



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

Indice

1	Introduzione	1
2	Abstract	2
3	Documento dei Requisiti	3
3.1	Raccolta dei requisiti	3
3.2	Tabella dei Requisiti	4
3.3	Analisi dei Requisiti	5
3.3.1	Vocabolario	5
3.3.2	Sistemi esterni	5
3.3.3	Casi d'uso	6
3.3.4	Scenari	7
3.4	Analisi del Rischio	18
3.4.1	Tabella Valutazione dei Beni	18
3.4.2	Tabella Minacce/Controlli	18
3.4.3	Analisi Tecnologica della Sicurezza	19
3.4.4	Security Use Case & Misuse Case	20
3.4.5	Security Use Case & Misuse Case Scenari	21
3.4.6	Requisiti di Protezione dei Dati	23
4	Analisi del Problema	24
4.1	Analisi Documento dei Requisiti: Analisi delle Funzionalità	24
4.1.1	Analisi Documento dei Requisiti: Analisi dei Vincoli	28
4.1.2	Analisi Documento dei Requisiti: Analisi delle Interazioni	29
4.1.3	Analisi Ruoli e Responsabilità	29
4.1.4	Scomposizione del Problema	30
4.1.5	Creazione Modello del Dominio	31
4.1.6	Architettura Logica: Struttura	32
4.1.7	Architettura Logica: Interazione	34
4.1.8	Piano di Lavoro	39
4.1.9	Sviluppi Futuri	39
5	Progettazione	40
5.1	Progettazione Architetturale	40
5.1.1	Requisiti non funzionali	40
5.1.2	Scelte tecnologiche	40
5.1.3	Scelta dell'architettura	41
5.1.4	Pattern architettonici e di design	41
5.2	Progettazione di dettaglio	43
5.2.1	Struttura: Dominio	43
5.2.2	Struttura: Interfacce	45
5.2.3	Struttura: Controller	46
5.2.4	Struttura: Interfacce Grafiche	47
5.3	Interazione	54
5.4	Progettazione della persistenza	60
5.4.1	Diagramma E-R	60

5.4.2	Formato dei file di log	61
5.5	Deployment	62
5.5.1	Artefatti	62
5.5.2	Deployment Type-Level	62
6	Implementazione	63
6.1	Progettazione Architetturale	63
6.1.1	Requisiti non funzionali	63
6.1.2	Scelte tecnologiche	63
6.1.3	Scelta dell'architettura	64
6.2	Monitoraggio	65
6.3	Sicurezza	65
6.4	Progettazione di dettaglio	66
6.4.1	Struttura: Dominio	66
6.4.2	Struttura: Controller - Client	69
6.4.3	Struttura: Controller - Server	71
6.5	Interazione	73
6.6	Deployment e Aggiornamento del Codice	82
6.6.1	Deployment Type-Level: Client	82
6.6.2	Deployment Type-Level: Server	82
6.7	Testing	83
6.7.1	AccountService	83
6.7.2	ProfileService	86
6.7.3	UserService	90
6.7.4	EventService	95
7	Performances	102
8	Costi	102

1 Introduzione

L'idea per il progetto di questa tesi nasce come risposta a un problema sempre più attuale in un mondo sempre più connesso. La molteplicità di contatti, la velocità delle comunicazioni e l'accesso universale alle notizie rendono la creazione, l'organizzazione e la partecipazione ad eventi estremamente semplice ma al contempo frenetico. Si fa fatica a seguire a tutte le occasioni a cui potremmo prendere parte. Pensiamo alle riunioni di lavoro, alle serate tra amici, agli appuntamenti per un caffè. Ma anche a una fiera, una convention aziendale, ad un concerto, alla partita di calcio o alla mostra dell'artista che ci è sempre piaciuto e che passa per una volta nella città vicina. Queste occasioni spesso si accavallano o si finisce per dimenticarsene, potenzialmente creando delusione e/o frustrazione.

Quando condividiamo un evento, a volte siamo noi a proporre, altre volte ci invitano.

Quando ci invitano, spesso magari abbiamo già un altro impegno, o magari un invito di un altro contatto a cui dobbiamo ancora dare conferma. E sul momento magari non ci si ricorda, si conferma per poi dover, purtroppo, disdire l'evento sovrapposto.

Quando invece siamo noi a proporre, potremmo trovarci nella difficoltà di trovare un evento da proporre, scrutando centinaia di profili social di tutti i locali di cui abbiamo sentito parlare nella speranza che propongano qualcosa, oppure non sappiamo se l'altra persona possa essere libera o meno. Questo problema si ripresenta ancora più grave nell'organizzazione di gruppo, in cui bisogna riuscire a far combaciare gli impegni di tre, quattro o più persone.

Ecco quindi l'opportunità di creare uno strumento che permetta di semplificare la proposta e la gestione degli eventi, separando la proposta dalla conferma, per dare modo di valutare l'effettiva disponibilità ma anche rendere più facile un'invito a partecipare. Allo stesso modo si può semplificare la ricerca di occasioni, creando uno spazio unico virtuale in cui pubblicare e consultare gli eventi.

Alla base della funzionalità sussiste l'idea di affiancare alla classica agenda degli impegni presi (e quindi confermati) un'altro calendario in cui sono presenti tutti gli eventi a cui si potrebbe partecipare. La conferma di un evento lo sposterà all'interno dell'agenda.

Gli eventi creati potranno essere condivisi a persone o gruppi di persone, e sarà possibile vedere chi conferma la sua presenza.

Inoltre, nell'epoca delle immagini e della condivisione, si prevede la possibilità di condividere le proprie foto con chiunque abbia partecipato all'evento.

La realizzazione di questo tipo di programma prevede particolarità che incrociano tante necessità diverse. In primis la persistenza dell'agenda dell'utente, che deve essere mantenuta e aggiornata per garantire affidabilità e coerenza per un uso distribuito del servizio. Si aggiunge poi l'aspetto della condivisione degli eventi, che vede necessario l'aggiornare tutti gli attori interessati dalle modifiche apportate. Infine, il caricamento e salvataggio delle foto introduce la gestione di richieste e di memoria di dimensioni importanti.

Tramite l'analisi dei requisiti e lo sviluppo del progetto verranno evidenziate le scelte tecnologiche che hanno portato all'implementazione di una applicazione efficace, affidabile e scalabile.

2 Abstract

Wyd è un'applicazione che permette ai clienti di organizzare i propri impegni, siano essi confermati oppure proposti.

Mette a disposizione due calendari, il primo con gli eventi in cui l'utente è convinto di partecipare, il secondo in cui vengono riuniti gli eventi a cui l'utente è stato invitato ma senza aver ancora dato conferma.

L'utente ha la possibilità di creare, modificare, confermare o disdire un evento, ma anche condividerlo con altri o allegarci foto. La condivisione di un evento può avvenire con applicazioni esterne tramite la generazione di un link o grazie all'ausilio di gruppi di profili. Inoltre, al termine di un evento, l'applicazione carica automaticamente le foto scattate durante l'evento, per allegarle a seguito della conferma dell'utente.

L'utente può infatti cercare altri profili e creare gruppi con i profili trovati.

Tutta l'interazione avviene tramite l'utilizzo di profili, che permettono di suddividere semanticamente gli eventi e le relazioni.

3 Documento dei Requisiti

3.1 Raccolta dei requisiti

- Per interagire con l'applicazione è necessario avere un utente registrato.
- Per registrarsi l'utente deve inserire una mail univoca, e una password lunga almeno 6 caratteri.
- L'utente ha a disposizione un'agenda con gli eventi confermati e una con gli eventi proposti.
- L'utente può creare un evento, definendo, al minimo, la data di inizio e di fine.
- La data di fine deve essere successiva alla data di inizio.
- L'utente può modificare un evento.
- L'utente può confermare un evento condiviso, o disdire un evento confermato.
- L'utente può condividere l'evento ad altri profili tramite link o condividendo a un gruppo o ad altri profili singoli.
- L'utente può caricare le foto relative ad un evento, con visibilità condivisa.
- Alla scadenza dell'evento, se su dispositivo mobile, l'applicazione controlla le foto scattate durante l'evento. Se ce ne sono, l'utente verrà notificato e potrà eliminare o confermare le foto, che verranno quindi caricate.
- L'utente può cercare altri profili e creare un'associazione tra il profilo cercato e quello che sta usando.
- I profili possono avere associazioni tra loro, a coppie o come gruppi.
- L'utente può avere più profili, vedere in contemporanea gli eventi di tutti i profili associati ma effettuare le azioni a nome di uno solo.
- La conferma/disdetta, il caricamento delle foto e la modifica di un evento deve avvenire in tempo reale se i profili condivisi sono online, altrimenti ricevono gli aggiornamenti all'avvio dell'applicazione.
- L'applicazione deve essere intuitiva e con brevi tempi di risposta.
- L'applicazione deve poter funzionare con un elevato numero di richieste concorrenti.

3.2 Tabella dei Requisiti

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

3.3 Analisi dei Requisiti

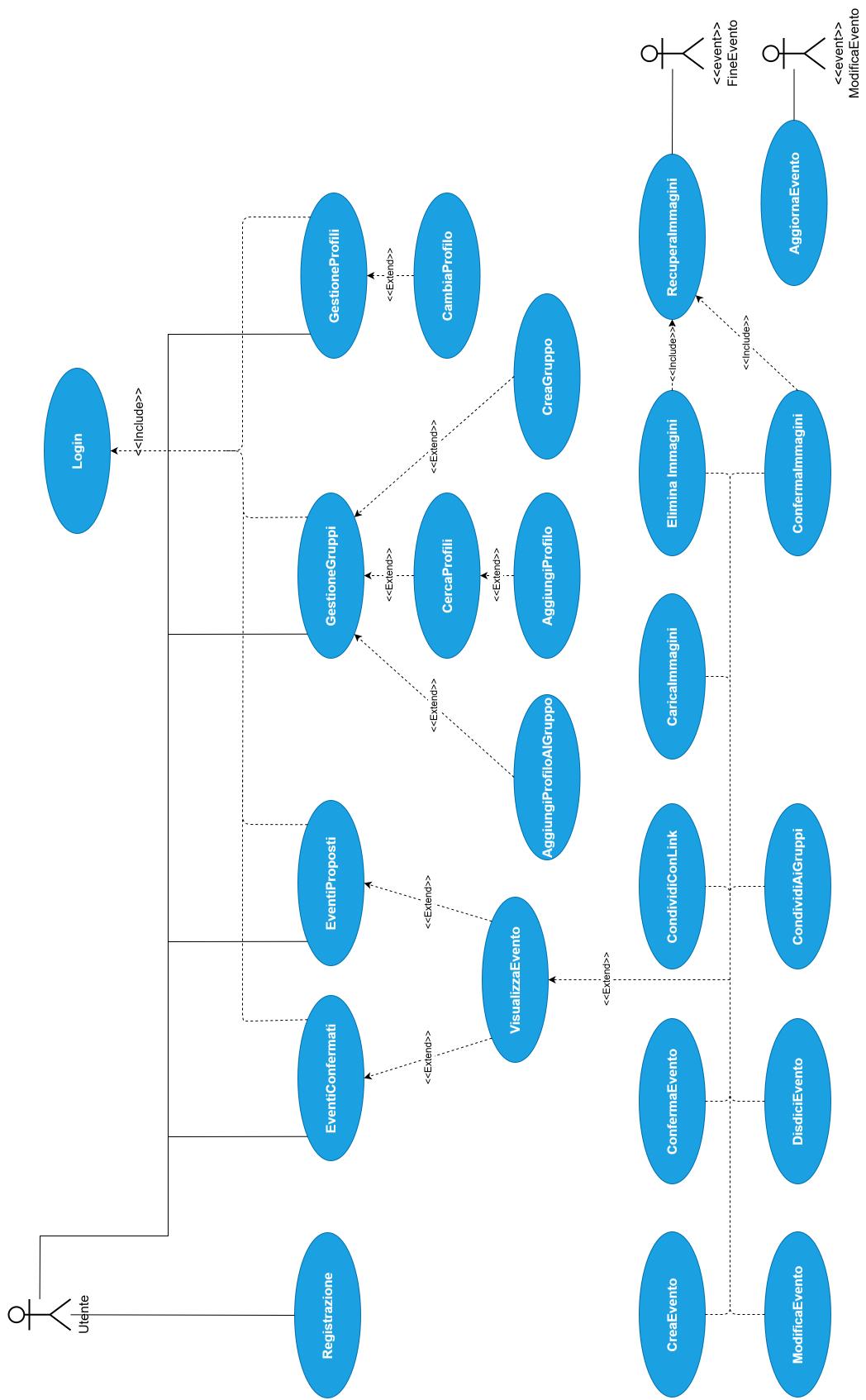
3.3.1 Vocabolario

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	

3.3.2 Sistemi esterni

Il sistema non dovrà interfacciarsi con sistemi esterni.

3.3.3 Casi d'uso



3.3.4 Scenari

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	<ol style="list-style-type: none"> 1. L'utente accede alla sezione di registrazione 2. L'utente inserisce email e password 3. L'utente termina la registrazione, se avvenuta con successo viene reindirizzato alla pagina principale
Scenari Alternativi	3. 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	

Titolo	Login
Descrizione	Permette di accedere al sistema
Attori	Utente
Relazioni	EventiConfermati, EventiProposti, GestioneGruppi, GestioneProfili
Precondizioni	
Postcondizioni	L'utente ha accesso al sistema, limitato in base ai suoi privilegi
Scenario principale	<ol style="list-style-type: none"> 1. L'utente inserisce le credenziali di accesso 2. Il sistema verifica le credenziali 3. Se le credenziali sono corrette, viene presentata la schermata iniziale
Scenari Alternativi	<ol style="list-style-type: none"> 1. L'utente inserisce le credenziali di accesso 2. Il sistema verifica le credenziali 3. Il sistema non riconosce le credenziali e rispedisce l'utente alla schermata di login con un messaggio di errore
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	EventiConfermati
Descrizione	Viene mostrato l'elenco degli eventi confermati dall'utente
Attori	Utente
Relazioni	Login, VisualizzaEvento
Precondizioni	
Postcondizioni	Viene mostrato l'elenco degli eventi confermati
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente va nella schermata di visualizzazione eventi confermati 2. Il sistema recupera l'elenco degli eventi confermati 3. Il sistema mostra a video l'elenco richiesto
Scenari Alternativi	
Requisiti non funzionali	Velocità di richiesta iniziale dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	EventiProposti
Descrizione	Viene mostrato l'elenco degli eventi proposti non confermati dall'utente
Attori	Utente
Relazioni	Login, VisualizzaEvento
Precondizioni	
Postcondizioni	Viene mostrato l'elenco degli eventi proposti non confermati
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente va nella schermata di visualizzazione eventi proposti 2. Il sistema recupera l'elenco degli eventi proposti non confermati 3. Il sistema mostra a video l'elenco richiesto
Scenari Alternativi	
Requisiti non funzionali	Velocità di richiesta iniziale dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	VisualizzaEvento
Descrizione	Viene mostrato l'evento con i suoi dettagli, con la possibilità di modificarli
Attori	Utente
Relazioni	EventiConfermati, EventiProposti, CreaEvento, ModificaEvento, ConfermaEvento, DisdiciEvento, CondividiConLink, CondividiAiGruppi, CaricaImmagini, EliminaImmagini, ConfermaImmagini
Precondizioni	
Postcondizioni	Viene mostrato l'evento e i suoi dettagli, le modifiche vengono temporaneamente salvate
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona un evento 2. Il sistema recupera i dati dell'evento 3. Il sistema mostra a video i dati dell'evento da la possibilità di modificare i dati dell'evento
Scenari Alternativi	<p>Scenario alternativo A:</p> <ol style="list-style-type: none"> 1. L'utente seleziona l'opzione di creare un nuovo evento 2. Il sistema mostra a video i dati dell'evento da la possibilità di modificare i dati dell'evento <p>Scenario alternativo B:</p> <ol style="list-style-type: none"> 1. L'utente viene indirizzato tramite link 2. Il sistema recupera i dati dell'evento 3. Il sistema mostra a video i dati dell'evento da la possibilità di modificare i dati dell'evento
Requisiti non funzionali	Velocità in lettura e scrittura dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	CreaEvento
Descrizione	Crea un evento e lo aggiunge
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento non esiste, i dati inseriti sono corretti
Postcondizioni	Viene creato l'evento e visualizzato nella pagina relativa
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. Il sistema controlla che i dati inseriti siano corretti 3. Se i dati sono corretti, l'evento viene salvato 4. L'evento è visualizzato nella schermata degli eventi 5. Tutti i dispositivi collegati al profilo visualizzano l'evento
Scenari Alternativi	<ol style="list-style-type: none"> 3. Se i dati risultano sbagliati, il sistema notifica l'utente indicando l'errore
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

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	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. Il sistema controlla che i dati modificati siano corretti 3. Le immagini vengono salvate 4. Tutti i dispositivi collegati ai profili collegati all'evento visualizzano le immagini
Scenari Alternativi	<ol style="list-style-type: none"> 3. Se i dati risultano sbagliati, il sistema notifica l'utente indicando l'errore
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	ConfermaEvento
Descrizione	Conferma la partecipazione ad un evento
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste e il profilo corrente non lo ha confermato
Postcondizioni	Il profilo conferma la sua presenza, tutti i profili collegati vengono aggiornati, l'evento è visualizzato tra gli eventi confermati
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. L'utente conferma la sua presenza 3. L'evento è visualizzato tra gli eventi confermati 4. Tutti i dispositivi collegati ai profili collegati all'evento visualizzano l'aggiornamento
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	DisdiciEvento
Descrizione	Disdice la partecipazione ad un evento
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste e il profilo corrente lo ha confermato
Postcondizioni	Il profilo disdice la sua presenza, tutti i profili collegati vengono aggiornati, l'evento è visualizzato tra gli eventi proposti
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. L'utente disdice la sua presenza 3. L'evento è visualizzato tra gli eventi proposti 4. Tutti i dispositivi collegati ai profili collegati all'evento visualizzano l'aggiornamento
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

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	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. L'utente seleziona le immagini che vuole caricare 3. Le immagini vengono salvate 4. Tutti i dispositivi collegati ai profili collegati all'evento visualizzano le immagini
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	RecuperaImmagini
Descrizione	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
Postcondizioni	Le immagini vengono salvate in locale e l'utente viene notificato
Scenario Principale	<ol style="list-style-type: none"> 1. Il sistema controlla che l'evento sia finito 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	

Titolo	EliminaImmagini
Descrizione	Rimuove le immagini dall'evento
Attori	Utente
Relazioni	RecuperaImmagini, VisualizzaEvento
Precondizioni	L'evento esiste ed esistono immagini collegate
Postcondizioni	Le immagini selezionate vengono rimosse dall'evento
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 1. L'utente seleziona le immagini caricate automaticamente da eliminare 1. Le immagini vengono rimosse dall'evento
Scenari Alternativi	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. L'utente seleziona le immagini da eliminare 3. Le immagini vengono rimosse dall'evento, e le modifiche propagate ai profili collegati
Requisiti non funzionali	
Punti aperti	

Titolo	ConfermaImmagini
Descrizione	L'utente conferma le immagini caricate automaticamente
Attori	Utente
Relazioni	RecuperaImmagini, VisualizzaEvento
Precondizioni	L'evento esiste ed esistono immagini caricate automaticamente
Postcondizioni	Le immagini selezionate vengono condivise con l'evento
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. L'utente seleziona conferma le immagini caricate automaticamente 3. Le immagini vengono aggiunte all'evento e tutti i profili collegati visualizzano le modifiche
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	CondividiAiGruppi
Descrizione	Permette di condividere l'evento ai gruppi
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste
Postcondizioni	L'evento è condiviso con tutti i profili appartenenti ai gruppi selezionati
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. Il sistema visualizza l'elenco dei gruppi di cui l'utente fa parte, permettendone la selezione 3. L'evento è condiviso con tutti i profili dei gruppi selezionati, che visualizzeranno l'evento tra i proposti
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	CondividiConLink
Descrizione	Permette di condividere l'evento tramite link
Attori	Utente
Relazioni	VisualizzaEvento
Precondizioni	L'evento esiste
Postcondizioni	L'utente ottiene un link che può condividere
Scenario Principale	<ol style="list-style-type: none"> 1. VisualizzaEvento 2. Il sistema mostra le opzioni di condivisione del link
Scenari Alternativi	<ol style="list-style-type: none"> 2. Il sistema salva il link in memoria temporanea
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	AggiornaEvento
Descrizione	Aggiorna l'evento in locale in base alle modifiche apportate da profili esterni
Attori	ModificaEvento
Relazioni	
Precondizioni	L'evento esiste ed è condiviso con uno dei profili collegati all'utente
Postcondizioni	L'evento viene aggiornato con le modifiche
Scenario Principale	<ol style="list-style-type: none"> 1. Il sistema riceve la notifica che un evento è stato modificato 2. Il sistema recupera le modifiche e aggiorna l'evento di conseguenza
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	GestioneGruppi
Descrizione	Viene mostrato l'elenco dei gruppi appartenenti al profilo corrente
Attori	Utente
Relazioni	Login, CercaProfili, CreaGruppo
Precondizioni	
Postcondizioni	Viene mostrato l'elenco degli gruppi appartenenti al profilo corrente
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente va nella schermata di gestione gruppi 2. Il sistema recupera l'elenco dei gruppi appartenenti al profilo corrente 3. Il sistema mostra a video l'elenco richiesto
Scenari Alternativi	
Requisiti non funzionali	Velocità di richiesta iniziale dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	CercaProfili
Descrizione	L'utente cerca i profili tramite tag
Attori	Utente
Relazioni	GestioneGruppi, AggiungiProfilo
Precondizioni	
Postcondizioni	Si visualizza la lista dei profili con tag corrispondente
Scenario principale	<ol style="list-style-type: none"> 1. GestioneGruppi 2. L'utente inserisce il tag parziale o completo del profilo per cui eseguire la ricerca 3. Il sistema ottiene la lista dei profili che corrispondono alla ricerca 4. La lista viene mostrata all'utente
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	AggiungiProfilo
Descrizione	si aggiunge il profilo selezionato alla lista gruppi del profilo
Attori	Utente
Relazioni	CercaProfili
Precondizioni	Il profilo selezionato esiste
Postcondizioni	Il profilo selezionato è visibile tra la lista dei gruppi
Scenario principale	<ol style="list-style-type: none"> 1. CercaProfili 2. L'utente seleziona il profilo da aggiungere 3. Il profilo viene aggiunto nella lista dei gruppi
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	CreaGruppo
Descrizione	L'utente crea un gruppo
Attori	Utente
Relazioni	GestioneGruppi, AggiungiProfiloAlGruppo
Precondizioni	
Postcondizioni	Il gruppo è creato ed aggiunto alla lista dei gruppi di tutti i profili interessati
Scenario principale	<ol style="list-style-type: none"> 1. GestioneGruppi 2. L'utente inserisce il nome ed eventualmente i profili interessati 3. Il sistema crea il gruppo e lo aggiunge alla lista dei gruppi di tutti i profili interessati
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	AggiungiProfiloAlGruppo
Descrizione	si aggiunge il profilo selezionato alla lista gruppi del profilo
Attori	Utente
Relazioni	GestioneGruppi
Precondizioni	Il profilo selezionato esiste
Postcondizioni	Il profilo selezionato è tra la lista i profili del gruppo
Scenario principale	<ol style="list-style-type: none"> 1. GestioneGruppi 2. L'utente seleziona il profilo da aggiungere 3. Il sistema aggiunge il profilo alla lista dei profili del gruppo 4. Il profilo aggiunto visualizza il gruppo nella sua lista dei gruppi
Scenari Alternativi	
Requisiti non funzionali	Velocità in lettura e scrittura dei dati
Punti aperti	

Titolo	GestioneProfili
Descrizione	Viene mostrato l'elenco dei profili collegati all'utente
Attori	Utente
Relazioni	Login, CambiaProfilo
Precondizioni	
Postcondizioni	Viene mostrato l'elenco degli profili collegati all'utente
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente va nella schermata di gestione profili 2. Il sistema recupera l'elenco dei profili collegati all'utente 3. Il sistema mostra a video l'elenco richiesto
Scenari Alternativi	
Requisiti non funzionali	Velocità di richiesta iniziale dei dati Semplicità e fluidità dell'interfaccia grafica
Punti aperti	

Titolo	CambiaProfilo
Descrizione	Modifica il profilo corrente con quello selezionato
Attori	Utente
Relazioni	GestioneProfili
Precondizioni	Il profilo selezionabile non è quello attualmente in uso
Postcondizioni	Il profilo corrente è quello che è stato selezionato
Scenario Principale	<ol style="list-style-type: none"> 1. GestioneProfili 2. L'utente seleziona il profilo 3. Il profilo corrente diventa quello selezionato
Scenari Alternativi	
Requisiti non funzionali	
Punti aperti	

3.4 Analisi del Rischio

3.4.1 Tabella Valutazione dei Beni

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

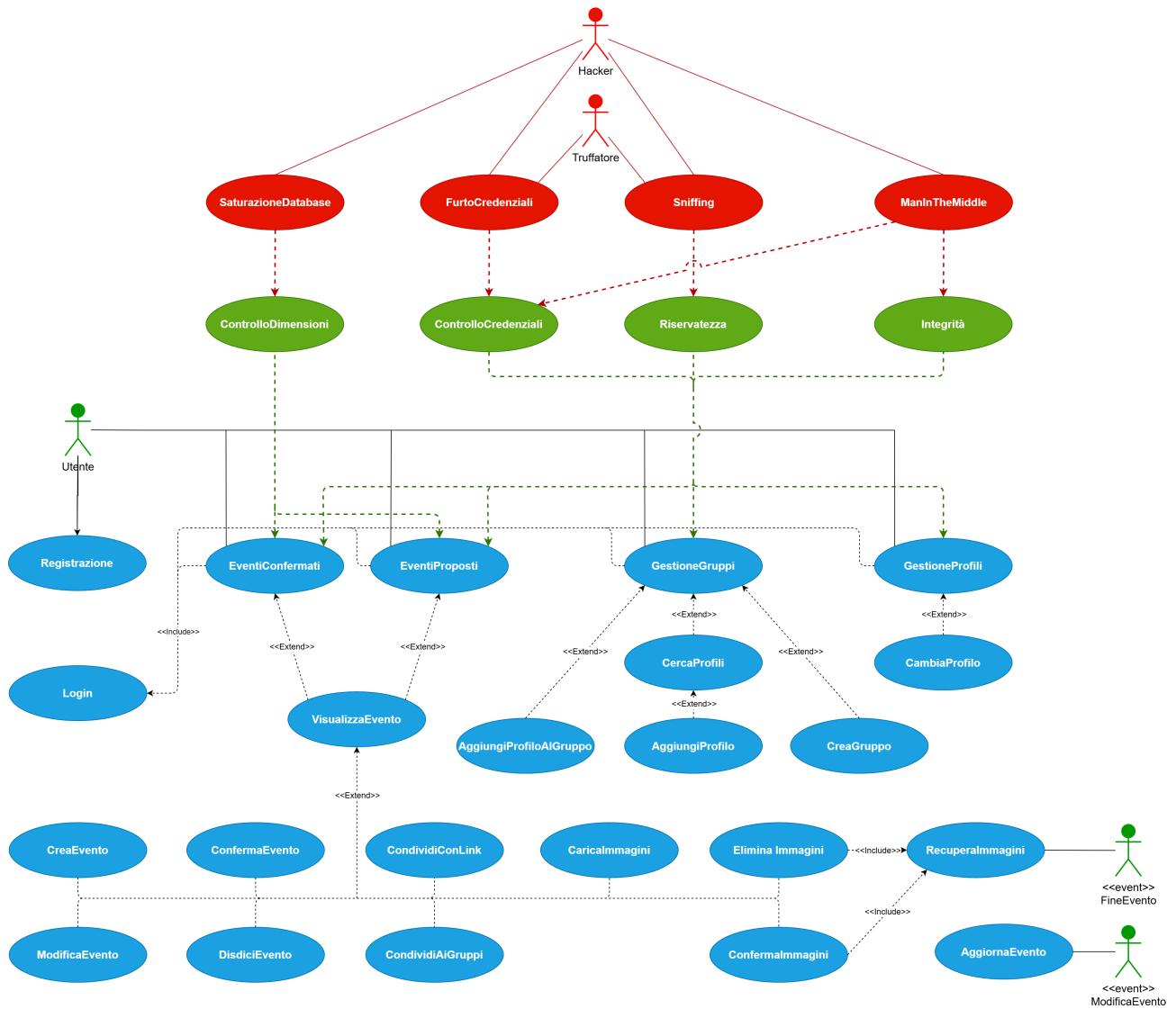
3.4.2 Tabella Minacce/Controlli

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

3.4.3 Analisi Tecnologica della Sicurezza

Tecnologia	Vulnerabilità
Autenticazione email/password	<ul style="list-style-type: none">• Utente rivela volontariamente la password• Utente rivela la password con un attacco di ingegneria sociale• Password banali
Cifratura comunicazioni	<ul style="list-style-type: none">• In caso di cifratura simmetrica particolare attenzione va alla lunghezza delle chiavi ed alla loro memorizzazione
Architettura Client/Server	<ul style="list-style-type: none">• DoS• Man in the Middle• Sniffing delle comunicazioni
Connessione Server/Persistenza	<ul style="list-style-type: none">• Limite massimo di connessioni contemporanee• Saturazione del Database

3.4.4 Security Use Case & Misuse Case



3.4.5 Security Use Case & Misuse Case Scenari

Titolo	ControlloDimensioni
Descrizione	Le richieste non possono superare un'una determinata dimensione
Misuse case	SaturazioneDatabase
Relazioni	
Precondizioni	L'attaccante ha i mezzi per carpire grandi quantità di file
Postcondizioni	Il sistema blocca la richiesta e limita la dimensione totale dei file caricati
Scenario principale	<ol style="list-style-type: none"> 1. L'attaccante fa una richiesta con dimensioni molto grandi 2. Il sistema controlla le dimensioni della richiesta, e la blocca
Scenari alternativi	<ol style="list-style-type: none"> 1. L'attaccante fa tante richieste di salvataggio dati in un breve lasso di tempo 2. Il sistema controlla le dimensioni totali delle richieste per ogni lasso di tempo 3. Se le dimensioni totali superano il limite, ogni richiesta successiva viene bloccata fino allo scadere del tempo
Scenari di un attacco avvenuto con successo	<ol style="list-style-type: none"> 1. L'attaccante riesce a farsi accettare le richieste con dimensioni elevate 2. Il sistema controlla la quantità totale di dati caricati dall'utente in un determinato lasso di tempo 3. Se la quantità supera il consentito, il sistema blocca l'utente

Titolo	ControlloCredenziali
Descrizione	L'accesso alle funzionalità del sistema deve essere controllato
Misuse case	FurtoCredenziali, ManInTheMiddle
Relazioni	
Precondizioni	L'attaccante ha i mezzi per carpire in tutto o in parte le credenziali di accesso di un utente
Postcondizioni	Il sistema blocca l'accesso non autorizzato e notifica il tentativo di accesso
Scenario principale	<ol style="list-style-type: none"> 1. L'attaccante tenta di accedere al servizio spacciandosi per un utente legittimo, di cui conosce le credenziali solo in parte (ad esempio mediante attacco con dizionario) 2. Il sistema non riconosce le credenziali, restituendo un errore 3. In seguito ad un numero fissato di tentativi falliti, il sistema blocca temporaneamente l'accesso a quell'utente e notifica l'anomalia a chi di dovere
Scenari di un attacco avvenuto con successo	<ol style="list-style-type: none"> 1. L'attaccante riesce a carpire le credenziali di accesso complete di un utente in un qualsiasi modo 2. Il sistema riconosce la correttezza delle credenziali, e fornisce l'accesso al soggetto malevolo 3. L'attaccante ha libero accesso al sistema, con privilegi diversi in base al tipo di utente

Titolo	Riservatezza
Descrizione	I dati non sono accessibili da chi non ne ha i permessi
Misuse case	Sniffing
Relazioni	
Precondizioni	L'attaccante ha i mezzi per intercettare i messaggi del sistema
Postcondizioni	Il sistema impedisce all'attaccante di decifrare (in tempi utili) i messaggi intercettati
Scenario principale	<ul style="list-style-type: none"> 1. Il Sistema protegge i messaggi 2. L'attaccante riesce ad intercettare un messaggio 3. L'attaccante prova a decifrare i messaggi, ma non riesce a trovare un modo per farlo abbastanza velocemente
Scenari di un attacco avvenuto con successo	<ul style="list-style-type: none"> 1. Il Sistema protegge i messaggi 2. L'attaccante riesce ad intercettare un messaggio 3. L'attaccante riesce a decifrare i messaggi e a leggerne il contenuto, ma solamente per una sessione di un utente

Titolo	Integrità
Descrizione	Integrità dei dati del sistema
Misuse case	ManInTheMiddle
Relazioni	
Precondizioni	<ul style="list-style-type: none"> 1. L'attaccante ha i mezzi per intercettare i messaggi del sistema 2. L'attaccante ha i mezzi per modificare i messaggi 3. L'attaccante ha i mezzi per spedire il messaggio modificato al destinatario
Postcondizioni	Il sistema rileva il messaggio contraffatto
Scenario principale	<ul style="list-style-type: none"> 1. Il Sistema protegge i messaggi 2. L'attaccante riesce ad intercettare un messaggio e lo modifica 3. Il sistema si accorge del messaggio contraffatto e lo segna nei log
Scenari di un attacco avvenuto con successo	<ul style="list-style-type: none"> 1. Il Sistema protegge i messaggi 2. L'attaccante riesce ad intercettare un messaggio e lo modifica 3. Il sistema accetta il messaggio e agisce di conseguenza, segnando il messaggio nei log

3.4.6 Requisiti di Protezione dei Dati

Sussistono inoltre i seguenti requisiti inerenti alla protezione dei dati:

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.

