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 Marzo 2025

Anno Accademico 2024/2025

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 alla completa e continua garanzia di manutenzione, offrono l'architettura ideale per il supporto di simili progetti.

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, e 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 dell'applicativo in questione, evidenziando l'impatto dell'integrazione delle risorse cloud sul risultato finale.

Indice

1 L'Analisi delle funzionalità

La realizzazione di qualunque prodotto software inizia da una fase in cui, partendo dall'abstract del progetto, si analizzano i requisiti e le funzionalità da realizzare. L'obiettivo è arrivare ad una definizione concisa e condivisa col cliente delle proprietà e del comportamento desiderato nell'applicazione. senza entrare nel merito delle scelte implementative. A quel punto si può procedere con la progettazione del programma vero e proprio.

1.1 I requisiti e i casi d'uso

Uno studio dell'abstract del progetto porta all'individuazione e alla descrizione delle caratteristiche essenziali. I requisiti formalizzano le funzionalità da realizzare, sintetizzando e schematizzando gli elementi descrittivi del prodotto. I casi d'uso descrivono le interazioni tra l'utente e il sistema, suddividendo le funzionalità in azioni elementari.

1.1.1 I requisiti e il vocabolario

Devono risultare chiari e precisi per permettere di procedere in modo corretto e trasparente.

Si suddividono in funzionali o non funzionali in base alle caratteristiche che descrivono. I requisiti funzionali descrivono le funzionalità che il sistema deve avere, mentre i requisiti non funzionali descrivono le caratteristiche che il sistema deve soddisfare per essere considerato valido.

Si noti come le caratteristiche di velocità e scalabilità vengono individuate ed introdotte fin da subito. (non è detto che tutte le applicazioni debbano servire milioni di utenti)

ID	Requisiti	Tipo
R1F	Registrazione di un account tramite l'interfaccia web	Funzionale
R2F	Identificazione attraverso mail univoca e password di almeno 6 caratteri	Funzionale
R3F	Visualizzazione degli eventi confermati	Funzionale
R4F	Visualizzazione degli eventi proposti	Funzionale
R5F	Creazione di un evento impostando almeno la data di inizio e quella di fine	Funzionale
R6F	La data di fine deve essere successiva alla data di inizio	Funzionale
R7F	Modifica di un evento	Funzionale
R8F	La conferma di un evento lo sposta negli eventi confermati	Funzionale
R9F	La disdetta di un evento lo sposta negli eventi proposti	Funzionale
R10F	Caricamento delle foto di un evento	Funzionale
R11F	Condivisione tramite link	Funzionale
R12F	Condivisione tramite gruppo o ad altri profili	Funzionale
R13F	Ricerca automatica delle foto sul dispositivo mobile	Funzionale
R14F	Conferma delle foto	Funzionale
R15F	Ricerca di altri profili	Funzionale
R16F	Creazione di un gruppo da due o più profili	Funzionale
R17F	Visualizzazione dei profili collegati	Funzionale
R18F	Creazione di un nuovo profilo	Funzionale
R19F	Cambio del profilo attualmente in uso	Funzionale
R20F	Aggiornamento in tempo reale delle modifiche agli eventi	Funzionale
R1NF	Per interagire l'utente deve essere autenticato	Non Funzionale
R2NF	Velocità di richiesta iniziale dei dati	Non Funzionale
R3NF	Semplicità e fluidità dell'interfaccia grafica	Non Funzionale
R4NF	Velocità in lettura e scrittura dei dati	Non Funzionale
R5NF	Velocità nella ricerca dei profili	Non Funzionale
R6NF	Scalabilità delle richieste	Non Funzionale

Tabella 1: Tabella dei requisiti di Wyd

Voce	**Definizione**	**Sinonimi**
Account	combinazione di mail e password che identifica un utente	
Utente	Persona che utilizza l'applicazione	
Profilo	Entità logica che raggruppa eventi e interazioni	
Profili collegati	Profili a cui l'utente può avere accesso	
Gruppo	Insieme di profili	
Evento	Azione(o previsione di azione) con una durata nel tempo	
Data e ora evento	Indicazione temporale del momento in cui avverrà l'azione	
Evento confermato	Evento a cui il profilo ha dato conferma di partecipazione	
Evento proposto	Evento a cui il profilo non ha dato conferma di partecipazione	Evento disdetto, evento condiviso
Email	Indirizzo di posta elettronica del cliente utilizzata anche per l'autenticazione	
Password	Codice alfanumerico di almeno 8 caratteri	
Credenziali	Insieme composto da email e password necessari per accedere al sistema	

Tabella 2: Vocabolario di Wyd

Si affianca alla tabella dei requisiti quella del vocabolario, che definisce i termini utilizzati nel progetto.

1.1.2 I casi d'uso

I casi d'uso descrivono le interazioni tra gli attori ed il sistema, suddividendo le funzionalità in azioni elementari. Gli attori sono tutti gli elementi che compiono una parte attiva all'interno del programma. I casi d'uso sono collegati tra loro dai rapporti di inclusione ed estensione. Si dice che un caso include un altro se contiene il suo comportamento. "An include relationship defines that a use case contains the behavior defined in another use case." Si dice che un caso d'uso estende un altro se "A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case."

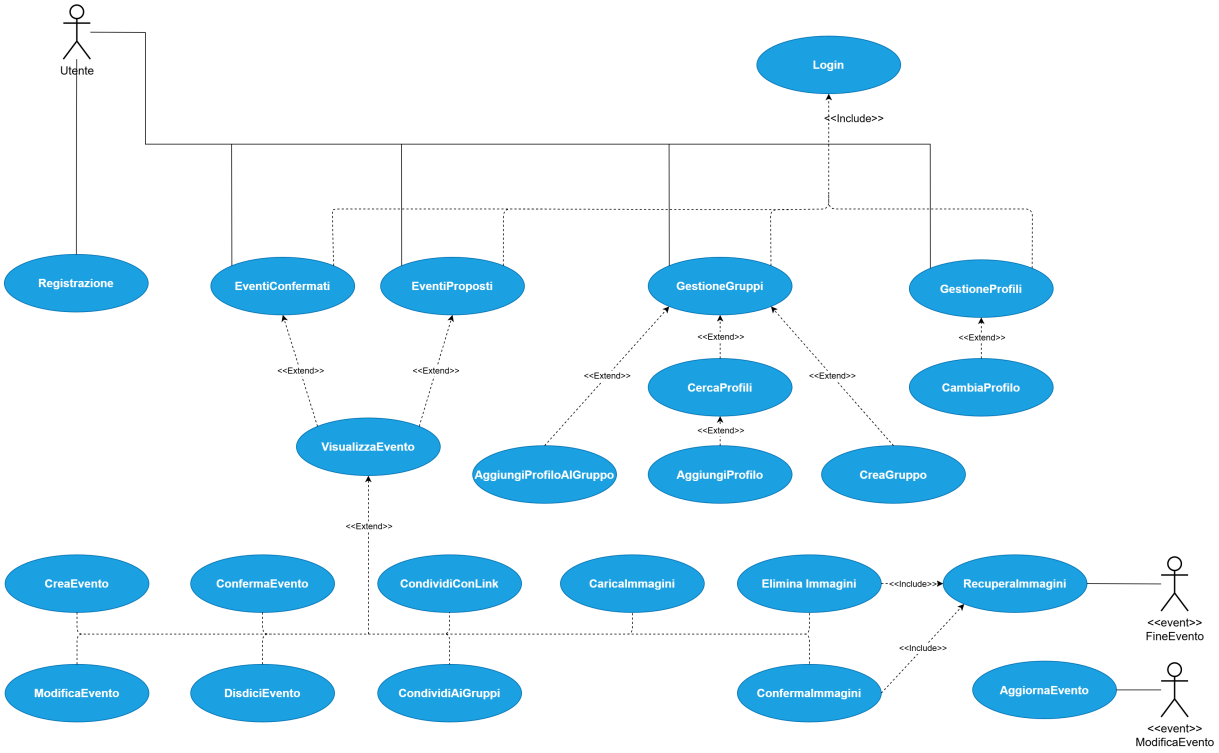

Figura 1: Diagramma dei casi d'uso

Per ogni caso d'uso viene identificato uno scenaro di utilizzo, che chiarifica il contesto, il comportamento e i punti critici dell'utilizzo. In particolare, si riportano gli scenari di utilizzo per i casi d'uso principali di Wyd. che vanno più ad impattare sulla struttura e sulle esigenze del progetto.

