Spot – Documento Progetto

Enrico Caminiti

Introduzione

Spot è una piattaforma web basata sulla geolocalizzazione che consente agli utenti di creare e condividere "spot" (luoghi di interesse), organizzare eventi collegati agli spot e interagire tramite commenti, follow e inviti. Il progetto è stato realizzato con l'obiettivo di mettere in pratica le competenze acquisite nello sviluppo di applicazioni web full-stack.

Architettura

Spot è strutturato secondo un'architettura **client–server**, in cui:

- Il **client** (frontend) è responsabile della presentazione e dell'interazione utente, sviluppato (n.r. "da sviluppare") con Angular.
- Il **server** (backend) espone un'API sviluppata in Spring Boot responsabile della gestione di dati, della logica business e dell' accesso al database PostgreSQL.
- Il **database** utilizza l'estensione PostGIS per supportare la geolocalizzazione avanzata, consentendo l'esecuzione efficiente di query spaziali.

Il protocollo HTTP viene utilizzato per le comunicazioni tra client e server, seguendo lo stile *RESTful* e utilizzando il formato *JSON* per la trasmissione dei dati.

Entità

Di seguito sono descritte le principali entità del sistema e i relativi attributi.

User – rappresenta un account registrato dell'applicazione. Comprende le credenziali di autenticazione (username, email, password) e mantiene riferimenti alle altre entità con cui interagisce: gli spot creati, gli eventi creati, i commenti scritti, la lista di utenti seguiti (following) e di follower, nonché gli spot/eventi che segue o a cui è invitato.

Campi principali:

Campo	Tipo	Descrizione
`id`	Integer (PK)	Identificativo univoco dell'utente.
`username`	String	Nome utente.
`email`	String	Indirizzo email univoco.
`password`	String	Password dell'account.
`spots`	Set<Spot>	Spot creati dall'utente (OneToMany).
`events`	Set<Event>	Eventi creati dall'utente (OneToMany).
`comments`	Set<Comment>	Commenti scritti dall'utente (OneToMany).
`followers`	Set<User>	Utenti che seguono questo utente (ManyToMany auto-relazione).
`following`	Set<User>	Utenti seguiti da questo utente (ManyToMany auto-relazione).
`invitedSpots`	Set<Spot>	Spot in cui l'utente è invitato (ManyToMany).
`invitedEvents`	Set<Event>	Eventi a cui l'utente è invitato (ManyToMany).
`followedSpots`	Set<Spot>	Spot seguiti dall'utente (ManyToMany).
`followedEvents`	Set<Event>	Eventi seguiti dall'utente (ManyToMany).

Spot – è un punto di interesse georeferenziato creato da un utente. Ogni spot appartiene a un proprietario (owner) e può contenere più eventi. Gli attributi principali includono il nome, la descrizione e le coordinate geografiche (position, di tipo Point offerto da PostGIS), oltre alla visibilità (privacy) che può assumere tre valori: **PUBLIC**, **FRIEND_ONLY** (visibile solo agli "amici" del proprietario) o **INVITE_ONLY** (visibile esclusivamente agli utenti invitati). Uno spot può

essere seguito da altri utenti e, se la privacy è *INVITE_ONLY*, possiede anche una lista di utenti invitati che possono visualizzarlo.

Campi principali:

Campo	Tipo	Descrizione
`id`	Integer (PK)	Identificativo univoco dello spot.
`name`	String	Nome dello spot.
`description`	String	Descrizione dello spot.
`position`	Point (geografia)	Coordinate geografiche.
`privacy`	Privacy (enum)	Visibilità dello spot: PUBLIC, FRIEND_ONLY, INVITE_ONLY.
`owner`	User (FK)	Proprietario/creatore dello spot.
`followers`	Set<User>	Utenti che seguono lo spot (ManyToMany).
`invitedUsers`	Set<User>	Utenti invitati allo spot (ManyToMany, rilevante se privacy=INVITE_ONLY).
`events`	Set<Event>	Eventi associati allo spot (OneToMany).

Event – rappresenta un'attività programmata in uno spot. È caratterizzato da un titolo, una descrizione, una data/ora (formato ISO 8601) e un livello di privacy. L'evento è legato a un unico spot (il luogo in cui si svolgerà) e a un creatore (il proprietario dello spot). È possibile invitare altri utenti a partecipare a un evento con privacy *INVITE_ONLY*, mentre per eventi PUBLIC o FRIEND_ONLY gli utenti possono seguirli senza un invito esplicito. Gli eventi supportano l'interazione tramite commenti e generano notifiche sia per gli invitati sia per chi li segue.

