

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II  
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE  
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE  
DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA  
INSEGNAMENTO DI BASI DI DATI I  
ANNO ACCADEMICO 2019-2020

# Progettazione e sviluppo di una base di dati relazionale di un software gestionale per recensioni turistiche

*Autori:*

DAVIDE PIO FAICCHIA  
MATRICOLA : N86003018  
*da.faicchia@studenti.unina.it*

MAURO GUIDA  
MATRICOLA : N86002889  
*maur.guida@studenti.unina.it*

*Docenti:*

PROF. FRANCESCO CUTUGNO  
DOTT. MARCO GRAZIOSO

*Questa pagina è stata lasciata intenzionalmente bianca.*

# Indice

## Capitolo 1

Analisi del problema	5
----------------------	---

## Capitolo 2

Progettazione concettuale	6
2.1 Dizionario delle classi	7
2.2 Dizionario delle associazioni	9
2.3 Dizionario dei vincoli	10

## Capitolo 3

Progettazione logica	12
3.1 Schema logico	12
3.1.1 Traduzione delle associazioni	13

## Capitolo 4

Progettazione fisica	14
4.1 Note sull'implementazione	14
4.2 Definizione delle tabelle	15
4.2.1 Definizione della tabella UTENTE	15
4.2.2 Definizione della tabella MAPPA	15
4.2.3 Definizione della tabella BUSINESS	16
4.2.4 Definizione della tabella RECENSIONE	16
4.2.5 Definizione della tabella ASSOCIAZIONERAFFINAZIONE	17
4.2.6 Definizione della tabella IMMAGINEPROPRIETA	17
4.2.7 Definizione della tabella IMMAGINERECENSIONE	17
4.3 Funzioni, procedure e Trigger	18
4.3.1 Integrità delle raffinazioni per tipologia 'Ristorante'	18
4.3.2 Integrità delle raffinazioni per tipologia 'Alloggio'	18
4.3.3 Integrità delle raffinazioni per tipologia 'Attrazione'	19
4.3.4 Generazione codice verifica per Utente	20
4.3.5 Effettua login	20
4.3.6 Effettua la registrazione	20
4.3.7 Imposta nuova password	21
4.3.8 Controlla documenti utente	21
4.3.9 Controlla codice verifica	21
4.3.10 Inserisci documenti utente	22
4.3.11 Inserisci business	22
4.3.12 Inserisci immagini a business	22
4.3.13 Recupera codice business	23
4.3.14 Inserisci raffinazioni	23
4.3.15 Funzione INSTR	24
4.3.17 Inserisci immagine recensione	24
4.3.16 Inserisci recensione	25
4.3.18 Ricerca locale	25
4.3.19 Aggiorna media stelle	26

4.3.20 Utente con recensione	26
4.4 Chiamate SQL integrate nell'Applicativo	27
4.4.1 Recupera locali da ricerca	27
4.4.2 Recupera locali da tipo	27
4.4.3 Recupera business da codUtente	27

## Capitolo 1

# Analisi del problema

In questo primo capitolo verrà presentata una panoramica generale analizzato il problema al livello di astrazione più alto.

E' stata richiesta la progettazione di una base di dati relazionale per un software di recensioni turistiche, il focus del problema è stato, quindi, gestire un numero indefinito di utenti, permettendo loro l'inserimento in piattaforma di recensioni e nuove attività a scopo pubblicitario e valutativo delle stesse, su tutto il territorio nazionale (Italia).

Da come è possibile analizzare dal *class diagram* al Capitolo 2, il concetto cardine del software presentato è il concetto di **Business**, al quale è permesso a uno o più utenti di associarvi una recensione, accompagnata, o meno, da immagini e descrizione.

Tale implementazione ha richiesto l'uso di 7 Tabelle e 2 Enumerazioni, come riportato graficamente al Capitolo 2 e descritto ampiamente al paragrafo 1 del medesimo capitolo. Lo sviluppo, in oltre, non ha richiesto la ristrutturazione del *class diagram* in quanto non presenti specializzazioni o associazioni molti a molti. La completa descrizione delle tabelle e le relative associazioni può essere osservata al Capitolo 3.

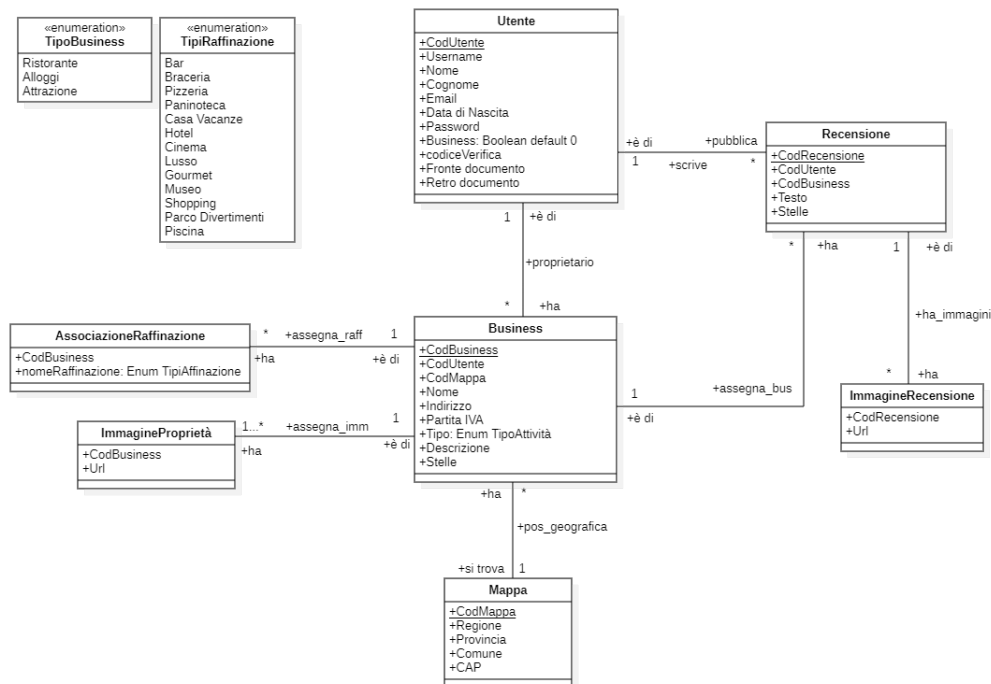
Ove possibile si è preferito evitare un'interazione diretta tra l'applicativo sviluppato ed il database. Per soddisfare tale scelta implementativa si è preferito dotare il database di funzioni specifiche (4.3), fornendo all'applicativo solo le informazioni strettamente necessarie alla corretta visualizzazione delle informazioni.

