



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

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 `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.

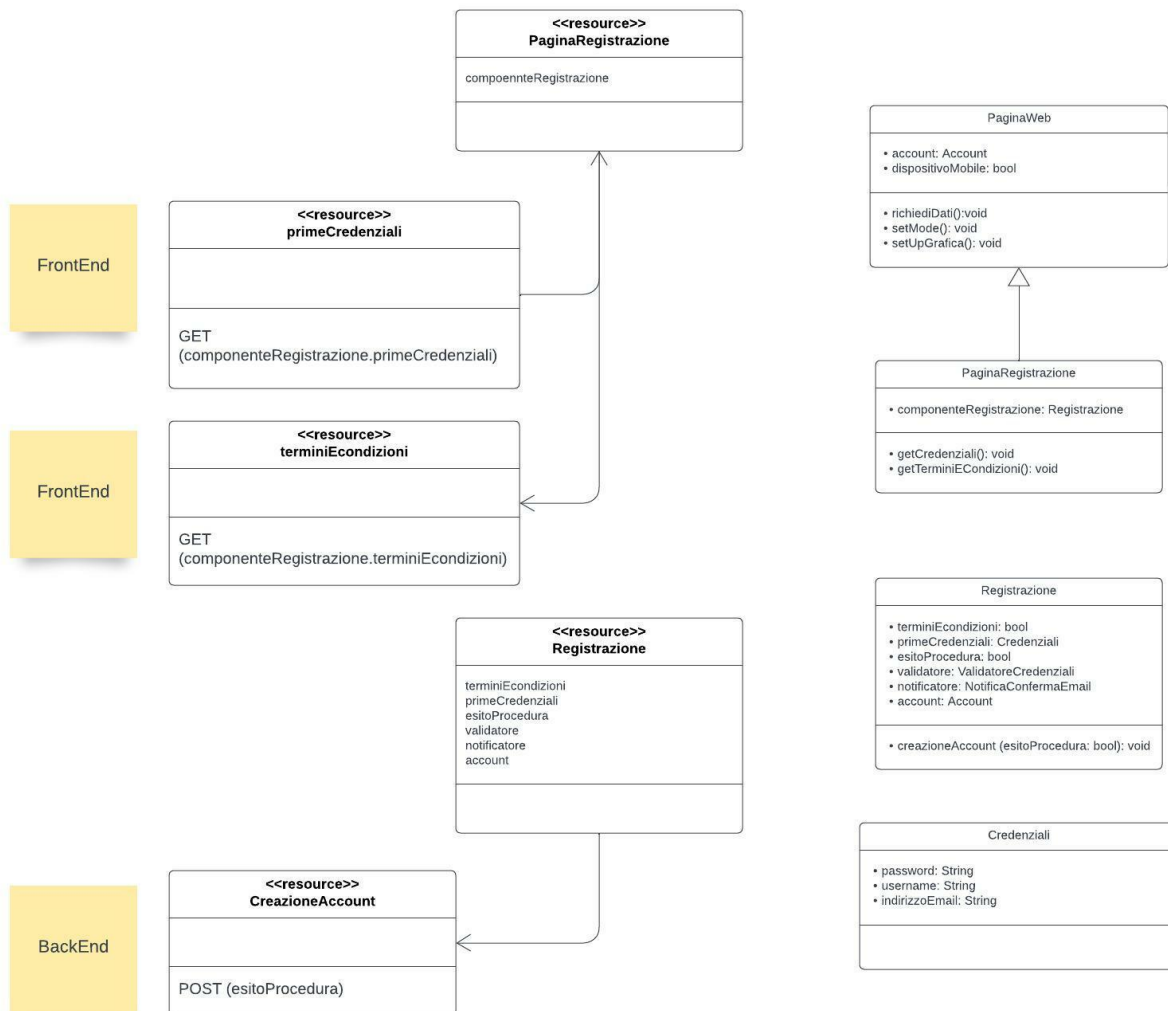


Figura: Classi e Resources Extraction Diagram fase registrazione

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 `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`.

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`.

