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 e i molteplici servizi cloud, assieme 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 funzionalità, 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 e implementative adottate nello sviluppo di Wyd, applicativo per la condivisione di eventi, 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 a 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 che la partecipazione.

In risposta a tali esigenze si è deciso di creare 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 e il salvataggio delle foto aggiungono la necessità di gestire richieste di archiviazione di dimensioni significative.

Descrizione dei capitoli

L'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 del progetto.
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 delle funzionalità legate alla gestione delle immagini che, oltre a introdurre problematiche impattanti sia sulle dimensioni delle richieste che 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.

Capitolo 4

I file multimediali rappresentano un elemento centrale in un'applicazione orientata alla condivisione sociale, contribuendo significativamente all'interazione tra i partecipanti e all'esperienza utente. La possibilità di acquisire e condividere contenuti visivi, come immagini e video, consente di documentare eventi e attività, favorendo una memoria collettiva e rafforzando il legame tra gli utenti. In particolare, l'integrazione di materiale multimediale associato a eventi condivisi permette di preservare una rappresentazione più completa e dettagliata dell'esperienza vissuta, migliorando la partecipazione sulla piattaforma, rappresentando uno dei punti di forza dell'applicazione.

Tuttavia, la gestione dei file multimediali introduce complessità operative sia per il sistema che per gli utenti stessi. La selezione e l'invio dei contenuti da parte dell'utente possono rappresentare un onere significativo, aumentando la difficoltà associata all'utilizzo dell'applicazione. Per ottimizzare il processo e migliorare l'usabilità, è essenziale ridurre al massimo l'interazione richiesta, automatizzando il recupero dei dati e limitando il ruolo dell'utente alla sola conferma dei contenuti selezionati. Questo approccio non solo semplifica l'esperienza d'uso, ma la rende più intuitiva e fruibile, contribuendo anche a un incremento del tasso di adozione e della frequenza di utilizzo dell'applicazione.

Parallelamente, la memorizzazione e il trasferimento di file multimediali pongono sfide significative a livello infrastrutturale. Queste entità comportano un impatto rilevante sulle risorse computazionali e sulla gestione dello storage. Il volume elevato di richieste di caricamento e accesso ai file può compromettere le prestazioni del sistema. I ritardi introdotti dal traffico di tali dati potrebbero influire negativamente sulle altre operazioni dell'applicazione. Per garantire un'archiviazione efficiente e scalabile è quindi necessa-

rio implementare una strategia di gestione della memoria che separi il salvataggio dei file multimediali dal database principale, evitando di sovraccaricare il server applicativo. Questa soluzione deve inoltre garantire un aggiornamento tempestivo delle informazioni, assicurando la sincronizzazione tra i dati archiviati e le modifiche effettuate dagli utenti.

Per affrontare tali problematiche, un primo esame osserva le modalità di recupero dei file multimediali, con un focus sulle tecniche di selezione e rilevamento automatico delle immagini, nonché sui vincoli normativi e di sicurezza che ne regolano l'utilizzo. L'analisi si concentrerà poi sulle strategie di salvataggio e gestione dello storage, analizzando le diverse tipologie di archiviazione disponibili e le soluzioni implementate per garantire scalabilità, efficienza e riduzione dell'impatto sulle prestazioni del sistema.

1.1 Modalità di recupero delle immagini

L'aggiunta di immagini a un evento prevede una fase preliminare di recupero che consente all'utente di selezionare i file multimediali da associare. Il sistema offre due modalità principali di acquisizione: la selezione manuale da parte dell'utente, che può scegliere le immagini direttamente dalla memoria del dispositivo, o un meccanismo automatizzato, che - sui dispositivi la cui tecnologia lo permette - identifica le foto scattate durante lo svolgimento dell'evento.

L'implementazione di questa funzionalità automatica di controllo della galleria per individuare i file multimediali desiderati richiede l'accesso alla galleria fotografica del dispositivo, un'operazione subordinata al consenso esplicito dell'utente. Per questo motivo, al primo avvio dell'applicazione, il sistema richiede l'autorizzazione per accedere ai file multimediali, unitamente al permesso per la gestione delle notifiche. Nel caso in cui l'utente neghi l'accesso, la richiesta verrà riproposta ogni qualvolta il sistema rilevi la necessità di accedere alla galleria per il recupero delle immagini.

