Progetto:

iSpesa

Titolo del documento:

Documento di architettura

Informazioni Documento

Nome Documento	*iSpesa_Architettura_D3*	Numero Documento	D3
Descrizione	Il documento include diagrammi delle classi e codice in OCL		

Indice

Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto iSpesa usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nel precedente documento è stato presentato il diagramma degli use case, il diagramma di contesto e quello dei componenti. Ora, tenendo conto di questa progettazione, viene definita l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un diagramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

1. Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto iSpesa. Ogni componente presente nel diagramma dei componenti diventa una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come le classi si relazionano tra loro.
Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti. In questo processo si è proceduto anche nel massimizzare la coesione e minimizzare l'accoppiamento tra classi.

1.1 Utenti e sistemi esterni

Analizzando il diagramma di contesto realizzato per il progetto iSpesa si nota la presenza di **tre attori "Utente, "UtenteRegistrato"** e **"Admin"**. L'**Utente** è colui che utilizza l'applicazione per accedere ai servizi di iSpesa, l'**UtenteRegistrato** è un tipo particolare di utente che dispone di maggiori funzionalità mentre l'**Admin** è chi modera e gestisce le funzioni del sito. Questi attori hanno specifiche funzioni e attributi ma hanno anche molto in comune. Sono state quindi individuate due classi *Admin* e *UtenteRegistrato* con funzioni e attributi specifici e una classe *Utente* con funzioni e attributi in comune collegate tramite una generalizzazione. Nella classe *UtenteRegistrato* sono state aggiunte anche le operazioni per aggiungere *Segnalazioni*, *Recensioni* e *gestire i prodotti preferiti*. Nella classe Admin sono state aggiunte le operazioni per *gestire le Segnalazioni, Recensioni e gli Utenti*.

