UNIVERSITÀ DI TRENTO

JustCook

10/11/2022

DOCUMENTO DI ARCHITETTURA

UNIVERSITÀ DI TRENTO

Indice

SCOPO DOCUMENTO

Questo documento ha lo scopo di definire l'architettura dell'applicazione Web JustCook. Per farlo viene utilizzato un diagramma delle classi in Unified Modeling Language (UML). Successivamente viene presentato lo stesso diagramma con l'aggiunta del codice in Object Constraint Language (OCL).

Questo documento tiene conto del documento precedente contenente i diagrammi degli Use Case, di contesto e delle componenti. In base a quest'ultimi si definisce l'architettura del sistema definendo nel modo più completo e dettagliato le classi che verranno implementate successivamente. Inoltre attraverso OCL si evidenzia nel suo insieme la logica che regola il comportamento del software.

1.DIAGRAMMA DELLE CLASSI

In questo paragrafo viene presentato il diagramma delle classi in Unified Modeling Language (UML). È stato realizzato partendo dal diagramma delle componenti. Quest'ultime sono rappresentate da una o più classi.

Le classi sono caratterizzate da un nome che le contraddistingue, da degli attributi che specificano i tipi di dati trattati e dai metodi che indicano il tipo di operazioni che possono eseguire. Inoltre vengono specificate diverse associazioni che forniscono informazioni sulle relazioni tra le varie classi.

Di seguito partendo da ogni componente vengono presentate le classi che le identificano.

1.1. DESCRIZIONE CLASSI

1. GESTIONE HOME

   Analizzando la componente **Gestione home** è stata individuata la classe `PaginaHome`. Si occupa di gestire il tutorial e mostrare all'utente la lista dei risultati della ricerca delle ricette. Dalla figura 1 si osserva che è sottoclasse di `PaginaWeb`. Inoltre nella figura 1 è presente anche la classe `Tutorial` che rappresenta il tutorial di JustCook.

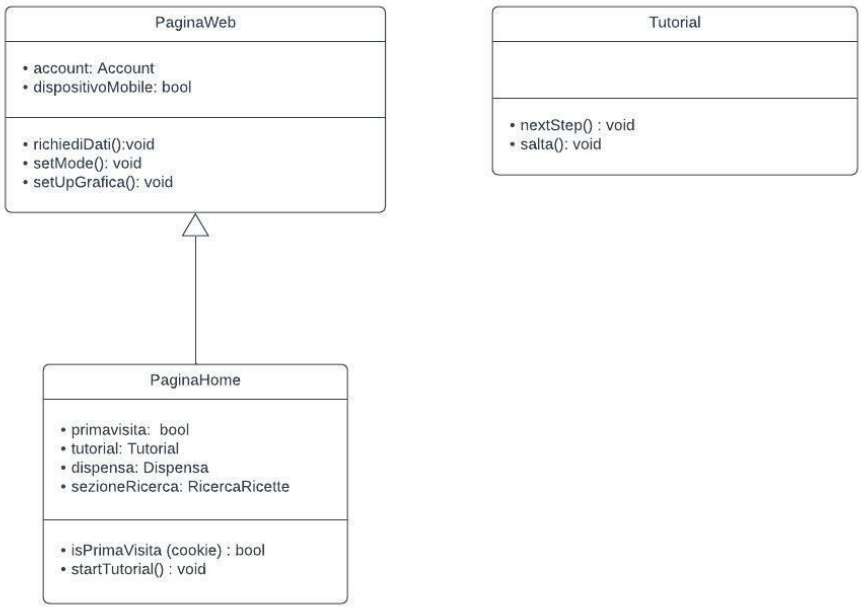


Figura 1: Classi `PaginaWeb, PaginaHome, Tutorial`

2. GESTIONE RICERCA RICETTE

Analizzando la componente **Gestione ricerca ricette** è stata individuata la classe `RicercaRicette`. Si ricorda che questa componente si occupa di fornire all'utente la sezione dove si ricercano le ricette con la possibilità di selezionare i filtri (per cui è stata creata la classe `Filtri`) e parametri di ordinamento. Inoltre si ricorda che questa componente fornisce a **Gestione home** la lista dei risultati delle ricette, individuati dalla classe `RisultatiRicette`.

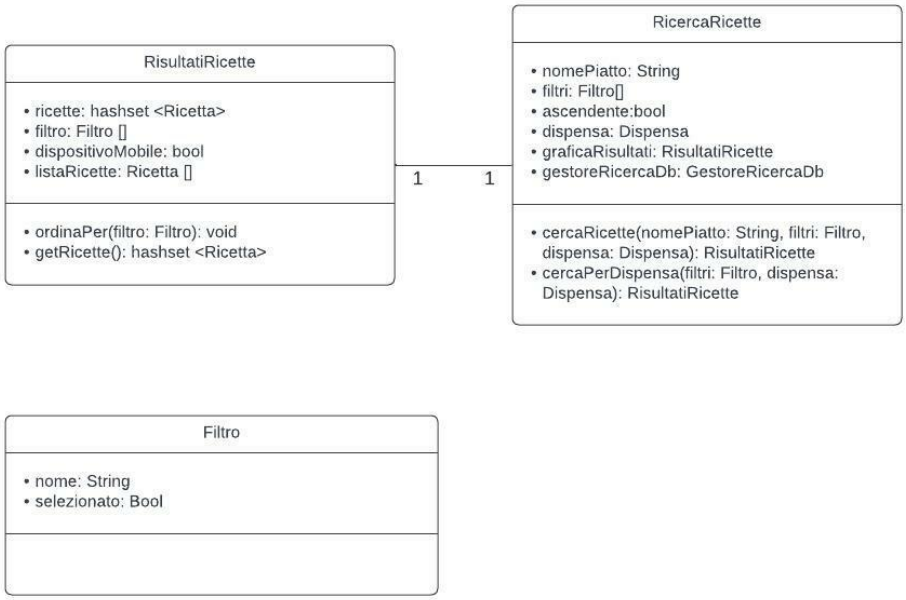


Figura 2: Classi `RisultatiRicette, RicercaRicette, Filtro`

3. GESTIONE RICERCA NEL DATABASE

La componente **Gestione ricerca nel db** si occupa di fornire i dati richiesti quando si effettua una ricerca di una ricetta o di un ingrediente (per cui è stata creata la classe `Ingrediente`) per aggiungerlo alla dispensa. È rappresentata dalla classe `GestoreRicercaDb`.

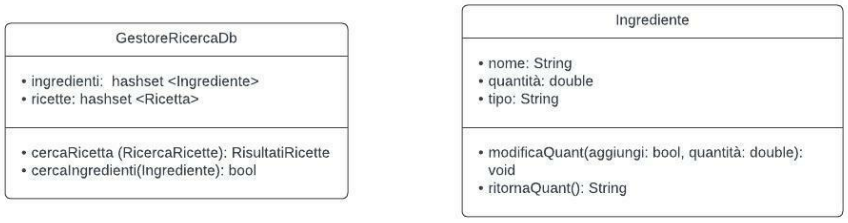


Figura 3: Classi `GestoreRicercaDb, Ingrediente`

4. GESTIONE DISPENSA

La componente **Gestione dispensa** si occupa di offrire all'utente la possibilità di aggiungere alla dispensa degli ingredienti e la loro quantità, cercandoli attraverso una barra di ricerca. Inoltre fornisce all'utente la possibilità di eliminarli. Tutte queste funzionalità vengono svolte dalla classe `Dispensa`, alla quale è associata anche la classe `DispensaGrafica` che si occupa della grafica della pagina della dispensa..

