UNIVERSITÀ DI TRENTO

JustCook

03/12/22

SVILUPPO APPLICAZIONE

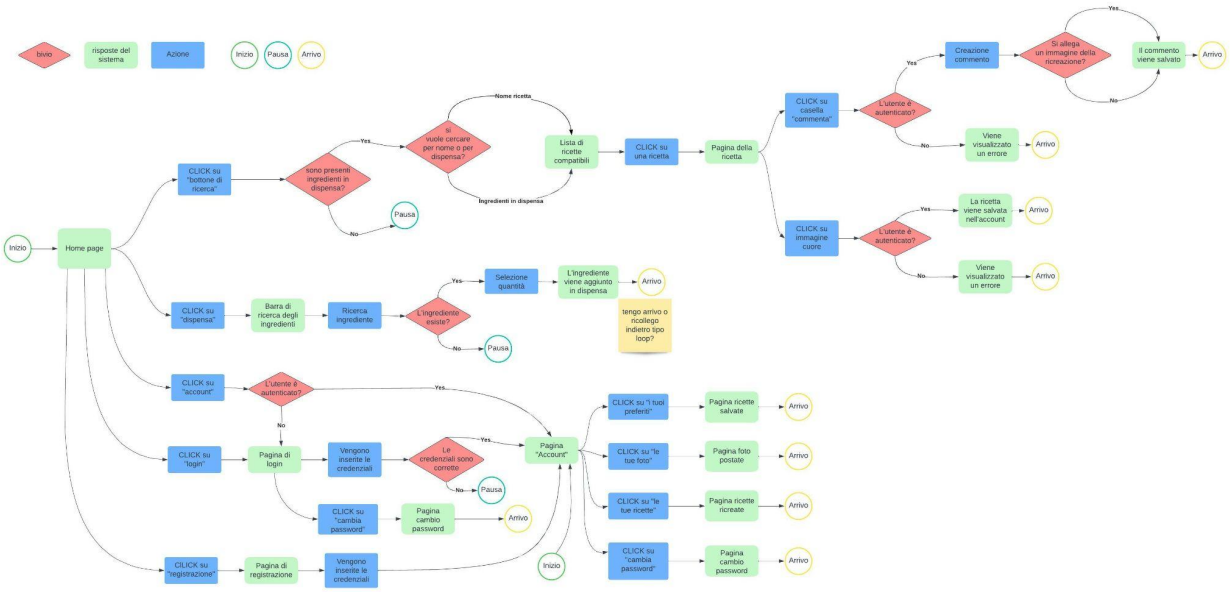
UNIVERSITÀ DI TRENTO

Indice

UNIVERSITÀ DI TRENTO

SCOPO DEL DOCUMENTO

1. USER FLOWS


bivio
risposte del sistema
Azione
Inizio
Pausa
Arrivo
Inizio
Home page
CLICK su "bottone di ricerca"
sono presenti ingredienti in dispensa?
Yes
No
Pausa
si vuole cercare per nome o per dispensa?
Nome ricetta
Ingredienti in dispensa
Lista di ricette compatibili
CLICK su una ricetta
Pagina della ricetta
CLICK su casella "commenta"
L'utente è autenticato?
Yes
No
Creazione commento
Si allega un immagine della ricreazione?
Yes
No
Il commento viene salvato
Arrivo
Viene visualizzato un errore
Arrivo
CLICK su immagine cuore
L'utente è autenticato?
Yes
No
La ricetta viene salvata nell'account
Arrivo
Viene visualizzato un errore
Arrivo
CLICK su "dispensa"
Barra di ricerca degli ingredienti
Ricerca ingrediente
L'ingrediente esiste?
Yes
No
Pausa
Selezione quantità
L'ingrediente viene aggiunto in dispensa
Arrivo
tengo arrivo o ricollego indietro tipo loop?
CLICK su "account"
L'utente è autenticato?
Yes
No
CLICK su "login"
Pagina di login
Vengono inserite le credenziali
Le credenziali sono corrette
Yes
No
Pausa
Pagina "Account"
CLICK su "cambia password"
Pagina cambio password
Arrivo
CLICK su "registrazione"
Pagina di registrazione
Vengono inserite le credenziali
Inizio
CLICK su "I tuoi preferiti"
Pagina ricette salvate
Arrivo
CLICK su "le tue foto"
Pagina foto postate
Arrivo
CLICK su "le tue ricette"
Pagina ricette ricreate
Arrivo
CLICK su "cambia password"
Pagina cambio password
Arrivo

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

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. PAGINA REGISTRAZIONE - 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. Per ottenere le prime credenziali appena inserite, l'API analizzata possiede il metodo HTTP `GET`.

2. PAGINA REGISTRAZIONE - METODO GET TERMINI E CONDIZIONI

   Analizzando la classe `PaginaRegistrazione` e il metodo `getTerminiECondizioni` è stata ideata l'API **Termini e condizioni**. L'utente per creare un proprio account deve accettare i termini e le condizione per il trattamento dei dati nella pagina della registrazione. Per ottenere questa informazione, l'API in considerazione utilizza il metodo HTTP `GET`.

3. REGISTRAZIONE - METODO CREAZIONE ACCOUNT

   Analizzando il metodo `creazioneAcount` 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.

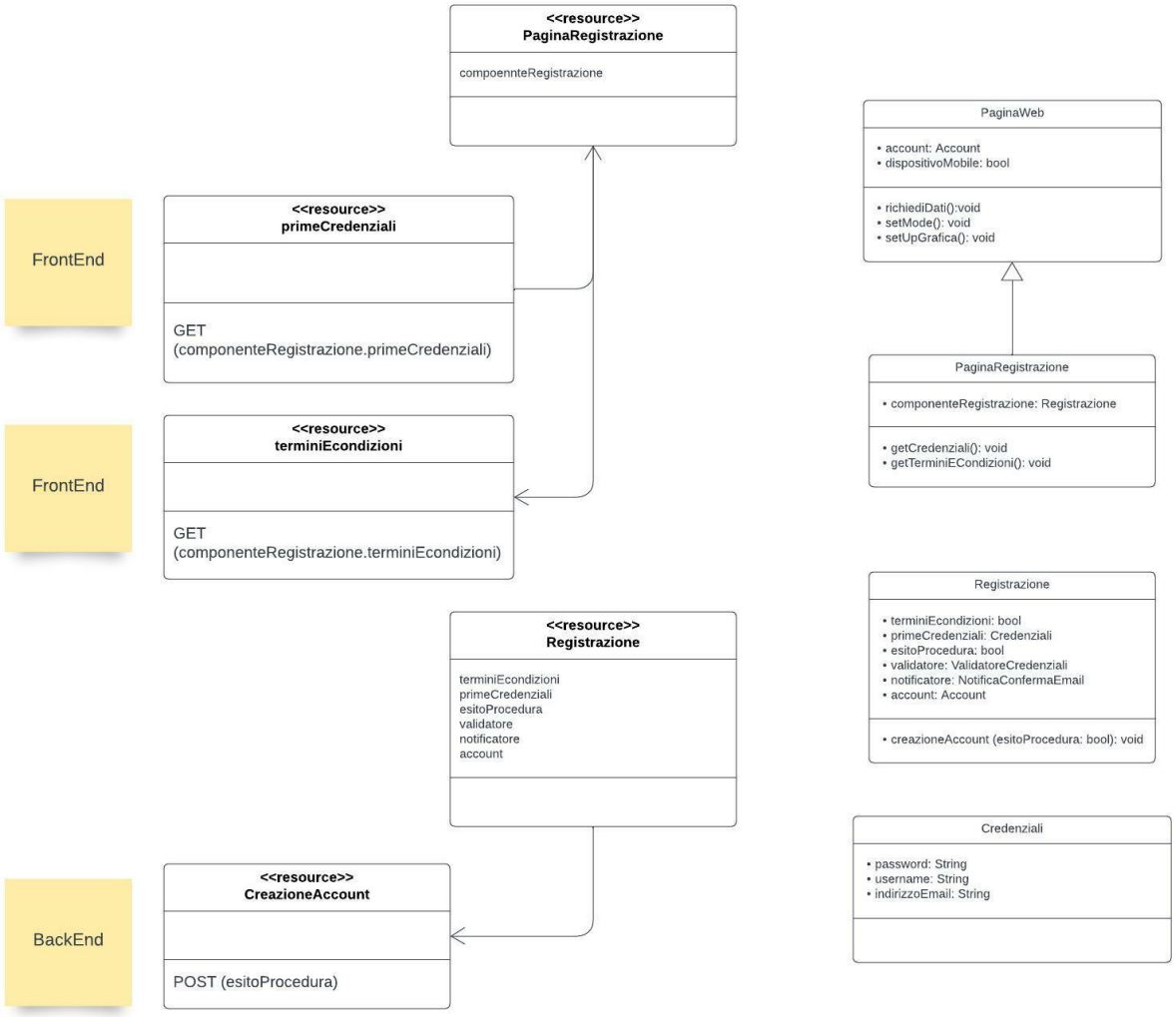


Figura: Classi e Resources Extraction Diagram fase registrazione

4. ACCOUNT - 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.

5. GESTORE MODIFICA DATI - MODIFICA CREDENZIALI

   La classe `GestoreModificaDati` possiede il metodo `modifcaCredenziali`. 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`.

