Scopo del documento

Questo documento ha lo scopo di riportare la definizione dell'architettura del progetto PricePal utilizzando i diagrammi delle classi UML (Unified Modeling Language) e il codice in OCL (Object Constraint Language).

1 - Diagramma delle classi

Nel presente capitolo vengono presentate in linguaggio UML tutte le classi previste per la realizzazione del progetto PricePal. Tali classi sono state costruite a partire dai diagrammi di contesto e dei componenti presentati nel precedente documento.

In particolare ogni classe è identificata da:

- Nome
- Attributi
- Metodi

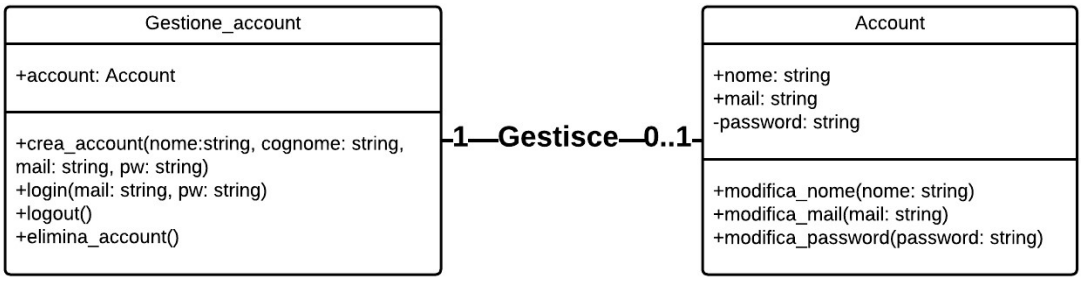
Inoltre, UML ci permette di mostrare come le differenti classi siano in relazione tra loro.

1.1 - Account e Gestione_account

Analizzando il diagramma di contesto si può notare come l'attore principale sia l'utente, il quale dovrà essere dotato di un account.

Si rende necessaria dunque la creazione di una classe Account, la quale contenga il nome, la mail e la password associate all'utente, il quale potrà eventualmente modificarle in futuro.

Una classe apposita Gestione_account si occuperà di gestire il login ed il logout oltre a tener traccia dell'utente attualmente collegato. Essa permetterà inoltre di aggiungere o rimuovere dal database un Account attraverso appositi metodi crea_account(...) ed elimina_account(...).

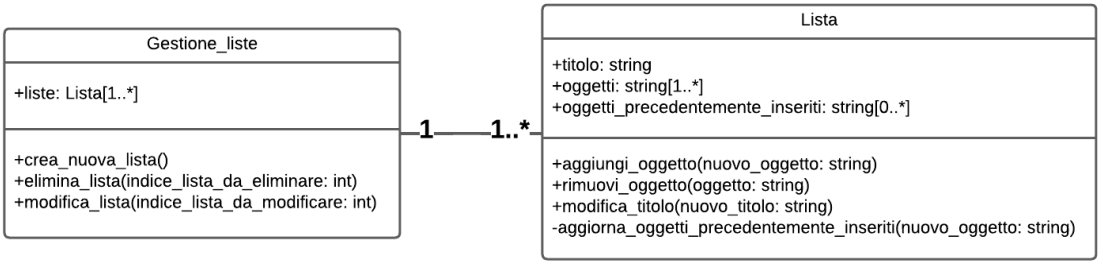


1.2 - Lista e Gestione_liste

È necessaria la creazione di classi apposite che si occupino di tener traccia delle liste create dall'utente.

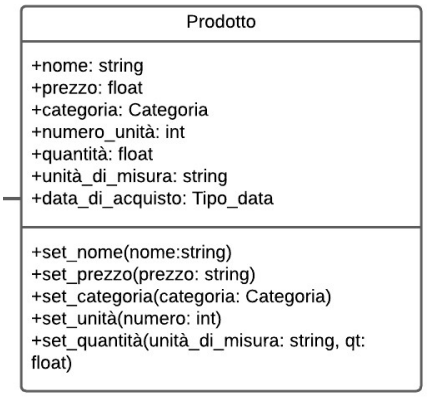
La classe Lista contiene oltre al titolo e all'elenco di oggetti presenti nella lista un attributo specifico contenente gli oggetti precedentemente inseriti in altre liste. Tale attributo è necessario per poter agevolare l'utente permettendogli di non dover riscrivere ogni volta il nome di un oggetto che ha già acquistato.
Il metodo aggiorna_oggetti_precedentemente_inseriti(...) permette di aggiornare il database ogni volta che un nuovo oggetto viene definito dall'utente.

