HomeRental

Progetto Basi di Dati 2023/2024

Prati Lorenzo

Analisi dei requisiti

Realizzare un un database a supporto di un sito web di house sharing per il soggiorno di brevi periodi.

Testo dell'intervista

Per usare l'applicazione, ogni utente necessita della creazione di un account con alcuni suoi dati personali: email, nome utente, password, nome, cognome. Email e nomi utenti devono essere univoci nell'ambito dell'applicazione. Se un utente desidera mettere a disposizione una o più proprietà ad eventuali ospiti, deve inserire anche il numero di telefono su cui essere contattato dagli ospiti in caso di bisogno, un metodo di pagamento su cui ricevere i pagamenti e una biografia, utile a farsi conoscere meglio; una volta fatto l'utente è a tutti gli effetti considerato un proprietario. Per ogni proprietà si vogliono memorizzare la città[1]*, l'indirizzo, una descrizione testuale, il tipo di struttura (appartamento, villa, baita, cottage, castello, barca, tenda), le amenità presenti (wi-fi, piscina, parcheggio, cucina, vasca, climatizzatore, lavatrice, self check-in, tv, camino) e le camere da letto. Ciascuna camera deve indicare il prezzo per notte, il numero massimo di persone che può ospitare e un numero di riconoscimento (ad esempio Camera 1, Camera 2 ecc.). Un proprietario può in qualsiasi momento cancellare una sua proprietà dal suo profilo, oltre a poter aggiungere e rimuovere nuove camere.*[2] *Gli utenti possono prenotare soggiorni nelle proprietà altrui, dei quali si vuole tenere uno storico. Per ogni soggiorno prenotato da un utente, si vogliono memorizzare il numero delle persone che, eventualmente, soggiorneranno insieme a lui, le date di inizio e fine soggiorno e le camere da lui scelte. La procedura sarà completata dal pagamento di tale soggiorno, durante il quale l'utente può decidere di applicare uno sconto utilizzando un coupon. Infatti, un utente può guadagnare in vari modi*[3] *dei coupon, ciascuno con una percentuale di sconto casuale e un numero variabile di tipi di struttura su cui può essere applicato. Una volta utilizzato per un pagamento, un coupon non può essere più utilizzato. Per ogni utente si vogliono memorizzare un numero a piacere di opzioni di pagamento. Il sito mette a disposizione due tipi di metodo di pagamento: carta di credito e conto paypal.*[4] *Per le carte di credito si vuole memorizzare il numero della carta, il nome e il cognome del possessore e la data di scadenza. Per i conti paypal si vuole memorizzare l'email associata al conto. Una carta di credito o un conto paypal devono essere univoci all'interno dell'applicazione.*[5] *Anche per i pagamenti si vuole tracciare lo storico. Una volta completata la prenotazione, l'utente ha comunque la possibilità di annullarla prima che arrivi il momento del check-in; allo stesso modo,*

[1] Per semplicità, si considera che il sito operi solamente in un determinato paese

[2] Per preservare l'integrità dei dati a fronte delle rimozioni (di proprietà e camere), il database deve comunque continuare a mantenere tutte le informazioni utili alla ricostruzione, per utenti e host, di uno storico delle prenotazioni

[3] Per semplicità, alla registrazione vengono assegnati a quell'utente tre coupon dalle caratteristiche casuali

[4] Per lo scope limitato del progetto, i dati sensibili degli utenti si memorizzano in modo semplice e non adeguato a un'applicazione reale

[5] Ho scelto questa opzione per evitare duplicati, anche se si poteva dare la possibilità di memorizzare la stessa carta/paypal in due account diversi

un proprietario può annullare una prenotazione effettuata in una sua proprietà. Superata la data di check-in, l'ospite può rilasciare una recensione (valutazione da 1 a 5 stelle e commento testuale) per la suddetta proprietà. Nel caso la proprietà sia stata nel frattempo rimossa, non sarà più possibile rilasciare una recensione per quella proprietà.

Glossario dei termini

Termine usato	Eventuali sinonimi nel dominio
utente	ospite, guest, account
proprietario	ospitante, host, proprietario di casa
proprietà	casa, alloggio, struttura
camera	stanza
città	luogo, meta, destinazione
amenità	comfort, tag, servizio
tipo struttura	tipologia di proprietà
soggiorno	prenotazione
recensione	valutazione
pagamento	-
metodo di pagamento	opzione di pagamento
carta di credito	-
paypal	conto paypal, account paypal
coupon	codice sconto, buono sconto

