



JustCook

03/12/22

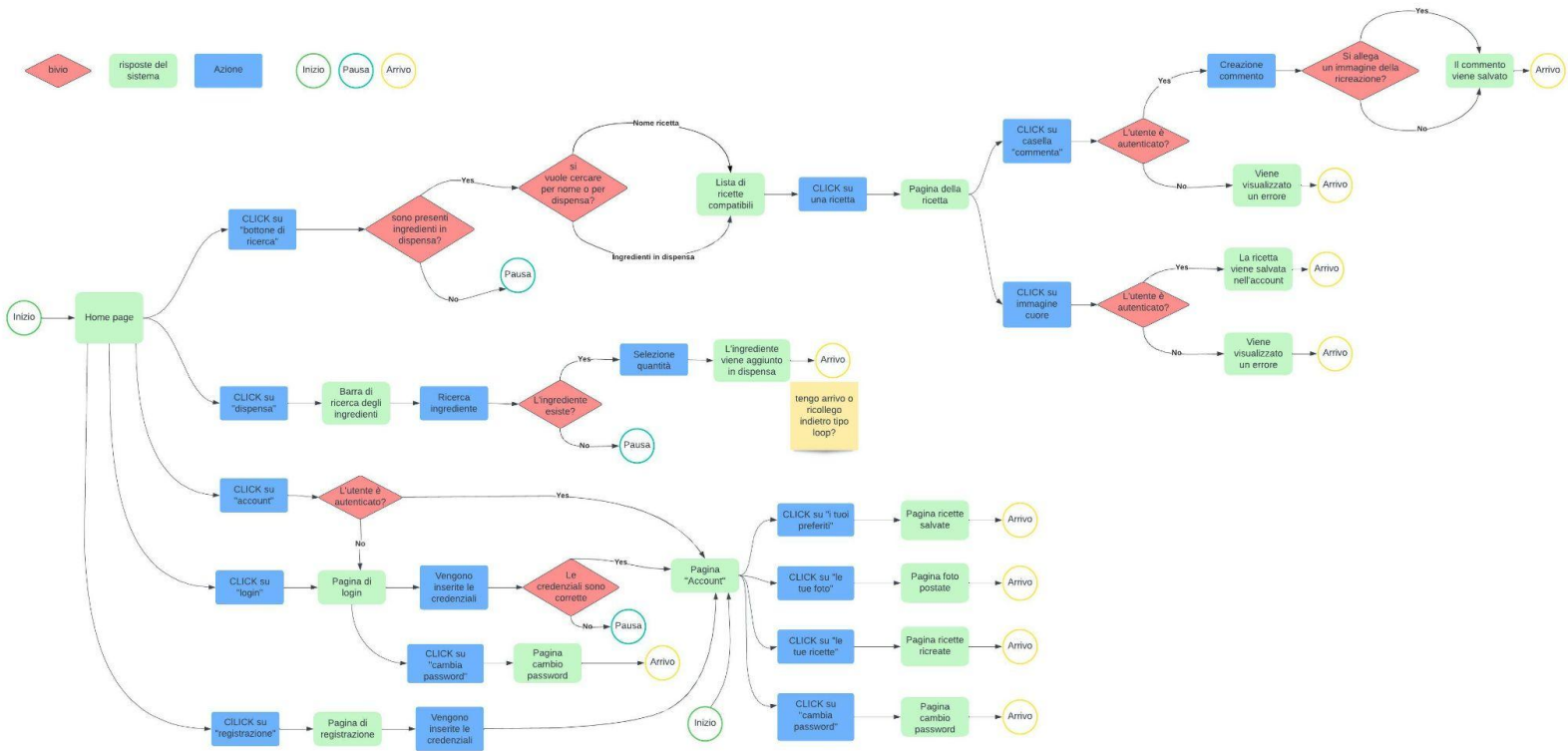
SVILUPPO APPLICAZIONE



Indice

SCOPO DEL DOCUMENTO

1. USER FLOWS



2. APPLICATION IMPLEMENTATION AND DOCUMENTATION

2.1. PROJECT STRUCTURE

La struttura del progetto si divide in tre cartelle: **api**, **ui**, **Photos**. La cartella **api** serve per gestire le API locali, la cartella **ui** contiene tutto il materiale per il front-end, la cartella **Photos** memorizza le immagini di supporto necessarie. Questa suddivisione è visibile in [figura 3](#).

2.2. PROJECT DEPENDENCIES

Oltre ai moduli Node standard che vengono scaricati al momento dell'installazione, sono stati utilizzati altri moduli e aggiunti al file Package.Json. Quest'ultimi sono:

- Express
- MongoDB

2.3. PROJECT DATA OR DB

In questo paragrafo vengono presentate le strutture dati presenti nel database di JustCook. Non sono state ideate tutte quelle necessarie per la completa realizzazione dell'applicazione Web, ma quelle relative ai dati utili per la "creazione di un prototipo". Nell'immagine seguente (figura 4) sono mostrate le collezioni dati appena introdotte. Successivamente per ogni struttura verrà riportata una descrizione con un esempio.

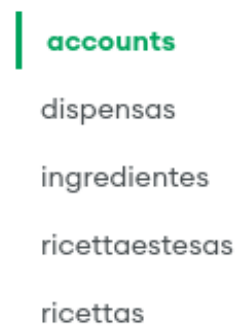


Figura 4: Collezioni Dati

Esempi tipi di dato

1. ACCOUNT

Per rappresentare gli account è stato creato il modello `account` con le seguenti "proprietà":

- `preferiti`: contiene gli id delle ricette preferite dall'utente
- `password`: la password dell'account
- `ratingDati`: contiene gli id delle ricette e i rating associati ad esse dati dagli utenti
- `primiCompletamenti`: contiene gli id delle ricette completate (per la prima volta) dall'utente
- `indirizzoEmail`: l'indirizzo email associato all'account
- `username`: il nome utente associato all'account

```
_id: ObjectId('63a6fa25db2b4423e4c6e5de')
✓ preferiti: Array
  0: ObjectId('639d9961f5edd649e010000e')
  1: ObjectId('639cd417d099f3b10a5e238c')
password: "eadJ039kl^^"
✓ ratingDati: Array
  ✓ 0: Object
    ricetta: ObjectId('639cd417d099f3b10a5e238c')
    ratingData: 3
    _id: ObjectId('63aeab3dd17d3be57b478d53')
  ✓ 1: Object
    ricetta: ObjectId('639d9961f5edd649e010000e')
    ratingData: 3
    _id: ObjectId('63aeab79d17d3be57b478d62')
✓ primiCompletamenti: Array
  0: ObjectId('639cd417d099f3b10a5e238c')
indirizzoEmail: "babbonatale:)@gmail.com"
username: "BabboNatale01"
```

Figura 5: Tipo di dato Account

2. DISPENSA

Per rappresentare le dispense è stato creato il modello `dispensa` con le seguenti “proprietà”:

- `ingredienti`: contiene gli ingredienti presenti nella dispensa
- `account`: è l'id dell'account associato ad una determinata dispensa
- `quantita`: contiene le quantità degli ingredienti presenti in dispensa

```
_id: ObjectId('63aee34a40713fc321b4d6cd')
✓ ingredienti: Array
  0: ObjectId('639cd6ee6335dc6ce6a36672')
  account: ObjectId('63a6fa25db2b4423e4c6e5de')
✓ quantita: Array
  0: 1
```

Figura 6: Tipo di dato Dispensa

3. INGREDIENTE

Per rappresentare gli ingredienti è stato creato il modello `ingrediente` con le seguenti “proprietà”:

- `nome`: è il nome dell'ingrediente
- `tipo`: è il tipo (grammi, numero, etti) dell'ingrediente

```
_id: ObjectId('639d9672f5edd649e0100005')  
nome: "Yogurt greco"  
tipo: "g"
```

Figura 7: Tipo di dato ingrediente

4. RICETTA ESTESA

Per rappresentare le ricette della pagina della ricetta è stato creato il modello `ricettaEstesa` con le seguenti “proprietà”:

- `ricetta`: è l'id della ricetta corrispondente
- `descrizione`: è la descrizione della ricetta in breve
- `ingredienti`: contiene gli ingredienti utilizzati nella ricetta corrispondente
- `passaggi`: è l'insieme delle istruzioni per la creazione del piatto
- `quantita`: contiene le quantità degli ingredienti della ricetta

```
_id: ObjectId('639cd58cd099f3b10a600972')  
ricetta: ObjectId('639cd417d099f3b10a5e238c')  
descrizione: "Il tiramisù è uno dei capisaldi della cucina italiana, uno dei dolci a..."  
▼ ingredienti: Array  
  0: ObjectId('639cd6dcd099f3b10a615c4a')  
  1: ObjectId('639cd6ee6335dc6ce6a36672')  
▼ passaggi: Array  
  0: "Per preparare il tiramisù preparate il caffè con la moka per ottenerne..."  
  1: "Montate i tuorli con le fruste elettriche, versando solo metà dose di ..."  
  2: "Non appena il composto sarà diventato chiaro e spumoso, e con le frust..."  
  3: "Incorporato tutto il formaggio avrete ottenuto una crema densa e compa..."  
  4: "Pulite molto bene le fruste e passate a montare gli albumi"  
  5: "Quando saranno schiumosi versate il restante zucchero un po' alla volt..."  
  6: "Dovrete montarli a neve ben ferma "  
  7: "Prendete una cucchiata di albumi e versatela nella ciotola con la cr..."  
  8: " Mescolate energicamente con una spatola"  
  9: "Dopodiché procedete ad aggiungere la restante parte di albumi, poco al..."  
▼ quantita: Array  
  0: 750  
  1: 250
```

Figura 8: Tipo di dato `ricettaEstesa`

5. RICETTA

Per rappresentare le ricette è stato ideato il modello `ricetta` con le seguenti “proprietà”:

- `nome`: è il nome della ricetta
- `autore`: è l’username dell’autore della ricetta
- `statistica`: contiene i dati della ricetta (tempo, difficoltà, costo)
- `filtri`: contiene i filtri della ricetta
- `rating`: votazione degli utenti alla ricetta

```
_id: ObjectId('639cd417d099f3b10a5e238c')
nome: "Tiramisù"
autore: "Maria"
statistica: Array
  0: 100
  1: 1
  2: 1
filtri: Array
  0: "senza uova"
  1: "dessert"
rating: 4
```

Figura 9: Tipo di dato `ricetta`

2.4. PROJECT APIs

In questo paragrafo vengono presentate le API dell'applicazione JustCook. Prima vengono descritte le risorse API attraverso la lingua italiana e dei digrammi in Unified Modeling Language (UML). Quest'ultimi sono il Resources Extraction Diagram (dal diagramma delle classi) e il Resources Model. Successivamente viene mostrato il codice con cui vengono implementate alcune delle API descritte.