6. ACCOUNT - GET IMG PROFILO

   L'utente può modificare la propria immagine profilo dalla pagina dell'account. La nuova foto viene ottenuta tramite il metodo `getImgProfilo` della classe `Account`. Per questo è stata ideata l'API **Nuova Img Profilo** con il metodo HTTP `GET`.

7. GESTORE MODIFICA DATI - 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`.

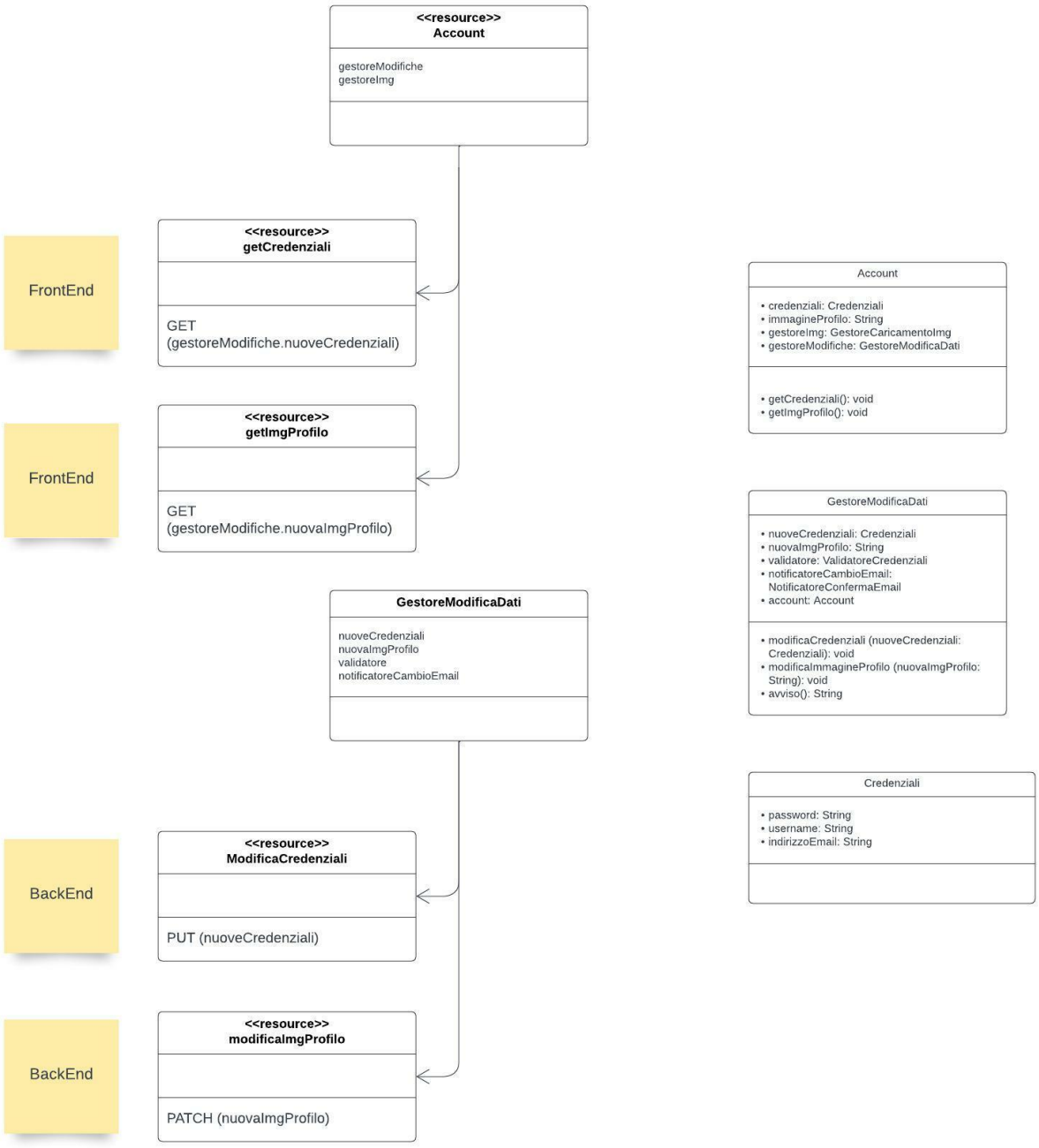


Figura: Classi e Resources Extraction Diagram fase modifica dati pagina account

8. SALVATAGGIO FILTRI - 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`.

9. SALVATAGGIO FILTRI - 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`.

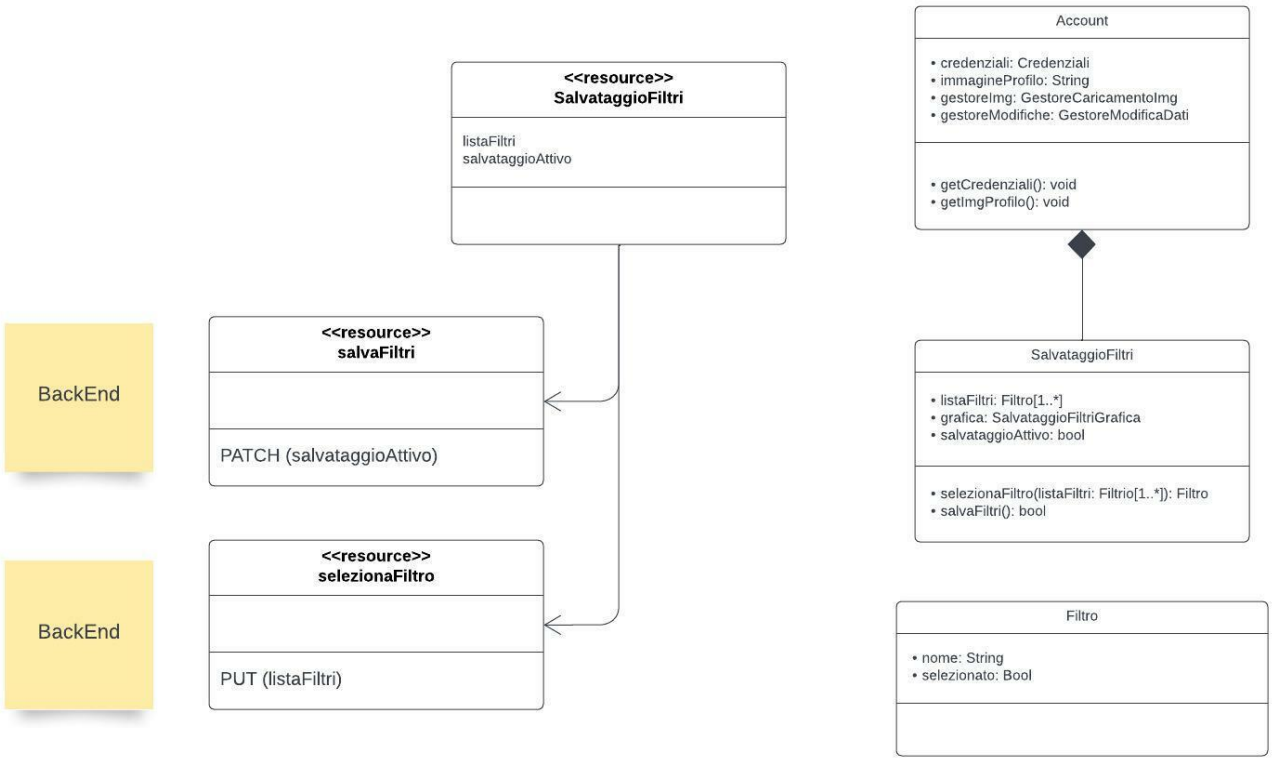


Figura: Classe e Resources Extraction Diagram fase salvataggio filtri

10. PAGINA LOGIN - GET CREDENZIALI

Analizzando la classe `PaginaLogin` e il suo metodo `getCredenziali` è stata ideata l'API **Credenziali inserite**. L'utente per accedere inserisce le credenziali (username, password) in delle apposite barre. Tali informazioni vengono prelevate dall'API attraverso il metodo HTTP `GET`.

11. LOGIN - 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`.

12. PAGINA LOGIN - GET PASSWORD

Analizzando la classe `PaginaLogin` e il suo metodo `getPassword` è stata creata l'API **Nuova password**. L'utente può modificare la password inserendo quella nuova in un'apposita barra. Questo dato viene ottenuto dall'API considerata attraverso il metodo HTTP `GET`.