La visione e l'analisi dei log verrà gestita con un editor di testo 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

4 Analisi del Problema

4.1 Analisi Documento dei Requisiti: Analisi delle Funzionalità

Tabella delle Funzionalità

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 Informazioni/Flusso: Registrazione

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Email	semplice	Protezione alta	Input	Deve essere di 256 caratteri e del formato giusto
Password	semplice	Protezione molto alta	Input	Deve essere almeno di 8 caratteri

Tabella Informazioni/Flusso: Login

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Email	semplice	Protezione molto alta	Input	Non più di 256 caratteri
Password	semplice	Protezione molto alta	Input	Non più di 50 caratteri

Tabella Informazioni/Flusso: EventiConfermati

Informazione	Tipo	Livello protezione/privacy	Input / Output	Vincoli
Lista Eventi Confermati	Composto	Protezione media	Output	

Tabella Informazioni/Flusso: EventiProposti

Informazione	Tipo	Livello protezione/privacy	Input / Output	Vincoli
Lista Eventi Proposti	Composto	Protezione media	Output	

Tabella Informazioni/Flusso: GestioneGruppi

Informazione	Tipo	Livello protezione/privacy	Input / Output	Vincoli
Lista Gruppi	Composto	Protezione media	Output	
Tag di ricerca	Semplice	Protezione bassa	Input	
Lista Profili	Composto	Protezione bassa	Output	Non più di 5 profili
Identificativo Profilo corrente	Semplice	Protezione alta	Output	

Tabella Informazioni/Flusso: GestioneProfili

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Lista Profili	Composta	Protezione media	Output	
Identificativo Utente	Semplice	Protezione alta	Output	
Identificativo Profilo corrente	Semplice	Protezione alta	Output	

Tabella Informazioni/Flusso: VisualizzaEvento

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Identificativo Evento	Semplice	Protezione alta	Output	
Titolo	Semplice	Protezione media	Input/Output	Non più di 256 caratteri
Descrizione	Semplice	Protezione media	Input/Output	Non più di 1024 caratteri
Data e orario di inizio	Semplice	Protezione media	Input/Output	Deve essere precedente alla data di fine
Data e orario di fine	Semplice	Protezione media	Input/Output	Deve essere successiva alla data di inizio
Confermato	Semplice	Protezione media	Input/Output	
Immagini	Composto	Protezione media	Input/Output	
Profili associati	Composto	Protezione media	Output	

Tabella Informazioni/Flusso: AggiornaEvento

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Identificativo Evento	Semplice	Protezione alta	Output	
Titolo	Semplice	Protezione media	Input/Output	Non più di 256 caratteri
Descrizione	Semplice	Protezione media	Input/Output	Non più di 1024 caratteri
Data e orario di inizio	Semplice	Protezione media	Input/Output	Deve essere precedente alla data di fine
Data e orario di fine	Semplice	Protezione media	Input/Output	Deve essere successiva alla data di inizio
Confermato	Semplice	Protezione media	Input/Output	
Immagini	Composto	Protezione media	Input/Output	

Tabella Informazioni/Flusso: RecuperaImmagini

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Immagini	Composto	Protezione media	Input/Output	

Tabella Informazioni/Flusso: ScrritturaLog

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Data	semplice	Protezione media	Input	Non più di 40 caratteri
Ora	semplice	Protezione media	Input	Non più di 40 caratteri
Attore	semplice	Protezione alta	Input	Non più di 20 caratteri
Identificativo Profilo	semplice	Protezione alta	Input	Non più di 20 caratteri
Identificativo Utente	semplice	Protezione alta	Input	Non più di 20 caratteri
Operazione Eseguita	composto	Protezione alta	Input	
Azione	composto	Protezione molto alta	Input	

4.1.1 Analisi Documento dei Requisiti: Analisi dei Vincoli

Tabella Vincoli

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

4.1.2 Analisi Documento dei Requisiti: Analisi delle Interazioni

Tabella Maschere

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 Sistemi Esterni

Sistema	Descrizione	Protocollo di Interazione	Livello di Sicurezza

4.1.3 Analisi Ruoli e Responsabilità

Tabella Ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Utente	Gestione di tutte le informazioni relative all'utente e ai profili, eventi e gruppi collegati	View Login, View Registrazione, View EventiConfermati, View EventiProposti, view VisualizzaEvento, View GestioneGruppi, View CercaProfili, View GestioneProfili	È richiesto un alto grado di riservatezza	Illimitati

Utente: Tabella Ruolo-Informazioni

Informazione	Tipo di Accesso
Email	Lettura/Scrittura
Password	Lettura/Scrittura
Lista Eventi confermati	Lettura
Lista Eventi Proposti	Lettura
Lista Gruppi	Lettura
Tag di ricerca	Scrittura
Lista Profili	Lettura
Titolo	Lettura/Scrittura
Descrizione	Lettura/Scrittura
Data e orario di inizio	Lettura/Scrittura
Data e orario di fine	Lettura/Scrittura
Confermato	Lettura/Scrittura
Immagini	Lettura/Scrittura
Profili Associati	Lettura

4.1.4 Scomposizione del Problema

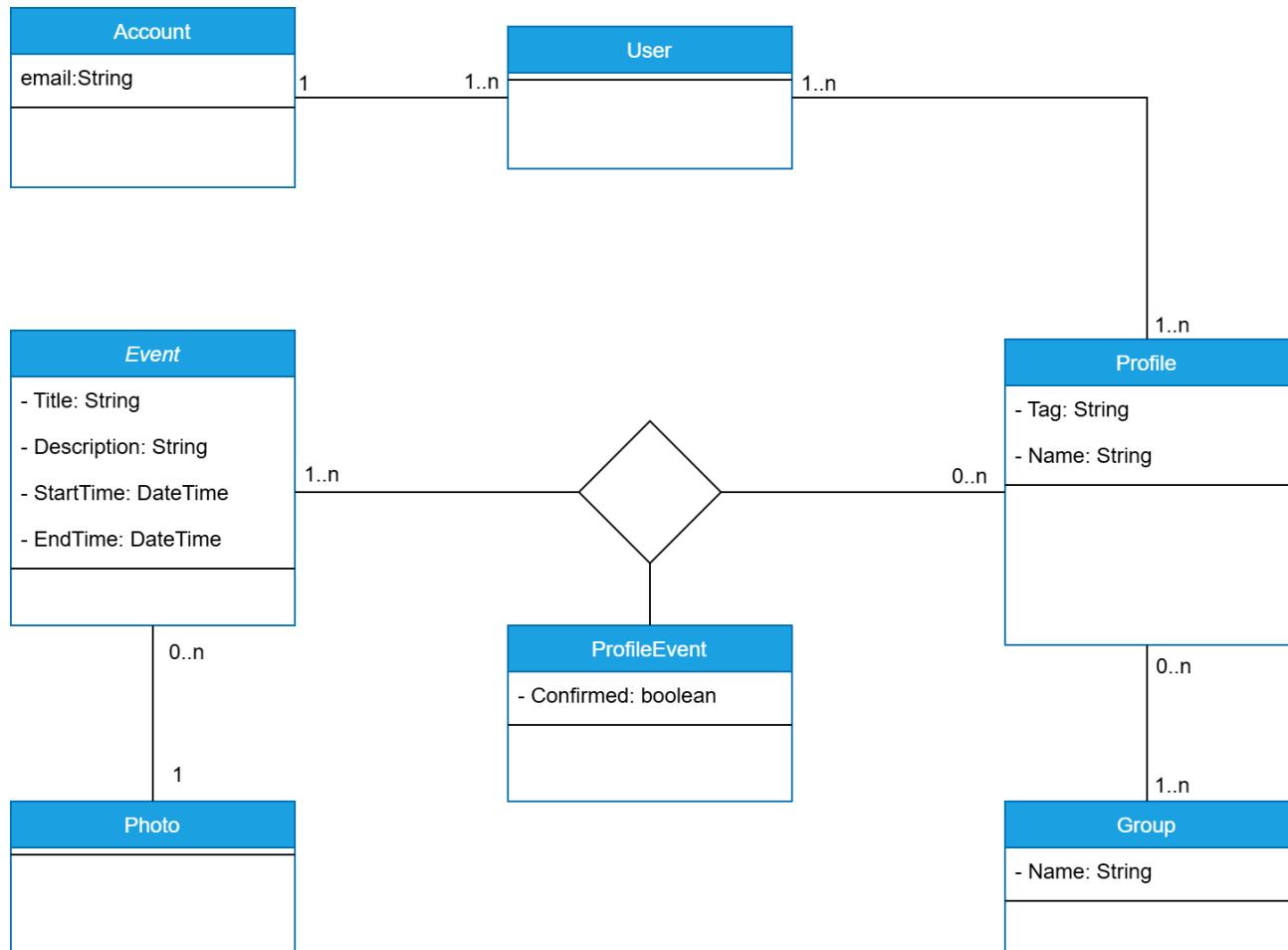
Tabella Scomposizione Funzionalità

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

Non sono presenti legami di esclusione o di necessità tra le sotto-funzionalità del sistema.

4.1.5 Creazione Modello del Dominio

Il seguente diagramma delle classi rappresenta la parte di modello del dominio relativa al sistema.



4.1.6 Architettura Logica: Struttura

Diagramma dei package

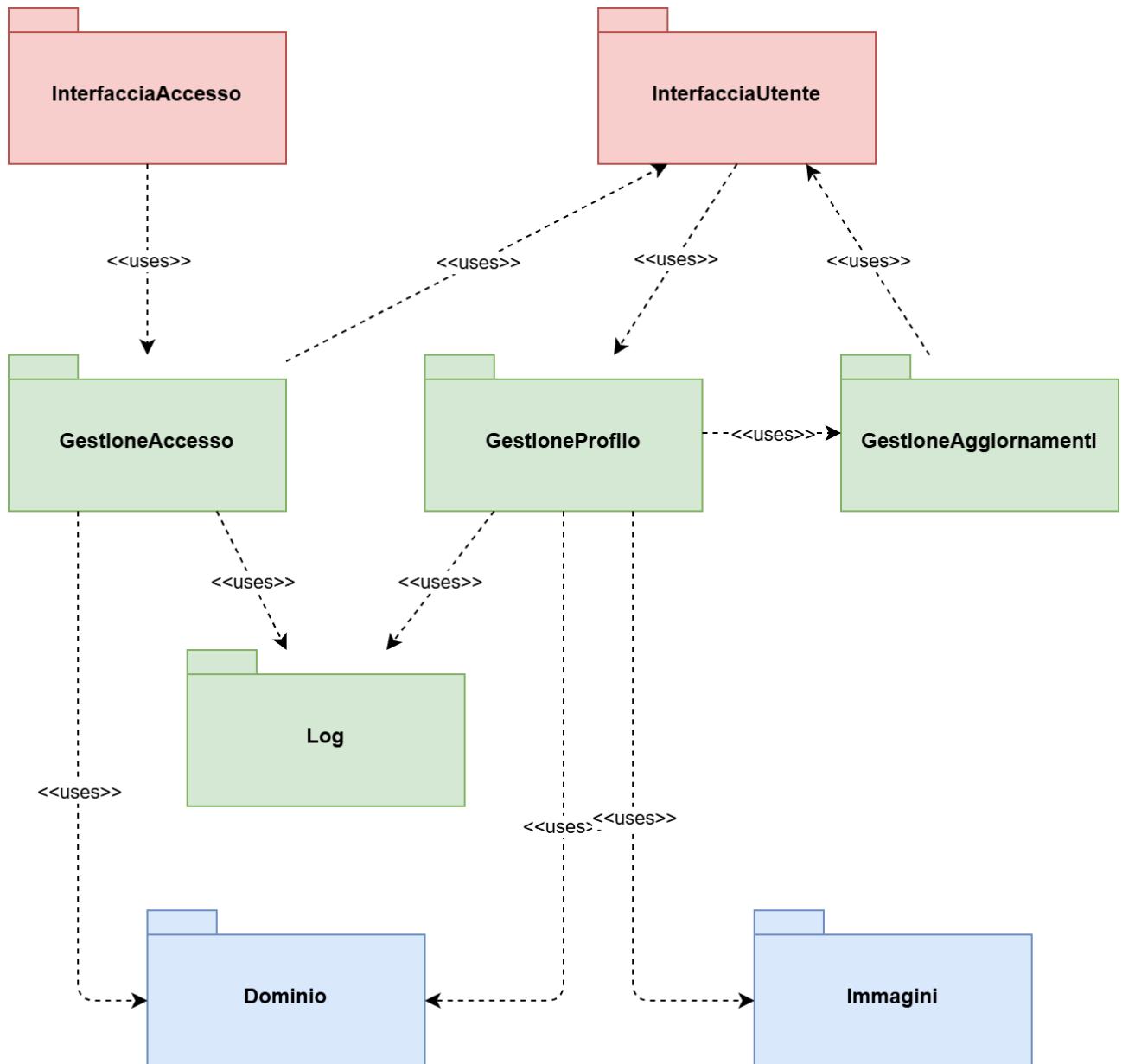


Diagramma delle classi: Dominio

Non viene riportato il diagramma delle classi associato al package **Dominio** in quanto è il modello del dominio creato nella fase precedente.

Diagramma delle classi: InterfacciaAccesso & GestioneAccesso

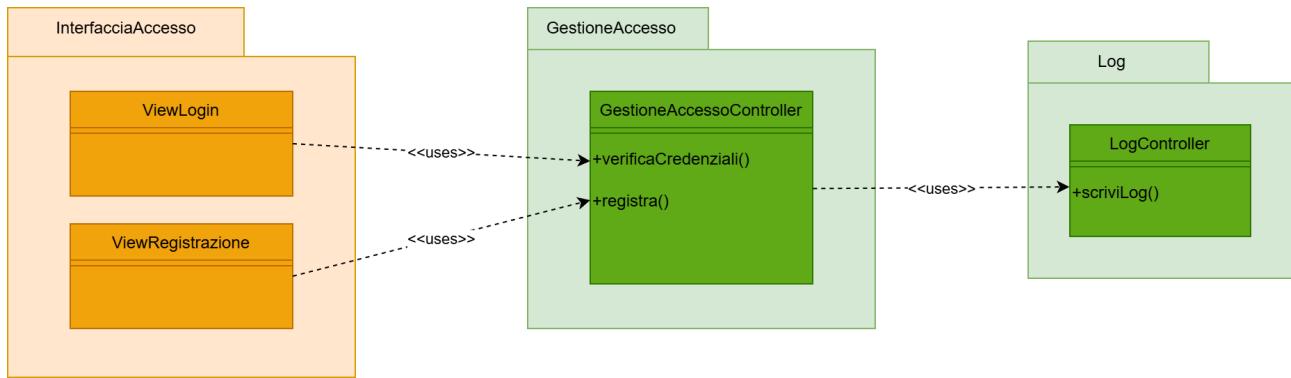
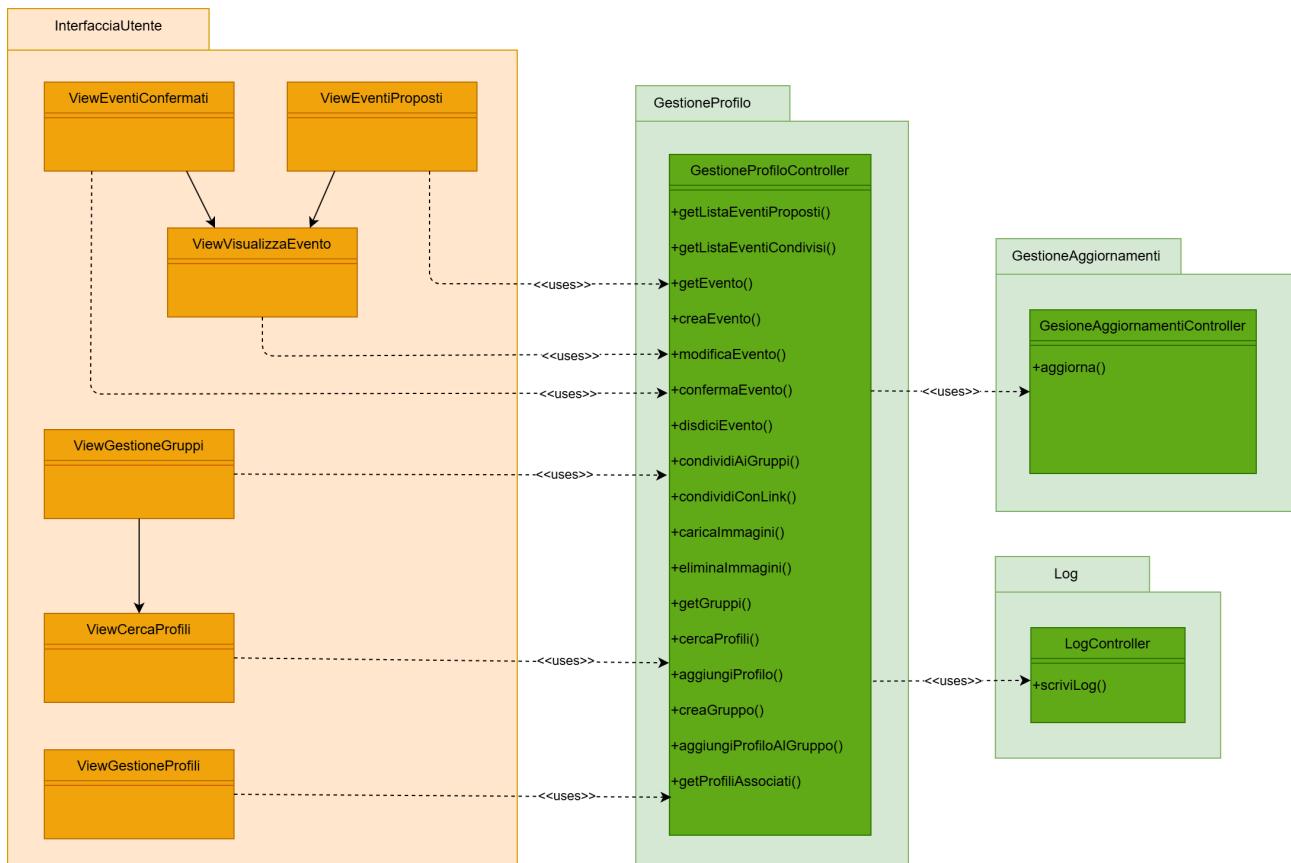


Diagramma delle classi: InterfacciaUtente & GestioneProfilo & GestioneAggiornamenti



4.1.7 Architettura Logica: Interazione

In seguito saranno riportati i principali diagrammi di sequenza durante un normale utilizzo dell'applicazione.