Lo scenario di registrazione vede la responsabilità, oltre che di creare un account, di associare un profilo all'utente. Questo permette di avere una struttura gerarchica che permette di associare più profili ad un'unico utente, che può in seguito crearne o associarne di nuovi.

Titolo	Registrazione
Descrizione	L'utente si registra al servizio
Attori	Utente
Relazioni	
Precondizioni	
Postcondizioni	L'utente è registrato nel sistema e può interagire con il resto dell'applicazione
Scenario principale	1.L'utente accede alla schermata di registrazione 2. L'utente inserisce email e password 3. Il sistema crea un account con le credenziali inserite, associando un utente ed un primo profilo 4. L'utente termina la registrazione, se avvenuta con successo viene reindirizzato alla pagina principale
Scenari Alternativi	Il sistema verifica che è già presente un account con la mail inserita, quindi procede con la procedura di login normale.
Requisiti non funzionali	Per interagire l'utente deve essere autenticato Velocità in lettura e scrittura dei dati
Punti aperti	

Tabella 3: Scenario di registrazione

A seguito della modifica dell'evento, che implica il salvataggio dei dati, viene chiesto l'aggiornamento in tempo reale di tutte le parti interessate. La modifica dei dati necessita inoltre un controllo sulle richieste contemporanee per evitare conflitti.

Titolo	ModificaEvento
Descrizione	Salva le modifiche ad un evento
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste e sono stati modificati dei dati
Postcondizioni	Le modifiche vengono salvate e propagate a tutti i profili collegati
Scenario Principale	1. VisualizzaEvento 2. Il sistema controlla che i dati modificati siano corretti 3. I cambiamenti vengono salvati 4. Tutti i dispositivi collegati ai profili collegati all'evento visualizzano le immagini
Scenari Alternativi	2. Se i dati risultano sbagliati, il sistema notifica l'utente originario indicando l'errore
Requisiti non funzionali	Velocità in lettura e scrittura dei dati Scalabilità delle richieste
Punti aperti	Le modifiche all'evento devono essere consistenti, soprattutto in caso di richieste simultanee

Tabella 4: Scenario della modifica di un evento

Il caricamento o salvataggio delle immagini è un'operazione di particolare importanza vista la sua rilevanza nel coinvolgimento degli utenti all'utilizzo delle funzionalità centrali dell'applicazione, e quindi al successo del progetto. Oltre a mostrare un'interfaccia intuitiva, il sistema deve essere in grado di gestire le richieste di caricamento, che generalmente richiedono più tempo e memoria, in modo efficiente e scalabile.

Titolo	CaricaImmagini
Descrizione	Permette all'utente di selezionare immagini da collegare all'evento, salvandole
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste
Postcondizioni	Le immagini vengono salvate e propagate a tutti i profili collegati
Scenario Principale	1. VisualizzaEvento 2. L'utente seleziona le immagini che vuole caricare 3. Le immagini vengono salvate 4. Tutti i dispositivi relativi ai profili collegati all'evento visualizzano le immagini
Scenari Alternativi	Scenario alternativo A: 3. Almeno una delle immagini crea problemi di lettura, l'utente viene notificato e può riprovare a caricare le immagini Scenario alternativo B: 3. Solo una parte delle immagini vengono salvate, altre comportano errori 4. l'utente viene notificato dell'errore e può riprovare a caricare le immagini 5. Tutti i dispositivi relativi ai profili collegati all'evento visualizzano le immagini Scenario alternativo C: 3. Nessuna immagine risulta salvata con successo 4. l'utente viene notificato dell'errore e può riprovare a caricare le immagini
Requisiti non funzionali	Semplicità e fluidità dell'interfaccia grafica Velocità in lettura e scrittura dei dati Scalabilità delle richieste
Punti aperti	

Tabella 5: Scenario del caricamento delle immagini

L'azione di recupero delle immagini facilita l'utilizzo dell'applicazione semplificando il procedimento di ricerca delle immagini, richiedendo all'utente solo la loro conferma. La sua corretta implementazione ne fa apprezzare l'utilità, con una significativo influenza sull'esperienza utente.
Richede la pianificazione e l'automazione del processo di cernita di dati, con effetti sull'analisi tecnologica, sui processi in background e sulla gestione della memoria locale.

Titolo	RecuperaImmagini
Descrizione	L'applicazione controlla la galleria e salva in locale le foto scattate durante l'evento
Attori	FineEvento
Relazioni	EliminaImmagini, ConfermaImmagini
Precondizioni	L'evento esiste ed è concluso l'utente ha dato il permesso all'accesso alla galleria
Postcondizioni	Le immagini sono salvate in locale e l'utente viene notificato
Scenario Principale	1. Il sistema attende la fine dell'evento 2. Il sistema controlla la galleria per trovare le immagini scattate nell'arco temporale dell'evento 3. Se ci sono immagini, vengono salvate in locale e l'utente viene notificato
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	L'implementazione dipende dal dispositivo su cui viene eseguita l'applicazione, alcuni dispositivi potrebbero non permetterne l'esecuzione

Tabella 6: Scenario di recupero delle immagini dal dispositivo dell'utente

1.1.3 I requisiti di sicurezza

Ogni sistema è esposto a vulnerabilità e rischi che impattano sul corretto funzionamento dell'applicazione e possono comportare disservizi in base alla loro rilevanza nel funzionamento del sistema. La definizione dei requisiti di sicurezza deriva quindi da un'analisi del rischio.

L'analisi del rischio avviene tramite la valutazione dei beni, l'identificazione delle minacce e dei punti deboli noti nelle tecnologie, per individuare i possibili vettori di attacco e orientare le risorse dove più necessario.

La valutazione dei beni determina i componenti fondamentali da proteggere, risaltandone il valore e l'esposizione relativa. Questo permette di stabilire le priorità dei componenti sui cui concentrare le attenzioni.

Bene	**Valore**	**Esposizione**
Sistema Informativo	Alto. Fondamentale per il funzionamento del servizio	Alta. Perdita finanziaria e di immagine
Informazioni dei clienti	Alto. Informazioni personali	Alta. Perdita di immagine dovuta alla divulgazione di dati sensibili
Informazioni relativi agli eventi	Medio-alto, necessari per offrire il servizio e contenenti informazioni personali e potenzialmente riservate	Molto Alta. Perdita di immagine possibile con la divulgazione dei dati relativi ai clienti
Dati dei gruppi	Medio. Necessario per condividere gli eventi	Alta. Perdita di immagine

Tabella 7: Valutazione dei beni

La tabella delle minacce individua gli attacchi principali previsti che possono avvenire sul sistema. Esamina la loro probabilità, le azioni richieste per controllarli ed il costo di realizzazione delle contromisure necessarie. Fornisce quindi una prima analisi sulle necessità implementative.

Minaccia	**Probab.**	**Controllo**	**Fattibilità**
Furto credenziali utente	Alta	Controllo sulla sicurezza della password - Log delle operazioni, autenticazione a due fattori	Costo implementativo medio
Alterazione o intercettazione delle comunicazioni	Alta	Utilizzo di un canale sicuro - Log delle operazioni, autenticazione integrata nel messaggio	Basso costo di realizzazione con determinati protocolli
Accesso non autorizzato al database	Bassa	Accesso da macchine sicure - Log di tutte le operazioni	Basso costo di realizzazione, il server deve essere ben custodito
DoS	Bassa	Controllo e limitazione delle richieste	Media complessità di implementazione
Saturazione del database	Bassa	1. Limitazione delle richieste in un dato intervallo di tempo. 2. Limitazione della grandezza delle richieste singole 3. Limitazione della grandezza richiesta dallo stesso utente in un dato intervallo di tempo	Media complessità di implementazione

Tabella 8: Tabella delle minacce

L'analisi tecnologica della sicurezza entra nel merito delle tecnologie che si prevede necessarie. Per ognuna esamina i punti deboli e i limiti intrinseci, producendo un quadro delle particolarità su cui porre maggiore attenzione.

Tecnologia	Vulnerabilità
Autenticazione email/password	• Utente rivela volontariamente la password • Utente rivela la password con un attacco di ingegneria sociale • Password banali
Cifratura comunicazioni	• In caso di cifratura simmetrica particolare attenzione va alla lunghezza delle chiavi ed alla loro memorizzazione
Architettura Client/-Server	• DoS • Man in the Middle • Sniffing delle comunicazioni
Connessione Server/-Persistenza	• Limite massimo di connessioni contemporanee • Saturazione del Database

Tabella 9: Analisi tecnologica della sicurezza

Si prevedono quindi i principali attori malevoli e i relativi casi d'uso, per poi definire i requisiti su cui si baseranno le contromisure necessarie. I casi d'uso identificano le principali modalità di attacco, e ad ognuno ne viene corrisposto un altro che ne comporta la mitigazione. Vengono quindi integrati con i casi d'uso individuati in precedenza, evidenziando le necessità e la loro applicazione.

