Friend Stuff DB

Autore: Federico Marra 0245408
Corso di laurea: Informatica
Email: marrafederico@proton.me

Indice

1 Introduzione

Friend Stuff è una piattaforma pensata per semplificare l'organizzazione di eventi, la gestione delle spese comuni e la condivisione dei ricordi all'interno di gruppi di amici.

1.1 Problematiche individuate

L'idea del progetto nasce dall'esigenza di avere un luogo unico dove poter organizzare e gestire le spese legate ad eventi tra persone. Oggi, per pianificare una semplice giornata fuori, ci si ritrova a utilizzare un insieme frammentato di applicazioni:

- app di messaggistica per comunicare,
- calendari condivisi per fissare le date,
- applicazioni finanziarie per gestire le spese,
- servizi di cloud storage per condividere foto e video.

Questo approccio frammentato porta a confusione, informazioni perse e tensioni nei gruppi. Friend Stuff nasce per risolvere proprio queste difficoltà, offrendo un unico spazio digitale, ordinato e trasparente.

1.2 Target di utenza

La piattaforma si rivolge a gruppi che organizzano regolarmente attività insieme e cercano un'alternativa ai social generalisti, mantenendo focus su comunicazione, pianificazione e ricordi condivisi.

2 Analisi dei requisiti

La piattaforma è progettata per offrire a utenti registrati un ambiente centralizzato in cui poter gestire dinamiche sociali e organizzative legate a gruppi ed eventi. Gli utenti, una volta autenticati, hanno la possibilità di instaurare connessioni sociali tramite l'aggiunta di amici e, successivamente, creare gruppi privati composti da altri utenti della piattaforma. All'interno di ciascun gruppo, è possibile organizzare eventi dedicati, a cui partecipano esclusivamente i membri del gruppo stesso. Ogni evento dispone di una chat integrata attraverso la quale i partecipanti possono comunicare in modo diretto. In aggiunta, la piattaforma consente di gestire in maniera centralizzata i contenuti multimediali prodotti durante gli eventi (come foto e video), riducendo la necessità di utilizzare servizi esterni e minimizzando la frammentazione delle informazioni. Gli utenti che prendono parte a un evento possono registrare le spese sostenute e richiedere la suddivisione automatica degli importi. Il sistema calcola le quote individuali da versare o ricevere, semplificando le operazioni di rimborso e contribuendo a evitare incomprensioni tra i partecipanti.

A seguito di una fase di raccolta e analisi dei requisiti, sono state individuate le principali entità che compongono il modello concettuale del sistema:

1. users identificata da username. Caratterizzata da nome, cognome e email.
2. user_groups identificata dal nome del gruppo e una descrizione.
3. events identificata dal nome dell'evento, caratterizzata da una descrizione, data di inizio, data di fine e una categoria.
4. locations identificata dal nome della location e l'indirizzo.
5. expenses identificata dal nome della spesa. È associata a un evento e a un utente (che effettua la spesa), ed è caratterizzata da un importo e una descrizione.
6. chat Ogni evento ha una chat ed è identificata da un codice.
7. messages Rappresenta un singolo messaggio inviato in una chat. È identificato da un codice ed è caratterizzato dal contenuto e dalla data di invio.
8. attachments Rappresenta gli allegati (foto/video etc..) mandati tramite messaggi in una chat. Sono identificati dal nome del file.

2.1 Glossario dei termini

Termine	Descrizione
Utente/Utente registrato	Persona autenticata nel sistema tramite email e password.
Gruppo di utenti	Collettivo privato di utenti registrati e autenticati.
Evento/Attività	Iniziativa (es. gita, cena, vacanza) organizzata all'interno di un gruppo.
Spesa	Transazione economica registrata da un utente durante un evento.
Chat	Canale di messaggistica associato a un evento.
Ricordo/Media	Contenuto multimediale (foto/video) condiviso in una chat.
Messaggio	Testo inviato da un utente nella chat dell'evento.
Admin	Utente che ha creato un gruppo e ne detiene i poteri amministrativi.
Member	Utente che fa parte di un gruppo senza poteri amministrativi.