Diagramma di Sequenza: Login Utente

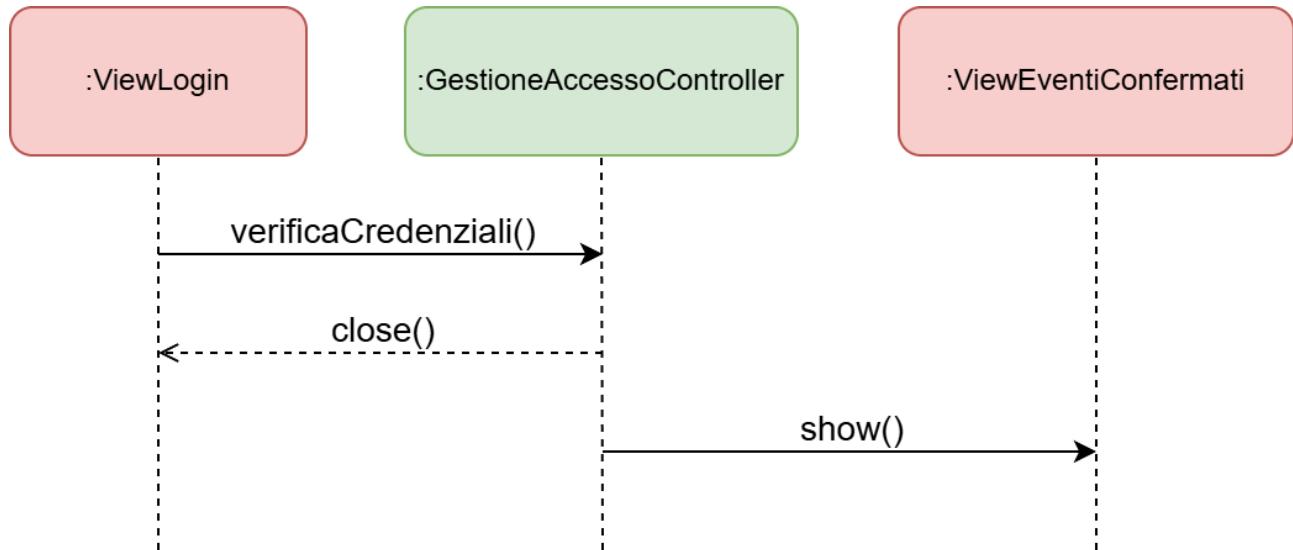


Diagramma di Sequenza: Registrazione

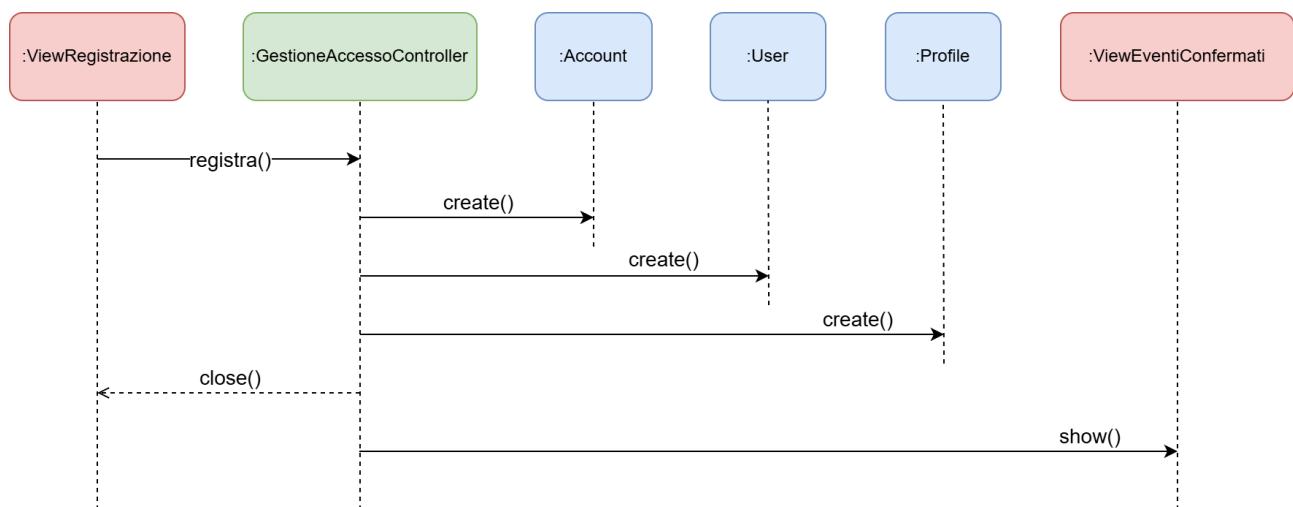


Diagramma di Sequenza: Visualizza Eventi Confermati

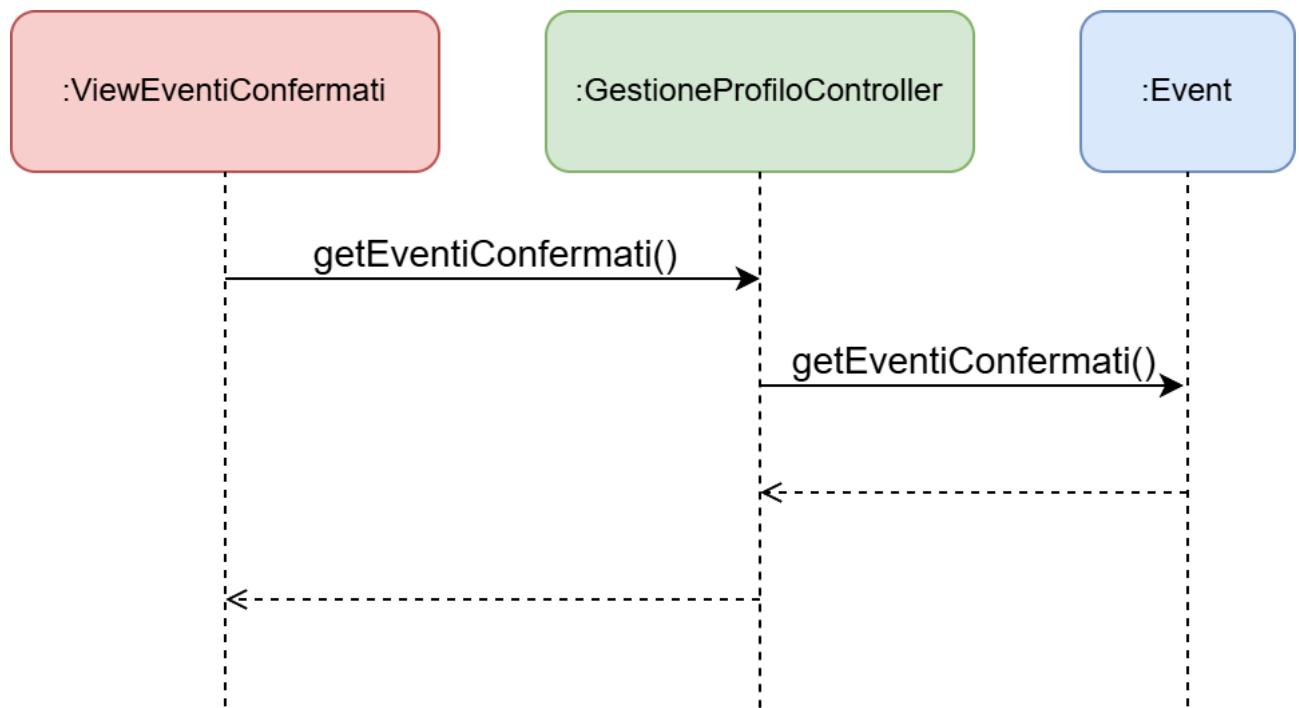


Diagramma di Sequenza: Visualizza Eventi Proposti

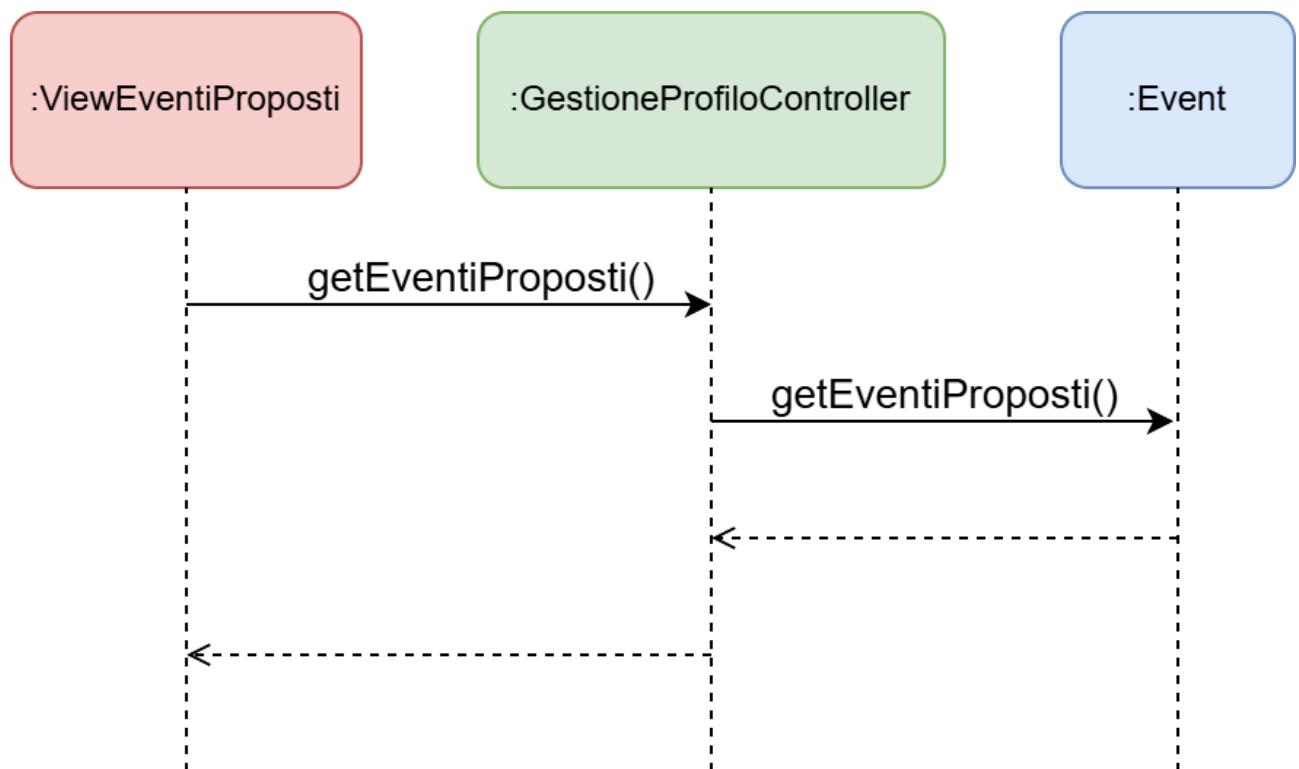


Diagramma di Sequenza: Visualizza Evento

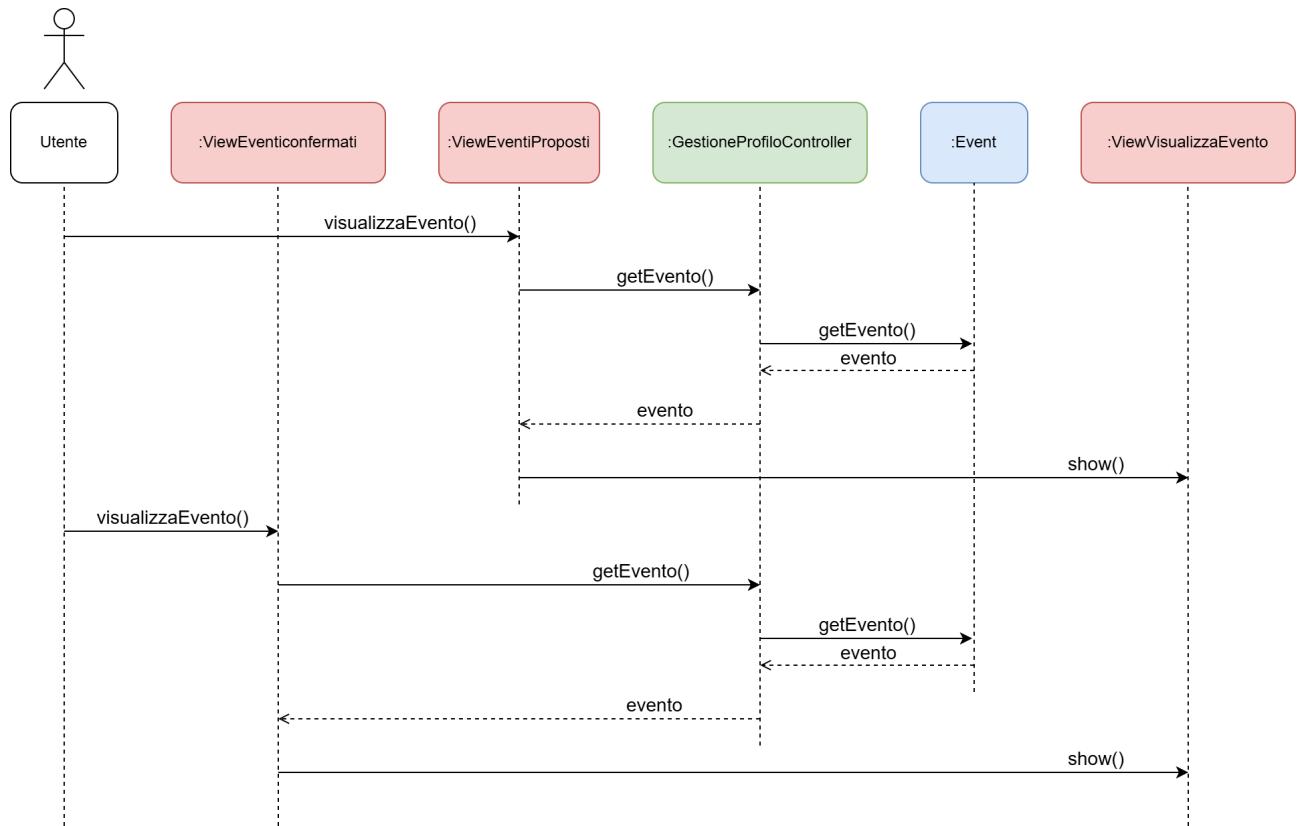


Diagramma di Sequenza: Condividi Evento ai gruppi

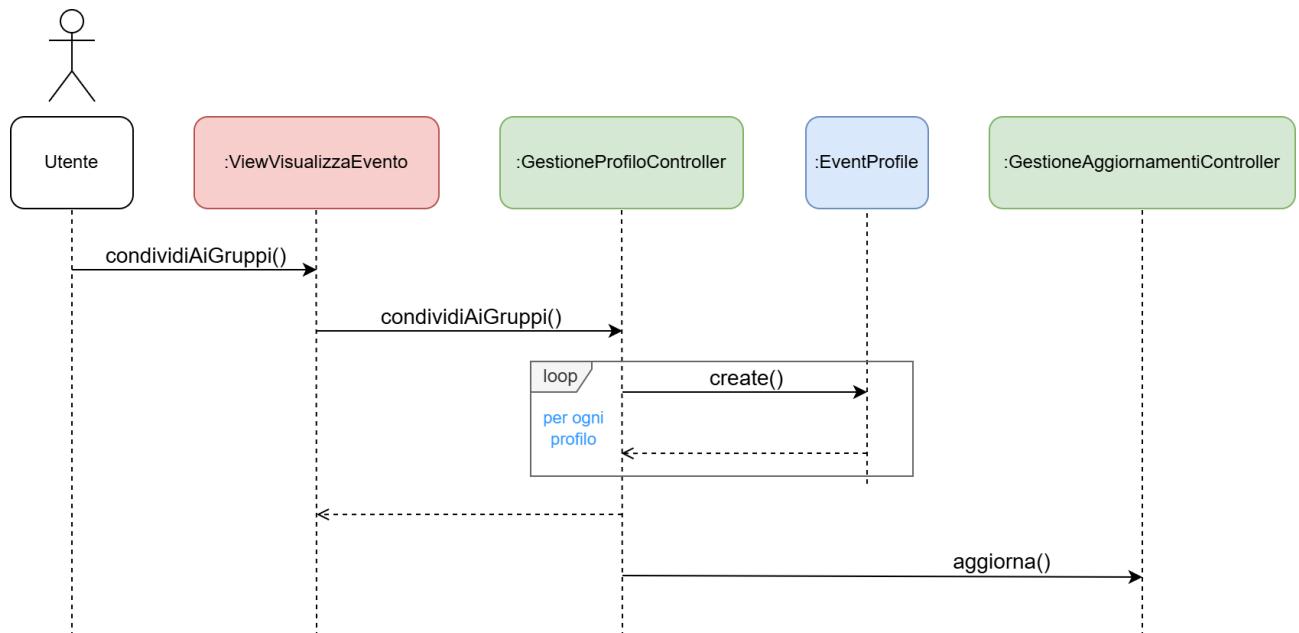


Diagramma di Sequenza: Crea / Modifica / Conferma / Disdici Evento

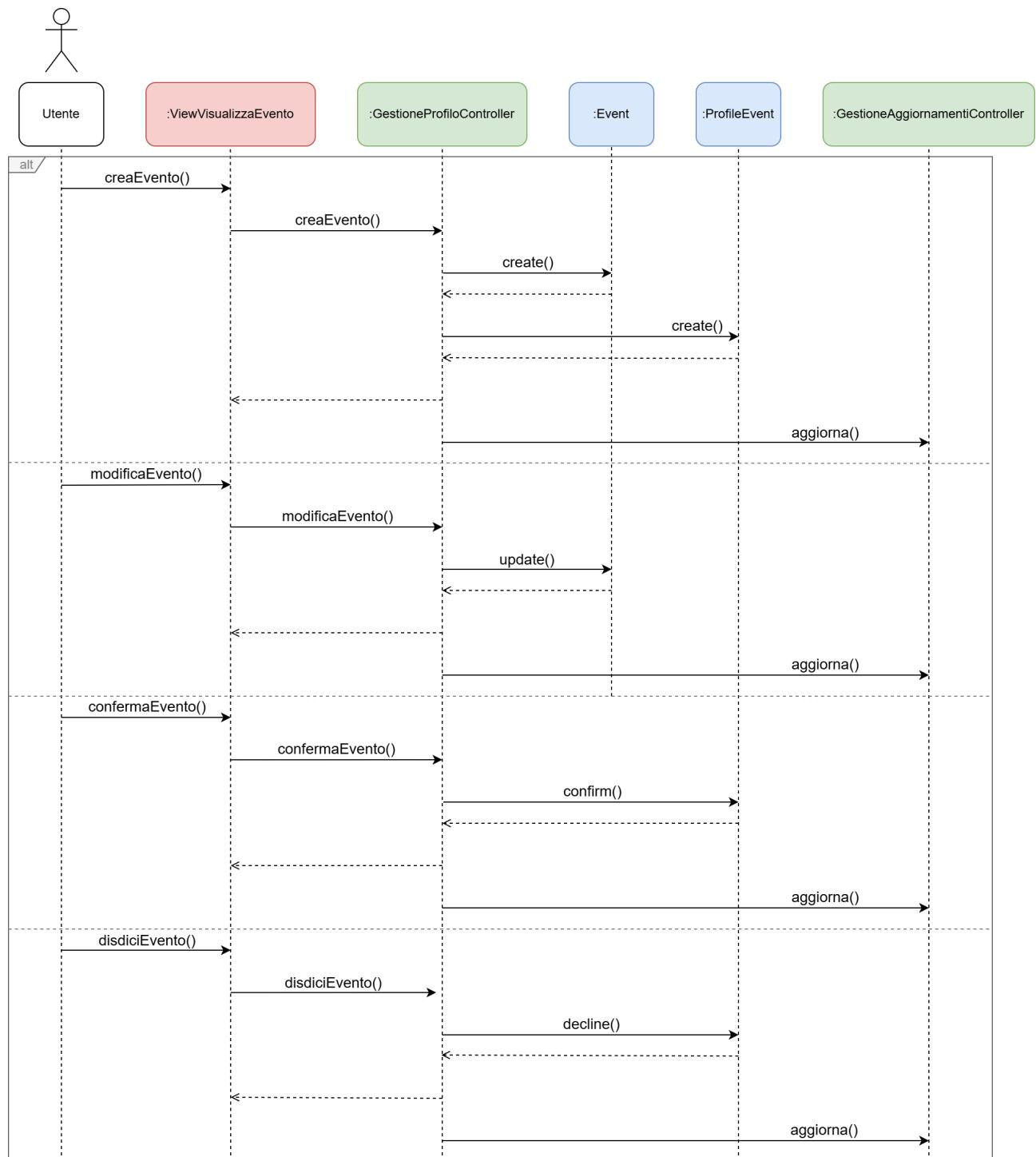


Diagramma di Sequenza: Carica Immagini

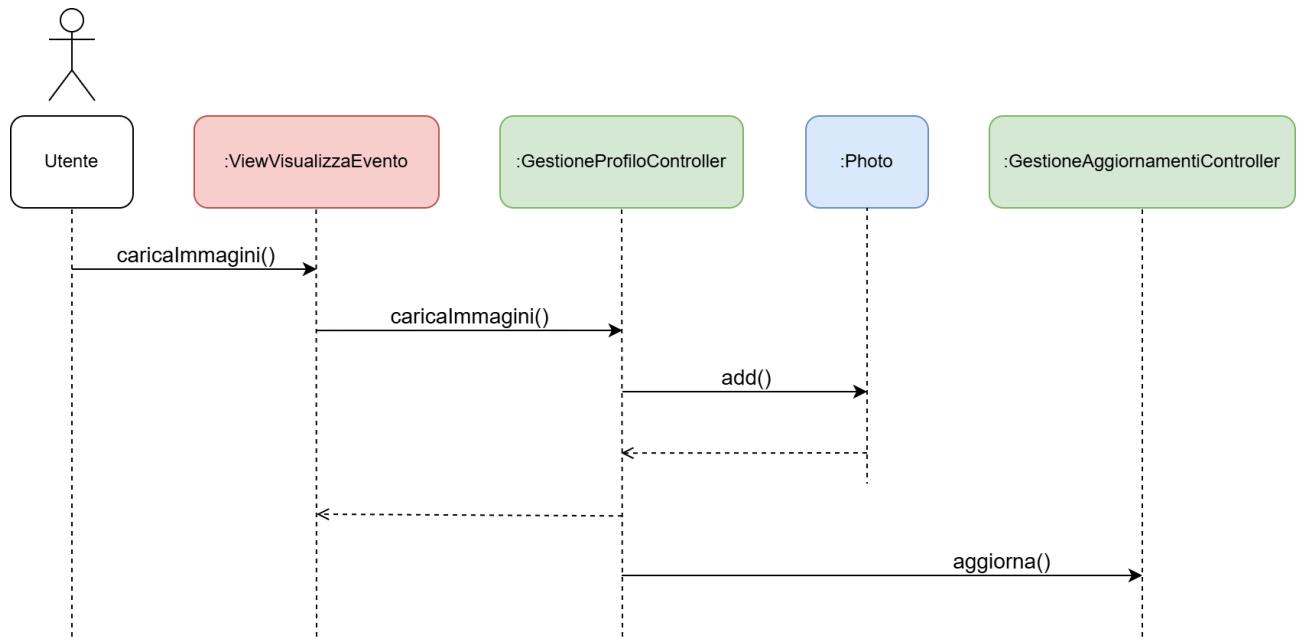
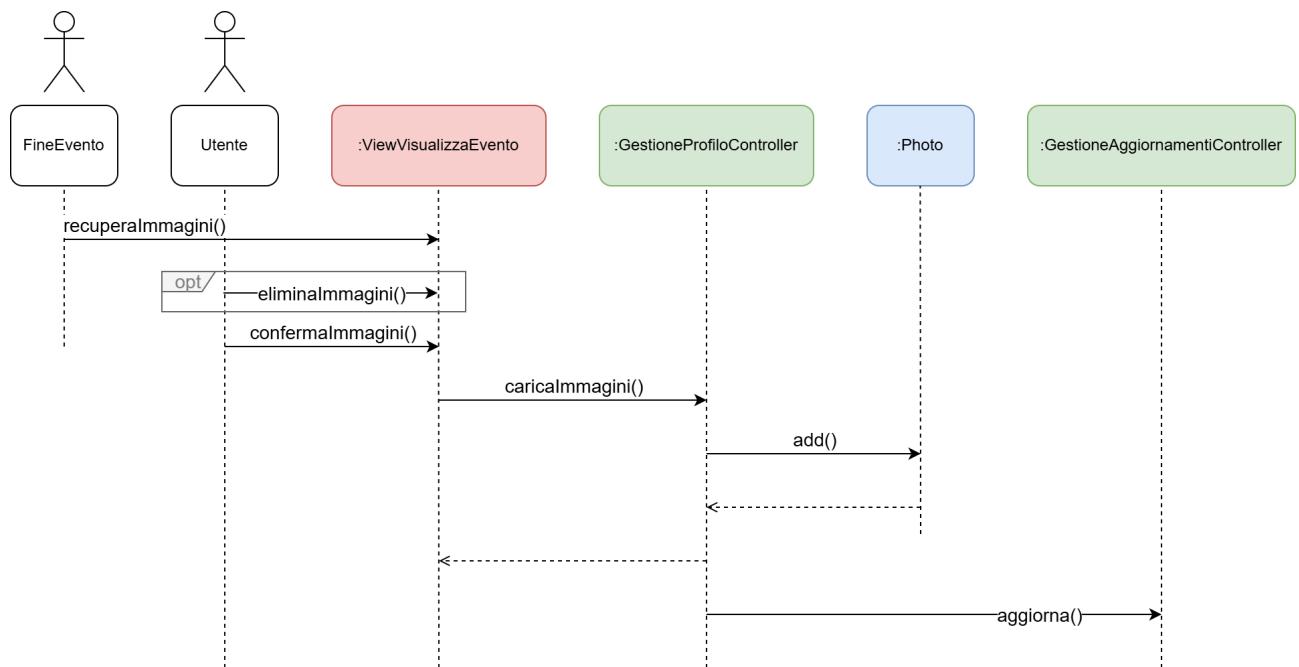


Diagramma di Sequenza: Recupera Immagini



4.1.8 Piano di Lavoro

I compiti sono stati divisi in base alle competenze di ogni membro del gruppo come indicato nella tabella sottostante:

Package	Progetto	Sviluppo
GestioneAccesso	Romanini	Romanini
GestioneEventi	Romanini	Romanini
GestioneAggiornamenti	Romanini	Romanini
InterfacciaUtente	Romanini	Romanini
InterfacciaAccesso	Romanini	Romanini
Dominio	Romanini	Romanini
Immagini	Romanini	Romanini
Log	Romanini	Romanini

I tempi di rilascio sono i seguenti:

- Progettazione entro due settimane dalla data odierna
- Sviluppo dei vari moduli con annessi test unitari entro due mesi dalla fine della fase di progettazione
- Integrazione e testing del sistema entro un mese dalla fine dello sviluppo

4.1.9 Sviluppi Futuri

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

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

5 Progettazione

5.1 Progettazione Architetturale

5.1.1 Requisiti non funzionali

Dall'analisi dei requisiti sono emersi i seguenti requisiti non funzionali:

- Tempo di risposta
- Usabilità
- Affidabilità
- Scalabilità
- Integrità dei dati
- Protezione dei dati
- Sicurezza delle comunicazioni

L'affidabilità e la scalabilità assumono fondamentale importanza vista la natura del software, che deve permettere agli utenti di poter organizzare, coordinare e condividere eventi. La compromissione di questi risulterebbe in un peggioramento dell'esperienza utente, da cui consegue una perdita di reputazione o di fedeltà del cliente.

Sarà necessario assicurare la sicurezza fisica dei dati immagazzinati nel sistema, così come la sicurezza software dei dati e delle comunicazioni. In caso di compromissione la perdita d'immagine e i risvolti legali sarebbero significativi. L'utilizzo di protocolli e tecnologie standard del settore dovrebbero garantire la sicurezza del sistema senza peggiorarne significativamente l'usabilità o le prestazioni.

Nonostante il sistema non presenti vincoli di tempo stringenti, una caratteristica essenziale dell'applicazione sarà la velocità di distribuire gli aggiornamenti ai vari utenti.

Infine, l'intuitività dell'interfaccia è fondamentale per l'usabilità e la diffusione del prodotto.

5.1.2 Scelte tecnologiche

La scelta tecnologica principale ricade sul tipo di applicazione che si andrà a sviluppare. In questo caso la scelta è quella di sviluppare sia un'interfaccia per browser web, sia un'applicazione per dispositivo mobile, per i seguenti motivi:

1. l'interfaccia web consente di avere una piattaforma standard accessibile da quasi tutti i dispositivi, con il solo requisito di un browser, potenzialmente permettendo (in base alla dimensione dello schermo) una maggior facilità di utilizzo, per rispondere all'esigenza organizzativa di medio o lungo termine. In questo modo allargano le possibilità di accesso al servizio.
2. l'applicazione mobile permette una gestione a breve termine e un aggiornamento costante, ed è fondamentale per recuperare le immagini scattate dall'utente.

5.1.3 Scelta dell'architettura

Dopo una rapida analisi, si è constatato che l'architettura più adeguata per il sistema è l'**architettura client-server a 3 livelli**.

L1 – Client

La componente lato Client implementerà l'interfaccia utente, gestendo le interazioni del cliente e richiedendo i dati al server, salvandoli in una cache locale laddove la tecnologia lo renda possibile. Inoltre avrà la responsabilità di recuperare le immagini scattate durante l'evento.

L2 – Server

Per distribuire meglio il carico, si è deciso di scomporre i server in base alle funzionalità offerte. Si hanno quindi tre server:

- Un server per le funzionalità di autenticazione
- Un server principale che risponde alle richieste dei client
- Un server che gestisce la propagazione degli aggiornamenti agli utenti

L3 – Persistenza

La gestione della persistenza verrà implementata in due server con responsabilità differenti:

- Un server sul quale sarà installato un DBMS relazionale che gestisca i dati e le relazioni dei componenti
- Un server per il salvataggio delle immagini e file

Il database relazionale sarà accessibile solo dal server che fornisce i servizi, per garantire il controllo dei ruoli e dei permessi. Le immagini saranno invece accessibili solo in lettura direttamente dai client, che dovranno però essere a conoscenza dei codici che le identificano. Questa conoscenza è considerata sufficiente per garantire la confidenza, ammesso che l'identificativo sia lungo abbastanza, ma anche un rischio accettabile, in quanto ridurrà di molto il carico del server principale.

5.1.4 Pattern architetturali e di design

Il pattern Client-Server è stato scelto come pattern architetturale per gestire l'accesso, le immagini e i servizi principali.

Il pattern Event Driven sarà invece usato per notificare i Client degli aggiornamenti.

A livello di design distinguiamo tre componenti principali: Model, View e Controller. Mentre la View sarà delegata completamente ai Client, Model e Controller saranno invece suddivisi tra il Client e il Server, in base alle necessità di caching dei dati e della località delle operazioni di business.

Si riportano di seguito i diagrammi di package e componenti che descrivono l'architettura del sistema.

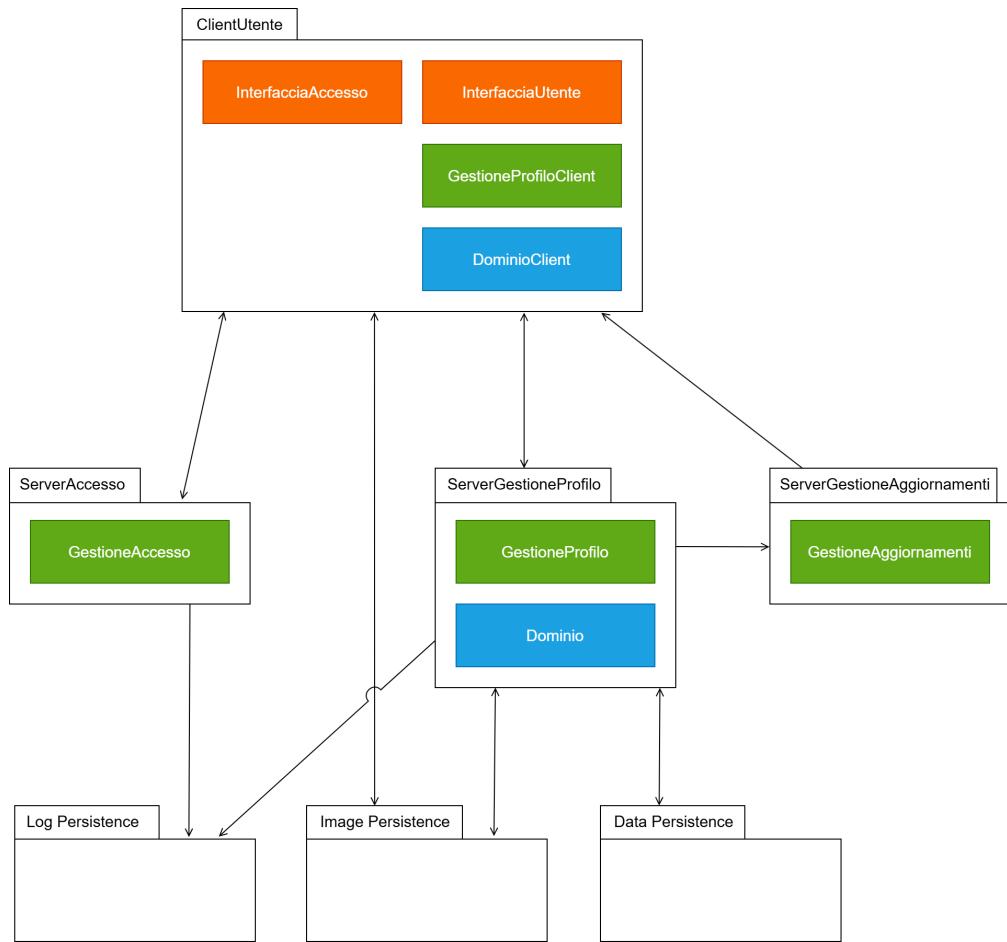


Figura 1: Diagramma dei package

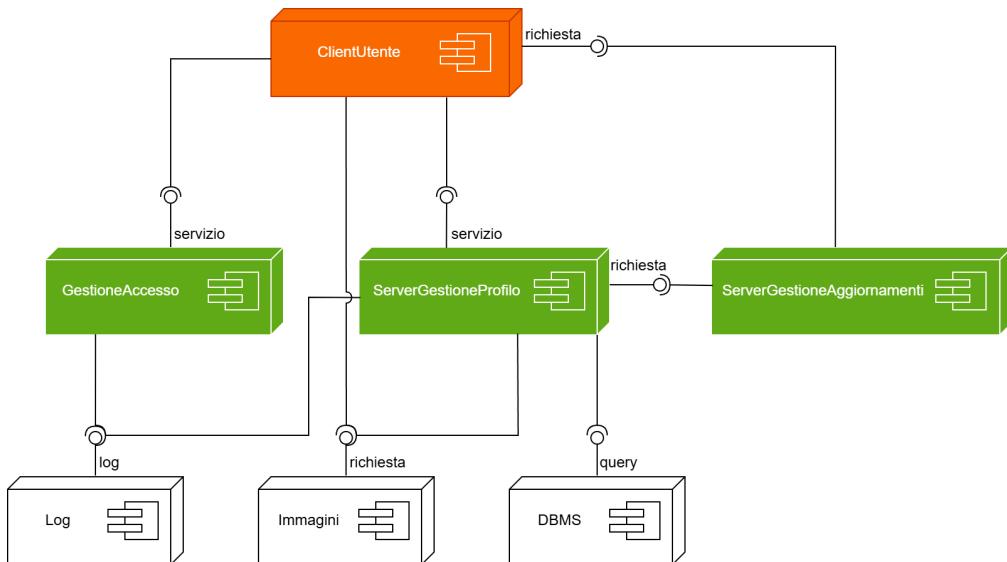


Figura 2: Diagramma dei componenti

5.2 Progettazione di dettaglio

5.2.1 Struttura: Dominio

Per quanto riguarda il dominio, distinguiamo il dominio del server dal dominio del client.

Diagramma di dettaglio: Dominio Client

A tutti i componenti principali si aggiunge un identificativo anfanumerico chiamato hash, per separare gli oggetti dall'identificativo assegnatogli dal database, che potrebbe portare a problemi di predittivit, e quindi compromettere l'autorizzazione all'accesso delle risorse da parte degli utenti.

Viene aggiunto il ruolo tra l'user e il profilo, per permettere una gestione dei permessi.

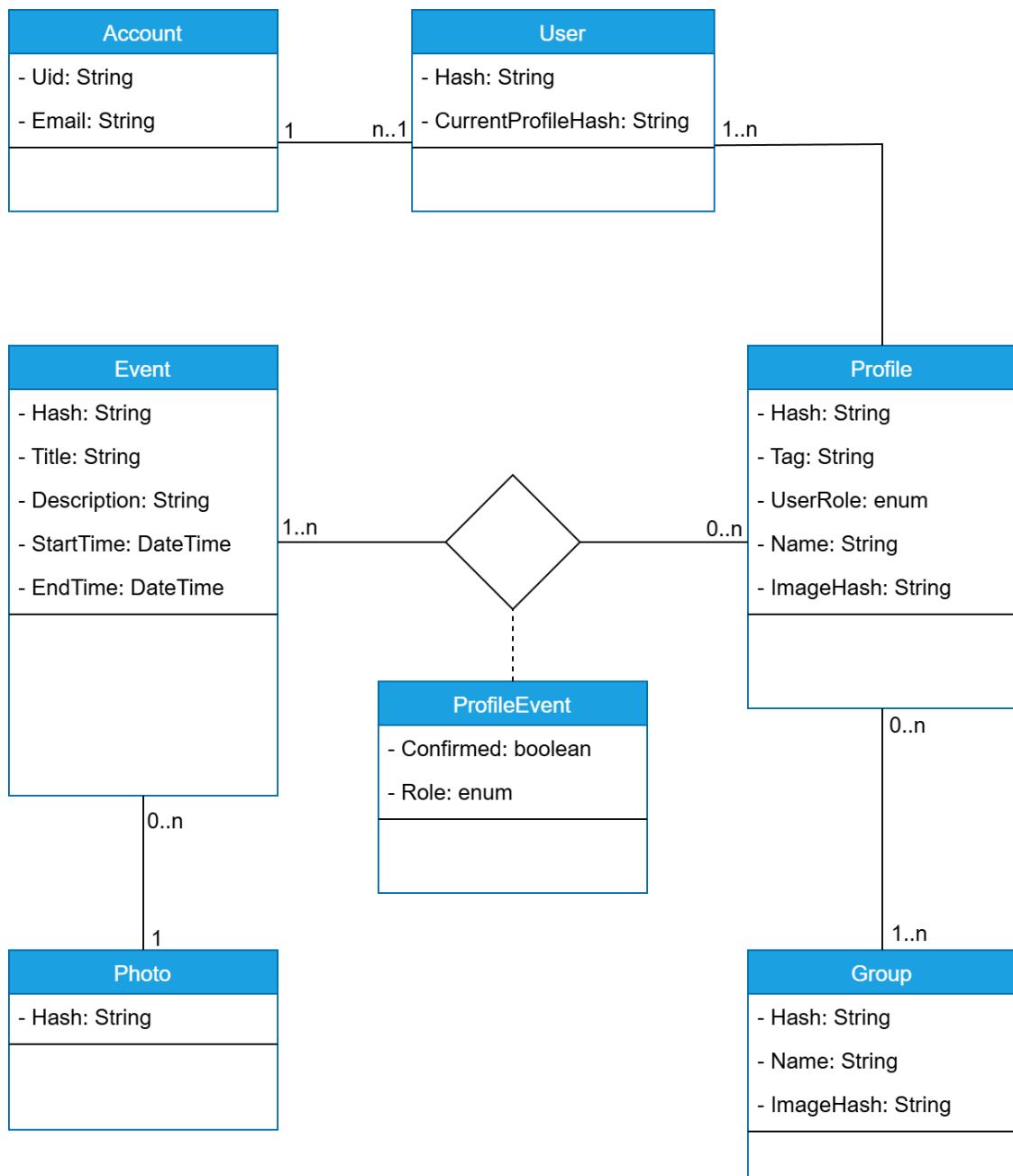
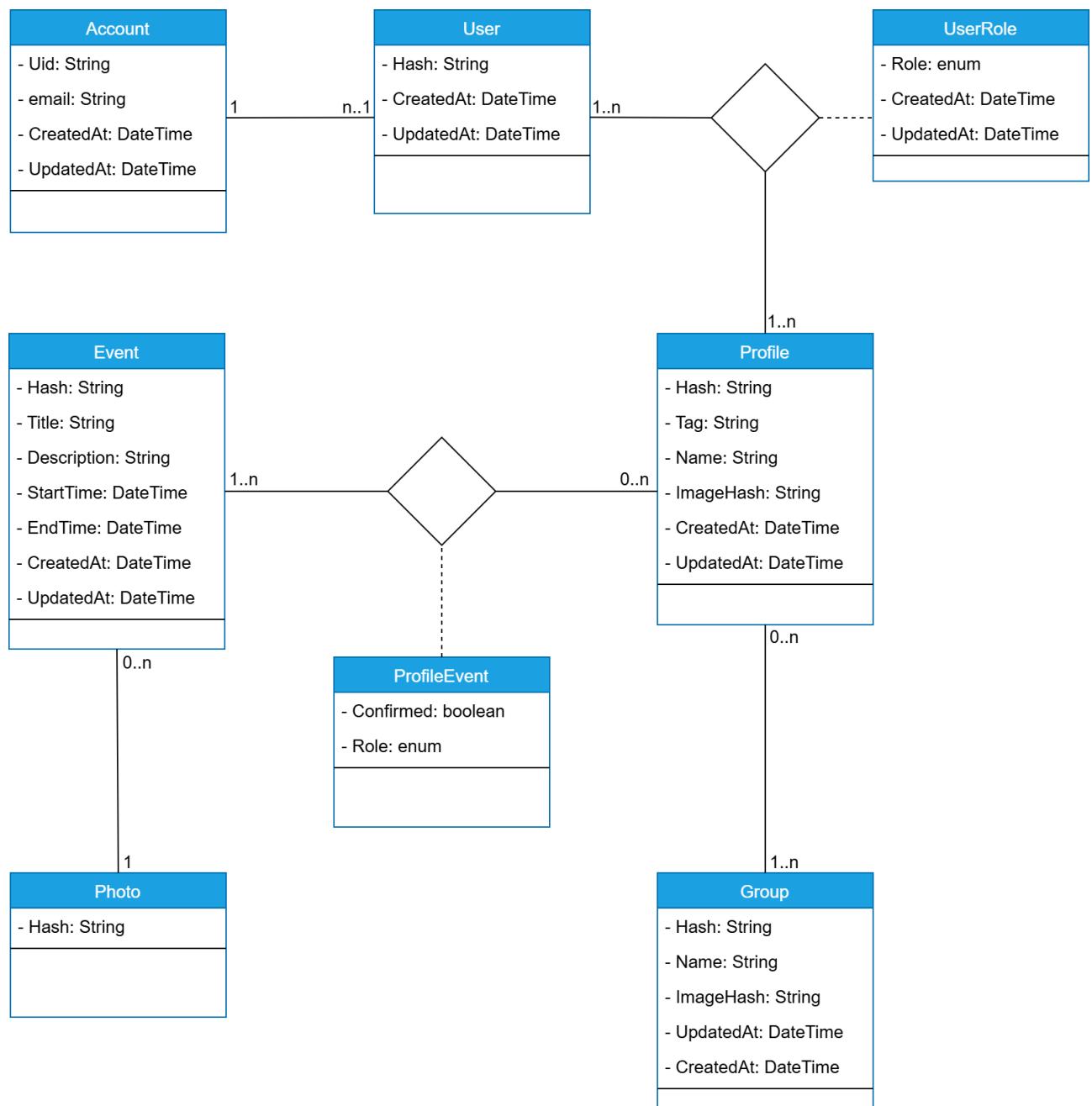


Diagramma di dettaglio: Dominio Server

Oltre all'Hash, per mantenere traccia delle attività e delle modifiche si aggiungono, ove necessario, le date di creazione e di ultima modifica.

Inoltre, aggiungiamo un componente per definire il ruolo dell'utente sul profilo.



5.2.2 Struttura: Interfacce

Diagramma di dettaglio: Interfacce Gestione Accesso e Gestione Aggiornamenti

<<Interface>> IGestioneAccesso	<<Interface>> IGestioneAggiornamenti
+ verificaCredenziali(String, String): Account + registra(String, String): Account	+ aggiornaEvento(Event, enum): void + aggiornaGruppo(Group, enum): void

Diagramma di dettaglio: Interfacce Client

<<Interface>> ILocalProfilo	<<Interface>> ILocalGruppi
+ getProfilo(String): Profile + getProfiles(List<String>): List<Profile>	+ getGruppi(Profile): List<Group>

<<Interface>> IAggiornamentiObserver	<<Interface>> ILocalEventi
+ aggiornaEvento(Event): void + aggiornaGruppo(Group): void	+ getEvento(String): Event + getEventi(User): List<Event> + dismissImage(Event, Photo): void + generateLink(Event): String

Diagramma di dettaglio: Interfacce Server

<<Interface>> IProfili	<<Interface>> IGruppi
+ getProfilo(String): Profile + getProfiles(List<String>): List<Profile> + lookForProfiles(String): List<Profile>	+ getGruppi(Profile): List<Group> + createGruppo(String, List<Profile>): Group + addProfilesToGroup(List<Profile>, Group): Group

<<Interface>> IEventi	<<Interface>> IUtente
+ getEventi(User): List<Event> + getEvento(String): Event + createEvent(String, String, DateTime, DateTime, boolean): Event + updateEvent(String, String, DateTime, DateTime, boolean): Event + uploadImages(Event, List<Photo>): Event + confirmEvent(Event, Profile): Event + dismissEvent(Event, Profile): Event + shareEvent(Event, List<Group>): Event	+ getUtente(Account): User

Le interfacce client e server risultano molto simili in quanto il client, prima di fare una richiesta dati al server, controllerà la cache locale per cercare i dati. L'aggiunta di tali interfacce consente di applicare il *Dependency Inversion Principle* in modo da disaccoppiare gli utilizzatori dalle implementazioni, che potrebbero cambiare.

5.2.3 Struttura: Controller

Ogni interfaccia verrà implementata da un relativo controller.

