1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

CORSO DI LAUREA IN
INFORMATICA APPLICATA
SCUOLA DI
SCIENZE TECNOLOGIE E FILOSOFIA DELL'INFORMAZIONE

STUDENTE: R. GENTILUCCI
MATRICOLA: 292613

CORSO DI PROGRAMMAZIONE E MODELLAZIONE AD OGGETTI

PROGETTO DI ESAME IN LINGUAGGIO C#: GESTIONE PARCHEGGIO

DOCENTE: S. DELPRIORI

INDICE

1. SPECIFICA DEL PROGETTO

La Specifica di Progetto prevede l'implementazione di una applicazione grafica (Windows Form) in linguaggio C# per la gestione delle aree di sosta a pagamento (parcheggi in custodia).
Il software dovrà poter gestire due differenti ambiti applicativi ovvero quello delle soste di breve periodo (soste orarie) e quello delle soste di lungo periodo (soste giornaliere).

Nello specifico l'applicazione dovrà garantire le seguenti funzionalità:

- interfaccia grafica principale (*dashboard*) che mostri la disponibilità del parcheggio ad ospitare nuovi mezzi ed inoltre visualizzi i posti totali, disponibili, occupati, tariffa oraria, tariffa giornaliera, intervallo di durata della tariffa oraria, servizi attivi;
- ingresso di un mezzo nel parcheggio ed emissione del biglietto di entrata;
- uscita di un mezzo nel parcheggio ed emissione del biglietto di uscita con calcolo del prezzo della sosta in base al tempo di permanenza;
- modifica dei parametri tecnici del parcheggio: numero posti totali, tariffa oraria e/o giornaliera, intervallo di durata della tariffa oraria e servizi offerti;

Ogni veicolo sarà caratterizzato dai seguenti parametri:

- numero di targa;
- tipologia del veicolo (automobile, motocicletta, camper);
- tipologia motore (alimentazione a benzina, gasolio, ibrida o elettrica).

L'applicativo dovrà permettere di elaborare statistiche giornaliere relative al numero e tipologia di sosta effettuata (oraria e/o giornaliera), tipologia di mezzi ospitati, ricavi con la possibilità di salvare tutti i dati su disco attraverso specifici file di sistema (archivio di sistema).

2. STUDIO DEL PROBLEMA

Gli aspetti progettuali risultati più critici ed interessanti nella modellazione del problema sono di seguito illustrati nel seguente elenco:

2.1. GESTIONE E SINCRONIZZAZIONE DELLA VISUALIZZAZIONE DELLO STATO DEL PARCHEGGIO (DASHBOARD)

In questo contesto l'applicazione sulla base del numero dei posti di sosta totali del parcheggio (configurati su apposito file di archivio *Parcheggio.txt*) e di quelli attualmente occupati (incrementati ad ogni ingresso e decrementati ad ogni uscita) deve permettere di conteggiare costantemente i posti disponibili in termini numerici e di visualizzare lo stato di capacità del parcheggio (attraverso la specifica colorazione del semaforo di ingresso) oltrechè visualizzare altre informazioni utili per l'uso del parcheggio (costo tariffa oraria e giornaliera, servizi attivi, ecc). A tale scopo si è deciso di aggiornare lo stato del parcheggio all'avvio dell'applicazione ed ad ogni operazione di ingresso ed uscita che si verifica o di eventuali modifiche dei parametri tecnici del parcheggio attraverso un punto di accesso globale su cui poter fare le sincronizzazioni (in termini di nuovi ingressi oppure uscite nel parcheggio o eventuali modifiche dei parametri del parcheggio) implementando (come descritto in seguito) il *Design Pattern Singleton* nell'interfaccia utente principale (*FormMain.cs*).

2.2. GESTIONE DEGLI INGRESSI ED USCITE DAL PARCHEGGIO PER SOSTE ORARIE E/O GIORNALIERE

Gli ingressi ed uscite dal parcheggio richiedono di essere gestite in modo dinamico con strutture dati che permettano di allocare e deallocare dinamicamente memoria RAM a seconda che si verifichi un nuovo ingresso oppure una uscita dal parcheggio stesso. Per venire incontro a tale necessità si è optato di gestire gli ingressi e le uscite attraverso una lista tipizzata (*Collection List*<>) i cui elementi sono oggetti contenenti informazioni relative al veicolo (targa, tipo, alimentazione motore) ed alla sosta (data, ora e minuti di ingresso) i quali possono essere dinamicamente aggiunti (ingresso) e rimossi (uscita) attraverso gli specifici metodi di libreria (*Add, Remove*). In fase di uscita, prima dell'eliminazione dello specifico elemento della lista, tutte le informazioni della sosta saranno salvate su disco fisso mediante file di archivio con opportuna formattazione *(Archivio_Parcheggio.txt*).
Inoltre, secondo il principio di modularità e riusabilità del codice alla base della programmazione ad oggetti, si è scelto di gestire gli ingressi ed uscite dal parcheggio per i contesti orari e giornalieri (la sosta giornaliera è riferita ad una permanenza almeno superiore ad un intervallo di tempo fissato nel file di archivio *Parcheggio.txt*) in modo univoco e sfruttando le stesse classi (*FormIngresso* e *FormUscita*).