Campi principali:

Campo	Tipo	Descrizione
`id`	Integer (PK)	Identificativo univoco dell'evento.
`title`	String	Titolo dell'evento.
`description`	String	Descrizione dell'evento.
`date`	LocalDateTime	Data e ora dell'evento (formato ISO 8601).
`privacy`	Privacy (enum)	Visibilità: PUBLIC, FRIEND_ONLY, INVITE_ONLY.
`owner`	User (FK)	Creatore dell'evento.
`spot`	Spot (FK)	Spot in cui si tiene l'evento.
`comments`	Set<Comment>	Commenti associati all'evento (OneToMany).
`followers`	Set<User>	Utenti che seguono l'evento (ManyToMany).
`invitedUsers`	Set<User>	Utenti invitati a partecipare all'evento (ManyToMany).

Comment – è un messaggio che un utente lascia su un evento. Un commento contiene il testo del messaggio, la data (timestamp ISO) in cui è stato pubblicato e un flag booleano `edited` che indica se è stato modificato dopo la pubblicazione. Ogni commento è associato a un autore (owner) e a un evento specifico.

Campi principali:

Campo	Tipo	Descrizione
`id`	Integer (PK)	Identificativo univoco del commento.
`text`	String	Testo del commento.
`date`	LocalDateTime	Data e ora di creazione del commento.
`edited`	boolean	Indica se il commento è stato modificato dopo la creazione.
`owner`	User (FK)	Autore del commento.
`event`	Event (FK)	Evento a cui il commento si riferisce.

Notification – informa un utente su un determinato evento di interesse (ad esempio, l'invito a uno spot o a un evento, un nuovo follower, la conferma di partecipazione a un evento o un nuovo commento). Una notifica contiene l'identificativo del destinatario (user_id), il tipo di notifica (type), un messaggio testuale descrittivo, la data/ora di creazione (createdAt) e un flag booleano `read` che indica se è stata visualizzata dall'utente.

Campi principali:

Campo	Tipo	Descrizione
`id`	Integer (PK)	Identificativo univoco della notifica.
`userId`	Integer (FK verso User)	Identificativo dell'utente destinatario della notifica.
`type`	NotificationType (enum)	Tipo di notifica (ad es. INVITE, FOLLOW, COMMENT...).
`message`	String	Messaggio testuale della notifica.
`createdAt`	LocalDateTime	Timestamp di creazione (impostato automaticamente).
`read`	boolean	Flag che indica se la notifica è stata letta (TRUE) o meno.