Diagramma di dettaglio: Gestione Accesso e Gestione Aggiornamenti

Introduciamo un Controller per la connessione con la persistenza del log

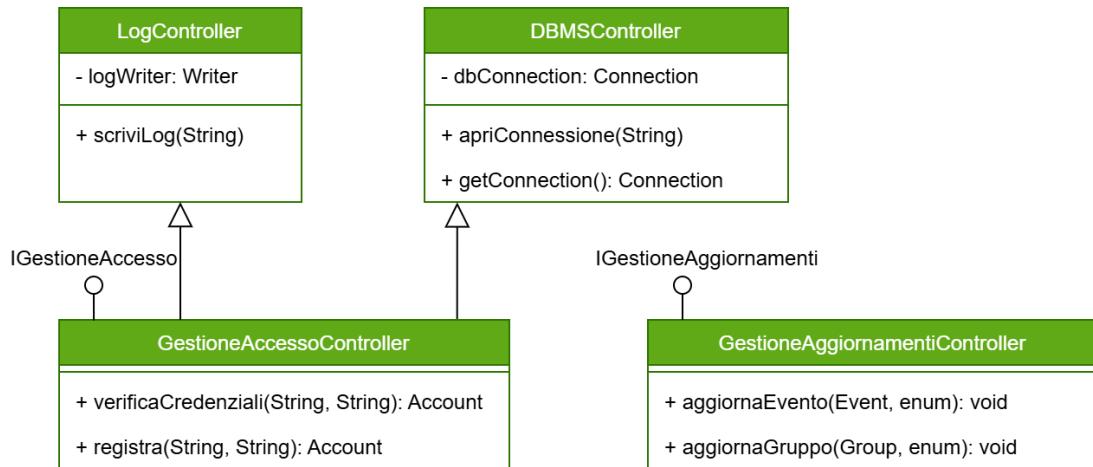
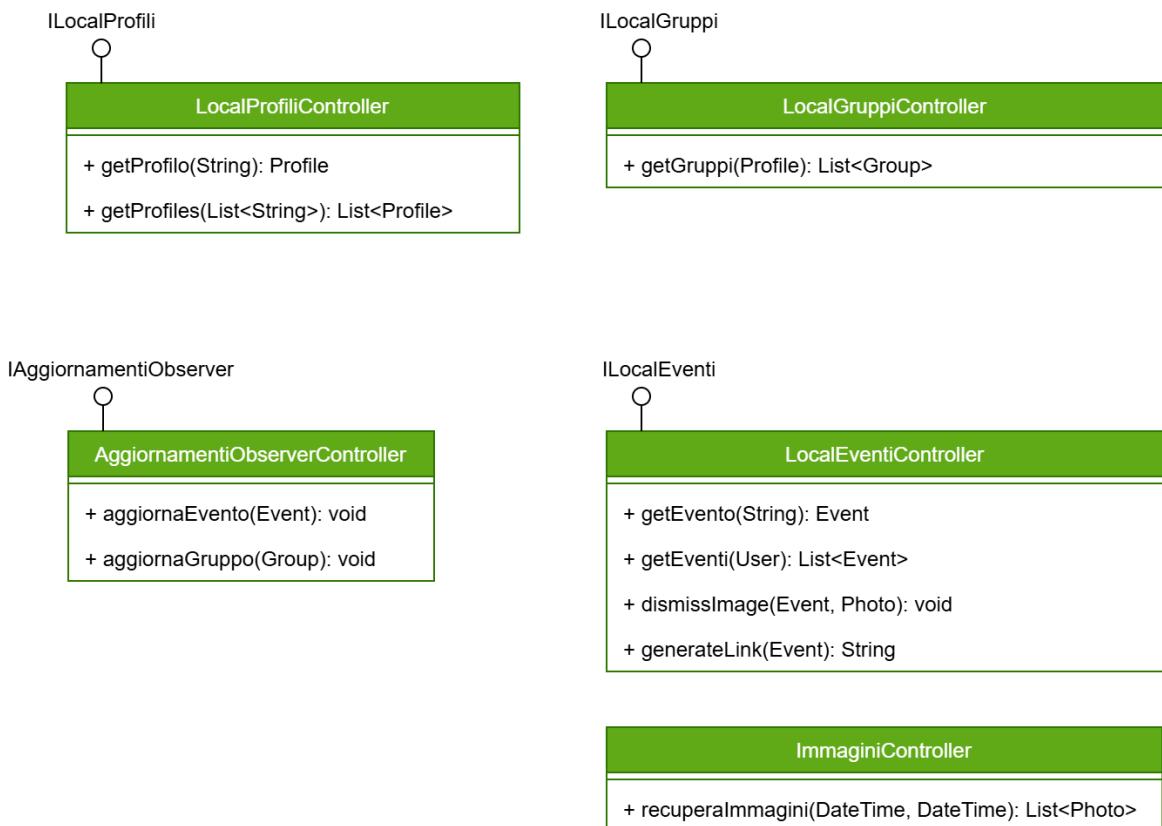


Diagramma di dettaglio: Client



5.2.4 Struttura: Interfacce Grafiche

Diagramma di dettaglio: Server

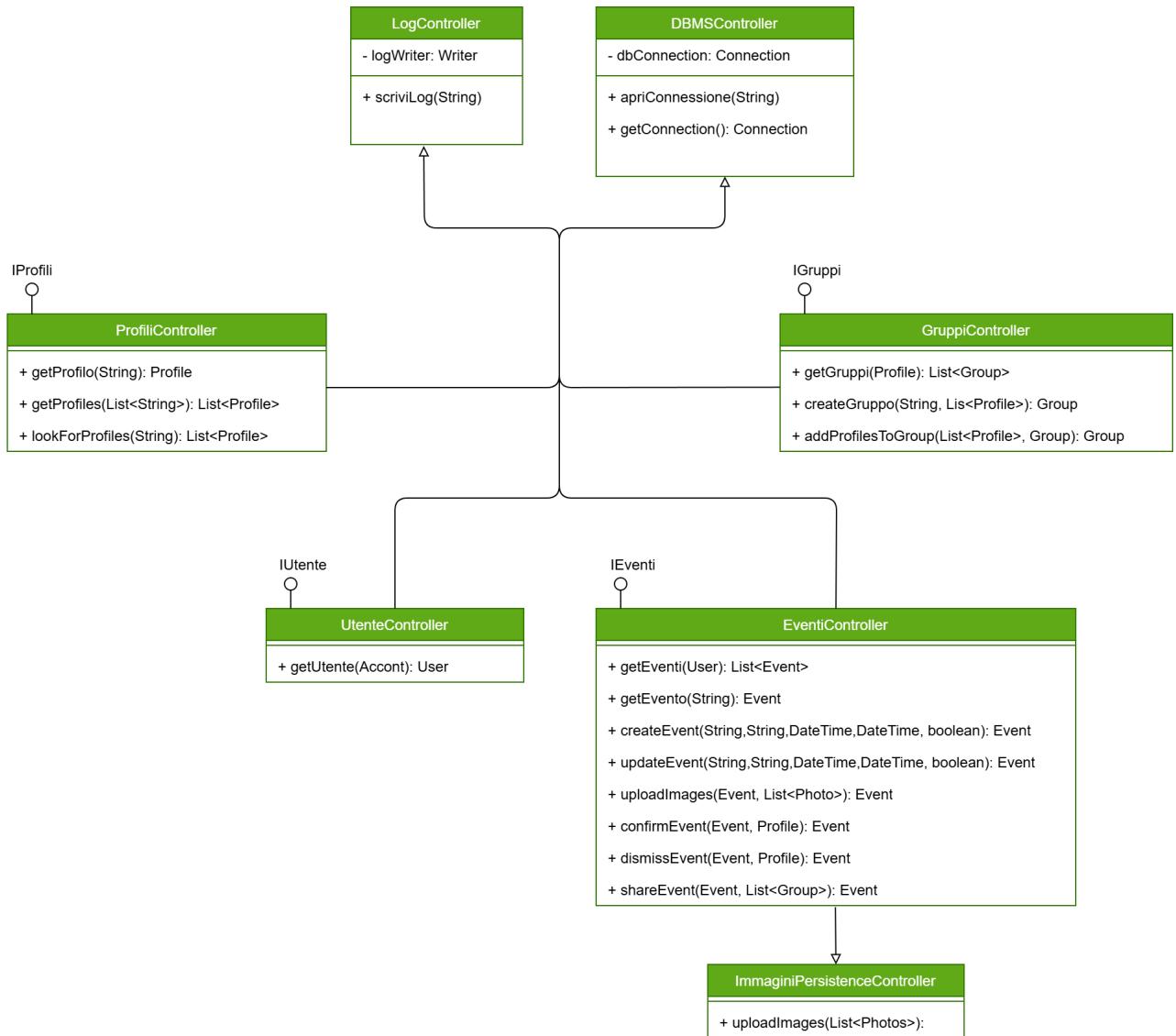


Diagramma di dettaglio: InterfacciaUtente - Eventi

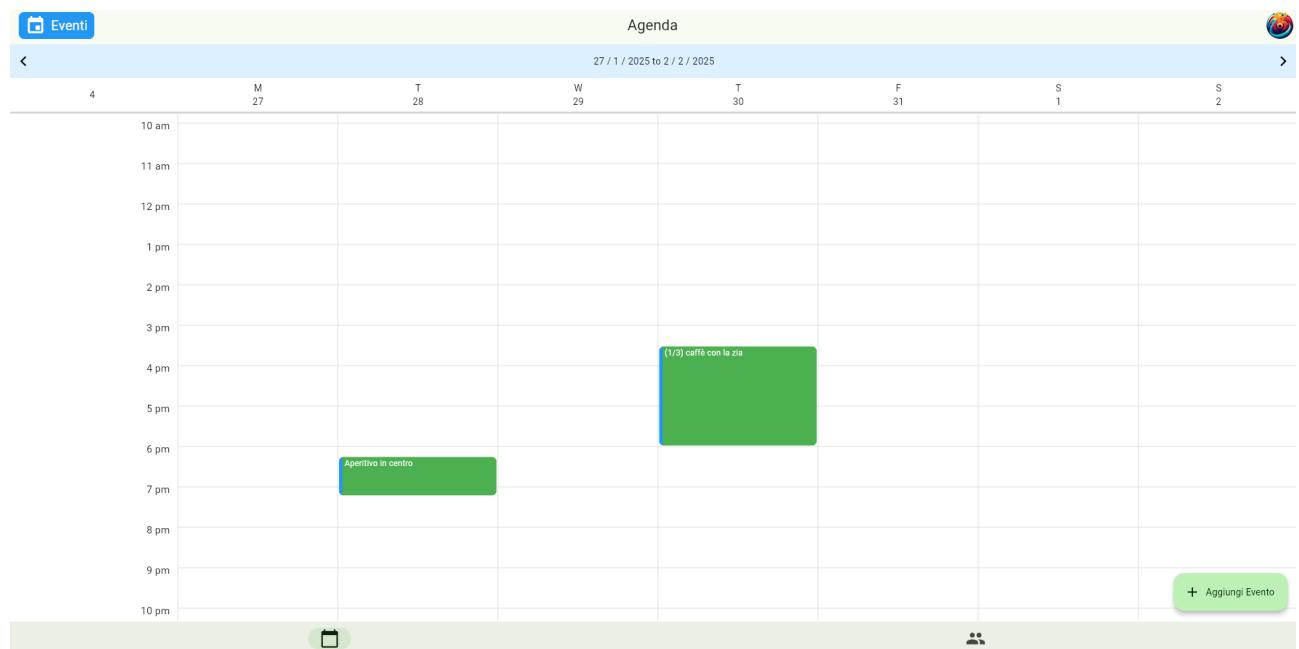
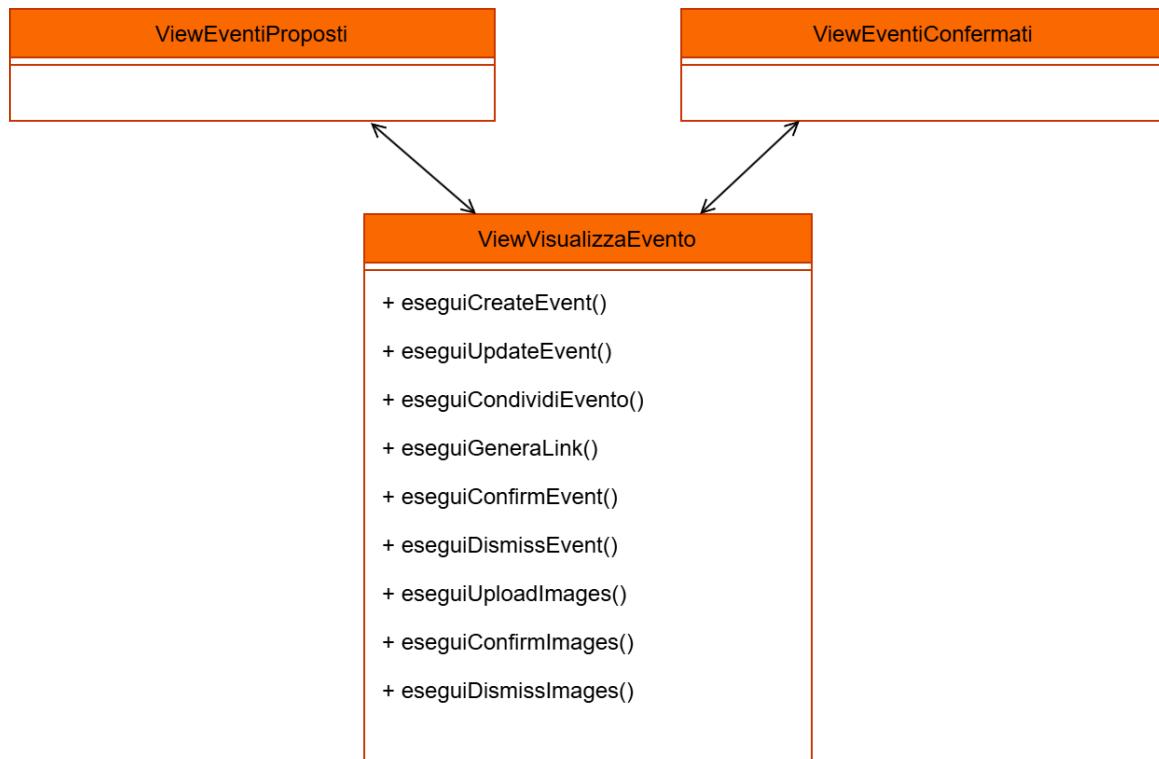


Figura 3: Eventi confermati

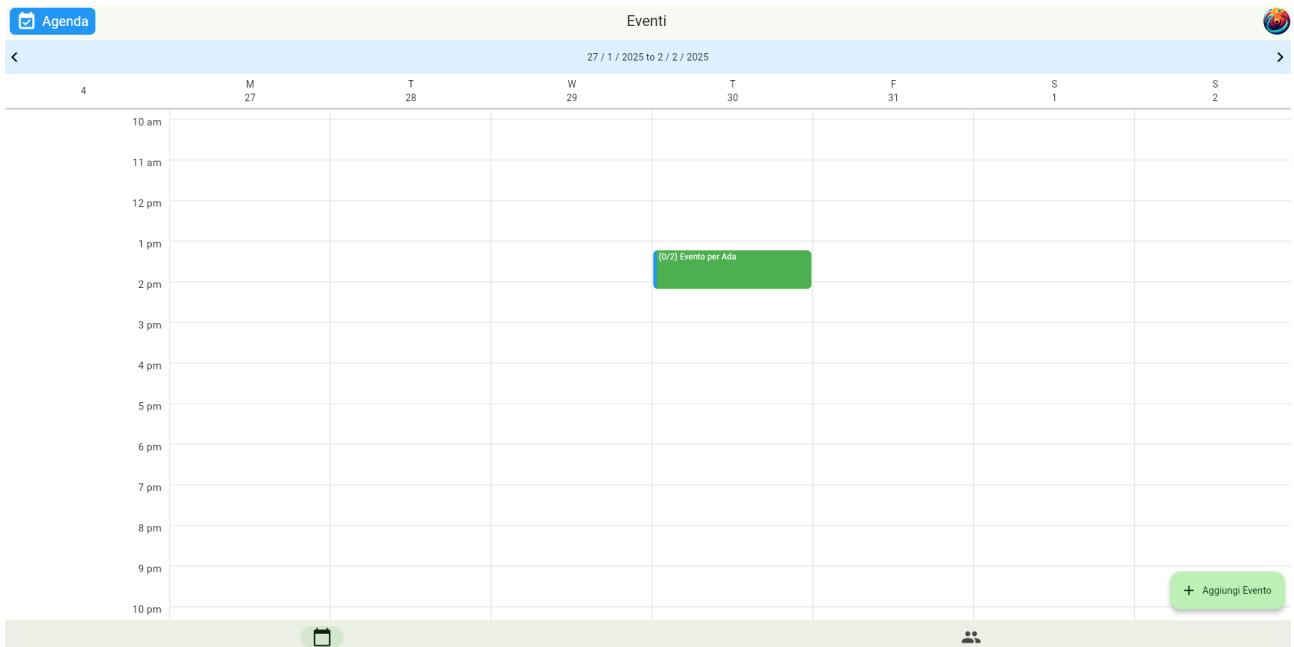


Figura 4: Eventi proposti

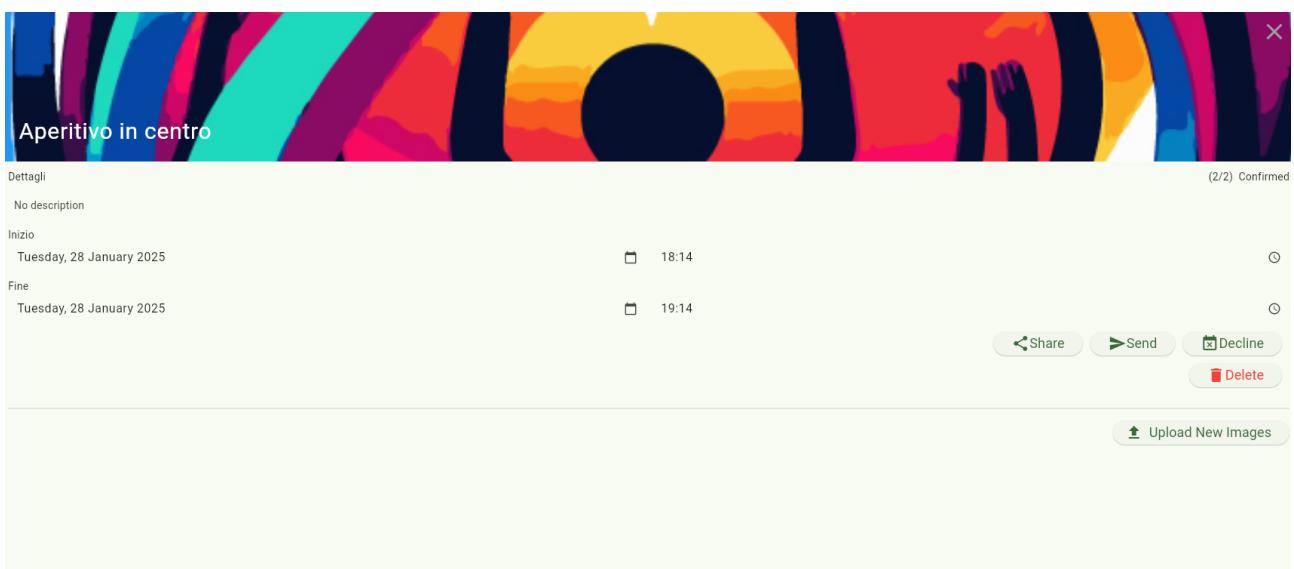


Figura 5: Dettaglio evento

Diagramma di dettaglio: InterfacciaUtente - Gruppi

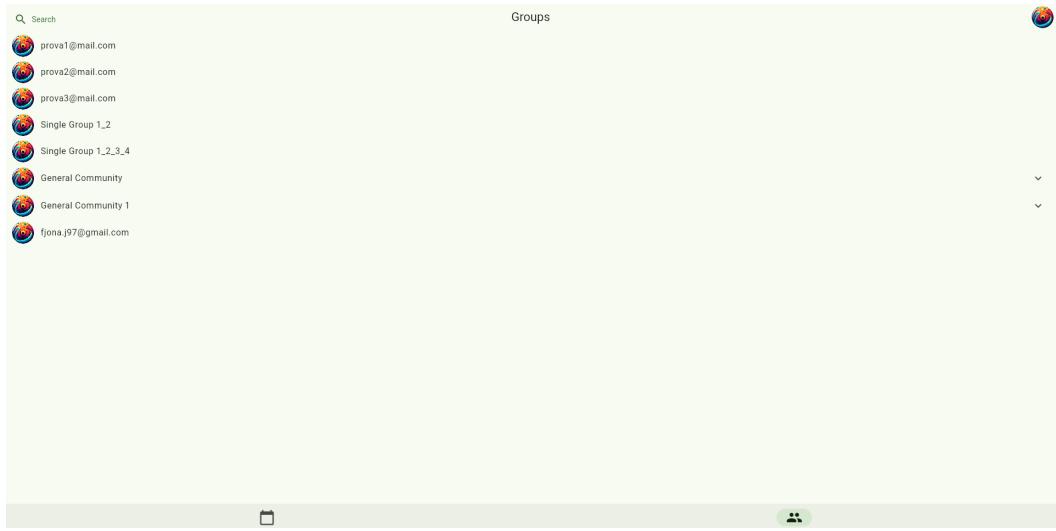
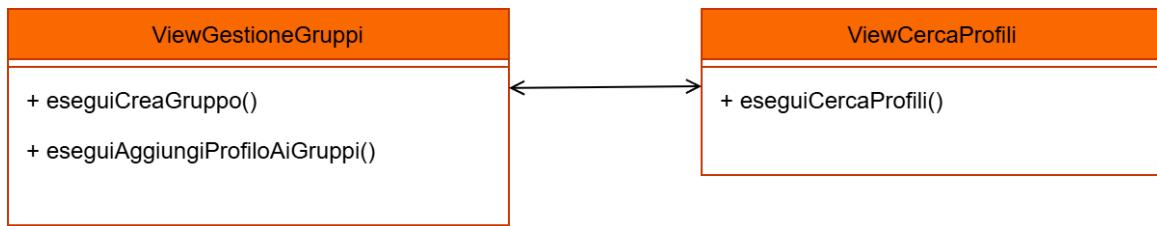


Figura 6: Gruppi



Figura 7: Cerca profili

Diagramma di dettaglio: InterfacciaUtente - Profili

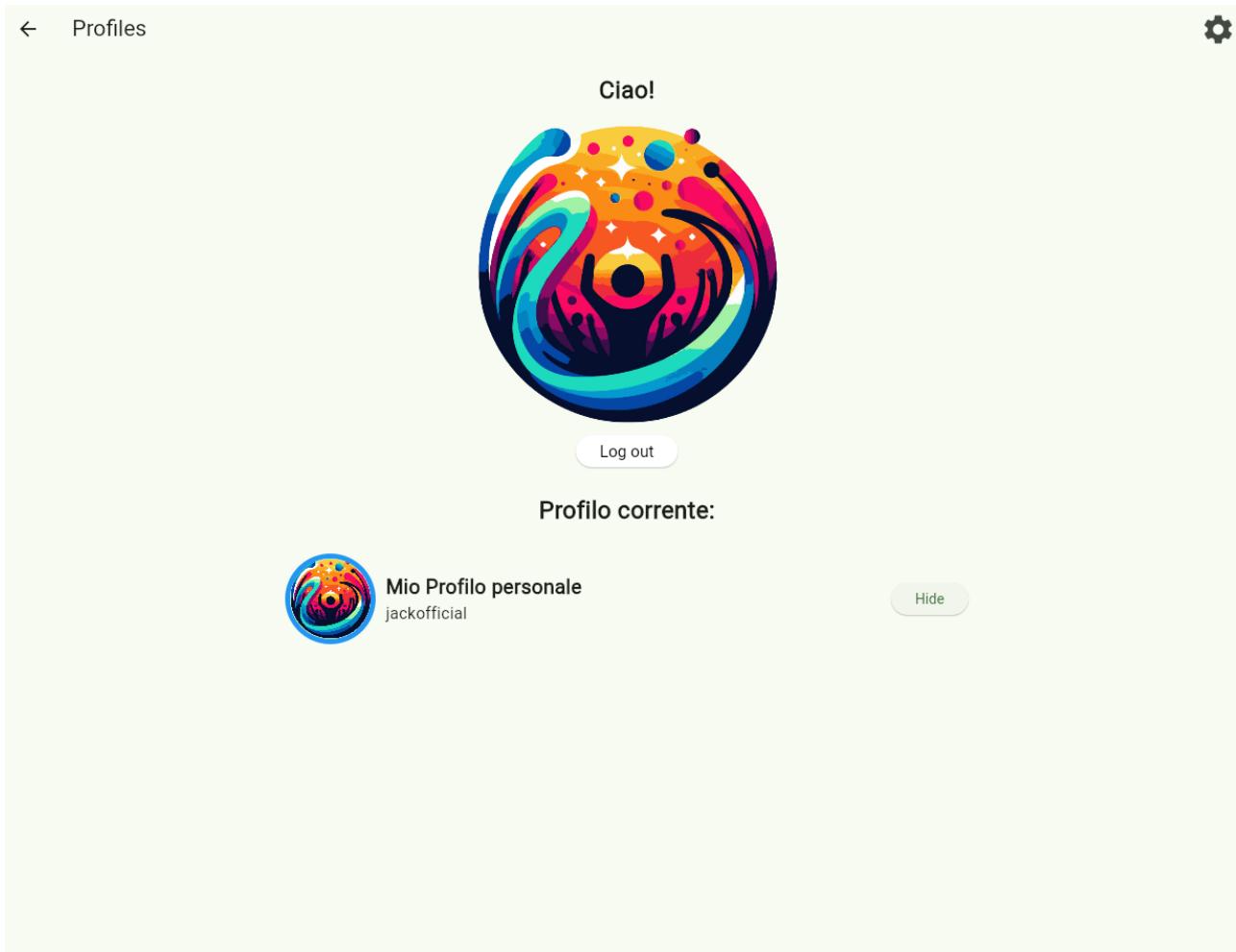


Figura 8: Profili collegati

Diagramma di dettaglio: InterfacciaGestioneAccesso

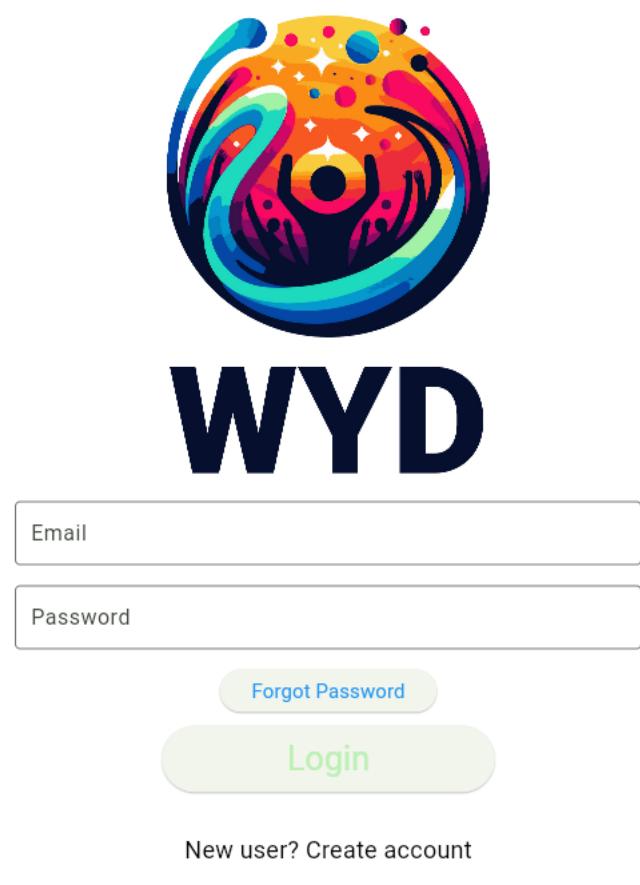
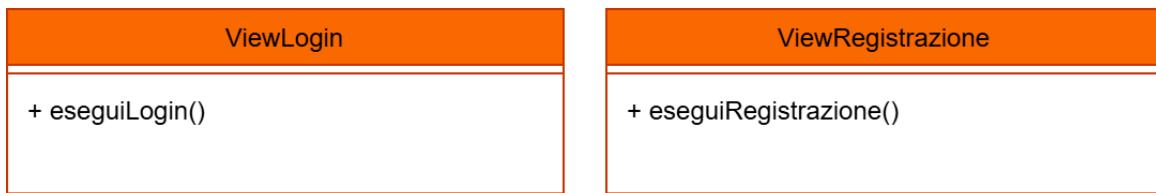


Figura 9: Login

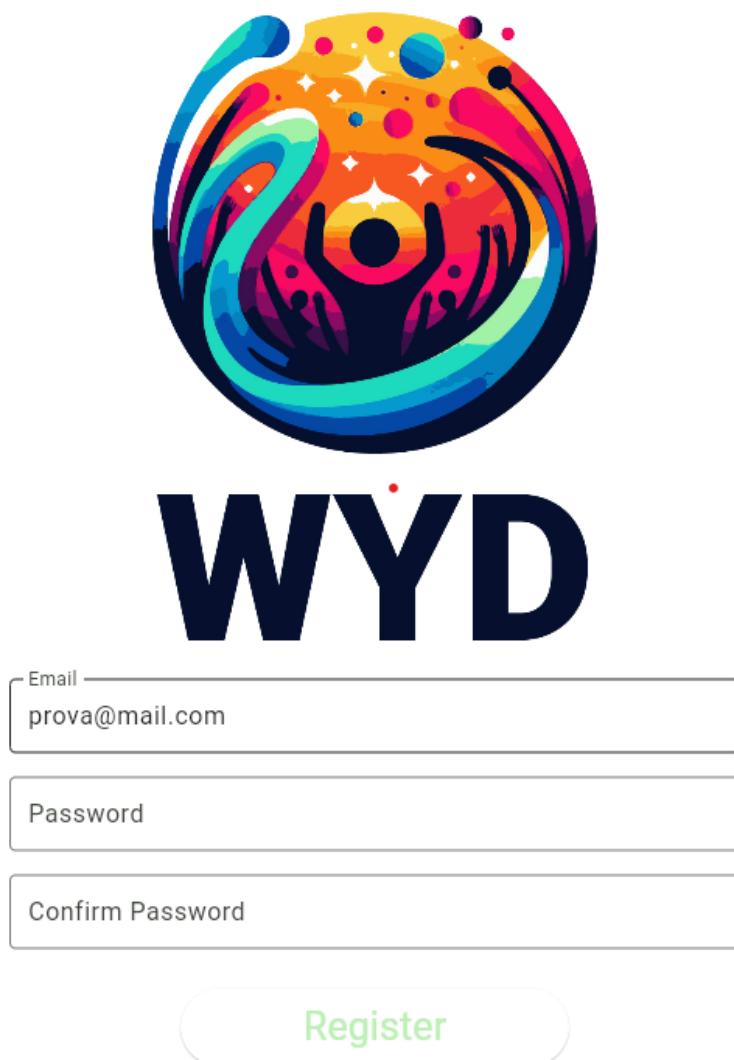


Figura 10: Registrazione

5.3 Interazione

Si riportano di seguito i vari diagrammi di sequenza, aggiornati rispetto a quelli visti in fase di analisi.

