

# Elaborato per il corso di basi di dati

---

A.A 2021/2022  
Heravolution

Componenti:

Samuele De Tuglie samuele.detuglie@studio.unibo.it 0000989483

Cristina Zoccola cristina.zoccola@studio.unibo.it 0000969874

# Indice

---

- [Analisi dei requisiti](#)
  - [Intervista](#)
  - [Estrazione dei concetti principali](#)
- [Progettazione concettuale](#)
  - [Schema scheletro](#)
  - [Schema finale](#)
- [Progettazione logica](#)
  - [Stima del volume dei dati](#)
  - [Descrizione delle operazioni principali e stima delle loro frequenze](#)
  - [Schemi di navigazione e tabelle degli accessi](#)
    - [Operazioni principali](#)
    - [Operazioni di controllo](#)
  - [Raffinamento dello schema](#)
  - [Analisi delle ridondanze](#)
  - [Traduzione di entità e associazioni in relazioni](#)
  - [Schema relazione finale](#)
  - [Traduzione delle operazioni in query SQL](#)
    - [Query principali](#)
    - [Query di controllo](#)
- [Progettazione dell'applicazione](#)

# Analisi dei requisiti

---

## Intervista

Si vuole gestire una compagnia che tratta la vendita e la spedizione di prodotti per la raccolta differenziata e lo smaltimento dei rifiuti.

Comprende tre tipi di utenti: clienti, guidatori e magazzinieri.

I clienti sono gli utenti base esterni alla compagnia, che usufruiscono dei servizi offerti, quali la vendita e la spedizione di prodotti (sacchetti della spazzatura, contenitori per la raccolta differenziata) e la raccolta dei rifiuti a domicilio.

La raccolta dei rifiuti a domicilio sarà disponibile solo se prima sono stati acquistati i relativi sacchetti della compagnia.

I guidatori si occupano della consegna dei prodotti acquistati e di gestire la raccolta dei rifiuti a domicilio scegliendo anche l'apposita discarica in cui smaltirli, inoltre sono anche considerati dei clienti (quindi possono usufruire dei servizi legati a tale ruolo).

Ogni guidatore deve inserire le patenti che possiede, con la possibilità di aggiornarle in futuro, quindi per effettuare le proprie mansioni potrà scegliere solo tra i veicoli che gli è permesso guidare, stando però attento a cosa deve trasportare siccome ogni mezzo ha una capacità di trasporto massima.

I magazzinieri sono coloro che si occupano di rifornire i diversi magazzini con i prodotti da vendere, durante l'inserimento deve specificare tutte le caratteristiche dei prodotti, proprio come i guidatori anche loro sono considerati dei clienti.

Ogni magazziniere deve scegliere un magazzino in cui rifornire i prodotti, che inoltre può cambiare in qualsiasi momento.

## Estrazione dei concetti principali

Termine	Descrizione	Sinonimi
Cliente	utente esterno che interagisce con la compagnia	client
Guidatore	utente interno che si occupa delle consegne	driver
Magazziniere	utente interno che si occupa del magazzino	warehouse worker
Veicolo	mezzo per trasportare prodotti e spazzatura	vehicle
Patente	licenza per la guida	driver license
Ordine	effettuato dai clienti per acquistare prodotti e la raccolta dei rifiuti	order
Magazzino	struttura che contiene i prodotti da vendere/acquistare	warehouse
Prodotto	oggetto che può essere venduto/acquistato	product
Contenitore	oggetto che contiene i sacchetti della spazzatura da ritirare	container
Sacchetto	oggetto che contiene la spazzatura da ritirare	trashbag

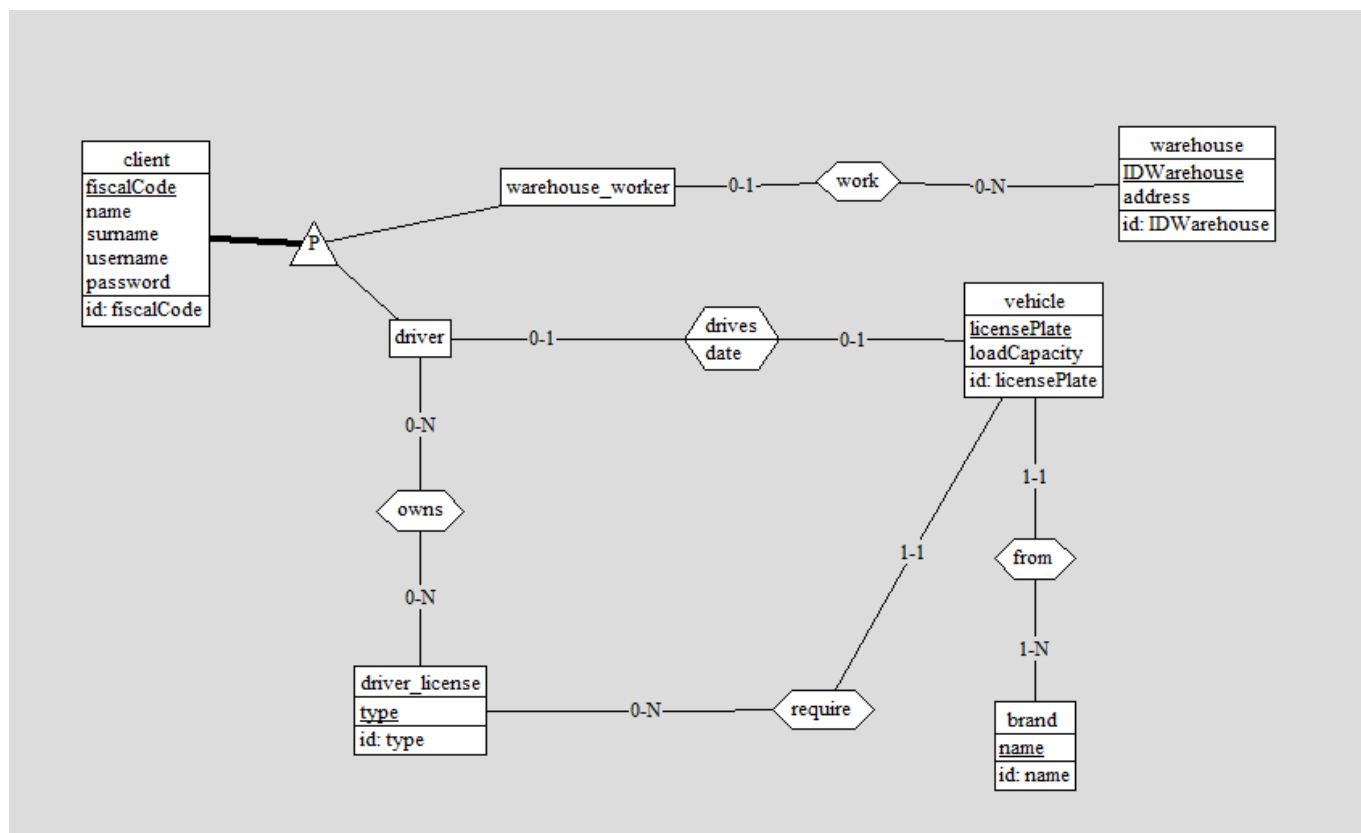
Ogni **cliente** può effettuare **ordini** di vari **prodotti**, ad ogni **guidatore** è associato un **veicolo** e ad ogni **magazziniere** un **magazzino**.

