Base di dati per la gestione dei parcheggi della città di Bari

Progetto per “Basi di Dati II”. A.A 2016/17

Realizzato da: Federico Maiorano

Matricola: 666337

Sommario

Progettazione Concettuale

Traccia

	Parcheggi
1.	Si vuole progettare una base di dati per la gestione dei parcheggi della città di Bari. Di ogni parcheggio,
2.	vogliono memorizzare: nome (univoco), via e posizione (latitudine/longitudine). Sono previste due categorie
3.	di parcheggi: parcheggi liberi (PL) e parcheggi a pagamento coperti (PPC). Nel primo caso (PL), si vuole tenere
4.	traccia della durata massima della sosta espressa in ore nel range [0-24]. Nel secondo caso (PPC), si
5.	vogliono memorizzare: numero di telefono, recapito email, nome della società gestore. Ogni PCC dispone di
6.	un certo numero di posti auto. Ogni posto auto dispone di un numero progressivo (univoco per un dato
7.	parcheggio), una larghezza, una lunghezza, e può disporre di un sensore associato per il rilevamento automatico
8.	dello stato (libero/occupato). Nel caso sia presente, si vuole tenere traccia dell'ID del sensore, del modello e
9.	del nome dell'azienda produttrice. Inoltre, per i PCC, si vogliono gestire le soste operate dagli utenti. Ogni
10.	sosta dispone di un codice univoco globale, una data di inizio, una data di fine, un costo, ed è associata ad
11.	un'auto e ad un utente. Il costo della sosta si ottiene moltiplicando la durata del parcheggio in ore per un
12.	costo unitario orario pari a 0.5 euro. Per quanto riguarda gli utenti, la piattaforma deve gestire: codice fiscale,
13.	nome, cognome, anno di nascita, numero di patente. Ogni PPC può vendere dei PASS per l'accesso alle zone a
14.	traffico limitato della città di Bari. Ogni PASS dispone di: codice univoco globale, data di rilascio, durata, auto e
15.	nome della zona di validità. Si vuole tenere traccia dello storico degli acquisti di PASS operati da ciascun utente.
16.	Inoltre, sono previste due categorie di utenti speciali: utenti premium (UP), ed utenti abbonati (UA). Dei
17.	primi (UP), si vogliono memorizzare anche nickname e password per l'accesso Web al portale.

Analisi dei requisiti

- *Scelta del giusto livello di astrazione:*

  - per "posizione" (rigo 2) si intendono le coordinate geografiche del parcheggio, ovvero la sua latitudine e longitudine.

- *Individuazione di omonimi e sinonimi:*

  - la sigla PL (rigo 3) sta per "parcheggio libero"
  - la sigla PCC (righe 3,4,5,9,13) sta per "parcheggio a pagamento a coperto"
  - la sigla UP (righe 16,17) sta per "utente premium"
  - la sigla UA (rigo 16) sta per "utente abbonato"

- *Resa esplicita dei riferimenti tra i termini:*

  - i termini "numero di telefono" e "recapito email" (rigo 5) si riferiscono al parcheggio, e non alla società gestore. Infatti una società può gestire più di un parcheggio nella città, tuttavia ogni parcheggio avrà un numero di telefono e una mail associata.
  - i termini "parcheggio" al rigo 1 e al rigo 11, hanno un significato differente: per "parcheggio" al rigo 1 si intende lo spazio destinato alla sosta dei veicoli, mentre per "parcheggio" al rigo 11 si intende la sosta fisica del veicolo. Quindi il termine "parcheggio" al rigo 11 verrà sostituito con il più adeguato termine "sosta".

- *Glossario dei termini*

Termine	Descrizione	Collegamento
Parcheggio	Area della città dove le auto posso sostare	Parcheggio libero, Parcheggio a pagamento coperto
Parcheggio libero	Area di sosta, dove la sosta può durare al massimo 24 ore	Parcheggio
Parcheggio a pagamento coperto	Area di sosta a pagamento coperta. Il guidatore che vi ci sosta, dovrà fornire alcuni dati ai responsabili del parcheggio	Parcheggio, Posto auto, Pass, Sosta
Posto auto	Parte di un parcheggio a pagamento coperto	Parcheggio a pagamento coperto
Sensore	Dispositivo per il rilevamento automatico dello stato (libero/occupato) di un posto auto	Posto auto
Sosta	Periodo nel quale l'auto di un utente è ferma in un'area di parcheggio a pagamento coperto	Parcheggio a pagamento coperto, Utente
Utente	Individuo che ha effettuato almeno una sosta in parcheggio a pagamento coperto	Sosta, Pass, Utente premium, Utente abbonato
Pass	Speciali permessi per l'accesso a zone a traffico limitato, venduti da parcheggi a pagamento coperti ad utenti	Parcheggio a pagamento coperto, Utente
Utente premium	Utente speciale che può accedere al portale web del sistema attraverso username e password	Utente
Utente abbonato	Utente che ha acquistato un abbonamento ad un parcheggio	Utente

- *Specifiche riorganizzate per concetti*:

Frasi di carattere generale: *"Si vuole progettare una base di dati per la gestione dei parcheggi della città di Bari."*

Frasi relative ai parcheggi: *"Di ogni parcheggio, si vogliono memorizzare: nome (univoco), via e posizione (latitudine/longitudine). Sono previste due categorie di parcheggi: parcheggi liberi (PL) e parcheggi a pagamento coperti (PPC)."*

Frasi relative ai parcheggi liberi: *"Si vuole tenere traccia della durata massima della sosta espressa in ore nel range [0-24]."*

