# Progetto Basi di dati

Angelo Sasso

4 Giugno 2018

Sommario

*Servizio di fornitura servizi di trasporto automobilistico privato (es. Uber)*

*INDICE*

# 1 Analisi dei requisiti

## 1.1 Specifiche

La piattaforma deve poter gestire le richieste di utenti consumatori (clienti) che desiderano prenotare delle corse erogate dai fornitori dei servizi (conducenti).

Quando un utente richiede un servizio di trasporto, viene inviata una notifica con tutte le informazioni necessarie (posizione di partenza, indirizzo di destinazione) al conducente più vicino. Il conducente può accettare la richiesta o rifiutarla.

Nel caso in cui venga rifiutata, la richiesta viene assegnata al driver più vicino e così via. Se il conducente accetta la richiesta, egli accede alla posizione del cliente per prelevarlo. Successivamente, sia il passeggero che il conducente hanno l'opzione reciproca di rilasciare un feedback.

Gli individui sono caratterizzati, oltre dai consueti dati anagrafici, dalla possibilità di poter associare un metodo di pagamento elettronico in alternativa al pagamento in contanti. Al momento di una ricerca di un conducente libero, il cliente specificherà eventuale budget massimo, numero di passeggeri o altre caratteristiche di interesse (animali, bagaglio etc.).

Il costo della corsa è quindi calcolato in base alla distanza percorsa, con eventuali sovrapprezzi che derivano dalle caratteristiche del mezzo impiegato e della corsa.

Per ogni conducente si vuole uno storico di tutte le richieste pervenute, con il dettaglio di accettazione o meno.

- possibilità di gestire meccanismi di scontistica automatizzata per frequenza (es. 5 corse in una settimana, la più breve viene scontata del 30%), kilometraggio (ogni 100km viene assegnato un punto, ciascuno dei quali corrisponde a un centro numero di km gratis) o altro
- i conducenti possono gestire più mezzi, non necessariamente con le stesse caratteristiche
- un cliente può prenotare corse ricorrenti (es. ogni venerdì alle 17.00 da una posizione di partenza a una di destinazione), eventualmente richiedendo sempre lo stesso autista

## 1.2 Glossario dei termini

Di seguito si trova un glossario dei termini:

| Termine | Descrizione | Sinonimo | Collegamenti |
|---|---|---|---|
| ***ospite*** | Coloro che vogliono ricercare passaggi<br>Aspettano la conferma dall'host. | ***viaggiatori*** | utenti<br>richieste<br>host |
| ***utenti*** | Possono essere host ( aggiungono e rimuovono<br>annunci) oppure ospiti (ricercano annunci). | | host<br>ospite |
| **richieste** | Contiene indirizzo di partenza e indirizzo<br>di destinazione. Può essere accettata o<br>declinata. | | host<br>ospite |
| ***host*** | *Coloro che mettono a disposizione degli ospiti*<br>le auto per i passaggi. In seguito possono<br>*dare la conferma o no della richiesta scelta* | | utenti<br>richieste<br>ospite |

## 1.3 Frasi

Di seguito verranno accoppiati i concetti associati alle varie entità manovrando le frasi della specifica per ogni entità.
UTENTI: Gli utenti possono registrarsi come host o come ospite per poter usufruire dei determinati servizi che offre la piattaforma.
HOST:possono offrire ospitalità aggiungendo confermando o rimuovendo annunci, condividere storie, opinioni e consigli online.Possono anche usare i Meetup (eventi) per trovare persone locali e viaggiatori simili a loro in tutto il mondo.
OSPITE:la sistemazione migliore per il loro viaggio può essere ricercata per data o per località . Il sistema restituirà un elenco di tutti gli appartamenti disponibili, correlati di immagini e recensioni. Oltre ad affittare e lasciare recensioni sull'esperienza vissuta possono anche usare i Meetup (eventi) per trovare persone locali e viaggiatori simili a loro in tutto mondo. ANNUNCI:Negli annunci devono essere descritti dettagliatamente gli alloggi(stanze private, interi appartamenti, castelli, ville, barche, baite, case sugli alberi, igloo, isole private etc.)includendo i servizi disponibili e gli orari di arrivo e partenza.

