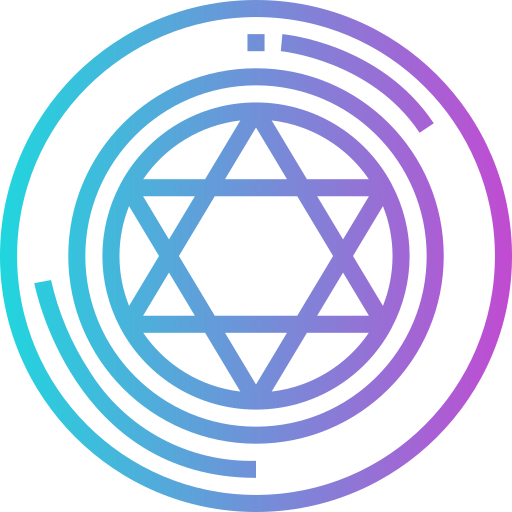
**Progetto:**



**Il Ricettacolo**

**Titolo del documento:**

Sviluppo Applicazione

**Informazioni documento:**

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

**Indice**

1. [**User Flows**](#Flows) **3**
2. [**Implementation and Documentation**](#Implementazione) **5**
3. [**API Documentation**](#Documentazione) **15**
4. [**FrontEnd Implementation**](#FrontEnd) **16**
5. [**GitHub Repository**](#GitHub) **19**
6. [**Testing**](#Testing) **20**

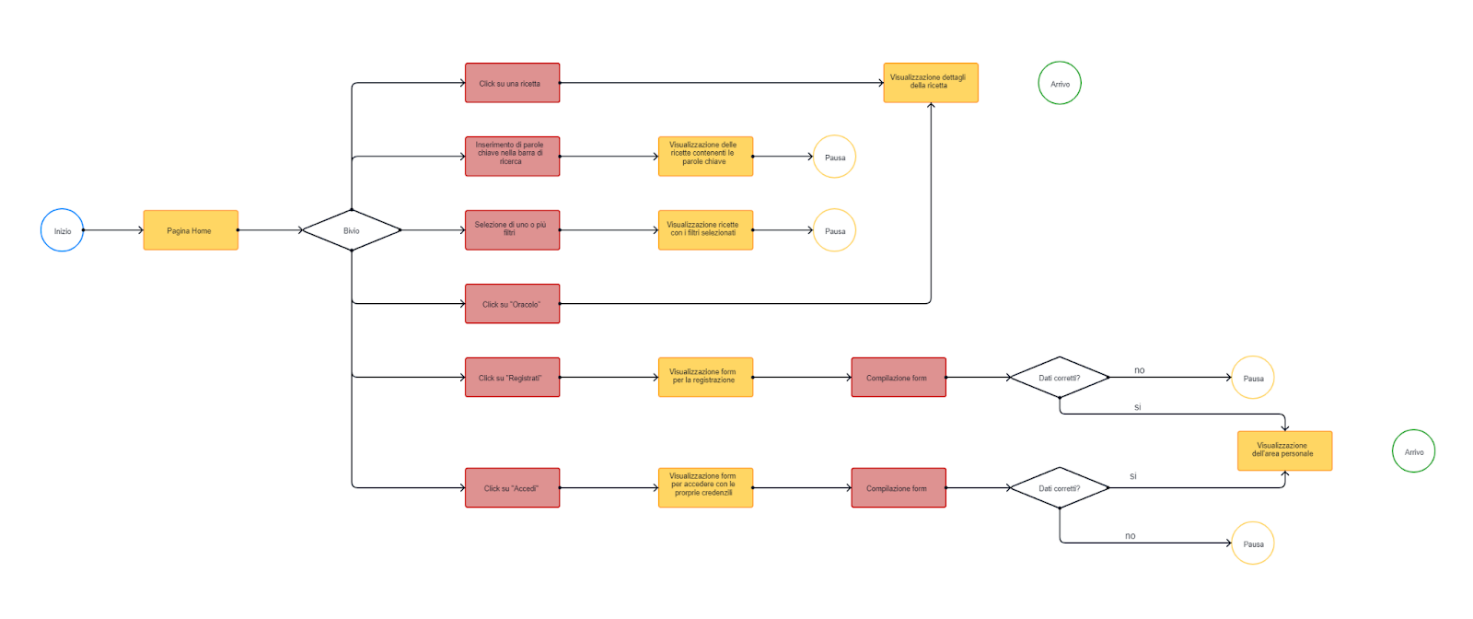
1. **User Flows**

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.

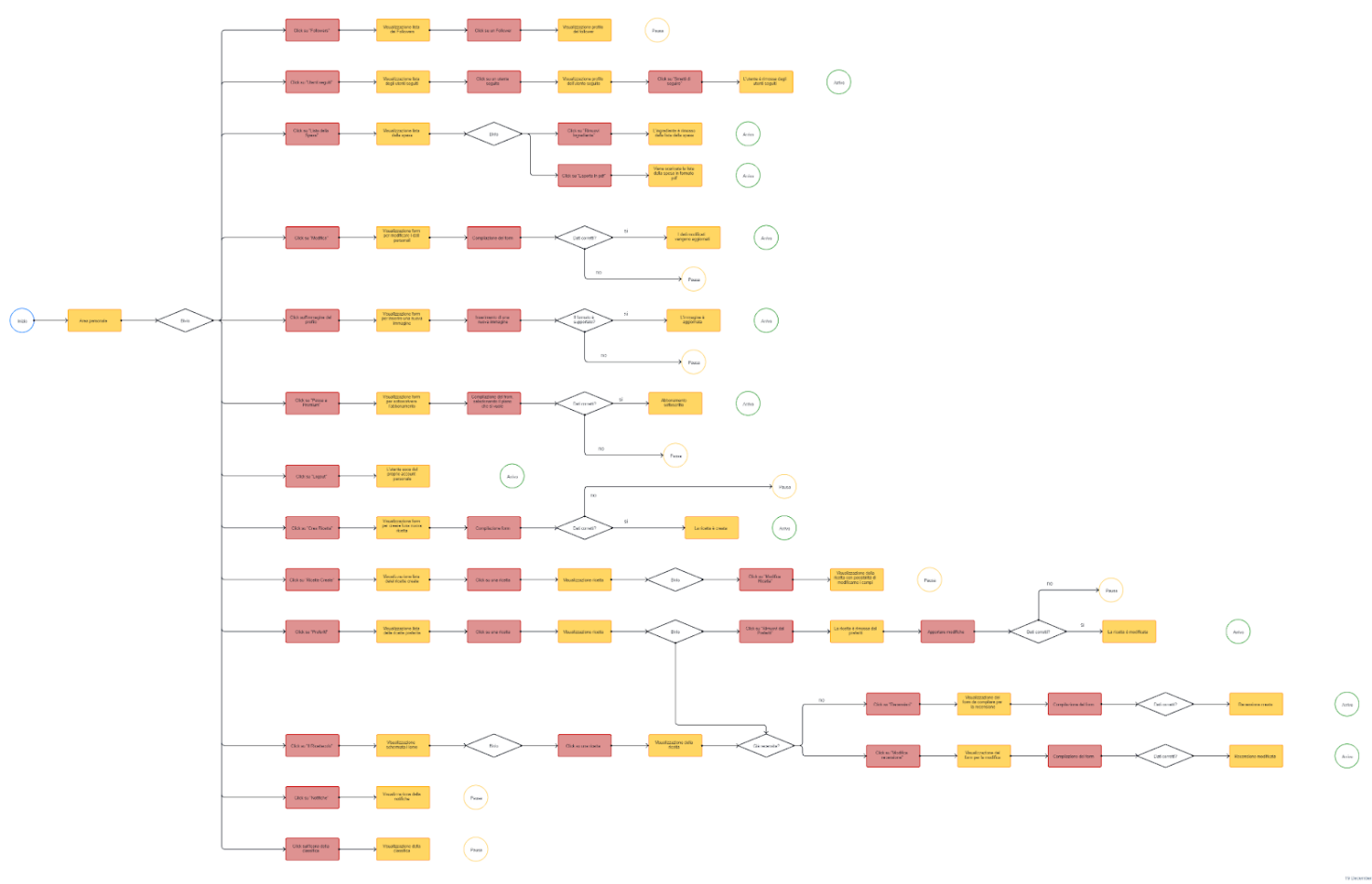
LEGENDA:



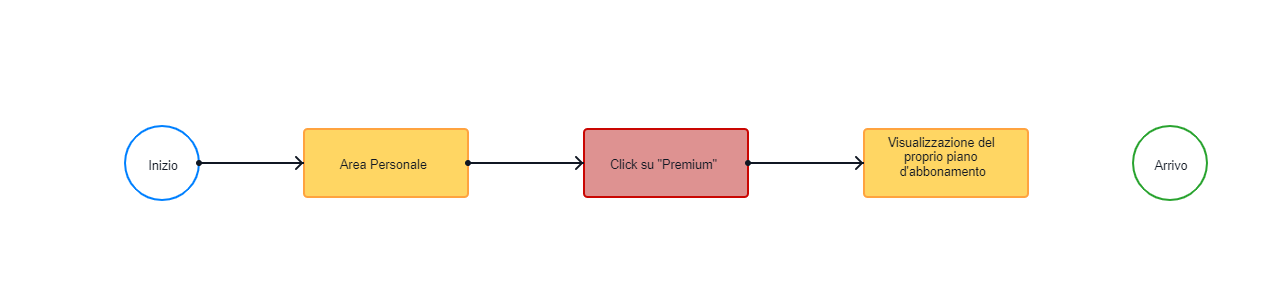
Utente Generico:



Utente Autenticato:



Utente Premium:

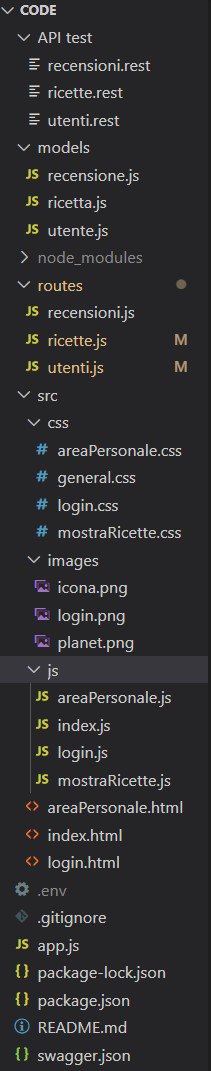


Per difficoltà di visualizzazione degli User Flow, è qui di seguito riportato il [link](https://app.flowmapp.com/share/projects/8028be7c-54c5-41ea-9a6b-71f8b1bca4cc/userflow/2d3ff083-e5de-404e-8590-4d77f3816007) dei file originali, più facilmente consultabili.

1. **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.

1. **Requisiti Non Funzionali**

**2.2 Project Dependencies**

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

- MongoDB

- Express

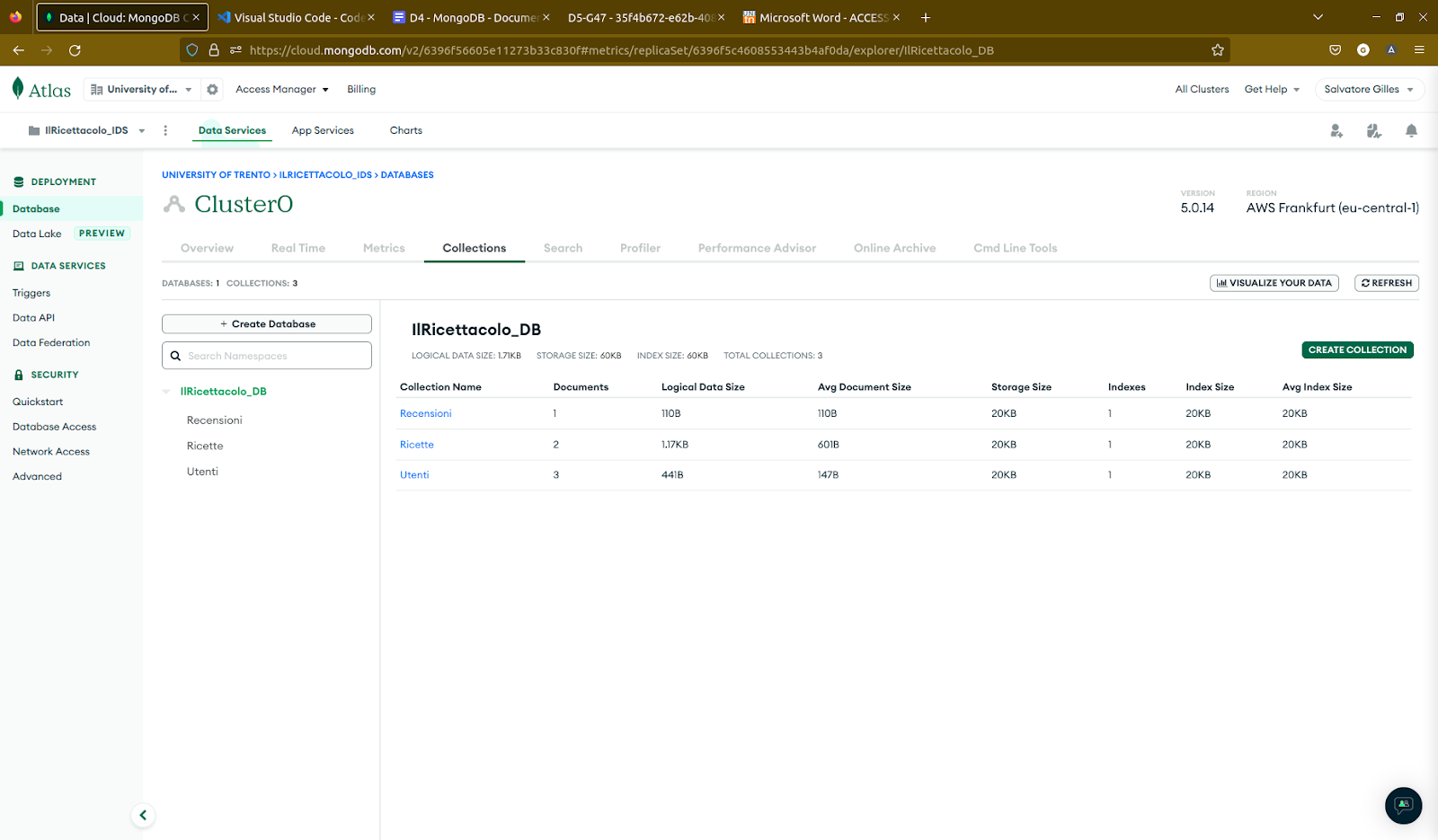
- dotenv

- Swagger

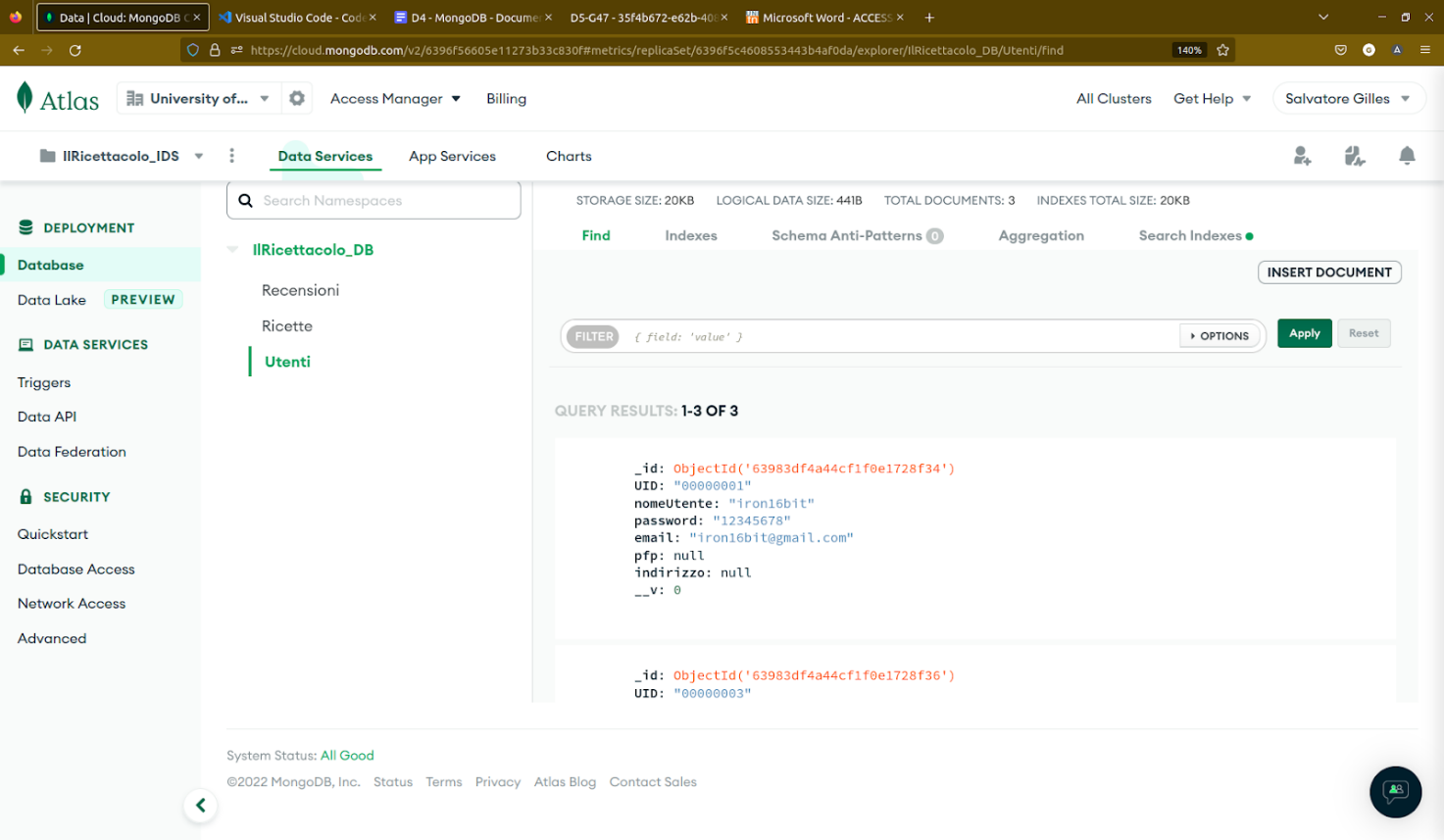
È inoltre stato utilizzato REST CLIENT, estensione per Visual Studio, per testare le API.