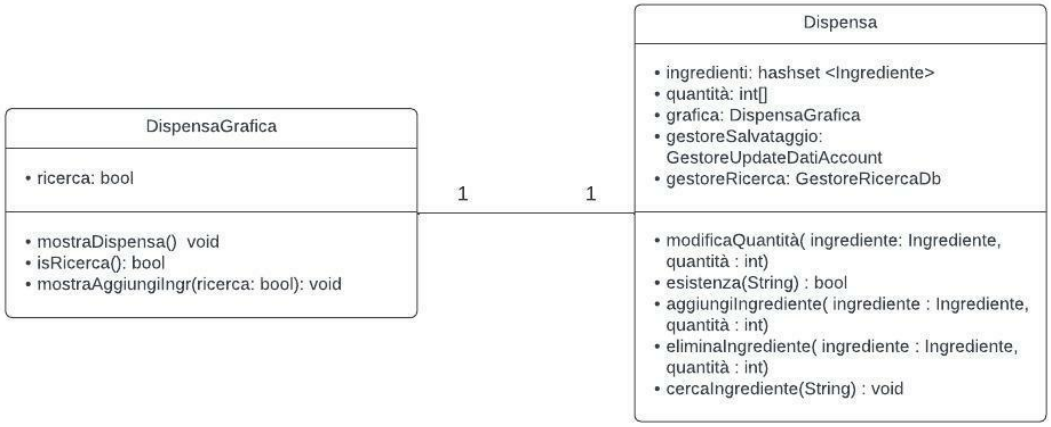


Figura 4: Classi `Dispensa, DispensaGrafica`

5. GESTIONE RICERCA PAGINE

Gestione ricerca pagine si occupa di passare i dati dal database alle pagine principali di JustCook. Per questa componente è stata individuata la classe `GestioneRicercaPagine`. Nella figura 6 si osserva anche la presenza della superclasse `PaginaWeb` e delle sue sottoclassi `PaginaRicetta, PaginaAccount, PaginaLogin, PaginaHome, PaginaRegistrazione`. Quest'ultime identificano tutte le pagine del sito, tra cui quelle che ricevano i dati dalla componente **Gestione ricerca pagine**.

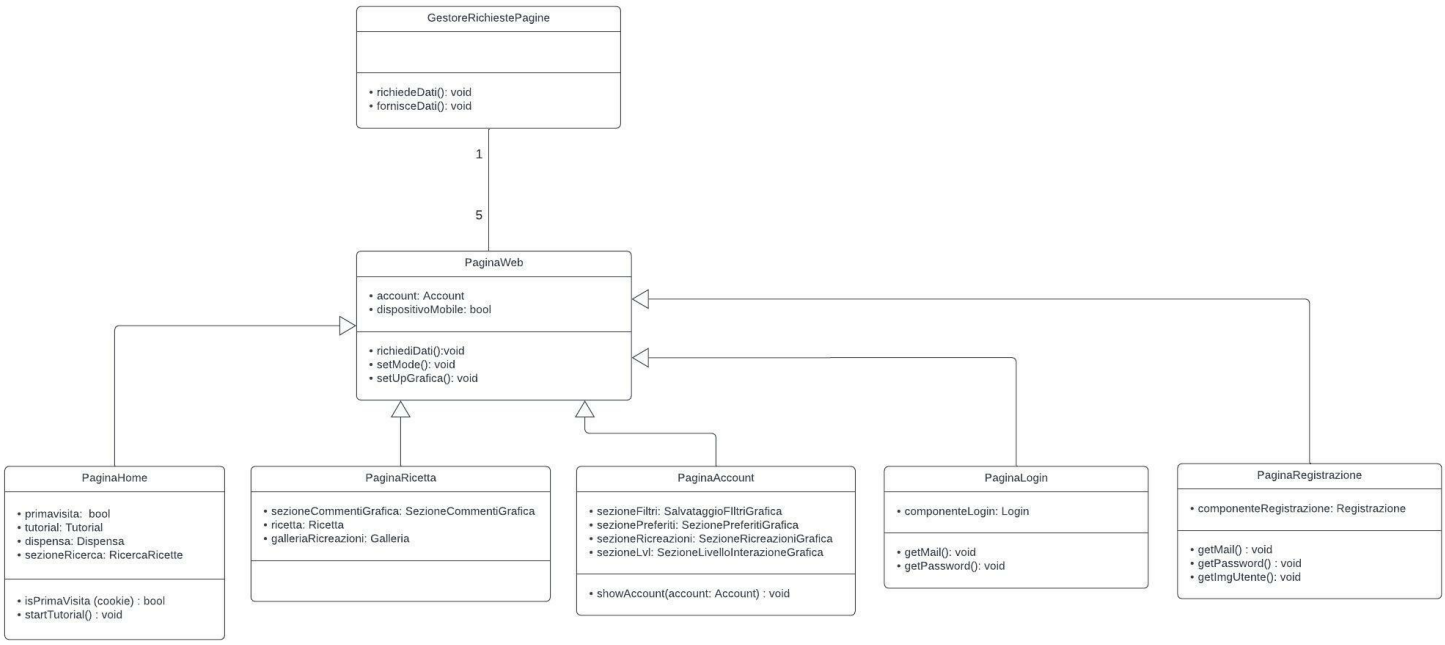


Figura 5: Classi `GestoreRichiestePagine`, `PaginaWeb`, `PaginaHome`, `PaginaRicetta`, `PaginaAccount`, `PaginaLogin`, `PaginaRegistrazione`

6. GESTIONE AUTENTICAZIONE

Analizzando **Gestione autenticazione** è stata individuata la classe `Login`. Si ricorda che questa componente si occupa di far accedere l'utente al proprio account. Inoltre fornisce la possibilità di cambiare password inserendo l'indirizzo email e la nuova password. Ovviamente l'utente per accedere deve inserire le proprie credenziali (username, password), individuate dalla classe `Credenziali`. Nella figura 7 si osserva che la grafica della pagina del login è gestita da `PaginaLogin`, sottoclasse di `PaginaWeb`.

