SAPIENZA
UNIVERSITÀ DI ROMA

Tesi di Laurea

CAMPETTO.IT

Una piattaforma web per la prenotazione di campi sportivi

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA E AUTOMATICA

Relatore
Prof. Leonardo Querzoni

Candidato
Antonio Zecca
Matricola 2085697

Anno Accademico 2024–2025

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo della tesi e motivazioni

Questa tesi presenta Campetto.it, una piattaforma web dedicata alla ricerca e alla prenotazione di campi sportivi, con un'interfaccia centralizzata e una struttura applicativa completa. L'intero sistema è stato sviluppato utilizzando il framework Ruby on Rails, secondo un'architettura unificata che incorpora sia la logica di backend che la generazione dinamica delle interfacce utente.
L'obiettivo generale del lavoro è dimostrare come un'applicazione ben organizzata, basata su componenti coerenti e comunicativi, possa affrontare in maniera efficace un insieme di esigenze pratiche reali: la ricerca geolocalizzata dei campi sportivi disponibili, la visualizzazione ordinata dei risultati, l'inserimento e la consultazione di recensioni da parte degli utenti e la prenotazione con pagamento online.
Le motivazioni nascono dal contesto dei centri sportivi di piccole e medie dimensioni, in cui le prenotazioni avvengono ancora in gran parte tramite canali informali: telefonate, messaggi Whatsapp, scambi via Social Network o semplici fogli cartacei. Sebbene questi metodi siano immediati e diffusi, comportano una serie di limitazioni ricorrenti: da un lato, gli utenti vogliono trovare facilmente spazi disponibili vicino a sé, in orari compatibili con i propri impegni; dall'altro, i gestori dei centri sportivi hanno necessità di tradurre le proprie regole interne in disponibilità prenotabili, riducendo errori e oneri manuali.

Campetto.it interviene su questo scenario con l'intento di offrire una soluzione digitale completa e centralizzata automatizzando il ciclo completo di prenotazione: l'utente finale può consultare i campi disponibili a partire dalla propria posizione, filtrare i risultati per sport e giorno, verificare in tempo reale gli slot liberi e completare la prenotazione online in pochi passaggi. Il gestore, dal canto suo, ha la possibilità di configurare il calendario orario del proprio centro, automatizzare la generazione delle fasce prenotabili e monitorare l'andamento delle attività tramite una dashboard di gestione.

Dal punto di vista dei risultati attesi, la piattaforma si propone di digitalizzare e centralizzare un processo oggi frammentato, riducendo i tempi necessari per la prenotazione, limitando gli errori dovuti a comunicazioni informali e offrendo ai gestori una visione strutturata dell'utilizzo dei propri campi. L'obiettivo è ottenere un flusso di lavoro più efficiente, trasparente e misurabile, che possa essere facilmente esteso o adattato a contesti sportivi differenti.

La piattaforma non mira a sostituire l'intera offerta esistente, né si propone come soluzione universale. Piuttosto, rappresenta un primo esempio concreto di come sia possibile digitalizzare e standardizzare un processo frammentario, migliorando nel contempo l'efficienza per chi offre il servizio e la fruibilità per chi lo cerca.
Il lavoro presentato in questa tesi documenta passo per passo questo processo: dall'analisi del dominio e dei bisogni funzionali fino alla progettazione dell'interfaccia e all'implementazione delle singole funzionalità. L'intento non è solo tecnico, ma anche progettuale: mostrare come una soluzione ben pensata, anche in contesti apparentemente semplici, possa contribuire a risolvere inefficienze concrete attraverso l'informatica applicata.

Figura 1: Fasi attraversate dall'utente nel processo di prenotazione.

1.2 Architettura di riferimento

L'architettura applicativa adottata segue il paradigma MVC tipico di Ruby on Rails (Ruby 2.7.2) con database PostgreSQL. Questo significa che la logica di dominio è incapsulata nei model (es. *Field, Slot, Booking, Review, User/Partner/Administrator*), l'orchestrazione dei flussi applicativi avviene nei controller, e la presentazione e interazione utente è gestita tramite viste dinamiche (*view*).
L'applicazione integra in modo controllato alcuni servizi esterni essenziali: un servizio di geocodifica per tradurre indirizzi in coordinate geografiche, un gateway di pagamento (Stripe) per le transazioni online e un servizio di posta elettronica per notifiche. Questi punti di contatto con sistemi terzi introducono vincoli di affidabilità e sicurezza da rispettare, influenzando le strategie di consistenza e tracciabilità adottate. In pratica, l'architettura prevede componenti dedicati per interagire con tali servizi o controllo degli errori per garantire la robustezza dell'applicazione anche in caso di malfunzionamenti esterni.

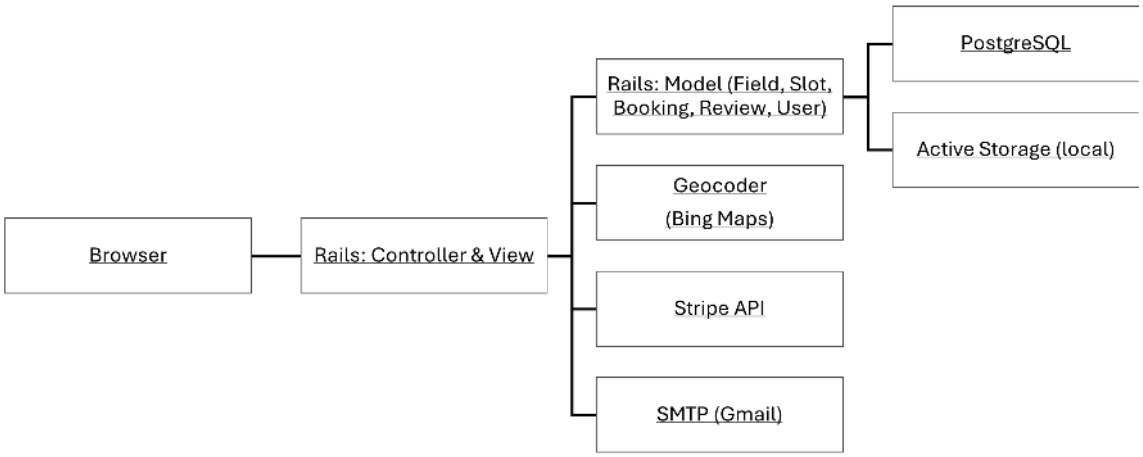


Figura 2: Architettura generale di *Campetto.it*.

1.2.1 Componenti architetturali

Per chiarezza, si riportano di seguito le principali componenti funzionali di Campetto.it, con una breve descrizione del loro inquadramento architetturale. Ciascun aspetto verrà approfondito nei capitoli successivi, sia dal punto di vista logico sia attraverso esempi di codice e schermate applicative.

- **Ricerca e geolocalizzazione**
  La ricerca dei campi è gestita dal controller `FieldsController#search`, che interroga il modello `Field` combinando sport, data e posizione. La geolocalizzazione utilizza un servizio esterno (Nominatim) incapsulato in metodi dedicati e in una rotta di supporto (`reverse_geocode`), così da mantenere nella logica di dominio la traduzione tra indirizzi e coordinate.

- **Gestione degli slot e delle disponibilità**
  Le fasce prenotabili sono rappresentate dal modello `Slot`, collegato a `Field`. La generazione degli slot avviene a partire dai parametri del centro (orari, durata, giorni), mentre controller e viste si limitano a mostrare gli slot liberi, delegando al modello il controllo sulle sovrapposizioni.

- **Sistema di prenotazione e pagamenti**
  Le prenotazioni sono modellate tramite `Booking`, che collega utente e slot.

Il flusso di pagamento è gestito da un controller dedicato (`CheckoutController`) integrato con Stripe, in modo che la prenotazione sia confermata solo dopo l'esito positivo della transazione.

- **Recensioni e valutazioni**
  Le recensioni sono gestite dal modello `Review`, associato a utente e campo. Il `ReviewsController` offre viste distinte per inserimento e consultazione, oltre alle sezioni di moderazione accessibili da partner e amministratori tramite le rispettive dashboard.

- **Segnalazioni e moderazione contenuti**
  Le segnalazioni sono rappresentate dal modello `Report`, tipicamente annidato sotto le prenotazioni. La dashboard amministrativa mostra le segnalazioni "in sospeso" e permette di aggiornarne lo stato (accettata/rifiutata).

- **Sicurezza, autenticazione e ruoli**
  L'accesso al sistema è gestito da modelli distinti (`User, Partner, Administrator`) e da percorsi dedicati di login/logout e dashboard.
  I controlli sui permessi sono implementati a livello di controller e filtrano l'accesso alle diverse aree (ricerca, gestione campi, amministrazione) in base al ruolo autenticato.

- **Layout, interfaccia e dashboard**
  L'interfaccia si appoggia a un layout unico (`application.html.erb`) condiviso da front-end e back-office, con componenti comuni (navbar, footer, gestione dei messaggi). Le dashboard di partner e amministratori riutilizzano lo stesso layout, personalizzato tramite classi di body (`partner-dashboard, admin-dashboard`) e fogli di stile dedicati.

Questi elementi costituiscono l'ossatura architetturale di Campetto.it: nei capitoli successivi ciascuna componente verrà descritta nel dettaglio, con riferimento alle scelte progettuali, al codice e alle schermate principali.

1.3 Organizzazione della tesi

Il resto del presente elaborato è strutturato come segue.
Nel **Capitolo 2** vengono raccolti e analizzati i requisiti del sistema: si descrive il dominio applicativo di Campetto.it, si identificano gli attori coinvolti (utente, partner, amministratore) e si presentano le user story dal loro punto di vista, da cui derivano i requisiti funzionali e non funzionali.

Il **Capitolo 3** illustra le metodologie di sviluppo adottate e le tecnologie impiegate: in particolare, viene descritto l'approccio di sviluppo e sono presentati gli strumenti e framework utilizzati (Ruby on Rails come framework MVC, PostgreSQL come database, gem e librerie per integrazioni esterne quali Stripe e servizi di geocodifica), evidenziando come queste tecnologie abbiano supportato la realizzazione del progetto.

Nel **Capitolo 4** si affronta l'analisi concettuale della base di dati: viene proposto lo schema E-R del dominio con il glossario delle entità e delle relazioni principali, discutendo le scelte progettuali e le regole di business rilevanti.

Sulla base di questo modello concettuale, il **Capitolo 5** dettaglia la progettazione del sistema: dapprima la progettazione logica del data layer (mappatura delle entità in tabelle relazionali all'interno di Rails, migrazioni e vincoli di integrità), quindi l'architettura applicativa secondo il pattern MVC (organizzazione di model, controller, view e definizione delle principali rotte), e infine la progettazione dei componenti chiave del software (motore di ricerca e geolocalizzazione, gestione delle disponibilità di prenotazione tramite slot, integrazione del pagamento online con Stripe, sistema di recensioni e moderazione, dashboard dedicate ai partner e agli amministratori, e principi di design dell'interfaccia utente).

Il **Capitolo 6** descrive la realizzazione pratica dell'applicazione e le sue funzionalità principali: si mostrano nel dettaglio i flussi operativi implementati per soddisfare le user story, ad esempio il processo di ricerca di un campo e prenotazione con pagamento, la visualizzazione di un campo con i relativi slot prenotabili, le funzionalità delle dashboard del partner e dell'amministratore, gli aspetti di front-end (interfaccia utente, responsività) e le misure di sicurezza e gestione dei ruoli.

Nel **Capitolo 7** viene affrontata la fase di validazione del progetto: si illustrano i test effettuati per verificare il corretto funzionamento di Campetto.it, includendo scenari di test automatizzati che coprono le principali funzionalità lato partner, utente e integrazione con servizi esterni; inoltre, si discutono i risultati dei test e si forniscono indicazioni per il deploy e l'esecuzione dell'applicazione in ambiente di produzione.

Infine, il **Capitolo 8** contiene le conclusioni della tesi e alcuni possibili sviluppi futuri: vengono riassunti i risultati raggiunti dal progetto e si suggeriscono miglioramenti ed estensioni che potrebbero essere apportati in lavori successivi, insieme a riflessioni finali sull'esperienza progettuale.

2 Raccolta e analisi dei requisiti

2.1 Dominio applicativo

L'ambito di riferimento del progetto è costituito dai centri sportivi di dimensioni contenute o intermedie, generalmente a carattere amatoriale o semi-professionale. Questi impianti offrono campi per discipline come calcetto, basket, tennis, padel, pallavolo, beach volley e altre attività simili, che si svolgono a rotazione in fasce orarie prestabilite. Le finestre di prenotazione sono spesso differenziate in base al giorno della settimana, alla stagione o ad eventi straordinari (tornei, manutenzioni, festività), generando una disponibilità dinamica ma poco formalizzata.
Da una parte, gli utenti sono interessati a trovare uno spazio disponibile vicino alla propria posizione, per una data e un orario compatibili con i propri impegni. Dall'altra, i gestori hanno la necessità di trasformare le regole operative interne in disponibilità prenotabili, evitando errori e riducendo la gestione manuale.

La piattaforma Campetto.it interviene proprio su questa interazione tra domanda e offerta, offrendo uno strumento semplice ma rigoroso per automatizzare il ciclo completo della prenotazione.

2.1.1 Processo di prenotazione

In questo contesto, Campetto.it formalizza e automatizza il processo di prenotazione completo. Il flusso operativo parte dalla pubblicazione dell'offerta da parte del gestore: ogni centro sportivo registra nel sistema uno o più campi, specificando per ciascuno lo sport praticato, la posizione, il prezzo e le regole temporali (orari di apertura/chiusura, durata di ciascun slot prenotabile, giorni di chiusura).
Sulla base di questa configurazione, il sistema genera automaticamente gli slot di disponibilità per il campo: vengono calcolati tutti gli intervalli futuri prenotabili secondo i parametri forniti, assicurando che non si sovrappongano e rispettino i vincoli di calendario (ad esempio escludendo giorni di chiusura). Ogni slot rappresenta una specifica fascia oraria con inizio e fine definiti in modo univoco, evitando ambiguità o doppie prenotazioni.
Dal lato utente, il processo inizia con la ricerca di un campo disponibile. L'utente può avviare la ricerca inserendo un indirizzo di interesse o utilizzando la propria posizione corrente. Grazie all'integrazione di un servizio di geocodifica, l'indirizzo viene convertito in coordinate geografiche e la piattaforma restituisce un elenco di campi ordinati per prossimità, filtrando solo quelli effettivamente disponibili nella data richiesta.

L'utente può quindi visualizzare i dettagli di un campo e selezionare uno slot orario specifico da prenotare. Al momento della prenotazione, il sistema verifica in tempo reale che lo slot sia ancora libero, quindi avvia la transazione tramite il gateway di pagamento Stripe e infine invia all'utente una notifica email sull'esito.

Oltre alle funzionalità principali, vengono soddisfatti alcuni requisiti trasversali di qualità che il sistema deve garantire durante il processo di prenotazione:

- **Coerenza temporale**: il sistema non deve mostrare slot già scaduti o sovrapposti e gestire le cancellazioni di prenotazioni.
- **Trasparenza informativa**: l'utente deve poter comprendere in modo immediato e univoco tutte le condizioni della prenotazione.
- **Tracciabilità end-to-end**: ogni operazione chiave (ricerca, prenotazione, pagamento, eventuale recensione) viene registrata in modo sicuro, così da poter essere verificata a posteriori.

In sintesi, la piattaforma rende più fluido, trasparente e verificabile un processo che oggi è spesso frammentato.

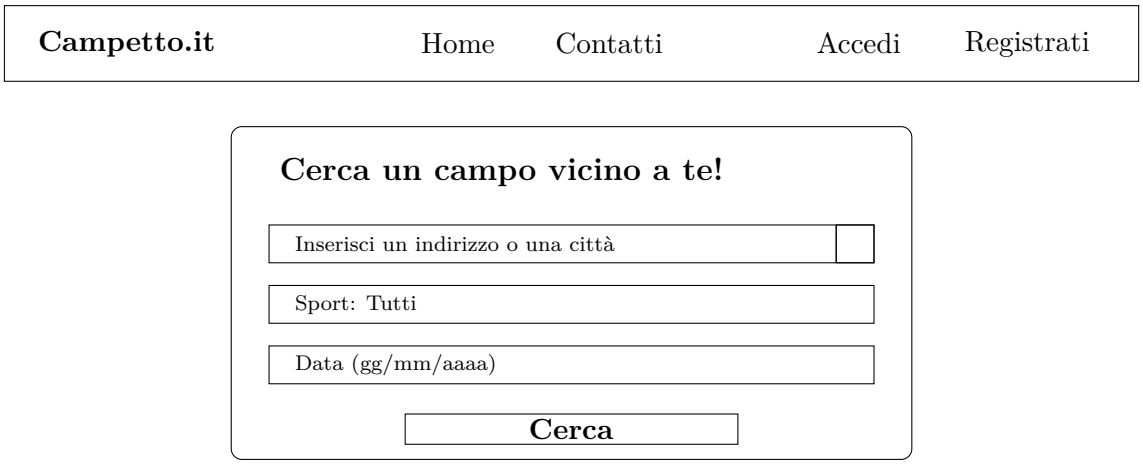


Figura 3: Wireframe iniziale della pagina di ricerca campi.

2.2 Attori coinvolti

Nel dominio di Campetto.it operano tre profili principali di utenti, ognuno con ruoli e responsabilità differenti, assistiti da alcuni servizi esterni integrati nella piattaforma.

Utente registrato (Cliente): è l'utilizzatore finale che cerca e prenota i campi. Può effettuare ricerche per indirizzo o posizione, filtrare i risultati per sport e data, visualizzare la scheda dettagliata di un campo (con info, disponibilità e recensioni), procedere al checkout con pagamento online e ricevere notifiche via email. Solo gli utenti autenticati possono completare prenotazioni e pagamenti, nonché lasciare recensioni sui campi prenotati. Ogni recensione viene associata al profilo dell'utente autore e al campo relativo; l'utente può anche gestire (modificare/eliminare) le proprie recensioni, mentre non ha poteri sulle recensioni altrui.

Partner (Gestore del centro sportivo): è il proprietario o gestore delegato di un impianto sportivo. Tramite un'area riservata, il partner può inserire nuovi campi associandoli al proprio centro sportivo, definire per ciascuno gli orari di apertura, la durata degli slot prenotabili e i giorni di chiusura, e quindi avviare la generazione automatica delle fasce orarie disponibili secondo tali regole. Inoltre, il partner può consultare le recensioni ricevute dai propri campi per monitorare la qualità percepita e intervenire se necessario (ad esempio rispondendo o contattando gli utenti insoddisfatti). Il sistema assicura che ogni campo inserito sia associato solo al centro sportivo del partner autenticato, prevenendo inserimenti non autorizzati da parte di terzi.

Amministratore: è una figura di governance con credenziali speciali, distinta dagli utenti comuni e dai partner. Accedendo all'area amministrativa, l'amministratore dispone di un pannello di controllo per funzioni di moderazione e manutenzione: ad esempio può rimuovere contenuti inappropriati (recensioni o risposte non conformi), gestire l'anagrafica degli utenti (ad esempio bloccare utenti molesti) e supervisionare le segnalazioni inviate dagli utenti o partner. Pur non intervenendo nelle operazioni quotidiane di ricerca e prenotazione, l'amministratore garantisce il rispetto delle policy e l'integrità dei dati, con tutte le operazioni di amministrazione tracciate e limitate al suo profilo con privilegi elevati.

2.2.1 Servizi esterni e confini tecnologici

Parallelamente agli attori, la piattaforma interagisce con alcuni servizi esterni che estendono le funzionalità principali:

- **Servizio di geocodifica**: converte indirizzi in coordinate geografiche (e viceversa). Ciò consente la ricerca per prossimità: l'utente può cercare campi "vicino a un indirizzo" e il sistema calcola le distanze e ordina i risultati in base alla posizione. La geocodifica è utilizzata anche inversamente per ottenere un indirizzo descrittivo a partire dalla posizione GPS corrente dell'utente.

- **Gateway di pagamento (Stripe)**: gestisce le transazioni economiche in modo sicuro.
  L'applicazione crea una sessione di pagamento sul gateway Stripe; se il pagamento va a buon fine, Stripe reindirizza l'utente alla pagina di conferma su Campetto.it. Il sistema marca quindi lo slot come prenotato e crea l'istanza di `Booking` associata, salvando l'ID univoco della transazione per tracciabilità. In caso di cancellazione di una prenotazione entro i termini previsti, l'applicazione può richiedere automaticamente a Stripe un rimborso tramite API e liberare lo slot relativo, garantendo coerenza tra disponibilità e pagamenti.

- **Servizio email (SMTP tramite Action Mailer)**: si occupa dell'invio di email transazionali agli utenti. Sono previste notifiche via email per eventi come la conferma di registrazione (con link di attivazione account), l'esito di un pagamento/prenotazione, il reset password e altre comunicazioni operative. In ambiente di sviluppo, Action Mailer è configurato con un account Gmail via SMTP, e gli indirizzi URL nelle email puntano a `localhost:3000` (configurazione di default per i test).

Questi servizi esterni sono integrati nei flussi applicativi mantenendo chiari confini tecnologici. Ad esempio, la piattaforma salva le chiavi di API esterne (come la `STRIPE_SECRET_KEY`) in variabili d'ambiente, evitando di inserirle nel codice sorgente o nel repository, migliorando la sicurezza e facilitando la configurazione su server diversi. Inoltre, per ciascun servizio sono previsti handling di errore e fallback: ad esempio, se il servizio di geocodifica non risponde, la ricerca può degradare mostrando risultati non ordinati per distanza; se un pagamento non viene confermato, la prenotazione viene annullata e l'utente informato del problema, evitando inconsistenze. Queste accortezze garantiscono che l'integrazione di servizi terzi non comprometta l'affidabilità complessiva del sistema.

