Dipartimento di Ingegneria e Scienza dell'Informazione

Progetto:

EventsMap

Titolo del documento:

Analisi e Progettazione

Document Info

Doc. Name	*D1EventsMAP_AnalisiProgettazione*	Doc. Number	AP1
Description	Documento di analisi e progettazione		

INDICE

Contents

Scopo del documento

Il presente documento riporta la specifica dei requisiti di sistema del progetto EventMap usando quanto più possibile diagrammi in Unified Modeling Language (UML). Nel precedente documento sono stati definiti gli obiettivi del progetto (PERCHÉ) e i requisiti (COSA) usando solo il linguaggio naturale. Ora i requisiti vengono specificati usando sia il linguaggio naturale sia linguaggi più formali e strutturati, in particolare UML per la descrizione dei requisiti funzionali. Inoltre, tenendo conto di tali requisiti, viene presentato il design del sistema con l'utilizzo del diagramma dei componenti e un diagramma delle classi.

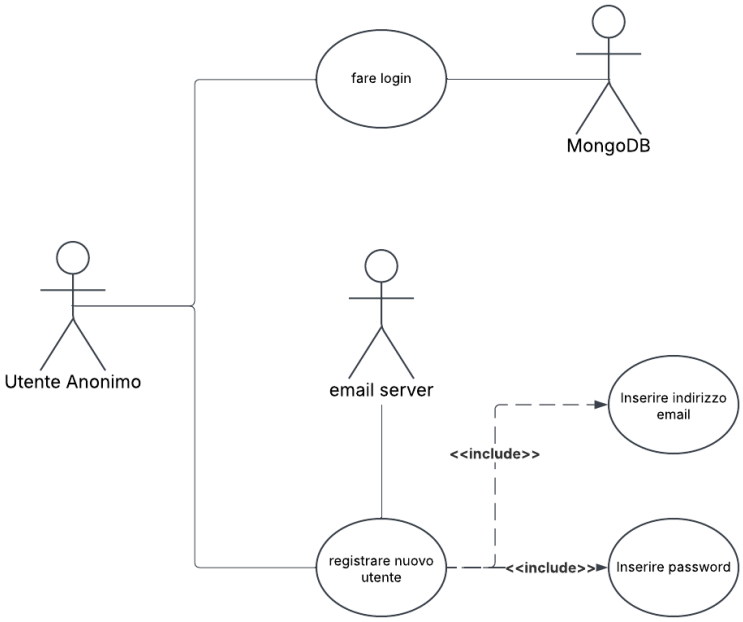
1. Requisiti Funzionali

Nel presente capitolo vengono riportati i requisiti funzionali (RF) del sistema utilizzando il linguaggio naturale e Use Case Diagram (UCD) scritti in UML.

Utente Anonimo

RF1: Effettuare login

RF2: Registrare nuovo utente



Use Case RF1: Effettuare login

Riassunto: Questo use case descrive come l'utente anonimo effettua il login nel sistema

Descrizione:

1. L'utente visualizza una schermata con un form di login
2. L'utente può:
    - Inserire e-mail e password negli appositi campi e premere "Login"
3. Se le credenziali sono corrette, l'utente viene reindirizzato alla mappa degli eventi

Eccezioni: Se le credenziali non sono valide, viene mostrato un messaggio di errore

RF2: Registrare nuovo utente

Riassunto: Questo use case descrive come l'utente anonimo effettua la registrazione nel sistema