La classe Gestione_liste è quella che permetterà all'utente di vedere tutte le liste che ha creato, di modificarle o eliminarle e di crearne di nuove.



1.3 - Prodotto

Poiché i carrelli sono composti da prodotti che devono contenere dei dati, è necessario dichiarare una classe Prodotto che conterrà tutti gli attributi relativi al singolo prodotto acquistato: nome, prezzo, categoria, unità, quantità, data di acquisto. La classe Prodotto è fornita di un setter per ogni suo attributo.



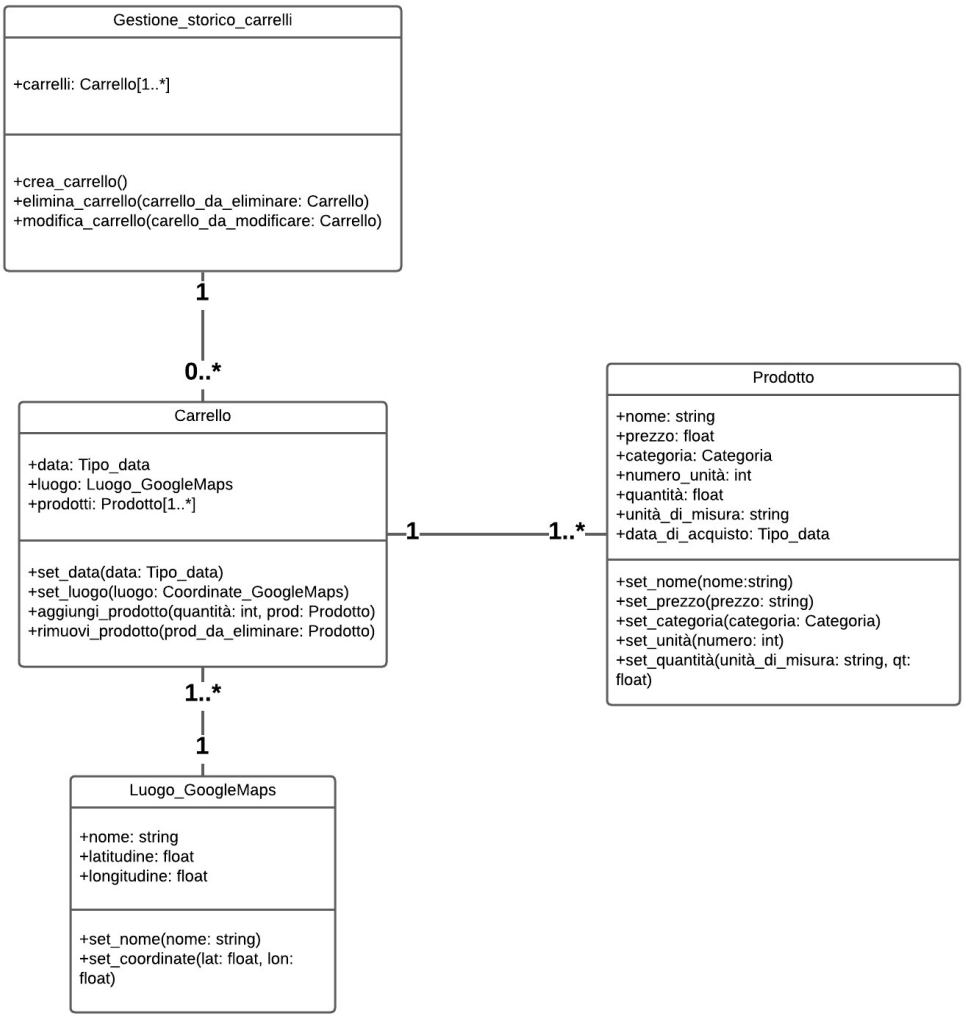
1.4 - Carrello, Gestione_storico_carrelli e Luogo_GoogleMaps

È necessaria la creazione di classi apposite che si occupino di tener traccia dei carrelli creati dall'utente.

La classe Gestione_storico_carrelli permette di creare, eliminare e modificare i carrelli precedentemente creati dall'utente.