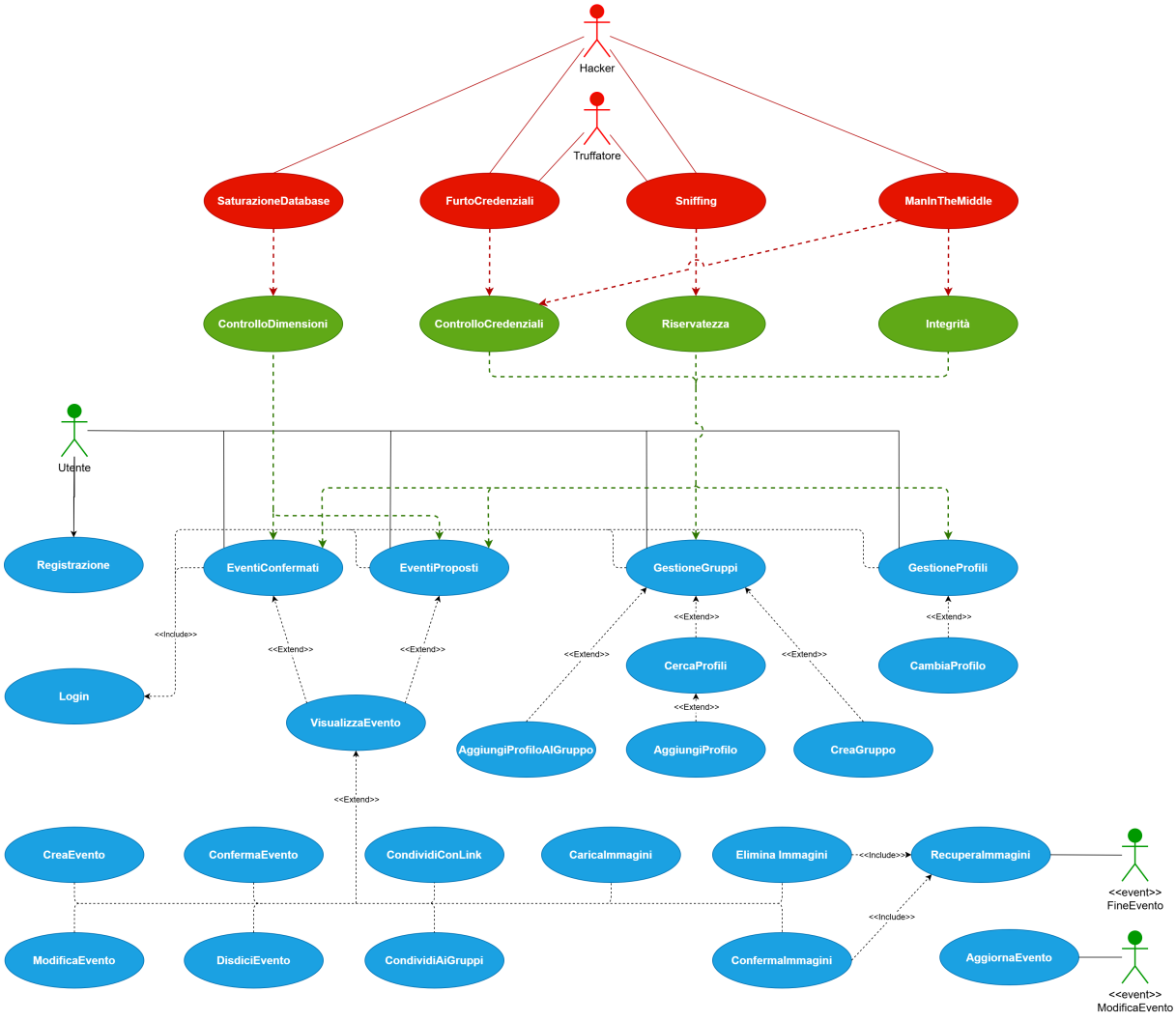

Figura 2: Casi d'uso relativi alla sicurezza

Visti i costi ed appurate le risorse a disposizoine sussistono i seguenti requisiti inerenti alla protezione dei dati e delle funzionalità:

1. Implementare un sistema di log per tracciare tutti i messaggi tra i client e i server, inclusi gli accessi, le richieste di prenotazione, di conferma, di sospensione e di invio e ricezione di dati

2. I dati salvati devono essere protetti da un attaccante che abbia accesso al sistema, prendendo misure di sicurezza fisica, eventualmente cifrando i dati

3. I dati inviati tra le parti remote devono essere protetti, utilizzando la cifratura dei dati

4. Tutte le azioni avvenute sul sistema devono essere tracciate tramite un sistema di log.

5. Il sistema deve essere resistente ad un alto numero di richieste contemporanee

6. La dimensione delle richieste non deve superare una determinata soglia

La visione e l'analisi dei log verrà gestita con uno strumento esterno, accessibile solo al personale autorizzato.

ID	**Requisiti**	**Tipo**
R21F	Implementazione di un sistema di log per tracciare tutti i messaggi tra i client e i server	Funzionale
R22F	Le richieste non devono superare una certa dimensione	Funzionale
R7NF	I dati salvati devono essere protetti da un attaccante che abbia accesso al sistema, prendendo misure di sicurezza fisica, eventualmente cifrando i dati	Non Funzionale
R8NF	I dati inviati tra le parti remote devono essere protetti, utilizzando la cifratura dei dati	Non Funzionale
R9NF	Il sistema deve essere resistente ad un alto numero di richieste contemporanee	Non funzionale

Tabella 10: Requisiti di sicurezza

1.2 L'analisi del problema

Una volta identificati i requisiti ed i casi d'uso, si procede con l'analisi del problema.

L'analisi del problema entra nel merito del comportamento dell'applicazione, evidenziando il rapporto tra requisiti, casi d'uso e funzionalità. Determina quindi l'architettura logica, che delinea le relazioni fondamentali del sistema, individuando i componenti logici principali e le loro responsabilità.

1.2.1 Analisi delle funzionalità

Le funzionalità sono dedotte dai casi d'uso, sintetizzando i servizi principali dell'applicazione. In particolare, le funzionalità vengono rilevate in base alla loro relazione con gli agli casi d'uso, alla specificità del compito che assolvono e alla pertinenza reciproca.

Si riportano in tabella le funzionalità che racchiudono altri casi d'uso.

Funzionalità	**Scomposizione**
EventiConfermati	VisualizzaEvento
EventiProposti	VisualizzaEvento
VisualizzaEvento	CreaEvento, ModificaEvento, ConfermaEvento, DisdiciEvento, CondividiConLink, CondividiAiGruppi, CaricaImmagini, EliminaImmagini, ConfermaImmagini
GestioneGruppi	CercaProfili, AggiungiProfiloAlGruppo, CreaGruppo
CercaProfili	AggiungiProfilo
GestioneProfili	CambiaProfilo

Tabella 11: Scomposizione delle funzionalità

Di ogni funzionalità vengono evidenziati il grado di complessità, la tipologia di azione che svolgono ed i requisiti collegati. Il grado di complessità riassume la quantità e la difficoltà implementativa delle azioni che una funzionalità ricopre. La tipologia riporta in maniera generale la qualità dei servizi offerti. Infine si riportano gli identificatori dei requisiti funzionali che ogni finzionalità soddisfa.

Funzionalità	**Tipo**	**Grado di complessità**	**Requisiti Collegati**
Login	Interazione esterno e lettura dati	semplice	R2F
Registrazione	Interazione esterno e memorizzazione dati	semplice	R1F
EventiConfermati	Interazione esterno e gestione dati	complessa	R3F, R8F
EventiProposti	Interazione esterno e gestione dati	complessa	R4F, R9F
GestioneGruppi	Interazione esterno e gestione dati	complessa	R15F, R16F
GestioneProfili	Interazione esterno e gestione dati	complessa	R17F, R18F, R19F
VisualizzaEvento	Interazione esterno e gestione, lettura e memorizzazione dati	complessa	R5F, R6F, R7F, R8F, R9F, R10F, R11F, R12F, R14F
AggiornaEvento	Gestione dati	complessa	R20F
RecuperaImmagini	Lettura dati	complessa	R13F
ScritturaLog	Memorizzazione dati	semplice	R21F

Tabella 12: Funzionalità

Si procede analizzando i dati che ogni funzionalità gestisce, indicandone la tipologia, la protezione richesta ed i vincoli correlati, per conoscere in maniera definitiva tutte le caratteristiche delle informazioni scambiate. A seguito dell'analisi delle informazioni, si procede con l'analisi dei vincoli, in cui si chiarificano i requisiti non funzionali, evidenziandone le

criticità e quali componenti ne vengono coinvolti.