**2.3 Project Data or DB**

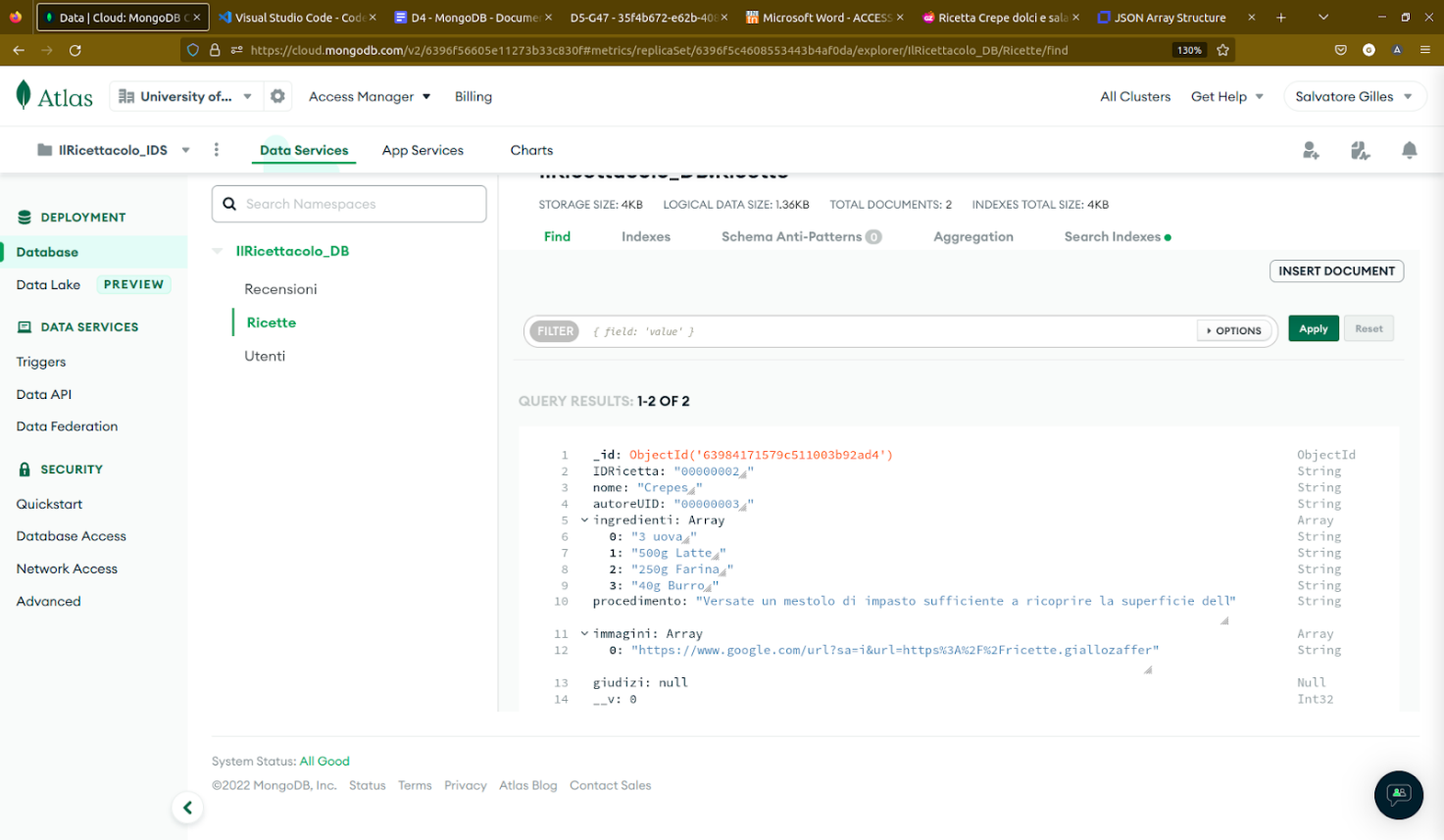
Per gestire i dati dell’applicativo web, è stato utilizzato MongoDB. Sono state definite tre collezioni: “Utenti”, “Ricette” e “Recensioni”, come rappresentato nella seguente immagine:



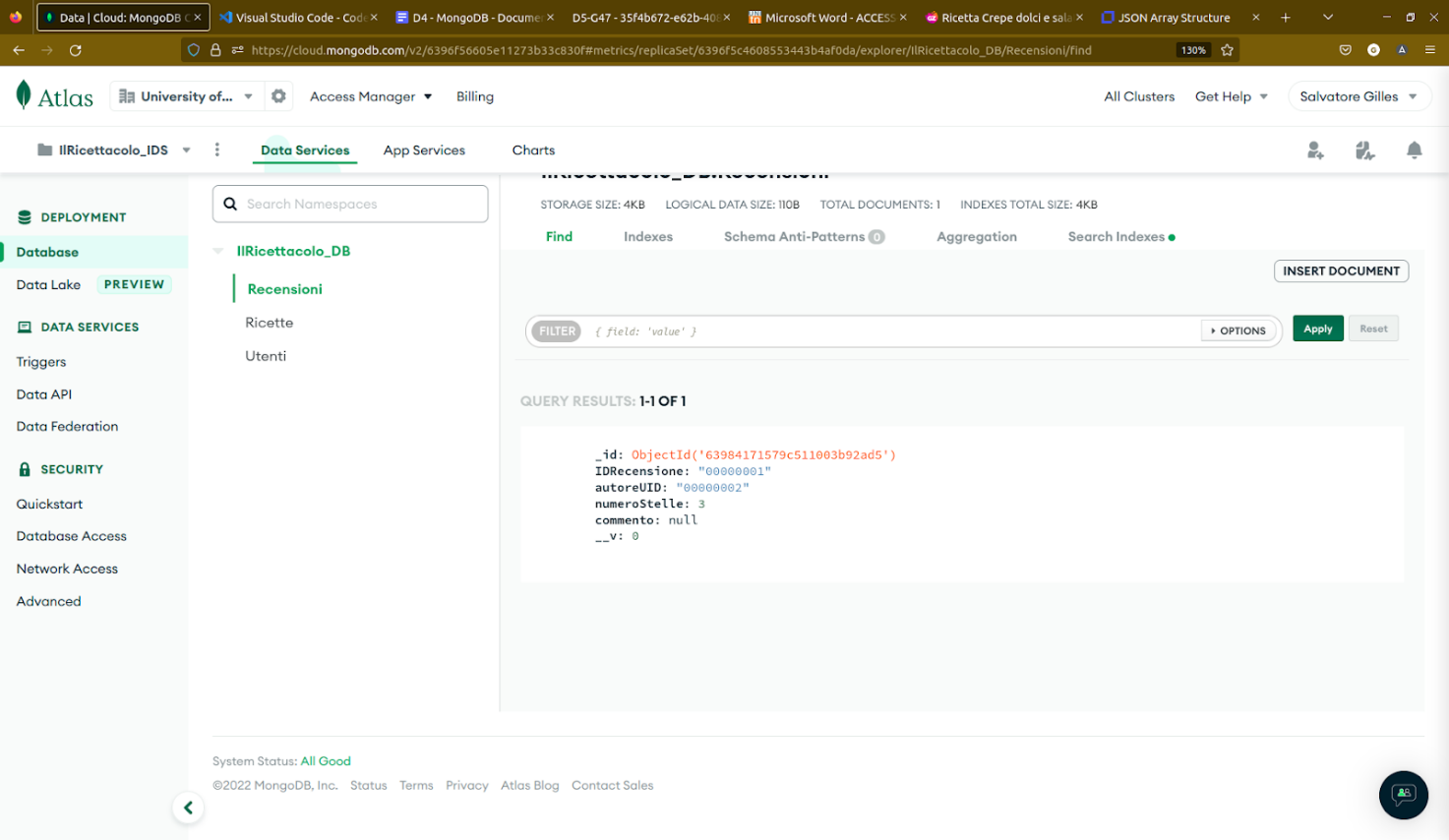
* Utenti rappresenta gli utenti che si sono registrati sul sito. Un esempio del dato utenti:



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



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



**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.

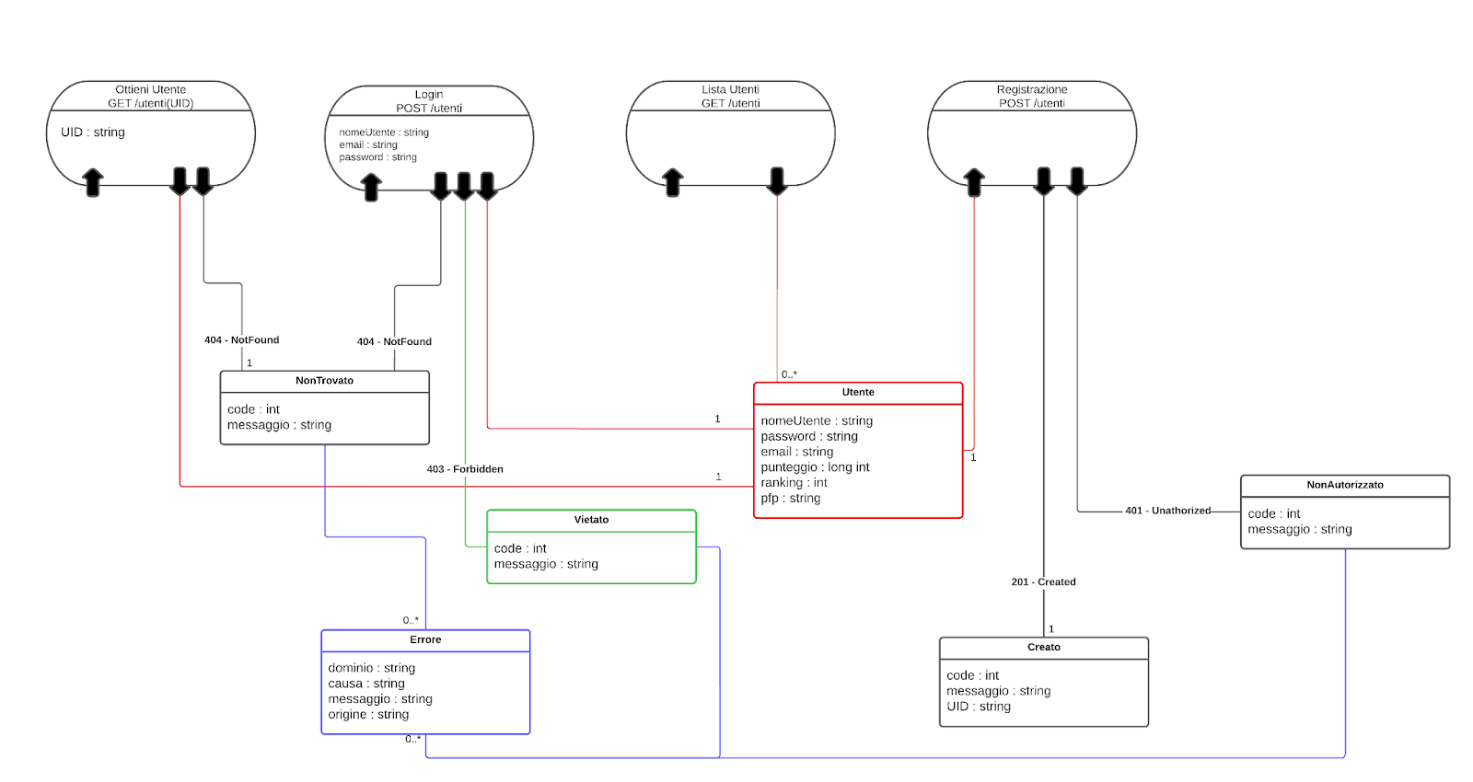
* La ricetta presenta i seguenti metodi HTTP:
  + scriviRicetta: un metodo POST che prende in input tutti i dati della ricetta (body) e crea la risorsa ricetta;
  + getRicetta: un metodo GET che prende in input un IDRicetta e ritorna la risorsa ricetta che corrisponde a quell’IDRicetta;
  + getRicetteAutore: 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**

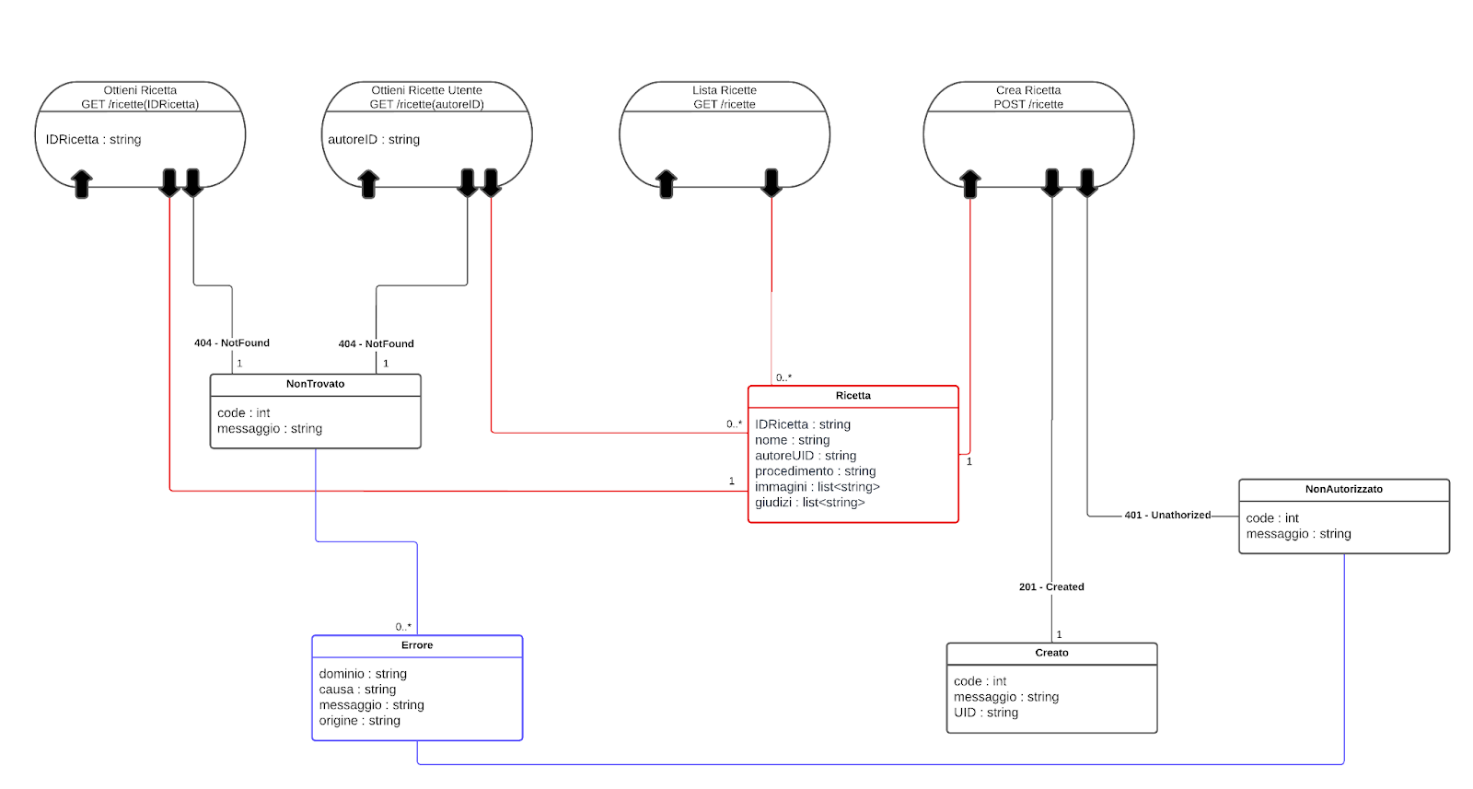
Tramite l’analisi delle risorse, è possibile definire i diagrammi delle API dell’utente, della ricetta e della recensione. In questi diagrammi, vengono descritti più nel dettaglio i metodi GET e POST.

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).

