

Elaborato di Basi di Dati – Prof. Vincenzo Moscato



Realizzato da:
Arnese Matteo
Barbato Emanuele
Castaldo Luigi Pio
Cecchini Lorenzo

Università degli Studi di Napoli Federico II
Facoltà di Ingegneria Informatica
Anno Accademico 2023/2024



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



Indice

- **Introduzione**
 - Chi siamo
 - Problematica da affrontare
- **Specifiche**
 - Specifiche sui dati
 - Specifiche sulle operazioni
- **Progettazione Concettuale**
 - Glossario dei termini
 - Schema E/R portante
 - Schema E/R completo
- **Progettazione Logica**
 - Fase di trasformazione
 - Fase di Traduzione
- **Progettazione Fisica**
 - Creazione delle Tabelle e Sequenze
 - Aggiunta delle chiavi esterne e politiche di reazione
 - Indici
 - Dimensionamento
- **Gestione della sicurezza**
- **Gestione della concorrenza**
- **Gestione dell'affidabilità**
- **Stored Procedure e Trigger**
 - Stored Procedure
 - Trigger
- **Query e viste**
- **Progettazione dell'applicazione**
 - Interfaccia Grafica
 - Implementazione del backend
 - PHP e query effettuabili tramite interfaccia
 - Gestione della sicurezza
- **Popolamento della base dati**



Introduzione

Chi siamo

Le nostre soluzioni ti consentono di viaggiare in tutta Italia in completa serenità. Dici addio alle file interminabili ai caselli autostradali, e dedica più tempo a ciò che è veramente importante.

Problematica da affrontare

Nell'era dell'innovazione tecnologica, la gestione efficiente dei pagamenti autostradali è diventata cruciale per semplificare l'esperienza di viaggio degli utenti e ottimizzare la fluidità del traffico. La nostra piattaforma informatica per il telepedaggio si propone di rivoluzionare questo processo, offrendo un sistema sicuro, rapido e intuitivo per gestire i pagamenti autostradali. Attraverso una combinazione di avanzate tecnologie di backend, sicurezza informatica robusta e un'interfaccia utente intuitiva, la nostra soluzione mira a migliorare l'efficienza dei pagamenti, consentendo agli utenti di attraversare le autostrade senza interruzioni e semplificando il monitoraggio delle transazioni. Unendo forze con istituzioni finanziarie e gestori autostradali, ci impegniamo a offrire un servizio affidabile e conforme alle normative vigenti, contribuendo a plasmare un futuro di viaggi autostradali più fluidi e convenienti.

Specifiche

Specifiche sui dati

Il richiedente vuole che si gestiscano le seguenti informazioni:

- Per i **dispositivi**: codice;
- Per i **clienti**: dati anagrafici e le informazioni di fatturazione;
- Per i **tragitti**: numero del tragitto, specificando il collegamento tra tragitto e casello;
- Per i **caselli**: codice e un numero;
- Per le **automobili**: targa, modello e il proprietario;

Specifiche sulle operazioni

Le principali operazioni previste sulla base di dati sono:

1. registrazione di un nuovo utente (frequenza: 100 volte al giorno);
2. inserimento di un nuovo tragitto (frequenza: 500 volte al giorno);
3. registrazione di un determinato metodo di pagamento (frequenza: 300 volte al giorno);
4. visualizzazione da parte degli utenti di tutte le principali informazioni relative ai propri dispositivi e relativi tragitti (frequenza: 500 volte al giorno);
5. visualizzazione da parte dell'admin di tutti i tragitti (frequenza: 10 volte al giorno)

È inoltre richiesto che (regola di business):

6. all'atto della cancellazione di un utente, le informazioni sui dispositivi passati ad esso relativi, devono essere eliminate dalla base di dati transazionale e conservate in apposite tabelle contenenti solo dati storici.



Progettazione concettuale

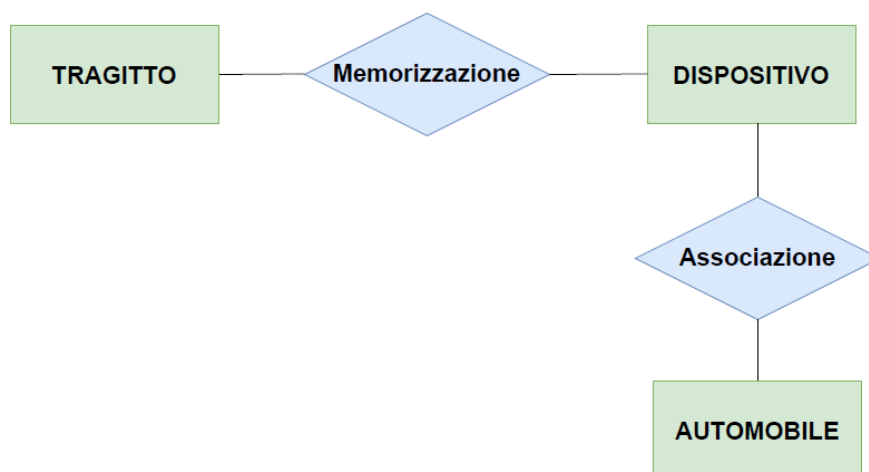
Glossario dei termini

Per avere una visione più schematica dei requisiti richiesti, è stato opportuno suddividere le informazioni in categorie e quindi elaborare il glossario dei termini usati:

Termine	Descrizione	Termini collegati
Dispositivo	Utilizzato per semplificare radicalmente il processo di pagamento, consentendo ai clienti di attraversare agevolmente le autostrade senza la necessità di contanti o fermarsi ai caselli.	Automobile, Tragitto
Automobile	Relativa al cliente, a cui è associato il dispositivo	Dispositivo, Cliente
Cliente	Persona fisica che registra un tragitto sulla piattaforma	Automobile
Tragitto	Ai fini della fatturazione al cliente	Dispositivo, Casello
Casello	Utilizzato per tenere traccia dell'effettivo inizio e fine tragitto compiuto dal cliente.	Tragitto

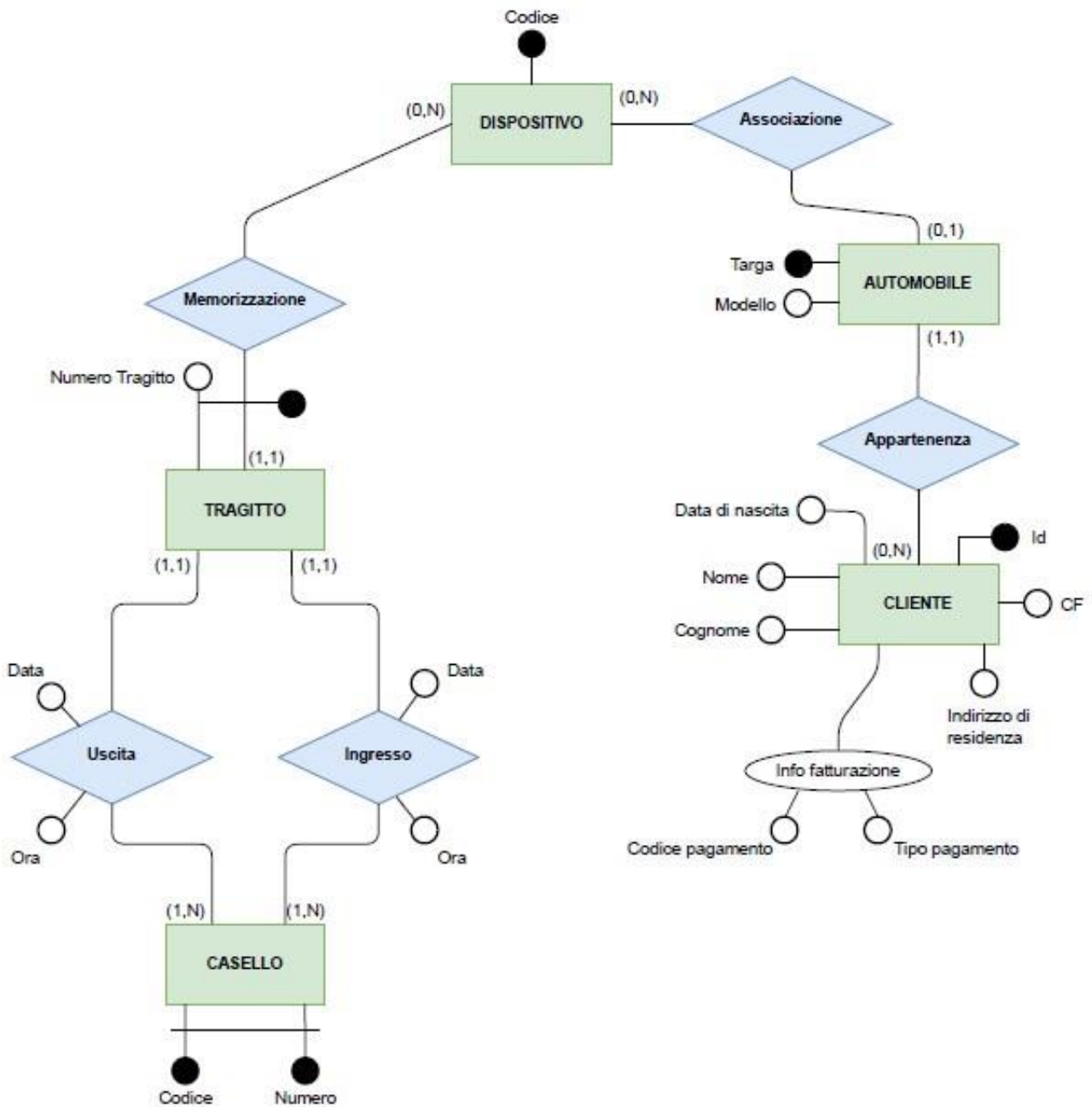
Schema E/R portante

Estraiamo dalle specifiche il **modello dello schema portante** con i concetti fondamentali del problema, e successivamente operando su di esso si effettua una fase di raffinamento, nella quale si inseriscono gli attributi relativi alle singole entità. Di seguito, si è voluto concentrare il tutto su quelle che sono le entità fondamentali alla base del concetto di memorizzazione del tragitto su un determinato dispositivo e su quale automobile è avvenuto lo spostamento:



Schema E/R completo

Di seguito è raffigurato lo schema E/R completo, comprendente tutti gli elementi e le relazioni alla base del progetto realizzato:

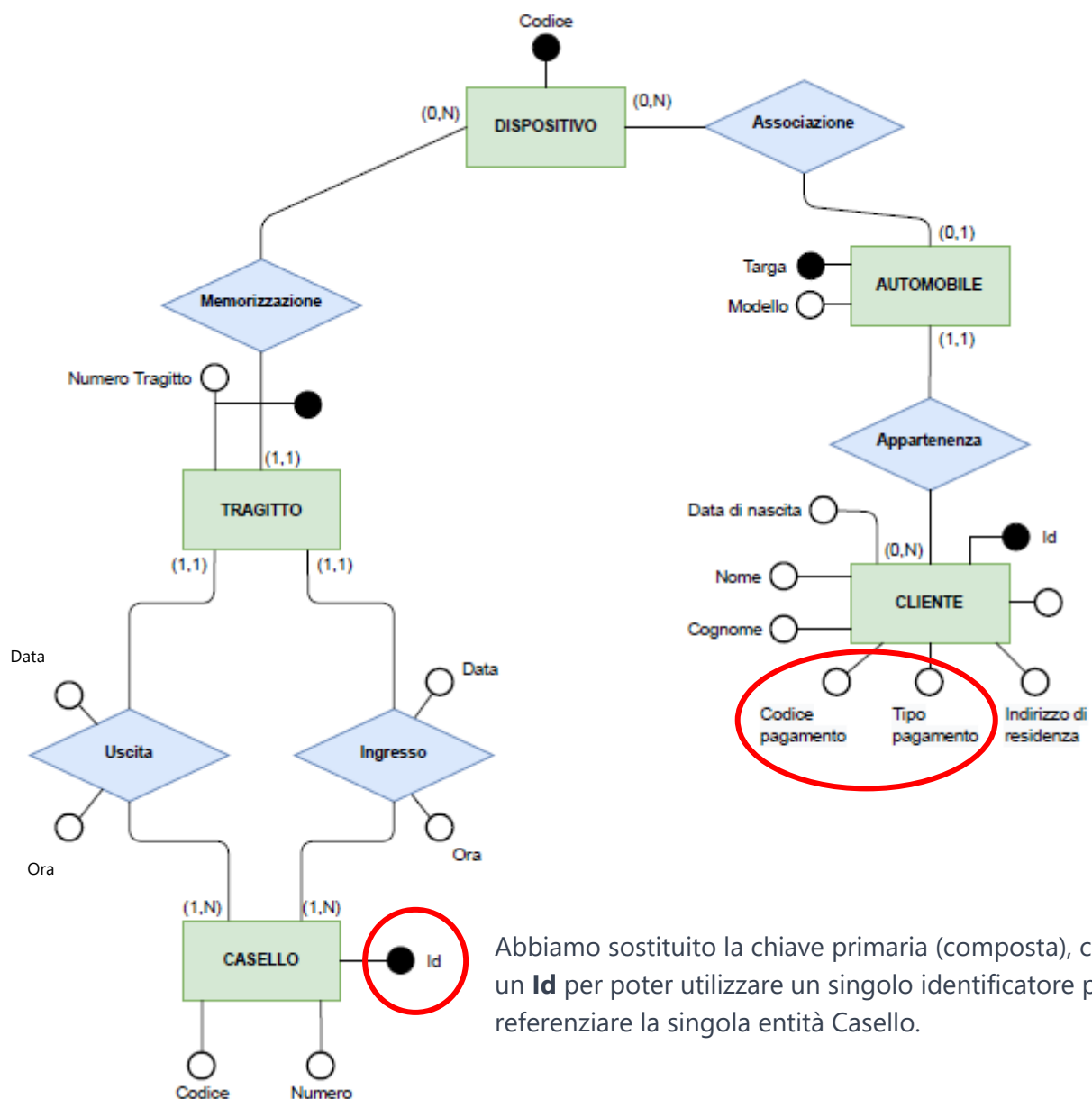


Progettazione logica

Fase di trasformazione

L'obiettivo di questa fase è quello di semplificare alcuni costrutti del modello ER, come ad esempio gli attributi composti e multivalore, non direttamente traducibili nel modello logico.

Nel nostro caso disponiamo di un solo attributo composto → **INFO FATTURAZIONE**, gestibile sostituendolo con tanti attributi semplici quanti sono gli attributi componenti → **Codice Pagamento e Tipo Pagamento**.



Fase di traduzione

Per la fase di traduzione:

- Una entità dello schema concettuale si traduce in una relazione dello schema logico avente lo stesso nome (ma al plurale) e gli stessi attributi dell'entità ed avente per chiave primaria il suo identificatore. Quindi vengono trasformate tutte le entità, in particolare otteniamo le relazioni:

DISPOSITIVI (Codice)

AUTOMOBILI (Targa, Modello)

CLIENTI (Id, Nome, Cognome, Data_Nascita, Codice_Fiscale, Indirizzo_Residenza, Codice_Pagamento, Tipo_Pagamento)

TRAGITTI (Num_Tragitto, Dispositivo: DISPOSITIVI);

CASELLI (Id, Codice, Numero)

Tra Dispositivo/Automobile, sussiste “**ASSOCIAZIONE**”, su cui è stato possibile adottare la regola 3, ottenendo una nuova relazione → **ASSOCIAZIONI_DISP_AUTO** (Automobile: **AUTOMOBILI**, Dispositivo: **DISPOSITIVI**);

Tra Cliente/Automobile, sussiste “**APPARTENENZA**”, su cui è stato possibile adottare la regola 3, ottenendo una nuova relazione → **APPARTENENZE_AUTO** (Automobile: **AUTOMOBILI**, Cliente: **CLIENTI**);

Tra Dispositivo/Tragitto, sussiste “**MEMORIZZAZIONE**”, su cui è stato possibile adottare la regola 4, secondo la quale si sarebbero dovuti aggiungere alla relazione **TRAGITTI**, gli attributi di **DISPOSITIVO**; tuttavia, non essendovene, poiché **TRAGITTI** era già provvista di chiave esterna referenziante **DISPOSITIVI**, l'associazione scompare.

Tra Casello/Tragitto, sussistono due associazioni “**INGRESSO**” e “**USCITA**”, gestite utilizzando la regola 4, aggiungendo alla relazione **TRAGITTI**, gli attributi di **CASELLO** → **TRAGITTI** (Num_Tragitto, Dispositivo: DISPOSITIVI, Casello Ingresso: CASELLI, Casello_Uscita: CASELLI, Data_Ingresso, Ora_Ingresso, Data_Uscita, Ora_Uscita);

Il risultato è il seguente modello logico-relazionale:

DISPOSITIVI (Codice)

AUTOMOBILI (Targa, Modello)

CLIENTI (Id, Nome, Cognome, Data_Nascita, Codice_Fiscale, Indirizzo_Residenza, Codice_Pagamento, Tipo_Pagamento)

TRAGITTI (Num_Tragitto, Dispositivo: DISPOSITIVI, Casello Ingresso: CASELLI, Casello_Uscita: CASELLI, Data_Ingresso, Ora_Ingresso, Data_Uscita, Ora_Uscita)

CASELLI (Id, Codice, Numero)

ASSOCIAZIONI_DISP_AUTO (Automobile: AUTOMOBILI, Dispositivo: DISPOSITIVI)

APPARTENENZE_AUTO (Automobile: AUTOMOBILI, Cliente: CLIENTI)



Progettazione fisica

Creazione delle tabelle e sequenze

```
CREATE TABLE DISPOSITIVI (  
    Codice NUMBER(6),  
  
    CONSTRAINT PK_DISPOSITIVI PRIMARY KEY(Codice)  
);  
  
CREATE TABLE AUTOMOBILI (  
    Targa VARCHAR2(7),  
    Modello VARCHAR2(20) NOT NULL,  
  
    CONSTRAINT PK_AUTOMOBILI PRIMARY KEY (Targa)  
);  
  
CREATE SEQUENCE clienti_id_seq START WITH 1;  
  
CREATE TABLE CLIENTI (  
    Id NUMBER(6) DEFAULT clienti_id_seq.nextval NOT NULL,  
    Nome VARCHAR2(20) NOT NULL,  
    Cognome VARCHAR2(20) NOT NULL,  
    DataNascita DATE NOT NULL,  
    CodiceFiscale VARCHAR2(16) NOT NULL,  
    Indirizzo VARCHAR(40) NOT NULL,  
    CodicePagamento VARCHAR2(30) DEFAULT NULL,  
    TipoPagamento VARCHAR2(5) DEFAULT NULL,  
  
    CONSTRAINT PK_CLIENTI PRIMARY KEY (Id),  
    CONSTRAINT CK_PAGAMENTO CHECK (TipoPagamento IS NULL OR TipoPagamento = 'Conto' OR TipoPagamento =  
'Carta'),  
    CONSTRAINT CK_CODICEFISCALE_UNIQUE UNIQUE (CodiceFiscale)  
);  
  
CREATE SEQUENCE caselli_id_seq START WITH 1;  
  
CREATE TABLE CASELLI (  
    Id NUMBER(4) DEFAULT caselli_id_seq.nextval,  
    Codice VARCHAR2(20) NOT NULL,  
    Numero NUMBER(2) NOT NULL,  
  
    CONSTRAINT PK_CASELLI PRIMARY KEY (Id)  
);
```




```

CREATE TABLE TRAGITTI (
    NumTragitto NUMBER(5) NOT NULL,
    Dispositivo NUMBER(6),
    Casello_Ingresso NUMBER(4) NOT NULL,
    Casello_Uscita NUMBER(4) NOT NULL,
    DataOraIngresso DATE NOT NULL,
    DataOraUscita DATE NOT NULL,

    CONSTRAINT PK_TRAGITTI PRIMARY KEY (NumTragitto, Dispositivo),
    CONSTRAINT FK_TRAGITTI_DISPOSITIVI FOREIGN KEY (Dispositivo) REFERENCES DISPOSITIVI(Codice),
    CONSTRAINT FK_TRAGITTI_CASELLO_1 FOREIGN KEY (Casello_Ingresso) REFERENCES CASELLI(Id),
    CONSTRAINT FK_TRAGITTI_CASELLO_2 FOREIGN KEY (Casello_Uscita) REFERENCES CASELLI(Id)
);

CREATE TABLE ASSOCIAZIONI_DISP_AUTO (
    Dispositivo NUMBER(6),
    Automobile VARCHAR2(7),

    CONSTRAINT PK_ASSOCIAZIONI_DISP_AUTO PRIMARY KEY (Automobile),
    CONSTRAINT FK_ASSOCIAZIONI_DISP_AUTO_DISPOSITIVO FOREIGN KEY (Dispositivo) REFERENCES DISPOSITIVI(Codice)
    CONSTRAINT FK_ASSOCIAZIONI_DISP_AUTO_AUTOMOBILE FOREIGN KEY (Automobile) REFERENCES AUTOMOBILI(Targa) ON
DELETE CASCADE
);

CREATE TABLE APPARTENENZE_AUTO (
    Automobile VARCHAR2(7),
    Cliente NUMBER(6) NOT NULL,

    CONSTRAINT PK_APPARTENENZE_AUTO PRIMARY KEY (Automobile),
    CONSTRAINT FK_APPARTENENZE_AUTO_AUTOMOBILE FOREIGN KEY (Automobile) REFERENCES AUTOMOBILI(Targa),
    CONSTRAINT FK_APPARTENENZE_AUTO_CLIENTE FOREIGN KEY (Cliente) REFERENCES CLIENTI(Id)
);

CREATE SEQUENCE utenti_id_seq START WITH 1;

CREATE TABLE UTENTI (
    Id NUMBER(6) DEFAULT utenti_id_seq.nextval,
    Email VARCHAR2(50),
    Password VARCHAR2(72) NOT NULL,
    Cliente NUMBER(6) NOT NULL,

    CONSTRAINT PK_UTENTI PRIMARY KEY (Id),
    CONSTRAINT FK_UTENTI FOREIGN KEY (Cliente) REFERENCES CLIENTI(Id) ON DELETE CASCADE,
    CONSTRAINT CK_EMAIL_UNIQUE UNIQUE (Email)
);

```



Aggiunta delle chiavi esterne e politiche di reazione

Il vincolo di integrità referenziale è un vincolo inter-relazionale che permette di mettere in relazioni dati tra tabelle diverse. In una base di dati relazionale le informazioni vengono distribuite su relazioni differenti al fine di evitare ridondanze di dati che possono portare ad inconsistenze in caso di operazioni di aggiornamento sulla base stessa. Il meccanismo attraverso cui si ha la possibilità di legare fra loro relazioni differenti fa uso della **chiave esterna**, che consiste in un attributo di una specifica che referencia un attributo di chiave primaria di un'altra tabella.

