



UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza
dell'Informazione

Progetto:



Titolo del documento:

Sviluppo Applicazione

Informazioni documento:

| | | | |
|------------------------|-----------------|--------------------------|-----------|
| Nome documento: | D4–T06.pdf | | |
| Membri: | Alberto Cimmino | Salvatore Gilles Cassarà | Sara Tait |

Indice

| | |
|--|----|
| 1. <u>User Flows</u> | 3 |
| 2. <u>Implementation and Documentation</u> | 5 |
| 3. <u>API Documentation</u> | 15 |
| 4. <u>FrontEnd Implementation</u> | 16 |
| 5. <u>GitHub Repository</u> | 19 |
| 6. <u>Testing</u> | 20 |

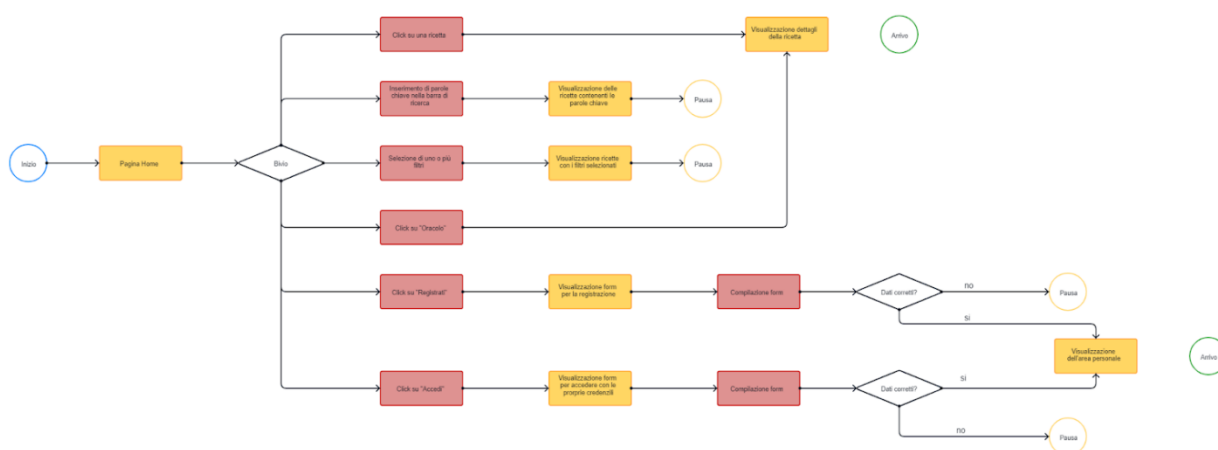
1. User Flows

In questa sezione sono riportati tre user flow, che illustrano le sequenze di azioni che gli utenti possono compiere navigando il sito. Ogni diagramma identifica le azioni specifiche per ogni tipo di utente, in ordine: utente generico, autenticato e premium.

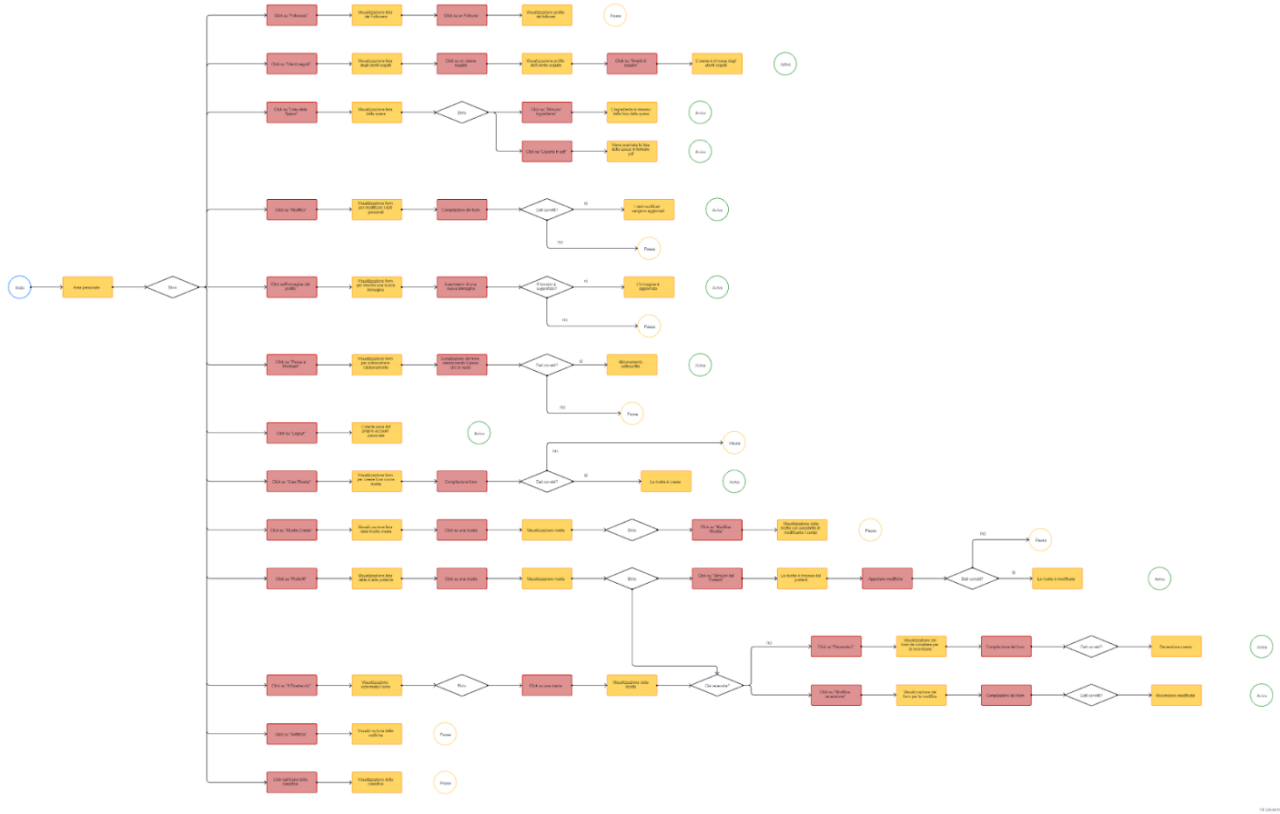
LEGENDA:



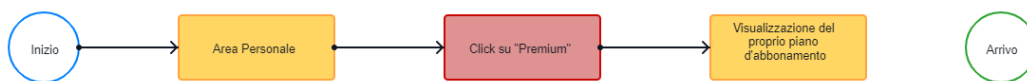
Utente Generico:



Utente Autenticato:



Utente Premium:

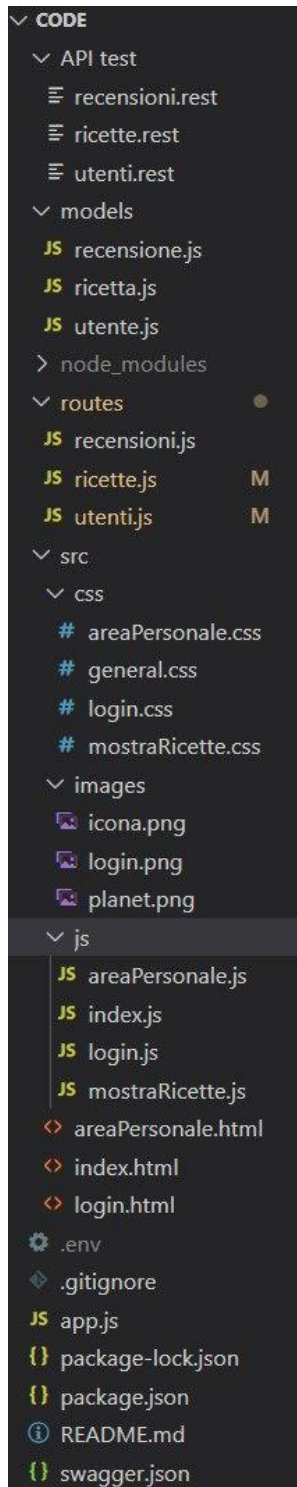


Per difficoltà di visualizzazione degli User Flow, è qui di seguito riportato il [link](#) dei file originali, più facilmente consultabili.

2. Implementation and Documentation

2.1 Project Structure

Nell'immagine, è mostrata la struttura del progetto.



In “API test” sono contenuti i file .rest utilizzati per testare le API.

La cartella “models” contiene i modelli utilizzati nel database per rappresentare utenti, ricette e recensioni.

Nella cartella “routes” sono contenute le implementazione delle API usate dal sito.

In “src” sono contenuti i file .css, le immagini utilizzate nel Front End, i file .js che gestiscono le relazioni tra il front-end e il back-end e i file .html corrispondenti alle pagine Home, Login e Area Personale del Front End. Il file principale è “app.js”, in cui è presente la connessione al database e la gestione delle richieste alle API. Si occupa inoltre di far partire un server locale.

Il file “swagger.json” contiene la documentazione delle API.

2.2 Project Dependencies

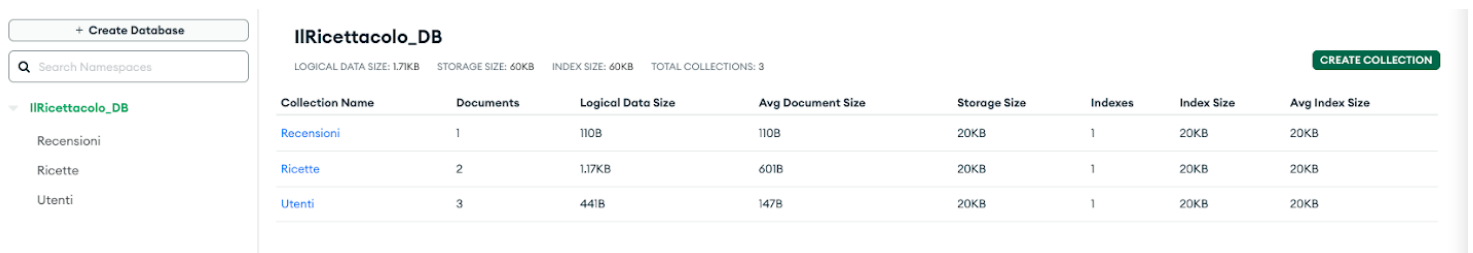
Per lo sviluppo della applicazione sono stati utilizzati i seguenti moduli node:

- MongoDB
- Express
- dotenv
- Swagger

È inoltre stato utilizzato REST CLIENT, estensione per Visual Studio, per testare le API.

2.3 Project Data or DB

Per gestire i dati dell'applicativo web, è stato utilizzato MongoDB. Sono state definite tre collezioni: "Utenti", "Ricette" e "Recensioni", come rappresentato nella seguente immagine:



The screenshot shows the MongoDB Atlas interface for a database named 'IIRicettacolo_DB'. On the left, there's a sidebar with a search bar and a list of collections: 'Recensioni', 'Ricette', and 'Utenti'. The main panel displays a table with the following data:

| Collection Name | Documents | Logical Data Size | Avg Document Size | Storage Size | Indexes | Index Size | Avg Index Size |
|-----------------|-----------|-------------------|-------------------|--------------|---------|------------|----------------|
| Recensioni | 1 | 110B | 110B | 20KB | 1 | 20KB | 20KB |
| Ricette | 2 | 1.17KB | 601B | 20KB | 1 | 20KB | 20KB |
| Utenti | 3 | 441B | 147B | 20KB | 1 | 20KB | 20KB |

- Utenti rappresenta gli utenti che si sono registrati sul sito. Un esempio del dato utenti:

```
_id: ObjectId('63983df4a44cf1f0e1728f34')
UID: "000000001"
nomeUtente: "iron16bit"
password: "12345678"
email: "iron16bit@gmail.com"
pfp: null
indirizzo: null
```

