



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 multiplatforma 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	1
Organizzazione dei capitoli	3
0.1 Costo previsto del sistema	4
Conclusione	10
0.2 Sviluppi futuri	12
Fonti bibliografiche e sitografia	14

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 contemporanea 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 architettureali 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 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

INDICE

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:

$$2500 \times (10 \text{ Mb} + 250 \text{ Kb} \times 29) = 43 \text{ Gb.}$$

I quarantatré 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.

The screenshot shows the Azure Functions pricing calculator. At the top, there are two dropdown menus: 'Region' set to 'Italy North' and 'Tier' set to 'Consumption'. Below these, a note states: 'The first 400,000 GB/s of execution and 1,000,000 executions are free.' The calculator is divided into two sections: 'Executions' and 'Requests'. In the 'Executions' section, there are three input fields: 'Memory size' (384 MB), 'Execution time (in milliseconds)' (250 ms), and 'Executions per month' (30,000,000). These are multiplied together to show a total cost of €34.15. In the 'Requests' section, there is a single input field for 'Execution count' (30,000,000), which results in a total cost of €5.13.

Category	Parameter	Value	Unit
Executions	Memory size	384	MB
	Execution time (in milliseconds)	250	ms
	Executions per month	30,000,000	executions
Requests	Execution count	30,000,000	requests

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

INDICE

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.

The screenshot shows the Azure Cosmos DB configuration interface. It includes four main sections: 'Api Choice' with a dropdown set to 'Azure Cosmos DB for MongoDB (RU)', 'Region' with a dropdown set to 'Italy North', 'Database Operations' with a dropdown set to 'Autoscale provisioned throughput', and 'Write Regions' with a dropdown set to 'Multiple Region Write (Multi-Master)'. Below the 'Write Regions' dropdown is a toggle switch labeled 'Enable Always-Free Quantity' which is currently turned on.

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 € 0,014, il costo totale previsto per l'utilizzo di Azure Cosmos si aggira intorno ai € 17.

The screenshot shows the Azure Cosmos DB cost calculator interface. It displays a calculation for the total cost of the database. The calculation is as follows: 1800 RU/s (Maximum RU/s) multiplied by 1 Month (Duration) multiplied by 20 (Average % Utilization) equals €16.54 (Total Cost). Below the calculation, the 'Primary Write Region' is set to 'Italy North'. The calculation also shows the 'Available Free Ru/s' of 1,000 and the 'Per 100 RU/s per hour' cost of €0.014.

Figura 4: Calcolo per il costo di Cosmos

INDICE

Per poter garantire la gestione totale dell'esecuzione dei messaggi è necessario impostare il piano Standard di Service Bus. Prevede un costo di base orario di € 0,012, più € 0,71 per ogni milione di messaggi, includendo però nel prezzo i primi 13 milioni. Stimando le operazioni in scrittura (che sono quelle che richiedono la propagazione) a un terzo di quelle totali, si prevedono 9 milioni di messaggi mensili, riducendo la spesa al solo costo della macchina, che ammonta così a circa € 9.

Le operazioni di aggiornamento sono le stesse che richiedono l'invio delle notifiche, scatenate tramite messaggio inviato su una Azure Storage Queue. Ogni richiesta comporta una scrittura sulla coda, che verrà presa in carico da una funzione, con un'operazione di lettura. Questo comporta 9 milioni di operazioni in lettura e altrettante in scrittura. A un costo di € 0,0035 ogni 10.000 operazioni, comporta una spesa di € 6,40, a cui si aggiunge € 1 per la capacità della coda che, per sicurezza, è stata stimata a 25 gigabyte. Questa quota difficilmente viene raggiunta in quanto un messaggio, dopo essere stato letto, dovrebbe essere eliminato.

The image shows a pricing calculator for Azure Storage Queue. At the top, there are four dropdown menus for configuration: 'Region' set to 'Italy North', 'Type' set to 'Queue Storage', 'Storage Account Type' set to 'General Purpose V2', and 'Redundancy' set to 'LRS'. Below these, the 'Capacity' section shows a value of '25' GB, which is multiplied by a unit price of '€0.0398 Per GB' to result in a total of '€0.99'. The 'Queue Class 1 operations' section shows a value of '1800' operations (in 10,000s), which is multiplied by a unit price of '€0.0035 Per 10,000 operations' to result in a total of '€6.37'.