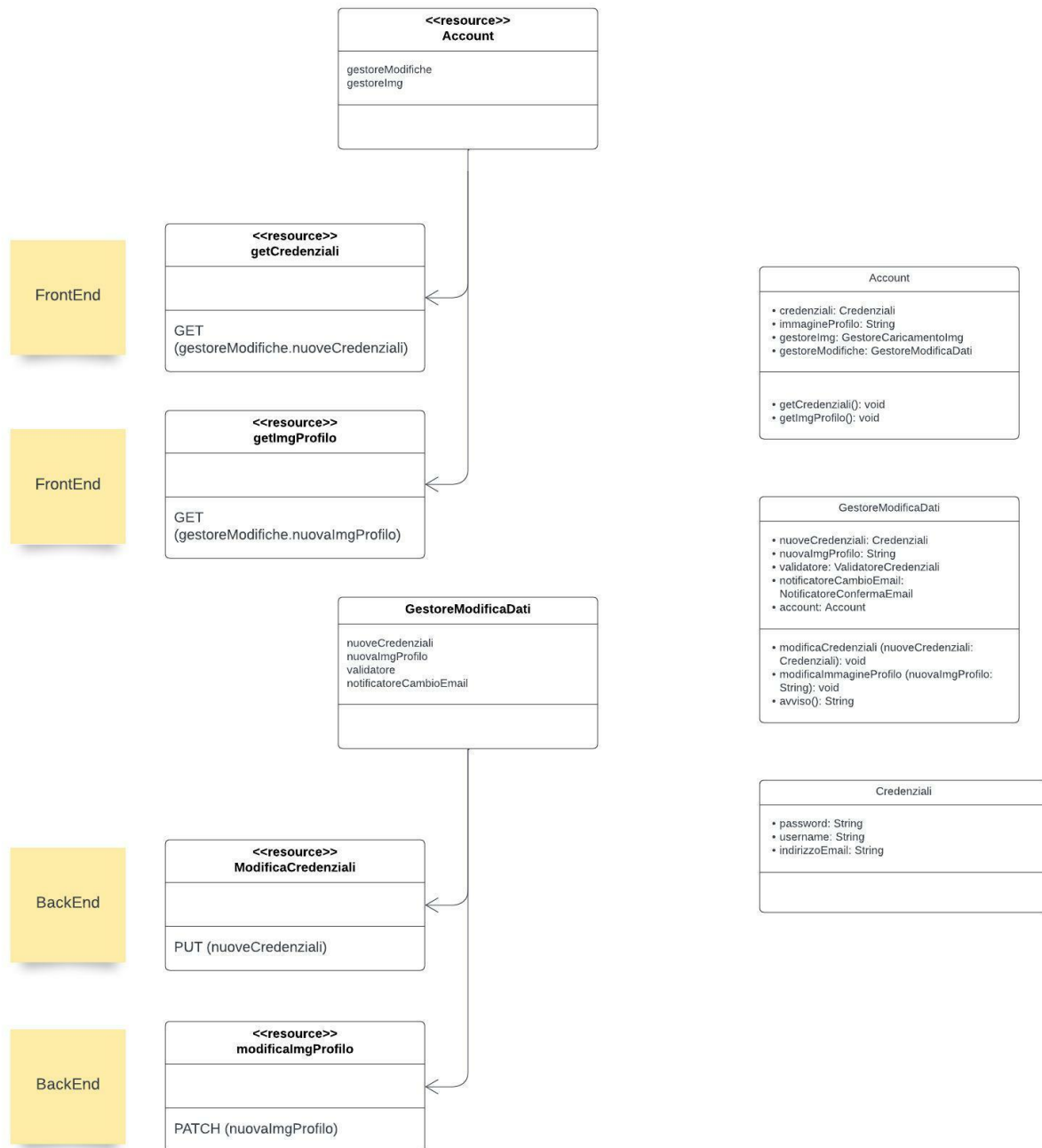


Figura: Classi e Resources Extraction Diagram fase modifica dati pagina account

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`.