Frasi relative ai parcheggi a pagamento coperti: *"Si vogliono memorizzare numero di telefono, recapito email, nome della società gestore. Ogni parcheggio a pagamento coperto dispone di un certo numero di posti auto. Per i parcheggi a pagamento coperti si vogliono gestire le soste operate dagli utenti. Ogni parcheggio a pagamento coperto può vendere dei PASS per l'accesso a zone a traffico limitato della città di Bari."*

Frasi relative ai posti auto: *"Ogni posto auto dispone di un numero progressivo (univoco per un dato parcheggio), una larghezza, una lunghezza, e può disporre di un sensore associato per il rilevamento automatico dello stato (libero/occupato)."*

Frasi relative ai sensori: *"Nel caso sia presente, si vuole tenere traccia dell'ID del sensore, del modello e del nome dell'azienda produttrice."*

Frasi relative alle soste: *"Ogni sosta dispone di un codice univoco globale, una data di inizio, una data di fine, un costo, ed è associata ad un'auto e ad un utente. Il costo della sosta si ottiene moltiplicando la durata della sosta in ore per un costo unitario orario pari a 0.5 euro."*

Frasi relative agli utenti: *"Si vogliono memorizzare il codice fiscale, nome, cognome, anno di nascita, numero di patente. Sono previste due categorie di utenti speciali: utenti premium (UP), ed utenti abbonati (UA)."*

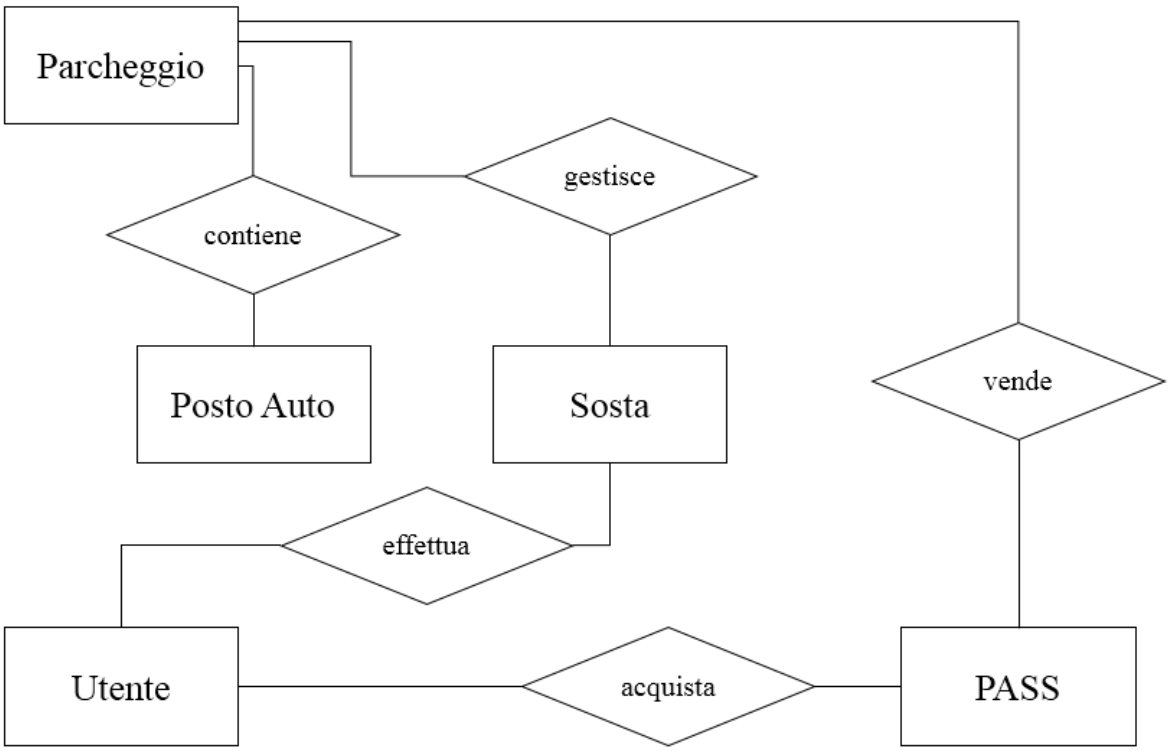
Frasi relative ai PASS: *"Ogni PASS dispone di: codice univoco globale, data di rilascio, durata, auto e nome della zona di validità. Si vuole tenere traccia dello storico degli acquisti di PASS operati da ciascun utente."*

Frasi relative agli utenti premium: *"Si vogliono memorizzare nickname e password per l'accesso Web al portale."*

