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 multipiattaforma per l'organizzazione di eventi condivisi e la condivisione multimediale automatica in tempo reale

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 Autenticare le richieste: la scelta del servizio e la sua integrazione

Poiché la modalità di autenticazione rappresenta un elemento importante per l'esperienza utente, in quanto deve assicurare un accesso sicuro all'applicazione mantenendone la semplicità, la facilità del processo di autenticazione deve essere garantita. L'applicazione deve consentire la possibilità di registrarsi creando un nuovo account dedicato, ma è altrettanto essenziale che permetta agli utenti di farlo anche tramite il proprio servizio di autenticazione preferito, migliorando sicuramente l'usabilità e l'apprezzamento. Di conseguenza, il sistema di gestione degli accessi deve supportare sia la registrazione e la gestione autonoma degli account specifici per il servizio, sia fornire l'integrazione con provider di autenticazione esterni.

Per lo scopo, Azure fornisce Microsoft Entra ID, parte della suite di servizi di autenticazione e autorizzazione Microsoft Entra. Sebbene teoricamente in grado di soddisfare i requisiti sopra indicati, la complessità della documentazione e le difficoltà riscontrate nell'integrazione con il servizio dell'applicativo hanno portato a valutare soluzioni alternative negli ambienti cloud.

La scelta è quindi ricaduta su Firebase Authentication, il servizio di autenticazione di Google Cloud Provider, che garantisce sia la possibilità di creare account dedicati che di collegarsi attraverso altri servizi di autenticazione. Presenta librerie di integrazione sia tramite Flutter che tramite C# che risultano facili da utilizzare, oltre a fornire una piattaforma di gestione con un'interfaccia chiara e intuitiva. Dal punto di vista economico, il servizio risulta vantaggioso, essendo gratuito fino ai cinquantamila utenti mensili attivi.

Firebase Authentication

Firebase si integra facilmente con Flutter, fornendo una libreria che gestisce completamente l'ottenimento e il mantenimento dei token di autenticazione, a partire dalle credenziali o dalle verifiche precedenti. Per ogni richiesta che richiede identificazione un servizio apposito intercetta il messaggio, recuperando il token e allegandoglielo. Alla ricezione del messaggio, il server estrae il token dalla richiesta, per poi contattare Firebase grazie l'astrazione fornita dalla libreria. Firebase controlla il token e, se corretto, ne restituisce i dati dell'account relativo.

Figura 2: Fasi di ottenimento e uso del token

Uno dei requisiti del progetto prevede che ogni account sia associato in modo univoco a un singolo utente. Durante la fase di registrazione, tuttavia, l'account viene inizialmente registrato nel database gestito da Firebase. Pertanto, al primo accesso, il server, dopo aver verificato l'autenticità della richiesta, provvede a creare una copia dell'account, generando poi il relativo nuovo oggetto utente e il primo profilo associato.