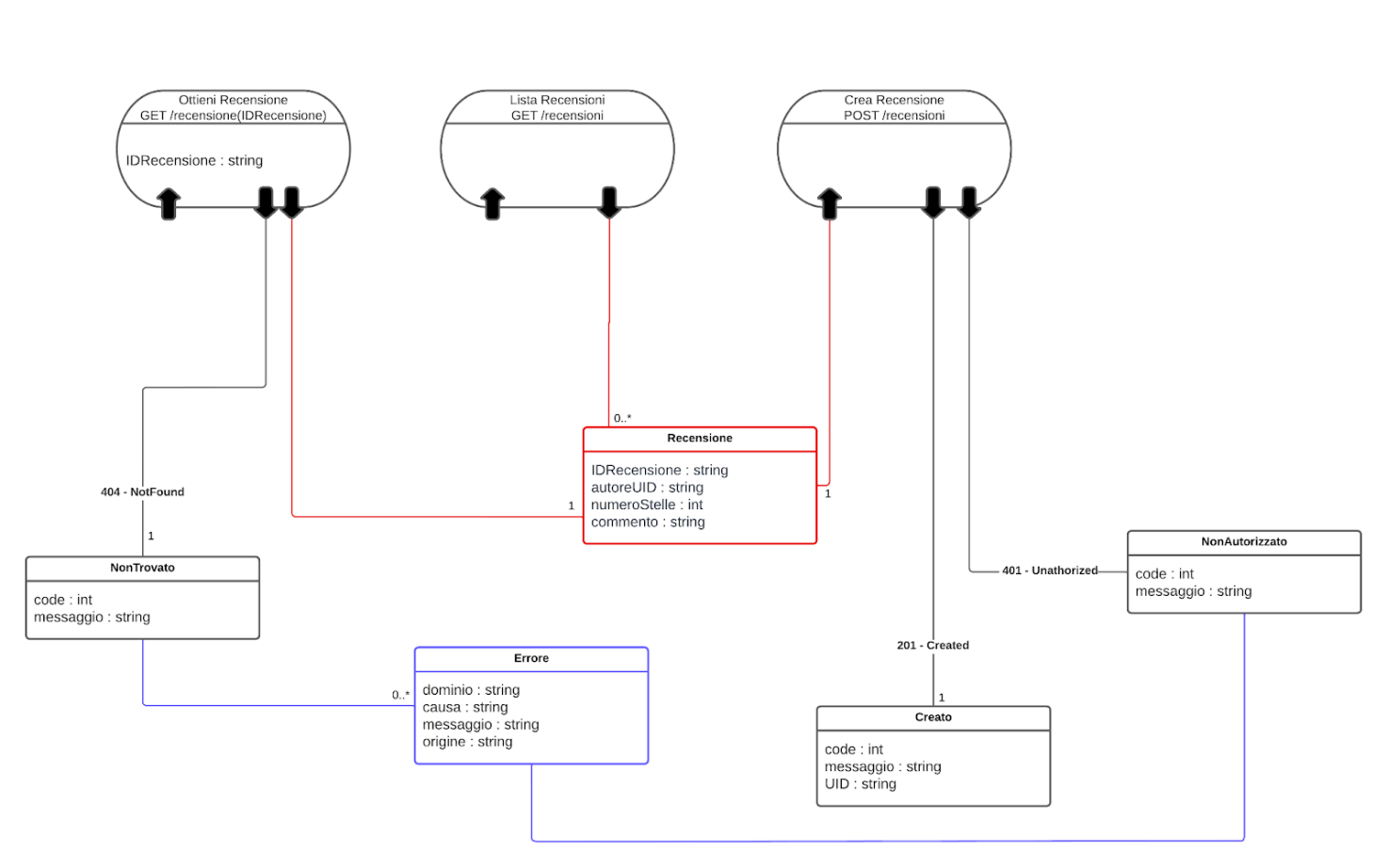
Infine, è possibile ottenere i dati degli utenti già esistenti con dei metodi GET. Questo può avvenire per tutti gli utenti tramite “Lista Utenti” o per un utente specifico tramite “Ottieni Utente” avendo come input l’UID corrispondente all’utente desiderato. Se quest’ultimo non corrisponde a nessun 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 traimte 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 di 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 traimte 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).

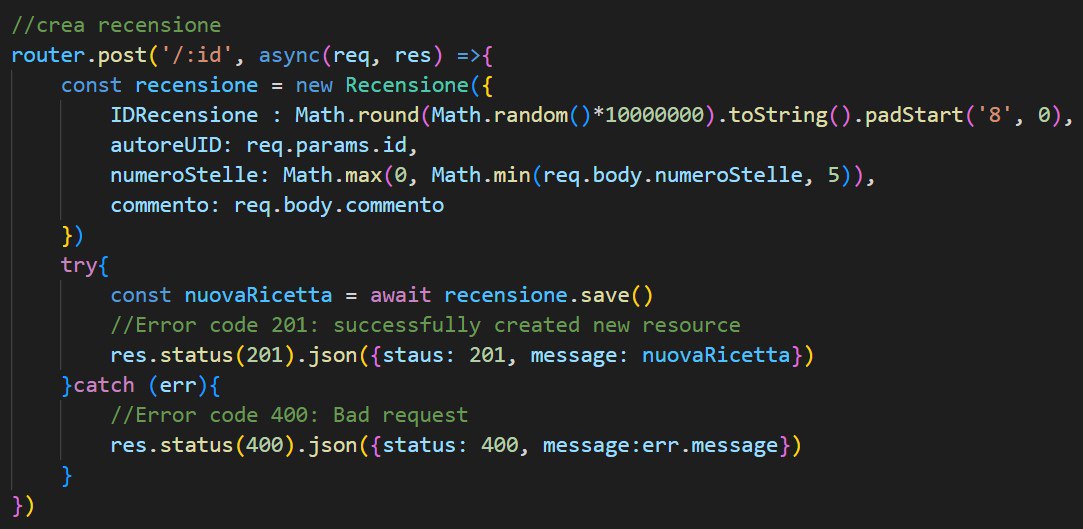
**2.5 Sviluppo API**

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.

**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.

**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.



**3.** **API documentation**

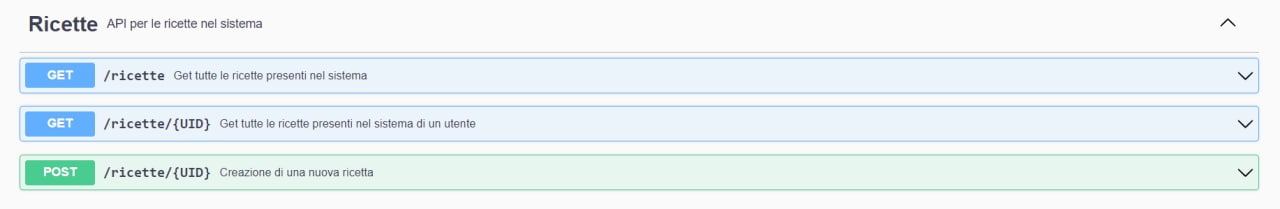
La documentazione delle API locali è stata inserita nel file “swagger.json”, così da renderla disponibile a chiunque veda il codice sorgente.

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**](http://localhost:3000/api-docs)

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



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.





* 1. **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.

****

****

**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>
  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/IlRicettacolo\_DB?retryWrites=true&w=majority

Installare le dipendenze con il comando:

npm install

Avviare il sito eseguendo:

npm run devStart

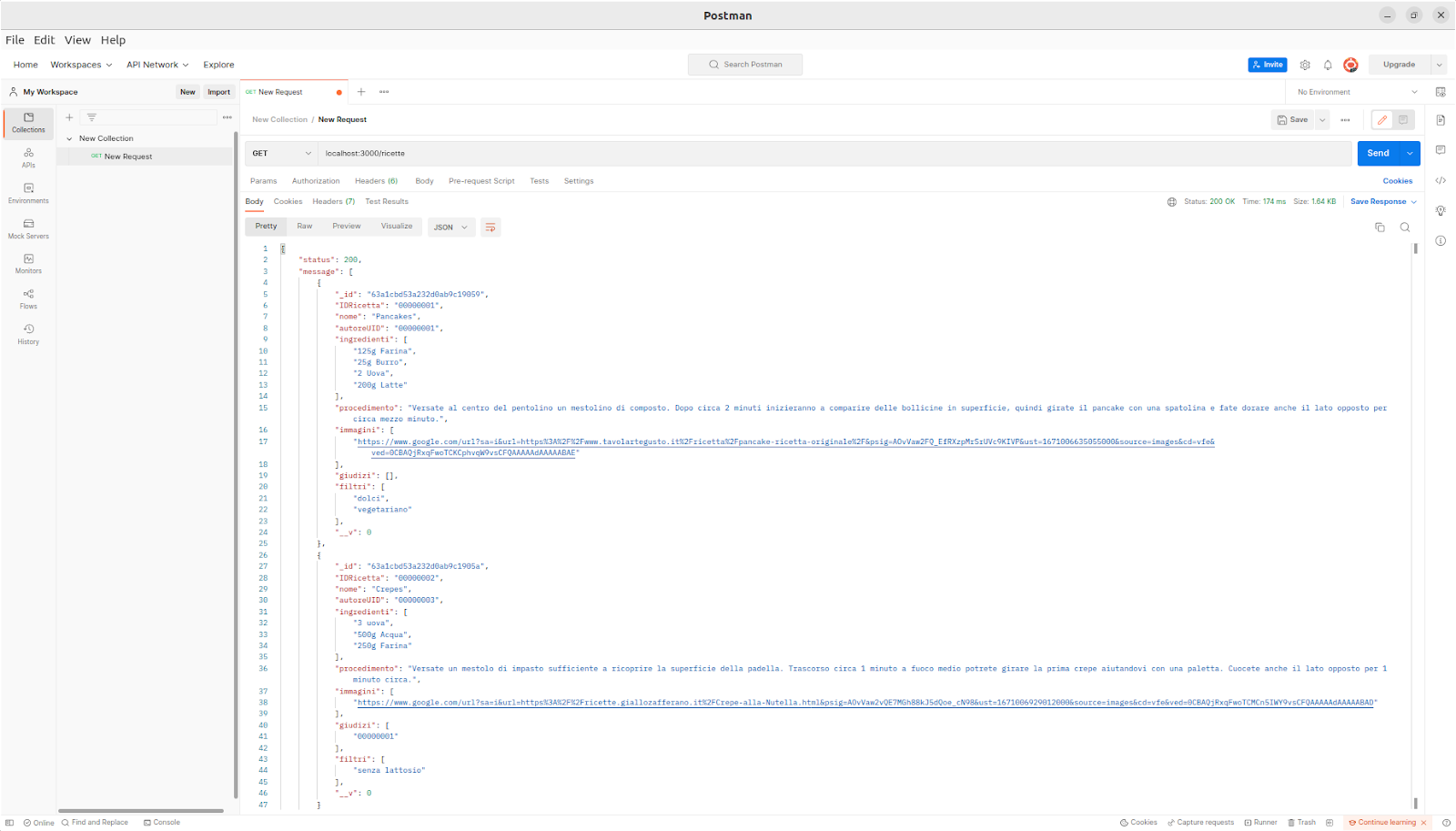
Collegarsi al sito tramite l'indirizzo su un browser supportato (vedere D1 e D2):

[localhost:3000](http://localhost:3000/)

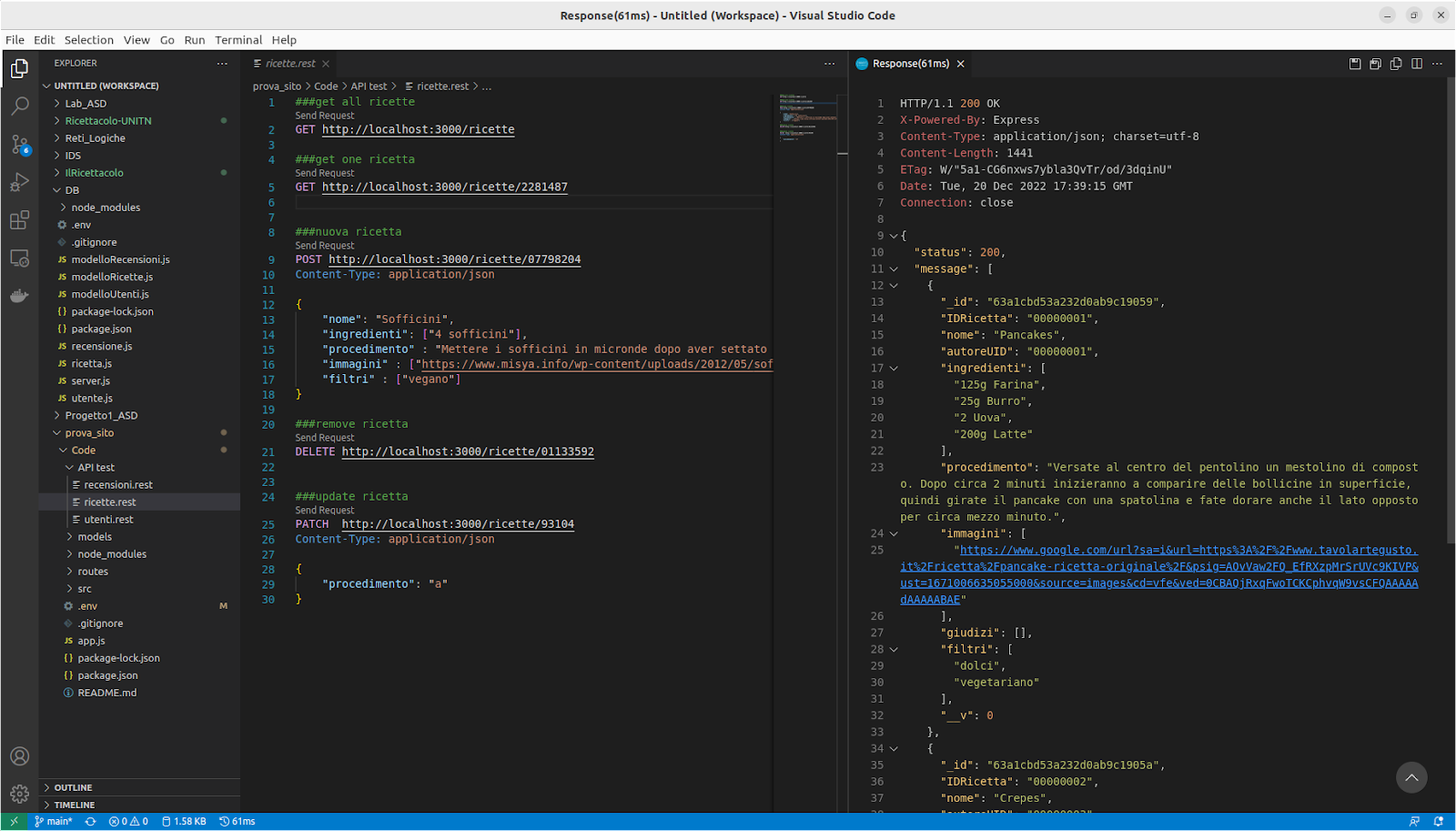
**6. Testing**

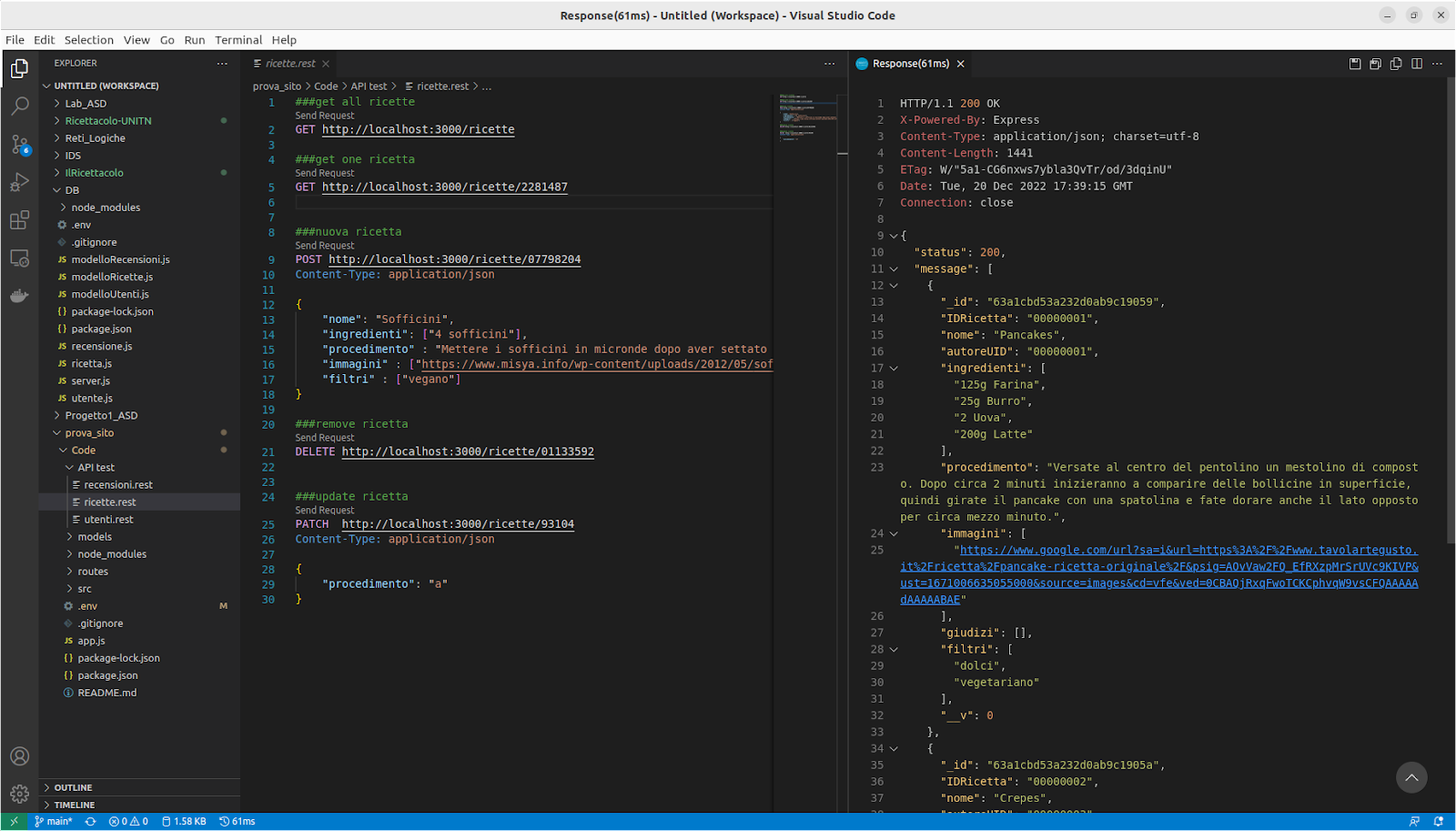
Le API dell’applicativo web sono state testate tramite l’estensione REST Client di VS code (<https://marketplace.visualstudio.com/items?itemName=humao.rest-client>) e Postman (<https://www.postman.com/>).

