UNIVERSITÀ
DEL SALENTO

Relazione del progetto "Car sharing"

Principi di progettazione del Software
Facoltà di Ingegneria - Laurea in Ingegneria dell'Informazione
Elaborato Software valido per l'a.a. 2019-2020
Docenti: Mainetti Luca, Vergallo Roberto
Alunni: Carrozzini Pierantonio, Piscopo Alessandro

Sommario

Traccia

Una catena di autonoleggio vuole realizzare un sistema per consentire ai suoi clienti di noleggiare automezzi online con opzione di "car sharing".
Chiunque può accedere al sistema ottenendo informazioni sul servizio e visitando il catalogo degli automezzi ed eventuali offerte disponibili. Un utente interessato a noleggiare un automezzo, si registra al sistema e diviene cliente specificando il proprio profilo (nome, cognome, telefono, e-mail, residenza, età, foto).
Ottenute le credenziali dal sistema, il cliente si connette al sistema e fornisce le proprie preferenze in termini di data d'inizio e di fine noleggio, modello di automezzo da noleggiare (può noleggiare auto, furgoni, camion, camper), categoria (questa informazione è valida solo per le auto; esistono tre categorie di auto: piccola, media, grande), tipo di motorizzazione (elettrica, benzina, diesel), accessori che desidera siano presenti (navigatore, seggiolino bimbo, catene da neve, carrellino, barre per il trasporto biciclette, ecc.), stazione di ritiro e riconsegna dell'automezzo, località principale che vuole raggiungere durante il noleggio.
Modelli, marche, categorie, accessori devono essere gestiti in modo esplicito e indipendente da altre informazioni nel sistema, inoltre per ogni marca e modello di automezzo il sistema memorizza una opiù fotografie.
All'atto della prenotazione, il sistema può proporre al cliente uno sharing se rileva altre richieste di noleggio con le stesse caratteristiche temporali, di stazione ritiro e consegna, e di località principale da raggiungere. In tal caso il cliente può vedere i profili degli altri clienti coinvolti nello sharing e accettare o rifiutare la proposta di sharing. In caso affermativo, i clienti coinvolti sono avvisati per e-mail. Quando lo ritiene opportuno, il cliente può modificare una prenotazione (eventualmente anche aggiungendo o rimuovendo accessori richiesti), oppure può cancellare una prenotazione.
Tutti gli interessati al car sharing della prenotazione sono avvisati per e-mail. Il sistema fornisce il calcolo del costo totale del noleggio per ogni nuova prenotazione o modifica di prenotazione.
Un operatore di sportello di una delle stazioni dell'autonoleggio periodicamente si connette al sistema, verifica le prenotazioni per la sua stazione e controlla che i pagamenti siano andati a buon fine.
Un addetto al parco automezzi di una delle stazioni dell'autonoleggio periodicamente si connette al sistema e prepara gli automezzi della sua stazione con gli accessori richiesti nelle prenotazioni; quando un automezzo è pronto, l'addetto lo segnala all'operatore di sportello tramite il sistema.
Un amministratore del servizio periodicamente si connette al sistema verificando le prenotazioni e controllando i pagamenti di tutte le stazioni della catena di autonoleggio; può filtrare gli elenchi per data, per stazione di noleggio, per modello, per marca; può stampare ogni elenco in formato PDF; quando riscontra anomalie tramite il sistema può inviare messaggi agli operatori di sportello oppure agli addetti al parco automezzi che i medesimi possono leggere dandone riscontro all'amministratore.
I messaggi scambiati sono registrati nella base di dati.

Specifica dei requisiti

Si vuole realizzare un'applicazione in grado di gestire una piattaforma di car sharing con le specifiche richieste dalla Traccia.

In particolare, si vuole tenere si vuole tenere traccia delle informazioni relative a:

1. le credenziali degli utenti divisi in Addetto, Operatore, Amministratore e Cliente (e-mail, password, username);
2. l'anagrafica dei clienti (nome, cognome, residenza, numero di telefono, foto);
3. le informazioni sulle stazioni (nome, longitudine e latitudine);
4. le informazioni sulle località (città, longitudine e latitudine);
5. le specifiche su ogni mezzo (targa, prezzo);
6. i dettagli sui modelli (nome, numero posti, tipo di veicolo, foto);
7. i diversi tipi di marche presenti (nome);
8. i diversi tipi di categorie presenti (cilindrata);
9. gli accessori disponibili (nome accessorio, prezzo);
10. le prenotazioni contenute nell' intera base di dati (data, numero posti occupati, data inizio e fine viaggio, il prezzo totale, la conferma di pagamento).
    Le prenotazioni contengono inoltre le scelte fatte dal cliente riguardo a località da raggiungere, stazione di ritiro e consegna del mezzo, mezzo e accessori scelti.
11. le richieste di sharing pervenute per ogni prenotazione.

Dal punto di vista dei requisiti funzionali:

1. L'utente dovrà essere in grado di visualizzare e filtrare il catalogo;
2. Il sistema fornirà le credenziali ad ogni registrazione via e-mail;
3. Il sistema riconoscerà automaticamente ogni utente loggato tramite una ricerca nella base di dati;
4. Il sistema gestisce la base di dati inibendo gli elementi non disponibili a nuove prenotazioni;
5. Il cliente avrà la possibilità di inserire una nuova prenotazione dopo essersi loggato e di inserire tutte le sue preferenze nel caso siano presenti nella base di dati;
6. Il sistema calcola i prezzi totali di ogni operazione,
7. Ogni prenotazione avrà un identificatore unico (idprenotazione) che il cliente potrà usare per modificare o cancellare la sua prenotazione;
8. Ad ogni prenotazione il sistema, se dovesse riscontrare delle somiglianze con prenotazioni precedenti, potrebbe mostrare al cliente interessato tutte le prenotazioni simili alla sua;
9. L'operatore può confermare i pagamenti di ogni prenotazione rendendola effettivamente pronta;
10. L'addetto ha la possibilità di segnalare che la prenotazione abbia gli accessori richiesti rendendo il mezzo pronto;
11. Amministratore, operatore e addetto possono scambiare messaggi;
12. Il sistema gestisce la tabella messaggi di amministratore, operatore e addetto;
13. Il sistema invia una e-mail a tutti gli interessati di uno sharing in caso di modifica o cancellazione di una prenotazione da parte di uno dei clienti interessati;
14. Il sistema invia per e-mail un pdf contenente le specifiche di ogni nuova prenotazione.

Dal punto di vista dei requisiti non funzionali:

1. L'interfaccia è stata implementata con un IDE Java, nello specifico è stato utilizzato il programma IntelliJ IDEA;
2. Il sistema non deve rivelare ai clienti il proprio codice utente;
3. Il sistema non può permettere ai clienti di vedere le prenotazioni altrui;
4. Il sistema potrà essere usato totalmente solo dai suoi creatori;
5. Solo il cliente può vedere la propria password.