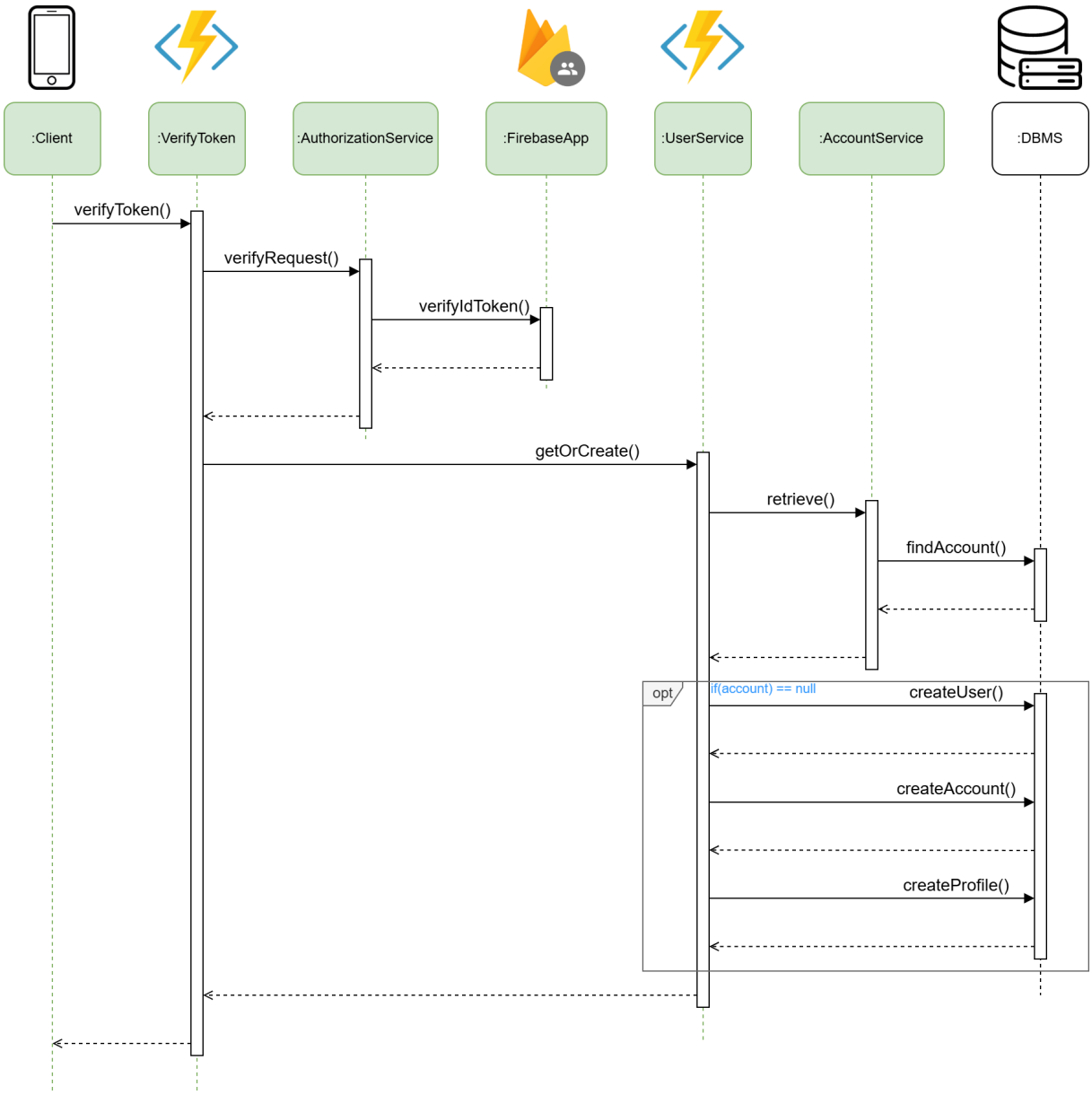

Figura 3: Diagramma di sequenza per la creazione di un account

0.2 Uno sguardo sulla sicurezza: segreti e protocolli

Il collegamento tra i vari componenti all'interno dell'ambiente Azure richiede l'utilizzo di chiavi e stringhe di connessione. Il salvataggio di tutte le chiavi sensibili è stato affidato al servizio Azure Key Vault, un server che permette la centralizzazione dei dati, cifrando il contenuto e garantendo un controllo maggiore sul loro utilizzo.

Azure Key Vault

Quando necessario i servizi, in particolare le Azure Functions, contatteranno il Key Vault per l'ottenimento delle chiavi necessarie, separando di fatto la logica implementativa dai segreti necessari per la sua esecuzione, riducendo così il rischio di una perdita delle chiavi derivata da un errore dello sviluppo.

Le comunicazioni tra i vari componenti devono avvenire in sicurezza, garantendo autenticità e confidenzialità. Per questo motivo tutte le comunicazioni tra dispositivi client e i vari servizi utilizzano la tecnologia TLS, che permette di cifrare i messaggi grazie a uno standard collaudato. In particolare, le comunicazioni tra i client e Azure Functions, così come con Firebase Authentication e il server per la persistenza delle immagini, avvengono tramite protocollo HTTPS, mentre le comunicazioni con il server per gli aggiornamenti in tempo reale usano il protocollo WSS.

Il rischio di saturazione delle risorse viene mitigato aggiungendo un duplice controllo sulle dimensioni delle richieste. In primo luogo si limita la dimensione massima della singola richiesta, facendo particolare attenzione alle richieste che contengono immagini, controllandola sia nel momento dell'invio che nel momento della ricezione. Inoltre, alla fine di ogni richiesta più grande di una determinata soglia, la dimensione viene sommata alle precedenti nell'ultimo periodo e, se la somma risulta troppo elevata, viene limitato l'utilizzo per quell'utente.

Per evitare un numero eccessivo di richieste totali, che possono provocare anch'esse una riduzione del servizio, è possibile integrare nel sistema risorse create appositamente da Azure, quali Azure DDOS Protection.

L'accesso al database è ristretto alle sole risorse Azure, garantendo l'isolamento dall'esterno, che comprometterebbe altrimenti l'affidabilità dei dati.

Infine, l'identificativo di ogni elemento del dominio è nascosto all'utente tramite la creazione di codici hash univoci che permettono comunque l'identificazione dell'oggetto senza rivelare ulteriori informazioni. In particolare, il recupero delle immagini (disponibili in teoria pubblicamente), avviene grazie a un link univoco dato dalla combinazione degli identificativi dell'evento e dell'immagine. Utilizzando i codici di hash diventa molto complicato ritrovare le immagini senza essere a conoscenza dei codici, che non avendo natura incrementale ma distribuita rende indovinare l'unica strategia per trovare un link valido.

0.3 Il monitoraggio dei servizi

Il monitoraggio del sistema è attuato in due modalità: tramite il salvataggio dei log e grazie al controllo delle prestazioni del sistema.

Relativamente a Firebase Authentication vengono forniti inclusi al servizio sia le interfacce per il controllo delle prestazioni che per la gestione dei log. Non è quindi richiesta alcuna ulteriore azione.

Per monitorare le Azure Functions sarà invece necessario affiancargli un'istanza di Azure Application Insights, servizio nato appositamente per controllare il funzionamento e la risposta dei servizi Azure. Una volta collegato il servizio, infatti, Application Insight permette la presentazione e l'analisi di numerose metriche, quali il tempo di risposta e il consumo di risorse. Consente inoltre di testare la risposta dell'applicativo simulando diversi scenari e riassumendo il loro comportamento.

Azure Application Insights

La creazione dei log è invece delegata al programmatore, in quanto è necessario integrarli nel codice. Nel momento della creazione, ogni funzione riceve, tramite dependency injection, un servizio Logger che permette la creazione e il salvataggio dei log. La funzione

non dovrà fare altro che chiamare il metodo apposito per generare e salvare un log. Tali log saranno poi consultabili e analizzabili tramite l'interfaccia fornita da Azure Application Insight.