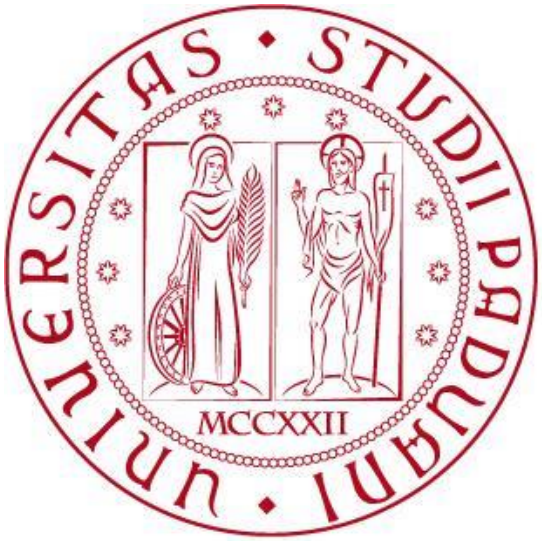


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Dipartimento di Ingegneria
dell'Informazione

Corso di Laurea in
Ingegneria Informatica

Basi di dati A.A. 2017/2018

Progettazione Base di Dati

HOME RENTING

A cura di:
Stefano Ivancich
Alexandru Enache

Docente:
Giorgio Maria Di Nunzio

INDICE

1. Descrizione del progetto

Si vuole realizzare una base di dati per un'applicazione online che mette in contatto persone in cerca di un alloggio per brevi periodi, con persone (privati) che mettono a disposizione una tale abitazione nelle principali città Italiane.
Per la quale si vogliono rappresentare i dati degli utenti, delle prenotazioni, degli alloggi, della città di collocazione dell'alloggio, dei servizi standard di cui è dotato l'alloggio e delle tipologie standard (appartamento, casa, ecc.) che può assumere l'alloggio, storicizzando i vari eventi possibili.

1.1. Requisiti strutturati

Frasi per Utente:
Per ogni utente, identificato univocamente tramite id, rappresentiamo i seguenti dati personali: l'indirizzo e-mail che deve essere unico, password, nome, cognome, sesso, la data di nascita, telefono, la fotografia, descrizione, lingue parlate.
Un utente viene chiamato "Host" se in una prenotazione è il possessore dell'alloggio, altrimenti viene chiamato "Ospite". Tale nomenclatura viene utilizzata nel seguito del documento allo scopo di presentare più chiaramente il progetto.
Ogni utente può essere sia Ospite che Host e può possedere zero, uno o più alloggi.
Ogni utente può prenotare un alloggio che non sia di sua proprietà.
Ogni Ospite può avere associato nessuna, una o più valutazioni, fatte dagli Host dopo la conclusione della prenotazione. L'Host deve effettuare una e una sola recensione sull'Ospite dopo che la prenotazione risulta conclusa. Mentre l'Ospite alla conclusione della prenotazione deve effettuare una e una sola recensione dell'alloggio.

Frasi per Alloggio:
Per ogni alloggio, identificato univocamente tramite id, rappresentiamo i seguenti dati: nome, prezzo per notte, descrizione, numero posti, la via, le fotografie.
Ogni alloggio appartiene a uno e un solo utente ed è collocato in una sola città.
Ogni alloggio è dotato di almeno un servizio e ogni alloggio deve essere di una e una sola tipologia (casa, appartamento, ecc.).
Ogni alloggio può essere incluso in nessuna, una o più prenotazioni e può avere associata nessuna, una o più recensioni ricevuta dall'Ospite alla conclusione della prenotazione. L'Ospite alla conclusione della prenotazione effettua una e una sola recensione dell'alloggio.
Nel caso il cui l'alloggio sia stato venduto o a causa di altri motivi che rendono impossibili effettuare altre prenotazioni allora le informazioni relative all'alloggio possono essere archiviate.
Un alloggio può essere eliminato definitivamente dal database solo se non sono state mai effettuate prenotazioni su di esso.

Frasi per Servizio:
Per ogni servizio, identificato univocamente dal nome, rappresentiamo la sua descrizione.
Ogni servizio può essere nelle dotazioni di nessuno, uno o più alloggi.
I servizi sono standard (stabiliti dal gestore della piattaforma) e sono 11, di seguito i nomi: Ascensore, Animali domestici ammessi, Internet, Parcheggio Gratuito Incluso, TV, È permesso fumare, Aria condizionata, Riscaldamento, Disponibile per eventi, Accessibile per chi ha mobilità ridotta, Per famiglie e bambini.

Frasi per Città:
Per ogni città, identificata dal nome, rappresentiamo una descrizione, necessaria a presentare il luogo di ubicazione dell'alloggio.
Ogni città può contenere nessuno, uno o più alloggi.

Frasi per TipoAlloggio:
Per ogni tipologia di alloggio, identificato univocamente dal nome, rappresentiamo la sua descrizione.
Le tipologie di alloggio non possono intersecarsi, ovvero ogni alloggio può essere di una e una sola tipologia.
Le tipologie di alloggio sono standard (stabilite dal gestore della piattaforma) e sono 7, di seguito i nomi: Appartamento intero, Camera singola, Camera condivisa, Dèpandance, Suite degli ospiti, Casa Vacanze, Pensione.

Frasi per Prenotazione:
Per ogni prenotazione, identificato univocamente tramite id, rappresentiamo la data di check in, l'alloggio associato e l'utente (Ospite) associato, la data di check out, il prezzo (complessivo pagato dall'Ospite), il numero di posti prenotati e la data di richiesta.
Ogni prenotazione è relativa a uno e un solo utente e ha associato uno e un solo alloggio.

Inoltre, la prenotazione ha associato uno stato che può assume uno dei seguenti valori: *Sospeso*, *Rifiutata*, *Annullata*, *Accettata*, *Cancellata*, *Conclusa*, *In Corso*. L'associazione di uno di questi stati avviene nel seguente modo:

al momento della effettuazione della prenotazione da parte dell'utente (Ospite) su un determinato alloggio, alla prenotazione viene associato lo stato *Sospeso*, in quanto il proprietario dell'alloggio (Host) dovrà leggere la proposta di prenotazione e decidere se accettarla o meno.
Nel caso in cui il Host accetta la prenotazione allora la prenotazione passa allo stato di *Accettata*, altrimenti allo stato di *Rifiutata*. Però se l'Ospite decide di eliminare la prenotazione prima che l'Host abbia deciso se accettare allora la prenotazione passa dallo stato di *Sospeso* allo stato di *Annullata*.
Successivamente, se la prenotazione si trova nello stato di *Accettata* e il cliente decide di eliminare la sua prenotazione prima della data di check-in allora lo stato della prenotazione assume il valore *Cancellata*.
Infine, se la prenotazione è nello stato di *Accettata* e l'Ospite soggiorna nell'alloggio allora la prenotazione assume lo stato di *In Corso* durante il periodo di check-in e di check-out, mentre alla fine di tale periodo (quando l'Ospite rilascia l'alloggio) la prenotazione passa allo stato di *Conclusa*. Inoltre, solo in questo ultimo caso l'Ospite dovrà effettuare una e una sola recensione dell'alloggio (associata alla prenotazione) e l'Host dovrà effettuare una e una sola recensione dell'Ospite (associata alla prenotazione).