# 2 Operazioni

In questa sezione vengono riportate le interrogazioni e gli aggiornamenti che la base di dati deve soddisfare.

## 2.1 Interrogazioni

Di seguito vanno riportate le interrogazioni.
La base deve:
-Permettere all'host di consultare richieste di passaggi;
-Permettere all'host di ricevere l'email e il numero di telefono del cliente per valutare la richiesta.

## 2.2 Aggiornamenti

Di seguito vanno riportati gli aggiornamenti. La base deve:
-Poter registrare un utente di tipo host;
-Poter eliminare un utente di tipo host;
-Poter registrare un utente di tipo ospite;
-Poter eliminare un utente di tipo ospite;
-Poter accettare una richiesta; -Poter declinare una richiesta;

# 3 Progettazione concettuale

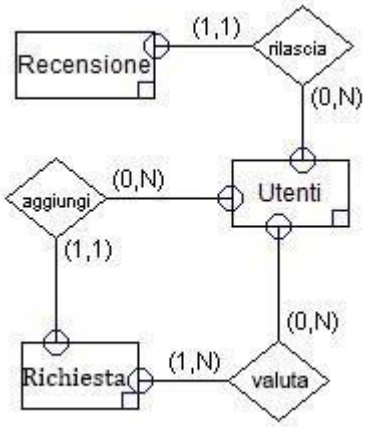
## 3.1 Scelta di progetto

Utilizzo di una strategia mista che cerca di combinare i vantaggi della strategia top-down con quelli della strategia bottom-up. Si suddividono i requisiti in componenti separate, come nella bottom-up, ma nello stesso tempo definisce uno schema scheletro contenente i concetti principali dell'applicazione.

## 3.2 Sviluppo dello schema E-R

Per prima cosa vengono individuati in uno schema scheletro i concetti più rilevanti. Poi verranno decomposti in vari sotto-schemi rappresentante ognuno le entità presenti nello scheletro.
Tramite un passo iterativo da ripetere per tutti i sotto-schemi vengono rappresentati e raffinati tutti i concetti precedentemente decomposti. Infine vengono integrati tutti i sotto-schemi prodotti in uno schema generale facendo riferimento allo scheletro.

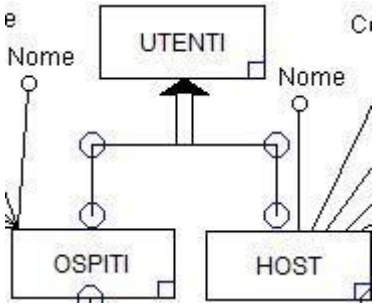
Si costruisce per prima cosa uno schema scheletro mostrante i concetti più rilevanti come in figura:




Decomponiamo e analizziamo adesso le varie entità:

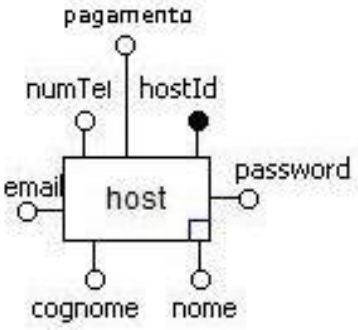
## 3.3 Utenti

L'entità utenti non è altro che l'entità padre delle entità Host e Ospiti.L'utente è colui che vuole utilizzare l'applicazione. Può essere utilizzata da un host o da un ospite dipendentemente dai servizi richiesti. Infatti :




## 3.4 host

L'entità host si è detta quindi classe figlia della classe utente. Una volta esser entrato come host infatti è possibile accettare o declinare richieste di passaggi.
I vari campi associati all' entità Host sono i seguenti:
-Nome(il nome di un host).
-Cognome(il cognome di un host).
-Email(l'email di un host).
-NumTelefono(il numero di telefono).
-HostId(una chiave identificativa dell'host).
-Password(password).
-Notifiche(notifiche).
-Storico(storico delle corse prese in carico o declinate).
In questo caso osserviamo che la chiave primaria dell'entità Host è hostId che rende univoca ogni occorrenza dell'entità.