13. PAGINA LOGIN - 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. Successivamente si utilizzerà un'email di conferma per completare la procedura. Per prelevare l'indirizzo email l'API considerata utilizza il metodo HTTP `GET`.

14. LOGIN - 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`.

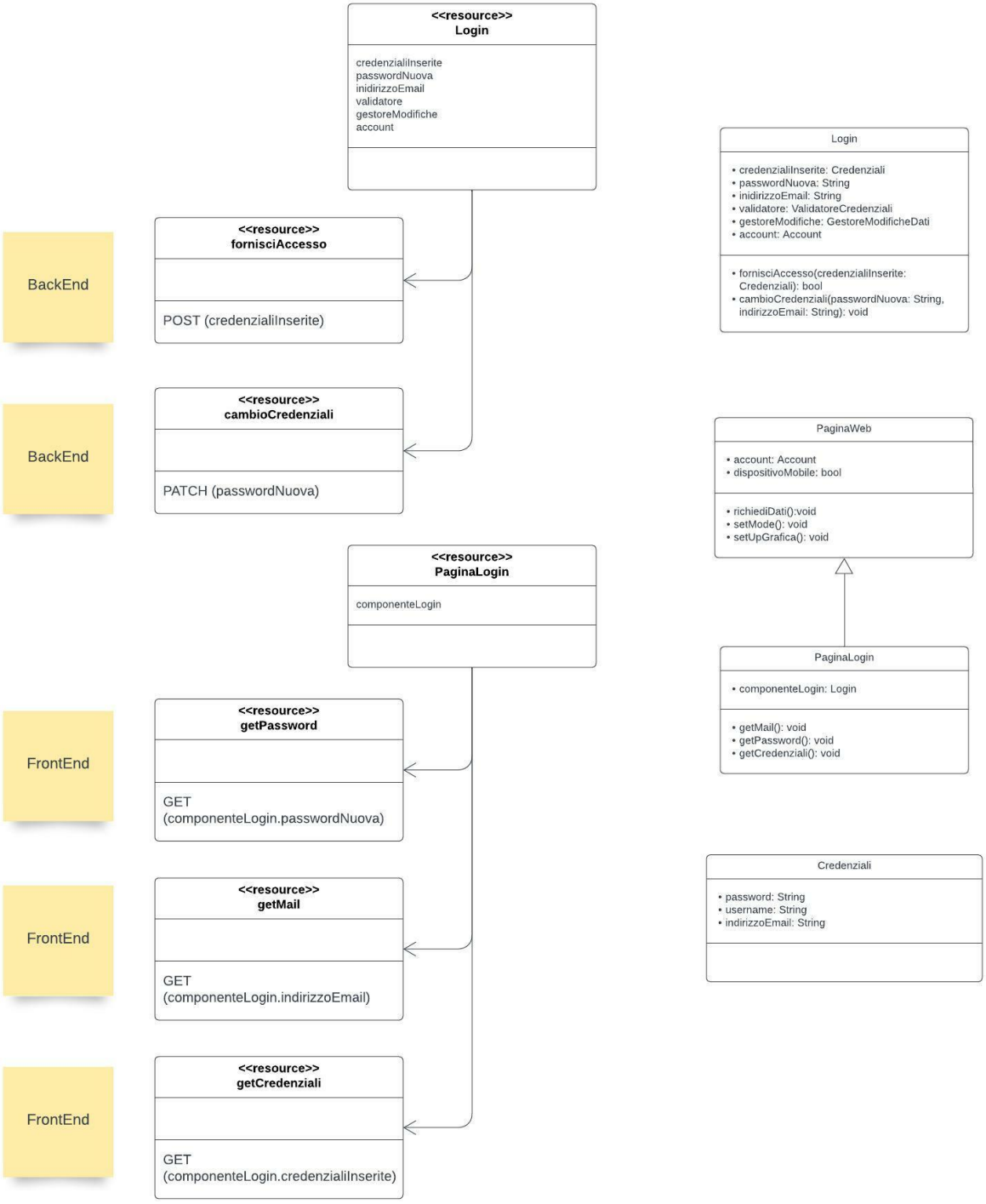


Figura: Classi e Resources Extraction Diagram fase login

15. RICETTA - 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.

16. RICETTA - AGGIUNGI AI 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.

17. RICETTA - 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.

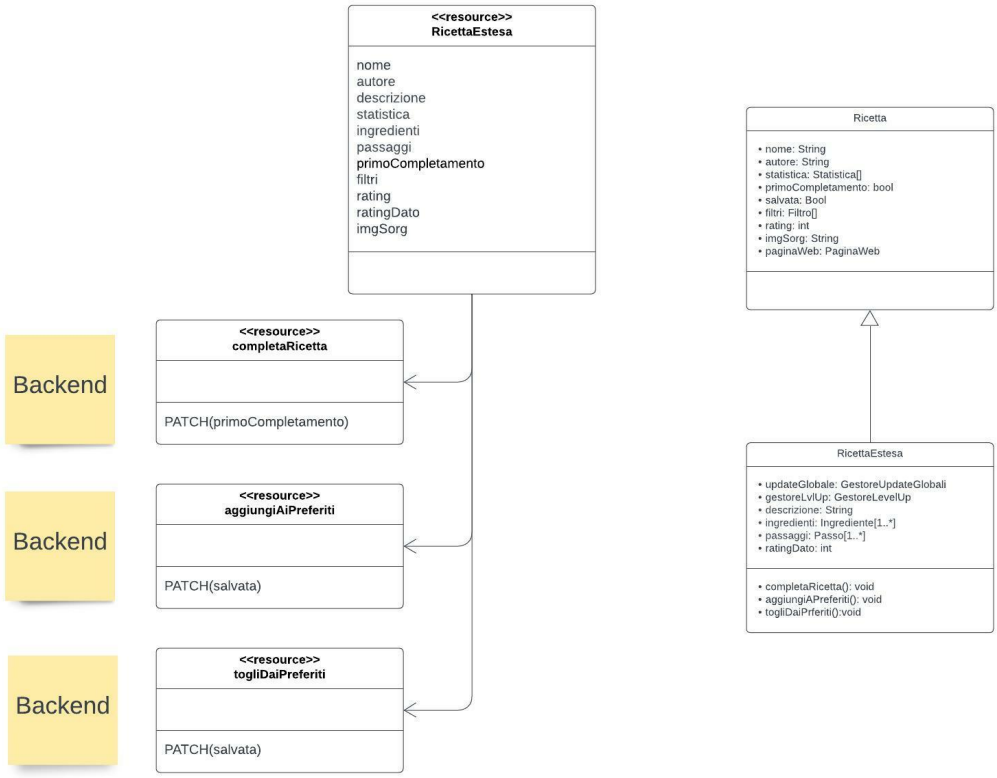


Figura: Classi e Resources Extraction Diagram fase interazione ricetta

18. GESTORE RICHIESTE PAGINE - RICHIEDE DATI

L'API **Carica dati** corrisponde al metodo `richiedeDati` della classe `GestoreRichiestePagine`. Attraverso il metodo HTTP `GET` restituisce tutti gli elementi necessari alla pagina Web richiesta dal database.

