SISTEMI WEB E BASI DI DATI

TENNIS BOOKING

Venditto Francesco - A13002603
francesco.venditto@studenti.unicampania.it

TENNIS BOOKING

SISTEMI WEB E BASI DI DATI

Università degli Studi della Campania *Luigi Vanvitelli*

INDICE

INTRODUZIONE

Obiettivo del documento

Nel seguente documento vengono descritte nel dettaglio le specifiche per lo sviluppo dell'applicativo software "Tennis Booking", una piattaforma che ha lo scopo di permettere ai suoi utenti di prenotare campi da tennis e partecipare a tornei creati dal proprietario dei campi.

Obiettivo del progetto

Il progetto mira a sviluppare una piattaforma che permetta agli utenti di prenotare campi da tennis e partecipare a tornei in modo rapido e intuitivo. Grazie a un'interfaccia user-friendly, gli utenti possono registrarsi, visualizzare la disponibilità dei campi, effettuare prenotazioni, gestire le proprie attività e iscriversi a tornei. Il sistema è progettato per migliorare l'efficienza della gestione delle prenotazioni e ridurre i tempi di attesa.

DESCRIZIONE GENERALE

Funzionalità del progetto

Il sistema deve consentire all'utente tramite interfaccia web di:

1. Registrarsi ed effettuare il login
2. Visualizzare i campi e gli orari disponibili
3. Prenotare un campo
4. Annullare le prenotazione
5. Visualizzare e iscriversi ai tornei attivi
6. Gestire il proprio profilo e cronologia prenotazioni

Il sistema deve consentire al proprietario, oltre a quelle elencate precedentemente, di creare tornei e, all'occorrenza, annullarli.

Classi di utenti

La tipologia di utenti che utilizzano il servizio non è unica. Esistono due categorie di utenti che possono utilizzare il servizio: **proprietari** e **clienti**. Sebbene un proprietario possa anche agire come cliente, non è possibile che un cliente assuma il ruolo di proprietario. In altre parole, il proprietario ha un accesso e funzionalità aggiuntive rispetto a un cliente, ma quest'ultimo può solo utilizzare il servizio senza avere i privilegi del proprietario.

Ambiente operativo

Il seguente sistema è sviluppato come un servizio web, tramite l'utilizzo della tecnologia PHP lato server, e dei linguaggi HTML, CSS e Javascript lato client. Il server utilizzato è apache mentre il database è mySQL. Il DBMS MySQL è gestito con DBeaver.

INTERFACCE RICHIESTE

Interfacce utente

L'interfaccia Web è intuitiva e facile da usare per i vari utenti. Viene formattata tramite HTML e CSS, e resa interattiva tramite l'utilizzo della tecnologia Javascript. Gli utenti possono facilmente navigare tra le diverse funzionalità del sistema (prenotazioni campi, visualizzazione tornei, visualizzare il profilo, etc…).

Interfacce hardware

Il sistema non richiede particolari requisiti hardware, essendo accessibile tramite browser web.

Interfacce software

Per la gestione del sistema sono stati utilizzati:

1. MySQL come DBMS;
2. DBeaver come interfaccia per il DBMS;
3. PHP per lo scripting lato server.

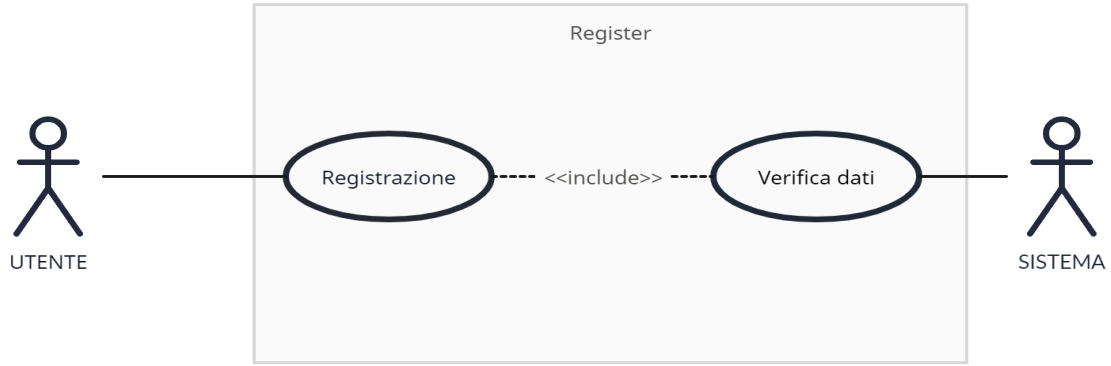
Interfacce di comunicazione

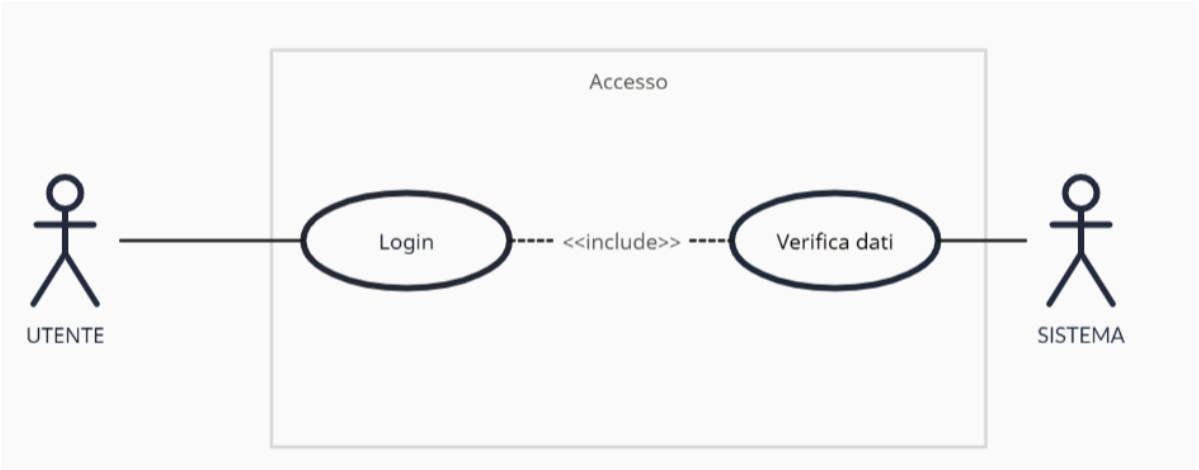
La gestione delle richieste e la loro elaborazione, così come l'interazione con gli utenti finali, è effettuata tramite l'utilizzo del protocollo HTTP. Lo scambio dei dati e le varie operazioni vengono eseguite adoperando i metodi GET, POST e SESSION di quest'ultimo. Molto spesso ho utilizzato le API JavaScript per interagire con il server tramite AJAX (Asynchronous JavaScript and XML). Questo permette di inviare richieste al server senza ricaricare la pagina.