Lista delle principali operazioni effettuate:

- aggiungere un nuovo utente (cliente, guidatore o magazziniere)
- effettuare l'accesso degli utenti con i relativi permessi
- effettuare un nuovo ordine di prodotti
- richiedere la raccolta della spazzatura a domicilio
- mostrare la cronologia degli ordini effettuati
- assegnare un veicolo ad un guidatore
- inserire le patenti di un guidatore
- mostrare la cronologia dei veicoli assegnati ad un guidatore
- consegnare gli ordini effettuati
- raccogliere la spazzatura a domicilio e rilasciarla in una discarica specifica
- associare un magazzino ad un magazziniere
- aggiungere un nuovo prodotto al magazzino
- visualizzare l'inventario di tutti i magazzini

# Progettazione concettuale

## Schema scheletro

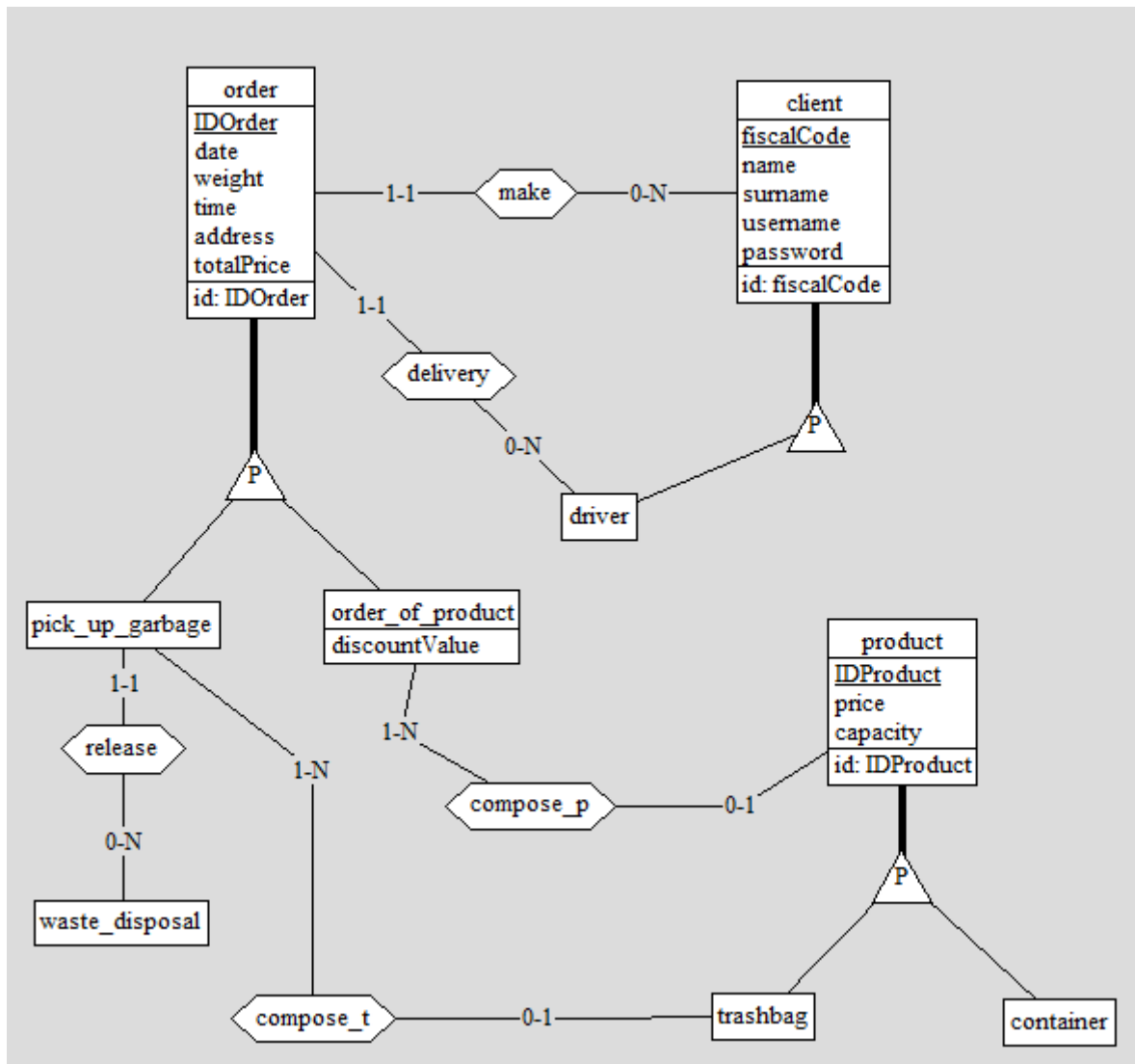


L'entità *driver* e l'entità *warehouse worker* sono la generalizzazione dell'entità *client*.

Sono identificati per *fiscal code* e vengono utilizzati l'*username* e la *password* per accedere all'applicazione.