Progettazione UML

Attori: Guest, Cliente, Amministratore, Addetto, Operatore.

Definizione dei casi d' uso:

1. Filtra catalogo [Utente] *include* Visualizza offerte [Utente, Cliente], *include* Stampa Catalogo [Amministratore];
2. Login [Cliente, Operatore, Addetto, Amministratore] *estende* Registrazione;
3. Prenotazione [Cliente] *include* Sharing [Cliente];
5. Gestione Profilo [Cliente] *include* Modifica Prenotazione;
6. Verifica pagamenti prenotazioni [Operatore];
7. Segnala mezzo pronto [Addetto];
8. Invia messaggio [Amministratore, Operatore, Addetto];

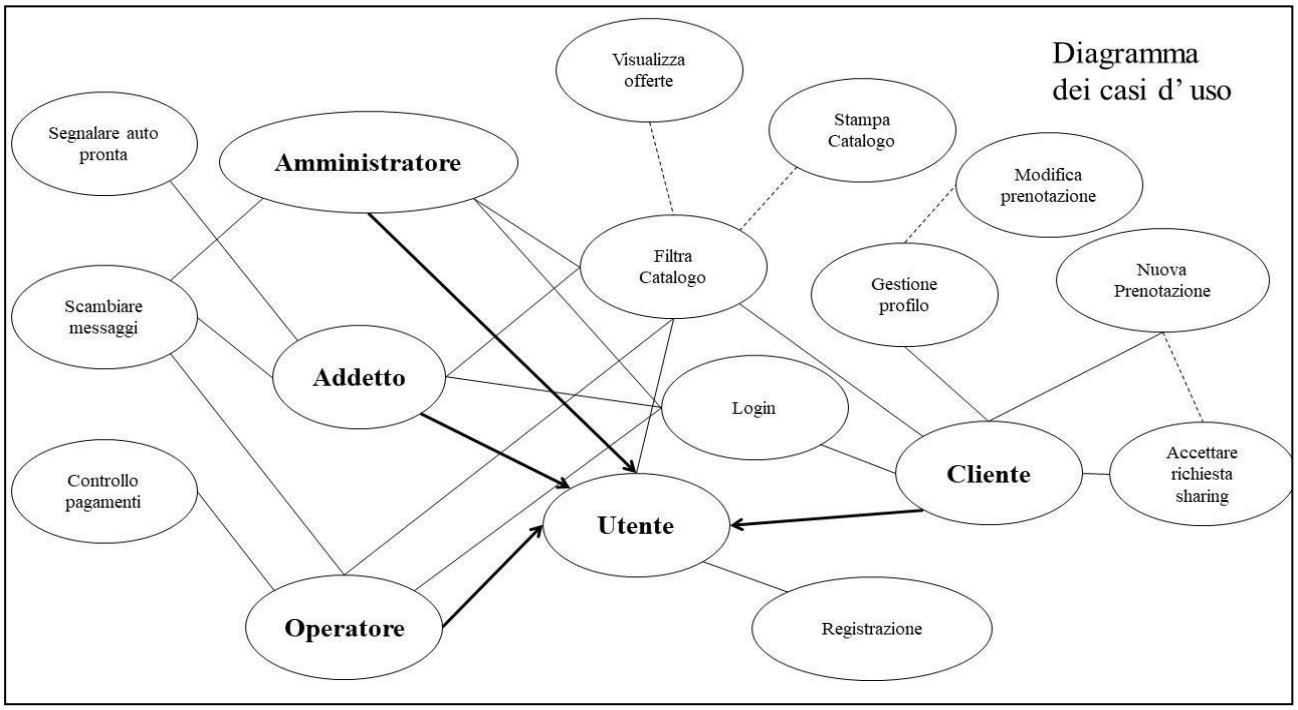

Figura 1. Diagramma dei casi d' uso.

Estraiamo 3 casi d' uso:

Filtra catalogo

1. Filtra Catalogo:
    - Utente si connette al sistema;
    - Dopo l'autenticazione il sistema mostra il catalogo;
    - Amministratore, Utente e Cliente possono filtrare il catalogo di tutti i mezzi presenti nel car sharing indicando nei filtri carburante, veicolo e cilindrata;
    - Il sistema mostrerà tutti i mezzi presenti indicandoli, qualora fossero occupati, con la dicitura "Occupato".
2. *Stampa catalogo:*
    - Amministratore può stampare i cataloghi una volta filtrati;
    - il sistema invierà una e-mail all' amministratore contenente un PDF con l'elenco scelto.
3. *Visualizza offerte:*
    - Utente e Cliente hanno la possibilità di visualizzare le offerte che il sistema produrrà casualmente.

Gestione profilo

1. Gestione Profilo:
    - Utente si connette al sistema;
    - Dopo l'autenticazione il sistema mostra il catalogo;
    - Il Cliente entra nella finestra "gestione profilo";
    - Il sistema mostra le vecchie prenotazioni e le prenotazioni in corso o un elenco vuoto nel caso in cui il cliente non avesse ancora fatto una prenotazione;

2. *Modifica Prenotazione*
    - Il Cliente, dopo aver visto le sue prenotazioni, seleziona il codice della prenotazione che ha intenzione di modificare o cancellare;
    - Il sistema controlla che sia ancora possibile, in termini temporali, modificare o cancellare la prenotazione e mostra al Cliente la finestra Modifica Prenotazione;
    - Il Cliente, una volta entrato nella finestra Modifica Prenotazione, inserisce le nuove preferenze relative alla prenotazione scelta in precedenza oppure cancella la prenotazione;
    - Il sistema aggiorna il prezzo della prenotazione e la pagina Gestione Profilo;
    - Il cliente informa, nel caso in cui la prenotazione fosse inserita in uno sharing, tutti gli interessati attraverso una e-mail.

Prenotazione

1. Prenotazione
    - Utente si connette al sistema;
    - Dopo l'autenticazione il sistema mostra il catalogo;
    - Il Cliente che vuole inserire una prenotazione entra nella finestra nuova Prenotazione;
    - Il sistema mostra al Cliente, tutti i campi da completare per inserire la nuova prenotazione;

- Il Cliente inserisce le preferenze;
- Il Sistema filtra le scelte del Cliente indicandogli quelle disponibili nella base di dati;
- Dopo aver scelto i requisiti della nuova prenotazione il cliente prosegue nella scelta degli accessori;
- Il sistema inserisce gli accessori nella prenotazione e inserisce la prenotazione nella base di dati;

2. *Sharing*
   - Dopo aver inserito la prenotazione il Sistema mostra al Cliente tutte le prenotazioni con le caratteristiche simili a quella appena inserita proponendo uno sharing;
   - Il cliente sfoglia le richieste e decide se accettare o non accettare lo sharing;
   - Dopo aver deciso tutte le richieste vengono memorizzate nella base di dati con il relativo stato.