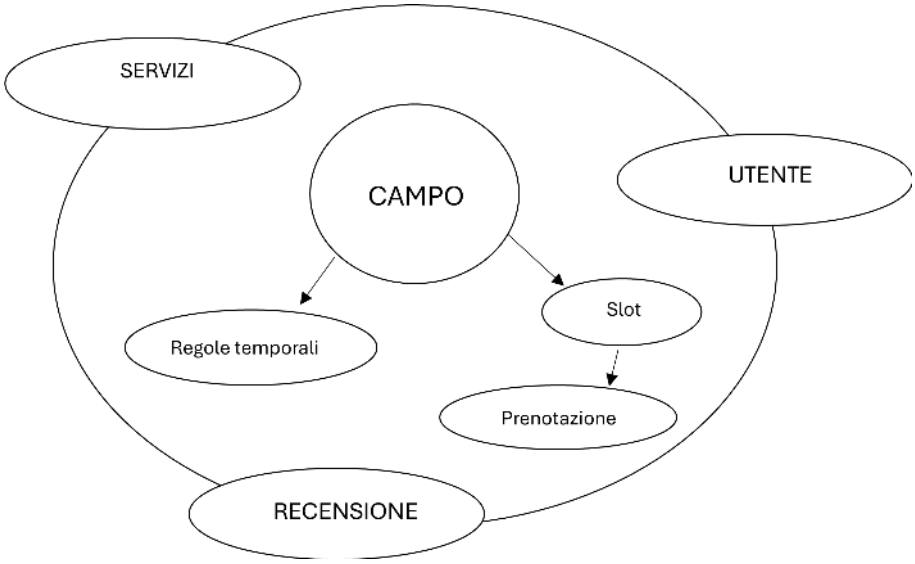


Figura 4: Mappa concettuale della realtà applicativa e delle entità coinvolte

2.3 User stories e requisiti del sistema

La raccolta e l'analisi dei requisiti funzionali è stata condotta attraverso la definizione di user story dal punto di vista dei tre attori (utente, partner, amministratore). Ciascuna user story adotta la forma tipica: "*Come <attore> voglio <obiettivo> affinché <beneficio>*", ed è accompagnata da criteri di accettazione che descrivono le condizioni di soddisfacimento.
Di seguito riporto un esempio di due storie per ogni attore accompagnato da una sintesi dei relativi criteri di accettazione:

Ricerca e visualizzazione dei campi sportivi
Come utente, autenticato o non, voglio poter ricercare un campo sportivo di una determinata disciplina, basandomi sulla mia posizione o su un indirizzo da me inserito, in modo da trovare campi vicini e disponibili.
Criterio di accettazione:
L'indirizzo fornito viene normalizzato tramite geocodifica; se l'indirizzo non è valido viene mostrato un messaggio chiaro e non vengono restituiti risultati parziali. L'elenco dei campi risultanti è ordinato per distanza e rispetta i filtri impostati (sport e data).

Gestione delle prenotazioni

Come utente autenticato, voglio poter cancellare una prenotazione futura, e se previsto dalla policy del partner, ottenere un rimborso.

Criterio di accettazione:

Se la cancellazione avviene con un anticipo sufficiente rispetto all'orario di inizio dello slot, il gateway di pagamento (Stripe) esegue un rimborso completo; in caso contrario, l'annullamento non comporta alcun rimborso. In entrambi i casi, lo slot viene liberato e torna prenotabile, e l'utente riceve un'email di conferma dell'avvenuta cancellazione.

Gestione dei campi sportivi e delle prenotazioni

Come partner autenticato, voglio poter creare, visualizzare, modificare ed eliminare i miei campi sportivi, per gestire l'offerta dei miei servizi.

Criterio di accettazione:

La piattaforma verifica la proprietà del centro sportivo prima di consentire la creazione di un nuovo campo. La generazione degli slot di disponibilità riflette le regole impostate dal partner e non produce slot duplicati o incoerenti.

Gestione delle recensioni e delle segnalazioni

Come partner autenticato, voglio poter visualizzare le recensioni dei miei campi, per monitorare il feedback dei clienti.

Criterio di accettazione:

Il partner può consultare in sola lettura le recensioni riguardanti i propri campi; eventuali interventi di moderazione su tali contenuti competono esclusivamente all'amministratore della piattaforma.

Gestione delle segnalazioni

Come amministratore autenticato, voglio poter visualizzare le segnalazioni effettuate dai clienti e dai partner, per valutare comportamenti scorretti durante l'uso della piattaforma.

Criterio di accettazione:

Tutte le azioni di moderazione sono tracciate e limitate al profilo con privilegi elevati; le operazioni potenzialmente impattanti (come la rimozione di contenuti o utenti) richiedono una conferma esplicita prima di essere eseguite.

Verifica degli account partner
Come amministratore, voglio poter cancellare o sospendere un account di un partner o di un cliente, per risolvere eventuali problemi o violazioni.
Criterio di accettazione:
Solo gli amministratori possono sospendere o cancellare un account; ogni operazione richiede conferma esplicita ed è tracciata, e un account sospeso o cancellato non può più accedere alla piattaforma.

La versione completa delle user story è consultabile nelle ultime pagine in **Appendice A**, mentre i corrispondenti storyboard e i mockup di interfaccia sono forniti in **Appendice B**.

In tutte le user story, i criteri di accettazione convergono su tre proprietà trasversali ritenute fondamentali: la **coerenza** dello stato, la **trasparenza** verso l'utente e la **tracciabilità** degli eventi chiave a fini di controllo e supporto.

2.4 Requisiti funzionali e non funzionali

Dalle user story sono stati derivati i requisiti *funzionali* del sistema, organizzati in un "piano funzionale" che elenca tutte le funzionalità che l'applicazione deve offrire. In sintesi, le principali funzionalità richieste sono:

- ***Ricerca dei campi sportivi***: permettere all'utente (anche non autenticato) di trovare campi per sport e area geografica, utilizzando indirizzo o posizione corrente, con risultati ordinati per distanza.
- ***Visualizzazione dettagli campo***: mostrare la scheda completa di un impianto sportivo con informazioni generali, foto, disponibilità di slot prenotabili e recensioni degli utenti.
- ***Registrazione e autenticazione utenti***: consentire agli utenti di registrarsi con email e password, confermare la propria email tramite link, e accedere con credenziali (login/logout), eventualmente con sessione persistente.
- ***Gestione prenotazioni***: consentire all'utente autenticato di prenotare uno slot disponibile effettuando il pagamento online (integrazione con Stripe), visualizzare lo storico delle proprie prenotazioni e cancellare una prenotazione futura (con eventuale rimborso automatico secondo la policy). In caso di cancellazioni da parte del gestore o amministratore, inviare notifiche all'utente coinvolto.

- ***Recensioni dei campi***: permettere agli utenti autenticati di scrivere una recensione con valutazione numerica su un campo prenotato e gestire le proprie recensioni (modifica/eliminazione). Mostrare le recensioni nella pagina del campo, con possibilità per i partner di rispondere (e per gli utenti di segnalare abusi).

- ***Area Partner (Gestione campi)***: fornire ai partner registrati un'area riservata per creare nuovi campi associati al proprio centro sportivo, configurarne gli orari e le regole di prenotazione, e quindi generare automaticamente gli slot prenotabili. Consentire inoltre di vedere le prenotazioni ricevute sui propri campi e le recensioni relative, e ricevere notifiche via email per eventi importanti (ad esempio una nuova prenotazione, una cancellazione, una nuova recensione).

- ***Area Admin (Moderazione e controllo)***: prevedere un back-office amministrativo dove l'amministratore possa visualizzare e filtrare contenuti generati dagli utenti (recensioni, segnalazioni), moderare contenuti inappropriati (ad esempio rimuovere recensioni offensive), gestire gli account utente e partner (ad esempio disabilitare account fraudolenti) e monitorare il funzionamento generale del sistema. L'admin deve anche poter visualizzare log o report sulle transazioni eseguite (prenotazioni e pagamenti) a fini di audit.

Oltre ai requisiti funzionali, sono stati stabiliti anche importanti requisiti *non funzionali*. Tra questi rientrano:

- ***Usabilità e responsività***: l'interfaccia deve essere semplice e intuitiva per tutte le tipologie di utente, con un design responsive che si adatti a dispositivi diversi (desktop, tablet, smartphone).

- ***Sicurezza***: la piattaforma deve proteggere i dati sensibili (password utenti, dati di pagamento) adottando best practice (password hash sicuri, comunicazioni su HTTPS, nessuna esposizione di chiavi segrete nel codice). L'integrazione con Stripe garantisce la sicurezza dei pagamenti, esternalizzando la gestione delle carte di credito e rispettando gli standard PCI.

- ***Affidabilità e consistenza***: grazie ai vincoli di coerenza temporale e tracciabilità citati prima, il sistema deve evitare situazioni di overbooking o dati incoerenti. Transazioni come la prenotazione e il pagamento devono essere atomiche: o tutte le operazioni associate (aggiornamento disponibilità, creazione prenotazione, registrazione pagamento) vanno a buon fine, oppure nessuna (roll-back in caso di errore).

- ***Prestazioni scalabili***: il sistema deve essere in grado di gestire un carico di utenti crescente (ad esempio molte ricerche contemporanee nelle ore di punta) mantenendo tempi di risposta accettabili. L'uso di query geospaziali ottimizzate e indici sul database (per coordinate, ecc.) aiuta a mantenere prestazioni efficienti.
- ***Manutenibilità ed estendibilità***: il codice organizzato secondo MVC e l'uso di gemme standard agevolano la manutenzione futura. La struttura modulare (ad esempio un modulo per pagamenti Stripe, uno per geocoding) consente di sostituire o aggiornare componenti (ad esempio cambiare provider di pagamento) con impatto minimo sul resto del sistema.

Per la versione completa dei requisiti e la loro categorizzazione si rimanda alla documentazione di analisi raccolta in **Appendice C**. In questo paragrafo ne è stata presentata una sintesi focalizzata sugli elementi più rilevanti per la comprensione del lavoro personale del progetto.

3 Metodologie e tecnologie utilizzate

3.1 Metodologia di sviluppo

Lo sviluppo di Campetto.it è stato condotto con un approccio iterativo e incrementale, tipico delle metodologie Agile: il progetto è stato suddiviso in cicli di sviluppo (iteration) in cui, ad ogni iterazione, si analizzavano un sottoinsieme di requisiti, si implementavano le relative funzionalità e si effettuavano test integrati. Questo approccio ha permesso di ottenere feedback continui e di adattare il prodotto in corso d'opera, nonché di avere sempre una versione funzionante (anche se parziale) dell'applicazione dopo ogni ciclo. Ad esempio, in una prima iterazione ci si è concentrati sulle funzionalità di base di ricerca e visualizzazione dei campi; nelle iterazioni successive si sono aggiunte la prenotazione con pagamento, poi le recensioni, e così via, assicurando che ogni blocco fosse verificato prima di procedere. La gestione dei requisiti tramite user story si è ben sposata con questo metodo: ad ogni sprint di sviluppo veniva assegnato un insieme di user story da implementare, e i criteri di accettazione di ciascuna storia costituivano la base per i test di validazione.
Dal punto di vista della collaborazione e controllo di versione, il codice è stato gestito su un repository Git condiviso, permettendo a più sviluppatori di lavorare in parallelo su componenti diversi. Sono stati effettuati code review periodici per garantire la qualità del codice e l'aderenza agli standard del progetto (convenzioni Ruby on Rails, stile di codifica, ecc.). L'evoluzione dello schema del database è stata gestita esclusivamente tramite le migration di Rails, così da mantenere la tracciabilità di ogni modifica al data layer.

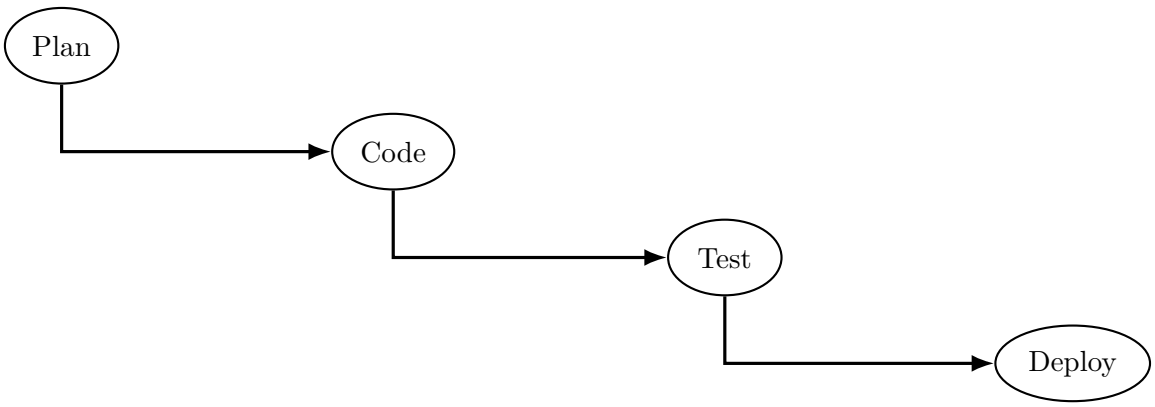


Figura 5: Fasi del processo di sviluppo fino al deploy

3.2 Stack tecnologico e strumenti

L'applicazione è costruita interamente in Ruby on Rails (versione 6.1) eseguito su Ruby 2.7. Come database relazionale si utilizza PostgreSQL (v.14) per la persistenza dei dati. L'application server è Puma, integrato in Rails, per gestire le richieste web concorrenti. Tali tecnologie di base sono ampiamente diffuse e vengono richiamate qui solo per completezza, mentre i paragrafi successivi approfondiscono le integrazioni e i servizi esterni più specifici del progetto.

Il lato frontend utilizza le viste server-side di Rails scritte in ERB (Embedded Ruby) per generare HTML dinamico. I fogli di stile sono sviluppati in SCSS e integrati tramite l'Asset Pipeline di Rails, combinando componenti personalizzati con le utility del framework CSS Bootstrap fornendo una base responsiva solida, riducendo il tempo necessario per il design UI e garantendo uniformità nell'aspetto delle pagine.

Per la gestione dei file (es. immagini dei campi sportivi caricate dai partner) si utilizza Active Storage, che in ambiente di sviluppo salva i file su disco locale (in futuri deploy in produzione potrà essere configurato con servizi cloud tipo AWS S3). Lo stack include anche Node.js e Yarn (specificamente Node 10 e Yarn 1.22) come dipendenze per la gestione di pacchetti frontend e compilazione di asset moderni qualora necessario.

In fase di sviluppo, l'ambiente principale è stato Linux (Ubuntu/LXLE 18.04 64-bit) come piattaforma di deployment locale, sfruttando strumenti come Bundler (2.1) per la gestione delle gemme Ruby e RVM/rbenv per la gestione delle versioni di Ruby. Per l'editor e il debugging, sono stati utilizzati strumenti comuni tra gli sviluppatori Rails: ad esempio Visual Studio Code con estensioni Ruby, e il Rails console interattivo per ispezionare gli oggetti del dominio durante l'implementazione.

3.3 Integrazioni e librerie esterne

Stripe (pagamenti): tramite la gemma *`stripe`* (v. 13.0.x), l'applicazione si connette alle API di Stripe per creare `PaymentIntent` e sessioni di checkout. Alla conferma di una prenotazione viene inizializzato un flusso di pagamento: in caso di esito positivo lo slot viene marcato come prenotato e la relativa `Booking` memorizza l'identificativo univoco della transazione, così da garantirne la tracciabilità; in caso di cancellazione con rimborso, l'applicazione invoca le API di `Stripe::Refund` e aggiorna in modo coerente sia lo stato dello slot sia quello della prenotazione.

Geocoder + Nominatim (geocodifica): la gemma *geocoder* che utilizza il servizio Nominatim basato su OpenStreetMap fornendo il supporto alle funzionalità di ricerca geospaziale. In particolare, viene impiegata sia per convertire un indirizzo inserito dall'utente in coordinate latitudine/longitudine (*forward geocoding*) sia per l'operazione inversa di ottenere un indirizzo approssimativo a partire dalla posizione corrente del browser (*reverse geocoding*). Ogni entità `Field` sfrutta quindi le query di ricerca tramite le coordinate per filtrare e ordinare i campi in base alla distanza.

Action Mailer (email): per l'invio delle comunicazioni automatiche il progetto sfrutta il modulo `Action Mailer` integrato in Rails, configurato in ambiente di sviluppo con un account Gmail via SMTP e credenziali gestite tramite variabili d'ambiente. I mailer dedicati inviano, tra le altre, le email di conferma registrazione, le notifiche di avvenuta prenotazione, i messaggi relativi a eventuali cancellazioni e i link di reset della password; ciascuna email è costruita a partire da viste dedicate (in formato testo e HTML) allineate allo stile grafico della piattaforma.

Altre librerie: oltre alle integrazioni principali, l'applicazione utilizza gemme e librerie di supporto per la sicurezza e l'interfaccia. In particolare, *bcrypt* viene impiegata tramite il meccanismo `has_secure_password` per l'hashing delle credenziali di `User`, `Partner` e `Administrator`, mentre Bootstrap e jQuery (inclusi nella pipeline degli asset) sostengono la realizzazione dell'interfaccia responsiva e delle interazioni dinamiche (ad esempio i componenti della dashboard e le richieste AJAX). Funzionalità più avanzate, come la visualizzazione dei campi su una mappa interattiva, sono state considerate come sviluppi futuri e non fanno parte del perimetro core della versione attuale.

3.4 Ambienti di deployment e configurazione

L'applicazione è stata sviluppata e testata in ambiente di development locale, ma è predisposta per essere deployata in ambiente production su un server o hosting supportato da Rails. La configurazione distingue i due ambienti principalmente nell'uso di chiavi e servizi:

- In **development**, come detto, si usano chiavi di test e servizi come Gmail SMTP. Il database locale PostgreSQL contiene dati di prova e può essere rigenerato facilmente. I log di Rails sono impostati al livello debug per fornire informazioni dettagliate durante lo sviluppo.

- In **production**, si utilizza un servizio di hosting con un database PostgreSQL gestito. Le credenziali sensibili (database, API keys) sono inserite come variabili d'ambiente sul server. Il livello di log viene elevato a info o warn per performance.

Uno script di seed iniziale è predisposto per popolare il database con dati minimi utili al collaudo. Ciò consente, subito dopo il deploy o su una nuova installazione, di avere un dataset di partenza per esplorare l'applicazione senza dover inserire tutto manualmente.

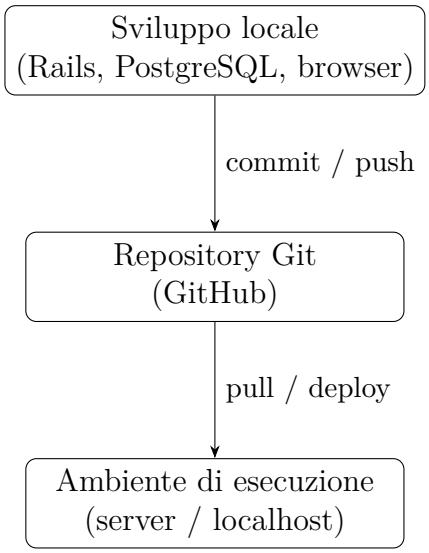


Figura 6: Pipeline di sviluppo e deploy di *Campetto.it*: dallo sviluppo locale, al repository Git, fino all'ambiente di esecuzione.

In conclusione di questo capitolo, riepiloghiamo che il progetto adotta uno stack moderno e consolidato (Rails/PostgreSQL) e integra efficacemente servizi esterni essenziali. L'ambiente di sviluppo locale è stato impostato in modo da rispecchiare il più possibile il futuro ambiente di produzione, facilitando la transizione e riducendo problemi di configurazione. Per garantire la riproducibilità del lavoro, la documentazione di progetto include inoltre una breve guida all'installazione e all'avvio dell'applicazione (clonazione del repository, configurazione delle variabili d'ambiente, esecuzione delle migration e del seed, avvio del server Rails in ambiente di sviluppo), così da rendere semplice l'esecuzione di un dimostratore a partire dalla codebase. Nelle fasi successive vedremo come questo stack viene sfruttato nella progettazione e realizzazione delle varie componenti del sistema.

4 Analisi concettuale della base dati

4.1 Modello concettuale dei dati

La base di dati di *Campetto.it* è stata progettata per modellare in modo coerente il processo di ricerca, prenotazione e valutazione di campi sportivi. La progettazione concettuale si è posta l'obiettivo di individuare le entità fondamentali del dominio, le rispettive responsabilità e relazioni, mantenendo il modello il più semplice possibile, privo di ridondanze e aderente alla logica applicativa implementata in *Ruby on Rails*. In base a questa analisi è stato sviluppato un modello entità–relazioni (E-R) che descrive gli attori principali del sistema (utente, gestore, amministratore), le risorse oggetto di ricerca e prenotazione (campi sportivi e fasce orarie prenotabili) e gli elementi di contorno del ciclo di vita della prenotazione (recensioni e segnalazioni). Nel seguito vengono presentati il modello concettuale dei dati, un glossario delle principali entità e le relazioni cardine che strutturano la base dati. Lo schema E-R complessivo della base dati è illustrato in Figura 7, che sintetizza graficamente le entità e le relazioni descritte in questo capitolo.

La progettazione concettuale è partita dall'identificazione delle entità chiave del dominio e dei principali vincoli che il sistema deve rispettare (ad esempio, prevenire l'*overbooking* sullo stesso intervallo temporale o garantire che ogni recensione abbia un autore definito).
Dall'analisi del dominio di Campetto.it sono emerse le seguenti entità principali: *User* (utente), *Partner* (gestore), *Administrator* (amministratore della piattaforma), *Field* (campo sportivo), *Slot* (fascia oraria prenotabile), *Booking* (prenotazione), *Review* (recensione) e *Report* (segnalazione). Il modello E-R sviluppato collega tra loro queste entità per coprire tutti gli aspetti funzionali richiesti dalla piattaforma. Alcune scelte progettuali rilevanti per la modellazione dei dati sono state le seguenti:

- **Configurazione dei campi tramite *Field***: L'entità *Field* incapsula l'intera configurazione che governa la generazione degli *Slot* (orari di apertura e chiusura, durata degli intervalli prenotabili, giorni esclusi). Quando un Partner crea un nuovo campo, la logica applicativa genera automaticamente gli Slot futuri in base a questa configurazione, evitando l'inserimento manuale di ogni singola fascia oraria e garantendo coerenza tra le regole impostate e la disponibilità effettiva.