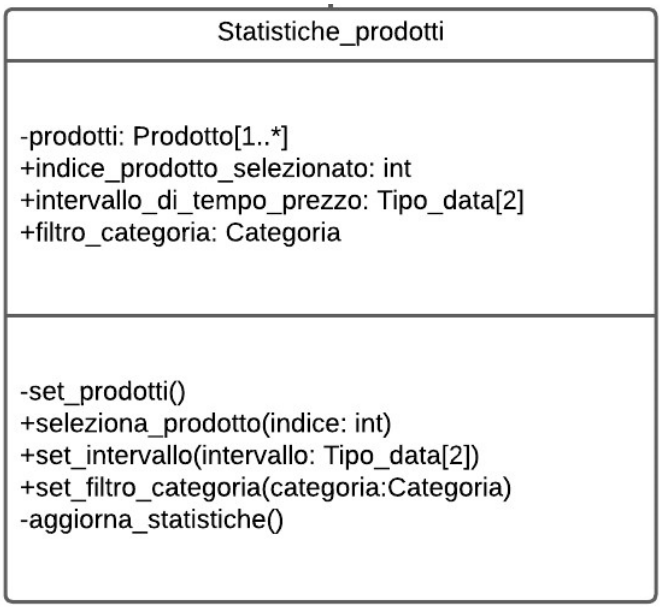
La classe Luogo_GoogleMaps permette di abbinare ai carrelli un luogo restituito dalle API di Google Maps, dopo che l'utente lo avrà selezionato dalla mappa dei negozi. Essa contiene il nome del luogo e le sue coordinate.

La classe Carrello contiene la data, il Luogo_GoogleMaps in cui è stata fatta la spesa alla quale il carrello fa riferimento, oltre alla lista di prodotti acquistati. Permetterà all'utente di aggiungere o rimuovere prodotti e di modificare data e luogo della spesa.



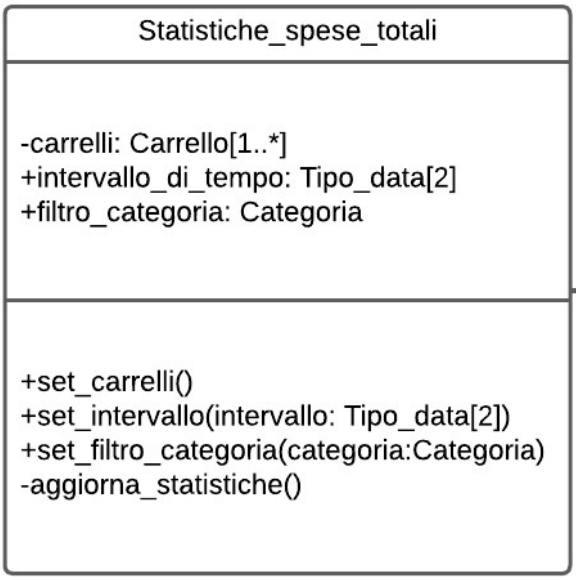
1.5 - Statistiche_prodotti

La classe Statistiche_prodotti permette di selezionare il prodotto tra quelli acquistati del quali si vogliono ottenere le statistiche. Permetterà inoltre all'utente di filtrare la lista di prodotti per categoria e di visualizzare le statistiche relative ad uno specifico periodo di tempo da lui definito.



