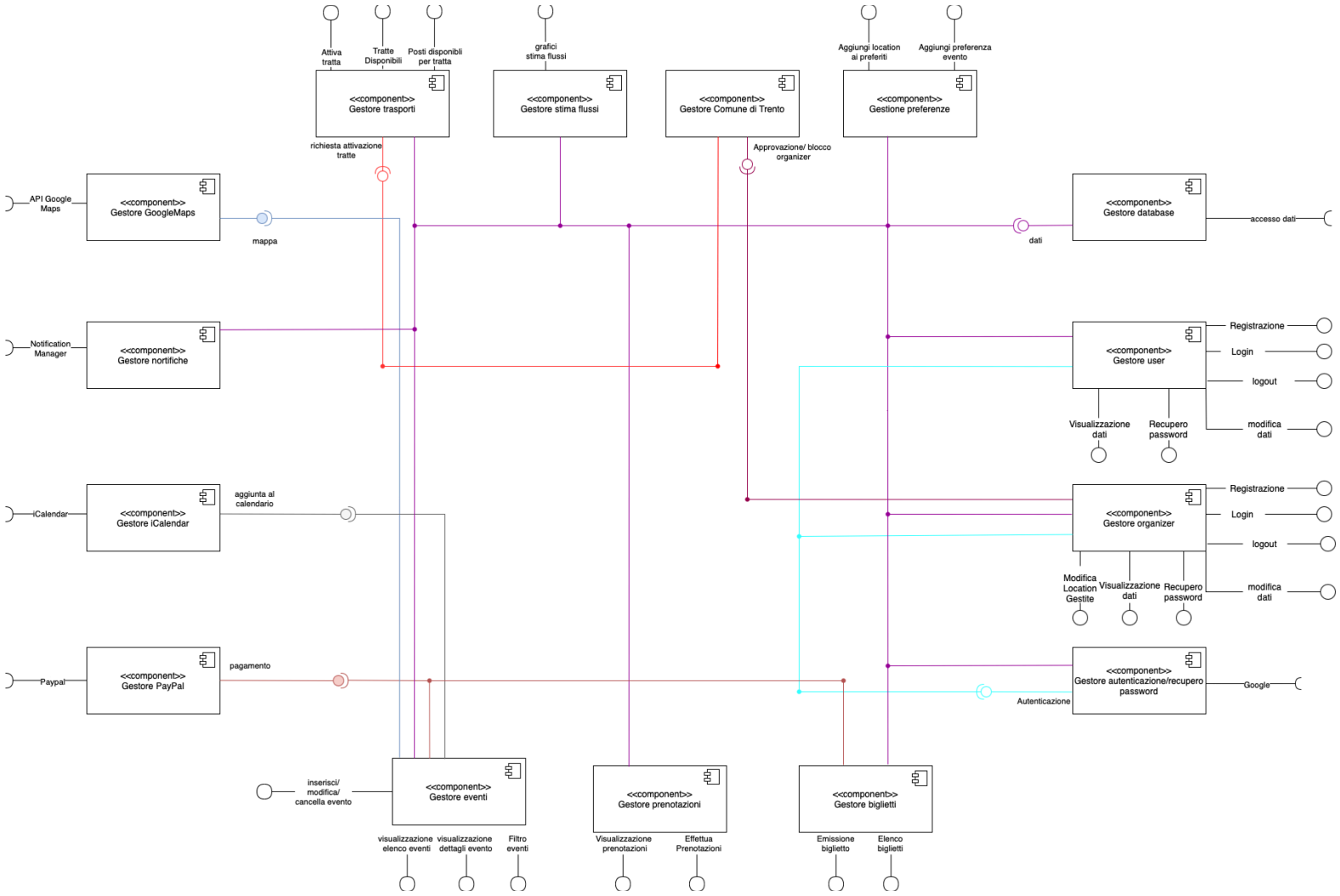
DELIVERABLE 2 - TRENTOUNICA

- Numero del gruppo → **15**
-

Nome	Cognome	Numero Matricola
Antonio	Amabile	238250
Luis	Bugnotti	238231
Niccolò	Cristoforetti	235806

1) Diagramma delle Componenti



Descrizione dei Componenti

Gestore PayPal

Questo componente gestisce le interazioni con il sistema esterno Paypal.

tipologia	**nome**	**descrizione**
fornita	pagamento	Il componente fornisce un'interfaccia per effettuare un pagamento con Paypal.
richiesta	paypal	Il componente utilizza l'interfaccia fornita da Paypal per richiedere un pagamento da parte dell'utente.

Gestore iCalendar

Questo componente gestisce l'integrazione degli eventi tramite l'API di iClendar

Tipologia	**Nome**	**Descrizione**
Fornita	Aggiunta al calendario	Fornisce un'interfaccia che permette agli utenti di aggiungere un evento al calendario tramite il formato iCalendar RF1
Richiesta	API iCalendar	Utilizza un'interfaccia standard per importare/esportare eventi da e verso calendari elettronici.

Gestore GoogleMaps

Questo componente gestisce l'interazione con il servizio Google Maps per la localizzazione e visualizzazione degli eventi e delle tratte.

Tipologia	**Nome**	**Descrizione**
Fornita	Visualizzazione Eventi su Mappa	Fornisce un'interfaccia per visualizzare eventi geolocalizzati RF1
Richiesta	API Google Maps	Utilizza l'API esterna Google Maps per la rappresentazione grafica

Gestore Notifiche

Questo componente gestisce tutte le notifiche che devono essere inviate agli utenti registrati e agli organizer.

Tipologia	**Nome**	**Descrizione**
Fornita	Notification Manager	Fornisce un'interfaccia per

		l'invio di notifiche relative a eventi annullati, nuovi eventi preferiti, approvazioni e blocchi degli organizer RF6
Richiesta	Dati	Richiede l'accesso ai dati degli utenti, degli organizer e degli eventi per determinare i destinatari e i contenuti delle notifiche da inviare

Gestore Organizer (user)

Questo componente gestisce la registrazione di utenti al sistema, il login, logout e gestione del proprio account agli utenti organizer e gli user (con l'unica differenza che questi ultimi non hanno l'interfaccia di approvazione/blocco organizer).

Tipologia	**Nome**	**Descrizione**
Fornita	Registrazione	Il componente fornisce un'interfaccia all'utente per registrarsi RF15
Fornita	Login	Il componente fornisce un'interfaccia all'utente per effettuare il login
Fornita	Logout	Il componente fornisce un'interfaccia all'utente per effettuare il logout
Fornita	Visualizzazione dati	Il componente fornisce un'interfaccia all'utente per visualizzare i propri dati sensibili. Un partecipante può vedere gli eventi a cui è registrato, mentre un gestore di eventi può visualizzare i propri eventi in corso
Fornita	Modifica dati	Il componente fornisce un'interfaccia in cui l'utente può modificare determinati dati, un partecipante può modificare qualsiasi suo dato a piacere, un gestore di eventi può modificare i dati di. Per gli altri richiede la conferma dal supervisore

Fornita	Recupero password	Il componente fornisce all'utente un'interfaccia per recuperare la password). Viene generato un link con un codice implicito. Con il codice implicito viene restituita la password.
Richiesta	Autenticazione	Il componente utilizza l'interfaccia del componente "Gestore registrazione/autenticazione " per autenticarsi ricevendo il token di accesso che deve essere poi confermato RF3
Richiesta	Dati	Il componente utilizza l'interfaccia del componente "Gestore database" per accedere ai dati salvati sul database RF1, RF7
Richiesta	Autenticazione Organizer	Richiede al "Gestore Comune di Trento" l'autorizzazione dell'organizzatore a operare sulla piattaforma, dopo la registrazione o modifiche rilevanti RF16
Richiesta	Blocco Organizer	Richiede al "Gestore Comune di Trento" di bloccare l'accesso all'organizzatore nel caso di revoca dei permessi

Gestore Database

Questo componente si occupa della gestione centralizzata dei dati relativi agli utenti, agli eventi, alle prenotazioni, ai biglietti e alle tratte. È responsabile della lettura, scrittura e aggiornamento delle informazioni persistenti all'interno del sistema.