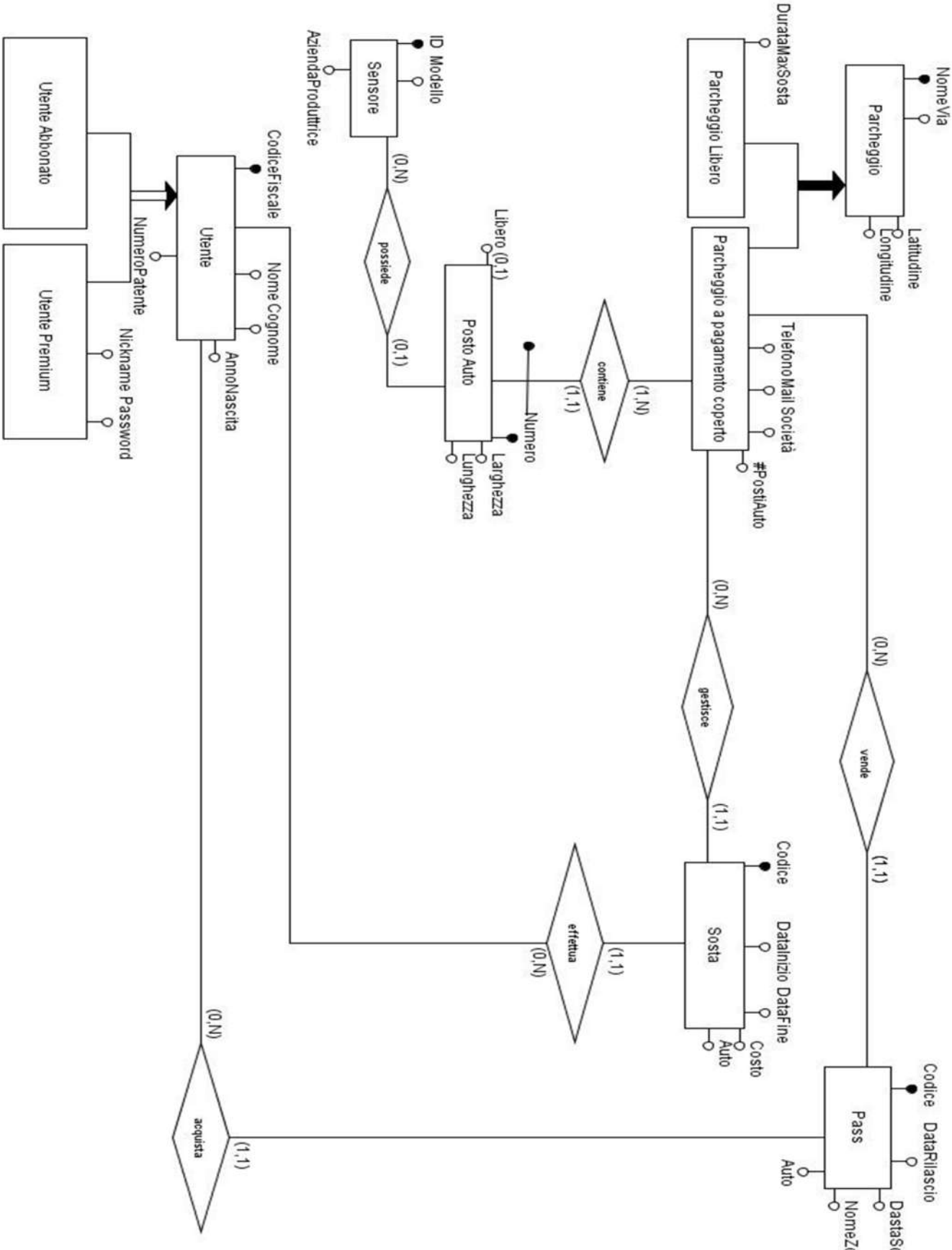
- *Schema E-R*

La strategia seguita nella fase di modellazione concettuale è la strategia ibrida. Si è prima definito uno schema scheletro iniziale, dal quale, attraverso raffinamenti successivi, si è ottenuto lo schema E-R finale.

Schema scheletro:



Schema finale:



Vincoli non espressi dallo schema E-R:

- Per un “Parcheggio Libero”, la durata massima della sosta è nel range [0-24] ore;
- “Costo”, attributo dell’entità “Sosta”, si ottiene moltiplicando la durata della sosta in ore per un costo unitario pari a 0,5 euro;

Scelte fatte nella fase di progettazione concettuale:

- La generalizzazione dell’entità “Parcheggio” è totale, perché si considera che un’area di parcheggio o è libera, o è a pagamento e al coperto.
- La generalizzazione dell’entità “Utente” è parziale, perché si è assunto che possano esistere anche degli utenti che non siano né abbonati né premium, poiché sostano in parcheggi a pagamento coperti raramente, senza la necessità di abbonarsi o di diventare utenti premium.
- Esiste una ridondanza tra l’attributo “#PostiAuto” dell’entità “Parcheggio a pagamento. coperto” e la relazione “contiene”. Infatti il numero di posti auto di uno specifico parcheggio è desumibile contando le occorrenze nella relazione “contiene” di tuple relative a quello specifico parcheggio.

Progettazione Logica

Operazioni

Op.1 Inserimento di un nuovo parcheggio a pagamento coperto (20 volte all'anno)

Op.2 Registrazione di un nuovo utente (20.000 volte all'anno)

Op.3 Inserimento di una sosta in un parcheggio a pagamento coperto (10.000 volte al giorno)

Op.4 Stampa dei posti disponibili in un parcheggio a pagamento coperto (1.000 volte al giorno)

Op.5 Cancellazione di un utente inattivo da 6 mesi (10.000 volte all'anno)

Op.6 Acquisto di un pass (10.000 volte all'anno)

Op.7 Stampa del guadagno giornaliero di un parcheggio a pagamento coperto (100 volte al giorno)

Op.8 Modifica di un parcheggio, a seguito dell'acquisto di un set di sensori per il rilevamento automatico dello stato dei posti auto (10 volte all'anno)

Op.9 Effettua una statistica sugli utenti premium, i pass da loro acquistati e il numero di soste da loro effettuate (5 volte al mese)

Op.10 Calcolo del totale dei posti auto in un parcheggio a pagamento coperto (100 volte al giorno)

Tavola dei volumi

Ciclo di vita del database: 5 anni

Concetto	Tipo	Volume
Parcheggio	E	200
Parcheggio Libero	E	100
Parcheggio a pagamento coperto	E	100
contiene	R	10.000
Posto Auto	E	10.000
possiede	R	2.500
Sensore	E	50
Utente	E	100.000
Utente Anonimo	E	7.000
Utente Premium	E	3.000
gestisce	R	20.000.000

Sosta	E	20.000.000
effettua	R	20.000.000
vende	R	50.000
PASS	E	50.000
acquista	R	50.000

Assunzioni fatte nella redazione della tavola dei volumi:

Si assume che il sistema gestisca 200 parcheggi, di cui 100 liberi e 100 coperti a pagamento. Ogni parcheggio a pagamento coperto contiene in media 100 posti auto, quindi il totale dei posti auto memorizzati sarà di 10.000 (100*100). Per la relazione uno a uno, anche il volume della relazione "contiene" sarà uguale a 10.000.
Si assume che solo venticinque parcheggi su cento possiedano sensori per il rilevamento automatico dello stato (libero/occupato) di un posto auto; perciò il volume di "possiede" equivale a 2.500. Di tipologie di sensori ne esistono 50 diverse.
Si assume che si registrano 20.000 utenti all'anno (vedi operazione 2), quindi il volume di "Utente" sarà uguale a 20.000*5 = 100.000, dato che il ciclo di vita del database è di 5 anni. Di tutti gli utenti, solo una parte è premium (3.000 utenti) o abbonato (7.000), poiché si è assunto che la maggior parte degli utenti usufruisce raramente di un parcheggio a pagamento coperto, senza quindi la necessità di abbonarsi o di diventare utente premium.
Si assume che ogni utente effettui in media 1 sosta alla settimana, considerando che ci siano utenti che ne effettuino 5 o 6 alla settimana per recarsi al lavoro, ma anche utenti che ne sostino in parcheggi una volta ogni 1 o 2 mesi. Perciò il volume di "Sosta" è il risultato arrotondato di 50*5*100.000, circa 20 Milioni. Stesso volume di "effettua" e "gestisce" per la relazione uno a uno.
Infine, in media solo 10.000 utenti acquistano 1 pass all'anno. Quindi il volume di "PASS", come quello di "acquista" e "vende", per la relazione uno a uno, sarà di 10.000*1*5 = 50.000.

Tavola delle operazioni

Operazione	Tipo	Frequenza
Op.1	I	20 / anno
Op.2	I	20.000 / anno
Op.3	I	10.000 / giorno
Op.4	I	1.000 / giorno

Op.5	I	10.000 / anno
Op.6	I	10.000 / anno
Op.7	I	100 / giorno
Op.8	I	10 / anno
Op.9	B	5 / mese
Op.10	I	100 / giorno

Le operazioni di scrittura vengono considerate con costo doppio.

Operazione 1: Inserimento di un nuovo parcheggio a pagamento coperto

Concetto	Tipo	Accesso
Parcheggio	S	1
Parcheggio a pagamento coperto	S	1
contiene	S	100
Posto Auto	S	100

Costo totale operazione 1 = ((1+1+100+100)*2)*20 = circa 8.000 accessi all'anno

Operazione 2: Inserimento di un nuovo utente

Concetto	Tipo	Accesso
Utente	S	1

Costo totale operazione 2 = (1*2)*20.000 = circa 40.000 accessi all'anno

Operazione 3: Inserimento di una sosta in un parcheggio a pagamento coperto

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
gestisce	S	1
Sosta	S	1
effettua	S	1
Utente	L	1

Costo totale operazione 3 = ((3*2)+3)*10.000 = circa 90.000 accessi al giorno

Operazione 4: Stampa dei posti disponibili in un parcheggio a pagamento coperto

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
contiene	L	100
Posto Auto	L	100

Costo totale operazione 4 = 202*1000= circa 200.000 accessi al giorno

Operazione 5: Cancellazione di un utente inattivo da 6 mesi

Concetto	Tipo	Accesso
Utente	S	1

Costo totale operazione 5 = 2*10.000= circa 20.000 accessi all'anno

Operazione 6: Acquisto di un pass

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
vende	S	1
Pass	S	1
acquista	S	1
Utente	L	1

Costo totale operazione 6 = ((3*2)+3)*10.000 = circa 90.000 accessi all'anno

Operazione 7: Stampa del guadagno giornaliero di un parcheggio a pagamento coperto

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
gestisce	L	1000
Sosta	L	1000

Costo totale operazione 7 = 2002*100= circa 200.000 accessi al giorno

Operazione 8: Modifica di un parcheggio, a seguito dell'acquisto di un set di sensori per il rilevamento automatico dello stato dei posti auto

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
contiene	L	100
Posto Auto	L	100
possiede	S	100
Sensore	S	100

Costo totale operazione 8 = (202+(200*2))*10= circa 6.000 accessi all'anno

Operazione 9: Effettua una statistica sugli utenti premium, i pass da loro acquistati e il numero di soste da loro effettuate

Concetto	Tipo	Accesso
Utente	L	1
Utente premium	L	1
acquista	L	1
Pass	L	1
effettua	L	4
Sosta	L	4

Costo totale operazione 9 = 12*5= circa 60 accessi al mese

Operazione 10: Calcolo del totale dei posti auto in un parcheggio a pagamento coperto

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1

Costo totale operazione 10 = 2*100= circa 200 accessi al giorno

Analisi delle ridondanze

Nello schema è presente un solo dato ridondante: l'attributo "#PostiAuto" di "Parcheggio a pagamento coperto" può essere derivato dalla relazione "contiene". L'operazione 10 sfrutta l'attributo ridondante "#PostiAuto". Le tavole degli accessi in presenza e assenza di ridondanza sono le seguenti:

Ridondanza presente

Operazione 10

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
Costo totale operazione = 2*100= circa 200 accessi al giorno		

Ridondanza assente

Operazione 10

Concetto	Tipo	Accesso
Parcheggio	L	1
Parcheggio a pagamento coperto	L	1
contiene	L	1
Costo totale operazione 10 = 3*100= circa 300 accessi al giorno		

In presenza di ridondanza abbiamo un numero minore di accessi al giorno. Si decide perciò di mantenere la ridondanza e lasciare lo schema invariato.