Requisito	**Categorie**	**Impatto**	**Funzionalità**
Semplicità dell'interfaccia	Usabilità	Intuitività di utilizzo	Login, Registrazione, EventiConfermati, EventiProposti, GestioneGruppi, GestioneProfili, VisualizzaEvento, RecuperaImmagini
Velocità della ricerca dei dati	Tempo di Risposta	Maggiore reattività	EventiConfermati, EventiProposti, GestioneGruppi, GestioneProfili, RecuperaImmagini
Velocità di memorizzazione dei dati	Tempo di Risposta	Maggiore reattività	Registrazione, AggiornaEvento, RecuperaImmagini
Controllo Accessi	Sicurezza	Peggiorano tempo di risposta e usabilità, migliorano la privacy dei dati	EventiConfermati, EventiProposti, GestioneGruppi, GestioneProfili, VisualizzaEvento
Protezione dei Dati	Sicurezza	Peggiorano tempo di risposta, migliorano la privacy dei dati	Login, Registrazione, EventiConfermati, EventiProposti, GestioneGruppi, GestioneProfili, VisualizzaEvento, AggiornaEvento, RecuperaImmagini
Scalabilità delle richieste	Tempo di Risposta	Minor degradamento delle prestazioni	EventiConfermati, EventiProposti, AggiornaEvento, RecuperaImmagini

Tabella 13: Vincoli

Infine si definiscono logicamente le maschere, ovvero i componenti visuali essenziali del programma. Ad ogni maschera corrisponderà un'interfaccia grafica tramite la quale l'utente potrà accedere alle funzionalità. Vengono quindi associate le maschere alle funzionalità di cui permettono l'esecuzione, indicando le informazioni relative.

Maschera	**Informazioni**	**Funzionalità**
View Login	email, password	Login
View Registrazione	email, password	Registrazione
View EventiConfermati	lista eventi confermati	EventiConfermati, AggiornaEvento
View EventiProposti	lista eventi proposti	EventiProposti, AggiornaEvento
View VisualizzaEvento	Identificativo utente, titolo, descrizione, data e orario di inizio, data e orario di fine, confermato, immagini, profili associati	VisualizzaEvento, RecuperaImmagini
View GestioneGruppi	lista gruppi	GestioneGruppi
View CercaProfili	tag di ricerca, lista profili	CercaProfili
View GestioneProfili	Lista profili, Identificativo utente, Identificativo profilo corrente	GestioneProfili

Tabella 14: Maschere

1.2.2 Architettura logica

Definite le relazioni e le informazioni relative alle funzionalità, si necessita di esprimere logicamente i componenti principali del prodotto e le loro relazioni. Le funzionalità vengono espresse a livello logico tramite package e diagrammi delle classi, mentre i dati vengono descritti all'interno del dominio.

Il modello del dominio individua le entità che rappresentano logicamente le dipendenze tra i dati. Ogni entità presenta i suoi dati tramite proprietà, identificate da un nome e dalla tipologia del dato. Inoltre, all'interno del modello vengono indicati i rapporti tra le entità specificando le cardinalità reciproche.

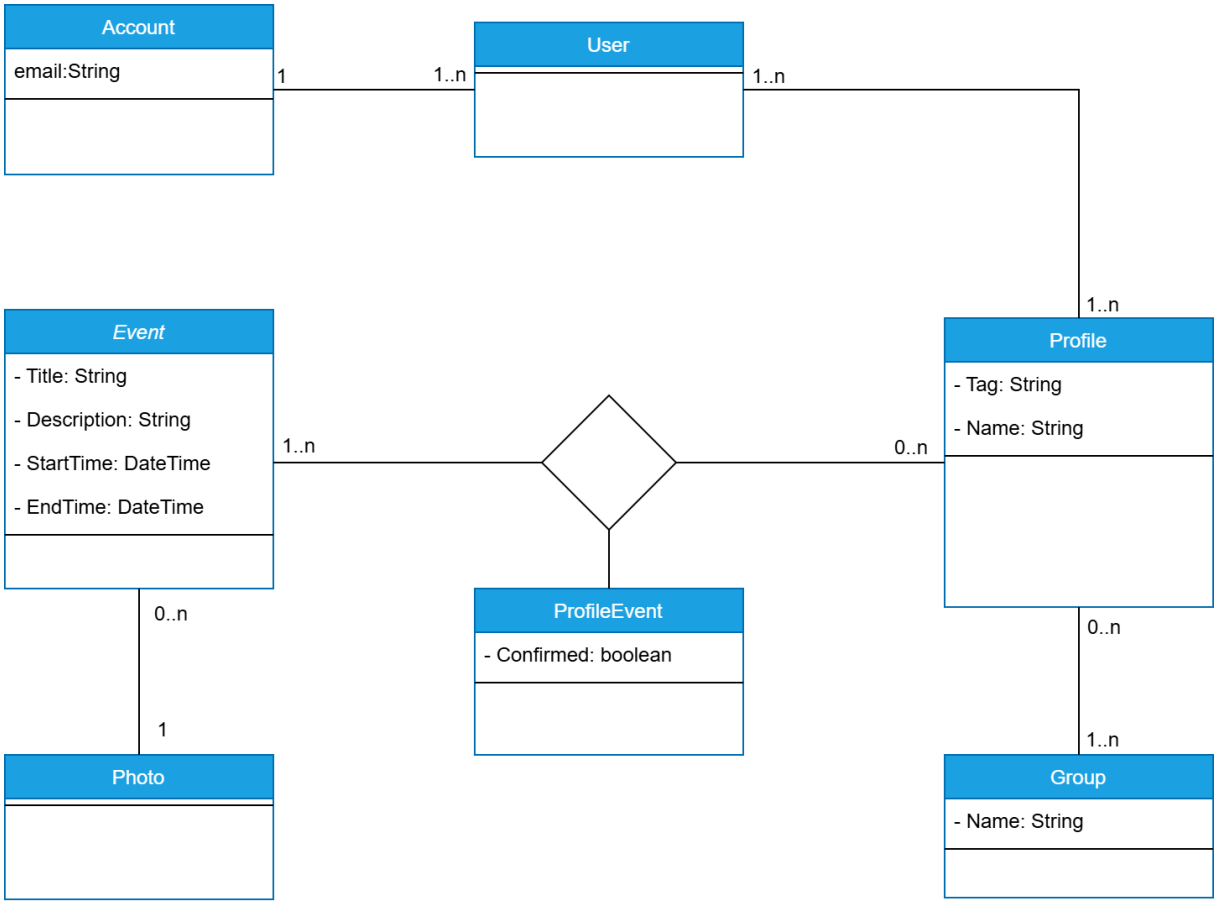

Figura 3: Modello del dominio

La divisione delle responsabilità logiche viene descritta nel diagramma dei package. Ogni package rappresenta una parte di prodotto che soddisfa una determinata responsabilità. La responsabilità viene individuata in base alla peculierità ed alle dipendenze delle funzionalità che ricopre. Si possono così distinguere, ad esempio, package relativi a interfacce grafiche, logiche applicative o gestione della persistenza, in base alle caratteristiche specifiche del prodotto. Il diagramma dei package offre una prima struttura delle parti del progetto e del loro rapporto.