Tipologia	**Nome**	**Descrizione**
Richiesta	Accesso dati	Richiede un meccanismo tecnico di accesso ai dati fisici, come database relazionale, ORM o servizi di persistenza, per poter salvare e recuperare le informazioni memorizzate.
Fornita	Dati	Fornisce l'accesso ai dati persistenti dell'applicativo,

		inclusi utenti, organizer, eventi, prenotazioni, preferenze, biglietti e tratte. Gli altri componenti consumano questi dati attraverso questa interfaccia

Gestore Autenticazione

Questo componente gestisce tutte le operazioni relative all'autenticazione degli utenti tramite provider esterni e il recupero della password dimenticata.

Tipologia	**Nome**	**Descrizione**
Richiesta	API di autenticazione esterna	Utilizza le API di Google per la validazione delle credenziali utente
Fornita	Autenticazione esterna	Fornisce un'interfaccia per autenticare gli utenti tramite provider esterni (Google) secondo RF3
Richiesta	Dati	Il componente utilizza l'interfaccia del componente "Gestore autenticazione" per accedere ai dati salvati sul database.

Gestore Prenotazioni

Questo componente gestisce le prenotazioni da parte degli utenti per gli eventi con prenotazione

Tipologia	**Nome**	**Descrizione**
Fornita	Visualizza prenotazioni	Il componente fornisce il dato del numero di prenotazioni dell'evento a pagamento specifico
Fornita	Effettua prenotazione	Il componente fornisce un'interfaccia all'utente che gli permette di prenotarsi per un evento a pagamento
Richiesta	Dati	Il componente utilizza l'interfaccia del componente "Gestore prenotazioni" per accedere ai dati salvati sul database.

Gestore Eventi

Questo componente gestisce tutte le interazioni degli eventi per ogni tipologia di attore coinvolto

Tipologia	**Nome**	**Descrizione**
Fornita	Visualizza eventi	Il componente fornisce all'utente,organizzatore,impiegato del comune la lista di eventi di propria competenza

Fornita	Visualizza dettagli evento	Il componente fornisce un'interfaccia all'utente,organizzatore o impiegato del comune nella quale mostra nel dettaglio tutte le informazioni di un evento creato e se ci sono tratte speciali
Fornita	Creazione evento	Il componente fornisce un'interfaccia per gli organizzatori per creare eventi, RF17
Fornita	Modifica evento	Il componente fornisce un'interfaccia al gestore eventi che permette la modifica degli attributi dell'evento. Viene coinvolto il gestore delle notifiche dopo la modifica dell'evento RF18,RF19
Fornita	Annullamento evento	Il componente fornisce al gestore eventi un'interfaccia che permette l'annullamento dell'evento, richiede una doppia conferma per annullare l'evento. Viene coinvolto il gestore delle notifiche dopo l'annullamento dell'evento RF18
Fornita	Filtro eventi	Il componente fornisce un'interfaccia per filtrare gli eventi per data, luogo, categoria RF 14
Richiesta	Pagamento	Richiede un pagamento da parte dell'utente
Richiesta	Mappa	Richiede l'interfaccia grafica di Google Maps
Richiesta	Aggiunta al calendario	Il componente utilizza l'interfaccia del componente "Gestione Google Calendar"
Richiesta	Dati	Il componente utilizza l'interfaccia del componente "Gestore eventi" per accedere ai dati salvati sul database.

Gestore Biglietti

Questo componente gestisce l'emissione dei biglietti elettronici associati alle prenotazioni di eventi e consente agli utenti di consultare l'elenco dei propri biglietti. Inoltre, interagisce con il servizio di pagamento e il database per gestire tutte le operazioni sui biglietti.

Tipologia	**Nome**	**Descrizione**
Fornita	Emissione biglietto	Fornisce un'interfaccia che permette la generazione di un biglietto elettronico al completamento di una prenotazione evento, associando un codice identificativo o QR univoco RF5
Fornita	Elenco biglietti	Fornisce un'interfaccia che consente agli utenti di consultare l'elenco aggiornato dei propri biglietti emessi per eventi prenotati RF7
Richiesta	Pagamento	Richiede l'interfaccia del servizio di pagamento (es. PayPal API) per verificare

		che il pagamento dell'evento sia stato completato prima di emettere il biglietto
Richiesta	Dati	Richiede l'interfaccia di accesso ai dati salvati nel database per recuperare informazioni relative a eventi prenotati, utenti e biglietti associati

Gestore Trasporti

Questo componente si occupa della gestione delle tratte in relazione agli eventi.

Tipologia	**Nome**	**Descrizione**
Fornita	Attiva tratta	Fornisce la possibilità di attivare una tratta in base alla richiesta di flusso utenti RF24
Fornita	Tratte disponibili	Fornisce l'elenco delle tratte attivate RF8
Fornita	Posti disponibili per tratta	Fornisce la visualizzazione dei posti ancora disponibili su una tratta speciale RF25
Richiesta	Richiesta attivazione tratta	Richiede informazioni dal Gestore Stima Flussi per determinare quando attivare una nuova tratta
Richiesta	Dati	Richiede accesso al database per aggiornare o consultare le tratte e le prenotazioni associate

Gestore stima flussi

Questo componente stima i flussi degli utenti previsti verso gli eventi sulla base delle prenotazioni, delle preferenze espresse e delle visualizzazioni.

Tipologia	**Nome**	**Descrizione**
Fornita	Grafici tima flussi	Fornisce dati aggregati e grafici sui flussi di partecipazione agli eventi. RF20, RF22, RF23.
Richiesta	Accesso dati	Richiede accesso ai dati di prenotazione, preferenze e visualizzazioni per calcolare i flussi

Gestore Comune di Trento

Questo componente rappresenta l'ente amministrativo che ha il compito di approvare o bloccare organizzatori di eventi e di autorizzare nuove tratte speciali di trasporto.

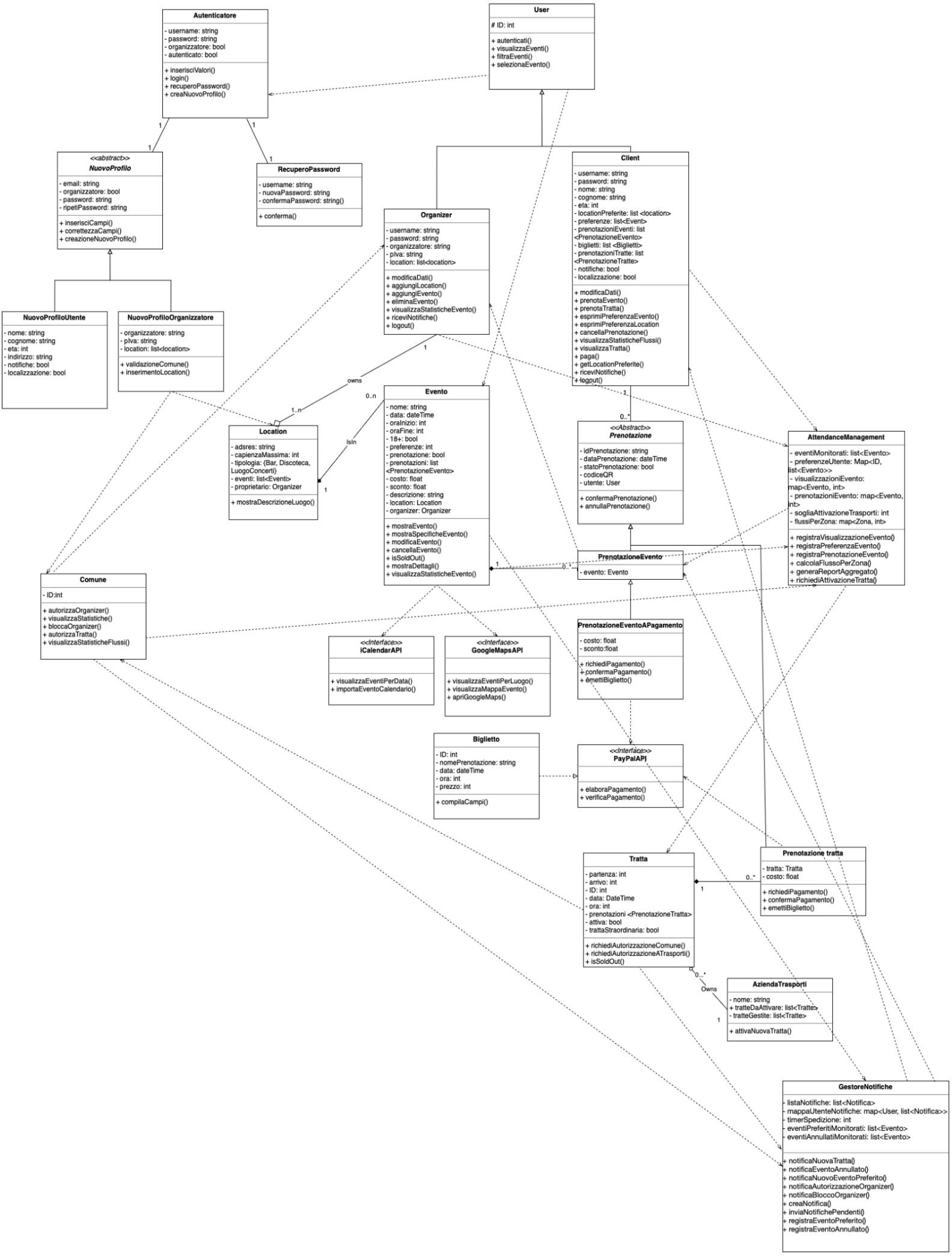
Tipologia	Nome	Descrizione
Forinita	Approvazione Organizer	Fornisce un'interfaccia per approvare nuovi organizer registrati, dopo la verifica della documentazione richiesta RF21
Fornita	Blocco Organizer	Fornisce un'interfaccia per bloccare organizer in caso di irregolarità o violazioni dei regolamenti RF21
Fornita	Richiesta attivazione tratta	Fornisce un'interfaccia per autorizzare l'attivazione di nuove tratte di trasporto basate sui dati di flusso utenti raccolti dal Gestore Stima Flussi RF24, RF25