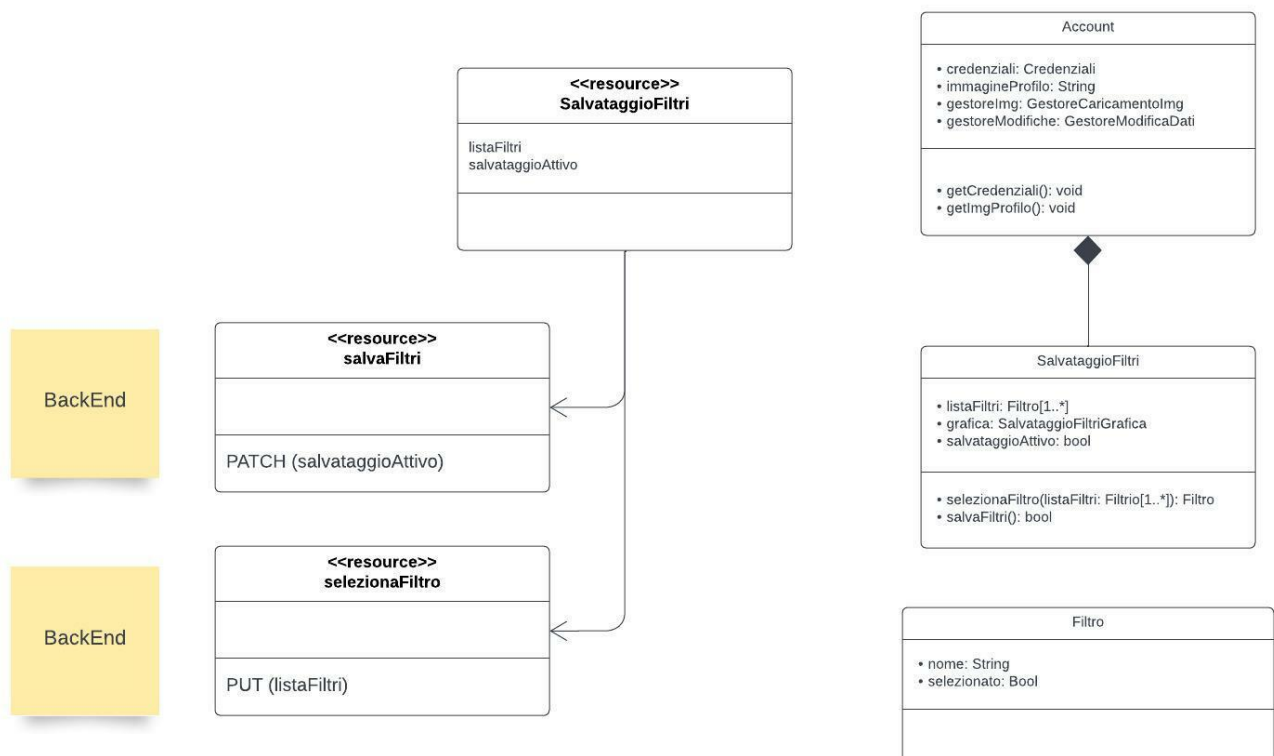


Figura: Classe e Resources Extraction Diagram fase salvataggio filtri

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`.