1.2. Operazioni sulla base di dati

Tabella delle Operazioni:

N°	Operazione	Tipo	Frequenza
1	Inserimento nuovo Utente	Inserimento	100/Giorno
2	Inserimento nuovo Alloggio da Utente	Inserimento	10/Giorno
3	Inserimento di una Città	Inserimento	1/Mese
4	Inserimento di un Servizio	Inserimento	1/Anno
5	Inserimento di un Tipo di alloggio	Inserimento	1/Anno
6	Aggiornamento di una Città (Descrizione)	Modifica	1/Mese
7	Aggiornamento dati Utente	Modifica	50/Giorno
8	Aggiornamento dati Alloggio	Modifica	5/Giorno
9	Aggiornamento Servizi	Modifica	1/Anno
10	Aggiornamento Tipo di alloggio	Modifica	1/Anno
11	Effettuazione prenotazione da parte di Utente	Inserimento	350/Giorno
12	Lista degli alloggi prenotabili in una Città	Interrogazione	2’000/Giorno
13	Lista degli alloggi prenotabili in una Città che offrono determinati servizi	Interrogazione	1’500/Giorno
14	Visualizzazione dati di un Alloggio	Interrogazione	5’000/Giorno
15	Visualizzazione dati di un Utente	Interrogazione	1’000/Giorno
16	Visualizzazione recensioni ricevute di un Alloggio	Interrogazione	5’000/Giorno
17	Visualizzazione recensioni ricevute di un Utente	Interrogazione	1’000/Giorno
18	Lista prenotazioni ricevute (eseguita dall’Host)	Interrogazione	10’000/Giorno
19	Lista prenotazioni effettuate (eseguita dall’Ospite)	Interrogazione	1’000/Giorno
20	Recensione dell’alloggio da parte di Utente (Ospite)	Inserimento	300/Giorno
21	Recensione dell’Utente (Ospite) da parte di altro utente (Host)	Inserimento	300/Giorno
22	Archiviazione di un Alloggio	Modifica	3/Giorno
23	Eliminazione di un Alloggio	Cancellazione	10/Giorno
24	Aggiornamento stato prenotazione	Modifica	350/Giorno

1.3. Glossario

Termine	Descrizione	Sinonimi	Collegamenti
Utente	Utente del portale che può svolgere il duplice ruolo di *Ospite*, ovvero effettuare la prenotazione di un alloggio, e di *Host*, ovvero pubblicare almeno un alloggio disponibile a essere affittato.	Ospite, Host	Alloggio, prenotazione, Recensione
Alloggio	Spazio abitativo reso disponibile per essere affitta dall’Utente (il Host)	Abitazione	Utente, Servizio, TipoAlloggio, Prenotazione
Servizio	Insieme dei servizi stabiliti dal gestore del portale e resi come standard che un Alloggio può possedere.		Alloggio
TipoAlloggio	Insieme delle tipologie di abitazione stabiliti dal gestore del portale e resi come standard.		Alloggio
Città	Città della nazione in cui opera il portale.		Alloggio
Prenotazione	Prenotazione effettuata dall’utente (Ospite) per affittare un Alloggio per un certo periodo di tempo		Alloggio, Utente

2. Progettazione Concettuale

2.1. Modello Concettuale: Entità-Associazione (E-R)

Di seguito lo schema concettuale prodotto per la rappresentazione della realtà di interesse:

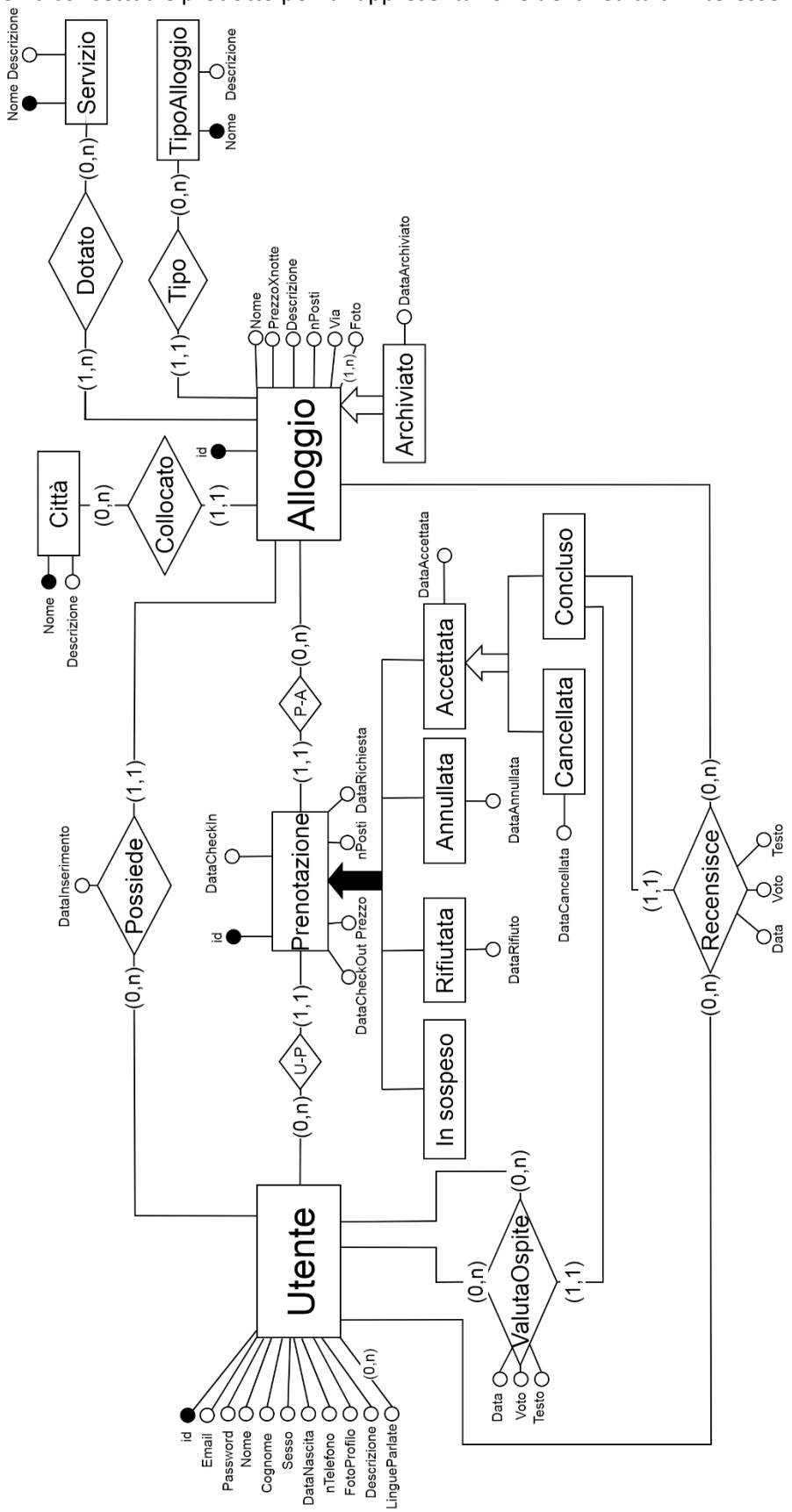


Figura 1: Schema Entità-Associazione (E-R).