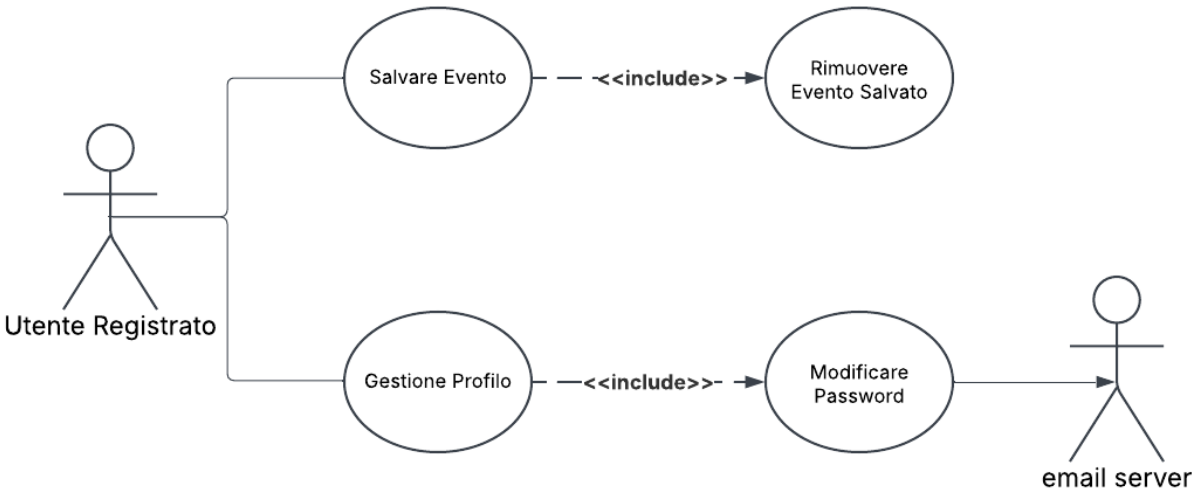
Descrizione: L'utente visualizza una schermata nella quale inserire e-mail e password per i futuri accessi

Eccezioni: Se l'utente non inserisce questi campi obbligatori la registrazione non prosegue

Utente Autenticato

RF3: Gestire eventi preferiti

RF4: Gestire profilo



Use Case RF3: Gestire eventi preferiti

Riassunto: Questo use case descrive come l'utente autenticato può gestire i propri eventi preferiti

Descrizione:

1. L'utente visualizza un evento sulla mappa o nella lista
2. L'utente può:
    - Salvare l'evento tra i preferiti
    - Rimuovere l'evento dai preferiti

- Visualizzare la lista degli eventi salvati
- Il sistema aggiorna lo stato dei preferiti nel database

Use Case RF3: Gestire profilo

Riassunto: Questo use case descrive come l'utente autenticato può gestire il proprio account

Descrizione:

1. L'utente visualizza la schermata del proprio account
2. L'utente può:
   - Cambiare password immettendo quella precedente e immettendo quella nuova
3. Il sistema aggiorna la password nel database con le dovute sicurezze

2. Analisi dei Componenti

2.1 Definizione dei Componenti

In questa sezione analizziamo in dettaglio ogni componente del sistema, specificando le sue responsabilità, interfacce e interazioni con gli altri componenti.

CMP1: Gestione Autenticazione

Descrizione:

Il componente gestisce tutti gli aspetti legati all'autenticazione degli utenti nel sistema. Si occupa sia del processo di registrazione di nuovi utenti che dell'accesso degli utenti esistenti. Implementa la modalità di autenticazione tramite credenziali locali (e-mail/password).

Interfacce richieste

- Credenziali di accesso → `{ email: string, password: string }`
- Verifica autenticazione → `{ token: string }`

Interfacce fornite

- Autenticazione Utente → `{ token: string, expires: string }`
- Validazione token → `{ userId: string, valid: boolean }`
- Gestione password
    - Cambio password → `{ currentPassword: string, newPassword: string }`
    - Validazione e hashing → `{ success: boolean, message: string }`

CMP2: Gestione Registrazione

Descrizione:

Il componente si occupa di richiedere all'utente i propri dati per effettuare una nuova registrazione. L'utente fornirà i propri dati personali e la propria e-mail, i quali verranno salvati all'interno del database. Il sistema creerà una password temporanea che verrà comunicata al componente "Gestione email" così da poterla inviare all'utente.