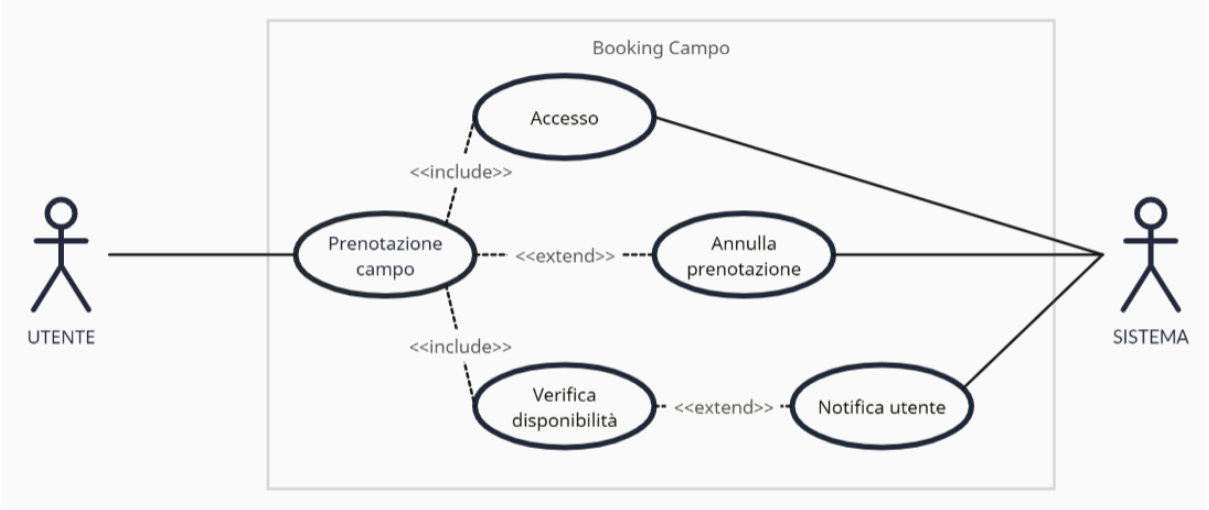
FUNZIONALITA' DEL PRODOTTO

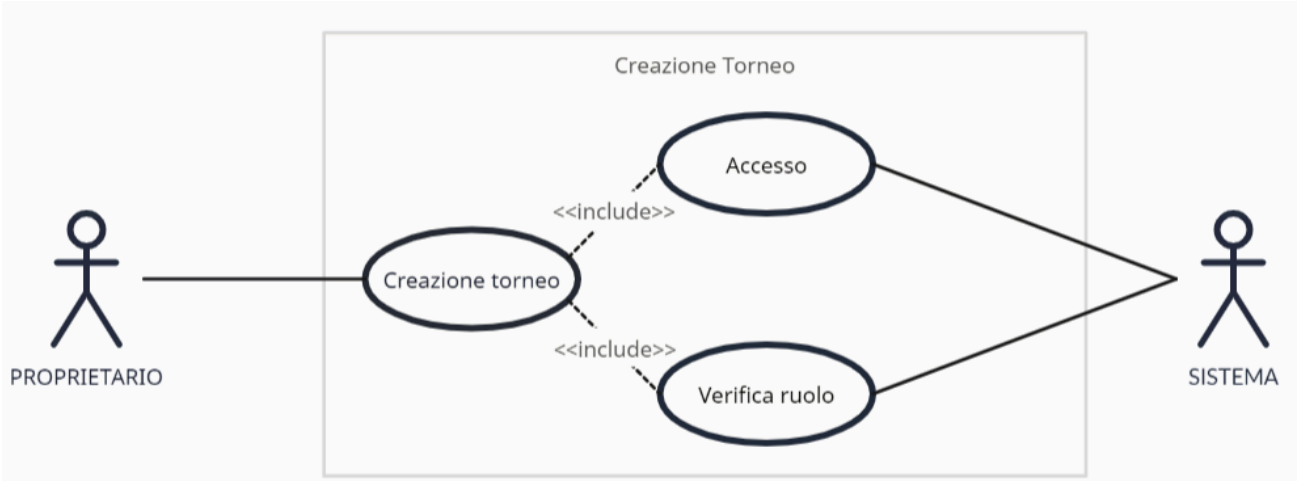
Diagramma dei casi d'uso