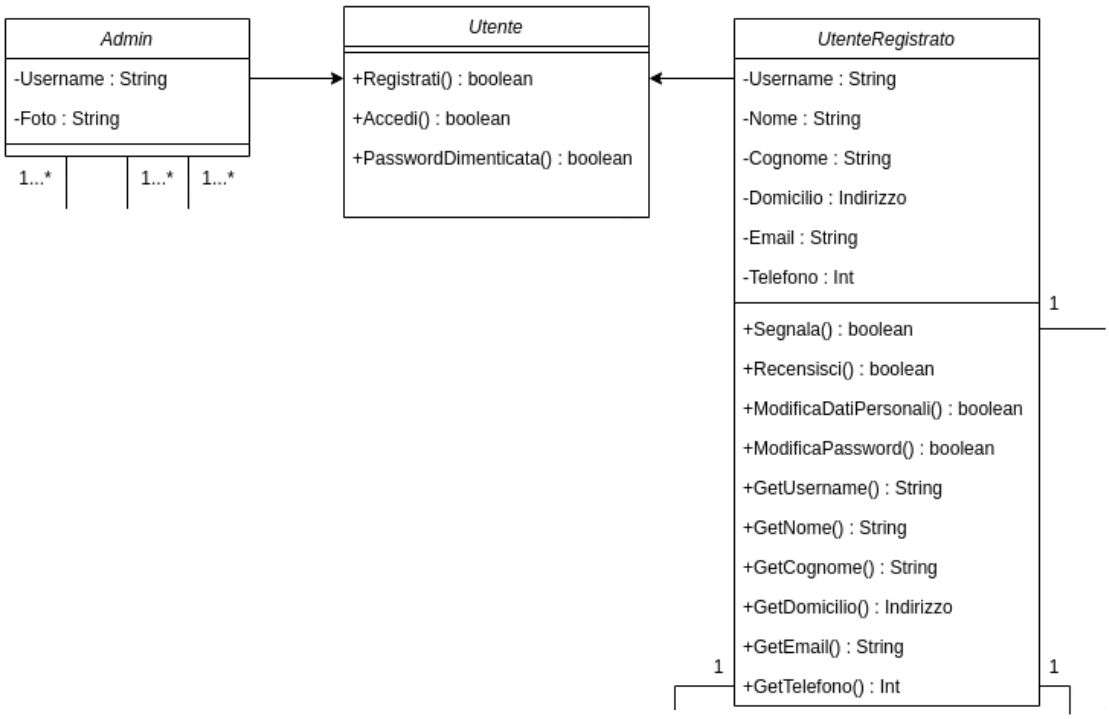

Figura 1. Classi per Utente, UtenteRegistrato e Admin

1.2 Gestione dell'aggiornamento del sito

Il diagramma di contesto presenta il sistema che permette al sito di aggiornarsi con i nuovi dati forniti dai negozi. La classe *Aggiornamento Sito* rappresenta il meccanismo di aggiornamento attraverso chiamate verso i vari negozi. Le classi *Gestione Prodotti*, *Gestione Volantini*, *Gestione Sconti* e *Gestione Negozi* prenderanno i propri dati da questa classe ogni 24 ore. Di seguito il dettaglio di queste classi con i propri attributi e metodi.

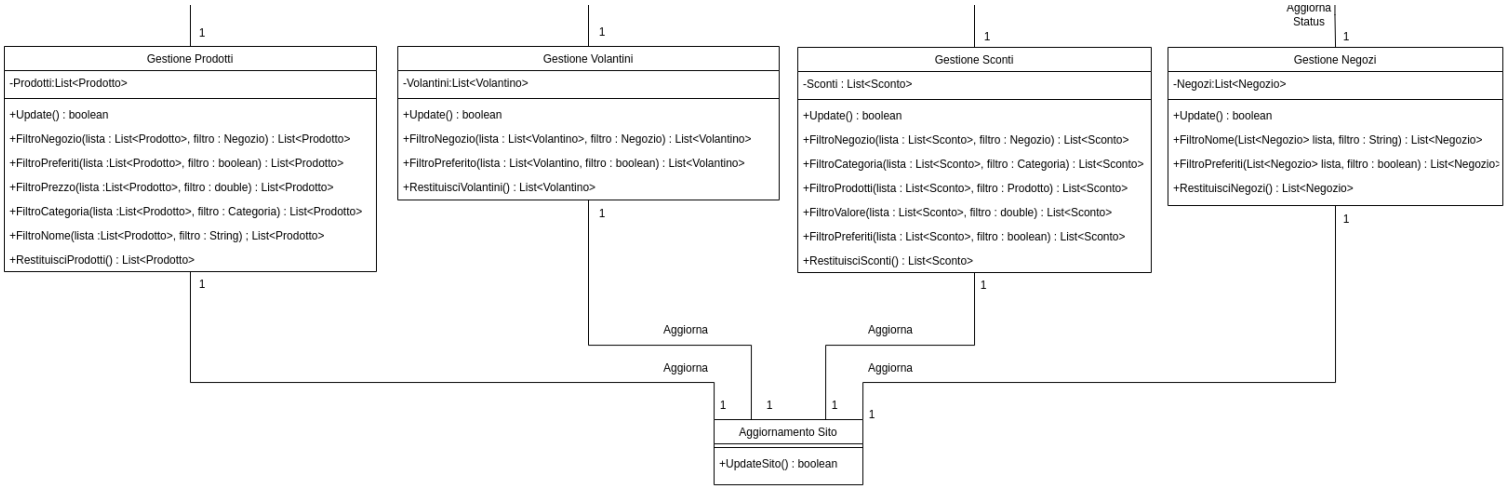

Figura 2. Classi per l'aggiornamento dei dati del sito

1.3 Gestione della visualizzazione del sito

Dopo aver memorizzato i dati necessari nelle classi della gestione del sito, i dati verranno visualizzati dall'utente. La classe Utente si interfaccia attraverso il sito con le classi *Prodotto*, *Volantino*, *Sconto* e *Negozio*. Queste classi riceveranno i loro dati dalle corrispettive classi *Gestione Prodotti*, *Gestione Volantini*, *Gestione Sconti* e *Gestione Negozi*. Di seguito il dettaglio di queste classi con i propri attributi e metodi.