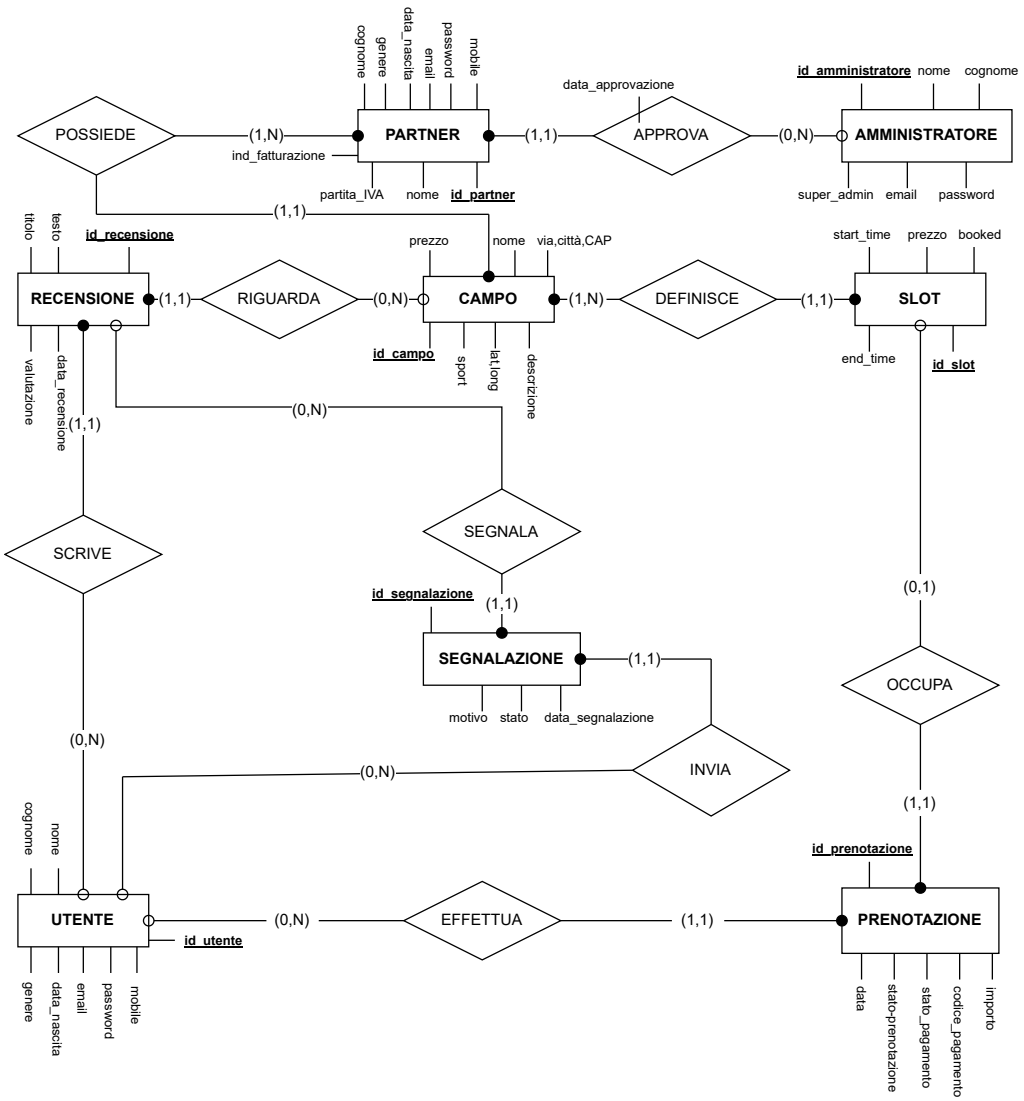
- **Uso dello *Slot* come livello intermedio**: L'entità *Booking* (prenotazione) non mantiene un riferimento diretto al campo, ma punta allo Slot specifico prenotato.
  Lo stato di occupazione è quindi un attributo dello Slot (ciascuno rappresenta un intervallo temporale univoco) e, a livello di dominio, è impossibile associare due prenotazioni diverse al medesimo Slot. In questo modo si previene l'overbooking già a livello di modellazione dei dati. In Figura 7 questa associazione è indicata dalla relazione *Occupa* tra *Booking* e *Slot*.

- **Recensioni collegate a utente e campo**: L'entità *Review* (recensione) collega un *User* a un *Field* in una relazione molti-a-uno da entrambe le parti: un utente può recensire più campi e ogni campo può ricevere più recensioni, ma ciascuna recensione ha un unico autore. La cardinalità è quindi uno-a-molti sia dal lato utente sia dal lato campo, garantendo che ogni recensione abbia un solo autore ed esista un solo campo di riferimento.

- **Segnalazioni collegate a utente e recensione**: L'entità *Report* (segnalazione) mette in relazione un *User* con una specifica *Review*, tramite le relazioni *Invia* e *Segnala* (Figura 7). Ogni segnalazione ha quindi un unico autore ed ha come oggetto una sola recensione, mentre una stessa recensione può essere segnalata più volte da utenti diversi. A livello implementativo, la struttura è stata mantenuta flessibile tramite le associazioni polimorfiche di Rails, in modo da poter estendere in futuro il meccanismo di segnalazione ad altre tipologie di contenuto.

- **Ruoli distinti per utenti, gestori e amministratori**: Le entità *User*, *Partner* e *Administrator* rappresentano tre ruoli separati all'interno della piattaforma. Ogni *Field* è di proprietà di un singolo Partner, che ne controlla la configurazione e il calendario delle disponibilità; gli User interagiscono con i campi tramite prenotazioni, recensioni e segnalazioni; gli Administrator non partecipano al flusso di gioco, ma hanno visibilità globale e permessi di moderazione su utenti, partner, contenuti e segnalazioni.

- **Rappresentazione geospaziale dei campi**: L'entità *Field* memorizza sia l'indirizzo strutturato sia le coordinate geografiche (latitudine e longitudine) ottenute tramite geocodifica. Questo duplice livello di rappresentazione consente da un lato di mostrare all'utente un indirizzo leggibile, dall'altro di supportare query di ricerca ordinate per distanza a partire da una posizione di riferimento (ad esempio la posizione corrente o un indirizzo inserito manualmente).



VINCOLI ESTERNI:

1. Prevenzione overbooking: per uno stesso campo non possono esistere slot sovrapposti e ogni Slot può essere associato al massimo a una sola Booking.
2. Unicità credenziali: l'indirizzo email è un identificatore univoco all'interno di ciascuna categoria (User, Partner, Administrator).
3. Recensioni vincolate: una Review su un campo è consentita solo agli utenti che hanno effettuato almeno una Booking per quel campo.
4. Segnalazioni attribuite: ogni Report deve avere sempre un reporter e un reportable non nulli (niente segnalazioni anonime).
5. Valori ammessi: prezzo, durata degli slot e altri attributi numerici devono essere positivi; i voti delle Review sono compresi in un intervallo discreto (es.1–5).

Figura 7: Diagramma Enità-Relazione della base dati

4.2 Principali entità e relazioni (Glossario)

In questa sezione vengono descritte in modo più dettagliato le principali entità del dominio, con l'obiettivo di fornire un glossario coerente con la base dati.

User (utente) .

Rappresenta l'utente registrato che utilizza la piattaforma per cercare campi, effettuare prenotazioni, lasciare recensioni e inviare segnalazioni. Ogni utente è autenticato tramite email e password, la cui hash sicura è memorizzata usando il meccanismo `has_secure_password` fornito da Rails. Oltre alle credenziali, il profilo utente contiene informazioni personali di base (nome, cognome) e lo stato di verifica dell'email. Gli User sono i principali fruitori del sistema: dall'interazione di ricerca dei campi alla prenotazione, con la possibilità di pubblicare recensioni e utilizzare il servizio di segnalazione. Si noti che un utente può recensire un campo *solo* dopo averlo effettivamente prenotato e utilizzato; in altre parole, per ogni recensione esiste una prenotazione precedente del medesimo campo effettuata dallo stesso utente. Questo vincolo assicura che le valutazioni provengano da utenti che hanno realmente usufruito del servizio.

Partner (gestore) .

È l'attore che detiene o gestisce uno o più campi sportivi pubblicati su Campetto.it. Anche il Partner è un utente autenticato con email e password, ma con privilegi di gestione: tramite una dashboard dedicata può creare, modificare e disattivare i propri Field, configurarne gli orari, aggiornare i prezzi e consultare le prenotazioni ricevute. Nel modello dati, il Partner include, oltre alle credenziali, i dati anagrafici di base (nome, cognome, genere, data di nascita), un recapito telefonico e le informazioni necessarie per eventuali comunicazioni amministrative. Dopo la registrazione, un Partner rimane nello stato *in attesa*: il suo account deve essere approvato da un Administrator prima che possa creare campi o accettare prenotazioni. Questo controllo (rappresentato dalla relazione *Approva* in Figura 7) garantisce che solo gestori verificati possano operare sulla piattaforma.

Administrator (amministratore della piattaforma) .

Rappresenta il personale che gestisce la piattaforma. Gli amministratori hanno il compito di verificare e approvare i nuovi Partner registrati, moderare i contenuti generati dagli utenti (ad esempio recensioni segnalate) e intervenire in caso di violazioni (ad esempio rimuovendo recensioni inappropriate o sospendendo account scorretti).

Nel modello E-R (Figura 7) è inclusa la relazione *Approva* tra Administrator e Partner, a indicare che ogni nuovo Partner deve essere abilitato da un amministratore prima di poter operare.

Field (campo sportivo) .

Rappresenta il singolo campo sportivo prenotabile dagli utenti. Ogni Field appartiene a un unico Partner, che ne è il proprietario e lo gestisce tramite la propria dashboard. Il modello contiene vari attributi descrittivi: nome, descrizione, tipo di sport praticato, prezzo orario (o per slot), indirizzo completo in forma testuale (via, CAP, città, indirizzo aggregato), coordinate geografiche (latitudine, longitudine) ottenute tramite geocodifica, oltre alla configurazione oraria per la generazione degli Slot. Dal punto di vista del dominio applicativo, il campo è l'unità fondamentale attorno a cui ruotano la ricerca dei campi, le prenotazioni, le recensioni e la geolocalizzazione.

Slot (fascia oraria prenotabile) .

Rappresenta un intervallo temporale specifico in cui è possibile effettuare una prenotazione, generato automaticamente a partire dalla configurazione di un Field. Ogni Slot è caratterizzato da un orario di inizio e uno di fine (`start_time`, `end_time`) e da uno stato di disponibilità (un flag booleano `booked` che indica se lo slot è stato occupato). Per ogni campo, la logica applicativa genera in anticipo una griglia di Slot futuri rispettando gli orari di apertura/chiusura e saltando i giorni contrassegnati come esclusi. Ciascuno Slot può essere associato al massimo a una Booking, garantendo che non esistano mai due prenotazioni sul medesimo intervallo temporale.

Booking (prenotazione) .

Rappresenta la prenotazione effettiva di uno Slot da parte di un utente. Una Booking mette in relazione un User con uno Slot specifico: non contiene un collegamento diretto al Field, ma raggiunge il campo attraverso lo slot (che è univocamente associato a un campo).Oltre alle chiavi esterne verso *User* e *Slot*, il modello memorizza la data della prenotazione, lo stato della prenotazione, le informazioni sul pagamento (codice restituito da Stripe, stato del pagamento e importo addebitato). I timestamp di creazione e aggiornamento (`created_at`, `updated_at`) sono gestiti a livello applicativo e non sono esplicitati nello schema concettuale per mantenere il diagramma più leggibile.

Review (recensione) .

È un contenuto generato da un utente dopo aver usufruito del servizio su uno specifico campo. Ogni Review riguarda sempre un Field determinato ed ha come autore un User ben definito. Gli attributi principali includono un voto numerico (tipicamente su scala 1–5), un titolo, un testo descrittivo e i relativi timestamp (data di creazione e di ultimo aggiornamento). Il modello dati impone vincoli di validazione sui valori ammessi (ad esempio range del voto, lunghezza massima del testo), verificati lato server. Un utente può scrivere più recensioni nel tempo (ad esempio giocando in campi diversi o tornando più volte nello stesso campo), ma ciascuna recensione ha un solo autore ed è associata a un solo campo.

Report (segnalazione) .

Modella le segnalazioni di contenuti inappropriati o problematici. Ogni segnalazione (ad esempio relativa a una recensione offensiva) è associata all'utente che l'ha inviata e alla recensione oggetto della segnalazione, includendo informazioni come la data, una motivazione testuale e uno stato gestito dall'amministratore (ad esempio *pending*, *accepted*, *rejected*). Nel modello E-R (Figura 7) questi legami sono rappresentati dalle relazioni *Invia* (*User–Report*) e *Segnala* (*Report–Review*). L'implementazione in Rails utilizza una struttura polimorfica (*reporter*, *reportable*) che consente, se necessario, di estendere in futuro il meccanismo di segnalazione ad altri tipi di contenuto oltre alle recensioni.

4.3 Relazioni cardine

In termini di relazioni di cardinalità tra le entità sopra descritte, si individuano i seguenti legami principali:

- **Partner : Field = 1 : N** (*Possiede*) – Un partner può gestire più campi; ogni campo appartiene esattamente a un partner.
- **Administrator : Partner = 1 : N** (*Approva*) – Un amministratore può approvare più partner; ciascun partner viene approvato da un solo amministratore (relazione *Approva*).
- **Field : Slot = 1 : N** (*Definisce*) – Ogni campo genera molti slot; ogni slot si riferisce a un solo campo.

- **Slot : Booking = 0..1 : 1** (*Occupa*) – Uno slot può essere associato al massimo a una prenotazione, mentre ogni prenotazione occupa esattamente uno slot.
- **User : Booking = 1 : N** (*Effettua*) – Un utente può effettuare molte prenotazioni; ogni prenotazione ha un solo utente come titolare.
- **Field : Booking = 1 : N** (indiretta) – Un campo può avere molte prenotazioni tramite i propri slot; ogni prenotazione riguarda uno specifico slot di un determinato campo.
- **User : Review = 1 : N** (*Scrive*) – Un utente può scrivere più recensioni; ogni recensione ha un unico autore.
- **Field : Review = 1 : N** (*Riguarda*) – Un campo può avere molte recensioni; ogni recensione è associata a un solo campo.
- **User : Report = 1 : N** (*Invia*) – Un utente può inviare più segnalazioni; ogni segnalazione ha un solo autore.
- **Review : Report = 1 : N** (*Segnala*) – Una recensione può ricevere più segnalazioni nel tempo; ogni segnalazione si riferisce a una singola recensione.

4.3.1 Vincoli aggiuntivi del dominio

Oltre alle cardinalità illustrate, il dominio applicativo è caratterizzato da alcuni vincoli che non emergono direttamente dal solo diagramma E-R, ma risultano fondamentali per il corretto funzionamento della piattaforma. In particolare:

- **Prevenzione dell'overbooking**: Per uno stesso campo non possono esistere due Slot sovrapposti temporalmente, ed ogni Slot può essere associato al massimo a una Booking. La generazione automatica degli slot, unita al vincolo uno-a-uno tra slot e prenotazione, impedisce di fatto doppie prenotazioni sul medesimo intervallo orario.
- **Unicità delle credenziali**: L'indirizzo email è un identificatore logico univoco per ciascuna delle entità User, Partner e Administrator. Il modello concettuale assume che non possano esistere due account con la stessa email all'interno della medesima categoria, vincolo che viene poi tradotto in opportune validazioni applicative e indici univoci a livello di schema logico.

- **Recensioni vincolate all'utilizzo del servizio**: La possibilità di recensire un campo è legata all'effettiva fruizione del servizio. In particolare, un utente può pubblicare una Review su un campo solo se ha effettuato almeno una Booking relativa a quel campo. Questa regola di dominio, implementata a livello applicativo, garantisce l'attendibilità delle valutazioni pubblicate.

- **Segnalazioni sempre attribuite**: Ogni *Report* deve essere sempre associato sia a un utente autore della segnalazione, sia alla recensione oggetto della segnalazione. Non sono previste segnalazioni anonime né segnalazioni prive di riferimento al contenuto segnalato. Dal punto di vista implementativo, questi legami corrispondono ai campi polimorfici *reporter*/*reportable* utilizzati nei model Rails.

- **Valori ammessi per attributi chiave**: Alcuni attributi numerici di dominio (ad esempio il prezzo di un campo o la durata di uno slot) devono assumere valori positivi (non nulli e non negativi), mentre i voti delle Review sono vincolati a un intervallo discreto predefinito (ad esempio 1–5). Tali vincoli vengono controllati tramite apposite regole di validazione nel model applicativo.

5 Progettazione dell'architettura software

In questo capitolo viene descritta la progettazione del sistema *Campetto.it*, con particolare attenzione al passaggio dal modello concettuale della base dati alla realizzazione concreta all'interno del framework *Ruby on Rails*.

5.1 Obiettivi di progettazione

La progettazione del sistema è stata guidata da alcuni obiettivi principali:

- **Semplicità e coerenza del modello dati**: le strutture dati persistenti riflettono in modo diretto le entità concettuali (*User*, *Partner*, *Administrator*, *Field*, *Slot*, *Booking*, *Review*, *Report*), evitando ridondanze non necessarie.
- **Supporto nativo alle regole di business**: vincoli fondamentali come l'assenza di overbooking, l'unicità delle credenziali o la consistenza delle relazioni devono essere garantiti direttamente dal modello relazionale e dalle associazioni offerte dal framework.
- **Separazione chiara dei ruoli**: utenti finali, partner e amministratori accedono a viste e funzionalità differenti; la progettazione deve riflettere tale separazione sia a livello di dati sia a livello applicativo.
- **Integrazione ordinata con servizi esterni**: funzionalità come geocodifica, pagamenti online e invio di email sono incapsulate in componenti dedicati, così da poter sostituire o estendere i fornitori di servizi senza stravolgere l'architettura complessiva.
- **Manutenibilità ed estendibilità**: il codice deve rimanere leggibile e organizzato in componenti riconoscibili (*model*, *controller*, *view*, servizi), in modo da poter aggiungere nuove funzionalità.

5.2 Progettazione logica del *data layer*

A partire dal modello concettuale descritto nel capitolo precedente, si è proceduto alla progettazione logica e fisica del *data layer* nell'ambiente Ruby on Rails, utilizzando PostgreSQL come database relazionale. Lo schema è definito nel file `schema.rb`, generato automaticamente a partire dalle migration. Lo schema logico complessivo delle principali relazioni del data layer è sintetizzato in Figura 8, che mette in evidenza le chiavi primarie, le principali chiavi esterne e i legami tra utenti, campi, slot, prenotazioni, recensioni e segnalazioni.

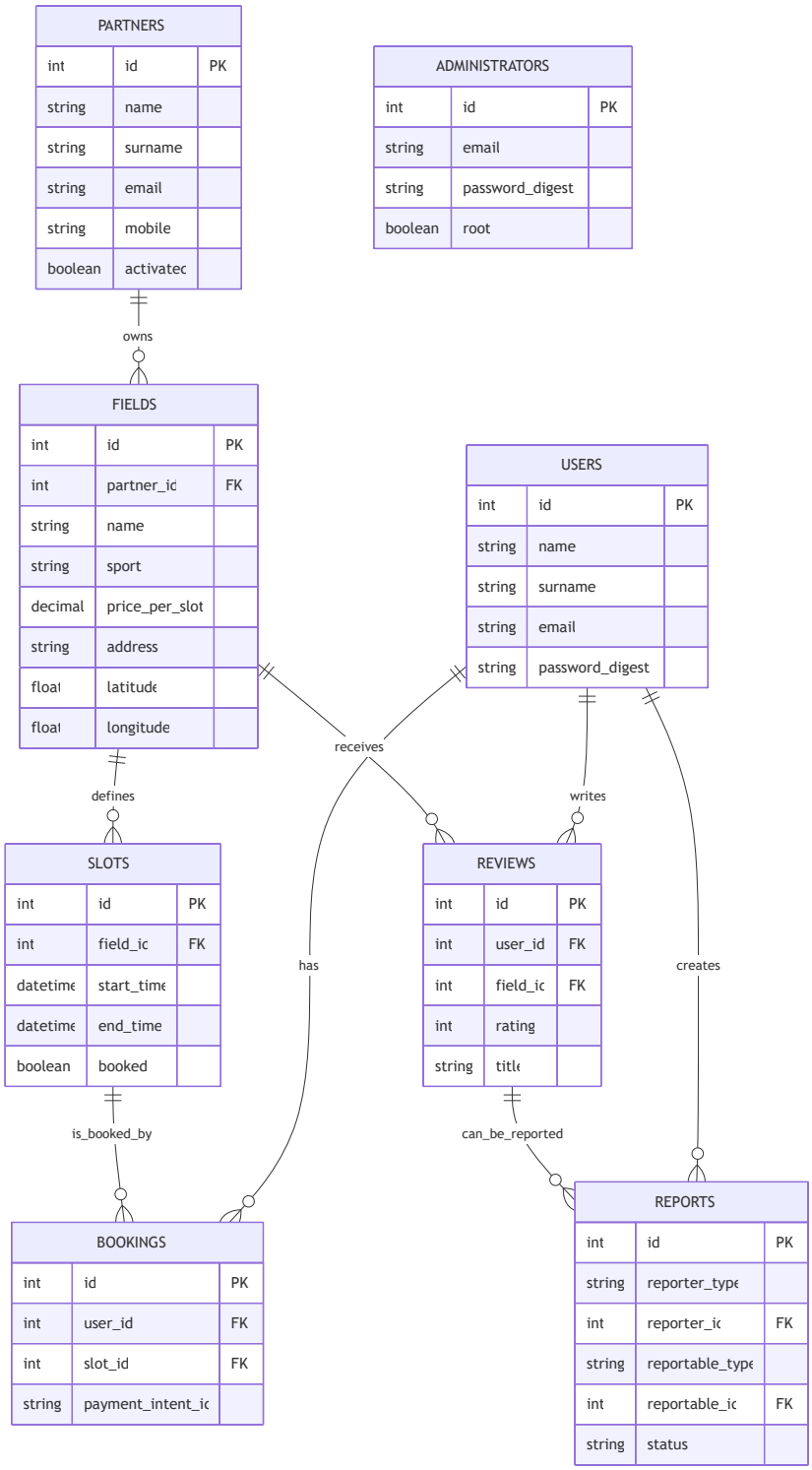


Figura 8: Schema logico relazionale della base dati di *Campetto.it*.