Ogni *driver* possiede una o più *driver license*.

Ogni *vehicle* è identificato dalla *license plate* e può essere guidato da un solo *driver* alla volta che deve possedere la *driver license* richiesta, inoltre ad ogni veicolo è associato un *brand*; tramite il campo *date* nell'associazione *drives* ci è possibile tenere traccia dei veicoli guidati da ogni guidatore durante quella giornata.



L'entità *order* ha associato il *client* che l'ha effettuato e il *driver* che lo ha consegnato.

Esistono due tipi di ordini: *pick up garbage* e *order of product* che sono la generalizzazione dell'entità *order*.

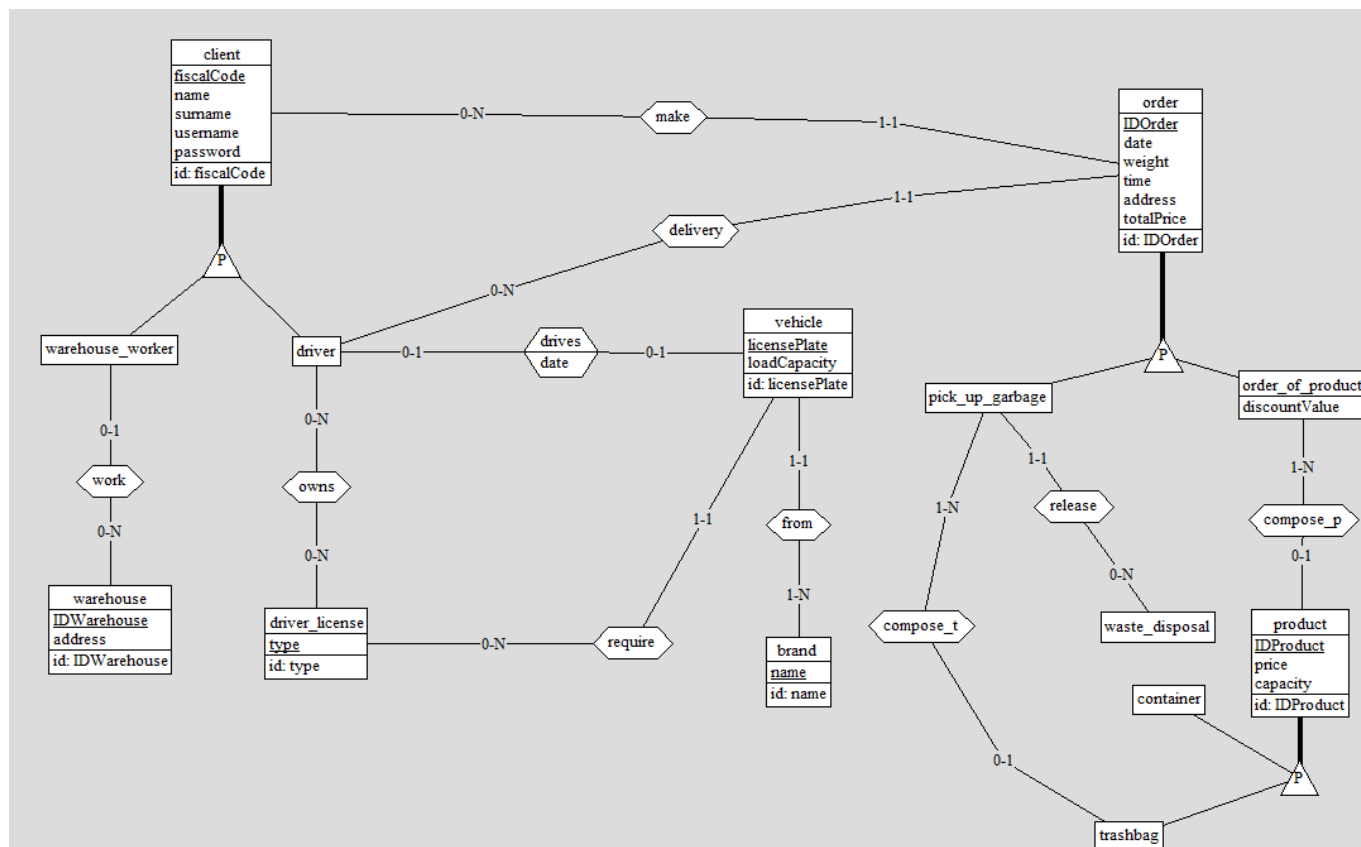
Ogni *pick up garbage* è poi rilasciato in una *waste disposal*.

Ad ogni *order of product* sono associati i prodotti acquistati, mentre ad un *pick up garbage* sono associati i *trashbag* da ritirare.

Ad ogni *product* quando viene acquistato viene associato l'*order of product* e/o il *pick up garbage* corrispondente.

Dopo una certa quantità di ordini il cliente ha uno sconto, memorizzato nel campo *discount* della tabella *order of product*.

## Schema finale



# Progettazione logica

## Stima del volume dei dati

Concetto	Costrutto	Volume
client	E	2.004.000
driver	E	2.000
warehouse worker	E	2.000
warehouse	E	40
driver license	E	5
vehicle	E	1200
brand	E	20
drives	R	500.000
order	E	2.000.000
pick up garbage	E	1.100.000
order of product	E	1.200.000
product	E	5.000.000
trashbag	E	3.700.000
container	E	1.300.000
waste disposal	E	20

Ci aspettiamo di avere un numero elevato e sempre in aumento di *order* (*pick up garbage*, *order of product*), *product* (*trashbag*, *container*) e *drives*, rispettivamente perchè gli *order* e i *drives* servono per gli storici, mentre i *product* una volta essere stati venduti non vengono eliminati dal database per poter risalire a quali prodotti sono stati acquistati nei singoli ordini ed inoltre per poter scegliere quali *trashbag* possono essere ritirate.