Ristrutturazione del testo con eliminazione delle ambiguità

Per usare l'applicazione, ogni **utente** necessita di registrarsi con alcuni suoi dati personali: email, nome utente, password, nome, cognome. Email e nomi utenti devono essere univoci nell'ambito dell'applicazione.
Un **proprietario** deve inserire anche il numero di telefono su cui essere contattato dagli ospiti in caso di bisogno, un metodo di pagamento su cui ricevere i pagamenti e una biografia, utile a farsi conoscere meglio.
Per ogni **proprietà** si vogliono memorizzare la **città**, l'indirizzo, una descrizione testuale, il **tipo di struttura** (appartamento, villa, baita, cottage, castello, barca, tenda), le **amenità** presenti (wi-fi, piscina, parcheggio, cucina, vasca, climatizzatore, lavatrice, self check-in, tv, camino) e le camere da letto. Ciascuna **camera** deve indicare il prezzo per notte, il numero massimo di persone che può ospitare e un numero di riconoscimento (ad esempio Camera 1, Camera 2 ecc.). Un proprietario può in qualsiasi momento cancellare una sua proprietà dal suo profilo, oltre a poter aggiungere e rimuovere nuove camere.
Gli utenti possono prenotare soggiorni nelle proprietà altrui, dei quali si vuole tenere uno storico. Per ogni **soggiorno** prenotato da un utente, si vogliono memorizzare il numero delle persone che, eventualmente, soggiorneranno insieme a lui, le date di inizio e fine soggiorno e le camere da lui scelte.
La procedura sarà completata dal **pagamento** di tale soggiorno, durante il quale l'utente può decidere di applicare uno sconto utilizzando un coupon. Infatti, un utente può guadagnare in vari modi dei **coupon**, ciascuno con una percentuale di sconto casuale e un numero variabile di tipi di struttura su cui può essere applicato. Una volta utilizzato per un pagamento, un coupon non può essere più utilizzato.
Per ogni utente si vogliono memorizzare un numero a piacere di metodi di pagamento. Il sito mette a disposizione due tipi di **metodo di pagamento**: carta di credito e paypal. Per le **carte di credito** si vuole memorizzare il numero della carta, il nome e il cognome del possessore e la data di scadenza. Per **paypal** si vuole memorizzare l'email associata al conto. Una carta di credito o un paypal devono essere univoci all'interno dell'applicazione. Anche per i pagamenti si vuole tracciare lo storico. Una volta prenotato il soggiorno, l'utente ha comunque la possibilità di annullarlo prima che arrivi il momento del check-in; allo stesso modo, un proprietario può annullare un soggiorno prenotato in una sua proprietà.
Superata la data di check-in, l'ospite può rilasciare una **recensione** (valutazione da 1 a 5 stelle e commento testuale) per la suddetta proprietà. Nel caso la proprietà sia stata nel frattempo rimossa, non sarà più possibile rilasciare una recensione per quella proprietà.

Operazioni

Di seguito sono elencate alcune operazioni che potranno/dovranno essere svolte.

1. Inserire un utente
2. Inserire una proprietà
3. Inserire un soggiorno
4. Visualizzare il prossimo soggiorno di un utente[6]
5. Visualizzare i soggiorni prenotati in una proprietà
6. Visualizzare le proprietà disponibili in base a criteri di ricerca[7]
7. Inserire una recensione
8. Visualizzare la valutazione media di una proprietà
9. Visualizzare la valutazione media di un proprietario
10. Visualizzare i coupon applicabili
11. Visualizzare i 10 proprietari migliori[8]
12. Visualizzare le 10 proprietà migliori[9]
13. Visualizzare le 5 città che hanno ospitato più soggiorni negli ultimi 30 giorni[10]
14. Visualizzare le amenità dalla più alla meno presente nelle proprietà con valutazione maggiore o uguale a 4.5/5 [11]

[6] Funzionalità di mostrare, ad esempio nella home page, quale sarà il prossimo viaggio dell'utente
[7] La query principale dell'applicazione, implementata con i seguenti filtri: città, numero di ospiti, tipo di struttura, lista di amenità
[8] Per *migliori* si intende una media ponderata che tiene conto della media globale
[9] Analogamente alla nota 3
[10] *mete popolari*
[11] *amenità apprezzate*

Progettazione concettuale

Schema scheletro

Inizialmente ho pensato a uno schema generale che contenesse tutti i concetti “core” del dominio e i loro legami. La dinamica identificativa dell’applicazione è, infatti, quella dello scenario di prenotazione delle proprietà. Dalla lettura del testo ristrutturato appare una buona idea quella di creare quattro entità per i concetti fondamentali: utente, proprietà, camera e soggiorno.
Un **utente** possiede zero o molte **proprietà**, la quale a sua volta ha a disposizione una o nessuna **camera**. Ho scelto di impostare una cardinalità minima pari a zero nella partecipazione di proprietà all’associazione disponibilità poichè in questo modo l’esistenza di una proprietà non è vincolata all’esistenza anche di una sua camera; questo da una parte potrebbe avere poco senso in un contesto reale poichè una proprietà del dominio ha senso se ha anche delle camere a disposizione degli ospiti, dall’altra consente agli utenti, a un certo momento, di rimuovere potenzialmente tutte le camere da una proprietà (con relativa cancellazione dei soggiorni prenotati per quelle camere) senza dover rimuovere anche la proprietà. Questo in ottica futura semplifica molto l’inserimento/gestione di proprietà e camere.
Infine, un utente può prenotare zero o molti **soggiorni**; ciascuno, occupa una o molte camere.

foto

Raffinamento di Utente, Proprietario, Proprietà

Dallo schema generale ho raffinato prima il concetto di **utente**, aggiungendo gli attributi email, password, nome_utente, nome, cognome. In questa fase sono presenti due identificatori: l’email e il nome_utente.
Ho deciso di modellare il concetto di **proprietario** con un subset di utente. Un proprietario è quindi un utente specializzato che memorizza anche gli attributi biografia (opzionalmente), numero di telefono, valutazione_media e num_valutazioni, oltre che avere associazioni specifiche. Ho deciso da subito di tenere traccia delle informazioni utili all’ottenimento di informazioni come media delle valutazioni e numero delle valutazioni, sebbene sia un’evidente ridondanza concettuale. Questa scelta verrà approfondita durante la fase di progettazione logica.
L’entità che modella la **proprietà** possiede gli attributi: indirizzo, descrizione, valutazione_media, numero_valutazioni e data_creazione. L’identificatore è un codice. Per poter gestire la rimozione delle proprietà, ma come specificato senza perdere informazioni utili alla ricostruzione dello storico dei soggiorni, ho creato l’entità **proprieta_rimossa** come subset di proprietà, contenente solo l’attributo data_rimozione. In questo modo viene realizzata una sorta di “archiviazione”.