5.2.1 Mappatura entità–tabelle

Le entità concettuali principali sono state mappate in tabelle relazionali secondo una corrispondenza uno-a-uno, con colonne, tipi di dato, indici e vincoli coerenti con il modello E-R:

- **`users`**: rappresenta gli utenti finali. Contiene, tra gli altri campi:
  - `email`: email univoca (indicizzata) utilizzata come credenziale di login;
  - `password_digest`: hash sicuro della password, gestito da Rails tramite `has_secure_password`;
  - `nome`, `cognome`: dati anagrafici di base;
  - `uid`, `provider`: supporto per eventuali login tramite provider esterni (social login);
  - `confirmation_token`, `confirmed_at`, `confirmation_sent_at`: campi utilizzati per il processo di conferma dell'email.

- **`partners`**: memorizza i partner (gestori) che pubblicano e gestiscono i campi. Include:
  - `name`, `surname`, `gender`, `birthdate`, `mobile`: dati anagrafici e di contatto del gestore;
  - `email` (univoca), `password_digest`, `remember_digest`: credenziali di accesso;
  - `activation_digest`, `activated`, `activated_at`: informazioni per l'attivazione dell'account;
  - `reset_digest`, `reset_sent_at`: campi per la procedura di reset della password.

  A livello applicativo, il model `Partner` è associato ai campi tramite `has_many :fields`.

- **`administrators`**: contiene le credenziali degli amministratori della piattaforma. I campi principali sono:
  - `name`, `surname`, `email`;
  - `password_digest`, `remember_digest`;
  - `root`: flag booleano che distingue un amministratore "root" con privilegi estesi.

  Anche in questo caso la colonna `email` è soggetta a vincolo di unicità.

- **`fields`**: rappresenta i campi sportivi pubblicati su Campetto.it. I campi principali sono:
  - `nome`, `descrizione`, `sport`, `prezzo`;
  - `via`, `città`, `CAP`, indirizzo completo in formato testuale;
  - `latitudine`, `longitudine`: coordinate geografiche persistite a seguito della geocodifica dell'indirizzo;
  - `start_time`, `end_time`, `interval`, `exclude_days`: configurazione oraria per la generazione degli slot;
  - `partner_id`: chiave esterna verso la tabella `partners`.

  A livello di modello, `Field` dichiara `belongs_to :partner`, `has_many :slots` e `has_many :reviews` (oltre ad un `has_one_attached :image` per la foto del campo).

- **`slots`**: memorizza le fasce orarie prenotabili associate a ogni campo. Ogni record contiene:
  - `field_id`: riferimento al campo sportivo;
  - `start_time`, `end_time`: estremi temporali dello slot;
  - `booked`: flag booleano che indica se lo slot è stato prenotato.

  La relazione uno-a-molti *Field – Slot* è espressa sia a livello di schema (chiave esterna con vincolo) sia a livello di modello (associazioni `belongs_to`/`has_many`).

- **bookings**: contiene le prenotazioni effettuate dagli utenti:
  - `user_id`: utente che ha effettuato la prenotazione;
  - `slot_id`: slot prenotato;
  - `payment_intent_id`: identificativo della transazione Stripe associata.

  Le chiavi esterne verso `users` e `slots` sono protette da vincoli referenziali ed indicizzate per velocizzare le query.

- **reviews**: memorizza le recensioni degli utenti sui campi:
  - `titolo`, `valutazione`, `testo`: contenuto della recensione (titolo, voto, testo libero);
  - `user_id`: riferimento all'autore (utente);
  - `field_id`: riferimento al campo recensito.

  Il model `Review` applica vincoli a livello applicativo (ad esempio presenza del titolo, range 1–5 per la valutazione, lunghezza massima del testo).

- **reports**: gestisce le segnalazioni tramite associazioni polimorfiche:
  - `reportable_type`, `reportable_id`: oggetto segnalato (ad esempio una `Booking` o una `Review`);
  - `reporter_type`, `reporter_id`: soggetto che effettua la segnalazione (tipicamente un `User`);
  - `status`: campo intero utilizzato come enum applicativo per lo stato della segnalazione (*pending*, *accepted*, *rejected*);
  - `details`: testo descrittivo inserito dal segnalante.

5.2.2 Migrazioni e controllo dell'evoluzione dello schema

Lo schema è stato costruito e modificato tramite le *migration* di Rails: ogni creazione di tabella, aggiunta di colonna o definizione di indice è stata introdotta come migrazione versionata. Ciò consente di ricreare l'intero database da zero applicando in sequenza tutte le migration e di evolvere lo schema in modo incrementale (ad esempio aggiungendo il campo `payment_intent_id` alla tabella `bookings` o i campi di conferma email alla tabella `users`) senza perdere i dati esistenti.

5.3 Architettura applicativa: pattern MVC e rotte

L'architettura applicativa segue il pattern *Model–View–Controller* tipico del framework Rails. I *model* Active Record rappresentano il nucleo della logica di dominio e gestiscono la persistenza sul database; i *controller* orchestrano i flussi applicativi e le interazioni con l'utente (attraverso le rotte); le *view* definiscono la presentazione e l'interfaccia utente.

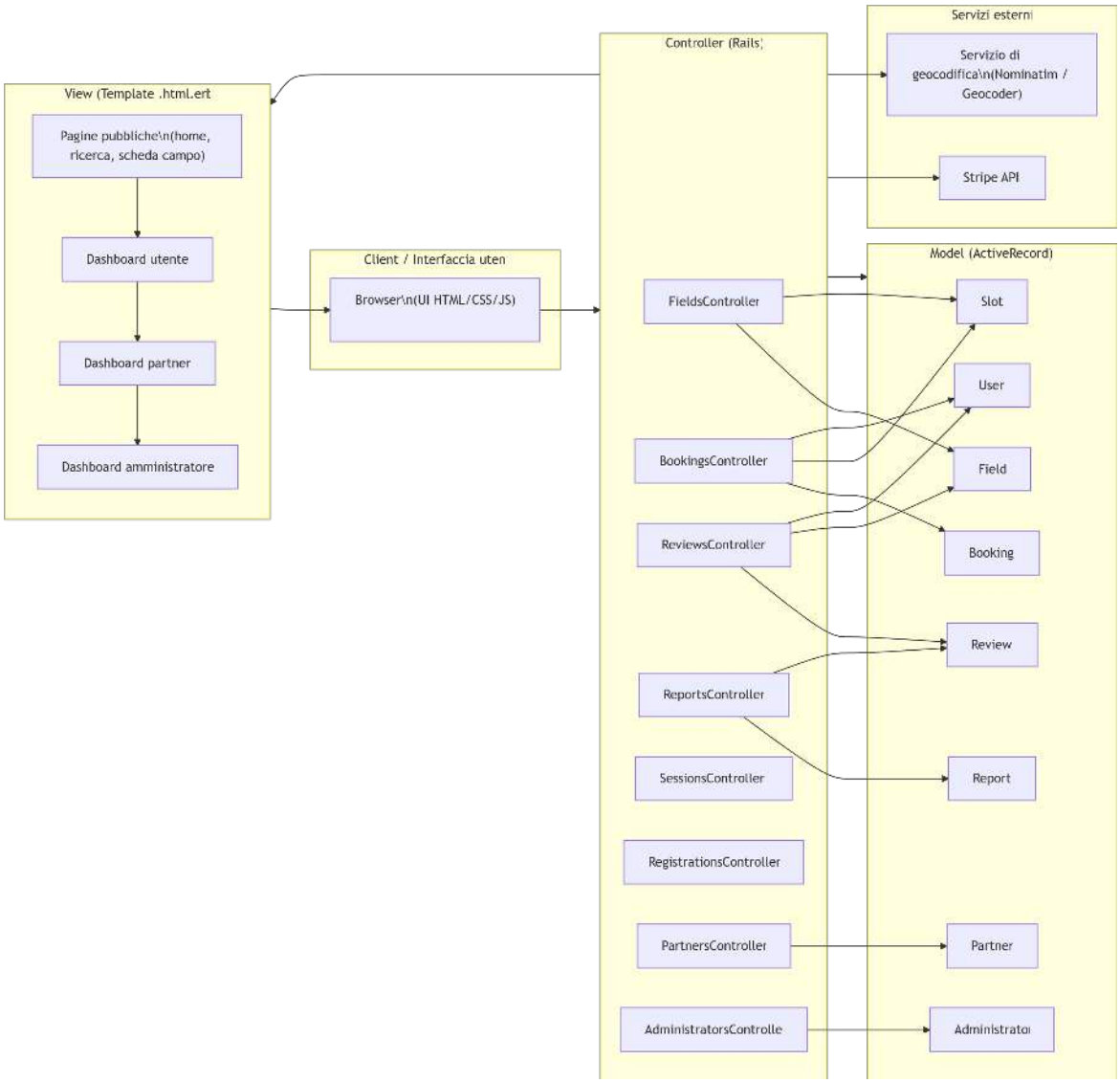


Figura 9: Diagramma package principali dell'architettura MVC di *Campetto.it*.

5.3.1 Model: logica di dominio e associazioni

I model Active Record costituiscono il nucleo della logica applicativa. In particolare:

- **User**: gestisce l'autenticazione degli utenti (`has_secure_password`), la conferma via email e le associazioni verso *Review*, *Booking* e *Report* (ad esempio `has_many :reviews`, `has_many :bookings`, `has_many :reports, as: :reporter`).
- **Partner**: incapsula la logica specifica dei gestori (validazioni anagrafiche, attivazione account, reset della password, gestione del *remember token*) e dichiara l'associazione `has_many :fields` verso i propri campi.
- **Administrator**: modella gli amministratori di piattaforma, includendo la logica di autenticazione, la gestione del *remember token* e un flag `root` per distinguere un amministratore "root" con permessi estesi.
- **Field**: oltre a definire le associazioni (`belongs_to :partner`, `has_many :slots`, `has_many :reviews`, `has_one_attached :image`), gestisce:
  - la geocodifica dell'indirizzo tramite la gemma Geocoder (`geocoded_by :full_address` con callback `after_validation` eseguito se `address_changed?`);
  - la generazione iniziale degli slot tramite un callback `after_create` che invoca `Slot.generate_slots(self)`;
  - la serializzazione dell'attributo `exclude_days` come array (lista di giorni della settimana esclusi).
- **Slot**: definisce le associazioni verso *Field* e *Booking* e contiene la logica per la generazione delle disponibilità future:
  - `generate_slots(field)`: genera gli Slot per i mesi successivi a partire dalla configurazione del campo specificato;
  - `generate_daily_slots`: metodo destinato a estendere periodicamente il calendario in avanti, evitando di creare slot duplicati per giorni già inseriti.
- **Booking**: collega un utente a uno slot, memorizza l'identificativo di pagamento Stripe associato (`payment_intent_id`) e può essere oggetto di segnalazioni (`has_many :reports, as: :reportable`).
- **Review**: collega ogni recensione a un utente e a un campo, applicando vincoli sul titolo, sul valore del voto e sulla lunghezza del testo.
- **Report**: implementa il meccanismo di segnalazione tramite associazioni polimorfiche (`belongs_to :reportable, polymorphic: true` e `belongs_to :reporter, polymorphic: true`) e gestisce lo stato tramite un enum (*pending*, *accepted*, *rejected*) inizializzato automaticamente a *pending*.

5.3.2 Controller e rotte: orchestrazione dei flussi

Il file `routes.rb` definisce le rotte principali, organizzando i percorsi per le tre tipologie di utenti (utente finale, partner, amministratore) in maniera distinta:

- **Utente**:
  - `root 'pages#home'`: pagina iniziale della piattaforma;
  - `get 'logReg'`: pagina combinata di login/registrazione per l'utente finale;
  - Rotte per l'autenticazione utente: login, logout, signup e callback OAuth (`/auth/:provider/callback`);
  - Rotte per conferma email e reset password: `confirm_email`, `confirm/:token`, `request_password_reset`, `edit_password`, `update_password`;
  - `get 'search_fields'` (instradata a `fields#search`): pagina di ricerca campi;
  - Risorsa REST `users`, con azioni per la gestione del profilo utente e rotte annidate per visualizzare le prenotazioni, le recensioni e le segnalazioni relative all'utente corrente (es. `/users/:id/bookings`, `/users/:id/reviews`, `/users/:id/reports`);
  - Risorsa REST `bookings`, limitata alle azioni `index` e `destroy`, per consultare e annullare le prenotazioni effettuate.
- **Partner**:
  - `partner_sign_up`, `partner_log_in`, `partner_log_out`: rotte per la registrazione e autenticazione dei gestori;
  - `partner_dashboard`: dashboard di gestione dei propri campi e prenotazioni;
  - Risorsa REST `partners` per la creazione di nuovi account Partner;
  - Rotte per l'attivazione account (`partner_activations`) e il reset password (`partner_password_resets`);
  - `get '/partner_reviews'`: endpoint (AJAX) per ottenere le recensioni relative ai campi di un dato partner (utilizzato per popolare la dashboard del partner).

- **Amministratore**:
  - `administrator_sign_up`, `administrator_log_in`, `administrator_log_out`: rotte per la gestione dell'accesso amministratore;
  - `administrator_dashboard`: dashboard principale dell'area amministrativa;
  - `administrator_index` e risorsa REST `administrators` per elencare, creare e rimuovere account amministratore;
  - Rotte per visualizzare e aggiornare il profilo amministratore: `administrator_profile` e `administrator_update`.

5.3.3 Esempi di flussi nei controller

Alcuni controller implementano in modo emblematico i flussi funzionali progettati per la piattaforma:

- **FieldsController**: gestisce la creazione e modifica dei campi lato partner, nonché la ricerca dei campi lato utente. In particolare:
  - le azioni `new`, `create`, `edit`, `update`, `destroy` sono protette dal filtro `before_action :authenticate_partner!`, che garantisce che solo un partner autenticato possa creare o modificare i propri campi;
  - l'azione `search` implementa la ricerca geolocalizzata dei campi:
    * se il browser fornisce le coordinate geografiche dell'utente (latitudine e longitudine), queste vengono utilizzate direttamente come punto di partenza;
    * in caso contrario, l'indirizzo inserito dall'utente viene geocodificato (con fallback alla sola città se l'indirizzo è incompleto) e gli eventuali errori sono segnalati tramite messaggi flash;
    * i risultati vengono filtrati per sport (se specificato) e ordinati per distanza, utilizzando gli scope di prossimità offerti dalla gemma Geocoder (metodo `near`);
    * la data selezionata viene normalizzata a un oggetto `Date` (memorizzato come `@selected_date`) e riutilizzata per precompilare il campo data nel form di ricerca nella vista.
  - l'azione `show` si occupa di:
    * validare la data passata come parametro nella query string, con fallback alla data odierna in caso di formato non valido;

        * caricare gli Slot disponibili per il campo e la data selezionata, filtrando quelli con `booked = false` e ordinandoli per orario di inizio.
    - l'azione `reverse_geocode` espone un endpoint JSON che, dato un paio di coordinate (`lat`, `lon`), restituisce un indirizzo approssimativo. Questo servizio viene utilizzato dal pulsante "usa la mia posizione" dell'interfaccia utente per facilitare l'inserimento dell'indirizzo nel form di ricerca.

- **BookingsController**: gestisce la visualizzazione e la cancellazione delle prenotazioni lato utente.
    - Il filtro `before_action :require_user` garantisce che solo un utente autenticato possa accedere alle azioni di questo controller.
    - L'azione `index` recupera le prenotazioni dell'utente corrente, includendo anche gli Slot associati (eager loading) per ridurre il numero di query al database.
    - L'azione `destroy` implementa la logica di cancellazione con eventuale rimborso:
        * verifica che la prenotazione da cancellare appartenga all'utente corrente;
        * se l'orario di inizio dello Slot è oltre una certa soglia futura (es. più di 24 ore di anticipo), invoca `refund_stripe(@booking)` per effettuare un rimborso tramite le API di Stripe (`Stripe::Refund`) e poi distrugge la prenotazione, liberando lo slot (impostando `booked = false`);
        * se la prenotazione è già scaduta (slot nel passato), consente comunque la cancellazione senza rimborso, aggiornando coerentemente lo stato dello slot (lo slot viene liberato ma nessun rimborso è emesso).

- **ReviewsController**: gestisce la creazione e la visualizzazione delle recensioni sui campi.
    - L'azione `field_reviews` mostra tutte le recensioni relative a un dato campo, ordinate per data di creazione (dal più recente al meno recente).
    - L'azione `user_index` elenca le recensioni scritte dall'utente attualmente autenticato.
    - L'azione `partner_reviews` restituisce (tramite una *partial* HTML) le recensioni relative ai campi gestiti dal partner corrente, così da popolare la sezione delle recensioni nella dashboard del partner.

  - L'azione `create` crea una nuova recensione associandola al campo e all'utente corrente, applicando le validazioni definite nel model (ad esempio verifica che l'utente abbia almeno una prenotazione pregressa su quel campo).
  - L'azione `destroy` è pensata per la moderazione: elimina una recensione solo se l'attore corrente dispone dei permessi necessari (ad esempio, un amministratore può rimuovere recensioni inappropriate).

- **ReportsController**: realizza il flusso di gestione delle segnalazioni utente e la moderazione dei contenuti segnalati.
  - L'azione `index` elenca tutte le segnalazioni con stato *pending* (di norma visibili solo agli amministratori).
  - L'azione `user_index` mostra l'elenco delle segnalazioni inviate dall'utente corrente.
  - L'azione `create` registra una nuova segnalazione a partire da un oggetto segnalabile (ad esempio una specifica `Booking`), utilizzando un metodo ausiliario `find_reportable` per individuare l'elemento oggetto della segnalazione; la nuova `Report` viene salvata con stato iniziale *pending*.
  - Le azioni `accept` e `reject` aggiornano lo stato di una segnalazione esistente (impostandolo, rispettivamente, su *accepted* o *rejected*) e inviano una notifica email al segnalatore tramite `UserMailer`. Queste azioni producono una risposta sia in HTML sia in JavaScript, così da aggiornare dinamicamente la dashboard amministrativa dopo l'operazione.

- **AdministratorsController**: gestisce le funzionalità riservate all'area amministrativa.
  - L'azione `dashboard` carica i dati principali della piattaforma (numero di utenti registrati, campi pubblicati, prenotazioni effettuate, recensioni e segnalazioni presenti) e li rende disponibili alla vista di back-office.
  - Le azioni `new`, `create`, `index`, `destroy` consentono a un amministratore con ruolo *root* di creare, elencare e rimuovere altri account amministrativi.
  - I filtri `before_action :logged_in_administrator` e `:root_administrator` assicurano che solo amministratori autenticati (e, per determinate azioni sensibili, solo l'amministratore principale *root*) possano accedere alle relative funzionalità.

5.3.4 View e layout

La componente di *view* (template `.html.erb`) e il layout front-end sono organizzati in modo da garantire riuso e coerenza nell'aspetto grafico delle varie sezioni dell'applicazione. In particolare, si è cercato di:

- riutilizzare *partial* comuni (ad esempio le card di presentazione del campo, le card delle prenotazioni e i box delle recensioni) nelle diverse pagine dell'applicazione;
- adottare un layout principale condiviso per la navigazione generale del sito (header con logo *Campetto.it*, menu di navigazione, footer con link utili);
- differenziare l'aspetto delle dashboard riservate (partner e amministratore) tramite apposite classi CSS (ad esempio `body.partner-dashboard` e `body.admin-dashboard`), definendo per queste viste una struttura a griglia con un pannello laterale di navigazione e un'area di lavoro centrale per i contenuti.

5.4 Progettazione dei componenti chiave

All'interno dell'architettura descritta, alcuni componenti funzionali rivestono un ruolo centrale nell'implementazione dei requisiti della piattaforma. Di seguito si approfondiscono quelli principali.

5.4.1 Ricerca geolocalizzata e filtraggio

La funzionalità di ricerca dei campi combina più aspetti:

- **Geolocalizzazione**: l'utente può inserire un indirizzo di riferimento oppure utilizzare la propria posizione attuale; il controller `FieldsController#search` traduce questi input in coordinate geografiche, effettuando controlli sul formato dell'indirizzo e segnalando errori (ad esempio in caso di indirizzo incompleto o non riconosciuto).
- **Ricerca per prossimità**: una volta determinate le coordinate di riferimento, i campi vengono filtrati entro un raggio prefissato (ad esempio 30 km) e ordinati per distanza dal punto cercato, utilizzando gli scope di prossimità forniti da Geocoder (metodo `near`).
- **Filtri per sport e data**: il parametro di ricerca per sport restringe i risultati alla disciplina selezionata, mentre la data indicata viene utilizzata per verificare la presenza di Slot liberi per ciascun campo nel giorno richiesto.

È inoltre presente un endpoint dedicato, `reverse_geocode`, che fornisce il servizio inverso: dato un paio di coordinate (`lat`, `lon`), restituisce un indirizzo approssimativo in formato JSON. Questo endpoint viene utilizzato per precompilare automaticamente il campo di indirizzo nel form di ricerca quando l'utente opta per la funzionalità "usa la mia posizione" tramite il browser.