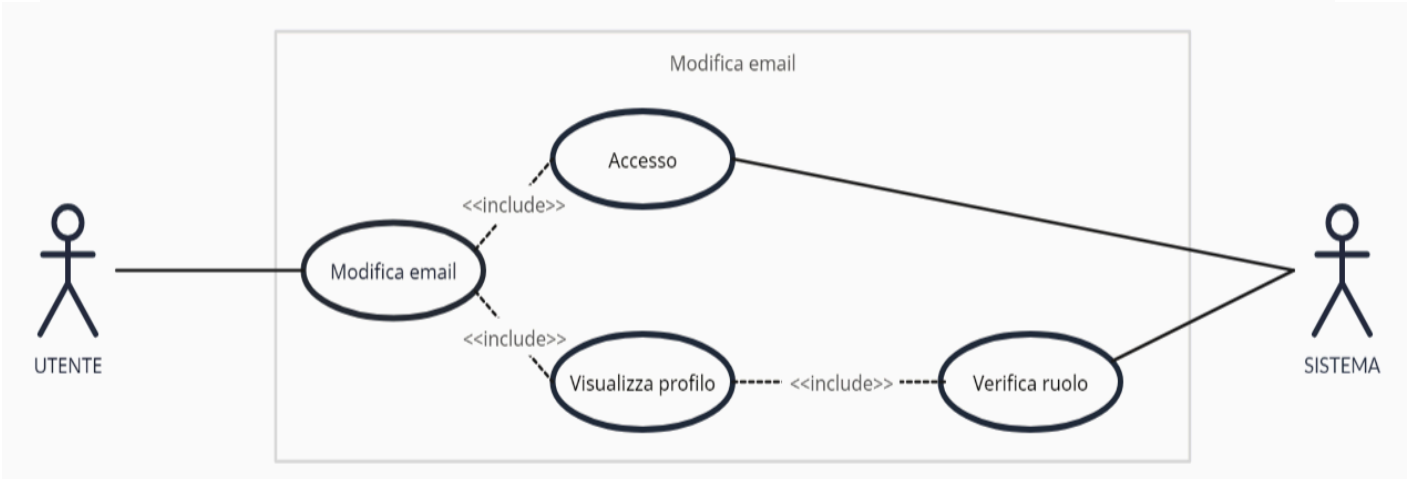
Di seguito sono illustrati i casi d'uso riguardanti le funzionalità più rilevanti del sistema.