## Descrizione delle operazioni principali e stima delle loro frequenze

Codice	Operazione	Frequenza
1	aggiungere un nuovo cliente	80 al giorno
2	aggiungere un nuovo magazziniere	30 all'anno
3	aggiungere un nuovo guidatore	30 all'anno
4	effettuare l'accesso degli utenti	150.000 al giorno
5	effettuare un nuovo ordine di prodotti	10.000 al giorno
6	richiedere la raccolta della spazzatura	5.000 al giorno
7	mostrare la cronologia degli ordini effettuati	3.000 al giorno
8	assegnare un veicolo ad un guidatore	8.000 al giorno
9	inserire le patenti di un guidatore	70 all'anno
10	mostrare la cronologia dei veicoli assegnati	100 all'anno
11	consegnare gli ordini effettuati	5.500 al giorno
12	raccogliere la spazzatura e rilasciarla	2.500 al giorno
13	associare un magazzino ad un magazziniere	45 all'anno
14	aggiungere un nuovo prodotto al magazzino	25.000 al giorno
15	visualizzare l'inventario di tutti i magazzini	2 alla settimana

## Schemi di navigazione e tabelle degli accessi

### *Operazioni principali*

Di seguito sono riportate le tabelle degli accessi per ogni operazione sopra riportata:

#### - Operazione 1: aggiungere un nuovo cliente

Concetto	Costrutto	Accessi	Tipo
client	E	2	L
client	E	1	S

totale: 2L + 1S → 80 al giorno

**- Operazione 2: aggiungere un nuovo magazziniere**

Concetto	Costrutto	Accessi	Tipo
client	E	2	L
client	E	1	S
warehouse worker	E	1	S

totale: 2L + 2S → 30 all'anno

**- Operazione 3: aggiungere un nuovo guidatore**

Concetto	Costrutto	Accessi	Tipo
client	E	2	L
client	E	1	S
driver	E	1	S

totale: 2L + 2S → 30 all'anno

**- Operazione 4: effettuare l'accesso degli utenti**

Concetto	Costrutto	Accessi	Tipo
client	E	1	L

totale: 1L → 150.000 al giorno

**- Operazione 5: effettuare un nuovo ordine di prodotti**

Concetto	Costrutto	Accessi	Tipo
product	E	3	L
order of product	E	1	L
order of product	E	1	S
product	E	1	S

totale: 4L + 2S → 10.000 al giorno

**- Operazione 6: richiedere la raccolta della spazzatura**

Concetto	Costrutto	Accessi	Tipo
product	E	2	L
trashbag	E	1	L
order of product	E	1	L
pick up garbage	E	1	S
trashbag	E	1	S

totale: 4L + 2S → 5.000 al giorno

**- Operazione 7: mostrare la cronologia degli ordini effettuati**

Concetto	Costrutto	Accessi	Tipo
order of product	E	1	L
pick up garbage	E	1	L
waste disposal	E	1	L

totale: 3L → 3.000 al giorno

**- Operazione 8: assegnare un veicolo ad un guidatore**

Concetto	Costrutto	Accessi	Tipo
driver	E	3	L
vehicle	E	1	L
owns	R	1	L
driver	E	1	S
drives	R	1	S

totale: 5L + 2S → 8.000 al giorno

**- Operazione 9: inserire le patenti di un guidatore**

Concetto	Costrutto	Accessi	Tipo
owns	R	2	L
driver license	E	1	L
owns	R	1	S

totale: 3L + 1S → 70 all'anno

**- Operazione 10: mostrare la cronologia dei veicoli assegnati**

Concetto	Costrutto	Accessi	Tipo
drives	R	1	L
vehicle	E	1	L

totale: 2L → 100 all'anno

**- Operazione 11: consegnare gli ordini effettuati**

Concetto	Costrutto	Accessi	Tipo
driver	E	1	L
vehicle	E	1	L
order of product	E	2	L
order of product	E	1	S
driver	E	1	S

totale: 4L + 2S → 5.500 al giorno

**- Operazione 12: raccogliere la spazzatura e rilasciarla**

Concetto	Costrutto	Accessi	Tipo
driver	E	1	L
vehicle	E	1	L
pick up garbage	E	2	L
waste disposal	E	1	L
pick up garbage	E	1	S
driver	E	1	S

totale: 5L + 2S → 2.500 al giorno

**- Operazione 13: associare un magazzino ad un magazziniere**

Concetto	Costrutto	Accessi	Tipo
warehouse worker	E	2	L
warehouse	E	1	L
warehouse worker	E	1	S

totale: 3L + 1S → 45 all'anno

**- Operazione 14: aggiungere un nuovo prodotto al magazzino**

Concetto	Costrutto	Accessi	Tipo
warehouse worker	E	1	L
garbage	E	1	L
product	E	1	S
trashbag/container	E	1	S

L'ultima operazione cambia in base a che tipo di prodotto si sta aggiungendo

totale: 2L + 2S → 25.000 al giorno

**- Operazione 15: visualizzare l'inventario di tutti i magazzini**

Concetto	Costrutto	Accessi	Tipo
warehouse	E	1	L
product	E	1	L

totale: 2L → 2 alla settimana

*Operazioni di controllo*

Le seguenti operazioni vengono eseguite ogni volta che un guidatore vuole rispondere ad un ordine.

**- Controllo se c'è una patente associata al guidatore**

Concetto	Costrutto	Accessi	Tipo
owns	R	1	L

totale: 1L

**- Controllo se il veicolo è assegnato al guidatore**

Concetto	Costrutto	Accessi	Tipo
driver	E	1	L

totale: 1L

## Raffinamento dello schema

### - Eliminazione delle gerarchie

È stata eliminata la gerarchia *order* tramite il collasso verso il basso, replicando gli attributi in *order\_of\_product* e *pick\_up\_garbage*.

Per le gerarchie *product* e *client* si è scelto di utilizzare un metodo ibrido, utilizzando il collasso verso l'alto, ma mantenendo le entità figlie in quanto alcuni campi proprietari all'entità sono stati mantenuti in esse (ad esempio il campo *licensePlate* di *driver* non poteva essere inserito in *client*, oppure il campo *IDOrderGarbage* di *trashbag* non poteva essere messo in *product*).

### - Trasformazione di relazioni in entità

È stata trasformata la relazione *owns* in un'entità che rappresenta la relazione molti a molti tra *driver* e *driver\_license*; è stata trasformata anche la relazione *drives* che ci permette di mantenere lo storico dei veicoli assegnati ai guidatori.

