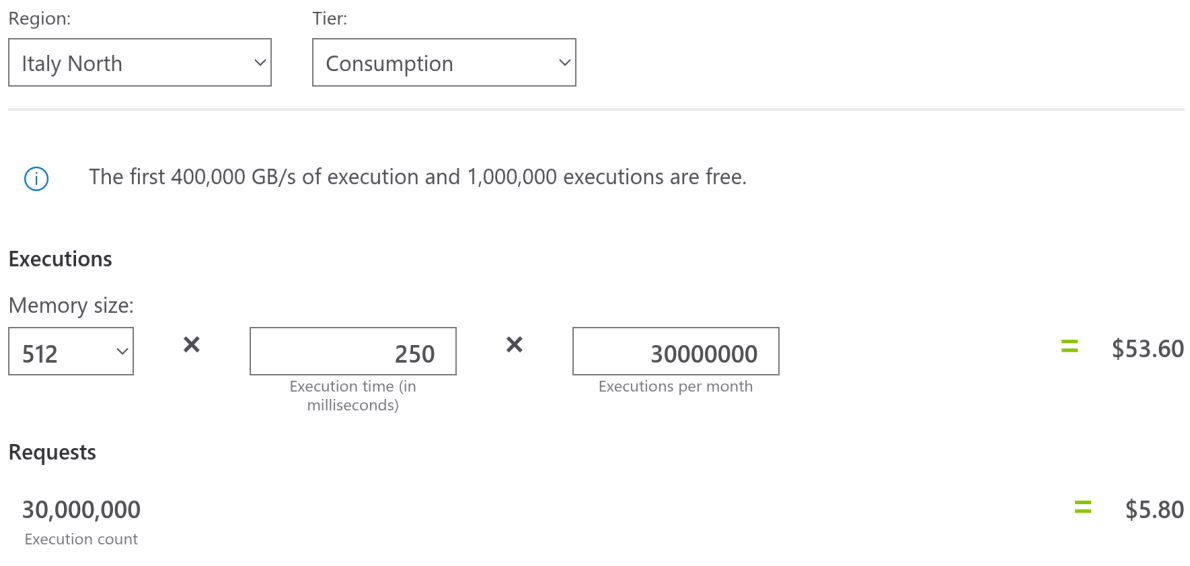

Figura 2: Calcolo per il costo delle Azure Functions

Il costo del database, implementato con Azure Cosmos DB, varia in base alle Request Units al secondo(RU/s) usate. Ogni richiesta consuma una certa quantità di RU, in base

al carico computazionale necessario. Bisogna quindi stimare le richieste al secondo, e il loro consumo di Request Unit. Considerando che la quasi totalità delle richieste verso le Azure Functions implicano un accesso al database, le invocazioni per secondo possono essere calcolate dividendo le richieste giornaliere per i secondi di una giornata, facendo così risultare una media di 12 richieste al secondo.

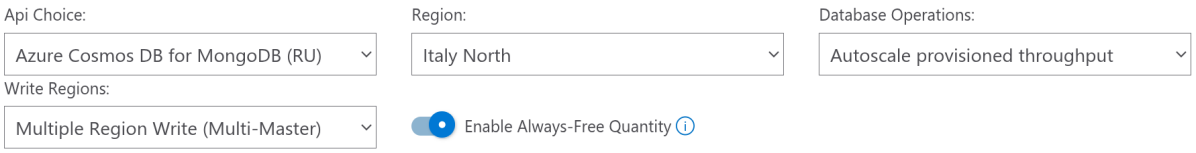

Figura 3: Impostazioni del server Cosmos

Il consumo medio di RU registrato durante i test varia tra i 10 e i 20 RU per richiesta. Per sicurezza, si considera il consumo medio di un'operazione sul database di 30 RU. Risulta così una media di 360 RU/s, che non fornisce però informazioni sul picco del carico. Utilizzando un piano di tipologia autoscale, che modifica in automatico le risorse del database in base al carico effettivo, bisogna stabilire il carico massimo che si prevede il database debba sostenere, tenendo però presente che la quantità minima di risorse sempre stanziate sarà del suo 10%.

Non essendo in possesso di dati relativi all'effettivo utilizzo dell'applicazione, Stimiamo un utilizzo massimo pari a 5 volte quello previsto, pari a 60 richieste al secondo. Questo porta a un massimo di 1800 RU/s, con una media prevista del 20%. Sottraendo i primi 1000 RU/s e moltiplicando i restanti per il costo orario di € 0.016, il costo totale previsto per l'utilizzo di Azure Cosmos si aggira intorno ai € 20.