2.4.1. RESOURCES EXTRACTION

In questa sezione vengono presentate e descritte le risorse API ideate a partire dai metodi delle classi del Class Diagram (analizzato nel documento precedente). Successivamente viene mostrato il Resources Extraction Diagram.

Descrizione risorse API estratte

1. CREDENZIALI INIZIALI - METODO GET CREDENZIALI

Analizzando la classe `PaginaRegistrazione` e il metodo `getCredenziali` è stata ideata l'API **Credenziali iniziali**. L'utente nella fase di creazione del proprio account deve inserire l'username, la password e l'indirizzo email associato in delle apposite barre. L'API analizzata possiede il metodo HTTP `GET` per ottenere e validare le prime credenziali appena inserite.

2. REGISTRAZIONE - METODO CREAZIONE ACCOUNT

Analizzando il metodo `creazioneAccount` della classe `Registrazione` è stata ideata l'API **Registrazione**. Quest'ultima possiede il metodo HTTP `POST` perchè deve creare un nuovo account per l'utente che ha effettuato la registrazione correttamente.

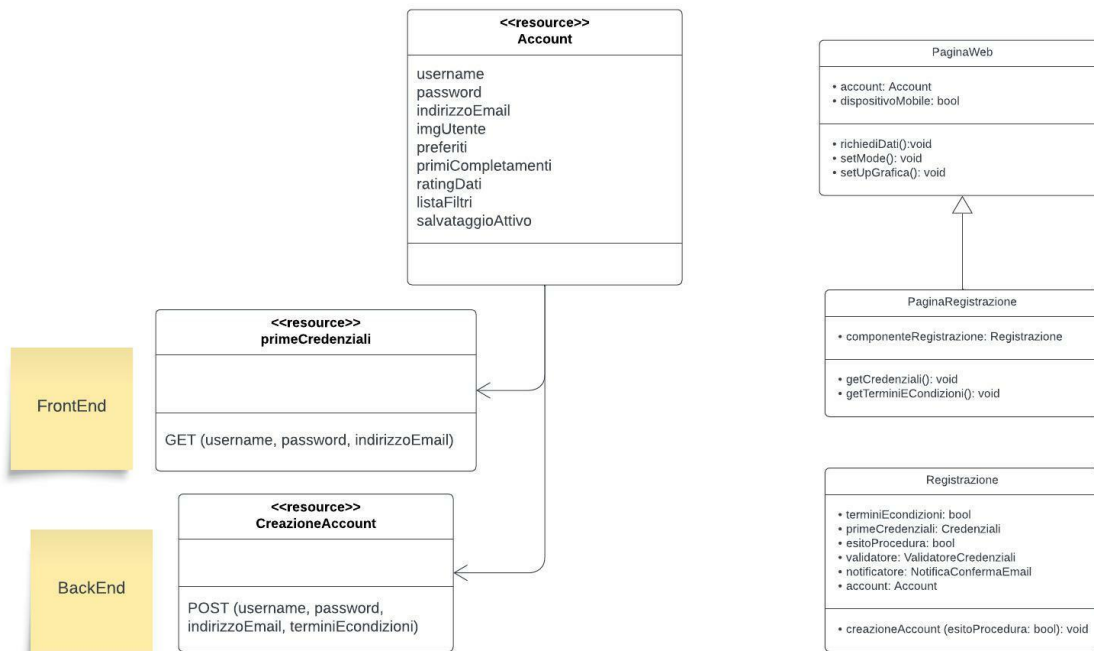


Figura 10: Classi e Resources Extraction Diagram registrazione

3. NUOVE CREDENZIALI - METODO GET CREDENZIALI

Dalla pagina dell'account è possibile modificare le credenziali (username, password, indirizzo email). Si possono cambiare inserendo i nuovi dati nelle apposite barre. Per questo è stato creato il metodo `getCredenziali` della classe `Account`. Di conseguenza è stata ideata l'API **Nuove credenziali** con il metodo HTTP `GET` per ottenere i dati e valutarli.

4. MODIFICA CREDENZIALI - METODO MODIFICA CREDENZIALI

La classe `GestoreModificaDati` possiede il metodo `modificaCredenziali`. Per questo è stata ideata l'API **Modifica credenziali** che si occupa di cambiare i dati in questione con quelli inseriti nella pagina dell'account. Possiede il metodo HTTP `PUT`.

5. MODIFICA IMMAGINE PROFILO - METODO MODIFICA IMG PROFILO

Analizzando la classe `GestoreModificaDati` e il metodo `modificaImgProfilo` è stata ideata l'API **Modifica Immagine profilo**. L'utente nella propria pagina dell'account può modificare la foto in questione. Per cambiarla l'API considerata utilizza il metodo HTTP `PATCH`.

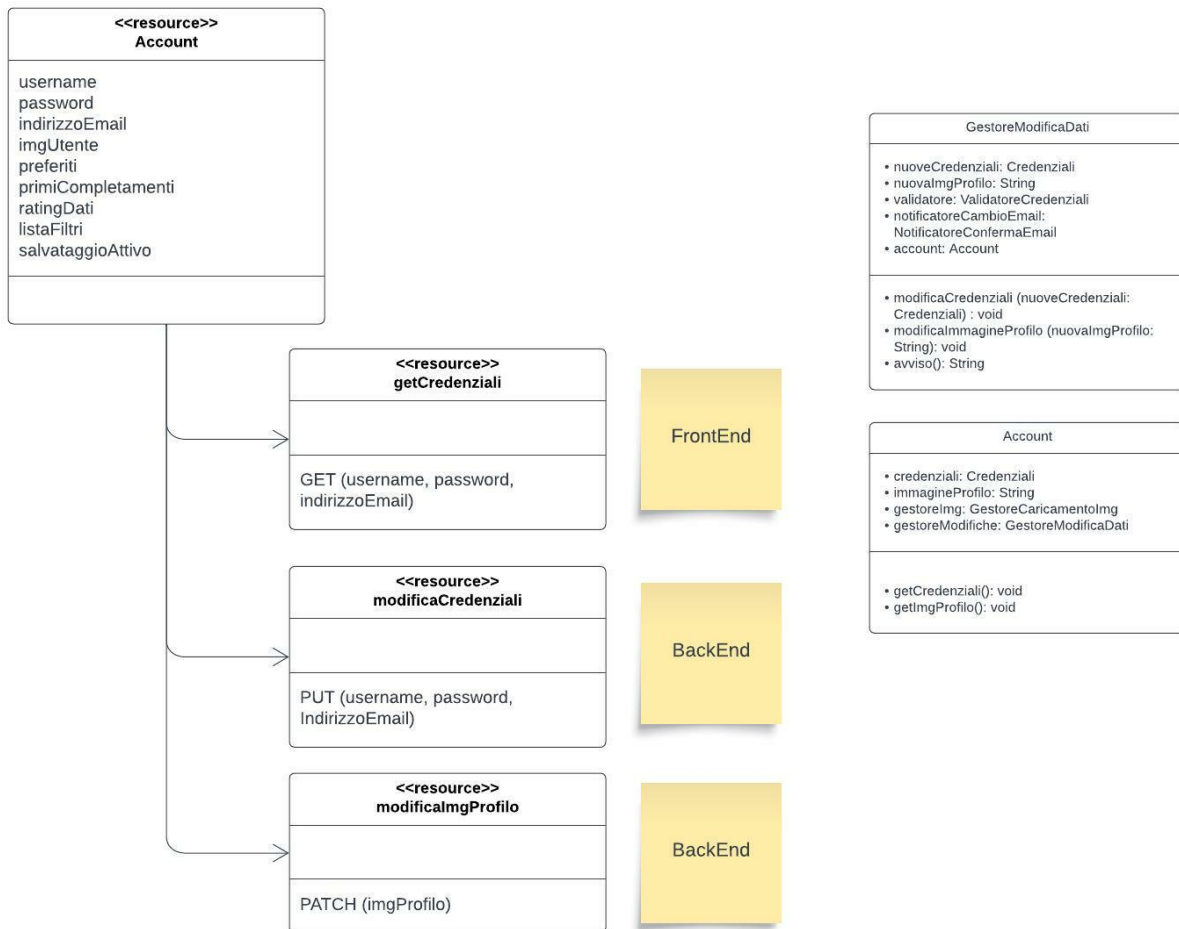


Figura 11: Classi e Resources Extraction Diagram modifica dati

6. SALVATAGGIO FILTRI - METODO SALVA FILTRI

Analizzando la classe `SalvataggioFiltri` e il suo metodo `salvaFiltri` è stata individuata l'API **Salvataggio filtri**. L'utente nella pagina dell'account può decidere se avere dei filtri predefiniti nella ricerca delle ricette oppure no. Questa modifica viene effettuata con il metodo HTTP `PATCH`.

7. SELEZIONA FILTRI - METODO SELEZIONA FILTRO

L'utente quando aziona il salvataggio dei filtri può selezionare quelli che vuole avere attivi nella ricerca delle ricette. È possibile farlo attraverso il metodo `selezionaFiltro` della classe `SalvataggioFiltri`. Di conseguenza è stata ideata l'API **Seleziona filtri** con il metodo HTTP `PUT`.