Schede CRC

Con le schede CRC iniziamo a definire le responsabilità delle classi a partire dalla classe Sistema (Figura 2).

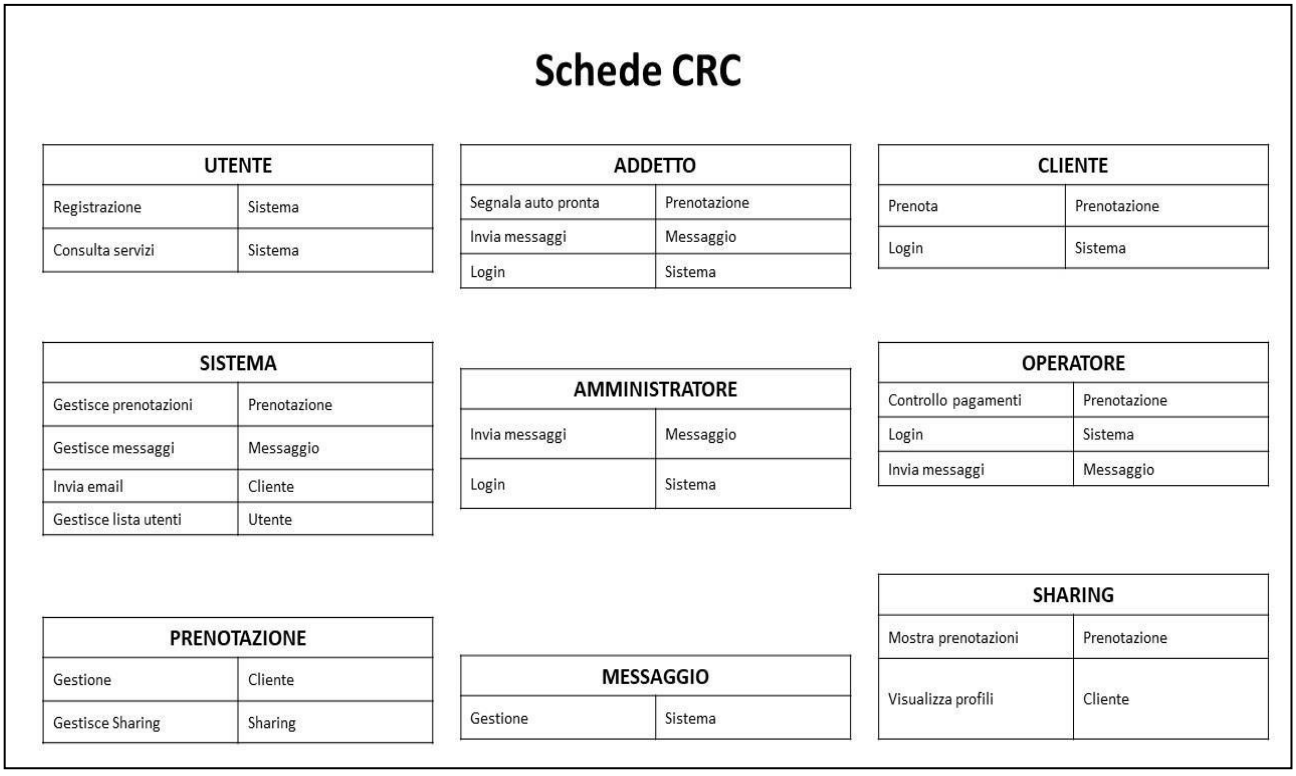

Figura 2. Schede CRC

Diagramma delle dipendenze

Abbiamo desunto le relazioni di dipendenza dai collaboratori nelle schede CRC (Figura 3).

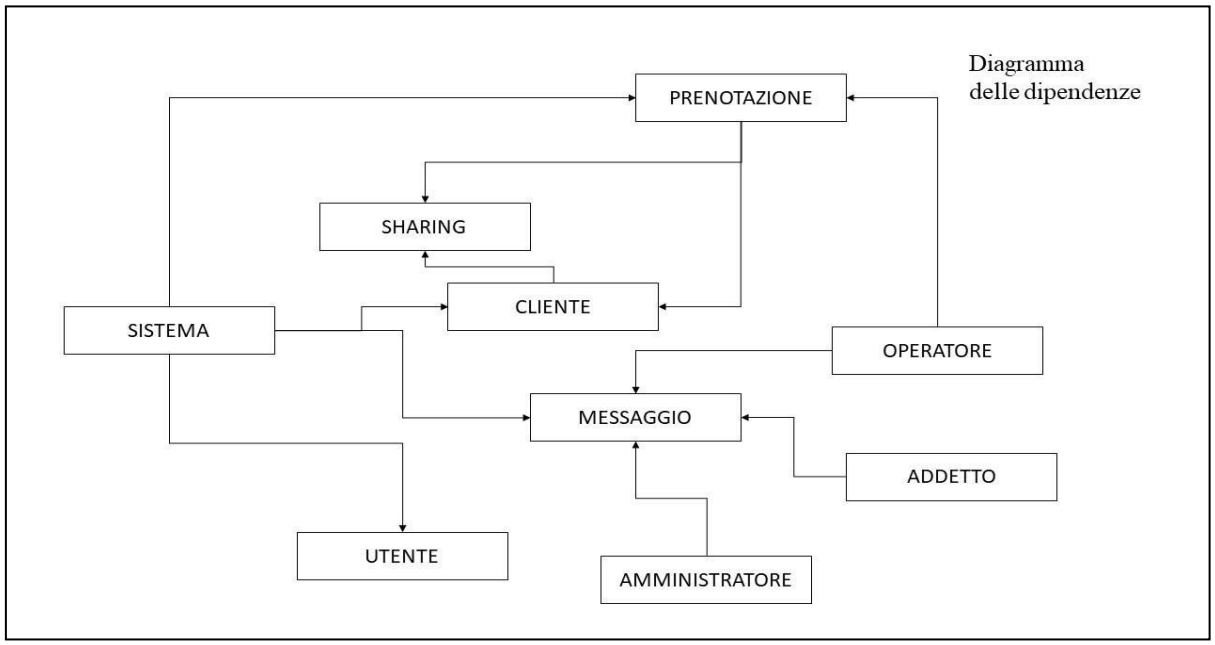

Figura 3.Diagramma delle dipendenze.

Diagramma di sequenza

Implementazione scenario nuova prenotazione cliente (Figura 4).