Le chiavi esterne necessarie per collegare le varie tabelle sono le seguenti:

Il campo "Dispositivo" di TRAGITTI_DISPOSITIVI referencia il campo "Codice" di DISPOSITIVI

```
CONSTRAINT FK_TRAGITTI_DISPOSITIVI FOREIGN KEY (Dispositivo) REFERENCES DISPOSITIVI(Codice)
```

Il campo "Casello_Ingresso" di TRAGITTI referencia il campo "Id" di CASELLI

```
CONSTRAINT FK_TRAGITTI_CASELLO_1 FOREIGN KEY (Casello_Ingresso) REFERENCES CASELLI(Id)
```

Il campo "Casello_Uscita" di TRAGITTI referencia il campo "Id" di CASELLI

```
CONSTRAINT FK_TRAGITTI_CASELLO_2 FOREIGN KEY (Casello_Uscita) REFERENCES CASELLI(Id)
```

Il campo "Dispositivo" di ASSOCIAZIONI_DISP_AUTO referencia il campo "Codice" di DISPOSITIVI

```
CONSTRAINT FK_ASSOCIAZIONI_DISP_AUTO_DISPOSITIVO FOREIGN KEY (Dispositivo) REFERENCES DISPOSITIVI(Codice)
```

Il campo "Cliente" di APPARTENENZE_AUTO referencia il campo "Id" di CLIENTI

```
CONSTRAINT FK_APPARTENENZE_AUTO_CLIENTE FOREIGN KEY (Cliente) REFERENCES CLIENTI(Id)
```

Il campo "Automobile" di APPARTENENZE_AUTO referencia il campo "Targa" di AUTOMOBILI

```
CONSTRAINT FK_APPARTENENZE_AUTO_AUTOMOBILE FOREIGN KEY (Automobile) REFERENCES AUTOMOBILI(Targa)
```

Per quanto riguarda le **politiche di reazione**, abbiamo a disposizione tre diverse soluzioni che forniscono opzioni distinte in merito al comportamento da assumere in seguito ad operazioni di cancellazione di tuple contenenti attributi caratterizzati da vincoli di integrità referenziale. Nel nostro database è stato opportuno utilizzare:

- **ON DELETE CASCADE:** consiste nel cancellare automaticamente tutte le tuple contenenti valori di chiave esterna che referenziano il valore di chiave primaria della tupla su cui si vuole effettuare la delete;

Ad esempio:

nel momento in cui viene rimosso un Cliente, sia la sua email che la sua password presenti nella tabella UTENTI, verranno cancellate in cascata:

```
CONSTRAINT FK_UTENTI FOREIGN KEY (Cliente) REFERENCES CLIENTI(Id) ON DELETE CASCADE
```

Qualora invece dovesse essere rimossa un'Automobile dalla relativa tabella, verrà anche rimossa la corrispondente tupla di associazione tra auto e dispositivo dalla tabella ASSOCIAZIONI_DISP_AUTO

```
CONSTRAINT FK_ASSOCIAZIONI_DISP_AUTO_AUTOMOBILE FOREIGN KEY (Automobile) REFERENCES AUTOMOBILI(Targa) ON DELETE CASCADE
```

La gestione dell'eliminazione in cascata di una tupla dalla tabella APPARTENENZE_AUTO viene gestita successivamente tramite trigger.



Indici

Come ben noto, tutte le chiavi primarie e gli attributi con il constraint UNIQUE sono automaticamente anche indici, e conseguentemente è possibile una ricerca molto più rapida grazie all'indicizzazione. Esempi di ciò sono l'attributo "Email" della tabella UTENTI ed il campo "CodiceFiscale" nella tabella CLIENTI. Si ricorda che le chiavi primarie sono automaticamente UNIQUE.

Se così non fosse, sarebbe comunque possibile definire un indice in maniera esplicita.

- Di seguito la creazione di un indice che permette di indicizzare la ricerca degli utenti sulla base della loro e-mail:

```
CREATE INDEX UTENTI_IDX ON UTENTI(Email);
```

- Di seguito la creazione di un indice che permette di indicizzare la ricerca dei caselli sulla base del codice e numero:

```
CREATE INDEX CASELLI_IDX ON CASELLI(Codice, Numero);
```

Dimensionamento

A partire dalle informazioni sul volume dei dati ed una volta scelti i tipi degli attributi da utilizzare nell'implementazione di ogni tabella, è possibile effettuare una stima dei costi in termini di occupazione di memoria della base di dati per la successiva fase di implementazione. L'occupazione in termini di byte per il DBMS Oracle dei tipi dati più diffusi è:

- **NUMBER(x)**: $(\lceil x/2 \rceil + 2)$ byte;
- **DATE**: 7 byte;
- **CHAR(x)**: x byte;
- **VARCHAR(x)**: da 0 a x byte.

Di seguito viene riportata per ogni tabella una stima dell'occupazione di memoria a regime, all'atto della messa in esercizio della base di dati (initial), e quella prevista nel corso del successivo biennio di funzionamento (next):

Dispositivi

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	4000 occ
Codice	NUMBER(6)	5	5K	15K
Totale			5K	15K



Automobili

Attributo	Tipo	Byte	Initial	Next
			1000 occ	3000 occ
Targa	VARCHAR2(7)	7	7K	21K
Modello	VARCHAR2(20)	20	20K	60K
Totale			27K	81K

Utenti

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	3000 occ
Id	NUMBER(6)	5	5K	15K
Email	VARCHAR(50)	50	50K	150K
Password	VARCHAR(72)	72	72K	216K
Cliente	NUMBER(6)	5	5K	15K
Totale			132K	396K

Clienti

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	3000 occ
Id	NUMBER(6)	5	5K	15K
Nome	VARCHAR(10)	10	10K	30K
Cognome	VARCHAR(20)	20	20K	60K
DataNascita	DATE	7	7K	21K
CF	VARCHAR(16)	16	16K	48K
IndirizzoResidenza	VARCHAR(40)	40	40K	120K
CodicePagamento	VARCHAR(30)	30	30K	90K
TipoPagamento	VARCHAR(5)	5	5K	15K
Totale			133K	400K



Tragitti

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	4000 occ
NumeroTragitto	NUMBER(5)	5	5K	20K
Dispositivo	NUMBER(6)	5	5K	20K
CaselloIngresso	DATE	7	7K	28K
CaselloUscita	VARCHAR(16)	16	16K	64K
DataOraIngresso	DATE	7	7K	28K
DataOraUscita	DATE	7	7K	28K
Totale			47000	188K

Caselli

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			737 occ	1000 occ
Id	NUMBER(4)	4	2948	4K
Codice	VARCHAR(20)	20	14740	20K
Numero	NUMBER(2)	3	2211	3K
Totale			20K	27K

Associazioni_disp_auto

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	3000 occ
Dispositivo	NUMBER(6)	5	5K	15K
Automobile	VARCHAR(7)	7	7K	21K
Totale			12K	36K

Appartenenze auto

Attributo	Tipo	Byte	Initial (Byte)	Next (Byte)
			1000 occ	3000 occ
Cliente	NUMBER(6)	5	5K	15K
Automobile	VARCHAR(7)	7	7K	21K
Totale			12K	36K



Gestione della sicurezza

Con riferimento alle specifiche sulle politiche di sicurezza, si è deciso di creare un utente *payngo_dba*, proprietario della base dati, un utente *admin* ed uno *utente*, ognuno di essi con diversi privilegi sulla base dati:

- **Database Administrator (DBA):** Il seguente script crea il **Database Administrator** e gli conferisce tutti i permessi possibili sulla base dati

```
CREATE USER payngo_dba IDENTIFIED BY payngo_dba;  
GRANT DBA TO payngo_dba;
```

- **Admin:** è l'amministratore di sistema, un operatore dell'azienda che deve avere tutti i permessi necessari ad agire sulla base dati al fine di garantire il massimo supporto all'utente. Il ruolo **RESOURCE** racchiude i permessi **CREATE, DROP, INSERT, UPDATE, DELETE** e **SELECT** ed inoltre i permessi di creare eventuali nuove stored procedure e trigger.

```
CREATE ROLE admin;  
GRANT CONNECT TO admin;  
GRANT RESOURCE ON payngo_dba.* TO admin;
```

- **Utente:** l'utente deve avere la possibilità di verificare i suoi dati anagrafici, i veicoli a lui associati, i dispositivi acquistati ed i tragitti effettuati. Per cui gli sono stati garantiti i permessi di **SELECT** sulle seguenti tabelle:

```
CREATE ROLE utente;  
GRANT CONNECT TO utente;  
GRANT SELECT ON payngo.UTENTI TO utente;  
GRANT SELECT ON payngo.CLIENTI TO utente;  
GRANT SELECT ON payngo.APPARTENENZE_AUTO TO utente;  
GRANT SELECT ON payngo.TRAGITTI TO utente;
```



Gestione della concorrenza

In base alle caratteristiche della base di dati, si è scelto di usare il **Protocollo 2PL Stretto**, che permette di evitare il verificarsi di ogni tipo di anomalia. Esempio se, durante l'acquisto di un dispositivo, la transazione viene abortita (per esempio dal cliente stesso che decide di non effettuare più l'acquisto), un'altra transazione iniziata poco dopo quella abortita non potrà vedere che il dispositivo della transazione abortita è tornato nuovamente disponibile).

- Per le transazioni che effettuano solo letture è stato stabilito come livello di isolamento il **READ COMMITTED**, per cui una transazione può leggere i dati modificati da un'altra transazione solo se quest'ultima ha effettuato il commit.

Gestione dell'affidabilità

Abbiamo scelto come storage il **RAID-1**, dato che sfrutta la tecnica di ridondanza con mirroring (ovvero si impiega un altro disco che è lo "specchio" del disco in cui memorizzo tutti i dati). Tale tipologia di RAID ha diversi vantaggi:

- Buon livello di affidabilità (migliore di RAID 2,3,4,5; minore di RAID 6);
- Facilità di sostituzione dei dischi;
- Buon throughput.

Oltre al problema dello storage, c'è bisogno di risolvere il problema legato agli eventuali guasti del sistema:

- Nel caso di guasti *hard* del sistema (es. rottura di un disco), si procede con la **ripresa a freddo** che si basa sul ripristino dei dati a partire dal backup, per poi applicare la ripresa a caldo;
- Nel caso di guasti *soft* del sistema (es. crash del sistema), si procede con la **ripresa a caldo**.