Onde evitare alterazioni *non* volontarie da parte del personale addetto alla manutenzione, con accesso diretto alla struttura dati, ogni vincolo è implementato direttamente nel database, così da impedire l'errato inserimento o modifica delle tabelle, limitando la possibilità di portare il sistema in uno stato instabile. Il *dizionario dei vincoli* può essere consultato al Capitolo 2 (2.3), dove è riportata anche una descrizione degli stessi.

Da qui in avanti saranno usati in modo intercambiabile i termini *Attività* e *Business*.

## Capitolo 2

# Progettazione concettuale



\* Durante lo sviluppo del progetto non è stato necessario ristrutturare il class diagram.

Le associazioni tra tabelle avvengono tramite l'uso delle chiavi primarie: `CodUtente`, `CodBusiness`, `CodRecensione` e `CodMappa`. Come si evince dal nome, tali chiavi indicano univocamente: Utenti, Attività, Recensioni di un determinato utente ad un determinato Business e uno specifico luogo sulla mappa.

Affinché fosse possibile categorizzare un Business, si sono dovuti creare delle tabelle enumerazione per definire dei tipi specifici per le 3 Attività da gestire (Ristorante, Alloggio, Attrazione) e le relative specializzazioni (4.3.1).

Un'ampia descrizione delle Classi utilizzate e i relativi vincoli, può essere consultata nei paragrafi immediatamente successivi.

## 2.1 Dizionario delle classi

Classe	Descrizione	Attributi
<b>Business</b>	Descrittore di ciascuna attività presente nel portale	<b>CodBusiness</b> ( <i>serial</i> ): chiave primaria. Identifica univocamente ciascuna attività. <b>Nome</b> ( <i>varchar</i> ): nome dell'attività. <b>Indirizzo</b> ( <i>varchar</i> ): indirizzo dell'attività. <b>PartitaIVA</b> ( <i>varchar</i> ): partitaIVA associata all'attività. <b>tipo</b> ( <i>tipoBusiness</i> ): tipologia dell'attività. <b>Descrizione</b> ( <i>varchar</i> ): descrizione attività. <b>Stelle</b> ( <i>numeric</i> ): valutazione corrente. <b>Telefono</b> ( <i>varchar</i> ): numero telefonico. <b>codUtente</b> ( <i>integer</i> ): chiave esterna. proprietario dell'attività. chiave esterna. <b>codMappa</b> ( <i>integer</i> ): chiave esterna. posizione dell'attività.
<b>Utente</b>	Descrittore degli utenti	<b>CodUtente</b> ( <i>serial</i> ): chiave primaria. Identifica univocamente ciascun utente. <b>Username</b> ( <i>varchar</i> ): username utilizzato dall'utente durante la registrazione. <b>Nome</b> ( <i>varchar</i> ): nome dell'utente. <b>Cognome</b> ( <i>varchar</i> ): cognome dell'utente. <b>Email</b> ( <i>varchar</i> ): email dell'utente. <b>DataDiNascita</b> ( <i>date</i> ): data nascita dell'utente. <b>Password</b> ( <i>varchar</i> ): password dell'utente. <b>CodiceVerifica</b> ( <i>varchar</i> ): codice di verifica utilizzato dall'utente durante la registrazione. <b>FronteDocumento</b> ( <i>varchar</i> ): immagine fronte documento inserita dall'utente per pubblicare una propria attività. <b>RetroDocumento</b> ( <i>varchar</i> ): immagine retro documento inserita dall'utente per pubblicare una propria attività.
<b>Recensione</b>	Descrittore delle recensioni inserite dagli utenti	<b>CodRecensione</b> ( <i>serial</i> ): chiave primaria. Identifica univocamente ciascuna recensione. <b>Nome</b> ( <i>varchar</i> ): testo della recensione. <b>Stelle</b> ( <i>numeric</i> ): valutazione locale inserita nella recensione. <b>codUtente</b> ( <i>integer</i> ): chiave esterna. identifica l'utente che ha scritto la recensione. <b>codBusiness</b> ( <i>integer</i> ): chiave esterna. identifica l'attività a cui è associata la recensione.

\*Tabella 1 : *continua a pagina successiva*

Tabella 1: *continua dalla pagina precedente.*

Classe	Descrizione	Attributi
<b>Mappa</b>	Descrittore della posizione geografica di ciascuna attività	<b>CodMappa</b> ( <i>serial</i> ): chiave primaria. Identifica univocamente ciascuna posizione geografica. <b>Stato</b> ( <i>varchar</i> ): Attributo di locazione. <b>CAP</b> ( <i>varchar</i> ): Attributo di locazione. <b>Comune</b> ( <i>varchar</i> ): Attributo di locazione. <b>Provincia</b> ( <i>varchar</i> ): Attributo di locazione. <b>Regione</b> ( <i>varchar</i> ): Attributo di locazione. <b>SiglaProvincia</b> ( <i>varchar</i> ): Attributo di locazione.
<b>Immagine Proprietà</b>	Descrittore delle immagini inserite dal proprietario associate ad una propria attività.	<b>URL</b> ( <i>varchar</i> ): URL dell'immagine. <b>codBusiness</b> ( <i>integer</i> ): chiave esterna. Identifica l'attività a cui è associata l'immagine.
<b>Associazione Raffinazione</b>	Descrittore delle raffinazioni di tipologia associate a ciascuna attività.	<b>raffinazione</b> ( <i>tipoRaffinazione</i> ): Raffinazione di una tipologia di attività. <b>codBusiness</b> ( <i>integer</i> ): chiave esterna. Identifica l'attività a cui è associata la raffinazione specificata.
<b>Immagine Recensione</b>	Descrittore delle immagini associate a ciascuna recensione.	<b>URL</b> ( <i>varchar</i> ): URL dell'immagine. <b>codRecensione</b> ( <i>integer</i> ): chiave esterna. Identifica la recensione a cui è stata associata l'immagine.