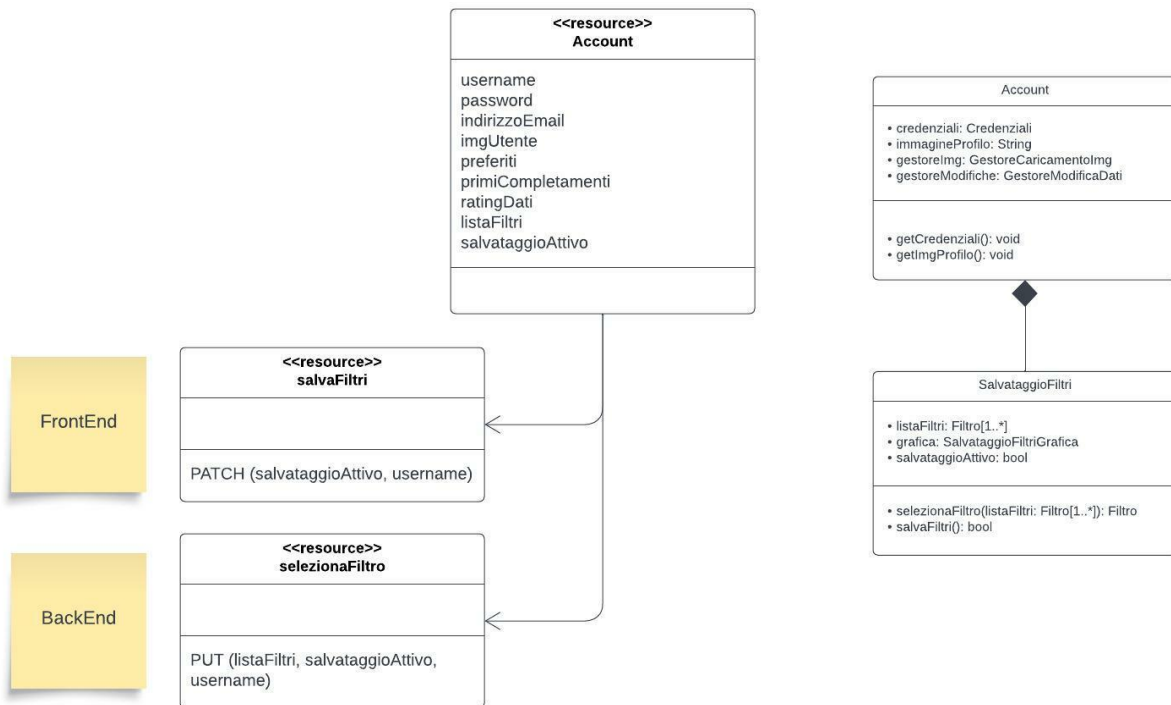


Figura 12: Classi e Resources Extraction Diagram salvataggio filtri

8. LOGIN - METODO FORNISCI ACCESSO

L'utente per accedere al proprio account deve inserire l'username e la password corretti. Per verificarne l'esattezza viene utilizzato il metodo `fornisciAccesso` della classe `Login`. Di conseguenza è stata creata l'API **Login** con il metodo HTTP `POST`.

9. EMAIL CAMBIO PASSWORD - METODO GET MAIL

Analizzando la classe `PaginaLogin` e il suo metodo `getMail` è stata ideata l'API **Email cambio password**. Quando l'utente desidera cambiare la password nella pagina del login deve inserire anche l'indirizzo email associato in un'apposita barra. Per prelevare l'indirizzo email e vedere se è associato ad un account, l'API considerata utilizza il metodo HTTP `GET`.

10. CAMBIO PASSWORD LOGIN - METODO CAMBIO CREDENZIALI

L'utente può cambiare la password dalla pagina del login. Per questo è stato ideato il metodo `cambioCredenziali` della classe `Login`. A quest'ultimo viene associata l'API **Cambio password login** con il metodo HTTP `PATCH`.

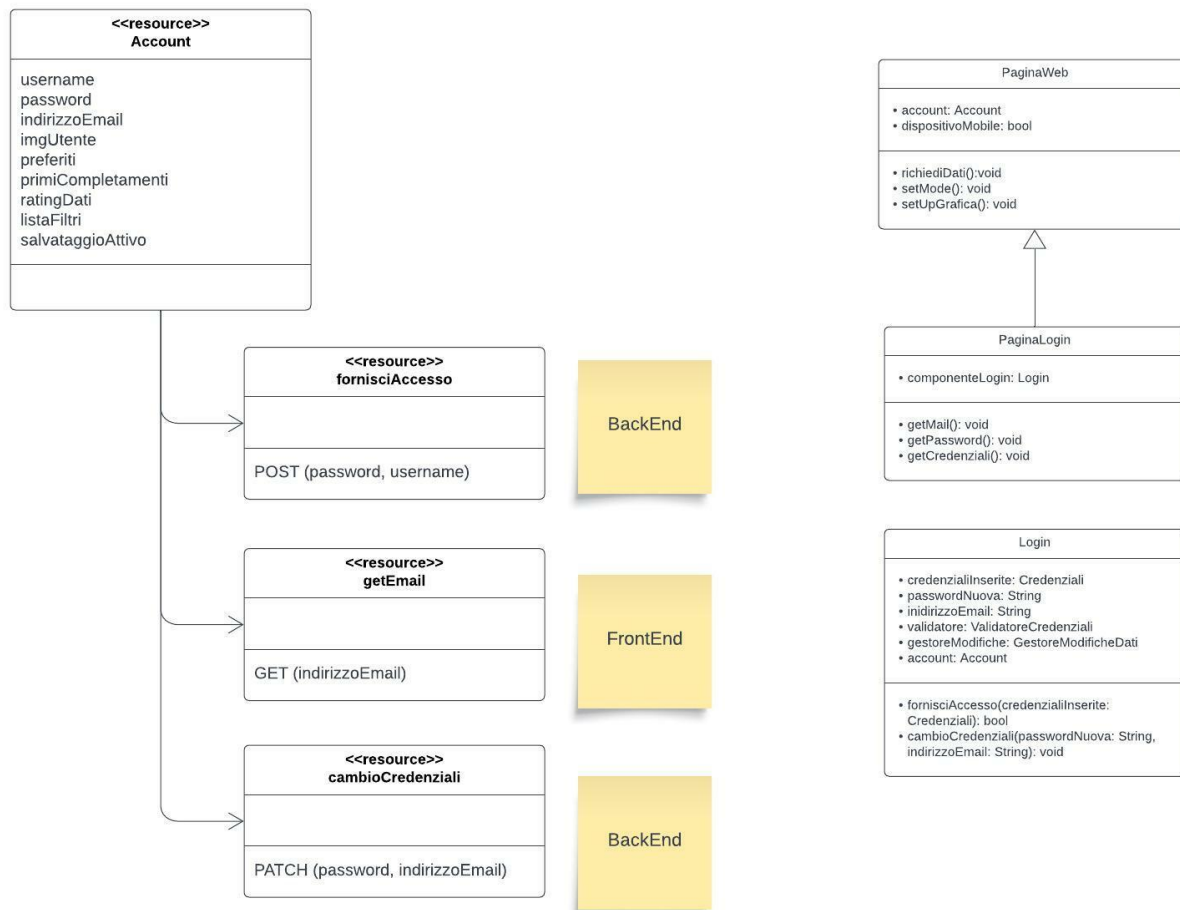


Figura 13: Classi e Resources Extraction Diagram login

11. CREA COMMENTO - METODO CREA COMMENTO

L'API **Crea commento** corrisponde al metodo `creaCommento` della classe `InterfacciaCreaCommento`. Attraverso il metodo HTTP `POST` aggiunge un nuovo commento alla relativa ricetta nel database.

12. ELIMINA COMMENTO - METODO ELIMINA COMMENTO

L'API **Elimina commento** corrisponde al metodo `eliminaCommento` della classe `InterfacciaCommento`. Attraverso il metodo HTTP `DELETE` viene eliminato il commento che corrisponde all'id passato dalla relativa ricetta nel database.

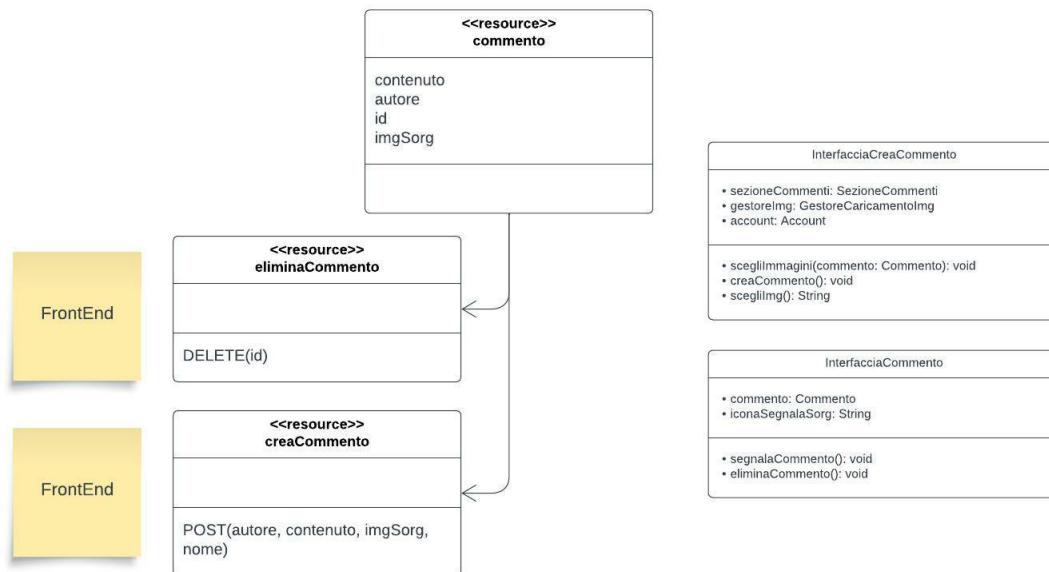


Figura 14: Classi e Resources Extraction Diagram commenti

13. MODIFICA QUANTITÀ - METODO MODIFICA QUANTITÀ

L'API **Modifica quantità** estratta dalla classe `Dispensa` è stata ideata allo scopo di modificare la quantità di un dato ingrediente in dispensa alla pressione di un tasto. Per implementare questa interazione dell'utente con il database si utilizza un metodo HTTP `PATCH` in modo che il resto della dispensa rimanga invariato.

14. AGGIUNGI INGREDIENTE - METODO AGGIUNGI INGREDIENTE

Sempre parte della classe `Dispensa`, analizzando il metodo `aggiungiIngrediente` si decide di implementare l'API **Aggiungi ingrediente**. Quest'ultima ha lo scopo di, una volta trovato l'ingrediente interessato fra tutti quelli disponibili, aggiungerlo alla propria dispensa. Per fare ciò si utilizza il metodo HTTP `PATCH`, in quanto nonostante un ingrediente venga aggiunto non si vuole perdere la dispensa già salvata.

15. ELIMINA INGREDIENTE - METODO ELIMINA INGREDIENTE

L'API **Elimina ingrediente** viene implementata analizzando la necessità, da parte sia dell'utente che del sistema, di eliminare un ingrediente dalla dispensa attuale. Per farlo viene usato il metodo HTTP `PATCH`, che viene chiamato alla pressione di un pulsante.

16. CERCA INGREDIENTE - METODO CERCA INGREDIENTE

L'API **Cerca ingrediente** viene ideata a partire dal metodo `cercaIngrediente` della classe `Dispensa`. Serve per verificare che l'ingrediente che si vuole aggiungere nella dispensa sia presente nel database di JustCook. Per fare questo utilizza il metodo HTTP GET.