Al termine di ogni evento, non appena possibile, l'applicazione avvia automaticamente un'analisi della galleria locale, individuando le immagini scattate durante tutta la durata

dell'evento. Se il sistema rileva la presenza di contenuti pertinenti ne memorizza temporaneamente i riferimenti in una memoria locale, per poi inviare una notifica all'utente informandolo del ritrovamento delle immagini. A questo punto, l'utente ha la possibilità di esaminare le immagini suggerite, escluderne alcune o confermare l'intero set per il caricamento.

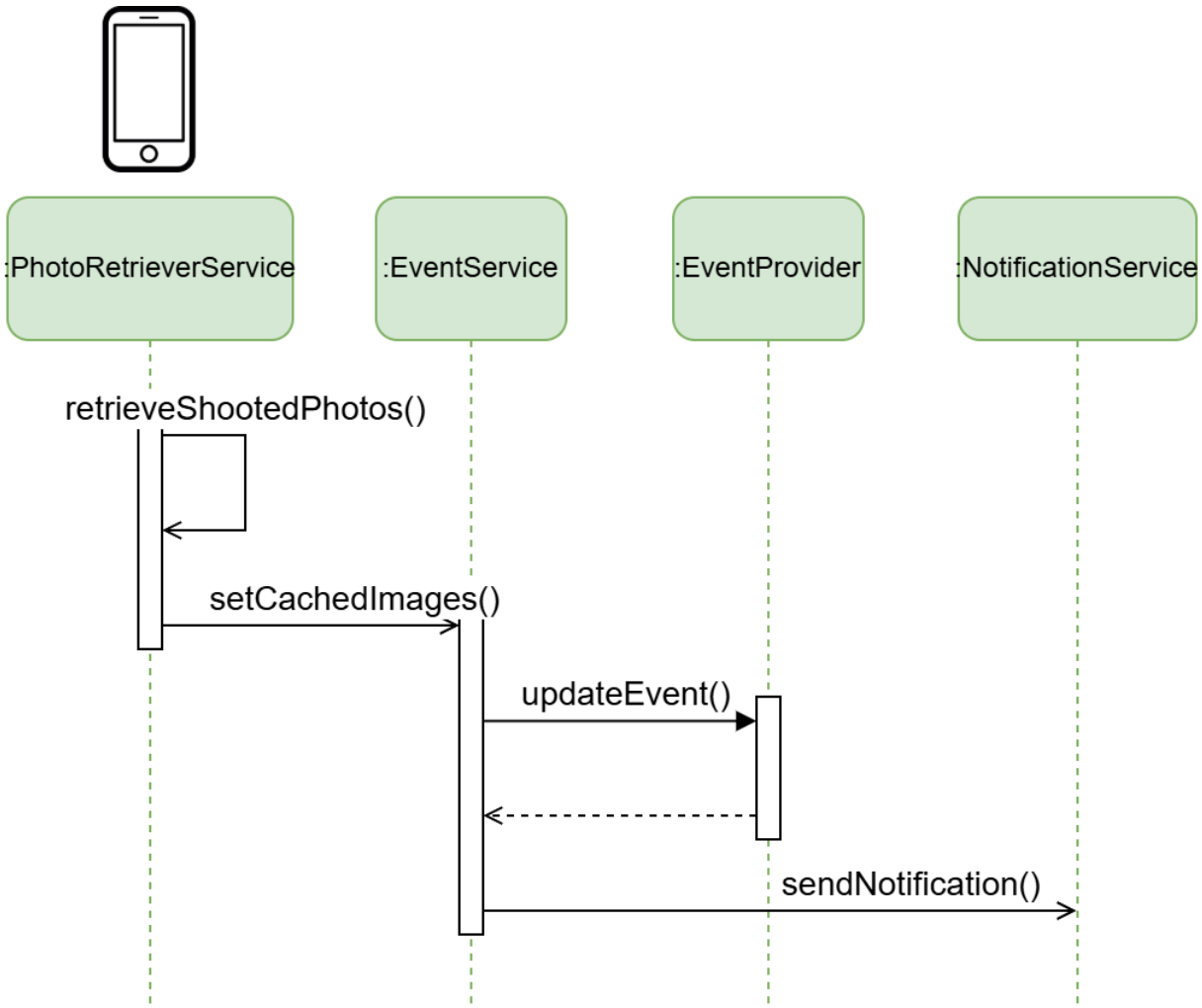

Figura 1.1: Interazione tra i componenti per il recupero delle immagini

Questa fase di conferma, oltre a garantire la trasparenza del servizio nei confronti dell'utente - riducendo il rischio di errori o caricamenti indesiderati - presenta anche vantaggi in termini di ottimizzazione delle prestazioni.