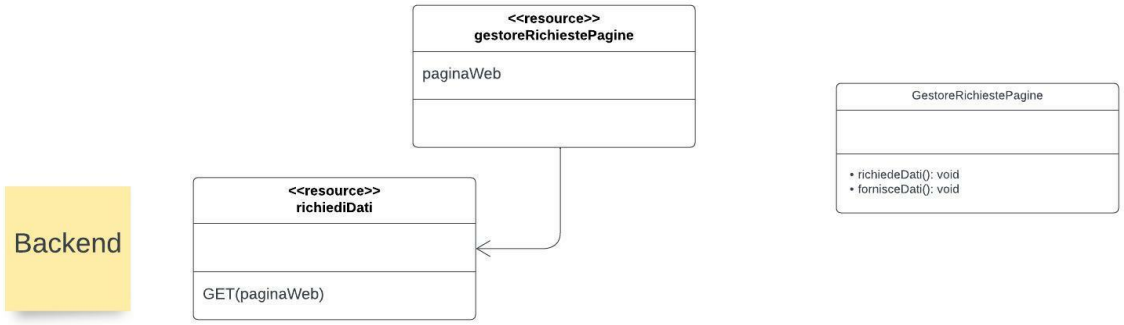


Figura: Classe e Resources Extraction Diagram fase richiesta dati

19. DISPENSA - 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.

20. DISPENSA - 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.

21. DISPENSA - 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.

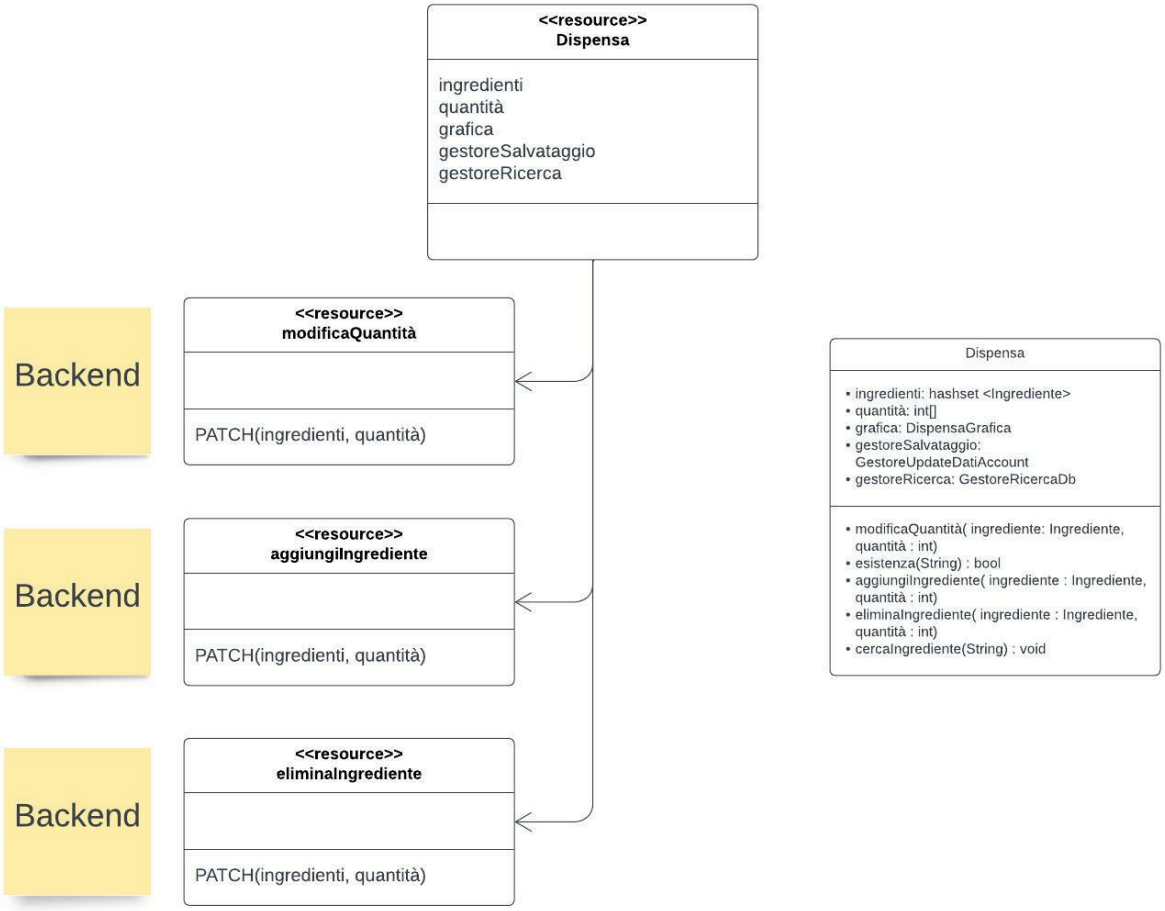


Figura: Classe e Resources Extraction Diagram fase interazione con dispensa

22. GESTORE RICERCA BD - CERCA RICETTE

A partire dalla classe `GestoreRicercaDB`, implementata per cercare e restituire ricette e ingredienti all'interno del database, viene creata l'API **Cerca ricette**. Questa sfrutta il metodo HTTP `GET` per verificare l'esistenza e restituire un insieme di ricette a partire dai parametri che prende in input, quali gli ingredienti della ricetta, l'autore o gli ingredienti usati.

23. GESTORE RICERCA DB - CERCA INGREDIENTI

Sempre parte della classe `GestoreRicercaDB`, l'API **Cerca ingredienti** ha lo scopo di verificare se un certo ingrediente esiste all'interno del database. Per fare ciò usa il metodo HTTP `GET`.

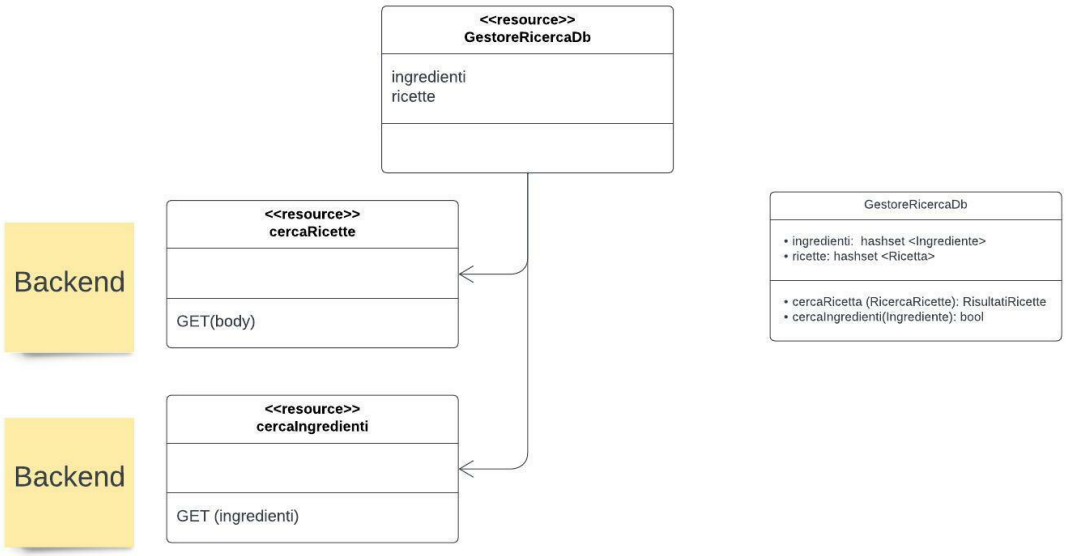


Figura: Classe e Resources Extraction Diagram fase ricerca dati nel database

24. RICERCA RICETTE - 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. In particolare questo metodo permette di cercare a partire dal nome della ricetta che si desidera leggere. Per implementare questa funzione viene usato un metodo HTTP `GET`.

25. RICERCA RICETTE - CERCA PER DISPENSA

`cercaPerDispensa` viene implementata con l'API **Cerca per dispensa** per motivi simili a quelli dell'API precedente, facente parte della stessa classe, ovvero `RicercaRicette`. La differenza sostanziale sta nel fatto che `cercaPerDispensa` permette di cercare tra le ricette a partire dagli ingredienti disponibili in dispensa e non dal nome di una pietanza. Per implementare questa API viene quindi usato il metodo HTTP `GET`.

26. RISULTATI RICETTE - GET RICETTE

L'API **Get ricette** è associata al metodo `getRicette` della classe `RisultatiRicette` che provvede a prendere dal database le ricette richieste, filtrandole utilizzando dei parametri in ingresso. Per poter usare i dati è necessario usare il metodo HTTP `GET`, così da poter selezionare fra tutte le ricette presenti nel database.