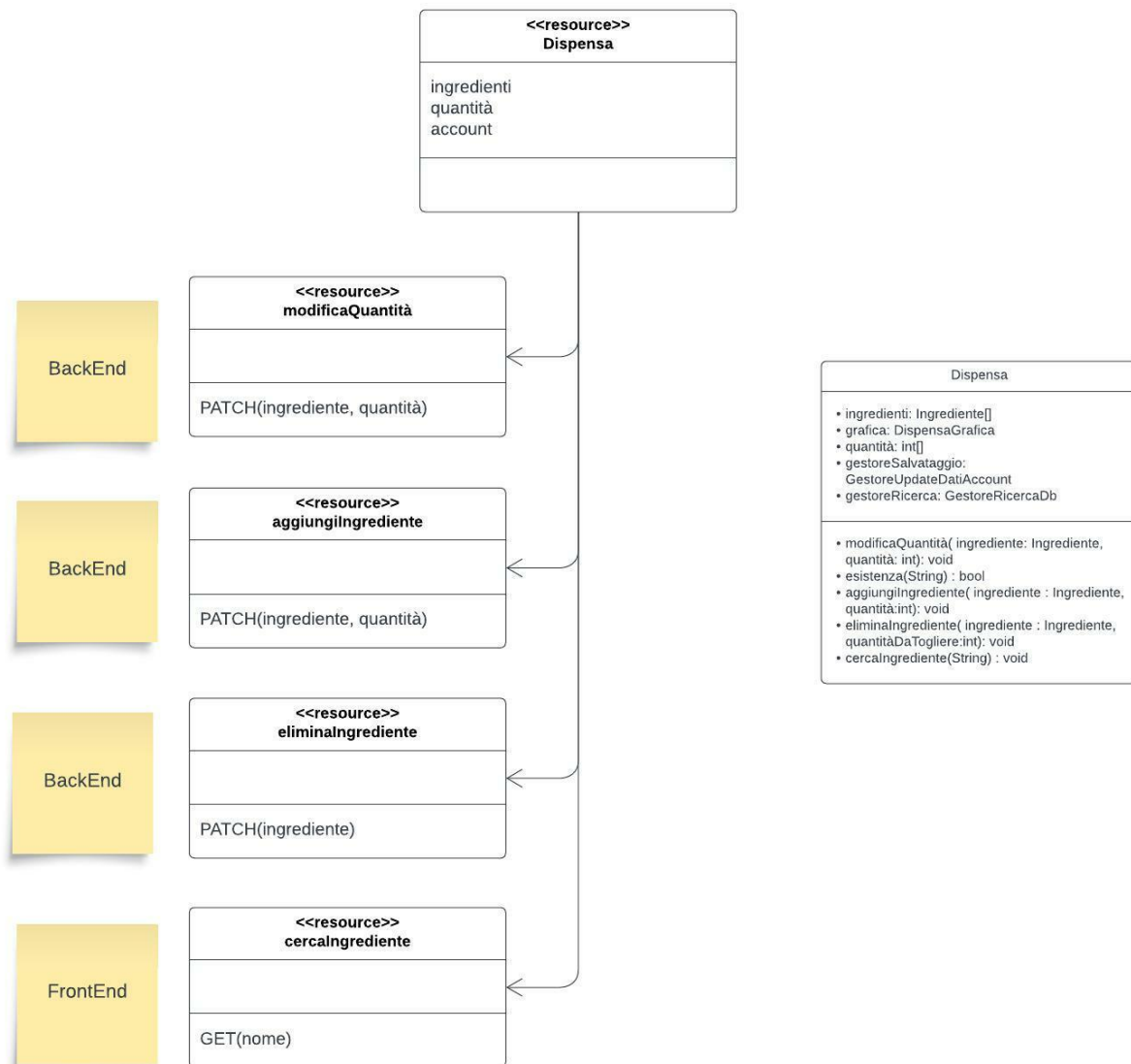


Figura 15: Classe e Resources Extraction Diagram interazione con dispensa

17. CERCA RICETTE - METODO CERCA RICETTE

Il metodo `cercaRicette` viene implementato con l'API **Cerca ricette** a partire dalla classe `RicercaRicette`. Analizzando la classe infatti si può notare la necessità di ricevere in ingresso alcune ricette a partire da dei criteri di ricerca. Questo metodo permette anche di cercare a partire dal nome della ricetta che si desidera leggere. Per implementare questa funzione viene usato un metodo HTTP `GET`.

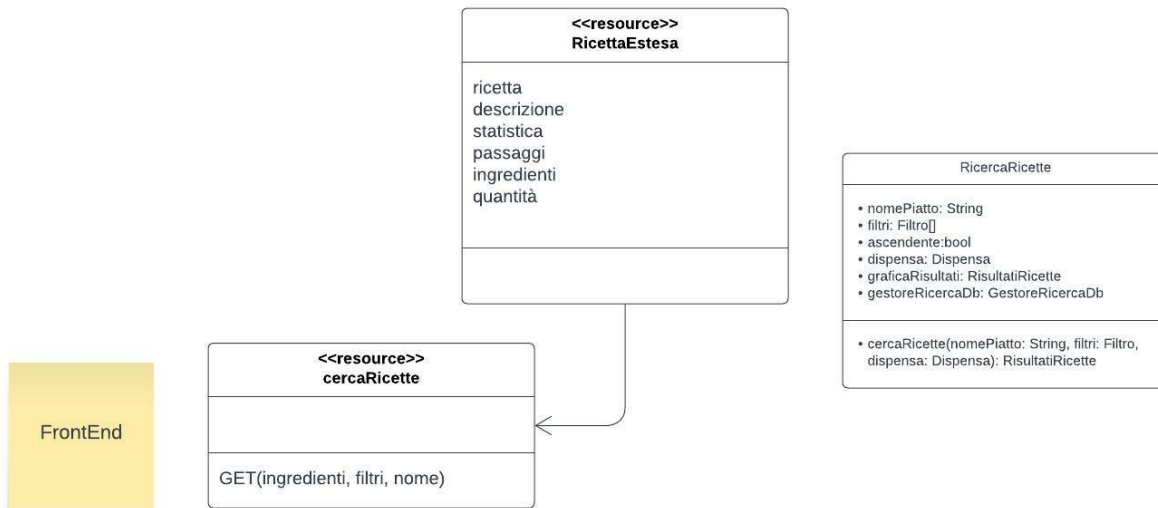


Figura 16: Classe e Resources Extraction Diagram ricerca ricette

18. COMPLETA RICETTA - METODO COMPLETA RICETTA

L'API **Completa ricetta** corrisponde, nel diagramma delle classi, al metodo `completaRicetta` della classe `RicettaEstesa`. Utilizza il metodo HTTP `PATCH` per aggiornare il database quando avviene il primo completamento di una determinata ricetta.

19. AGGIUNGI AI PREFERITI - METODO AGGIUNGI A PREFERITI

L'API **Aggiungi ai preferiti** corrisponde al metodo `aggiungiAPreferiti` della classe `RicettaEstesa`. Attraverso il metodo HTTP `PATCH`, modifica lo stato di una determinata ricetta per segnalare la sua presenza tra i preferiti dell'utente.

20. TOGLI DAI PREFERITI - METODO TOGLI DAI PREFERITI

L'API **Togli dai preferiti** corrisponde al metodo `togliDaiPreferiti` della classe `RicettaEstesa`. Attraverso il metodo HTTP `PATCH`, aggiorna lo stato della ricetta nel database per segnalare la sua assenza tra i preferiti dell'account.

21. AGGIUNGI RATING - METODO DAI RATING

L'API **Aggiungi rating** corrisponde al metodo `daiRating` della classe `ricettaEstesa`. Attraverso il metodo HTTP `PATCH` aggiunge il nuovo rating alla ricetta nel database.