## Analisi delle ridondanze

La prima ridondanza che abbiamo deciso di inserire è il campo *userType* in *client*, mantenendo le tabelle *driver* e *warehouse\_worker*.

Si può vedere come in fase di accesso all'applicazione questo renda l'operazione meno costosa:

Concetto	Costrutto	Accessi	Tipo
client	E	1	L

totale: 1L → 150.000 al giorno

Senza ridondanza sarebbe necessario controllare ogni entità in base a quanti tipi di utenti esistono.

Concetto	Costrutto	Accessi	Tipo
client	E	1	L
driver	E	1	L
warehouse worker	E	1	L

totale: 3L → 450.000 al giorno

La seconda ridondanza è presente nell'entità *product*, che contiene il campo *productType* (stesso caso della ridondanza sopra citata).

## Traduzione di entità e associazioni in relazioni

### Traduzione tabelle

---

client (fiscalCode, name, surname, username, password, userType)

---

driver (fiscalCode : client, licensePlate\* : vehicle)

---

drives (IDDrives, date, fiscalCode : client, licensePlate : vehicle)

---

driver\_license (type)

---

owns (IDOwns, type : driver\_license, fiscalCode:driver)

---

vehicle (licensePlate, loadCapacity, brandName : brand, driverLicense : driver\_license)

---

brand (name)

---

warehouse\_worker (fiscalCode : client, IDWarehouse\* : warehouse)

---

warehouse (IDWarehouse, address)

---

product (IDProduct, price, productType, capacity, garbageType : garbage, IDOrder\* : order\_of\_product, IDWarehouse : warehouse)

---

trashbag (IDProduct : product, IDOrderGarbage\* : pick\_up\_garbage)

---

container (IDProduct : product)

---

garbage (type)

---

pick\_up\_garbage (IDOrderGarbage, date, time, address, totalPrice, weight, licensePlate\* : vehicle, fiscalCode : client, IDWasteDisposal\* : waste\_disposal)

---

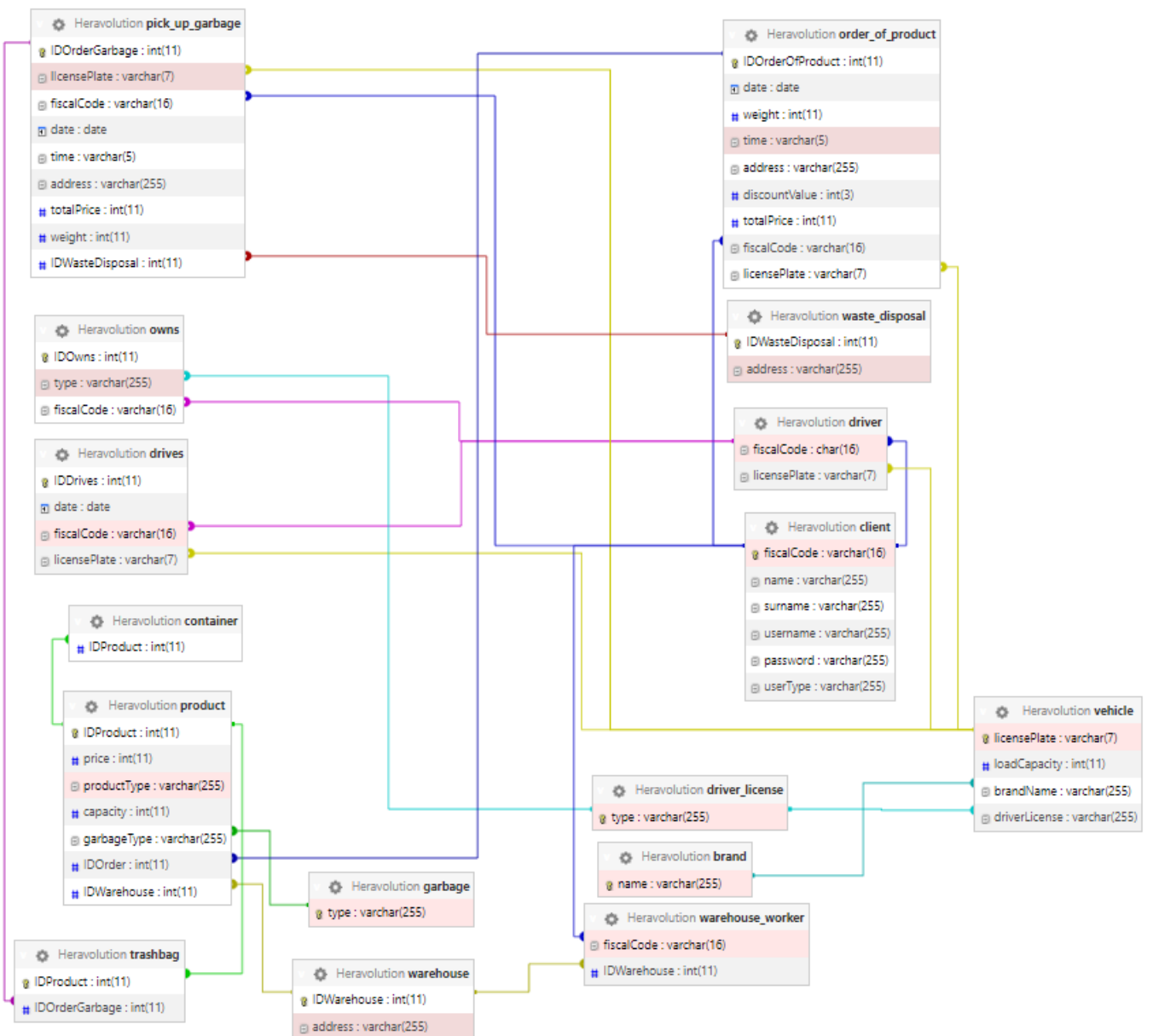
order\_of\_product (IDOrderOfProduct, date, weight, time, address, discountValue\*, totalPrice, fiscalCode : client, licensePlate\* : vehicle)

---

waste\_disposal (IDWasteDisposal, address)



## Schema relazione finale



# Traduzione delle operazioni in query SQL

## Query principali

Di seguito nelle operazioni effettuate i seguenti valori indicano:

`$_POST[]` : è l'input dell'utente all'interno delle corrispettive form

`$_SESSION[]` : è il modo di mantenere in memoria i dati dell'utente che ha effettuato l'accesso

`$last_id`: è la variabile che contiene l'ultimo *ID* inserito

`get_array()` : è una funzione che permette di ottenere l'array contenente i valori richiesti

`implode()` : permette di spezzare un array per ottenere i valori richiesti

### - Operazione 1: aggiungere un nuovo cliente

Query per controllare se esiste già un cliente con lo stesso codice fiscale:

```
SELECT *
FROM client
WHERE fiscalCode = '".$_POST["fiscalCode"].''
```

Query per controllare se esiste già un cliente con lo stesso username:

```
SELECT *
FROM client
WHERE username = '".$_POST["username"].''
```

Query per inserire il cliente:

```
INSERT INTO client(name, surname, fiscalCode, username, password, userType)
VALUES (?, ?, ?, ?, ?, ?)
```

### - Operazione 2: aggiungere un nuovo magazziniere

Si svolge come l'operazione 1 ma con l'aggiunta della query per inserire il magazziniere:

```
INSERT INTO warehouse_worker(fiscalCode)
VALUES (?)
```

### - Operazione 3: aggiungere un nuovo guidatore

Si svolge come l'operazione 1 ma con l'aggiunta della query per inserire il guidatore:

```
INSERT INTO driver(fiscalCode)
VALUES (?)
```

### - Operazione 4: effettuare l'accesso degli utenti

```
SELECT fiscalCode, userType, password
FROM client
WHERE username = '$_POST["username"]'
LIMIT 1
```

### - Operazione 5: effettuare un nuovo ordine di prodotti

Query per visualizzare i prodotti disponibili (per essere acquistabili devono aver il campo *IDOrder* nullo):

```
SELECT *
FROM product
WHERE IDOrder IS NULL
ORDER BY productType
```

Query per ottenere l'eventuale sconto:

```
SELECT COUNT(*) AS counter
FROM order_of_product
WHERE fiscalCode = '$_SESSION['fiscalCode']'
```

Query per ottenere il peso totale dell'ordine:

```
SELECT SUM(capacity) AS weight
FROM product
WHERE IDProduct IN (".get_array($product).")
```

Query per ottenere il prezzo totale dell'ordine:

```
SELECT SUM(price) AS totalPrice
FROM product
WHERE IDProduct IN (".get_array($_POST["product"]).")
```

Query per inserire l'ordine:

```
INSERT INTO order_of_product(date, time, address, discountValue, totalPrice,
fiscalCode, weight)
VALUES (?, ?, ?, ?, ?, ?, ?)
```

Query per inserire l'/ID dell'ordine ai prodotti:

```
UPDATE product
SET IDOrder = ".$last_id."
WHERE IDProduct IN (".get_array($_POST["product"]).")
```

#### - Operazione 6: richiedere la raccolta della spazzatura

Query per visualizzare i sacchetti che possono essere raccolti (devono essere stati acquistati da quel cliente e devono avere il campo *IDOrderGarbage* nullo):

```
SELECT product.* FROM product, trashbag
WHERE product.productType = 'trashbag'
AND trashbag.IDOrderGarbage IS NULL
AND product.IDOrder = ANY
( SELECT IDOrderOfProduct FROM order_of_product
  WHERE fiscalcode = ".$_SESSION['fiscalCode'].
  AND licensePlate IS NOT NULL)
AND product.IDProduct = trashbag.IDProduct
```

Query per ottenere il peso totale dell'ordine:

```
SELECT SUM(capacity) AS weight
FROM product
WHERE IDProduct IN (".get_array($product).")
```

Query per inserire l'ordine:

```
INSERT INTO pick_up_garbage(fiscalCode, date, time, address, totalPrice, weight)
VALUES (?, ?, ?, ?, ?, ?)
```

Query per inserire l'ID dell'ordine ai prodotti:

```
UPDATE trashbag
SET IDOrderGarbage = ".$last_id."
WHERE IDProduct IN (".get_array($_POST["garbage"]).")
```

### - Operazione 7: mostrare la cronologia degli ordini effettuati

Query per visualizzare gli ordini di *order of product*:

```
SELECT date, address, discountValue, totalPrice, weight, licensePlate
FROM order_of_product
WHERE fiscalCode = '".$_SESSION["fiscalCode"]."'
ORDER BY IDOrderOfProduct
```

Query per visualizzare gli ordini di *pick up garbage*:

```
SELECT licensePlate, date, pick_up_garbage.address AS a, totalPrice, weight,
waste_disposal.address AS b
FROM pick_up_garbage, waste_disposal
WHERE fiscalCode = '".$_SESSION["fiscalCode"]."'
    AND (pick_up_garbage.IDWasteDisposal = waste_disposal.IDWasteDisposal
    OR pick_up_garbage.IDWasteDisposal IS NULL)
GROUP BY IDOrderGarbage
ORDER BY IDOrderGarbage
```

### - Operazione 8: assegnare un veicolo ad un guidatore

Query per controllare se l'utente ha già un veicolo associato:

```
SELECT licensePlate
FROM driver
WHERE fiscalCode = '".$_SESSION["fiscalCode"]."'
```

Query per mostrare i veicoli disponibili (per esserlo non devono essere associati a nessun altro guidatore e il guidatore deve avere la patente corrispondente):

```
SELECT vehicle.*
FROM vehicle, driver
WHERE driverLicense IN
  (SELECT type
   FROM owns
   WHERE fiscalCode = '$_SESSION["fiscalCode"]')
AND vehicle.licensePlate NOT IN (
  SELECT licensePlate
  FROM driver
  WHERE licensePlate IS NOT NULL)
GROUP BY vehicle.licensePlate
```

Query per inserire il veicolo:

```
UPDATE driver
SET licensePlate = '$_POST['vehicle']'
WHERE fiscalCode = '$_SESSION["fiscalCode"]'
```

Query per inserire la corsa del guidatore sul veicolo:

```
INSERT INTO drives(fiscalCode, licensePlate, date)
VALUES(?, ?, ?)
```