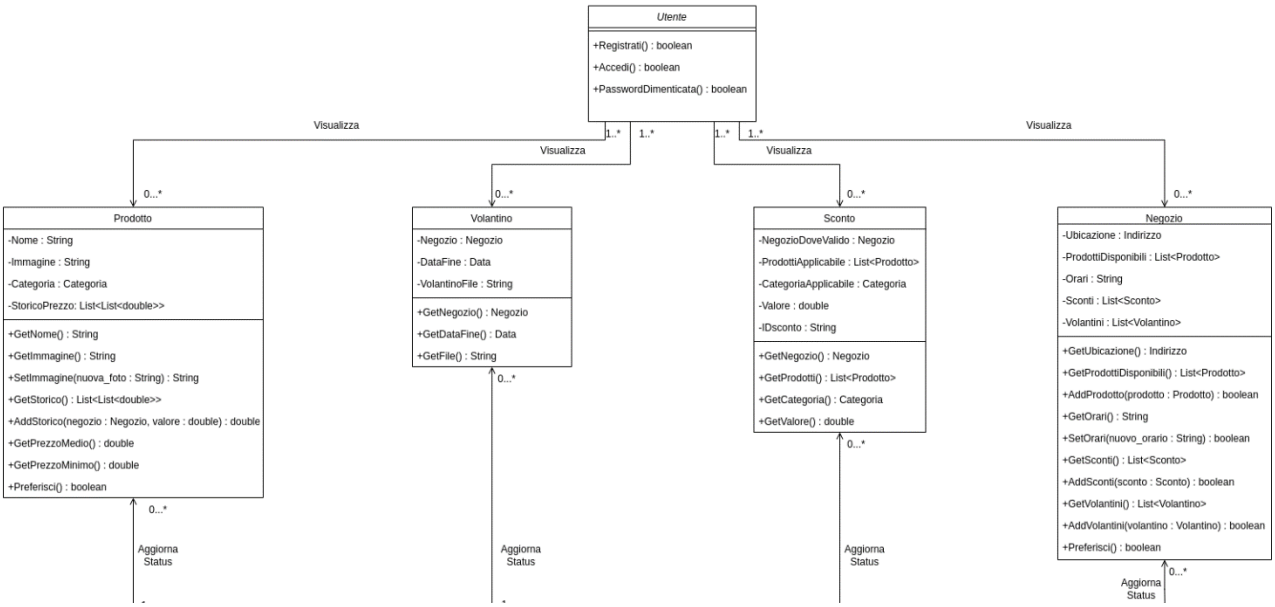

Figura 3. Classi per la visualizzazione del sito

1.4 Classe UtenteRegistrato

La classe *UtenteRegistrato* prevede la possibilità di inviare recensioni, segnalazioni e gestire una lista di preferiti. La classe *GestionePreferiti* contiene i *Negozi* e i *Prodotti* che sono stati scelti dall'*UtenteRegistrato*. Definiamo le classi che permettono all'utente la creazione di nuove recensioni, segnalazioni e la gestione dei preferiti con tutti gli attributi necessari.

