Scopo del documento

Il seguente documento riporta tutte le informazioni riguardanti lo sviluppo di una parte dell'applicazione web PricePal.

Nel **primo capitolo** viene riportato lo user flow, ovvero una descrizione tramite diagramma di tutte le azioni che si possono eseguire sulla parte implementata di PricePal, descrivendo le varie richieste effettuabili a front-end in ogni pagina e le varie risposte possibili.

Nel **secondo capitolo** viene spiegato come è stato implementato il front-end.

Viene poi data nel **terzo capitolo** una rappresentazione delle risorse estratte dal diagramma delle classi, e successivamente sviluppate, tramite "diagramma di estrazione delle risorse", seguito dalla visualizzazione dettagliata delle api implementate attraverso "diagramma delle risorse" nel **quarto capitolo**.

Nel **quinto capitolo** viene presentata la documentazione delle API prodotta tramite il software "Swagger".

Nel **sesto capitolo** viene mostrato come è stato implementato il back-end.

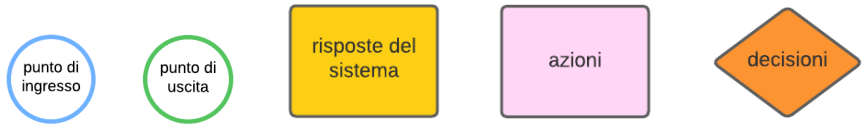
Nel **settimo capitolo** vengono mostrati i risultati della fase di testing.

Nell'**ottavo capitolo** viene spiegato come è stato effettuato il deployment.

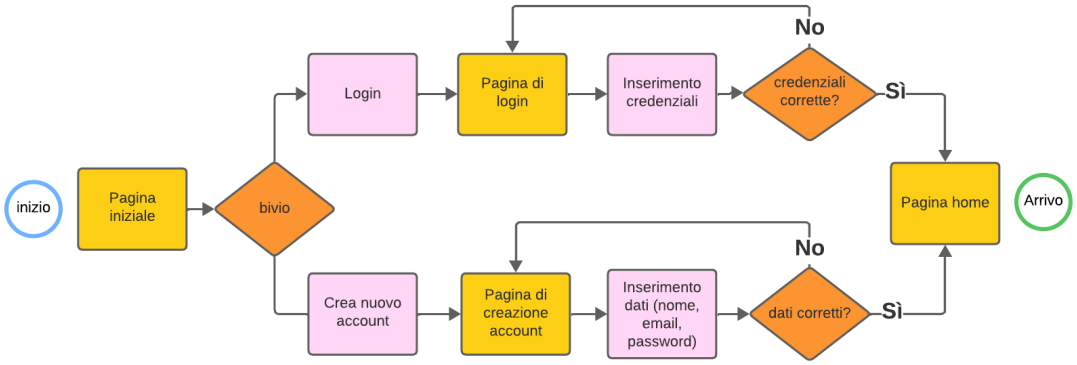
1. User-flow

In questa sezione viene riportato lo user-flow, il quale rappresenta le azioni che l'utente più compiere nell'implementazione descritta in questo documento.

Didascalia:

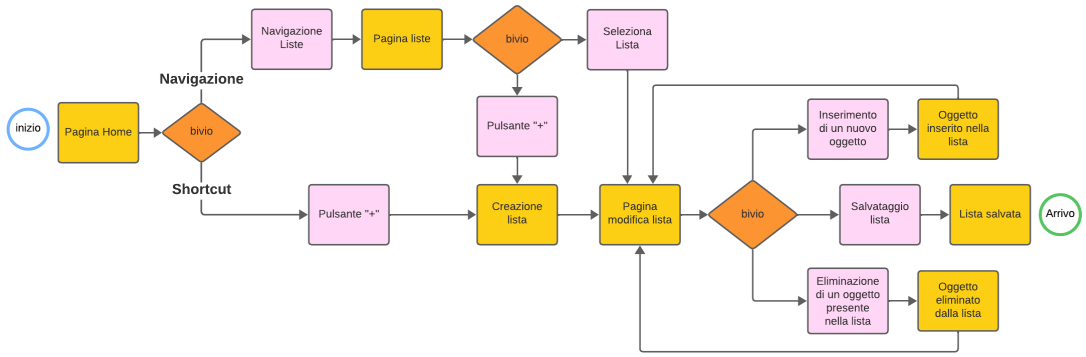


User-flow relativo all’accesso/registrazione a PricePal:



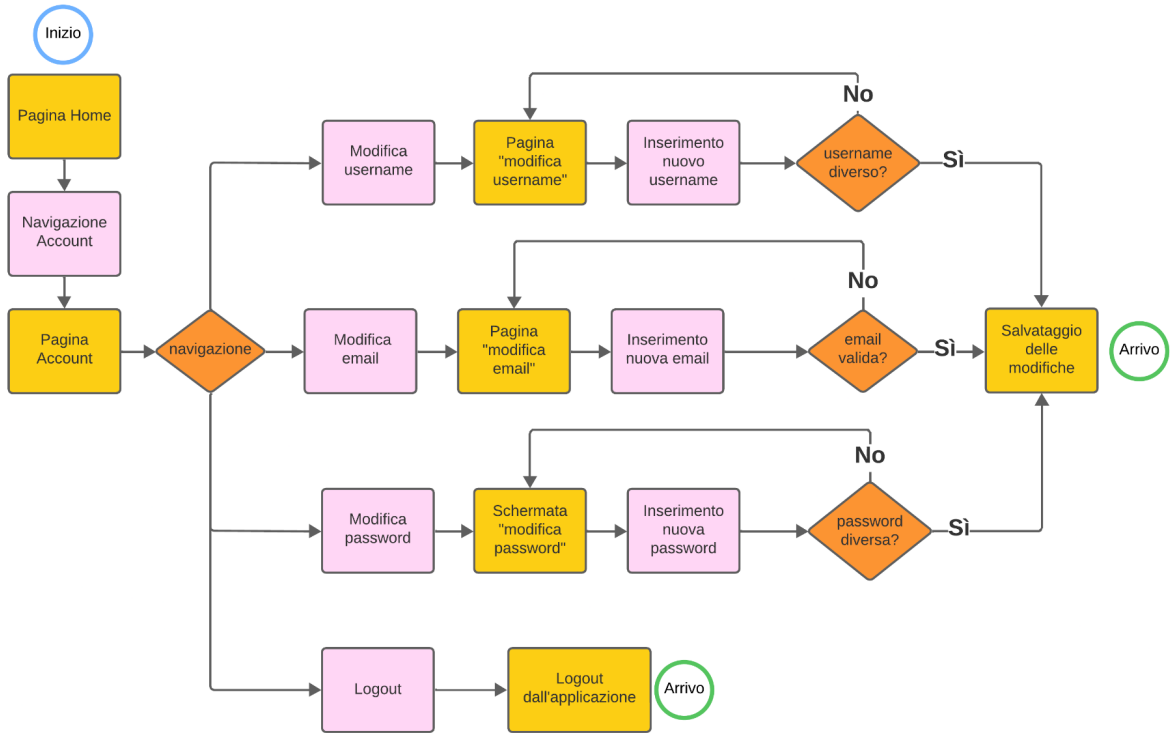
L’utente può effettuare l'accesso e la registrazione di un nuovo account come da RF1.1 e RF1.2.

User-flow relativo alla creazione e alla modifica di una lista:



L’ utente può accedere alla schermata contenente tutte le liste per selezionarne una da modificare, come da RF3, creare liste, come da RF2.1, e aggiungere o rimuovere oggetti da una lista, come da RF2.2.

User-flow relativo alla modifica delle credenziali del proprio account:



L'utente può modificare nome, email e password, come da RF1.4, ed effettuare il logout dall'applicazione, come da RF1.3.

2. Implementazione del front-end

In questa sezione vengono descritte in dettaglio tutte le schermate del front-end che abbiamo implementato e, per ognuna di esse, le rispettive azioni che l'utente può eseguire.

Schermata iniziale

La schermata iniziale è la schermata che verrà caricata in automatico quando si accede al sito. È presente il logo di Pricepal e due pulsanti: "accedi" e "registrati". Dato che l'applicazione può essere utilizzata solo da utenti registrati, l'utente deve obbligatoriamente creare un nuovo account o accedere con il proprio per continuare.

Accesso

Nella schermata di accesso è presente un form dove l'utente può inserire email e password relativi al proprio account personale. Se le credenziali sono corrette, l'applicazione reindirizza l'utente alla schermata home.

Registrazione

Nella schermata di registrazione è presente un form dove l'utente può creare un account inserendo nome, email e password. Se le credenziali sono corrette, l'applicazione crea un nuovo account e reindirizza l'utente alla schermata home.