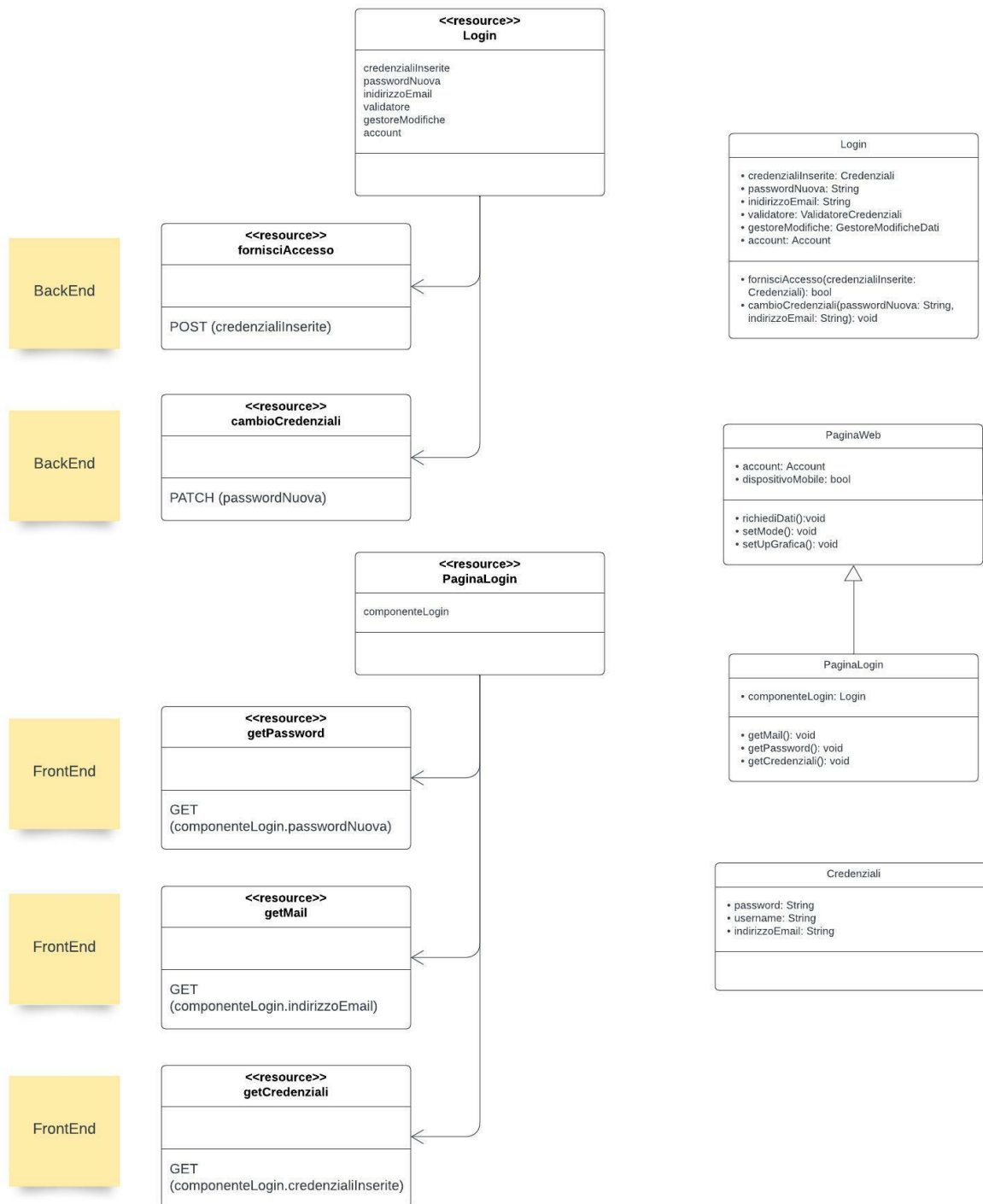


Figura: Classi e Resources Extraction Diagram fase login

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.

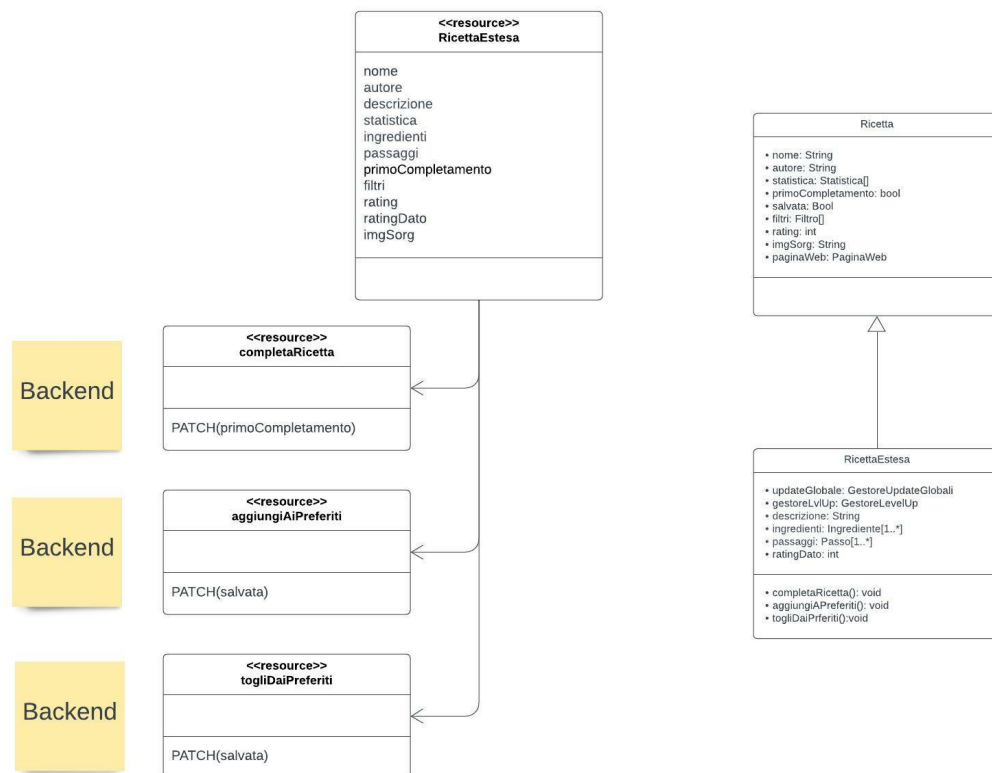


Figura: Classi e Resources Extraction Diagram fase interazione ricetta

18. GESTORE UPDATE GLOBALI - AGGIORNA RATING

L'API **Aggiungi rating** corrisponde al metodo `aggiornaRating` della classe `GestoreUpdateGlobali`. L'utente può dare una valutazione ad una determinata ricetta. L'API considerata attraverso il metodo HTTP PATCH aggiunge il nuovo rating alla ricetta nel database.