Schema logico

```
CREATE OR REPLACE TYPE utente_ty AS OBJECT (
cf varchar2(16),
nome varchar2(30),
cognome varchar2(30),
anno_nascita integer,
numero_patente varchar2(15)
```

```
) NOT FINAL;


CREATE OR REPLACE TYPE utentepremium_ty UNDER utente_ty (
nickname varchar2(30),
passwd varchar2(8)) FINAL;


CREATE OR REPLACE TYPE utenteabbonato_ty UNDER utente_ty () NOT FINAL;


CREATE OR REPLACE TYPE parcheggio_ty AS OBJECT (
nome varchar2(30),
via varchar2(50),
latitudine decimal(9,6),
longitudine decimal(9,6)) NOT FINAL;


-- 'pl' : parcheggi liberi
CREATE OR REPLACE TYPE pl_ty UNDER parcheggio_ty (
durataMaxSosta number(2)
) FINAL;


-- 'ppc' : parcheggi a pagamento coperti
CREATE OR REPLACE TYPE ppc_ty UNDER parcheggio_ty (
telefono number(8),
mail varchar2(30),
società varchar2(50),
postiauto integer
) FINAL;


CREATE OR REPLACE TYPE postoauto_ty AS OBJECT (
numero integer,
larghezza integer,
lunghezza integer,
libero char(1),
ppc varchar2(30),
sensore ref sensore_ty
) FINAL;

CREATE OR REPLACE TYPE sensore_ty AS OBJECT (
codice varchar2(5),
modello varchar2(20),
azienda varchar2(30)
) FINAL;


CREATE OR REPLACE TYPE sosta_ty AS OBJECT (
codice varchar(8),
dataInizio date,
dataFine date,
costo integer,
targaAuto varchar2(10),
ppc ref ppc_ty,
utente ref utente_ty,
MEMBER FUNCTION
CalcolaCosto
RETURN NUMBER
) FINAL;
CREATE OR REPLACE TYPE BODY sosta_ty AS
MEMBER FUNCTION
CalcolaCosto
```

```
RETURN NUMBER IS
BEGIN
RETURN  24 * (to_date(to_char(dataFine, 'YYYY-MM-DD hh24:mi'), 'YYYY-MM-DD hh24:mi') -
to_date(to_char(dataInizio, 'YYYY-MM-DD hh24:mi'), 'YYYY-MM-DD hh24:mi'))*0.5;
END;
END;


CREATE OR REPLACE TYPE pass_ty AS OBJECT (
codice varchar(5),
dataRilascio date,
scadenza date,
NomeZona varchar2(25),
targaAuto varchar2(10),
ppc ref ppc_ty,
utente ref utente_ty
) FINAL;

--table
CREATE TABLE Utente OF utente_ty (
cf PRIMARY KEY,
nome NOT NULL,
cognome NOT NULL,
anno_nascita NOT NULL,
numero_patente NOT NULL
);

CREATE TABLE UtentePremium OF utentepremium_ty (
nickname NOT NULL,
passwd NOT NULL
);


CREATE TABLE UtenteAbbonato OF utenteabbonato_ty;

CREATE TABLE parcheggio OF parcheggio_ty (
nome PRIMARY KEY,
via NOT NULL,
latitudine NOT NULL,
longitudine NOT NULL
);

CREATE TABLE PL OF pl_ty (
durataMaxSosta NOT NULL
);

CREATE TABLE PPC OF ppc_ty (
nome primary key,
telefono NOT NULL,
mail NOT NULL,
società NOT NULL,
postiauto NOT NULL
);

CREATE TABLE Sensore OF sensore_ty (
codice PRIMARY KEY,
modello NOT NULL,
azienda NOT NULL
);

CREATE TABLE PostoAuto OF postoauto_ty (
larghezza NOT NULL,
```

```
lunghezza NOT NULL,
libero NOT NULL,
sensore SCOPE IS SENSORE,
constraint pa_fk1 foreign key(ppc) references ppc(nome),
    primary key(numero,ppc)
);

CREATE TABLE Sosta OF sosta_ty (
codice PRIMARY KEY,
dataInizio NOT NULL,
dataFine NOT NULL,
targaAuto NOT NULL,
ppc SCOPE IS PPC,
utente SCOPE IS UTENTE
);

CREATE TABLE Pass OF pass_ty (
codice PRIMARY KEY,
dataRilascio NOT NULL,
scadenza NOT NULL,
NomeZona NOT NULL,
targaauto NOT NULL,
ppc SCOPE IS PPC,
utente SCOPE IS UTENTE
);
```

Realizzazione Database

Il database è stato realizzato in Oracle 11g.

Procedure

Procedure di popolamento automatico di tabelle del database:

- "*popolaPPC*", popola la tabella "PPC" con valori generati casualmente;

```
CREATE OR REPLACE PROCEDURE popolaPPC IS
BEGIN
DECLARE
n NUMBER;
BEGIN
n := 0;
WHILE n < 100 LOOP
INSERT INTO ppc VALUES(
  ppc_ty(dbms_random.string('U', 10), dbms_random.string('L',15),
  round(dbms_random.value(1, 100),2),round(dbms_random.value(1, 100),2),
  round(dbms_random.value(1,99999999)), dbms_random.string('L',15), dbms_random.string('L',15),
round(dbms_random.value(50,150))));
n := n + 1;
END LOOP;
END;
END;
```

- "*popolaSensore*", popola la tabella "Sensore" con valori generati casualmente;

```
CREATE OR REPLACE PROCEDURE popolaSensore IS
BEGIN
DECLARE
n NUMBER;
BEGIN
n := 0;
WHILE n < 50 LOOP
INSERT INTO Sensore VALUES(
  sensore_ty(dbms_random.string('X', 5), dbms_random.string('L',15), dbms_random.string('L',15)));
n := n + 1;
END LOOP;
END;
END;
```