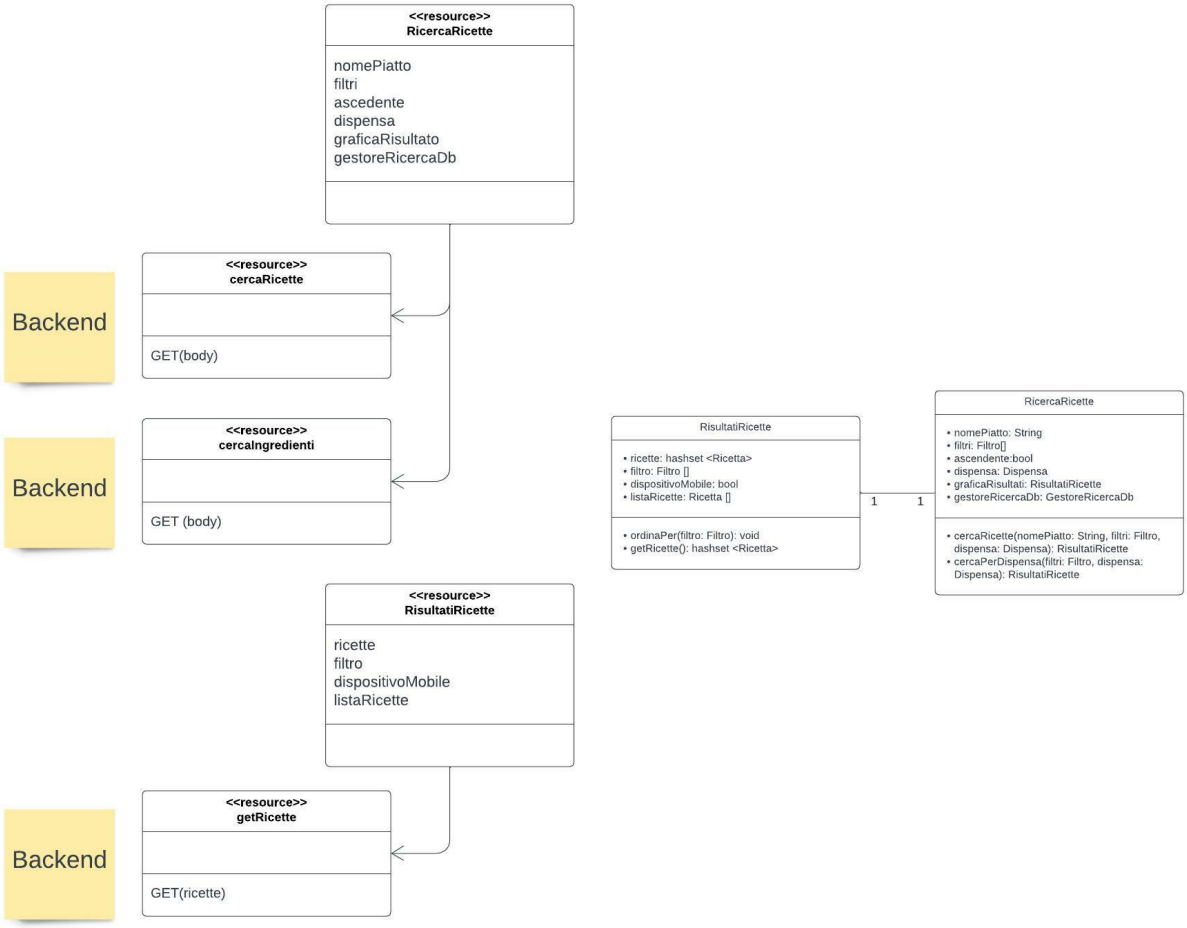


Figura: Classi e Resources Extraction Diagram fase ricerca ricette

27. GESTORE NUOVE RICETTE - GET RICETTA

Analizzando i metodi della classe `GestoreNuoveRicette`, si decide di implementare l'API **Get ricetta**, necessaria per il sistema per recuperare una data ricetta e poterla elaborare. Per fare ciò si utilizza il metodo HTTP `GET`.

28. GESTORE NUOVE RICETTE - PASSA RICETTA

Simile all'API precedente, **Passa ricetta** viene implementata per sopperire al bisogno di salvare nel database una data ricetta. Essendo questa API parte della classe `GestoreNuoveRicette`, nello specifico si occuperà di salvare nel database le ricette postate dagli utenti e approvate dagli amministratori. Per salvare queste ricette si usa il metodo HTTP `POST`.

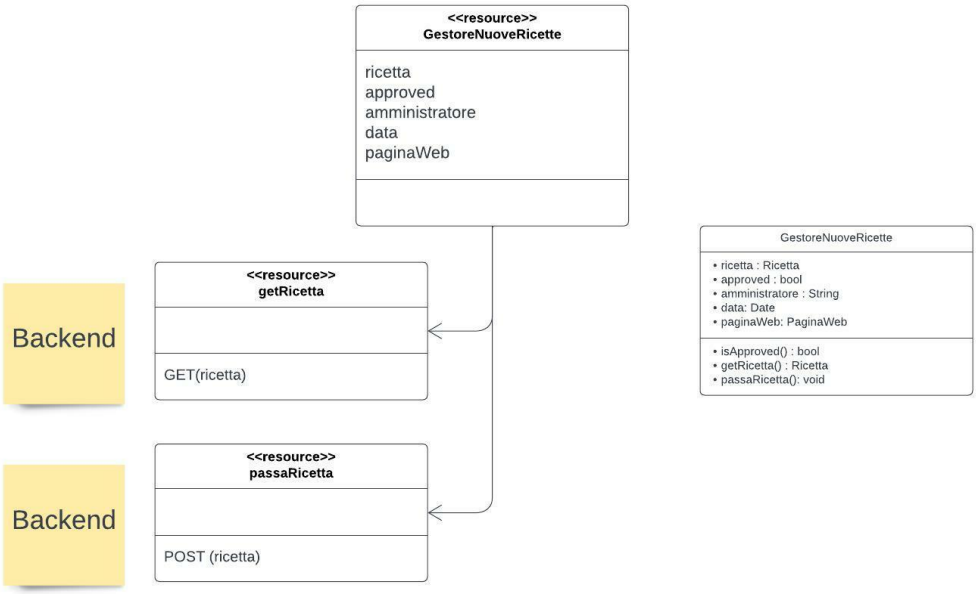


Figura: Classe e Resources Extraction Diagram fase creazione ricette

29. PAGINA HOME - START TUTORIAL

L'API **Start tutorial** volta al front-end della classe `PaginaHome` svolge il compito di inizializzare il tutorial e in seguito salvare tutti i dati a questo connessi. Per implementare questa API viene utilizzato il metodo HTTP `POST`.

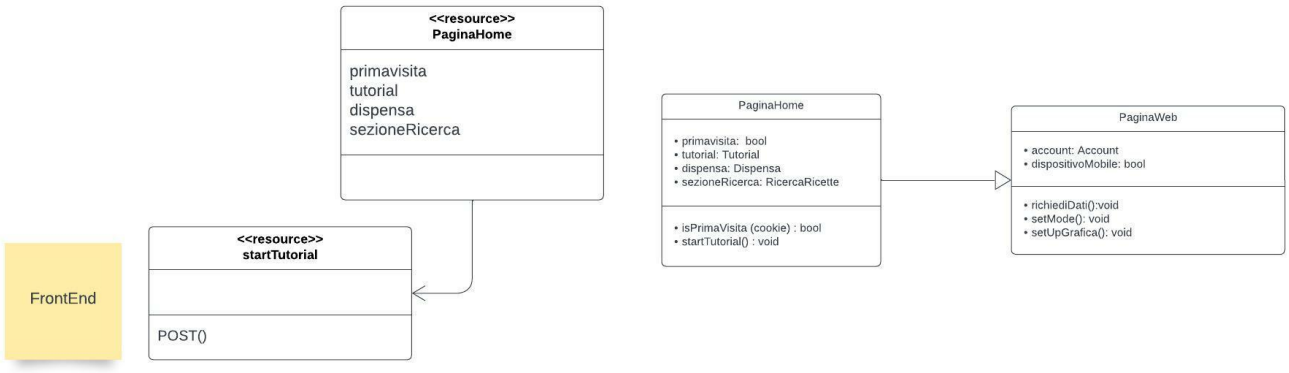


Figura: Classi e Resources Extraction Diagram fase tutorial

UNIVERSITÀ DI TRENTO

Diagramma 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 3). Nel caso non venissero trovati i dati ricercati viene ritornato il codice di risposta `404 NOT FOUND`.

2. TERMINI E CONDIZIONI

   L'API **Termini e condizioni** prende in input la variabile booleana `terminiEcondizioni`. Quest'ultima ha valore `true` se l'utente ha accettato i termini e le condizioni nella pagina dell'account, `false` altrimenti. L'API preleva questa informazione attraverso il metodo HTTP `GET`. Se viene eseguito tutto correttamente viene ritornato il codice di risposta `200 OK` e `terminiEcondizioni` viene fornita a chi si occupa di gestire la creazione dell'account (risorsa API 3). Nel caso la procedura sia errata viene ritornato `404 NOT FOUND`.

3. CREAZIONE ACCOUNT

   L'API **Registrazione** si occupa di creare l'account attraverso il metodo HTTP `POST`. Per farlo prende in input `primeCrenziali` (risorsa API 1), `terminiEcondizioni` (risorsa API 2), `validatore` che si occupa di validare i dati inseriti dall'utente e la variabile booleana `esitoProcedura`. Quest'ultima ha valore `true` se la procedura è corretta. Di conseguenza verrà ritornato il codice di risposta `201 CREATED`. Invece se ha valore `false` sono avvenuti degli errori e viene ritornato il messaggio `400 BAD REQUEST`.