2.2. Dizionario dei dati

2.2.1. Entità

Entità	Descrizione	Attributi	Identificatore
Utente	Può possedere un alloggio; può effettuare una prenotazione; può recensire un alloggio; può valutare un utente ospite.	id, Email, Password, Nome, Cognome, Sesso, DataNascita, nTelefono, FotoProfilo, Descrizione, LinguaParlate	id
Alloggio	Collocato in una città; appartiene a un utente Host; è di una tipologia unica; si possono effettuare prenotazioni su di esso; è dotato di almeno un servizio.	id, Nome, PrezzoXnotte, Descrizione, nPosti, Via, Foto	id
Città	Vi possono essere collocati degli alloggi	Nome, Descrizione	Nome
Archiviato	Alloggio che non è più reso disponibile per prenotazioni o non più posseduto dall'Host; non può essere prenotato.	DataArchiviato	id
Servizio	Servizio che può essere offerto dagli alloggi	Nome, Descrizione	Nome
TipoAlloggio	Tipologia di alloggio (Appartamento, camera singola, ecc.).	Nome, Descrizione	Nome
Prenotazione	Prenotazione di un alloggio	DataCheckIn, DataCheckOut, Prezzo, nPosti, DataRichiesta	id
In sospeso	Prenotazione in attesa di conferma da parte dell'Host.		id
Rifiutata	Prenotazione che un Host si è rifiutato di accettare.	DataRifiutata	id
Annullata	Prenotazione annullata dall'ospite, prima di essere accettata dall'Host.	DataAnnullata	id
Accettata	Prenotazione Accettata dall'Host.	DataAccettata	id
Cancellata	Prenotazione cancellata dall'Ospite dopo essere stata accettata dall'Host.	DataCancellata	id
Concluso	Soggiorno concluso da parte dell'Ospite nel periodo previsto dalla prenotazione.		id

2.2.2. Associazioni

Associazione	Attributi	Entità collegate
Possiede	DataInserimento	Utente (0,N), Alloggio (1,1)
Collocato		Alloggio (1,1), Città (0,N)
Dotato		Alloggio (1,N), Servizio (0,N)
Tipo		Alloggio (1,1), TipoAlloggio (0,N)
ValutaOspite	Data, Voto, Testo	Utente (0,N), Utente (0,N), Concluso (1,1)
Recensisce	Data, Voto, Testo	Utente (0,N), Alloggio (0,N), Concluso (1,1)
U-P		Utente (0,N), Prenotazione (1,1)
P-A		Prenotazione (1,1), Alloggio (0,N)

2.3. Schema Concettuale, Regole di vincolo

Per rispettare la rappresentazione fedele della realtà di interesse descritta in sezione 1.1 è necessario definire delle regole di vincolo per concetti altrimenti non esprimibili utilizzando il modello E.R.

- Regola di Vincolo 1:
  Un utente non deve effettuare una prenotazione su un alloggio di cui è proprietario.
- Regola di Vincolo 2:
  Un alloggio non deve essere eliminato definitivamente dal database se è stata fatta almeno una prenotazione su di esso.

3. Progettazione Logica

3.1. Ristrutturazione schema E.R.

Tabella dei volumi

Concetto	Tipo	Volume	Descrizione
Utente	E	50’000	100 utenti nuovi al giorno, in un anno sono dell’ordine dei 50k
Alloggio	E	5’000	10% degli utenti ha un alloggio
Città	E	50	Principali città italiane
Archiviato	E	1000	20% degli alloggi sono archiviati
Servizio	E	11	Predefiniti
TipoAlloggio	E	7	Predefiniti
Concluso	E	100’000	In media 2 soggiorni all’anno per utente
Cancellata	E	5’000	5% dei soggiorni viene cancellato
Accettata	E	106’000	Concluso + cancellato + soggiorni in corso
Rifiutata	E	10’000	10% delle prenotazioni viene rifiutato
Annullata	E	10’000	10% delle prenotazioni viene annullato
In sospeso	E	300	(2 prenotazioni all’anno * 50k utenti) /365 contando che le prenotazioni vangano accettare o rifiutate nell’arco di 24h
Prenotazione	E	126’300	Accettata + rifiutata + annullata + in sospeso
Possiede	A	5’000	Volume alloggi, contando che alcuni utenti possono avere più di un alloggio
Collocato	A	5’000	Una sola città per alloggio
Dotato	A	20’000	In media 4 servizi per alloggio
Tipo	A	5’000	Un solo tipo per alloggio
ValutaOspite	A	100’000	Una recensione per ogni soggiorno concluso
Recensisce	A	100’000	Una recensione per ogni soggiorno concluso
U-P	A	100’000	Per ogni prenotazione un associazione
P-A	A	100’000	Per ogni prenotazione un associazione

Tabella Accessi Op. 12
Lista degli alloggi prenotabili in una Città

Concetto	Costrutto	Accessi	Tipo
Città	E	1	L
Collocato	A	5’000	L
Alloggio	E	5’000	L
P-A	A	100’000	L
Prenotazione	E	100’000	L
Rifiutata	E	10’000	L
Annullata	E	10’000	L
Cancellata	E	5’000	L

Tabella Accessi Op. 13
Lista degli alloggi prenotabili in una Città offrono determinati servizi

Concetto	Costrutto	Accessi	Tipo
Città	E	1	L
Collocato	A	5’000	L
Alloggio	E	5’000	L
Dotato	A	20’000	L
P-A	A	100’000	L
Prenotazione	E	100’000	L
Rifiutata	E	10’000	L
Annullata	E	10’000	L
Cancellata	E	5’000	L

Eliminazione della generalizzazione Prenotazione
Date le tabelle degli accessi precedenti è evidente che mantenendo la generalizzazione dell'entità "Prenotazione" in entità separate comporterebbe un costo notevole perché si dovrebbe andare a visualizzare per ogni entità se la Prenotazione è stata *Annullata*, *Rifiutata*, *Cancellata* ovvero se l'alloggio è prenotabile o no in quella data.
Quindi si è deciso di accorpare tutte le entità figlie nell'entità padre "Prenotazione", aggiungendo un attributo "Stato" che indica uno dei seguenti stati (esposti nella sezione 1.1): *In sospeso*, *Rifiutata*, *Annullata*, *In corso*(Accettata), *Cancellata*(Accettata) e *Concluso* (Accettata).
Gli attributi delle entità figlie, quindi, vengono aggiunti alla entità "Prenotazione" e potranno assumere valori nulli, in quanto non significativi per alcune occorrenze del genitore, quindi la loro cardinalità sarà (0,1).