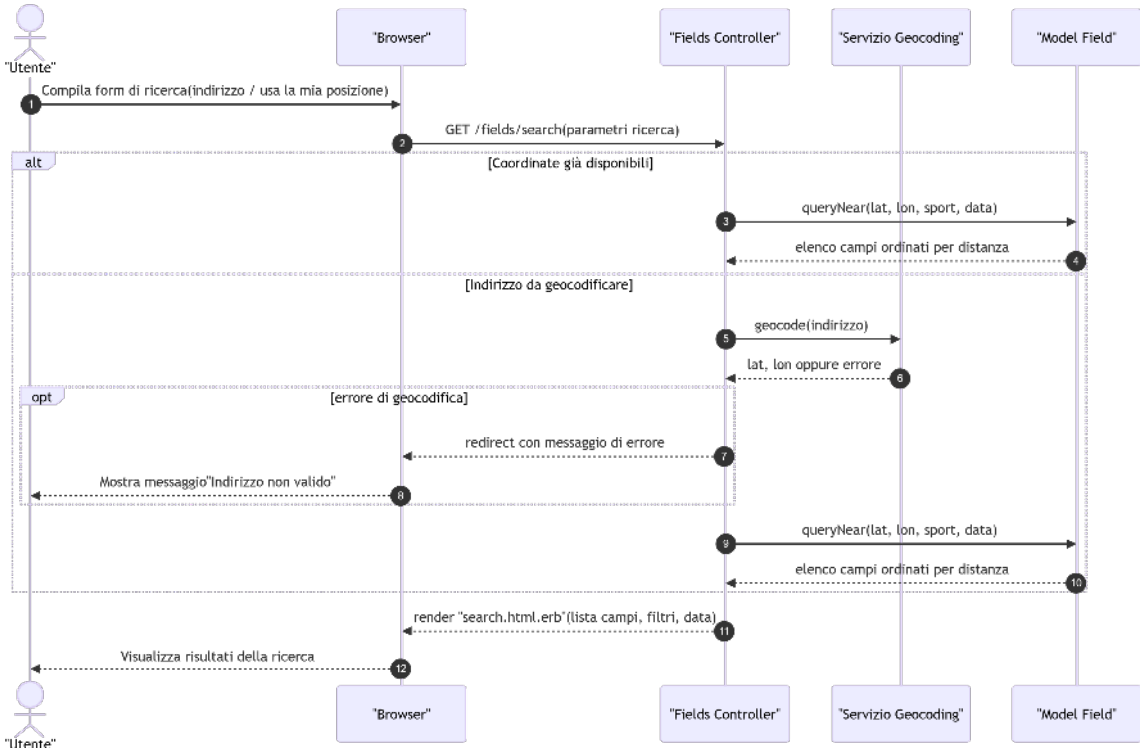


Figura 10: Diagramma di sequenza della ricerca geolocalizzata.

5.4.2 Generazione e gestione degli slot

La generazione delle disponibilità di prenotazione è progettata per essere automatica e controllabile. Alla creazione di un nuovo *Field*, un callback `after_create` invoca `Slot.generate_slots(self)`, il quale genera le fasce orarie prenotabili per un intervallo temporale prefissato (ad esempio i quattro mesi successivi) rispettando gli orari di apertura/chiusura del campo, la durata di ciascuno slot e i giorni esclusi definiti nella configurazione.

Dal lato utente, la pagina di dettaglio di un campo mostra per la data selezionata solo gli Slot futuri ancora disponibili (`booked = false`), ottenuti tramite una specifica query nel metodo `FieldsController#show`.

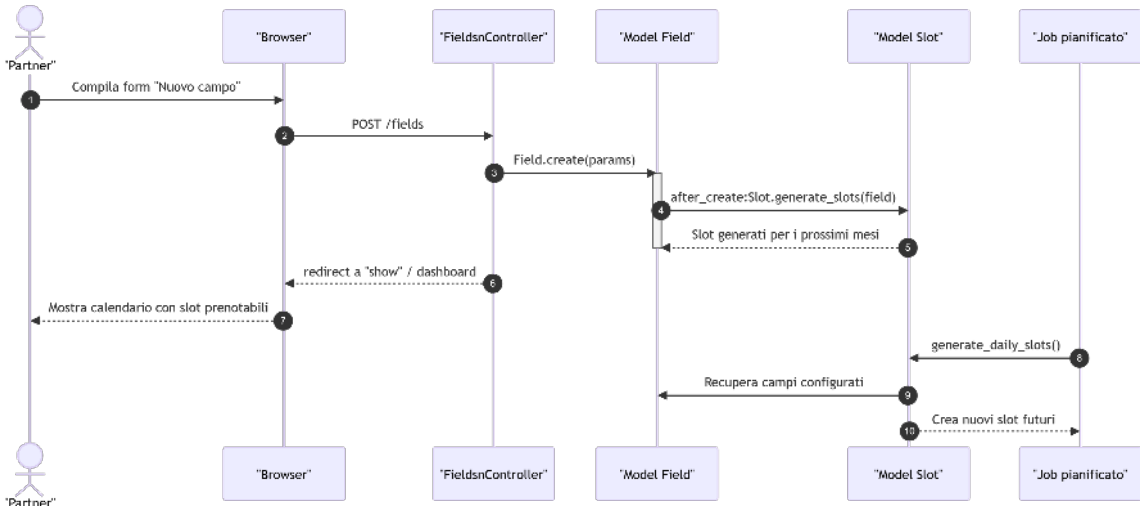


Figura 11: Diagramma di sequenza per la generazione e gestione degli slot.

5.4.3 Flusso di pagamento e cancellazione con Stripe

Il flusso di pagamento è stato realizzato delegando a Stripe la gestione sicura dei dati di carta di credito e dell'incasso. Quando l'utente conferma la prenotazione di uno slot, un controller dedicato al checkout crea un oggetto *PaymentIntent* tramite le API di Stripe (utilizzando la gemma `stripe`). L'utente viene reindirizzato alla pagina di pagamento ospitata sui server Stripe; al termine della procedura (con un redirect di successo o via webhook), il sistema salva l'identificativo del pagamento nella colonna `payment_intent_id` della tabella `bookings`. La *Booking* viene creata solo in caso di esito positivo del pagamento, e lo Slot associato viene marcato come occupato (`booked = true`). La cancellazione di una prenotazione già effettuata è gestita dal metodo `BookingsController#destroy` che, come descritto in precedenza, verifica la titolarità della prenotazione, eventualmente effettua il rimborso attraverso Stripe e libera lo slot (impostando nuovamente `booked = false`).

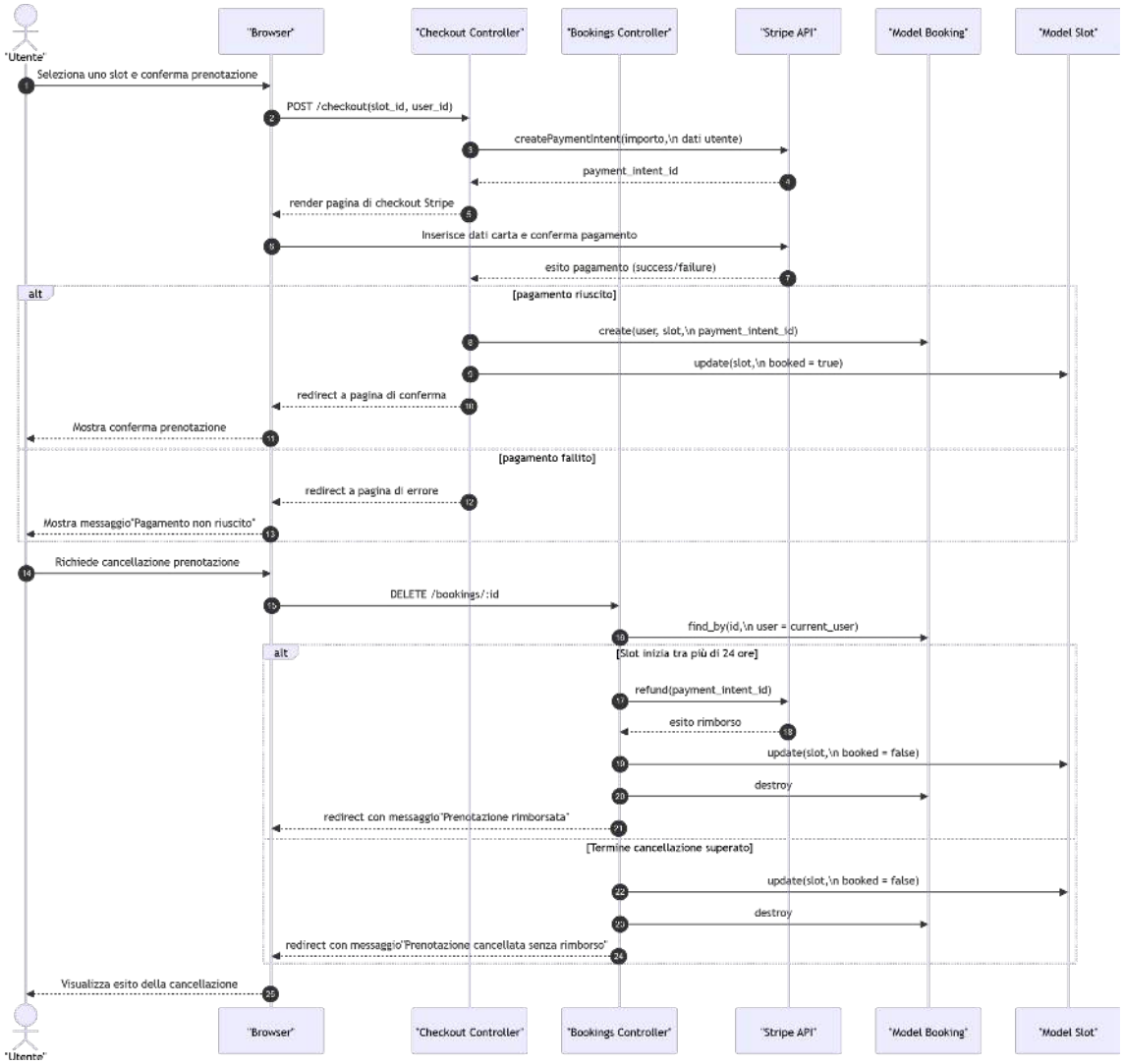


Figura 12: Diagramma di sequenza del flusso di pagamento e cancellazione con Stripe.

5.4.4 Sistema di recensioni e moderazione

La piattaforma include un sistema di recensioni per i campi sportivi ed un meccanismo integrato di segnalazione per la moderazione dei contenuti:

- Ogni *Review* collega un utente a un campo; il controller `ReviewsController` fornisce le azioni per visualizzare le recensioni disponibili e per consentire all'utente di creare nuove recensioni (rispettando il vincolo che l'utente abbia una prenotazione sul campo da recensire).
- Il modello `Report` consente di segnalare recensioni o prenotazioni problematiche:

– l'azione `ReportsController#create` riceve come parametri il tipo e l'identificativo dell'oggetto segnalato, costruisce un nuovo `Report` con stato iniziale *pending* e lo salva nel database;

– le azioni `ReportsController#index` e `ReportsController#accept/reject` sono utilizzate dall'amministratore per visualizzare le segnalazioni pendenti e risolverle, aggiornando lo stato (accettando o rifiutando la segnalazione) ed inviando notifiche email al *reporter* tramite `UserMailer`.

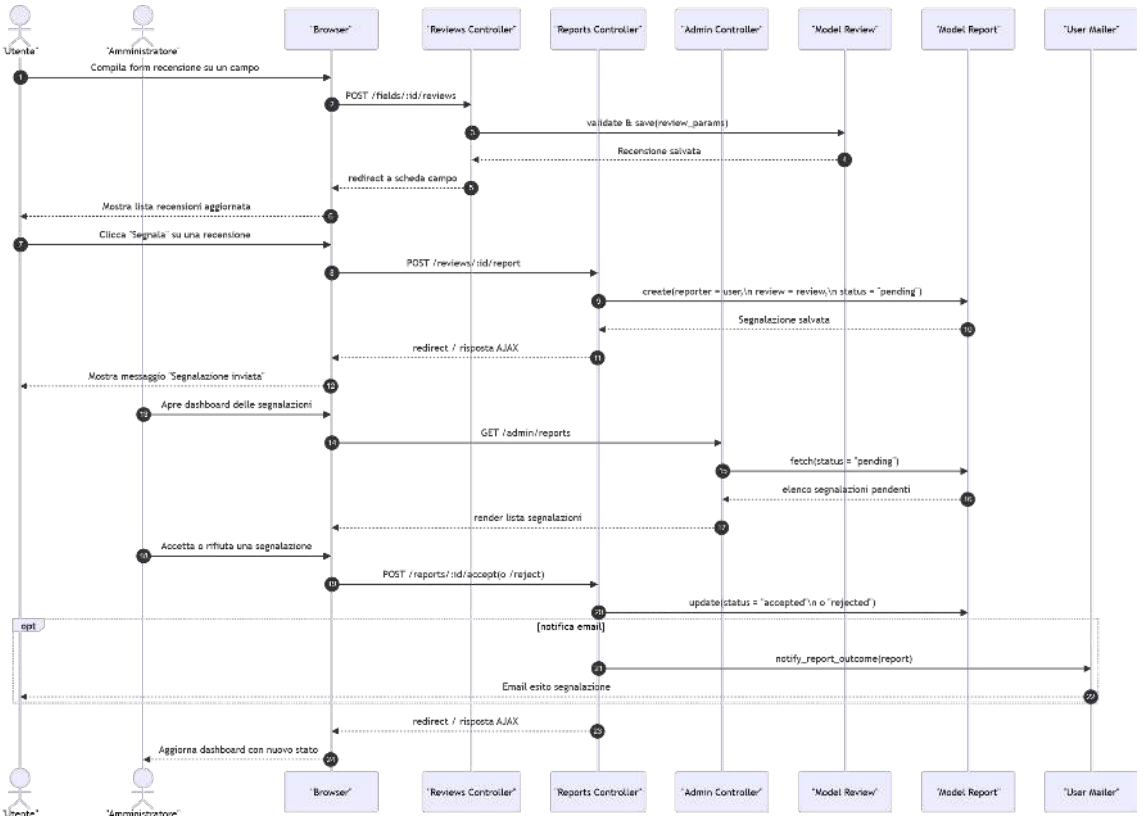


Figura 13: Diagramma di sequenza per recensioni, segnalazioni e moderazione.

6 Implementazione del sistema

6.1 Architettura applicativa

L'applicazione *Campetto.it* è stata realizzata seguendo il paradigma MVC (Model–View–Controller) proprio di Ruby on Rails, adottando una separazione chiara tra livello di dominio, logica applicativa e interfaccia utente. L'obiettivo di progetto è stato mantenere un'architettura coerente con il modello concettuale, estendibile nel tempo e facilmente manutenibile.

Il sistema utilizza PostgreSQL come database relazionale, ActiveStorage per la gestione delle immagini, ActionMailer per le notifiche email e Stripe Checkout per la gestione sicura dei pagamenti. Ogni funzionalità è stata incapsulata nel proprio controller: la ricerca dei campi è gestita dal *FieldsController*, il pagamento dal *CheckoutController*, le recensioni e le segnalazioni dai relativi moduli, mantenendo una struttura coerente con il dominio applicativo in cui i campi generano slot e gli slot danno luogo alle prenotazioni.

6.2 Ricerca e geolocalizzazione dei campi

La ricerca dei campi rappresenta una delle funzionalità principali del sistema, permettendo all'utente di identificare i campi disponibili nelle vicinanze filtrando per posizione, sport e data. La progettazione integra la geocodifica dell'indirizzo, l'interrogazione del database e la logica applicativa del controller.

Per ogni campo registrato, l'indirizzo (via, CAP, città) viene convertito in coordinate geografiche tramite la gemma `Geocoder` e salvato nel database. Questo consente di evitare chiamate esterne durante la fase di ricerca, che si limita a interrogare i dati già normalizzati.

Quando l'utente effettua una ricerca, il *FieldsController* recupera l'indirizzo o le coordinate del browser, applica la geocodifica se necessario e seleziona i campi appartenenti allo sport richiesto. Successivamente viene applicato il filtro di prossimità mediante il metodo `near`, basato sulla Haversine formula, ordinando i risultati per distanza crescente.
È previsto anche il *reverse geocoding* per ottenere automaticamente un indirizzo a partire dalla posizione corrente dell'utente.

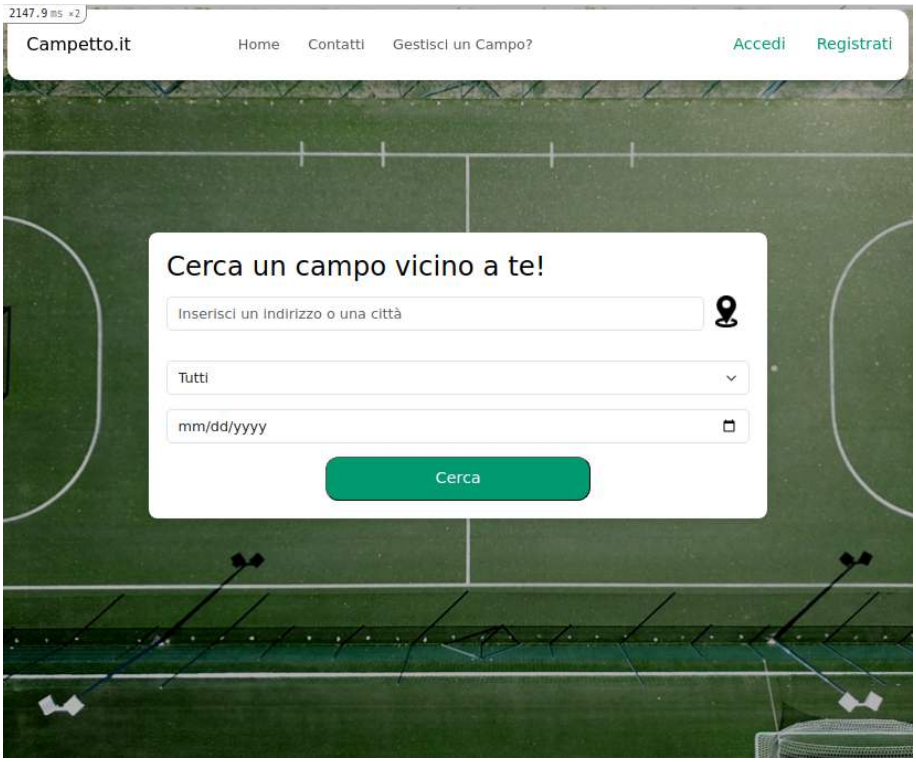


Figura 14: Pagina di ricerca e filtraggio dei campi

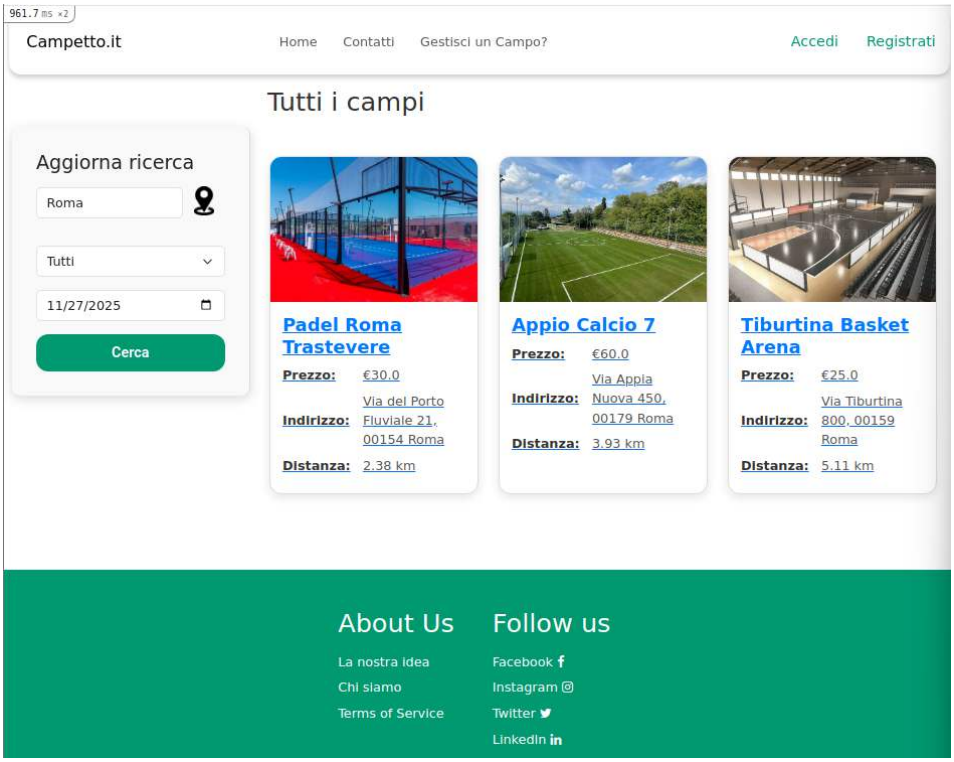


Figura 15: Pagina dei campi disponibili sulla piattaforma

6.3 Visualizzazione del campo e gestione degli slot

La pagina di dettaglio del campo rappresenta il punto di accesso alle disponibilità di prenotazione e raccoglie informazioni strutturate sul campo, sulle sue caratteristiche e sugli slot prenotabili.

Il controller gestisce in modo rigoroso la selezione della data: eventuali date non valide o antecedenti all'odierno vengono normalizzate, garantendo che l'utente visualizzi solo disponibilità future. Gli slot vengono quindi filtrati per data, disponibilità (`booked = false`) e ordinati temporalmente.

Ogni slot è presentato con orario, prezzo e un pulsante per avviare la prenotazione. Gli slot prenotati non vengono mostrati, evitando ambiguità nell'interfaccia. Il comportamento è coerente con la logica di generazione degli slot, che tiene conto dell'orario di apertura, chiusura, intervallo temporale e giorni esclusi.

Casi particolari come prenotazioni concorrenti, modifiche dei campi da parte del partner o date fuori intervallo sono stati trattati in modo da garantire coerenza applicativa ed evitare fenomeni di overbooking.