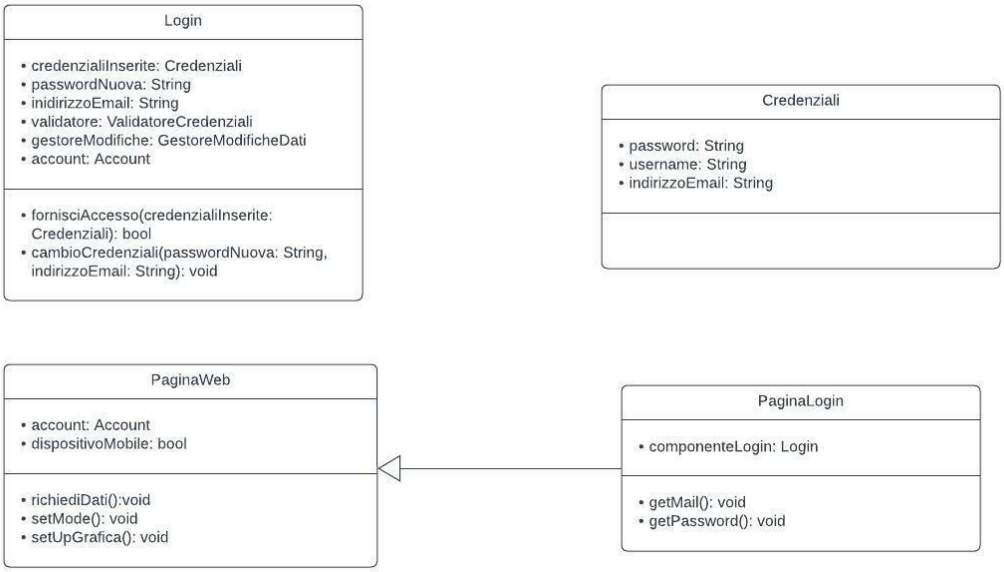


Figura 6: Classi `Login, PaginaWeb, PaginaLogin, Credenziali`

7. GESTIONE REGISTRAZIONE

Gestione registrazione si occupa della creazione degli account degli utenti. La classe che la rappresenta è `Registrazione`. Come nel punto precedente, c'è una classe che si occupa della grafica: `PaginaRegistrazione`, sottoclasse di `PaginaWeb`.

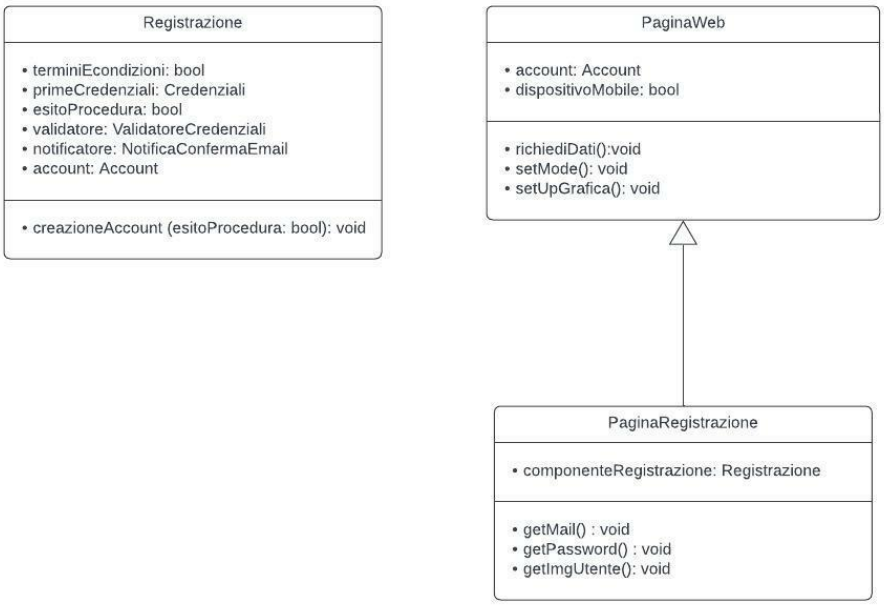


Figura 7: Classi `Registrazione, PaginaRegistrazione, PaginaWeb`

8. GESTIONE VALIDITÀ CREDENZIALI

Gestione validità credenziali si occupa di validare i dati quando si crea un account, quando si vuole modificare la password durante il login e quando si vogliono cambiare i dati dalla propria pagina dell'account. Per svolgere queste azioni è stata realizzata la classe `ValidatoreDati`.

Figura 8: Classe `ValidatoreCredenziali`

9. GESTIONE MODIFICA DATI ACCOUNT

Analizzando la componente **Gestione modifica dati account** è stata individuata la classe `GestoreModificaDati`. Quest'ultima si occupa di gestire i dati da modificare quando si vogliono cambiare dalla propria pagina dell'account o dalla pagina del login.

Figura 9: Classe `GestoreModificaDati`

10. GESTIONE VISUALIZZAZIONE DATI ACCOUNT

Gestione visualizzazione dati account si occupa di fornire all'utente la sezione riguardante i propri dati (immagine profilo, username, password, indirizzo email associato). Quest'ultima comprende anche la possibilità di modificarli. L'esito di tale operazione viene fornita all'utente nella pagina dell'account. La classe che individua tutte queste funzionalità è `Account`. Dalla figura 10 si osserva che `Account` è associata a `SezioneRicreazioni, SezionePreferiti, SezioneLivelloInterazione, SalvataggioFiltri`. Quest'ultime identificano le sezioni della pagina dell'account. A queste sono associate le classi `SezioneRicreazioniGrafica, SezionePreferitiGrafica, SalvataggioFiltriGrafica, SezioneLivelloGrafica` che si occupano della grafica delle rispettive sezioni. Inoltre si osserva la classe `AccountAmministratore`. Rispetto alla sua superclasse `Account` è associata anche a `SezioneValutazioneRicette`. Infine la pagina dell'account ha la propria grafica, individuata da `PaginaAccount`, sottoclasse di `PaginaWeb`.

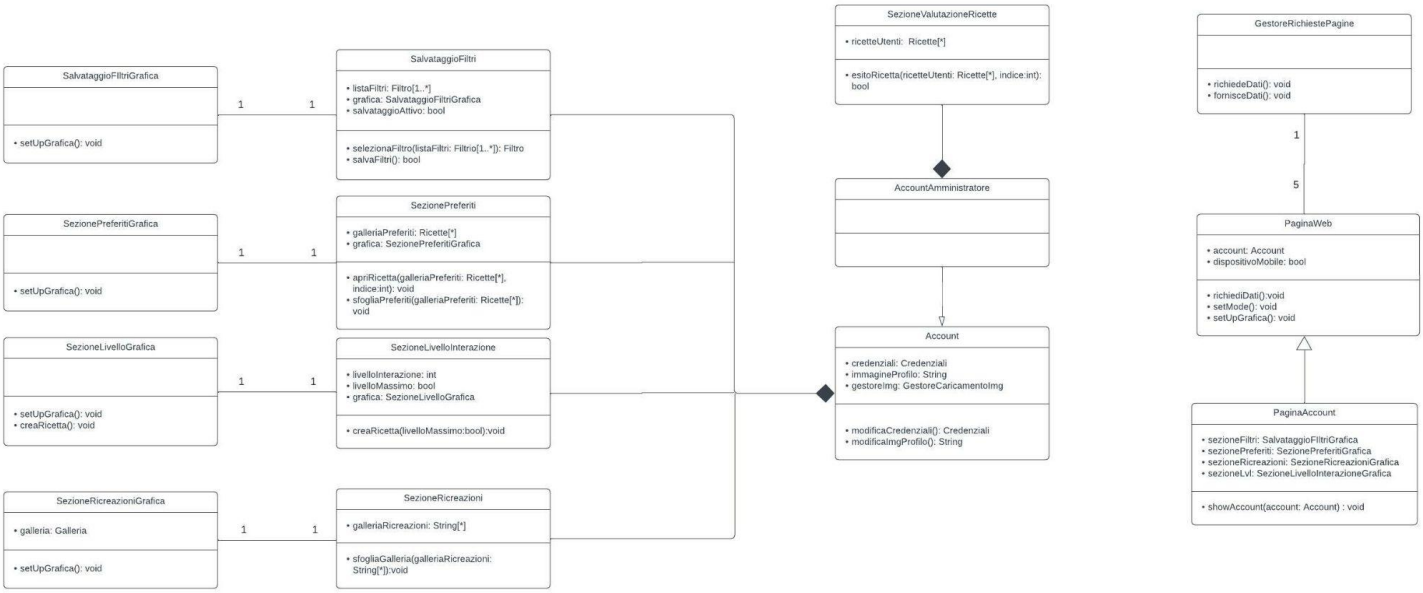


Figura 10: Classi `Account, SalvataggioFiltri, SezionePrefriti, SezioneLivelloInterazione, SezioneRicreazioni, SalvataggioFiltriGrafica, SezioneLivelloGrafica, SezioneRicreazioneGrafica, AccountAmministratore, SezioneValutazioneRicette, PaginaWeb, PaginaAccount`

11. GESTIONE NUOVE RICETTE

La componente **Gestione nuove ricette** si occupa di dirigere tutte le operazioni che riguardano la creazione di nuove ricette. Fornisce all'utente i moduli per inviare ricette e fornisce agli amministratori i moduli con le ricette da valutare. Per questo è stata individuata la classe `GestoreNuoveRicette`.