Per quanto riguarda la **camera**, l'entità è dotata degli attributi ordinale, num_ospiti e prezzo_per_notte. Analogamente a come detto per la proprietà, è presente anche un'entità **camera_rimossa**.
Per modellare la **citta**, ho creato un'entità identificata dal nome della città.
Una proprietà deve anche appartenere ad un **tipo_struttura**, quindi ho creato un'entità identificata dal nome del tipo di struttura.
Infine, l'entità **amenita** è identificata dal nome dell'amenità.

foto

Raffinamento di Soggiorno, Recensione, Metodi di Pagamento, Coupon

Il **soggiorno** è modellato da un'entità con gli attributi: check_in, check_out, num_ospiti e prezzo (somma totale da pagare per il soggiorno, calcolata al momento dell'inserimento in modo da evitare ambiguità dovute a futuri cambi di prezzo delle camere). Per gestire la cancellazione dei soggiorni, sia da parte dell'utente prenotante sia del proprietario, è utile l'entità **soggiorno_cancellato**, con l'attributo data_cancellazione. Con questo modello non si è in grado di capire se la cancellazione sia stata fatta dal proprietario o dall'utente, ma questo non mi è risultato un dato utile. Facendo le opportune considerazioni sulle date del soggiorno, è possibile individuare lo "status" dello stesso: "prenotato", "in corso", "completato" e "annullato". Non è quindi necessario inserire ulteriori costrutti per rappresentare lo "status" del soggiorno.
Ho deciso di modellare la parte riguardante i metodi di pagamento con una gerarchia (totale, esclusiva) dove l'entità padre **metodo_pagamento** memorizza: data_creazione e codice identificativo. Inoltre, partecipa alle associazioni con le entità utente e proprietario.
L'entità **carta_credito** memorizza banalmente gli attributi: numero_carta, nome, cognome, data_scadenza. Il numero_carta è identificatore.
L'entità **paypal** memorizza invece solamente l'email, che è anche identificatore.
Un utente può avere zero o molti metodi di pagamento. Un proprietario deve necessariamente avere un metodo di pagamento. Un'istanza di metodo_pagamento può essere associata a un solo utente, ma può essere, per quell'utente, sia metodo di pagamento generico (addebito) che metodo di accredito per le sue proprietà (se questo è anche proprietario).
Un **pagamento** è modellato da un'entità con attributo data_creazione (timestamp del pagamento), identificata dall'entità soggiorno a cui si riferisce. Un soggiorno corrisponde sempre a un pagamento, e viceversa.
Un **coupon** è modellato da un'entità con attributi: percentuale_sconto e codice identificativo. Un coupon è applicabile a uno o molti tipi_struttura.
Un pagamento è associato, oltre che al soggiorno a cui si riferisce, a zero o un coupon, ad un metodo di pagamento (quello usato per l'addebito) e ad un altro metodo di pagamento (accredito). In questo modo, anche se il proprietario cambia il suo metodo di accredito, è sempre possibile ricostruire la transazione per eventuali ricevute, rimborsi ecc.
Infine, una **recensione** è modellata con un'associazione molti a molti tra utente e proprietà, con gli attributi: valutazione, testo e data_ultima_modifica. In questo modo, un utente può rilasciare una sola recensione per una certa proprietà, ma può modificarla.

Schema concettuale finale

foto

Progettazione logica

Stima dei volumi

Concetto	**Costrutto**	**Volume (dopo 1 anno)**
UTENTE	E	500.000 (1000 nuovi al giorno)
PROPRIETARIO	E	50.000 (10% degli utenti)
PROPRIETA	E	50.000 (~1 per proprietario)
PROPRIETA_RIMOSSA	E	1000
CAMERA	E	100.000 (~2 per proprietà)
CAMERA_RIMOSSA	E	2000
AMENITA	E	10
TIPO STRUTTURA	E	7
CITTA[12]	E	5000
SOGGIORNO	E	1.000.000 (2 per utente)
SOGGIORNO_CANCELLATO	E	10.000
PAGAMENTO	E	1.000.000
METODO_PAGAMENTO	E	1.000.000 (2 per utente)
CARTA_CREDITO	E	500.000
PAYPAL	E	500.000
COUPON	E	1.500.000 (3 per utente)
possedimento	A	50.000
disponibilità	A	100.000
servizio	A	150.000 (3 per proprietà)
ubicazione	A	50.000
tipologia	A	50.000

[12] Città italiane (approssimazione)

prenotazione	A	1.000.000
occupazione	A	2.000.000 (2 camere per soggiorno)
recensione	A	1.000.000 (1 per soggiorno)
accredito_predefinito	A	50.000
utente-metodo_pagamento	A	1.000.000
finalizzazione	A	1.000.000
addebito	A	1.000.000
accredito	A	1.000.000
utente-coupon	A	1.500.000
spesa-coupon	A	500.000 (1 coupon ogni 2 pagamenti)
spendibilità-coupon	A	4.500.000 (3 tipi struttura per coupon)

Frequenza delle operazioni

1. Inserire un utente (1000/giorno)
2. Inserire una proprietà (100/giorno)
3. Inserire un soggiorno (2800/giorno)
4. Visualizzare il prossimo soggiorno di un utente (500.000/giorno)
5. Visualizzare i soggiorni prenotati in una proprietà (100.000/giorno)
6. Visualizzare le proprietà disponibili in base a criteri di ricerca (3.000.000/giorno)
7. Inserire una recensione (2800/giorno)
8. Visualizzare la valutazione media di una proprietà (10.000.000/giorno)
9. Visualizzare la valutazione media di un proprietario (5.000.000/giorno)
10. Visualizzare i coupon applicabili (5000/giorno)
11. Visualizzare i 10 proprietari migliori (3.000.000/giorno)
12. Visualizzare le 10 proprietà migliori (3.000.000/giorno)
13. Visualizzare le 5 città che hanno ospitato più soggiorni negli ultimi 30 giorni (3.000.000/giorno)
14. Visualizzare le amenità dalla più alla meno presente nelle proprietà con valutazione maggiore o uguale a 4.5/5 (3.000.000/giorno)