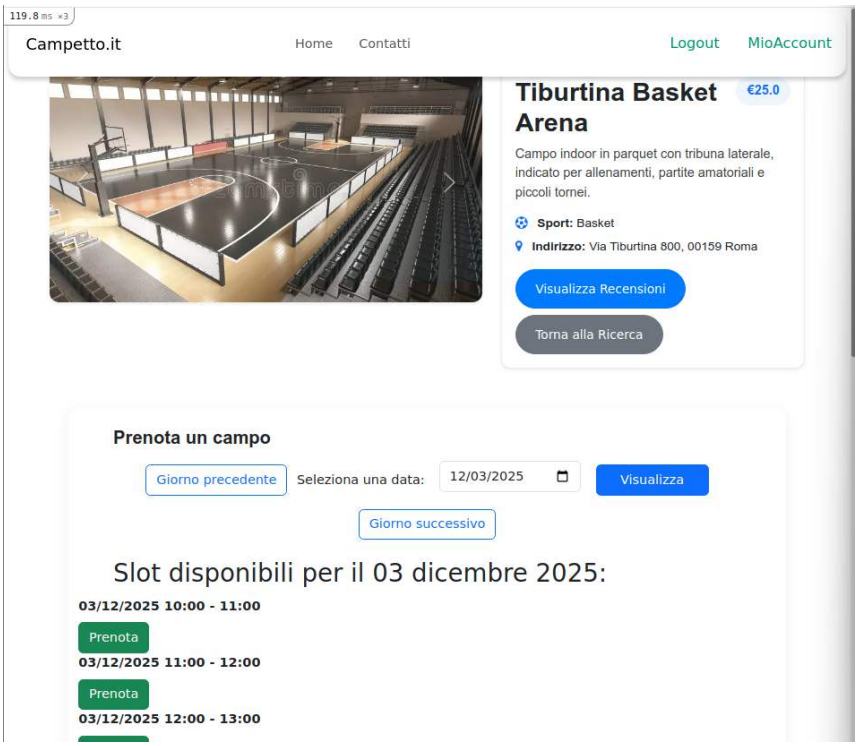
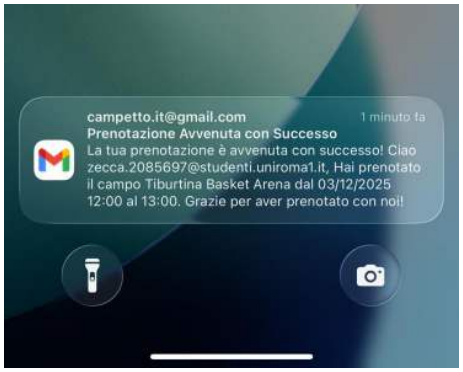


Figura 16: Pagina di dettaglio del campo

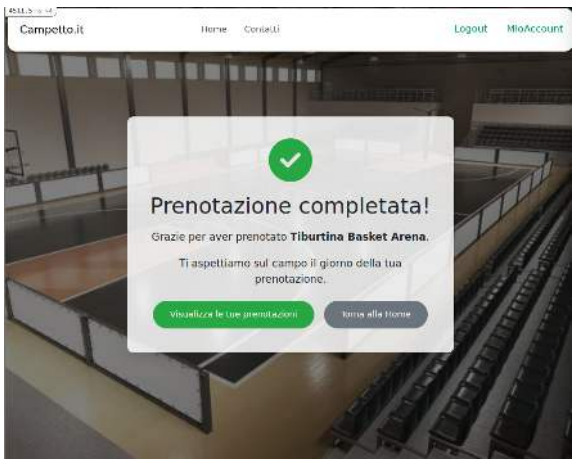
6.4 Flusso di prenotazione e pagamento online

La prenotazione avviene tramite un flusso integrato con Stripe Checkout, progettato per essere sicuro, lineare e aderente alle regole di dominio.
Quando un utente seleziona uno slot, il sistema crea una *Checkout Session* contenente l'importo, lo slot di riferimento e gli URL di ritorno. Lo slot non viene marcato come prenotato in questa fase, così da evitare blocchi per pagamenti abbandonati.
Una volta completato il pagamento, l'utente viene reindirizzato all'URL di successo: il sistema verifica lo stato del *PaymentIntent* e crea la prenotazione, segnando lo slot come occupato. Inoltre, vediamo come in FIGURA, al successo della prenotazione, l'utente riceve una mail di conferma.
Il sistema gestisce anche modifiche e cancellazioni. Se la prenotazione viene annullata con sufficiente anticipo, viene avviato un rimborso tramite `Stripe::Refund`, mentre cancellazioni tardive non prevedono rimborso.

(a) Pagamento tramite Stripe



(b) Notifica sul dispositivo mobile



(c) Pagina di conferma della prenotazione

Figura 17: Flusso di pagamento, notifica e conferma della prenotazione.

6.5 Sistema di recensioni e gestione delle segnalazioni

Il sistema di recensioni permette agli utenti di valutare i campi dopo una prenotazione. Ogni recensione è collegata a un utente e a un campo e contiene titolo, valutazione e testo descrittivo. L'inserimento è consentito soltanto agli utenti autenticati, mentre la visualizzazione è pubblica nella pagina del campo.

Le segnalazioni (*Report*) costituiscono un modulo leggero per denunciare contenuti inappropriati o problematiche legate alle prenotazioni. Il modello è polimorfico: una segnalazione può riferirsi a una *Review* o a una *Booking*.

6.6 Dashboard Partner: gestione campi e prenotazioni

La dashboard partner offre una vista completa sui campi gestiti e sulle relative prenotazioni. Ogni campo è mostrato tramite una card contenente immagine, descrizione e controlli di modifica o eliminazione. La cancellazione è gestita con vincoli di integrità che impediscono la rimozione di slot prenotati.
Una sezione specifica elenca tutte le prenotazioni dei campi del partner, mostrando data, fascia oraria, utente e prezzo. Il partner può annullare una prenotazione: in tal caso la booking viene rimossa e lo slot reso nuovamente disponibile, con eventuale gestione del rimborso.
La dashboard include anche una vista dedicata alle recensioni ricevute, utile per valutare la qualità percepita dagli utenti.

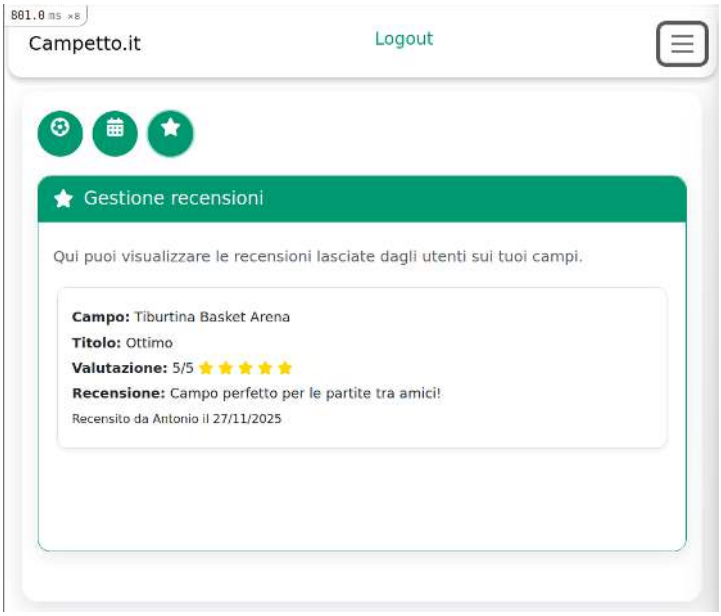


Figura 18: Dashboard del partner con le recensioni lasciate dagli utenti.

6.7 Dashboard Amministratore: gestione e moderazione

La dashboard amministrativa rappresenta il centro di controllo del sistema e raggruppa funzionalità di supervisione, moderazione e gestione degli attori.

L'interfaccia è organizzata tramite sezioni che elencano utenti, partner, campi, prenotazioni, recensioni e segnalazioni. Il caricamento avviene tramite richieste AJAX per mantenere fluida l'esperienza di navigazione.
Gli amministratori possono eliminare contenuti inappropriati, cancellare prenotazioni, gestire i campi e interagire con le segnalazioni. La sezione dedicata ai report consente di analizzare ogni segnalazione e di accettarla o respingerla; l'utente coinvolto viene avvisato tramite email generata da ActionMailer.

Un modulo dedicato permette la gestione degli account amministrativi, con creazione ed eliminazione di nuovi amministratori, mantenendo un chiaro controllo dei privilegi e delle responsabilità.

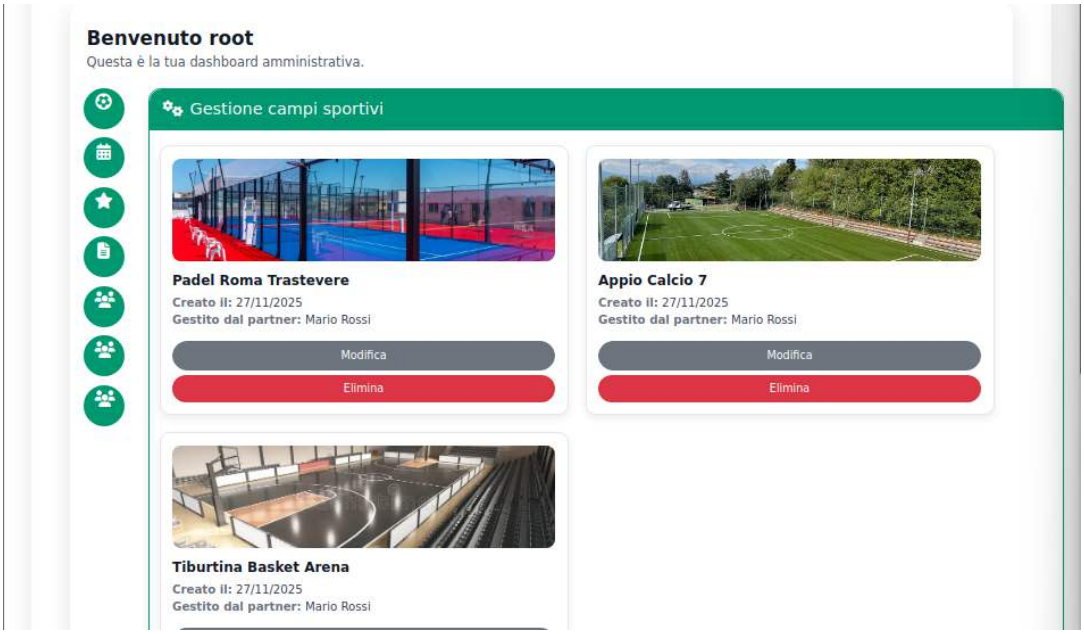


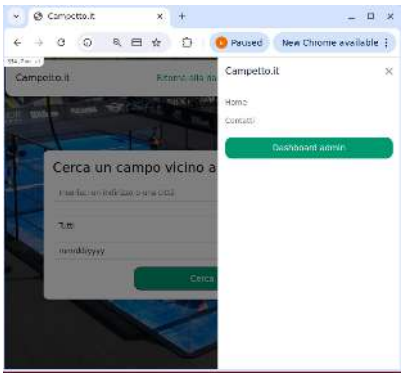
Figura 19: Dashboard amministrativa nella sezione dei campi sportivi disponibili in tutta la piattaforma.

6.8 Progettazione del front-end: struttura, UI/UX e responsive design

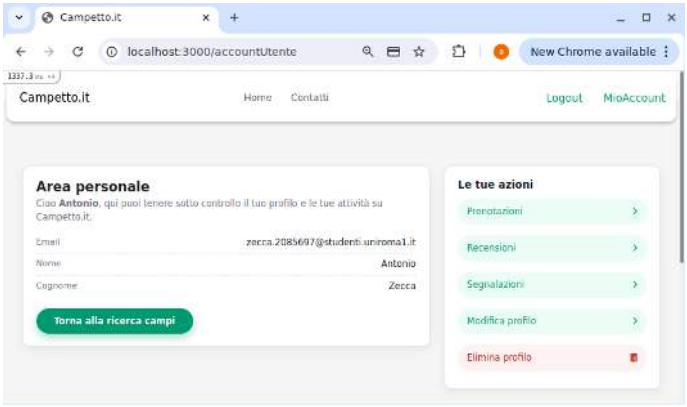
Il front-end combina template ERB, fogli di stile SCSS e componenti Bootstrap 5, ottenendo un'interfaccia moderna, uniforme e facilmente adattabile ai vari dispositivi. Le viste sono suddivise per domini logici (campi, prenotazioni, recensioni, report) e supportano componenti riutilizzabili come navbar, footer, card e modali.

La progettazione UI/UX si basa su tre principi: coerenza visiva, pulizia tipografica e riduzione del carico cognitivo. Il colore principale identifica il brand ed è applicato in pulsanti, badge e componenti interattivi, mentre la distribuzione dei contenuti privilegia spazi ampi e blocchi ben separati.

Il layout è completamente responsive grazie al sistema di griglie di Bootstrap. Le pagine ad alto volume informativo, come la ricerca dei campi e la dashboard amministrativa, possono riadattarsi dinamicamente per essere leggibili anche su schermi ridotti. Alcune funzionalità sono supportate da componenti JavaScript per migliorare l'esperienza utente: modali, geolocalizzazione, toggle di sezioni e caricamento asincrono nelle dashboard.



(a) Pagina di ricerca in versione responsiva



(b) Dashboard utente con finestra ridotta

Figura 20: Esempio di visualizzazione responsiva della pagina di ricerca e della dashboard utente.

6.9 Sicurezza, autenticazione e gestione dei ruoli

La sicurezza ha costituito un pilastro centrale nella progettazione del sistema. I modelli *User*, *Partner* e *Administrator* utilizzano `has_secure_password` per garantire un hashing sicuro delle password e evitare la memorizzazione in chiaro. Sono state inoltre applicate validazioni sulla complessità e meccanismi per limitare tentativi di accesso non autorizzato.

Il sistema prevede tre flussi di autenticazione separati:

- Utenti : `/login`
- Partner : `/partner_log_in`
- Amministratori : `/administrator_log_in`

Ciò previene escalation di privilegi e garantisce che ogni attore operi nel proprio perimetro.

L'autorizzazione è gestita tramite filtri *before_action* che verificano l'identità dell'utente e la proprietà delle risorse. Rails fornisce inoltre protezioni contro CSRF, XSS e SQL injection, integrate con validazioni lato server su tutti i modelli. Dati sensibili come informazioni di pagamento non transitano mai nel sistema, poiché gestiti interamente da Stripe. Le comunicazioni email, gestite da ActionMailer, sono inviate in modo asincrono e comprendono conferme di registrazione, esiti prenotazioni e aggiornamenti delle segnalazioni.

7 Validazione e test

In questo capitolo descrivo come ho affrontato la fase di validazione di *Campetto.it*, con particolare attenzione ai test automatizzati scritti in Cucumber. L'obiettivo è mostrare non solo *che cosa* viene testato, ma anche *come* funzionano i test, qual è la strategia complessiva che ho seguito e in che modo questi scenari diventano una documentazione eseguibile del comportamento atteso del sistema.

7.1 Obiettivi e funzionamento

L'applicazione copre diversi flussi critici: la ricerca dei campi a partire da un indirizzo, la gestione dei campi da parte dei partner, la supervisione della piattaforma da parte dell'amministratore e l'intero processo di prenotazione.

Nel concreto, i test funzionano così:

1. Definisco uno **scenario** in linguaggio naturale, usando le parole chiave `Funzionalità`, `Scenario`, `Dato`, `Quando`, `Allora`.

2. Per ciascun passo (`Dato`/`Quando`/`Allora`) collego un pezzo di codice Ruby, chiamato *step definition*, che prepara il contesto, esegue l'azione o verifica il risultato.

3. Quando lancio i test, Cucumber:

   (a) prepara il database di test (usando l'ambiente `test` di Rails);

   (b) esegue gli step in ordine;

   (c) controlla che le asserzioni sugli elementi della pagina e sui dati siano soddisfatte.

In questo modo ogni scenario diventa una sorta di "storia" di utilizzo dell'applicazione: ad esempio, un partner che accede alla propria dashboard e vede i suoi campi oppure un amministratore che accede alla dashboard amministrativa e visualizza un riepilogo delle prenotazioni.
Tutti gli scenari Cucumber descritti in questo capitolo sono eseguiti sull'ambiente test di Rails. Dopo aver inizializzato il database di test con: `rails db:test:prepare` la suite di test può essere lanciata con il comando: `bundle exec cucumber` Questo comando esegue tutti i file .feature presenti nella cartella features/ e produce un report testuale in console.

7.1.1 Strategia di test adottata

Dal punto di vista teorico, la strategia che ho seguito è quella di concentrarmi principalmente su test di tipo *end-to-end* o *acceptance*, scritti in chiave BDD (Behaviour Driven Development).

Per la scrittura dei test di accettazione ho utilizzato principalmente **Cucumber**, un framework che permette di descrivere il comportamento dell'applicazione in un linguaggio molto vicino all'italiano.
Ogni gruppo di scenari è raccolto in un file con estensione `.feature`. All'interno di questi file definisco:

- una `Funzionalità`, che descrive in poche righe lo scopo generale;
- uno o più `Scenario`, che rappresentano casi d'uso specifici;
- gli step `Dato`, `Quando`, `Allora`, che descrivono rispettivamente il contesto iniziale (dati presenti a database, utente loggato, ecc.), l'azione che l'utente compie (ad esempio visita una pagina o clicca un pulsante) e il risultato che mi aspetto di vedere (testo, elementi in pagina, riepiloghi, ecc.).

Dal lato Ruby, per ogni frase testuale definita nel file `.feature` esiste una *step definition* nella cartella `features/step_definitions/`.

7.2 Esempi di scenari di test

7.2.1 Test sulle dashboard del partner

Lo scenario che ho definito per questo caso verifica che un partner correttamente autenticato possa accedere alla propria dashboard e vedere elencati i campi che gestisce. In particolare lo scenario:

- crea un partner di test con una email e una password note;
- crea un campo associato a quel partner (ad esempio *Campo Padel Partner*);
- autentica il partner al sito;
- visita la dashboard del partner;
- verifica che nella pagina siano presenti:
  - il testo che identifica la gestione dei campi e nome del campo.

Nel file `dashboard_partner.feature` questo scenario è descritto in linguaggio naturale nel seguente modo:

Listing 1: Test - Dashboard partner

```
Funzionalità: Dashboard partner
  Per gestire i miei campi
  Come partner autenticato
  Voglio poter vedere nella dashboard i campi che gestisco

  Scenario: Il partner accede alla dashboard e vede il proprio campo
    Dato esiste un partner con email "partner@example.com" e password "
        Password123!"
    E esiste un campo "Campo␣Padel␣Partner" associato al partner "
        partner@example.com"
    E sono un partner autenticato con email "partner@example.com" e
        password "Password123!"
    Quando visito la dashboard del partner
    Allora dovrei vedere il testo "Gestione␣campi␣sportivi" nella pagina
    E dovrei vedere il campo "Campo␣Padel␣Partner" nella lista dei campi
        del partner
```

Le corrispondenti *step definition* in Ruby si occupano di:

- creare il record `Partner` rispettando le validazioni dell'applicazione (nome, cognome, data di nascita, telefono, ecc.);
- creare il record `Field` collegato al partner (con nome, sport, prezzo, indirizzo e orari di apertura);
- autenticare il partner di test, simulando una sessione già aperta;
- visitare il percorso `partner_dashboard_path` e controllare che l'HTML contenga il testo di gestione dei campi e il nome del campo creato.