Tabella 1: Glossario dei termini

2.2 Schema Entity-Relationship (ER)

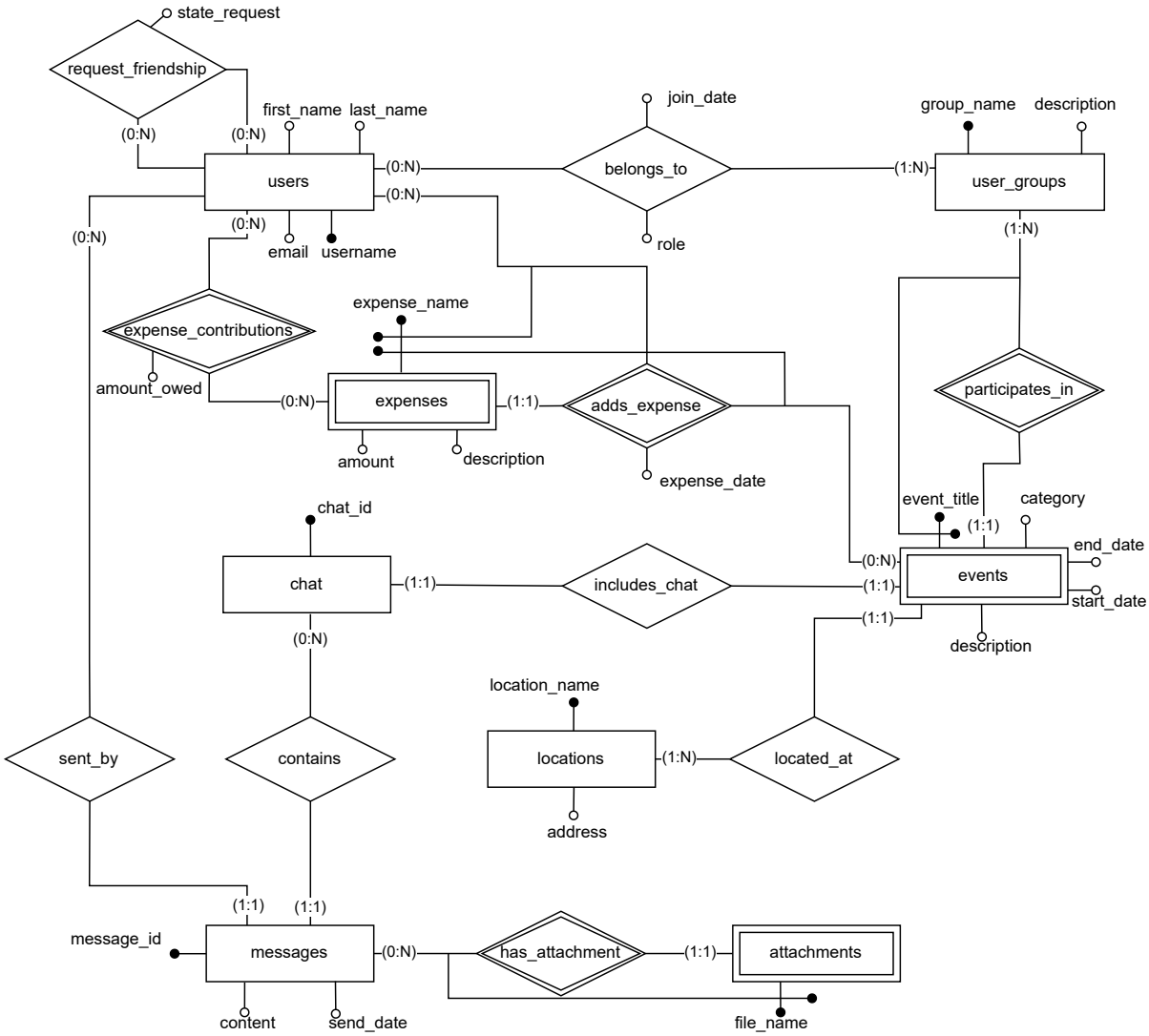


Figura 1: Schema ER

2.3 Glossario delle entità

Entità	Attributi	Descrizione
users	`username`, email, first_name, last_name	Identifica un utente registrato sulla piattaforma.
user_groups	`group_name`, description	Identifica un gruppo di utenti all'interno della piattaforma.
expenses	`expense_name`, amount, description	Identifica una spesa nel contesto di un evento.
events	`event_name`, description, start_date, end_date, category	Descrive l'evento al quale un gruppo di utenti parteciperà.
locations	`location_name`, address	Descrive il luogo in cui si svolge l'evento.
chat	`chat_id`	Identifica una chat all'interno di un evento.
messages	`message_id`, content, send_date	Identifica un messaggio inviato da un utente all'interno di una chat di un evento.
attachments	`file_name`	Identifica gli allegati inviati tramite messaggi in una chat.