Tavole degli accessi

S = scrittura, L = lettura, 1 S = 2 L

Concetto	Costrutto	Numero Accessi	Tipo Accesso
1. Inserire un utente: 1000/giorno			
UTENTE	E	1	S
totale: 2000 L			
2. Inserire una proprietà: 100/giorno Una proprietà mediamente ha 3 amenità e 2 camere.			
PROPRIETA	E	1	S
possedimento	A	1	S
servizio	A	3	S
ubicazione	A	1	S
tipologia	A	1	S
disponibilità	A	2	S
CAMERA	E	2	S
totale: 2200 L			
3. Inserire un soggiorno: 2800/giorno Ciò comporta anche l'inserimento di un pagamento. Il soggiorno occupa 2 camere e usa un coupon, già selezionato compatibile con il tipo struttura.			
prenotazione	A	1	S
SOGGIORNO	E	1	S
occupazione	A	2	S
finalizzazione	A	1	S
PAGAMENTO	E	1	S
addebito	E	1	S
accredito	E	1	S
spesa_coupon	E	1	S
totale: 50.400 L			

4. Visualizzare il prossimo soggiorno di un utente: 500.000/giorno Si vogliono anche visualizzare le informazioni sulle camere e sulla proprietà.			
UTENTE	E	1	L
prenotazione	A	2	L
SOGGIORNO	E	2	L
occupazione	A	2	L
CAMERA	E	2	L
disponibilità	A	1	L
PROPRIETA	E	1	L
totale: 5.500.000 L			
5. Visualizzare i soggiorni prenotati in una proprietà: 100.000/giorno Si vogliono visualizzare anche le camere prenotate e il nome dell'utente che ha prenotato il soggiorno.			
PROPRIETA	E	1	L
disponibilità	A	2	L
CAMERA	E	2	L
occupazione	A	40	L
SOGGIORNO	E	20	L
prenotazione	A	20	L
UTENTE	E	20	L
totale: 10.500.000 L			
6. Visualizzare le proprietà disponibili in base a criteri di ricerca: 3.000.000/giorno Considero di cercare le proprietà (con informazioni anche riguardo all'utente che le possiede) che hanno camere libere in un certo periodo, che possono contenere un certo numero di ospiti, in una certa città, appartenenti a 2 tipi struttura e con 3 amenità selezionate			
CITTA	E	1	L
ubicazione	A	10	L
PROPRIETA	E	10	L

tipologia	A	10	L
TIPO_STRUTTURA	E	10	L
servizio	A	30	L
AMENITA	E	30	L
disponibilità	A	20	L
CAMERA	E	20	L
occupazione	A	400	L
SOGGIORNO	E	200	L
totale: ? L			
7. Inserire una recensione: 2800/giorno È necessario aggiornare anche gli attributi valutazioni_media e num_valutazioni in proprietario e proprietà.			
recensione	A	1	S
PROPRIETA	E	1	L
PROPRIETA	E	1	S
PROPRIETARIO	E	1	L
PROPRIETARIO	E	1	S
totale: 22.400 L			
8. Visualizzare la valutazione media di una proprietà: 10.000.000/giorno			
PROPRIETA	E	1	L
totale: 10.000.000 L			
9. Visualizzare la valutazione media di un proprietario: 5.000.000/giorno			
PROPRIETARIO	E	1	L
totale: 5.000.000 L			
10. Visualizzare i coupon applicabili: 5000/giorno Bisogna verificare che il coupon non sia già stato usato in un pagamento.			
UTENTE	E	1	L

utente-coupon	A	3	L
COUPON	E	3	L
spendibilita-coupon	A	9	L
spesa_coupon	E	1	L
totale: 85.000 L			
11. Visualizzare i 10 proprietari migliori: 3.000.000/giorno			
PROPRIETARIO	E	50.000	L
totale: 50.000 L			
12. Visualizzare le 10 proprietà migliori: 1.000.000/giorno			
PROPRIETA	E	50.000	L
totale: 50.000 L			
13. Visualizzare le 5 città che hanno ospitato più soggiorni negli ultimi 30 giorni: 1.000.000/giorno			
PROPRIETA	E	50.000	L
disponibilità	A	100.000	L
CAMERA	E	100.000	L
occupazione	A	2.000.000	L
totale: 2.250.000.000.000 L			
14. Visualizzare le amenità dalla più alla meno presente nelle proprietà con valutazione media superiore a 4.5/5: 1.000.000/giorno			
PROPRIETA	E	50.000	L
servizio	A	150.000	L
totale: 200.000.000.000 L			

Schemi di accesso

Raffinamento dello schema

Eliminazione dei costrutti E-R

Per quanto tiguarda i tre subset proprietà_rimossa, camera_rimossa e soggiorno_cancellato, ho effettuato un collasso verso l'alto. Poichè l'attributo delle entità figlie è solo uno, non è necessario aggiungere un attributo per riconoscere il "tipo". Invece, la gerarchia totale esclusiva dei pagamenti l'ho tradotta con l'uso di associazioni, poichè risulta comodo mantenere entrambe le entità figlie e l'entità padre con le sue associazioni. In questo caso ho aggiunto l'attributo tipo con valori in {CARTA_CREDITO, PAYPAL} in modo da poter riconoscere il sottotipo anche partendo dall'entità padre, anche se non era necessario.

Scelta delle chiavi primarie

Le entità pagamento, carta_credito, paypal e proprietario hanno chiave primaria (id) uguali a quelle dell'entità a cui sono legate. L'entità utente ha due identificatori: email e nome_utente. Per comodità ho scelto di creare un codice come chiave primaria. Le altre entità hanno solo un identificatore, quindi quello diventa la loro chiave primaria.

Analisi delle ridondanze

La ridondanza concettuale che ho mantenuto consinste nel tenere traccia della valutazione media e del numero di valutazioni per le proprietà e i propietari. Poichè le operazioni 8 e 9 sono molto frequenti, la ridondanza risulta utile. Infatti, senza ridondanza si avrebbe un numero di letture molto superiore.