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

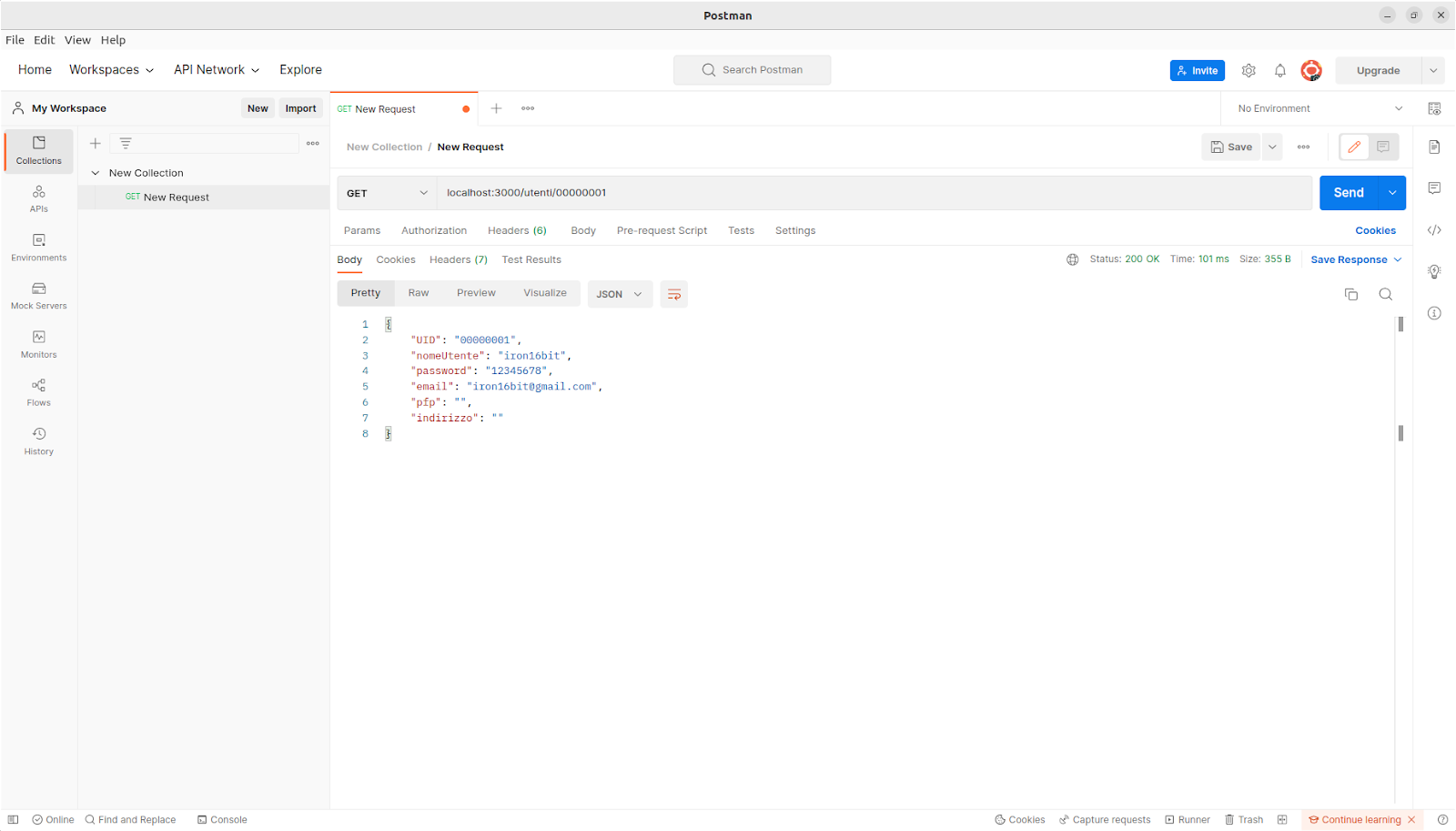
Il test da Postman ritorna due ricette, ovvero tutte quelle presenti in questo momento nel database.