Home

Una volta che l'utente ha fatto l'accesso (oppure ha registrato un nuovo account) verrà caricata la pagina Home. In cima alla pagina Home è presente un header contenente il logo e l'icona dell'utente da cui si può accedere alla schermata di gestione account. Sotto l'header è presente una navbar composta da due bottoni: "Home" e "Liste". I bottoni sono automaticamente selezionati quando ci troviamo all'interne delle rispettive pagine. L'header e la navbar sono fissi e sono visibili in ogni schermata post-login. La schermata Home vera e propria è composta dalle 3 liste più recenti (se presenti) e da un pulsante "+" che reindirizza l'utente alla pagina di gestione lista.

Liste

Nella schermata liste sono presenti tutte le liste associate all'utente. Cliccando sul pulsante "+" si viene reindirizzati alla schermata "gestione lista" con una nuova lista vuota. Se invece si clicca su una lista si viene reindirizzati alla schermata "gestione lista" e viene caricata la lista selezionata per essere modificata dall'utente.

Gestione lista

Nella schermata gestione lista viene visualizzata una lista precedentemente selezionata o creata dall'utente. Sono presenti appositi input testuali e bottoni per modificare il titolo e gli oggetti contenuti nella lista. Quando si inizia a digitare il nome di un oggetto comparirà un menù di selezione che suggerisce all'utente l'inserimento di oggetti che combaciano con la stringa che l'utente sta scrivendo. Questi suggerimenti sono ricavati dalla lista di tutti gli oggetti precedentemente inseriti dall'utente. Per salvare una lista alla quale sono state applicate delle modifiche è necessario cliccare sul pulsante "salva". Sono inoltre presenti dei pulsanti per eliminare la lista e per eliminare la lista di oggetti precedentemente inseriti dall'utente.

Gestione account

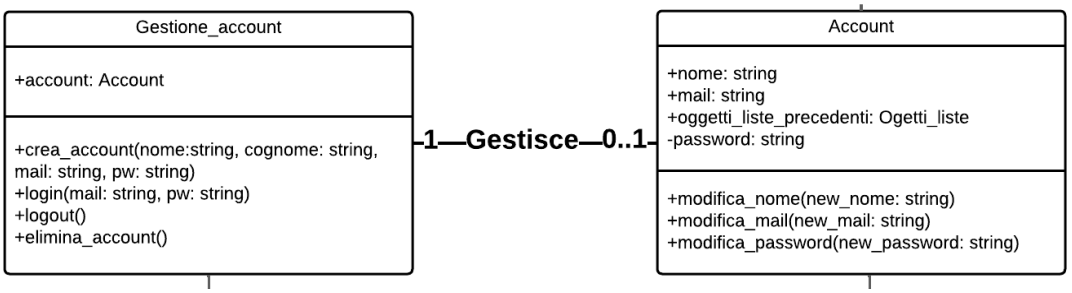
Nella pagina Gestione account sono presenti 5 bottoni, rispettivamente: "Cambia nome", "Cambia email", "Cambia password", "Elimina account", "Logout". Cliccando uno dei primi quattro bottoni si aprirà un form sulla parte destra della pagina in cui sarà possibile eseguire l'azione selezionata dopo aver inserito la password attuale. Cliccando su "Logout" l'utente verrà reindirizzato alla schermata iniziale e dovrà eseguire di nuovo l'accesso se desidera utilizzare l'applicazione.

3. Estrazione delle risorse dal class diagram

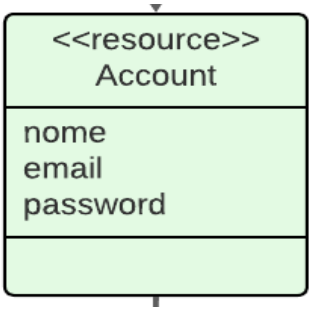
Il seguente capitolo mostra come le risorse implementate nel back-end siano state estratte dal class diagram.

Per prima cosa sono state individuate le classi che sarebbero diventate gli schemi necessari a rappresentare i dati contenuti nel back-end. In particolare sono state individuate le seguenti classi:

- Account e Gestione_account



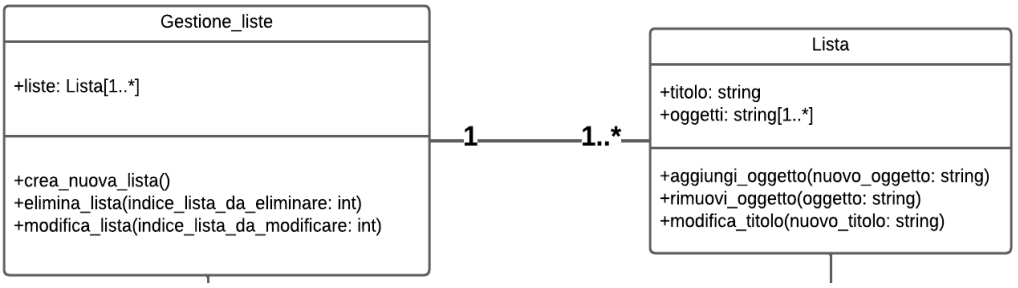
Da queste due classi abbiamo deciso di ricavare la risorsa **Account**, contenente gli attributi “nome”, “mail” e “password”.