Stored Procedure e Trigger

Stored Procedure

Di seguito sono indicate alcune stored procedure che si sono ritenute necessarie per la gestione del database:

- La seguente stored procedure permette la registrazione dell'utente, inserendo nella tabella CLIENTI i suoi dati anagrafici, e nella tabella UTENTI l'e-mail e la password fornite, nonché l'ID del cliente, ottenuto dall'inserimento precedente, tramite **contatore**:

```
CREATE OR REPLACE PROCEDURE REGISTRA_UTENTE (  
    In_Nome IN CLIENTI.Nome%TYPE,  
    In_Cognome IN CLIENTI.Cognome%TYPE,  
    In_DataNascita IN CLIENTI.DataNascita%TYPE,  
    In_CodiceFiscale IN CLIENTI.CodiceFiscale%TYPE,  
    In_Indirizzo IN CLIENTI.Indirizzo%TYPE,  
    In_Email IN UTENTI.Email%TYPE,  
    In_Password IN UTENTI.Password%TYPE  
)  
AS  
BEGIN  
    DECLARE  
        -- Acquisisci il successivo Id Cliente dal contatore apposito e memorizzalo nella variabile temporanea  
        -- (nel frattempo il contatore viene incrementato, garantendo l'assenza di problematiche di concorrenza)  
        Id_Cliente CLIENTI.Id%TYPE := clienti_id_seq.nextval;  
    BEGIN  
        -- Inserimento dati nella tabella CLIENTI  
        INSERT INTO CLIENTI (Id, Nome, Cognome, DataNascita, CodiceFiscale, Indirizzo)  
        VALUES (Id_Cliente, In_Nome, In_Cognome, In_DataNascita, In_CodiceFiscale, In_Indirizzo);  
        -- Inserimento dati nella tabella UTENTI  
        INSERT INTO UTENTI (Email, Password, Cliente) VALUES (In_Email, In_Password, Id_Cliente);  
        COMMIT;  
    END;  
END REGISTRA_UTENTE;
```



- La seguente stored procedure permette la registrazione di un tragitto, selezionando l'Id dei caselli di ingresso e di uscita a partire dal loro codice e numero, ottenendo l'ultimo numero di tragitto (e di conseguenza il più elevato) di quel determinato dispositivo ed aggiungendovi uno, ed inserendo i dati ottenuti nella tabella TRAGITTI:

```
CREATE OR REPLACE PROCEDURE registrazione_tragitti(
    In_Cod_Dis IN TRAGITTI.Dispositivo%TYPE,
    In_Cod_Cas_E IN CASELLI.Codice%TYPE,
    In_Num_Cas_E IN CASELLI.Numero%TYPE,
    In_Cod_Cas_U IN CASELLI.Codice%TYPE,
    In_Num_Cas_U IN CASELLI.Numero%TYPE,
    In_DataOra_I IN TRAGITTI.DataOraIngresso%TYPE,
    In_DataOra_U IN TRAGITTI.DataOraUscita%TYPE
) AS
BEGIN
DECLARE
    id_ingresso CASELLI.Id%TYPE;
    id_uscita CASELLI.Id%TYPE;
    num_trag TRAGITTI.NumTragitto%TYPE;
BEGIN
    SELECT CASELLI.Id INTO id_ingresso FROM CASELLI WHERE CASELLI.Codice = In_Cod_Cas_E AND CASELLI.Numero =
In_Num_Cas_E;
    SELECT CASELLI.Id INTO id_uscita FROM CASELLI WHERE CASELLI.Codice = In_Cod_Cas_U AND CASELLI.Numero =
In_Num_Cas_U;
    SELECT MAX(TRAGITTI.NumTragitto) INTO num_trag FROM TRAGITTI WHERE TRAGITTI.Dispositivo = In_Cod_Dis;
    INSERT INTO TRAGITTI VALUES (num_trag + 1, In_Cod_Dis, id_ingresso, id_uscita, In_DataOra_I,
In_DataOra_U);
    COMMIT;
END;
END registrazione_tragitti;
```

- La seguente stored procedure restituisce per ogni casello il numero di volte che è stato attraversato in ingresso da un'automobile dotata di un dispositivo.

```
CREATE OR REPLACE PROCEDURE casello_trafficato(
    Out_somma OUT NUMBER,
    Out_max_id OUT CASELLI.Id%TYPE
) AS
BEGIN
BEGIN
    SELECT C.Id, COUNT(*) AS NUMERO_VOLTE INTO Out_max_id, Out_somma
    FROM (TRAGITTI T JOIN CASELLI C ON T.Casello_Ingresso = C.Id)
    GROUP BY C.Id
    ORDER BY NUMERO_VOLTE DESC
    FETCH FIRST 1 ROW ONLY;
END;
END casello_trafficato;
```



- La seguente procedure si occupa di gestire l'aggiunta di nuove auto in seguito alla richiesta del cliente, ed associarvi un dispositivo. Riceve in ingresso un parametro speciale, "In_option", che può assumere i valori 0 ed 1. Nel primo caso, ricerca in DISPOSITIVI un dispositivo non ancora associato ad alcuna auto (e quindi non presente in ASSOCIAZIONI_DISP_AUTO). Nel secondo caso, invece, prende in ingresso il codice di un dispositivo. In questo modo, il cliente può associare un dispositivo già in suo possesso ed associato ad una sola automobile ad una seconda, o scegliere di ricevere un nuovo dispositivo.

```
CREATE OR REPLACE PROCEDURE associazione_auto(  
    In_user_id IN CLIENTI.Id%TYPE,  
    In_targa IN AUTOMOBILI.Targa%TYPE,  
    In_modello IN AUTOMOBILI.Modello%TYPE,  
    In_device IN DISPOSITIVI.Codice%TYPE,  
    In_option NUMBER  
) AS  
BEGIN  
    DECLARE  
        new_id DISPOSITIVI.Codice%TYPE;  
    BEGIN  
        IF In_option = 0 THEN  
            BEGIN  
                SELECT DISPOSITIVI.Codice INTO new_id FROM DISPOSITIVI WHERE DISPOSITIVI.Codice NOT IN (  
                    SELECT ASSOCIAZIONI_DISP_AUTO.Dispositivo  
                    FROM ASSOCIAZIONI_DISP_AUTO  
                )  
                ORDER BY DISPOSITIVI.Codice ASC  
                FETCH FIRST 1 ROW ONLY;  
  
                INSERT INTO AUTOMOBILI (Targa, Modello) VALUES (In_targa, In_modello);  
                INSERT INTO APPARTENENZE_AUTO (Automobile, Cliente) VALUES (In_targa, In_user_id);  
                INSERT INTO ASSOCIAZIONI_DISP_AUTO (Dispositivo, Automobile) VALUES (new_id, In_targa);  
                COMMIT;  
            END;  
        ELSE  
            BEGIN  
                INSERT INTO AUTOMOBILI VALUES (In_targa, In_modello);  
                INSERT INTO APPARTENENZE_AUTO VALUES (In_targa, In_user_id);  
                INSERT INTO ASSOCIAZIONI_DISP_AUTO VALUES (In_device, In_targa);  
                COMMIT;  
            END;  
        END IF;  
    END;  
END associazione_auto;
```



Trigger

Di seguito sono indicati alcuni trigger che si sono ritenuti necessarie per la gestione del database:

- Il seguente trigger si occupa di verificare se l'utente è maggiorenne, sia al momento della registrazione che durante la modifica dei dati:

```
-- Trigger che verifica se l'utente è maggiorenne
CREATE OR REPLACE TRIGGER verifica_eta
BEFORE INSERT OR UPDATE ON CLIENTI
FOR EACH ROW
BEGIN
    -- Calcola l'età sottraendo la data di nascita dalla data corrente
    IF MONTHS_BETWEEN(SYSDATE, :NEW.DataNascita) / 12 < 18 THEN
        raise_application_error(-20001,
            'Devi essere maggiorenne per poterti registrare ');
    END IF;
END;
```

- Il seguente trigger si occupa di verificare se il casello inserito ha codice e numero unici:

```
-- Trigger che verifica se il casello inserito ha codice e numero unici
CREATE OR REPLACE TRIGGER check_caselli
BEFORE INSERT ON CASELLI
FOR EACH ROW
DECLARE
    error_not_unique EXCEPTION;
    found_cod NUMBER(4);
BEGIN
    SELECT COUNT(*) INTO found_cod FROM CASELLI WHERE Codice = :new.Codice AND Numero = :new.Numero;
    IF found_cod > 0 THEN RAISE error_not_unique;
    END IF;
EXCEPTION
    WHEN error_not_unique THEN raise_application_error(-20002, 'Si è provato ad inserire un casello già esistente');
END;
```

- Il seguente trigger si occupa di eliminare dalla tabella APPARTENENZE_AUTO le occorrenze delle automobili il cui proprietario corrisponde all'Id del cliente eliminato:

```
-- Trigger che elimina tutte le auto associate al cliente a seguito della sua eliminazione
CREATE OR REPLACE TRIGGER elimina_appartenenze_auto
AFTER DELETE ON CLIENTI
FOR EACH ROW
BEGIN
    DELETE FROM APPARTENENZE_AUTO WHERE :old.Id = APPARTENENZE_AUTO.Cliente;
END;
```



- Il seguente trigger si occupa di eliminare dalla tabella AUTOMOBILI le occorrenze di automobili la cui targa corrisponde a quella appena eliminata in APPARTENENZE_AUTO.

```
-- Trigger che elimina tutte le auto associate al cliente dalla tabella AUTOMOBILI quando viene eliminato
CREATE OR REPLACE TRIGGER elimina_auto
AFTER DELETE ON APPARTENENZE_AUTO
FOR EACH ROW
BEGIN
    DELETE FROM AUTOMOBILI WHERE AUTOMOBILI.Targa = :old.Automobile;
END;
```

L'insieme dei due trigger precedenti hanno l'effetto di eliminare ogni traccia delle automobili associate ad un cliente se quest'ultimo decidesse di eliminare il proprio account.

- Il seguente trigger si occupa di controllare che l'utente abbia inserito almeno un metodo di pagamento prima che possa registrare un'automobile.

```
-- Trigger che controlla che se l'utente ha registrato un metodo di pagamento
CREATE OR REPLACE TRIGGER check_appartenenze
BEFORE INSERT ON APPARTENENZE_AUTO
FOR EACH ROW
DECLARE
    error_insert EXCEPTION;
    codice CLIENTI.CodicePagamento%TYPE;
    tipo_pagamento CLIENTI.TipoPagamento%TYPE;
BEGIN
    SELECT CLIENTI.CodicePagamento, CLIENTI.TipoPagamento INTO codice, tipo_pagamento FROM CLIENTI WHERE
CLIENTI.Id = :new.Cliente;
    IF codice IS NULL OR tipo_pagamento IS NULL THEN
        RAISE error_insert;
    END IF;
EXCEPTION
    WHEN error_insert THEN raise_application_error(-20003, 'Il metodo di pagamento è NULL, non è possibile
registrare il veicolo');
END;
```



- Il seguente trigger, al momento dell'associazione di un'automobile ad un dispositivo, controlla che a quest'ultimo non siano già associate due automobili; in caso affermativo, blocca l'associazione.

```
-- Trigger che controlla se il dispositivo è associato a più di due auto
CREATE OR REPLACE TRIGGER check_dispositivi
BEFORE INSERT OR UPDATE ON ASSOCIAZIONI_DISP_AUTO
FOR EACH ROW
DECLARE
    error_cant_insert EXCEPTION;
    disp_count NUMBER(4);
BEGIN
    SELECT COUNT(ASSOCIAZIONI_DISP_AUTO.Dispositivo) INTO disp_count FROM ASSOCIAZIONI_DISP_AUTO WHERE
ASSOCIAZIONI_DISP_AUTO.Dispositivo = :new.Dispositivo;
    IF disp_count = 2 THEN RAISE error_cant_insert;
    END IF;
EXCEPTION
    WHEN error_cant_insert THEN raise_application_error(-20004, 'Impossibile inserire il dispositivo, è già
associato a due automobili');
END;
```

Query e Viste

Le viste sono state fondamentali per il funzionamento del sito web creato dal development team in quanto permettono di mostrare all'utente finale solo determinate informazioni contenute nella base dati.