## 3.5 Ospiti

L'entità ospiti si è detta anch'essa classe ereditata dalla classe utente. Una volta aver loggato come user è possibile ricercare per data o per località un annuncio

ed è data anche la possibilità di entrare in contatto con l'host dell'annuncio per poter affittare l'alloggio mostrato in quell'annuncio. I vari campi associati all'entità Ospiti sono i seguenti:

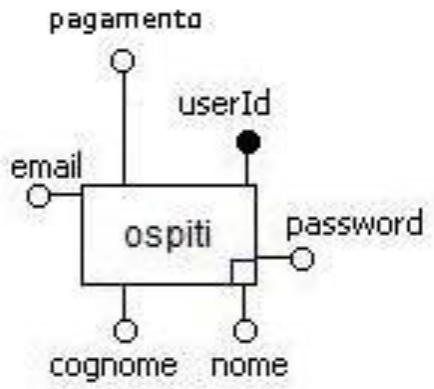
-Nome(nome di un utente di tipo ospite).

-Cognome(cognome di un ospite).

-Email(email di un ospite). -UserId(una chiave identificativa dell'ospite)(chiave primaria).

-Password(password).

-Pagamento(metodi di pagamento preferiti).




## 3.6 Richiesta

L'entità richiesta è un'entità il quale accesso è possibile sia da un utente di tipo host che da un utente di tipo ospite.

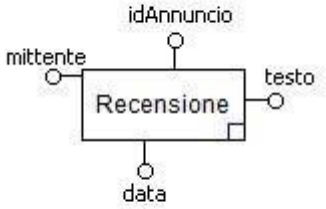
I primi possono accettare o declinare richieste di passaggi sfruttando questa classe, gli utenti invece possono creare richieste di passaggi da inoltrare.
Struttura:
-Id(questo id rende univoco ogni annuncio).
-Città(nome località).
-PosizionePartenza(indica la posizione dove il cliente vorrebbe essere prelevato
-IndirizzoDestinazione(destinazione del cliente).
-Descrizione(dichiarazioni del cliente per numero di persone ed eventuali animali).
-nomeCliente(ogni annuncio è collegato al cliente che lo crea tramite il clienteId.

-costo(costo calcolato in base ai criteri preposti).


## 3.7 Recensioni

Per implementare l'operazione di recensione di un annuncio da parte di un ospite è stata creata un'entità chiamata Recensioni:
Dovrà avere i seguenti attributi:

-mittente: l'id dell'ospite che scrive la recensione.
-idAnnuncio: l'id dell'annuncio da recensire.
-testo: il testo vero e proprio.
-data: quando viene pubblicata la recensione.




## 3.8 Relazioni

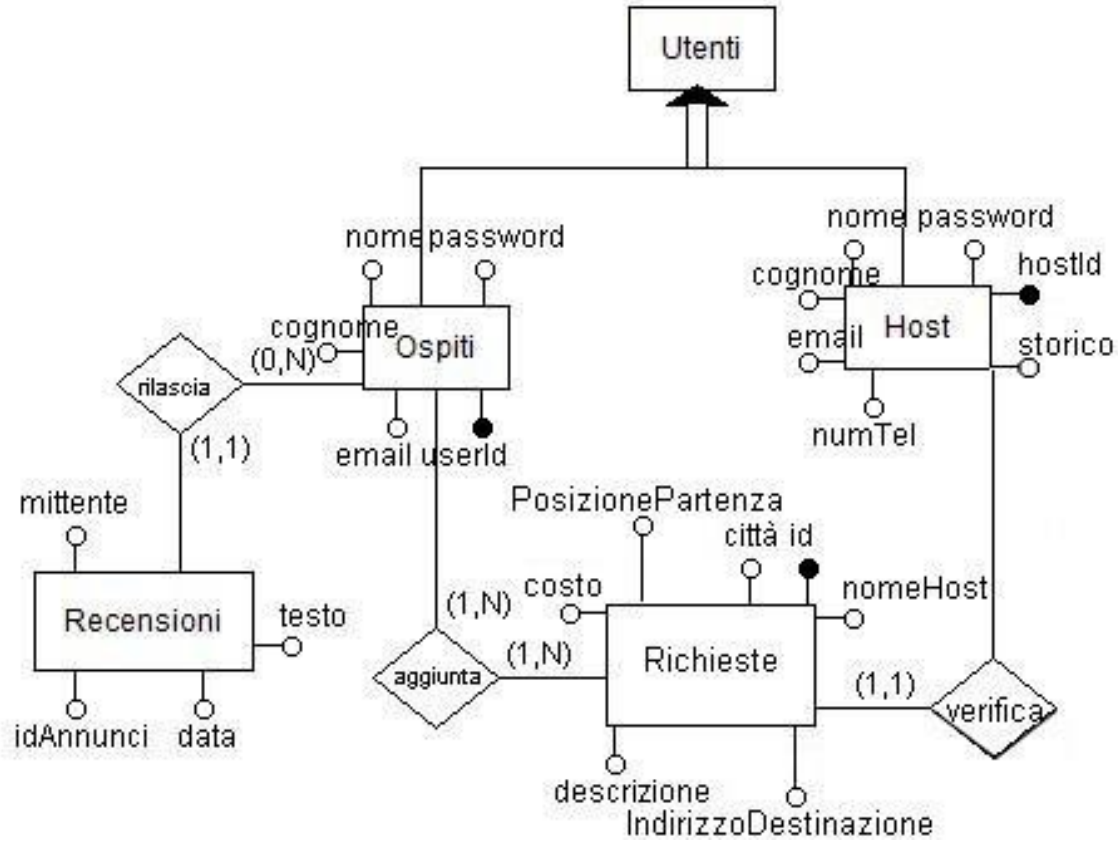
Analizziamo adesso le varie relazioni presenti:

La prima relazione che analizziamo è l'aggiunta di annuncio da parte di un host.

Infatti le entità coinvolte in questa relazione sono Host e Annuncio. La relazione (Host-Aggiunta) ha cardinalità (1,N) infatti ogni host può pubblicare non solo 1,ma anche N annunci.
La relazione (Annuncio-Aggiunta) ha cardinalità(1,1) perchè ad ogni annuncio aggiunto corrisponde solamente un host.
L'altra relazione che analizziamo è la ricerca di un ospite di un annuncio. (Ospite-Ricerca) ha cardinalità(1,N) infatti ogni ospite può ricercare anche N alloggi senza vincoli.
(Annuncio-Ricerca)ha cardinalità(1,N) perchè ogni annuncio può essere ricercato anche da N ospiti.
Le altre relazioni sono la rimozione di un annuncio da parte di un Host che ha cardinalità uguale alla relazione 'aggiunta' (1,N) e (1,1) e la relazione 'scrivi' associata alle entità host e Messaggi che permette all'host di scrivere nel forum. (Host-Scrivi) ha cardinalità (0,N) perchè la scrittura di un post nel forum è opzionale invece (Messaggi-Scrivi) ha cardinalità (1,1) perchè ogni messaggio nel forum è scritto da un solo host.
La relazione 'rilascia' associata alle entità ospiti e Recensioni permette di aggiungere una recensione ad un annuncio.
(Ospiti-rilascia) ha cardinalità (0,N) perchè il rilascio di una recensione è opzionale e (Recensione-rilascia) ha invece cardinalità (1,1) perchè una recensione viene rilasciata da un ospite.

## 3.9 Schema E-R finale

Schema E-R nella sua forma finale prima della progettazione logica si presenta così:


Utenti
nome password
cognome
Ospiti
(0,N)
rilascia
(1,1)
email userId
nome password
cognome
hostId
email
Host
storico
numTel
mittente
Recensioni
testo
idAnnunci
data
PosizionePartenza
città id
costo
nomeHost
(1,N)
aggiunta
(1,N)
Richieste
(1,1)
verifica
descrizione
IndirizzoDestinazione

| Entità | Descrizione | Attributi | Identificatore |
|---|---|---|---|
| Host | *Valuta le richieste e può accettare o rifiutare di offrire il passaggio col proprio mezzo* | Nome<br>Cognome<br>Email<br>numTel<br>hostId<br>Password<br>mezzo | hostId |
| Cliente | *Può ricercare annunci e lasciare recensioni* | Nome<br>Cognome<br>Email<br>userId<br>Password<br>metodoPagamento | userId |
| Richiesta | *Una richiesta di passaggio che viene pubblicata da un cliente e può essere accettata da un host* | id<br>città<br>PosizionePartenza<br>IndirizzoDestinazione<br>descrizione<br>costo<br>nomeHost | id |
| Recensioni | *Una recensione che può essere rilasciata dopo un passaggio* | mittente<br>idRichiesta<br>data<br>testo | |