Diagramma di Sequenza: Registrazione - Client

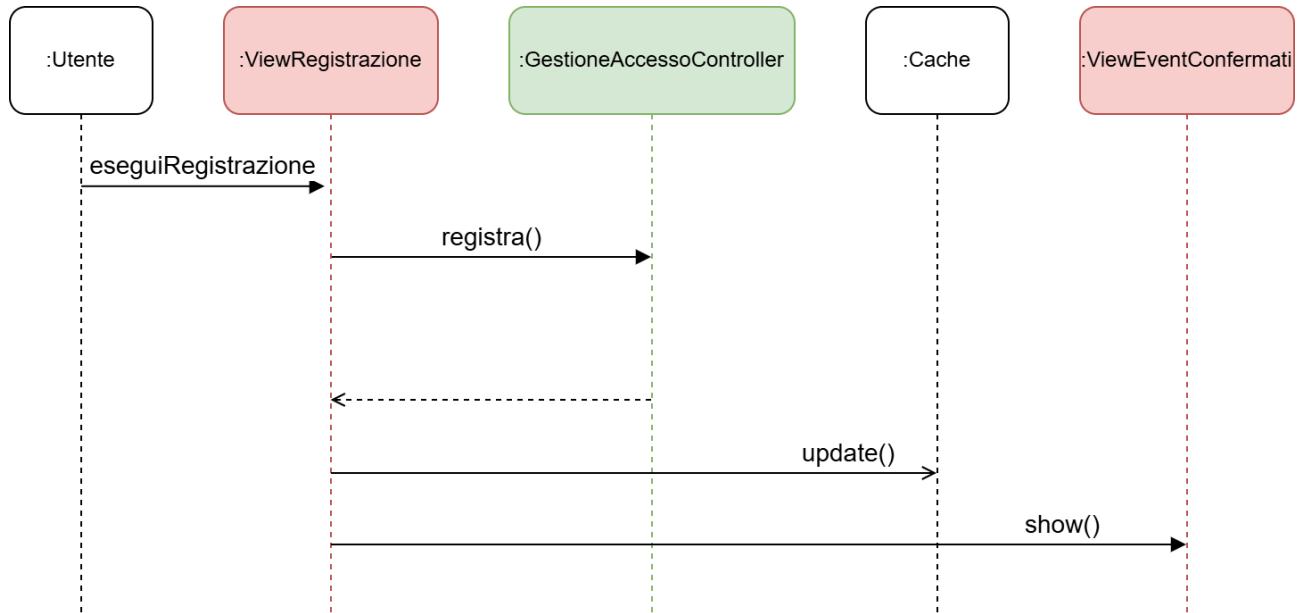


Diagramma di Sequenza: Registrazione - Server

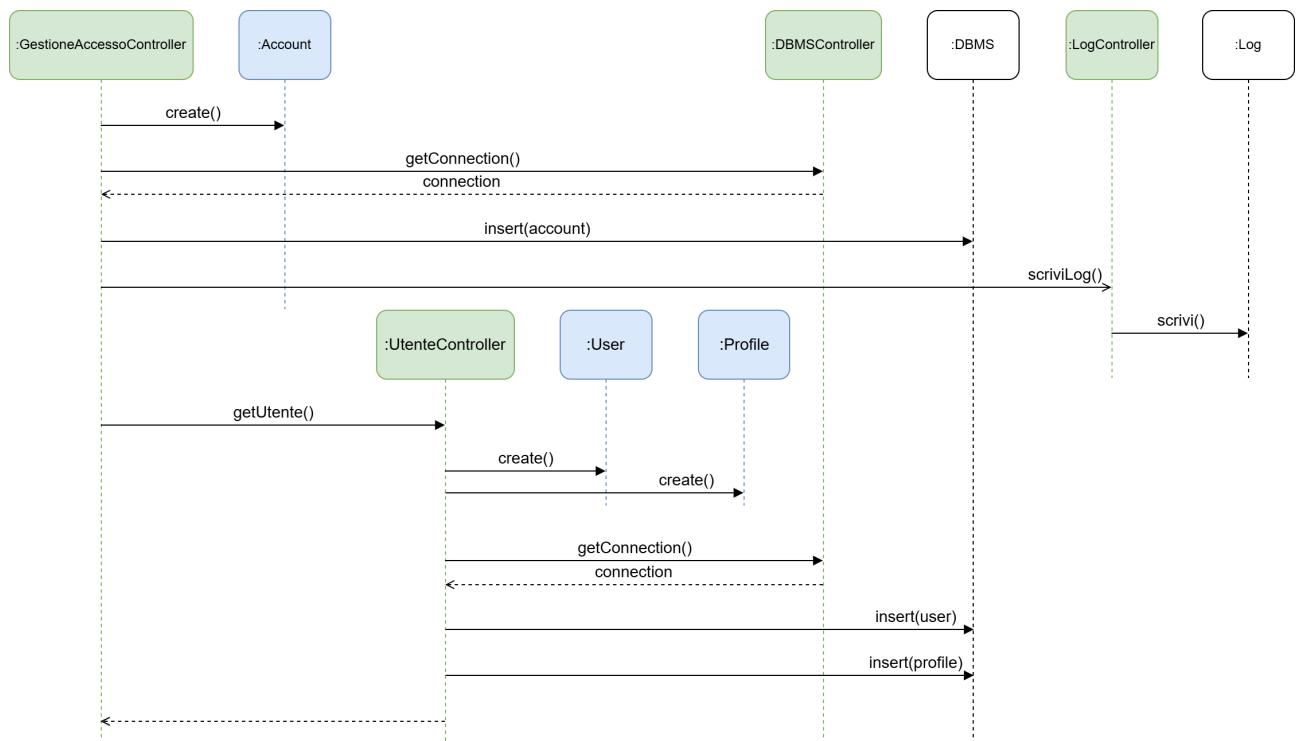


Diagramma di Sequenza: Visualizza Eventi Confermati

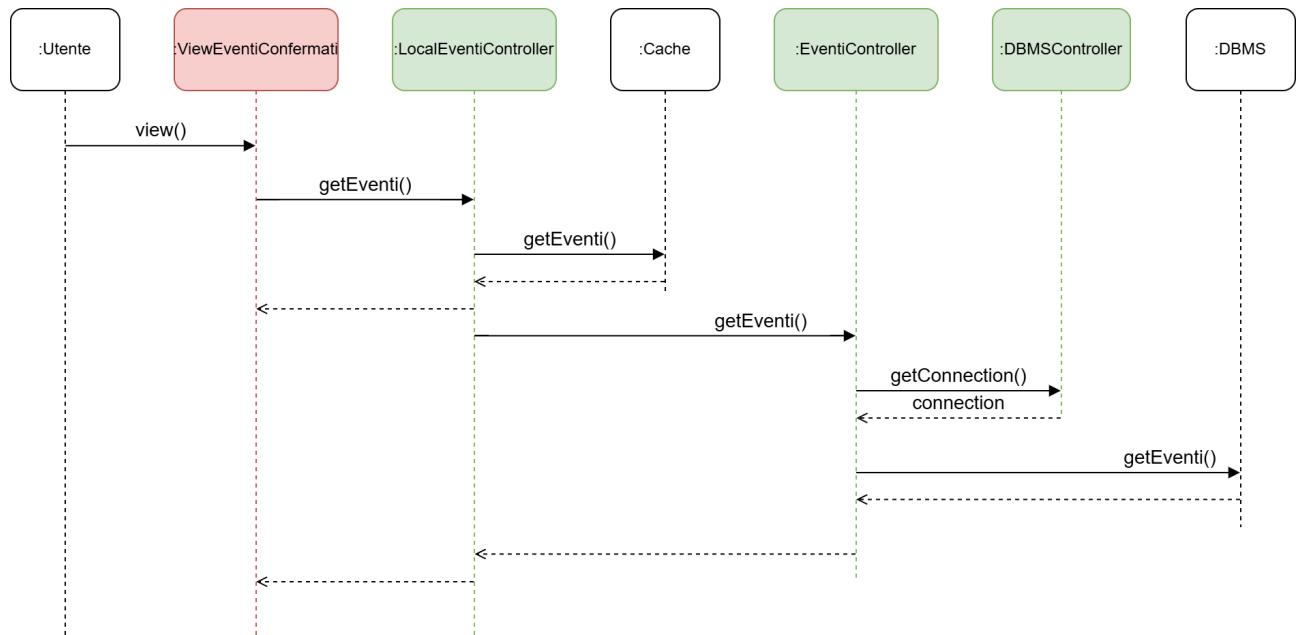


Diagramma di Sequenza: Visualizza Evento

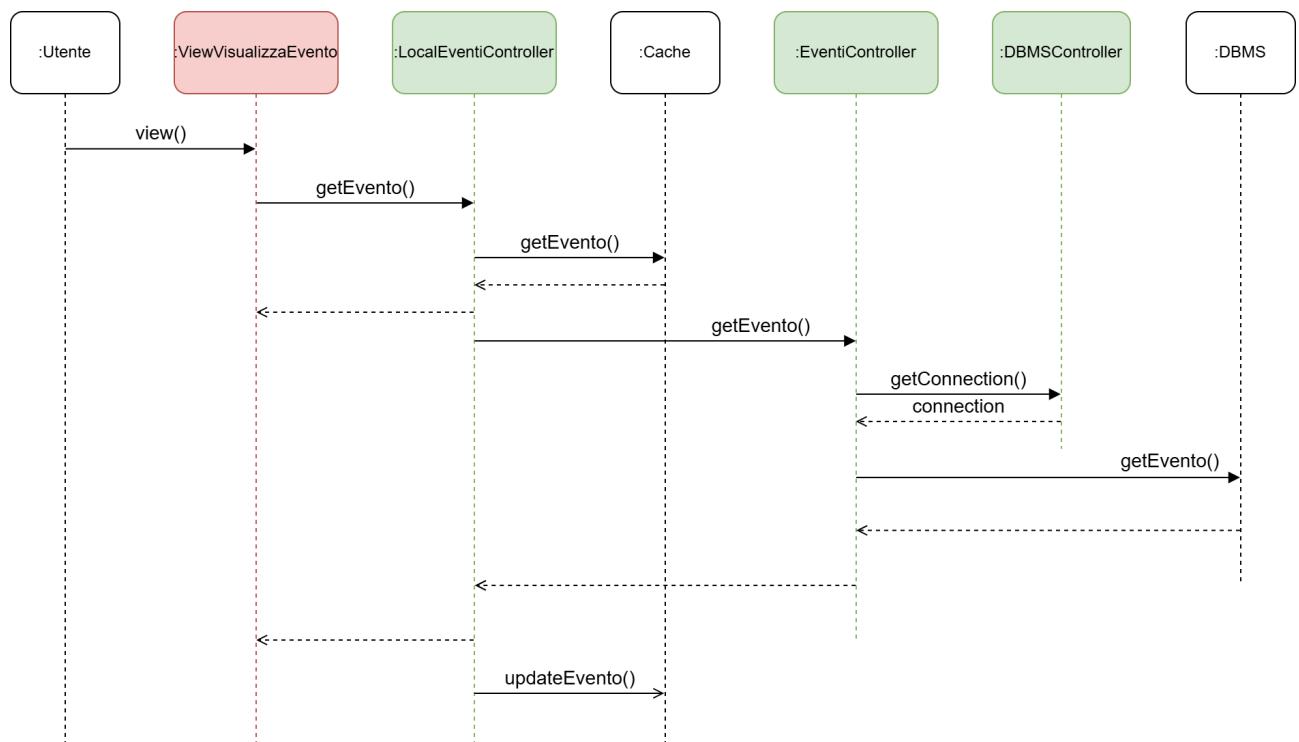


Diagramma di Sequenza: Crea Evento

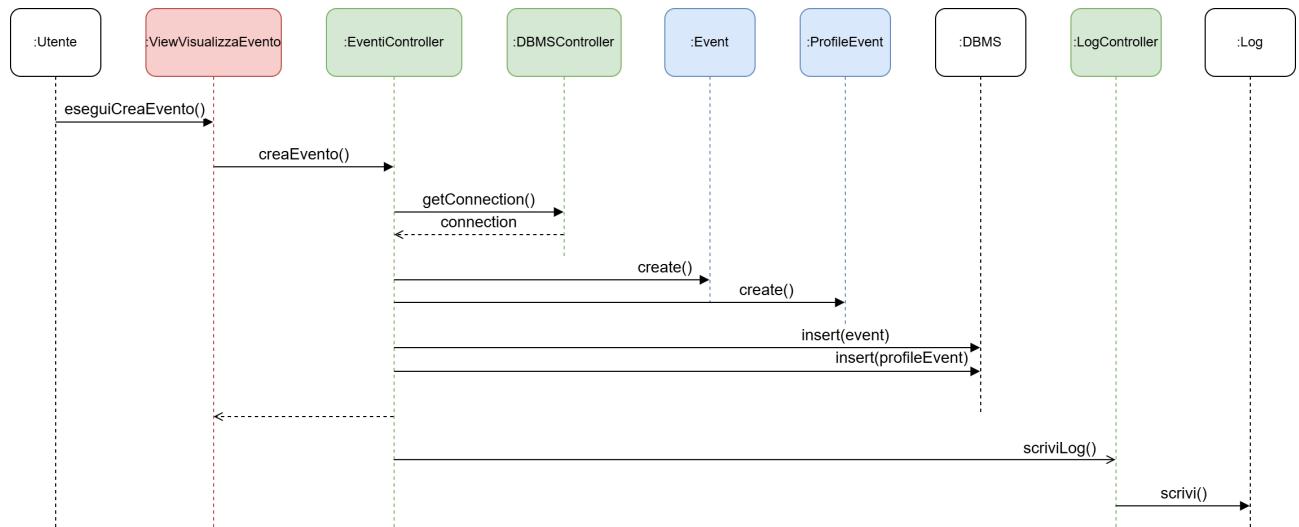


Diagramma di Sequenza: Modifica Evento

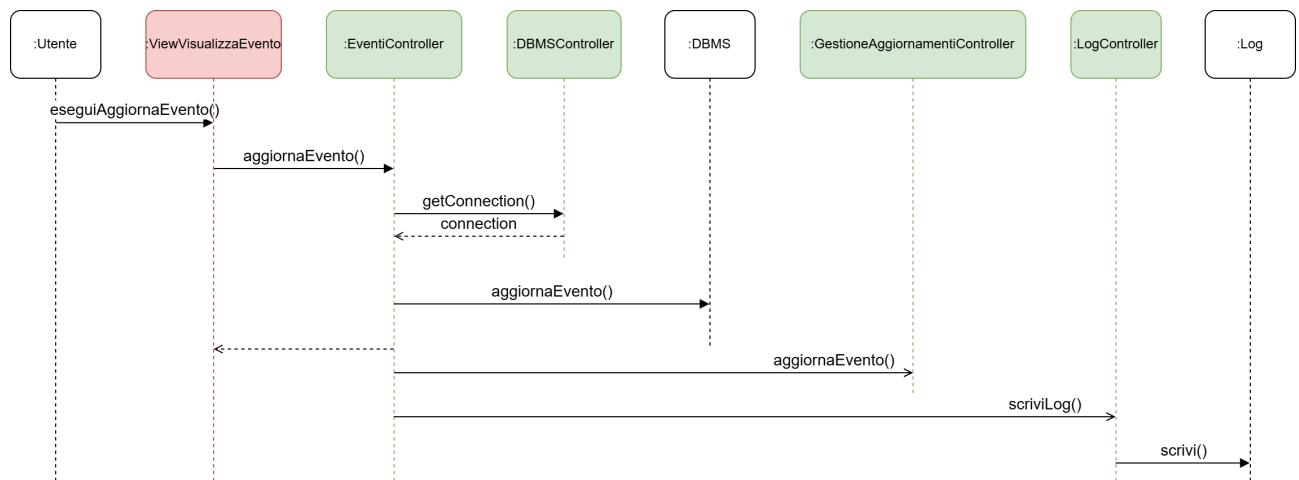


Diagramma di Sequenza: Aggiorna Evento

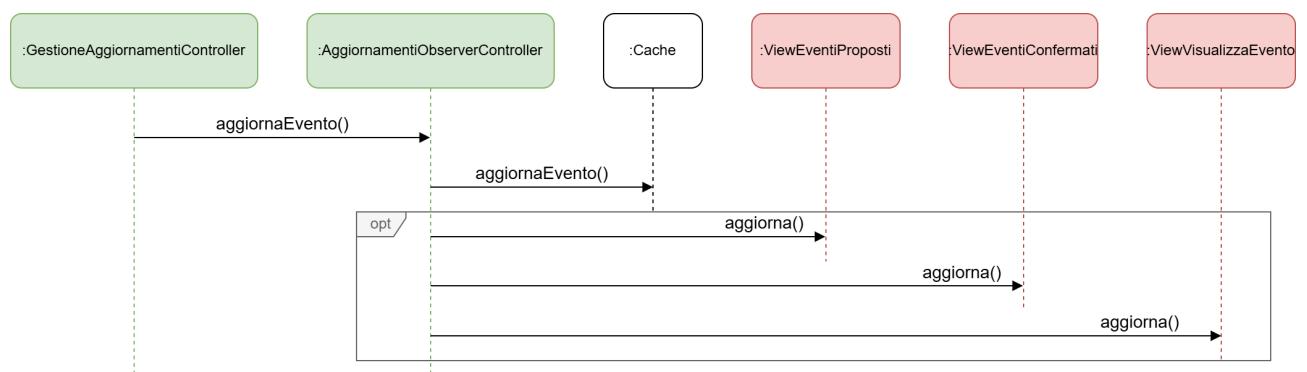


Diagramma di Sequenza: Conferma Evento

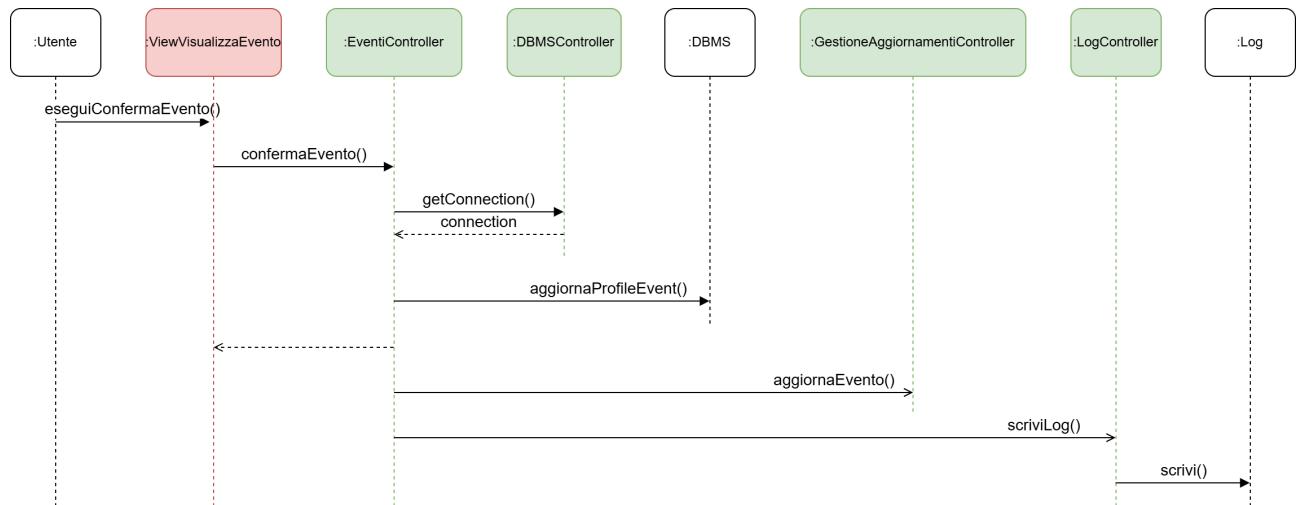


Diagramma di Sequenza: Recupera Immagini - Client

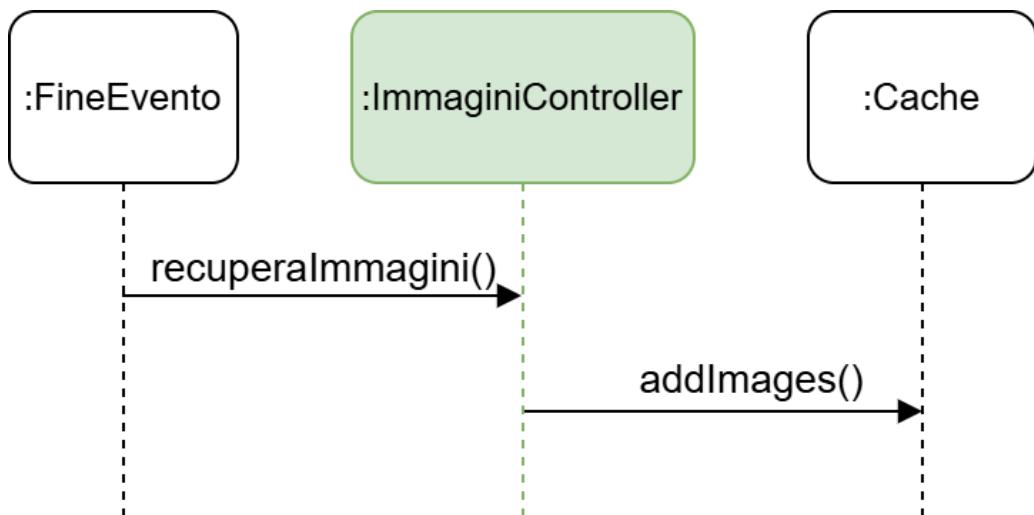


Diagramma di Sequenza: Conferma Immagini - Client

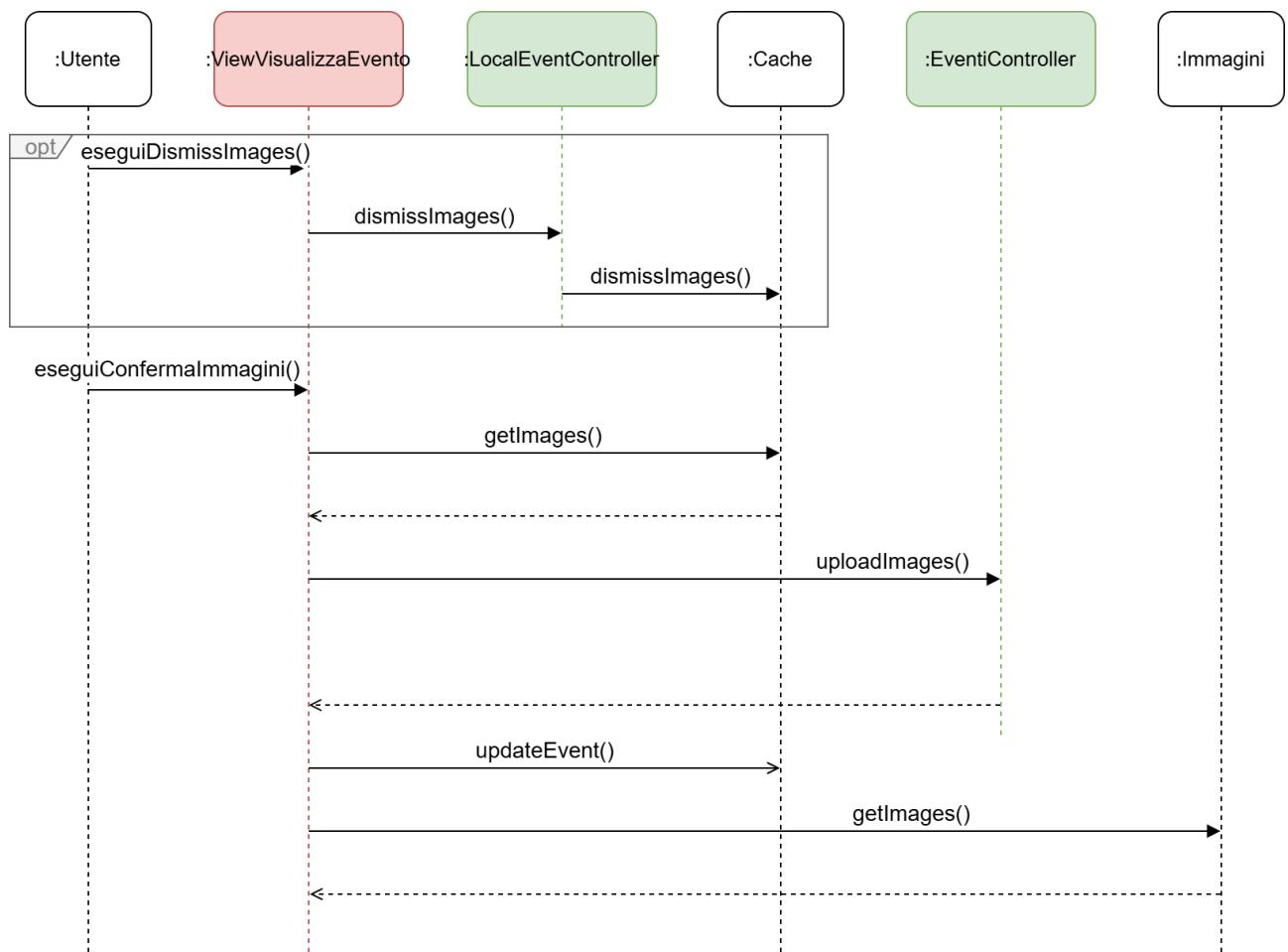


Diagramma di Sequenza: Conferma Immagini - Server

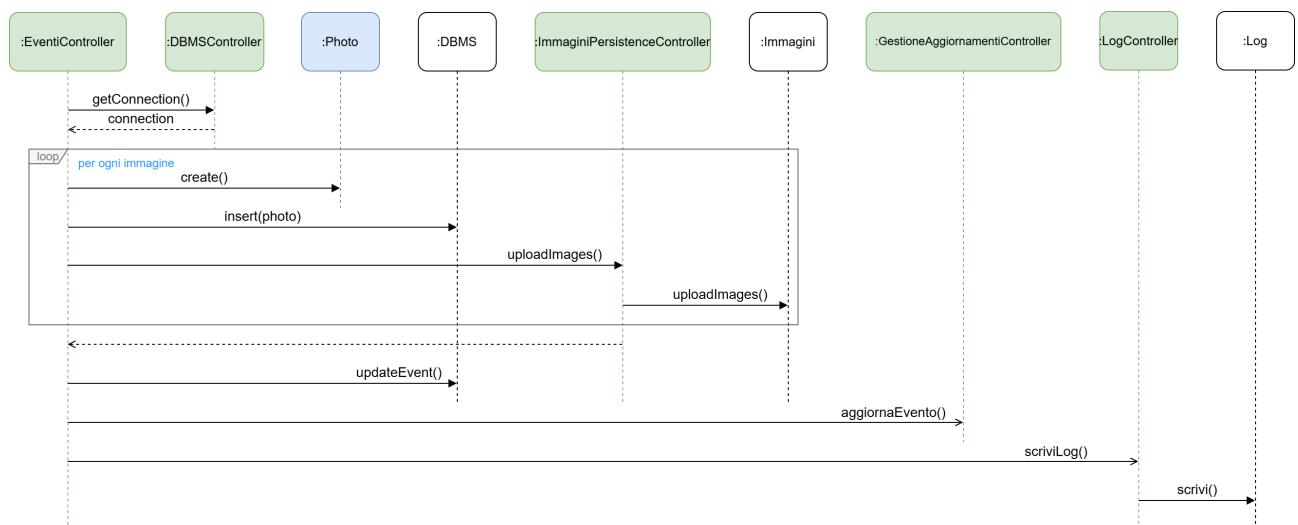


Diagramma di Sequenza: Condividi Evento ai Gruppi

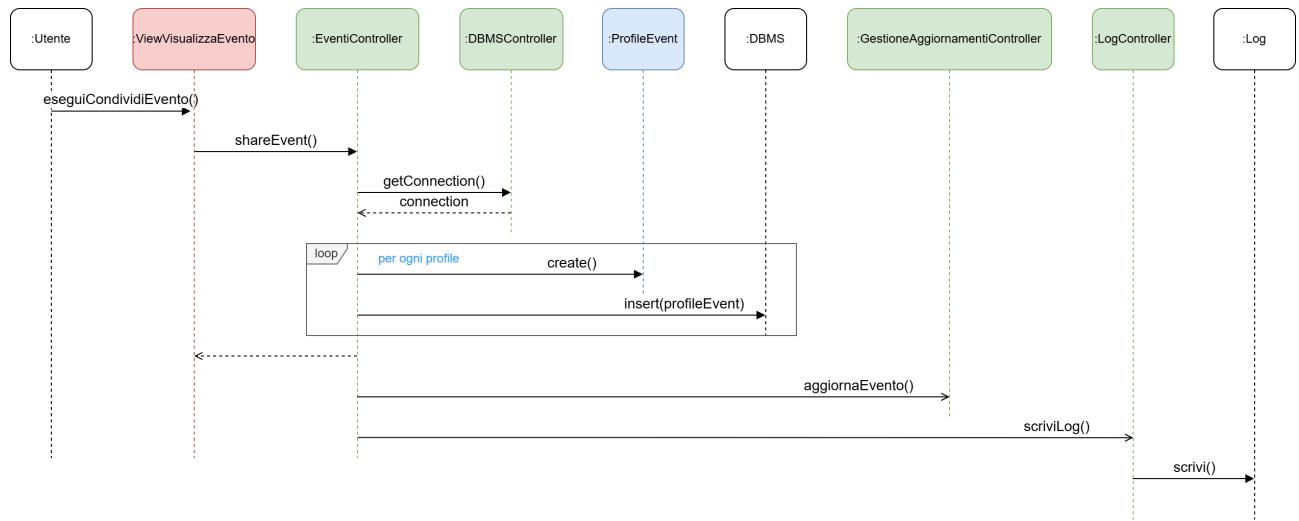
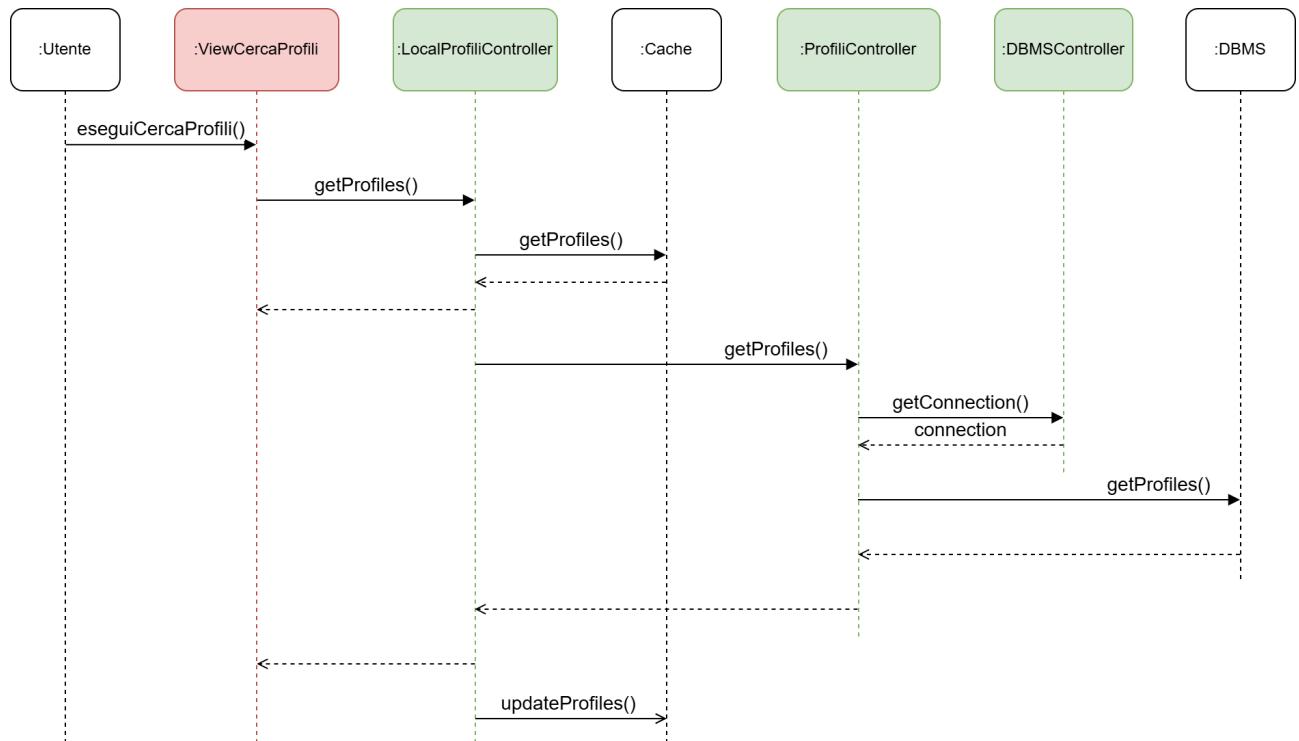
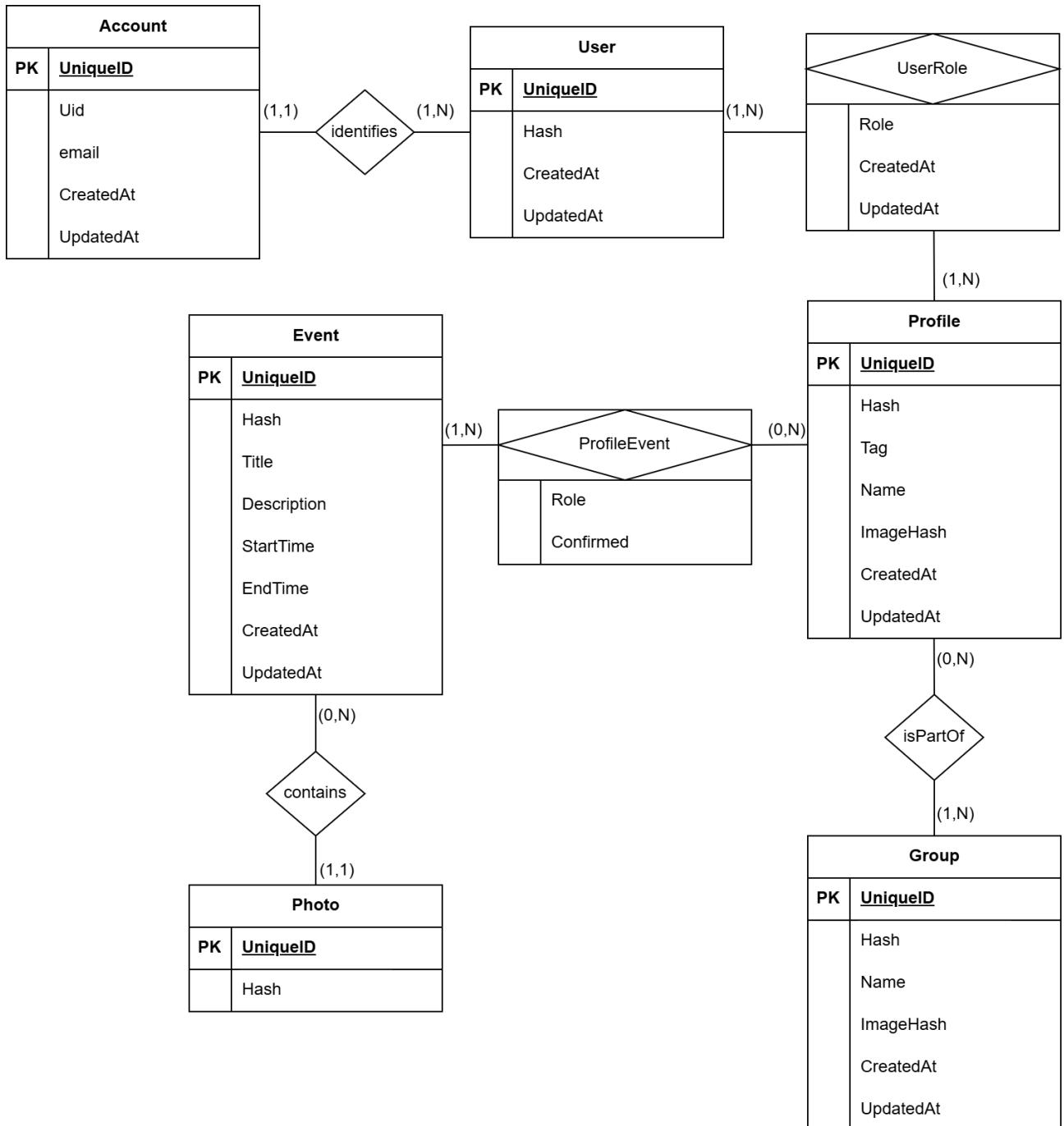


Diagramma di Sequenza: Cerca profili



5.4 Progettazione della persistenza

5.4.1 Diagramma E-R



Come si può notare, il diagramma E-R della persistenza segue precisamente la struttura del modello del dominio mostrato precedentemente. La differenza sta nelle associazioni, che presumibilmente in fase di progettazione logica ed implementazione del database verranno concretizzate in classi di associazione, si avranno quindi due tabelle ulteriori (**ProfileEvent** e **UserRole**) per modellare le associazioni.