Tabella 1 - Dizionario delle classi



## 2.2 Dizionario delle associazioni

Nome	Descrizione	Classi coinvolte
<b>assegna_raff</b>	esprime l'assegnazione di una raffinazione di tipologia a un business.	<b>Business[1]</b> ruolo <b>è di</b> : indica l'attività a cui viene assegnata la raffinazione. <b>AssociazioneRaffinazione[0..*]</b> ruolo <b>ha</b> : esprime la possibilità di associare a un'attività delle raffinazioni
<b>pos_geografica</b>	esprime l'assegnazione di un'attività alla propria posizione geografica (in termini di CAP, Regione, Comune,...)	<b>Business[0..*]</b> ruolo <b>ha</b> : esprime la condizione per la quale una stessa posizione geografica può essere assegnata a più attività. <b>Mappa[1]</b> ruolo <b>si trova</b> : indica la posizione geografica associata al business.
<b>assegna_imm</b>	esprime l'assegnazione di una certa immagine a una attività.	<b>Business[1]</b> ruolo <b>è di</b> : esprime il business al quale è associata la coppia ( <i>url,codBusiness</i> ). <b>ImmagineProprietà[1..*]</b> ruolo <b>ha</b> : esprime l'immagine associata al business in termini di coppia ( <i>url,codBusiness</i> ).
<b>proprietario</b>	esprime l'assegnazione di una certa attività al suo proprietario	<b>Business[0..*]</b> ruolo <b>ha</b> : esprime la condizione per la quale un certo utente può o non avere un'attività. <b>Utente[1]</b> ruolo <b>è di</b> : definisce il proprietario dell'attività.
<b>scrive</b>	esprime l'assegnazione di una recensione all'utente che l'ha scritta.	<b>Recensione[0..*]</b> ruolo <b>pubblica</b> : esprime la possibilità per un utente di scrivere o meno delle recensioni. <b>Utente[1]</b> ruolo <b>è di</b> : definisce l'utente che ha scritto la recensione.
<b>assegna_bus</b>	esprime l'assegnazione di una certa recensione ad un'attività.	<b>Business[1]</b> ruolo <b>è di</b> : definisce il business a cui è associata la recensione <b>Recensione[0..*]</b> ruolo <b>ha</b> : definisce la possibilità di associare o meno una recensione ad un'attività.
<b>ha_immagini</b>	esprime l'assegnazione di una certa immagine ad una recensione.	<b>ImmagineRecensione[0..*]</b> ruolo <b>ha</b> : indica la possibilità per una recensione di avere o meno delle immagini associate. <b>Recensione[1]</b> ruolo <b>è di</b> : definisce la recensione ha cui viene associata una certa immagine.

Tabella 2 - Dizionario delle associazioni

## 2.3 Dizionario dei vincoli

Nome Vincolo	Descrizione
<b>legit emails</b>	Le emails inserite dagli utente devono avere un formato legittimo, ergo devono contenere almeno un carattere prima della @ , almeno un carattere tra essa e il punto e almeno due caratteri nella parte finale. vedi: <i>4.2.1 Definizione della tabella UTENTE.</i>
<b>LunghezzaPassword</b>	Le password inserite dagli utenti devono avere lunghezza maggiore o uguale di 6. vedi: <i>4.2.1 Definizione della tabella UTENTE.</i>
<b>RecensioneUnicaUtenteLuogo</b>	Un utente può pubblicare al più una recensione per attività. vedi: <i>4.2.4 Definizione della tabella RECENSIONE.</i>
<b>RaffinazioneUnica</b>	Non è possibile associare più di una volta una stessa raffinazione a una attività. vedi: <i>4.2.5 Definizione della tabella ASSOCIAZIONERAFFINAZIONE.</i>
<b>NumeroDiTelefonoNonValido</b>	I numeri di telefono dei business devono avere un formato legittimo. Devono contenere 10 cifre. vedi: <i>4.2.3 Definizione della tabella BUSINESS.</i>
<b>NumeroDiTelefono Troppo-Corto</b>	I numeri di telefono dei business devono avere un formato legittimo. Devono contenere solo cifre numeriche. vedi: <i>4.2.3 Definizione della tabella BUSINESS.</i>
<b>lunghezzaPartitaIVA</b>	La partitaIVA deve avere esattamente 10 caratteri. vedi: <i>4.2.3 Definizione della tabella BUSINESS.</i>
<b>DataDiNascita NonValida</b>	Non è possibile che l'utente sia nato nell'istante presente o nel futuro. vedi: <i>4.2.1 Definizione della tabella UTENTE.</i>
<b>noImmagineProprieta</b>	Non è possibile associare più volte una stessa immagine a un business. vedi: <i>4.2.6 Definizione della tabella IMMAGINEPROPRIETA.</i>
<b>checkRaffinazioneRistoranti</b>	vedi: <i>4.3.1 Integrità delle raffinazioni per tipologia 'Ristorante'.</i>
<b>checkRaffinazioneAlloggio</b>	vedi: <i>4.3.2 Integrità delle raffinazioni per tipologia 'Alloggio'.</i>
<b>checkRaffinazioneAttrazioni</b>	vedi: <i>4.3.3 Integrità delle raffinazioni per tipologia 'Attrazione'.</i>
<b>legit Stelle Business</b>	Un business può avere (Stelle IS NULL) o meno una valutazione((Stelle BETWEEN 1 AND 5). vedi: <i>4.2.3 Definizione della tabella BUSINESS.</i>

\*Tabella 3 - continua a pagina successiva

Tabella 3: *continua dalla pagina precedente.*

Nome Vincolo	Descrizione
<b>legit Stelle Recensione</b>	Una recensione <i>deve</i> avere una valutazione((Stelle BETWEEN 1 AND 5). vedi: <i>4.2.4 Definizione della tabella RECENSIONE</i> .
<b>unique Username</b>	Non è possibile che più utenti abbiano lo stesso Username. vedi: <i>4.2.1 Definizione della tabella UTENTE</i> .
<b>unique PartitaIVA</b>	Non è possibile che più business abbiano lo stesso Username. vedi: <i>4.2.3 Definizione della tabella BUSINESS</i> .
<b>domain tipoRaffinazione</b>	Vincolo di dominio attributo <b>tipo</b> nella tabella BUSINESS. vedi: <i>4.2.3 Definizione della tabella BUSINESS</i> .
<b>domain tipoRaffinazione</b>	Vincolo di dominio attributo <b>raffinazione</b> nella tabella ASSOCIAZIONERAFFINAZIONE. vedi: <i>4.2.3 Definizione della tabella ASSOCIAZIONERAFFINAZIONE</i> .
<b>domain tipoBusiness</b>	Vincolo di dominio attributo <b>tipo</b> nella tabella BUSINESS. vedi: <i>4.2.3 Definizione della tabella BUSINESS</i> .