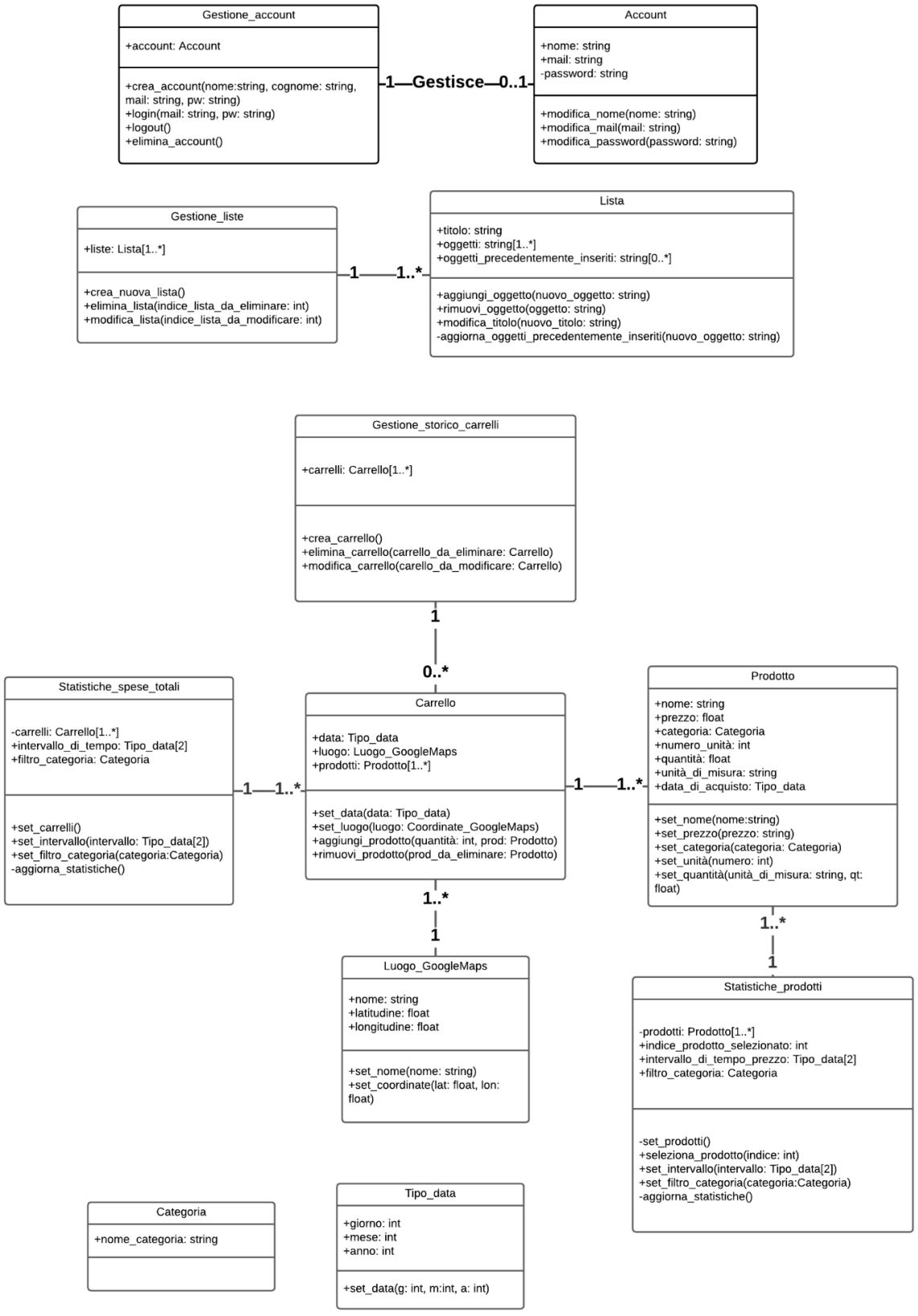
1.6 - Statistiche_spese_totali

La classe Statistiche_spese_totali contiene tutti i carrelli creati dall'utente, dai quali si ricaveranno i dati sui quali effettuare statistiche. Permetterà inoltre all'utente di applicare un filtro per vedere le spese relative ad una determinata categoria o ad uno specifico periodo di tempo da lui definito.



1.7 Diagramma delle classi complessivo

Riportiamo di seguito il diagramma delle classi con tutte le classi fino ad ora presentate. Oltre alle classi già descritte, sono state inserite classi ausiliarie, ad esempio Tipo_data e Categoria. Queste classi servono per descrivere eventuali tipi strutturati usati, ad esempio, negli attributi delle altre classi.



2 - Codice in Object Constraint Language

In questo capitolo sono descritte, utilizzando l'Object Constraint Language (OCL), alcune invarianti, precondizioni e postcondizioni relative alle classi precedentemente descritte. In particolare, abbiamo deciso di soffermarci sulle informazioni meno banali.