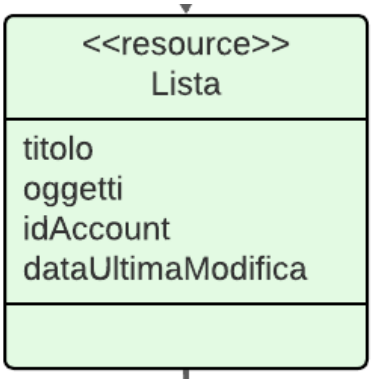
I metodi “modifica_nome”, “modifica_mail”, “modifica_password” di Account vengono trasformati in un’API **modificaAccount**. I metodi “crea_account”, “login”,

“elimina_account” di Gestione_account vengono trasformati nelle seguenti API: **registra**, **login**, **eliminaAccount**.

- Lista e Gestione_liste

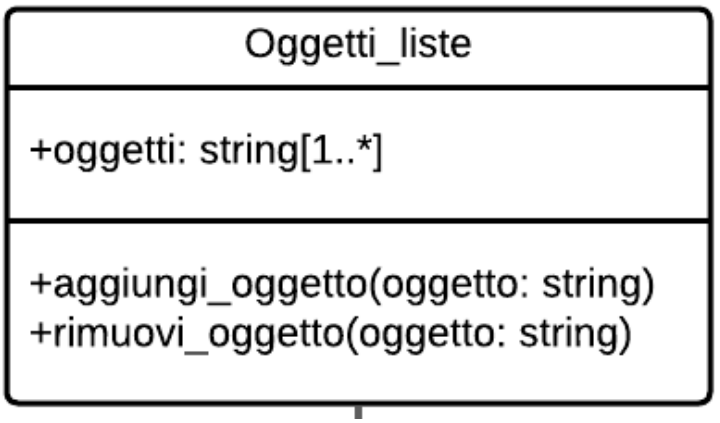


Da queste due classi abbiamo deciso di ricavare la risorsa **Lista**, contenente gli attributi “titolo”, “oggetti”, “idAccount”, “dataUltimaModifica”.

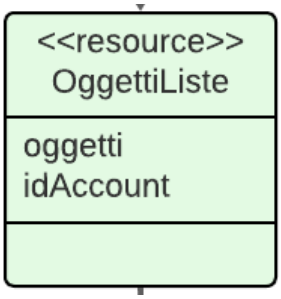


I metodi “crea_nuova_lista”, “elimina_lista” e “modifica_lista” di Gestione_liste vengono trasformati nelle seguenti API: **creaLista**, **eliminaLista** e **modificaLista.** Risulta inoltre necessario definire la seguente API: **ottieniLista**