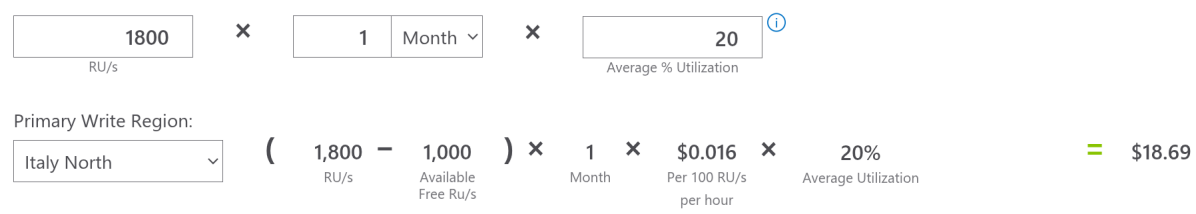

Figura 4: Calcolo per il costo di Cosmos

INDICE

Key vault 1 richiesta ogni 10 operazioni

storage container 30 immagini al giorno, 5000 * 4mb = 600 gb necessario comprimere l'immagine a 200 kb
PubSub 1 utenza, messaggi da 1kb, si considerano metà delle richieste come richieste in scrittura, ma vengono inviati solo agli utenti presenti e quindi 15.000.000 * 1Kb * 10 canali broadcast = 30 GB.
Authentication Gratis per i primi 50000 di utenti attivi mensilie.
Queue al mese 30.000.000 operazioni di lettura 30.000.000 opeazioni di scrittura.
capacità totale di 50 Gb, sufficienti perché, una volta letto il messaggio, questo viene rimosso dalla coda.
servicebus -confirm/update event 300.000 giornaliere = 9 milioni mensili
Virtual network free of charge
riconsiderare PubSub. compressione immagini. totale.

Conclusioni

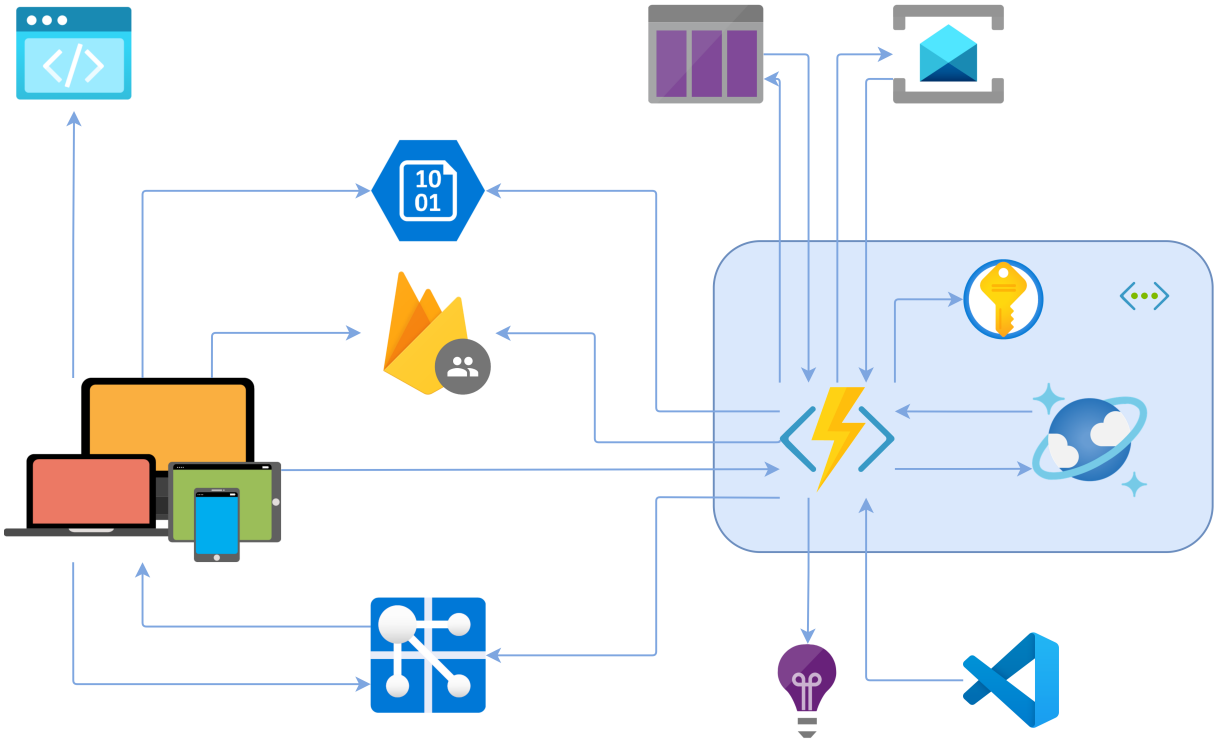

Figura 5: Grafico dell'architettura di WYD

0.2 Sviluppi futuri

Gli sviluppi futuri potranno comprendere, in base a decisioni di marketing:

- La visualizzazione degli impegni degli altri profili
- L'implementazione di una chat per ogni gruppo
- Sviluppo di strumenti utili all'organizzazione dei gruppi, quali:
  - form per combinare le disponibilità reciproche
  - appunti condivisi(liste della spesa o note su chi porta cosa)
  - calcolo delle spese compiute da ciascun componente
- La creazione di profili pubblici che possono essere seguiti
- La creazione di eventi pubblici
- Una funzionalità di ricerca degli eventi o dei profili pubblici
- Supporto alla gestione di prenotazione e organizzazione degli eventi, dalle liste di attesa alla vendita dei biglietti
- La possibilità per le aziende di gestire in locale il proprio server e i relativi dati

Fonti bibliografiche e sitografia

Object Management Group, OMG Unified Modelling Language Version 2.5.1, December 2017, https://www.omg.org/spec/UML/2.5.1/PDF

https://learn.microsoft.com/en-us/azure/reliability/reliability-cosmos-db-nosql https://learn.microsoft.gb/azure/cosmos-db/throughput-serverless https://learn.microsoft.com/en-gb/azure/cosmos-db/provision-throughput-autoscale