- “*popolaUtentePremium*”, popola la tabella “UtentePremium” con valori generati casualmente;

```
CREATE OR REPLACE PROCEDURE popolaUtentePremium IS
BEGIN
DECLARE
n NUMBER;
BEGIN
n := 0;
WHILE n < 5 LOOP
INSERT INTO UtentePremium VALUES(
  utentepremium_ty(dbms_random.string('X', 16), dbms_random.string('L',8),
dbms_random.string('L',8), round(dbms_random.value(1930, 1999)), dbms_random.string('X', 15),
 dbms_random.string('L',8), dbms_random.string('L',8)));
n := n + 1;
END LOOP;
END;
END;
```

- “*popolaUtenteAbbonato*”, popola la tabella “UtenteAbbonato” con valori generati casualmente;

```
CREATE OR REPLACE PROCEDURE popolaUtenteAbbonato IS
BEGIN
DECLARE
n NUMBER;
BEGIN
n := 0;
WHILE n < 300 LOOP
INSERT INTO UtenteAbbonato VALUES(
  utenteabbonato_ty(dbms_random.string('X', 16), dbms_random.string('L',8),
dbms_random.string('L',8), round(dbms_random.value(1930, 1999)), dbms_random.string('X', 15)));
n := n + 1;
END LOOP;
END;
END;
```

- “*popolaPass*”, popola la tabella “Pass” con valori generati casualmente, con utenti e ppc presi casualmente dalle tabelle “Utente” e “PPC”;

```
CREATE OR REPLACE PROCEDURE popolaPass IS
BEGIN
DECLARE
n NUMBER;
ppc ref ppc_ty;
ppc_nome VARCHAR2(30);
utente ref utente_ty;
utente_cf VARCHAR2(16);
d date;
BEGIN
n := 0;
WHILE n < 10 LOOP
  SELECT nome INTO ppc_nome FROM (SELECT * FROM  ppc ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  SELECT cf INTO utente_cf FROM (SELECT * FROM  utente ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  d := TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(DATE '2000-01-
01','J'),TO_CHAR(DATE '9999-12-31','J'))),'J');
INSERT INTO pass select pass_ty(
dbms_random.string('X', 5),
d,
d + 60,
dbms_random.string('L', 15),
dbms_random.string('X', 7),
ref(p),
ref(u)) from ppc p,utente u where p.nome = ppc_nome AND u.cf = utente_cf;
n := n + 1;
END LOOP;
END;
END;
```

- “*popolaSosta*”, popola la tabella “Sosta” con valori generati casualmente, con utenti e pcc presi casualmente dalle tabelle “Utente” e “PPC”;

```
CREATE OR REPLACE PROCEDURE popolaSosta IS
BEGIN
DECLARE
n NUMBER;
ppc ref ppc_ty;
ppc_nome VARCHAR2(30);
utente ref utente_ty;
utente_cf VARCHAR2(16);
d date;
BEGIN
n := 0;
WHILE n < 10 LOOP
  SELECT nome INTO ppc_nome FROM (SELECT * FROM  ppc ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  SELECT cf INTO utente_cf FROM (SELECT * FROM  utente ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  d := TO_DATE(TRUNC(DBMS_RANDOM.VALUE(TO_CHAR(DATE '1900-01-
01','J'),TO_CHAR(DATE '2017-12-31','J'))),'J');
INSERT INTO sosta select sosta_ty(
dbms_random.string('X', 8),
d,
d + 0.25,
0,
dbms_random.string('X', 7),
ref(p),
ref(u)) from ppc p,utente u where p.nome = ppc_nome AND u.cf = utente_cf;
n := n + 1;
END LOOP;
END;
END;
```

- “*popolaPostoAuto*”, popola la tabella “PostoAuto” con valori generati casualmente, con ppc e sensori presi casualmente dalla tabella “PPC” e “Sensore”.

```
create or replace PROCEDURE popolaPostoAuto IS
BEGIN
DECLARE
n NUMBER;
m NUMBER;
ppc_nome VARCHAR2(30);
sensore ref utente_ty;
sensore_codice VARCHAR2(16);
BEGIN
n := 0;
WHILE n < 10 LOOP
  SELECT nome INTO ppc_nome FROM (SELECT * FROM  ppc ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  SELECT codice INTO sensore_codice FROM (SELECT * FROM  sensore ORDER BY
DBMS_RANDOM.RANDOM) WHERE rownum<2;
  SELECT count(*) INTO m FROM postoauto WHERE ppc = ppc_nome;
INSERT INTO postoauto select postoauto_ty(
m+1,
round(dbms_random.value(3,5)),
round(dbms_random.value(3,5)),
'1',
ppc_nome,
ref(s)) from sensore s where s.codice = sensore_codice;
n := n + 1;
END LOOP;
END;
END;
```

Altre procedure, utili per effettuare alcune delle operazioni definite in precedenza:

- “*InserisciPPC*”, per effettuare l’operazione 1. Registra un nuovo ppc con i parametri forniti in input;

```
CREATE OR REPLACE PROCEDURE InserisciPPC (nome PPC.nome%TYPE, via
PPC.via%TYPE,
lat ppc.latitudine%TYPE, lon ppc.longitudine%TYPE, tel ppc.telefono%TYPE,
mail ppc.mail%TYPE, soc ppc.società%TYPE, posti ppc.postiauto%TYPE) IS
BEGIN
INSERT INTO PPC VALUES(nome, via, lat, lon, tel, mail, soc, posti);
dbms_output.put_line('PPC correttamente inserito');
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('PPC non inserito');
END;
```

- "*InserisciUtente*", per effettuare l'operazione 2. Registra un nuovo utente con i parametri forniti in input;

```
CREATE OR REPLACE PROCEDURE InserisciUtente (cf utente.cf%TYPE, nome
utente.nome%TYPE,
cognome utente.cognome%TYPE, annoNasc utente.anno_nascita%TYPE, numPat
utente.numero_patente%TYPE) IS
BEGIN
INSERT INTO utente VALUES(cf, nome, cognome, annoNasc, numPat);
dbms_output.put_line('Utente correttamente registrato');
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Utente non registrato');
END;
```