- OggettiListe



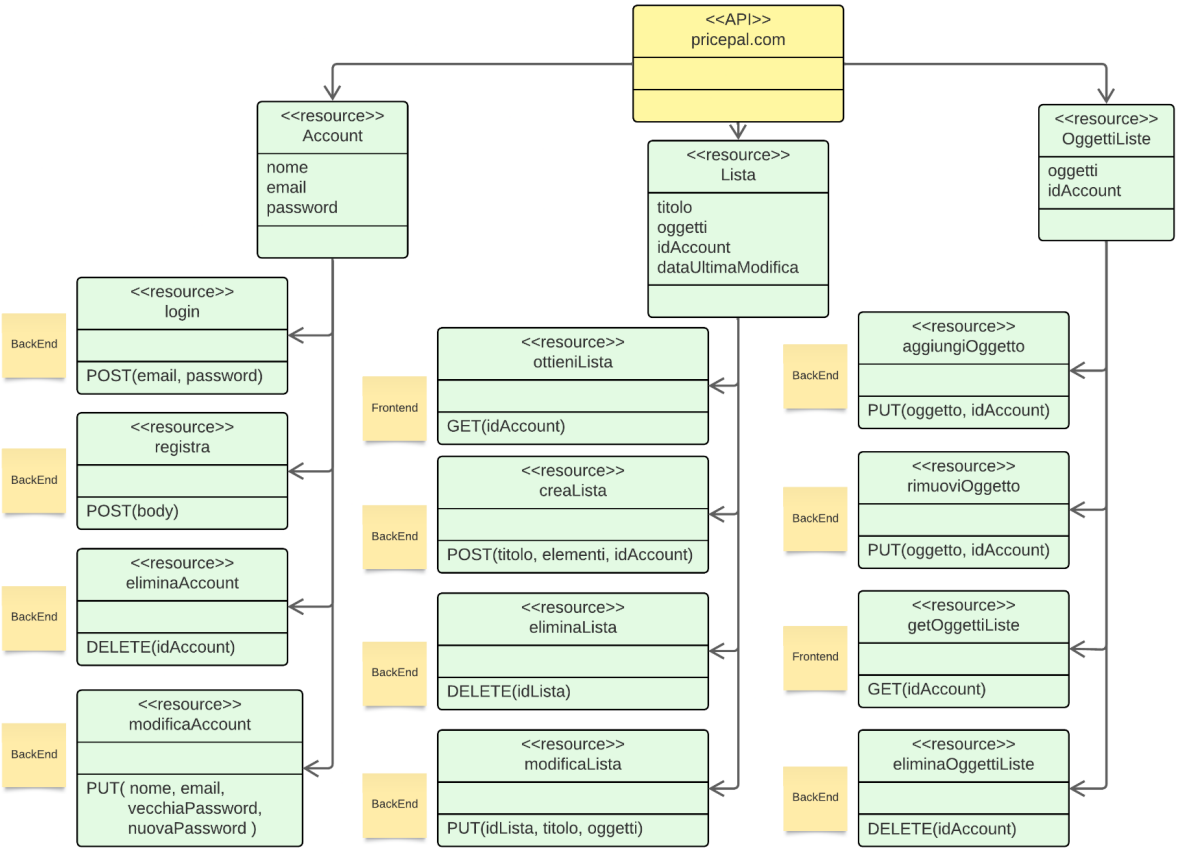
Da questa classe abbiamo deciso di ricavare la risorsa **OggettiListe**, contenente gli attributi “oggetti” e “idAccount”.



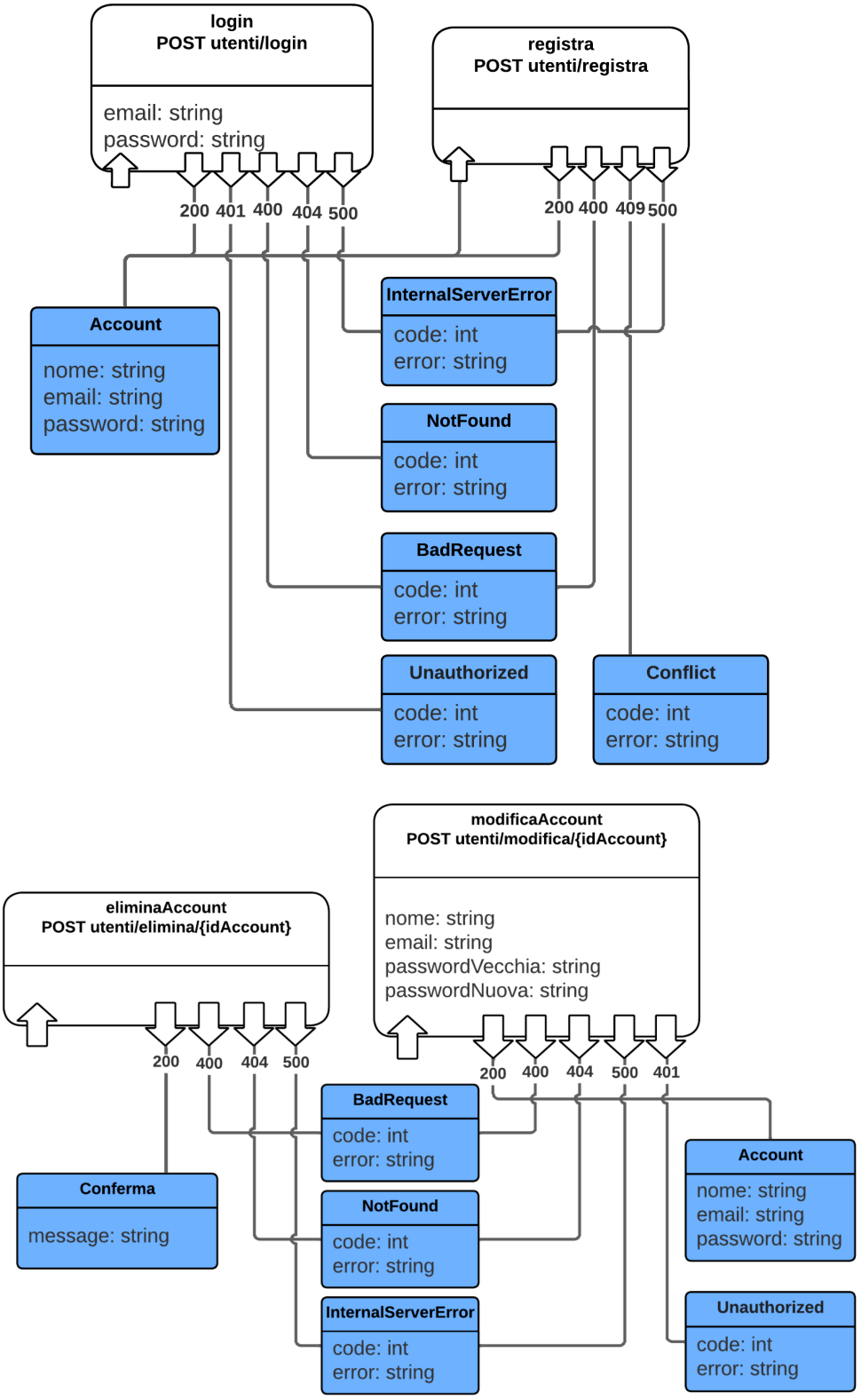
I metodi “aggiungi_oggetto” e “rimuovi_oggetto” vengono trasformati nelle API **aggiungiOggetto** e **rimuoviOggetto**. Risulta inoltre necessario creare le seguenti API: **getOggettiListe** e **eliminaOggettiListe**.