Tabella 2: Glossario delle entità

2.4 Glossario delle relazioni

Relazione	Entità coinvolte	Descrizione
request_friendship	users	Un utente può inviare nessuna o più richieste d'amicizia. Un utente può ricevere nessuna o molte richieste. Una richeista d'amicizia ha uno stato: Accettata, Rifiutata o in attesa (default).
belongs_to	users, user_groups	Un utente può appartenere a più gruppi. Un gruppo ha almeno un membro. Inoltre ogni membro del gruppo ha un ruolo (member o admin) e una data di adesione.
participates_in	users, events	Un gruppo di utenti partecipa a uno o più eventi. Ad un evento può partecipare solo il gruppo che lo ha organizzato.
adds_expense	users, expenses, events	Un utente registra nessuna o più spese durante un evento. Un evento ha nessuna o più spese. Una spesa è correlata ad un solo evento ed utente. Inoltre si memorizza anche la data della spesa effettuata.
expense_contributions	users, expenses	Nessuno o più utenti possono partecipare alla divisione della spesa. La spesa può essere suddivisa tra nessuno o più utenti. Naturalmente la spesa può essere suddivisa solo tra membri di uno stesso gruppo.
located_at	events, locations	Un evento si svolgerà in un luogo. In un luogo possono essere stati svolti uno o più eventi.
includes_chat	events, chat	Ogni evento ha al più una chat associata. Ad un evento è associata al più una chat.
contains	chat, messages	Una chat contiene nessuno o molti messaggi. Un messaggio è correlato ad una chat specifica.
sent_by	users, messages	Ogni messaggio è correlato ad un utente specifico. Un utente può aver mandato nessuno o molti messaggi.
has_attachment	messages, attachment	Ogni messaggio può contenere nessuno o più allegati. Un allegato è associato ad un solo messaggio.

Tabella 3: Glossario delle relazioni

2.5 Vincoli d'integrità sui dati

Sono stati identificati i seguenti vincoli d'integrità sui dati per le entità discusse precedentemente.

2.5.1 users

Attributo	Tipo di vincolo	Dettagli
username	chiave primaria	Attributo di tipo stringa.
email	obbligatorio	Attributo di tipo stringa, deve essere un formato email compatibile.
first_name	obbligatorio	Attributo di tipo stringa.
last_name	obbligatorio	Attributo di tipo stringa.

Tabella 4: Vincoli entità users

2.5.2 user groups

Attributo	Tipo di vincolo	Dettagli
group_name	chiave primaria	Attributo di tipo stringa.
description	opzionale	Attributo di tipo stringa.

Tabella 5: Vincoli entità user_groups

2.5.3 events

Attributo	Tipo di vincolo	Dettagli
event_name	chiave parziale	Attributo di tipo stringa. Identificata tramite relazione participates_in con user_groups.
category	obbligatorio	Attributo di tipo stringa.
description	obbligatorio	Attributo di tipo stringa.
start_date	obbligatorio	Data di inzio dell'evento.
end_date	obbligatorio	Data fine evento. La data di fine evento non può essere precedente alla data di inizio evento.

Tabella 6: Vincoli entità events

2.5.4 locations

Attributo	Tipo di vincolo	Dettagli
locations_name	chiave primaria	Attributo di tipo stringa.
address	opzionale	Attributo multivalore che contiene informazioni sulla posizione del luogo (city, street name, number).

Tabella 7: Vincoli entità locations

2.5.5 expenses

Attributo	Tipo di vincolo	Dettagli
expenses_name	chiave parziale	Attributo di tipo stringa. Identificata tramite relazione adds_expense con users e events.
amount	obbligatorio	numero decimale.
description	obbligatorio	Attributo di tipo stringa.

Tabella 8: Vincoli entità expenses

2.5.6 chat

Attributo	Tipo di vincolo	Dettagli
chat_id	chiave primaria	Attributo di tipo intero.

Tabella 9: Vincoli entità chat

2.5.7 messages

Attributo	Tipo di vincolo	Dettagli
message_id	chiave primaria	Attributo di tipo intero.
content	obbligatorio	Attributo di tipo stringa.
send_date	obbligatorio	Data di invio del messaggio.

Tabella 10: Vincoli entità messages

2.5.8 attachments

Attributo	Tipo di vincolo	Dettagli
file_name	chiave parziale	Attributo di tipo stringa. Identificata tramite la relazione has_attachment con messages.

Tabella 11: Vincoli entità attachments

2.6 Classi di utenza

Durante l'analisi dei requisiti sono state individuate due classi di utenza all'interno di un gruppo:

- **Admin**: un utente con privilegi amministrativi limitati al gruppo che ha creato. Oltre a poter gestire gli utenti del proprio gruppo, l'admin ha anche tutte le funzionalità di un normale membro.
- **Member**: un utente standard del sistema non dispone di permessi amministrativi in un gruppo.