Inoltre, vengono aggiunti i seguenti vincoli:
1. "Una prenotazione non deve ricevere una recensione se non è nello stato *Concluso*";
2. "Un utente non deve ricevere una valutazione, quando in veste di Ospite, riguardo a un soggiorno se la relativa prenotazione non è in stato *Concluso*".

Eliminazione della generalizzazione Alloggio
Siccome sull'entità archiviato non si eseguono operazioni, si è deciso di accorpare l'entità "Archiviato" in "Alloggio" aggiungendo l'attributo *Archiviato* che può assumere i valori TRUE oppure FALSE.
L'attributo "DataArchiviato" viene aggiunto ad Alloggio e può assumere valori nulli.

Inoltre, vengono aggiunti i seguenti vincoli:
1. "Un alloggio non deve essere prenotato se è Archiviato (valore TRUE sull'attributo *Archiviato*)";
2. "Un alloggio non deve essere modificato se è Archiviato (valore TRUE sull'attributo *Archiviato*)";
3. "Un alloggio non deve essere incluso nei risultati della ricerca da parte dell'utente se è Archiviato (valore TRUE sull'attributo *Archiviato*)".

Eliminazione dell'attributo multivalore "Foto"
Il modello relazionale non permette di rappresentare attributi multivalore, quindi viene aggiunta un entità "Foto" identificata da un id e come attributo "Link". L'entità "Foto" è collegata all'entità "Alloggio" tramite un'associazione uno a molti in quanto una foto può appartenere a un solo alloggio, mentre un alloggio può avere zero, una oppure molte foto associate.

Eliminazione dell'attributo multivalore "Lingue parlate"
Viene aggiunta l'entità "Lingue" identificata dall'attributo *Nome*. Tale entità è collegata all'entità "Utente" tramite una relazione molti a molti in quanto una lingua può essere parlata da nessuno, uno o più più utenti mentre un utente può parlare nessuna (se l'utente decide di non indicare la lingua), una o più lingue.

Di seguito, in figura 2, è rappresentato lo schema E.R. concettuale ristrutturato.

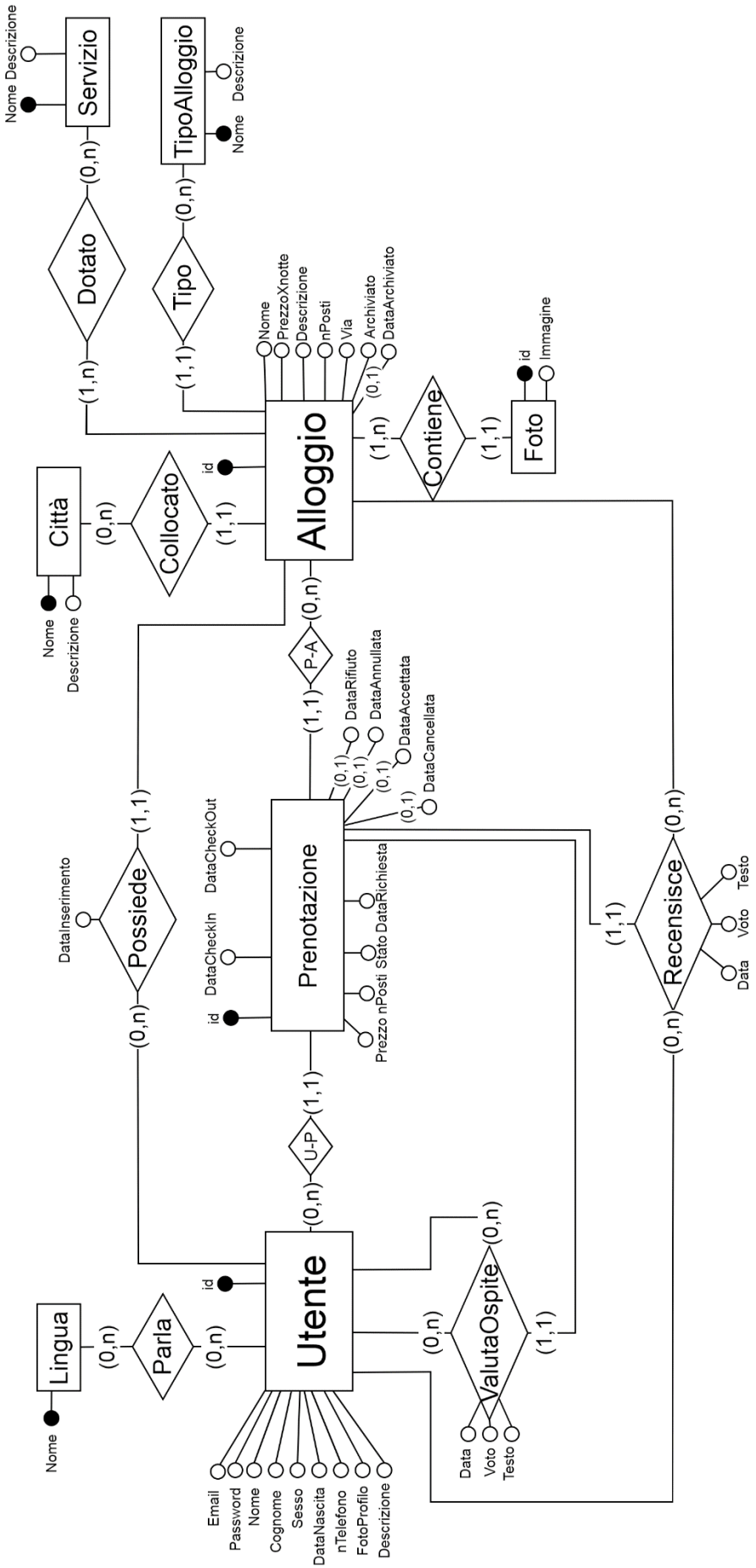


Figura 2: Modello Entità-Associazione (E-R) ristrutturato.