19. GESTORE UPDATE GLOBALI - AGGIORNA COMMENTI

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

20. GESTORE UPDATE GLOBALI - AGGIORNA COMMENTI

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

21. GESTORE RICHIESTE PAGINE - RICHIESTE 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.

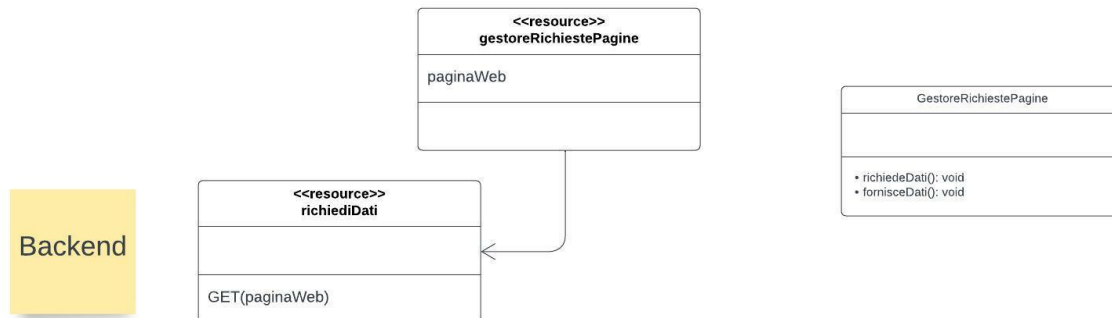


Figura: Classe e Resources Extraction Diagram fase richiesta dati

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 `primeCredenziali` (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`.

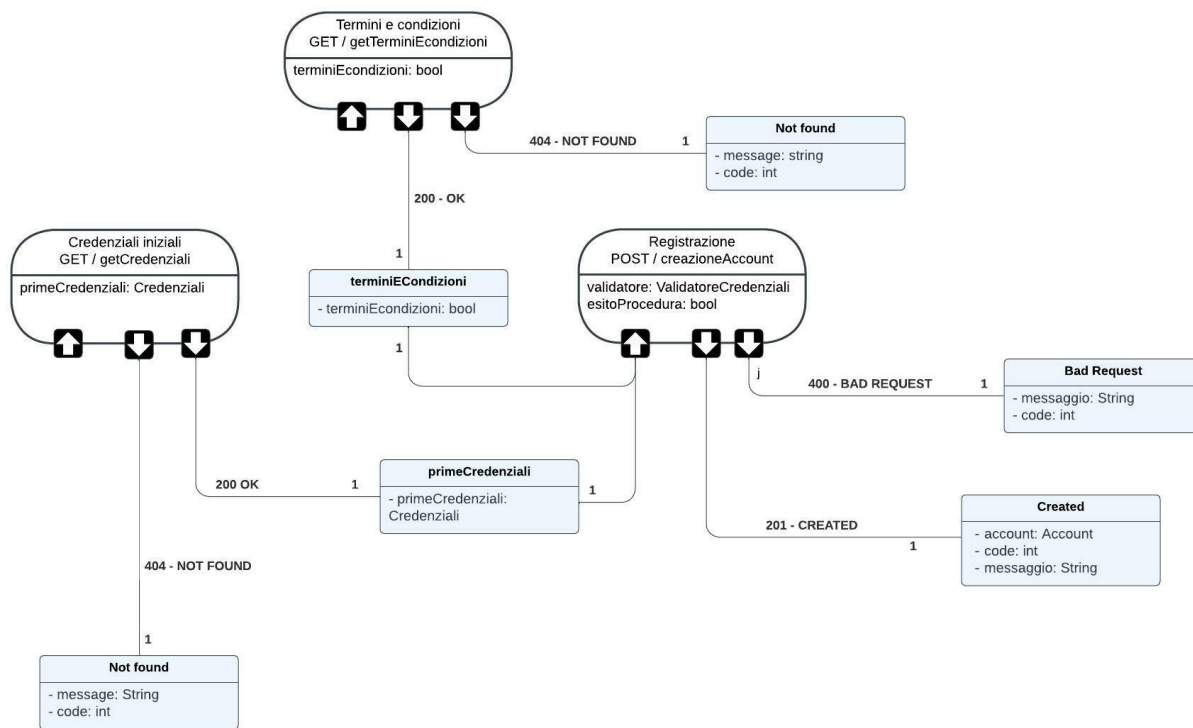


Figura: Resources Model Termini e condizioni, Registrazione, Credenziali iniziali

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`.