2.1 - Account e Gestione_account

In relazione alla classe Gestione_account, descritta nel paragrafo 1.1, abbiamo definito i seguenti vincoli:

Logout:

Nella classe Gestione_account non è possibile invocare il metodo "logout" se non è presente nessun account associato.

```
context Gestione_account :: logout()
pre: account != null
```

Eliminazione dell'account:

Nella classe Gestione_account non è possibile invocare il metodo "elimina_account" se non è presente nessun account associato.

```
context Gestione_account :: elimina_account()
pre: account != null
```

Modifica della password:

Nella classe Account non è possibile invocare il metodo "modifica_password" se la nuova password è uguale alla vecchia password.

```
context Account :: modifica_password(new_password: string)
pre: Account.password != new_password
```

Modifica della mail:

Nella classe Account non è possibile invocare il metodo "modifica_mail" se la nuova mailè uguale alla vecchia mail.

```
context Account :: modifica_mail(new_mail: string)
pre: Account.mail != new_mail
```

Modifica del nome:
Nella classe Account non è possibile invocare il metodo “modifica_nome” se il nuovo nome è uguale al vecchio nome.

```
context Account :: modifica_nome(new_nome: string)
pre: Account.nome != new_nome
```

2.2 - Lista e Gestione_liste

In relazione alla classe Lista, descritta nel paragrafo 1.2, abbiamo definito i seguenti vincoli:

Rimozione di oggetti:
Nella classe Lista è possibile invocare il metodo “rimuovi_oggetto” solo se la lista non è vuota e se l’oggetto è effettivamente presente nella lista.

```
context Lista :: rimuovi_oggetto(oggetto)
pre: Lista.oggetti→notEmpty() AND Lista.oggetti → includes(oggetto)
```

In relazione alla classe Gestione_liste, descritta nel paragrafo 1.2, abbiamo definito i seguenti vincoli:

Eliminazione della lista:
Nella classe Gestione_liste è possibile eliminare una lista se la lista di liste non è vuota e se la lista è effettivamente presente nell’attributo “liste”.

```
context Gestione_liste :: elimina_lista(indice_lista_da_eliminare)
pre:  liste→notEmpty() AND indice_lista_da_eliminare <  liste→size()
```

Modifica della lista:
Nella classe Gestione_liste è possibile invocare il metodo “modifica_lista” solo se l’attributo “liste” non è vuoto e se l’indice della lista da modificare è minore della dimensione dell’attributo “liste”.

```
context Gestione_liste :: modifica_lista(indice_lista_da_modificare)
pre:  liste→notEmpty() AND indice_lista_da_modificare <  liste→size()
```

2.3 - Prodotto

Non abbiamo individuato vincoli particolari relativi alla classe Prodotto descritta nel paragrafo 1.3.

2.4 - Carrello, Gestione_storico_carrelli e Luogo_GoogleMaps

In relazione alla classe Carrello, descritta nel paragrafo 1.4, abbiamo definito i seguenti vincoli.

Rimozione di un prodotto:

Nella classe Carrello è possibile invocare il metodo “rimuovi_prodotto” solo se la lista di prodotti non è vuota e se il prodotto da eliminare è effettivamente presente in essa.

```
context Carrello :: rimuovi_prodotto(prod_da_eliminare)
pre: Carrello.prodotti→notEmpty() AND Carrello.prodotti → includes(prod_da_eliminare)
```

In relazione alla classe Gestione_storico_carrelli, descritta nel paragrafo 1.4, abbiamo definito i seguenti vincoli.

Eliminazione carrello:

Nella classe gestione_storico_carrelli è possibile eliminare un carrello solo se la lista di carrelli non è vuota e il carrello è effettivamente presente nella lista.

```
context Gestione_storico_carrelli :: elimina_carrello(carrello_da_eliminare)
pre:  Gestione_storico_carrelli.carrelli→notEmpty()
       AND Gestione_storico_carrelli.carrelli → includes(carrello_da_eliminare)
```

Modifica carrello:

Nella classe Gestione_storico_carrelli è possibile invocare il metodo “modifica_carrello” solo se l’attributo “carrelli” non è vuoto e se il carrello è effettivamente presente nella lista “carrelli”.

```
context Gestione_storico_carrelli ::  modifica_carrello(carrello_da_modificare)
pre: Gestione_storico_carrelli.carrelli→notEmpty()
      AND Gestione_storico_carrelli.carrelli → includes(carrello_da_modificare)
```

In relazione alla classe Luogo_GoogleMaps, descritta nel paragrafo 1.4, abbiamo definito i seguenti vincoli.