- Ricette rappresenta le ricette create dagli utenti, come la seguente:

| | | |
|----|---|----------|
| 1 | <code>_id: ObjectId('63984171579c511003b92ad4')</code> | ObjectId |
| 2 | <code>IDRicetta: "00000002"</code> | String |
| 3 | <code>nome: "Crepes"</code> | String |
| 4 | <code>autoreUID: "00000003"</code> | String |
| 5 | <code>ingredienti: Array</code> | Array |
| 6 | <code>0: "3 uova"</code> | String |
| 7 | <code>1: "500g Latte"</code> | String |
| 8 | <code>2: "250g Farina"</code> | String |
| 9 | <code>3: "40g Burro"</code> | String |
| 10 | <code>procedimento: "Versate un mestolo di impasto sufficiente a ricoprire la superficie dell"</code> | String |
| 11 | <code>immagini: Array</code> | Array |
| 12 | <code>0: "https://www.google.com/url?sa=i&url=https%3A%2F%2Fricette.giallozaffer"</code> | String |
| 13 | <code>giudizi: null</code> | Null |

- Recensioni rappresenta le recensioni lasciate da un utente ad una ricetta. Un esempio del dato recensioni può essere:

```

_id: ObjectId('63984171579c511003b92ad5')
IDRecensione: "000000001"
autoreUID: "000000002"
numeroStelle: 3
commento: null

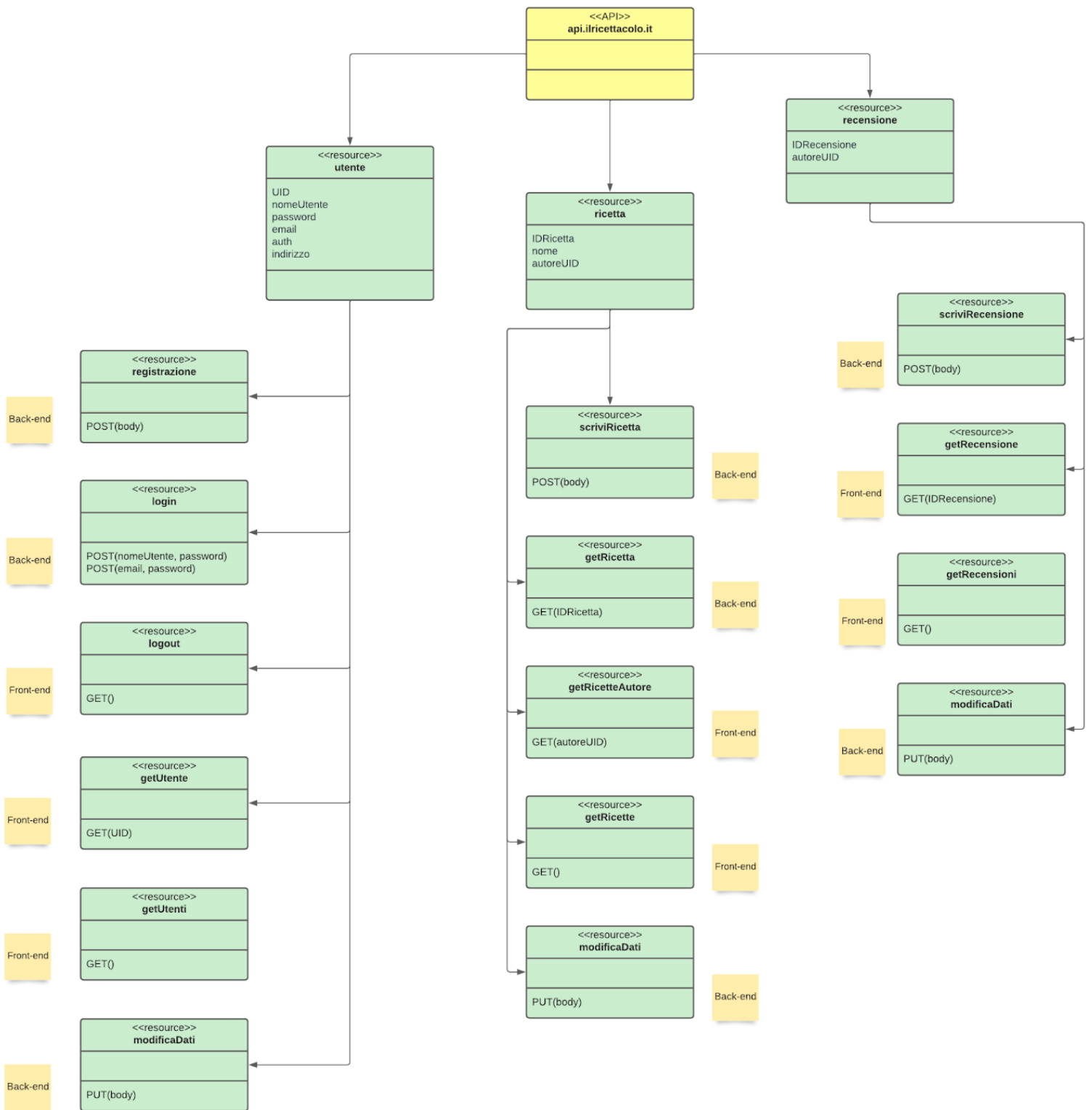
```


2.4 Project APIs

2.4.1 Resources Extraction from the Class Diagram

Analizzando il diagramma delle classi, sono state individuate tre risorse principali: l'utente, le ricette e le recensioni.