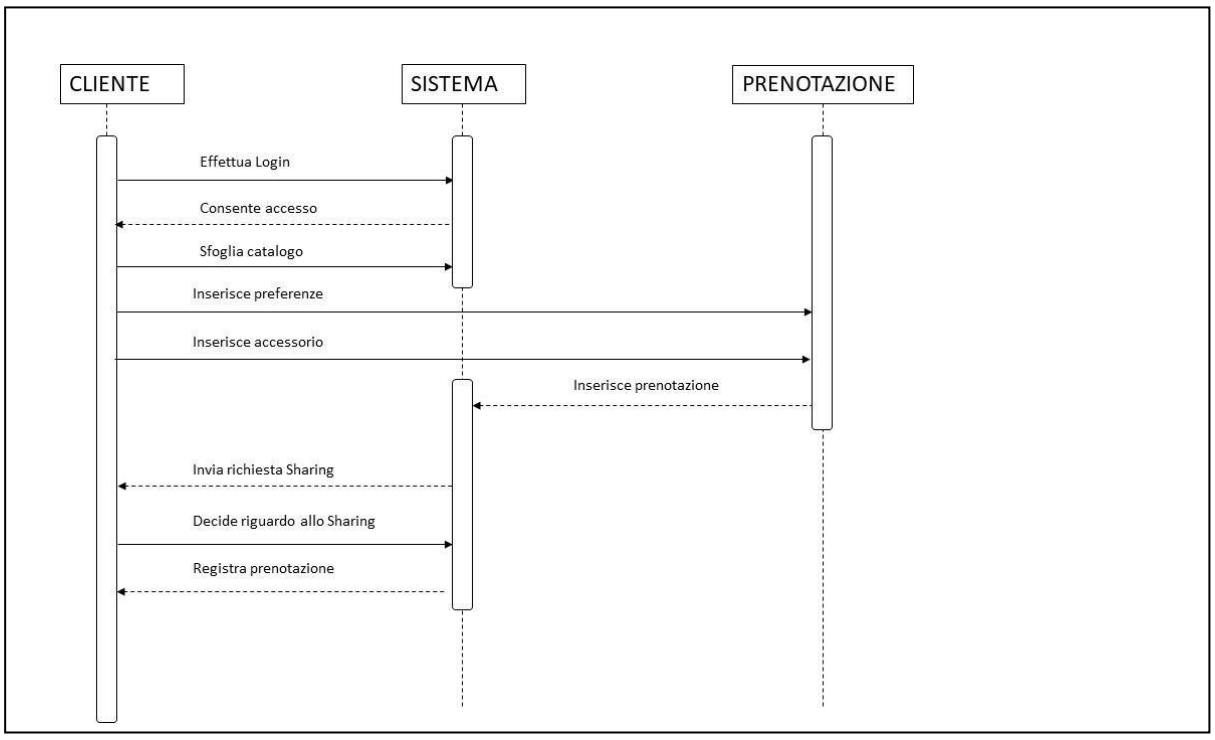

Figura 4. Diagramma di sequenza

Progettazione Logica

Schema Relazionale

Per convenzione, nella definizione dello schema, i nomi delle tabelle sono riportati in grassetto, le chiavi primarie di ciascuna tabella sono indicate con la sottolineatura, mentre le chiavi esterne con il prefisso "FK". I nomi degli attributi e delle tabelle riportati nelle descrizioni sono in corsivo.

Accessorio (idaccessorio, nome, prezzo)
idaccessorio è chiave primaria di accessorio.

Addetto (utente_idutente)
FK_utente_idutente è chiave primaria della tabella Addetto.

Amministratore (utente_idutente)
FK_utente_idutente è chiave primaria della tabella Amministratore.

Categoria (idcategoria, cilindrata)
idcategoria è chiave primaria di categoria.

Cliente (utente_idutente, nome, cognome, residenza, data_nascita, cellulare, foto)
FK_utente_idutente è chiave primaria della tabella cliente.

Contiene (idcontiene, FK_accessorio_idaccessorio, FK_prenotazione_idprenotazione)
Idcontiene è chiave primaria della tabella contiene;
Vincolo di integrità referenziale tra FK_accessorio_idaccessorio e l'attributo idaccessorio della tabella accessorio;
Vincolo di integrità referenziale tra FK_prenotazione_idprenotazione e l'attributo idprenotazione della tabella prenotazione.

Localita (idlocalita, tipo, latitudine, longitudine)
idlocalita è chiave primaria della tabella localita.

Marca (idmarca, tipo)
idmarca è chiave primaria di marca.

Messaggio (idmessaggio, utente_idutente, testo, data, stazione)
Idmessaggio è chiave primaria della tabella messaggio;
Vincolo di integrità referenziale tra FK_utente_idutente e l'attributo idutente della tabella Utente.

Mezzo (idmezzo, targa, FK_modello_idmodello, prezzo, carburante)
idmezzo è chiave primaria della tabella mezzo;
Vincolo di integrità referenziale tra FK_modello_idmodello e l'attributo idmodello della tabella modello.

Modello (idmodello, nome, num_posti, FK_categoria_idcategoria, FK_marca_idmarca, veicolo, foto)
Idmodello è chiave primaria della tabella modello;
Vincolo di integrità referenziale tra FK_categoria_idcategoria e l'attributo idcategoria della tabella categoria;
Vincolo di integrità referenziale tra FK_marca_idmarca e l'attributo idmarca della tabella marca.

Operatore (utente_idutente)
FK_utente_idutente è chiave primaria della tabella operatore.

Prenotazione (idprenotazione, data, cliente_idcliente, FK_mezzo_idmezzo, num_posti_occupati, FK_idstazione_partenza, FK_idstazione_arrivo, FK_localita_idlocalita, dataInizio, dataFine, costoTotale, pronta, pagamento)
Idstazione è chiave primaria della tabella prenotazione;
Vincolo di integrità referenziale tra FK_cliente_idcliente e l'attributo utente_idutente della tabella Cliente;
Vincolo di integrità referenziale tra FK_mezzo_idmezzo e l'attributo idmezzo della tabella mezzo;
Vincolo di integrità referenziale tra FK_idstazione_partenza e l'attributo idstazione della tabella stazione;
Vincolo di integrità referenziale tra FK_idstazione_arrivo e l'attributo idstazione della tabella stazione;
Vincolo di integrità referenziale tra FK_localita_idlocalita e l'attributo idlocalita della tabella localita.

Stazione (idstazione, nome, latitudine, longitudine, FK_operatore_utente_idutente, FK_addetto_utente_idutente)
Idstazione è chiave primaria della tabella Utente;
Vincolo di integrità referenziale tra FK_operatore_utente_idutente e l'attributo utente_idutente della tabella Operatore;
Vincolo di integrità referenziale tra FK_addetto_utente_idutente e l'attributo utente_idutente della tabella Addetto.

Utente (idutente, username, password, e-mail)
idutente è chiave primaria della tabella utente.

Schema logico

Di seguito viene mostrato lo schema logico della base di dati (figura 5).