2.7 Elenco delle operazioni

Basandosi sull'analisi dei requisiti svolta finora, sarà possibile svolgere le principali operazioni sui dati per la realizzazione delle seguenti funzioni.

Funzione	Admin (gruppo)	Descrizione
registration/login	no	Registrazione/autenticazione di un utente sulla piattaforma.
add/remove friend	no	Gestione delle amicizie tra utenti.
create group	no	Creazione di un gruppo.
remove group	si	Rimozione di un gruppo.
create/remove events	sì	Gestione degli eventi all'interno del gruppo.
add/remove expense	no	Registrazione o cancellazione di spese personali in un evento.
send/delete message	no	Invio o eliminazione di un messaggio nella chat dell'evento.
assign admin	sì	Promozione o revoca del ruolo di admin per altri membri.
leave group	no	Permette all'utente di uscire da un gruppo.
add/remove group member	sì	L'admin può aggiungere o rimuovere utenti dal gruppo.
update/remove profile	no	Modifica dei propri dati o cancellazione del profilo.
refund friend	no	Rimborso di una quota da parte di utenti.

Tabella 12: Operazioni sui dati

3 Progettazione logica

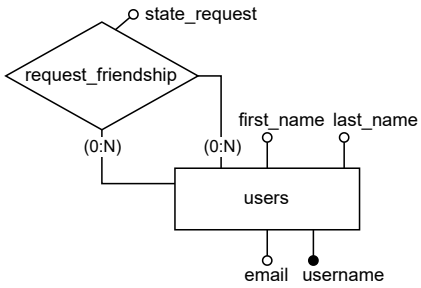
In questa fase di progettazione andremo a tradurre la schema concettuale definito nella precedente fase in un modello relazionale. L'obiettivo è trasformare ogni entità identificata in una tabella. Ciascuna di queste tabelle avrà come colonne gli attributi precedentemente definiti per le rispettive entità. Infine bisognerà tradurre le associazioni di cardinalità tra le entità. Questo viene fatto tramite l'utilizzo di chiavi primarie e chiavi esterne, garantendo così l'integrità referenziale e la coerenza dei dati.

3.1 Traduzione relazioni di cardinalità

Traduciamo le associazioni e le entità in tabelle.

3.1.1 users — users

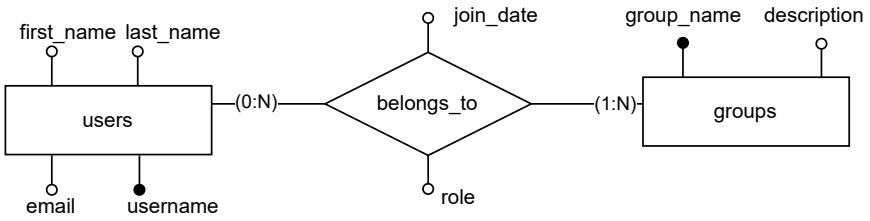
Abbiamo una relazione ricorsiva su users per il rapporto di amicizia tra utenti. In questo caso bisognerà creare una tabella friendships che avrà due chiavi esterne verso users, quindi la chiave primaria sarà una chiave primaria composta.



users(<u>username</u>, email, first_name, last_name)
friendships(<u>username_1, unsername_2</u>, state_request)
FK: username_1 verso tabella users
FK: username_2 verso tabella users

3.1.2 users — user_groups

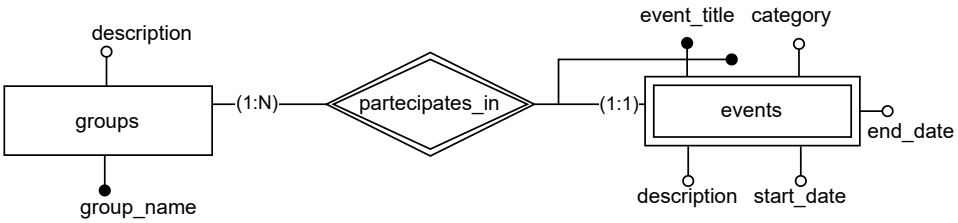
Tra users e user_groups abbiamo una relazione di cardinalità N:N, dove un gruppo per esistere deve avere almeno un utente, ma un utente può anche non appartenere a nessun gruppo. In questa occasione si andrà a creare una tabella di relazione **group_members** che avrà come chiave primaria i riferimenti alle due entità coinvolte. Inoltre la tebella user_groups verrà ridenominata **groups** per evitare confusione con group_members.