Da un punto di vista normativo, la procedura di recupero e selezione automatica delle immagini viene esplicitamente descritta nelle condizioni d'uso dell'applicazione, alle quali

l'utente deve aderire con accettazione espressa prima di utilizzare il servizio. Tuttavia, la fase di conferma dell'utente non rappresenta un obbligo giuridico, poiché la responsabilità della pubblicazione di contenuti multimediali ricade sul soggetto che realizza la fotografia. In conformità con la normativa vigente in materia di tutela dell'immagine (art. 10 c.c. e artt. 96-97 della Legge n. 633/1941) e protezione dei dati personali (Regolamento UE 2016/679 – GDPR), chi scatta una fotografia è tenuto a ottenere il consenso delle persone ritratte prima di procedere alla sua pubblicazione.

Come già affrontato nel capitolo precedente, le operazioni che coinvolgono la modifica di uno stesso componente del sistema sono soggette a vincoli di concorrenza per l'accesso alla risorsa di interesse. La conclusione di un evento condiviso tra più utenti genera richieste per aggiungere immagini - sul medesimo evento - che potrebbero dover essere eseguite simultaneamente.

La fase di selezione introduce un ritardo prima della richiesta di caricamento, dilatando la distribuzione temporale delle richieste e riducendo la probabilità di collisioni dovute a operazioni concorrenti sullo stesso elemento. L'attesa della conferma dell'utente prevede infatti un ritardo fisiologico tra la fine dell'evento e l'effettivo caricamento delle immagini, contribuendo a distribuire le richieste nel tempo e limitando il rischio di congestione del server.

Una volta completata la selezione da parte dell'utente, le immagini devono essere salvate sul server, per essere rese disponibili anche agli altri profili con cui l'evento è condiviso.

1.2 Integrazione con il sistema

A differenza dei dati usualmente scambiati all'interno del sistema, i file multimediali presentano dimensioni significativamente superiori, con una differenza che si manifesta su ordini di grandezza rilevanti. Salvare questi file nella stessa modalità degli altri dati, affiancandoli quindi agli elementi logici, comporterebbe un impatto significativo sulla dimensione totale del database, andando a influenzare tutte le operazioni eseguite su di esso, dal momento in cui il recupero delle informazioni deve avvenire su un volume di dati

molto più vasto del necessario. Il rallentamento generale delle operazioni comporterebbe un impatto significativo sulle prestazioni complessive del sistema. Per questo motivo, è necessaria una gestione della memoria specificamente progettata per l'archiviazione e il recupero di contenuti multimediali.

Inoltre, le dimensioni delle immagini e dei video influenzano direttamente il tempo di elaborazione e il volume delle richieste, aumentando il carico computazionale su tutti i componenti del sistema. Infatti, la possibilità di allegare più file a un singolo evento implica che i tempi di caricamento elevati dei file multimediali possano prolungare sensibilmente la durata delle transazioni necessarie per la modifica degli eventi, incidendo sulla reattività del sistema.

1.2.1 Scelta del servizio di persistenza

La visualizzazione dei file multimediali riveste un'importanza secondaria rispetto ad altre funzionalità offerte dall'applicazione. Di conseguenza, è possibile accettare un tempo di caricamento maggiore, a condizione che ciò contribuisca a ridurre la latenza delle operazioni invece più rilevanti. Il salvataggio dei file multimediali direttamente sul database centrale comporterebbe un aumento significativo del volume delle richieste, determinando un maggiore impiego di risorse computazionali e un incremento dei tempi di caricamento. Questo fenomeno potrebbe incidere negativamente sulle prestazioni complessive del sistema, penalizzando l'esecuzione simultanea di altre operazioni.