Configuration	Value	Unit Price	Total Cost
Capacity	25 GB	€0.0398 Per GB	€0.99
Queue Class 1 operations	1800 Operations (in 10,000s)	€0.0035 Per 10,000 operations	€6.37

Figura 5: Calcolo per il costo di Azure Storage Queue

Per l'invio delle notifiche si utilizza Azure PubSub. La versione gratuita non è più sufficiente, potendo garantire un massimo di 20 connessioni contemporanee. Si stima però che, modificando il contratto a Standard, una istanza sia sufficiente, la quale, con un costo orario di € 0,0594, ammonta da sola a € 44 mensili. Stimando la dimensione media dei messaggi delle notifiche a 1 Kb, e prevedendo una media di 10 utenti attivi per ogni notifica, la dimensione totale dei messaggi così inviati con PubSub dovrebbe ammontare attorno a $9.000.000 \times 1 \text{ Kb} \times 10 = 90 \text{ Gb}$. Il costo per l'invio dei messaggi risulterà così

INDICE

pari a € 13, per un totale di circa € 56 mensili.

Per il calcolo del costo di Azure Blob Storage, che viene usato per salvare le immagini, viene stimato un carico di un'immagine al giorno per utente, o di 30 immagini al mese. Azure Blob Storage prevede un costo in base alla dimensione totale dei dati salvati, a cui si aggiunge il costo computazionale necessario per aggiungere o trovare le immagini. Considerata una dimensione media per immagine (compressa) di 200 Kb, ogni mese vengono aggiunti $30 \times 5000 \times 200 \text{ Kb} = 30 \text{ Gb}$. Dopo un anno di utilizzo stimiamo quindi una dimensione totale di 360 Gb, per una spesa mensile di € 6.

Si stima inoltre che un utente guardi giornalmente 3 eventi nel dettaglio, i quali sono mediamente condivisi con 10 utenti. Considerando il carico di una immagine al giorno per utente, ogni utente esegue, al giorno, 3 richieste di elenco e 30 di lettura. Si calcolano così 150.000 richieste di scrittura, 450.000 richieste di elenco, e 4.500.000 richieste di lettura. Con un prezzo di € 0,047 ogni 10.000 invocazioni per i primi due casi, e di € 0,004 ogni 10.000 invocazioni per le letture, il costo computazione ammonta a € 4,50. La spesa totale per il servizio viene quindi stimata a € 10,5.

Per il Key Vault bisogna stimare la frequenza con cui le Functions necessitano di recuperare i segreti per accedere agli altri servizi. Si stima che, come minimo, ogni richiesta abbia bisogno di almeno un segreto, ad esempio per accedere al database. Le Azure Functions sono però strutturate per riutilizzare l'ambiente di esecuzione, se le risorse computazionali lo permettono. Questo consente di riutilizzare i segreti recuperati dalle funzioni invocate precedentemente. Si stima quindi che il Key Vault venga interrogato una volta ogni dieci richieste, per un totale di 3 milioni al mese. Per un costo di € 0,027 ogni 10.000 invocazioni, la spesa totale di Azure Key Vault viene prevista per circa € 8.

L'ultima risorsa utilizzata all'interno dell'ecosistema Azure è Virtual Network, il cui utilizzo è però gratuito. Infine, il servizio per l'autenticazione, Firebase Authentication, risulta anch'esso gratuito, non superando la soglia dei 50.0000 utenti attivi mensili.

























Static Web Apps		Free tier	 	Upfront: €0.00	Monthly: €0.00
Azure Functions		Consumption tier, Pay as yo...	 	Upfront: €0.00	Monthly: €39.29
Azure Cosmos DB		Azure Cosmos DB for Mong...	 	Upfront: €0.00	Monthly: €16.54
Service Bus		Standard tier: Messaging Op...	 	Upfront: €0.00	Monthly: €8.68
Storage Accounts		Queue Storage, General Pur...	 	Upfront: €0.00	Monthly: €7.37
Azure Web PubSub		Standard, 1 Unit(s) x 1 Hours...	 	Upfront: €0.00	Monthly: €56.23
Storage Accounts		Block Blob Storage, General ...	 	Upfront: €0.00	Monthly: €10.58
Key Vault		Vault: 3,000,000 operations, ...	 	Upfront: €0.00	Monthly: €7.96

Figura 6: Riassunto dei costi previsti delle risorse Azure

Il costo totale previsto per fornire il servizio a 5000 utenti ammonta quindi a circa € 150. Viste tutte le stime assunte sull'utilizzo del sistema questa cifra è da considerarsi indicativa. Fornisce però una dimensione iniziale sul costo da prevedere per l'esecuzione del sistema. Come appena presentato, il prezzo può infatti essere correlato al numero di utenti attivi, tenendo però presente la scalabilità specifica di ogni servizio.