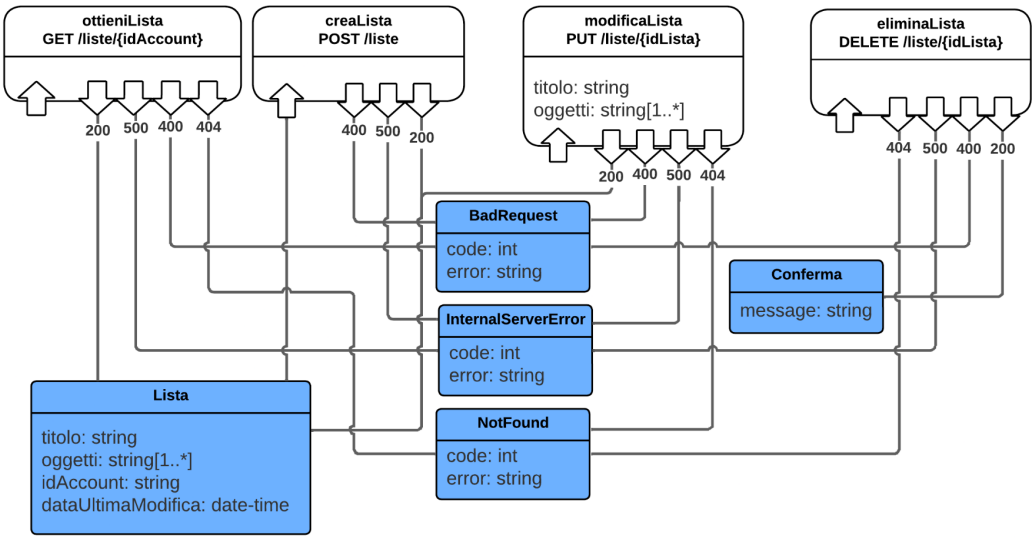
Diagramma completo

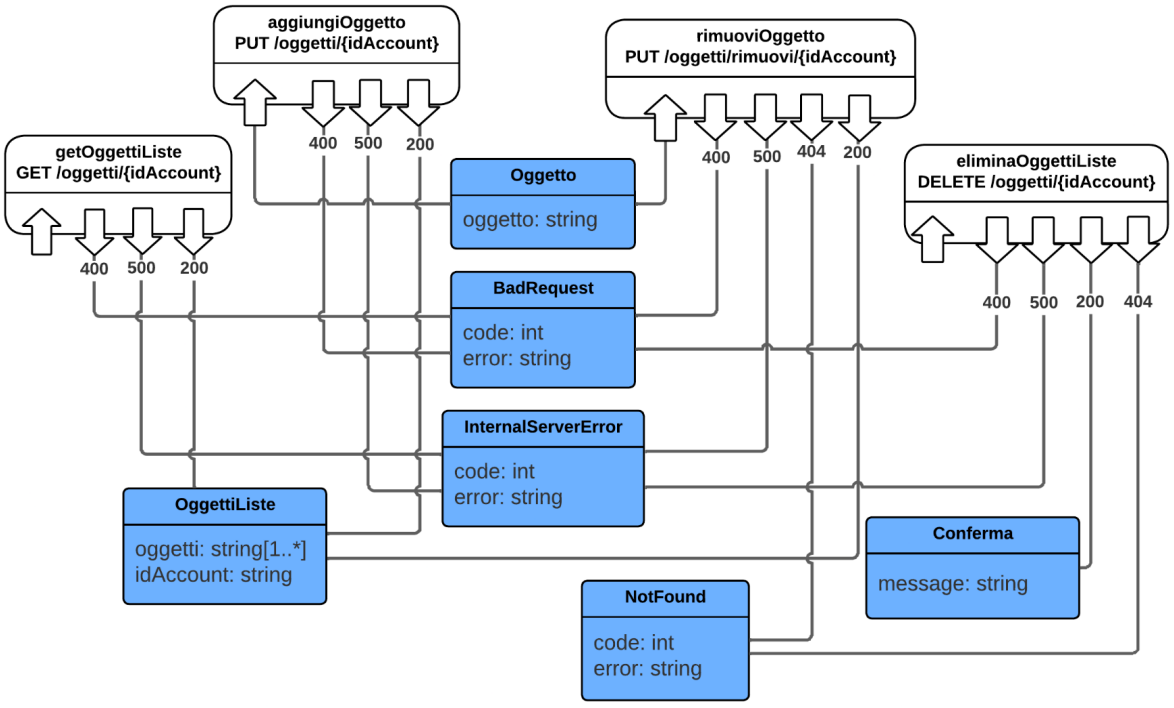
Segue il diagramma di estrazione delle risorse dal class diagram completo.



4. Diagramma delle risorse







API di Login (/login):

L'API riceve una richiesta POST contenente nel body le credenziali dell'account (email e password). Viene fatto un controllo se le credenziali, nel caso di una risposta positiva viene cercato l'utente nel database utilizzando l'email fornita. Se l'account esiste, viene controllato se la password fornita corrisponde alla password memorizzata nel database. In caso di successo l'API restituisce un messaggio di successo e le informazioni dell'account.
L'API restituirà invece un errore se:

- Il campo email non è presente nel body oppure non è di tipo string;
- Il campo password non è presente nel body oppure non è di tipo string;
- Non esiste nessun account associato a email;
- password è errata.

API di Registrazione (/registra):
Riceve una richiesta POST contenente nel body i dettagli dell'account da registrare (nome, email e password). Verifica se esiste già un account con la stessa email nel database e, nel caso l'account non esista, aggiunge le informazioni dell'account nel database criptando la password utilizzando bcrypt. Restituisce un messaggio di successo e le informazioni dell'account registrato.
L'API restituirà invece un errore se:

- Il campo email non è presente nel body oppure non è di tipo string
- Il campo password non è presente nel body oppure non è di tipo string;
- il campo nome non è presente nel body oppure non è di tipo string;
- Esiste già un'account registrato con la email presente nel body.

API di Eliminazione Account (/elimina/{idAccount}):
Riceve una richiesta DELETE contenente nell'url l'ID dell'account da eliminare. Controlla se l'ID è fornito e valido e cerca l'utente nel database utilizzando l'ID fornito. Se l'utente esiste, lo elimina dal database insieme a tutte le liste e la lista di oggetti abbinate all'utente. Infine restituisce un messaggio di successo.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Non esiste nessun account associato a idAccount.

API di Modifica dell'Account (/modifica/{idAccount}):
Riceve una richiesta PUT contenente nell'url l'ID dell'account da modificare e nel body i dati da aggiornare (nome, email, nuovaPassword) e la password di conferma (vecchiaPassword). Controlla se l'ID è fornito è valido e cerca l'utente nel database utilizzando l'ID. Se l'utente esiste, verifica la vecchia password, aggiorna i dati dell'utente e restituisce un messaggio di successo.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Non esiste nessun account associato a idAccount;
- vecchiaPassword non è presente o è errata.