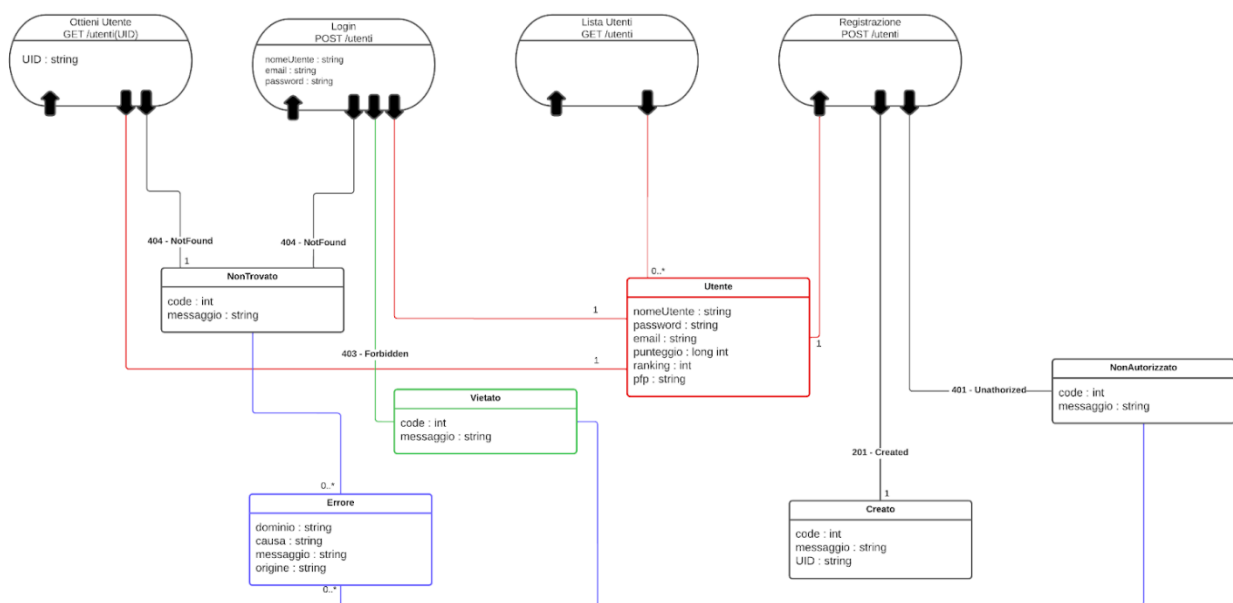
- L'utente presenta i seguenti metodi HTTP:
 - registrazione: un metodo POST che prende in input tutti i dati dell'utente (body) e crea la risorsa utente;
 - login: un metodo POST che può prendere in input nome utente e password od email e password. Anche questo metodo crea la risorsa utente;
 - logout: un metodo GET che libera la risorsa utente;
 - getUtente: un metodo GET che prende in input un UID e ritorna la risorsa utente che corrisponde a quell'UID;
 - getUtenti: un metodo GET che ritorna tutte le risorse utente;
 - modificaDati: metodo PUT che prende in input i nuovi dati dell'utente (body) e li aggiorna.
- La ricetta presenta i seguenti metodi HTTP:
 - scriviRicetta: un metodo POST che prende in input tutti i dati della ricetta (body) e crea la risorsa ricetta;
 - getRicetta: un metodo GET che prende in input un IDRicetta e ritorna la risorsa ricetta che corrisponde a quell'IDRicetta;
 - getRicetteAutore: un metodo GET che prende in input un autoreID e ritorna la risorsa ricetta che contengono quell'autoreID;
 - getRicette: un metodo GET che ritorna tutte le risorse ricetta;
 - modificaDati: metodo PUT che prende in input i nuovi dati della ricetta (body) e li aggiorna.
- La recensione presenta i seguenti metodi HTTP:
 - scriviRecensione: un metodo POST che prende in input tutti i dati della recensione (body) e crea la risorsa recensione;
 - getRecensione: un metodo GET che prende in input un IDRecensione e ritorna la risorsa recensione che corrisponde a quell>IDRecensione;
 - getRecensioni: un metodo GET che ritorna tutte le risorse recensione;
 - modificaDati: metodo PUT che prende in input i nuovi dati della recensione (body) e li aggiorna.



2.4.2 Resources Models

Tramite l'analisi delle risorse, è possibile definire i diagrammi delle API dell'utente, della ricetta e della recensione. In questi diagrammi, vengono descritti più nel dettaglio i metodi GET e POST.

Utente

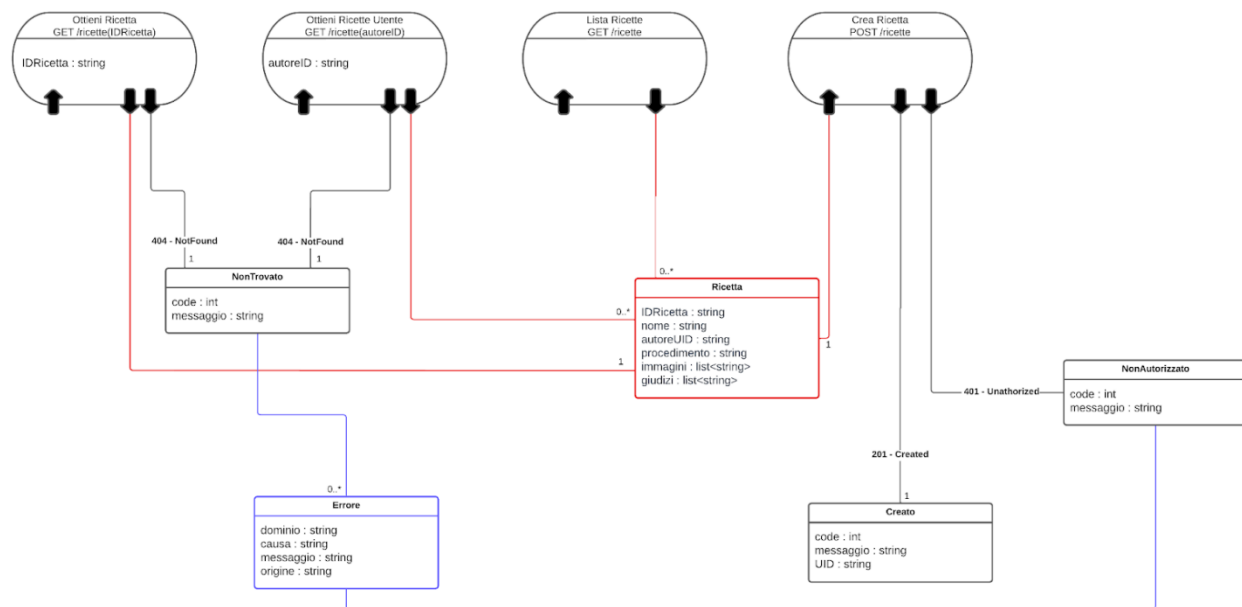


Quando viene effettuata la registrazione, viene eseguito un POST dei dati dell'utente, creando così la risorsa utente e salvandone i dati (risposta 201). Questo processo può non avere successo nel caso di mancanza dell'autorizzazione necessaria a creare una risorsa (risposta 401).