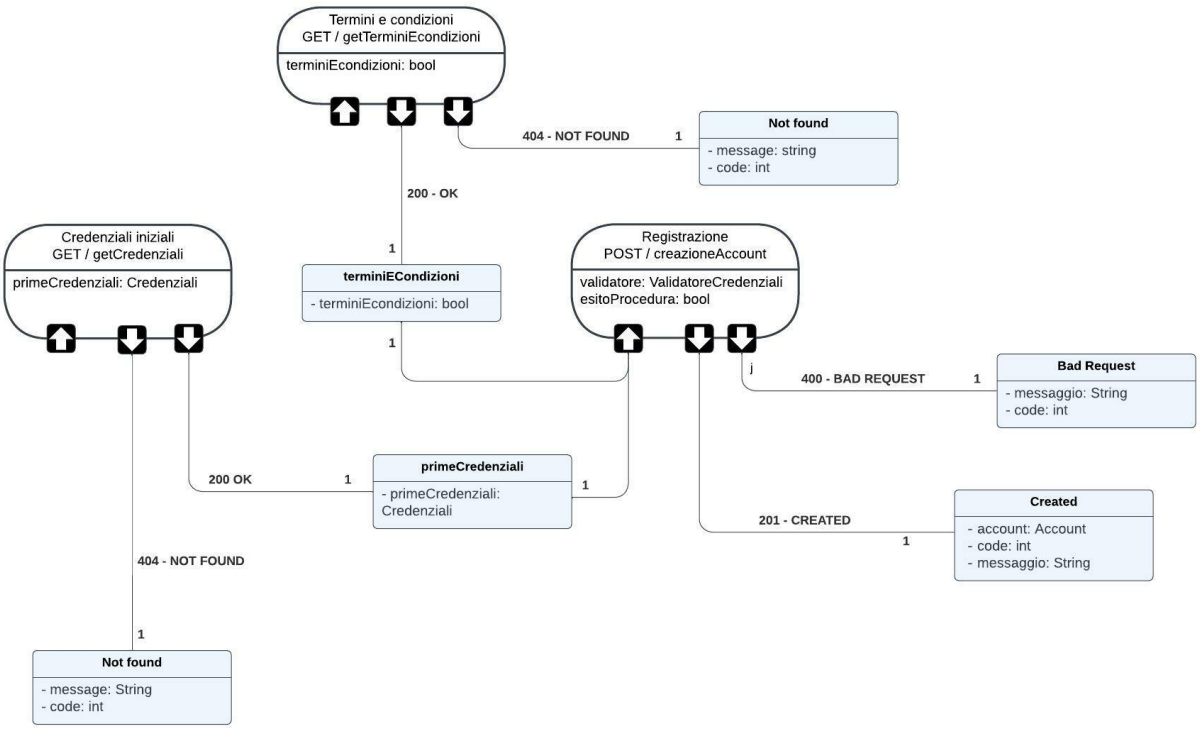


Figura: Resources Model Termini e condizioni, Registrazione, Credenziali iniziali

4. NUOVE CREDENZIALI

L'API **Nuove credenziali** prende in input `gestoreModifiche.nuoveCredenziali` che contiene 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 5). Invece se qualcosa è andato storto sarà ritornato il codice di risposta `404 NOT FOUND`.

5. 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 4), `validatore` per validare i nuovi dati, `account` che corrisponde all'account con le credenziali da modificare e `notificatoreCambioEmail` che verifica la correttezza dell'indirizzo email. Se l'operazione avviene correttamente viene ricevuto il messaggio `200 OK` e `account` con i dati modificati, altrimenti `400 BAD REQUEST`.

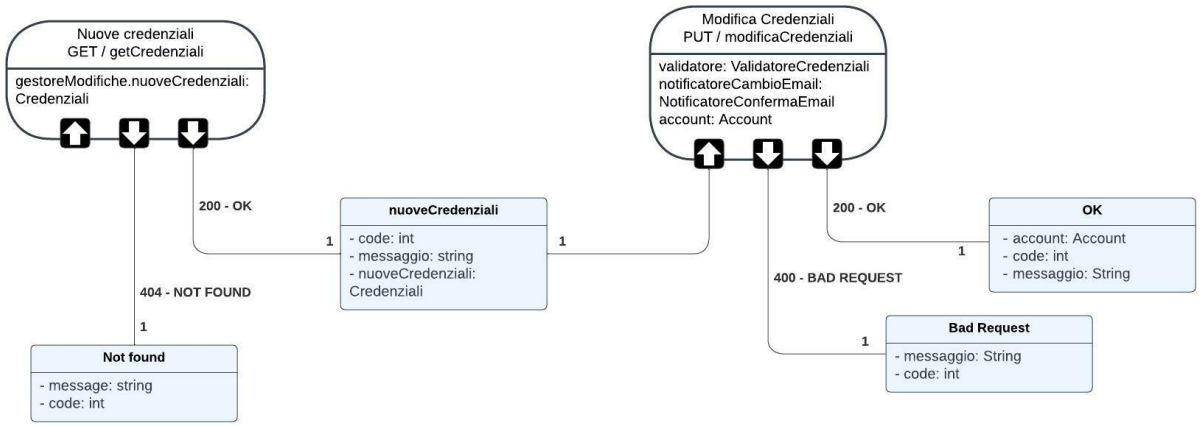


Figura: Resources Model Nuove credenziali, Modifica Credenziali

6. NUOVA IMG PROFILO

L'API **Nuova Img Profilo** prende in input `gestoreImg` necessaria per far selezionare all'utente l'immagine direttamente dalla propria galleria. Riceve anche `gestoreModifiche.nuovaImgProfilo` che contiene la nuova foto. Attraverso il metodo HTTP `GET` l'API preleva l'immagine scelta. Se la procedura risulta corretta viene visualizzato il messaggio `200 OK` e `nuovaImgProfilo` verrà passata a chi si occupa di gestire la modifica dell'immagine profilo (risorsa API 7). Invece se ci sono stati errori viene ritornato il codice di risposta `404 NOT FOUND`.

7. MODIFICA IMMAGINE PROFILO

L'API **Modifica Immagine profilo** attraverso il metodo HTTP `PATCH` modifica l'immagine profilo. Prende in input `nuovaImgProfilo` (risorsa API 6) e `account` che corrisponde al profilo 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`.

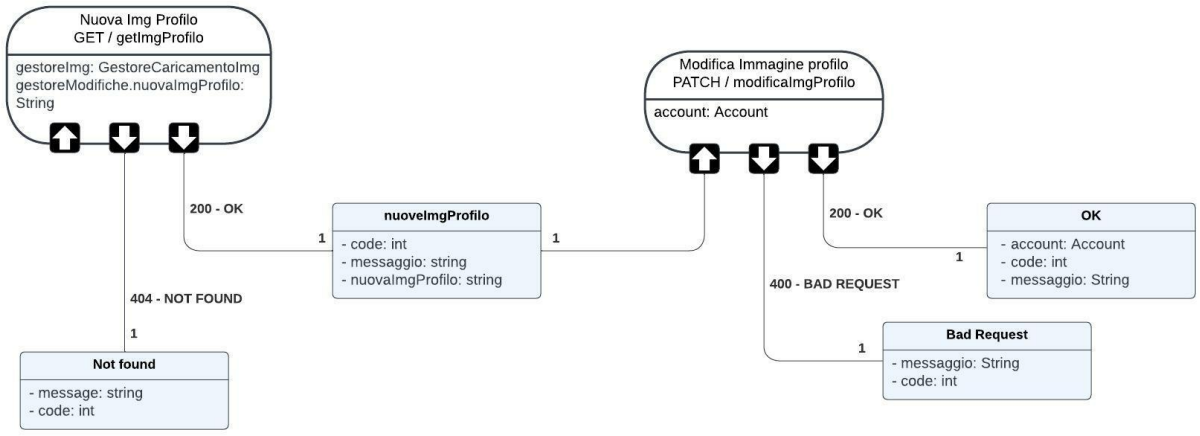


Figura: Resources Model Nuova Img Profilo, Modifica Immagine profilo

8. SALVATAGGIO FILTRI

L'API **Salvataggio filtri** prende in input la variabile booleana `salvataggioFiltri`. Quest'ultima ha valore `true` se l'utente attiva il salvataggio dei filtri di ricerca, `false` altrimenti. La modifica del valore della variabile appena analizzata avviene attraverso il metodo HTTP `PATCH`. Se l'operazione eseguita è corretta viene ritornato il messaggio `200 OK` e `salvataggioFiltri` viene mandato a chi si occupa della selezione dei filtri (risorsa API 9). Invece, se il processo è errato, viene inviato il codice di risposta `400 BAD REQUEST`.

9. SELEZIONA FILTRI

L'API **Seleziona filtri** utilizza il metodo HTTP `PUT` per cambiare la selezione dei filtri da tenere nella ricerca delle ricette. Prende in input `salvataggioFiltri` 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 `listaFiltri` contenente i filtri e le loro variabili booleane `selezionato` modificate (vedi figura x) . Queste hanno il valore a `true` se l'utente desidera averli nella ricerca delle ricette, `false` altrimenti. Invece, se l'operazione non va a buon fine, viene ricevuto il codice di risposta `400 BAD REQUEST`.