Requisiti funzionali

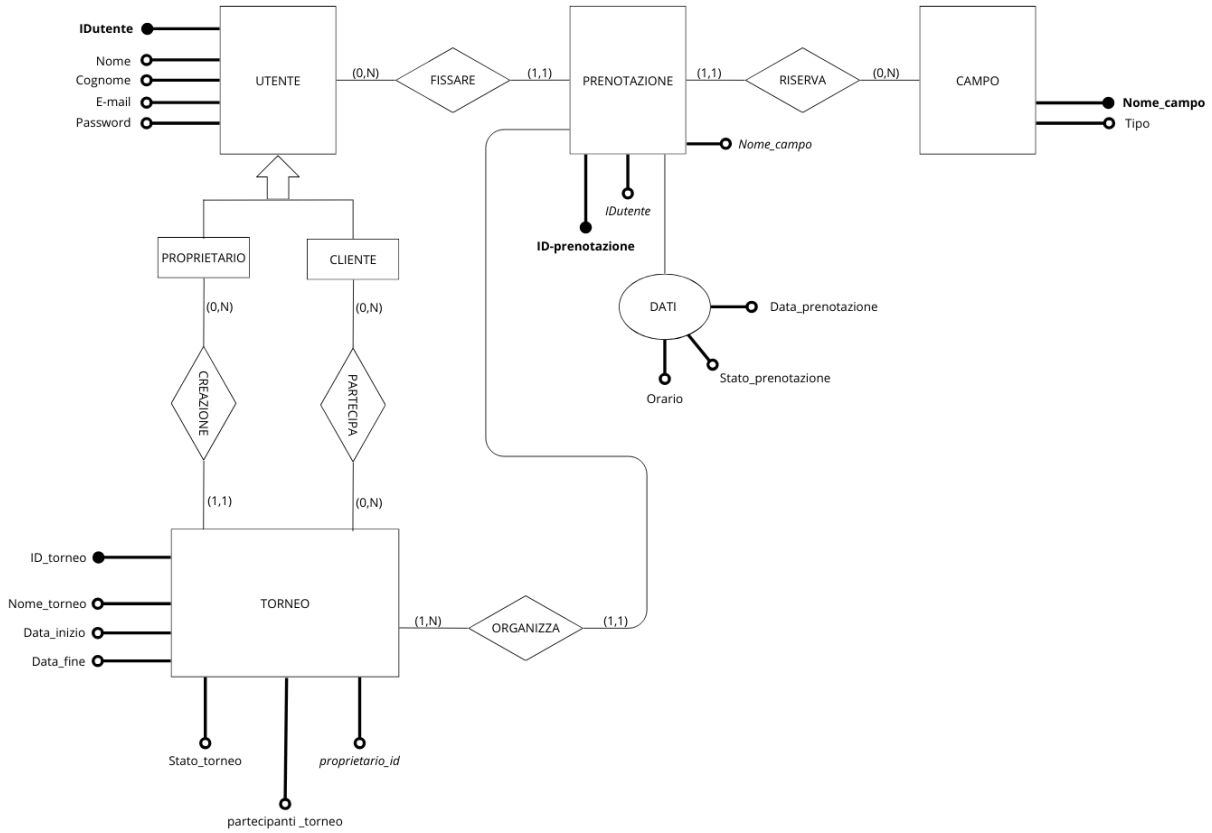
- **Registrazione**
  1. L'utente deve poter effettuare la registrazione al servizio web inserendo nome, cognome, e-mail e password.
- **Login**
  1. L'utente deve poter effettuare il login al servizio web inserendo e-mail e password.
- **Visualizzazione del profilo**
  1. L'utente ha la possibilità di visualizzare il proprio profilo con i dati anagrafici.
  2. L'utente può modificare la propria e-mail.
  3. L'utente può gestire le proprie prenotazioni, con la possibilità di annullarle se necessario.
  4. Il proprietario ha la possibilità di gestire i tornei creati, con la possibilità di annullarli se necessario.
- **Prenotare un campo**
  1. L'utente può verificare la disponibilità di un campo in terra, erba o cemento e, se disponibile, prenotarlo selezionando il giorno e l'orario desiderati.
  2. Il sistema aggiornerà automaticamente lo stato del campo e gestirà la prenotazione.