L'analisi di questo calcolo evidenzia quanto sia significativo l'impatto di Azure PubSub sul totale. Visto il ruolo nel sistema, il suo utilizzo risulta sicuramente degno di revisione, per trovare soluzioni alternative che possano fornire un servizio di pari qualità, a un prezzo minore.

Conclusione

L'obiettivo di questa tesi era mostrare a che livello e in che misura il requisito della scalabilità e l'utilizzo delle risorse in cloud impattano sulla realizzazione di un'applicazione.

Durante la fase di analisi è stato evidenziato quali fossero i punti critici del progetto. Si sono quindi affiancate a ogni componente scelto le necessità a cui dovevano rispondere, eventualmente approfondendo le molteplici offerte presenti sul mercato. La scelta è stata determinata dalle tecnologie e funzionalità che presentavano, influenzando in maniera decisiva l'approccio da usare nella progettazione del codice. Sono state quindi affiancate le scelte implementative implicate dall'utilizzo dei vari servizi.

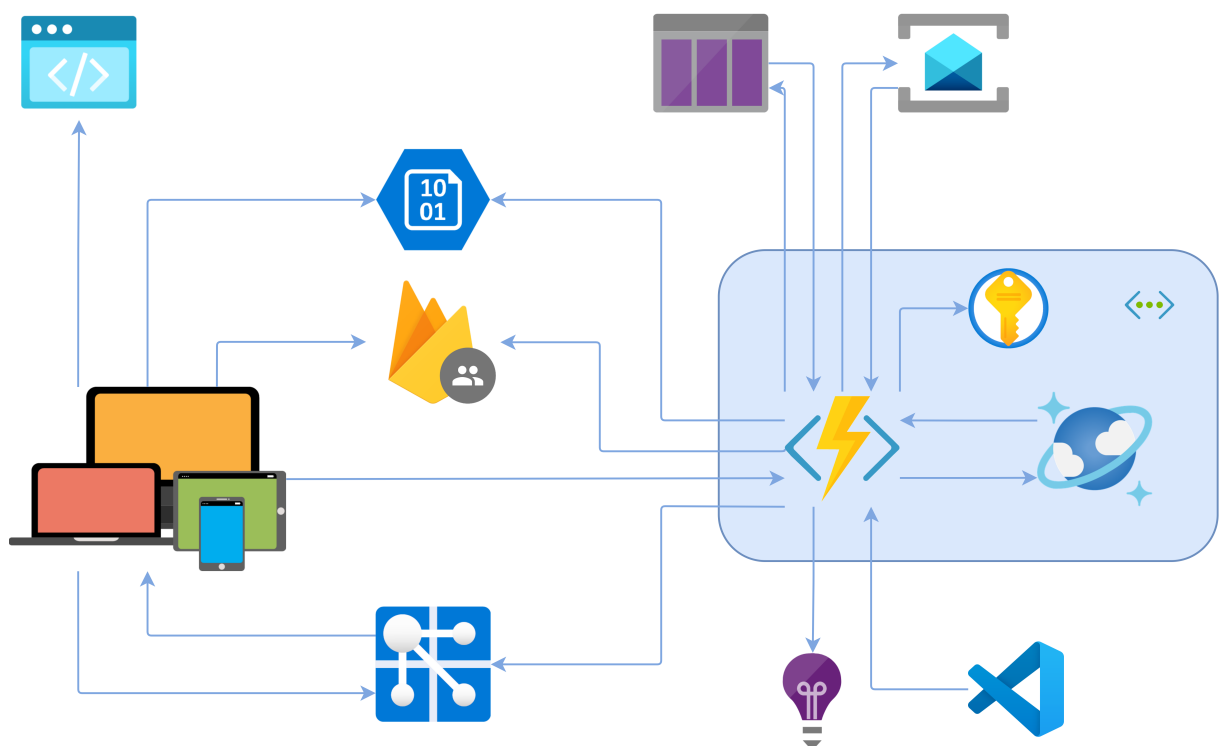


Figura 7: Grafico dell'architettura di WYD

La fase di realizzazione ha visto lo sviluppo di un client utente intuitivo e di un server scalabile ed efficiente, sfruttando la tecnologia serverless. Una particolare attenzione è stata prestata ai meccanismi di autenticazione, alla sicurezza dei dati e al monitoraggio continuo dei servizi. La gestione della persistenza è stata affrontata attraverso l'implementazione di un database non relazionale distribuito, scelta che ha comportato il principale impatto sulle dinamiche del progetto. Gli è stato inoltre affiancato lo sviluppo di una memoria locale per la cache e la comunicazione in tempo reale, elementi cruciali per la reattività dell'applicativo. Un capitolo significativo è stato dedicato al recupero e al salvataggio dei dati multimediali, essenziali per la funzionalità di condivisione in tempo reale di Wyd.