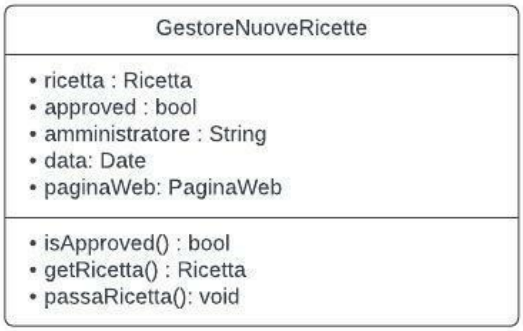


Figura 11: Classe `GestoreNuoveRicette`

12. GESTIONE NOTIFICA AMMINISTRATORI

Analizzando la componente **Gestione notifica amministratori** è stata individuata la componente `NotificatoreAmministratori`. Si occupa di regolare l'interazione tra il sistema e Gmail API quando bisogna notificare l'amministratore delle segnalazioni dei commenti per un'eventuale eliminazione e delle nuove ricette degli utenti da valutare.

Figura 12: Classe `NotificatoreAmministratori`

13. GESTIONE NOTIFICA UTENTI

Analizzando **Gestione notifica utenti** si nota che questa componente si occupa di interagire con Gmail API per mandare degli avvisi all'utente, per verificare l'esistenza e la conformità dell'indirizzo email inserito nella fase di registrazione e modifica dei dati dell'account. Inoltre per la registrazione e per la modifica della password in fase di login è necessaria un'email di conferma per completare l'operazione. Per gestire al meglio queste funzionalità è stata realizzata la classe `NotificatoreUtenti` e la sua sottoclasse `NotificatoreConfermaEmail`.

Figura 13: Classi `NotificatoreUtenti, NotificatoreConfermaEmail`

14. GESTIONE INTERAZIONE CON COMMENTI

La componenete **Gestione interazione con commenti** si occupa di fornire all'utente la possibilità di creare un commento (con possibilie foto allegata), eliminarlo (solo amministartori) e di segnalarlo. Per questo è stata ideata la classe `InterfacciaCommento`. La grafica è gestita da `InterfacciaCreaCommento`. Inoltre per le segnalazioni è stata creata la classe `Segnalazione`.

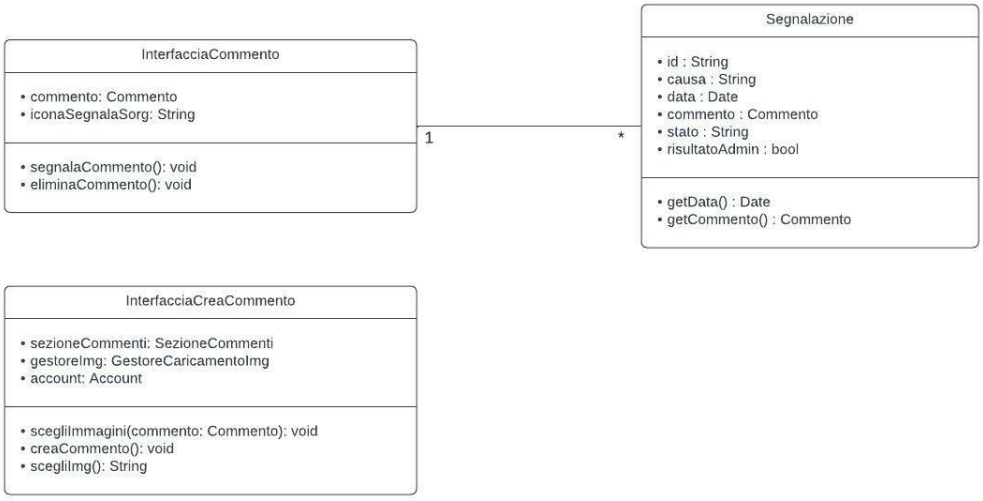


Figura 14: Classi `InterfacciaCommento, Segnalazione, InterfacciaCreaCommento`

15. GESTIONE AGGIORNAMENTO COMMENTI

Analizzando **Gestione aggiornamento commenti** è stata ideata la classe `GestioneUpdateCommenti`. Questa componente si occupa di prendere in considerazione tutte le possibili modifiche (creazione, eliminazione) e di avvissare le altre componenti che hanno bisogno di un aggiornamento “real time” (livello interazione, pagina ricetta).

Figura 15: Classe `GestoreUpdateCommenti`

16. GESTIONE SEZIONE COMMENTI

La componente **Gestione sezione commenti** si occupa di fornire all’utente la sezione dei commenti, compresa la galleria delle ricreazioni. Per questo è stata ideata la classe `SezioneCommenti`. Nella figura 16 si osserva che per i commenti è stata creata una classe `Commento`, con la rispettiva `CommentoGrafica` che si occupa della grafica. Anche `SezioneCommenti` ha una classe che si occupa della grafica, ovvero `SezioneCommentiGrafica`.

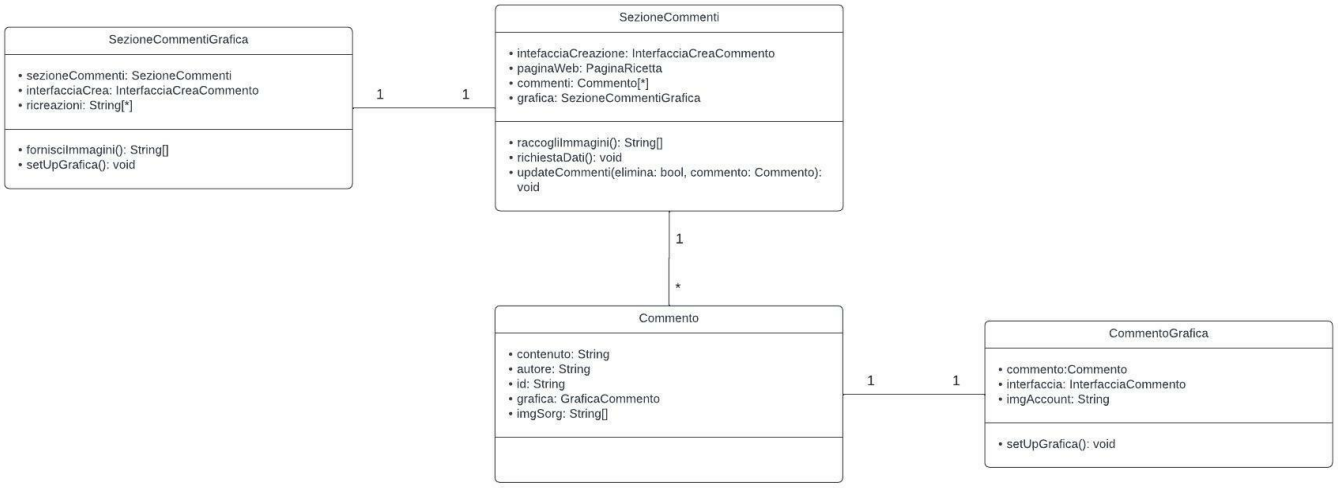


Figura 16: Classi `SezioneCommenti, SezioneCommentiGrafica, Commento, CommentoGrafica`