Valori delle coordinate:

Gli attributi latitudine e longitudine della classe Luogo_GoogleMaps devono avere valori rientranti nei limiti delle coordinate geografiche.

```
context Luogo_GoogleMaps :: latitudine
inv: latitudine >= -90 AND latitudine <= 90
```

```
context Luogo_GoogleMaps :: longitudine
inv: longitudine >= -180 AND longitudine<= 180
```

2.5 - Statistiche_prodotti

In relazione alla classe Statistiche_prodottii, descritta nel paragrafo 1.5, abbiamo definito i seguenti vincoli.

Impostazione intervallo:

Nella classe Statistiche_spese_totali è presente il metodo set_intervallo che prende come parametro una coppia di Tipo_data. Per eseguire questo metodo è necessario che la prima data sia antecedente alla seconda. Di seguito è riportato questo vincolo in codice OCL:

```
context Statistiche_spese_totali :: set_intervallo(intervallo)
pre: intervallo[0] < intervallo[1]
```

Selezione prodotto:

Nella classe Statistiche_spese_totali è presente il metodo seleziona_prodotto. Per selezionare un prodotto la lista deve necessariamente contenere almeno un elemento, e l'indice del prodotto selezionato non deve superare la dimensione della lista.

```
context Statistiche_prodotti :: seleziona_prodotto(indice)
pre:  prodotti→notEmpty() AND indice < prodotti→size()
```

2.6 - Statistiche_spese_totali

In relazione alla classe Statistiche_spese_totali, descritta nel paragrafo 1.6, abbiamo definito i seguenti vincoli.

Impostazione intervallo:

Lo stesso vincolo definito nella sezione 2.5 vale per il metodo set_intervallo della classe Statistiche_prodotto. Questa è la codifica in OCL:

```
context Statistiche_spese_totali :: set_intervallo(intervallo)
pre: intervallo[0] < intervallo[1]
```

2.7 Tipo_data

La classe ausiliaria Tipo_data presenta i seguenti vincoli:

Giorno:

L'attributo giorno deve essere compreso tra 1 e 31.

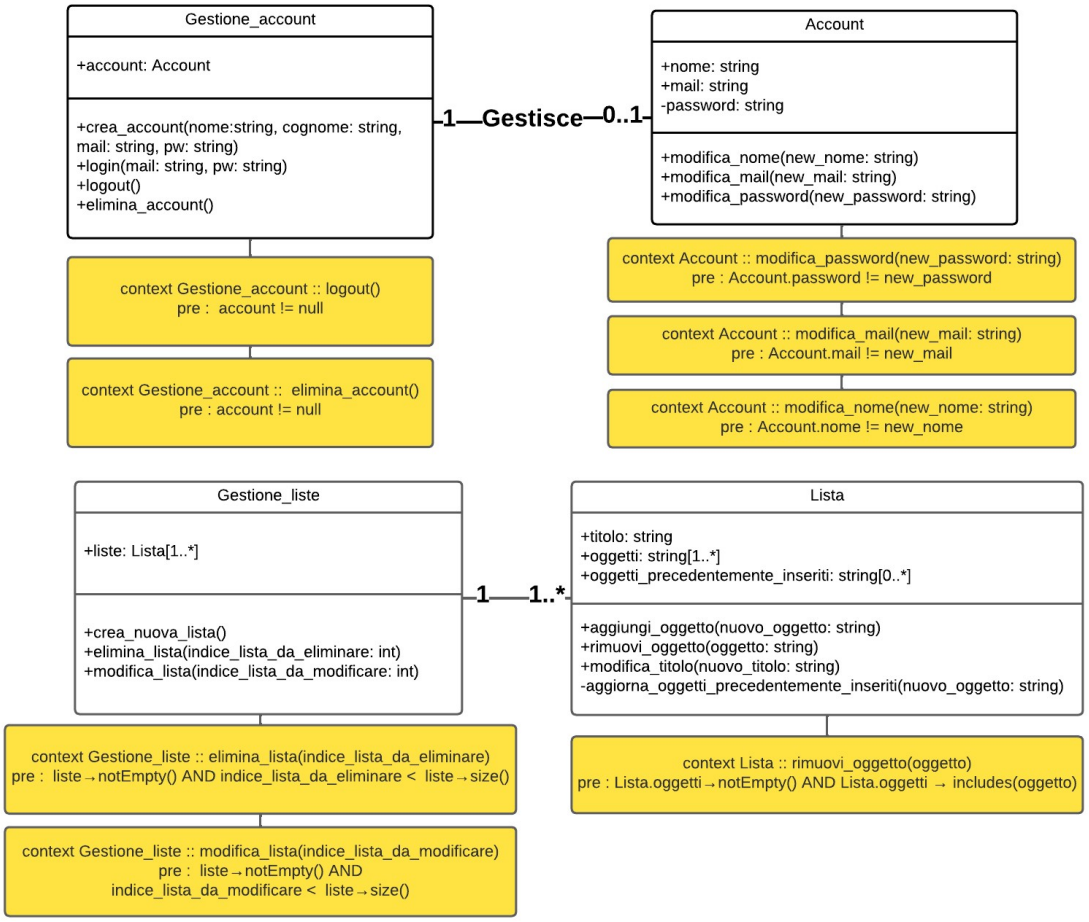
```
context Tipo_data :: giorno
inv: giorno >= 1 AND giorno <= 31
```