L'integrazione tra le tecnologie cloud ha richiesto un approccio progettuale orientato alla separazione delle responsabilità, alla gestione della consistenza eventuale e alla resilienza in caso di errori o richieste concorrenti. Ciò ha condotto all'adozione di pattern moderni, come la denormalizzazione, i trigger su modifiche, l'uso di code per la propagazione degli eventi e notifiche in tempo reale attraverso canali dedicati.

I risultati ottenuti durante i test di carico hanno confermato la qualità delle scelte architetturali adottate: il sistema dimostra di sostenere centinaia di richieste al secondo senza degrado percettibile delle prestazioni, mantenendo al tempo stesso un basso tempo di latenza.

In definitiva, il percorso intrapreso ha mostrato come la realizzazione di un'applicazione distribuita e scalabile non sia il frutto di un processo lineare, ma di un dialogo costante tra le esigenze funzionali, i vincoli tecnologici e le opportunità offerte dai servizi cloud-native. Ogni tecnologia adottata ha influito sulle logiche implementative, imponendo precise scelte progettuali, sia in termini di prestazioni che di costi.

L'auspicio è che le considerazioni sviluppate possano offrire un contributo utile non solo alla valutazione di Wyd come progetto, ma anche come punto di partenza per l'approfondimento dei principi e delle pratiche legate allo sviluppo di applicazioni moderne, condivise e scalabili.

0.2 Sviluppi futuri

Wyd può essere ampliata nelle sue funzionalità, con conseguenze sull'infrastruttura che possono essere nulle o molto impattanti.

L'architettura attuale guadagnerebbe sicuramente in prestazioni dall'introduzione di una cache tra le Azure Functions e il database. È inoltre bene rivedere l'utilizzo delle websockets per le informazioni in tempo reale, considerato l'impatto previsto di Azure PubSub sui costi totali.

L'implementazione di eventi pubblici è sicuramente il principale passo logico successivo per rendere Wyd un'applicazione di ampio utilizzo. Questo comporta la realizzazione di una piattaforma dedicata, che riceva gli eventi e ne gestisca la condivisione, tenendo traccia di tutti i rapporti tra profili che seguono e profili che possono pubblicare.

Con la creazione di eventi pubblici ha senso prevedere un sistema per gestire la prenotazione e l'organizzazione degli eventi, dalle liste di attesa alle vendite dei biglietti. La necessità di garantire la robustezza del servizio, introducendo logiche di controllo dei ruoli e di traffico monetario, comporta requisiti specifici di affidabilità e consistenza a cui l'infrastruttura dovrà rispondere.

Vista la crescente attenzione alla privacy, alla protezione dei dati e alla dipendenza dovuta all'utilizzo di servizi esterni, si prevede di sviluppare un applicativo gemello che possa essere distribuito su un unico server dedicato, per fornire la possibilità a terzi di eseguire su macchine proprie. Il programma vedrebbe quindi un'infrastruttura fisica completamente diversa, ma la sua implementazione continuerebbe a seguire tutti i principi adottati per mantenere il codice scalabile, fornendo così un prodotto comunque resistente a carichi differenti, seppure ridotti. Questo diminuirebbe inoltre il numero di modifiche da introdurre nel codice.

Infine, ulteriori sviluppi potranno comprendere, in base alle necessità degli utenti:

- La visualizzazione degli impegni degli altri profili
- L'implementazione di una chat per ogni gruppo
- 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
- Una funzionalità di ricerca degli eventi o dei profili pubblici

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.com/en-gb/azure/cosmos-db/throughput-serverless> <https://learn.microsoft.com/en-gb/azure/cosmos-db/provision-throughput-autoscale> <https://learn.microsoft.com/en-us/azure/azure-functions/functions-dotnet-dependency-injection> <https://azure.microsoft.com/en-us/pricing/calculator/>