3.2. Modello Logico: Relazionale

Di seguito in figura 3 lo schema logico prodotto per la rappresentazione della realtà di interesse:

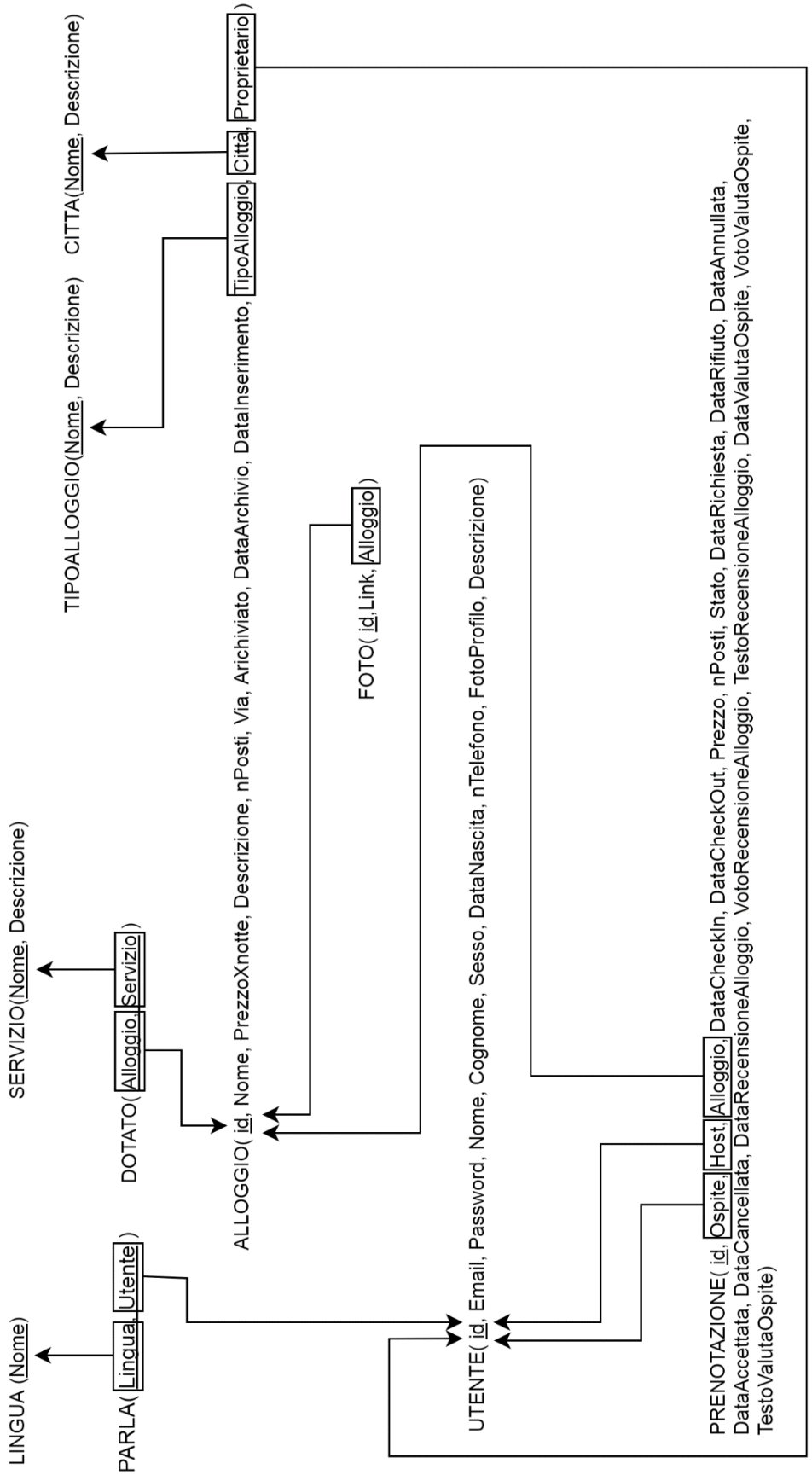


Figura 3: Modello Logico: Relazionale.

3.3. Schema Logico, Regole di vincolo

- RV1: Un utente non deve effettuare una prenotazione su un alloggio di cui è proprietario.
- RV2: Un alloggio non deve essere eliminato definitivamente dal database se sono state effettuate prenotazioni su di esso.
- RV3: Gli attributi DataRecensioneAlloggio, VotoRecensionAlloggio, TestoRecensioneAlloggio, DataValutaOspite, VotoValutaOspite, TestoValutaOspite possono essere riempiti solo se la prenotazione se è nello stato *Concluso*.
- RV4: Un alloggio non deve essere prenotato se è Archiviato.
- RV5: Un alloggio non deve essere incluso nei risultati della ricerca se è *Archiviato*.
- RV6: Gli attributi id, Email, Password, Nome, Cognome, Sesso, DataNascita, nTelefono di Utente non devono essere nulli.
- RV7: Gli attributi Nome, Prezzoxnotte, Descrizione, nPosti, Via, DataInserimento di Alloggio non devono essere nulli.
- RV8: Gli attributi DataCheckIn, DataCheckOut, Prezzi, nPosti, Stato, DataRichiesta di Prenotazione non devono essere nulli