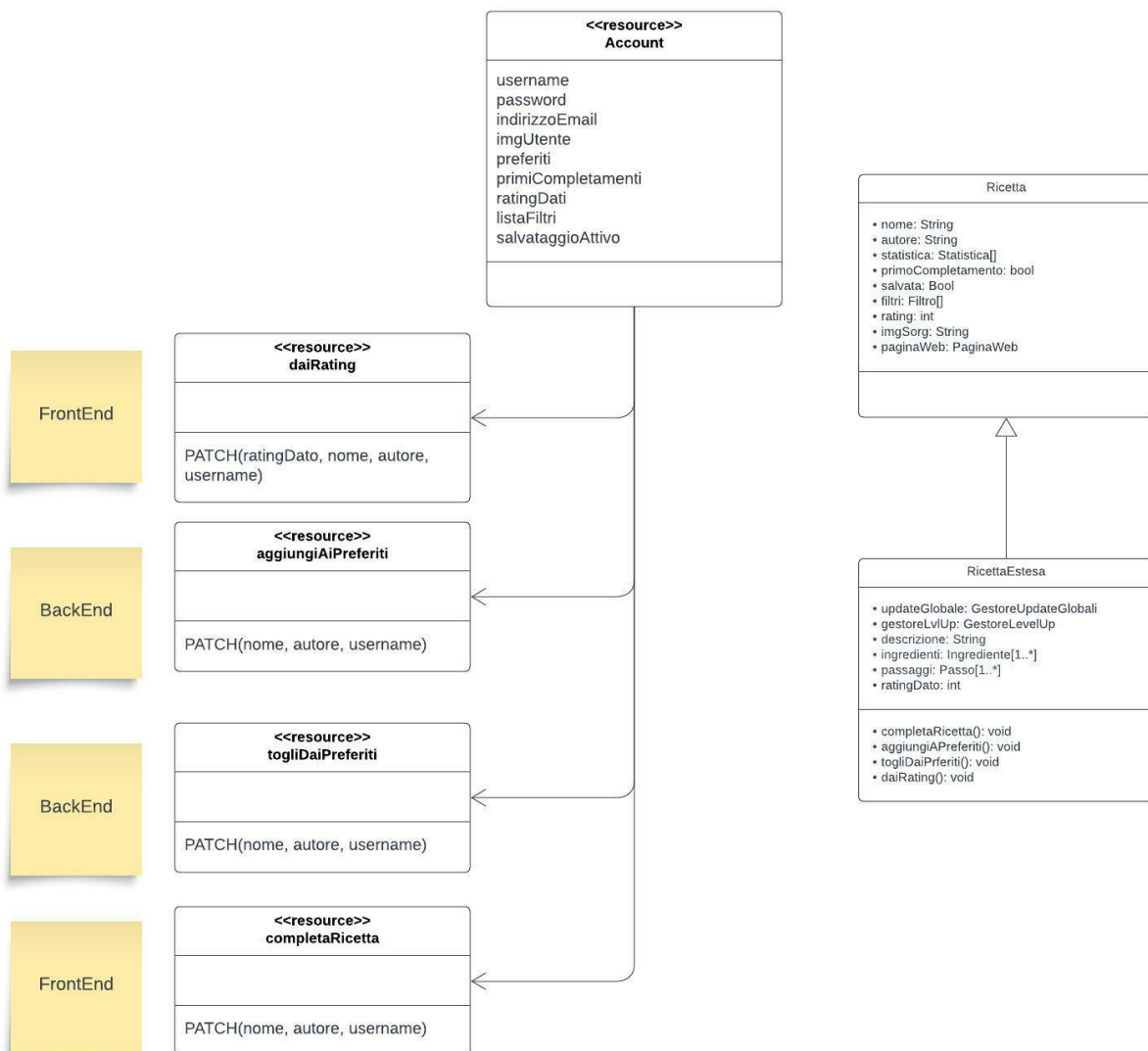


Figura 17: Classi e Resources Extraction Diagram interazione con ricetta

Diagramma Resources Extraction Diagram

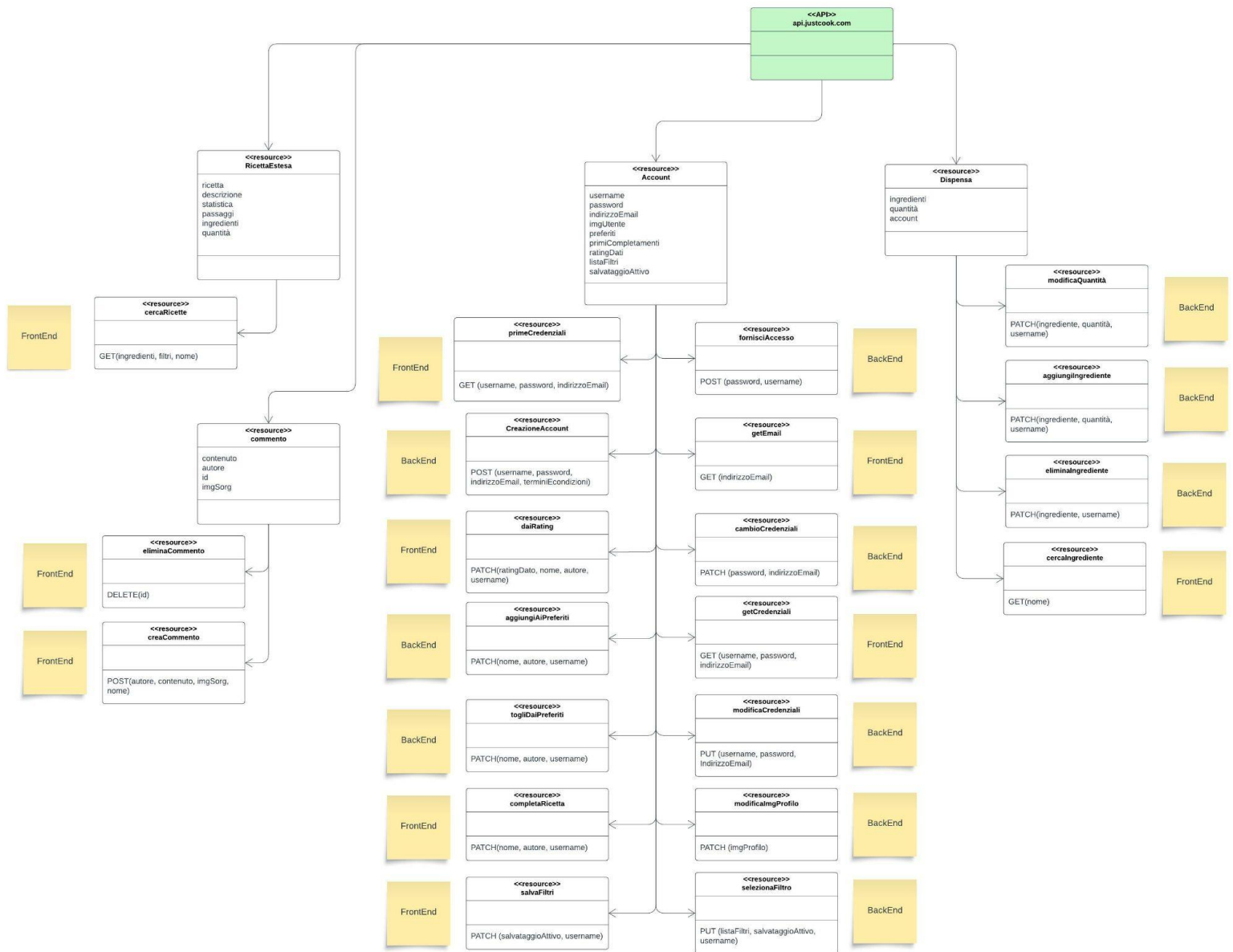


Figura 18: Resources Extraction Diagram

2.4.2. RESOURCES MODEL

In questa sezione vengono analizzati nel dettaglio i modelli delle risorse API descritte nel paragrafo precedente. I modelli vengono prima analizzati, successivamente ne vengono mostrati i diagrammi.

Descrizione modelli risorse API

1. CREDENZIALI INIZIALI

L'API **Credenziali iniziali** prende in input `primeCredenziali` che corrisponde alla password, allo username e all'indirizzo email inseriti dall'utente nella pagina di registrazione. La preleva attraverso il metodo HTTP `GET`. Se le istruzioni vengono eseguite correttamente viene ritornato il messaggio `200 OK` e vengono passate le credenziali a chi si occupa di gestire la creazione dell'account (risorsa API 2). Nel caso non venissero trovati i dati ricercati viene ritornato il codice di risposta `404 NOT FOUND`. Invece se la procedura presenta degli errori o i dati non sono validi il codice è `400 BAD REQUEST`.

2. CREAZIONE ACCOUNT

L'API **Registrazione** si occupa di creare l'account attraverso il metodo HTTP `POST`. Per farlo prende in input `primeCredenziali` (risorsa API 1) e `terminiEcondizioni` (variabili booleana). Quest'ultima ha valore `true` se l'utente ha accettato i termini e le condizioni nella pagina della registrazione, false altrimenti. Se la procedura è corretta il codice di risposta `201 CREATED` e verrà ritornato il nuovo account. Invece se sono presenti degli errori viene ritornato il messaggio `400 BAD REQUEST`.

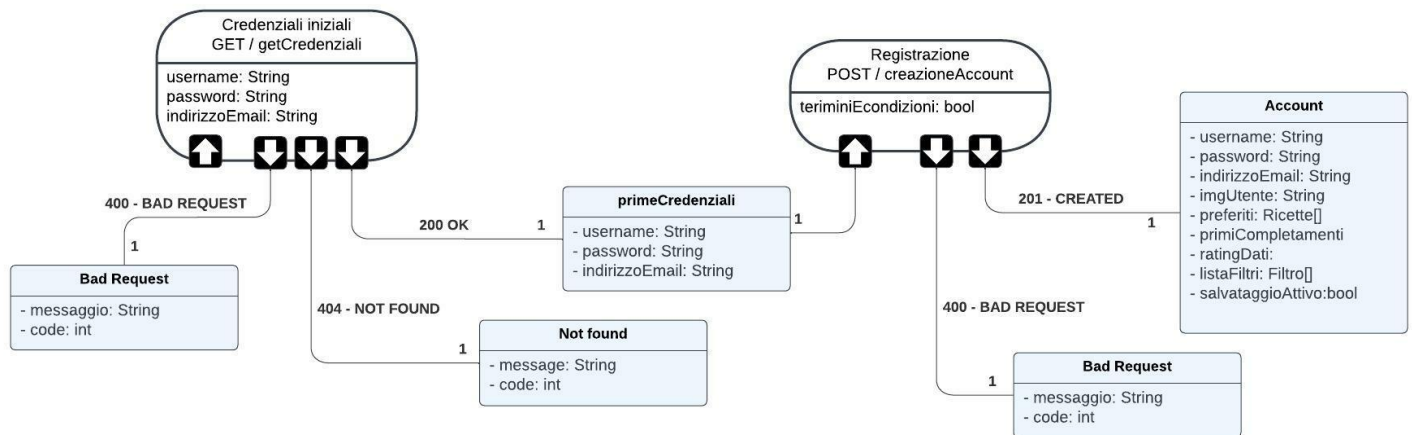


Figura 19: Resources Model Credenziali iniziali, Registrazione

3. NUOVE CREDENZIALI

L'API **Nuove credenziali** prende in input le credenziali dell'utente. Alcune di queste contengono i nuovi dati inseriti dall'utente nella pagina dell'account. Li recupera attraverso il metodo HTTP `GET`. Successivamente se l'operazione è stata eseguita correttamente verrà ritornato il messaggio `200 OK` e `nuoveCredenziali` verranno passate a chi si occupa di modificare i dati dell'account ([risorsa API 4](#)). Se i nuovi dati non vengono trovati sarà ritornato il codice di risposta `404 NOT FOUND`. Invece se l'operazione presenta degli errori o le nuove credenziali non sono valide il codice sarà `400 BAD REQUEST`.

4. MODIFICA CREDENZIALI

L'API **Modifica Credenziali** si occupa di modificare i dati dell'account attraverso il metodo HTTP `PUT`. Prende in input `nuoveCredenziali` ([risorsa API 3](#)). Se l'operazione avviene correttamente viene ricevuto il messaggio `200 OK` e `Account` con i dati modificati, altrimenti `400 BAD REQUEST`.

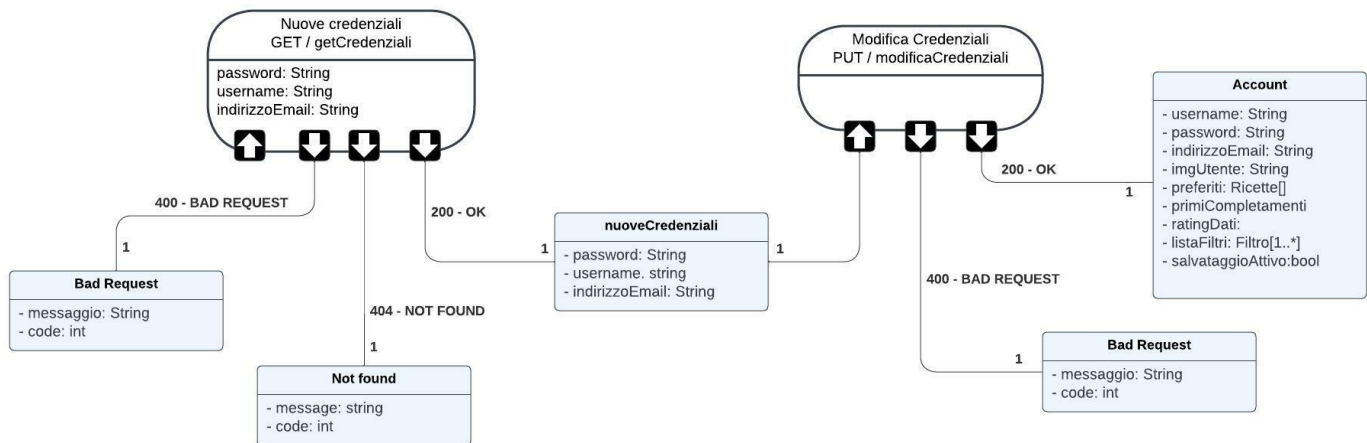


Figura: Resources Model Nuove credenziali, Modifica credenziali

5. MODIFICA IMMAGINE PROFILO

L'API **Modifica Immagine profilo** attraverso il metodo HTTP **PATCH** modifica l'immagine profilo. Prende in input `nuovaImgProfilo` che corrisponde alla nuova foto e `username` per identificare l'account da modificare. Se l'operazione è eseguita correttamente viene ritornato il messaggio **200 OK** e **Account** con la foto modificata. Se l'operazione non è eseguita correttamente viene ritornato il codice di risposta **400 BAD REQUEST**. Se l'immagine non viene trovata il codice è **404 NOT FOUND**.