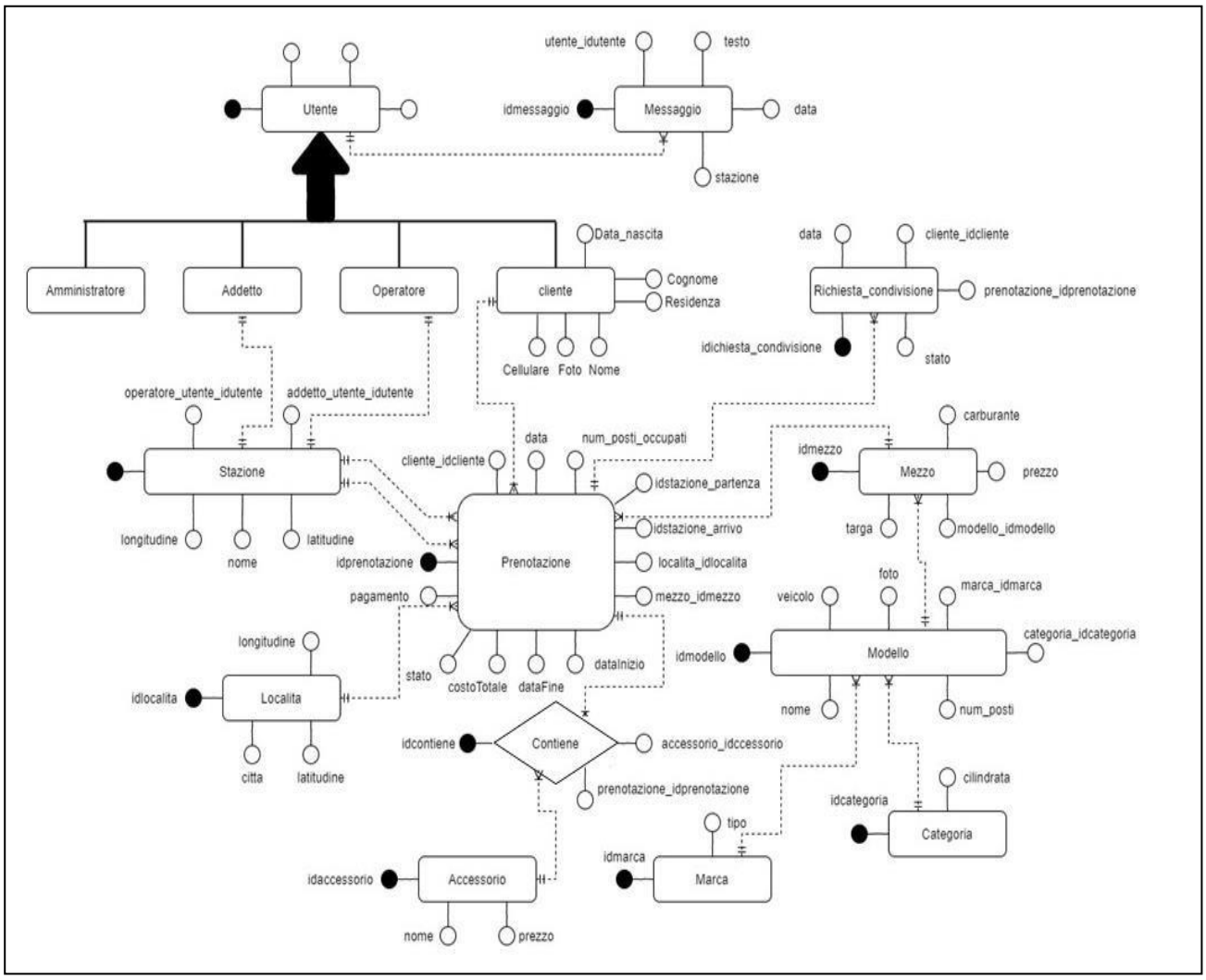

Figura 5.Diagramma E/R

Sprint backlog

Di seguito lo Sprint Backlog delle ore di lavoro stimate ad inizio progetto e le relative ore effettuate per ogni Task. L' obiettivo finale è una applicazione in java che permette di gestire un car sharing (Figura 6).

	ORE STIMATE	ORE EFFETTIVE
Analisi Requisiti	5	3
Progettazione	5	7
Database	15	20
DAO	35	28
Model	20	16
Fxml	25	30
Controller	55	50
Util	25	20
Test/Code cleanup	15	20
Debugging	10	7
Improving	20	25
Relazione	5	2
TOTALE	235	228

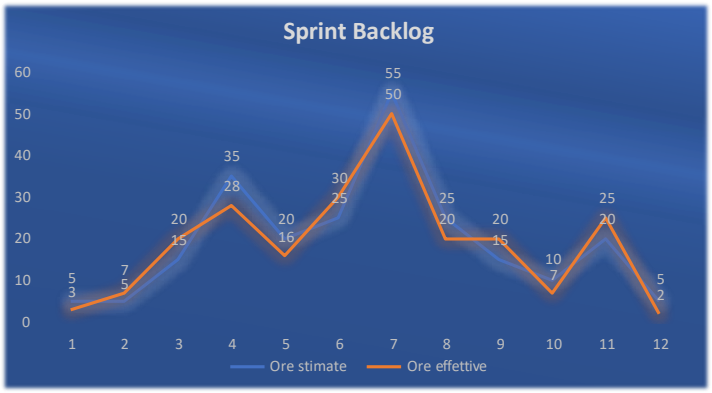

Figura 6.Sprint Backlog

Burndown chart

Di seguito la rappresentazione grafica delle ore di lavoro, indicando sull' asse verticale il lavoro rimanente e sull' asse orizzontale i vari step (Figura 7).

		ORE EFFETTIVE	ORE STIMATE
1	**Analisi Requisiti**	225	230
2	**Progettazione**	218	225
3	**Database**	198	210
4	**DAO**	170	175
5	**Model**	154	155
6	**Fxml**	124	130
7	**Controller**	74	75
8	**Util**	54	50
9	**Test/Code cleanup**	34	35
10	**Debugging**	27	25
11	**Improving**	2	5
12	**Relazione**	0	0
	Totale	228	235

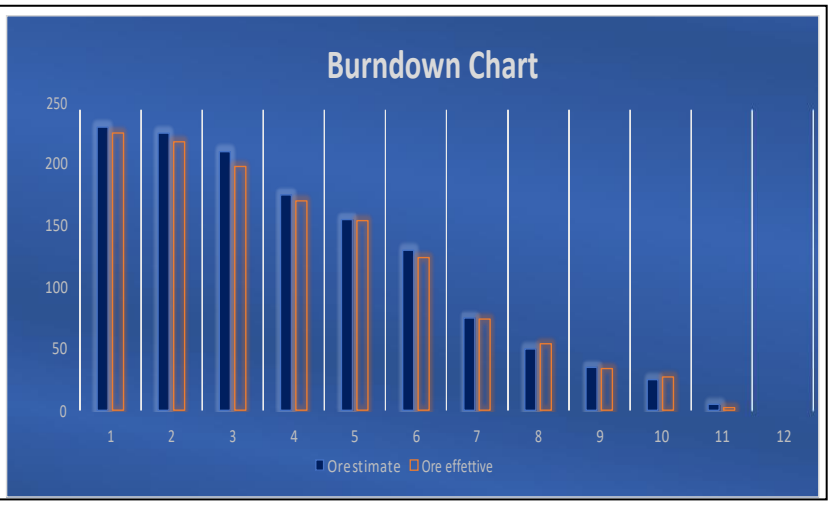

Figura 7.Burndown Chart

Base di dati

Di seguito viene mostrata la base di dati completa, contenente anche i tipi di dati (Figura 8).