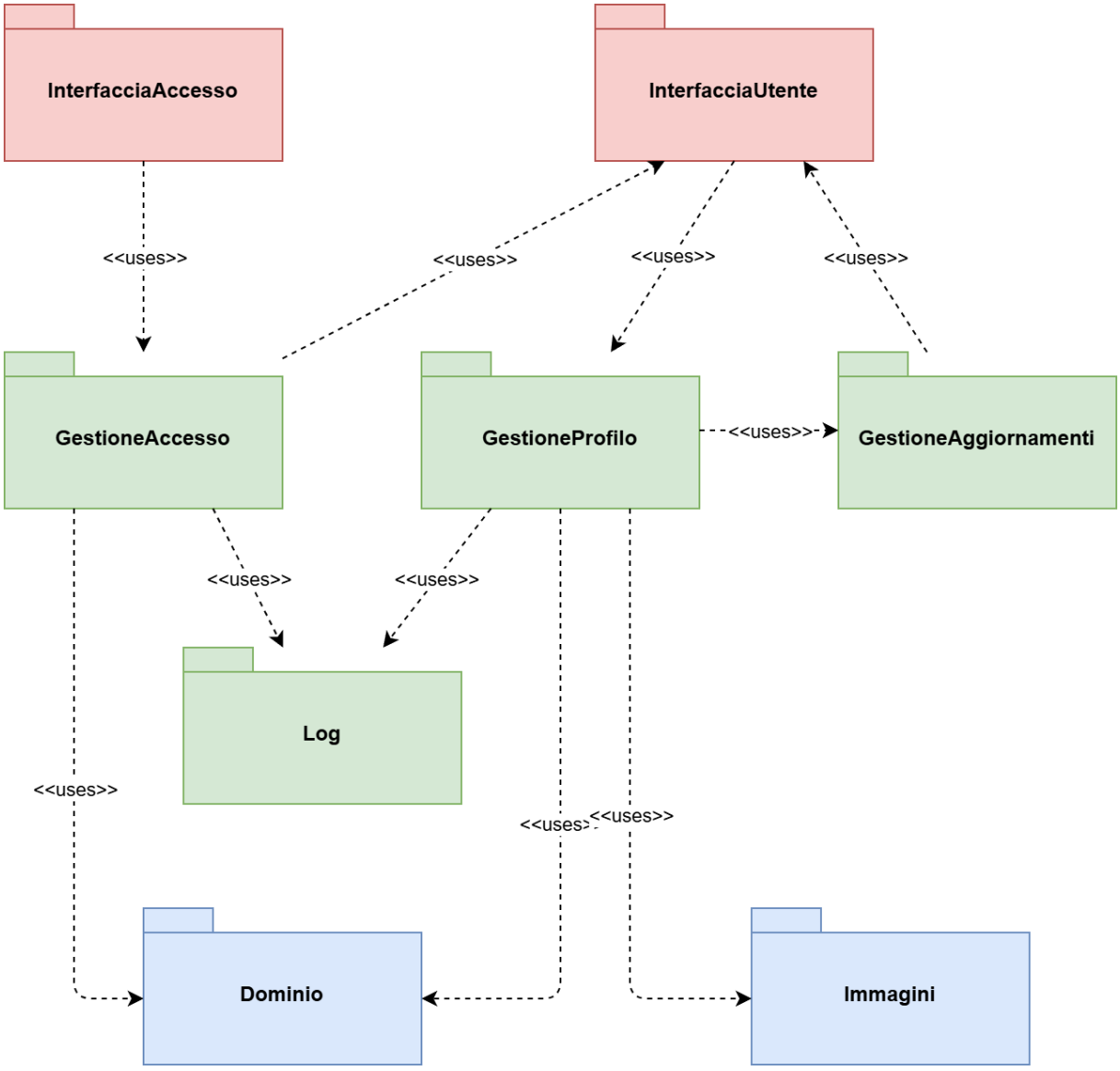

Figura 4: Diagramma dei Package

Ogni package contiene una o più classi che lo implementano. Ogni classe rappresenta un componente logico che assume uno specifico scopo. Le classi possono presentare dei metodi, ovvero delle istanze che descrivono le funzionalità fornite, alle quali altri componenti possono fare richiesta di esecuzione. La definizione delle classi permette di creare una struttura iniziale presentando le funzionalità minime e le dipendenze tra le parti.

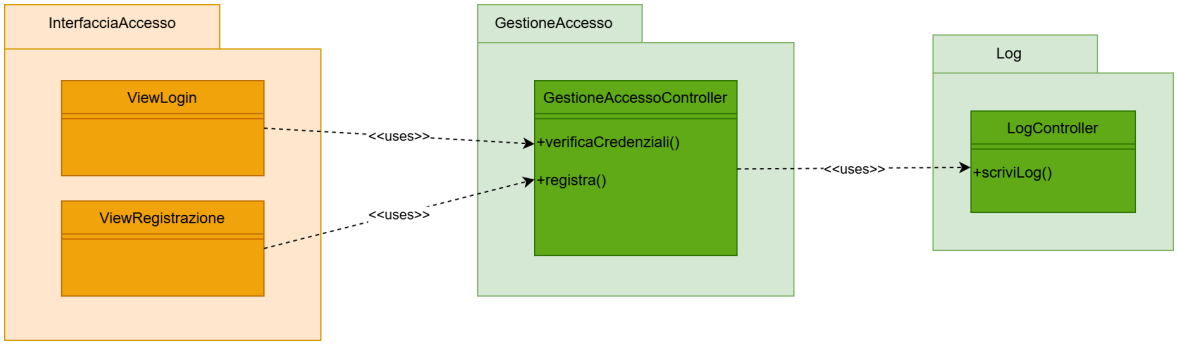

Figura 5: Diagramma delle classi: interfacca e gestione accesso

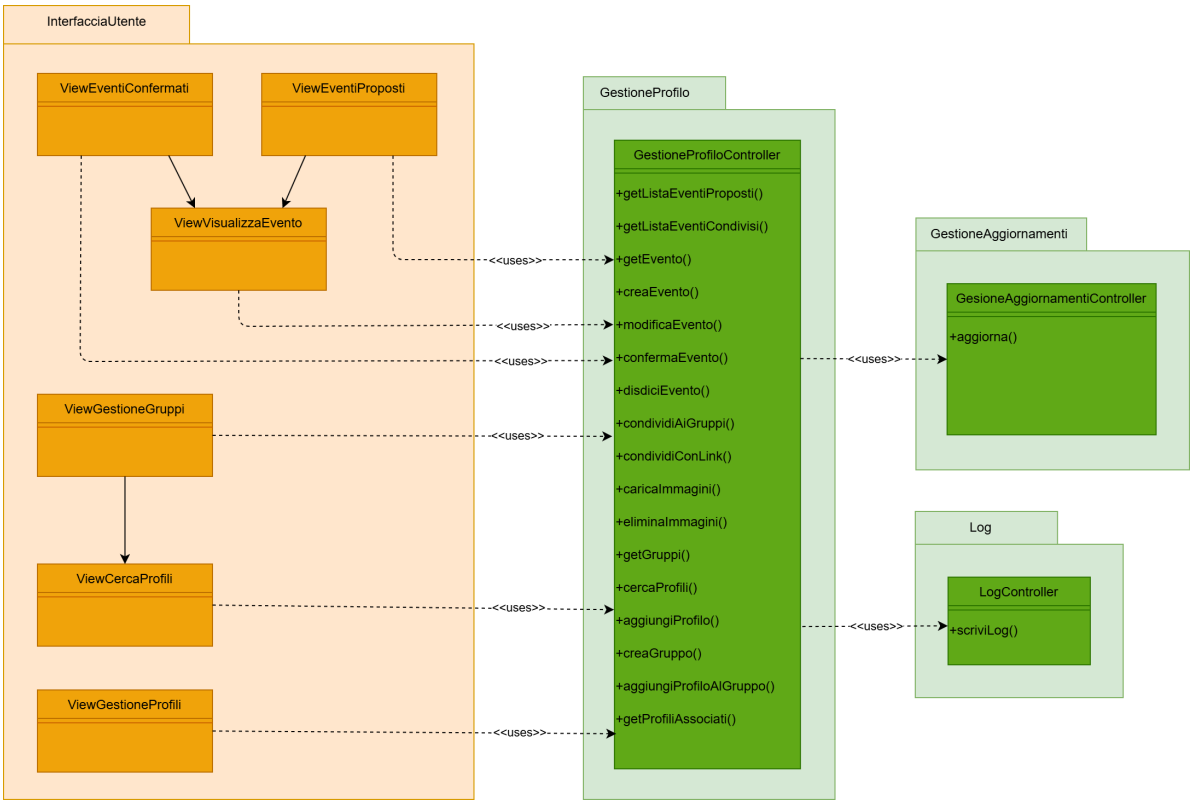

Figura 6: Diagramma delle classi: interfacca utente, gestione profilo ed aggiornamenti

2 Struttura centrale

Lo sviluppo di un'applicazione segue un processo strutturato, articolato in diverse fasi. In fase di progettazione, l'obiettivo principale è identificare le caratteristiche funzionali e comportamentali del sistema, delineando le componenti principali e le rispettive responsabilità. Questa fase permette di definire un'architettura coerente, facilitando le successive scelte tecnologiche e implementative.