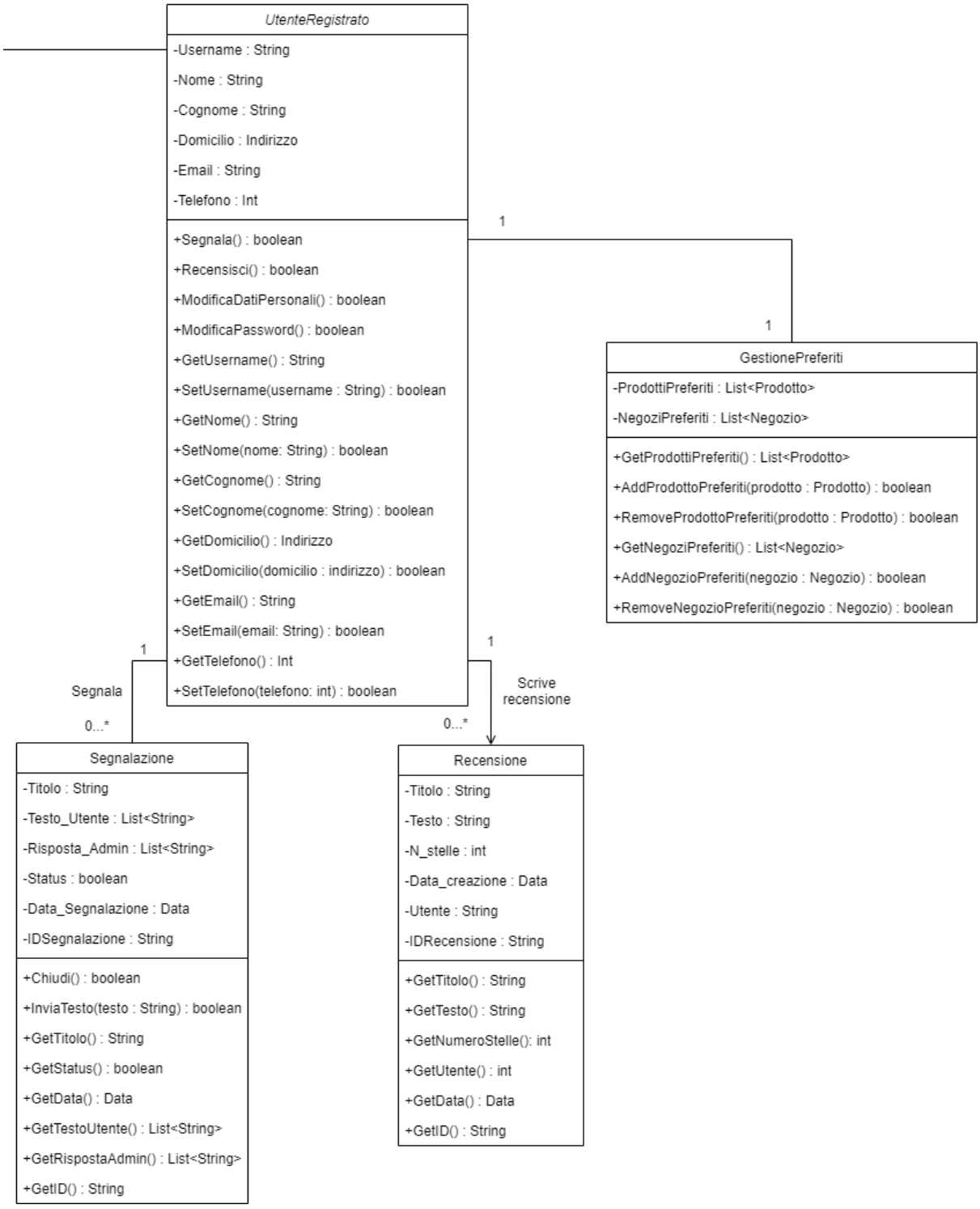

Figura 4. Classi per UtenteRegistrato

1.5 Classe Admin

La classe *Admin* prevede la possibilità di gestire le recensioni, segnalazioni e gli utenti. La classe *GestioneRecensioni* e *GestioneSegnalazioni* permette di eliminare le recensioni o segnalazioni. La classe *GestioneUtenti* permette di eliminare un utente, oscurarlo dagli altri utenti o di modificare la sua password su richiesta. Definiamo le classi che permettono all'admin di gestire le recensioni, le segnalazioni e gli utenti.