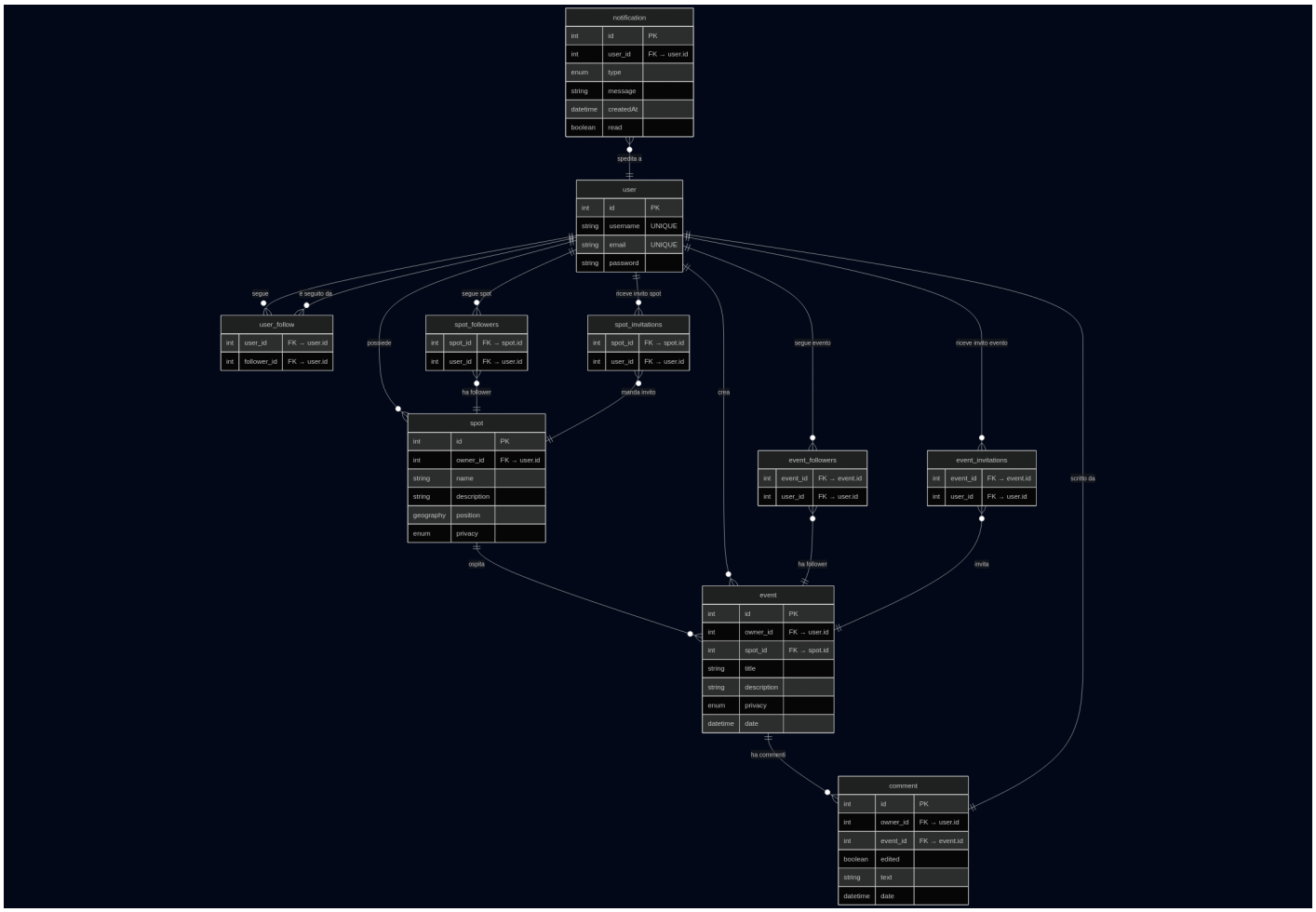

Figure 1: Schema erm del database

Funzionalità e API REST

Le principali funzionalità della piattaforma sono esposte tramite una API REST, organizzata in controller Spring annotati con `@RestController`. Ogni gruppo di endpoint si occupa di una specifica area logica (autenticazione, gestione utenti, spot, eventi, commenti, feed). Per tutte le richieste che richiedono autenticazione è necessario includere nell'header HTTP il token di sessione (`X-Session-Token`) ottenuto al momento del login o della registrazione.

Di seguito si fornisce una panoramica ragionata degli endpoint, suddivisa per area tematica, con descrizione delle operazioni disponibili. Si noti che nell'applicazione due utenti sono considerati **amici** se entrambi si seguono reciprocamente (follow reciproco). Un utente è considerato **amico** con se stesso.

Autenticazione

Questi endpoint gestiscono la registrazione e l'autenticazione degli utenti. In caso di successo, restituiscono un token di sessione da utilizzare per le successive chiamate protette.

Endpoint	Metodo	Descrizione	Note
`/login`	POST	Autentica un utente esistente e genera un nuovo token di sessione.	Body JSON: `username`, `password`. Ritorna il token di sessione oppure errore 401 (Unauthorized) se credenziali non valide.
`/register`	POST	Registra un nuovo account utente e restituisce il token di sessione.	Body JSON: `username`, `email`, `password`. Errore 409 (Conflict) se lo username o l'email sono già in uso.

Dettagli aggiuntivi: tutti i campi sopra indicati sono obbligatori (una richiesta con campi mancanti restituisce errore 400 – Bad Request).

Gestione degli utenti

La piattaforma consente di visualizzare il proprio profilo e quello di altri utenti, di cercare utenti per nome e di gestire la funzionalità di follow/unfollow. In particolare, un utente può seguire altri utenti per vedere i loro spot ed eventi pubblici; se due utenti si seguono a vicenda, diventano "amici" e possono vedere anche i contenuti con privacy *FRIEND_ONLY*. Ogni call è preceduta dal `base url segment` riportato a parte per motivi di spazio. Eg. `/profile` è in realtà `/users/profile`.