Nella fase successiva, di implementazione, si procede con il passaggio dall'analisi teorica alla realizzazione concreta, dove la selezione dell'architettura e delle tecnologie di riferimento assumono un ruolo centrale. Le decisioni prese in questa fase determinano il comportamento dei diversi componenti e le modalità con cui essi interagiscono tra loro. Un'attenta selezione delle soluzioni ottimali, più adatte ai requisiti definiti in fase di progettazione, permette di impostare fin dalle prime iterazioni uno sviluppo efficiente e strutturato, minimizzando la necessità di revisioni successive.

Mentre alcune decisioni risultano immediate o intercambiabili, altre richiedono analisi approfondite per individuare la soluzione più adatta. Un approccio efficace consiste nello sviluppare inizialmente i componenti con requisiti ben definiti, per poi affinare progressivamente l'integrazione e la configurazione degli altri elementi del sistema. L'identificazione, anche parziale, di una struttura iniziale consente di delineare i vincoli di integrazione e di semplificare la definizione delle soluzioni residue.

L'architettura dell'applicativo si basa su una chiara suddivisione in componenti, ciascuno con un ruolo specifico all'interno del sistema. L'interfaccia grafica è responsabile della presentazione e dell'interazione con l'utente, ponendo particolare attenzione alla coerenza visiva e alla fluidità dell'esperienza.

La logica applicativa viene gestita da un server centrale, il quale si occupa di coordinare le comunicazioni tra i diversi servizi e di garantire il corretto flusso delle operazioni. La gestione dell'autenticazione degli utenti è delegata a un servizio apposito, che consente

di separare questa responsabilità dal resto del sistema, migliorando sia le prestazioni che la sicurezza.

Un ulteriore aspetto fondamentale nella progettazione del sistema riguarda la protezione delle comunicazioni e dei dati sensibili. L'adozione di misure di sicurezza adeguate è essenziale per garantire la protezione delle informazioni scambiate tra i vari componenti e per ridurre i rischi derivanti da eventuali attacchi esterni. Inoltre, per monitorare il corretto funzionamento dell'applicazione e identificare tempestivamente eventuali anomalie, il sistema è integrato con strumenti di logging e analisi delle prestazioni.

Questa organizzazione modulare consente di ottimizzare la scalabilità e la manutenibilità dell'applicativo, facilitando eventuali evoluzioni future. La suddivisione chiara delle responsabilità, unita a un'architettura flessibile e sicura, rappresenta quindi un elemento chiave per garantire la stabilità e l'efficienza del sistema nel lungo periodo.

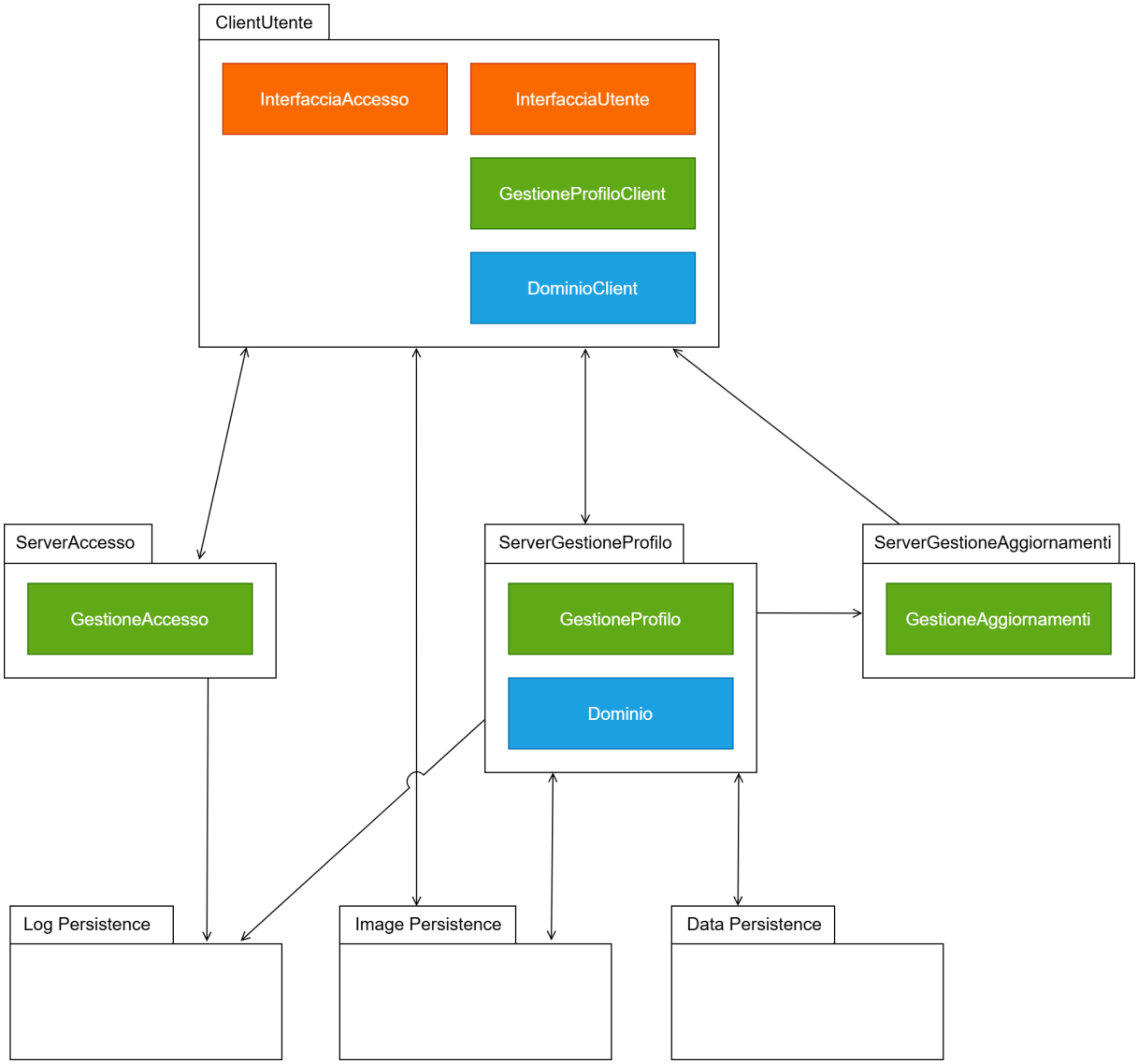

Figura 7: Struttura e responsabilità delle parti del progetto

2.1 Lo sviluppo del client

L'utilizzo delle applicazioni per la gestione degli eventi può essere suddiviso in due fasi distinte, ciascuna con specifiche esigenze funzionali.
La prima fase riguarda infatti la pianificazione a lungo termine e l'organizzazione degli impegni. In questa circostanza l'utente decide come distribuire il proprio tempo, pianificando attività e appuntamenti, e strutturando il proprio calendario nel modo più efficiente per le proprie necessità. La seconda fase riguarda invece la gestione degli eventi non ancora certi e definiti; ciò include l'invito a un evento, l'eventuale conferma da parte dell'utente, l'identificazione degli impegni a breve termine e l'aggiornamento del loro stato (ad esempio, se l'evento sia ancora confermato, quante persone vi partecipano, se qualcuno ha annullato o se l'evento è già concluso) con la gestione degli eventuali contenuti multimediali successivi all'evento. Queste due fasi implicano un approccio diverso da parte dell'utente, comportando di conseguenza esigenze differenti a cui l'applicazione deve rispondere adeguatamente.

Per rispondere a tali necessità, è fondamentale che l'applicazione offra un'interfaccia utente versatile, fruibile sia da desktop che da dispositivi mobili. La versione desktop consente una pianificazione a lungo termine, offrendo una visione d'insieme chiara e completa di tutti gli impegni, tale da facilitare la gestione del tempo. D'altra parte, la versione mobile deve permettere una gestione rapida e dinamica degli eventi quotidiani, garantendo che l'utente possa rimanere sempre connesso e aggiornato sugli sviluppi in tempo reale.

Inoltre, considerando che l'applicazione è destinata a un utilizzo diffuso e a un'utenza potenzialmente elevata, è necessario garantire tempi di risposta ridotti e una gestione efficiente delle richieste concorrenti. Ciò implica la progettazione di un sistema in grado di scalare facilmente, per supportare un ampio numero di utenti simultanei senza compromettere le prestazioni.