Gestore Preferenze

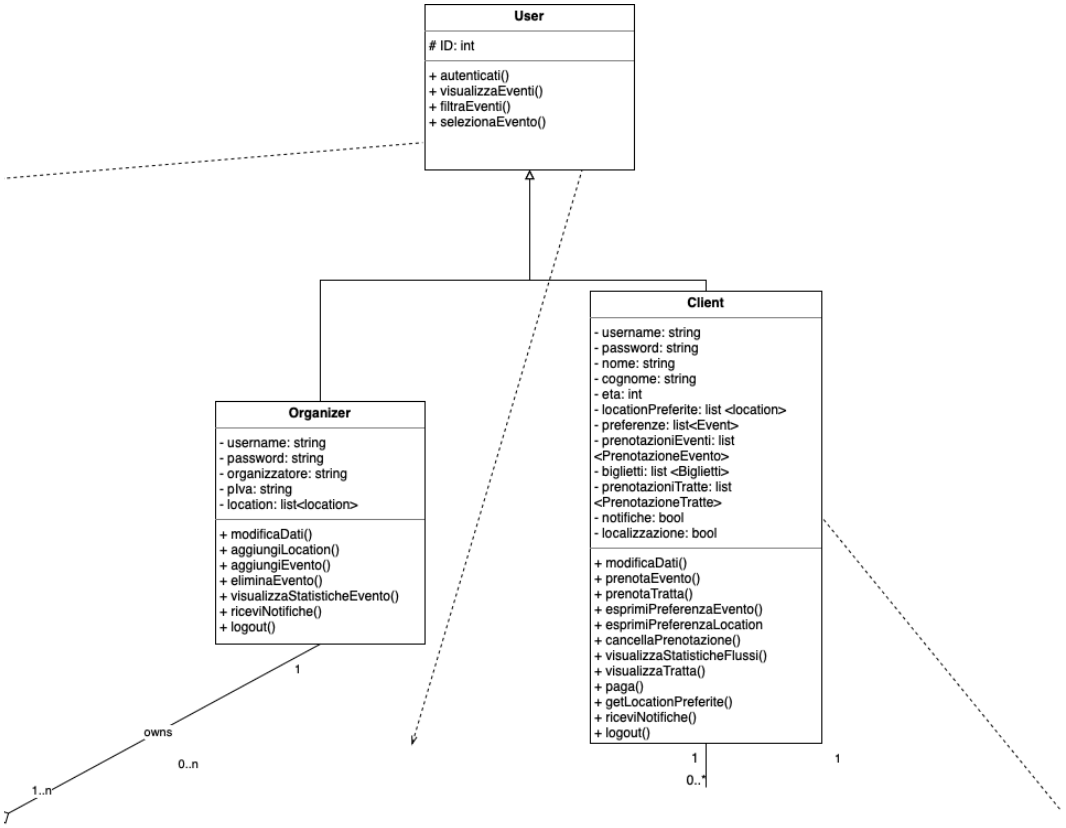
Questo componente gestisce le preferenze degli utenti riguardo eventi e location, permettendo di migliorare il sistema di suggerimenti e la stima dei flussi.

Tipologia	Nome	Descrizione
Fornita	Aggiunta preferenza evento	Fornisce un'interfaccia che permette agli utenti di esprimere la preferenza per eventi specifici RF4
Fornita	Aggiunta location ai preferiti	Fornisce un'interfaccia che permette agli utenti di aggiungere location ai propri preferiti RF11
Richiesta	Accesso dati	Richiede l'accesso ai dati per memorizzare e consultare le preferenze espresse dagli utenti

2) Diagramma delle classi



Descrizione delle classi



User

La classe User rappresenta un utente generico registrato nel sistema. Si occupa delle operazioni di base sugli eventi.

Operazione	**Definizione**	**Descrizione**
autenticati()	autenticati(operazione: string) → bool	Permette all'utente di avviare un'operazione di login, registrazione o recupero password, delegando completamente la gestione dei dati al Gestore Autenticatore. L'operazione viene specificata tramite il parametro operazione. Restituisce true se l'operazione viene completata correttamente, false altrimenti.
visualizzaEventi()	visualizzaEventi(modalita: string) → list	Permette all'utente di visualizzare gli eventi disponibili, specificando la

		modalità di visualizzazione desiderata. Il parametro modalita può assumere i valori "elenco", "data", "mappa". In base alla modalità, il metodo richiede alla classe Evento di restituire gli eventi visualizzati nel formato corrispondente. L'utente non interagisce direttamente con API esterne. Restituisce una lista di eventi.
filtraEvevnti()	filtraEventi(filtro: string) → list	Permette all'utente di filtrare gli eventi disponibili secondo un criterio specificato (es. categoria, luogo, data). Ritorna la lista di eventi filtrati
selezionaEvento()	selezionaEvento(idEvento: int) → Evento	Permette all'utente di selezionare un evento tramite l'ID evento. Ritorna l'oggetto Evento corrispondente

Organizer

La classe Organizer estende la classe User. Rappresenta un organizzatore registrato che può creare, modificare eventi e gestire location

Operazione	**Definizione**	**Descrizione**
modificaDati()	modificaDati(datiModificati: dict) → bool	Permette all'organizer di modificare i propri dati (es. password, pIva, nome). Restituisce true se la modifica è avvenuta correttamente.
aggiungiLocation()	aggiungiLocation(location: Location) → bool	Permette all'organizer di aggiungere una nuova location associata al proprio profilo. Restituisce true se l'aggiunta è avvenuta con successo.
aggiungiEvento()	aggiungiEvento(evento: Evento) → bool	Permette all'organizer di creare un nuovo evento in una location già registrata. Restituisce true in caso di successo.
eliminaEvento()	eliminaEvento(idEvento: int) → bool	Permette all'organizer di eliminare un evento precedentemente creato, identificato tramite ID. Restituisce true se