- **Visualizzazione tornei**
    1. L’utente deve poter visualizzare una lista dei tornei attualmente attivi con la possibilità di iscriversi.
    2. Il proprietario deve poter creare tornei aggiungendo informazioni rilevanti come nome del torneo, data di inizio e data data di fine.

APPENDICE

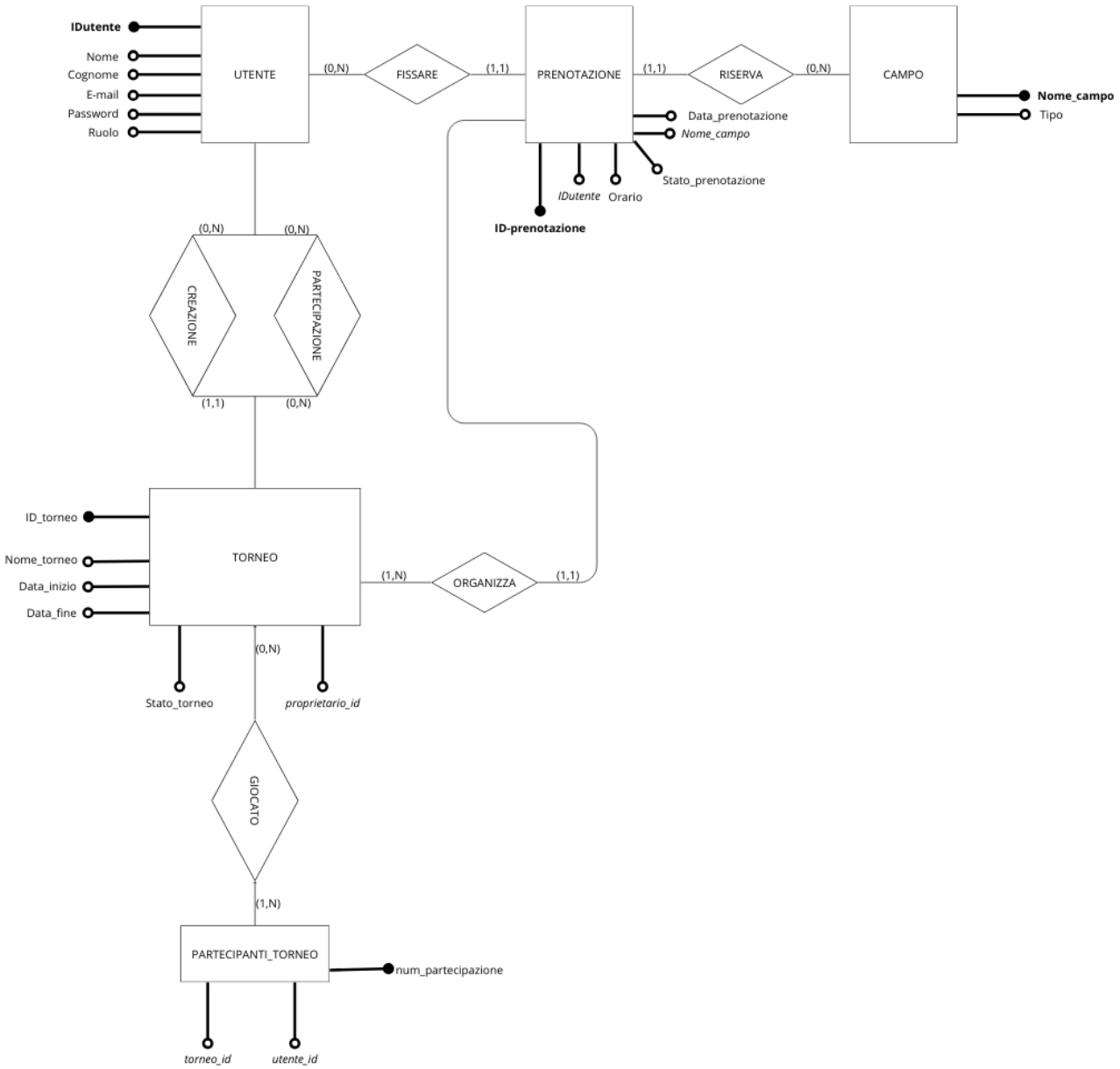
Schema concettuale

Il sistema è schematizzato concettualmente nel seguente modello E/R



Schema concettuale ristrutturato

La ristrutturazione dello schema concettuale si rende necessaria perché non tutti gli schemi E-R possono essere tradotti nel modello logico, nel nostro caso ciò che deve essere modificato sono: gli attributi di Data_prenotazione, Orario, e Stato_prenotazione, che possono essere direttamente inclusi in PRENOTAZIONI senza l'entità DATI, gli attributi composti Cliente e Proprietario che abbiamo in utente poiché viola la prima forma normale e l'attributo partecipanti_torneo che viene tradotto in una nuova entità.



Dizionario dei dati

➢ Tabella Entità

ENTITA'	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Utente	Un utente è una persona che si registra nella web app per prenotare campi da tennis e partecipare a tornei. Gli utenti possono essere suddivisi in due categorie principali: Clienti (che prenotano i campi e partecipano ai tornei) e Proprietari (che gestiscono i campi e i tornei).	IDutente, Nome, Cognome, E-Mail, Password, Ruolo	IDutente
Prenotazione	La prenotazione rappresenta un appuntamento per utilizzare un campo da tennis in una data e ora specifica. Ogni prenotazione è associata a un utente e a un campo.	ID-prenotazione, *IDutente*, *Nome_campo*, Orario, Data_prenotazione, Stato_prenotazione	ID-prenotazione
Campo	Un campo è la struttura fisica utilizzata per le partite di tennis. Ogni campo può avere diverse caratteristiche e può essere associato a un torneo.	Nome_campo, Tipo	Nome_campo
Tornei	Un torneo è un evento organizzato per competere in partite di tennis. Gli utenti possono iscriversi e partecipare ai tornei, e i proprietari possono crearli.	ID_torneo, *Proprietario_ID*, Data_inizio, Data_fine, Stato_torneo, Nome_torneo	ID_torneo
Partecipanti_torneo	Esso rappresenta la lista dei partecipanti ai vari tornei	num_partecipazione, *Torneo_id*, *Utente_id*	num_partecipazione