Inserire una recensione <u>senza ridondanza</u>: 2800/giorno			
Concetto	**Costrutto**	**Numero accessi**	**Tipo accesso**
recensione	A	1	S
totale: 5600 L			

Visualizzare la valutazione media di una proprietà <u>senza ridondanza</u>: 10.000.000/giorno			
Concetto	**Costrutto**	**Numero accessi**	**Tipo accesso**
PROPRIETA	E	1	L
recensione	A	20	L
totale: 210.000.000			

Visualizzare la valutazione media di un proprietario <u>senza ridondanza</u>: 5.000.000/giorno			
Concetto	**Costrutto**	**Numero accessi**	**Tipo accesso**
PROPRIETARIO	E	1	L
PROPRIETA	A	1	L
recensione	A	20	L
totale: 110.000.000			

In totale, senza ridondanza si fanno circa 300.000.000 letture al giorno, mentre dalle tabelle degli accessi precedentemente mostrate si ricava che aggiungendo la ridondanza si fanno solamente circa 15.000.000 di letture al giorno. Pertanto, si mantiene la ridondanza.

Traduzione di entità e associazioni in relazioni

Partendo dallo schema concettuale ristrutturato, con evidenziate le chiavi primarie, ho operato la seguente traduzione.
Ho eliminato:

- utente-proprietario: l'id del proprietario è lo stesso del suo utente
- possedimento: l'id del proprietario viene importato in proprietà
- ubicazione: l'id della città viene importato in proprietà
- tipologia: l'id del tipo struttura viene importato in proprietà
- disponibilità: l'id della proprietà viene importato in camera
- prenotazione: l'id dell'utente viene importato in soggiorno
- finalizzazione: l'id del pagamento è lo stesso del suo soggiorno
- spesa_coupon: l'id del coupon viene importato in pagamento (opzionale)
- addebito: l'id del metodo_pagamento viene importato in pagamento
- accredito: l'id del metodo_pagamento viene importato in pagamento
- utente-coupon: l'id dell'utente viene importato in coupon
- utente-metodo_pagamento: l'id dell'utente viene importato in metodo_pagamento
- accredito_predefinito: l'id del metodo_pagamento viene importato in proprietario
- carta_credito-metoto_pagamento: l'id della carta_credito è lo stesso del suo metodo_pagamento
- paypal-metodo_pagamento: l'id di paypal è lo stesso del suo metodo_pagamento

Ho reificato:

- servizio: importando l'id della proprietà e l'id dell'amenità e facendoli chiave primaria
- occupazione: importando l'id della camera e l'id del soggiorno e facendoli chiave primaria
- recensione: importando l'id dell'utente e l'id della proprietà e facendoli chiave primaria
- spendibilità_coupon: importando l'id del coupon e l'id del tipo struttura e facendoli chiave primaria

Schema relazionale finale

utenti (**id**, email, password, nome_utente, nome, cognome)
UNIQUE: email
UNIQUE: nome_utente
proprietari (**id: utenti**, biografia*, valutazione_media, num_valutazioni, id_metodo_accredito: metodi_pagamento)
UNIQUE: id_metodo_accredito
proprietà (**id**, indirizzo, descrizione*, valutazione_media, num_valutazioni, data_creazione, id_proprietario: proprietari, id_città: città, id_tipo_struttura: tipi_struttura, data_rimozione*)
camere (**id**, ordinale, num_ospiti, prezzo_per_notte, id_proprietà: proprietà, data_rimozione*)
soggiorni (**id**, check_in, check_out, num_ospiti, prezzo, id_utente: utenti, data_cancellazione*)
occupazioni (**id_soggiorno: soggiorni, id_camera: camere**)
recensioni (**id_utente: utenti, id_proprietà: proprietà**, valutazione, testo*, data_ultima_modifica)
servizi (**id_proprietà: proprietà, id_amenità: amenità**)
metodi_pagamento (**id**, tipo, id_utente: utente)
carte_credito (**id: metodi_pagamento**, numero_carta, nome, cognome, data_scadenza)
UNIQUE: numero_carta
paypals (**id: metodi_pagamento**, email)
UNIQUE: email
coupons (**id**, percentuale_sconto, id_utente: utente)
spendibilità_coupons (**id_coupon: coupons, id_tipo_struttura: tipi_struttura**)
pagamenti (**id: soggiorni**, id_metodo_addebito: metodi_pagamento, id_metodo_accredito: metodi_pagamento, id_coupon*: coupons, data_creazione)
amenità (**nome**)
tipi_struttura(**nome**)
città (**nome**)

Traduzione delle operazioni in query SQL

1. **Inserire un utente**

```
INSERT INTO utenti (email, password, nome_utente, nome, cognome)
VALUES (%(email)s, %(password)s, %(nome_utente)s, %(nome)s, %(cognome)s)
```

2. **Inserire una proprietà**

```
INSERT INTO proprieta (indirizzo, descrizione, valutazione_media,
num_valutazioni, data_creazione, id_citta, id_tipo_struttura,
id_proprietario, data_rimozione)
VALUES (%(indirizzo)s, %(descrizione)s, %(valutazione_media)s,
%(num_valutazioni)s, NOW(), %(id_citta)s, %(id_tipo_struttura)s,
%(id_proprietario)s, %(data_rimozione)s)

INSERT INTO servizi (id_proprieta, id_amenita)
VALUES (%(id_proprieta)s, %(id_amenita)s)

INSERT INTO camere (ordinale, num_ospiti, prezzo_per_notte, id_proprieta,
data_rimozione)
VALUES (%(ordinale)s, %(num_ospiti)s, %(prezzo_per_notte)s,
%(id_proprieta)s, %(data_rimozione)s)
```

3. **Inserire un soggiorno**

```
INSERT INTO soggiorni (check_in, check_out, num_ospiti, prezzo, id_utente,
data_cancellazione)
VALUES (%(check_in)s, %(check_out)s, %(num_ospiti)s, %(prezzo)s,
%(id_utente)s, %(data_cancellazione)s)

INSERT INTO occupazioni (id_soggiorno, id_camera)
VALUES (%(id_soggiorno)s, %(id_camera)s)

INSERT INTO pagamenti (id, id_metodo_addebito, id_metodo_accredito,
id_coupon, data_creazione)
VALUES (%(id)s, %(id_metodo_addebito)s, %(id_metodo_accredito)s,
%(id_coupon)s, NOW())
```