Per ottimizzare la gestione dei file multimediali, si è deciso di distinguere il dato binario dalla sua rappresentazione logica. In questo modo, la relazione tra il file e l'evento associato può essere mantenuta indipendentemente dai dati binari che lo compongono. Una volta recuperati i riferimenti logici ai file multimediali associati all'evento interessato, sarà possibile ottenere i loro contenuti binari in un secondo momento, solo quando necessario. Il modello del dominio illustrato in precedenza evidenzia la relazione logica tra gli eventi e i file associati (Image).

Considerando la necessità e la possibilità di archiviare i file multimediali su risorse differenti dal database centrale, è fondamentale individuare la soluzione più adatta per la loro

persistenza. I principali servizi cloud per l'archiviazione di file multimediali si suddividono in tre categorie: Object Storage, File Storage e Block Storage.

Gli Object Storage gestiscono i file in un unico livello, con la possibilità di aggiungere metadati agli oggetti. A ciascun elemento viene associato un identificativo univoco che ne consente il recupero. L'accesso ai dati avviene tipicamente tramite API RESTful, che oltre a offrire la possibilità di gestire i permessi, garantisce l'utilizzo su ampia scala. La presenza di un unico livello di indirizzamento permette una scalabilità pressoché illimitata, e un costo variabile in base alla quantità di dati memorizzati.

I File Storage organizzano i file in una struttura gerarchica di cartelle e sottocartelle, semplificando la gestione dei file e il controllo degli accessi. Oltre a facilitare un controllo ulteriore agli utenti, questa soluzione è compatibile con protocolli di accesso particolari. Tuttavia, la sua capacità e scalabilità, così come il costo effettivo, sono legati alla struttura dei file e alla capacità prevista dal piano selezionato.

Infine, i Block Storage gestiscono la memoria tramite la suddivisione dei dati in blocchi logici, salvati separatamente e ognuno dotato di identificativo univoco. Questa tecnologia offre elevate prestazioni per il recupero e la modifica dei dati, ma i costi aumentano all'incremento della quantità di dati presenti. La scalabilità è quindi limitata alla capacità assegnata al volume. Oltretutto, i costi sono elevati, particolarmente nel caso di grandi moli di dati.

Tra queste soluzioni, la categoria degli Object Storage risulta la più adatta alle esigenze del progetto di salvataggio dei file multimediali. La sua scalabilità illimitata consente di gestire grandi volumi di elementi con una ridotta dipendenza tra loro. Inoltre, l'identificazione univoca di ciascun oggetto garantisce una rapida individuazione dei dati e un'efficiente risposta prestazionale a numerose richieste contemporanee.

Nel contesto di Azure, il servizio di Object Storage fornito è rappresentato da Azure Blob Storage (ABS). ABS adotta un'organizzazione centrata su container, entità logiche che raggruppano più file multimediali e introducono un livello di indirizzamento aggiuntivo.

L'accesso in lettura ai dati avviene tramite protocollo API RESTful, con autenticazione per l'aggiunta di nuovi elementi. I container creati sono relativi alle categorie per le quali le immagini vengono salvate. Ad esempio, ci sarà un container relativo alle immagini degli eventi, così come uno per quelle legate ai profili.

Azure Blob Storage

All'interno del nome del blob si possono però aggiungere percorsi logici, separati dal carattere "\". Le immagini vengono quindi identificate univocamente attraverso la combinazione del container, dell'hash dell'elemento a cui fanno riferimento (l'evento, il profilo...) e del loro hash, concatenati per formare il percorso finale in cui sarà possibile recuperarle.

1.2.2 Procedura di salvataggio

Una volta inizializzato Azure Blob Storage, Azure Functions potrà poi connettersi per creare e gestire i container. La richiesta parte dal client verso le Azure Functions, a seguito della selezione dei file multimediali da parte dell'utente. La richiesta include, oltre all'hash dell'evento interessato, la lista delle estensioni dei file che si vogliono salvare.