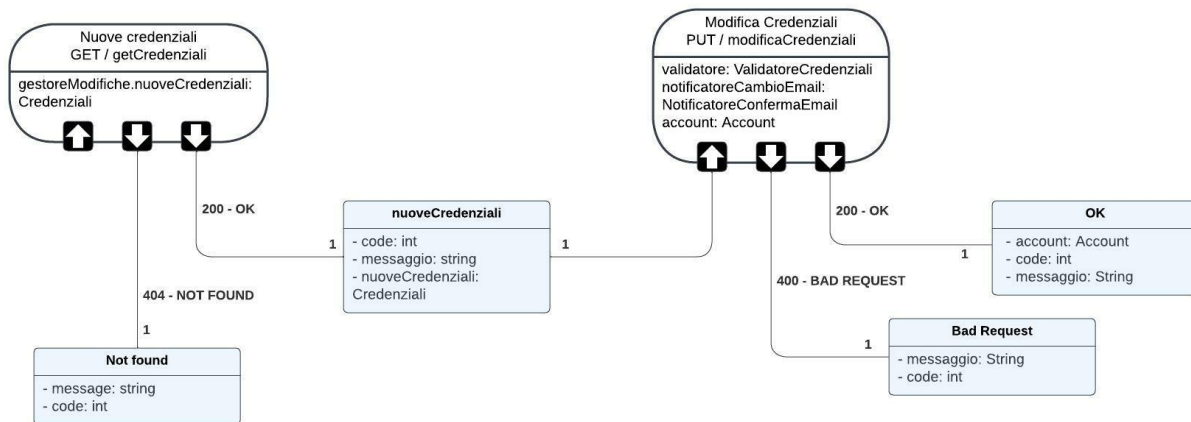


Figura: Resources Model Nuove credenziali, Modifica Credenziali

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.

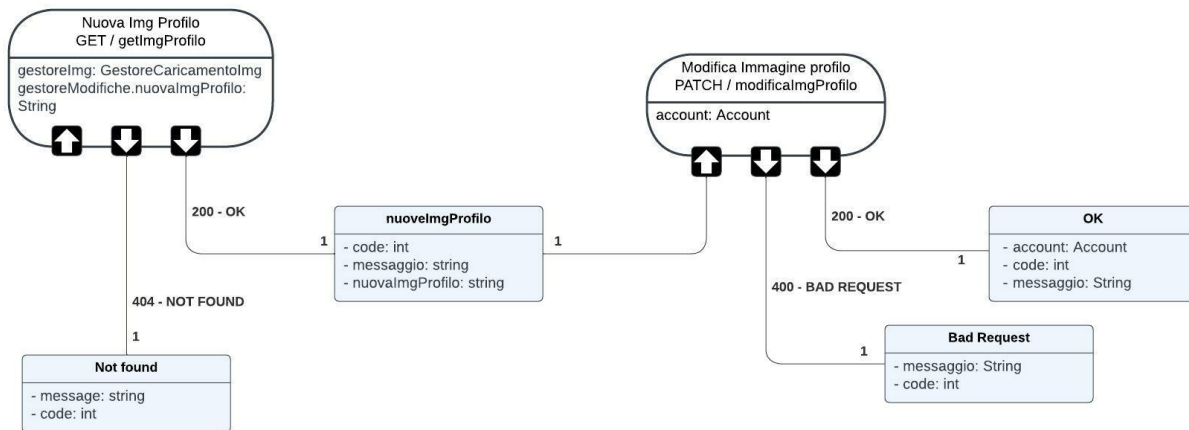


Figura: Resources Model Nuova Img Profilo, Modifica Immagine profilo

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`.