Il file `dashboard_steps.rb` contiene le step definition utilizzate sia per la dashboard del partner sia per quella dell'amministratore. In particolare, le prime definizioni riguardano le operazioni del partner (creazione del campo, accesso alla dashboard, visualizzazione dei campi gestiti), mentre le successive sono dedicate alle funzionalità di amministrazione (gestione dei partner, moderazione e verifica degli account):

Listing 2: Estratto delle step definition per la dashboard del partner

```ruby
# features/step_definitions/dashboard_steps.rb

# ======================
# PARTNER
# ======================

Dato('esiste un partner con email {string} e password {string}') do |email, password|
  @partner = Partner.create!(
    name: 'Partner',
    surname: 'Test',
    gender: 'm',
    mobile: '3330000000', # 10 cifre valide
    birthdate: 25.years.ago.to_date, # maggiorenne
    email: email,
    password: password,
    password_confirmation: password
  )
end

Dato('esiste un campo {string} associato al partner {string}') do |nome_campo, email_partner|
  partner = Partner.find_by!(email: email_partner)

  Field.create!(
    nome: nome_campo,
    descrizione: 'Campo di prova per BDD',
    sport: 'Padel',
    prezzo: 20,
    via: 'Via Test 123',
    cap: '00100',
    citta: 'Roma',
    start_time: Time.current.change(hour: 8, min: 0),
    end_time: Time.current.change(hour: 23, min: 0),
    interval: 60,
    exclude_days: [],
    partner: partner
  )
end
```

```
38
39 Dato('sono un partner autenticato con email {string} e password {string}')
       do |email, _password|
40   partner = Partner.find_by!(email: email)
41
42   # Simuliamo che il partner sia già loggato:
43   allow_any_instance_of(ApplicationController)
44     .to receive(:current_partner)
45     .and_return(partner)
46
47   if ApplicationController.method_defined?(:partner_logged_in?)
48     allow_any_instance_of(ApplicationController)
49       .to receive(:partner_logged_in?)
50       .and_return(true)
51   end
52 end
53
54 Quando('visito la dashboard del partner') do
55   visit partner_dashboard_path
56 end
57
58 Allora('dovrei vedere il testo {string} nella pagina') do |testo|
59   # Controllo che il testo sia presente nell'HTML generato,
60   expect(page.html).to include(testo)
61 end
62
63 E('dovrei vedere il campo {string} nella lista dei campi del partner') do
       |nome_campo|
64   # Controllo che il nome del campo sia presente nell'HTML della pagina,
65   expect(page.html).to include(nome_campo)
66 end
```

Listing 3: Estratto delle step definition per la dashboard dell'amministratore

```ruby
# features/step_definitions/dashboard_steps.rb

# ======================
# PARTNER
# ======================

Dato('esiste un partner con email {string} e password {string}') do |email, password|
  @partner = Partner.create!(
    name: 'Partner',
    surname: 'Test',
    gender: 'm',
    mobile: '3330000000', # 10 cifre valide
    birthdate: 25.years.ago.to_date, # maggiorenne
    email: email,
    password: password,
    password_confirmation: password
  )
end

Dato('esiste un campo {string} associato al partner {string}') do |nome_campo, email_partner|
  partner = Partner.find_by!(email: email_partner)

  Field.create!(
    nome: nome_campo,
    descrizione: 'Campo di prova per BDD',
    sport: 'Padel',
    prezzo: 20,
    via: 'Via Test 123',
    cap: '00100',
    citta: 'Roma',
    start_time: Time.current.change(hour: 8, min: 0),
    end_time: Time.current.change(hour: 23, min: 0),
    interval: 60,
    exclude_days: [],
    partner: partner
  )
end
```

```
38
39 Dato('sono un partner autenticato con email {string} e password {string}')
       do |email, _password|
40   partner = Partner.find_by!(email: email)
41
42   # Simuliamo che il partner sia già loggato:
43   allow_any_instance_of(ApplicationController)
44     .to receive(:current_partner)
45     .and_return(partner)
46
47   if ApplicationController.method_defined?(:partner_logged_in?)
48     allow_any_instance_of(ApplicationController)
49       .to receive(:partner_logged_in?)
50       .and_return(true)
51   end
52 end
53
54 Quando('visito la dashboard del partner') do
55   visit partner_dashboard_path
56 end
57
58 Allora('dovrei vedere il testo {string} nella pagina') do |testo|
59   # Controllo che il testo sia presente nell'HTML generato,
60   expect(page.html).to include(testo)
61 end
62
63 E('dovrei vedere il campo {string} nella lista dei campi del partner') do
       |nome_campo|
64   # Controllo che il nome del campo sia presente nell'HTML della pagina,
65   expect(page.html).to include(nome_campo)
66 end
```

Dal punto di vista del risultato atteso, quando eseguo questo scenario mi aspetto che Cucumber segnali il test come superato. La Figura 21 mostra un estratto dell'output della console con lo scenario "Dashboard partner" eseguito con esito positivo.

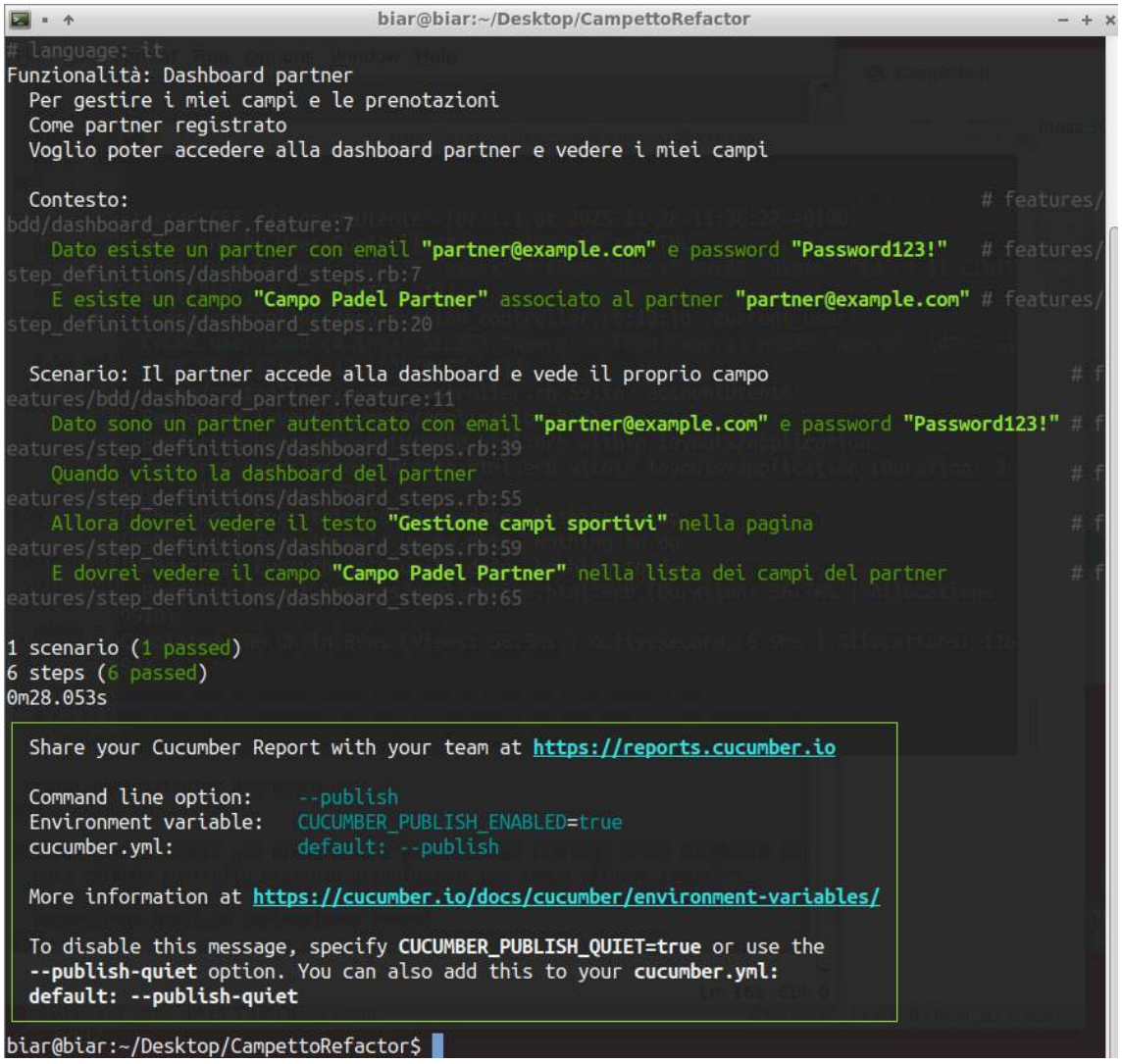


Figura 21: Output della console con lo scenario "Dashboard partner" eseguito con successo.

7.2.2 Test dal lato utente

Dal punto di vista dell'utente finale ho scelto di presentare tre scenari che, nel loro complesso, rappresentano il flusso principale di utilizzo di *Campetto.it*:

- la **ricerca dei campi** a partire da un indirizzo e da un filtro per sport e data;
- la visualizzazione degli **slot disponibili** per un determinato campo e una certa data;
- la creazione e la conferma di una **prenotazione** fino alla pagina di riepilogo.

Nel primo scenario, l'utente inserisce un indirizzo, seleziona uno sport e una data, e si aspetta di vedere un elenco di campi ordinati per distanza, con eventuali messaggi di errore se l'indirizzo non è valido. Nel file `search_fields.feature` questo comportamento è descritto con uno scenario del tipo:

Listing 4: Test - ricerca campi

```
Feature: Ricerca dei campi
  Per trovare rapidamente un campo vicino
  Come utente finale
  Voglio poter cercare i campi per indirizzo e sport

  Background:
    Given esiste il campo di padel "Padel␣Leverano" in "Via␣Giovanni␣
        Gronchi,␣73045␣Leverano" al prezzo di "10"

  Scenario: Ricerca con indirizzo valido e sport corretto
    Given sono nella pagina di ricerca dei campi
    When inserisco "Leverano" come indirizzo
    And seleziono "Padel" come sport
    And clicco su "Cerca"
    Then dovrei vedere il campo "Padel␣Leverano"

  Scenario: Ricerca con città in cui non ci sono campi
    Given sono nella pagina di ricerca dei campi
    When inserisco "Milano" come indirizzo
    And seleziono "Padel" come sport
    And clicco su "Cerca"
    Then dovrei vedere un messaggio che indica che non ci sono campi
        disponibili
```

Listing 5: Step definition - ricerca campi

```ruby
# features/step_definitions/ricerca_campi_step.rb

Given('esiste il campo di padel {string} in {string} al prezzo di {string}') do |nome_campo, indirizzo_completo, prezzo|
  # Cerco se esiste già nel DB di test
  field = Field.find_or_initialize_by(nome: nome_campo)

  field.assign_attributes(
    descrizione: 'Campo di padel',
    sport: 'Padel',
    prezzo: prezzo.to_f,
    latitudine: 0.0,
    longitudine: 0.0,
    start_time: Time.zone.now.change(hour: 10, min: 0),
    end_time: Time.zone.now.change(hour: 22, min: 0),
    interval: 60,
    exclude_days: nil,
    indirizzo: indirizzo_completo,
    via: 'Via Giovanni Gronchi',
    citta: 'Leverano',
    cap: '73045'
  )

end


Given('sono nella pagina di ricerca dei campi') do
  # Qui assumo che la ricerca sia nella home (root_path)
  visit root_path
end

When('inserisco {string} come indirizzo') do |indirizzo|
  fill_in 'indirizzo', with: indirizzo
end

When('seleziono {string} come sport') do |sport|
  select sport, from: 'sport'
end

```

```
When('clicco su {string}') do |text|
  if page.has_button?(text)
    click_button text
  elsif page.has_link?(text)
    click_link text
  else
    # fallback: qualunque elemento cliccabile con quel testo
    click_on text
  end
end


Then('dovrei vedere il campo {string}') do |_nome_campo|
  expect(page).to have_content('Campi per Padel')
end

Then('dovrei vedere un messaggio che indica che non ci sono campi
    disponibili') do

  expect(page).to have_content('Campi per Padel')

end
```

Un secondo scenario riguarda la prenotazione di un campo da parte di un utente: una volta autenticato e accertato che esiste il campo, l'utente cerca il campo e prenota lo slot. Un ultimo scenario mostra infine la valutazione che l'utente vuole lasciare, valutando la corretta implementazione delle recensioni.
In ambiente di test non viene effettuata una chiamata reale a Stripe, ma si verifica che, al termine del flusso, esista una `Booking` associata all'utente e allo slot e che la pagina mostri un messaggio di conferma.

I prossimi file raccoglono un estratto dei file relativi a questi due scenari, la Figura 22a, Figura 22b e la Figura 22c mostrano l'output Cucumber con tutti gli scenari utente eseguiti con esito positivo.

Listing 6: Test - prenotazione

```
#language: it
Funzionalità: Prenotazione di un campo
  Per poter giocare
  Come utente registrato
  Voglio poter cercare un campo e prenotare uno slot disponibile

   Contesto:
    Dato esiste il campo di padel "Padel Leverano" in "Via Giovanni Gronchi, 73045 Leverano" al prezzo di "10"
    E esiste uno slot disponibile per "Padel Leverano" il "2024-12-02" alle "18:00"
    E sono un utente registrato e autenticato con email "utente@example.com" e password "Password123!"

  Scenario: L utente cerca il campo e prenota uno slot
    Dato sono nella pagina di ricerca dei campi
    Quando inserisco "Leverano" come indirizzo
    E seleziono "Padel" come sport
    E clicco su "Cerca"
    Allora dovrei vedere il campo "Padel Leverano"

    Quando seleziono il campo "Padel Leverano"
    E seleziono lo slot "02/12/2024 18:00 - 19:00"
    E confermo la prenotazione
```

Listing 7: Estratto delle step definition per prenotazione da parte di un utente

```
# features/step_definitions/prenotazione_e_recensioni_steps.rb

# ==========================
# SLOT E PRENOTAZIONI
# ==========================

Given('esiste uno slot disponibile per {string} il {string} alle {string}') do |nome_campo, data, ora|
  field = Field.find_by!(nome: nome_campo)

  start_time = Time.zone.parse("#{data} #{ora}") rescue Time.parse("#{data} #{ora}")
  end_time = start_time + 1.hour
```

```
12
13    Slot.create!(
14      field: field,
15      start_time: start_time,
16      end_time: end_time,
17      booked: false
18    )
19  end
20
21  Given('sono un utente registrato e autenticato con email {string} e
        password {string}') do |email, password|
22    user = User.find_or_initialize_by(email: email)
23
24    if user.new_record?
25      user.password = password
26      user.password_confirmation = password
27      user.nome ||= 'Test'
28      user.cognome ||= 'User'
29      user.confirmed_at = Time.current if user.respond_to?(:confirmed_at)
30      user.save!
31    end
32
33    @current_user = user
34
35   (rack_session_access)
36    page.set_rack_session(user_id: user.id)
37  end
38
39  Given(esiste una prenotazione confermata per {string} da parte dell utente
        {string}) do |nome_campo, email|
40    field = Field.find_by!(nome: nome_campo)
41
42    user = User.find_or_create_by!(email: email) do |u|
43      u.password = 'Password123!'
44      u.password_confirmation = 'Password123!'
45      u.nome = 'Utente' if u.respond_to?(:nome)
46      u.cognome = 'Test' if u.respond_to?(:cognome)
47      u.confirmed_at = Time.current if u.respond_to?(:confirmed_at)
48    end
49
```

```
50    start_time = Time.zone.now + 2.days
51    end_time = start_time + 1.hour
52
53    slot = Slot.create!(
54      field: field,
55      start_time: start_time,
56      end_time: end_time,
57      booked: true
58    )
59
60    Booking.create!(
61      user: user,
62      slot: slot,
63      payment_intent_id: 'test_intent'
64    )
65  end
66
67  Given('mi␣trovo␣nella␣pagina␣del␣campo␣{string}') do |nome_campo|
68    @field = Field.find_by!(nome: nome_campo)
69    visit field_path(@field)
70  end
71
72  When('seleziono␣il␣campo␣{string}') do |nome_campo|
73    field = Field.find_by!(nome: nome_campo)
74    visit field_path(field)
75  end
76
77  When('seleziono␣lo␣slot␣{string}') do |slot_label|
78    if page.has_button?(slot_label)
79      click_button slot_label
80    elsif page.has_link?(slot_label)
81      click_link slot_label
82
83    elsif page.has_button?('Prenota')
84      first(:button, 'Prenota').click
85    elsif page.has_link?('Prenota')
86      first(:link, 'Prenota').click
87    else
88      first('button,␣a').click
89    end
```

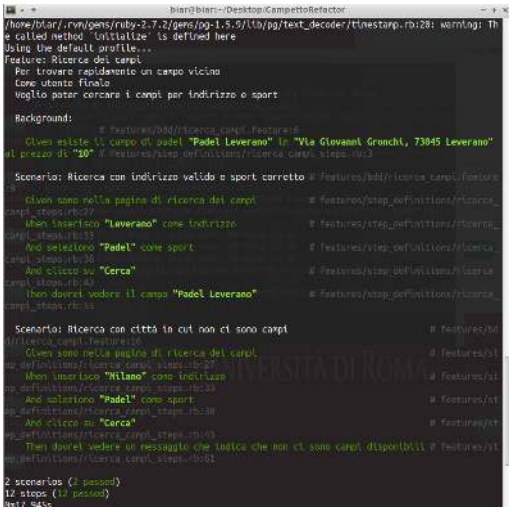
```
90  end
91
92  When('confermo la prenotazione') do
93    if page.has_button?('Conferma prenotazione')
94      click_button 'Conferma prenotazione'
95    elsif page.has_button?('Prenota')
96      click_button 'Prenota'
97    end
98  end
99
100 Then('dovrei vedere un messaggio di conferma della prenotazione') do
101   expect(page).to have_content('Prenotazione')
102 end
```

Listing 8: Test - review

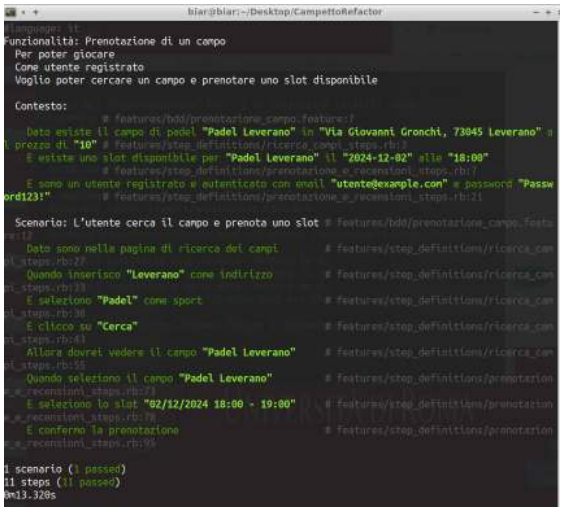
```
1  Funzionalità: Recensioni dei campi
2    Per aiutare altri utenti a scegliere dove giocare
3    Come utente registrato
4    Voglio poter lasciare una recensione su un campo che ho prenotato
5
6    Contesto:
7      Dato esiste il campo di padel "Padel Leverano" in "Via Giovanni
         Gronchi, 73045 Leverano" al prezzo di "10"
8      E esiste una prenotazione confermata per Padel Leverano da parte dell
         utente 'utente@example.com'
9      E sono un utente registrato e autenticato con email "utente@example.
         com" e password "Password123!"
10
11   Scenario: L utente aggiunge una recensione
12     Dato mi trovo nella pagina del campo "Padel Leverano"
13     Quando inserisco "Ottimo campo, illuminazione perfetta" come testo
         della recensione
14     E seleziono un voto pari a "5"
15     E invio la recensione
16     Allora dovrei vedere la recensione "Ottimo campo, illuminazione
         perfetta"
17     E dovrei vedere il voto "5/5"
```

Listing 9: Estratto delle step definition per recensione da parte di un utente

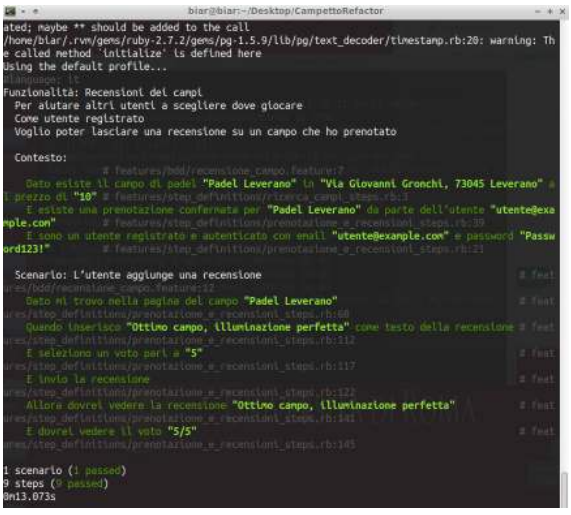
```
When('inserisco {string} come testo della recensione') do |testo|
  @review_text = testo
end

When('seleziono un voto pari a {string}') do |voto|

  @review_vote = voto.to_i
end

When('invio la recensione') do
  raise 'Campo non impostato' unless @field
  raise 'Utente non impostato' unless @current_user

  voto = @review_vote || 5

  Review.create!(
    field: @field,
    user: @current_user,
    titolo: "Recensione automatica",
    testo: @review_text,
    valutazione: voto
  )

  visit field_reviews_path(@field)
end

Then('dovrei vedere la recensione {string}') do |testo|
  expect(page).to have_content(testo)
end

Then('dovrei vedere il voto {string}') do |voto|

  numero_stelle = voto.split('/').first.to_i

  within '.reviews-list' do
    expect(page).to have_css('i.fas.fa-star', minimum: numero_stelle)
  end
end
```



(a) Output del test Cucumber relativo alla ricerca dei campi da parte dell'utente.



(b) Output del test Cucumber relativo alla conferma di prenotazione.



(c) Output del test Cucumber relativo all'inserimento di una recensione.

Figura 22: Esecuzione con esito positivo dei test Cucumber per ricerca, prenotazione e recensioni lato utente.

8 Conclusioni e sviluppi futuri

8.1 Conclusioni

Il lavoro svolto per lo sviluppo di Campetto.it ha permesso di costruire una piattaforma completa, capace non solo di digitalizzare il processo di prenotazione dei campi sportivi, ma anche di proporre un modello più moderno, ordinato e fruibile per utenti, gestori e amministratori.

Dal punto di vista personale e formativo, le parti che hanno richiesto maggiore approfondimento, e che hanno rappresentato la crescita più significativa, riguardano l'integrazione tra geocodifica, filtri di ricerca e logica di backend. La realizzazione di un modulo robusto, tollerante agli errori dell'utente e capace di restituire risultati coerenti, ha richiesto un lavoro iterativo e approfondito, che ha migliorato la comprensione del funzionamento interno di Rails e delle dinamiche dell'integrazione con servizi esterni.
Per quanto riguarda l'esperienza di sviluppo front-end, la cura dell'interfaccia, delle card, dei layout responsivi e della coerenza grafica ha ulteriormente rafforzato il rapporto tra forma e funzionalità.
Naturalmente, diverse sfide hanno accompagnato il percorso. La gestione della geocodifica e delle API esterne sono state spesso soggette a limiti tecnici o di configurazione. Tuttavia, la risoluzione progressiva di tali problemi ha contribuito alla maturazione del progetto e alla stabilità del sistema.
Nel complesso, il lavoro sviluppato mostra come sia possibile digitalizzare un processo ancora spesso frammentato, creando uno strumento moderno, affidabile e potenzialmente utile alla comunità sportiva locale.

8.2 Sviluppi futuri

Il progetto offre numerose possibilità di estensione.
Un primo sviluppo riguarda la gestione avanzata delle disponibilità. In molti sport, come il padel o il calcio a 5, l'organizzazione delle partite richiede di riunire più giocatori. Oggi questa dinamica avviene soprattutto in gruppi WhatsApp gestiti dagli istruttori o dai gestori dei centri sportivi. Una direzione futura consiste quindi nel digitalizzare questo processo: consentire agli utenti di segnalare la propria disponibilità per uno specifico slot, creare gruppi di gioco, confermare la prenotazione solo al raggiungimento del numero minimo di partecipanti e, più in generale, rendere l'organizzazione degli incontri più semplice e immediata.