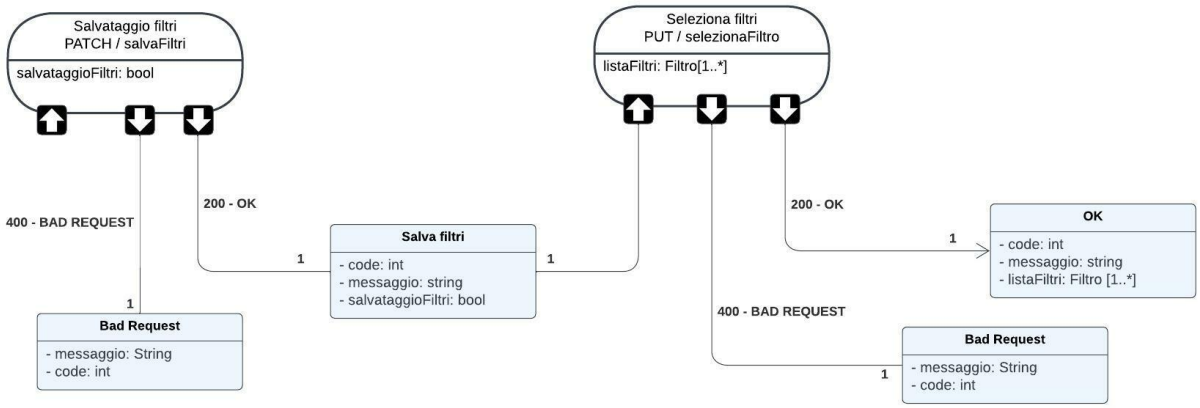


Figura: Resources Model Salvataggio filtri, Seleziona filtri

10. CREDENZIALI INSERITE

L'API **Credenziali inserite** prende in input `componenteLogin.credenzialiInserite` che contiene i dati (username, password) inseriti dall'utente nella pagina del login per effettuare l'accesso. Li preleva attraverso il metodo HTTP `GET`. Se l'operazione è eseguita correttamente, viene ritornato il messaggio `200 OK` e `credenzialiInserite` per chi si occupa di gestire il login (risorsa API 11). Se ci sono errori nella procedura viene ritornato il codice di risposta `404 NOT FOUND`.

11. LOGIN

L'API **Login** fornisce l'accesso all'utente nella fase di login attraverso il metodo HTTP `POST`. Riceve in input `credenzialiInserite` (risorsa API 10). Se è tutto corretto viene ritornato il messaggio `200 OK`, altrimenti il codice di risposta è `400 BAD REQUEST`. Inoltre, se l'operazione è giusta, nella risposta viene fornito anche l'account associato ai dati inseriti (`account`).

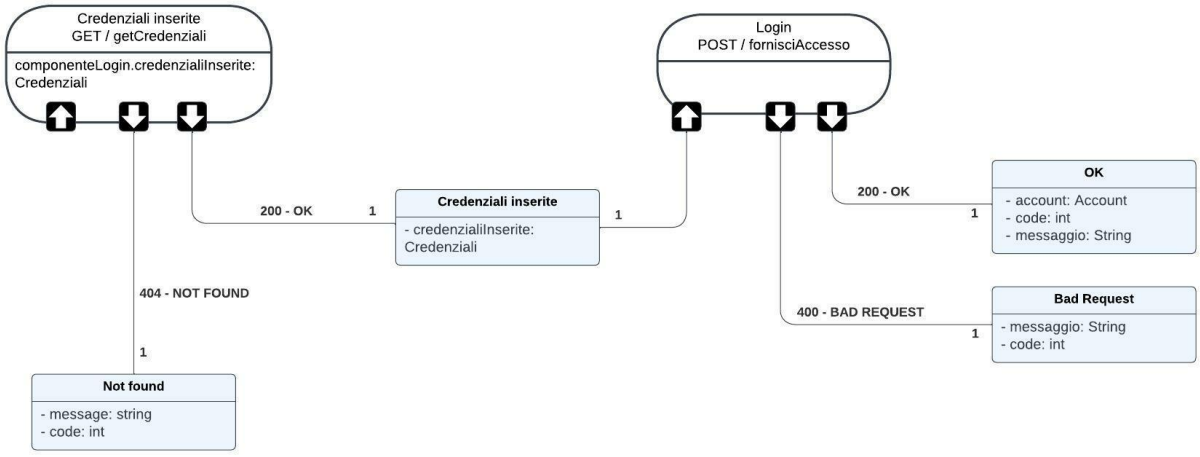


Figura: Resources Model Credenziali inserite, Login

12. NUOVA PASSWORD

L'API **Nuova password** prende in input `componenteLogin.passwordNuova`. Quest'ultima contiene la nuova password che l'utente inserisce nella pagina del login. Viene prelevata attraverso il metodo HTTP `GET`. Se è tutto corretto viene ritornato il messaggio `200 OK` e `passwordNuova` viene inviata a chi si occupa di modificare la password (risorsa API 14). Invece, se la procedura presenta errori, viene ricevuto il messaggio `404 NOT FOUND`.

13. EMAIL CAMBIO PASSWORD

L'API **Email cambio password** prende in input `compoennteLogin.indirizzoEmail` contente 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 14). Invece, se non viene trovato l'indirizzo email, il codice di risposta è `404 NOT FOUND`.

14. CAMBIO PASSWORD LOGIN

L'API **Cambio password** login modifica la password dell'utente attraverso il metodo HTTP `PATCH`. Prende in input `passwordNuova` (risorsa API 12), `indirizzoEmail` (risorsa API 13), `validatore` (per validare la nuova password) e `gestoreModifiche` (per modificare la password). Se l'operazione è corretta viene ritornato `200 OK`, altrimenti `400 BAD REQUEST`.

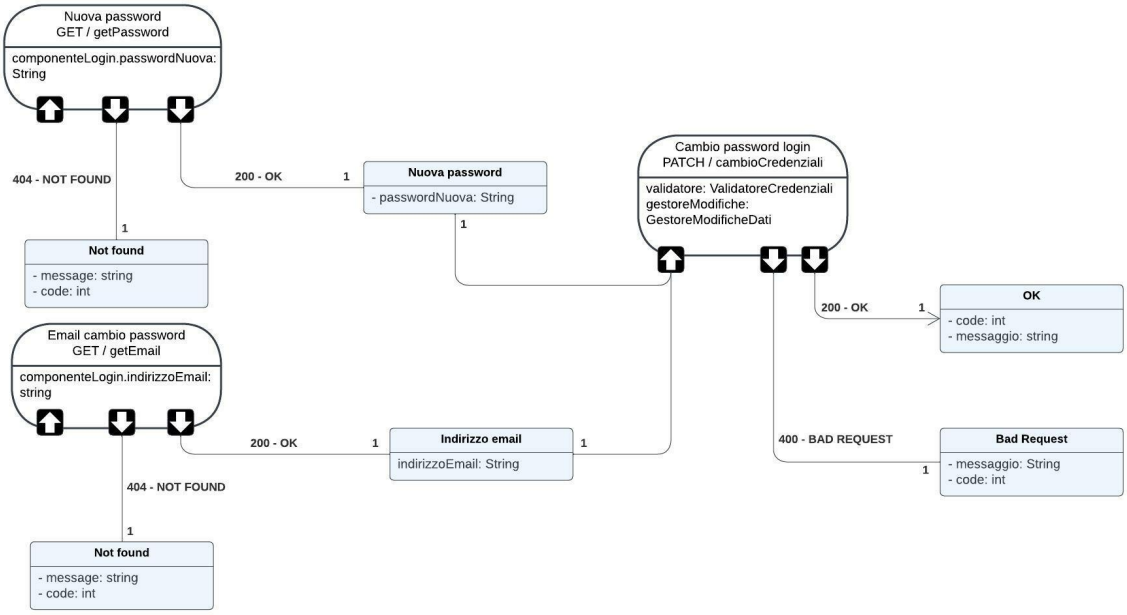


Figura: Resources Model Nuova password, Cambio password login, Email cambio password

15. COMPLETA RICETTA

L'API **Completa ricetta** prende in input il nome e l'autore della ricetta che è stata appena "completata" e la variabile booleana `primoCompletamento` che sarà `true` se è la prima volta che l'utente completa il piatto e `false` altrimenti. Nel caso in cui `primoCompletamento` abbia valore `true` attraverso il metodo `PATCH` viene aggiornato a `false` il relativo attributo della ricetta nel database. 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`.

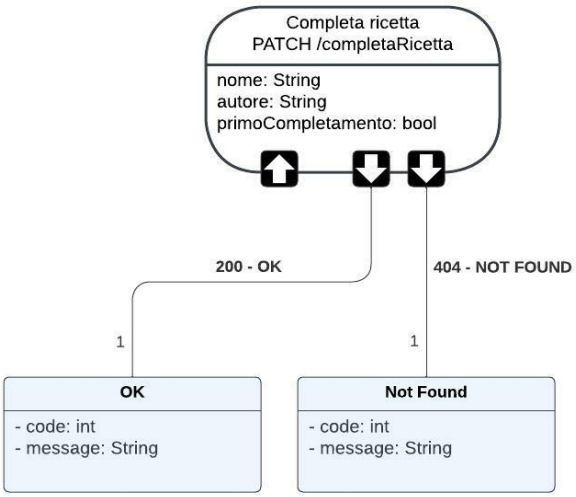


Figura: Resources Model Completa ricetta

16. 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 e la variabile booleana `salvata` che sarà `true` se il piatto è già presente tra i preferiti e `false` altrimenti. Nel caso in cui `salvata` abbia valore `true` non viene eseguita alcuna operazione. Se `salvata` ha valore `false` attraverso il metodo `PATCH` viene aggiornato a `true` il relativo attributo della 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 cercata.