## - Operazione 9: inserire le patenti di un guidatore

Query per controllare se l'utente ha già almeno una patente associata:

```
SELECT IDOwns
FROM owns
WHERE fiscalCode = '$_SESSION["fiscalCode"]'
```

Query per mostrare le patenti inseribili (quelle diverse da quelle già associate):

```
SELECT type
FROM driver_license
WHERE type NOT IN (
    SELECT type
    FROM owns
    WHERE fiscalCode = '".$_SESSION["fiscalCode"]."')
```

Query per inserire la patente:

*per permetterci di effettuare una sola scrittura nel database la query è una stringa a cui vengono concatenati tutti i valori da inserire, l'inizio della query è quello che segue*

```
INSERT INTO owns (type, fiscalCode)
VALUES (?, ?)
```

#### - Operazione 10: mostrare la cronologia dei veicoli assegnati

```
SELECT date, drives.licensePlate, loadCapacity, driverLicense, brandName
FROM drives, vehicle
WHERE fiscalCode = '".$_SESSION["fiscalCode"]."'
    AND drives.licensePlate = vehicle.licensePlate
GROUP BY date, licensePlate
ORDER BY IDDrives
```

#### - Operazione 11: consegnare gli ordini effettuati

Query per ottenere la *licensePlate* del veicolo associato al guidatore e la sua *loadCapacity*:

```
SELECT driver.licensePlate, loadCapacity
FROM driver, vehicle
WHERE driver.fiscalCode = '".$_SESSION["fiscalCode"]."'
    AND driver.licensePlate = vehicle.licensePlate
LIMIT 1
```

Query per mostrare gli ordini da consegnare (per esserlo devono avere il campo *licensePlate* nullo):

```
SELECT IDOrderOfProduct, address, weight
FROM order_of_product
WHERE licensePlate IS NULL
```

Query per ottenere il peso totale di tutti gli ordini selezionati:

```
SELECT SUM(weight) as totalWeight
FROM ".$table."
WHERE ".$id." IN (".implode(",", $product).")
LIMIT 1
```

(\$table è la variabile che contiene il nome della tabella in cui cercare l'ordine, \$id è la variabile che contiene l'id dell'ordine che ci interessa, \$product è la variabile che contiene l'array degli ordini)

Query per consegnare gli ordini:

```
UPDATE order_of_product
SET licensePlate = ".$licensePlate."
WHERE IDOrderOfProduct IN (".implode(",", $_POST["order"]).")
```

Query per togliere il veicolo associato al guidatore:

```
UPDATE driver
SET licensePlate = NULL
WHERE fiscalCode = ".$_SESSION["fiscalCode"]."
```

## - Operazione 12: raccogliere la spazzatura e rilasciarla

Query per ottenere la *licensePlate* del veicolo associato al guidatore e la sua *loadCapacity*:

```
SELECT driver.licensePlate, loadCapacity
FROM driver, vehicle
WHERE driver.fiscalCode = ".$_SESSION["fiscalCode"]."
    AND driver.licensePlate = vehicle.licensePlate
LIMIT 1
```

Query per mostrare i sacchetti da ritirare (per esserlo devono avere il campo *licensePlate* nullo):

```
SELECT *
FROM pick_up_garbage
WHERE licensePlate IS NULL
```



Query per mostrare le discariche disponibili:

```
SELECT *  
FROM waste_disposal
```

Query per ottenere il peso totale di tutti gli ordini selezionati:

```
SELECT SUM(weight) AS totalWeight  
FROM ".$table."  
WHERE ".$id." IN (".implode(",", $product).")  
LIMIT 1
```

Query per raccogliere la spazzatura:

```
UPDATE pick_up_garbage  
SET licensePlate = ".$licensePlate.",  
    IDWasteDisposal = ".$_POST["disposal"]."  
WHERE IDOrderGarbage IN (".implode(",", $_POST["garbage"]).")
```

Query per togliere il veicolo associato al guidatore:

```
UPDATE driver  
SET licensePlate = NULL  
WHERE fiscalCode = ".$_SESSION["fiscalCode"]."
```

### - Operazione 13: associare un magazzino ad un magazziniere

Query per controllare se il magazziniere ha già un magazzino associato:

```
SELECT IDWarehouse  
FROM warehouse_worker  
WHERE fiscalCode = ".$_SESSION["fiscalCode"]."  
LIMIT 1
```

Query per mostrare i magazzini disponibili (tutti tranne quello già associato):

```
SELECT warehouse.IDWarehouse, address FROM warehouse, warehouse_worker
WHERE (warehouse.IDWarehouse != warehouse_worker.IDWarehouse
      OR warehouse_worker.IDWarehouse IS NULL)
AND fiscalCode = '$_SESSION["fiscalCode"]'
```

Query per associare il magazzino:

```
UPDATE warehouse_worker
SET IDWarehouse = '$_POST["warehouse"]'
WHERE fiscalCode = '$_SESSION["fiscalCode"]'
```

#### - Operazione 14: aggiungere un nuovo prodotto al magazzino

Query per ottenere l'ID del magazzino associato:

```
SELECT IDWarehouse
FROM warehouse_worker
WHERE fiscalCode = '$_SESSION["fiscalCode"]'
```

Query per mostrare i tipi di spazzatura:

```
SELECT type
FROM garbage
```

Query per aggiungere il prodotto:

*anche in questo caso per poter effettuare una sola scrittura la query è una stringa con concatenati tutti i valori da aggiungere, la query inizia con quanto segue*

```
INSERT INTO product (price, productType, capacity, garbageType, IDWarehouse)
VALUES
```

Query per aggiungere il prodotto al magazzino:

*query realizzata allo stesso modo della precedente, ma in questo caso la query inizia in questo modo*

```
INSERT INTO ".$_POST["productType"]."(IDProduct)
VALUES
```

(*productType* è il nome della tabella a cui aggiungere i prodotti)

#### - Operazione 15: visualizzare l'inventario di tutti i magazzini

```
SELECT address, productType, capacity, price, garbageType
FROM warehouse, product
WHERE IDOrder IS NULL AND product.IDWarehouse = warehouse.IDWarehouse
ORDER BY warehouse.IDWarehouse
```