Ricevuta la lista delle estensioni, la funzione chiamata esegue una verifica dei permessi di accesso necessari, per poi connettersi a Cosmos DB; dopo aver controllato che le estensioni siano tra quelle accettate dal sistema, essersi accertato che l'evento esista e aver generato gli hash per le future immagini. Inserisce quindi nel database i nuovi riferimenti logici alle immagini che includono, oltre al loro hash, il riferimento all'evento, la loro estensione e lo stato di elaborazione, inizializzato a "Created". Si connette quindi ad Azure Blob Storage per ottenere un Shared Access Signarure (SAS) token.

Il SAS token è un codice che viene usato per fornire permessi su servizi a entità terze. Chiunque sia in possesso del token ha la possibilità di eseguire le modifiche a esso collegate. È infatti possibile definire i permessi, la durata e la località da associare al token. In particolare, essendo le immagini salvate in un percorso logico che segue l'evento, il token richiesto dalla funzione e generato da ABS avrà la possibilità di creare e scrivere file sul container "eventi", nel percorso dell'evento corrispondente. La validità di utilizzo

ha durata dieci minuti.

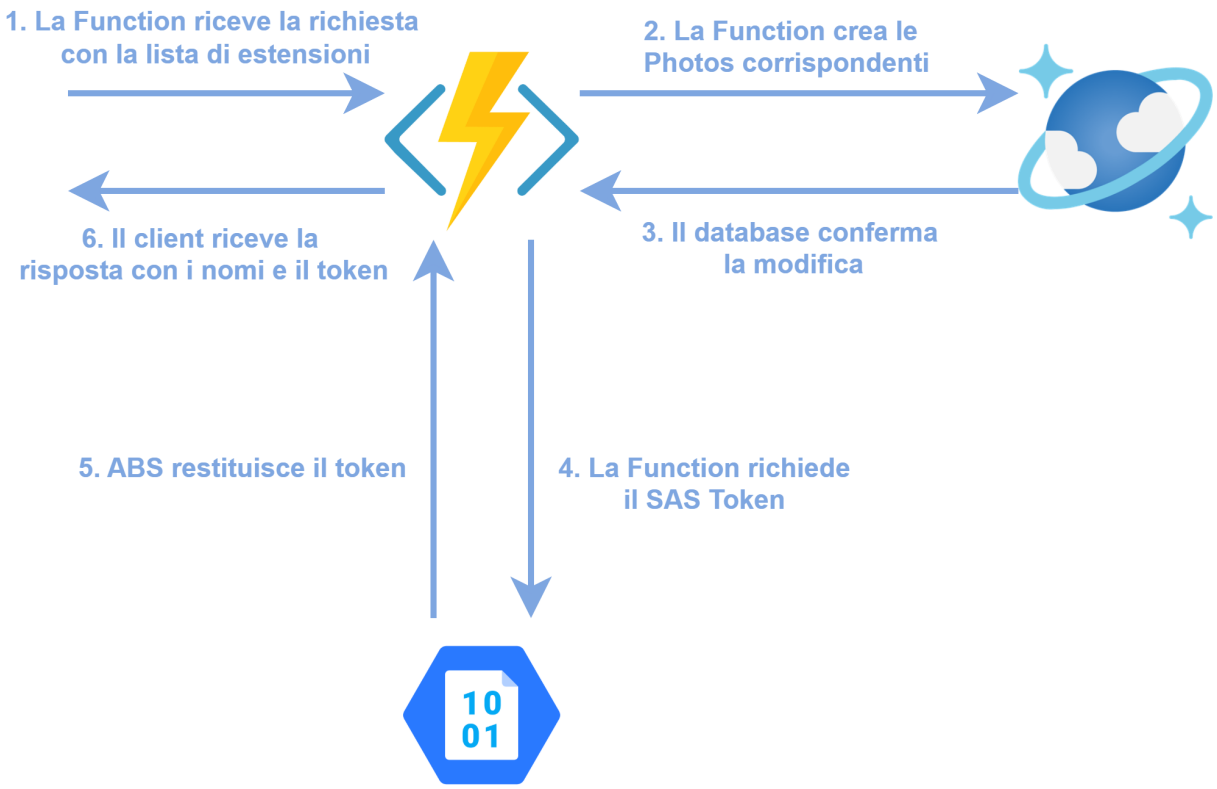

Figura 1.2: Interazione logica per l'ottenimento del token

La risposta verso il client sarà quindi composta, oltre che dal token di accesso, anche dai nomi delle immagini, affiancate dalle loro estensioni. Ricevuta la risposta il client procederà a comprimere in simultanea i file selezionati, per ridurre il consumo di banda e il volume dei dati totali trasmessi. Questa strategia consente di diminuire il carico computazionale sul server, migliorando l'efficienza complessiva del sistema. Nel momento in cui finisce la procedura di compressione, il device contatterà il Blob Storage e, usando il SAS token, caricherà le immagini usando il nome associato con l'estensione corretta.