Di seguito è riportata la creazione di alcune viste, particolarmente utili nel caso di query ripetute che devono accedere agli stessi dati anche provenienti da più tabelle, rendendo dunque non necessario l'utilizzo dell'operatore di join:

- La seguente vista permette la creazione di una tabella "Viaggi" consistente che contiene tutti i dispositivi, dunque le automobili, che hanno attraversato un certo casello di ingresso

```
CREATE MATERIALIZED VIEW Viaggi(Dispositivo) AS
SELECT T.Dispositivo
FROM (Tragitti T JOIN Dispositivi D ON T.Dispositivo = D.Codice) JOIN Caselli C ON T.Casello_Ingresso = C.Id
WHERE C.Codice = '$Codice' AND C.Numero = '$Numero'
```

- Mentre la seguente query si avvale della vista "Viaggi" e restituisce tutte le targhe di automobili che non hanno mai attraversato un Casello

```
SELECT A.TARGA AS AUTO_MAI_PASSATE_PER_CASELLO
FROM (Associazioni_disp_auto ADA JOIN Automobili A ON ADA.Automobile = A.Targa) JOIN Dispositivi D ON
ADA.Dispositivo = D.Codice
WHERE D.Codice NOT IN (SELECT V.Dispositivo FROM Viaggi V);
```



- La seguente vista materializzata contiene l'insieme delle targhe associate ad un dispositivo, escludendo quelle che invece non lo sono.

```
CREATE MATERIALIZED VIEW TargheConDispositivo(Targa) AS
SELECT A.Targa
FROM (Associazioni_disp_auto ADA JOIN Automobili ON ADA.Automobile = A.Targa) JOIN Dispositivi D ON
ADA.Dispositivo = D.Codice
```

- La seguente query si avvale della vista "TargheConDispositivo" e restituisce il numero di dispositivi associato ad ogni cliente.

```
SELECT C.Nome, COUNT(*) AS N_Dispositivi
FROM (Appartenenze_auto AP JOIN Clienti C ON AP.Cliente = C.Id) JOIN Automobili AP.Automobile = A.Targa
WHERE A.Targa IN (SELECT N.Targa FROM TargheConDispositivo N)
GROUP BY C.Nome;
```

- La seguente query restituisce il numero di volte in cui un determinato modello di automobile ha attraversato un casello in ingresso, scelto durante la creazione della vista "Viaggi".

```
SELECT A.Modello, COUNT(*) AS Numero_Attraversamenti_Ingresso
FROM (Associazioni_disp_auto ADA JOIN Automobili ON ADA.Automobile = A.Targa) JOIN Dispositivi D ON
ADA.Dispositivo = D.Codice
WHERE A.Modello = "$Modello" AND D.Codice IN (SELECT V.Dispositivo FROM Viaggi V)
GROUP BY A.Modello;
```

- La seguente query restituisce il numero di volte in cui ogni dispositivo ha attraversato un determinato casello in ingresso.

```
SELECT T.Dispositivo, COUNT(*) AS Numero_Attraversamenti
FROM (Tragitti T JOIN Dispositivi D ON T.Dispositivo = D.Codice) JOIN Caselli C ON T.Casello_Ingresso = C.Id
WHERE C.Codice = "$Casello" AND C.Numero = "$Numero"
GROUP BY T.Dispositivo;
```



Progettazione dell'applicazione

Come esempio di realizzazione delle soluzioni proposte, abbiamo deciso di creare una piattaforma web, accessibile all'indirizzo:

<https://basidati.netsons.org>

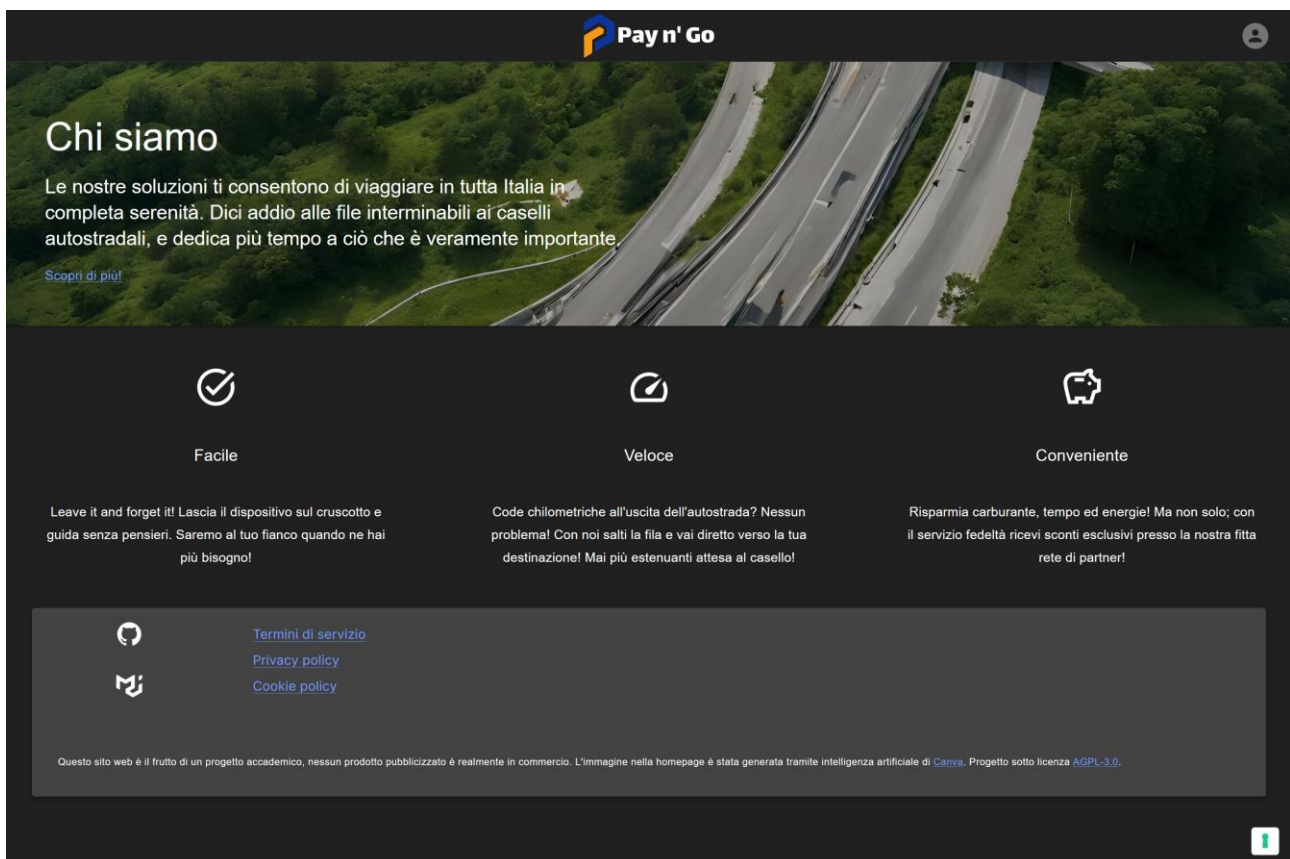
Il codice sorgente è disponibile sulla repository **Github**: <https://github.com/xMattMaster/Pay-n-Go>

Interfaccia grafica

Per la creazione delle interfacce utente si è deciso di utilizzare la libreria JavaScript **React** contornata dal framework **Next.js**. È stata effettuata l'esportazione statica del progetto a causa delle restrizioni della piattaforma. Ogni interfaccia grafica è disponibile sia in tema chiaro che in tema scuro (in base alle preferenze del browser), ed è compatibile sia alla visualizzazione da desktop che da mobile. Di seguito l'elenco delle interfacce grafiche:

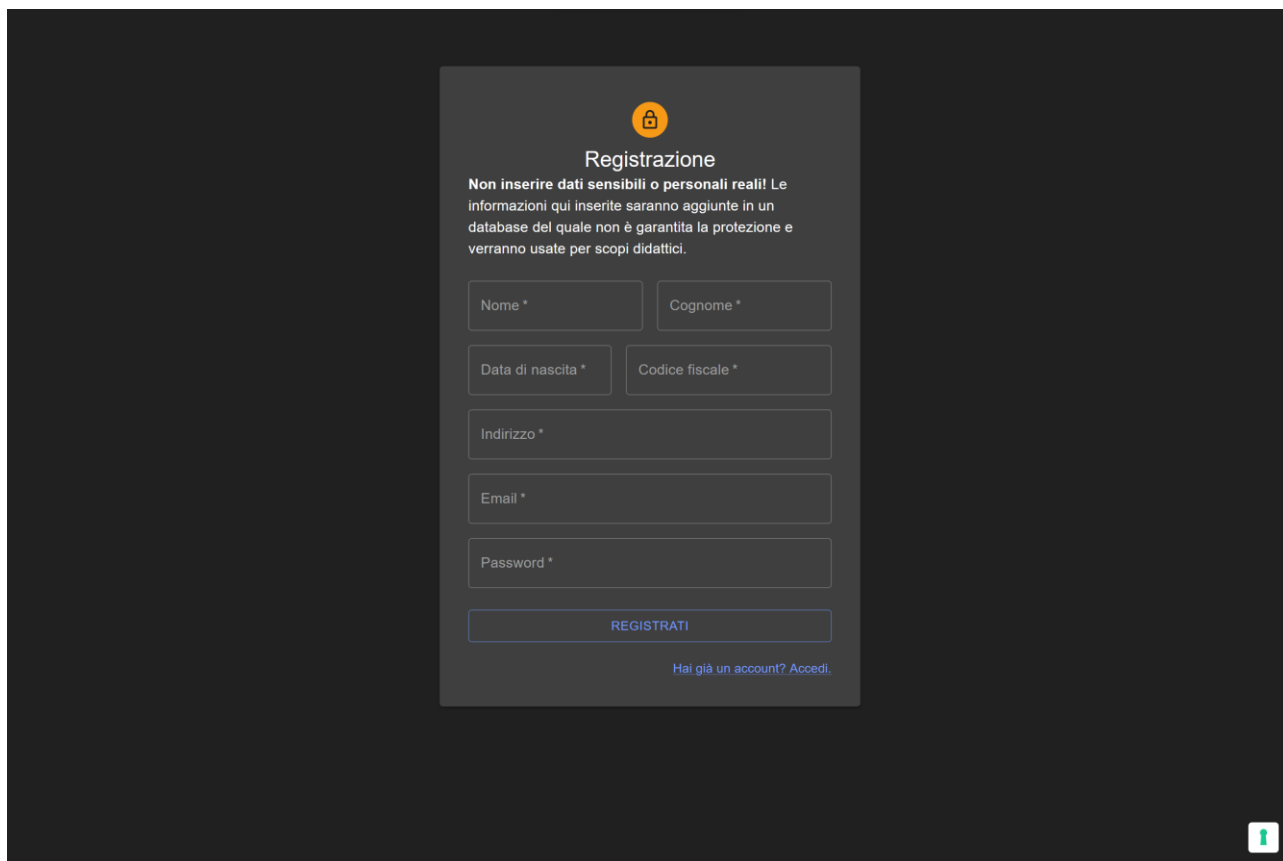
- **Homepage**

Permette di conoscere le caratteristiche del sito ed eventualmente registrarsi o accedere.



- **Pagina di registrazione**

Consente la registrazione sul sito, con conseguente memorizzazione dei dati inseriti dall'utente nel database.

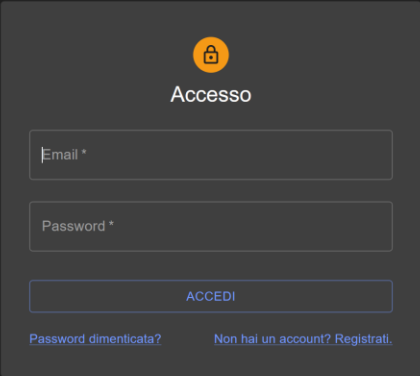


The image shows a registration form centered on a dark background. At the top of the form is a yellow padlock icon. Below it, the title "Registrazione" is displayed. A warning message in Italian states: "Non inserire dati sensibili o personali reali! Le informazioni qui inserite saranno aggiunte in un database del quale non è garantita la protezione e verranno usate per scopi didattici." The form contains several input fields: "Nome *" and "Cognome *" in a row, "Data di nascita *" and "Codice fiscale *" in a row, "Indirizzo *" on a single line, "Email *" on a single line, and "Password *" on a single line. Below these fields is a button labeled "REGISTRATI". At the bottom right of the form, there is a link that says "Hai già un account? Accedi." In the bottom right corner of the entire dark area, there is a small green square icon with a white letter 'i'.



- **Pagina di accesso**

Consente l'accesso all'area riservata dell'utente se si inseriscono delle credenziali valide; controlla l'esistenza dell'email e in caso affermativo controlla la corrispondenza della password inserita con quella memorizzata nel database.