Interfacce richieste

- Dati personali utente → `{ email: string, password: string }`
- Verifica duplicati nel database → `{ email: string }`

Interfacce fornite

- Registrazione utente → `{ userId: string, token: string }`
- Creazione account → `{ success: boolean, message: string }`

CMP3: Gestione Eventi

Descrizione:

Il componente Gestione Eventi rappresenta il cuore del sistema TrentoEvents. È responsabile dell'acquisizione, normalizzazione e gestione di tutti gli eventi provenienti dall'API del Comune di Trento. Questo componente implementa la logica necessaria per sincronizzare periodicamente i dati con l'API comunale, gestire il salvataggio locale degli eventi nel database MongoDB, e fornire funzionalità di ricerca e filtro avanzate.

Interfacce richieste

- Dati dall'API di Trento → `{ apiId: string, title: string, date: Date, location: string, category: string }`
- Richieste database → `{ filters: object }`

Interfacce fornite

- Lista eventi disponibili → `{ events: [] }`
- Gestione eventi salvati → `{ eventId: string, userId: string }`
- Dettagli evento → `{ eventId: string }`

CMP4: Gestione Mappa

Descrizione:

Gestisce la visualizzazione geografica degli eventi utilizzando OpenStreetMap. Si occupa del rendering della mappa, posizionamento dei marker degli eventi e gestione delle interazioni utente con la mappa.

Interfacce richieste

- Dati evento → `{ coordinates: { lat: number, lng: number }, title: string }`
- Interazione utente → `{ eventId: string }`

Interfacce fornite

- Visualizzazione mappa con marker → `{ lat: number, lng: number }`
- Interazione utente con marker → `{ onMarkerClick(eventId) }`

CMP5: Gestione Preferiti

Descrizione:

Gestisce il salvataggio e la rimozione degli eventi preferiti per gli utenti autenticati. Mantiene la sincronizzazione tra il frontend e il database per garantire una user experience fluida.

Interfacce richieste

- Identificazione evento e utente → `{ eventId: string, userId: string }`
- Token JWT per autenticazione → `{ token: string }`

Interfacce fornite

- Lista eventi salvati → `{ savedEvents: [] }`
- Stato sincronizzazione → `{ lastSync: Date }`

CMP6: Gestione Database

Descrizione:

Il componente Gestione Database costituisce il nucleo centrale per la persistenza dei dati nel sistema TrentoEvents. Questo componente agisce come un intermediario tra il database MongoDB fisico e tutti gli altri componenti del sistema, fornendo un'interfaccia unificata e coerente per tutte le operazioni sui dati. La sua responsabilità principale è garantire l'integrità, la consistenza e l'efficienza nell'accesso ai dati, implementando anche strategie di caching per ottimizzare le performance del sistema.

Interfacce richieste

- Dati autenticazione → `{ email: string, passwordHash: string }`
- Dati eventi → `{ apiId: string, title: string }`
- Dati utente → `{ userId: string }`

Interfacce fornite

- Gestione utenti → `{ createUser, findUserByEmail, updateUserPassword }`
- Gestione eventi → `{ saveEvent, updateEvent, deleteEvent }`
- Gestione preferiti → `{ addFavorite, removeFavorite, getUserFavorites }`

CMP7: Gestione Impostazioni

Descrizione:

Il componente si occupa di gestire la modifica della password dell'utente. Permette all'utente autenticato di cambiare la propria password in modo sicuro, verificando la password corrente prima di procedere con l'aggiornamento. Se l'utente effettua il primo login dopo la registrazione, viene obbligato a cambiare la password temporanea.