Una volta che i dati di un utente sono stati salvati, è possibile riaccedere a quei dati tramite il login, ovvero inseriti nome utente e password od email e password, se questi corrispondono a dei dati esistenti, viene effettuato un POST che crea la risorsa utente caricando i dati corrispondenti. Nel caso nessuno dei dati di accesso inseriti sia corretto, non viene trovato nessun utente (risposta 404). Se solo parte dei dati non corrispondono, non è consentito il login (risposta 403).

Infine, è possibile ottenere i dati degli utenti già esistenti con dei metodi GET. Questo può avvenire per tutti gli utenti tramite "Lista Utenti" o per un utente specifico tramite "Ottieni Utente" avendo come input l'UID corrispondente all'utente desiderato. Se quest'ultimo non corrisponde a nessun utente, non viene ritornato nulla (risposta 404).

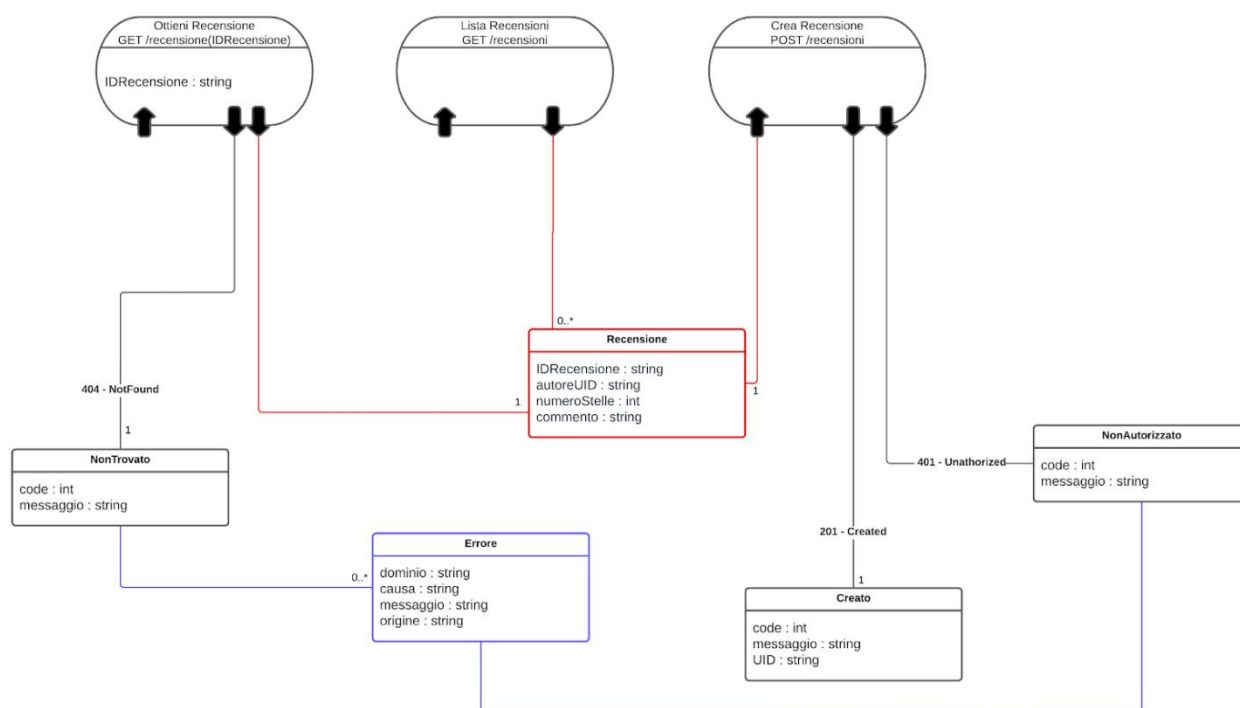
Ricetta



Quando si scrive una ricetta, tramite “Crea Ricetta”, i dati della ricetta in input vengono usati per creare la risorsa ricetta corrispondente tramite un POST (risposta 201) ed i suoi dati vengono salvati. Questo processo può non avere successo nel caso di mancanza dell’autorizzazione necessaria a creare una risorsa (risposta 401).

È possibile ottenere i dati di una ricetta tramite i metodi GET di “Lista Ricette”, che ritorna tutte le ricette esistenti, “Ottieni Ricette Utente”, che prende come input un autoreID e ritorna tutte le ricette scritte da quell’utente, e “Ottieni Ricetta”, che prende come input un IDRicetta e ritorna la corrispondente ricetta. Questi ultimi metodi possono non trovare nessuna ricetta (risposta 404).

Recensione



Con “Crea Recensione”, i dati della recensione dati in input vengono usati per creare la risorsa ricetta corrispondente tramite un POST (risposta 201) ed i suoi dati vengono salvati. Questo processo può non avere successo nel caso di mancanza dell'autorizzazione necessaria a creare una risorsa (risposta 401).

È possibile ottenere i dati di una recensione tramite i metodi GET di “Lista Recensioni”, che ritorna tutte le ricette esistenti, “Ottieni Recensione”, che prende come input un `IDRecensione` e ritorna la recensione corrispondente. Se l'`IDRecensione` non corrisponde a nessuna ricetta, non viene ritornato nulla (risposta 404).

2.5 Sviluppo API

Qui di seguito il codice delle principali API utilizzate nel progetto. Il codice completo di tutte le API è consultabile su GitHub.