La terminazione del caricamento è collegata all'esecuzione di una Azure Function. Il ruolo di questa funzione è quello di controllare che il file caricato corrisponda a un'immagine logica sul database. Controllerà quindi che il nome del file e il percorso associato corrispondano al nome dell'immagine e all'evento relativo, assicurandosi inoltre che l'estensione non sia cambiata e che il suo stato sia lo stesso con cui è stato inizializzato. In caso uno di questi parametri non coincida, procederà eliminando entrambe le risorse,

ovvero sia l'immagine logica sul database che il blob caricato. Altrimenti, se tutto risulta corretto, procederà ad aggiornare lo stato dell'immagine a "Uploaded".

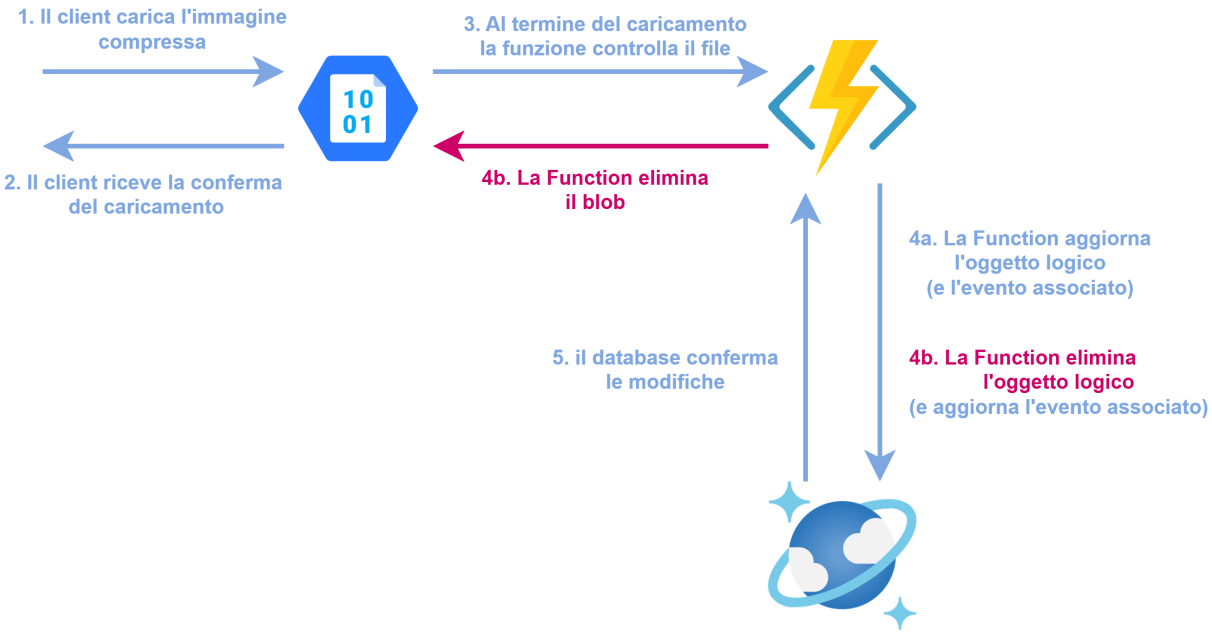

Figura 1.3: Interazione logica per il caricamento dell'immagine

Il caricamento di un'immagine costituisce una modifica all'evento. Per questo motivo è necessario aggiornare la data di aggiornamento dell'evento. Per evitare che troppe richieste ricadano sullo stesso evento, sarà solo l'ultima funzione chiamata a effettuare la modifica. A seguito delle operazioni precedenti la funzione controllerà infatti se, fra tutte quelle associate all'evento, ci siano immagini con lo stato ancora in "Started". In caso contrario si deduce che la funzione corrente è l'ultima ad aver effettuato la modifica, e sarà lei quindi ad aggiornare l'evento relativo. L'aggiornamento dell'evento comporta la propagazione della notifica a tutti i profili coinvolti.