4. Codice SQL

4.1. Struttura

```
CREATE TABLE lingua
(
    nome VARCHAR(10) PRIMARY KEY
);

CREATE TABLE utente
(
    id SERIAL PRIMARY KEY, /*SERIAL equivale ad INTEGER AUTO_INCREMENT per
    postgre*/
    email VARCHAR(40) UNIQUE, /*No 2 utenti con la stessa email*/
    password VARCHAR(40) NOT NULL,
    nome VARCHAR(40) NOT NULL,
    cognome VARCHAR(40) NOT NULL,
    sesso BOOLEAN NOT NULL, /*FALSE=maschio*/
    datanascita DATE NOT NULL,
    ntelefono NUMERIC(10,0) NOT NULL CHECK(ntelefono>999999999), /*i cellulari
    hanno 10 cifre*/
    fotoprofilo VARCHAR(256) DEFAULT 'utente.png', /*immagine di default*/
    descrizione VARCHAR(256) /*utente può descrivere in max 256 caratteri*/
);

CREATE TABLE parla
(
    lingua VARCHAR(10) NOT NULL,
    utente INTEGER NOT NULL,
    PRIMARY KEY (lingua, utente),
    FOREIGN KEY (lingua) REFERENCES lingua(nome)
        ON DELETE CASCADE ON UPDATE CASCADE, /*Se viene eliminata una lingua và
        eliminata anche la riga*/
    FOREIGN KEY (utente) REFERENCES utente(id)
        ON DELETE CASCADE ON UPDATE CASCADE /*Se viene eliminato un utente vanno
        eliminate le righe che dicono quali lingue parlava*/
);

CREATE TABLE tipoalloggio
(
    nome VARCHAR(20) PRIMARY KEY,
    descrizione VARCHAR(256) /*max 256 caratteri*/
);

CREATE TABLE citta
(
    nome VARCHAR(20) PRIMARY KEY,
    descrizione VARCHAR(256) /*max 256 caratteri*/
);

CREATE TABLE alloggio
(
    id SERIAL PRIMARY KEY, /*SERIAL equivale ad INTEGER AUTO_INCREMENT per
    postgre*/
    nome VARCHAR(50) NOT NULL,
    prezzoxnotte NUMERIC(6,2) NOT NULL CHECK(prezzoxnotte>0), /*numero che va da
    0 a 9999.99*/
    nposti INTEGER NOT NULL CHECK(nposti>0 AND nposti<10), /*Almeno 1 posto per
    essere valido*/
    descrizione VARCHAR(256), /*max 256 caratteri*/
    via VARCHAR(40) NOT NULL,
    archiviato BOOLEAN NOT NULL,
    dataarchivio DATE, /*Se NULL vuol dire che non è mai stato archiviato*/
```

```
    datainserimento DATE NOT NULL,
    citta VARCHAR(20) NOT NULL,
    tipoalloggio VARCHAR(20) NOT NULL,
    proprietario INTEGER NOT NULL,
    FOREIGN KEY (citta) REFERENCES citta(nome)
        ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY (tipoalloggio) REFERENCES tipoalloggio(nome)
        ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY (proprietario) REFERENCES utente(id)
        ON DELETE NO ACTION ON UPDATE CASCADE /*Se un utente viene eliminato,
        non vengono eliminati i suoi alloggi perchà potrebbero riferire a delle
        prenotazioni*/
);

CREATE TABLE foto
(
    id SERIAL PRIMARY KEY,
    link VARCHAR(256) DEFAULT 'ImmagineVuota.png',
    alloggio INTEGER NOT NULL,
    FOREIGN KEY (alloggio) REFERENCES alloggio(id)
        ON DELETE CASCADE ON UPDATE CASCADE /*Se viene eliminato un alloggio,
        devono essere eliminate tutte le sue foto*/
);

CREATE TABLE servizio
(
    nome VARCHAR(40) PRIMARY KEY,
    descrizione VARCHAR(256) /*max 256 caratteri*/
);

CREATE TABLE dotato
(
    alloggio INTEGER NOT NULL,
    servizio VARCHAR(40) NOT NULL,
    PRIMARY KEY (alloggio, servizio),
    FOREIGN KEY (alloggio) REFERENCES alloggio(id)
        ON DELETE CASCADE ON UPDATE CASCADE, /*Se viene eliminato un alloggio,
        vanno eliminate le righe che dicono quali servizi aveva*/
    FOREIGN KEY (servizio) REFERENCES servizio(nome)
        ON DELETE NO ACTION ON UPDATE CASCADE
);

CREATE DOMAIN statoprenotazione AS VARCHAR(12)
CHECK(VALUE ='Sospeso' OR VALUE ='Rifiutata' OR VALUE ='Annullata' OR VALUE
='Accettata' OR
VALUE ='Cancellata' OR VALUE ='Conclusa' OR VALUE='In Corso');

CREATE TABLE prenotazione
(
    id SERIAL PRIMARY KEY,
    datacheckin DATE NOT NULL,
    datacheckout DATE NOT NULL,
    prezzo NUMERIC(7,2) NOT NULL CHECK(prezzo>0),
    nposti INTEGER NOT NULL CHECK(nposti>0 AND nposti<10),
    stato statoprenotazione NOT NULL,
    datarichiesta DATE NOT NULL,
    datarifiuto DATE, /*Se NULL vuol dire che non è stata ancora Rifiutata*/
    dataannullata DATE, /*Se NULL vuol dire che non è stata ancora Annullata*/
    dataaccettata DATE, /*Se NULL vuol dire che non è stata ancora Accettata*/
    datacancellata DATE, /*Se NULL vuol dire che non è stata ancora Cancellata*/
    datarecensionealloggio DATE, /*Se NULL vuol dire che non è stato ancora
    valutato*/
```

```
    /*Voto deve essere tra 0 e 10, se NULL vuol dire che non è stato ancora
    valutato*/
    votorecensionealloggio INTEGER
        CHECK((votorecensionealloggio>=0 AND votorecensionealloggio<=10) OR
    votorecensionealloggio=NULL),
    testorecensionealloggio VARCHAR(256), /*max 256 caratteri, Se NULL vuol dire
    che non è stato ancora valutato*/
    datavalutaospite DATE, /*Se NULL vuol dire che non è stato ancora valutato*/
    /*Voto deve essere tra 0 e 10, se NULL vuol dire che non è stato ancora
    valutato*/
    votovalutaospite INTEGER
        CHECK((votovalutaospite>=0 AND votovalutaospite<=10) OR
        votovalutaospite=NULL), testovalutaospite VARCHAR(256), /*max 256
        caratteri, Se NULL vuol dire che non è stato ancora valutato*/
    ospite INTEGER NOT NULL,
    host INTEGER NOT NULL,
    alloggio INTEGER NOT NULL,
    FOREIGN KEY (ospite) REFERENCES utente(id)
        ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY (host) REFERENCES utente(id)
        ON DELETE NO ACTION ON UPDATE CASCADE,
    FOREIGN KEY (alloggio) REFERENCES alloggio(id)
        ON DELETE NO ACTION ON UPDATE CASCADE
);

/* Privilegi per eseguire INSERT, UPDATE, SELECT,... da codice PHP*/
GRANT ALL PRIVILEGES ON alloggio TO webdb;
GRANT ALL PRIVILEGES ON citta TO webdb;
GRANT ALL PRIVILEGES ON dotato TO webdb;
GRANT ALL PRIVILEGES ON foto TO webdb;
GRANT ALL PRIVILEGES ON lingua TO webdb;
GRANT ALL PRIVILEGES ON parla TO webdb;
GRANT ALL PRIVILEGES ON prenotazione TO webdb;
GRANT ALL PRIVILEGES ON servizio TO webdb;
GRANT ALL PRIVILEGES ON tipoalloggio TO webdb;
GRANT ALL PRIVILEGES ON utente TO webdb;
```

4.2. Query

Op. 1 Inserire un Utente

```
INSERT INTO utente
(email, password, nome, cognome, datanascita, sesso, ntelefono)
VALUES ('utente@email', 'password', 'nome', 'cognome', '2018-01-01', TRUE,
3333333333);
```

Op. 2 Inserire un nuovo Alloggio

```
INSERT INTO alloggio
(nome, prezzoxnotte, nposti, via, archiviato, descrizione,
datainserimento, citta, tipoalloggio, proprietario)
VALUES (value1, value2, value3, value3, TRUE, value3, value3, value3,
value3, value3);
```