17. GESTIONE PREFERITI E RICREAZIONI

Analizzando la componente **Gestione preferiti e ricreazioni** sono state individuate le classi `SezionePreferiti, SezionePrefritiGrafica` (parte grafica), `SezioneRicreazioni, SezioneRicreazioniGrafica` (parte grafica). Questa componente si occupa della sezione delle immagini allegate ai commenti e della sezione contente le ricette salvate come preferite. Nella figura 17 si osserva che le classi sopra elencate fanno parte della classe `Account`.

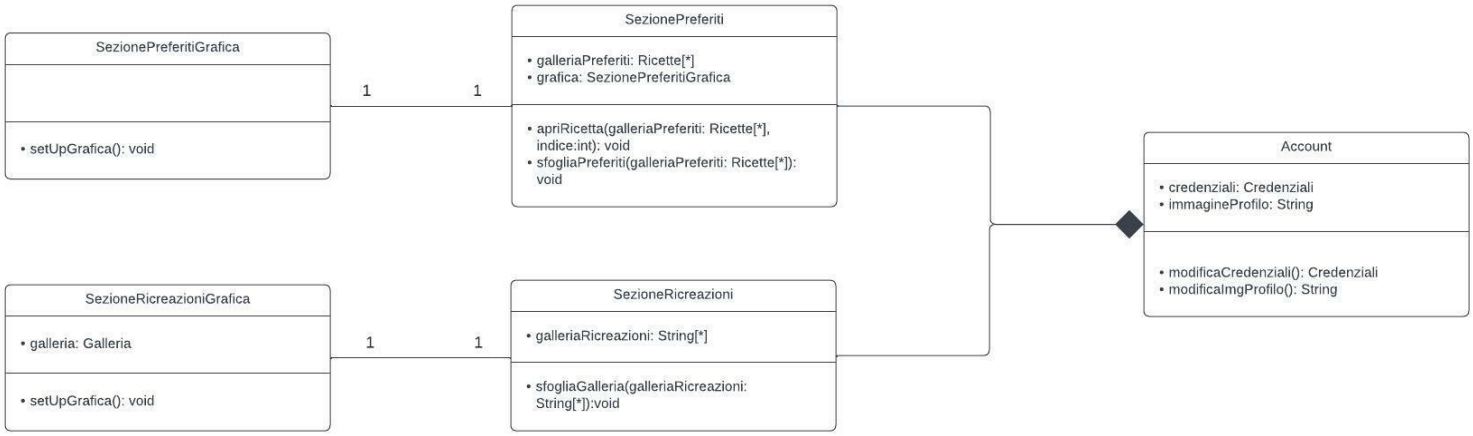


Figura 17: Classi `SezionePreferiti, SezionePreferitigrafica, SezioneRicreazioni, SezioneRicreazioniGrafica, Account`

18. GESTIONE CARICAMENTO IMMAGINI

La componente **Gestione caricamento immagini** si occupa di interagire con File System API quando un utente vuole caricare, modificare la propria immagine profilo e quando vuole allegare le ricreazioni ai commenti. Per questo è stata individuata la classe `GestoreCaricamentoImg`.

Figura 18: Classe `GestoreCaricamentoImg`

19. GESTIONE VISUALIZZAZIONE INFO RICETTA

Analizzando Gestione visualizzazione info ricetta sono state individuate le classi `Ricetta` e la sua sottoclasse `RicettaEstesa`. Questa componente si occupa di fornire all'utente tutte le informazioni rigurdanti le ricette (ingredienti, procedimento, rating, filtri) e la possibilità di poterle completare, valutare, aggiungere tra i preferiti. Nella figura 19 si osserva che per i passaggi del procedimento sono state create le classi `Passo, PassoGrafica` (per la grafica). Inoltre per le informazioni riguardanti i filtri è stata identificata la classe `Statistica`.