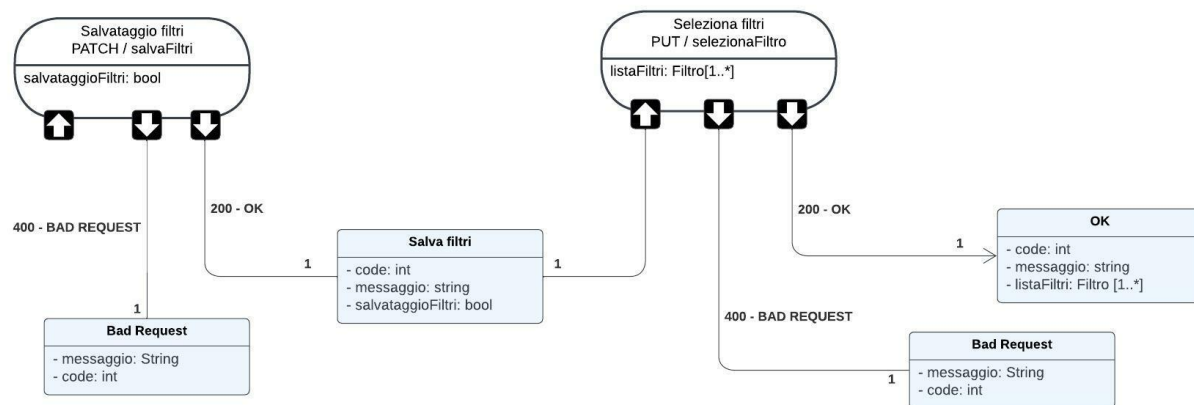


Figura: Resources Model Salvataggio filtri, Selezione filtri

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`).

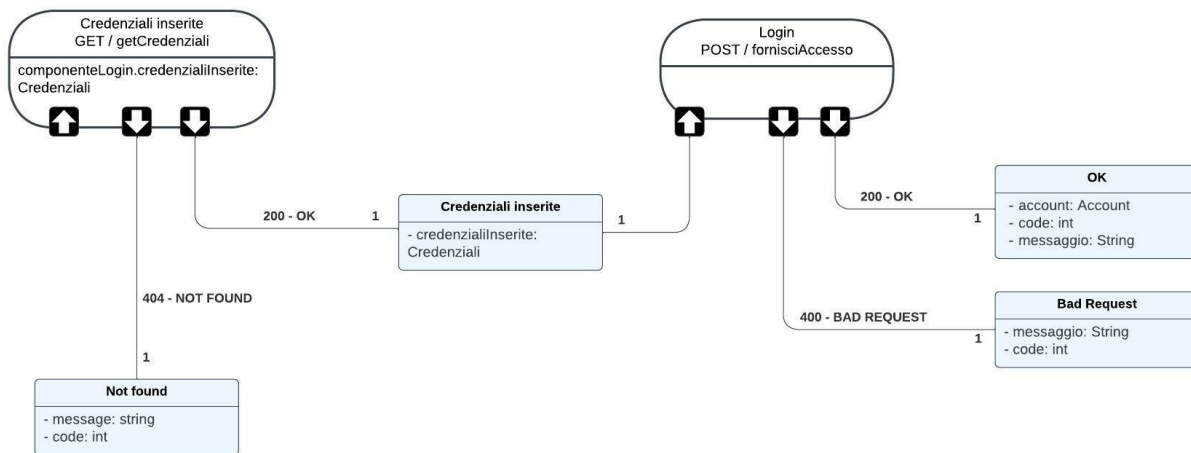


Figura: Resources Model Credenziali inserite, Login

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 `compoenntLogin.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 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.