- "*InserisciSosta*", per effettuare l'operazione 3. Registra una nuova sosta con i parametri forniti in input;

```
CREATE OR REPLACE PROCEDURE InserisciSosta (dataI sosta.datainizio%TYPE,
dataF sosta.datafine%TYPE, costo sosta.costo%TYPE, targaauto sosta.targaauto%TYPE,
ppc_nome ppc.nome%TYPE, utente_cf utente.cf%TYPE) IS
BEGIN
INSERT INTO sosta select sosta_ty(
dbms_random.string('X',5),
dataI,
dataF,
costo,
targaauto,
ref(p),
ref(u)) from ppc p,utente u where p.nome = ppc_nome AND u.cf = utente_cf;
dbms_output.put_line('Sosta correttamente inserita');
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Sosta non inserita');
END;
```

- "*StampaPostiDisponibiliPPC*", per effettuare l'operazione 4. Stampa il numero di posti liberi del ppc fornito in input;

```
CREATE OR REPLACE PROCEDURE StampaPostiDisponibiliPPC (nome ppc.nome%TYPE) IS
BEGIN
DECLARE
n number;
BEGIN
SELECT count(*) INTO n FROM postoauto p WHERE p.ppc = nome AND libero = '1';
dbms_output.put_line('Posti disponibili nel ppc ' || nome || ' : ' || n);
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Stampa non riuscita');
END;
END;
```

- "*CancellaUtentiInattivi*", per effettuare l'operazione 5. Elimina dal sistema gli utenti inattivi, ovvero coloro i quali non effettuino una sosta da più di 6 mesi. Vengono eliminate anche le

soste e i pass acquistati da questi utenti;

```
CREATE OR REPLACE PROCEDURE cancellaUtentiInattivi IS
BEGIN
  FOR utenti IN (SELECT deref(utente).cf AS cf FROM sosta WHERE datafine<sysdate-(30*6))
  LOOP
    delete from pass WHERE deref(utente).cf = utenti.cf;
    delete from sosta WHERE deref(utente).cf = utenti.cf;
    delete from utente where cf = utenti.cf;
  END LOOP;
END;
```

- "*AcquistaPASS*", per effettuare l'operazione 6. Registra un nuovo pass con i parametri forniti in input;

```
CREATE OR REPLACE PROCEDURE AcquistaPASS (ppc_nome ppc.nome%TYPE, utente_cf
utente.cf%TYPE, dataScadenza pass.scadenza%TYPE, nomezona pass.nomeZona%TYPE,
targaauto pass.targaAuto%TYPE) IS
BEGIN
DECLARE
utente_nome utente.nome%TYPE;
BEGIN
SELECT nome INTO utente_nome FROM utente u WHERE u.cf = utente_cf;
INSERT INTO pass select pass_ty(
dbms_random.string('X', 5),
SYSDATE,
dataScadenza,
nomeZona,
targaauto,
ref(p),
ref(u)) from ppc p,utente u where p.nome = ppc_nome AND u.cf = utente_cf;
dbms_output.put_line('Acquisto pass riuscito');
dbms_output.put_line('Venduto da ppc ' || ppc_nome || ' ad utente ' || utente_nome);
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Acquisto pass non riuscito');
END;
END;
```

- "*StampaGuadagnoGiornalieroPPC*", per effettuare l'operazione 7. Stampa il guadagno (la somma dei costi delle soste) di un ppc in una precisa data;

```
CREATE OR REPLACE PROCEDURE StampaGuadagnoGiornalieroPPC (nome ppc.nome%TYPE,
dataF sosta.datafine%TYPE) IS
BEGIN
DECLARE
n number;
BEGIN
SELECT sum(costo) INTO n FROM sosta s WHERE deref(s.ppc).nome = nome AND s.dataFine like
dataF;
dbms_output.put_line('Guadagno ppc ' || nome || ' in data ' || to_date(dataF,'DD-Mon-YY') || ' uguale
a ' || n);
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Stampa non riuscita');
END;
END;
```

- "*ModificaPPCSensori*", per effettuare l'operazione 8. Modifica tutti i posti auto di un dato ppc aggiornandone i sensori;

```
CREATE OR REPLACE PROCEDURE ModificaPPCSensori (nome ppc.nome%TYPE, sensoreCod
sensore.codice%TYPE) IS
BEGIN
DECLARE
n number;
m number;
BEGIN
n := 0;
SELECT count(*) INTO m FROM postoauto pa WHERE pa.ppc = nome;
WHILE n < m LOOP
update postoauto p set sensore = (SELECT ref(s) FROM sensore s WHERE s.codice = sensoreCod
) WHERE p.ppc = nome;
n := n + 1;
END LOOP;
dbms_output.put_line('Modifiche correttamente avvenute');
EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('Modifiche non riuscite');
END;
END;
```

Trigger

- "*CalcolaCostoSosta*", calcola il costo di una sosta a seguito dell'inserimento della stessa all'interno della tabella "Sosta". Si sfrutta il metodo "calcolacosto" del tipo di dato "sosta_ty";

```
CREATE OR REPLACE TRIGGER calcolacostososta
after insert on sosta
BEGIN
update sosta s set costo = s.calcolacosto() where codice = codice;
END;
```

- "*AggiornaTabellaUtente1*", aggiorna la tabella utente a seguito dell'inserimento di un utente premium nella tabella "UtentePremium";

```
CREATE OR REPLACE TRIGGER aggiornaTabellaUtente1
after insert on utentepremium
for each row
BEGIN
INSERT INTO UTENTE
VALUES(:new.cf,:new.nome,:new.cognome,:new.anno_nascita,:new.numero_patente);
END;
```