Tabella 3 - Dizionario dei vincoli

## Capitolo 3

# Progettazione logica

In questo capitolo saranno presentati gli schemi logico e relazionali.

### 3.1 Schema logico

<b>UTENTE</b>	( <b><u>codUtente</u></b> , Username, Nome, Cognome, Email, Password, DataDiNascita, codiceVerifica, FronteDocumento, RetroDocumento)
<b>BUSINESS</b>	( <b><u>codBusiness</u></b> , Nome, Indirizzo, PartitaIVA, tipo, Descrizione, Stelle, Telefono, <u>codUtente</u> , <u>codMappa</u> )
<b>MAPPA</b>	( <b><u>codMappa</u></b> , Stato, CAP, Regione, Comune, Provincia, SiglaProvincia)
<b>RECENSIONE</b>	( <b><u>codRecensione</u></b> , Testo, Stelle, <u>codUtente</u> , <u>codBusiness</u> )
<b>IMMAGINE PROPRIETA</b>	(URL, <u>codBusiness</u> )
<b>ASSOCIAZIONE RAFFINAZIONE</b>	(raffinazione, <u>codBusiness</u> )
<b>IMMAGINE RECENSIONE</b>	(URL, <u>codRecensione</u> )

Tabella 4 - Schema logico

### 3.1.1 Traduzione delle associazioni

Associazione	Implementazione
<b>assegna_raff</b>	Chiave esterna in ASSOCIAZIONERAFFINAZIONE → BUSINESS
<b>pos_geografica</b>	Chiave esterna in BUSINESS → MAPPA
<b>assegna_imm</b>	Chiave esterna in IMMAGINEPROPRIETA → BUSINESS
<b>propretario</b>	Chiave esterna in BUSINESS → UTENTE
<b>scrive</b>	Chiave esterna in RECENSIONE → UTENTE
<b>assegna_bus</b>	Chiave esterna in RECENSIONE → BUSINESS
<b>ha_immagini</b>	Chiave esterna in IMMAGINERECENSIONE → RECENSIONE

Tabella 5 - Traduzione delle associazioni

## Capitolo 4

# Progettazione fisica

La base di dati verrà implementata sul DBMS PostgreSQL 11.7

### 4.1 Note sull'implementazione

Al fine di agevolare lo sviluppo delle funzioni necessarie, ove possibile, saranno utilizzate funzioni e procedure caratteristiche dello standard del DBMS in questione. Tuttavia, alcune funzionalità come `INSTR` appartenente ad `ORACLE`, non disponibile su `PostgreSQL`, saranno implementate manualmente come funzioni del seguente Database, discorso equivalente per l'implementazione di `ASSERTION`, anch'essa rimpiazzata da una specifica procedure poiché non disponibili.

Nel seguente capitolo saranno mostrate le funzionalità implementate lato Database, quali: Vincoli su Tabelle, Procedure, Funzioni e Trigger, oltre ad evidenziare come ogni Tabella sia stata definita.

Nel paragrafo 4 saranno approfondite le funzioni implementate direttamente dell'applicativo e pertanto non presenti nel Database, tuttavia questa pratica, ove possibile, sia stata scongiurata per semplificare future manutenzioni ed implementazioni, rendendo possibile effettuare aggiornamenti esclusivamente lato Database senza rendere obsolete vecchie release degli applicativi.

L'obiettivo principale è rendere ogni richiesta dell'utente una chiamata ad una funzione presente nella Base di Dati, limitando possibili manomissioni e garantendo una maggiore stabilità del sistema, oltre ad un supporto a lungo termine degli applicativi.

## 4.2 Definizione delle tabelle

Seguono le definizioni delle tabelle utilizzate nella base di dati, con le relative descrizioni dei vincoli adottati, per gli attributi, di ognuna di esse.

*\* Alcune parole risulteranno essere grammaticalmente errate data l'incompatibilità dei caratteri speciali col DBMS.*

### 4.2.1 Definizione della tabella UTENTE

```
1 /**
2  *   TABELLA: UTENTE
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5 CREATE TABLE Utente(
6     codUtente SERIAL PRIMARY KEY,
7     Username VARCHAR(50) NOT NULL UNIQUE,
8     Nome VARCHAR(50) NOT NULL,
9     Cognome VARCHAR(50) NOT NULL,
10    Email VARCHAR(100) NOT NULL UNIQUE CHECK(Email LIKE '_%@%._%'),
11    DataDiNascita date NOT NULL,
12    Password VARCHAR(100) NOT NULL,
13    codiceVerifica VARCHAR(10) DEFAULT NULL,
14    FronteDocumento VARCHAR(1000) DEFAULT NULL,
15    RetroDocumento VARCHAR(1000) DEFAULT NULL
16 );
17
18 -- Vincolo di lunghezza della password
19 ALTER TABLE Utente
20     ADD CONSTRAINT LunghezzaPassword CHECK(length>Password)>=6);
21
22 -- Vincolo di integrita della DataDiNascita inserita dall'utente
23 ALTER TABLE Utente
24     ADD CONSTRAINT DataDiNascitaNonValida
25     CHECK(NOT(DataDiNascita >= date('now')));
```

### 4.2.2 Definizione della tabella MAPPA

```
1 /**
2  *   TABELLA: MAPPA
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5 CREATE TABLE Mappa(
6     codMappa SERIAL PRIMARY KEY
7     Stato VARCHAR(5),
8     CAP VARCHAR(7),
9     Comune VARCHAR(100),
10    Regione VARCHAR(100),
11    Provincia VARCHAR(100),
12    SiglaProvincia VARCHAR(5),
13 );
```