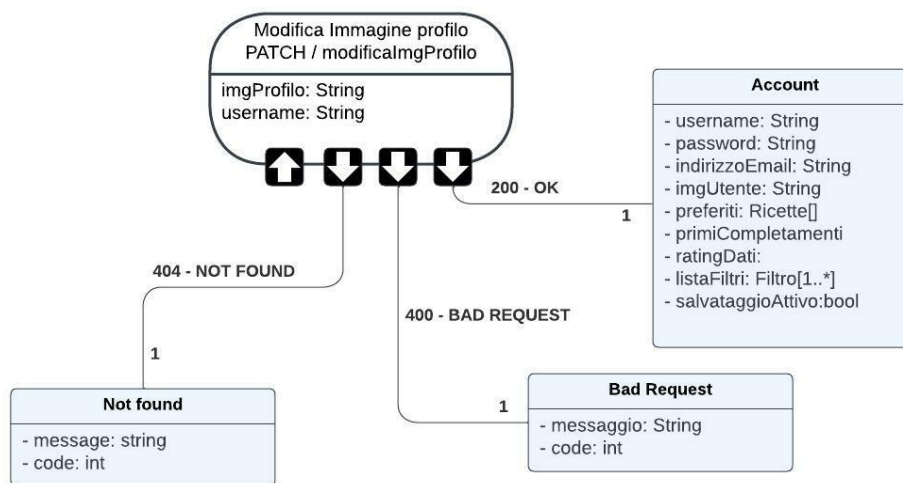


Figura 20: Resources Model Modifica Immagine profilo

6. SALVATAGGIO FILTRI

L'API **Salvataggio filtri** prende in input la variabile booleana `salvataggioAttivo`. Quest'ultima ha valore `true` se l'utente attiva il salvataggio dei filtri di ricerca, `false` altrimenti. L'altro parametro di input è `username`. Serve per identificare l'account di cui si vuole cambiare il salvataggio dei filtri. La modifica avviene attraverso il metodo HTTP `PATCH`. Se l'operazione eseguita è corretta viene ritornato il messaggio `200 OK` e `salvataggioAttivo` viene mandato a chi si occupa della selezione dei filtri ([risorsa API 7](#)). Invece, se il processo è errato, viene inviato il codice di risposta `400 BAD REQUEST`.

7. SELEZIONA FILTRI

L'API **Seleziona filtri** utilizza il metodo HTTP `PUT` per cambiare i filtri da tenere nella ricerca delle ricette. Prende in input `salvataggioAttivo` ([API 6](#)), `username` (per identificare l'account da modificare) e `listaFiltri`. Quest'ultima contiene tutti i filtri presenti in JustCook. Se l'operazione di selezionamento va a buon fine viene ritornato il messaggio `200 OK` e `Account` con `listaFiltri` modificata. Invece, se l'operazione non va a buon fine, viene ricevuto il codice di risposta `400 BAD REQUEST`.

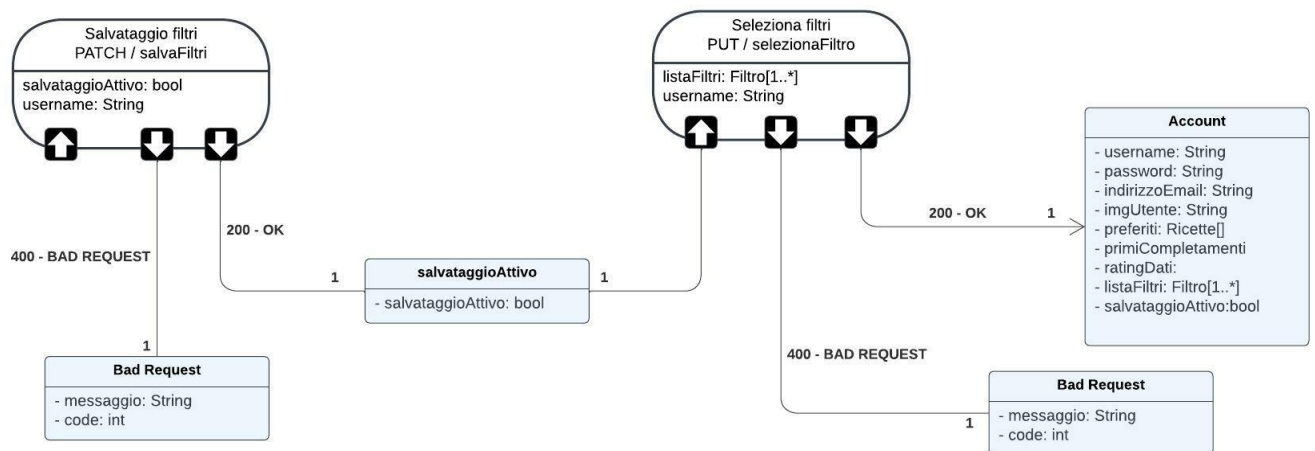


Figura 21: Resources Model Salvataggio filtri, Seleziona filtri

8. LOGIN

L'API **Login** fornisce l'accesso all'utente nella fase di login attraverso il metodo HTTP POST. Riceve in input `password` e `username`. Se quest'ultime sono corrette (corrispondono ad un account) viene ritornato 200 OK con `Account`, altrimenti il codice di risposta è 400 BAD REQUEST. Se le credenziali inserite nella pagina del login dall'utente non vengono trovate il messaggio di risposta è 404 NOT FOUND.

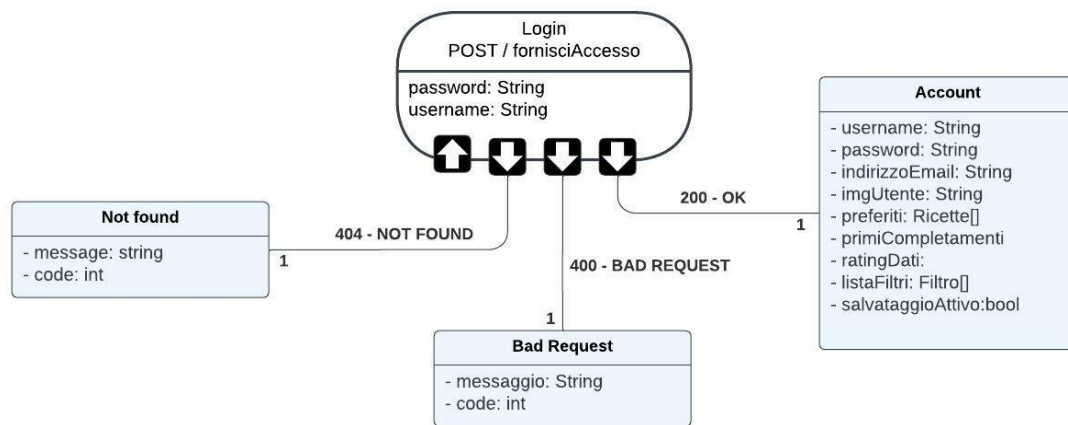


Figura 22: Resources Model Login

9. EMAIL CAMBIO PASSWORD

L'API **Email cambio password** prende in input `indirizzoEmail` contenente l'indirizzo email dell'utente inserito nella pagina del login. Viene prelevato attraverso il metodo HTTP GET. Se viene trovato il codice di risposta è 200 OK e `indirizzoEmail` viene inviato a chi si occupa di gestire il cambio password (risorsa API 10). Invece, se non viene trovato l'indirizzo email, il codice di risposta è 404 NOT FOUND. Se l'indirizzo email non è associato ad un account il codice è 400 BAD REQUEST.

10. CAMBIO PASSWORD LOGIN

L'API **Cambio password login** modifica la password dell'utente attraverso il metodo HTTP PATCH. Prende in input `password` (contiene la nuova password) e `indirizzoEmail` (risorsa API 9). Se l'operazione è corretta viene ritornato 200 OK con `Account`, altrimenti 400 BAD REQUEST (password nuova non valida). Se la password non è stata inserita il codice è 404 NOT FOUND.

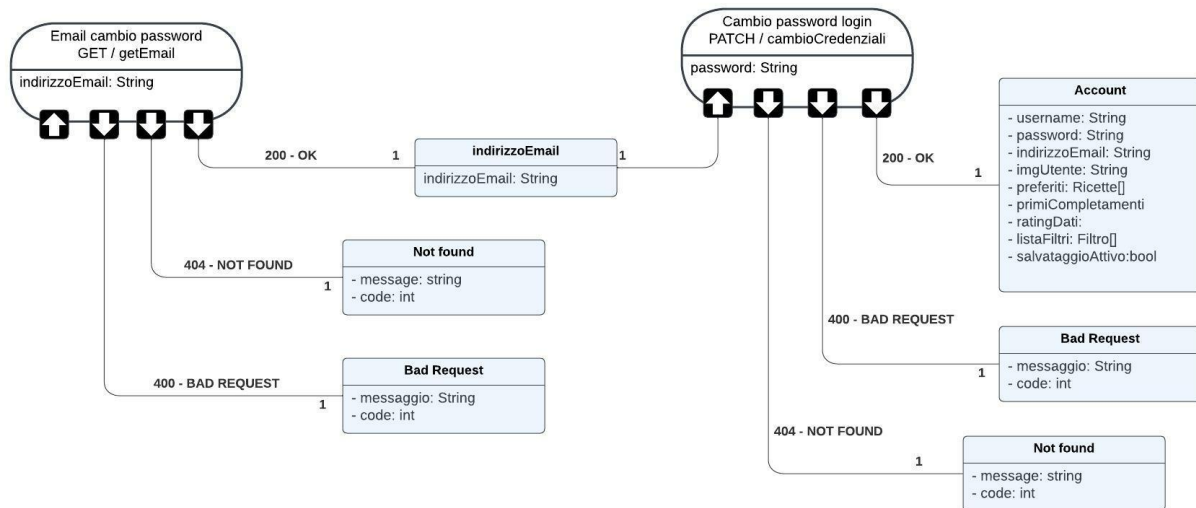


Figura 23: Resources Model Cambio password login, Email cambio password

11. CREA COMMENTO

L'API **Crea commento** prende in input il nome della ricetta su cui commentare. Inoltre prende in input il contenuto del commento, l'autore di quest'ultimo e l'eventuale immagine allegata. Poi attraverso il metodo HTTP `POST` viene aggiunto un nuovo commento alla ricetta nel database. Viene ritornato il messaggio di `201 CREATED` e `Commento` alla buona riuscita dell'operazione e l'errore `404 NOT FOUND` se non viene trovata la ricetta da commentare. Se l'operazione presenta degli errori il messaggio ritornato è `400 BAD REQUEST`.