Essendo le immagini così create indipendenti tra loro, la loro modifica non richiede il blocco o il controllo di altre entità. Essendo la modifica concentrata sulla singola entità logica delle immagini, questa operazione risulta altamente parallelizzabile, garantendo così la scalabilità delle richieste.

L'accesso in lettura ai file multimediali in ABS risulta pubblico per impostazione predefinita, poiché la mancata definizione di ruoli comporta l'assenza di controlli espliciti

sulle autorizzazioni delle richieste. Questo aspetto pone delle problematiche sulla privacy delle immagini, che viene però mitigato grazie all'uso di hash randomici sufficientemente lunghi, che riducono drasticamente la probabilità di collisione. Infatti, senza la conoscenza dell'hash corretto, un accesso non autorizzato alle immagini richiederebbe tentativi casuali estremamente numerosi, nella speranza di trovare una combinazione corretta, rendendo il successo un attacco altamente improbabile. Inoltre, anche in caso di compromissione di un hash, e quindi la possibilità di recupero di un'immagine da parte di un attore il cui accesso non dovrebbe essere permesso, l'accesso sarebbe limitato a una singola immagine, senza fornire ulteriori informazioni sugli altri file memorizzati.

Bibliografia

Amazon Web Services. Serverless Computing - AWS Lambda Pricing. Amazon Web Services, a. URL https://aws.amazon.com/lambda/pricing/.

Amazon Web Services. Amazon DynamoDB Pricing. Amazon Web Services, b. URL https://aws.amazon.com/dynamodb/pricing/.

Daniel Barcelona-Pons and Pedro García-López. Benchmarking Parallelism in FaaS Platforms. *Future Generation Computer Systems*, 124:268–84, November 2021. doi: 10.1016/j.future.2021.06.005.

Awel Eshetu Fentaw. Cross Platform Mobile Application Development: A Comparison Study of React Native Vs Flutter. *Jurnal University of Jyväskylä*, 27791:37–38, 2020.

Firebase. Authentication. Firebase Documentation. URL https://firebase.google.com/docs/auth.

Flutter. App Architecture. Flutter Documentation. URL https://docs.flutter.dev/app-architecture.

Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad Reza Sadeghi, and Jörg Schwenk. Universally Composable Security Analysis of TLS. In *Lecture Notes in Computer Science [Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics]*, volume 5324 LNCS, pages 313–27. Springer Verlag. doi: 10.1007/978-3-540-88733-1_22.

Google Cloud. Cloud Firestore Pricing. Google Cloud, a. URL https://cloud.google.com/firestore/pricing.

Google Cloud. Cloud Run pricing. Google Cloud, b. URL https://cloud.google.com/run/pricing/.

Simon Hedlund and Ylva Rasmusson Wright. Cross-platform Frameworks Comparison: Android Applications in a Cross-platform Environment, Xamarin Vs Flutter. Bachelor's thesis, Blekinge Institute of Technology, 2021.

ISO. ISO/IEC/IEEE International Standard - Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. ISO/IEC/IEEE 29148:2018(E), 2018.

B. Jose and S. Abraham. Exploring the merits of nosql: A study based on mongodb. In *2017 International Conference on Networks and Advances in Computational Technologies, NetACT 2017*, pages 266–271. Institute of Electrical and Electronics Engineers Inc., 2017. doi: 10.1109/NETACT.2017.8076778.

Benymol Jose and Sajimon Abraham. Performance Analysis of NoSQL and Relational Databases with MongoDB and MySQL. *Materials Today: Proceedings*, 24:2036–43, 2019. doi: 10.1016/j.matpr.2020.03.634.

Benymol Jose and Sajimon Abraham. Intelligent Processing of Unstructured Textual Data in Document Based NoSQL Databases. *Materials Today: Proceedings*, 80:1777–85, January 2023. doi: 10.1016/j.matpr.2021.05.605.

Microsoft Azure. Azure Cosmos DB. Azure, a. URL `https://azure.microsoft.com/en-us/products/cosmos-db`.