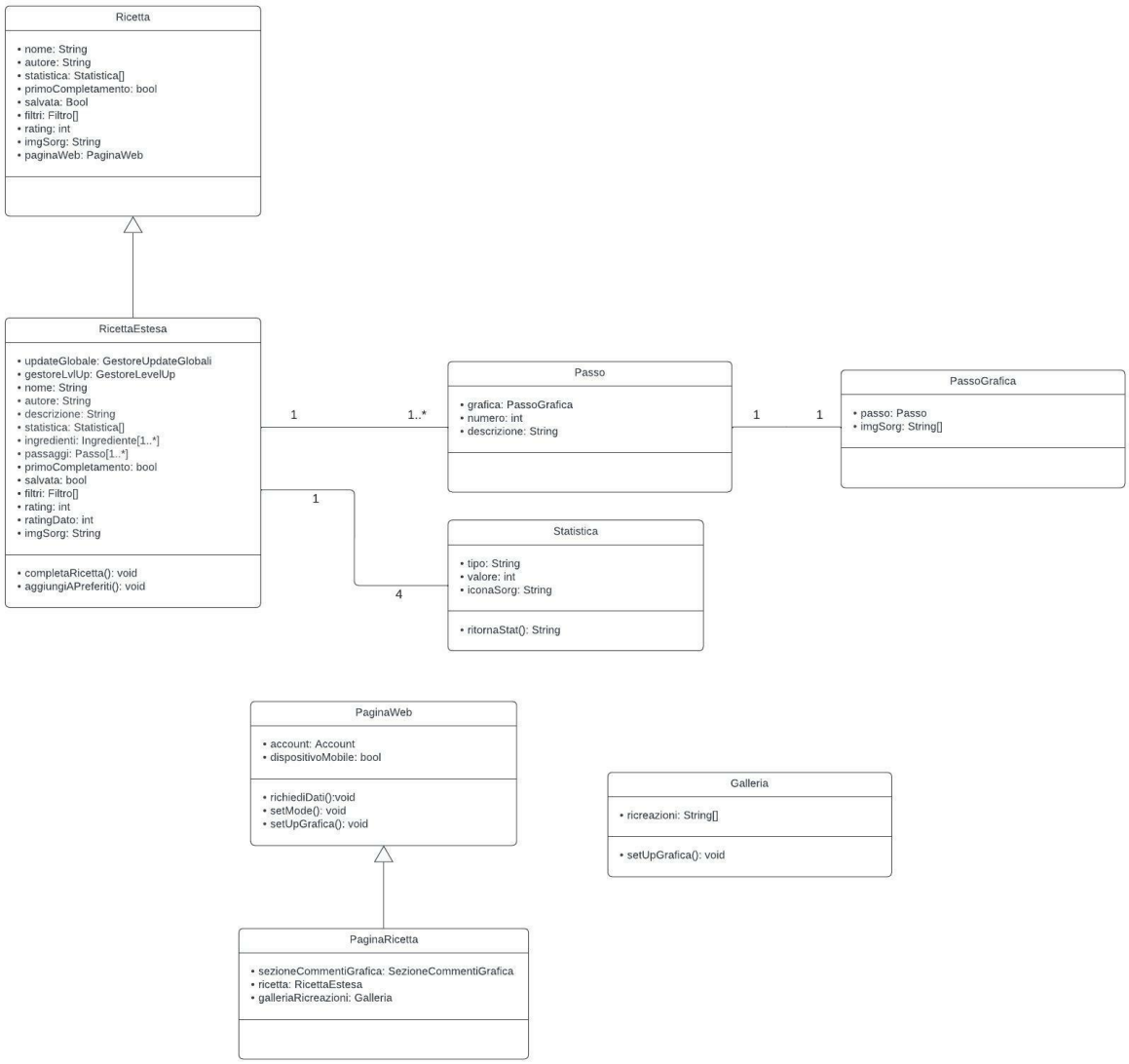


Figura 19: Classi `Ricetta, RicettaEstesa, Passo, PassoGrafica, Statistica, PaginaRicetta, PaginaWeb, Galleria`

20. GESTIONE AGGIORNAMENTI GLOBALI SU DB

La componente Gestione aggiornamenti globali su database si occupa di interagire con il database inviandogli i dati da aggiornare quando le ricette vengono valutate o quando vengono modificati i commenti (eliminati, creati). Per svolgere queste funzioni è stata ideata la classe GestoreUpdateGlobali.

Figura 20: Classe `GestoreUpdateGlobali`

21. GESTIONE LEVEL UP

La componente **Gestione level up** si occupa di raccogliere le informazioni rigurdanti l'aggiornamento del livelllo interazione (ricreazioni, ricette completate, valutazioni). Per questo è stata creata la componente `GestioneLevelUp`.

Figura 21: Classe `GestioneLevelUp`

22. GESTIONE AGGIORNAMENTO ACCOUNT

La componente Gestione aggiornamento account si occupa di interfacciarsi con il database per passargli gli aggiornamenti rigurdanti i dati dell'account (username, indirizzo email, password, immagine profilo) e il livello interazione. Per questo è stata creata la classe GestoreUpdateDatiAccount.

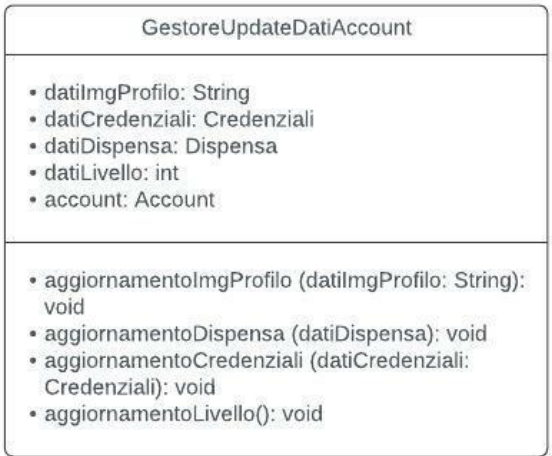
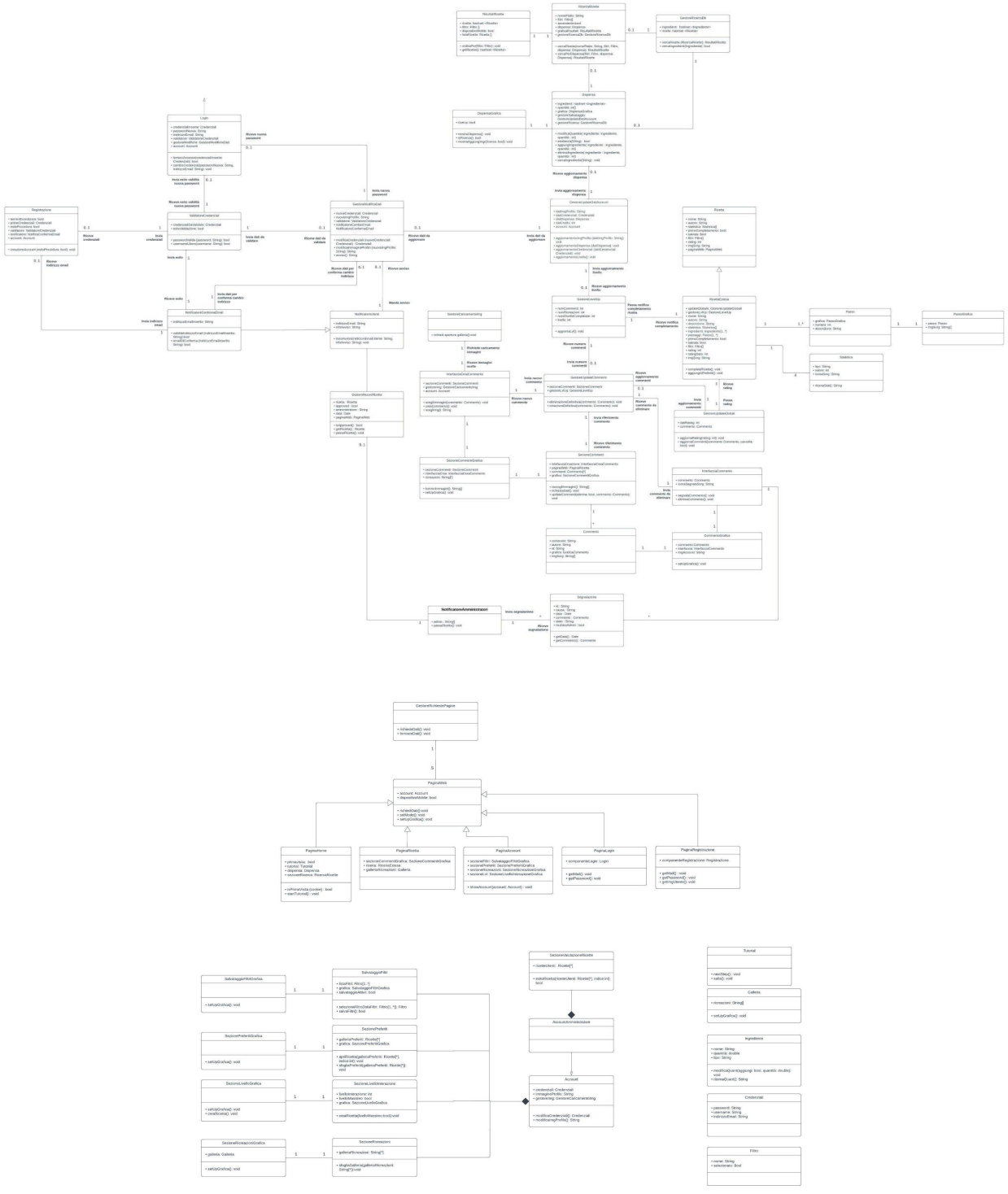


Figura 22: Classe GestoreUpdateDatiAccount

1.2. DIAGRAMMA CLASSI

In questa sezione viene riportato il digramma completo contenente le classi descritte precedentemente e le relazioni tra di esse.



2.CODICE IN OBJECT CONSTRAINT LANGUAGE

In questo paragrafo vengono descritti alcuni metodi delle classi nel dettaglio. Nello specifico ci si concentra sulla logica delle singoli operazioni. Per farlo al meglio viene utilizzato l'Object Constraint Language (OCL).