group_members(<u>username, group_name</u>, join_date, role)
FK: username verso users
FK: group_name verso tabella user_groups

3.1.3 user_groups — events

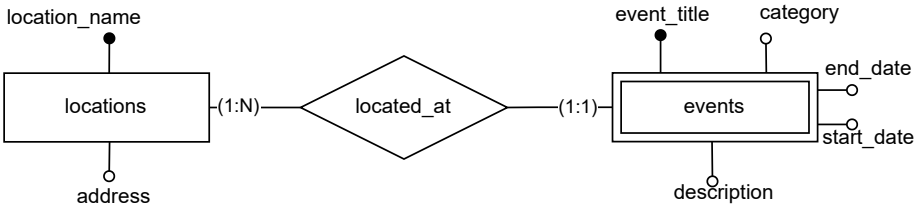
In questo caso abbiamo una relazione di cardinalità N:1, inoltre l'entità events è un'entità debole perché non può essere identificata da sola. Per esistere serve sapere anche il nome del gruppo che lo ha organizzato. Quindi aggiungeremo ad events un riferimento a user_groups.



user_groups(group_name, description)
events(event_title, group_name, category, description, start_date, end_date)
FK: group_name verso tabella user_groups

3.1.4 locations — events

Tra locations e events esiste una relazione di cardinalità N:1. Quindi nella traduzione andremo ad aggiungere una chiave esterna a events che fa riferimento a locations.

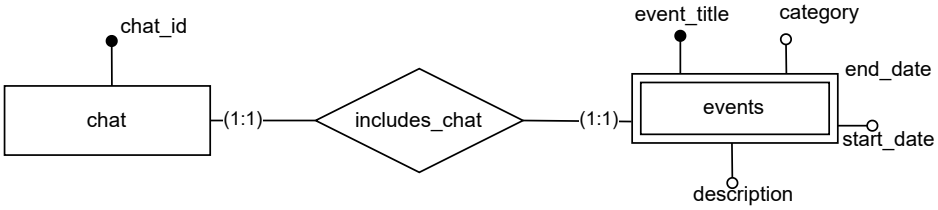


events((event_title, group_name) , location_name, description, start_date, end_date, category)
FK: location_name verso tabella locations.

Nell'entità locations abbiamo un attributo composto address, formato da city, street e street_number. Per gestirlo invece di creare un'altra tabella dividiamo l'attributo e lo inseriamo nella tabella locations.
locations(location_name, city, street, stree_number);

3.1.5 events — chat

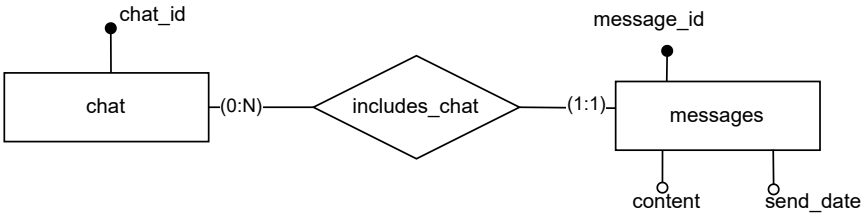
In questo caso abbiamo una relazione di cardinalità 1:1, quindi andremo ad aggiungere solamente un riferimento esterno all'entità chat.



chat(chat_id, (event_title, group_name))
FK: (event_title, group_name) verso tabella events

3.1.6 chat — messages

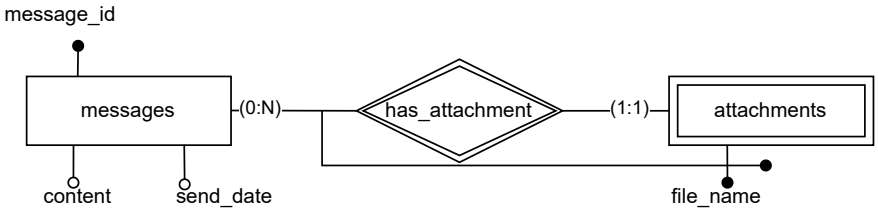
Tra queste due entità esiste una relazione di cardinalità N:1, quindi andremo ad aggiungere un riferimento esterno a messages.



messages(<u>message_id</u>, chat_id, content, send_date);
FK: chat_id verso tabella chat