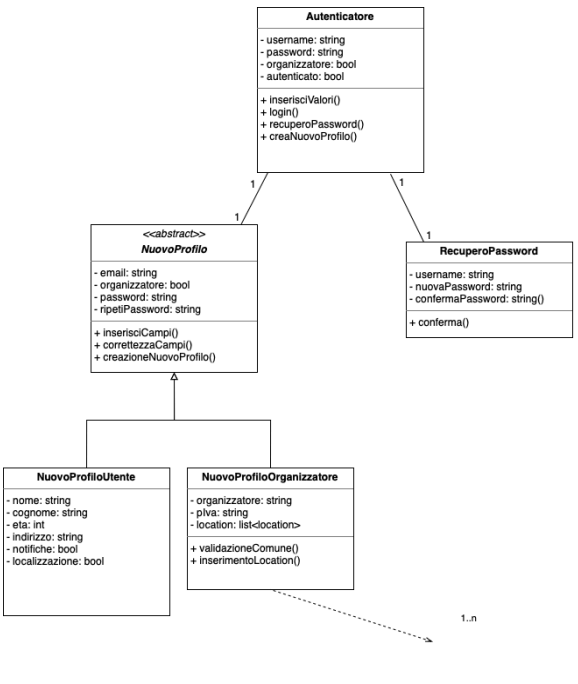
		l'eliminazione avviene correttamente.
visualizzaStatisticheEvento()	visualizzaStatisticheEvento(idEvento: int) → StatisticheEvento	Permette all'organizer di consultare statistiche relative a uno specifico evento come visualizzazioni o prenotazioni.
riceviNotifiche()	riceviNotifiche() → list	Permette all'organizer di ricevere notifiche amministrative e aggiornamenti sugli eventi.
logout()	logout() → void	Permette all'organizer di terminare la sessione corrente.

Client

La classe Client estende la classe User. Rappresenta un partecipante che può prenotare eventi, esprimere preferenze e gestire biglietti.

Operazione	**Definizione**	**Descrizione**
modificaDati()	modificaDati(datiModificati: dict) → bool	Permette al client di modificare i propri dati (es. email, password, preferenze notifiche). Restituisce true in caso di successo.
prenotaEvento()	prenotaEvento(idEvento: int) → bool	Permette al client di prenotare un evento specifico tramite ID. Ritorna true se la prenotazione è stata registrata.
prenotaTratta()	prenotaTratta(tratta: Tratta) → bool	Permette al client di prenotare un posto su una tratta di trasporto speciale legata agli eventi. Restituisce true se la prenotazione è registrata correttamente.
esprimiPreferenzaEvento()	esprimiPreferenzaEvento(idEvento: int) → bool	Permette al client di indicare interesse per un evento, aumentando il contatore delle preferenze.
espreimiPreferenzaLocation()	esprimiPreferenzaLocation(idLocation: int) → bool	Permette al client di aggiungere una location tra le proprie preferenze. Ritorna true se la preferenza è stata inserita.
cancellaPrenotazioone()	cancellaPrenotazione(idPrenotazione: int) → bool	Permette al client di annullare una prenotazione già effettuata, sia per eventi sia per tratte di trasporto.

		Restituisce true se l'annullamento è avvenuto con successo.
visualizzaStatisticheFLussi()	visualizzaStatisticheFlussi() → FlussiEvento	Permette al client di visualizzare dati aggregati sui flussi di partecipazione agli eventi
visualizzaTratta()	visualizzaTratta(idEvento: int) → list	Permette al client di visualizzare le tratte di trasporto disponibili per uno specifico evento.
paga()	paga(idPrenotazione: int, metodoPagamento: string) → bool	Permette al client di effettuare il pagamento relativo a una prenotazione. La prenotazione può riferirsi sia a un evento sia a una tratta di trasporto. È necessario specificare l'identificativo della prenotazione (idPrenotazione) e il metodo di pagamento scelto (metodoPagamento). Restituisce true se il pagamento avviene con successo.
getLocationPreferite()	getLocationPreferite() → list	Permette al client di visualizzare la lista delle location preferite salvate nel profilo.
riceviNotifiche()	riceviNotifiche() → list	Permette al client di ricevere notifiche relative a eventi preferiti o prenotati.
logout()	logout() → void	Permette al client di terminare la sessione corrente.



Autenticatore

Il componente Autenticatore gestisce tutte le operazioni relative alla validazione degli utenti: login, creazione nuovo profilo e recupero password.

Operazione	Definizione	Descrizione
inserisciValori()	inserisciValori(username: string, password: string, organizzatore: bool) → void	Permette di inserire le credenziali e il ruolo (organizzatore o cliente) per l'operazione successiva. Non restituisce alcun valore.
login()	login() → bool	Tenta di autenticare l'utente utilizzando i dati precedentemente inseriti. Se l'autenticazione ha successo, il metodo restituisce true e l'attributo interno autenticato viene impostato a true. In caso contrario, restituisce false e autenticatoviene impostato a false.
recuperoPassword()	recuperoPassword(username: string, nuovaPassword: string) → bool	Permette di recuperare la password per l'utente specificato. Richiede username e nuova password. Ritorna true se l'operazione ha successo.
creaNuovoProfilo()	creaNuovoProfilo(datiProfilo : NuovoProfilo) → bool	Permette la creazione di un nuovo profilo utente o organizer. Richiede un oggetto NuovoProfilo con i dati inseriti. Restituisce true se il profilo viene creato correttamente.

Nuovo Profilo (classe astratta)

La classe NuovoProfilo rappresenta il modello astratto per la creazione di nuovi profili utente (sia client sia organizer).
Non può essere istanziata direttamente.

Operazione	Definizione	Descrizione
inserisciCampi()	inserisciCampi(email: string, password: string, ripetiPassword: string, organizzatore: bool) → void	Permette di inserire i dati iniziali richiesti per la registrazione. Non ritorna valori.
correttezzaCampi()	correttezzaCampi() → bool	Verifica la correttezza dei campi inseriti, come corrispondenza tra password e ripetizione, validità dell'email.

		Restituisce true se i dati sono validi.
creazioneNuovoProfilo()	creazioneNuovoProfilo() → bool	Permette la creazione del profilo utente o organizer se i dati sono validi. Restituisce true se la creazione è avvenuta correttamente.

NuovoProfiloUtente

Estensione di Nuovo Profilo che rappresenta i dati specifici per un nuovo utente cliente.

NuovoProfiloOrganizzatore

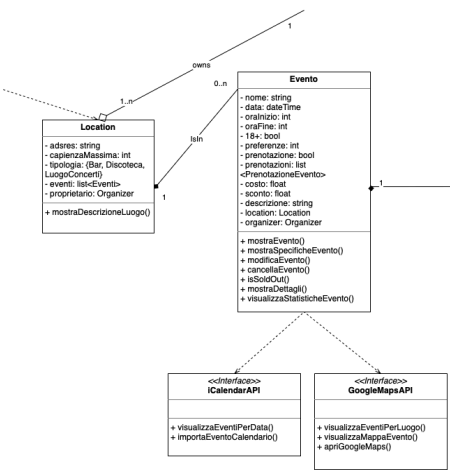
Estensione di Nuovo Profilo che gestisce i dati per un nuovo organizer e le relative location

Operazione	**Definizione**	**Descrizione**
validazioneComune()	validazioneComune() → bool	Permette di richiedere al Comune la validazione dell'organizzatore utilizzando i dati presenti nella classe (pIva e organizzatore). Restituisce true se l'organizzatore viene approvato.
inserimentoLocation()	inserimentoLocation() → bool	Permette all'organizzatore di inserire una nuova location associata al proprio profilo, agendo sui dati già memorizzati (location). Restituisce true se l'inserimento avviene correttamente.

RecuperoPassword

Questa classe supporta la gestione della procedura di recupero password dell'utente.

Operazione	**Definizione**	**Descrizione**
conferma()	conferma(username: string, nuovaPassword: string, confermaPassword: string) → bool	Permette di confermare l'aggiornamento della password di un utente. Verifica la corrispondenza tra nuova password e conferma. Restituisce true se la modifica avviene con successo.



Location

Questa classe rappresenta una location fisica dove possono essere ospitati eventi (bar, discoteche, luoghi per concerti).