2.5.1 Ricette

Tramite questa API si esegue una richiesta GET che ritorna tutte le ricette presenti nel database.

```
//get di tutti le ricette --getRicette()
router.get('/', async(req, res) =>{
  try{
    const ricette = await Ricetta.find()
    res.status(200).json({status: 200, message: ricette})
  }catch(err){
    res.status(500).json({status: 500,message : err.message})
  }
})
```

2.5.2 Crea Recensione

Tramite questa API si esegue una richiesta POST che crea una nuova recensione impostando come valori quelli passati nel corpo della richiesta.

```
//crea recensione
router.post('/:id', async(req, res) =>{
  const recensione = new Recensione({
    IDRecensione : Math.round(Math.random()*10000000).toString().padStart('8', 0),
    autoreUID: req.params.id,
    numeroStelle: Math.max(0, Math.min(req.body.numeroStelle, 5)),
    commento: req.body.commento
  })
  try{
    const nuovaRicetta = await recensione.save()
    //Error code 201: successfully created new resource
    res.status(201).json({staus: 201, message: nuovaRicetta})
  }catch (err){
    //Error code 400: Bad request
    res.status(400).json({status: 400, message:err.message})
  }
})
```


2.5.3 Login

Tramite questa API si effettua una richiesta POST che carica i dati dell'utente corrispondente ai dati passati nel corpo della richiesta.

```
//login -- login()
router.post('/:id', async (req, res) => {
  try{
    let utente = await Utente.findOne({nomeUtente : req.body.nomeUtente})
    if(!utente)
      return res.status(404).json({status: 404, message: "nome utente o password non corretti"})
    if(utente.password == req.body.password){
      //Error code 200: OK
      res.status(200).json({status:200, UID: utente.UID, nomeUtente: utente.nomeUtente})
    }else{
      //Error code 404: not found
      res.status(404).json({status: 404, message: "nome utente o password non corretti"})
    }
  }catch (err){
    //Error code 500: Server side error
    res.status(500).json({status: 500, message:err.message})
  }
})
```


3. API documentation

La documentazione delle API locali è stata inserita nel file “swagger.json”, così da renderla disponibile a chiunque veda il codice sorgente.

È stato poi utilizzato il modulo Swagger UI Express di NodeJS per generare una pagina web dove è possibile visualizzare le definizioni delle specifiche API.

L’endpoint da invocare per raggiungere la pagina web relativa alla documentazione è:

<http://localhost:3000/api-docs>

In particolare mostriamo di seguito la pagina di Swagger relativa alle API che gestiscono le ricette:

| Ricette API per le ricette nel sistema | | ^ |
|--|---|---|
| GET | /ricette Get tutte le ricette presenti nel sistema | ▼ |
| GET | /ricette/{UID} Get tutte le ricette presenti nel sistema di un utente | ▼ |
| POST | /ricette/{UID} Creazione di una nuova ricetta | ▼ |

4. FrontEnd Implementation

Il Front End implementato consiste nella pagina Home, la pagina di Login e la pagina dell'Area Personale; queste forniscono le funzionalità di selezione di una ricetta, visualizzazione dei dettagli della stessa e di eventuali recensioni, login e accesso all'area personale.

4.1 HOME

La Home è la prima pagina che si visualizza accedendo al sito. Vengono mostrate tutte le ricette presenti nel database e, cliccando su una di esse, è possibile visualizzarne i dettagli (quindi ingredienti e procedimento) ed eventuali tag ed immagini. A lato sono mostrati i possibili tag delle ricette (benché la ricerca per filtri non sia implementata).

Da qui è possibile accedere alla pagina di Login cliccando sull'icona utente.



4.2 LOGIN

In questa pagina viene richiesto di inserire il proprio nome utente e la propria password. Se i dati sono corretti, cliccando sul pulsante “Login” si accede alla propria area personale, altrimenti viene visualizzato un messaggio di errore.

Premere sull'icona de “Il Ricettacolo” in alto a sinistra riporta l'utente alla pagina Home.



The screenshot shows the login page of 'Il Ricettacolo'. The header is purple with the site logo on the left, the title 'Il Ricettacolo' in the center, and a user icon on the right. A vertical sidebar on the left contains buttons for 'PRIMI PIATTI', 'SECONDI PIATTI', 'DOLCI', 'SENZA LATTOSIO', 'SENZA GLUTINE', and 'VEGETARIANO'. The main content area is light blue and features the title 'LOGIN' with a decorative icon. Below the title is a login form with two input fields: 'Nome utente' and 'Password'. A link 'Password dimenticata? Recupera Password' is positioned below the password field. A blue 'LOGIN' button is at the bottom of the form.



This screenshot shows the login page after an unsuccessful login attempt. The layout is identical to the previous one, but the input fields now contain error messages: 'NomeErrato' in the username field and 'passwordErrata' in the password field. Below the password field, the error message 'Nome utente e/o password errati' is displayed. The 'LOGIN' button remains visible at the bottom of the form.

4.3 AREA PERSONALE

Una volta effettuato il login l'utente viene portato all'area personale dove è presente l'elenco delle ricette create dall'utente. A lato vengono inoltre mostrati i dati personali quali nome utente, email e password. Sono presenti i pulsanti per modificare i dati personali e passare all'account premium (le funzionalità non sono tuttavia disponibili). Nell'immagine viene mostrata l'Area Personale dell'utente che ha come nome "iron16bit" e password "12345678".

Premere sull'icona de "Il Ricettacolo" riporta l'utente alla pagina Home.



5. GitHub Repository

Il progetto è presente su GitHub al seguente link: <https://github.com/Ricettacolo-UNITN>.
Esso è suddiviso in tre repository: "Documents", "Derivables" e "Code".

- "Documents" contiene le versioni in pdf e word di tutti i derivable;
- "Derivables" contiene solamente i pdf di tutti i derivable;
- "Code" contiene tutto il codice relativo al FrontEnd ed al BackEnd.

Gli account dei membri del gruppo sono:

- Alberto Cimmino: <https://github.com/bettozzo>
- Salvatore Gilles Cassarà: <https://github.com/Iron16Bit>
- Sara Tait: <https://github.com/SaraTait>

5.1 Informazioni per l'avvio del sito web

Così come descritto nel file "README.md" nella repository "Code", per avviare il sito web è necessario seguire la procedura qui descritta:

Prerequisiti:

Avere NodeJS con npm.