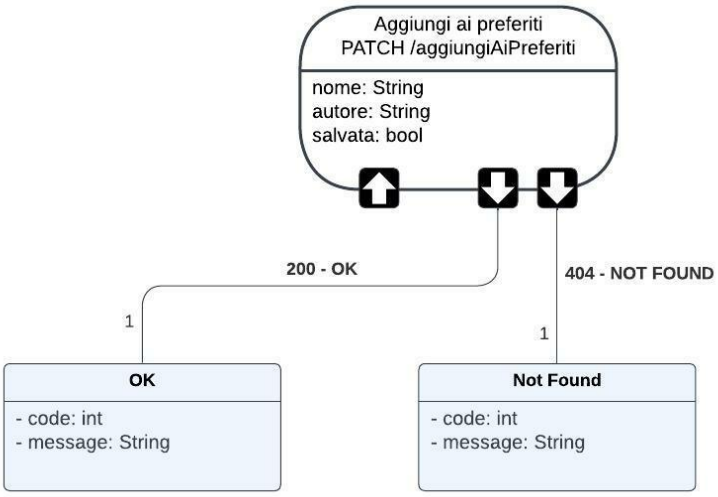


Figura: Resources Model Aggiungi ai preferiti

17. TOGLI DAI PREFERITI

L'API **Togli dai preferiti** prende in input il nome e l'autore della ricetta da togliere dai preferiti dell'utente e la variabile booleana `salvata`. Nel caso in cui salvata abbia valore `false` non viene eseguita alcuna operazione. Se `salvata` ha valore `true` attraverso il metodo `PATCH` viene aggiornato a `false` il relativo attributo della ricetta nel database. 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.

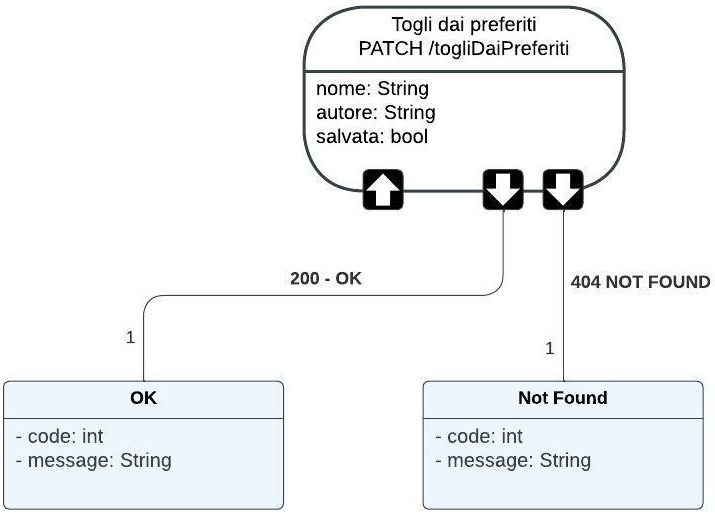


Figura: Resources Model Togli dai preferiti

18. CARICA DATI

Carica dati corrisponde al metodo `richiedeDati` della classe `GestoreRichiestePagine`. L'API prende in input la pagina web di cui richiedere le risorse. Attraverso il metodo `GET` restituisce tutti gli elementi necessari al database. Viene ritornato un messaggio di `200 OK` alla buona riuscita dell'operazione e l'errore `404 NOT FOUND` se uno o più risorse non vengono trovate.

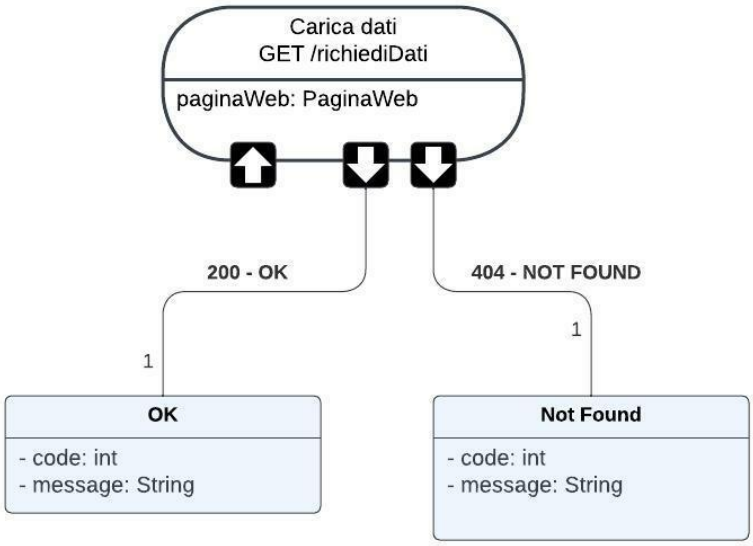


Figura: Resources Model Carica dati

19. 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 passata la dispensa di un utente. 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`.

20. 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 richiesta una dispensa a cui aggiungere l'ingrediente. 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.

21. ELIMINA INGREDIENTE

Come per il metodo `modificaQuantità` e `aggiungiIngrediente`, l'API **Elimina ingrediente** utilizza il metodo HTTP `PATCH` per modificare i dati della dispensa. A questo metodo viene passata in input la dispensa di un utente e l'ingrediente che si desidera eliminare. 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`.

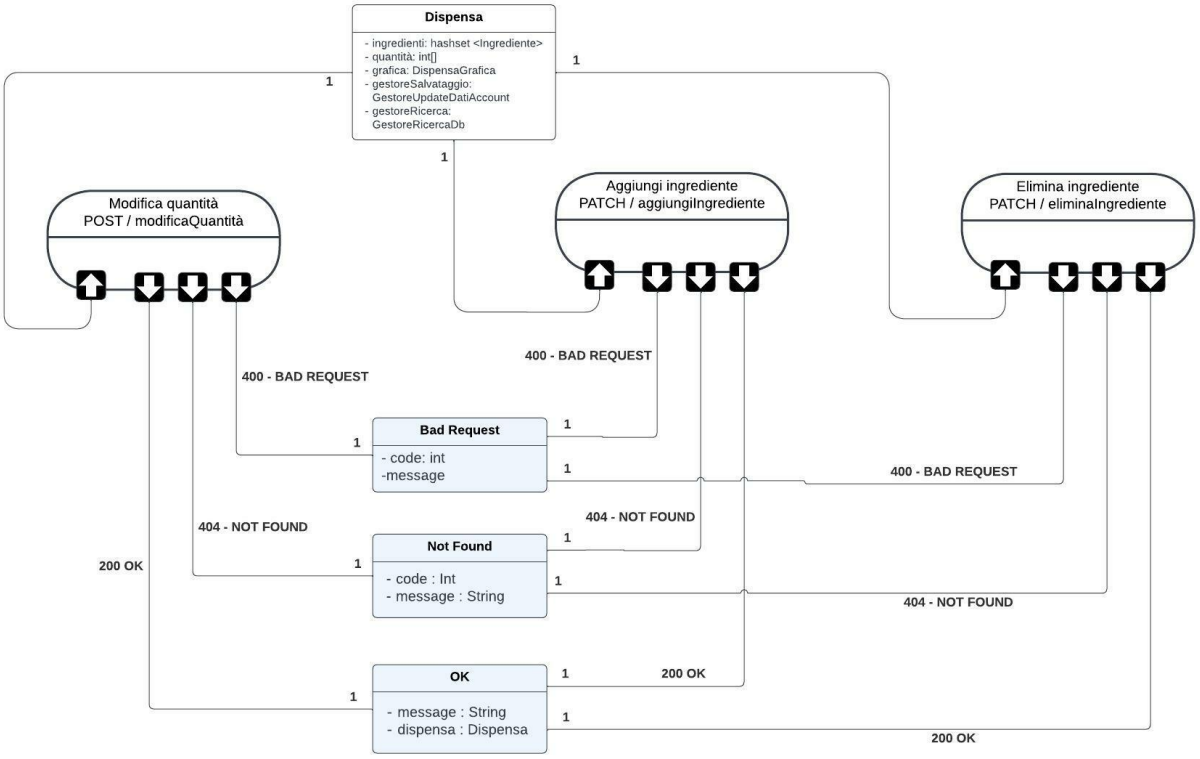


Figura: Resources Model Modifica quantità, Aggiungi ingrediente, Elimina ingrediente

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