### 4.2.3 Definizione della tabella BUSINESS

```
1 /**
2  *   TABELLA: BUSINESS
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5
6 -- Vincolo di dominio per l'attributo 'tipo'
7 CREATE TYPE tipoBusiness AS ENUM ('Attrazione', 'Alloggio', 'Ristorante');
8
9 CREATE TABLE Business(
10     codBusiness SERIAL PRIMARY KEY,
11     Nome VARCHAR(50) NOT NULL,
12     Indirizzo VARCHAR(100) NOT NULL,
13     PartitaIVA VARCHAR(100) NOT NULL UNIQUE,
14     tipo tipoBusiness NOT NULL,
15     Descrizione VARCHAR(2000) NOT NULL,
16     Stelle NUMERIC DEFAULT NULL CHECK((Stelle BETWEEN 1 AND 5)
17                                     OR Stelle IS NULL),
18     Telefono VARCHAR(10) NOT NULL,
19     codUtente INTEGER REFERENCES Utente(codUtente) ON DELETE CASCADE,
20     codMappa INTEGER REFERENCES Mappa(codMappa)
21 );
22
23 -- Vincolo di correttezza del numero telefonico (deve contenere solo cifre)
24 ALTER TABLE Business
25     ADD CONSTRAINT NumeroDiTelefonoNonValido CHECK(Telefono ~ '^[0-9 ]*$');
26
27 -- Vincolo di correttezza del numero telefonico (deve contenere esattamente
28     10 cifre)
29 ALTER TABLE Business
30     ADD CONSTRAINT NumeroDiTelefonoTroppoCorto CHECK(LENGTH(Telefono) = 10);
31
32 -- Vincolo di correttezza sulla lunghezza della partitaIVA inserita
33 ALTER TABLE Business
34     ADD CONSTRAINT lunghezzaPartitaIVA CHECK(LENGTH(PartitaIVA) = 11);
```

### 4.2.4 Definizione della tabella RECENSIONE

```
1 /**
2  *   TABELLA: RECENSIONE
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5 CREATE TABLE Recensione(
6     CodRecensione SERIAL PRIMARY KEY,
7     Testo VARCHAR(2000) NOT NULL,
8     Stelle NUMERIC NOT NULL CHECK(Stelle BETWEEN 1 AND 5),
9     CodBusiness INTEGER REFERENCES Business(codBusiness) ON DELETE CASCADE,
10    CodUtente INTEGER REFERENCES Utente(codUtente) ON DELETE CASCADE
11 );
12
13 -- Non sara possibile per un utente recensire piu volte lo stesso business
14 ALTER TABLE Recensione
15     ADD CONSTRAINT RecensioneUnicaUtenteLuogo UNIQUE(codBusiness, codUtente);
```



#### 4.2.5 Definizione della tabella ASSOCIAZIONERAFFINAZIONE

```
1 /**
2  *   TABELLA: ASSOCIAZIONERAFFINAZIONE
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5
6 -- Vincolo di dominio per l'attributo 'raffinazione'
7 CREATE TYPE tipoRaffinazione AS
8 ENUM ('Pizzeria', 'Braceria', 'FastFood',
9 'Paninoteca', 'Osteria', 'TavolaCalda',
10 'Taverna', 'Trattoria', 'Pesce',
11 'Cinema', 'Shopping', 'Monumento',
12 'Museo', 'ParcoGiochi', 'Piscina',
13 'Lounge', 'Hotel', 'Bed&Breakfast',
14 'Ostello', 'CasaVacanze', 'Residence');
15
16 CREATE TABLE AssociazioneRaffinazione(
17     codBusiness INTEGER REFERENCES Business(codBusiness) ON DELETE CASCADE,
18     raffinazione tipoRaffinazione
19 );
20
21 -- Non sara possibile associare piu di una volta una stessa raffinazione a
    un business
22 ALTER TABLE AssociazioneRaffinazione
23     ADD CONSTRAINT RaffinazioneUnica UNIQUE(codBusiness,raffinazione);
24 );
```

#### 4.2.6 Definizione della tabella IMMAGINEPROPRIETA

```
1 /**
2  *   TABELLA: IMMAGINEPROPRIETA
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5 CREATE TABLE ImmagineProprieta(
6     Url VARCHAR(1000),
7     codBusiness INTEGER REFERENCES Business(codBusiness) ON DELETE CASCADE
8 );
9
10 -- Non sara possibile associare piu di una volta una stessa immagine a un
    business
11 ALTER TABLE ImmagineProprieta
12     ADD CONSTRAINT noImmagineProprieta UNIQUE(Url,CodBusiness);
```

#### 4.2.7 Definizione della tabella IMMAGINERECENSIONE

```
1 /**
2  *   TABELLA: IMMAGINERECENSIONE
3  *   Crea la tabella e implementa i vincoli piu semplici
4  */
5 CREATE TABLE ImmagineRecensione(
6     Url VARCHAR(1000) NOT NULL,
7     codRecensione INTEGER REFERENCES Recensione(CodRecensione) ON DELETE
        CASCADE
8 );
```

### 4.3 Funzioni, procedure e Trigger

In questo paragrafo verrà fornita una descrizione accurata dei metodi di automazione adottati, quali trigger e delle funzioni e procedure richiamate dall'applicativo con lo scopo di ricavare le informazioni necessarie al suo funzionamento.

#### 4.3.1 Integrità delle raffinazioni per tipologia 'Ristorante'

Se l'utente seleziona come tipologia 'Ristorante', allora le raffinazioni devono essere necessariamente pescate tra le seguenti :

('Pizzeria', 'Braceria', 'FastFood', 'Paninoteca', 'Osteria', 'Tavola', 'Taverna', 'Trattoria', 'Pesce')

```
1 CREATE FUNCTION checkRaffinazioneRistoranti() RETURNS TRIGGER AS
2 $BODY$
3 DECLARE
4     raff AssociazioneRaffinazione.raffinazione%TYPE;
5 BEGIN
6     FOR raff IN SELECT raffinazione FROM AssociazioneRaffinazione WHERE
7         codBusiness = NEW.codBusiness
8     LOOP
9         IF raff.raffinazione NOT IN
10            ('Pizzeria', 'Braceria', 'FastFood',
11             'Paninoteca', 'Osteria', 'Tavola Calda',
12             'Taverna', 'Trattoria', 'Pesce') THEN
13             RAISE EXCEPTION 'Errore: Raffinazione non consentita';
14         END IF;
15     END LOOP;
16 END;
17 $BODY$
18 LANGUAGE PLPGSQL;
19
20 CREATE TRIGGER checkRaffinazioneRistoranti
21 BEFORE INSERT ON Business
22 FOR EACH ROW
23 WHEN (NEW.tipo = 'Ristorante')
24 EXECUTE PROCEDURE checkRaffinazioneRistoranti();
```