2.1.1 Il framework di sviluppo

Al fine di ottenere tutte le prestazioni precedentemente elencate, la scelta è ricaduta sull'adozione del framework di sviluppo Flutter. Diversi fattori motivano tale decisione.
In primo luogo, l'architettura di Flutter si basa su un motore grafico indipendente dalla piattaforma di esecuzione, il che consente di ottenere elevate prestazioni e garantire un'esperienza utente uniforme su dispositivi diversi.
In secondo luogo, Flutter adotta un approccio dichiarativo nella progettazione dell'interfaccia grafica, che facilita lo sviluppo di componenti reattivi attraverso un codice conciso, facilmente manutenibile.

Un ulteriore vantaggio di Flutter è rappresentato dalla crescente adozione nel settore, dalla solidità della community di sviluppo e dal supporto offerto da Google, che ne assicurano la stabilità, l'efficienza, la sicurezza e la disponibilità di componenti personalizzabili per l'intero ciclo di vita del prodotto. Infine, Flutter consente uno sviluppo rapido e interattivo grazie alla sua sintassi intuitiva e al meccanismo di hot reload, che riduce significativamente i tempi di compilazione e facilita il testing in tempo reale.

Tra le altre tecnologie valutate per lo sviluppo dell'interfaccia grafica, vi erano React Native e Xamarin. Tuttavia, entrambe presentano alcune limitazioni: le applicazioni finali sviluppate con React Native tendono ad avere dimensioni più elevate e le prestazioni risultano inferiori, in particolare nella gestione della memoria. Xamarin, pur essendo una valida opzione, presenta una curva di apprendimento più ripida e una comunità di sviluppatori ridotta rispetto a Flutter, con una conseguente minore disponibilità di componenti e librerie.

Per quanto Flutter consenta di uniformare l'esperienza utente su dispositivi diversi e semplifichi la compilazione per le varie piattaforme, alcune configurazioni rimangono comunque dipendenti dalla tecnologia su cui l'applicazione viene eseguita. Di conseguenza, ciascun eseguibile richiede una manutenzione aggiuntiva, inclusi gli aggiornamenti delle dipendenze specifiche, sia a livello di deployment che di gestione delle versioni.

In una fase iniziale dello sviluppo, nell'ottica di coprire il più ampio mercato possibile con il minor numero di piattaforme, si è deciso di sviluppare una versione fruibile via web e una per dispositivi Android.

2.1.2 L'implementazione

2.1.3 La distribuzione

Già consolidata e affidabile, sin dalle prime fasi di sviluppo del progetto è stata adottata la piattaforma cloud Azure, per garantire un'infrastruttura solida e scalabile.

Nell'ambito delle tecnologie offerte da Azure per la distribuzione del codice web, si è scelto Azure Static Web App, un servizio di hosting progettato per applicazioni web statiche.

La preferenza per questa soluzione è derivata dalla affidabilità dell'infrastruttura di Azure e dai bassi costi operativi per un utilizzo limitato, fattori che la rendono particolarmente vantaggiosa.

Questa selezione è stata resa possibile in quanto l'interfaccia sviluppata non prevede la creazione dinamica di contenuti, ovvero la pagina che viene restituita rimane invariata indipendentemente dall'utente che effettua la richiesta. I dati specifici dell'utente verranno infatti richiesti ad un server terzo (qualora non siano già presenti in una cache locale). Questa scelta consente di rendere l'interfaccia grafica completamente indipendente dall'identità dell'utente, migliorando così le prestazioni e riducendo il carico computazionale delegato a Azure Static Web App.

Inoltre, in attesa della pubblicazione dell'applicazione sull'App Store di Android, l'eseguibile è stato temporaneamente reso disponibile tramite Azure Storage Container. Questo servizio consente l'archiviazione e la distribuzione di file di varie tipologie, fornendo un link diretto per il recupero.

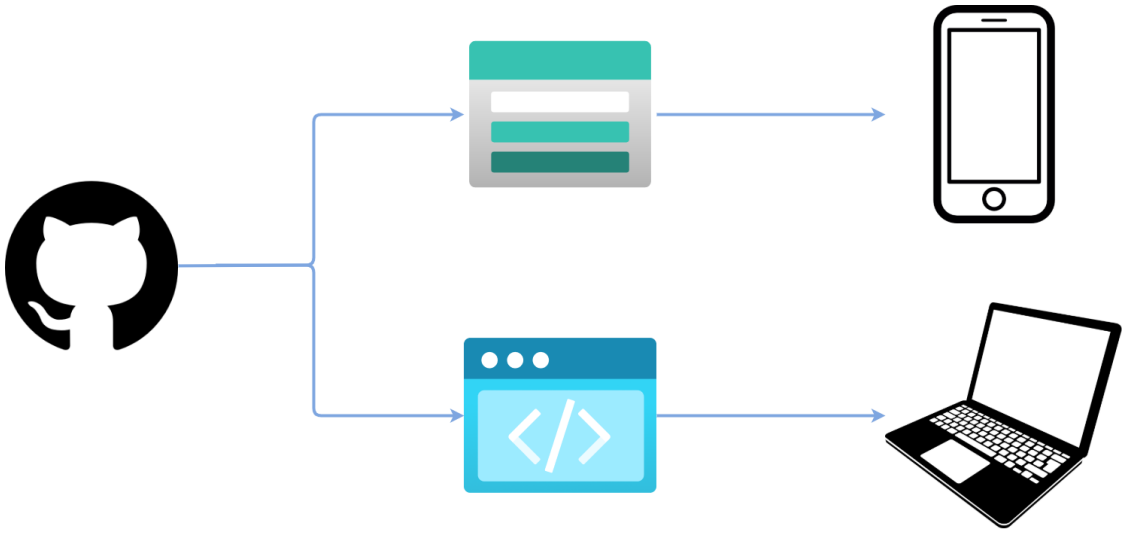

Figura 8: Diagramma di aggiornamento e distribuzione del client

2.1.4 L'aggiornamento

Per garantire un processo di aggiornamento efficiente e automatizzato, sia il codice distribuito sulla web app che l'applicativo ospitato nel container vengono gestiti tramite GitHub Actions, che ne assicura l'aggiornamento automatico a ogni nuova versione.

Nella fruizione tramite browser, il codice si aggiorna automaticamente a ogni accesso, mentre per l'applicativo su dispositivo mobile è necessaria una reinstallazione manuale. Per questa ragione, gli utenti dell'applicazione verranno notificati tempestivamente ogni volta che sarà disponibile una nuova versione.

2.2 L'architettura del server principale

Per poter essere in grado di gestire un numero elevato di richieste in tempi ridotti, l'applicazione deve garantire un'elevata scalabilità.

Capace di soddisfare al meglio questa esigenza è Azure Functions, un servizio serverless che consente di suddividere il codice in unità indipendenti, ciascuna eseguita in un ambiente di esecuzione unico per ogni richiesta. Questa caratteristica, combinata con la virtualizzazione dell'ambiente di esecuzione offerta dal cloud, consente una scalabilità potenzialmente illimitata.

L'indipendenza e la natura stateless di ogni funzione, ovvero la sua esecuzione svincolata dalle informazioni sullo stato o sulla sessione, sono responsabilità dello sviluppatore. Ogni funzione segue il principio di singola responsabilità, eseguendo un unico compito specifico. Tuttavia, nel caso di richieste che necessitino l'esecuzione coordinata di più funzioni, Azure Durable Functions fornisce una soluzione efficace.

Integrata all'interno delle Azure Functions, Azure Durable Function consente la creazione di una funzione orchestrator, incaricata di gestisce l'ordine, lo stato, il ciclo di vita e le risposte delle varie funzioni coinvolte nell'elaborazione della richiesta.

Mantenendo un'architettura indipendente e scalabile, questa soluzione consente di gestire efficacemente scenari in cui un'esecuzione sequenziale delle operazioni è cruciale, dove è necessario effettuare tentativi aggiuntivi in caso di errore o fallimento o si richiede l'attesa del completamento di operazioni con un tempo di esecuzione prolungato.

Tuttavia, l'architettura stateless e l'accoppiamento debole tra orchestrator e funzioni in esecuzione causano un tempo di risposta delle Azure Durable Functions più elevato. Per ottimizzare l'allocazione delle risorse, il sistema avvia perciò solo le funzioni strettamente necessarie all'esecuzione del compito determinato, stanziando al minimo il consumo computazionale.

2.2.1 La scelta del linguaggio

Come linguaggio di programmazione per lo sviluppo delle funzioni è stato utilizzato C#. La consapevolezza che sia l'ambiente di sviluppo di C#, ovvero il framework .Net, sia la piattaforma Azure siano entrambi sviluppati e mantenuti dalla stessa azienda, Microsoft, garantisce elevati livelli di stabilità, supporto e coordinamento delle tecnologie adottate.