5.4.2 Formato dei file di log

Il formato del file di log su cui il sistema terrà traccia delle operazioni sarà il seguente:

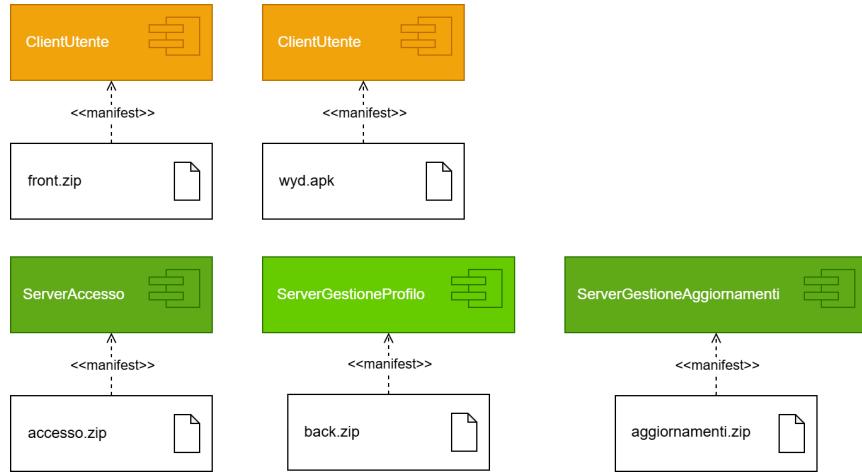
Esempio: File `/var/log/wyd.log`

`$ Data - Ora - Operazione - Descrizione - ID utente`

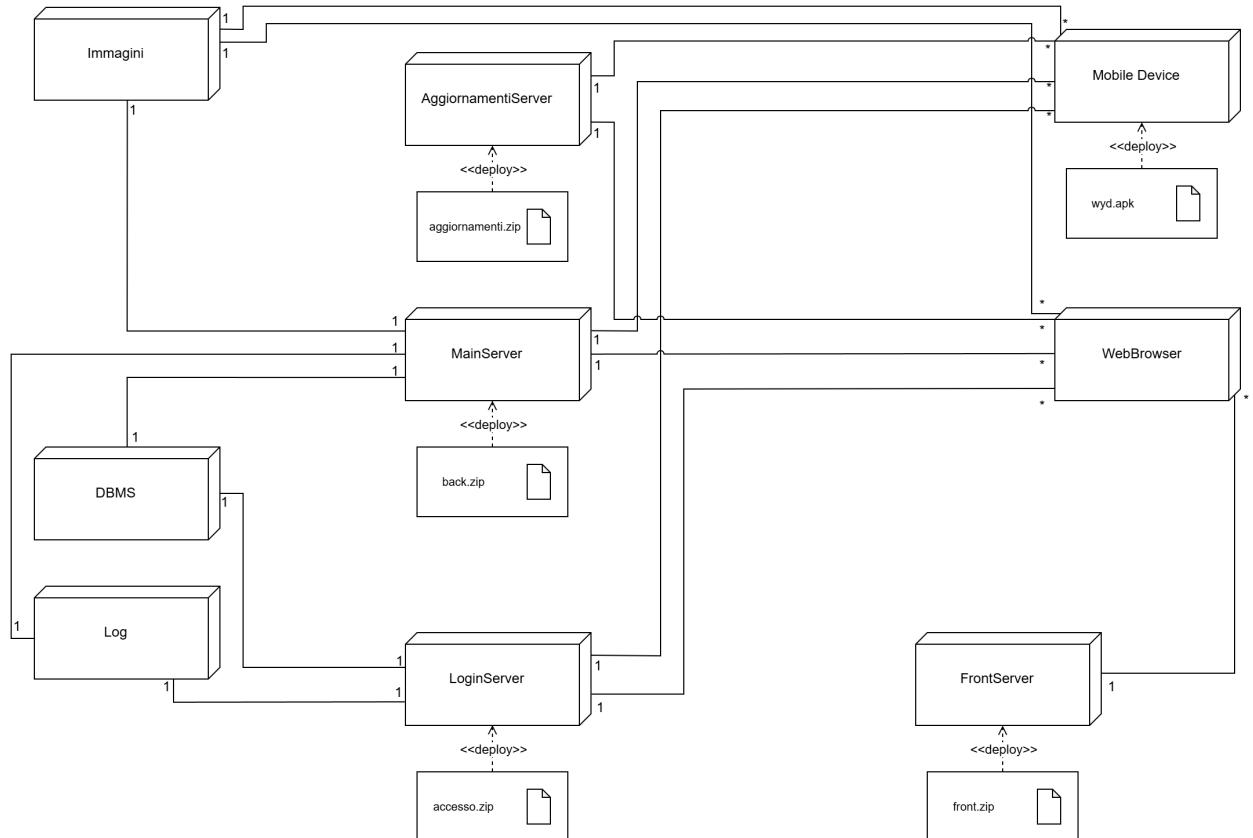
Nota: l'ID utente è l'identificativo dell'esecutore di tale operazione.

5.5 Deployment

5.5.1 Artefatti



5.5.2 Deployment Type-Level



6 Implementazione

6.1 Progettazione Architetturale

6.1.1 Requisiti non funzionali

Per garantire la scalabilità e l'affidabilità del progetto useremo tecnologie in cloud, che garantiscono prestazioni altrimenti irraggiungibili ad un costo contenuto, soprattutto nelle fasi iniziali del progetto.

Questo comporta un sostanziale riduzione del carico di lavoro, in quanto al gestore cloud verranno delegati i requisiti di affidabilità, scalabilità, protezione da attacchi Ddos e la velocità delle comunicazioni.

6.1.2 Scelte tecnologiche

Per familiarità con il sistema, disponibilità dei fondi e servizi offerti, la scelta del gestore cloud ricade su:

- Google Firebase, per la gestione degli accessi
- Microsoft Azure, per la logica applicativa e la persistenza.

In particolare, Google Firebase Authentication fornisce un sistema di autenticazione personalizzabile, oltre a dare la possibilità di gestire più identity providers. I dati relativi agli utenti, quindi in particolare l'email e la password verranno conservati nella memoria interna al servizio, senza che quindi sia richiesta una gestione da parte del programma.

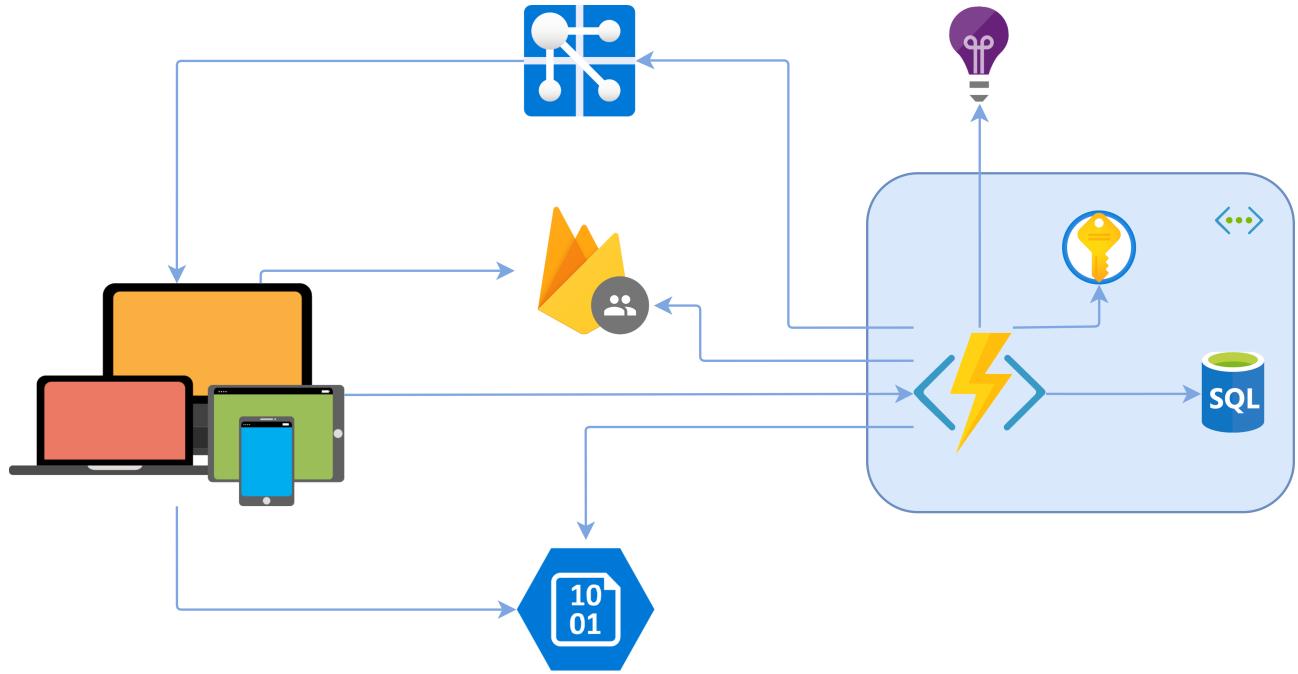
La logica di business dell'applicazione verrà sviluppata su Azure Functions, un servizio Faas che scala automaticamente le richieste al server. La persistenza logica sarà affidata a Azure SQL Server, un database relazionale gestito dalla piattaforma, mentre le foto e i file multimediali verranno salvati su Azure Storage Container. Per aggiornare i devices in tempo reale verrà usato Azure PubSub, che permette di creare un canale di comunicazione con ogni dispositivo.

Infine, per la fruizione del servizio tramite browser useremo Azure Static Web App, e, in attesa del rilascio ufficiale sul Play Store dell'applicazione, l'applicazione mobile sarà resa disponibile tramite Azure Storage Container.

Per quanto riguarda lo sviluppo, i principali linguaggi di programmazione saranno:

- Flutter per l'interfaccia grafica, che permette di racchiudere in un'unica implementazione applicazioni per dispositivi di tecnologia differente
- C# per la logica di business: nativamente supportato dalle Functions, permette un collegamento diretto tra oggetti logici e database, semplificando la gestione della persistenza.

6.1.3 Scelta dell'architettura



Nel grafico possiamo notare come al centro dell'architettura ci siano le Azure Functions che, all'interno della stessa rete, comunicano con il database relazionale e il Key Vault. In base alla richiesta, che partirà sempre dal dispositivo dell'utente, le Functions potranno poi interagire con il servizio Pub-Sub per gli aggiornamenti in tempo reale, Google Firebase per l'autenticazione o Azure Storage Container per le immagini.

I devices degli utenti avranno quindi la possibilità di contattare autonomamente, oltre alle Functions, sia Google Firebase che il server per la persistenza delle immagini.

6.2 Monitoraggio

Il monitoraggio del sistema avverrà in due modi: tramite salvataggio dei log e controllo delle prestazioni del sistema.

Per quanto riguarda Firebase Authentication vengono forniti con il servizio sia le interfacce per il controllo delle prestazioni che la gestione dei log.

Per monitorare le Azure Functions sarà necessario collegare Azure Application Insights che provvede a controllare il funzionamento e la risposta del servizio. La creazione dei log è invece affidata al programmatore, che verranno poi salvati tramite function su un file sistem dedicato su un Azure Container.

6.3 Sicurezza

Per garantire la sicurezza nelle comunicazioni ogni interazione avverrà utilizzando un canale sicuro cifrato grazie al protocollo HTTPS.

Tutte le credenziali e variabili di ambiente usate dalle Azure Functions per collegarsi ai database e ai vari servizi saranno conservate e ottenute tramite Azure Key Vault, per garantire la confidenzialità delle credenziali.

Il server di accesso rilascerà un token, unico per ogni utente, che verrà allegato ad ogni richiesta, per essere usato dal server principale per identificare l'utente ed eventualmente controllare i permessi relativi. Il token deve permettere di identificare ed autenticare univocamente l'utente.

La generazione degli hash di identificazione dei componenti deve garantire una diffusione tale da rendere altamente improbabile una collisione.

Le richieste di salvataggio dati non possono superare i 100 Kb di dimensione, e ogni utente non può fare in un arco orario richieste in scrittura che vadano complessivamente oltre i 20 MB. Come evidenziato precedentemente, la difesa da attacchi di tipo denial of service verranno affidati al cloud provider.

Inoltre, sarà necessario che tutto il codice, in fase di sviluppo, venga controllato da esperti di sicurezza, per ridurre il più possibile l'integrazione di falle nel sicurezza.

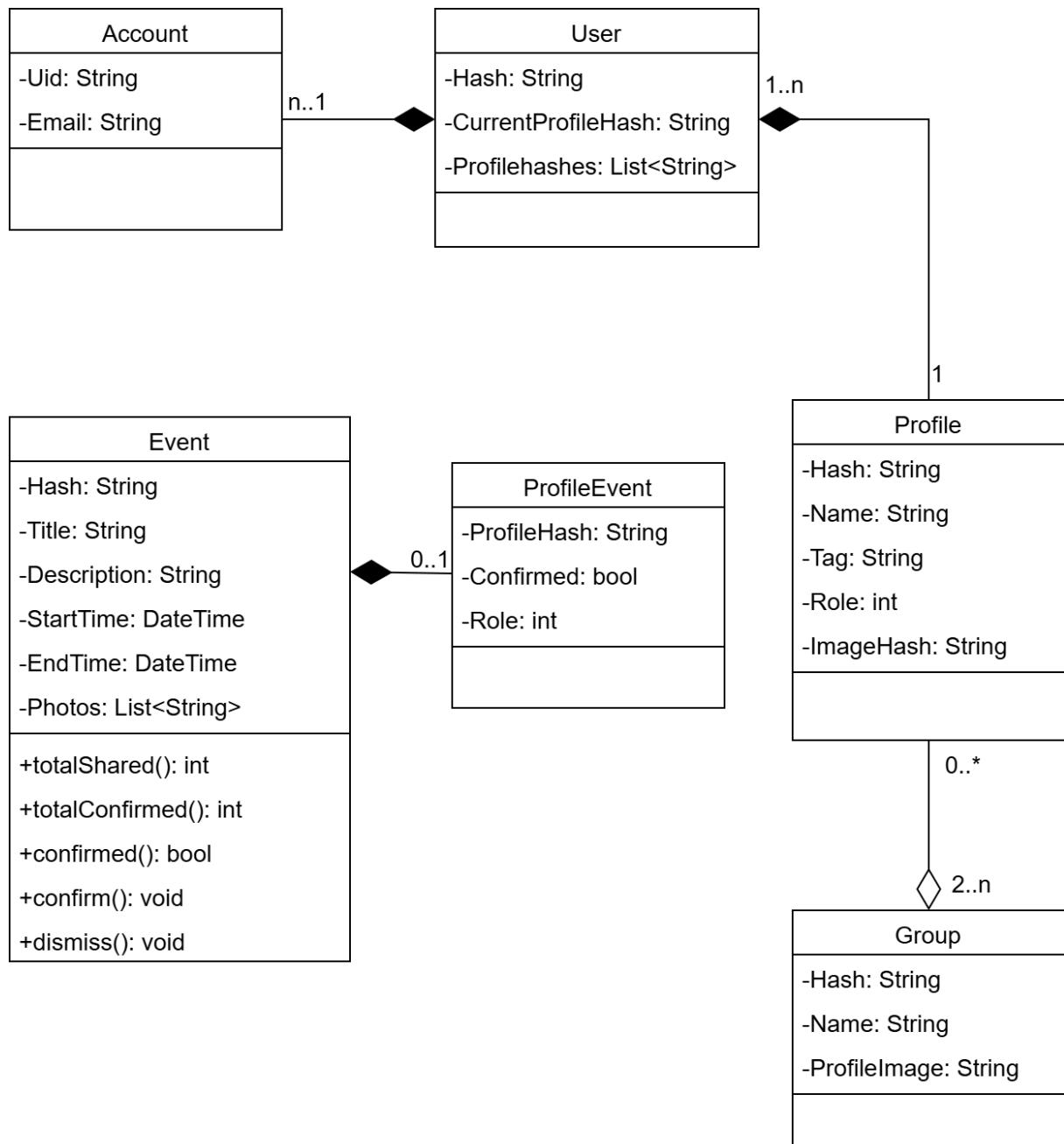
6.4 Progettazione di dettaglio

Di seguito verranno schematizzati alcuni esempi generali per mostrare le modifiche da apportare in seguito alle scelte implementative, come linee guida per il resto del programma. I casi particolari di interazione verranno dettagliati nella relativa sezione.

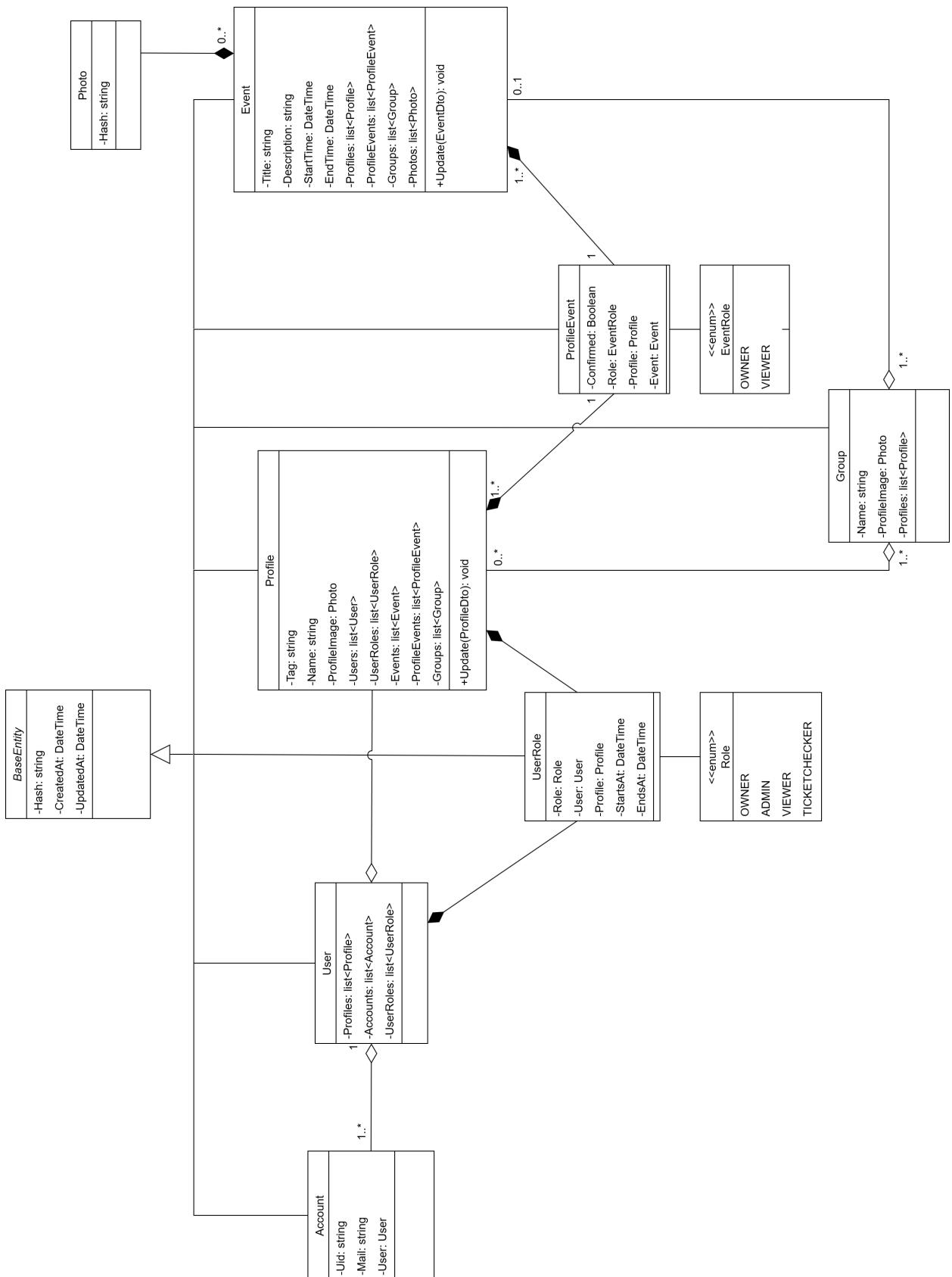
6.4.1 Struttura: Dominio

Sia per il client che per il server, ogni entità descritta nel dominio verrà mappata con una classe corrispondente.

Modello delle Classi: Dominio Client



Modello delle Classi: Dominio Server



Infine, per la conversione delle classi di dominio a quelle di risposta, si implemetteranno dei data tranfert objects(DTO) che permetteranno una mappatura corretta con il dominio dei client.

Modello delle Classi: DTO

ProfileEventDto	GroupDto	ProfileDto	EventDto
-ProfileHash: String	-Hash: String	-Hash: String	-Hash: String
-Confirmed: bool	-Name: String	-Name: String	-Title: String
-Role: int	-ProfileImage: String	-Tag: String	-Description: String
		-Role: int	-StartTime: DateTime
		-ImageHash: String	-EndTime: DateTime
			-Photos: List<String>
			-ProfileEvents: List<ProfileEventDto>
UserDto	AccountDto		+totalShared(): int
-Hash: String	-Uid: String		+totalConfirmed(): int
-CurrentProfileHash: String	-Email: String		+confirmed(): bool
-Profiles: List<ProfileDto>			+confirm(): void
			+dismiss(): void

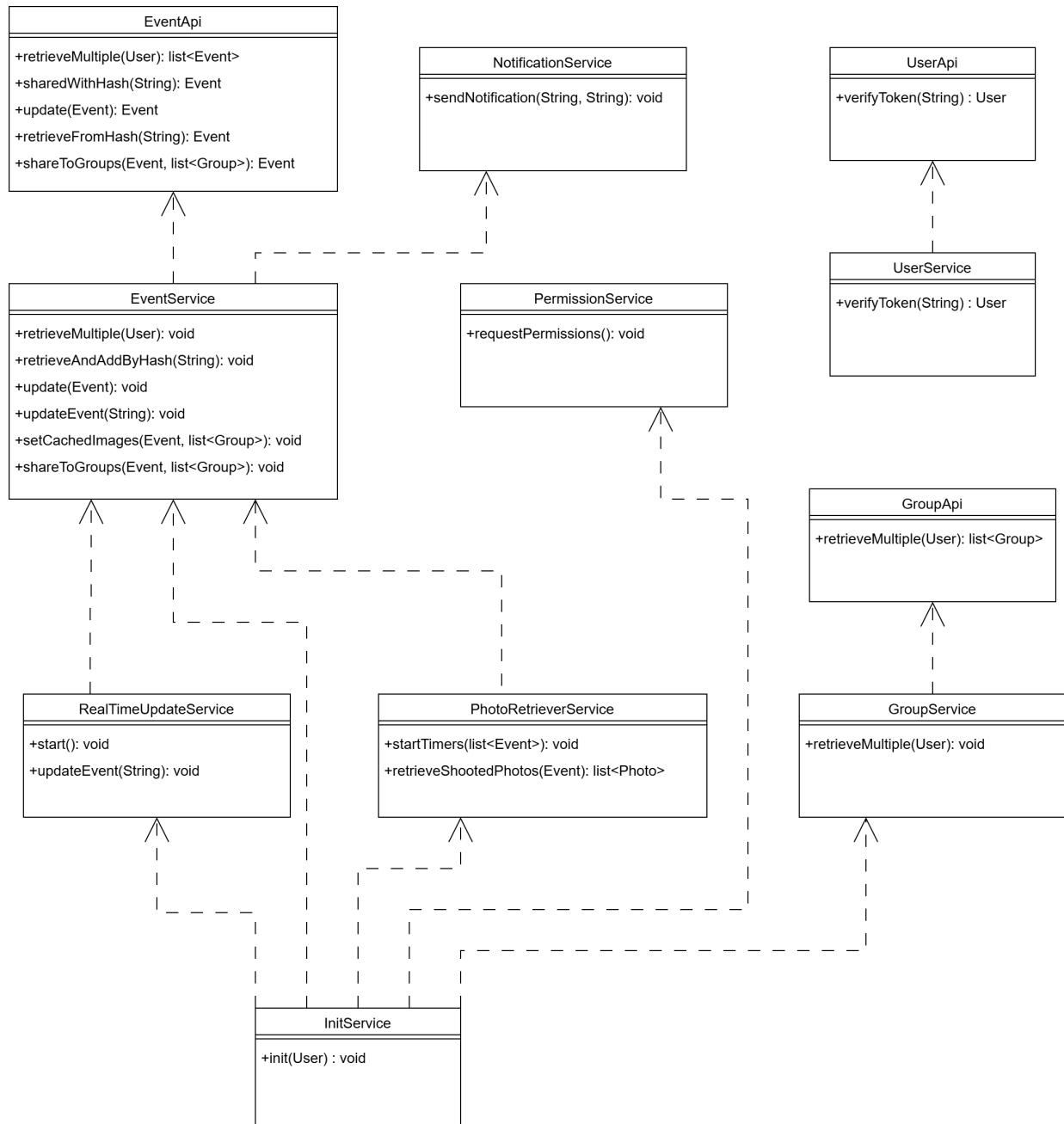
6.4.2 Struttura: Controller - Client

Ogni richiesta di dati verso l'esterno verrà implementata grazie a classi che nascondono i dettagli necessari per la comunicazione, chiamate API.

Inoltre, funzionalità che vengono ripetute in parti diverse del programma verranno delegate a classi chiamate servizi, che raggrupperanno le richieste per affinità logica.

A titolo esemplificativo, si riportano alcune classi.

Modello delle Classi: Servizi



Flutter ha una gestione dello stato che si basa sulla vita dei componenti a cui è associato. La persistenza a livello locale ha quindi durata limitata e ricade nella normale programmazione del framework.

La persistenza di livello applicativo verrà suddivisa in base alle classi logiche del dominio e mantenuta in classi Provider. Le classi seguiranno un pattern singleton e verranno create all'avvio dell'applicazione.

Modello delle Classi: Provider

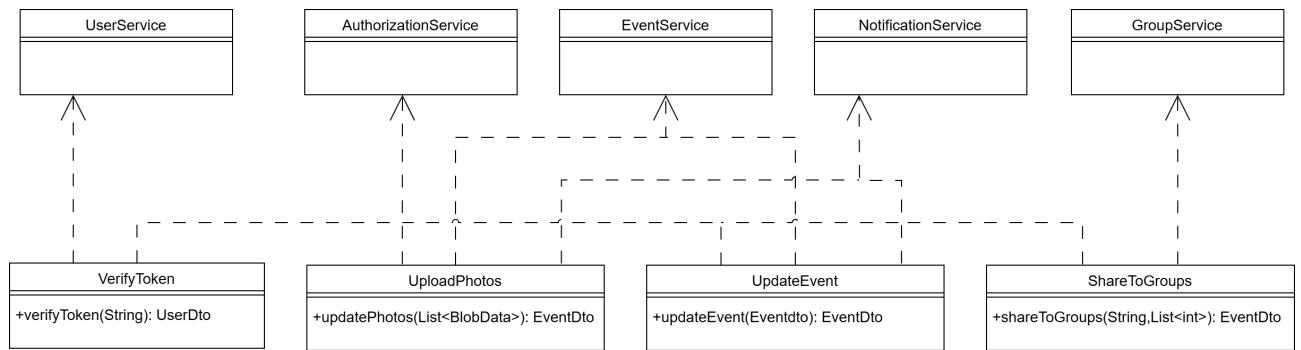
AuthProvider	UriProvider	EventProvider
-instance: AuthProvider	-instance: UriProvider	-instance: EventProvider
_auth: FirebaseAuth	-uri: String	-events: list<Event>
+login(String, String): void	+setUri(String): void	+addAll(list<Event>): void
+register(String, String): void	+getUri(): String	+add(Event): void
		+update(Event): void
		+updateEvent(Event): void
UserProvider	ProfilesProvider	
-instance: UserProvider	-instance: ProfilesProvider	
-user: User	-profiles: list<Profile>	
+updateUser(User): void	+addAll(list<Profile>): void	

6.4.3 Struttura: Controller - Server

Per sfruttare le caratteristiche di scalabilità delle Functions, il codice dovrà essere suddiviso in nuclei di codice indipendenti tra loro, senza la permanenza di uno stato tra una richiesta e l'altra.

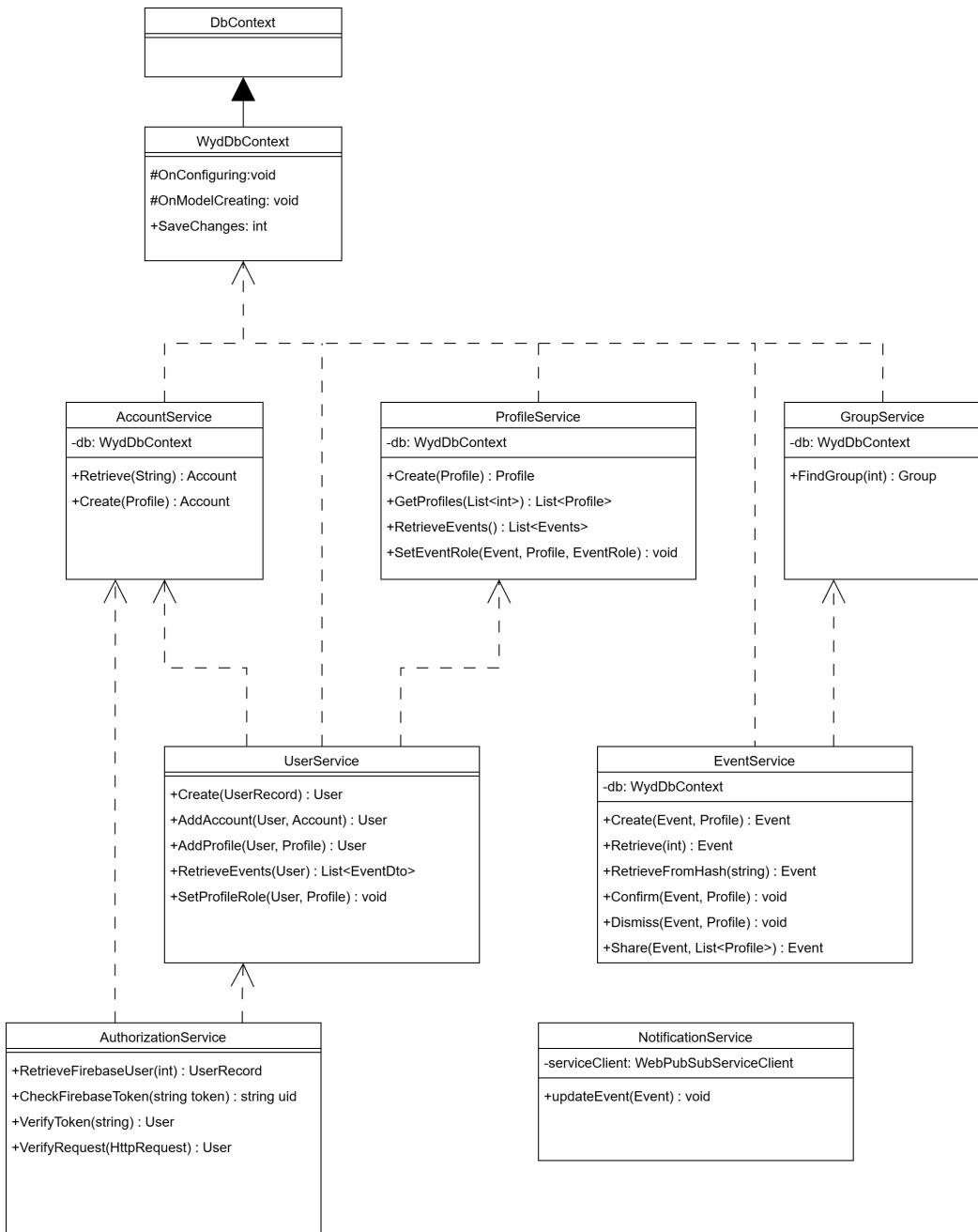
Anche in questo caso, alcune funzionalità potrebbero essere condivise da più function, per cui verranno raggruppate in classi service.