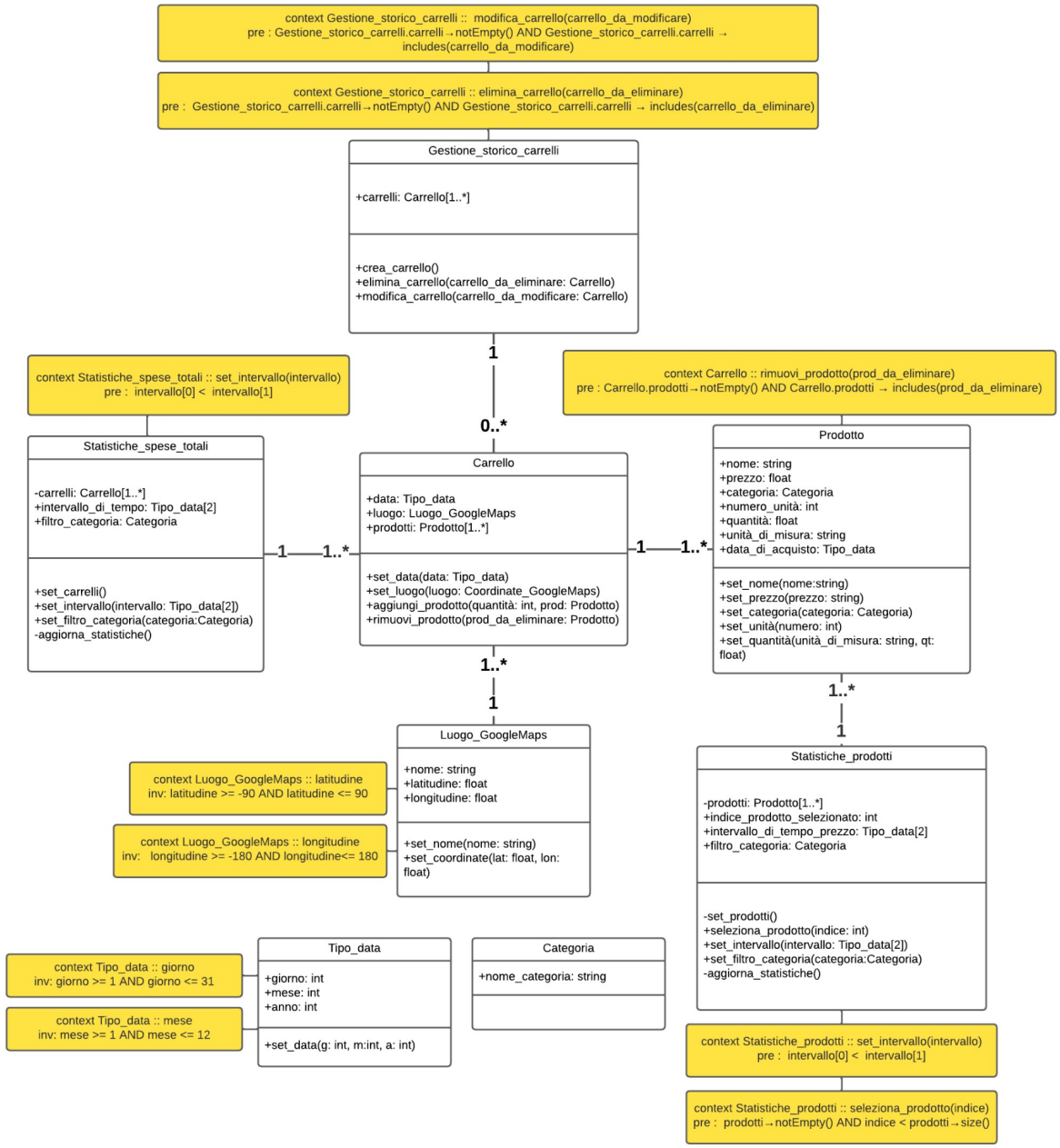
Mese:
L'attributo mese deve essere compreso tra 1 e 12.

```
context Tipo_data :: mese
inv: mese >= 1 AND mese <= 12
```