3.1.7 messages — attachments

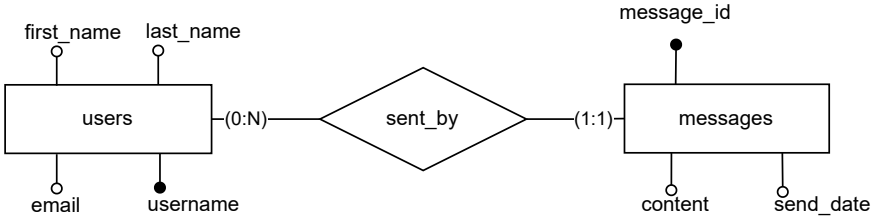
Tra l'entità messages e attachments abbiamo una relazione di cardinalità N:1, dove un messaggio può anche non contenere allegati. Andremo ad aggiungere un riferimento esterno ad attachments. Ricordiamo che attachments è un entità debole, quindi la sua chiave primaria sarà una chiave composta.



attachments(<u>file_name, message_id</u>);
FK: message_id verso tabella messages

3.1.8 users — messages

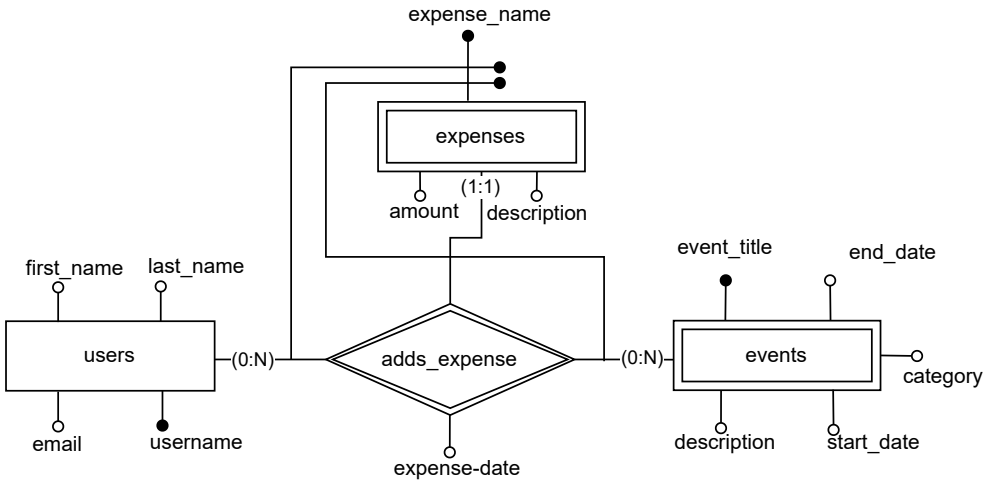
Tra l'entità users e messages esiste una relazione di cardinalità N:1. Andremo ad aggiungere alla tabella messages un riferimento esterno per users. Oltre al riferimento alla tabella chat, messages avrà anche il riferimento ad users.



messages(<u>message_id</u>, content, send_date, chat_id, username)
FK: username verso tabella users
FK: chat_id verso tabella chat

3.1.9 users — events — expenses

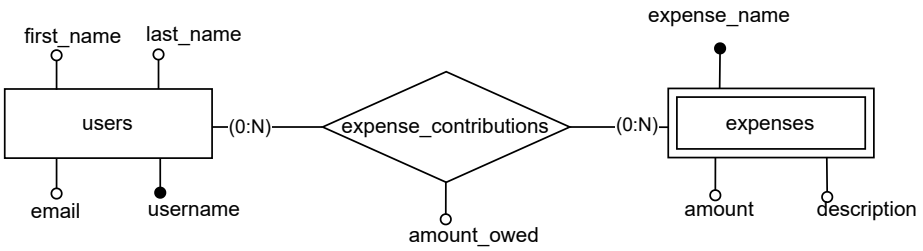
Tra queste tre entità esiste una relazione ternaria. È una relazione che permette di aggiungere una spesa ad un evento. La relazione ci dice che un utente può aggiungere opzionalmente una spesa ed un evento può avere opzionale una o più spese. Quindi una spesa è associata al più ad un utente e a un evento. Inoltre l'entità expenses senza una chiave composta non è identificabile. Quindi, la tabella expenses avrà una come chiave primaria composta dagli identificatori di users e events.



expenses(expense_name, payer_username, event_title, group_name, amount, description, expense_date)
FK: payer_username verso tabella users (utente che paga una spesa)
FK: (event_title, group_name) verso tabella events