Interfacce richieste

- Verifica autenticazione → `{ token: string }`
- Dati modifica password → `{ currentPassword: string, newPassword: string }`

Interfacce fornite

- Modifica password → `{ success: boolean, message: string }`
- Validazione password → `{ valid: boolean }`

CMP8: Gestione Filtri e Ricerca

Descrizione:

Il componente si occupa di gestire tutte le operazioni di filtro e ricerca degli eventi nel sistema. Questo modulo è fondamentale per permettere agli utenti di trovare facilmente gli eventi di loro interesse attraverso vari criteri come data, categoria e circoscrizione. Il componente elabora le richieste di ricerca, le traduce in query per il database e restituisce i risultati in formato appropriato.

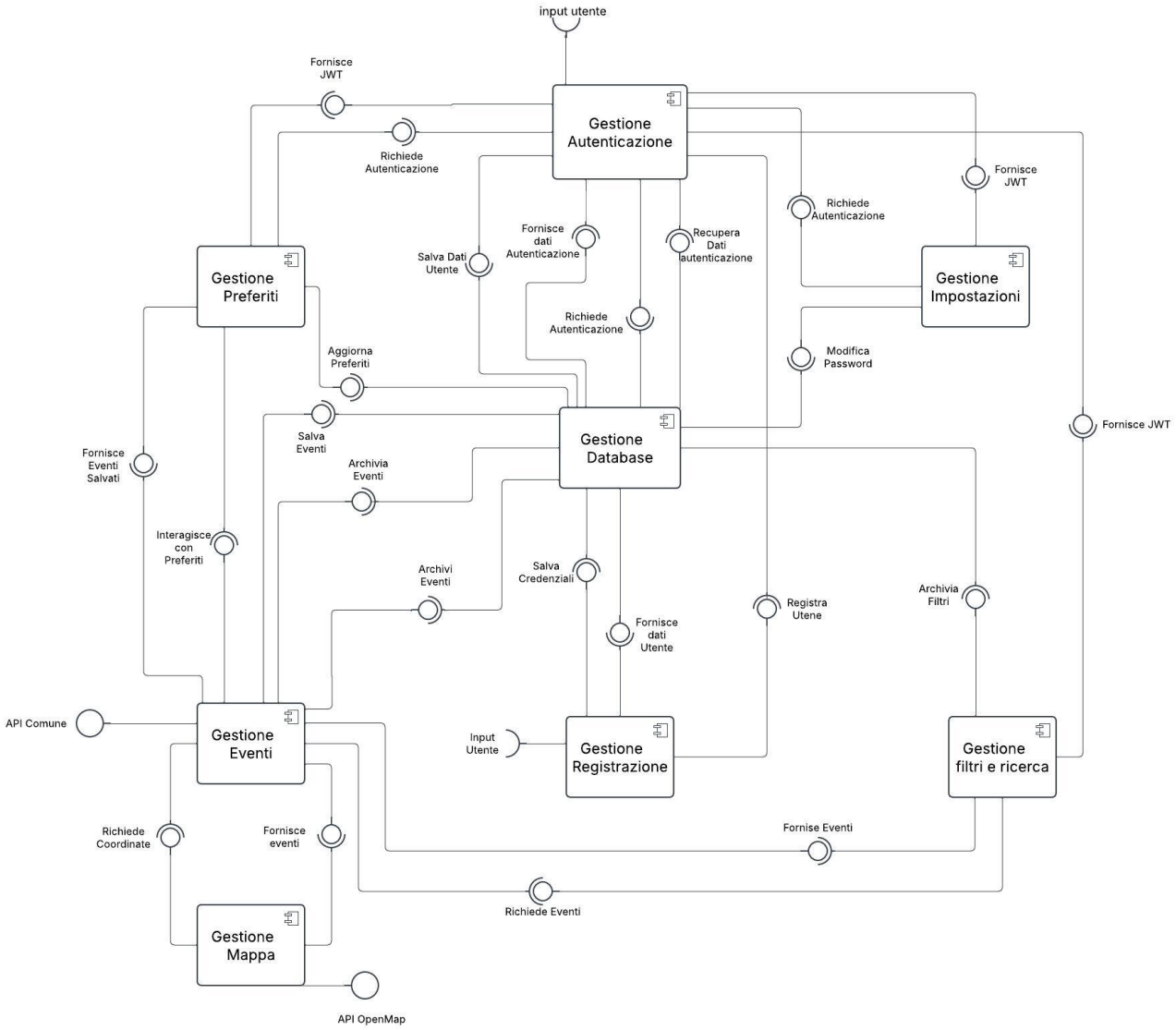
Interfacce richieste

- Parametri di ricerca → `{ date: Date, category: string }`

Interfacce fornite

- Ricerca eventi filtrati → `{ events: [] }`
- Suggerimenti di ricerca → `{ categories: [], dates: [] }`

1.2 Diagramma dei componenti

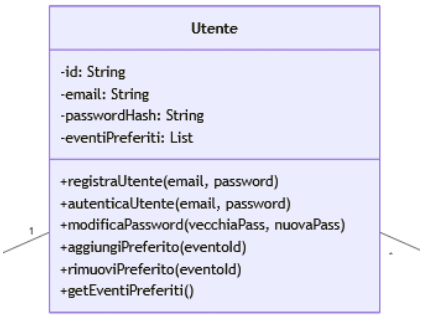


3. Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto TrentoEvents. Ogni componente presente nel diagramma dei componenti diventa una o più classi.

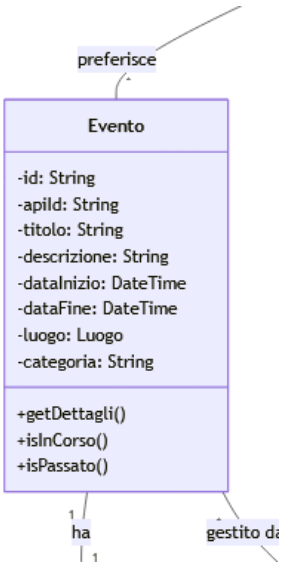
1. Utente