Istruzioni:

Clonare la repository:

```
git clone https://github.com/Ricettacolo-UNITN/Code.git
```

Accedere alla cartella "Code" così creata e creare un file ".env" contenente:

```
DATABASE_URL=mongodb+srv://admin:admin@cluster0.1emobth.mongodb.net/IRicettacolo_DB?retryWrites=true&w=majority
```

Installare le dipendenze con il comando:

```
npm install
```

Avviare il sito eseguendo:

```
npm run devStart
```

Collegarsi al sito tramite l'indirizzo su un browser supportato (vedere D1 e D2):

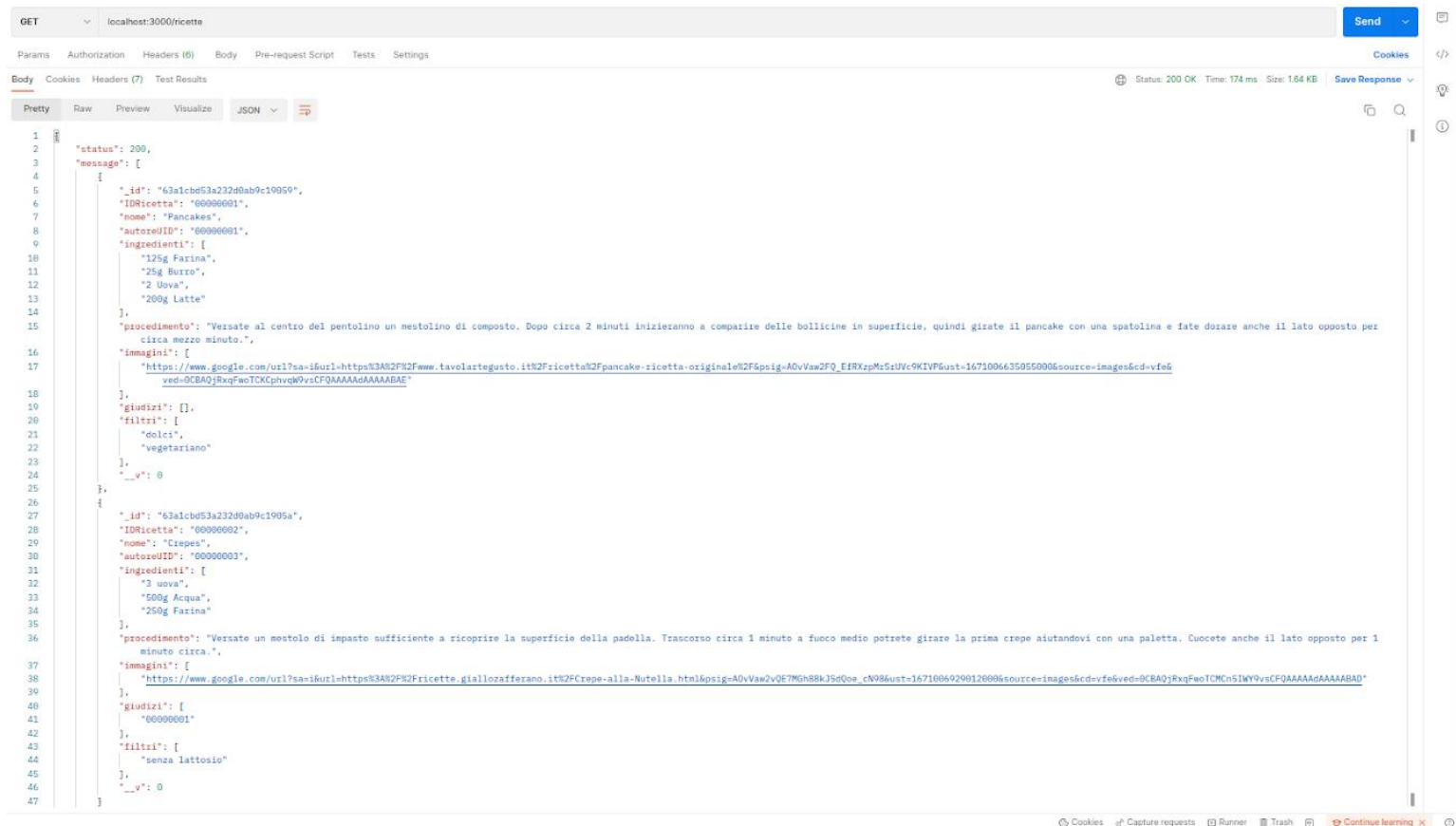
localhost:3000

6. Testing

Le API dell'applicativo web sono state testate tramite l'estensione REST Client di VS code (<https://marketplace.visualstudio.com/items?itemName=humao.rest-client>) e Postman (<https://www.postman.com/>).

Tutte le API sono state testate, ma qui di seguito sono riportati solo alcuni dei casi di test:

GET /ricette Dovrebbe ritornare tutte le ricette presenti nel database



```
1 {
2   "status": 200,
3   "message": [
4     {
5       "_id": "63a1cbd53a232d9ab9c19959",
6       "IDRicetta": "00000001",
7       "nome": "Pancakes",
8       "autoreID": "00000001",
9       "ingredienti": [
10        "125g Farina",
11        "25g Burro",
12        "2 Uova",
13        "200g Latte"
14      ],
15       "procedimento": "Versate al centro del pentolino un mestolino di composto. Dopo circa 2 minuti inizieranno a comparire delle bollicine in superficie, quindi girate il pancake con una spatolina e fate dorare anche il lato opposto per circa mezzo minuto.",
16       "immagini": [
17        "https://www.google.com/url?sa=i&url=https://3A82F92Fwww.tavolartegusto.it%2Fricetta%2Fpancake-ricetta-originale%2F6psig-A0vVaw2FQ_EIRKzPm5rUV-9KIVP6ust-1671006635955000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCXKphvq9vscFQAAAAAAGAAABAE"
18      ],
19       "giudizi": [],
20       "filtri": [
21        "dolci",
22        "vegetariano"
23      ],
24       "_v": 0
25     },
26     {
27       "_id": "63a1cbd53a232d9ab9c1995a",
28       "IDRicetta": "00000002",
29       "nome": "Crepes",
30       "autoreID": "00000003",
31       "ingredienti": [
32        "3 uova",
33        "500g Acqua",
34        "250g Farina"
35      ],
36       "procedimento": "Versate un mestolo di impasto sufficiente a ricoprire la superficie della padella. Trascorso circa 1 minuto a fuoco medio potrete girare la prima crepe aiutandovi con una paletta. Cuocete anche il lato opposto per 1 minuto circa.",
37       "immagini": [
38        "https://www.google.com/url?sa=i&url=https://3A82F92Fricette.giallozafferano.it%2FCrepe-alla-Nutella.html&psig=A0vVaw2vQe7M6h88k350oe_cN98&ust-1671006929012600&source=images&cd=vfe&ved=0CBAQjRxqFwoTCMCn5IWY9vsCFQAAAAAAGAAABAD"
39      ],
40       "giudizi": [
41        "00000001"
42      ],
43       "filtri": [
44        "senza lattosio"
45      ],
46       "_v": 0
47     }
48   ]
49 }
```