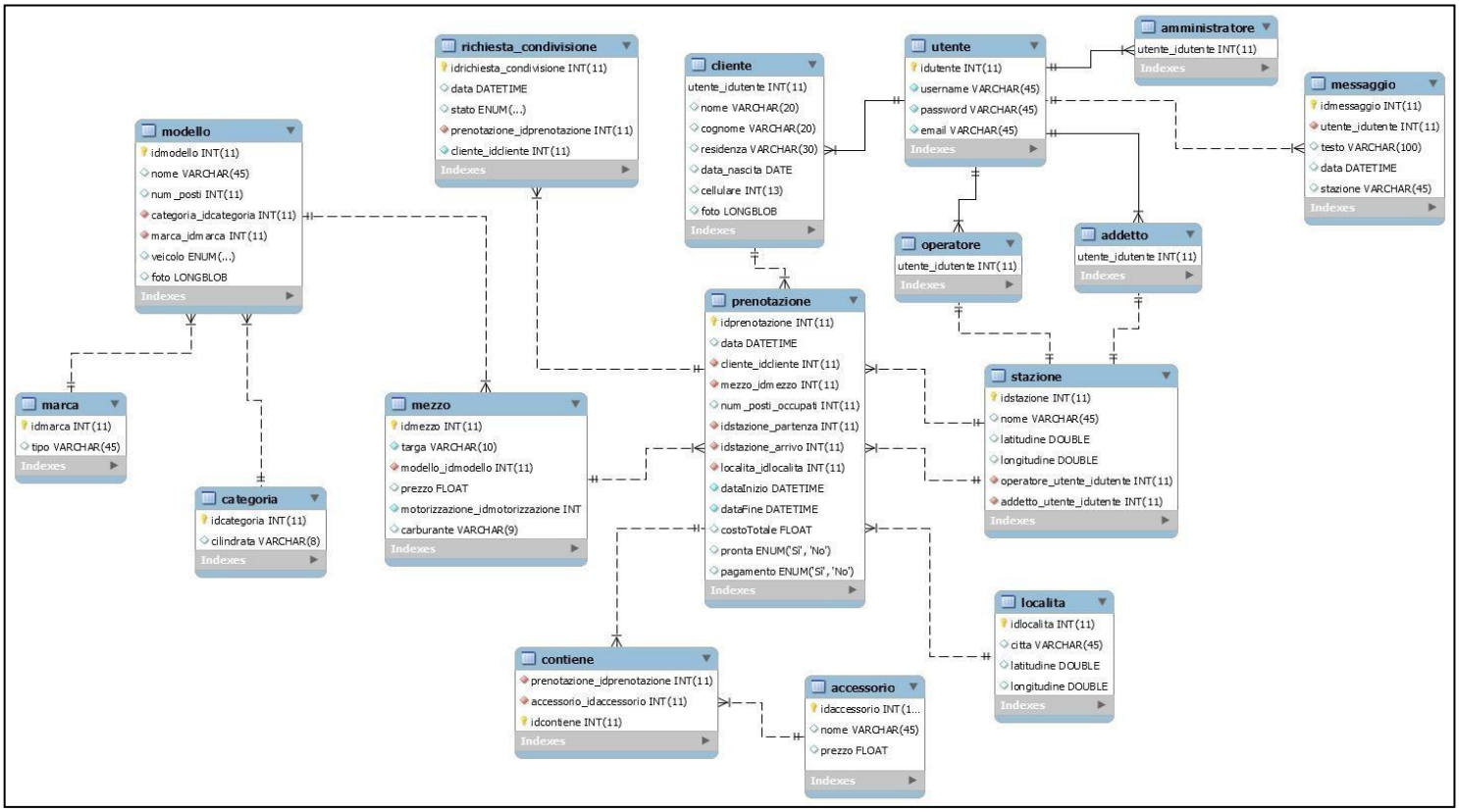

Figura 8.Base di dati completa

Definizione dei DAO utilizzati

IBaseDAO:

- **public T findById(int id);**
- **public ArrayList<T> findAll().**

IAddettoDAO, IOperatoreDAO, IAmministratoreDAO:

- **Addetto findUtenteByField(String name, String value);**
- **Addetto findByUsername(String username).**

IMezzoDAO:

- **ArrayList<Mezzo> findByFiltri(String a,String b,String c,String d,String e,String f,String g,String h,String i,String j);** (*Questo DAO permette di trovare i mezzi che soddisfano TUTTI i filtri scelti dall'utente/cliente nella prima finestra dell'applicazione).*

IModelloDAO:

- **ArrayList<Modello> findByFiltri(String a,String b,String c,String d,String e,String f,String g,String h,String i,String j);** (*Questo DAO permette di trovare i modelli che soddisfano TUTTI i filtri scelti dall'utente/cliente nella prima finestra dell'applicazione.).*

IStazioneDAO:

- **Stazione findStazioneByField(String name, String id);** (T*rova la stazione scelta a partire da un filtro e un valore per quest'ultimo*);
- **Stazione findByAddetto(String id);** (*Cerca la stazione dell'addetto a partire dal suo id*);
- **Stazione findByOperatore(String id);** (*Trova la stazione dell'operatore a partire dal suo id*).

IMessaggiDAO:

- **ArrayList<Messaggio> findByField(String name, String value);** (T*rova il messaggio a partire da un filtro e un valore dato a quest'ultimo*);
- **ArrayList<Messaggio> findByStazione(String stazione);** (*Carica tutti i messaggi per la stazione della quale viene passato il nome*);
- **void addMessage(Messaggio mess);** (*Aggiunge un nuovo messaggio al database ogni qualvolta viene utilizzato*).

IAccessorioDAO:

- **ArrayList<Integer> findByPrenotazione(int id);** (*Restituisce un array di interi quindi una lista con id di accessori*);
- **ArrayList<Accessorio> findAccessorioByPrenotazione(int id);** *(Restituisce un array di accessori che comprende anche il loro nome e prezzo*);
- **ArrayList<Accessorio> findAccessorio(ArrayList<Integer> a);** (*Prende in input una lista di id e ritorna una lista di accessori*).