La classe Utente rappresenta l'attore principale che utilizza il sistema dopo essersi registrato. Può visualizzare gli eventi sulla mappa e, una volta autenticato, salvare i propri eventi preferiti. All'interno della classe Utente sono state inserite le operazioni di gestione delle credenziali e dei preferiti. L'utente può modificare la propria password e gestire la lista degli eventi salvati.



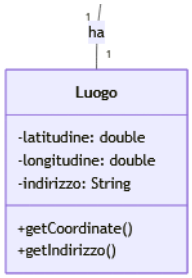
2. Evento

La classe Evento gestisce tutte le informazioni relative agli eventi disponibili nel sistema. Ogni evento viene recuperato dall'API del Comune di Trento e salvato localmente con le informazioni necessarie per la visualizzazione e gestione. L'evento contiene sia un ID interno che l'ID originale dell'API. Include tutti i dettagli necessari per la visualizzazione e mantiene un riferimento alla sua localizzazione attraverso la classe Luogo.



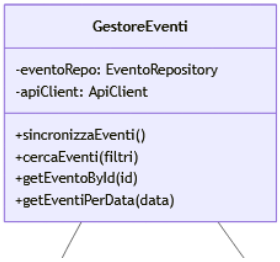
3. Luogo

La classe Luogo si occupa di gestire le informazioni geografiche degli eventi. È fondamentale per la visualizzazione degli eventi sulla mappa e per il filtraggio per circoscrizione. Questa classe mantiene sia le coordinate geografiche per il posizionamento sulla mappa che l'indirizzo testuale e la circoscrizione per la ricerca e il filtraggio.



4. GestoreEventi

Il GestoreEventi è responsabile del ciclo di vita degli eventi nel sistema, dalla sincronizzazione con l'API comunale alla ricerca e filtraggio. Questa classe coordina le operazioni tra l'API client e il repository degli eventi, gestendo la sincronizzazione e fornendo metodi di ricerca avanzati.