Tabella delle entità.

| Relazione | Descrizione | Attributi | Componenti |
|---|---|---|---|
| Aggiunta | *Aggiunge una richiesta di passaggio* | | clienti<br>Richiesta |
| Rimozione | *Rimuove una richiesta di passaggio* | | clienti<br>Richiesta |
| Pagamento | *Rilascia la tariffa che l'host trattiene al cliente* | | clienti<br>host |
| Valuta | *Permette all'host di accettare o rifiutare una richiesta* | | host<br>Richiesta |
| Rilascia | *Permette di recensire un passaggio* | | ospite<br>Recensione |

Tabella delle relazioni.

# 4 Progettazione Logica

La progettazione logica della base di dati analizzata è suddivisa in 4 parti:
1)ottimizzazione;
2)semplificazione;
3) traduzione; 4)normalizzazione;

## 4.1 Ottimizzazione

La fase di ottimizzazione deve puntare innanzitutto sulla creazione della tavola dei volumi e della tavola delle operazioni.
Queste tavole aiutano ad approssimare l'indice di prestazione della base creata.

| Concetto | Tipo | Volume |
|---|---|---|
| Host | E | 500 |
| Ospiti | E | 1000 |
| Richiesta | E | 1500 |
| Recensioni | E | 500 |
| Aggiunta | R | 1500 |
| Pagamento | R | 100 |
| Rilascia | R | 500 |
| Rimozione | R | 100 |
| Valuta | R | 100 |

Tabella dei volumi.

| Operazione | Tipo | Frequenza |
|---|---|---|
| Registrazione host | I | 10 al giorno |
| Registrazione ospite | I | 100 al giorno |
| Aggiunta richiesta | I | 100 al giorno |
| Rimozione richiesta | I | 100 al giorno |
| Valuta richiesta | I | 1000 al giorno |
| Pagamento | I | 1000 al giorno |
| Lasciare una recensione | I | 100 al giorno |

Tabella delle operazioni.

## 4.2 Semplificazione: Eliminazione delle gerarchie

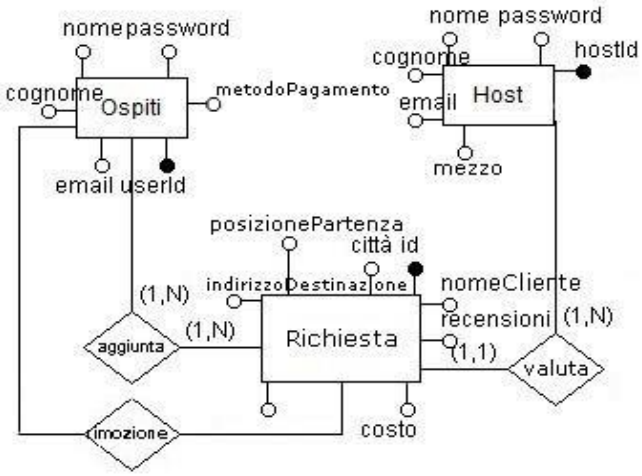
Dall'inizio della progettazione alla stesura dello schema E-R finale sono stati fatti molti miglioramenti per quanto riguarda la maggiore efficienza e la maggior snellezza della base e in generale della struttura dello schema E-R.

Tra i vari cambiamenti ci sono i seguenti:

1. L'eliminazione della generalizzazione che c'era tra la tabella utenti e le tabelle figlie 'host' e 'ospiti', infatti nello schema finale non avremo più traccia della tabella utenti, ma solamente delle tabelle figlie.
   Così facendo rimarranno solo le tabelle host e ospiti in quanto l'entità padre utenti non viene mai utilizzata non essendo messa in relazione con nessun' altra entità, al contrario delle entità figlie.