Operazione	Definizione	Descrizione
mostraDescrizioneLuogo()	mostraDescrizioneLuogo() → string	Permette di ottenere una descrizione sintetica della location, contenente indirizzo, capienza e tipologia. Non richiede parametri di ingresso. Restituisce una stringa contenente la descrizione del luogo.

Evento

La classe Evento rappresenta un singolo evento organizzato all'interno di una location. Gestisce informazioni come prenotazioni, orari, costi e dettagli dell'evento.

Operazione	Definizione	Descrizione
mostraEvento()	mostraEvento(modalita: string) → string	Permette di visualizzare l'evento corrente (this) secondo la modalità specificata. In modalità "elenco", restituisce una descrizione sintetica dell'evento (nome, data, ora, location, costo e preferenze). In modalità "data" o "mappa", integra i dati tramite iCalendarAPI o GoogleMapsAPI per la visualizzazione.
mostraSpecificheEvento()	mostraSpecificheEvento() → dict	Restituisce un dizionario con le specifiche dettagliate dell'evento: nome, data, orario, location, prezzo, disponibilità posti.
modificaEvento()	modificaEvento(campiModificati: dict) → bool	Permette di modificare i dati dell'evento (es. orari, descrizione, costo). I dati da modificare vengono passati sotto forma di dizionario. Restituisce true se la modifica ha successo.
cancellaEvento()	cancellaEvento() → bool	Permette di cancellare l'evento dal sistema. Non richiede parametri. Restituisce true se l'eliminazione va a buon fine.

isSoldOut()	isSoldOut() → bool	Verifica se l'evento ha esaurito i posti disponibili (sold-out). Restituisce true se non ci sono più posti disponibili, false altrimenti.
mostrDettagli()	mostraDettagli() → string	Restituisce una descrizione dettagliata dell'evento, includendo: nome, descrizione, data, ora di inizio e fine, location, prezzo, sconto applicato e numero di preferenze ricevute.
visualizzaStatisticheEvento()	visualizzaStatisticheEvento() → StatisticheEvento	Permette di ottenere dati statistici legati all'evento corrente, come: numero di prenotazioni, numero di preferenze espresse, numero di visualizzazioni.

iCalendarAPI (interfaccia)

Questa interfaccia fornisce le funzionalità di integrazione con calendari elettronici, permettendo l'importazione e la consultazione degli eventi.

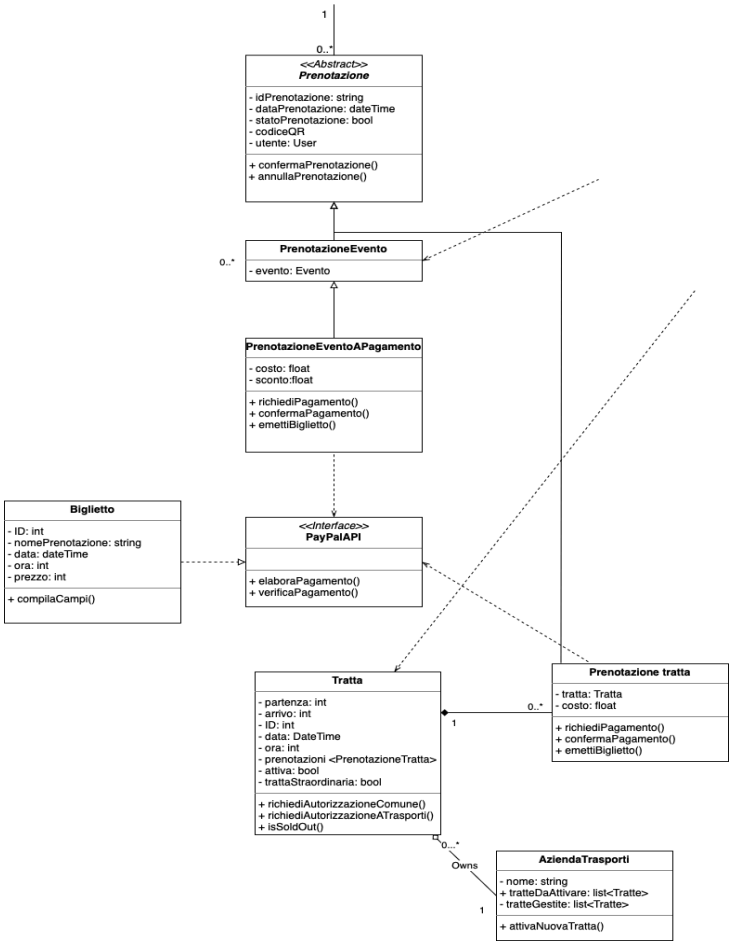
Operazione	**Definizione**	**Descrizione**
visualizzaEventiPerData()	visualizzaEventiPerData(data: dateTime) → list	Permette alla classe Evento di filtrare e visualizzare eventi in base alla data selezionata. È utilizzato internamente da Eventi per supportare la modalità "data" nella visualizzazione.
importaEventoCalendario()	importaEventoCalendario(evento: Evento) → bool	Permette di importare un evento nel calendario personale.

GoogleMapsAPI (interfaccia)

Questa interfaccia permette l'interazione con la piattaforma Google Maps per localizzare e visualizzare eventi sulla mappa.

Operazione	**Definizione**	**Descrizione**
visualizzaEventiPerLuogo()	visualizzaEventiPerLuogo(luogo: string) → list	Permette alla classe Evento di filtrare e visualizzare eventi in base alla loro localizzazione. È utilizzato internamente da Eventi per supportare la modalità "mappa" nella visualizzazione.
visualizzaMappaEventi()	visualizzaMappaEvento(evento: Evento) → string	Restituisce un link o una rappresentazione grafica

		della posizione geografica dell'evento.
apriGoogleMaps()	apriGoogleMaps(url: string) → void	Apre il servizio Google Maps al link della mappa relativa all'evento selezionato.



Prenotazione (classe astratta)

La classe Prenotazione è una classe astratta che rappresenta il concetto generico di prenotazione effettuata da un utente per un servizio (evento o tratta).Contiene gli attributi e i comportamenti comuni a tutte le prenotazioni.

Operazione	**Definizione**	**Descrizione**
confermaPrenotazione()	confermaPrenotazione() → bool	Permette di confermare una prenotazione impostandone lo stato su "confermato". Restituisce true se l'operazione ha successo.
annullaPrenotazione()	annullaPrenotazione() → bool	Permette di annullare una prenotazione cambiandone lo stato. Restituisce true se l'annullamento è avvenuto correttamente.

PrenotazioneEventoAPagamento

PrenotazioneEvento estende Prenotazione aggiungendo l'attributo evento: Evento, senza definire nuovi metodi propri.

PrenotazioneEventoAPagamento

Estende la classe Prenotazione evento, aggiungendo gestione dei costi e sconti per eventi a pagamento.

Operazione	**Definizione**	**Descrizione**
richediPagamento()	richiediPagamento() → bool	Richiede il pagamento relativo alla prenotazione tramite PayPal API. Restituisce true se la richiesta è stata inviata correttamente.
confermaPagamento()	confermaPagamento() → bool	Conferma l'avvenuto pagamento della prenotazione, aggiornando lo stato. Restituisce true se il pagamento viene validato correttamente.
emettiBiglietto()	emettiBiglietto() → Biglietto	Genera un biglietto associato alla prenotazione di un evento solo dopo la conferma del pagamento. Il biglietto contiene informazioni sull'evento (nome, data, ora, location) e il codice QR identificativo. Se il pagamento non risulta confermato, il biglietto non viene emesso.

Biglietto

La classe Biglietto rappresenta il biglietto digitale emesso a seguito della conferma di una prenotazione.

Operazione	**Definizione**	**Descrizione**
compilaCampi()	compilaCampi(id: int, nomePrenotazione: string, data: dateTime, ora: int, prezzo: int) → void	Permette di inserire i dati relativi alla prenotazione (id, nome evento, data, ora, prezzo) nel biglietto digitale. Non restituisce valore.

PayPalAPI (interfaccia)

Questa interfaccia rappresenta il collegamento con un sistema di pagamento esterno (come PayPal) per la gestione delle transazioni.