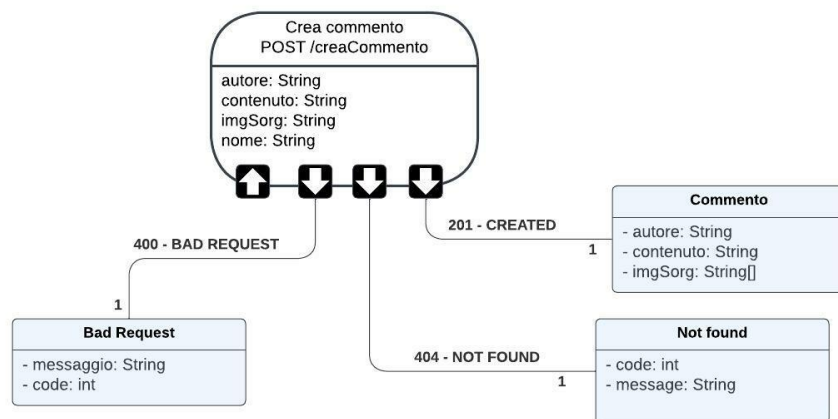


Figura 24: Resources Model Crea commento

12. ELIMINA COMMENTO

L'API **Elimina commento** prende in input l'id del commento da eliminare, poi attraverso il metodo HTTP `DELETE` viene eliminato il commento che corrisponde all'id passato dalla relativa ricetta nel database. Viene ritornato un messaggio di `200 OK` alla buona riuscita dell'operazione e l'errore `404 NOT FOUND` se non viene trovata la ricetta in cui è presente il commento. Invece se l'operazione presenta degli errori il codice è `400 BAD REQUEST`.

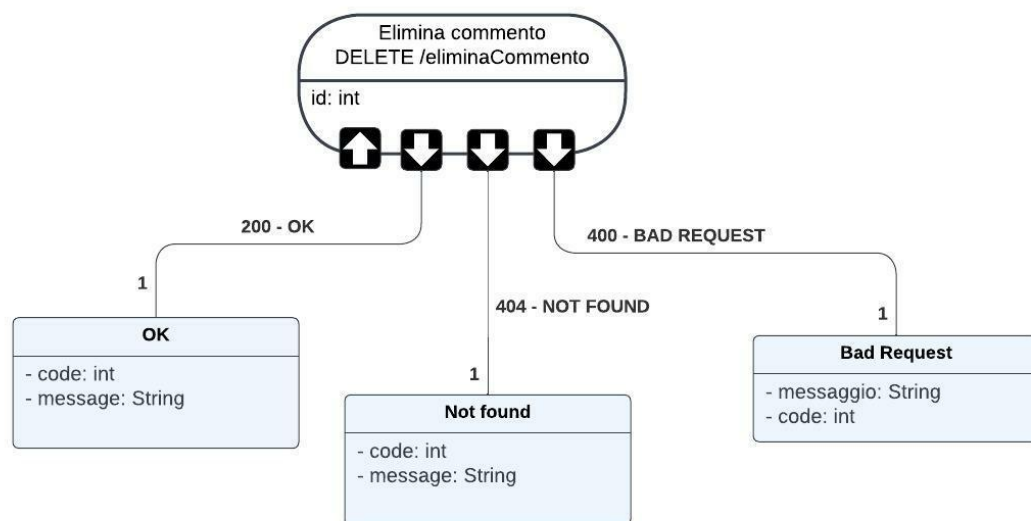


Figura 25: Resources Model Elimina commento

13. MODIFICA QUANTITÀ

L'API **Modifica quantità** viene implementata per permettere all'utente di diminuire o aumentare la quantità di un ingrediente alla pressione di un tasto sviluppato in front-end. Per funzionare richiede che in input gli venga passato l'ingrediente e la nuova quantità. Inoltre prende in ingresso `username`. Quest'ultimo serve per individuare la dispensa dell'utente che sta effettuando la modifica. Procede a fare una HTTP `PATCH` sulla quantità di un ingrediente. A operazione eseguita viene restituito un codice `200 OK` per confermare l'esito e restituire la dispensa modificata. In caso l'operazione subisca degli errori verrà restituito il codice `400 BAD REQUEST` con collegato messaggio di errore. In caso in cui si provi a modificare la quantità di un ingrediente non in dispensa il codice ritornato sarà `404 NOT FOUND`.

14. AGGIUNGI INGREDIENTE

Per aggiungere un determinato ingrediente nella dispensa di un utente si sviluppa l'API **Aggiungi ingrediente** di tipo HTTP `PATCH`. In ingresso viene richiesto l'ingrediente, la quantità associata da aggiungere e l'username dell'utente che sta effettuando l'aggiunta.. La funzione `PATCH` verrà applicata alla quantità legata a un determinato ingrediente e, una volta svolta la business logic, viene restituito in output il codice `200 OK` per confermare l'operazione, allegando la dispensa modificata. Come per altri metodi, in caso l'operazione subisca degli errori verrà invece restituito il codice `400 BAD REQUEST`. Nel caso in cui l'ingrediente che si prova ad aggiungere non esista, verrà restituito codice `404 NOT FOUND` con allegato messaggio di avvenuto errore.

15. ELIMINA INGREDIENTE

L'API **Elimina ingrediente** utilizza il metodo HTTP `PATCH` per modificare i dati della dispensa. A questa API viene passato in input l'ingrediente che si desidera eliminare e l'username dell'utente che sta effettuando l'operazione. Al termine dell'esecuzione si otterrà un codice `200 OK` che conferma l'eliminazione dell'ingrediente e restituisce la dispensa modificata. In caso la procedura sia errata si ottiene il codice `400 BAD REQUEST`, se invece si passa un ingrediente o una dispensa non validi viene ritornato codice `404 NOT FOUND`.

16. RICERCA INGREDIENTE

L'API **Cerca ingrediente** prende in input il nome dell'ingrediente inserito dall'utente nell'apposita barra di ricerca per aggiungerlo nella dispensa . Per fare questo l'API utilizza il metodo HTTP `GET`. Se l'operazione va a buon fine viene restituito il messaggio `200 OK` con `Ingrediente`. Se il nome del cibo non è stato inserito correttamente nella barra di ricerca viene ritornato l'errore `404 NOT FOUND`. Infine se l'operazione subisce degli errori o l'ingrediente cercato non è presente nel database il messaggio ritornato è `400 BAD REQUEST`.

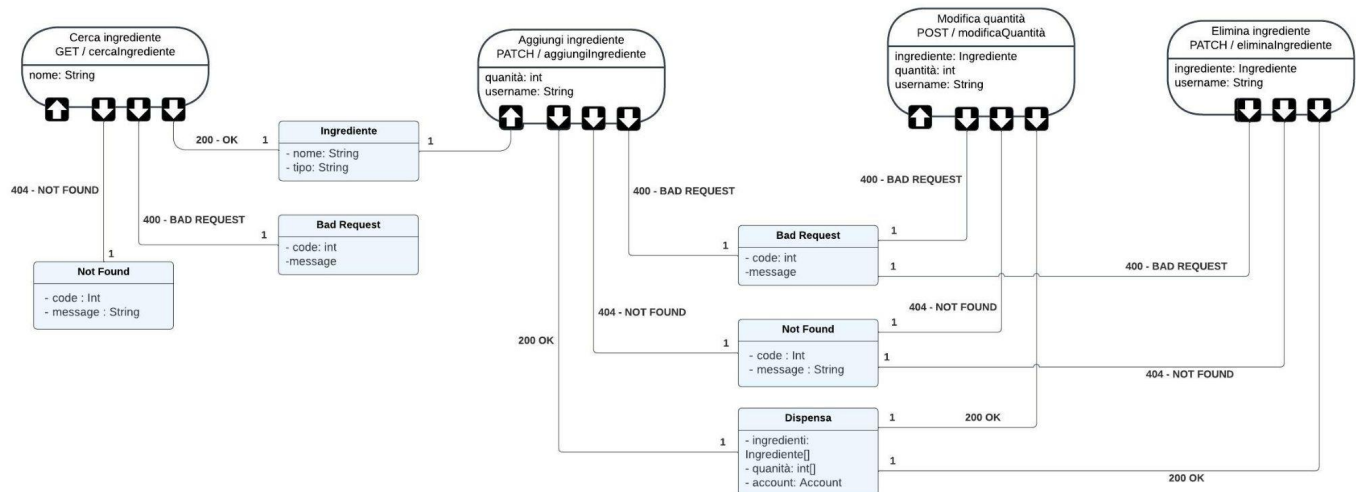


Figura 26: Resources Model Cerca ingrediente, Aggiungi ingrediente, Modifica quantità, Elimina ingrediente

17. CERCA RICETTE

L'API **Cerca ricette** prende in input `ingredienti` che rappresenta gli ingredienti nella dispensa dell'utente, `filtri` che corrisponde ai filtri selezionati nella ricerca e `nome` che è il nome del piatto ricercato (può anche non essere specificato). Se l'operazione non presenta errori viene ricevuto il codice di risposta `200 OK` con `listaRicette` che rappresenta i risultati della ricerca. Se ci sono stati degli errori durante l'esecuzione il codice ricevuto è `500 INTERNAL SERVER ERROR`.