2. Un'altra scelta è stata quella di pensare 'recensione', non come una tabella a se stante e quindi come un'entità, ma come un concetto associato all'entità Richieste, trasformandolo in un attributo di quest'ultima. Così facendo abbiamo snellito lo schema finale, che adesso avrà una tabella in meno.

Di seguito vedremo la variazione grafica dello schema E-R.




## 4.3 Traduzione

Il nostro schema E-R viene adesso tradotto nel relativo schema relazionale:

-Host(nome,cognome,email,telefono,mezzo,hostId,password).

-Ospiti(nome,cognome,email,userId,telefono,password,pagamento). -

Richieste(id,città,posizionePartenza,indirizzoDestinazione,descrizione,costo,

recensione).

## 4.4 Normalizzazione

La normalizzazione è un processo che ha lo scopo finale di certificare la qualità dello schema del database.
Per far tutto ciò si esaminano le relazioni delle tabelle precedenti prendendo in esame una tabella per volta e si verifica il soddisfacimento delle seguenti forme normali, che garantiscono l'eliminazione di qualsiasi anomalia o incoerenze all'interno delle relazioni.

1. Gli attributi devono essere definiti su un dominio con valori atomici.

2. quando è in 1NF e per ogni relazione tutti gli attributi non chiave dipendono funzionalmente dall'intera chiave composta, cioè dall'insieme di chiavi minimale che identifica in modo univoco i record della tabella.

3. quando è in 2NF e ogni attributo non chiave dipende solamente dalla chiave e non da altri attributi non chiave.

4. Forma normale di Boyce e Codd (BCNF) se per ogni dipendenza funzionale $X->A$(relazioni tra attributi di una tabella),$X$ deve contenere una chiave della relazione.

Adesso verifichiamo che il nostro schema soddisfi queste 4 forme normali.
A quel punto lo schema sarà normalizzato.

Iniziando dalla tabella:

1. HOST(nome,cognome,email,numTel,hostId,password,mezzo);
2. OSPITI(nome,cognome,email,numTel,idCliente,password);
3. RICHIESTE(idRichiesta,città,posizionePartenza,indirizzoDestinazione,descrizione,costo, nomeCliente, recensione);
4. PAGAMENTO(idRichiesta,nomeCliente);

Si può vedere che tutti gli attributi delle restanti tabelle hanno un dominio di valori atomici ed è quindi soddisfatta la 1FN. In maniera al quanto banale è soddisfatta la 2FN visto che in ogni tabella restante la chiave minimale è unica.
La terza forma normale è soddisfatta perchè ogni attributo non chiave non dipende da altri attributi non chiave.
Infine per lo stesso motivo della 3NF è garantita anche la forma normale di Boyce e Codd.

# 5 MySql query

Di seguito verranno mostrate le query usate per la creazione delle tabelle:

1)TABELLA HOST:
CREATE TABLE
HOST(nome
char(20),cognome
char(20), email char(40),
numTel char(20), hostId
char(20) unique, mezzo
char(30), password
char(20))

2)TABELLA OSPITI:
CREATE TABLE
OSPITI(nome char(20),
cognome char(20), email
char(40), idCliente
char(20) unique, password
char(20))



3)TABELLA RICHIESTE:
CREATE TABLE RICHIESTE(id
int(11) unique, città char(20),
posizionePartenza date,
indirizzoArrivo date, descrizione
text, costo int(11), idCliente
char(20), recensione text)

4)TABELLA MESSAGGI:
CREATE TABLE PAGAMENTO(idPagamento
char(20) unique, idRichiesta char(20),
data timestamp, topic char(20))

Di seguito verranno mostrate le query che manipolando il data base permettono la creazione delle operazioni elencate nel capitoletto 1.4:
1)Creazione di un utente di tipo host.
INSERT INTO
HOST(nome,cognome,email,numTel,hostId,password,mezzo)values(”,”,”,”,”,”,”)

2)Creazione di un utente di tipo ospite.
INSERT INTO
CLIENTE(nome,cognome,email,userId,password,pagamento)values(”,”,”,”,”,”)
3)Per inserire una richiesta.
INSERT INTO
RIHIESTE(città,posizionePartenza,indirizzoDestinazione,descrizione,costo,idClient
e)values(”,”,”,”,”,”)

4)Per rimuovere una richiesta.