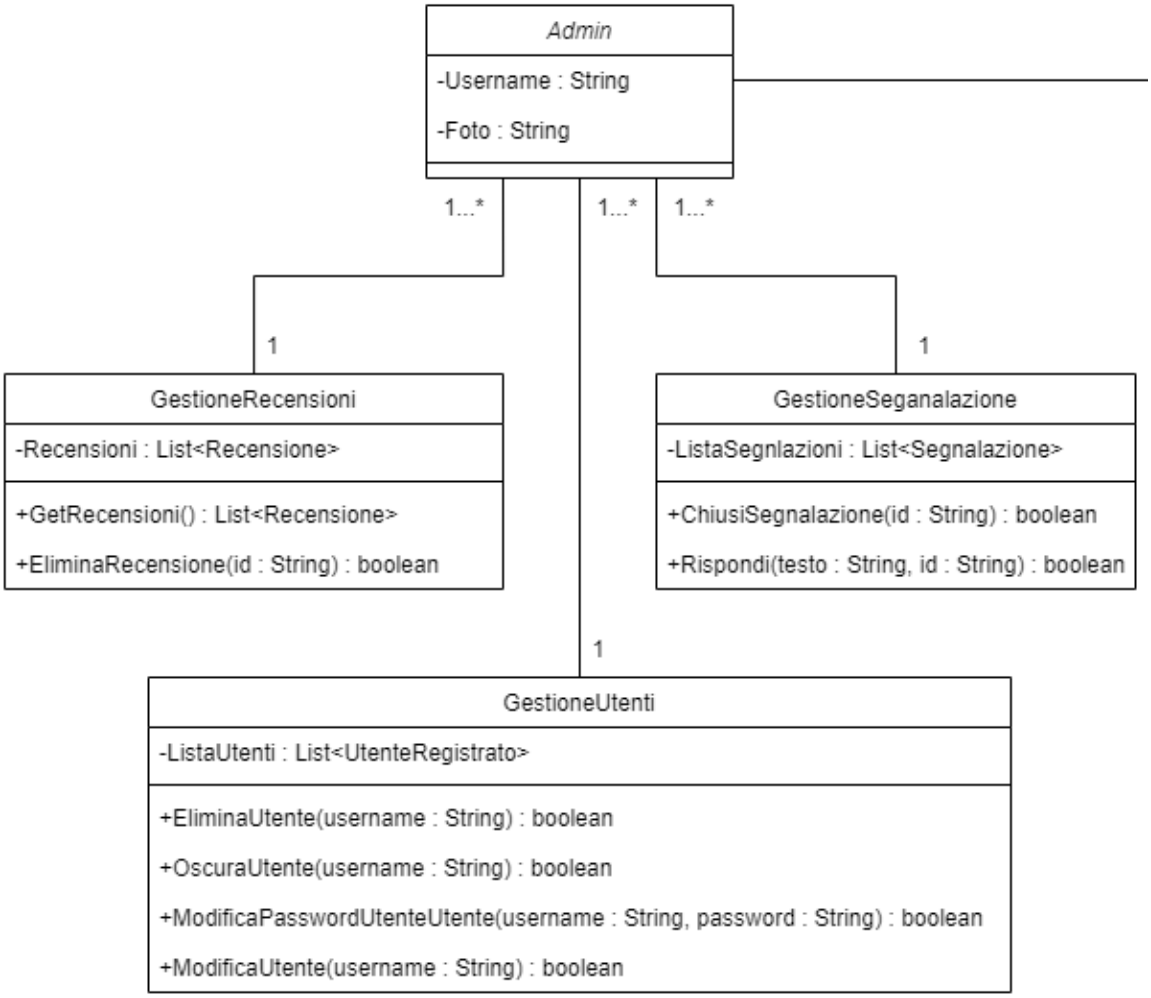

Figura 5. Classi per Admin

1.6 Classi di Supporto

Di seguito presentiamo le classi di supporto utilizzate nel progetto. La classe *Data* serve a memorizzare le date che verranno utilizzati per eventuali timer all'interno del progetto. Prevede un metodo *Passata()* che permette di controllare se la data in questione è già passata, oppure no. La classe *Indirizzo* ha lo scopo di permetterci di salvare gli indirizzi degli utenti o dei negozi in maniera precisa, permettendoci di ottenere singolarmente ogni dato dello stesso. La classe *Categoria* è un enumerativo che useremo per differenziare i vari prodotti in categorie differenti