Operazione	**Definizione**	**Descrizione**
elaboraPagamento()	elaboraPagamento(importo: float) → bool	Permette di elaborare un pagamento per un importo specificato. Restituisce true se l'elaborazione avviene correttamente.

verificaPagamento()	verificaPagamento(idTransazione: string) → bool	Permette di verificare l'effettivo completamento di un pagamento associato a una transazione. Restituisce true se il pagamento risulta valido.

Tratta

La classe Tratta rappresenta un percorso di trasporto associato a eventi, eventualmente attivato in base alla domanda.

Operazione	**Definizione**	**Descrizione**
richiediAutorizzazioneComune()	richiediAutorizzazioneComune(idComune: int) → bool	Permette di richiedere al Comune l'autorizzazione per attivare una nuova tratta. Restituisce true se la richiesta viene presa in carico.
richiediAutorizzazioneAiTrasporti()	richiediAutorizzazioneATrasporti(nomeAzienda: string) → bool	Permette di inviare una richiesta all'azienda di trasporti per attivare una tratta, dopo l'approvazione del Comune.
isSoldOut()	isSoldOut() → bool	Verifica se i posti disponibili sulla tratta sono esauriti. Restituisce true se la tratta è completamente prenotata.

PrenotazioneTratta

La classe Prenotazione tratta rappresenta la prenotazione di un posto su una tratta di trasporto.

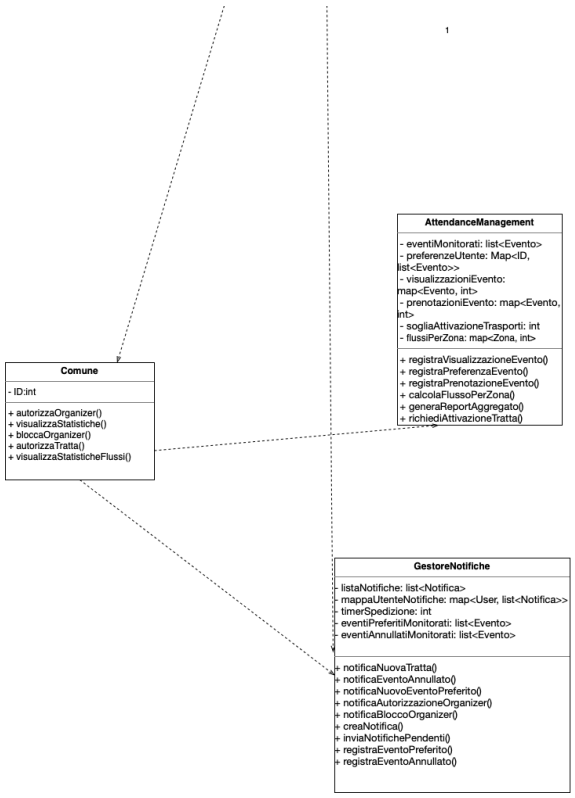
Operazione	**Definizione**	**Descrizione**
richiediPagamento()	richiediPagamento() → bool	Richiede il pagamento della prenotazione del posto sulla tratta, tramite PayPal API o metodo equivalente. Restituisce truese la richiesta è completata correttamente.
confermaPagamento()	confermaPagamento() → bool	Conferma l'avvenuto pagamento della prenotazione sulla tratta. Aggiorna lo stato a "confermato". Restituisce true se il pagamento è validato.
emettiBiglietto()	emettiBiglietto() → Biglietto	Genera un biglietto per la prenotazione di un posto su una tratta solo se il pagamento è stato confermato. Il biglietto conterrà informazioni sulla

		tratta (partenza, destinazione, orario) e il codice QR per la validazione. Se il pagamento non risulta completato, il biglietto non viene emesso.

AziendaTrasporti

La classe AziendaTrasporti rappresenta un gestore di tratte di trasporto, responsabile dell'attivazione delle tratte approvate.

Operazione	Definizione	Descrizione
attivaNuovaTratta()	attivaNuovaTratta(tratta: Tratta) → bool	Permette di attivare una nuova tratta nel sistema di trasporti gestito, aggiungendola alla lista delle tratte gestite. Restituisce true se l'attivazione va a buon fine.



Comune

La classe Comune rappresenta l'ente amministrativo responsabile dell'approvazione di organizer, eventi e tratte di trasporto speciali.

Operazione	Definizione	Descrizione
autorizzaOrganizer()	autorizzaOrganizer(idOrgani	Permette di approvare un

	zer: int) → bool	nuovo organizzatore registrato. Restituisce true se l'approvazione avviene con successo.
visualizzaStatistiche()	visualizzaStatistiche(idEvento: int) → StatisticheEvento	Permette di consultare le statistiche relative a un singolo evento (prenotazioni, visualizzazioni, preferenze).
bloccaOrganizer()	bloccaOrganizer(idOrganizer: int) → bool	Permette di bloccare un organizzatore esistente per irregolarità. Restituisce true se l'operazione va a buon fine.
autorizzaTratta()	autorizzaTratta(idTratta: int) → bool	Permette di approvare l'attivazione di una nuova tratta di trasporto. Restituisce true se la tratta viene autorizzata.
visualizzaStatisticheFlussi()	visualizzaStatisticheFlussi() → list	Permette di visualizzare dati aggregati sui flussi di partecipazione agli eventi.

AttendanceManagement

Questa classe si occupa della gestione del monitoraggio degli eventi, delle preferenze espresse dagli utenti, delle visualizzazioni e delle prenotazioni. Permette inoltre di analizzare i flussi di partecipazione e richiedere l'attivazione di tratte di trasporto.

Operazione	**Definizione**	**Descrizione**
registraVisualizzazioneEvento()	registraVisualizzazioneEvento(evento: Evento) → void	Registra una visualizzazione dell'evento indicato, aggiornando il contatore delle visualizzazioni e il flusso per la zona associata.
registraPreferenzaEvento()	registraPreferenzaEvento(utente: User, evento: Evento) → void	Registra l'espressione di una preferenza da parte di un utente verso un evento, aggiornando anche il monitoraggio dei flussi.
regitraPrenotazioneEvento()	registraPrenotazioneEvento(utente: User, evento: Evento) → void	Registra una prenotazione effettuata da un utente per un evento, aggiornando i dati di monitoraggio e i flussi per zona.
calcolaFlussoPerZona()	calcolaFlussoPerZona() → map<Zona, int>	Calcola il numero di utenti registrati (visualizzazioni, preferenze, prenotazioni) per ogni zona geografica. Ritorna una mappa zona → numero utenti.
generaReportAggregato()	generaReportAggregato() →	Genera un report aggregato

	ReportAggregato	dei flussi di partecipazione e di preferenze, da inviare al Comune di Trento per l'analisi. Il report aggregato include anche i flussi per zona salvati nell'attributo flussiPerZona
richiediAttivazioneTratta()	richiediAttivazioneTratta(zona: Zona) → bool	Richiede l'attivazione di una nuova tratta di trasporto per una zona che ha superato la soglia minima di partecipanti. Restituisce true se la richiesta viene inoltrata correttamente.