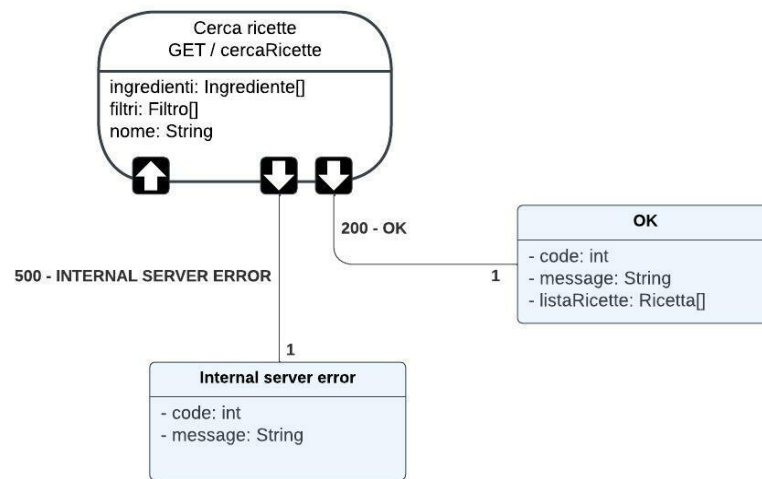


Figura 27: Resources Model Cerca ricette

18. COMPLETA RICETTA

L'API **Completa ricetta** prende in input il nome e l'autore della ricetta che è stata appena "completata". Prende in ingresso anche `username` per identificare l'utente che ha completato la ricetta. L'API attraverso il metodo HTTP `PATCH` aggiorna i completamenti dell'account dell'utente. All'esecuzione corretta delle istruzioni viene ritornato un messaggio di `200 OK`, mentre se non viene trovata la ricetta cercata si riscontra l'errore `404 NOT FOUND`. Inoltre se la procedura presenta errori viene ritornato il codice `500 INTERNAL SERVER ERROR`.

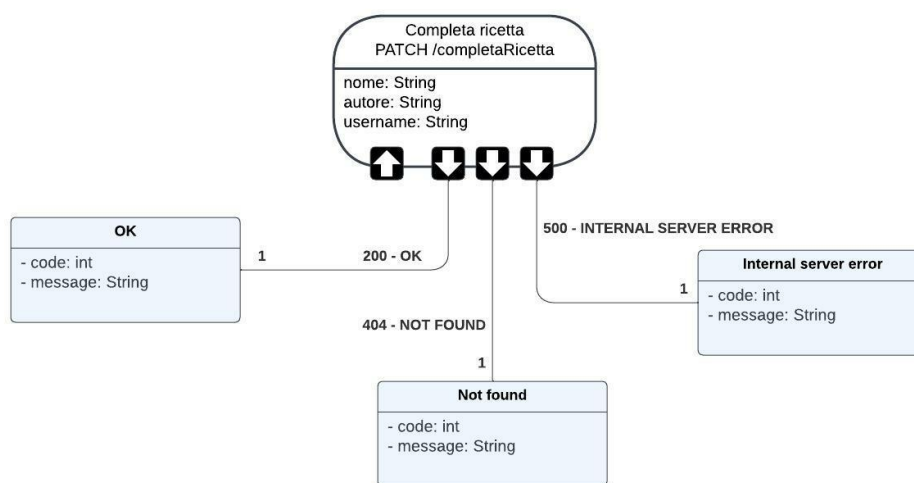


Figura 28: Resources Model Completa ricetta

19. AGGIUNGI AI PREFERITI

L'API **Aggiungi ai preferiti** prende in input il nome e l'autore della ricetta da aggiungere tra i preferiti dell'utente. Inoltre prende in ingresso username per identificare l'utente che ha aggiunto la ricetta come preferita. Attraverso il metodo HTTP `PATCH` viene aggiornato l'elenco delle ricette preferite dell'account dell'utente. Viene ritornato un messaggio di `200 OK` alla buona riuscita dell'operazione e l'errore `404 NOT FOUND` se non viene trovata la ricetta cercata. Se la ricetta è già presente nei preferiti viene ritornato il codice `400 BAD REQUEST`. Invece se la procedura presenta degli errori il codice ritornato è `500 INTERNAL SERVER ERROR`.

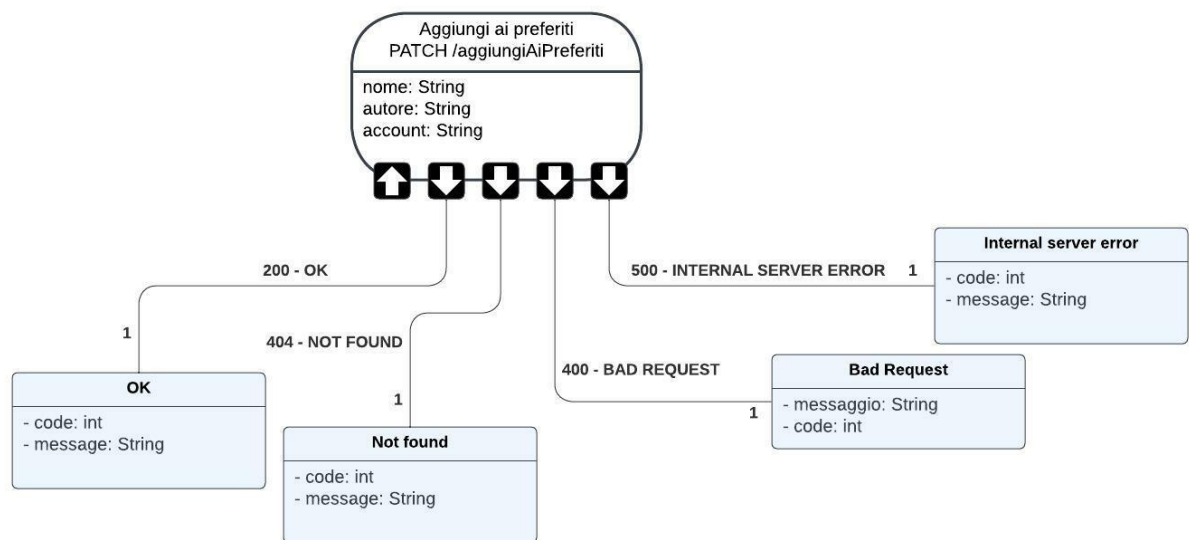


Figura 29: Resources Diagram Aggiungi ai preferiti

20. TOGLI DAI PREFERITI

L'API **Togli dai preferiti** prende in input il nome e l'autore della ricetta da togliere dai preferiti dell'utente (identificato attraverso username). Attraverso il metodo HTTP `PATCH` viene aggiornata la lista dei preferiti dell'utente rimuovendo la ricetta in questione. Viene ritornato un messaggio `200 OK` alla buona riuscita dell'operazione e l'errore `404 NOT FOUND` se non viene trovata la ricetta da togliere dai preferiti. Inoltre se l'operazione presenta degli errori la risposta ricevuta è `500 INTERNAL SERVER ERROR`.

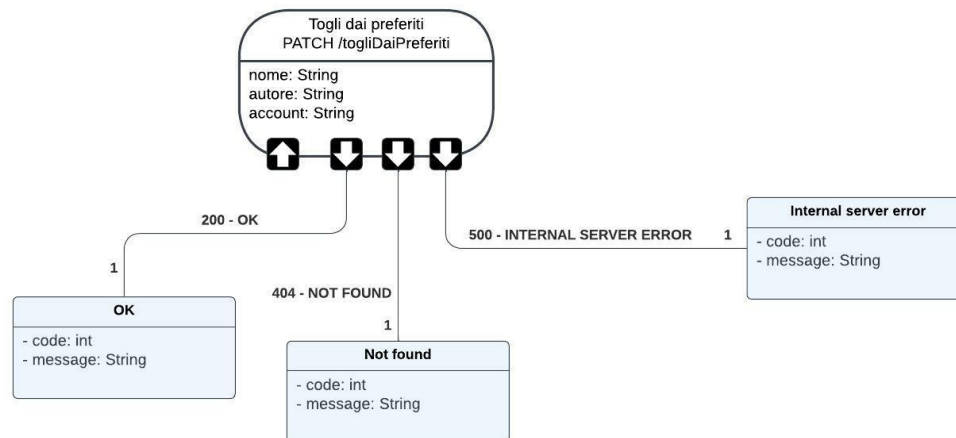


Figura 30: Resources Model Togli dai preferiti

21. AGGIUNGI RATING

L'API **Aggiungi rating** prende in input il nome e l'autore della ricetta di cui salvare il rating dato dall'utente (identificato da `username`) e la valutazione stessa appena fornita come intero da 0 a 5. Attraverso il metodo HTTP `PATCH` viene aggiunto il nuovo rating alla ricetta nel database e aggiunto il rating all'account dell'utente. Al buon termine dell'operazione viene inviato un messaggio di `200 OK`. Se non viene trovata la ricetta cercata si riscontra l'errore `404 NOT FOUND`. Se il rating è già stato dato viene ritornato il codice `400 BAD REQUEST`. Infine se la procedura presenta degli errori il messaggio di risposta è `500 INTERNAL SERVER ERROR`.

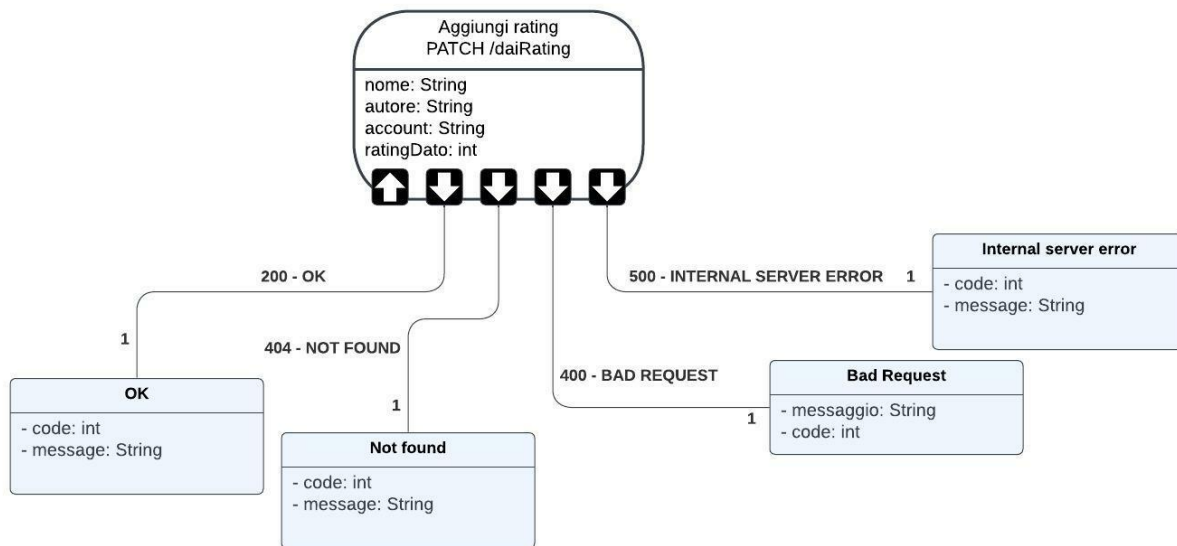


Figura 31: Resources Model Aggiungi rating

3. API DOCUMENTATION
4. FRONTEND IMPLEMENTATION
5. GITHUB REPOSITORY AND DEPLOYMENT INFO
6. TESTING