Op. 3 Inserire una Prenotazione

```
INSERT INTO prenotazione
(ospite, host, alloggio, datacheckin, datacheckout, prezzo, nposti, stato,
datarichiesta)
VALUES (1234, 1234, 1234, '2018-02-01', '2018-02-01', 250, 2, 'Sospeso',
'2018-01-01');
```

Op. 4 Lista delle date in cui l'Alloggio di id=1234 non può essere prenotato

$$\pi_{\substack{datacheckin,\\ datacheckout}} \left(\sigma_{\substack{alloggio=1234\ AND\\ (stato='Accettata'\ OR\ stato='In\ corso')}} (prenotazione) \right)$$

```
SELECT DISTINCT datacheckin, datacheckout
FROM prenotazione
WHERE alloggio=1234 AND (stato='Accettata' OR stato='In Corso');
```

Op. 5 Lista delle Prenotazioni ricevute (come Host)

$$\pi_{\substack{prenotazione.stato\\ prenotazione.id,\\ prenotazione.alloggio,\\ alloggio.nome,\\ prenotazione.ospite,\\ utente.nome,\\ prenotazione.datacheckin,\\ prenotazione.datacheckout,\\ prenotazione.nposti,\\ prenotazione.prezzo}} \left(\sigma_{host=1234}(prenotazione) \bowtie_{alloggio=id} (alloggio) \bowtie_{ospite=id} (utente) \right)$$

```
SELECT DISTINCT p.stato, p.id, p.alloggio, alloggio.nome, p.ospite,
                utente.nome AS nomeOspite, p.datacheckin, p.datacheckout,
                p.nposti, p.prezzo, p.datacancellata, p.datarifiuto,
                p.dataannullata
FROM prenotazione AS p
INNER JOIN alloggio
    ON p.alloggio=alloggio.id
INNER JOIN utente
    ON p.ospite=utente.id
WHERE p.host=1234;
```

Op. 6 Lista delle Recensioni fatte dagli Ospiti sull'Alloggio di id=1234

$$\pi_{\substack{utente.id,\\ utente.nome,\\ utente.fotoprofilo,\\ prenotazione.datarecensionealloggio,\\ prenotazione.testorecensionealloggio,\\ prenotazione.votorecensionealloggio}} \left(\sigma_{\substack{alloggio=1234\ AND\\ datarecensionealloggio\ IS\ NOT\ NULL}} (prenotazione) \bowtie_{ospite=id} (utente) \right)$$

```
SELECT DISTINCT u.id, u.nome, u.fotoprofilo, p.datarecensionealloggio,
p.testorecensionealloggio, p.votorecensionealloggio
FROM prenotazione AS p
INNER JOIN utente AS u
ON p.ospite=u.id
WHERE p.alloggio=1234 AND p.datarecensionealloggio IS NOT NULL;
```

Op. 7 Lista dei 20 Alloggi (non Archiviati) che hanno la media delle recensioni più alta

$$\pi_{\substack{prenotazione.alloggio\\ mediavoti,\\ numero\ voti,\\ alloggio.nome,\\ alloggio.tipoalloggio,\\ alloggio.nposti,\\ alloggio.prezzoxnotte,\\ foto.link}} \left(\sigma_{archiviato=FALSE} \left(\begin{matrix} (alloggio)\mathcal{F}_{AVARAGE(votorecensionealloggio),\ COUNT(votorecensionealloggio)}(prenotazione) \\ \bowtie_{alloggio=id} (alloggio) \bowtie_{alloggio=alloggio} (foto) \end{matrix} \right) \right)$$

```
SELECT DISTINCT p.alloggio, AVG(p.votorecensionealloggio) AS mediavoti,
                COUNT(p.votorecensionealloggio), a.nome, a.tipoalloggio,
                a.nposti, a.prezzoxnotte
FROM prenotazione AS p
RIGHT JOIN alloggio AS a
    ON p.alloggio=a.id
WHERE a.archiviato = FALSE
GROUP BY p.alloggio, a.nome, a.tipoalloggio, a.nposti, a.prezzoxnotte
ORDER BY mediavoti DESC
LIMIT 20;
```

Op. 8 Lista dei 20 Alloggi (non Archiviati) di una Città in ordine di media delle recensioni

$$\pi_{\substack{prenotazione.alloggio\\ mediavoti,\\ numero\ voti,\\ alloggio.nome,\\ alloggio.tipoalloggio,\\ alloggio.nposti,\\ alloggio.prezzoxnotte,}} \left(\sigma_{\substack{alloggio.archiviato=FALSE\\ AND\ alloggio.citta='nome'}} \left(\begin{matrix} (alloggio)\mathcal{F}_{AVARAGE(votorecensionealloggio),\ COUNT(votorecensionealloggio)}(prenotazione) \\ \bowtie_{alloggio=id} (alloggio) \end{matrix} \right) \right)$$

```
SELECT DISTINCT p.alloggio, AVG(p.votorecensionealloggio) AS mediavoti,
                COUNT(p.votorecensionealloggio), a.nome, a.tipoalloggio,
                a.nposti, a.prezzoxnotte
FROM prenotazione AS p
RIGHT JOIN alloggio AS a
    ON p.alloggio=a.id
WHERE a.archiviato = FALSE AND a.citta='nomecitta'
GROUP BY p.alloggio, a.nome, a.tipoalloggio, a.nposti, a.prezzoxnotte
ORDER BY mediavoti DESC
LIMIT 20;
```

Op. 9 Lista degli Alloggi (non Archiviati) di un certo Tipo, di una determinata Città con determinati Servizi, che possono ospitare un certo numero di persone.