In particolare la tipologia di sosta (oraria/giornaliera) viene mappata in fase di uscita del veicolo del parcheggio mediante un campo specifico all'interno della *FormUscita*, e sulla base di tale attributo, viene applicata la tariffa oraria o giornaliera (configurate sull' apposito file di archivio *Parcheggio.txt)* per il calcolo del prezzo della sosta.

2.3. FUNZIONI DI GESTIONE DELL'AREA DI SOSTA

Tali funzioni sono rivolte alla struttura di gestione del parcheggio (gestore) e consistono nella possibilità di modificare le configurazioni dei parametri tecnici del parcheggio (numero di posti totali disponibili, tariffa oraria e/o giornaliera, intervallo di durata della tariffa oraria, servizi attivi), modificare la password di sistema (archiviata su apposito file di archivio *Password.txt*) e permettere elaborazioni dei dati su base giornaliera. Tali funzionalità richiedono dei diritti di accesso in modo da poter essere interdette per gli utenti generici del parcheggio e rese disponibili solo per il gestore dello stesso. In questo senso si è deciso di implementare un processo di accounting al sistema permettendo l'accesso solo al personale accreditato. Tale aspetto è stato realizzato sfruttando il *Pattern Design Proxy* (come descritto in seguito) effettuando una separazione tra il richiedente dell'accesso (client) e il fornitore del servizio (real subject) interponendo tra loro una terza entità (proxy) che funge da intermediario.
Per quanto riguarda la funzione di modifica dei parametri del parcheggio e quella di modifica della password, al fine di avere il pieno controllo delle operazioni, si è deciso di implementare il *Design Pattern Singleton* in modo di avere un unico punto di accesso allo svolgimento di tali particolari funzionalità.
Il processo di elaborazione dei dati su base giornaliera, invece, è stato gestito attraverso l'acquisizione di tutti i dati delle soste (presenti nel file di archivio *Archivio_Parcheggio.txt*) in una specifica *Collection List<>.* Attraverso un calendario è possibile selezionare il giorno di interesse e quindi ricercare nella lista tutte le ricorrenze aventi la stessa data in modo da poterne acquisire i dati di interesse per le statistiche. Inoltre, attraverso un controllo di tipo *ListView,* si mettono a disposizione, oltre a quelli di solo interesse delle statistiche, tutti i dati delle soste relativi alla data selezionata da calendario.

3. SCELTE ARCHITETTURALI

La struttura dell'applicazione, basata su n. 11 classi, è riportata nel seguente Diagramma delle Classi UML:

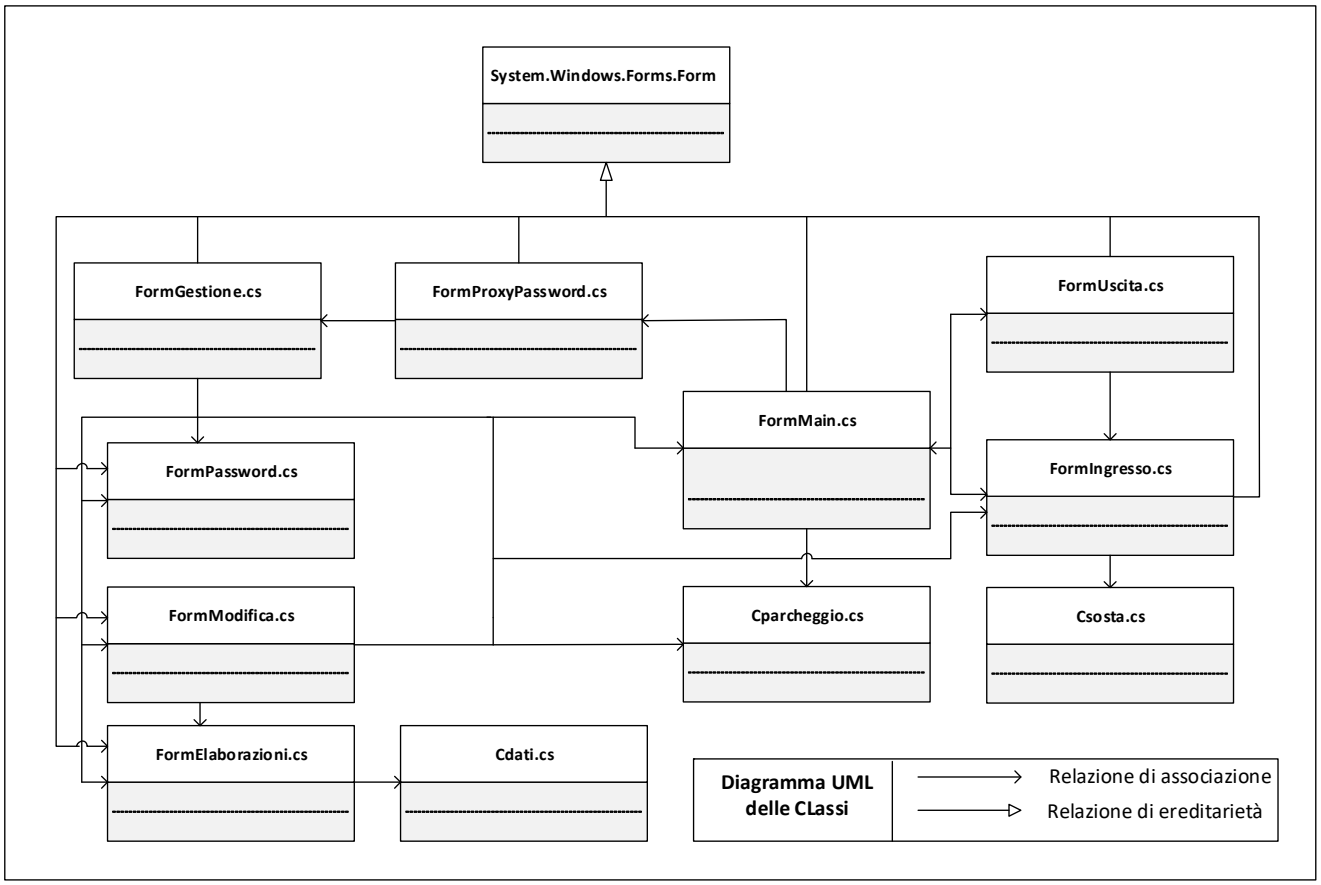
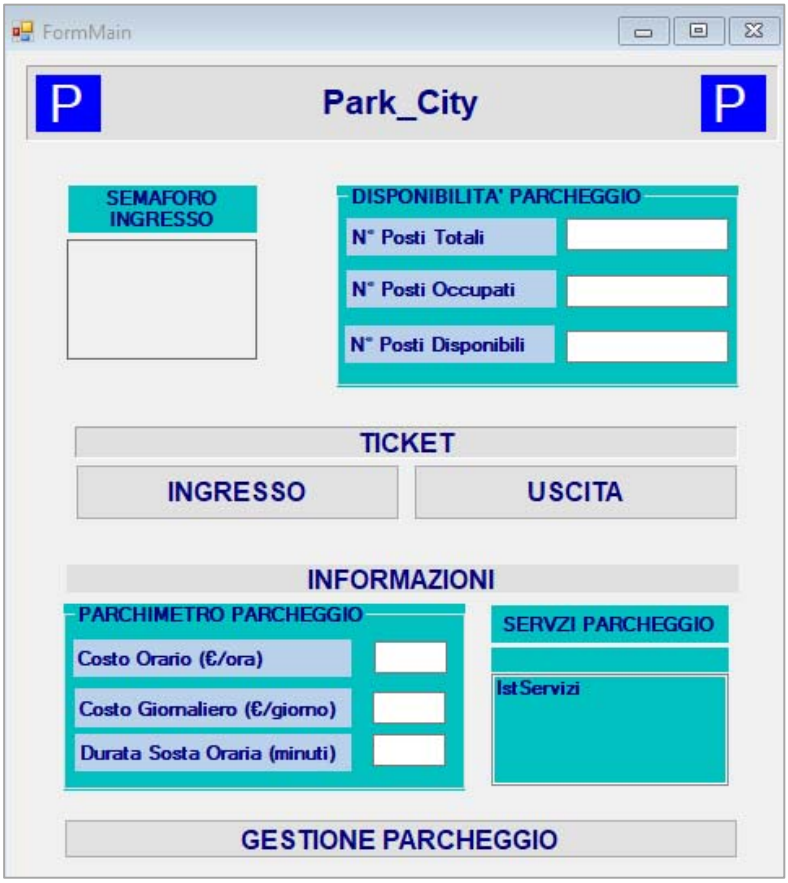


Figura 1: diagramma UML delle classi che costituiscono la struttura dell'applicazione software