*Query di controllo*

#### - Controllo se c'è una patente associata al guidatore

```
SELECT IDOwns
FROM owns
WHERE fiscalCode = '".$_SESSION["fiscalCode"].''
```

#### - Controllo se il veicolo è assegnato al guidatore

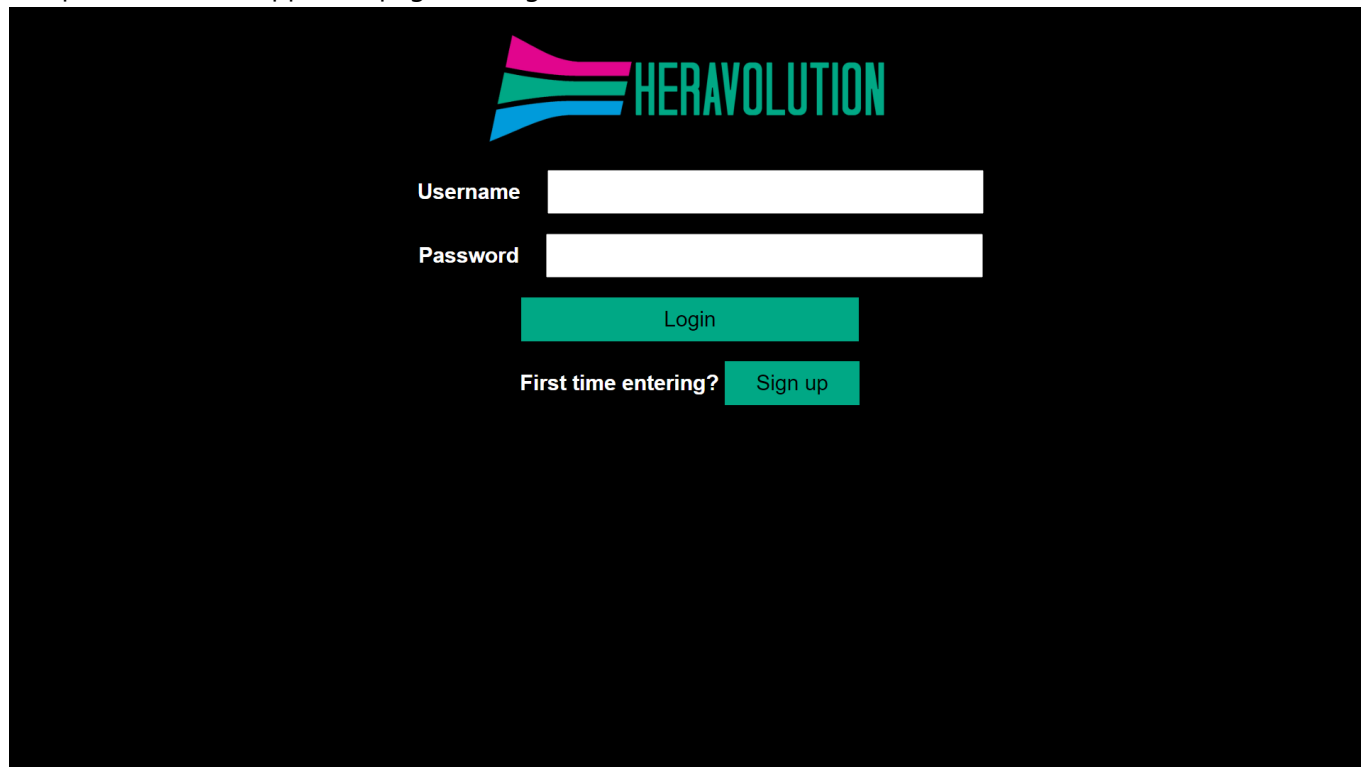
```
SELECT licensePlate
FROM driver
WHERE fiscalCode = '".$_SESSION["fiscalCode"].''
```

# Progettazione dell'applicazione

---

Per interfacciarsi al database si è optato per realizzare un sito web (fatto con html, css, php e javascript); il server apache per reindirizzare il sito risiede in remoto insieme al database mySQL.

All'apertura del sito appare la pagina di login :



The login page features the HERAVOLUTION logo at the top, which consists of three horizontal bars in red, green, and blue, followed by the word 'HERAVOLUTION' in green. Below the logo are two white input fields for 'Username' and 'Password'. A red 'Login' button is positioned below the password field. At the bottom, there is a link 'First time entering?' followed by a red 'Sign up' button.

Una volta entrati si viene reindirizzati nella pagina del cliente:



The client page is titled 'Client page' in red. It has two red buttons: 'Logout' and 'History of orders'. Below these is a section titled 'Order of products' in red. This section contains an 'Address:' label followed by a white input field. Below the address field is a dropdown menu with a red background and white text, showing a list of items: 'container | price: 1 | capacity: 2', 'container | price: 1 | capacity: 2', 'container | price: 1 | capacity: 2', 'container | price: 1 | capacity: 2', and 'trashbag | price: 2 | capacity: 5'. A red 'Submit' button is located below the dropdown. At the bottom of the page, there is a section titled 'Pick up garbage' in red, followed by the text 'No garbage to pick up'.

Che in base ai propri permessi (se si è guidatori o magazzinieri) permette di accedere alle relative pagine:

- Guidatore:

### Driver Home

License plate: AA111BB | Load capacity: 25

[Logout](#)[Client page](#)[Add license](#)[Change vehicle](#)[History of driven vehicles](#)

#### Deliver order

Address: Via Umberto Saba 589 | Weight: 11

Submit

#### Pick up garbage

No garbage to pick up

- Magazziniere:

### Warehouse worker home

[Logout](#)[Client page](#)[Change Warehouse](#)[Warehouse inventory](#)

#### Add products

Container ▼

Battery ▼

Quantity:

Capacity:

Price:

submit

Con i relativi bottoni nelle pagine sopra citate è possibile accedere alle altre funzionalità dell'applicazione. Ai fini di evitare possibili *SQL injection* prima di inserire valori all'interno del database ogni valore viene opportunamente controllato, inoltre l'applicazione stessa controlla la correttezza dei dati inseriti.