- "*AggiornaTabellaUtente2*", aggiorna la tabella utente a seguito dell'inserimento di un utente abbonato nella tabella "UtenteAbbonato";

```
CREATE OR REPLACE TRIGGER aggiornaTabellaUtente2
after insert on utenteabbonato
for each row
BEGIN
INSERT INTO UTENTE
VALUES(:new.cf,:new.nome,:new.cognome,:new.anno_nascita,:new.numero_patente);
END;
```

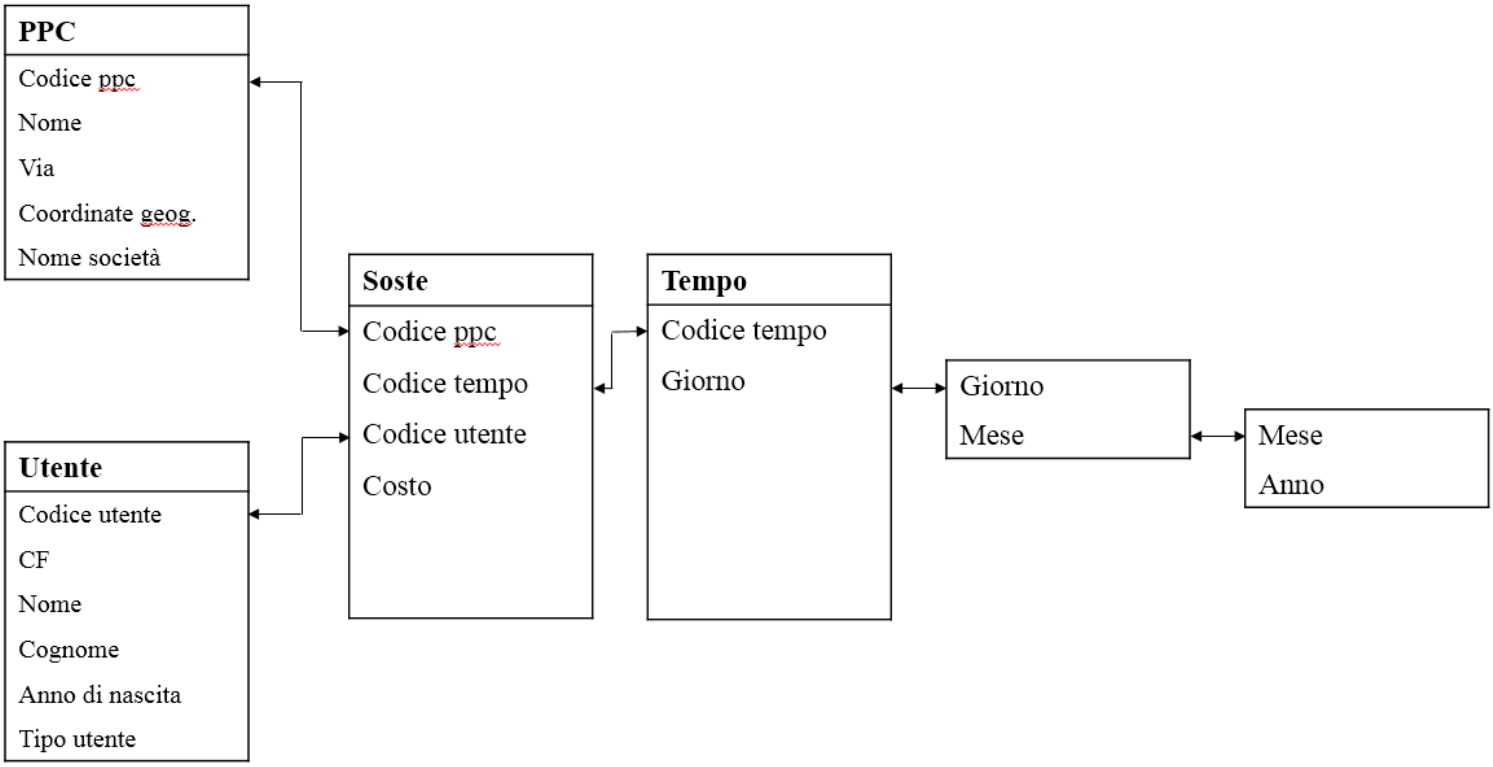
Datawarehouse

Obiettivo di business: Analizzare le soste effettuate in parcheggi a pagamento coperti per studiarne il costo, in base al parcheggio, al periodo di tempo in cui sono avvenute, e all'utente che le ha effettuate.

Fatto: Soste di un parcheggio a pagamento coperto.

Dimensioni: Parcheggio a pagamento coperto, Tempo, Utente

Schema a fiocco di neve



Realizzazione

La datawarehouse è stata realizzata in "Oracle Warehouse Builder" versione 11gR2.

Servlet

1.Servlet *"ServletStampaInfoPremium"*

Utile per la stampa di informazioni riguardo agli utenti premium registrati nel database. In particolare, per ogni utente premium, vengono stampate il numero di soste effettuate e, per ogni sosta, presso quale parcheggio a pagamento coperto è stata effettuata, la data di inizio e la data di fine. Viene inoltre stampato il numero di pass acquistati dall'utente, e, per ogni pass, la zona di validità e la scadenza.

2.Servlet *"ServletInserimentoSosta"*

Utile per inserire una sosta all'interno del database. Viene presentato un form che richiede l'inserimento di informazioni riguardo alla sosta:

- nome del parcheggio a pagamento coperto presso il quale la sosta è avvenuta
- codice fiscale dell'utente che ha effettuato la sosta
- data di inizio della sosta
- data di fine della sosta
- targa dell'auto

Dopo aver sottomesso i dati, verrà mostrato un messaggio sull'esito dell'operazione, ovvero se la sosta è stata inserita o no nel database.