1. CRITERI ORDINAMENTO RICERCA RICETTE

Nella funzione `ordinaPer` della classe `RisultatiRicette` è necessario, per la corretta esecuzione del programma, che prima dell'esecuzione del metodo sia impostato almeno un filtro. In caso contrario non sarebbe necessario chiamare la funzione. Possiamo descrivere questa condizione con il seguente codice OCL.

```
context RisultatiRicette::ordinaPer(filtro)
pre : filtro != NULL
```

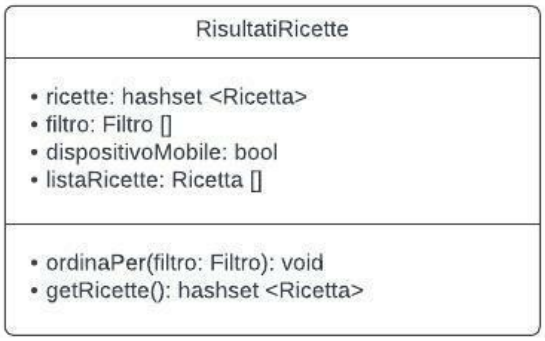


Figura 23: Classe relativa ai criteri di ordinamento

2. RICERCA RICETTE PER NOME PIATTO

Nella classe `RicercaRicette` è presente il metodo `cercaRicette`. Per permettere a questo metodo di venire chiamato correttamente dobbiamo imporre che uno dei parametri presi in input, in particolare il parametro dispensa, non sia nullo. Questa limitazione è espressa formalmente come segue:

```
context RicercaRicette::cercaRicette(nomePiatto, filtri, dispensa)
pre : dispensa != NULL
```

Figura 24: Classe relativa alla ricerca ricette

3. RICERCA RICETTE PER DISPENSA

Sempre appartenente alla classe sopracitata `RicercaRicette` (figura 24), il metodo `cercaPerDispensa` richiede una limitazione simile a quella dell'analogo `cercaRicette` poiché il funzionamento di questi due è quasi del tutto identico. Viene quindi espressa l'impossibilità del parametro ricetta di essere nullo in questo modo:

```
context RicercaRicette::cercaPerDispensa(nomePiatto, filtri, dispensa)
pre : dispensa != NULL
```

4. RIMOZIONE INGREDIENTE

La funzione `rimuoviIngrediente` della classe `Dispensa` richiede che la quantità dell'ingrediente che si desidera rimuovere sia minore della quantità che si possiede (attributo della classe `Dispensa`). Inoltre si prevede che ogni volta che venga chiesto di rimuovere una certa quantità di un certo ingrediente questa debba essere almeno un unità, al contrario infatti il metodo perderebbe di significato. Per descrivere queste condizioni si usa il seguente codice:

```
context Dispensa::rimuoviIngrediente(ingrediente, quantità)
pre : ingrediente -> count(ingredienti) > 1 &&
count(ingredienti)<=quantità
inv : quantità[ingrediente] >= 1
```

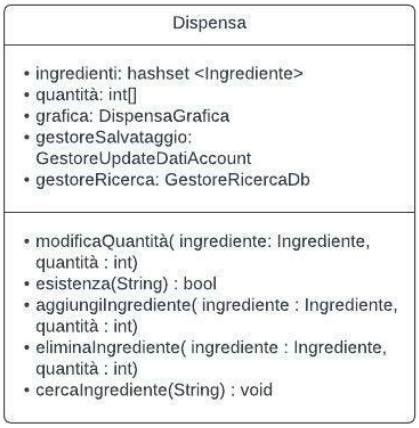


Figura 24: Classe relativa alla rimozione ingrediente

5. APPROVAZIONE NUOVE RICETTE

Nella classe `GestioneNuoveRicette` il metodo `isApproved` è usato per aggiornare lo stato di una richiesta di inserimento di una nuova ricetta. Si impone il che dopo l'esecuzione del metodo il parametro booleano approved abbia uno stato definito. Di seguito il codice per descrivere questa condizione.

```
context GestoreNuoveRicette::isApproved()
post : approved == true || approved == false
```

Figura 25: Classe relativa all'approvazione nuove ricette

6. SEGNALAZIONE

Nella classe Segnalazione si decide di imporre all'utente un vincolo, ovvero che il commento legato a una determinata segnalazione non ecceda i 251 caratteri. Questa imposizione utile alla semplicità e mantenibilità del sistema è implementata con il codice OCL:

```
context Segnalazione inv commento.size() < 251
```

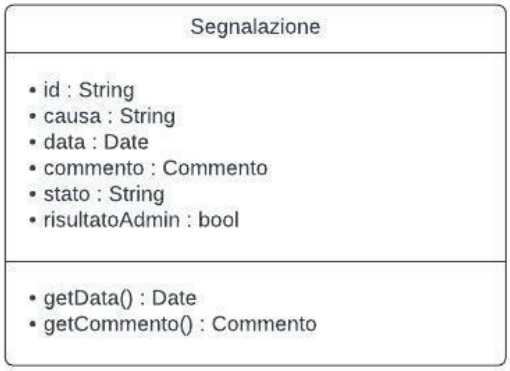


Figura 26: Classe relativa alla segnalazione

7. PRIMA VISITA PAGINA HOME

Il metodo `isPrimaVisita` della classe `PaginaHome` permette al sistema di capire se l'utente ha già visitato il sito in precedenza. Per ottenere un metodo affidabile, si pone come condizioni che la variabile `primavisita` sia falsa prima appunto di aver eseguito il metodo, e sia `true` alla fine dell'esecuzione. Per descrivere queste condizioni in OCL si usa il seguente codice:

```
context PaginaHome::isPrimaVisita(cookie)
pre : primavisita == true
post : primavisita == false
```

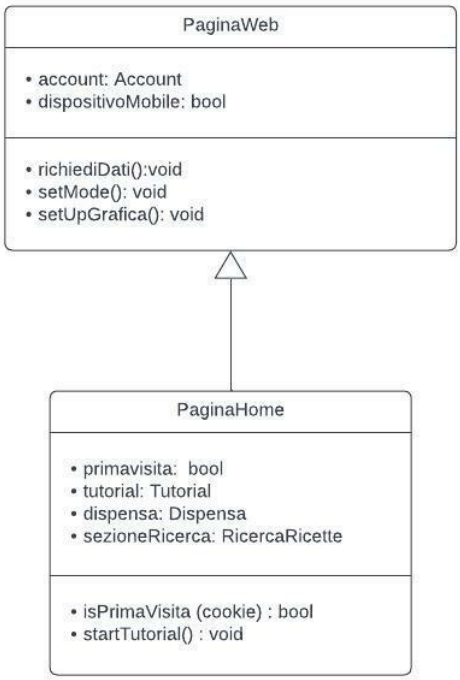


Figura 27: Classi relative alla prima visita pagina home

8. LIVELLO INTERAZIONE

La classe `SezioneLivelloInterazione` ha come attributo il `livelloInterazione`. Quest'ultimo ha sempre un valore che varia tra un minimo di 0 è un massimo di 10. Questa condizione è espressa in OCL attraverso l'invariante con questo codice:

```
context SezioneLivelloInterazione inv:
SezioneLivelloInterazione.livelloInterazione>=0 AND
SezioneLivelloInterazione.livelloInterazione<=10
```