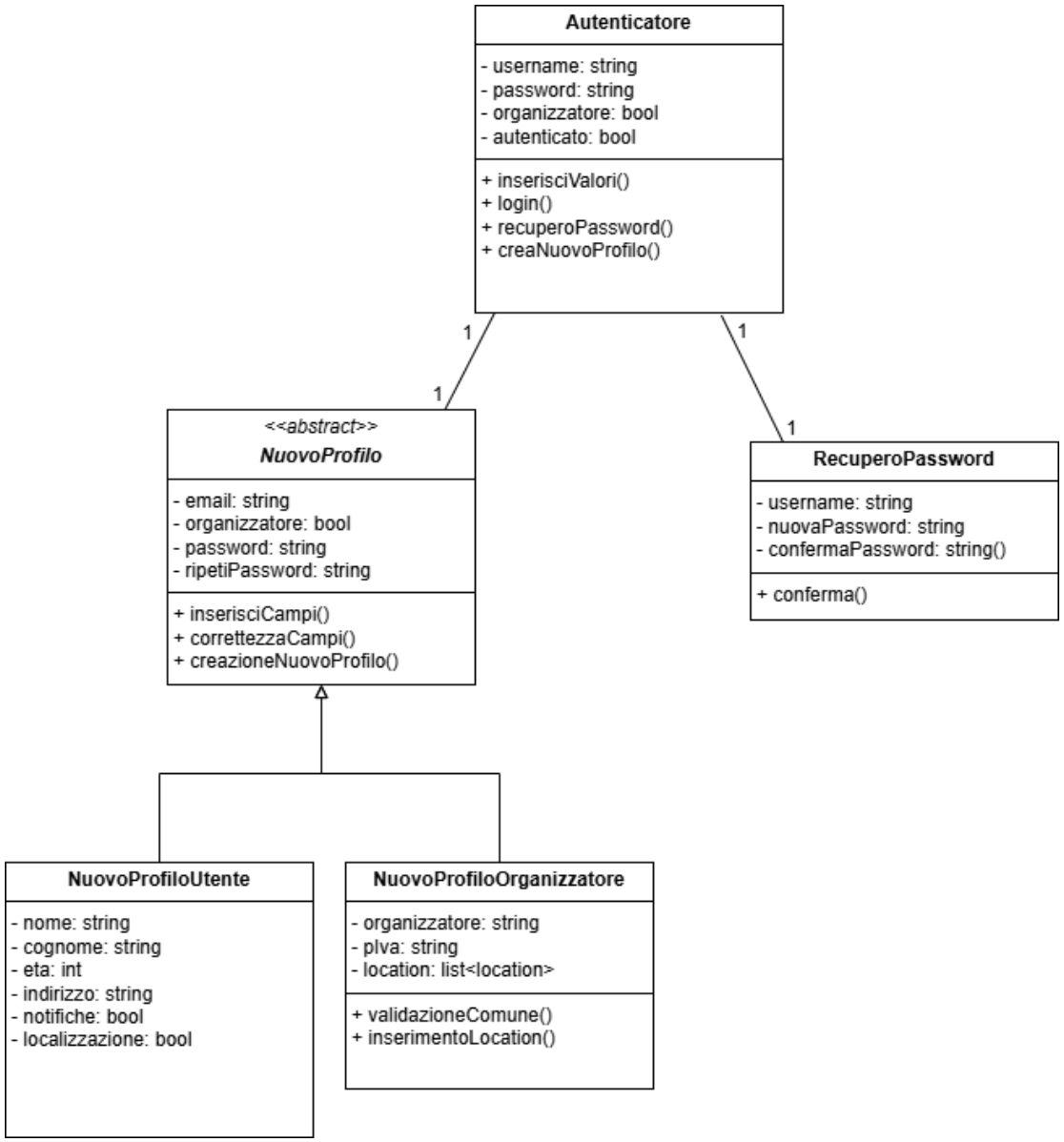
GestoreNotifiche

Questa classe si occupa della gestione, creazione e invio delle notifiche agli utenti in relazione a eventi preferiti, eventi annullati o nuovi eventi.

Operazione	**Definizione**	**Descrizione**
notificaNuovaTratta()	notificaNuovaTratta(evento: Evento) → void	Registra una nuova notifica relativa alla disponibilità di una tratta di trasporto per un evento. La notifica viene aggiunta alla lista delle notifiche pendenti e inviata successivamente tramite inviaNotifichePendenti().
notificaEventoAnnullato()	notificaEventoAnnullato(evento: Evento) → void	Recupera gli utenti che hanno prenotato o espresso preferenza per l'evento annullato. Crea una notifica di annullamento dell'evento destinata solo a questi utenti. La notifica viene registrata nella lista delle notifiche pendenti per l'invio successivo.
notificaNuovoEventoPreferito()	notificaNuovoEventoPreferito(utente: User, evento: Evento) → void	Registra una notifica per avvisare un utente della pubblicazione di un nuovo evento corrispondente alle sue preferenze. L'invio avverrà successivamente.
notificaAutorizzzazioneOrganizer()	notificaAutorizzazioneOrganizer(organizer: Organizer) → void	Registra una notifica per comunicare a un organizzatore l'avvenuta autorizzazione da parte del Comune di Trento.
notificaBloccoOrganizer()	notificaBloccoOrganizer(organizer: Organizer) → void	Registra una notifica per comunicare a un organizzatore che è stato bloccato dal Comune di Trento.

creaNotifica()	creaNotifica(testo: string, destinatari: list) → Notifica	Crea una nuova notifica personalizzata destinata a un insieme di utenti. La notifica viene memorizzata tra quelle pendenti in attesa di invio.
inviaNotifichePendenti()	inviaNotifichePendenti() → void	Scorre tutte le notifiche pendenti registrate e le invia ai rispettivi destinatari. Dopo l'invio, aggiorna o rimuove le notifiche inviate. La funzione può essere chiamata manualmente o automatizzata tramite un timer interno
registraEventoPreferito()	registraEventoPreferito(utente: User, evento: Evento) → void	Registra che un utente ha espresso interesse per un evento. In base alle preferenze, può essere generata una notifica che sarà inviata successivamente.
registraEventoAnnullato()	registraEventoAnnullato(evento: Evento) → void	Registra che un evento è stato annullato. Predispone la creazione delle notifiche relative da inviare agli utenti interessati.

3) Constraint OCL



3.1 Classe Nuovo profilo

Controlla che l'email non sia vuota e che abbia almeno la struttura testo@testo.estensione
context NuovoProfilo inv EmailValida: not self.email.oclIsUndefined() and self.email <> '' and self.email.matches('^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$')

Obbliga la password ad avere tra 8 e 20 caratteri inclusi, e a contenere almeno una lettera minuscola, una lettera maiuscola, un numero e un carattere speciale.
context NuovoProfilo inv PasswordLunghezzaCorretta: self.password.size() >= 8 and

```
    self.password.size() <= 20 and
    self.password.matches('.*[a-z].*') and
    self.password.matches('.*[A-Z].*') and
    self.password.matches('.*[0-9].*') and
    self.password.matches('.*[^a-zA-Z0-9].*')
```

Restituisce vero se password e ripeti password coincidono
context NuovoProfilo::correttezzaCampi(): Boolean post: result = (self.password = self.ripetiPassword)

Se organizzatore è true, allora deve essere creato un Nuovo Profilo Organizzatore altrimenti un Nuovo Profilo Utente
context NuovoProfilo inv: CorrettezzaTipoProfilo: (organizzatore = true implies self.oclIsTypeOf(NuovoProfiloOrganizzatore)) and (organizzatore = false implies self.oclIsTypeOf(NuovoProfiloUtente))

3.2 Classe Autenticatore

Autenticato sarà true se username e password sono corretti
context Autenticatore::login() : Boolean post: result = true implies self.autenticato = true

3.3 Classe Recupero Password

Nuova password e conferma password devono coincidere
context RecuperoPassword inv: PasswordsCoincidenti: self.nuovaPassword = self.confermaPassword