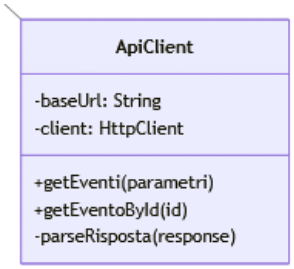


5. GestorePreferiti

Il GestorePreferiti si occupa di tutte le operazioni relative al salvataggio e alla gestione degli eventi preferiti degli utenti. Coordina le operazioni tra il repository degli utenti e degli eventi per gestire i preferiti, mantenendo la coerenza dei dati.

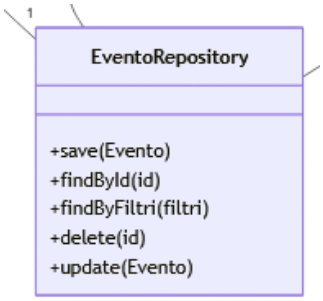
6. APIClient

La classe ApiClient gestisce tutte le comunicazioni con l'API del Comune di Trento. Si occupa di effettuare le richieste HTTP, gestire le risposte e convertire i dati nel formato interno del sistema.



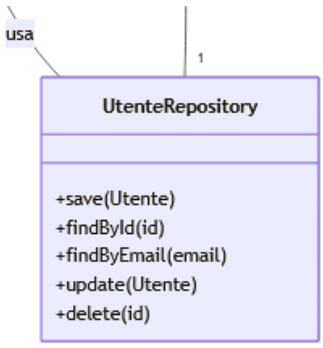
7. EventoRepositiry

L'EventoRepository implementa il pattern Repository per la gestione della persistenza degli eventi nel database. Fornisce metodi CRUD per gli eventi e implementa query specializzate per la ricerca.



8. UtenteRepository

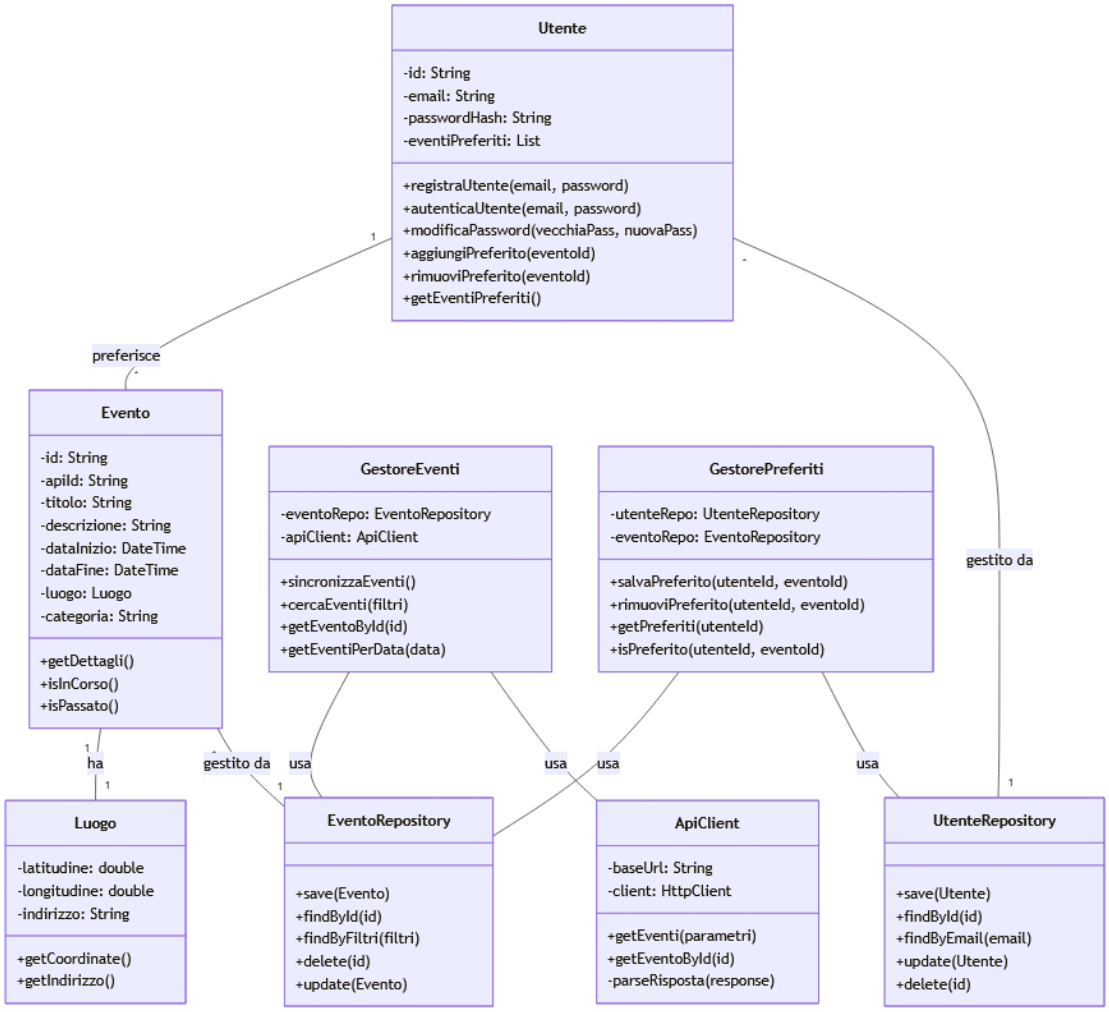
L'UtenteRepository gestisce la persistenza dei dati degli utenti nel database. Fornisce metodi CRUD per gli utenti e query specializzate come la ricerca per e-mail.