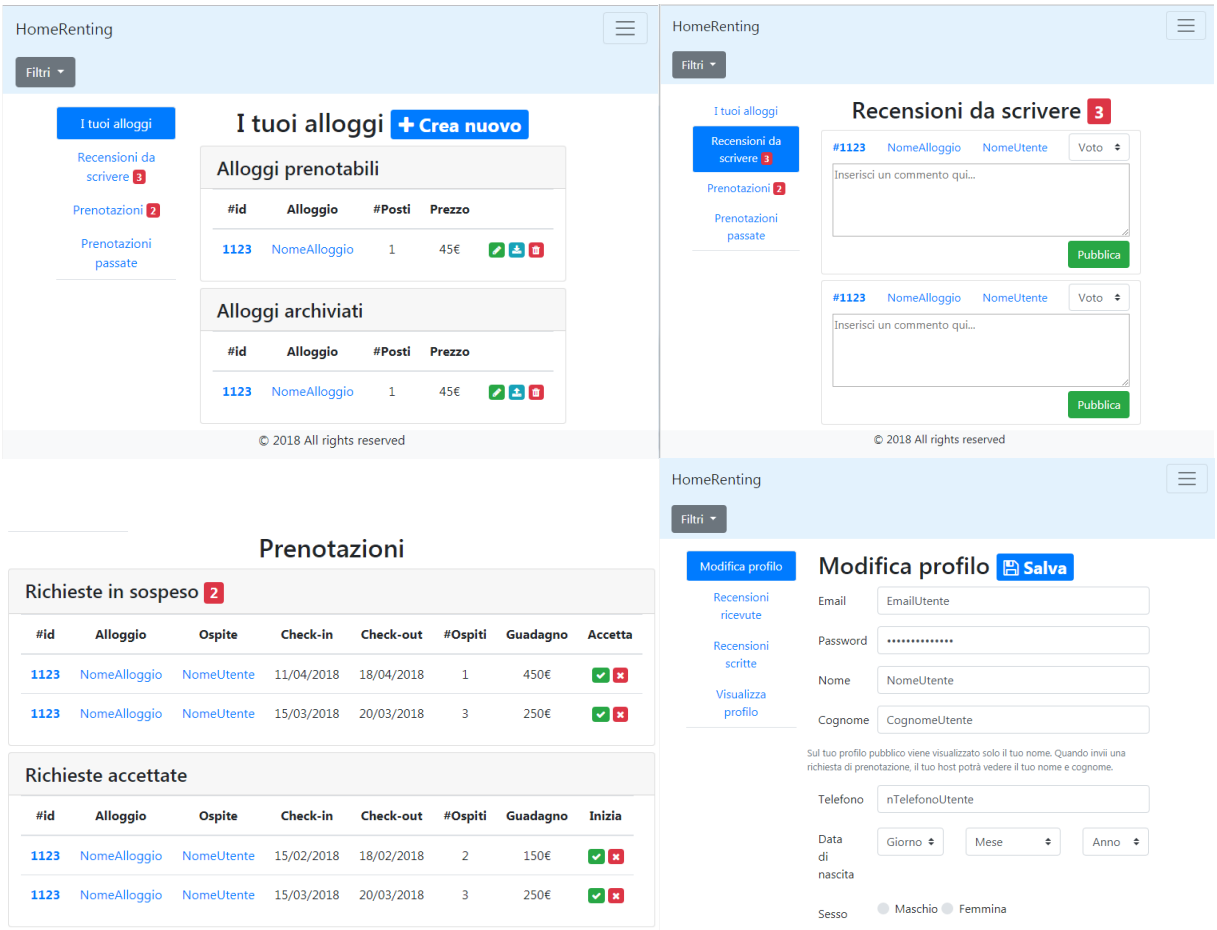
$$\pi_{\substack{prenotazione.alloggio\\ mediavoti,\\ numero\ voti,\\ alloggio.nome,\\ alloggio.tipoalloggio,\\ alloggio.nposti,\\ alloggio.prezzoxnotte,}} \left(\sigma_{\substack{alloggio.archiviato=FALSE\\ AND\ alloggio.citta='nome'\\ AND\ dotato.servizio='nome'\\ AND\ alloggio.tipo='nome'\\ AND\ alloggio.nposti\geq 2}} \left(\begin{matrix} (alloggio)\mathcal{F}_{AVARAGE(votorecensionealloggio),\\ COUNT(votorecensionealloggio)}(prenotazione) \\ \bowtie_{alloggio=id}\ (alloggio) \bowtie_{id=alloggio}\ (dotato) \end{matrix} \right) \right)$$

```
SELECT DISTINCT a.id, AVG(p.votorecensionealloggio) AS mediavoti,
                COUNT(p.votorecensionealloggio), a.nome, a.tipoalloggio,
                a.nposti, a.prezzoxnotte
FROM prenotazione AS p
RIGHT JOIN alloggio AS a
    ON p.alloggio=a.id
INNER JOIN dotato
    ON a.id=dotato.alloggio
WHERE a.archiviato = FALSE AND a.citta='Milano' AND
      dotato.servizio = 'nomeservizio' AND
      a.tipoalloggio = 'tipoalloggio' AND
      a.nposti >= 2
GROUP BY p.alloggio, a.nome, a.tipoalloggio, a.nposti, a.prezzoxnotte
ORDER BY mediavoti DESC
LIMIT 20;
```

5. Interfaccia grafica

L'applicazione è stata sviluppata combinando HTML, CSS, PHP e Bootstrap e pensata per 2 categorie di utilizzatori, gli utenti registrati e gli utenti non registrati.
Per ogni tipo di utilizzatore sono state realizzata delle homepage personali attraverso le quali si può accedere inserendo la email e la password.
All'avvio dell'applicazione appare una schermata generica di ricerca, successivamente, è possibile registrarsi e/o loggarsi per poter prenotare, recensire ed effettuare tutte quelle azioni per usufruire del servizio.

L'interfaccia è stata pensata per rendere l'applicazione intuitiva ed efficiente e per questo si è deciso di utilizzare una grafica semplice e minimale. Tuttavia, è possibile migliorarla aggiungendo (o modificando) altre funzionalità. Di seguito vengono riportati gli screenshot dell'applicazione.



6. Note

Strumenti utilizzati: Microsoft Word, draw.io, PostgreSQL, SublimeText3.

Possibili miglioramenti:

- Provvedere alla creazione di un'area amministratore riservato al gestore della piattaforma.
- Includere un'area dedicata alla community in cui gli utenti possono condividere impressioni e discutere dei soggiorni trascurati in determinate città utilizzando gli alloggi offerti dalla piattaforma.
- Al fine di far vivere agli utenti Ospiti una esperienza unica nelle città in cui decidono di prendere alloggio, può essere pianificato di realizzare una area della piattaforma dedicata a ciascuna città in cui gestori di teatri, guide turistiche, artisti, ecc. possono pubblicare una esperienza da far fare agli utenti Ospiti della piattaforma gratis o a pagamento. In tale modo oltre ad accrescere la popolarità della città, viene alimentata ancora di più l'economia locale.
- Una tale possibilità è offerta dalla piattaforma Airbnb tramite il concetto di *Experience*.

Difficoltà incontrate:

- La raccolta dei requisiti ha richiesto una certa quantità di tempo in quanto è stato necessario analizzare la piattaforma Airbnb (principale piattaforma nel campo del *home renting*) per capire come funziona chiaramente un tale tipo di servizio.
- La gestione della entità della prenotazione in quanto è stato necessario analizzare i diversi stati per cui può passare (come lo si può vedere nella generalizzazione della entità "Prenotazione" dello schema concettuale in Figura 1).
- Ristrutturazione dello schema concettuale in Figura 1, infatti è stato necessario realizzare (basandoci sulle ipotesi desunte dalla raccolta dei requisiti) la *Tabella dei Volumi* e alcune *Tabelle degli accessi* per effettuare una buona ristrutturazione.
- Sviluppando l'interfaccia grafica sono emerse molte operazioni in più necessarie per il funzionamento basilare di essa.