Base url segment : /users

Endpoint	Metodo	Descrizione	Note
`/profile`	GET	Restituisce il profilo dettagliato dell'utente loggato.	
`/{userId}`	GET	Recupera informazioni pubbliche sul profilo di un altro utente.	Non include dati sensibili.
`/{userId}/follow`	GET	Esegue il follow dell'utente specificato.	Genera una notifica all'utente seguito.
`/{userId}/unfollow`	GET	Interrompe il follow dell'utente specificato.	
`/search?query=...`	GET	Cerca utenti in base alla query fornita.	Supporta parametri di paginazione (es. `page`, `size`).
`/friends`	GET	Elenca gli **amici** reciproci dell'utente loggato.	Vengono mostrati solo gli utenti **amici**.
`/{userId}/events`	GET	Elenca gli eventi creati dall'utente amico specificato.	Accessibile solo agli utenti tra loro **amici**.
`/{userId}/spots`	GET	Elenca gli spot creati (o seguiti) dall'utente specificato.	Accessibile solo agli utenti tra loro **amici**.
`/{userId}/followers`	GET	Restituisce la lista dei follower di un dato utente.	Accessibile solo agli utenti tra loro **amici**.

Gestione degli Spot

Questa sezione di API permette di cercare spot in base alla posizione geografica, nonché di creare e gestire gli spot (inclusa la lista di utenti follower o invitati). Un utente autenticato può creare nuovi spot, modificarli o eliminarli, e seguire gli spot di altri utenti per ricevere aggiornamenti sui relativi eventi. In queste query vengono inclusi solo spot con privacy PUBLIC o FRIEND_ONLY/INVITE_ONLY se l'utente ha diritto di vederli (amico o invitato). Ogni call è preceduta dal `base url segment` riportato a parte per motivi di spazio. Eg. `/profile` è in realtà `/users/profile`.

Base url segment : /spots

Endpoint	Metodo	Descrizione	Note
`?lat=&lng=&meters=`	GET	Restituisce l'elenco degli spot visibili entro un certo raggio dalla posizione specificata.	Parametri query: `lat` e `lng` per le coordinate geografiche, `meters` (opzionale) per il raggio in metri.
`/nearest?lat= &lng=&meters=`	GET	Restituisce lo spot visibile più vicino alla posizione specificata.	Stessi parametri (`lat`, `lng`, `meters`).

Endpoint	Metodo	Descrizione	Note
`/{spotId}`	GET	Recupera i dettagli dello spot identificato da `{spotId}`.	Lo spot deve essere visibile al richiedente.
`/`	POST	Crea un nuovo spot con i dati specificati.	Body JSON: `name`, `lat`, `lng`, `description` (opzionale), `privacy`. Il nuovo spot sarà associato all'utente loggato come owner.
`/{spotId}`	PATCH	Aggiorna i campi di uno spot esistente.	Solo il proprietario (owner) dello spot può modificarlo.
`/{spotId}`	DELETE	Elimina lo spot specificato e tutti gli eventi ad esso associati.	Solo il proprietario può eliminarlo.
`/{spotId}/follow`	GET	L'utente loggato inizia a seguire lo spot.	Errore se lo spot era già seguito dal chiamante.
`/{spotId}/unfollow`	GET	Smette di seguire lo spot indicato.	Errore se il chiamante non seguiva lo spot.
`/{spotId}/followers`	GET	Elenca gli utenti che seguono lo spot indicato.	
`/{spotId}/invited`	GET	Elenca gli utenti invitati allo spot indicato (solo per spot privati).	Accessibile solo al proprietario e agli utenti invitati.

Gestione degli Eventi

Gli endpoint dedicati agli eventi permettono di creare nuovi eventi all'interno di uno spot, di visualizzare gli eventi esistenti (filtrando in base ai permessi), nonché di modificarli, eliminarli e gestire le partecipazioni (follow e inviti). La visibilità di ciascun evento dipende dalla privacy impostata e dal rapporto dell'utente chiamante con l'owner dello spot. Ogni call è preceduta dal `base url segment` riportato a parte per motivi di spazio. Eg. `/profile` è in realtà `/users/profile`.