DELETE FROM ANNUNCI WHERE ID = ”

5)Per visualizzare lo storico delle richieste.
SELECT *FROM Richieste where posizionePartenza between “ and “

6) Per valutare la richiesta del cliente creatore del passaggio:
SELECT email,numTel FROM host INNER JOIN Richieste ON (host.hostId = Richieste.nomeHost AND Richieste.id = ”) WHERE nomeHost =”;
In questo modo l’ospite riceverà email e numero di telefono dell’host per po- ter in seguito contattarlo al fine di offrire il passaggio.

7) Per aggiungere un messaggio al forum:
INSERT INTO messaggi(mittente,testo,topic)values(”,”,”) 8) Per

mostrare tutti i viaggi dell’host: SELECT * FROM Richieste

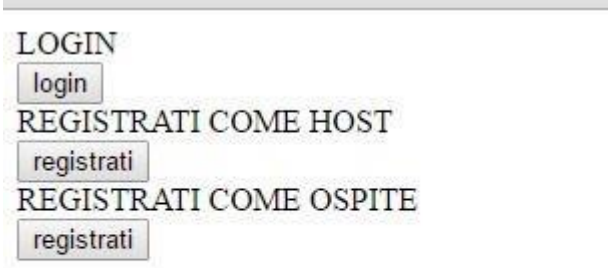
10)Per aggiungere una recensione:
UPDATE Richieste SET recensione=” WHERE id=”

# 6 Interfaccia Grafica

A supportare il data base ci pensa un’interfaccia grafica sviluppata con html e php in locale, usando ’phpMyAdmin’ e ’wamp’ e permette all’utente di svolgere le operazioni viste precedentemente in maniera semplificata.

## 6.1 Home

La home si presenterà come nella seguente figura.
L’utente potrà scegliere tra effettuare il login, registrarsi come host oppure registrarsi come ospite premendo i rispettivi ’submit’.




## 6.2 Registrazione Host

Se nella home l'utente sceglie di registrarsi come host gli apparirà un modulo da compilare al fine di registrarsi nel data base e poter quindi in seguito,nella schermata di login,accedere come host e sfruttare tutti i servizi che la piattaforma offre per questo tipo di utente.

## 6.3 Registrazione Ospite

Se nella home l'utente sceglie invece di registrarsi come ospite gli apparirà un modulo da compilare al fine di registrarsi nel data base e poter quindi in seguito,nella schermata di login,accedere come ospite e sfruttare tutti i servizi che la piattaforma offre per questo tipo di utente.


## 6.4 login

Se invece l'utente è già registrato o come host o come ospite può cliccare direttamente su login dove apparirà una finestra di login.

Attenzione un utente che è registrato nel database come host non può pensare di loggare come ospite e viceversa.

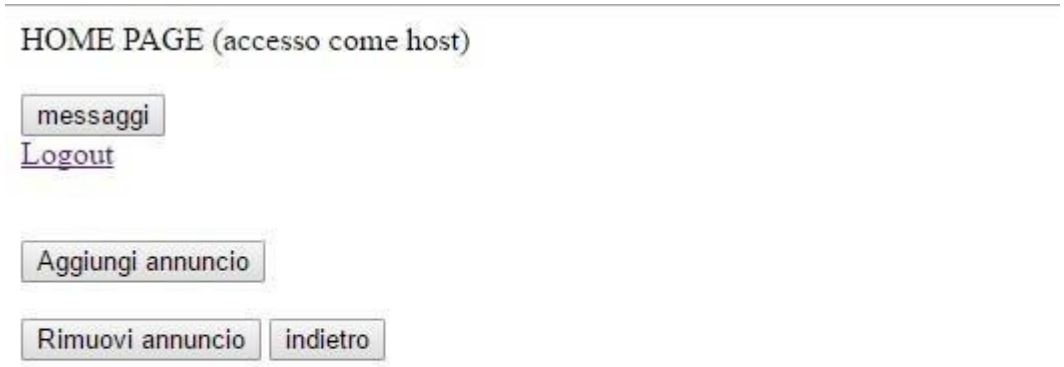
Proprio per questo sono stati introdotti dei gruppi:

se al momento del login il gruppo è settato a @localhost allora il file php associato alla pagina di login controllerà se l'username e la password inserite sono presenti nella tabella host del database.