➢ Tabella Relazioni

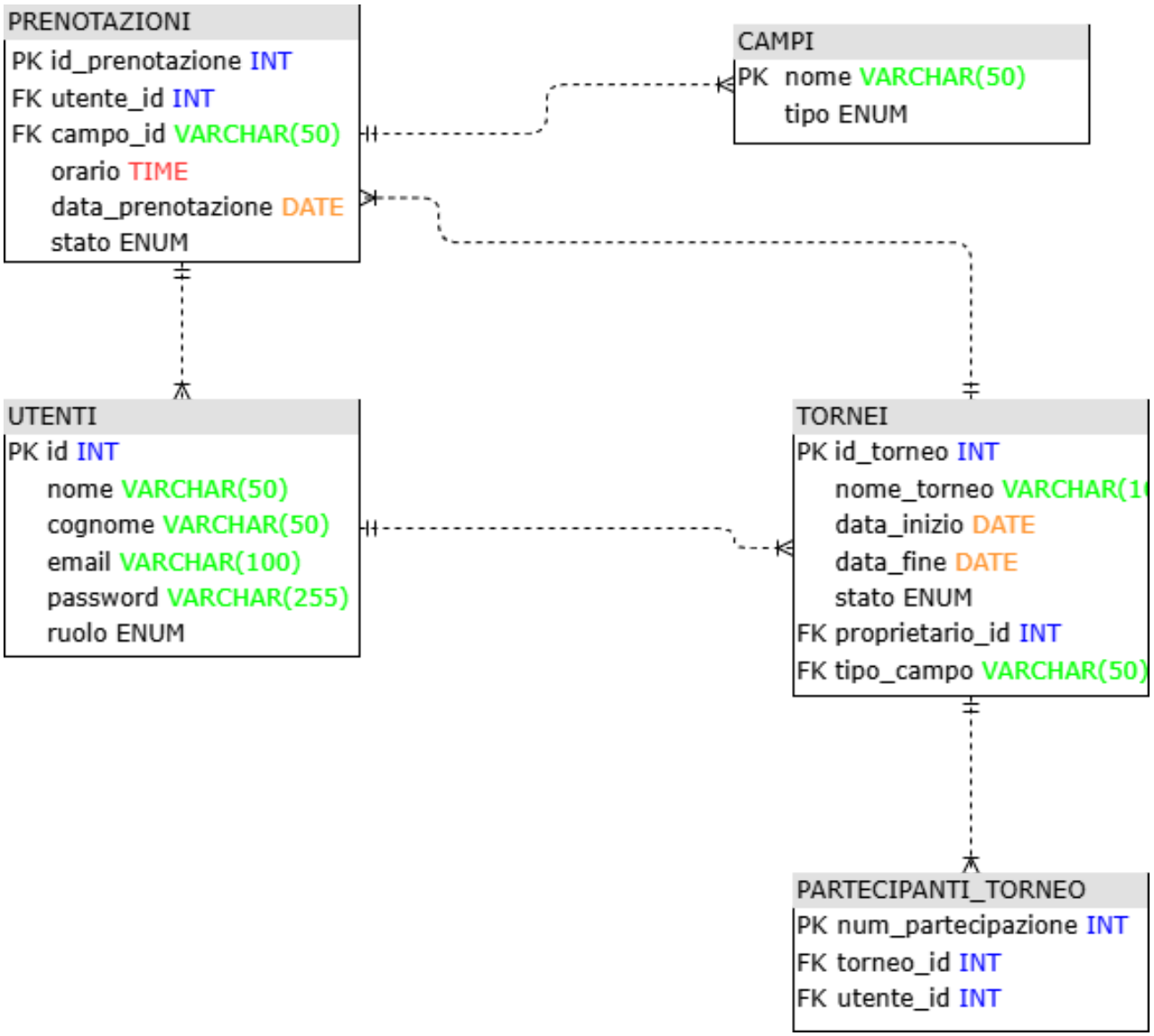
RELAZIONE	DESCRIZIONE	CARDINALITA’
Partecipazione	Legame tra Utente e Tornei, indica la possibilità all’utente di partecipare ad uno o più torneo.	(0,N) ↔ (0,N) Un utente può partecipare a 0 o tanti tornei, il torneo può avere 0 o tanti partecipanti.
Fissare	Legame tra Utente e Prenotazioni, indica la possibilità all’utente di prenotare uno o più campi da gioco.	(0,N) ↔ (1,1) Un utente può effettuare 0 o tante prenotazioni, ogni prenotazione è associata esattamente a un utente.
Organizza	Legame tra Tornei e Prenotazioni, indica che, quando un torneo è stato creato dal proprietario, il sistema si occuperà di prenotare i campi nelle date selezionate.	(1,N) ↔ (1,1) Un torneo può effettuare almeno 1 o tante prenotazioni, ogni prenotazione è associata esattamente a un torneo.
Riserva	Legame tra Prenotazioni e Campi, indica che la prenotazione avrà in esclusiva un campo da gioco.	(1,1) ↔ (0,N) Una prenotazione può riservare un solo campo da gioco, ogni campo da gioco può avere 0 o tante prenotazioni.
Giocato	Legame tra Tornei e Partecipanti_torneo, indica le persone che hanno intenzione di partecipare ai tornei selezionati.	(0,N) ↔ (1,N) Un torneo può avere 0 a tanti partecipanti, ogni partecipante della lista è associato ad 1 o più tornei.
Creazione	Legame tra Utente e Tornei, indica la possibilità di creare un torneo da parte del proprietario.	(0,N) ↔ (1,1) Un proprietario può creare 0 o tanti tornei, ogni torneo può essere creato da uno e uno solo proprietario.