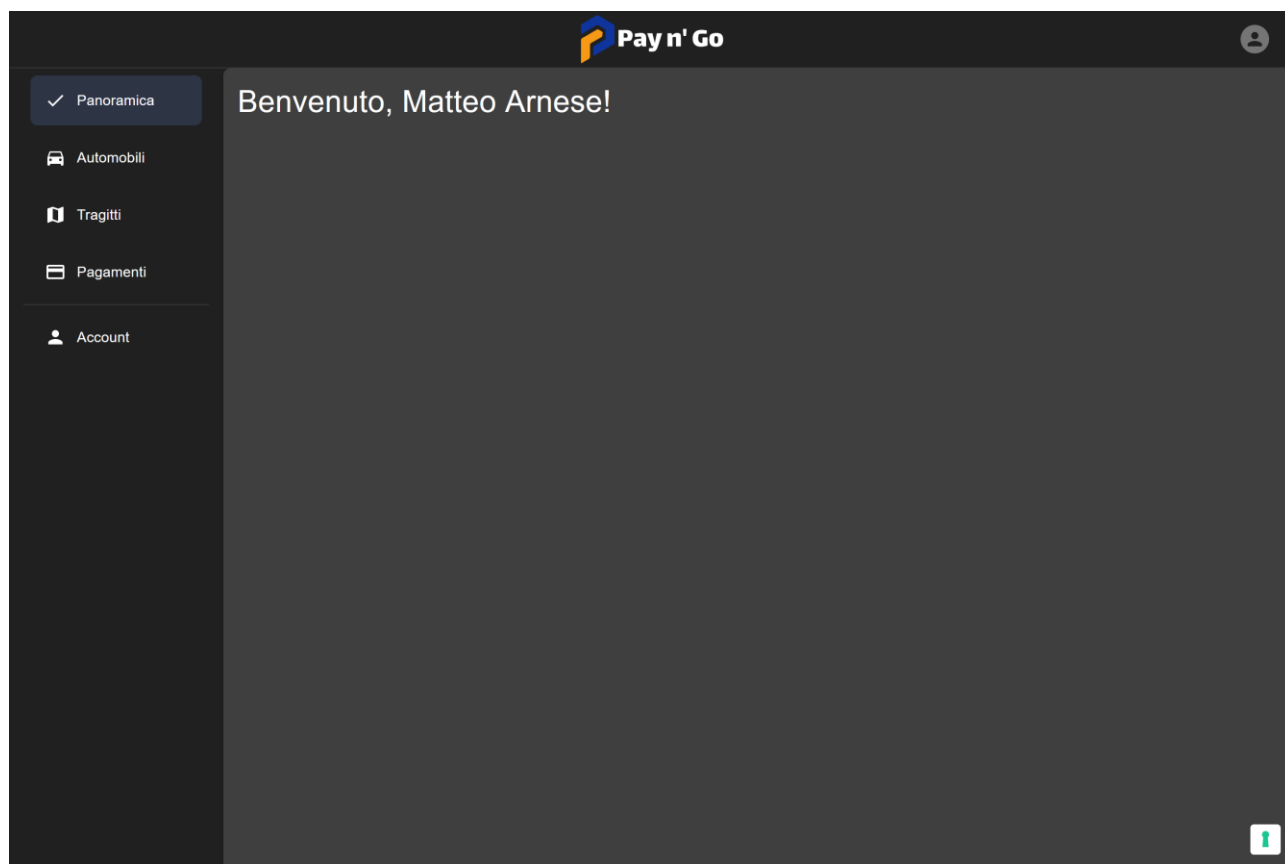


The image shows a login form titled "Accesso" centered on a dark background. At the top of the form is an orange lock icon. Below the title are two input fields: "Email *" and "Password *". Under these fields is a button labeled "ACCEDI". At the bottom of the form, there are two links: "Password dimenticata?" and "Non hai un account? Registrati.". A small green icon is visible in the bottom right corner of the dark background area.



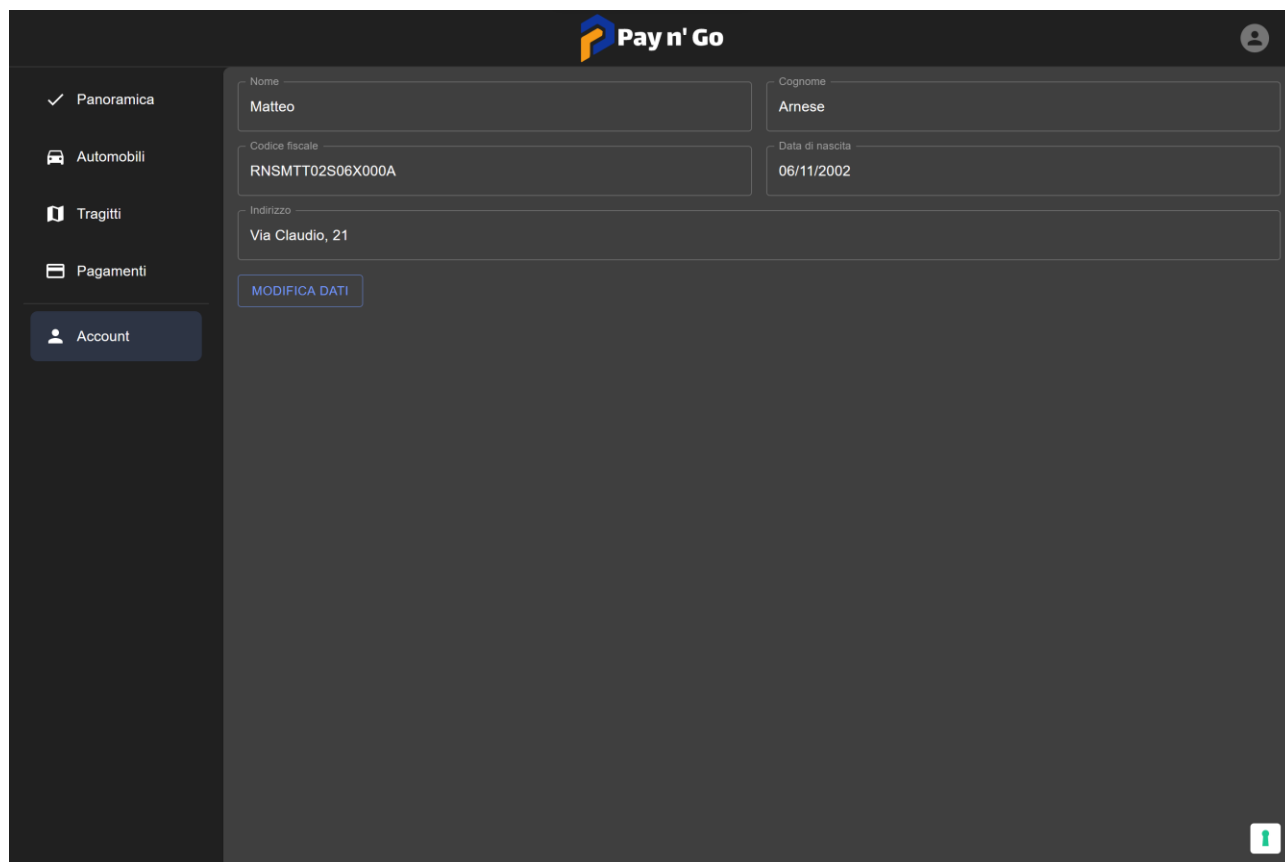
- **Dashboard**

Accessibile solo se l'utente ha effettuato l'accesso, garantendo la visualizzazione di tutte le informazioni che esso ha disposizione, tra cui informazioni inerenti ai **Pagamenti**, alle **Automobili** e ai **Tragitti**. Il tutto avendo a disposizione un proprio **Account**.



- **Dashboard – account**

Permette la visualizzazione e la modifica delle informazioni anagrafiche associate all'utente.



The screenshot displays the 'Pay n' Go' account dashboard. On the left is a dark sidebar with navigation options: 'Panoramica' (checked), 'Automobili', 'Tragitti', 'Pagamenti', and 'Account' (highlighted with a user icon). The main content area has a dark header with the 'Pay n' Go' logo and a user profile icon. Below the header, user details are shown in a light gray box: 'Nome' (Matteo), 'Cognome' (Arnese), 'Codice fiscale' (RNSMTT02S06X000A), 'Data di nascita' (06/11/2002), and 'Indirizzo' (Via Claudio, 21). A 'MODIFICA DATI' button is located below the address field. A small green help icon is in the bottom right corner of the main area.

Nome	Matteo	Cognome	Arnese
Codice fiscale	RNSMTT02S06X000A	Data di nascita	06/11/2002
Indirizzo	Via Claudio, 21		

[MODIFICA DATI](#)



- **Dashboard – automobili**

Consente la visualizzazione delle automobili associate all'utente, con tanto di dispositivo ad esso associate. Permette l'aggiunta di altre automobili e l'eliminazione di quelle esistenti.

✓ Panoramica

Automobili

Tragitti

Pagamenti

Account

Pay n' Go

Targa	Modello	Dispositivo
AX840BG	Tesla Model X	1
CD529IN	Ford Fiesta	3
FE456IT	Fiat Panda	3

1-3 di 3

AGGIUNGI AUTOMOBILEELIMINA AUTOMOBILE



- **Dashboard – tragitti**

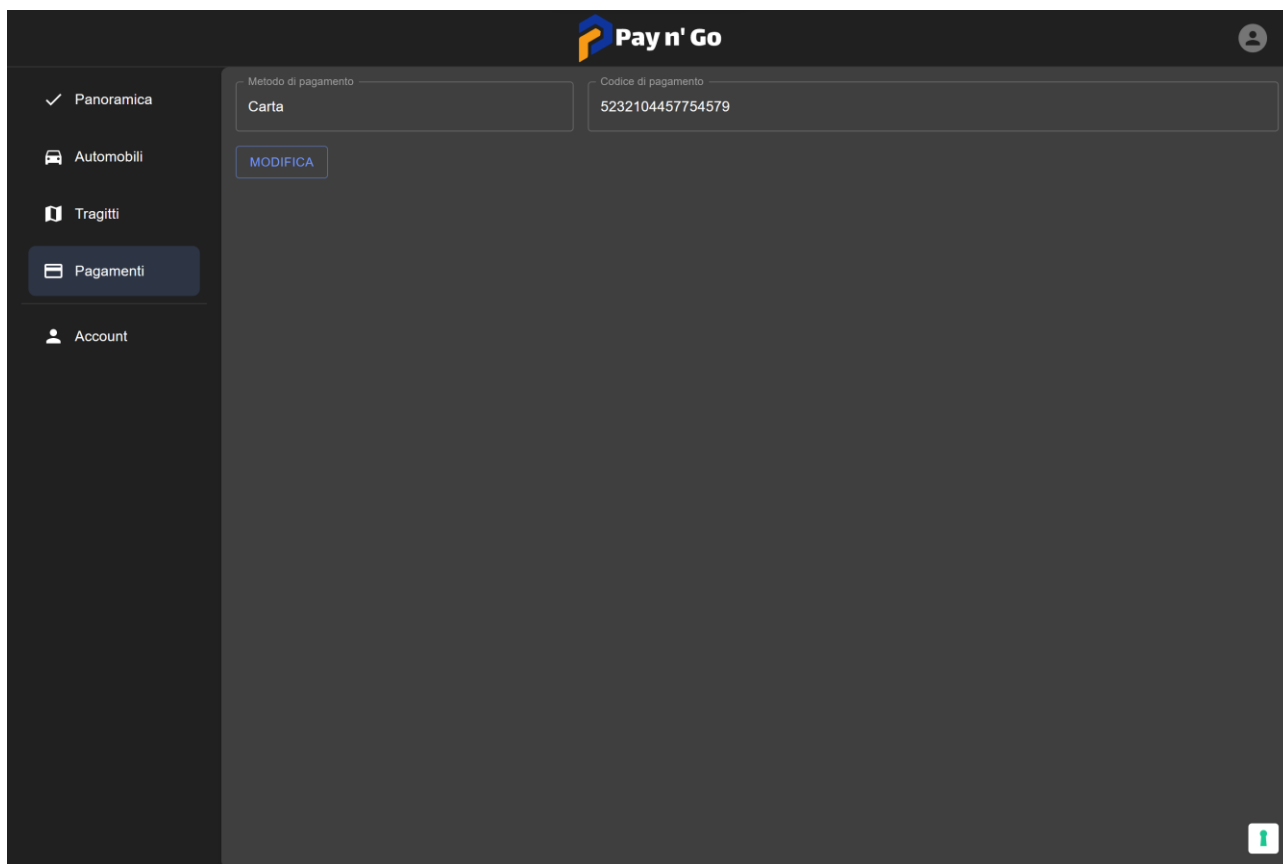
Permette la visualizzazione dei tragitti associati a tutte le automobili del cliente, Per ogni tragitto, specifica il dispositivo ad esso associato, il numero di tragitto relativo a quel determinato dispositivo, i caselli di ingresso e di uscita e data ed ora di ingresso e di uscita.