Ottieni Lista (GET /{idAccount}):
Riceve una richiesta GET contenente idAccount come parametro nell'url. Se l'idAccount è valido cerca tutte le liste associate a quell'account. Restituisce le liste trovate se presenti.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Non esiste nessuna lista associata ad idAccount.

Crea Lista (POST /):
Riceve una richiesta POST contenente i dati necessari per creare una nuova lista nel body della richiesta (titolo, oggetti, idAccount). Verifica la presenza dei dati necessari e la validità

di idAccount. Nel caso di conferma crea una nuova istanza di Lista con i dati forniti e la salva nel database. Restituisce un messaggio di successo e la lista appena creata.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Nel body mancano i dati oppure sono presenti dati non validi.

Elimina Lista (DELETE /{idLista}):
Riceve una richiesta DELETE contenente come parametro idLista da eliminare. Una volta fatta una verifica sulla validità dell'idLista cerca la lista corrispondente all'ID fornito e la elimina. Restituisce un messaggio di successo se la lista è stata eliminata.
L'API restituirà invece un errore se:

- idLista non è presente o non rispetta il formato degli id di MongoDB;
- idLista non identifica una lista presente nel database.

Modifica Lista (PUT /{idLista}):
Riceve una richiesta PUT contenente l'idLista come parametro nell'url e i dati da aggiornare (titolo, oggetti) nel body della richiesta.
Verifica la presenza e la validità dell'ID, la presenza di titolo e la presenza di oggetti nel body della richiesta. Aggiorna la lista con i nuovi valori forniti, inclusa la data di ultima modifica. Restituisce un messaggio di successo insieme ai dettagli della lista appena modificata.
L'API restituirà invece un errore se:

- idLista non è presente o non rispetta il formato degli id di MongoDB;
- titolo non è presente nel body;
- oggetti non è presente nel body;
- idLista non identifica una lista presente nel database.

API per ottenere tutti gli oggetti mai inseriti da un account
(GET /api/oggetti/:idAccount)
Riceve una richiesta GET con l'ID dell'account come parametro nell'url. Verifica che l'ID dell'account sia valido e restituisce la lista di oggetti mai inseriti da quell'account. In caso di successo, viene restituita una risposta JSON contenente un messaggio di successo e la lista di oggetti.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB.

Aggiungi oggetto (PUT /api/oggetti/{idAccount})
L' API aggiunge un oggetto alla lista di oggetti mai inseriti da un determinato account. Riceve una richiesta PUT con l'ID dell'account nell'url e l'oggetto da aggiungere nel body della richiesta. Verifica che l'ID dell'account e l'oggetto siano validi. Se l'account esiste, aggiunge l'oggetto alla lista degli oggetti mai inseriti da quell'account. Se l'account esiste, ma non ha una lista associata, la funzione ne crea una. Restituisce una risposta JSON contenente un messaggio di successo e la lista di oggetti aggiornata.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Il campo oggetto non è presente nel body oppure non è di tipo string.

Rimozione di un oggetto dalla lista associata a un account:
(PUT /api/oggetti/rimuovi/:idAccount)
Riceve una richiesta PUT con l'ID dell'account nel percorso e l'oggetto da rimuovere nel body della richiesta. Verifica se l'ID dell'account e l'oggetto sono validi. Rimuove l'oggetto dalla lista degli oggetti mai inseriti da quell'account e restituisce una risposta JSON contenente un messaggio di successo e la lista di oggetti aggiornata.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- Il campo oggetto non è presente nel body oppure non è di tipo string.

Eliminazione dell'intera lista di oggetti associata a un account:
(DELETE /api/oggetti/:idAccount)
Riceve una richiesta DELETE con l'ID dell'account nell'url. Verifica se l'ID dell'account è valido. Elimina completamente la lista di oggetti mai inseriti da quell'account e restituisce una risposta JSON con un messaggio di successo.
L'API restituirà invece un errore se:

- idAccount non è presente o non rispetta il formato degli id di MongoDB;
- La lista di oggetti mai inseriti da quell'account non esiste.

5. Documentazione delle API

Le API che abbiamo descritto nella sezione precedente sono state documentate usando "Swagger".

La documentazione è attualmente disponibile al seguente link:
INSERIRE LINK

Nel caso in cui venga scaricata la repository "backend" e si avvii il server in locale sulla propria macchina, l'endpoint da invocare per raggiungere la documentazione è:
http://localhost:3000/api-docs

Più avanti sono date tutte le informazioni su come scaricare ed avviare sia il front-end che il back-end in locale.

6. Implementazione backend

TODO AGO

7. Testing

TODO AGO

8. Deployment