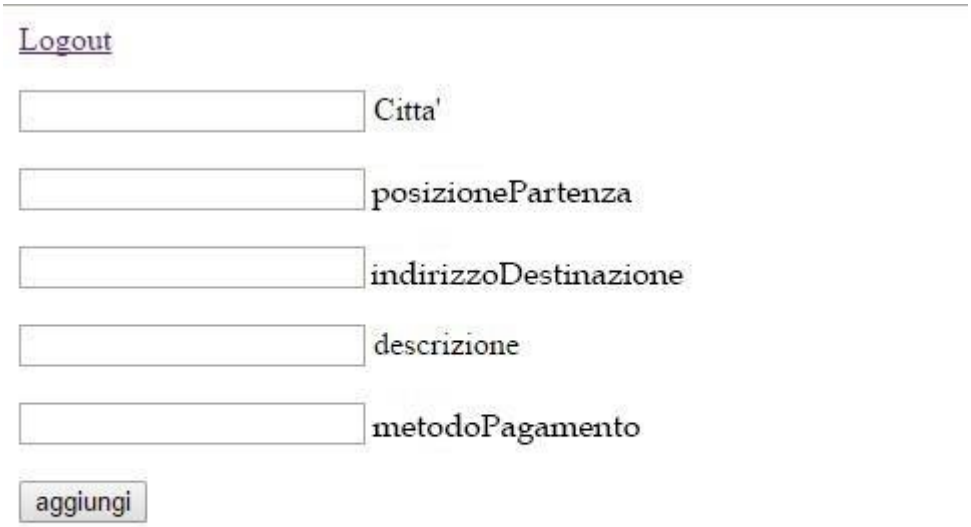
Vice versa se è settato a @localuser controllerà se l'username e la password inserite sono presenti nella tabella ospiti.


## 6.5 Host Page

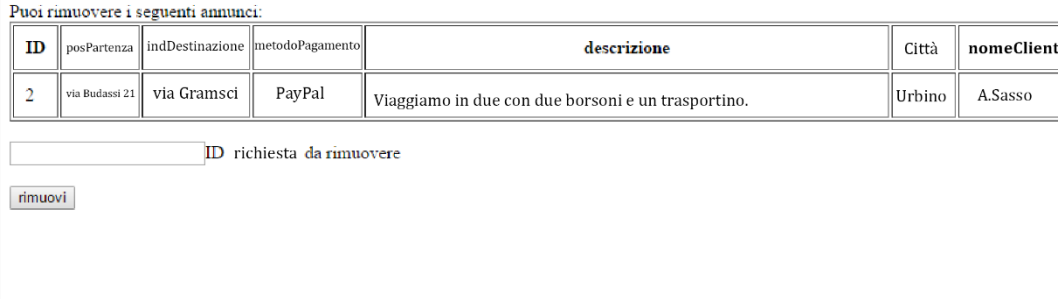
Se l'utente accede come host lo porterà nella seguente pagina.



Potrà scegliere tra:
-l'aggiunta di un annuncio.

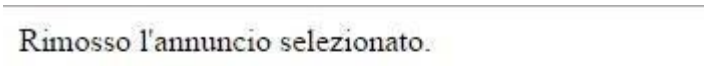



-la rimozione di un annuncio:




Ogni cliente ha la possibilità di rimuovere un annuncio precedentemente pubblicato da lui.
Premendo su rimuovi richiesta uscirà una tabella contenente gli annunci pubblicati come in figura.

Scrivendo nella casella di testo l'Id dell'annuncio da rimuovere e cliccando su rimuovi si rimuoverà l'annuncio selezionato.




-Infine sarà possibile rilasciare una recensione:

# Strumenti utilizzati

Per la realizzazione della relazione sono stati utilizzati i seguenti software:
Microsoft Word.
Java diagram er.
Per la realizzazione del progetto sono stati utilizzati i seguenti software:
Wampp (per la realizzazione del database).
Notepad ++(per la realizzazione dell'interfaccia grafica con i linguaggi php e html).