Microsoft Azure. Azure Functions pricing. Azure, b. URL `https://azure.microsoft.com/en-us/pricing/details/functions`.

Microsoft Azure. Azure Key Vault. Azure, c. URL `https://azure.microsoft.com/en-us/products/key-vault`.

Microsoft Azure. Azure pricing calculator. Azure, d. URL `https://azure.microsoft.com/en-us/pricing/calculator/`.

Microsoft Azure. Azure Web PubSub. Azure, e. URL `https://azure.microsoft.com/en-us/products/web-pubsub`.

Microsoft Learn. Application Insights overview. Microsoft Learn, a. URL `https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview`.

Microsoft Learn. Choose between Azure messaging services - Azure Service Bus, Azure Event Hubs, and Azure Queue Storage. Microsoft Learn, b. URL https://learn.microsoft.com/en-us/azure/service-bus-messaging/compare-messaging-services.

Microsoft Learn. Introduction to Azure Queue Storage. Microsoft Learn, c. URL https://learn.microsoft.com/en-us/azure/storage/queues/storage-queues-introduction.

Microsoft Learn. What is Azure Service Bus? Introduction to Azure Service Bus messaging. Microsoft Learn, d. URL https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview.

Microsoft Learn. Azure Functions .NET class library development. Microsoft Learn, e. URL https://learn.microsoft.com/en-us/azure/azure-functions/functions-dotnet-class-library.

Microsoft Learn. Dependency injection in .NET Azure Functions. Microsoft Learn, f. URL https://learn.microsoft.com/en-us/azure/azure-functions/functions-dotnet-dependency-injection.

Microsoft Learn. Durable Functions overview. Microsoft Learn, g. URL https://learn.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-overview.

Microsoft Learn. Develop .NET isolated process Azure Functions. Microsoft Learn, h. URL https://learn.microsoft.com/en-us/azure/azure-functions/dotnet-isolated-process-guide.

Microsoft Learn. Provision throughput on Azure Cosmos DB for NoSQL resources. Microsoft Learn, i. URL https://learn.microsoft.com/en-gb/azure/cosmos-db/provision-throughput-autoscale.

Microsoft Learn. Reliability for Azure Blob Storage. Microsoft Learn, j. URL https://learn.microsoft.com/en-us/azure/well-architected/service-guides/azure-blob-storage.

Microsoft Learn. Reliability in Azure Cosmos DB for NoSQL. Microsoft Learn, k. URL https://learn.microsoft.com/en-us/azure/reliability/reliability-cosmos-db-nosql.

Microsoft Learn. Reliability in Azure Cosmos DB for NoSQL. Microsoft Learn, l. URL https://learn.microsoft.com/en-us/azure/reliability/reliability-cosmos-db-nosql.

Microsoft Learn. Serverless capacity mode in Azure Cosmos DB for NoSQL. Microsoft Learn, m. URL https://learn.microsoft.com/en-gb/azure/cosmos-db/throughput-serverless.

Microsoft Learn. Grant limited access to Azure Storage resources using shared access signatures (SAS). Microsoft Learn, n. URL https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview.

NIST. NIST Special Publication 800-30 Revision 1 - Guide for Conducting Risk Assessments. Technical Report September, 2012. NIST Guide for Conducting Risk Assessments.

Object Management Group Inc (OMG). Unified Modeling Language (UML) Specification Version 2.5.1. Technical report, Object Management Group, Inc. (OMG), 2017. URL https://www.omg.org/spec/UML/2.5.1.

Rabi Prasad Padhy, Manas Ranjan Patra, and Suresh Chandra Satapathy. RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's. *International Journal of Advanced Engineering Sciences and Technologies*, 11(11):15–30, 2011.

Moneer Rifai. Serverless showdown: AWS Lambda vs Azure Functions vs Google Cloud Functions. Pluralsight Blog, June 2023. URL https://www.pluralsight.com/resources/blog/cloud/serverless-showdown-aws-lambda-vs-azure-functions-vs-google-cloud-functions.

Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. Architectural Implications of Function-as-a-Service Computing. In *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, pages 1063–75. IEEE Computer Society, 2019. doi: 10.1145/3352460.3358296.