3.1 Diagramma delle classi complessivo

Relazioni tra le classi:

1. Utente - Evento (1:N)
   - Un utente può salvare molti eventi nei preferiti
   - La relazione è implementata attraverso una lista di ID nella classe Utente
2. Evento - Luogo (1:1)
   - Ogni evento ha esattamente una localizzazione
   - Il luogo è un attributo dell'evento
3. GestoreEventi - Repositories
   - Utilizza sia EventoRepository che ApiClient per la gestione completa degli eventi
   - Coordina le operazioni tra API e database
4. GestorePreferiti - Repositories
   - Utilizza entrambi i repository per gestire i preferiti
   - Mantiene la coerenza tra utenti ed eventi



4. Dal Class diagram alle API

Classe	Metodo	HTTP Method	URI + query params	Request	Response	Note
Utente	effettuaLogin	POST	/api/v1/auth/login	{email, password }	200 User {token }	È possibile fornire e-mail e password per ottenere un token di accesso
Utente	registraNuovo	POST	/api/v1/auth/register	{email, password }	200 User { id, email }	-
Utente	modificaPassword	PUT	/api/v1/user/{id}	{ password }	200 User	-
Utente	getPreferiti	GET	/api/v1/user/{id}/favorites	-	200 Event[]	Lista degli eventi salvati
Utente	salvaPreferito	POST	/api/v1/user/{id}/favorites/{eventId}	-	200 {success: true }	-
Utente	rimuoviPreferito	DELETE	/api/v1/user/{id}/favorites/{eventId}	-	204	-
Evento	getDettagli	GET	/api/v1/events/{id}	-	200 Event	Dettagli completi evento
Evento	cercaPerData	GET	/api/v1/events?date={date}	-	200 Event[]	Filtra eventi per data
Evento	cercaPerCategoria	GET	/api/v1/events?category={cat}	-	200 Event[]	Filtra per categoria
GestoreEventi	sincronizza	POST	/api/v1/events/sync	-	200 {updated: number }	Solo admin - Sincronizza con API comunale
GestoreEventi	rimuoviPassati	DELETE	/api/v1/events/cleanup	-	200 {deleted: number }	Solo admin - Rimuove eventi passati