Il programma ha una classe principale chiamata *FormMain.cs* dove al suo interno si trovano i collegamenti (tramite appositi pulsanti) alle altre forms di interfaccia utente: *FormIngresso.cs* (per la gestione degli ingressi nel parcheggio), *FormUscita.cs* (per la gestione delle uscite dal parcheggio), *FormProxyPassword.cs* e *FormGestione.cs* per accedere alle funzioni di gestione del parcheggio implementate specifcatamente in *FormModifica.cs* (per la modifica dei parametri di configurazione dell'area di sosta), *FormPassword.cs* (per la gestione della password di sistema), *FormElaborazioni.cs* (per le elaborazioni dei dati e creazioni di statistiche giornaliere). Le altre classi sono: *Cparcheggio.cs* (che gestisce i parametri di configurazione dell'area di sosta), *Csosta.cs* (che contiene tutte le informazioni di ingresso relative ad una sosta) e *Cdati.cs* (che contiene tutte le informazioni relative ad una sosta utili nell'elaborazione dei dati) e *Program.cs* (per l'avvio iniziale dell'applicazione). Il sistema si completa con tre file di archivio su disco fisso che sono: *Parcheggio.txt* (in cui vengono registrati i parametri di configurazione del parcheggio), *Archivio_Parcheggio.txt* (in cui vengono memorizzati i dati relativi alle varie soste nel parcheggio) e *Password.txt* (in cui viene archiviata la password di sistema).

3.1. FORMMAIN.CS



Questa forms costituisce l'interfaccia utente principale ed è divisa da un punto di vista logico in tre sezioni principali.

La prima sezione svolge la funzionalità di *dashboard* ovvero un cruscotto in grado di rappresentare lo stato di capacità del parcheggio (libero oppure occupato), sia in termini numerici che in termini visivi (attraverso un indicatore semaforico) nonché le caratteristiche principali del parcheggio (posti totali, tariffa oraria, tariffa giornaliera, intervallo di durata della sosta oraria, servizi attivi nel parcheggio) (informazioni acquisite dal file di archivio *Parcheggio.txt)*.

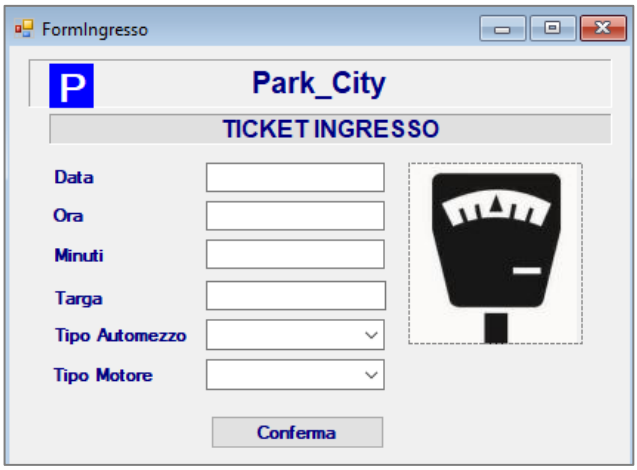
La sincronizzazione ed aggiornamento della dashboard avviene all'avvio dell'applicazione, ad ogni operazione di ingresso ed uscita ed a seguito di eventuali modifiche dei parametri tecnici del parcheggio. Inoltre, per evitare stati di inconsistenza, al raggiungimento della condizione di completa occupazione del parcheggio (oltre alla notifica nella visualizzazione grafica del semaforo di ingresso) viene interdetta ogni altra possibilità di accesso (disabilitazione del pulsante di ingresso orario e giornaliero) finchè non viene liberato un posto di sosta derivante da una successiva uscita. Per gestire al meglio tale aspetto si è optato per l'implementazione del *Design Pattern Singleton* che permette di avere un'unica istanza della interfaccia utente principale (*FormMain*) e quindi di fornire un punto di

accesso globale su cui poter fare gli aggiornamenti in termini di nuovi ingressi oppure uscite nel parcheggio ed anche eventuali modifiche in termini di posti totali disponibili nel parcheggio stesso.

La seconda sezione (qualora il parcheggio risulti ancora con posti disponibili) permette l'ingresso di un nuovo veicolo (*FormIngresso.cs*) oppure consente di effettuare l'uscita dall'area di fermata al termine di una sosta (*FormUscita.cs*).

La terza sezione, invece, viene riservata al gestore del parcheggio e consente di modificare i parametri caratteristici del parcheggio (*FormModifica.cs)*, cambiare la password di sistema *FormPassword.cs*) ed elaborare i dati relativi alle soste nel parcheggio (*FormElaborazioni.cs*) creando statistiche giornaliere. Tale sezione è protetta da diritti di accesso (password) e le relative funzionalità sono interdette fintantochè l'accounting non ha avuto esito positivo (*FormProxyPassword.cs, FormGestione.cs*).

3.2. FORMINGRESSO.CS



Con la *FormIngresso.cs* (istanziata e caricata attraverso il click del pulsante di ingresso presente nella *FormMain.cs*) è possibile acquisire tutte le informazioni relative ad un nuovo ingresso nel parcheggio sia che esso sia di tipo orario o di tipo giornaliero. Nello specifico, vengono acquisite le informazioni relative al veicolo (targa, tipologia di automezzo, tipologia di alimentazione del motore) e quelle relative alla data, ora e minuti di ingresso opportunamente catturate dall'orologio di sistema messo a disposizione dal sistema operativo che manda in esecuzione l'applicazione. I dati di ingresso di ciascuna specifica sosta sono inseriti all'interno della Collection List<*Csosta*> *sosta* in modo da poter gestire dinamicamente gli ingressi e le uscite che si susseguono nella gestione giornaliera del parcheggio (attraverso i metodi di libreria *Add* e *Remove* che agiscono sull'indice della lista).