IClienteDAO:

- **void salvaUtente(Cliente c, String pass);** (*salvaUtente salva il nuovo cliente nella tabella utente, con username mail e password generata dal metodo*);
- **void salvaCliente(Cliente c);** (*salvaCliente procede a salvare tutti i restanti dati del cliente*);
- **Utente findUtenteByField(String name, String value);** (*Trova un utente a partire da username [findByUsername] o a partire da un id [findUtenteById]*);
- **Utente findByUsername(String username);**
- **Utente findUtenteById(String id);**
- **ArrayList<Cliente> prenotazioneCondivisa(int id);** (*Cerca in richiesta_condivisione per controllare se il cliente di cui verrà dato l'id in input ha preso parte o meno a degli sharing*).

IPrenotazioneDAO:

- **ArrayList<Prenotazione> findByFiltri(String nome, String valore);** (*Applica i filtri che vengono passati da [findPrenotazioneByStaz] e [findPrenotazioneByCliente])*
- **ArrayList<Prenotazione> findPrenotazioneByStaz(String valore);**
- **ArrayList<Prenotazione> findPrenotazioneByCliente(String valore);**
- **ArrayList<Prenotazione> findPrenotazioneByModello(String valore);** (Consente *all' amministratore di filtrare per modello*);
- **ArrayList<Prenotazione> findPrenotazioneByMarca(String valore);** (*Consente all' amministratore di filtrare per marca*);
- **ArrayList<Prenotazione> findPrenotazioneByData(Date d);** (*Consente all' amministratore di filtrare per data);*
- **ArrayList<Prenotazione> findSharing(int id);** (T*rova lo sharing a partire dall'id di un cliente*);
- **void salvaPrenotazione(Prenotazione p, ArrayList<Integer> a);** (*Salva la prenotazione effettuata da un utente e i suoi accessori nella tabella contiene*);
- **void salvaSharing(Prenotazione p, ArrayList<Integer> a, Utente u);** (*Salva lo sharing nel caso in cui un cliente accetta*);
- **void modificaPrenotazione(Prenotazione p, ArrayList<Integer> a);** (P*ermette di modificare la prenotazione*);
- **void modificaOperatore(Prenotazione p);** (*Si utilizza nel caso in cui l'operatore voglia modificare lo stato del pagamento di un auto questo metodo modificherà il valore del pagamento da No di dafault a sì*);
- **void delete(int id);** (*Cancella la prenotazione effettuata da un cliente*);
- **void deleteSharing(Prenotazione p, Utente u);** (*Cancella un cliente che aveva accettato uno sharing e modifica il valore nella tabella richiesta_condivisione da "accettata" a "rifiutata"*).

Query utilizzate

IBaseDAO:

- **public T findById(int id);**
  SELECT *
  FROM tabella
  WHERE chiavePrimaria= "+id+";
- **public ArrayList<T> findAll();**
  SELECT *
  FROM tabella;

IAddettoDAO (IOperatoreDAO/ IAmministratoreDAO):

- **Findbyid;**
  SELECT A.utente_idutente, U.username, U.password, U.email
  FROM addetto AS A
  INNER JOIN utente as U ON U.idutente = A.utente_idutente
  WHERE A.utente_idutente = "+id+";
- **Addetto findUtenteByField(String name, String value);**
  SELECT addetto.utente_idutente, utente.username, utente.password, utente.email
  FROM addetto

INNER JOIN utente ON utente_idutente=utente.idutente
WHERE ("+name+"='"+value+"');

IMezzoDAO/ IModelloDAO:

- **ArrayList<Mezzo> findByFiltri(String a,String b,String c,String d,String e,String f,String g,String h,String i,String j);**
  SELECT mezzo.modello_idmodello, modello.categoria_idcategoria, modello.marca_idmarca, nome,carburante,num_posti, veicolo, cilindrata, tipo, prezzo
  FROM mezzo INNER JOIN (modello LEFT JOIN categoria ON modello.categoria_idcategoria=categoria.idcategoria NATURAL JOIN marca) ON mezzo.modello_idmodello=modello.idmodello
  WHERE veicolo IN ('"+d+"','"+e+"','"+f+"','"+g+"')
  AND tipo IN ('Fiat','Alfa Romeo','Bmw')
  AND cilindrata IN ('"+h+"','"+i+"','"+j+"') A
  ND carburante IN ('"+a+"','"+b+"','"+c+"')
  group by idmezzo";

IStazioneDAO:

- **findbyid**;
  SELECT S.idstazione, S.nome, S.latitudine, S.longitudine, S.addetto_utente_idutente, S.operatore_utente_idutente
  FROM stazione AS S INNER JOIN addetto AS A
  ON S.addetto_utente_idutente = A.utente_idutente
  INNER JOIN operatore AS O
  ON S.operatore_utente_idutente = O.utente_idutente
  WHERE S.idstazione = "+id+";
- **Stazione findStazioneByField(String name, String id);**
  SELECT *
  FROM stazione
  WHERE ("+name+"='"+id+"');

IMessaggiDAO:

- **ArrayList<Messaggio> findByField(String name, String value);**
  SELECT *
  FROM messaggio
  WHERE ("+nome+"='"+valore+"');
- **void addMessage(Messaggio mess);**
  INSERT INTO messaggio
  VALUES (NULL, "+mess.getUtente().getId()+", '"+mess.getOggetto()+"', '"+mess.getData()+"','"+mess.getStazione()+"');

IAccessorioDAO:

- **ArrayList<Integer> findByPrenotazione(int id);**
  SELECT accessorio_idaccessorio
  FROM contiene
  WHERE (prenotazione_idprenotazione='"+id+"');
- **ArrayList<Accessorio> findAccessorioByPrenotazione(int id);**

SELECT accessorio_idaccessorio
FROM contiene
WHERE (prenotazione_idprenotazione='"+id+"');

- **ArrayList<Accessorio> findAccessorio(ArrayList<Integer> a**);
  for (int i = 0; i <a.size() ; i++) ; {
  lista.add(findById(Integer.parseInt(a.get(i).toString())));}

IClienteDAO:

- **boolean addFoto(int id, InputStream foto);**
  UPDATE cliente
  SET foto = ?
  WHERE utente_idutente="+id+";
- **void salvaUtente(Cliente c, String pass);**
  INSERT INTO utente
  VALUES (NULL,'"+c.getEmail()+"','"+pass+"','"+c.getEmail()+"');
- **void salvaCliente(Cliente c);**
  INSERT INTO cliente(utente_idutente)
  SELECT last_insert_id() ;
  UPDATE cliente
  SET nome='"+c.getNome()+"', cognome='"+c.getCognome()+"',
  residenza='"+c.getResidenza()+"',data_nascita='"+strDataNascita+"',cellulare='"+c.getCellul
  are()+"'
  WHERE utente_idutente= last_insert_id()";
- **ArrayList<Cliente> prenotazioneCondivisa(int id);**
  SELECT cliente_idcliente
  FROM richiesta_condivisione
  WHERE (prenotazione_idprenotazione='"+id+"');