#### 4.3.2 Integrità delle raffinazioni per tipologia 'Alloggio'

Se l'utente seleziona come tipologia 'Alloggio', allora le raffinazioni devono essere necessariamente pescate tra le seguenti :

('Hotel', 'Bed&Breakfast', 'Ostello', 'CasaVacanze', 'Residence')

```
1 CREATE FUNCTION checkRaffinazioneAlloggio() RETURNS TRIGGER AS
2 $BODY$
3 DECLARE
4     raff AssociazioneRaffinazione.raffinazione%TYPE;
5 BEGIN
6     FOR raff IN SELECT raffinazione FROM AssociazioneRaffinazione WHERE
7         codBusiness = NEW.codBusiness
8     LOOP
9         IF raff.raffinazione NOT IN
10            ('Hotel', 'Bed&Breakfast', 'Ostello', 'CasaVacanze', 'Residence') THEN
11             RAISE EXCEPTION 'Errore: Raffinazione non consentita';
12         END IF;
13     END LOOP;
14 END;
```

```

14 $BODY$
15 LANGUAGE PLPGSQL;
16
17
18 CREATE TRIGGER checkRaffinazioneAlloggio
19 BEFORE INSERT ON Business
20 FOR EACH ROW
21 WHEN (NEW.tipo = 'Alloggio')
22 EXECUTE PROCEDURE checkRaffinazioneAlloggio();

```

#### 4.3.3 Integrità delle raffinazioni per tipologia 'Attrazione'

Se l'utente seleziona come tipologia 'Attrazione', allora le raffinazioni devono essere necessariamente pescate tra le seguenti :

('Cinema', 'Shopping', 'Monumento', 'Museo', 'Parco Giochi', 'Piscina', 'Bar/Lounge')

```

1 CREATE FUNCTION checkRaffinazioneAttrazioni() RETURNS TRIGGER AS
2 $BODY$
3 DECLARE
4     raff AssociazioneRaffinazione.raffinazione%TYPE;
5 BEGIN
6     FOR raff IN SELECT raffinazione
7                 FROM AssociazioneRaffinazione
8                 WHERE codBusiness = NEW.codBusiness LOOP
9         IF raff.raffinazione NOT IN
10            ('Cinema', 'Shopping', 'Monumento', 'Museo',
11             'Parco Giochi', 'Piscina', 'Bar/Lounge') THEN
12             RAISE EXCEPTION 'Errore: Raffinazione non consentita';
13         END IF;
14     END LOOP;
15 END;
16 $BODY$
17 LANGUAGE PLPGSQL;
18
19
20 CREATE TRIGGER checkRaffinazioneAttrazioni
21 BEFORE INSERT ON Business
22 FOR EACH ROW
23 WHEN (NEW.tipo = 'Attrazione')
24 EXECUTE PROCEDURE checkRaffinazioneAttrazioni();

```

#### 4.3.4 Generazione codice verifica per Utente

Tale procedura genera un codice di 10 cifre e lo inserisce nel record della tabella UTENTE grazie al codUtente passato in input.

```
1 CREATE OR REPLACE PROCEDURE generaCodiceVerifica(INT)
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     Update utente
6     SET codiceVerifica=(SELECT substr(md5(random()::text), 0, 10))
7     WHERE codUtente=$1;
8
9     COMMIT;
10 END;
11 $$;
```

#### 4.3.5 Effettua login

Tale funzione effettua l'accesso al portare tramite le credenziali d'accesso fornite dall'utente. (Attraverso il suo codUtente)

```
1 CREATE OR REPLACE FUNCTION login(INusername VARCHAR(50), INpassword VARCHAR
  (100))
2 RETURNS INTEGER AS $$
3 DECLARE codUtenteDaRestituire INTEGER;
4 BEGIN
5     SELECT codUtente INTO codUtenteDaRestituire
6     FROM utente
7     WHERE username = $1 AND password = $2;
8
9     RETURN codUtenteDaRestituire;
10 END;
11 $$ LANGUAGE plpgsql;
```

#### 4.3.6 Effettua la registrazione

Tale procedura aggiorna il database inserendo i parametri dati in input dall'utente in modo tale che egli possa effettuare il login in un secondo momento.

```
1 CREATE OR REPLACE PROCEDURE registrati(INusername VARCHAR(50), INnome
2 VARCHAR(50), INcognome VARCHAR(50), INemail VARCHAR(100), INdata DATE,
3 INpassword VARCHAR(100))
4 LANGUAGE plpgsql
5 AS $$
6 BEGIN
7     INSERT INTO Utente(Username, Nome, Cognome, Email,
8                          DataDiNascita, Password)
9     Values($1,$2,$3,$4,$5,$6);
10
11     COMMIT;
12 END;
13 $$;
```

### 4.3.7 Imposta nuova password

Tale procedura permette all'utente di aggiornare la password.

```
1 CREATE OR REPLACE PROCEDURE impostaNuovaPassword(INT, VARCHAR(50))
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     UPDATE utente
6     SET password = $2
7     WHERE codutente=$1;
8
9     COMMIT;
10 END;
11 $$;
```

### 4.3.8 Controlla documenti utente

Questa funzione controlla che l'utente abbia verificato la propria identità avendo fornito un documento. (FronteDocumento & RetroDocumento)

```
1 CREATE OR REPLACE FUNCTION controllaDocumentiUtente(INT)
2 RETURNS BOOLEAN AS $$
3 DECLARE flag BOOLEAN = '1';
4 BEGIN
5     IF EXISTS (SELECT 1
6                FROM Utente U
7                WHERE U.codUtente = $1 AND U.FronteDocumento IS NULL
8                      AND U.RetroDocumento IS NULL) THEN
9         flag = '0';
10    END IF;
11    RETURN flag;
12 END;
13 $$ LANGUAGE plpgsql;
14 $$;
```