Schema logico

La progettazione logica del sistema ha portato allo sviluppo delle seguenti tabelle

---

Utenti (Id, nome, cognome, email, password, ruolo)
Prenotazioni (id_prenotazione, *utente_id*, *campo_id*, orario, data_prenotazione, stato)
Tornei (id_torneo, nome_torneo, datainizio, data_fine, stato, *proprietario_id*, *tipo_campo*)
Campi (nome, tipo)
Partecipanti_torneo (num_partecipazione, *utente_id*, *torneo_id*)



Schema fisico

L'installazione fisica del database è eseguita tramite il seguente codice SQL:

```
CREATE TABLE `utenti` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nome` varchar(50) NOT NULL,
 `cognome` varchar(50) NOT NULL,
 `email` varchar(100) NOT NULL,
 `password` varchar(255) NOT NULL,
 `ruolo` enum('cliente','proprietario') DEFAULT 'cliente',
 PRIMARY KEY (`id`),
```

```
 UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `prenotazioni` (
 `id_prenotazione` int(11) NOT NULL AUTO_INCREMENT,
 `utente_id` int(11) NOT NULL,
 `campo_id` varchar(50) NOT NULL,
 `orario` time DEFAULT NULL,
 `data_prenotazione` date DEFAULT NULL,
 `stato` enum('prenotata','annullata') DEFAULT 'prenotata',
 PRIMARY KEY (`id_prenotazione`),
 KEY `utente_id` (`utente_id`),
 KEY `fk_prenotazioni_campo` (`campo_id`),
 CONSTRAINT `fk_prenotazioni_campo` FOREIGN KEY (`campo_id`) REFERENCES `campi` (`nome`)
ON DELETE CASCADE,
 CONSTRAINT `utente_id` FOREIGN KEY (`utente_id`) REFERENCES `utenti` (`id`) ON DELETE
CASCADE,
) ENGINE=InnoDB AUTO_INCREMENT=143 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `tornei` (
 `id_torneo` int(11) NOT NULL AUTO_INCREMENT,
 `nome_torneo` varchar(100) NOT NULL,
 `data_inizio` date NOT NULL,
 `data_fine` date NOT NULL,
 `stato` enum('in corso','annullato') DEFAULT 'in corso',
 `proprietario_id` int(11) DEFAULT NULL,
 `tipo_campo` varchar(50) NOT NULL,
 PRIMARY KEY (`id_torneo`),
 KEY `proprietario_id` (`proprietario_id`),
 CONSTRAINT `tornei_ibfk_1` FOREIGN KEY (`proprietario_id`) REFERENCES `utenti` (`id`) ON
DELETE SET NULL
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `partecipanti_torneo` (
 `num_partecipazione` int(11) NOT NULL AUTO_INCREMENT,
 `utente_id` int(11) NOT NULL,
 `torneo_id` int(11) NOT NULL,
 PRIMARY KEY (`num_partecipazione`),
 KEY `partecipanti_torneo_ibfk_1` (`utente_id`),
 KEY `partecipanti_torneo_ibfk_2` (`torneo_id`),
 CONSTRAINT `partecipanti_torneo_ibfk_1` FOREIGN KEY (`utente_id`) REFERENCES `utenti` (`id`)
ON DELETE CASCADE,
 CONSTRAINT `partecipanti_torneo_ibfk_2` FOREIGN KEY (`torneo_id`) REFERENCES `tornei`
(`id_torneo`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
CREATE TABLE `campi` (
```

```
 `nome` varchar(50) NOT NULL,
 `tipo` enum('erba','cemento','terra') NOT NULL,
 PRIMARY KEY (`nome`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```