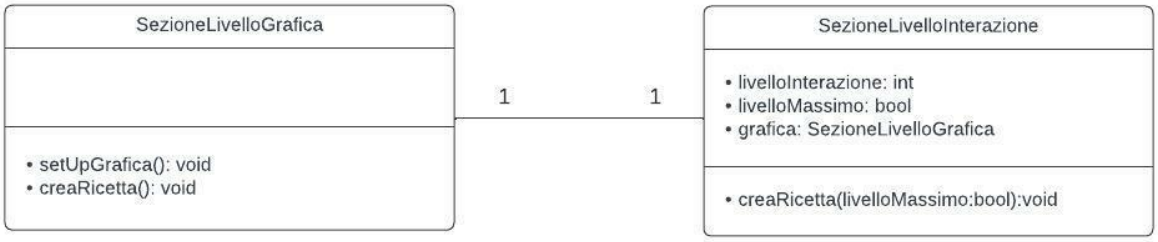


Figura 28: Classi relative al livello interazione

9. CREAZIONE RICETTE

La classe precedente presenta anche il metodo `creaRicetta` che prende come parametro `livelloMassimo`. Quest'ultimo rimane a false finchè non si raggionge il livello 10, attraverso il quale si possono creare ricette. Queste informazioni sono espresse tramite OCL attraverso la seguente precondizione:

```
context SezioneLivelloInterazione::creaRicetta(lm: livelloMassimo)
pre: SezioneLivelloInterazione.lm=true
```

10. SELEZIONE FILTRI

`SalvataggioFiltri` è la classe che identifica la sezione della pagina dell'account dove si possono selezionare i filtri da attivare automaticamente nella ricerca delle ricette. Quando si scelgono i filtri deve essere attivo "il salvataggio". Per descrivere questa condizione viene utilizzata la seguente precondizione in codice OCL:

```
context SalvataggioFiltri::selezionaFiltro(lf: listaFiltri)
pre: SalvataggioFiltri.salvataggioAttivo=true
```

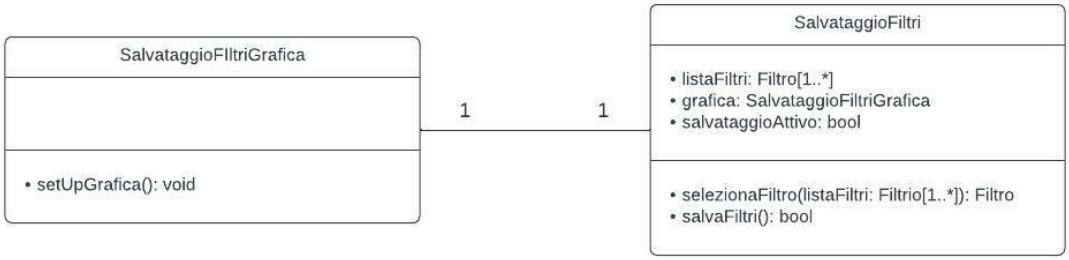


Figura 29: Classi relative al salvataggio filtri

11. APERTURA RICETTE PREFERITE

La classe `SezionePreferiti` contiene le ricette preferite dell'utente. Le ricette possono essere cliccate per visualizzarne la pagina. Ovviameente se non ci sono preferiti non si può aprire nesuna pagina. Questa condizione è espressa attraverso la precondizione seguente scritta in codice OCL:

```
context SezionePreferiti::apriRicetta(gp: galleriaPreferiti, i: indice)
pre: SezionePrefriti.gp!=NULL
pre: i>=0
```

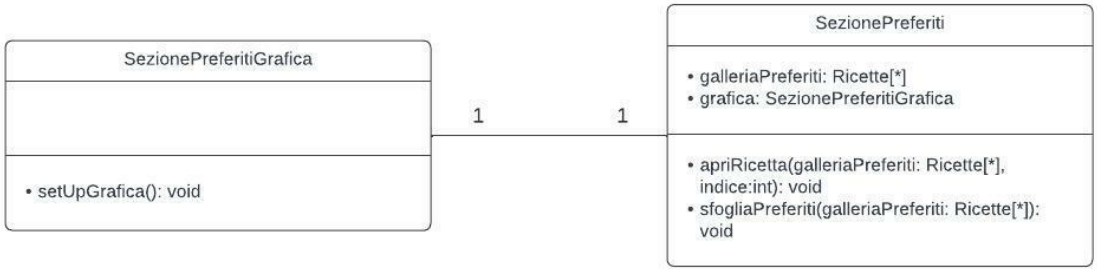


Figura 30: Classi relative all'apertura ricette preferite

12. ESITO VALUTAZIONE RICETTE

La classe `SezioneValutazionePreferiti` rappresenta la sezione della pagina dell'account dove l'amministratore può accetare o rifiutare le ricette proposte dagli utenti. Ovviamente se non ci sono nuove ricette non si può dare una valutazione. Questa condizione è espressa attraverso la precondizione in codice OCL seguente:

```
context SezioneValutazionePreferiti::esitoRicetta(ru: ricetteUtenti,i: indice)
pre: SezioneValutazionePreferiti!=NULL
pre: i>=0
```

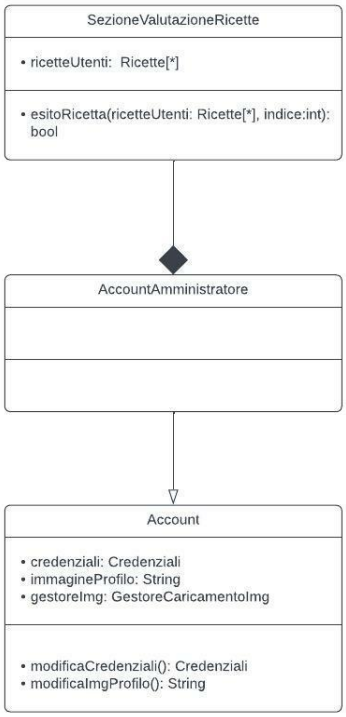


Figura 31: Classi relative all'esito valutazione ricette

13. CREAZIONE ACCOUNT

La classe `Registrazione` riguarda la creazione degli account. Quando si sono validati i dati (indirizzo email associato, username, password) è possibile completare la procedura. Nella classe questo è possibile quando gli attributi `esitoProcedura` e `validatore.esitoValidazione` hanno valore `true`. Si può osservare nella precondizione in codice OCL seguente:

```
context Registrazione::creazioneAccount(ep: esitoProcedura)
pre: Registrazione.ep=true
pre: Registrazione.validatore.esitoValidazione=true
```

Figura 32: Classe relativa alla creazione account

14. CAMBIO PASSWORD LOGIN

Con la classe `Login` e il suo metodo `cambioCredenziali` è possibile modificare la password. Questo avviene quando arriva la conferma tramite email e la password è stata validata. A livello di classe è quando l'attributo `validatore.esitoValidazione` ha valore `true`. Inoltre dopo la modifica la password nuova è quella inserita nella pagina del login. Queste condizioni sono espresse con la precondizione e postcondizione (OCL):

```
context Login::cambioCredenziali (pn: passwordNuova, ie: indirizzoEmail)
pre: Login.validatore.esitoValidazione = true
post: Login.account.credenziali.password = pn
```

3. DIAGRAMMA DELLE CLASSI CON CODICE OCL

In questo pargrafo viene introdotto nuovamente il digramma delle componenti, ma con l'aggiunta del codice OCL.