4. **Visualizzare il prossimo soggiorno di un utente**

```
SELECT *
FROM soggiorni
WHERE soggiorni.id_utente = %(id_utente_1)s
AND soggiorni.data_cancellazione IS NULL
AND soggiorni.check_in > NOW()
```

```
ORDER BY soggiorni.check_in
LIMIT 1

SELECT *
FROM camere, occupazioni
WHERE %(param_1)s = occupazioni.id_soggiorno
AND camere.id = occupazioni.id_camera

SELECT *
FROM proprieta
WHERE proprieta.id = %(pk_1)s
```

5. **Visualizzare i soggiorni prenotati in una proprietà**
6. **Visualizzare le proprietà disponibili in base a criteri di ricerca**

```
WITH proprieta_valide AS
(SELECT DISTINCT proprieta.id AS id, proprieta.id_citta AS id_citta,
proprieta.indirizzo AS indirizzo, proprieta.id_tipo_struttura AS
id_tipo_struttura, proprieta.valutazione_media AS valutazione_media,
proprieta.num_valutazioni AS num_valutazioni, utenti.nome AS nome,
utenti.cognome AS cognome
FROM proprieta
INNER JOIN proprietari ON proprietari.id = proprieta.id_proprietario
INNER JOIN utenti ON utenti.id = proprietari.id
INNER JOIN servizi ON proprieta.id = servizi.id_proprieta
WHERE proprieta.id_proprietario != %(id_proprietario_1)s
AND proprieta.data_rimozione IS NULL
AND proprieta.id_citta IN (%(id_citta_1_1)s)
AND proprieta.id_tipo_struttura IN (%(id_tipo_struttura_1_1)s)
AND servizi.id_amenita IN (%(id_amenita_1_1)s, %(id_amenita_1_2)s)
GROUP BY proprieta.id
HAVING count(*) = %(count_1)s),
camere_libere AS
(SELECT DISTINCT camere.id AS anon_1, camere.num_ospiti AS num_ospiti,
proprieta_valide.id AS id, proprieta_valide.id_citta AS id_citta,
proprieta_valide.indirizzo AS indirizzo, proprieta_valide.id_tipo_struttura
AS id_tipo_struttura, proprieta_valide.valutazione_media AS
valutazione_media, proprieta_valide.num_valutazioni AS num_valutazioni,
proprieta_valide.nome AS nome, proprieta_valide.cognome AS cognome
FROM proprieta_valide
INNER JOIN camere ON proprieta_valide.id = camere.id_proprieta
LEFT OUTER JOIN occupazioni ON camere.id = occupazioni.id_camera
LEFT OUTER JOIN soggiorni ON soggiorni.id = occupazioni.id_soggiorno
```

```
WHERE camere.data_rimozione IS NULL
AND (
  soggiorni.id IS NULL
  OR soggiorni.data_cancellazione IS NOT NULL
  OR soggiorni.check_out < %(check_out_1)s
  OR soggiorni.check_in > %(check_in_1)s)
)
SELECT camere_libere.id, camere_libere.id_citta AS id_citta,
camere_libere.indirizzo AS indirizzo, camere_libere.id_tipo_struttura AS
id_tipo_struttura, camere_libere.nome AS nome, camere_libere.cognome AS
cognome, camere_libere.valutazione_media AS valutaione_media,
camere_libere.num_valutazioni AS num_valutazioni
FROM camere_libere
GROUP BY camere_libere.id
HAVING sum(camere_libere.num_ospiti) >= %(sum_1)s
```

7. **Inserire una recensione**
8. **Visualizzare la valutazione media di una proprietà**

```
SELECT valutazione_media
FROM proprieta
WHERE proprieta.id = %(id)s
```