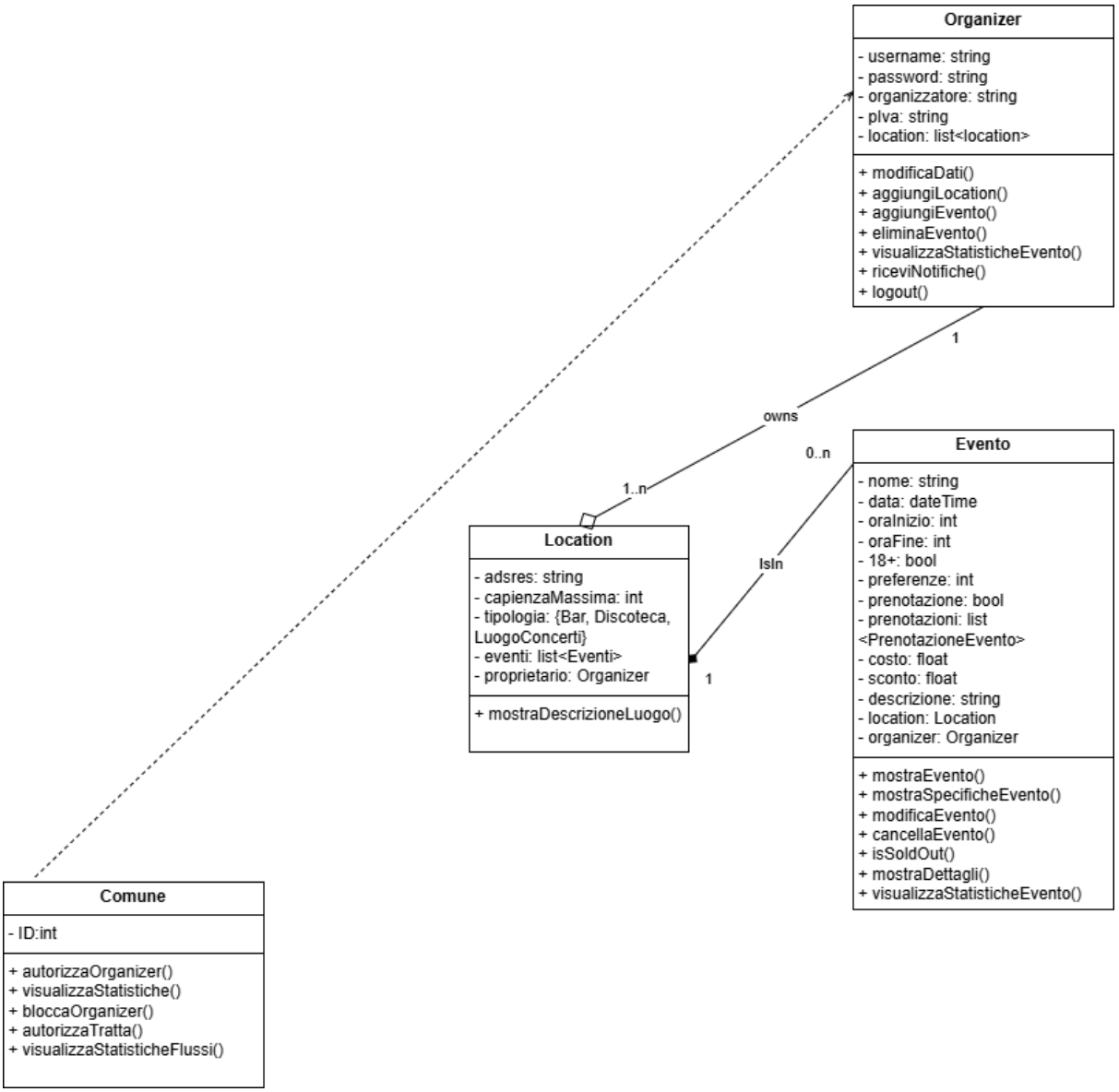
3.4 Classe Client

Verifica che l'evento che il Client sta prenotando non sia sold out e che un evento non sia riservato ai maggiorenni (soloMaggiorenni = false) oppure se ha un'età pari o superiore a 18 anni (eta >= 18)
context Client::prenotaEvento(idEvento: Integer) pre: let e : Evento = Evento.allInstances()->any(ev | ev.ID = idEvento) in not e.isSoldOut() and (not e.soloMaggiorenni or self.eta >= 18)

Verifica che le notifiche siano visibili solamente se il Client ha attivato la ricezione delle notifiche
context Client::riceviNotifiche() pre: self.notifiche = true

Verifica che non si possa aggiungere una preferenza ad un evento che non esiste, dopo l'operazione, il sistema aumenta di uno rispetto al dato precedente il conteggio delle preferenze sull'evento selezionato.
context Client::esprimiPreferenzaEvento(idEvento: Integer) pre: Evento.allInstances() -> exists(e | e.ID = idEvento) post: let e = Evento.allInstances()->any(ev | ev.ID = idEvento) in e.preferenze = e.preferenze@pre + 1

3.5 Classe Organizer



Verifica che il metodo aggiungiEvento(e: Evento), l'evento che si sta aggiungendo abbia: la data che non sia antecedente al giorno in cui si sta creando l'evento, che l'ora di fine sia maggiore dell'ora di inizio, che la location non sia nulla e che la lunghezza del nome dell'evento non sia 0
context Organizer::aggiungiEvento(e: Evento) pre: e.data >= DateTime::now() and e.oraFine > e.oraInizio and e.location <> null and e.nome.size() > 0

Verifica che il metodo aggiungiLocation(l: Location), la location che si vuole aggiungere abbia il nome, l'indirizzo e la città non nulli, la capienza massima deve essere un valore maggiore di zero, la tipologia deve essere una tra: bar,discoteca o LuogoConcerti

```
context Organizer::aggiungiLocation(l: Location)
pre: l.nome.size() > 0 and
     l.indirizzo..size() > 0 and
     l.città.size() > 0 and
     l.capienzaMassima > 0 and
    (l.tipologia = Tipologia::Bar or
     l.tipologia = Tipologia::Discoteca or
     l.tipologia = Tipologia::LuogoConcerti)
```

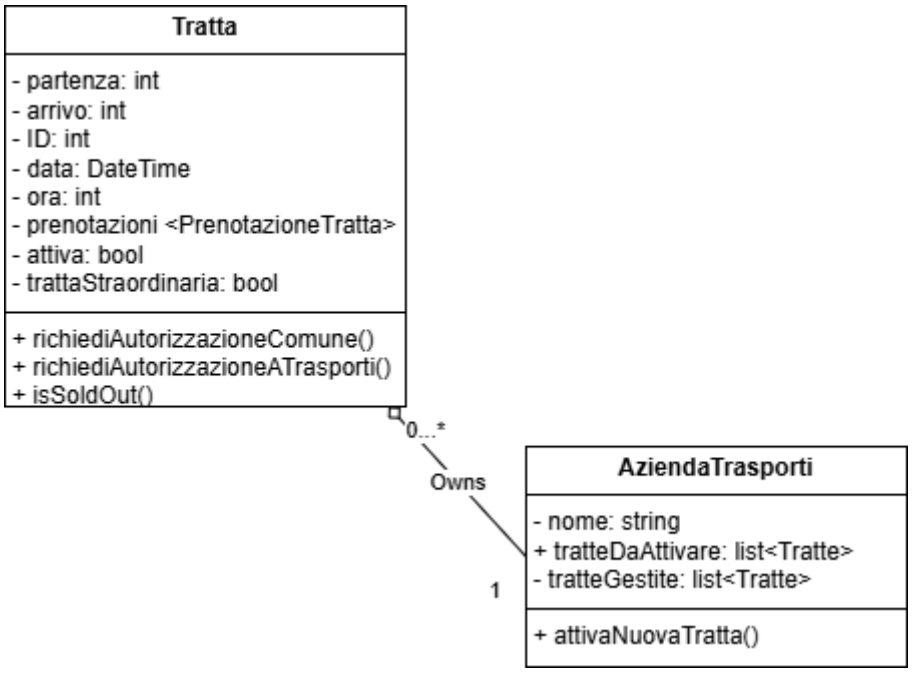
3.6 Classe Comune

Verifica che il metodo autorizzaOrganizer(o: Organizer) consenta al Comune di autorizzare un organizzatore solo se la sua partita IVA (o.pIva) è composta esattamente da undici caratteri, e abbia al suo interno solo numeri.
context Comune::autorizzaOrganizer(o: Organizer) pre: o.pIva.matches('[0-9]{11}')

3.7 Classe Evento

Garantisce che il metodo isSoldOut() della classe Evento rifletta correttamente la capienza della location in base alla capienza massima
context Evento inv: self.isSoldOut() = (self.prenotazioni->size() = self.location.capienzaMassima)

3.8 Classe AziendaTrasporti



Garantisce che il metodo attivaNuovaTratta, aggiunge una nuova Tratta sole se

l'orario di partenza (t.partenza) è antecedente rispetto all'orario di arrivo (t.arrivo), controlla che il giorno della tratta sia successivo o uguale al giorno dell’attivazione. Dopo l'attivazione, la tratta t deve risultare registrata nella lista delle tratte gestite (tratteGestite).

```
context AziendaTrasporti::attivaNuovaTratta(t: Tratta)
pre: t.partenza < t.arrivo and
     t.data >= DateTime::now()
post: self.tratteGestite->exists(tratta | tratta = t)
```