### 4.3.9 Controlla codice verifica

Questa funzione controlla che l'utente sia verificato, ovvero abbia inserito il 'codiceVerifica' inviatogli per email.

```
1 CREATE OR REPLACE FUNCTION controllaCodiceVerifica(INT, VARCHAR(10))
2 RETURNS BOOLEAN AS $$
3 DECLARE flag BOOLEAN = '0';
4 BEGIN
5     IF EXISTS (SELECT 1
6                FROM Utente U
7                WHERE codUtente = $1 AND U.codiceVerifica = $2) THEN
8         flag = '1';
9         UPDATE Utente
10        SET codiceVerifica = NULL
11        WHERE codUtente = $1;
12    END IF;
13    RETURN flag;
14 END;
15 $$ LANGUAGE plpgsql;
```

#### 4.3.10 Inserisci documenti utente

Tale procedura aggiorna il database inserendo le foto del documento date in input dall'utente.

```
1 CREATE OR REPLACE PROCEDURE inserisciDocumentiUtente(INT, VARCHAR(1000),
  VARCHAR(1000))
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     UPDATE Utente
6     SET FronteDocumento = $2, RetroDocumento = $3
7     WHERE codUtente = $1;
8
9     COMMIT;
10 END;
11 $$;
```

#### 4.3.11 Inserisci business

Tale procedura inserisce o aggiorna un business. Se la PartitaIVA risulta già presente nel database e l'utente che tenta di reinserirla, è il proprietario del business, allora questo verrà aggiornato, altrimenti verrà inserito ex novo.

```
1 CREATE OR REPLACE PROCEDURE inserisciBusiness(VARCHAR(50), VARCHAR(100),
  VARCHAR(10), VARCHAR(100), VARCHAR(100), VARCHAR(2000), INTEGER, INTEGER)
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     IF EXISTS ( SELECT 1 FROM Business WHERE PartitaIVA = $4 ) THEN
6         UPDATE BUSINESS
7         SET Nome = $1, Indirizzo = $2, Telefono = $3,
8             tipo = ($5)::tipoBusiness, Descrizione = $6
9         WHERE PartitaIVA = $4;
10    ELSE
11        INSERT INTO Business (Nome, Indirizzo, Telefono, PartitaIVA, tipo,
12                               Descrizione, codUtente, codMappa)
13        VALUES ( $1, $2, $3, $4, ($5)::tipoBusiness , $6, $7, $8);
14    END IF;
15    COMMIT;
16 END;
17 $$;
```

#### 4.3.12 Inserisci immagini a business

Tale procedura associa una immagine fornite dal proprietario al suo business.

```
1 CREATE OR REPLACE PROCEDURE inserisciImmaginiABusiness(INTEGER, VARCHAR
  (1000))
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     INSERT INTO ImmagineProprieta(codBusiness, Url)
6     VALUES($1, $2);
7 END;
8 $$;
```

### 4.3.13 Recupera codBusiness

Tale funzione recupera il codice del business attraverso la PartitaIVA.

```
1 CREATE OR REPLACE FUNCTION recuperaCodBusiness(VARCHAR(100))
2 RETURNS INTEGER
3 AS $$
4 DECLARE codiceBusiness INTEGER;
5 BEGIN
6     SELECT codBusiness INTO codiceBusiness
7     FROM Business
8     WHERE PartitaIVA = $1;
9     RETURN codiceBusiness;
10 END;
11 $$ LANGUAGE plpgsql;
```

### 4.3.14 Inserisci raffinazioni

Questa procedura, dati in input un codBusiness ed una stringa di raffinazioni separate da virgole, scompone quest'ultima estraendo le singole raffinazioni così da poterle inserire singolarmente nella tabella ASSOCIAZIONERAFFINAZIONE.

```
1 CREATE OR REPLACE PROCEDURE inserisciRaffinazioni(INTEGER, VARCHAR(3000))
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE lunghezza INTEGER;
5 DECLARE stringa VARCHAR(3000);
6 DECLARE pos1 INTEGER;
7 DECLARE pos2 INTEGER;
8 DECLARE raffinazione VARCHAR(200);
9 DECLARE occorrenza INTEGER = 1;
10 BEGIN
11     DELETE
12     FROM AssociazioneRaffinazione
13     WHERE codBusiness = $1;
14     lunghezza = LENGTH($2);
15     stringa = $2;
16     pos1 = 1;
17     LOOP
18         pos2 = INSTR(stringa, ',', 1, occorrenza);
19         EXIT WHEN pos2 = 0;
20         raffinazione = SUBSTRING(stringa, pos1, pos2-pos1);
21         INSERT INTO AssociazioneRaffinazione
22         VALUES ($1, raffinazione::tipoRaffinazione);
23         EXIT WHEN pos2 = lunghezza;
24         occorrenza = occorrenza+1;
25         pos1 = pos2+1;
26     END LOOP;
27 END;
28 $$;
```

### 4.3.15 Funzione INSTR

Implementazione funzione INSTR per PostgreSQL.

```
1 create or replace function instr(str text, sub text, startpos int,
   occurrence int)
2 returns int language plpgsql
3 as $$
4 declare
5     tail text;
6     shift int;
7     pos int;
8     i int;
9 begin
10    shift:= 0;
11    if startpos = 0 or occurrence <= 0 then
12        return 0;
13    end if;
14    if startpos < 0 then
15        str:= reverse(str);
16        sub:= reverse(sub);
17        pos:= -startpos;
18    else
19        pos:= startpos;
20    end if;
21    for i in 1..occurrence loop
22        shift:= shift+ pos;
23        tail:= substr(str, shift);
24        pos:= strpos(tail, sub);
25        if pos = 0 then
26            return 0;
27        end if;
28    end loop;
29    if startpos > 0 then
30        return pos+ shift- 1;
31    else
32        return length(str)- length(sub)- pos- shift+ 3;
33    end if;
34 end $$;
```

### 4.3.16 Inserisci immagine recensione

Tale procedura associa un' immagine fornita dall' utente a una sua recensione.

```
1 CREATE OR REPLACE PROCEDURE inserisciImmagineRecensione(VARCHAR(1000),
   INTEGER)
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     INSERT INTO ImmagineRecensione
6     VALUES ( $1, $2);
7     COMMIT;
8 END;
9 $$;
```