2.8 Diagramma delle classi complessivo

Riportiamo infine il diagramma delle classi con tutte le classi fino ad ora presentate ed il codice OCL individuato.




context Gestione_storico_carrelli :: modifica_carrello(carrello_da_modificare)
pre : Gestione_storico_carrelli.carrelli→notEmpty() AND Gestione_storico_carrelli.carrelli → includes(carrello_da_modificare)
context Gestione_storico_carrelli :: elimina_carrello(carrello_da_eliminare)
pre : Gestione_storico_carrelli.carrelli→notEmpty() AND Gestione_storico_carrelli.carrelli → includes(carrello_da_eliminare)
Gestione_storico_carrelli
+carrelli: Carrello[1..*]
+crea_carrello()
+elimina_carrello(carrello_da_eliminare: Carrello)
+modifica_carrello(carrello_da_modificare: Carrello)
1
0..*
context Statistiche_spese_totali :: set_intervallo(intervallo)
pre : intervallo[0] < intervallo[1]
context Carrello :: rimuovi_prodotto(prod_da_eliminare)
pre : Carrello.prodotti→notEmpty() AND Carrello.prodotti → includes(prod_da_eliminare)
Statistiche_spese_totali
-carrelli: Carrello[1..*]
+intervallo_di_tempo: Tipo_data[2]
+filtro_categoria: Categoria
+set_carrelli()
+set_intervallo(intervallo: Tipo_data[2])
+set_filtro_categoria(categoria:Categoria)
-aggiorna_statistiche()
1
1..*
Carrello
+data: Tipo_data
+luogo: Luogo_GoogleMaps
+prodotti: Prodotto[1..*]
+set_data(data: Tipo_data)
+set_luogo(luogo: Coordinate_GoogleMaps)
+aggiungi_prodotto(quantità: int, prod: Prodotto)
+rimuovi_prodotto(prod_da_eliminare: Prodotto)
1
1..*
Prodotto
+nome: string
+prezzo: float
+categoria: Categoria
+numero_unità: int
+quantità: float
+unità_di_misura: string
+data_di_acquisto: Tipo_data
+set_nome(nome:string)
+set_prezzo(prezzo: string)
+set_categoria(categoria: Categoria)
+set_unità(numero: int)
+set_quantità(unità_di_misura: string, qt: float)
1..*
1
1..*
1
Luogo_GoogleMaps
+nome: string
+latitudine: float
+longitudine: float
+set_nome(nome: string)
+set_coordinate(lat: float, lon: float)
context Luogo_GoogleMaps :: latitudine
inv: latitudine >= -90 AND latitudine <= 90
context Luogo_GoogleMaps :: longitudine
inv: longitudine >= -180 AND longitudine<= 180
Statistiche_prodotti
-prodotti: Prodotto[1..*]
+indice_prodotto_selezionato: int
+intervallo_di_tempo_prezzo: Tipo_data[2]
+filtro_categoria: Categoria
-set_prodotti()
+seleziona_prodotto(indice: int)
+set_intervallo(intervallo: Tipo_data[2])
+set_filtro_categoria(categoria:Categoria)
-aggiorna_statistiche()
Tipo_data
+giorno: int
+mese: int
+anno: int
+set_data(g: int, m:int, a: int)
Categoria
+nome_categoria: string
context Tipo_data :: giorno
inv: giorno >= 1 AND giorno <= 31
context Tipo_data :: mese
inv: mese >= 1 AND mese <= 12
context Statistiche_prodotti :: set_intervallo(intervallo)
pre : intervallo[0] < intervallo[1]
context Statistiche_prodotti :: seleziona_prodotto(indice)
pre : prodotti→notEmpty() AND indice < prodotti→size()