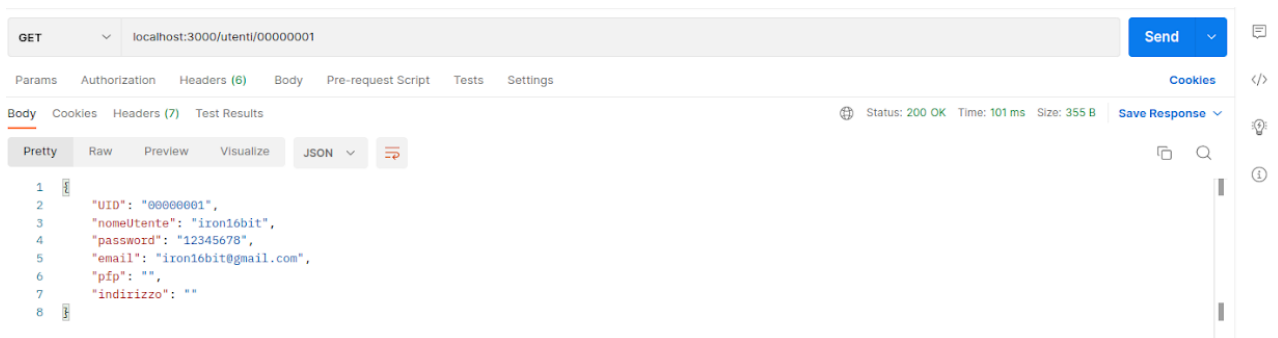
Il test da Postman ritorna due ricette, ovvero tutte quelle presenti in questo momento nel database.


```
1   ###get all ricette
    Send Request
2   GET http://localhost:3000/ricette
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 1441
5 ETag: W/"5a1-CG6nxws7ybla3QvTr/od/3dqinU"
6 Date: Tue, 20 Dec 2022 17:39:15 GMT
7 Connection: close
8
9 {
10   "status": 200,
11   "message": [
12     {
13       "_id": "63a1cbd53a232d0ab9c19059",
14       "IDRicetta": "00000001",
15       "nome": "Pancakes",
16       "autoreUID": "00000001",
17       "ingredienti": [
18         "125g Farina",
19         "25g Burro",
20         "2 Uova",
21         "200g Latte"
22       ],
23       "procedimento": "Versate al centro del pentolino un mestolino di compost
o. Dopo circa 2 minuti inizieranno a comparire delle bollicine in superficie,
quindi girate il pancake con una spatolina e fate dorare anche il lato opposto
per circa mezzo minuto.",
24       "immagini": [
25         "https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.tavolartegusto.
it%2Fricetta%2Fpancake-ricetta-originale%2F&psig=AQvVaw2F0_EfrXzpMrSriWc9KIVP&
ust=1671006635055000&source=images&cd=vfe&ved=0CBA0jRxqFwoTCKCphvqW9vsCF0AAAAA
dAAAAABAE"
26       ],
27       "giudizi": [],
28       "filtri": [
29         "dolci",
30         "vegetariano"
31       ],
32       "__v": 0
33     },
34     {
35       "_id": "63a1cbd53a232d0ab9c1905a",
36       "IDRicetta": "00000002",
37       "nome": "Crepes",
```

Allo stesso modo, anche il testing tramite REST Client ritorna due ricette.

GET utenti/UID



Richiedendo tramite Postman l'utente con UID "00000001", ritorna correttamente l'utente corrispondente.

```
4   ###get one utente
    Send Request
5   GET http://localhost:3000/utenti/00000001
```

```
1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  Content-Type: application/json; charset=utf-8
4  Content-Length: 119
5  ETag: W/"77-TBAmkHxveyDoc6Wr0jPn3jlcnAs"
6  Date: Tue, 20 Dec 2022 17:42:25 GMT
7  Connection: close
8
9  {
10   "UID": "00000001",
11   "nomeUtente": "iron16bit",
12   "password": "12345678",
13   "email": "iron16bit@gmail.com",
14   "pfp": "",
15   "indirizzo": ""
16 }
```

Anche il testing tramite REST Client ritorna lo stesso utente

GET recensioni/IDRecensione



Il testing tramite Postman ritorna la recensione corrispondente allo IDRecensione "00000001".

```
4   ###get one recensione
    Send Request
5   GET http://localhost:3000/recensioni/00000001
6
```

```
1  HTTP/1.1 200 OK
2  X-Powered-By: Express
3  Content-Type: application/json; charset=utf-8
4  Content-Length: 216
5  ETag: W/"d8-Z1it2S843qRRvdgjbN1ZG1q+TAU"
6  Date: Tue, 20 Dec 2022 17:49:13 GMT
7  Connection: close
8
9  {
10   "status": 200,
11   "message": [
12     {
13       "_id": "63a1cbd53a232d0ab9c1905b",
14       "IDRecensione": "00000001",
15       "autoreUID": "00000002",
16       "numeroStelle": 3,
17       "commento": "Interessante ricetta senza lattosio, ma preferisco quelle cl
18       assiche",
19       "__v": 0
20     }
21   ]
22 }
```

Anche il testing tramite REST Client ritorna la stessa recensione.

Documento: "D4-T06.pdf"