### 4.3.17 Inserisci recensione

Tale funzione inserisce una recensione nel database e restituisce il codice della stessa così da poterle associare delle immagini successivamente.

```
1 CREATE OR REPLACE FUNCTION inserisciRecensione(VARCHAR(2000), NUMERIC,  
2         INTEGER, INTEGER)  
3 RETURNS INTEGER  
4 AS $$  
5 DECLARE codRecen INTEGER;  
6 BEGIN  
7     INSERT INTO Recensione (Testo, Stelle, CodBusiness, CodUtente)  
8     VALUES ( $1, $2, $3, $4) RETURNING codRecensione INTO codRecen;  
9     RETURN codRecen;  
10 END;  
11 $$ LANGUAGE plpgsql;
```

### 4.3.18 Ricerca locale

Tale funzione restituisce tutte le attività trovate in base alla tipologia e/o luogo passati in input dall'utente.

```
1 CREATE OR REPLACE FUNCTION ricercaLocale(VARCHAR(20), VARCHAR(20))  
2 RETURNS SETOF record  
3 AS $$  
4 BEGIN  
5     IF ( $1 = '' AND $2 = '' ) THEN  
6         RETURN QUERY  
7         SELECT B.codBusiness, B.Nome,  
8             B.Indirizzo, B.Stelle, I.Url  
9         FROM Business B, ImmagineProprieta I  
10        WHERE B.codBusiness = I.codBusiness AND  
11            I.URL = (SELECT IP.URL  
12                    FROM ImmagineProprieta IP  
13                    WHERE IP.codBusiness = B.codBusiness LIMIT 1);  
14    ELSE  
15        RETURN QUERY  
16        SELECT B.codBusiness, B.Nome,  
17            B.Indirizzo, B.Stelle, I.Url  
18        FROM ((Business B JOIN ImmagineProprieta I  
19              ON (B.codBusiness = I.codBusiness))  
20             JOIN Mappa M ON (B.codMappa = M.codMappa))  
21        WHERE I.URL = (SELECT IP.URL  
22                      FROM ImmagineProprieta IP  
23                      WHERE IP.codBusiness = B.codBusiness LIMIT 1)  
24              AND ((B.Nome ILIKE '%' || $1 || '%')  
25                 OR (SELECT 1  
26                     FROM AssociazioneRaffinazione A  
27                     WHERE A.codBusiness = B.codBusiness AND  
28                     CAST(A.raffinazione AS VARCHAR(100)) ILIKE  
29                     '%' || $1 || '%') IS NOT NULL) AND  
30              ((M.Provincia ILIKE '%' || $2 || '%') OR  
31              (M.Comune ILIKE '%' || $2 || '%'));  
32    END IF;  
33 END;  
34 $$ LANGUAGE plpgsql;
```

#### 4.3.19 Aggiorna media stelle

Tale funzione aggiorna la valutazione del locale in seguito all'aggiunta di una nuova recensione, per mezzo del trigger 'calcolaNuovaMediaDopoInserimentoRecensione'.

```
1 CREATE OR REPLACE FUNCTION aggiornaMediaStelle()
2 RETURNS TRIGGER
3 AS $$
4 DECLARE nuovaMedia NUMERIC;
5 BEGIN
6     SELECT AVG(Stelle) INTO nuovaMedia
7     FROM Recensione
8     WHERE codBusiness = NEW.codBusiness;
9
10    UPDATE Business
11    SET Stelle = nuovaMedia
12    WHERE codBusiness = NEW.codBusiness;
13
14    RETURN NEW;
15 END;
16 $$
17 LANGUAGE 'plpgsql';
18
19 CREATE TRIGGER calcolaNuovaMediaDopoInserimentoRecensione
20 AFTER INSERT ON Recensione
21 FOR EACH ROW
22 EXECUTE PROCEDURE aggiornaMediaStelle();
```

#### 4.3.20 Utente con recensione

Tale funzione controlla se un utente ha già scritto una recensione per un certo business.

```
1 CREATE OR REPLACE FUNCTION utenteConRecensione(INT, INT)
2 RETURNS BOOLEAN AS $$
3 DECLARE flag BOOLEAN = '0';
4 BEGIN
5     IF EXISTS (SELECT 1
6                FROM Recensione
7                WHERE codUtente = $1 AND codBusiness = $2) THEN
8         flag = '1';
9     END IF;
10    RETURN flag;
11 END;
12 $$ LANGUAGE plpgsql;
```

## 4.4 Chiamate SQL integrate nell'Applicativo

In questo paragrafo sono riportate e descritte alcune interrogazioni al Database eseguite dall'applicativo sviluppato in Java.

*\* il ? è un placeholder per l'input dell'utente che verrà sostituito da una specifica funzione Java.*

### 4.4.1 Recupera locali da ricerca

L'utente, grazie a questa SELECT recupera tutti i locali con un opportuna scelta di luogo e raffinazione o nome locale.

```
1 SELECT codBusiness, Nome, Indirizzo, Stelle, URL
2 FROM ricercaLocale( ? , ? )
3 AS Locali(codBusiness INTEGER ,Nome VARCHAR(50),Indirizzo VARCHAR(100),
4           Stelle NUMERIC ,URL VARCHAR(1000))
```

### 4.4.2 Recupera locali da tipo

Questa SELECT recupera le informazioni di un locale attraverso una specifica richiesta di tipo tra Attrazione, Ristorante o Alloggio.

```
1 SELECT B.codBusiness, B.Nome, B.Indirizzo, B.Stelle, I.Url
2 FROM Business B, ImmagineProprieta I
3 WHERE B.codBusiness = I.codBusiness AND
4        I.URL = (SELECT URL
5                  FROM ImmagineProprieta IP
6                  WHERE IP.codBusiness = B.codBusiness LIMIT 1)
7        AND Tipo = ?::tipoBusiness
```

### 4.4.3 Recupera business da codUtente

Questa SELECT recupera tutte le informazioni dei business di un determinato utente.

```
1 SELECT B.codBusiness, B.Nome, B.Indirizzo, B.Stelle, I.Url
2 FROM Business B, ImmagineProprieta I
3 WHERE B.codBusiness = I.codBusiness AND
4        I.URL = (SELECT URL
5                  FROM ImmagineProprieta IP
6                  WHERE IP.codBusiness = B.codBusiness LIMIT 1)
7        AND codUtente = ?
```