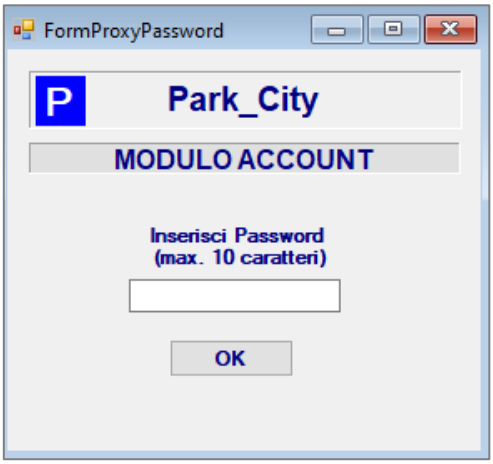
In chiusura della form si effettua una sincronizzazione (aggiornamento) della interfaccia utente principale (*FormMain.cs*) a seguito dei nuovo ingresso e, nel caso che i posti disponibili del parcheggio risultassero esauriti, viene effettuata una segnalazione visiva nel semaforo di ingresso e contemporaneamente disabilitato il pulsante per effettuare un ulteriore ingresso.

3.3. FORMUSCITA.CS

Con la *FormUscita.cs* (istanziata e caricata attraverso il click del pulsante di uscita presente nella *FormMain.cs*) è possibile acquisire e completare tutte le informazioni relative ad una sosta di un veicolo nel parcheggio. Nello specifico, attraverso il codice alfanumerico che identifica la targa del veicolo, si effettua una ricerca nella Collection List<*Csosta*> *sosta* e si recuperano i dati di ingresso al parcheggio di quello specifico mezzo completandoli con i dati di uscita (ora e minuti). Successivamente si calcolano i minuti di permanenza nell'area di sosta quindi si stabilisce se la sosta è di tipo oraria o giornaliera e si calcola il prezzo del biglietto applicando la tariffa opportuna. Infine, tutte le informazioni complete di una sosta vengono memorizzate (modalità *append*) nel file di archivio *Archivio_Parcheggio.txt* secondo uno specifico formato.

In chiusura della form si effettua una sincronizzazione (aggiornamento) della interfaccia utente principale (*FormMain.cs*) a seguito dell'uscita di un veicolo dal parcheggio e, nel caso che i posti disponibili del parcheggio risultavano esauriti, viene effettuata una segnalazione visiva nel semaforo di ingresso di nuova disponibilità e contemporaneamente riabilitato il pulsante per effettuare un nuovo ingresso.

3.4. FORMPROXYPASSWORD.CS E FORMGESTIONE.CS



Attraverso il pulsante *Gestione Parcheggio* della *FormMain.cs* è possibile accedere alle funzionalità utili alla gestione del parcheggio e riservate al gestore dello stesso. In particolare viene istanziata la *FormProxyPassword.cs* che secondo lo schema del *Design Pattern Proxy* (con funzionalità di *Proxy Protection*) si occupa di verificare i diritti di accesso delle funzionalità di gestione. In particolare l'oggetto istanza della *FormProxyPassword.cs* (proxy) funge da controllore degli accessi richiesti dalla *FormMain.cs* (client) attraverso la verifica di corrispondenza della password inserita rispetto a quella di sistema (registrata nel file di archivio *Password.txt*) ed una volta riscontrati i diritti di accesso inoltra le richieste alla *FormGestione.cs* (real subject) che consente di accedere e svolgere le funzioni di gestione.