Modello delle Classi: Functions Endpoint



Sia nel caso del database, che nel caso del server per le notifiche, la connessione sarà nascosta da una classe apposita, rispettivamente DbContext e WebPubSubServiceClient.

Modello delle Classi: Servizi



6.5 Interazione

Diagramma di Sequenza: Avvio - Client

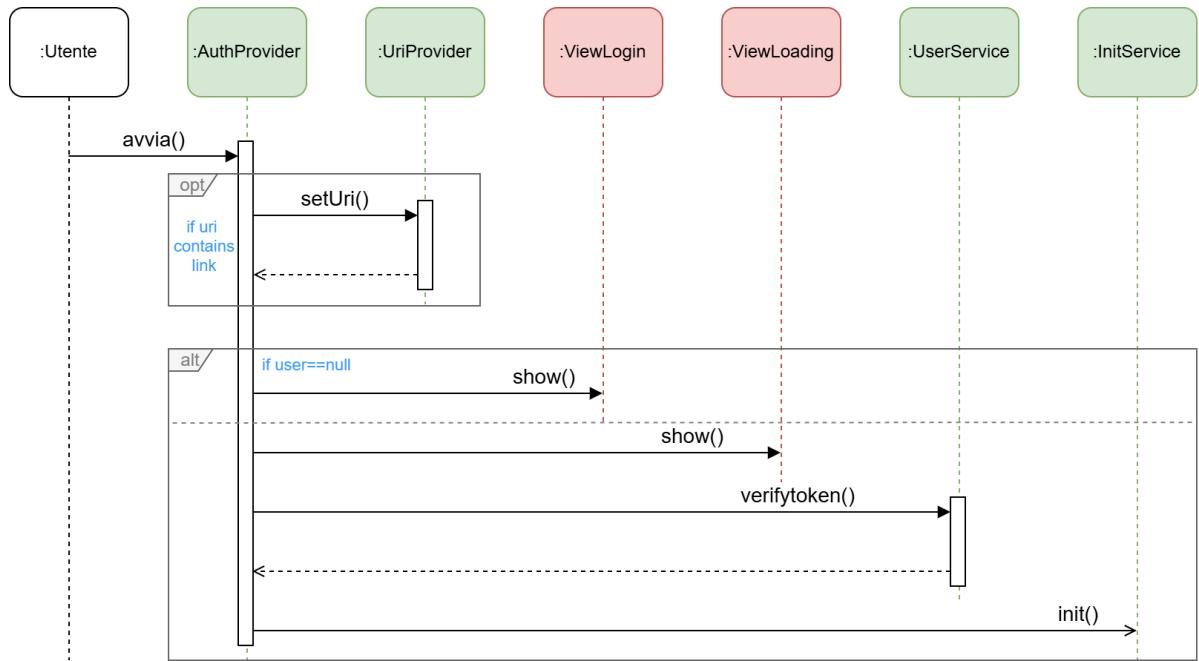


Diagramma di Sequenza: Login - Client

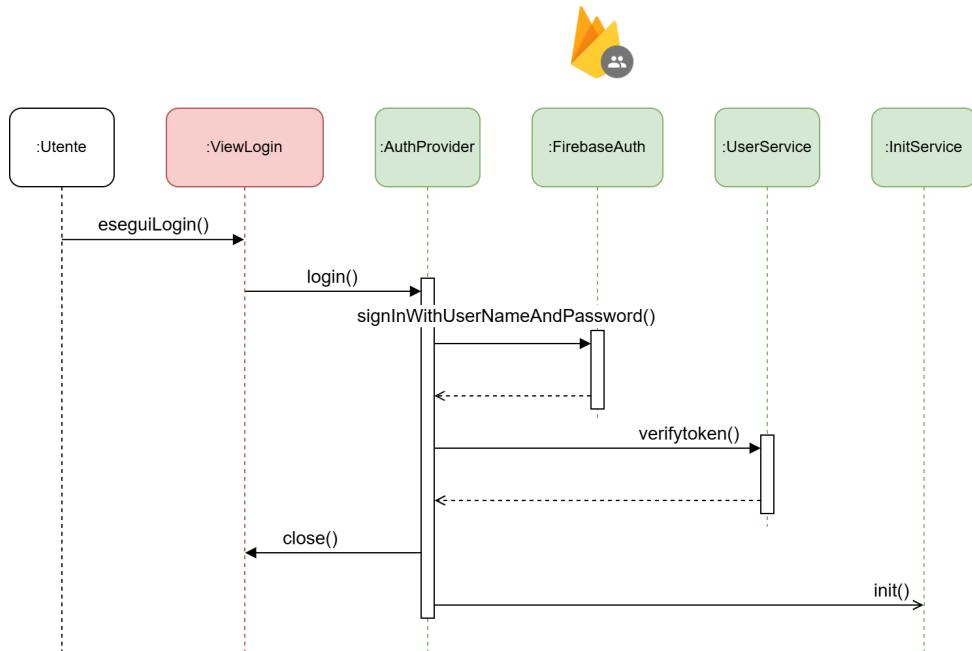


Diagramma di Sequenza: Registrazione - Client

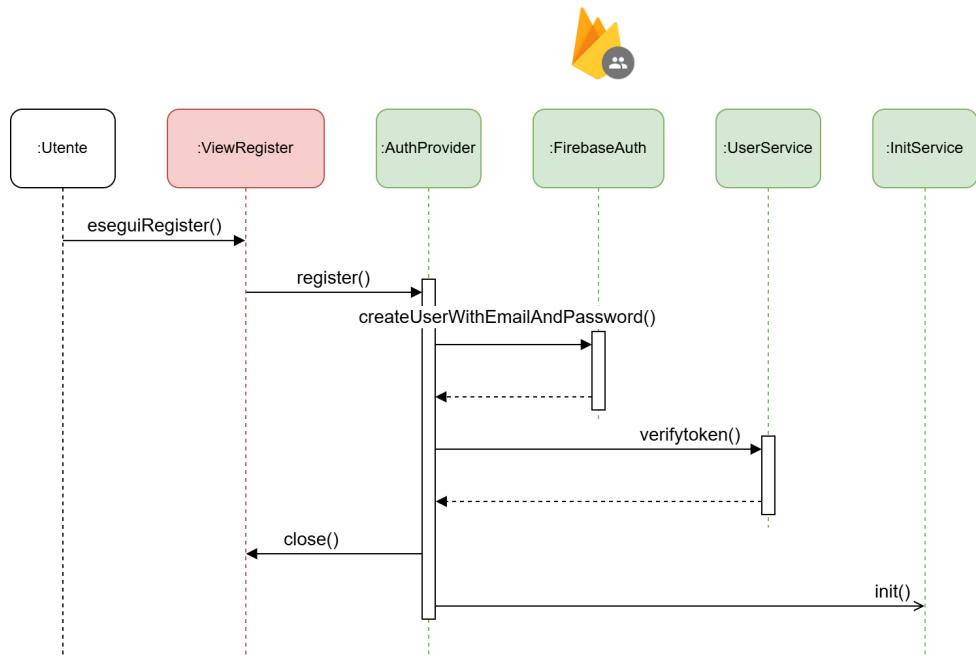


Diagramma di Sequenza: VerifyToken - Client

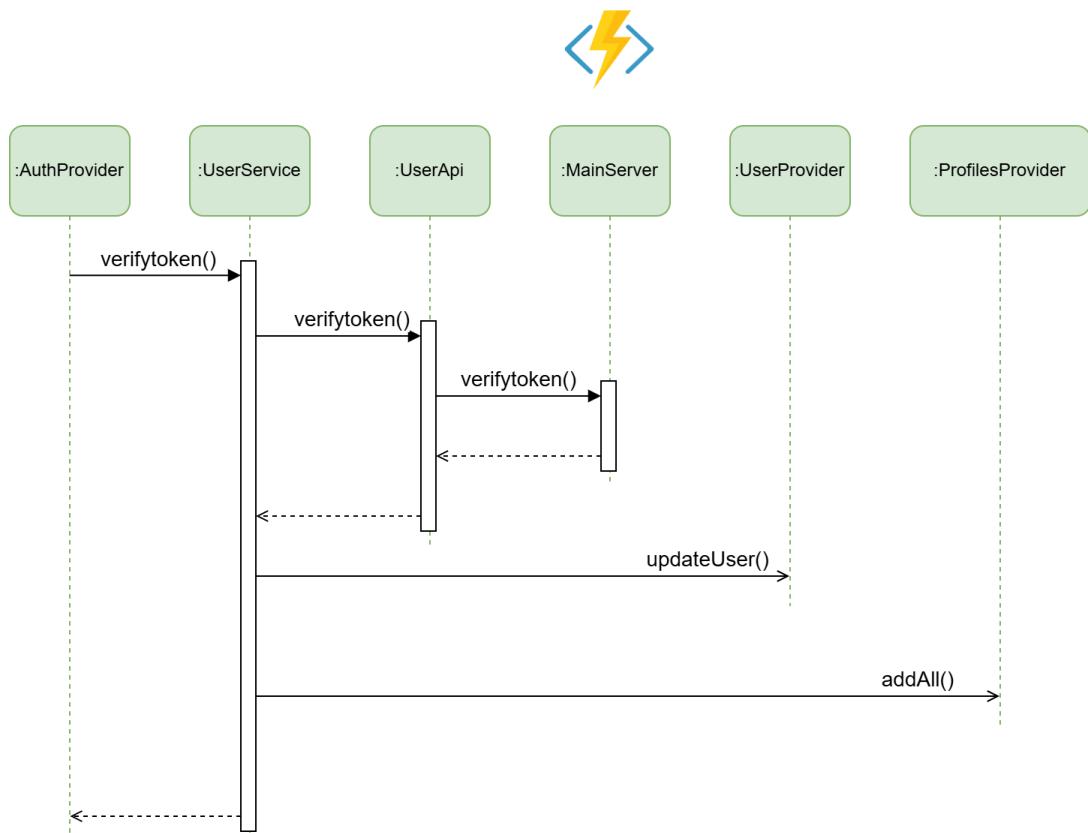
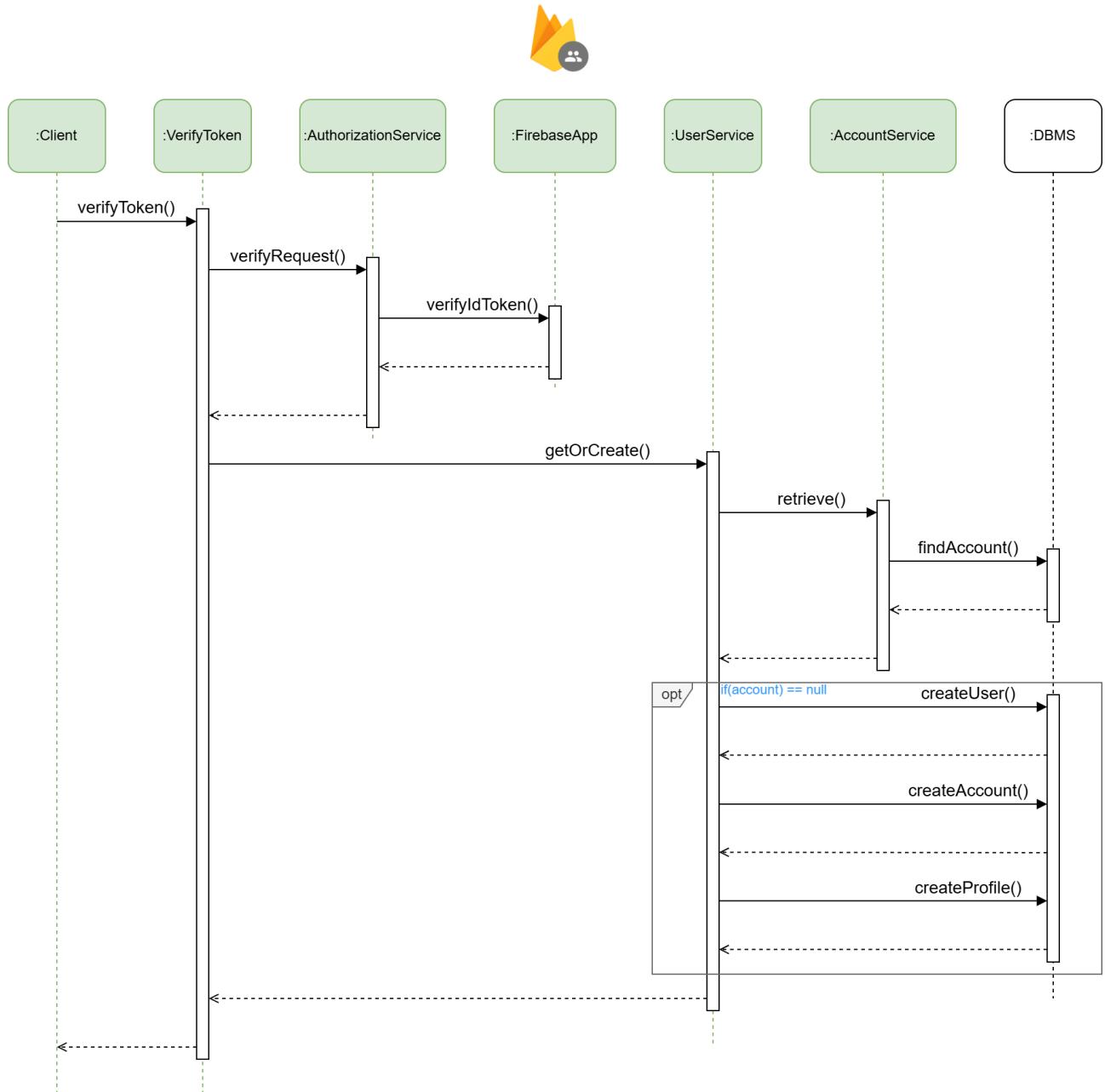


Diagramma di Sequenza: VerifyToken - Server



Come mostrato nel diagramma, la richiesta di verifica token al server Firebase avviene ogni volta che viene invocata la `verifyRequest` (quindi anche nei successivi grafici).

Diagramma di Sequenza: Initialize - Client

Initialize è la sequenza che avviene al primo accesso dopo che l'applicazione è stata chiusa. viene eseguita o dopo il login/registrazione, o all'avvio dell'applicazione con un utente già registrato.

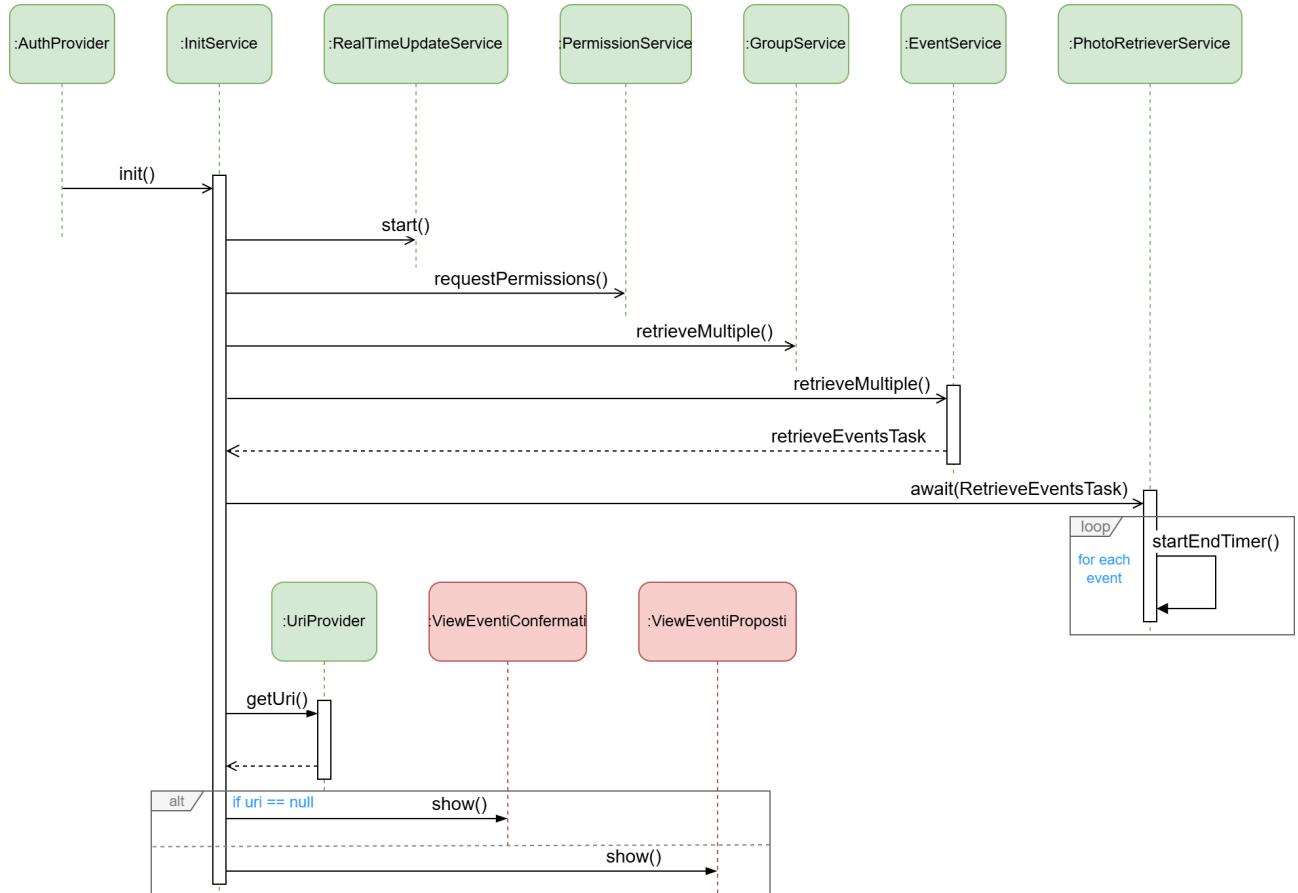
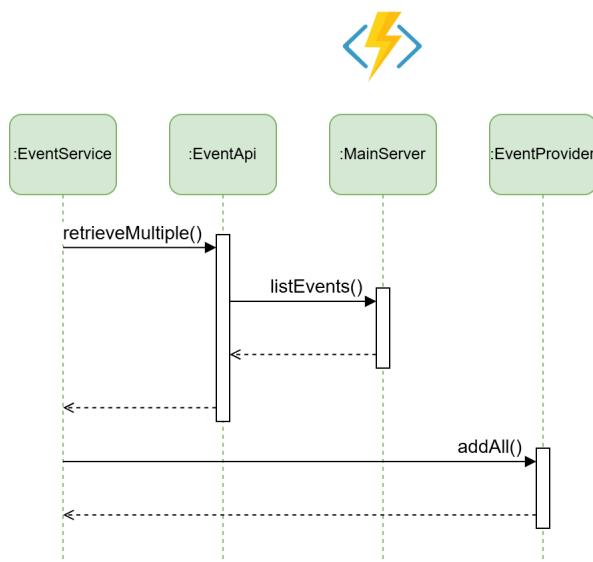


Diagramma di Sequenza: EventService.retrieveMultiple() - Client



RetrieveMultiple di GroupService mantiene la stessa logica, tramite GroupApi e GroupProvider

Diagramma di Sequenza: Visualizza eventi proposti - Client

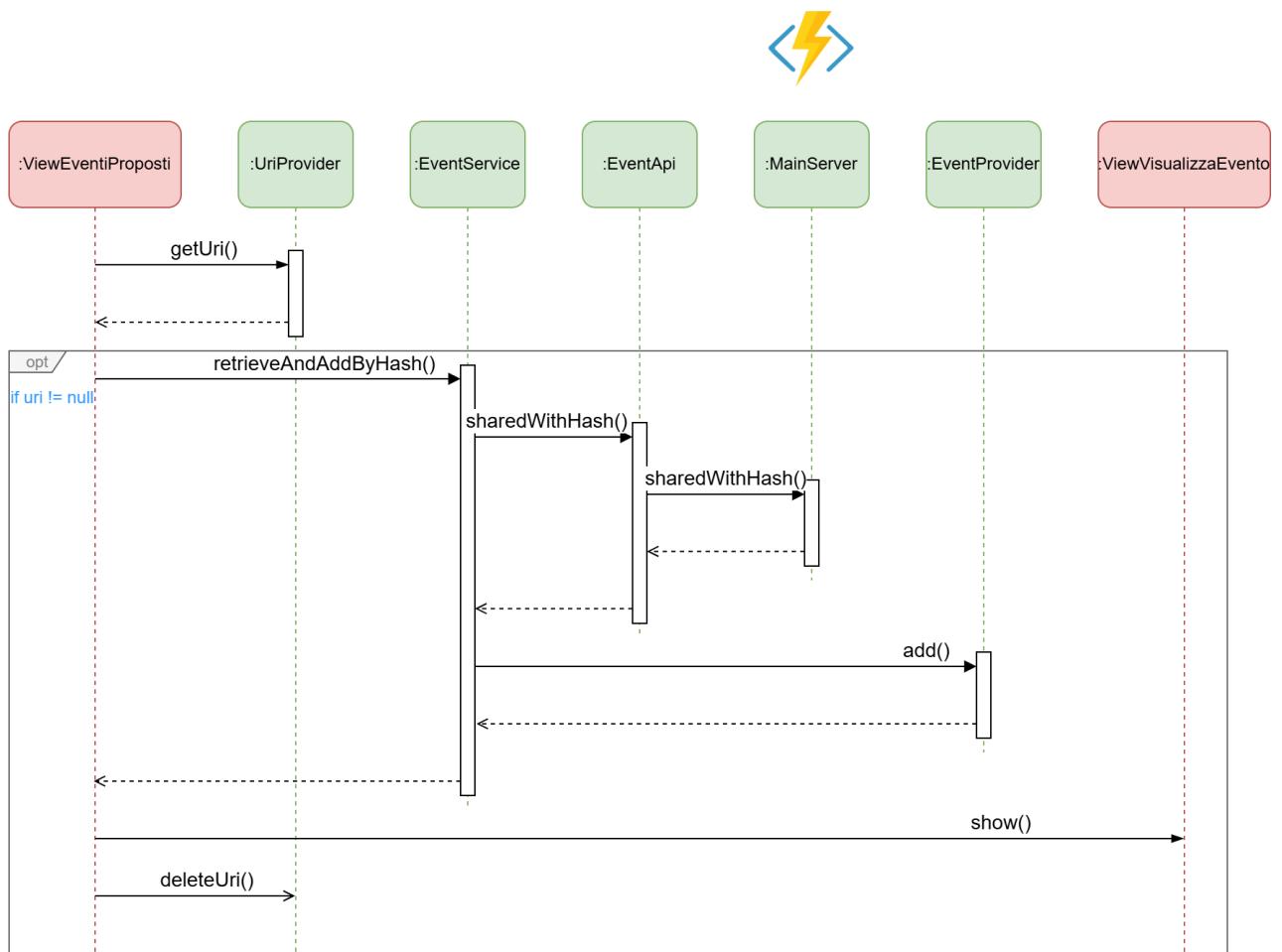


Diagramma di Sequenza: Modifica Evento - Client

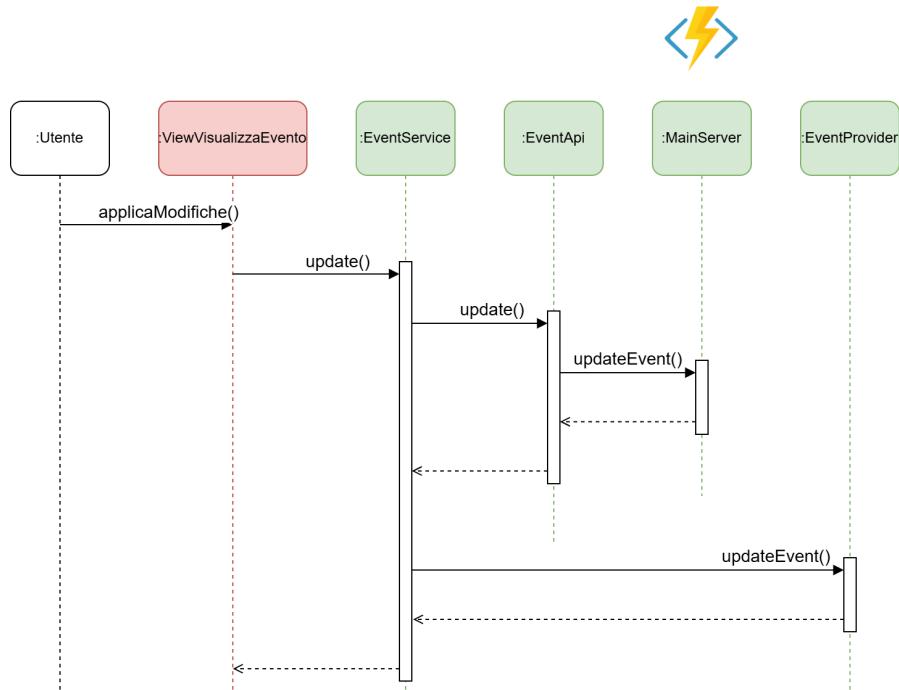


Diagramma di Sequenza: Modifica Evento - Server

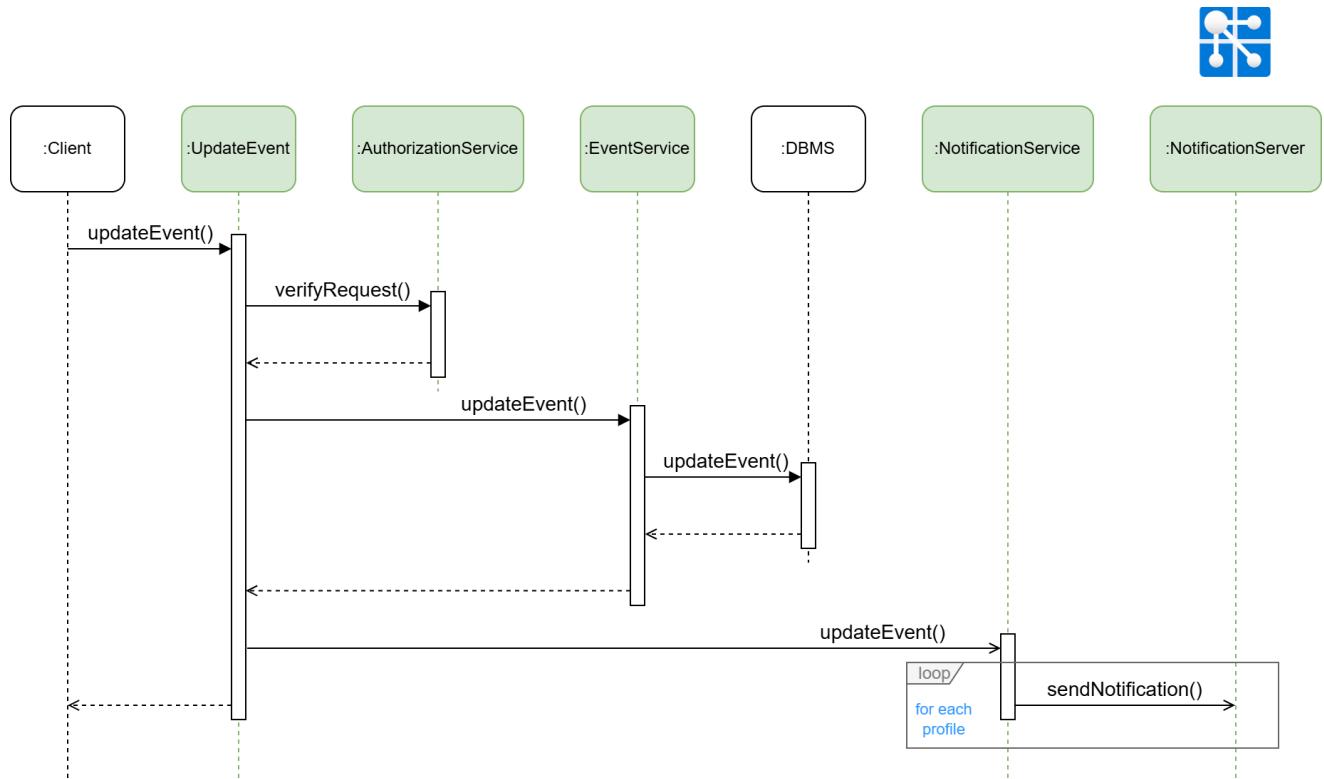


Diagramma di Sequenza: Aggiorna Evento - Client

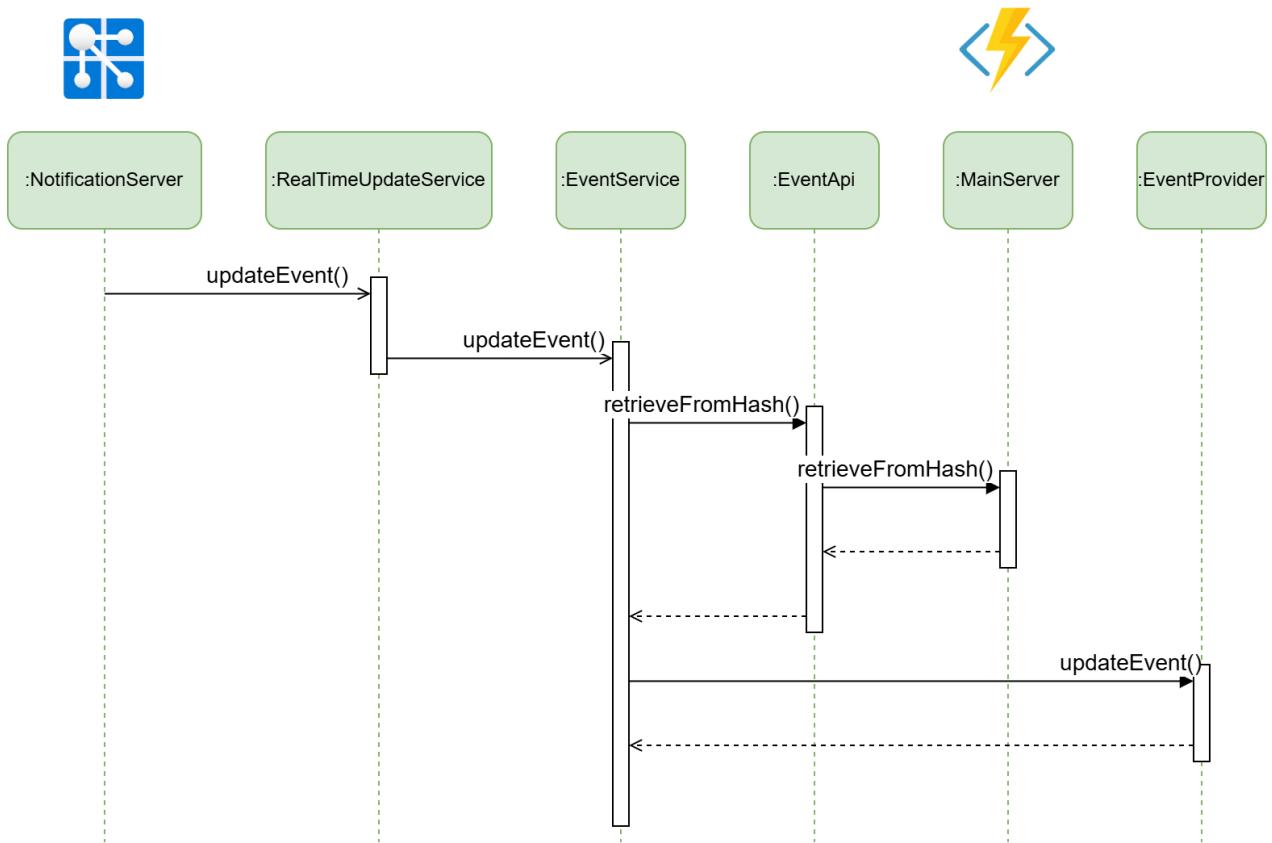


Diagramma di Sequenza: Recupera Immagini - Client

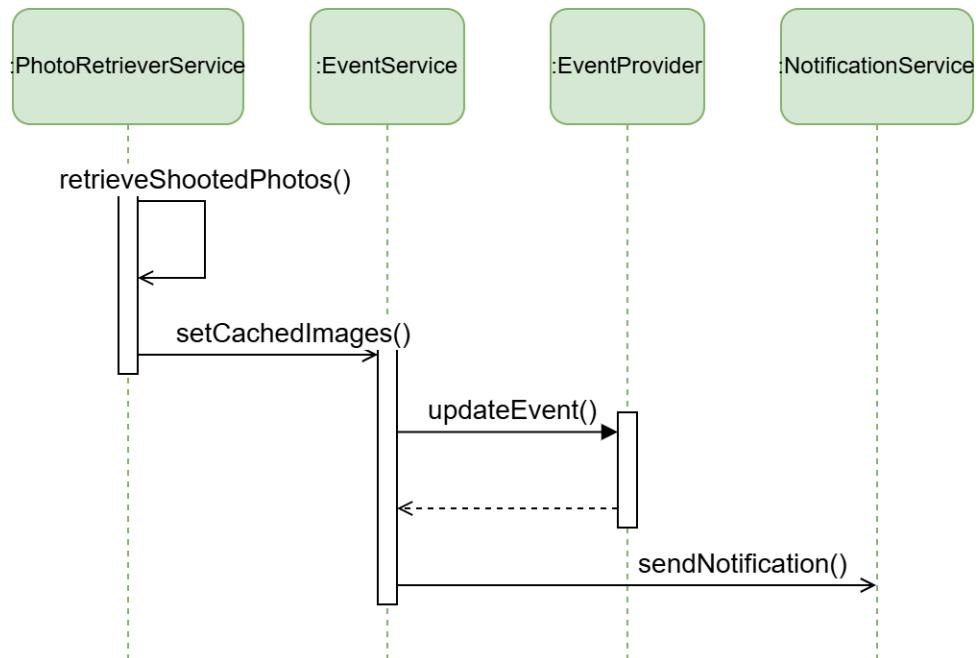


Diagramma di Sequenza: CaricaImmagini - Server

Il salvataggio dell'immagine richiede la creazione di un oggetto con un hash univoco da poi collegare all'evento. La creazione logica dell'oggetto e il suo effettivo salvataggio sul database avvengono in due momenti diversi. Questo permette il salvataggio in contemporanea di più immagini verso il container, la resilienza nel caso di un errore durante il caricamento di una singola immagine e l'ottimizzazione dell'utilizzo del database, concentrando le scritture in un unico momento. Questo riduce inoltre la probabilità di collisione di richieste di modifica sullo stesso evento.