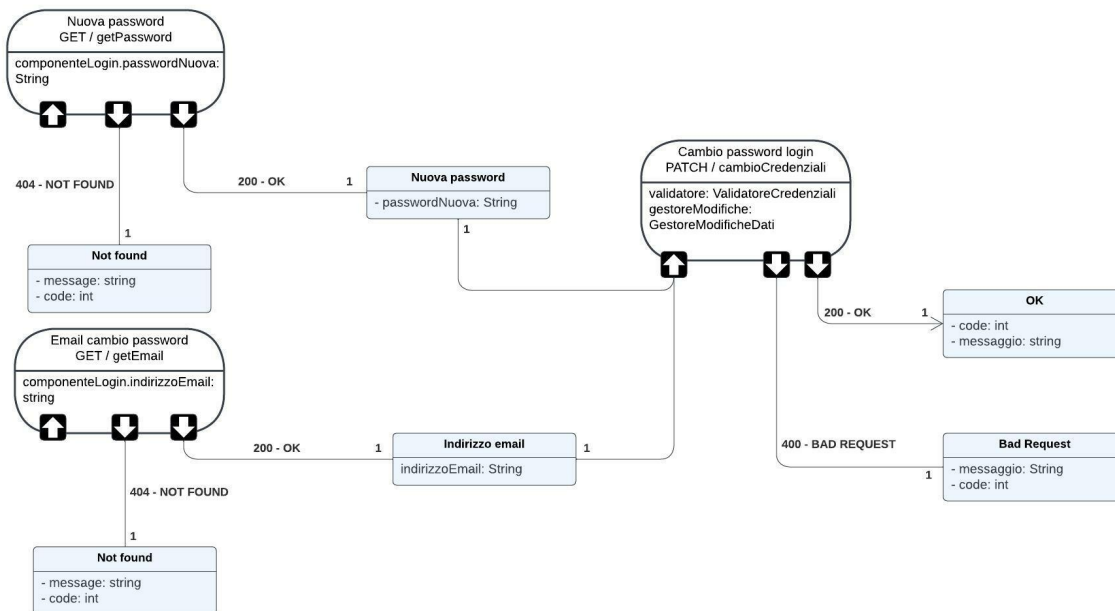


Figura: Resources Model Nuova password, Cambio password login, Email cambio password

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`.

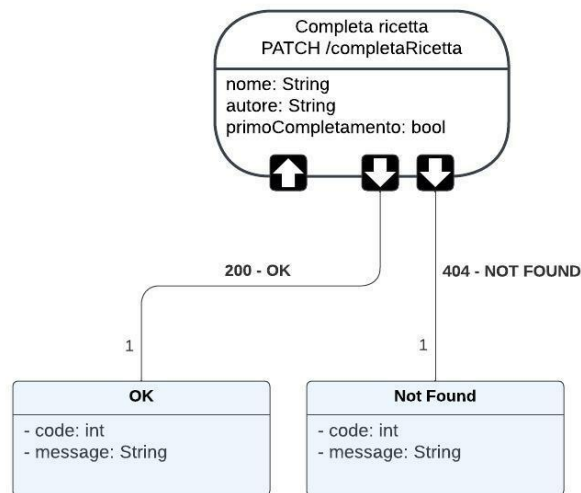


Figura: Resources Model Completa ricetta

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.

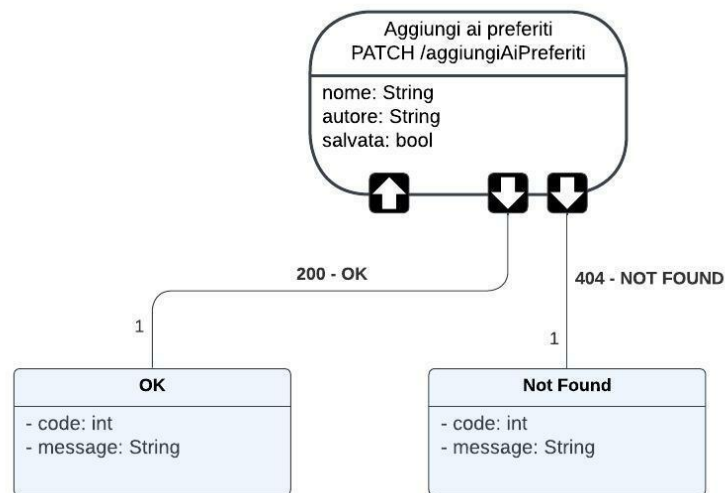


Figura: Resources Model Aggiungi ai preferiti

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.

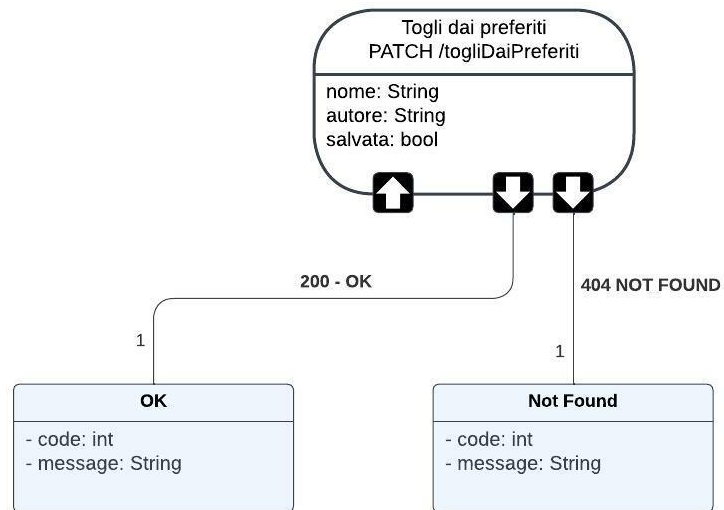


Figura: Resources Model Togli dai preferiti



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