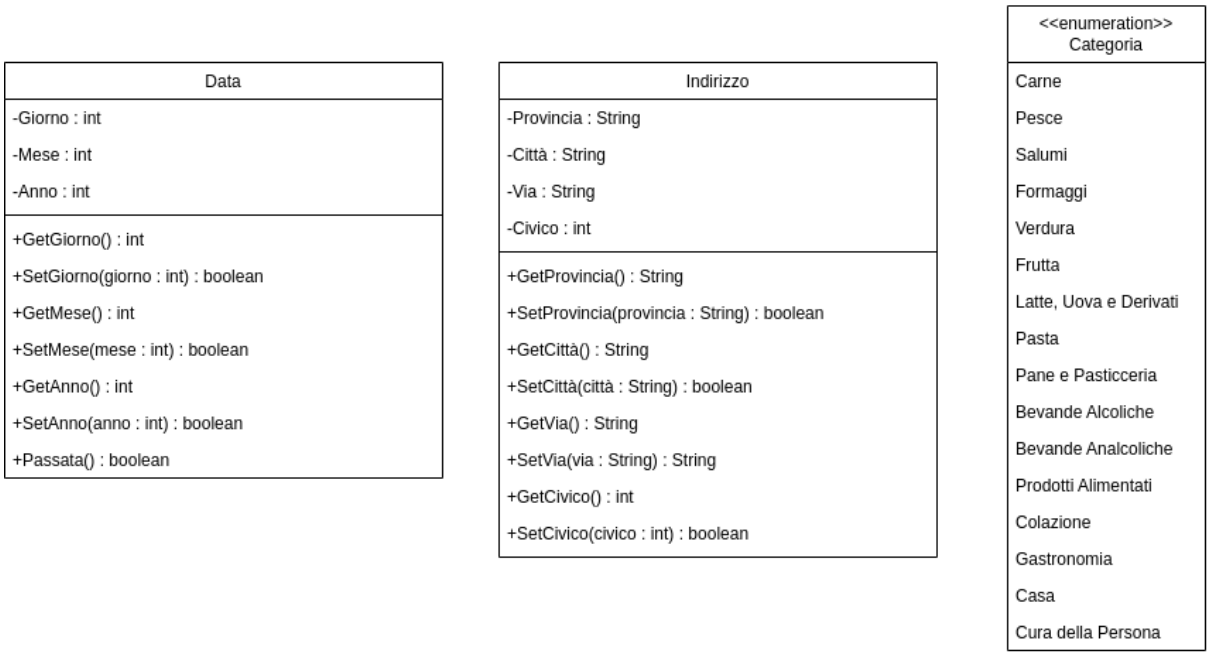

Figura 6. Diagramma delle classi di supporto

1.7 Diagramma 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 *Data* e *Indirizzo*. Queste classi servono per descrivere eventuali tipi strutturati usati, ad esempio, negli attributi delle altre classi.