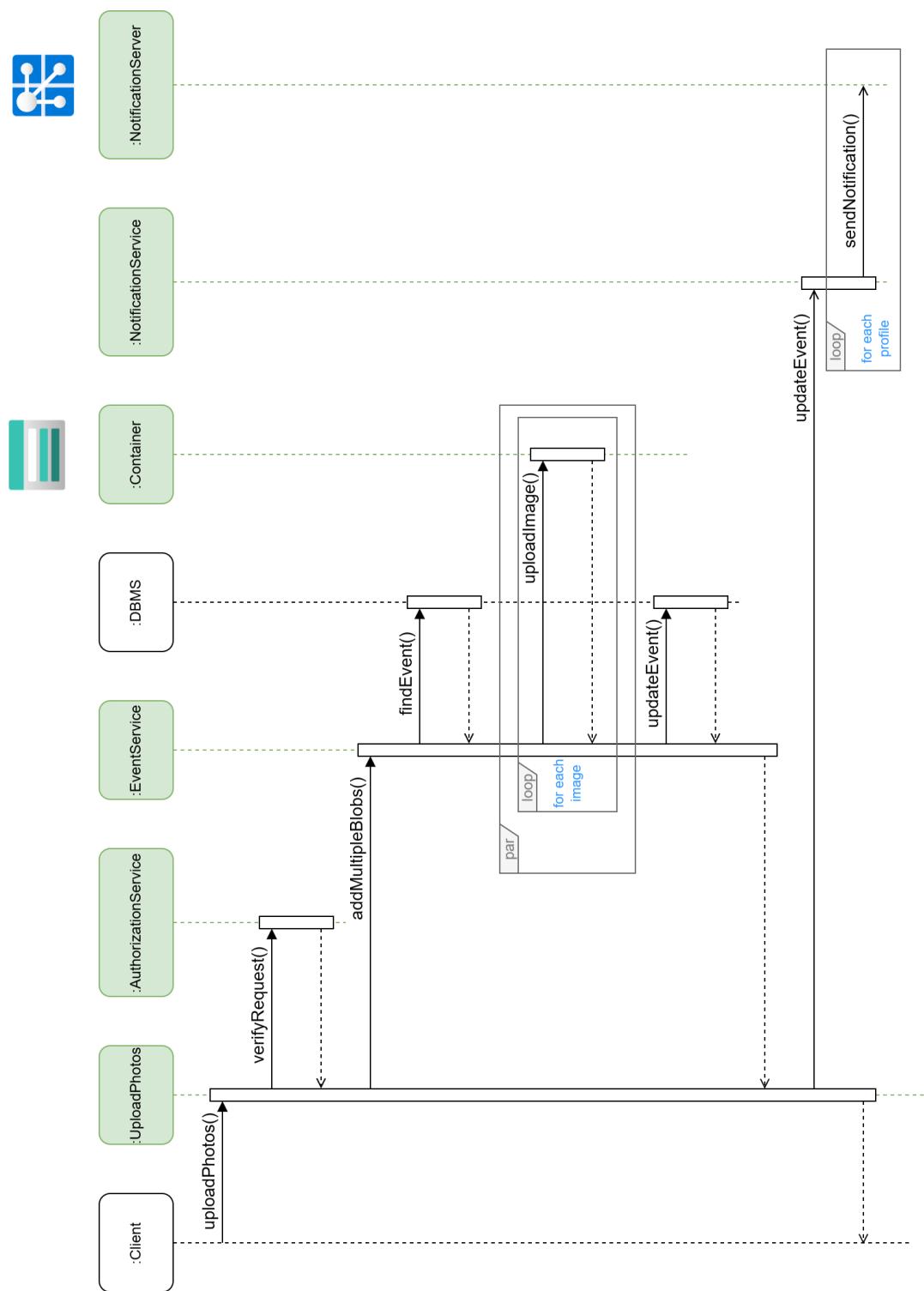


Diagramma di Sequenza: Condivisione ai Gruppi - Client

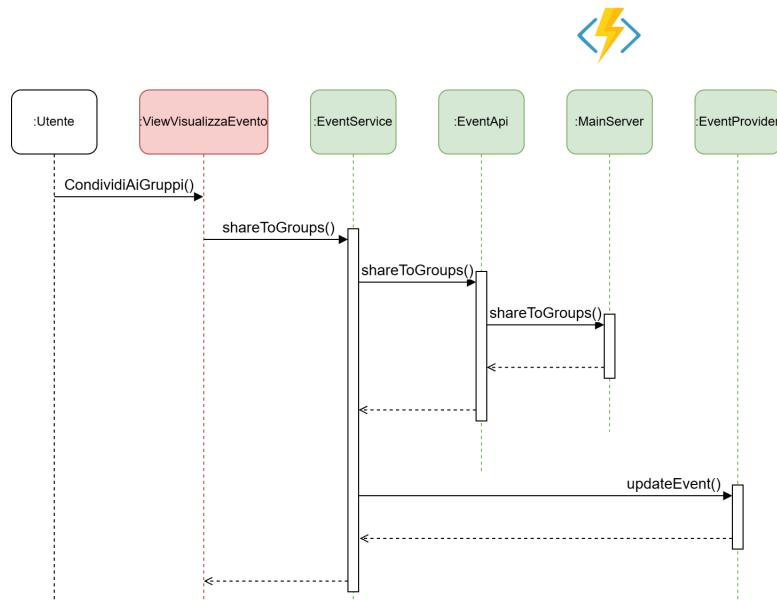
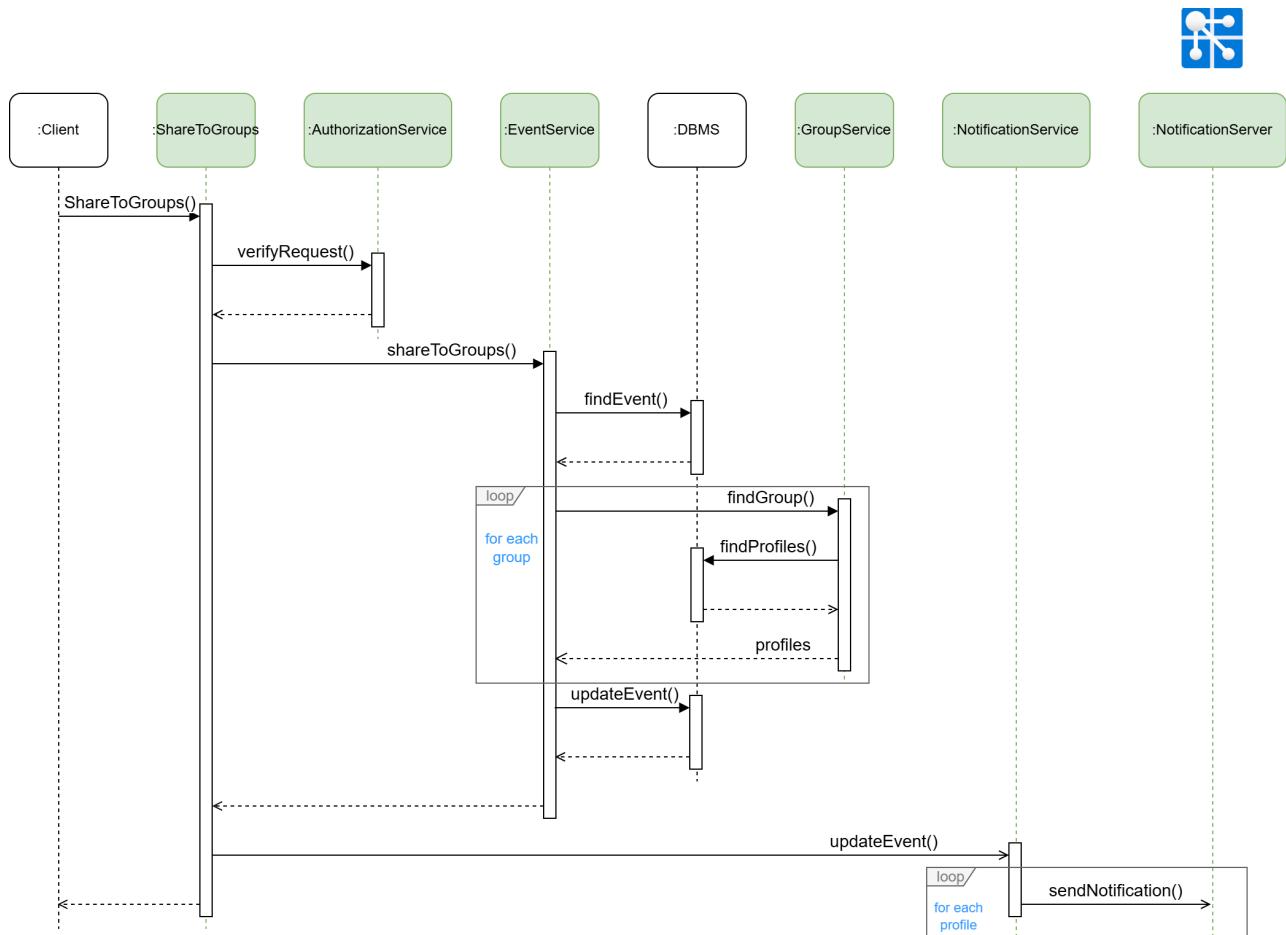


Diagramma di Sequenza: Condivisione ai Gruppi - Server

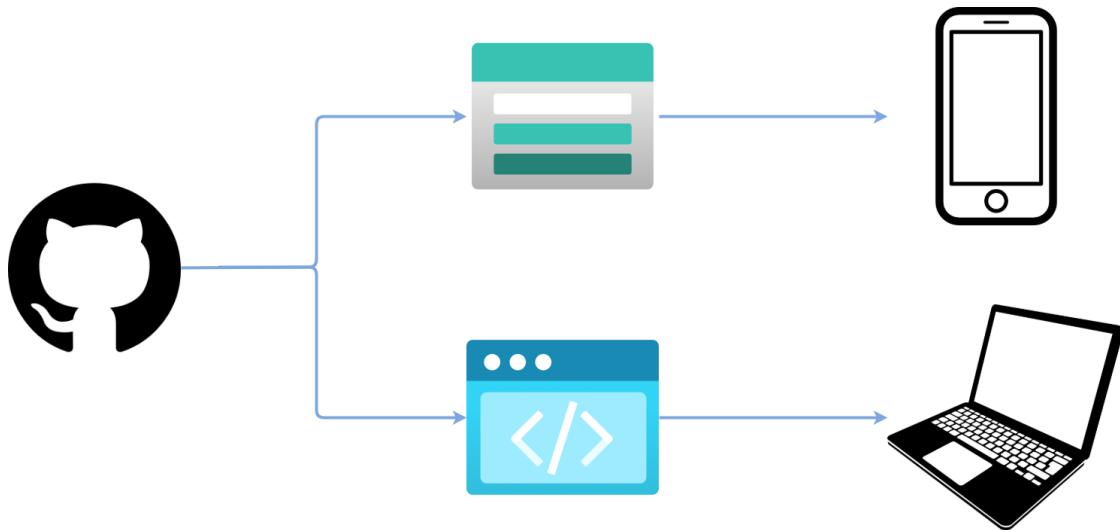


6.6 Deployment e Aggiornamento del Codice

6.6.1 Deployment Type-Level: Client

Il codice distribuito sulla web app e l'applicativo salvato sul container verranno aggiornati automaticamente tramite GitHub Actions.

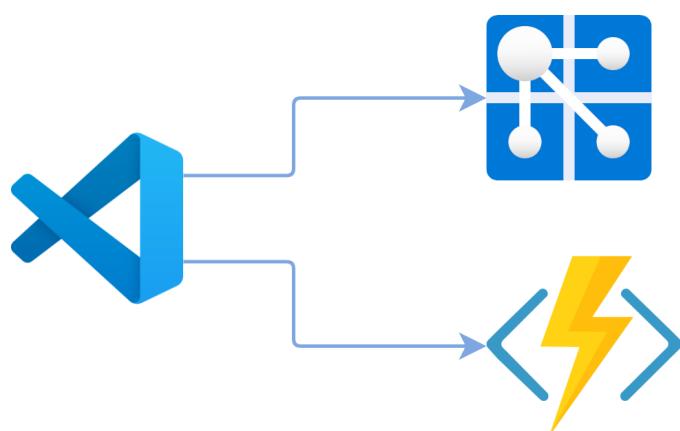
Lato browser il codice si aggiorna ad ogni accesso, mentre l'applicativo necessita di venire reinstallato. Per questa ragione, gli utenti che usano l'applicazione verranno notificati nel momento del caricamento di una nuova versione.



6.6.2 Deployment Type-Level: Server

L'aggiornamento del codice eseguito sulle Azure Functions avverrà con un comando tramite interfaccia grafica su Visual Studio Code grazie all'estensione Azure Tools.

Le modifiche al dominio del database verranno applicate tramite migrations, generate automaticamente dal framework C# grazie alla libreria EFCore che permette di mappare le classi in tabelle SQL.



6.7 Testing

Si implementano i test principali per il corretto funzionamento in locale delle funzionalità del server.

Tutti i test condividono una funzione per simulare il database:

```
5 references
private static WydDbContext GetInMemoryDbContext()
{
    var options = new DbContextOptionsBuilder<WydDbContext>()
        .UseInMemoryDatabase(databaseName: Guid.NewGuid().ToString())
        .Options;

    return new WydDbContext(options);
}
```

Figura 11: Simulazione del Database

6.7.1 AccountService

```
[Fact]
0 references
public void Create_ValidAccount_ReturnsAccount()
{
    var dbContext = GetInMemoryDbContext();
    var service = new AccountService(dbContext);

    var newUser = new User { };
    dbContext.Users.Add(newUser);
    dbContext.SaveChanges();

    var account = new Account
    {
        Mail = "test@example.com",
        Uid = "uid123",
        User = newUser
    };

    var result = service.Create(account);

    Assert.NotNull(result);
    Assert.Equal("test@example.com", result.Mail);
    Assert.Equal("uid123", result.Uid);
}
```

Figura 12: Create: happy Path

```

[Fact]
0 references
public void Create_AccountWithEmptyMail_ThrowsArgumentException()
{
    var dbContext = GetInMemoryDbContext();
    var service = new AccountService(dbContext);

    var newUser = new User { };
    dbContext.Users.Add(newUser);
    dbContext.SaveChanges();

    var account = new Account
    {
        Mail = string.Empty,
        Uid = "uid1234",
        User = newUser
    };

    var exception = Assert.Throws<ArgumentException>(() => service.Create(account));
    Assert.Equal("Invalid account data provided.", exception.Message);
}

```

Figura 13: Create: mail nulla

```

[Fact]
0 references
public void Create_AccountWithEmptyUid_ThrowsArgumentException()
{
    var dbContext = GetInMemoryDbContext();
    var service = new AccountService(dbContext);

    var newUser = new User { };
    dbContext.Users.Add(newUser);
    dbContext.SaveChanges();

    var account = new Account
    {
        Mail = "test@example.com",
        Uid = string.Empty,
        User = newUser
    };

    var exception = Assert.Throws<ArgumentException>(() => service.Create(account));
    Assert.Equal("Invalid account data provided.", exception.Message);
}

```

Figura 14: Create: uid nullo

```
[Fact]
0 references
public void Retrieve_ReturnsAccount()
{
    var dbContext = GetInMemoryDbContext();

    var newUser = new User { };
    dbContext.Users.Add(newUser);
    var newAccount = new Account { Uid = "1234", Mail = "mail1@mail.com", User = newUser };
    dbContext.Accounts.Add(newAccount);
    dbContext.SaveChanges();

    var service = new AccountService(dbContext);

    var result = service.Retrieve("1234");

    Assert.NotNull(result);
    Assert.Equal("1234", result!.Uid);
}
```

Figura 15: Retrive: happy path

6.7.2 ProfileService

```
[Fact]
0 references
public void Create_ShouldAddProfileToDatabase()
{
    var context = GetInMemoryDbContext();
    var service = new ProfileService(context);
    var profile = new Profile { Name = "profileName", Tag = "profileTag" };

    var result = service.Create(profile);

    Assert.NotNull(result);
    Assert.Equal(profile.Name, result.Name);
    Assert.Equal(profile.Tag, result.Tag);
}
```

Figura 16: Create: happy path

```
[Fact]
0 references
public void Create_ShouldThrowException_WhenProfileIsNull()
{
    var context = GetInMemoryDbContext();
    var service = new ProfileService(context);

    Assert.Throws<ArgumentNullException>(() => service.Create(null));
}
```

Figura 17: Create: input nullo

```

[Fact]
0 references
public void SetEventRole_ShouldSetRoleForProfileEvent()
{
    var context = GetInMemoryDbContext();
    var service = new ProfileService(context);

    var ev = new Event
    {
        Title = "Test Event",
        StartTime = DateTimeOffset.Now,
        EndTime = DateTimeOffset.Now.AddHours(2)
    };
    context.Events.Add(ev);
    context.SaveChanges();

    var profile = new Profile
    {
        Name = "profileName2",
        Tag = "profileTag2",
        ProfileEvents = []
    };
    context.Profiles.Add(profile);
    context.SaveChanges();

    ev.Profiles.Add(profile);
    context.SaveChanges();

    service.SetEventRole(ev, profile, EventRole.Owner);

    var updatedProfileEvent = ev.ProfileEvents
        .Where(pe => pe.Profile.Hash == profile.Hash)
        .First();
    Assert.NotNull(updatedProfileEvent);
    Assert.Equal(EventRole.Owner, updatedProfileEvent.Role);
}

```

Figura 18: SetEventRole: happy path

```

[Fact]
0 references
public void RetrieveEvents_ShouldReturnEventDtos()
{
    var profile = new Profile
    {
        Name = "profileName2",
        Tag = "profileTag2",
        Events =
        [
            new Event
            {
                Title = "Event 1",
                StartTime = DateTimeOffset.Now,
                EndTime = DateTimeOffset.Now.AddHours(2)
            },
            new Event
            {
                Title = "Event 2",
                StartTime = DateTimeOffset.Now.AddDays(1),
                EndTime = DateTimeOffset.Now.AddDays(1).AddHours(2)
            }
        ]
    };

    var result = ProfileService.RetrieveEvents(profile);

    Assert.NotNull(result);
    Assert.Equal(2, result.Count);
    Assert.Equal("Event 1", result[0].Title);
    Assert.Equal("Event 2", result[1].Title);
}

```

Figura 19: RetrieveEvents: happy path

```
[Fact]
0 references
public void RetrieveEvents_ShouldReturnEmptyList_WhenNoEvents()
{
    var profile = new Profile
    {
        Name = "profileName2",
        Tag = "profileTag2",
        Events = []
    };

    var result = ProfileService.RetrieveEvents(profile);

    Assert.NotNull(result);
    Assert.Empty(result);
}
```

Figura 20: RetrieveEvents: nessun evento

6.7.3 UserService

```
8 references
private static UserService GetUserService()
{
    var dbContext = GetInMemoryDbContext();
    var accountService = new AccountService(dbContext);
    var profileService = new ProfileService(dbContext);
    var authenticationService = new AuthenticationService();
    return new UserService(dbContext, accountService, profileService, authenticationService);
}
```

Figura 21: Creazione di UserService

```
[Fact]
0 references
public void AddProfile_ShouldAddProfileToUser()
{
    var userService = GetUserService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag" };
    var user = new User { };

    var result = userService.AddProfile(user, profile);

    Assert.NotNull(result);
    Assert.Equal(profile, result.MainProfile);
    Assert.Contains(profile, result.Profiles);
}
```

Figura 22: AddProfile: happy path

```
[Fact]
0 references
public void AddProfile_ShouldThrowException_WhenUserIsNull()
{
    var userService = GetUserService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag1" };

    var exception = Assert.Throws<ArgumentNullException>(() =>
        userService.AddProfile(null, profile));
    Assert.Equal("User cannot be null. (Parameter 'user')", exception.Message);
}
```

Figura 23: AddProfile: user nullo

```

[Fact]
0 references
public void AddProfile_ShouldThrowException_WhenProfileIsNull()
{
    var userService = GetUserService();
    var user = new User { };

    var exception = Assert.Throws<ArgumentNullException>(() =>
        userService.AddProfile(user, null));
    Assert.Equal("Profile cannot be null. (Parameter 'profile')", exception.Message);
}

```

Figura 24: AddProfile: profilo nullo

```

[Fact]
0 references
public void Create_ShouldCreateUserWithAccountAndProfile()
{
    var userService = GetUserService();
    var email = "test@example.com";
    var uid = "testUid";

    var result = userService.Create(email, uid);

    Assert.NotNull(result);
    Assert.Contains(result.MainProfile, result.Profiles);

    var account = result.Accounts.FirstOrDefault(a => a.Mail == email && a.Uid == uid);
    Assert.NotNull(account);
    Assert.Equal(result.Id, account.User.Id);
}

```

Figura 25: Create: happy path

```

[Fact]
0 references
public async Task RetrieveEventsAsync_ShouldReturnEventDtos()
{
    var user = new User { };
    var profiles = new List<Profile>{
        new() {
            Name = "profileName1",
            Events =
            [
                new Event
                {
                    Title = "Event 1",
                    StartTime = DateTimeOffset.Now,
                    EndTime = DateTimeOffset.Now.AddHours(2)
                }
            ],
            new() {
                Name = "profileName2",
                Events =
                [
                    new Event
                    {
                        Title = "Event 2",
                        StartTime = DateTimeOffset.Now.AddDays(1),
                        EndTime = DateTimeOffset.Now.AddDays(1).AddHours(2)
                    }
                ]
            }
        };
    var context = GetInMemoryDbContext();
    context.Users.Add(user);
    context.Profiles.AddRange(profiles);
    user.Profiles.AddRange(profiles);
    context.SaveChanges();

    var result = await UserService.RetrieveEventsAsync(user);

    Assert.NotNull(result);
    Assert.Equal(2, result.Count);
    Assert.Equal("Event 1", result[0].Title);
    Assert.Equal("Event 2", result[1].Title);
}

```

Figura 26: RetrieveEvents: happy path

```

[Fact]
0 references
public void SetProfileRole_ShouldSetRoleForUserProfile()
{
    var userService = GetUserService();
    var context = GetInMemoryDbContext();
    var profile = new Profile { Name = "profileName", Tag = "profileTag3" };

    var user = new User
    {

    };
    user.Profiles.Add(profile);
    context.SaveChanges();

    userService.SetProfileRole(user, profile, Role.Admin);

    var userRole = user.UserRoles.Find(ur => ur.Profile == profile);
    Assert.NotNull(userRole);
    Assert.Equal(Role.Admin, userRole.Role);
}

```

Figura 27: SetProfileRole: happy path

```

[Fact]
0 references
public void SetProfileRole_ShouldThrowException_WhenUserIsNull()
{
    var userService = GetUserService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag4" };

    var exception = Assert.Throws<ArgumentNullException>(() =>
    {
        userService.SetProfileRole(null, profile, Role.Admin));
    Assert.Equal("User cannot be null. (Parameter 'user')", exception.Message);
}

```

Figura 28: SetProfileRole: user nullo

```

[Fact]
0 references
public void SetProfileRole_ShouldThrowException_WhenProfileIsNull()
{
    var userService = GetUserService();
    var user = new User { };

    var exception = Assert.Throws<ArgumentNullException>(() =>
        userService.SetProfileRole(user, null, Role.Admin));
    Assert.Equal("Profile cannot be null. (Parameter 'profile')", exception.Message);
}

```

Figura 29: SetProfileRole: profilo nullo

```

[Fact]
0 references
public void SetProfileRole_ShouldThrowKeyNotFoundException_WhenProfileNotFound()
{
    var userService = GetUserService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag" };
    var user = new User { };

    var exception = Assert.Throws<KeyNotFoundException>(() =>
        userService.SetProfileRole(user, profile, Role.Admin));
    Assert.Equal("User does not have the specified profile.", exception.Message);
}

```

Figura 30: SetProfileRole: profilo non trovato

6.7.4 EventService

```
14 references
private static EventService GetEventService()
{
    var dbContext = GetInMemoryDbContext();
    var groupService = new GroupService(dbContext);
    var profileService = new ProfileService(dbContext);
    return new EventService(dbContext, groupService, profileService);
}
```

Figura 31: Creazione di EventService

```
[Fact]
0 references
public void Create_ValidEventAndProfile_CreatesEvent()
{
    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag" };
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var ev = new EventDto { StartTime = startTime, EndTime = endTime };

    var result = eventService.Create(ev, profile);

    Assert.NotNull(result);
    Assert.NotEqual("event1Hash", result.Hash);
    Assert.Equal(result.StartTime, startTime);
    Assert.Equal(result.EndTime, endTime);
    Assert.Contains(profile, result.Profiles);
}
```

Figura 32: Create: happy path

```
[Fact]
0 references
public void Create_NullEventDto_ThrowsArgumentNullException()
{
    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag1" };

    Assert.Throws<ArgumentNullException>(() => eventService.Create(null, profile));
}
```

Figura 33: Create: evento nullo

```

[Fact]
0 references
public void Create_NullProfile_ThrowsArgumentNullException()
{
    var eventService = GetEventService();
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var ev = new EventDto { StartTime = startTime, EndTime = endTime };

    Assert.Throws<ArgumentNullException>(() => eventService.Create(ev, null));
}

```

Figura 34: Create: profilo nullo

```

[Fact]
0 references
public void Create_EventWithConfirmedProfile_ConfirmsEvent()
{
    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag2" };
    var context = GetInMemoryDbContext();
    context.Profiles.Add(profile);
    context.SaveChanges();
    var ev = new EventDto
    {
        StartTime = DateTimeOffset.Now,
        EndTime = DateTimeOffset.Now.AddHours(2),
        ProfileEvents = [new ProfileEventDto { ProfileHash = profile.Hash, Confirmed = true }]
    };

    var result = eventService.Create(ev, profile);

    Assert.NotNull(result);
    Assert.True(result.ProfileEvents
        .Where((pe) => pe.Profile.Hash == profile.Hash).First().Confirmed);
}

```

Figura 35: Create: profilo confermato

```

[Fact]
0 references
public void RetrieveByHash_ValidHash_ReturnsEvent()
{
    var eventService = GetEventService();
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();

    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var result = eventService.RetrieveByHash(newEvent.Hash);

    Assert.NotNull(result);
    Assert.Equal(newEvent.Hash, result.Hash);
}

```

Figura 36: RetrieveByHash: happy path

```

[Fact]
0 references
public void RetrieveByHash_InvalidHash_ThrowsKeyNotFoundException()
{
    var eventService = GetEventService();
    var invalidHash = "invalidHash";

    var exception = Assert.Throws<KeyNotFoundException>(() =>
        eventService.RetrieveByHash(invalidHash));
    Assert.Equal($"Event with ID {invalidHash} not found.", exception.Message);
}

```

Figura 37: RetrieveByHash: hash invalido

```

[Fact]
0 references
public void RetrieveByHash_NullHash_ThrowsArgumentNullException()
{
    var eventService = GetEventService();

    Assert.Throws<ArgumentNullException>(() => eventService.RetrieveByHash(null));
}

```

Figura 38: RetrieveByHash: hash nullo

```

[Fact]
0 references
public void ConfirmOrDecline_ValidEventAndProfile_ConfirmsEvent()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag3" };
    newEvent.Profiles.Add(profile);
    dbContext.SaveChanges();

    var result = eventService.ConfirmOrDecline(newEvent.Hash, profile, true);

    Assert.NotNull(result);
    Assert.True(result.ProfileEvents.First(pe => pe.Profile.Id == profile.Id).Confirmed);
}

```

Figura 39: Confirm: happy path

```

[Fact]
0 references
public void ConfirmOrDecline_InvalidEventHash_ThrowsKeyNotFoundException()
{
    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag4" };

    var exception = Assert.Throws<KeyNotFoundException>(() =>
        eventService.ConfirmOrDecline("invalidHash", profile, true));
    Assert.Equal("Event with ID invalidHash not found.", exception.Message);
}

```

Figura 40: Confirm: hash invalido

```

[Fact]
0 references
public void ConfirmOrDecline_InvalidProfile_ThrowsKeyNotFoundException()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();
    var profile = new Profile { Name = "profileName", Tag = "profileTag5" };
    newEvent.Profiles.Add(profile);
    dbContext.SaveChanges();

    var invalidProfile = new Profile { Hash = "profile71Hash" };

    var exception = Assert.Throws<KeyNotFoundException>(() =>
        eventService.ConfirmOrDecline(newEvent.Hash, invalidProfile, true));
}

```

Figura 41: Confirm: profilo invalido

```

[Fact]
0 references
public void ConfirmOrDecline_ValidEventAndProfile_DeclinesEvent()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();
    var profile = new Profile { Hash = "profile8Hash" };
    newEvent.Profiles.Add(profile);
    dbContext.SaveChanges();

    var result = eventService.ConfirmOrDecline(newEvent.Hash, profile, false);

    Assert.NotNull(result);
    Assert.False(result.ProfileEvents.First(pe => pe.Profile.Hash == profile.Hash).Confirmed);
}

```

Figura 42: Decline: happy path

```

[Fact]
0 references
public void Share_ValidEventAndProfiles_SharesEvent()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();
    var profiles = new HashSet<Profile>
    {
        new () { Name = "profileName", Tag = "profileTag6" },
        new () { Name = "profileName", Tag = "profileTag7" }
    };

    var result = eventService.Share(newEvent, profiles);

    Assert.NotNull(result);
    Assert.Equal(2, result.Profiles.Count);
    Assert.Contains(result.Profiles, p => p.Tag == "profileTag6");
    Assert.Contains(result.Profiles, p => p.Hash == "profileTag7");
}

```

Figura 43: Share: happy path

```

[Fact]
0 references
public void Share_EmptyProfiles_DoesNotThrowException()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();
    var profiles = new HashSet<Profile>();

    var result = eventService.Share(newEvent, profiles);

    Assert.NotNull(result);
    Assert.Empty(result.Profiles);
}

```

Figura 44: Share: lista profili vuota

```
[Fact]
0 references
public void Share_NullProfiles_ThrowsArgumentNullException()
{
    var startTime = DateTimeOffset.Now;
    var endTime = DateTimeOffset.Now.AddHours(2);
    var newEvent = new Event { StartTime = startTime, EndTime = endTime };

    var dbContext = GetInMemoryDbContext();
    dbContext.Events.Add(newEvent);
    dbContext.SaveChanges();

    var eventService = GetEventService();

    Assert.Throws<ArgumentNullException>(() => eventService.Share(newEvent, null));
}
```

Figura 45: Share: lista profili nulla

7 Performances

8 Costi