Pay n' Go						
✓ Panoramica	Disp...	Num...	Casello di ingresso	Data e ora di ingresso	Casello di uscita	Data e ora di uscita
Automobili	1	5	Napoli Est, 1	2023-12-17 11:13:15	Napoli Ovest, 1	2023-12-17 11:13:15
Tragitti	1	4	Napoli Est, 1	2023-12-17 11:12:33	Napoli Ovest, 1	2023-12-17 11:12:33
Pagamenti	3	1	Napoli Est, 1	2023-12-17 01:01:42	Napoli Ovest, 3	2023-12-17 01:01:42
Account	1	2	Napoli Ovest, 3	2023-12-17 01:01:42	Napoli Est, 2	2023-12-17 01:01:42
	3	2	Napoli Est, 3	2023-12-17 01:01:42	Napoli Ovest, 1	2023-12-17 01:01:42
	1	1	Napoli Est, 1	2023-12-05 01:01:42	Napoli Ovest, 1	2023-12-05 02:01:42
1-6 di 6						



- **Dashboard – pagamenti**

Permette di visualizzare il metodo di pagamento associato all'account.



The screenshot displays the 'Pay n' Go' dashboard interface. At the top, the 'Pay n' Go' logo is visible on the left, and a user profile icon is on the right. A dark sidebar on the left contains navigation links: 'Panoramica' (with a checkmark), 'Automobili' (with a car icon), 'Tragitti' (with a ticket icon), 'Pagamenti' (with a card icon and highlighted in blue), and 'Account' (with a person icon). The main content area has a dark background. At the top of this area, there are two input fields: 'Metodo di pagamento' containing the text 'Carta' and 'Codice di pagamento' containing the number '5232104457754579'. Below the 'Metodo di pagamento' field is a blue button labeled 'MODIFICA'. In the bottom right corner of the main area, there is a small green icon of a person inside a white square.



Implementazione del backend

La piattaforma di hosting scelta, **Netsons**, fornisce un database MariaDB, con il quale è possibile interfacciarsi attraverso il tool **phpMyAdmin**. L'interrogazione dello stesso è possibile attraverso degli script PHP. Il DBMS utilizzato riceve istruzioni in linguaggio **MySQL**, di conseguenza è stata effettuata la scrittura di tutti gli elementi precedentemente riportati anche in questo linguaggio. Di seguito qualche esempio delle traduzioni effettuate, comunque accessibili integralmente nella repository Github:

- Trigger **APPARTENENZE_AUTO**

```
DROP TRIGGER IF EXISTS APPARTENENZE_AUTO;
DELIMITER $$
CREATE TRIGGER check_appartenenze
BEFORE INSERT ON APPARTENENZE_AUTO
FOR EACH ROW
BEGIN
    DECLARE codice VARCHAR(30);
    DECLARE tipo_pagamento VARCHAR(5);

    SELECT CodicePagamento, TipoPagamento INTO codice, tipo_pagamento
    FROM CLIENTI
    WHERE Id = NEW.Cliente;

    IF codice IS NULL OR tipo_pagamento IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Il metodo di pagamento è NULL, non è possibile registrare il veicolo';
    END IF;
END $$
DELIMITER ;
```



- Procedure **REGISTRA_UTENTE**

```
-- Procedure per la registrazione di un nuovo utente nei due database
DROP PROCEDURE IF EXISTS REGISTRA_UTENTE;
DELIMITER &&
CREATE PROCEDURE REGISTRA_UTENTE (
    IN In_Nome VARCHAR(20),
    IN In_Cognome VARCHAR(20),
    IN In_DataNascita DATE,
    IN In_CodiceFiscale VARCHAR(16),
    IN In_Indirizzo VARCHAR(40),
    IN In_Email VARCHAR(50),
    IN In_Password VARCHAR(72)
)
MODIFIES SQL DATA
BEGIN
    -- Dichiarazione variabile temporanea
    DECLARE Id_Cliente INT;
    -- Inizializza transazione, garantisce l'atomicità
    START TRANSACTION;
    -- Inserimento dati nella tabella CLIENTI
    INSERT INTO CLIENTI (Nome, Cognome, DataNascita, CodiceFiscale, Indirizzo)
    VALUES (In_Nome, In_Cognome, In_DataNascita, In_CodiceFiscale, In_Indirizzo);
    -- Acquisisci l'Id Cliente appena assegnato tramite AUTO_INCREMENT e memorizzalo nella variabile
    -- temporanea
    SET Id_Cliente = LAST_INSERT_ID();
    -- Inserimento dati nella tabella UTENTI
    INSERT INTO UTENTI (Email, Password, Cliente) VALUES (In_Email, In_Password, Id_Cliente);
    COMMIT;
END; $$
DELIMITER;
```

Da notare l'utilizzo di **LAST_INSERT_ID()**: gli attributi che richiedono autoincremento, come l'Id_Cliente sono stati implementati come **AUTOINCREMENT**, quindi è possibile utilizzare questa funzione per recuperare l'identificativo appena generato per la tabella CLIENTI ed inserirlo anche nella tabella UTENTI.



PHP e query effettuabili tramite interfaccia

Di seguito un esempio di scheletro in PHP per inizializzare una connessione con il database per l'esecuzione di codice MySQL:

```
<?php
    header('Access-Control-Allow-Origin: *');
    header('Access-Control-Allow-Methods: POST');
    header('Access-Control-Allow-Headers: Content-Type');
    header('Content-Type: application/json');
    $servername = "REDACTED_FOR_SECURITY_REASONS";
    $username = "REDACTED_FOR_SECURITY_REASONS";
    $password = "REDACTED_FOR_SECURITY_REASONS";
    $dbname = "REDACTED_FOR_SECURITY_REASONS";

    $content = trim(file_get_contents("php://input"));
    $decoded = json_decode($content, true);

    $input_params = $decoded['input_params'];

    mysqli_report(MYSQLI_REPORT_STRICT | MYSQLI_REPORT_ALL);

    $output = new stdClass();
    $output->res = 0;
    $output->message = "";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    // SQL CODE
    $sql = "Some MySQL Commands";
    try {
        $res = $conn->query($sql);
        $output->message = "OK";
        $output->res = 1;
        echo JSON_encode($output);
    } catch(mysqli_sql_exception $e) {
        $output->message = $e->getMessage();
        $output->res = -1;
        echo JSON_encode($output);
    }
?>
```



Di seguito una lista delle query che vengono effettuate nel file PHP:

- **registration.php**

Viene effettuata una chiamata a procedura REGISTRA_UTENTE. Da notare che la password è stata precedentemente sottoposta ad hashing bcrypt.

```
CALL REGISTRA_UTENTE ('$firstName', '$lastName', '$dateOfBirth', '$cfId', '$address', '$email', '$pass');
```

- **login.php**

Viene innanzitutto eseguita la query:

```
SELECT U.Cliente, U.Password FROM UTENTI U WHERE U.Email = '$email';
```

Si ricorda che Email è un attributo **UNIQUE** della tabella UTENTI, di conseguenza la query restituirà o nessun risultato o una singola tupla. Nel primo caso, viene segnalato l'evento al frontend; se la query al contrario restituisce una tupla, questa è memorizzata in una variabile `data_retrieved`, che conterrà quindi sia Cliente, chiave esterna che riferenzia l'Id del cliente, chiave primaria della tabella CLIENTI, sia l'hash bcrypt della password memorizzata in fase di registrazione. Viene dunque effettuato un check della validità della password, lato server, attraverso la funzione `password_verify`, che compara una stringa (la password inserita dall'utente in fase di login) con l'hash bcrypt presente nel database e precedentemente estratto. In caso di esito positivo del controllo, viene effettuata la ricerca dei dati dell'utente nel database attraverso una nuova query; in caso contrario, viene segnalato l'esito negativo.

```
$data_retrieved = $result_utenti->fetch_array(MYSQLI_ASSOC);
if (password_verify($clearPassword, $data_retrieved['Password']))
{
    // If the password is correct, search for the user's first and last names
    $clientId = $data_retrieved['Cliente'];
    $query_clienti = "SELECT C.Nome, C.Cognome FROM CLIENTI C WHERE C.Id = '$clientId'";
    $result_clienti = $conn->query($query_clienti);
    $client_info = $result_clienti->fetch_array(MYSQLI_ASSOC);
    $output->id = $data_retrieved['Cliente'];
    $output->nome = $client_info['Nome'];
    $output->cognome = $client_info['Cognome'];
    $output->message = "OK";
    $output->res = 1;
    echo JSON_encode($output);
}
else
{
    $output->message = "Password errata";
    $output->res = 0;
    echo JSON_encode($output);
}
```



- **dashboard_account.php**

Vengono recuperati tutti i dati necessari al pannello “**Account**” della Dashboard.

```
SELECT C.Nome, C.Cognome, C.DataNascita, C.CodiceFiscale, C.Indirizzo FROM CLIENTI C WHERE Id = $Id;
```

- **dashboard_account_modify.php**

Viene modificata la tupla di CLIENTI contrassegnata dall'Id utente memorizzato con i nuovi dati inseriti.

```
UPDATE CLIENTI SET CLIENTI.Nome = '$new_name', CLIENTI.Cognome = '$new_surname', CLIENTI.DataNascita = '$new_birthdate', CLIENTI.CodiceFiscale = '$new_cf', CLIENTI.Indirizzo = '$new_location' WHERE CLIENTI.Id = '$user_id';
```

- **dashboard_add_vehicle.php**

Viene effettuata una chiamata a procedura **PROCEDURE_ASSOCIAZIONE_AUTO**; da notare, come già evidenziato, il campo “In_option”.

```
if ($device == "-1") {  
    $sql = "CALL procedure_associazione_auto('$user_id','$targa','$model','$device', 0);"  
} else {  
    $sql = "CALL procedure_associazione_auto('$user_id','$targa','$model','$device', 1);";  
}
```

- **dashboard_delete_vehicle.php**

Viene effettuata la cancellazione dell'associazione tra l'automobile ed il cliente. Di conseguenza, attraverso i trigger e la politica di reazione **ON DELETE CASCADE** di cui sopra, vengono eliminate anche l'automobile, non più necessaria nella base dati, e l'associazione tra essa ed il dispositivo.

```
DELETE FROM APPARTENENZE_AUTO WHERE APPARTENENZE_AUTO.Automobile = '$targa' AND APPARTENENZE_AUTO.Cliente = '$user_id';
```