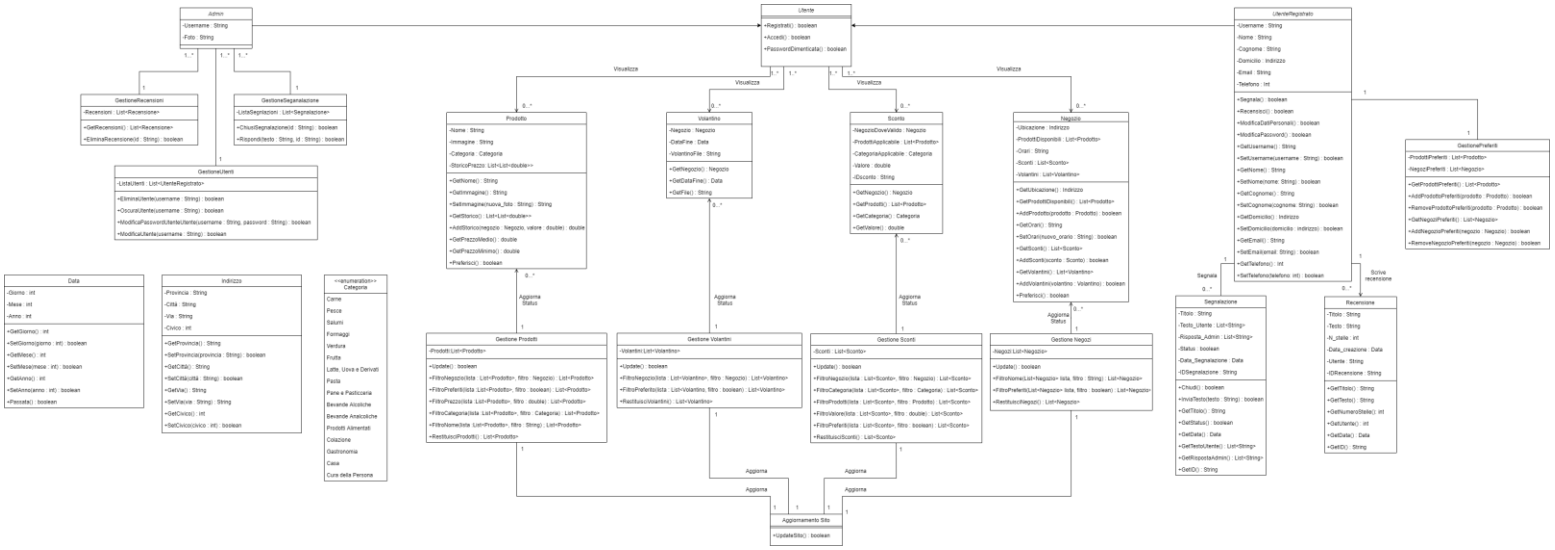

Figura 7. Diagramma delle classi complessivo

2. Codice in Object Constraint Language

In questo capitolo è descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL) perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

2.1 Aggiornamento sito

L'aggiornamento del sito inizia quando UpdateSito viene chiamata. Questa condizione su queste classi:

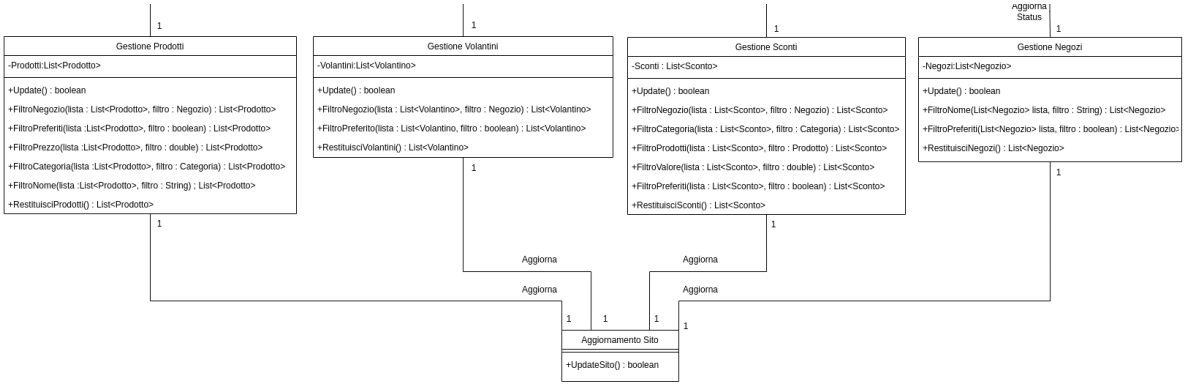

Figura 8. Classi relative all'aggiornamento dei dati

è espressa in OCL attraverso una precondizione con questo codice:

```
context AggiornamentSito::UpdateSito()
post: (GestioneProdotti.Update()) AND (GestioneVolantini.Update())
AND (GestioneSconti.Update()) AND (GestioneNegozi.Update())
```

2.2 Conferma valore sconto

Nella classe *Sconto* appena riportata deve essere sempre verificata una condizione. Il valore dello sconto deve variare da 1% a 99%. Questa condizione è espressa in OCL attraverso una invariante con questo codice:

```
context Sconto inv:
valore > 0 AND valore < 100
```