IPrenotazioneDAO:

- **ArrayList<Prenotazione> findPrenotazioneByModello(String valore);**
  SELECT idprenotazione
  FROM prenotazione AS p
  INNER JOIN mezzo AS M
  INNER JOIN modello AS o
  ON modello_idmodello = o.idmodello
  WHERE idmodello='"+valore+"' group by idprenotazione;
- **ArrayList<Prenotazione> findPrenotazioneByMarca(String valore);**
  SELECT idprenotazione
  FROM prenotazione as p
  join mezzo as m on p.mezzo_idmezzo=m.idmezzo
  join modello as mo on m.modello_idmodello=mo.idmodello
  left join marca as marc on mo.marca_idmarca=marc.idmarca
  WHERE idmarca='"+valore+"' group by idprenotazione;
- **ArrayList<Prenotazione> findPrenotazioneByData(Date d);**
  SELECT idprenotazione
  FROM prenotazione
  WHERE dataInizio='"+data+"';

- **ArrayList<Prenotazione> findSharing(int id);**
  SELECT prenotazione_idprenotazione
  FROM richiesta_condivisione
  WHERE (cliente_idcliente='"+id+"');
- **void salvaPrenotazione(Prenotazione p, ArrayList<Integer> a);**
  INSERT INTO prenotazione
  VALUES (NULL, '"+strDataPrenotazione+"', '"+p.getCliente().getId()+"',
  '"+p.getMezzo().getId()+"','"+p.getNumPostiOccupati()+"','"+p.getPartenza().getId()+"','"+p
  .getArrivo().getId()+"','"+p.getLocalita().getId()+"','"+strDataInizio+"','"+strDataFine+"','"+
  p.getPrezzoPagato()+"','No','In attesa')";
  for(int i=0;i<a.size();i++){
  INSERT INTO contiene
  VALUES (NULL,'" + p.getId() + "','" + a.get(i) + "');}
- **void salvaSharing(Prenotazione p, ArrayList<Integer> a, Utente u);**
  INSERT INTO richiesta_condivisione
  VALUES (NULL, '"+strDataPrenotazione+"', 'ACCETTATA', '"+p.getId()+"',
  '"+u.getId()+"');
  UPDATE prenotazione
  SET num_posti_occupati='"+p.getNumPostiOccupati()+"'
  WHERE idprenotazione='"+p.getId()+"';
- **void modificaPrenotazione(Prenotazione p, ArrayList<Integer> a);** 3 UPDATE
  prenotazione
  SET idprenotazione='"+p.getId()+"', data='"+strDataPrenotazione+"', cliente_idcliente=
  '"+p.getCliente().getId()+"',
  mezzo_idmezzo='"+p.getMezzo().getId()+"',num_posti_occupati='"+p.getNumPostiOccupat
  i()+"',idstazione_partenza='"+p.getPartenza().getId()+"',idstazione_arrivo='"+p.getArrivo().
  getId()+"',localita_idlocalita='"+p.getLocalita().getId()+"',dataInizio='"+strDataInizio+"',dat
  aFine='"+strDataFine+"',prezzoTotale='"+p.getPrezzoPagato()+"',pagamento='No',pronta='I
  n attesa'
  WHERE idprenotazione="+p.getId()+";
  DELETE FROM contiene
  WHERE prenotazione_idprenotazione='"+p.getId()+"';
  (Dentro un ciclo for)
  INSERT INTO contiene
  VALUES (NULL, '" + p.getId() + "','" + a.get(i) + "');
- **void modificaOperatore(Prenotazione p);**
  UPDATE prenotazione
  SET idprenotazione='"+p.getId()+"', data='"+strDataPrenotazione+"',cliente_idcliente=
  '"+p.getCliente().getId()+"',
  mezzo_idmezzo='"+p.getMezzo().getId()+"',num_posti_occupati='"+p.getNumPostiOccupat
  i()+"',idstazione_partenza='"+p.getPartenza().getId()+"',idstazione_arrivo='"+p.getArrivo().
  getId()+"',localita_idlocalita='"+p.getLocalita().getId()+"',dataInizio='"+strDataInizio+"',dat
  aFine='"+strDataFine+"',prezzoTotale='"+p.getPrezzoPagato()+"',pagamento='"+p.getPaga
  mento()+"',pronta='"+p.getPronta()+"'
  WHERE idprenotazione="+p.getId()+";
- **void delete(int id);**
  DELETE FROM prenotazione

WHERE idprenotazione="+id";

- **void deleteSharing(Prenotazione p, Utente u);**
  UPDATE richiesta_condivisione
  SET stato='RIFIUTATA'
  WHERE prenotazione_idprenotazione="+p.getId()+"
  AND cliente_idcliente="+u.getId()+";
- **findprenotazionebyfiltri(Prenotazione p);**
  SELECT idprenotazione
  FROM prenotazione
  WHERE dataInizio='"+strDataInizio+"
  AND dataFine='"+strDataFine+"'
  AND idstazione_partenza='"+p.getPartenza().getId()+"'
  AND idstazione_arrivo='"+p.getArrivo().getId()+"'
  AND localita_idlocalita='"+p.getLocalita().getId()+"'
  AND pagamento='No';

Conclusioni e sviluppi futuri

Con il presente lavoro è stata creata un' applicazione *stand-alone* per la gestione di un car sharing secondo le specifiche fornite dalla traccia.
Per la modellazione della base di dati è stata usata l' applicazione MySql Workbench.
Per la creazione dell' applicazione è stato utilizzato il linguaggio di programmazione Java tramite l'IDE IntelliJ IDEA.
Il comparto grafico è stato gestito utilizzando FXML (implentando la libreria JavaFX).
Nello sviluppo del progetto è stato utilizzato un modello di sviluppo agile, in particolare Scrum.
La progettazione concettuale è stata gestita utilizzando il linguaggio di modellazione UML per mezzo di un insieme di diagrammi (classi, sequenza, dipendenze).
In futuro si potrebbe scegliere di includere, per rendere più sicura l' applicazione,un metodo per la criptazione delle password.
Potremmo permettere all'amministratore di aggiungere mezzi, stazioni, località, accessori, marche, categorie, modelli che per il momento vengono inserite solo da noi.
Negli sviluppi futuri va anche l' aggiunta di recapiti più precisi riguardo alle stazioni, aggiungendo l' indirizzo e anche l' introduzione delle offerte magari offrendo codici sconto e chiedendo di inserirli in ogni prenotazione.