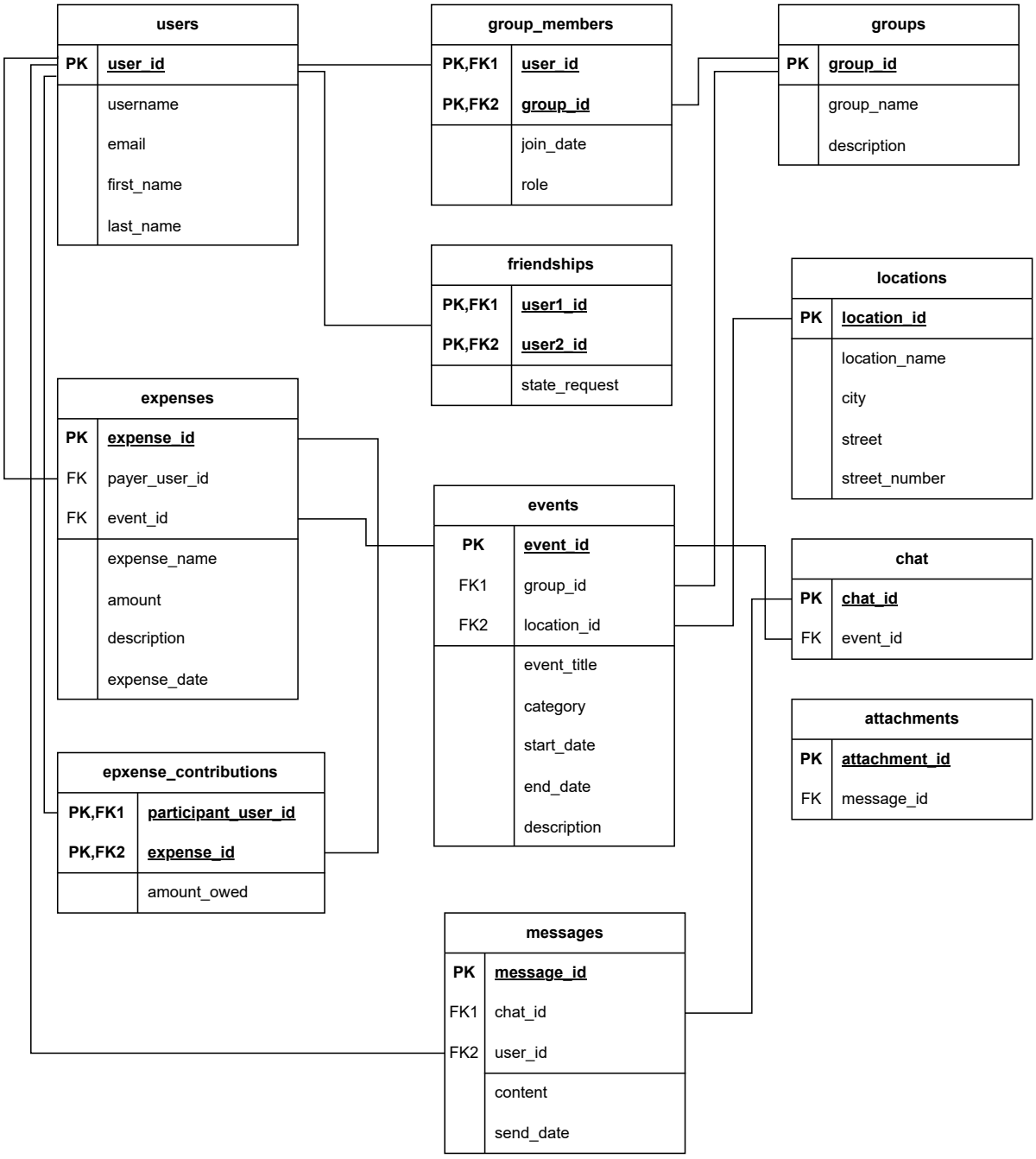
3.1.10 users — expenses

Queste due entità hanno una cardinalità N:N, quindi si creerà una nuova tabella **expense_contributions** che avrà come chiave primaria i riferimenti a users e expenses.



expense-contributions(participant_username, expense_name, event_title, payer_username, amount_owed)
FK: participant_username verso tabella users (utente che paga la sua quota contribuendo ad una spesa)
FK: (expense_name, event_title, payer_username) verso tabella expenses

3.2 Schema ristrutturato



4 Progettazione fisica

Dopo aver completato la progettazione logica, possiamo concentrarci sull'implementazione vera e propria. Procederemo nell'aggiunta di chiavi surrogate (ID fittizi) per ogni tabelle per migliorare le prestazioni delle operazioni di query e join, semplificare la gestione delle relazioni e garantire una maggiore flessibilità in caso di modifiche ai dati.