- **dashboard_payments.php**

Vengono recuperati tutti i dati necessari al pannello “**Pagamenti**” della Dashboard.

```
SELECT C.TipoPagamento, C.CodicePagamento FROM CLIENTI C WHERE Id = $Id;
```

- **dashboard_payments_modify.php**

Viene modificata la tupla di CLIENTI contrassegnata dall'Id utente memorizzato con i nuovi dati di pagamento inseriti.

```
UPDATE CLIENTI SET CLIENTI.CodicePagamento = '$code', CLIENTI.TipoPagamento = '$method' WHERE CLIENTI.Id = '$user_id';
```



- **dashboard_trips.php**

Vengono recuperati tutti i dati necessari al pannello **"Tragitti"** della Dashboard.

```
SELECT CONCAT(T.NumTragitto, '.', T.Dispositivo) AS Id, T.NumTragitto,
        T.Dispositivo,
        CONCAT(C1.Codice, ', ', C1.Numero) AS CasIngresso,
        T.DataOraIngresso,
        CONCAT(C2.Codice, ', ', C2.Numero) AS CasUscita,
        T.DataOraUscita
FROM (TRAGITTI T JOIN CASELLI C1 ON T.Casello_Ingresso=C1.Id) JOIN CASELLI C2 ON
T.Casello_Uscita=C2.Id
WHERE T.Dispositivo IN (
    SELECT ADA.Dispositivo
    FROM (APPARTENZE_AUTO AA JOIN AUTOMOBILI A ON AA.Automobile = A.Targa) JOIN
ASSOCIAZIONI_DISP_AUTO ADA ON A.Targa = ADA.Automobile
    WHERE AA.Cliente = $Id
)
ORDER BY T.DataOraUscita DESC;
```

- **dashboard_vehiclesModify_getUsableDevices.php**

Viene fornita al pannello **"Aggiungi automobile"** una lista dei dispositivi associati all'utente e che hanno meno di due automobili associate.

```
SELECT ADA.Dispositivo
FROM (APPARTENZE_AUTO AA JOIN AUTOMOBILI A ON AA.Automobile = A.Targa) JOIN
ASSOCIAZIONI_DISP_AUTO ADA ON A.Targa = ADA.Automobile
WHERE AA.Cliente = $Id
GROUP BY ADA.Dispositivo
HAVING COUNT(*) < 2;
```

Gestione della sicurezza

Il **backend** gestisce la sicurezza della base dati assegnando ad ogni sezione del sito che interagisce con la base dati (login, registrazione e la dashboard) un account, il quale avrà i permessi per eseguire un numero limitato di operazioni. Ad esempio, la sezione di login comunica con la base dati connettendosi al database con uno script php:

```
$servername = "localhost";
$username = "user_login";
$password = "*****";
$dbname = "basidati_db";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

L'accesso al database ("basidati_db") è disciplinato dai permessi conferiti all'utente "user_login". Per facilitare il settaggio di tutti i parametri, la creazione di questi utenti è avvenuta utilizzando un'utilità di supporto messa a disposizione dalla piattaforma di hosting scelta.

Il **backend** fornisce inoltre la cifratura delle informazioni personali dell'utente con l'ausilio di librerie apposite. In particolare, la password dell'utente viene criptata con la funzione di hashing bcrypt.



Popolamento della base dati

- Con le seguenti chiamate a procedura, si intendono popolare le tabelle CLIENTI ed UTENTI, a meno delle informazioni sul metodo di pagamento, gestite via web all'atto dell'inserimento manuale mediante una specifica interfaccia e conseguente UPDATE TABLE:

```
CALL REGISTRA_UTENTE ('Emanuele','Barbato','13-December-2002','BRBMNL02T13F839D','Via G.Pascoli  
56','emanuelebarbato@gmail.com','BasiD4ti!');  
CALL REGISTRA_UTENTE ('Lorenzo','Cecchini','02-October-2002','CCCLNZ02R02F839C','Via Lambrate  
44','lorenzocecchini2@gmail.com','Pa55w0rD');  
CALL REGISTRA_UTENTE ('Luigi Pio','Castaldo','14-April-2002','CCSLGR03E275539Q','Via Petrarca  
6','luigipiocastaldo78@gmail.com','S3Cur3!');  
CALL REGISTRA_UTENTE ('Paolo','Fox','05-February-1961','FX0PLA61B05H501L','Via Ricchi  
88','paolofoxinfo@hotmail.com','0r05cop0');  
CALL REGISTRA_UTENTE ('Matteo','Arnese','06-November-2002','RNSMTT02S06F839A','Via Claudio  
21','matteoarnese@gmail.com','N3ssunoLASAPR4ma1!!');  
CALL REGISTRA_UTENTE ('Kvicha','Kvaratskhelia','12-February-2001','KVRKCH01B12Z136A','Corso Posillipo  
77','kvichakvara@gmail.com','B0mb3r');  
CALL REGISTRA_UTENTE ('Roberto','Canonico','28-March-70','CNCRBT84T18539D','Via Scarlatti  
8','roberto.canonico@unina.it','Ret1DIC4lcol4tori');  
CALL REGISTRA_UTENTE ('Joe','Biden','20-November-1942','BDNJ0E42S20Z404M','WhiteHouse Street  
76','joebiden@mixmail.com','Am3r1cA');
```

- Con le seguenti istruzioni, si intende popolare la tabella DISPOSITIVI:

```
INSERT INTO DISPOSITIVI VALUES (000001);  
INSERT INTO DISPOSITIVI VALUES (000002);  
INSERT INTO DISPOSITIVI VALUES (000003);  
INSERT INTO DISPOSITIVI VALUES (000004);  
INSERT INTO DISPOSITIVI VALUES (000005);  
INSERT INTO DISPOSITIVI VALUES (000006);  
INSERT INTO DISPOSITIVI VALUES (000007);  
INSERT INTO DISPOSITIVI VALUES (000008);  
INSERT INTO DISPOSITIVI VALUES (000009);  
INSERT INTO DISPOSITIVI VALUES (000010);  
INSERT INTO DISPOSITIVI VALUES (000011);  
INSERT INTO DISPOSITIVI VALUES (000012);  
INSERT INTO DISPOSITIVI VALUES (000013);  
INSERT INTO DISPOSITIVI VALUES (000014);  
INSERT INTO DISPOSITIVI VALUES (000015);  
INSERT INTO DISPOSITIVI VALUES (000016);  
INSERT INTO DISPOSITIVI VALUES (000017);  
INSERT INTO DISPOSITIVI VALUES (000018);  
INSERT INTO DISPOSITIVI VALUES (000019);
```



- Con le seguenti istruzioni, si intende popolare la tabella CASELLI:

```
INSERT INTO CASELLI (Codice,Numero) VALUES ('NAPOLI',4);
INSERT INTO CASELLI (Codice,Numero) VALUES ('MILANO',6);
INSERT INTO CASELLI (Codice,Numero) VALUES ('VERONA',18);
INSERT INTO CASELLI (Codice,Numero) VALUES ('ROMA',2);
INSERT INTO CASELLI (Codice,Numero) VALUES ('SALERNO',9);
INSERT INTO CASELLI (Codice,Numero) VALUES ('GENOVA',10);
INSERT INTO CASELLI (Codice,Numero) VALUES ('POTENZA',4);
INSERT INTO CASELLI (Codice,Numero) VALUES ('NAPOLI',5);
INSERT INTO CASELLI (Codice,Numero) VALUES ('CROTONE',1);
INSERT INTO CASELLI (Codice,Numero) VALUES ('LECCE',16);
INSERT INTO CASELLI (Codice,Numero) VALUES ('BARI',3);
INSERT INTO CASELLI (Codice,Numero) VALUES ('BOLOGNA',8);
```

- Con le seguenti chiamate a procedura, si intende popolare la tabella AUTOMOBILI:

```
CALL ASSOCIAZIONE_AUTO (4,'GG007JB','Bugatti Veyron',000001,0);
CALL ASSOCIAZIONE_AUTO (1,'SB042AB','Ford Kuga',000002,0);
CALL ASSOCIAZIONE_AUTO (5,'UL958TT','Audi Q8',000003,0);
CALL ASSOCIAZIONE_AUTO (6,'P0335TY','Maserati Ghibli',000004,0);
CALL ASSOCIAZIONE_AUTO (6,'KV077KK','Kvaramobile',000004,1);
CALL ASSOCIAZIONE_AUTO (2,'PX884FO','Renault Twingo',000005,0);
CALL ASSOCIAZIONE_AUTO (3,'JA068LU','Kupra Formentor',000007,0);
CALL ASSOCIAZIONE_AUTO (3,'AI707AE','Dacia Sandero',000007,1);
CALL ASSOCIAZIONE_AUTO (7,'P0995DF','Fiat Punto',000008,0);
CALL ASSOCIAZIONE_AUTO (8,'ER231LE','Ferrari Portofino',000009,0);
CALL ASSOCIAZIONE_AUTO (7,'LK521BA','Fiat Multipla',000010,1);
CALL ASSOCIAZIONE_AUTO (4,'WQ562KJ','Alfa Romeo Stelvio',000011,1);
```



- Con le seguenti chiamate a procedura, si intende popolare la tabella TRAGITTI:

```
CALL REGISTRAZIONE_TRAGITTI (000001, 'NAPOLI', 4, 'CROTONE', 1, '20-December-2023 10:20:33', '20-December-2023 15:48:24');
CALL REGISTRAZIONE_TRAGITTI (000003, 'BOLOGNA', 8, 'MILANO', 6, '15-October-2022 22:01:45', '15-October-2022 23:59:59');
CALL REGISTRAZIONE_TRAGITTI (000006, 'LECCE', 16, 'BARI', 3, '17-March-2020 18:11:40', '17-March-2020 20:47:15');
CALL REGISTRAZIONE_TRAGITTI (000004, 'GENOVA', 10, 'VERONA', 18, '28-February-2020 08:35:09', '28-February-2020 12:55:10');
CALL REGISTRAZIONE_TRAGITTI (000003, 'SALERNO', 9, 'POTENZA', 4, '20-August-2018 07:45:34', '20-August-2018 10:40:10');
CALL REGISTRAZIONE_TRAGITTI (000002, 'ROMA', 2, 'BARI', 3, '09-June-2023 16:52:09', '09-June-2023 19:30:00');
CALL REGISTRAZIONE_TRAGITTI (000007, 'NAPOLI', 4, 'NAPOLI', 5, '16-December-2021 23:09:46', '17-December-2021 00:01:05');
CALL REGISTRAZIONE_TRAGITTI (000005, 'MILANO', 6, 'ROMA', 2, '02-July-2006 04:10:40', '02-July-2006 11:43:11');
CALL REGISTRAZIONE_TRAGITTI (000008, 'BOLOGNA', 8, 'VERONA', 18, '15-March-2022 09:27:54', '15-March-2022 11:27:15');
CALL REGISTRAZIONE_TRAGITTI (000009, 'LECCE', 16, 'NAPOLI', 4, '23-May-2008 20:22:22', '13-May-2008 23:59:16');
CALL REGISTRAZIONE_TRAGITTI (000010, 'GENOVA', 18, 'BARI', 3, '06-September-2020 06:57:48', '06-September-2020 16:44:11');
CALL REGISTRAZIONE_TRAGITTI (000011, 'CROTONE', 1, 'SALERNO', 9, '31-December-2001 23:59:59', '01-January-2002 05:11:08');
CALL REGISTRAZIONE_TRAGITTI (000006, 'BARI', 3, 'LECCE', 16, '17-March-2020 20:10:00', '17-March-2020 22:26:10');
```