9. **Visualizzare la valutazione media di un proprietario**

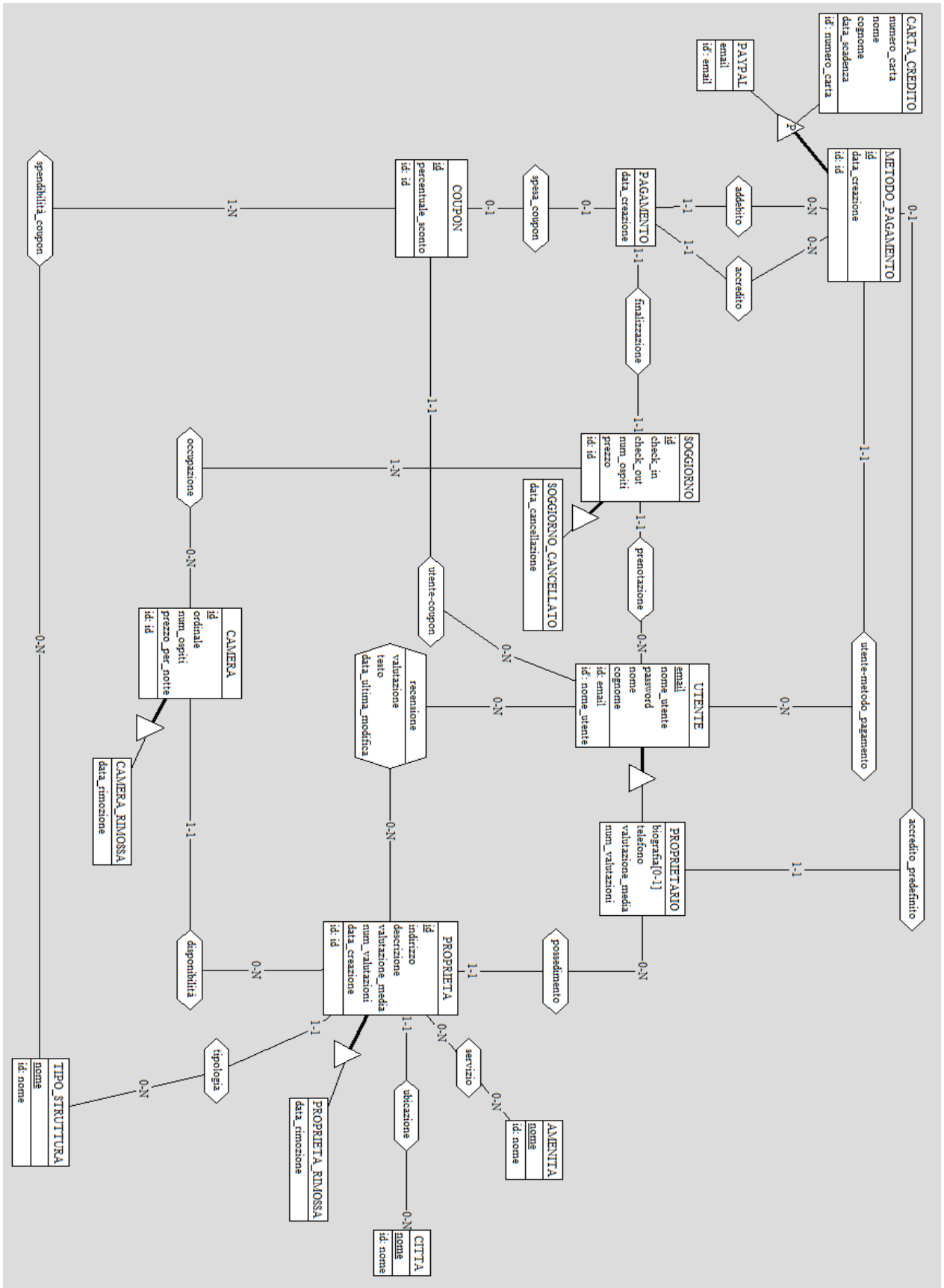
```
SELECT valutazione_media
FROM proprietario
WHERE proprietario.id = %(id)s
```

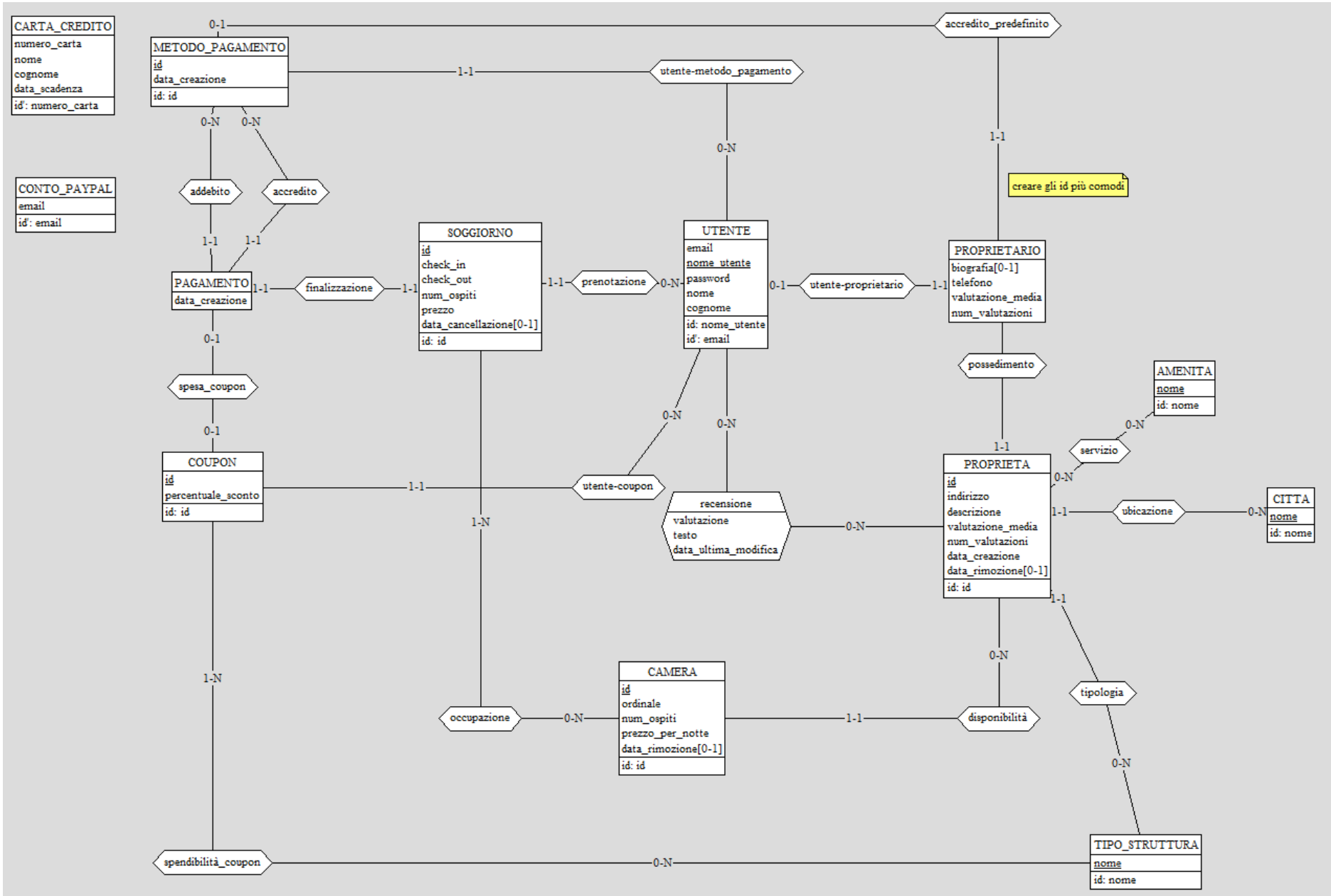
10. **Visualizzare i coupon applicabili**

```
SELECT *
FROM coupons
LEFT OUTER JOIN pagamenti ON coupons.id = pagamenti.id_coupon
LEFT OUTER JOIN spendibilita_coupons ON coupons.id =
spendibilita_coupons.id_coupon
WHERE spendibilita_coupons.id_tipo_struttura = %(id_tipo_struttura_1)s
AND coupons.id_utente = %(id_utente_1)s
AND pagamenti.id_coupon IS NULL
```