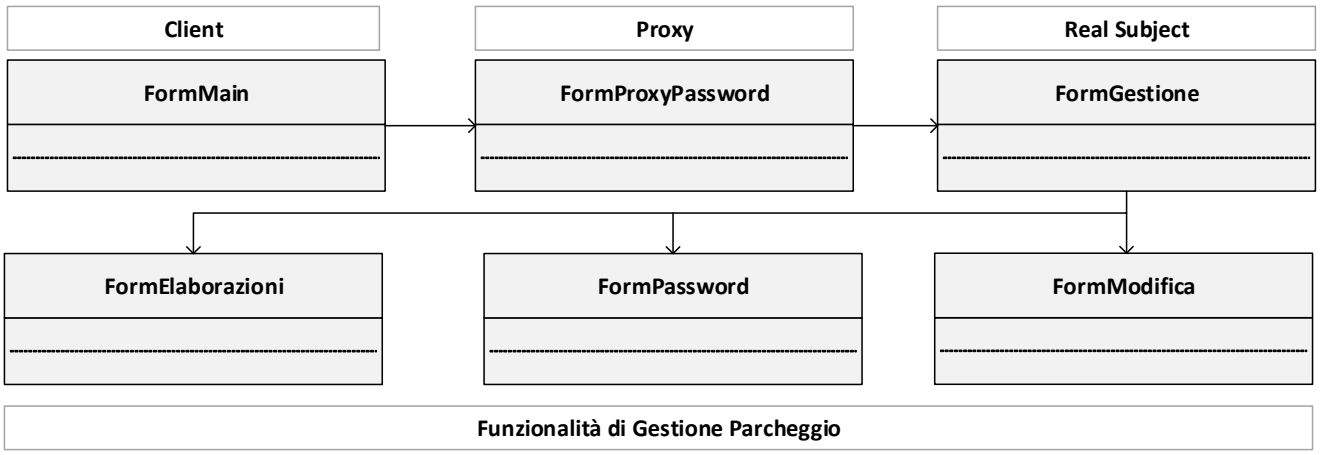
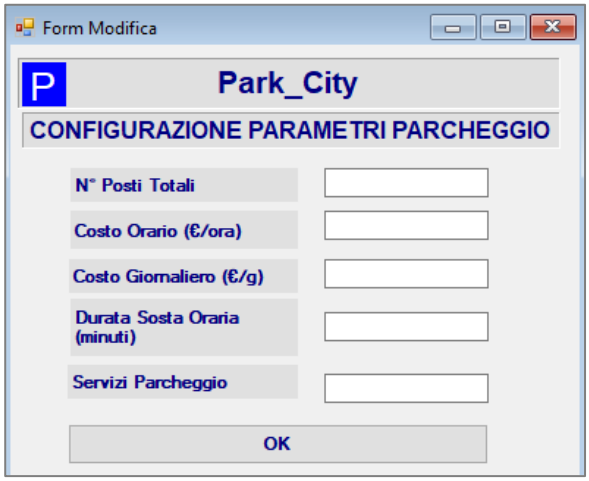


Figura 2: diagramma UML delle classi che partecipano all'implementazione del Design Pattern Proxy

Le specifiche funzionalità di gestione del parcheggio sono svolte attraverso le seguenti classi.

3.5. FORMMODIFICA.CS



Attraverso l'istanza di questa classe è possibile modificare i parametri tecnici caratteristici del parcheggio ovvero il numero di posti totali disponibili, tariffa oraria, tariffa giornaliera, intervallo di durata della tariffa oraria, servizi attivi (per es. WI-FI, BAR, ecc.).
I valori dei nuovi parametri vengono raccolti negli specifici campi della form e vengono quindi successivamente archiviati (sovrascrittura) nel file *Parcheggio.txt*.

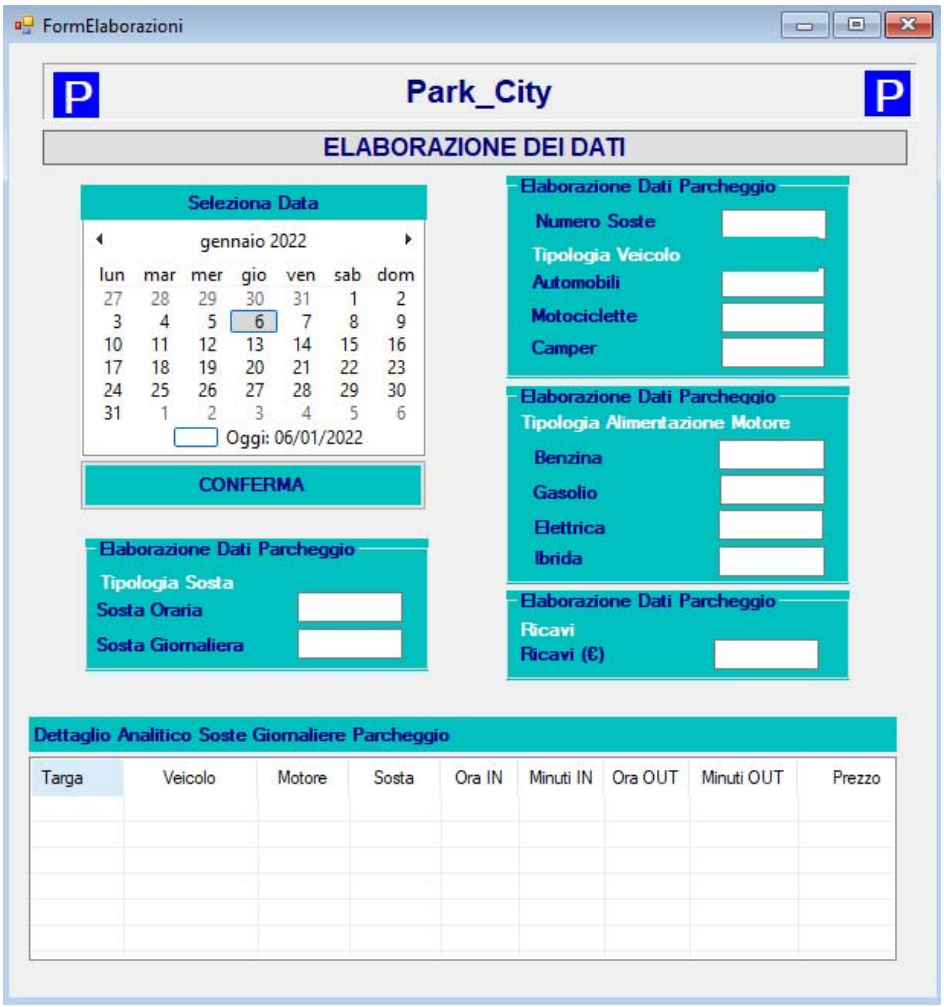
In ultimo si effettua una sincronizzazione (aggiornamento) della interfaccia utente principale (*FormMain.cs*) a seguito dei nuovi parametri inseriti.
Per avere un'unica istanza della *FormModifica.cs,* in modo da avere il pieno controllo dell'operazione di modifica, si è implementato in tale form il *Design Pattern Singleton* garantendo così un accesso globale alla funzione di modifica dei parametri tecnici del parcheggio.