L'integrazione con Entity Framework Core permette la mappatura direttamente in oggetti dei componenti del dominio, semplificando così la logica delle relazioni e astraendo le comunicazioni con il database. Grazie all'utilizzo delle proprietà virtuali degli oggetti si può applicare il lazy loading, riducendo il numero di richieste al database solo a quando esse sono strettamente necessarie, mantenendo in codice a livello logico ed ottimizzandone le prestazioni.

2.2.2 Lo sviluppo

Lo sviluppo è stato condotto utilizzando Visual Studio Code, piattaforma che, grazie ad apposite estensioni, consente un collegamento diretto ai servizi cloud Azure, semplificando il processo di aggiornamento del codice, rendendolo estremamente lineare ed immediato.

Azure Function in ambiente .Net supporta due modelli di esecuzione e sviluppo: in-process worker o isolated worker. Il worker è il processo all'interno dell'applicativo che gestisce la creazione delle risorse e l'esecuzione delle funzioni in risposta alle richieste. Nella modalità in-process, la funzione viene eseguita all'interno dello stesso processo del worker che l'ha generata, riducendo la quantità di allocazione delle risorse necessarie ma condividendo l'ambiente di esecuzione. Nel modello isolated, invece, ogni funzione viene eseguita attraverso un processo indipendente dedicato, garantendo maggiore isolamento e quindi riducendo le possibili dipendenze tra le funzioni.

Inoltre, il modello isolated worker offre ulteriori vantaggi, grazie al maggiore supporto

fornito: innanzitutto esso prevede una maggiore compatibilità, grazie al più ampio numero di versioni del framework .Net, a differenza del modello in-process, limitato alle sole versioni con supporto a lungo termine. In secondo luogo, il supporto per la creazione di middleware personalizzati permette l'elaborazione di un codice intermedio tra la chiamata e l'esecuzione della funzione, funzionalità invece non disponibile nel modello in process. Considerati questi vantaggi, le funzioni sono state sviluppate utilizzando il modello isolated worker per garantire maggiore flessibilità, compatibilità e modularità dell'architettura.

2.3 Autenticazione

Poiché la modalità di autenticazione rappresenta un elemento cruciale per l'esperienza dell'utente, in quanto deve assicurare un accesso sicuro all'applicazione mantenendone la semplicità, la facilità del processo autenticazione deve essere garantita. Offrire agli utenti la possibilità di autenticarsi tramite il proprio authentication provider di fiducia migliora sicuramente l'usabilità e l'apprezzamento degli utenti, ma è altrettanto essenziale consentire la possibilità di creare account dedicati esclusivamente all'applicazione. 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 ricaduta su Firebase Authentication, che garantisce sia la possibilità di creare account dedicati che di collegarsi attraverso altri authentication providers. Inoltre, la piattaforma presenta un'interfaccia chiara e offre servizi di integrazione di facile utilizzo sia tramite Flutter che tramite C#. Dal punto di vista economico, il servizio risulta vantaggioso, essendo gratuito fino ai cinquantamila utenti mensili attivi.

Uno dei requisiti fondamentali del progetto prevede che ogni account sia associato in modo univoco a un singolo utente. Durante la fase di creazione del profilo, 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.

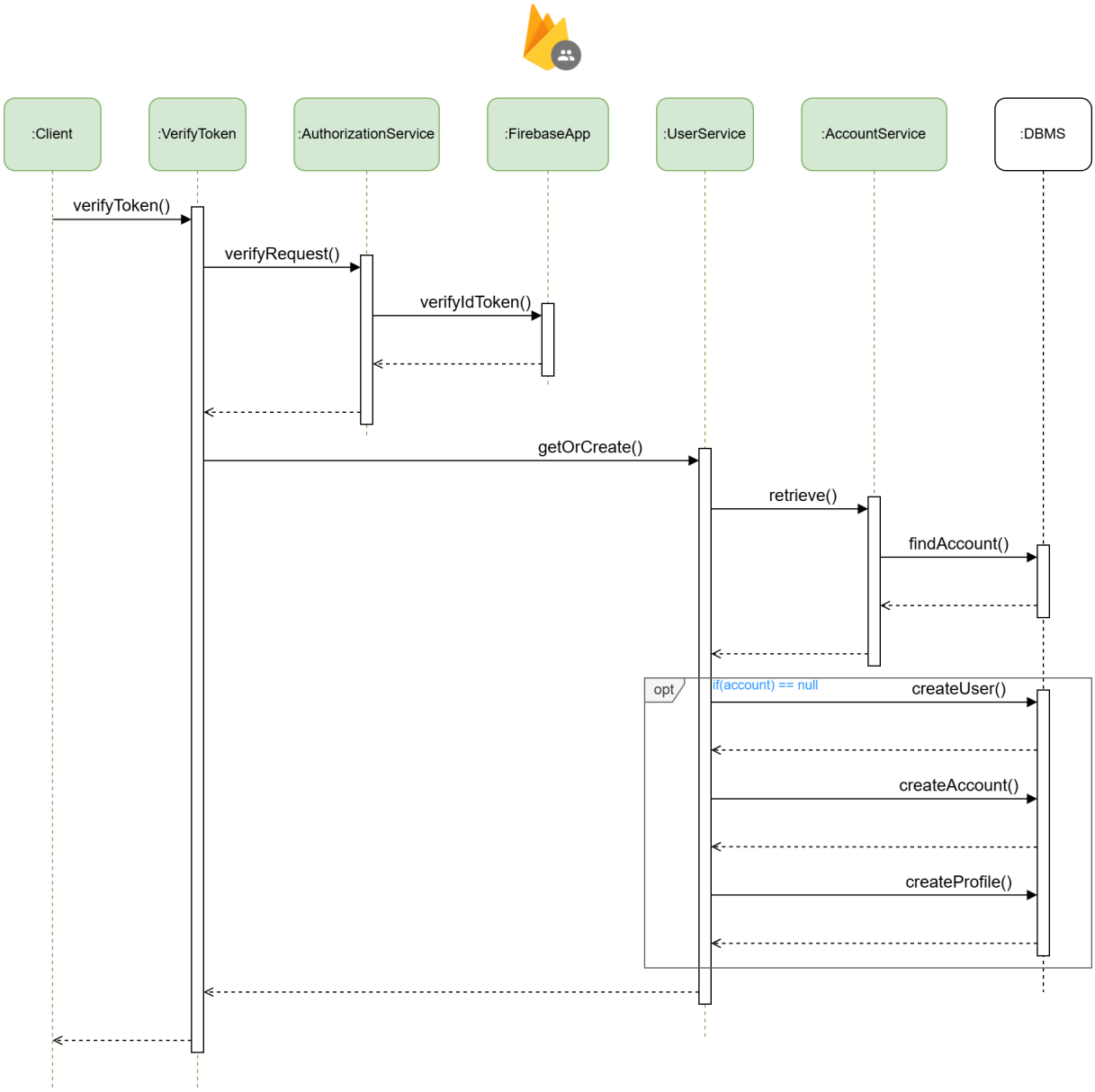

Figura 9: Diagramma di sequenza per la creazione di un account

Per garantire un processo di autenticazione sicuro ed efficiente, a ogni richiesta che richiede identificazione, il dispositivo utente aggiunge anche il token di autenticazione, salvato durante il primo accesso e la verifica iniziale. Il server verifica quindi la validità del token prima di procedere con l'esecuzione della richiesta.

2.4 La sicurezza

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. Quando necessario i servizi, in particolare le Azure Function, contatteranno il Key Vault per l'ottenimento delle chiavi necessarie, riducendo il rischio di un'intercettazione data magari da un errore durante lo 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 ad uno standard collaudato. In particolare, le comunicazioni tra i client e Azure Function, 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.

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 caricamento delle immagini avviene grazie ad un link univoco dato dalla combinazione degli identificativi dell'evento e dell'immagine. Utilizzando i codici di hash diventa molto complicato il ritrovamento delle immagini senza essere a conoscenza dei codici, che non avendo natura incrementale ma distribuita rende indovinare un link valido.

TODO Dos e limite alle dimensioni delle richieste

2.5 Monitoraggio

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

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

Per monitorare le Azure Functions sarà invece necessario collegare Azure Application Insights, servizio che provvede a controllare il funzionamento e la risposta del servizio. Una volta unito il servizio, infatti, Azure 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.

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. Tali log saranno poi consultabili e analizzabili tramite l'interfaccia fornita da Azure Application Insight.