11. Visualizzare i 10 proprietari migliori
12. Visualizzare le 10 proprietà migliori
13. Visualizzare le 5 città che hanno ospitato più soggiorni negli ultimi 30 giorni

14. Visualizzare le amenità dalla più alla meno presente nelle proprietà con valutazione maggiore o uguale a 4.5/5


CARTA_CREDITO
numero_carta
nome
cognome
data_scadenza
id': numero_carta
PAYPAL
email
id': email
P
METODO_PAGAMENTO
id
data_creazione
id: id
addebito
accredito
0-N
1-1
PAGAMENTO
id
data_creazione
finalizzazione
SOGGIORNO
id
check_in
check_out
num_ospiti
prezzo
id: id
SOGGIORNO_CANCELLATO
data_cancellazione
prenotazione
UTENTE
email
nome_utente
password
nome
cognome
id: email
id': nome_utente
utente-metodo_pagamento
accredito_predefinito
0-1
spesa_coupon
COUPON
id
percentuale_sconto
id: id
spendibilità_coupon
utente-coupon
occupazione
CAMERA
id
ordinale
num_ospiti
prezzo_per_notte
id: id
CAMERA_RIMOSSA
data_rimozione
recensione
valutazione
testo
data_ultima_modifica
PROPRIETARIO
biografia[0-1]
telefono
valutazione_media
num_valutazioni
possedimento
PROPRIETA
id
indirizzo
descrizione
valutazione_media
num_valutazioni
data_creazione
id: id
disponibilità
tipologia
servizio
ubicazione
AMENITÀ
nome
id: nome
CITTÀ
nome
id: nome
PROPRIETA_RIMOSSA
data_rimozione
TIPO_STRUTTURA
nome
id: nome


CARTA_CREDITO
numero_carta
nome
cognome
data_scadenza
id': numero_carta
METODO_PAGAMENTO
id
data_creazione
id: id
utente-metodo_pagamento
accredito_predefinito
CONTO_PAYPAL
email
id': email
addebito
accredito
creare gli id più comodi
SOGGIORNO
id
check_in
check_out
num_ospiti
prezzo
data_cancellazione[0-1]
id: id
UTENTE
email
nome_utente
password
nome
cognome
id: nome_utente
id': email
PROPRIETARIO
biografia[0-1]
telefono
valutazione_media
num_valutazioni
PAGAMENTO
data_creazione
finalizzazione
prenotazione
utente-proprietario
possedimento
AMENITA
nome
id: nome
spesa_coupon
servizio
COUPON
id
percentuale_sconto
id: id
utente-coupon
recensione
valutazione
testo
data_ultima_modifica
PROPRIETA
id
indirizzo
descrizione
valutazione_media
num_valutazioni
data_creazione
data_rimozione[0-1]
id: id
ubicazione
CITTA
nome
id: nome
CAMERA
id
ordinale
num_ospiti
prezzo_per_notte
data_rimozione[0-1]
id: id
occupazione
disponibilità
tipologia
spendibilità_coupon
TIPO_STRUTTURA
nome
id: nome

Progettazione dell'applicazione

Ho realizzato un semplice sito web con il framework Flask. Ho scelto questa tecnologia perchè permette in modo semplice e veloce di visualizzare schermate con pulsanti, tabelle, navigazione tra pagine ecc. L'applicazione usa Python e HTML con Bootstrap. In particolare, per la gestione del database ho fatto uso interamente dell'**object-relational-mapper (ORM)** fornito dalla libreria Python **SQLAlchemy**. In questo modo, ho potuto adottare un approccio "code-first" definendo prima le classi del modello in Python (file models.py), che successivamente sono mappate automaticamente da SQLAlchemy in tabelle. L'applicazione "sotto" si interfaccia con un server MySql, ma essendo il modello e le query definite nel codice indipendenti dal linguaggio SQL, il DBMS sottostante può essere facilmente cambiato (SQLite ecc.). Ho scelto questa soluzione per poter sfruttare il semplice approccio a oggetti fornito da Python e per semplificare la scrittura di query con l'interfaccia messa a disposizione da SQLAlchemy. Per interrogazioni particolarmente complicate, è sempre possibile eseguire "raw" SQL.

Il sito è molto semplice e implementa solo i meccanismi fondamentali per testare il dominio di applicazione, tralasciando aspetti come la sicurezza ecc.

Homepage:
- puoi prenotare un soggiorno scegliendo città, check-in, check-out, numero di ospiti e (opzionalmente) un numero a piacere di amenità e/o tipi di struttura
- in basso viene visualizzato il tuo prossimo soggiorno

Schermata di prenotazione:
- seleziona le camere da prenotare, metodo di pagamento e opzionale coupon
- effettua il pagamento

Dashboard:
- una semplice dashboard di gestione delle proprietà
- per ogni proprietà è possibile
    - vedere le camere, aggiungerne di nuove o rimuoverle, aggiungere o rimuovere amenità, modificare la descrizione ecc.
    - vedere le prenotazioni
    - vedere le recensioni
    - rimuovere la proprietà

Prenotazioni:
- una tabella con i tuoi soggiorni e i loro status
    - scrivi le recensioni dei soggiorni in corso o completati
    - annulla le prenotazioni

Profilo:
- informazioni di base
- alcune modifiche possibili
- gestisci i tuoi metodi di pagmento