4.1 Definizione delle tabelle

Adesso ridefiniamo ogni tabella con i propri attributi e tipo di dati.

4.1.1 users

Attributo	Vincolo	Tipo di dato
user_id	AUTO_INCREMENT, NOT NULL, PK	INT
username	NOT NULL, UNIQUE	VARCHAR(100)
email	NOT NULL, UNIQUE	VARCHAR(100)
first_name	NOT NULL	VARCHAR(100)
last_name	NOT NULL	VARCHAR(100)
created_at	NOT NULL	DATETIME
update_at	NOT NULL	DATETIME

Tabella 13: Tabella users

4.1.2 friendships

Attributo	Vincolo	Tipo di dato
user1_id	NOT NULL, PK, FK REFERENCES users(user_id)	INT
user2_id	NOT NULL, PK, FK REFERENCES users(user_id)	INT
state_request	NOT NULL	ENUM(pending, accepted, denied)

Tabella 14: Tabella friendships

4.1.3 groups

Attributo	Vincolo	Tipo di dato
group_id	AUTO_INCREMENT, NOT NULL, PK	INT
group_name	NOT NULL	VARCHAR(100)
description	NULL	VARCHAR(255)
creation_date	NOT NULL	DATETIME

Tabella 15: Tabella groups

4.1.4 group_members

Attributo	Vincolo	Tipo di dato
user_id	NOT NULL, PK, FK, REFERENCES users(user_id)	INT
group_id	NOT NULL, PK, FK, REFERENCES groups(group_id)	INT
join_date	NOT NULL	DATETIME
role	NOT NULL	ENUM(member, admin)

Tabella 16: Tabella group_members

4.1.5 locations

Attributo	Vincolo	Tipo di dato
location_id	AUTO_INCREMENT, NOT NULL, PK	INT
location_name	NOT NULL	VARCHAR(100)
city	NULL	VARCHAR(100)
street	NULL	VARCHAR(100)
street_number	NULL	INT

Tabella 17: Tabella locations

4.1.6 events

Attributo	Vincolo	Tipo di dato
event_id	AUTO_INCREMENT, NOT NULL, PK	INT
group_id	NOT NULL, FK, REFERENCES groups(group_id)	INT
location_id	NOT NULL, FK, REFERENCES locations(location_id)	INT
event_title	NOT NULL	VARCHAR(100)
category	NOT NULL	ENUM(food, party, holidays, relax, camping, other)
start_date	NOT NULL	DATE
end_date	NOT NULL, CHECK(end_date >= start_date)	DATE
description	NULL	VARCHAR(255)

Tabella 18: Tabella events

4.1.7 chat

Attributo	Vincolo	Tipo di dato
chat_id	AUTO_INCREMENT, NOT NULL, PK	INT
event_id	NOT NULL, FK, REFERENCES events(event_id)	INT

Tabella 19: Tabella chat

4.1.8 messages

Attributo	Vincolo	Tipo di dato
message_id	AUTO_INCREMENT, NOT NULL, PK	INT
user_id	NOT NULL, FK, REFERENCES users(user_id)	INT
chat_id	NOT NULL, FK, REFERENCES chat(chat_id)	INT

Tabella 20: Tabella messages

4.1.9 attachments

Attributo	Vincolo	Tipo di dato
attachment_id	AUTO_INCREMENT, NOT NULL, PK	INT
message_id	NOT NULL, FK, REFERENCES messages(message_id)	INT
file_name	NOT NULL	VARCHAR(255)
file_size	NOT NULL	BIGINT
file_type	NOT NULL	VARCHAR(255)

Tabella 21: Tabella attachments

4.1.10 expenses

Attributo	Vincolo	Tipo di dato
expense_id	AUTO_INCREMENT, NOT NULL, PK	INT
payer_id	NOT NULL, FK, REFERENCES users(user_id)	INT
event_id	NOT NULL, FK, REFERENCES events(event_id)	INT
amount	NOT NULL, CHECK(amount > 0)	DECIMAL
expense_name	NOT NULL	VARCHAR(100)
description	NULL	VARCHAR(255)
expense_date	NOT NULL	DATETIME

Tabella 22: Tabella expenses

4.1.11 expense_contributions

Attributo	Vincolo	Tipo di dato
participant_id	NOT NULL, PK, FK, REFERENCES users(user_id)	INT
expense_id	NOT NULL, PK, FK, REFERENCES expense(expense_id)	INT
amount_owed	NOT NULL, CHECK(amount_owed >= 0)	DECIMAL

Tabella 23: Tabella expense_contributions