Base url segment : /events

Endpoint	Metodo	Descrizione	Note
`/spot/{spotId}`	GET	Elenca tutti gli eventi visibili all'interno dello spot specificato.	Gli eventi restituiti dipendono dal ruolo del chiamante e dalla privacy dell'evento.
`/`	POST	Crea un nuovo evento associato a uno spot esistente.	Body JSON: `title`, `description` (opzionale), `date` (ISO string), `privacy`, `spotId`.
`/{eventId}`	GET	Recupera i dettagli di un evento specifico.	L'evento deve essere visibile al chiamante.
`/{eventId}`	PATCH	Aggiorna i dettagli di un evento.	Solo il creatore (owner) dell'evento può modificarlo.
`/{eventId}`	DELETE	Elimina l'evento specificato.	Solo il creatore dell'evento o il proprietario dello spot può eliminarlo.
`/{eventId}/follow`	GET	Inizia a seguire l'evento specificato.	Errore se l'utente seguiva già l'evento.
`/{eventId}/unfollow`	GET	Smette di seguire l'evento specificato.	Errore se l'utente non stava seguendo l'evento.
`/{eventId}/followers`	GET	Restituisce la lista degli utenti che seguono l'evento.	Restituisce i followers dell'evento.
`/{eventId}/invited`	GET	Restituisce la lista degli utenti invitati all'evento.	Accessibile solo al creatore e agli utenti invitati stessi.

Gestione dei Commenti

La piattaforma consente agli utenti autenticati di aggiungere commenti agli eventi e di gestirli. Ogni commento è associato a un evento specifico e può essere modificato o rimosso solo dal suo autore. I commenti più recenti di un evento possono essere recuperati tramite paginazione. Ogni call è preceduta dal `base url segment` riportato a parte per motivi di spazio. Eg. `/profile` è in realtà `/users/profile`.

Base url segment : /comments

Endpoint	Metodo	Descrizione	Note
`/`	POST	Aggiunge un nuovo commento a un evento.	Body JSON: `text` (testo del commento), `date` (timestamp ISO di creazione), `eventId`. L'utente loggato diventerà l'autore.
`/event/{eventId}`	GET	Recupera la lista dei commenti associati a un determinato evento.	Supporta parametri di paginazione (es. `page` e `size`). Errore se l'evento non è visibile al chiamante.
`/{commentId}`	GET	Ottiene i dettagli di un singolo commento (testo, autore, data, ecc.).	Errore se il commento non esiste o non è accessibile.
`/{commentId}`	PATCH	Modifica il testo (o altri campi editabili) di un commento esistente.	Solo l'autore può modificarlo. Dopo la modifica, viene impostato `edited=true`.
`/{commentId}`	DELETE	Elimina un commento esistente.	Solo l'autore può eliminarlo.

Feed e notifiche

La sezione *Feed* fornisce all'utente autenticato un elenco aggregato di aggiornamenti riguardanti gli spot e gli eventi di suo interesse (spot/eventi creati dall'utente, quelli che segue o in cui è stato invitato). La sezione *Notifiche* permette di recuperare le notifiche non lette e di segnarle come lette. Ogni call è preceduta dal `base url segment` riportato a parte per motivi di spazio. Eg. `/profile` è in realtà `/users/profile`.

Base url segment : /feed

Endpoint	Metodo	Descrizione	Note
`/`	GET	Restituisce un feed di spot ed eventi rilevanti per l'utente loggato.	Include eventi creati, seguiti o in cui è invitato, in ordine cronologico.
`/notifications`	GET	Restituisce l'elenco delle notifiche non ancora lette dell'utente.	Mostra solo notifiche destinate all'utente loggato con `read=false`.
`/notifications/read`	GET	Segna come lette tutte le notifiche correnti dell'utente.	Dopo la chiamata, tutte le notifiche dell'utente saranno marcate come `read=true`.

Collaudo con Postman

Una raccolta **Postman** (in formato JSON) è stata predisposta per testare l'intero ciclo di vita dell'applicazione: dalla registrazione e autenticazione, fino alla creazione e gestione di spot, eventi, commenti, inviti e follow. Dopo il login o la registrazione, il token di sessione restituito viene salvato automaticamente in una variabile di ambiente (`token`) della raccolta e incluso come intestazione in tutte le richieste successive (`X-Session-Token`).

La raccolta organizza gli endpoint in scenari realistici per facilitare il collaudo e l'intera suite di test può essere eseguita in locale puntando a un server in esecuzione su `http://localhost:8080`. La variabile `baseUrl` nella raccolta Postman è configurata di default su questo indirizzo per semplificare l'esecuzione di tutte le chiamate senza dover modificare manualmente ogni URL.