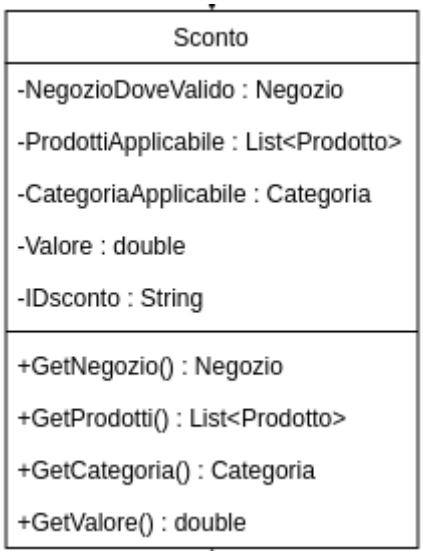

Figura 9. Classe Sconto

2.3 Risposta segnalazioni

Nella classe *GestioneSegnalazione* appartenente ad *Admin* appena riportata è presente anche il metodo Rispondi(testo, id). L'esecuzione di questo metodo comporta l'inizializzazione del valore dell'attributo Risposta_Admin nella Segnalazione. Questa condizione è espressa in OCL attraverso una postcondizione con questo codice:

```
context GestioneSegnalazione::Rispondi(testo, id)
post: Segnalazione.Risposta_Admin = testo
```

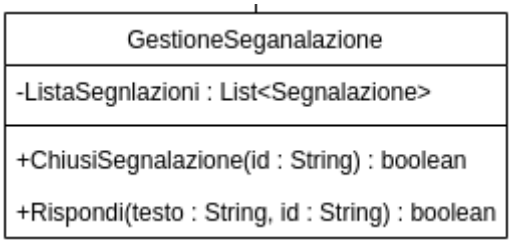

Figura 10. Classe GestioneSegnalazione

3. Diagramma delle classi con codice OCL

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

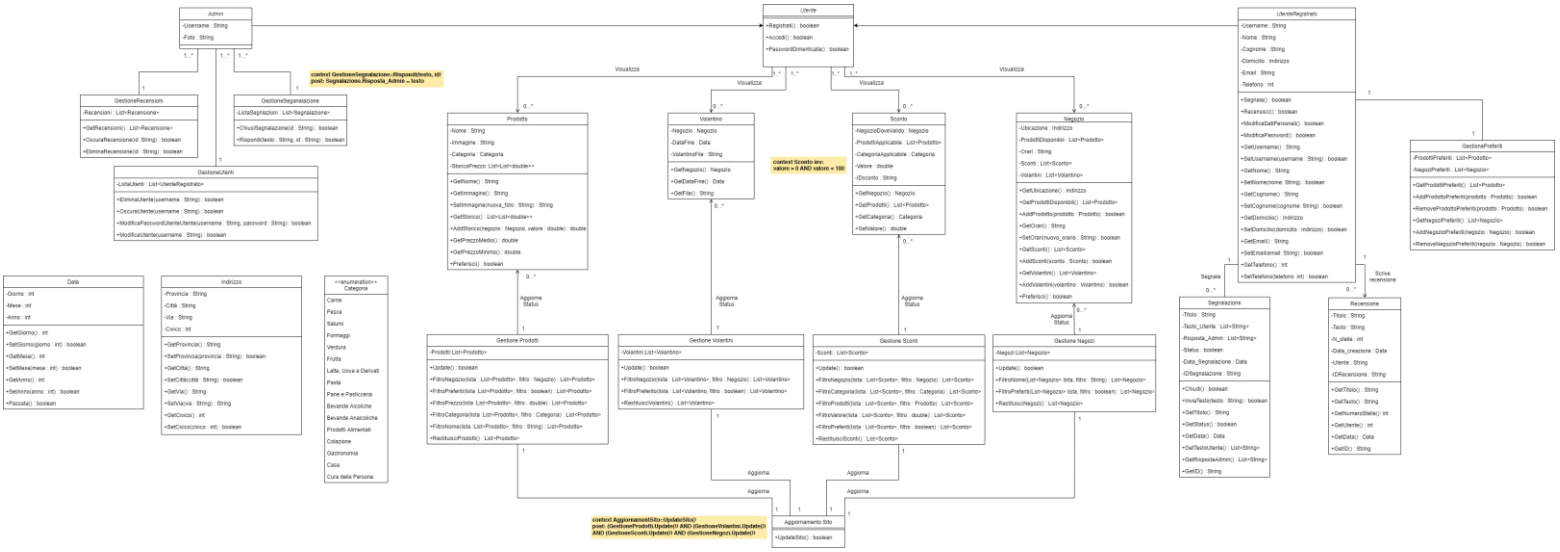

Figura 11. Diagramma delle classi con codice OCL