Ulteriori sviluppi potrebbero trasformare la piattaforma in un vero social sportivo: chat interne, notifiche, spazi dedicati agli istruttori e strumenti per favorire l'incontro tra utenti che praticano gli stessi sport. Tali funzionalità amplierebbero la dimensione comunitaria dell'applicazione.
Dal punto di vista tecnico, il sistema potrebbe beneficiare di ulteriori miglioramenti strutturali: affinamento delle logiche di ricerca, arricchimento dei filtri (copertura, illuminazione, pavimentazione, servizi aggiuntivi), potenziamento dell'affidabilità del backend e maggiore granularità nella gestione degli slot. Un passaggio importante sarebbe lo sviluppo di un'app mobile nativa, in grado di offrire un uso più immediato del servizio e notifiche in tempo reale.

8.3 Considerazioni finali

Campetto.it nasce come progetto accademico, ma rappresenta una concreta opportunità di crescita per riflettere su come digitalizzare un settore ancora poco strutturato. Il percorso dimostra come una buona progettazione e un approccio iterativo possano trasformare un'idea in un'applicazione completa, utile e potenzialmente scalabile.

Bibliografia

[1] L. Querzoni. *Dispense del corso "Laboratorio di Architetture Software e Sicurezza Informatica"*. Sapienza Università di Roma, a.a. 2022/2023.

[2] D. Calvanese, G. De Giacomo, M. Lenzerini. *Dispense del corso "Basi di Dati"*. Sapienza Università di Roma.

[3] R. Ramakrishnan, J. Gehrke. *Sistemi di basi di dati*. McGraw–Hill, 2004.

[4] G. De Giacomo. *Dispense del corso "Progettazione del Software"*. Sapienza Università di Roma.

[5] M. Hartl. *Ruby on Rails Tutorial (Rails 5): Learn Web Development with Rails*. 4a edizione, Addison–Wesley, 2016.

[6] S. Ruby, D. Thomas, D. Hansson. *Agile Web Development with Rails 5*. Pragmatic Bookshelf, 2017.

[7] D. Chelimsky et al. *The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber, and Friends*. Pragmatic Bookshelf, 2010.

[8] Ruby on Rails Guides. *Documentazione ufficiale di Ruby on Rails*. Disponibile online: `https://guides.rubyonrails.org/`. Ultimo accesso: novembre 2025.

[9] Stripe. *Stripe Payments API*. Documentazione ufficiale, disponibile online: `https://stripe.com/docs/api`. Ultimo accesso: novembre 2025.

[10] Bootstrap. *Bootstrap 5 Documentation*. Disponibile online: `https://getbootstrap.com/docs/5.0/getting-started/introduction/`. Ultimo accesso: novembre 2025.

Appendici

A Requisiti

Requisiti per la Piattaforma di Prenotazione dei Campi Sportivi

1. Descrizione della Piattaforma

La piattaforma è progettata per consentire la prenotazione dei campi sportivi di diverse discipline (calcio, tennis, pallavolo, ecc.). Gli utenti principali della piattaforma sono:

- **Clienti**: Possono cercare, visualizzare e prenotare campi sportivi.
- **Partner**: Gestori dei campi sportivi che offrono disponibilità e ricevono prenotazioni.
- **Amministratori**: Gestiscono la piattaforma, verificano gli account, moderano contenuti e risolvono dispute.

2. Requisiti degli Utenti

- **Utente non autenticato**:
    - Deve poter navigare e visualizzare i campi sportivi disponibili.
    - Deve poter effettuare una ricerca dei campi in base alla disciplina sportiva e alla posizione geografica (utilizzando l'API di Bing Maps).
    - Se tenta di prenotare un campo o lasciare una recensione, deve essere reindirizzato alla pagina di login.
- **Registrazione e autenticazione**:
    - L'utente deve poter registrarsi compilando un form con i dati richiesti (nome, cognome, e-mail, password).
    - Il sistema deve inviare un'e-mail di verifica per attivare l'account.
    - Deve esserci una funzionalità di recupero password tramite e-mail.
- **Utente autenticato**:
    - Deve poter cercare e visualizzare i campi sportivi.
    - Deve poter prenotare un campo selezionando uno slot disponibile e completando il pagamento tramite l'API di Stripe.
    - Deve poter visualizzare le proprie prenotazioni (passate e future) e cancellare prenotazioni future, con eventuale rimborso se previsto dalla policy.
    - Deve poter lasciare una recensione per un campo sportivo visitato (titolo, valutazione da 1 a 5 stelle, testo).
    - Deve poter segnalare recensioni o risposte considerate inadeguate.
    - Deve poter ricevere notifiche via e-mail per conferme di prenotazione, cancellazioni, esiti delle segnalazioni.

3. Requisiti dei Partner

- **Registrazione e autenticazione**:
    - I partner devono registrarsi compilando un apposito form.
    - Dopo la registrazione, il partner deve verificare la propria e-mail e attendere l'approvazione dell'amministratore.
    - Deve poter accedere tramite una sezione dedicata, diversa da quella degli utenti generici.
- **Gestione del profilo**:
    - Deve poter visualizzare e modificare i propri dati personali.
    - Deve poter eliminare il proprio account e tutti i dati personali associati.
- **Gestione dei campi sportivi**:
    - Deve poter creare, visualizzare, modificare ed eliminare i campi sportivi.
    - Deve poter gestire gli slot di prenotazione (creazione, modifica, eliminazione).
    - Deve poter visualizzare tutte le prenotazioni, accettare o rifiutare prenotazioni specifiche, e gestire eventuali rimborsi.
- **Gestione delle recensioni e segnalazioni**:
    - Deve poter visualizzare, rispondere o segnalare recensioni ricevute sui propri campi sportivi.
    - Deve poter visualizzare gli introiti e i pagamenti ricevuti dalla piattaforma.

4. Requisiti degli Amministratori

- **Gestione degli account**:
    - Deve poter creare, modificare, sospendere o eliminare account utente e partner.
    - Deve poter verificare e approvare i partner registrati.
- **Gestione dei campi e prenotazioni**:
    - Deve poter visualizzare, modificare o eliminare i campi sportivi e gli slot di prenotazione.
    - Deve poter visualizzare e gestire le prenotazioni effettuate (modifica, cancellazione).
- **Gestione delle recensioni e segnalazioni**:
    - Deve poter visualizzare e moderare le recensioni e le risposte segnalate dagli utenti.
    - Deve gestire le segnalazioni di comportamenti scorretti da parte di clienti o partner, e comunicare l'esito agli utenti coinvolti.
- **Gestione dei pagamenti**:
    - Deve poter gestire i pagamenti verso i partner, visualizzare gli introiti della piattaforma e gestire eventuali controversie finanziarie.

5. Requisiti Tecnici

- **Integrazione con Bing Maps API** per la ricerca basata su posizione.
- **Integrazione con Stripe API** per la gestione dei pagamenti.
- **Supporto per notifiche e-mail** per conferme, cancellazioni, notifiche di segnalazioni, ecc.
- **Sistema di autenticazione sicura** con possibilità di mantenere la sessione (funzionalità "Ricorda la mia password").

6. Requisiti di Sicurezza e Privacy

- Tutte le informazioni personali e i dati delle transazioni devono essere crittografati.
- Gli utenti devono poter eliminare il proprio account e i relativi dati personali.
- Il sistema deve garantire la conformità al GDPR e alle normative locali sulla privacy.

B Storyboard

Storyboard per la Piattaforma di Prenotazione dei Campi Sportivi

Parte 1: Registrazione degli Utenti

1. **Accesso alla Schermata di Registrazione**
    - Il nuovo utente accede alla schermata di registrazione dell'applicazione.
    - L'utente inserisce i propri dati personali (nome, cognome, e-mail, password).
    - Dopo aver compilato il form, l'utente clicca sul pulsante "Registrati".
2. **Verifica dell'Email**
    - Il sistema invia un messaggio all'e-mail inserita con un link per la verifica.
    - L'utente clicca sul link per verificare l'indirizzo e-mail.
    - Il sistema conferma l'attivazione del profilo e reindirizza l'utente alla schermata di login.
3. **Accesso alla Piattaforma**
    - L'utente inserisce le credenziali di accesso (e-mail e password).
    - Il sistema autentica l'utente e crea la sessione, reindirizzandolo all'ultima pagina visitata o alla homepage della piattaforma.

Parte 2: Ricerca di un Campo Sportivo

1. **Ricerca del Campo**
    - L'utente utilizza la barra di ricerca nell'header per cercare un campo sportivo.
    - L'utente seleziona la disciplina sportiva e fornisce alternativamente la propria posizione o un indirizzo vicino cui cercare.
2. **Visualizzazione dei Risultati**
    - Il sistema mostra i campi sportivi vicini alla posizione di ricerca.
    - L'utente visualizza le informazioni di base dei campi nei risultati della ricerca (nome, distanza, foto, valutazione media).
    - L'utente clicca su un campo specifico per accedere alla pagina dettagliata.
3. **Visualizzazione delle Informazioni del Campo**
    - L'utente accede alla pagina del campo e visualizza:
        - Foto del campo.
        - Informazioni dettagliate (indirizzo, tipo di superficie, orari di apertura, ecc.).
        - Recensioni degli altri utenti.
        - Disponibilità degli slot di prenotazione.

Parte 3: Prenotazione di un Campo Sportivo

1. **Selezione dello Slot di Prenotazione**
    - L'utente autenticato seleziona uno slot di prenotazione disponibile dalla pagina del campo.
2. **Procedura di Pagamento**
    - L'utente viene reindirizzato alla pagina di pagamento gestita tramite l'API di Stripe.
    - L'utente inserisce i dati di pagamento e conferma la transazione.
3. **Conferma della Prenotazione**
    - Il sistema invia una conferma di prenotazione all'utente via e-mail.
    - L'utente viene reindirizzato alla pagina "Le mie prenotazioni" per visualizzare e gestire le proprie prenotazioni.

Parte 4: Gestione delle Prenotazioni

1. **Visualizzazione delle Prenotazioni**
    - L'utente autenticato accede alla sezione "Le mie prenotazioni" dal menu principale.
    - Il sistema mostra una lista di tutte le prenotazioni effettuate, con informazioni come data, orario, campo, stato della prenotazione.
2. **Cancellazione di una Prenotazione**
    - L'utente seleziona una prenotazione futura dalla lista.
    - L'utente clicca sul pulsante "Cancella Prenotazione".
    - Il sistema chiede conferma dell'azione attraverso una finestra modale.
    - Se la policy del partner lo prevede, l'utente riceve un rimborso.
    - Il sistema aggiorna la lista delle prenotazioni e invia una notifica di cancellazione via e-mail.

Parte 5: Creazione e Gestione delle Recensioni

1. **Creazione di una Recensione**
    - Dopo aver visitato un campo, l'utente autenticato accede alla pagina del campo e clicca su "Lascia una recensione".
    - L'utente inserisce il titolo della recensione, una valutazione da 1 a 5 stelle, e un commento.
    - L'utente invia la recensione; il sistema la pubblica immediatamente sulla pagina del campo.

2. **Modifica o Cancellazione di una Recensione**
    - L’utente autenticato può visualizzare la propria recensione sulla pagina del campo.
    - L’utente clicca su "Modifica" per apportare cambiamenti o su "Cancella" per rimuovere la recensione.
    - Il sistema aggiorna o rimuove la recensione in base all'azione dell'utente.

Parte 6: Segnalazione di Recensioni o Risposte Inappropriate

1. **Segnalazione di Contenuti Inappropriati**
    - L’utente (autenticato o non) visualizza una recensione o una risposta dei partner considerata inappropriata.
    - L’utente clicca su "Segnala" accanto alla recensione o risposta.
    - Il sistema invia la segnalazione agli amministratori per la revisione.
2. **Notifica dell'Esito della Segnalazione**
    - L’amministratore esamina la segnalazione e decide l'azione appropriata (rimozione, mantenimento, ecc.).
    - Il sistema invia una notifica via e-mail all’utente che ha effettuato la segnalazione, informandolo dell'esito.

Parte 7: Gestione da Parte del Partner

1. **Gestione dei Campi Sportivi**
    - Il partner accede alla propria dashboard dopo l’autenticazione.
    - Nella sezione “Gestione Campi”, il partner può creare un nuovo campo, modificare i campi esistenti, o eliminarli.
2. **Gestione degli Slot di Prenotazione**
    - Il partner visualizza una lista di slot prenotabili.
    - Può creare nuovi slot, modificarli, o eliminarli.
    - Se un cliente ha già prenotato uno slot, la cancellazione dello slot comporta un rimborso automatico al cliente.
3. **Visualizzazione e Risposta alle Recensioni**
    - Il partner accede alla sezione “Recensioni” nella dashboard.
    - Visualizza le recensioni dei clienti e ha la possibilità di rispondere, segnalare o eliminare le risposte.

Parte 8: Gestione da Parte dell'Amministratore

1. **Verifica e Approvazione degli Account**
    - L'amministratore accede alla propria dashboard tramite una sezione dedicata.
    - Nella sezione "Gestione Account", visualizza le richieste di verifica degli account dei partner e approva o rifiuta le richieste.
2. **Moderazione delle Recensioni e delle Segnalazioni**
    - L'amministratore accede alla sezione "Segnalazioni".
    - Visualizza le recensioni o risposte segnalate e decide se eliminarle o mantenerle.
    - Invia notifiche agli utenti coinvolti con l'esito delle segnalazioni.
3. **Gestione delle Prenotazioni**
    - L'amministratore visualizza tutte le prenotazioni effettuate sulla piattaforma.
    - Può modificare, cancellare prenotazioni o gestire rimborsi in caso di controversie.

Parte 9: Log-out

1. **Log-out degli Utenti Generici e Partner**
    - In qualsiasi momento durante la sessione, l'utente (cliente o partner) può cliccare sul pulsante "Log-out" posizionato nell'header della piattaforma.
    - Il sistema chiede una conferma dell'azione di log-out per evitare disconnessioni accidentali.
    - Dopo la conferma, il sistema termina la sessione dell'utente, cancella eventuali token di autenticazione salvati (se l'utente ha scelto di non ricordare la password), e reindirizza l'utente alla homepage o alla pagina di login.
    - Se l'utente ha attivato la funzione "Ricorda la mia password", la sessione non richiede nuovamente l'autenticazione al prossimo accesso, a meno che non venga esplicitamente disattivata durante il log-out.
2. **Log-out dell'Amministratore**
    - L'amministratore può effettuare il log-out in qualsiasi momento cliccando su "Log-out" dalla propria dashboard.
    - Il sistema termina la sessione dell'amministratore, reindirizzandolo alla pagina di login dell'amministrazione per motivi di sicurezza.
    - Tutti i dati di sessione vengono eliminati per garantire che non ci sia accesso non autorizzato in futuro.

C User stories

User Stories per la Piattaforma di Prenotazione dei Campi Sportivi

Clienti:

1. **Ricerca e Visualizzazione dei Campi Sportivi**
    - *Come utente, autenticato o non, voglio poter ricercare un campo sportivo di una determinata disciplina, basandomi sulla mia posizione o su un indirizzo da me inserito, in modo da trovare campi vicini e disponibili.*
    - *Come utente, autenticato o non, voglio poter visualizzare la pagina di un campo sportivo e visualizzarne le informazioni (foto, recensioni, disponibilità), in modo da decidere se prenotare il campo.*
2. **Registrazione e Autenticazione**
    - *Come utente non autenticato, voglio poter effettuare la registrazione al sito compilando un apposito form, in modo da creare un account e accedere a funzionalità avanzate.*
    - *Come utente registrato, voglio poter confermare la mia e-mail tramite un link inviato dal sistema, per attivare il mio account.*
    - *Come utente non autenticato, voglio poter effettuare l'accesso al mio profilo, potendo scegliere se mantenere l'accesso o no, per facilitare future sessioni.*
3. **Gestione delle Prenotazioni**
    - *Come utente autenticato, voglio poter prenotare uno slot disponibile e completare il pagamento, per assicurarmi di avere accesso al campo sportivo.*
    - *Come utente autenticato, voglio poter visualizzare le mie prenotazioni (passate e future), in modo da avere una chiara visione delle mie attività sportive.*
    - *Come utente autenticato, voglio poter cancellare una prenotazione futura, e se previsto dalla policy del partner, ottenere un rimborso.*
    - *Come utente autenticato, voglio ricevere una e-mail se la mia prenotazione viene cancellata per qualsiasi motivo, per essere informato tempestivamente.*
4. **Gestione del Profilo e delle Recensioni**
    - *Come utente autenticato, voglio poter modificare la password e i miei dati personali, per mantenerli aggiornati.*
    - *Come utente autenticato, voglio poter eliminare il mio account e tutti i dati personali associati, per esercitare il mio diritto alla privacy.*
    - *Come utente autenticato, e che ha visitato un campo, voglio poter lasciare una recensione sul campo, in modo da condividere la mia esperienza con altri utenti.*
    - *Come utente autenticato, voglio poter cancellare una recensione che ho fatto, se cambio idea o ritengo non sia più rilevante.*

5. **Segnalazioni e Notifiche**
    - *Come utente, autenticato o non, voglio poter segnalare le risposte dei partner alle recensioni che considero inappropriate, per mantenere l'integrità della piattaforma.*
    - *Come utente, autenticato o non, voglio poter segnalare le recensioni degli altri clienti che considero inappropriate, per contribuire alla moderazione della comunità.*
    - *Come utente autenticato, voglio ricevere una e-mail sull'esito della mia segnalazione riguardante una recensione o risposta, per sapere come è stata gestita.*
    - *Come utente autenticato, voglio poter segnalare un comportamento scorretto rispetto a una mia prenotazione, in modo da ottenere giustizia.*
    - *Come utente autenticato, voglio ricevere una e-mail sull'esito della mia segnalazione riguardante una prenotazione, per essere informato sullo stato della questione.*

Partner:

1. **Registrazione e Autenticazione**
    - *Come partner non autenticato, voglio poter effettuare la registrazione all'applicazione compilando l'apposito form, per iniziare a offrire i miei campi sportivi sulla piattaforma.*
    - *Come partner registrato, voglio poter confermare la mia e-mail per avviare la procedura di verifica del mio account, per poter accedere alle funzionalità della piattaforma.*
    - *Come partner, voglio essere notificato tramite e-mail quando il mio account viene verificato e reso attivo, per iniziare a usare la piattaforma.*
2. **Gestione del Profilo e dei Dati**
    - *Come partner autenticato, voglio poter visualizzare la pagina relativa al mio account e modificare le mie informazioni, per tenerle sempre aggiornate.*
    - *Come partner autenticato, voglio poter eliminare il mio account e tutti i miei dati personali, per esercitare il mio diritto alla privacy.*
3. **Gestione dei Campi Sportivi e delle Prenotazioni**
    - *Come partner autenticato, voglio poter creare, visualizzare, modificare ed eliminare i miei campi sportivi, per gestire l'offerta dei miei servizi.*
    - *Come partner autenticato, voglio poter gestire gli slot di prenotazione (creare, modificare, eliminare), per ottimizzare la disponibilità dei campi.*
    - *Come partner autenticato, voglio visualizzare tutte le prenotazioni effettuate, per monitorare le attività e i flussi di prenotazione.*
    - *Come partner autenticato, voglio poter cancellare uno slot prenotabile, eventualmente cancellando e rimborsando la prenotazione se già prenotato, per gestire situazioni impreviste.*

4. **Gestione delle Recensioni e delle Segnalazioni**
    - *Come partner autenticato, voglio poter visualizzare le recensioni dei miei campi, per monitorare il feedback dei clienti.*
    - *Come partner autenticato, voglio poter rispondere alle recensioni, per fornire chiarimenti o ringraziare i clienti.*
    - *Come partner autenticato, voglio poter segnalare le recensioni inadeguate, per mantenere l'integrità del profilo del mio campo sportivo.*
    - *Come partner autenticato, voglio ricevere un'e-mail riguardante una segnalazione rispetto ai comportamenti scorretti dei miei clienti, per essere informato tempestivamente.*
5. **Gestione degli Introiti**
    - *Come partner autenticato, voglio visualizzare gli introiti totali del periodo e i pagamenti effettuati verso di me dalla piattaforma, per monitorare le mie finanze.*

Amministratori

1. **Gestione degli Accessi**
    - *Come amministratore, voglio poter effettuare l'accesso alla mia dashboard, per gestire la piattaforma.*
    - *Come amministratore, voglio poter effettuare il log-out dalla mia dashboard, per mantenere la sicurezza.*
2. **Verifica e Moderazione degli Account**
    - *Come amministratore, voglio poter approvare o non approvare i partner che richiedono la verifica del proprio account, per garantire la qualità e l'affidabilità degli operatori sulla piattaforma.*
    - *Come amministratore, voglio visualizzare i partner e i clienti iscritti alla piattaforma, per avere una panoramica completa degli utenti.*
3. **Gestione degli Account**
    - *Come amministratore, voglio poter cancellare o sospendere un account di un partner o di un cliente, per risolvere eventuali problemi o violazioni.*
    - *Come amministratore, voglio poter visualizzare, creare, eliminare o modificare i campi sportivi registrati, per garantire la qualità delle informazioni.*
4. **Gestione delle Prenotazioni e Recensioni**
    - *Come amministratore, voglio visualizzare gli slot prenotabili e le prenotazioni associate a un determinato campo sportivo, per gestire le attività della piattaforma.*

- *Come amministratore autenticato, voglio visualizzare le recensioni e le risposte segnalate, ed eventualmente eliminarle o contrassegnarle come verificate, per mantenere un ambiente appropriato sulla piattaforma*.

5. **Gestione delle Segnalazioni**

   - *Come amministratore autenticato, voglio poter visualizzare le segnalazioni effettuate dai clienti e dai partner, per valutare comportamenti scorretti durante l'uso della piattaforma*.
   - *Come amministratore autenticato, voglio assegnare un esito a una segnalazione di scorrettezza, per garantire la giustizia e il rispetto delle regole sulla piattaforma*.

Ringraziamenti

Testo ringraziamenti