3.6. FORMPASSWORD.CS

Attraverso l'istanza di questa classe è possibile modificare la password di sistema con la quale è possibile fare l'accounting alle funzionalità di gestione del parcheggio. Nello specifico viene richiesta di inserire la nuova password che viene successivamente registrata nel file di archivio *Password.txt.*
L'aggiornamento della password è immediatamente effettivo e per accedere di nuovo alle funzionalità gestionali del parcheggio si dovrà far riferimento alla nuova password.

Per avere un'unica istanza della *FormPassword.cs,* in modo da avere il pieno controllo dell'operazione di modifica, si è implementato in tale form il *Design Pattern Singleton* garantendo così un accesso globale alla funzione di modifica della password del parcheggio.

3.7. FORMELABORAZIONI.CS



Attraverso l'istanza di tale form è possibile effettuare delle elaborazioni statistiche dei dati giornalieri delle soste e rappresentarle a video. In particolare vengono acquisiti i dati precedentemente archiviati nel file di sistema *Archivio_Parcheggio.txt* ed inseriti all'interno di una Collection List<*Cdati*> *elaborazioni.* Una volta selezionato il giorno di interesse per l'elaborazione dei dati (attraverso lo specifico calendario) viene fatta una scansione della lista per individuare le soste corrispondenti alla data di interesse in modo da poterne individuare il numero complessivo delle soste totali, della tipologia di veicoli, della tipologia di alimentazione del motore dei veicoli, della tipologia di soste (oraria o giornaliera) ed il ricavo giornaliero totale.

Attraverso la *ListViewRiepilogo* vengono mostrate a video tutte le informazioni analitiche delle soste oggetto dell'elaborazione dei dati.

4. DOCUMENTAZIONE SULL'UTILIZZO

L'applicativo software per l'uso richiede che siano presenti oltre i file sorgenti anche i seguenti file di archivio:

- *Parcheggio.txt*
  in cui vi sono riportati i parametri tecnici del parcheggio secondo il seguente formato:
  numero posti disponibili totali
  tariffa oraria
  tariffa giornaliera
  intervallo di durata della tariffa oraria
  servizi attivi (es: Servizio1,Servizio2,Servizio3,ecc)
- *Password.txt*
  in cui è riportata la password di sistema per poter accedere alle funzioni di gestione del parcheggio.

5. USE CASES

In questa sezione viene riportato il diagramma UML dei Casi d'Uso (Use Cases) che descrivono le principali modalità di interazione degli utenti con l'applicativo software:

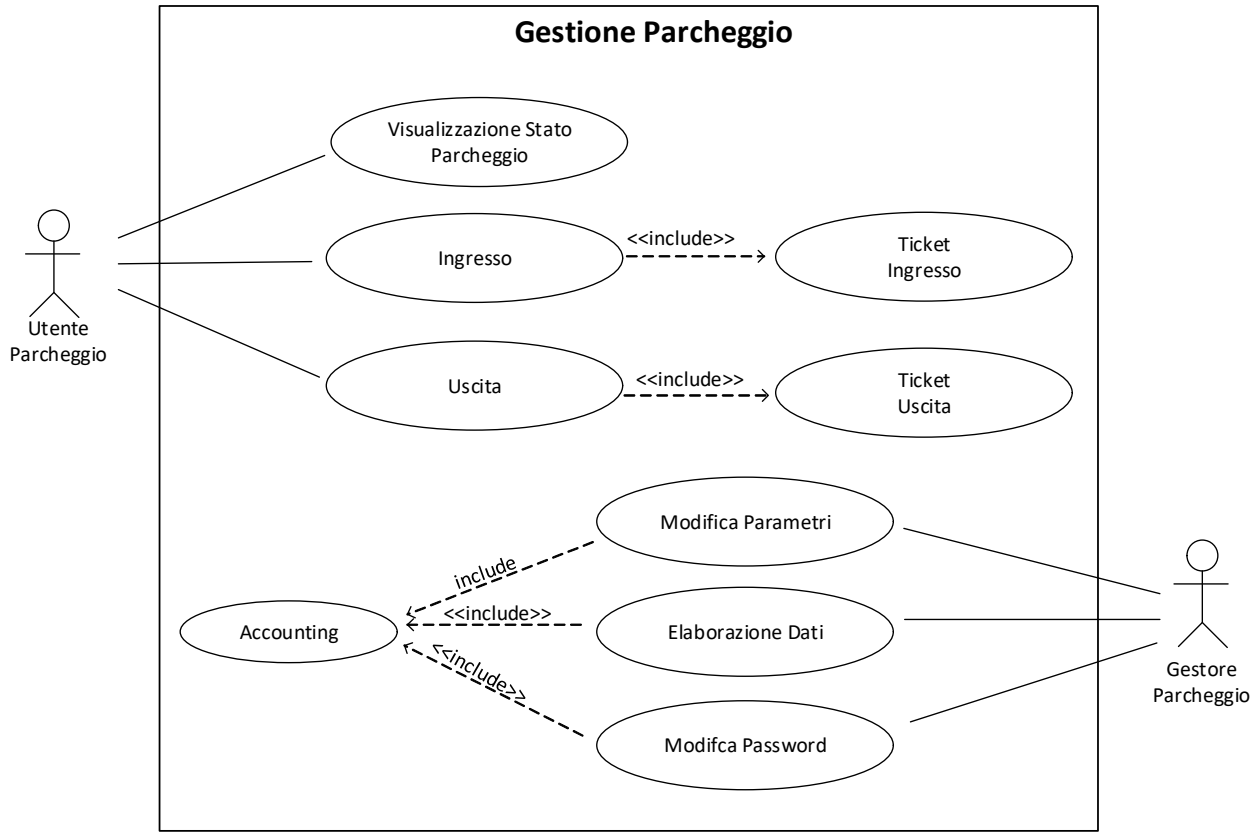


Figura 3: diagramma UML dei casi d'uso dell'applicazione software

5.1. DESCRIZIONE DEI CASI D'USO

Di seguito viene fatta una descrizione testuale dei casi d'uso attraverso un formato tabellare:

Use Case: Visualizzazione Stato Parcheggio	**ID**: UC1	**Attore**: Utente Parcheggio
Precondizione: /		
Corso degli eventi: L'utente verifica la disponibiltà del parcheggio ed eventualmente effettua un'ingresso		
Post-condizione: Generazione del ticket d'ingresso o rinuncia all'ingresso		
Alternativa: /		

Use Case: Ingresso	**ID**: UC2	**Attore**: Utente Parcheggio
Precondizione: UC1		
Corso degli eventi: L'utente attiva la procedura di ingresso al parcheggio		
Post-condizione: UC3		
Alternativa: /		

Use Case: Ticket Ingresso	**ID**: UC3	**Attore**: Utente Parcheggio
Precondizione: UC2		
Corso degli eventi: L'utente genera il ticket d'ingresso		
Post-condizione: UC4		
Alternativa: /		

Use Case: Uscita	**ID**: UC4	**Attore**: Utente Parcheggio
Precondizione: UC3		
Corso degli eventi: L'utente attiva la procedura di uscita dal parcheggio		
Post-condizione: UC5		
Alternativa: /		

Use Case: Uscita	**ID**: UC5	**Attore**: Utente Parcheggio
Precondizione: UC4		
Corso degli eventi: L'utente genera il ticket di uscita, paga il biglietto ed esce dal parcheggio		
Post-condizione: UC5		
Alternativa: /		

Use Case: Accounting	**ID**: UC6	**Attore**: Gestore Parcheggio
Precondizione: /		
Corso degli eventi: Inserimento password e verifica correttezza con quella del sistema		
Post-condizione: UC7, UC8, UC9, UC6		
Alternativa: /		

Use Case: Modifica Parametri	**ID**: UC7	**Attore**: Gestore Parcheggio
Precondizione: UC4		
Corso degli eventi: Il gestore modifica uno o piu' parametri del parcheggio e ritorna nella pagina principale		
Post-condizione: /		
Alternativa: UC8, UC9		

Use Case: Elaborazioni Dati	**ID**: UC8	**Attore**: Gestore Parcheggio
Precondizione: UC6		
Corso degli eventi: Il gestore seleziona una data e vengono visualizzate le statistiche giornaliere		
Post-condizione: /		
Alternativa: UC9, UC7		

Use Case: Modifica Password	**ID**: UC9	**Attore**: Gestore Parcheggio
Precondizione: UC6		
Corso degli eventi: Il gestore inserisce una nuova password e ritorna nella pagina principale		
Post-condizione: /		
Alternativa: UC7, UC8		