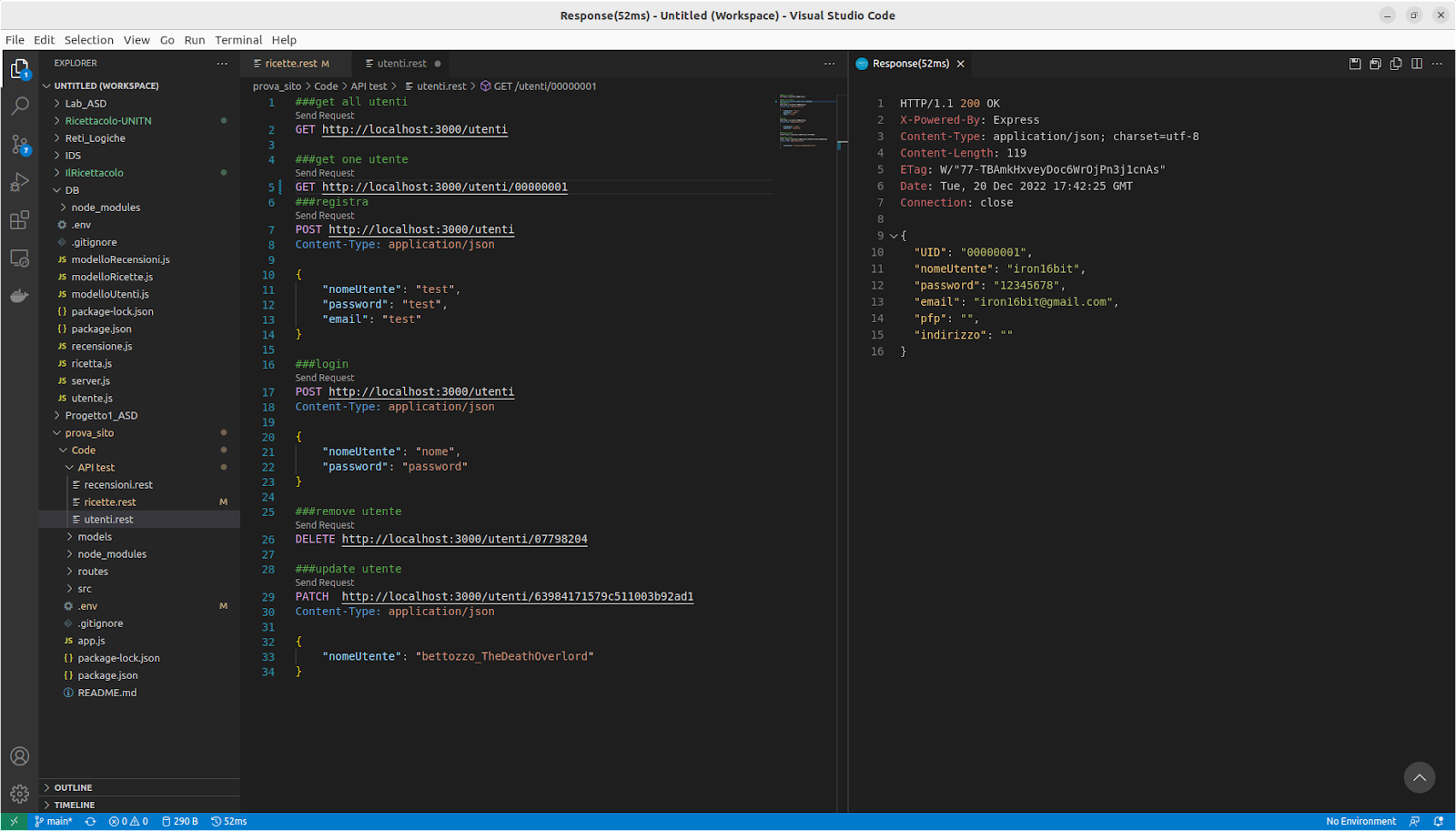


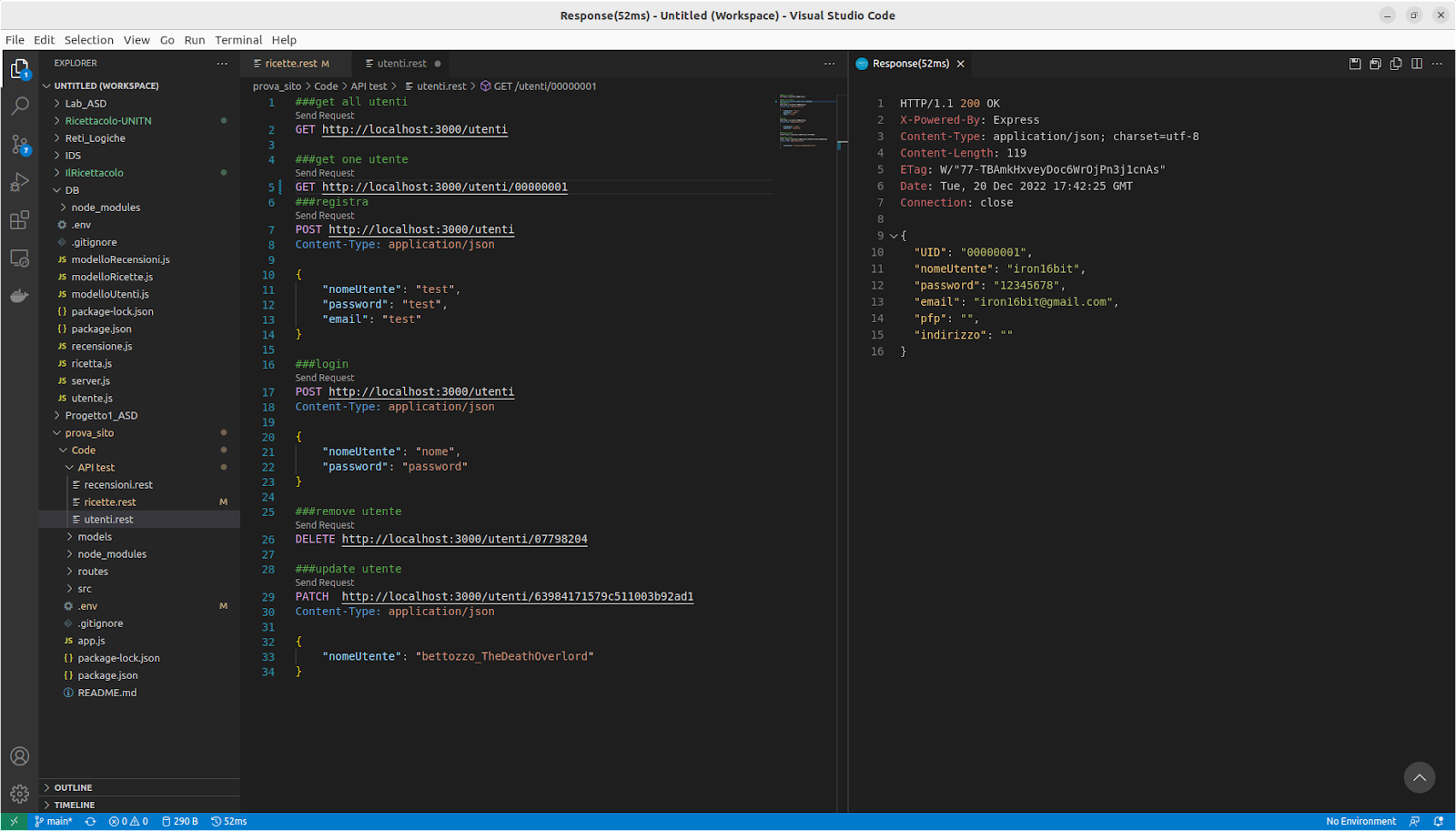
Allo stesso modo, anche il testing tramite REST Client ritorna due ricette.

GET utenti/UID



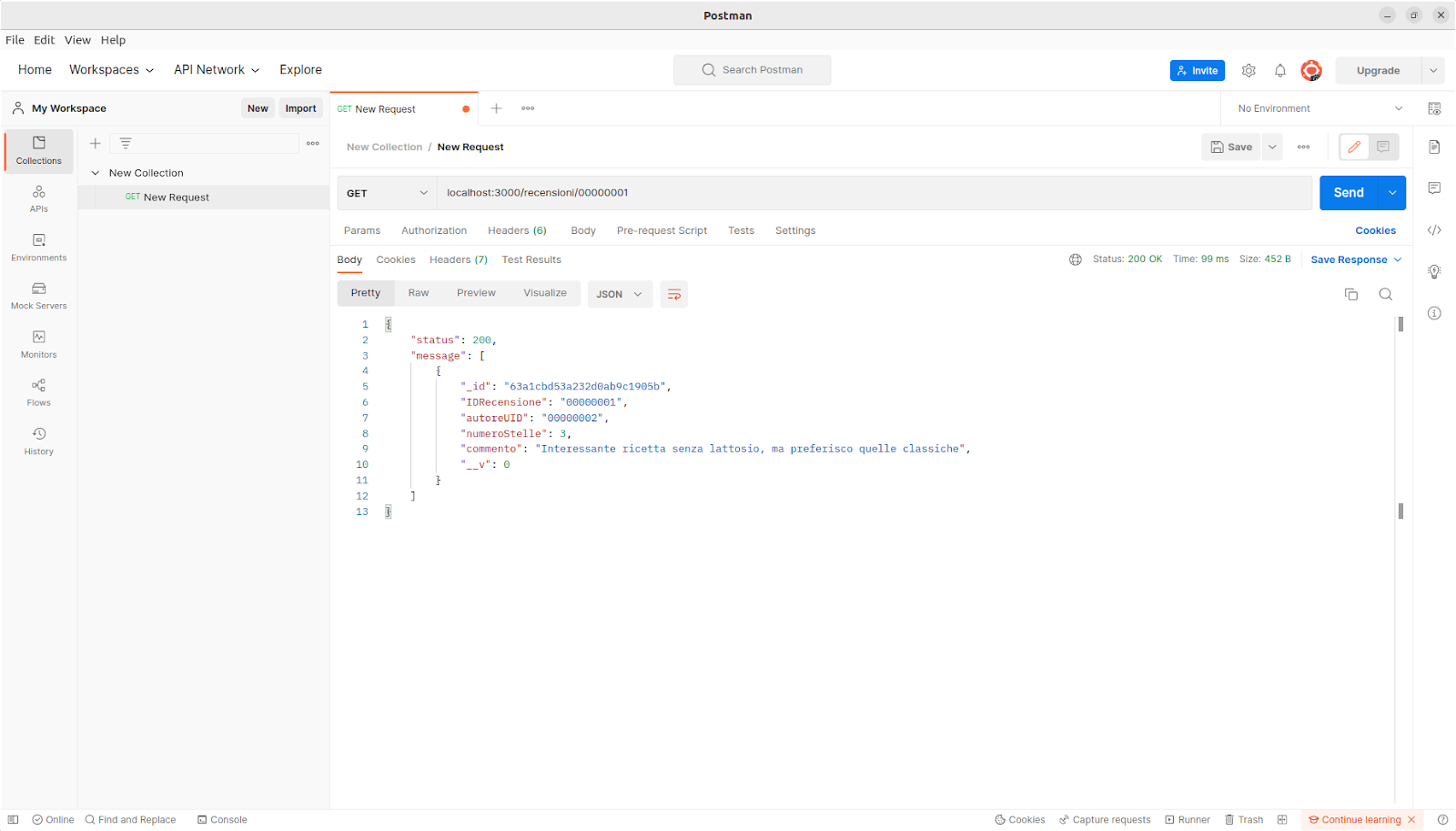
Richiedendo tramite Postman l’utente con UID “00000001”, ritorna correttamente l’utente corrispondente.



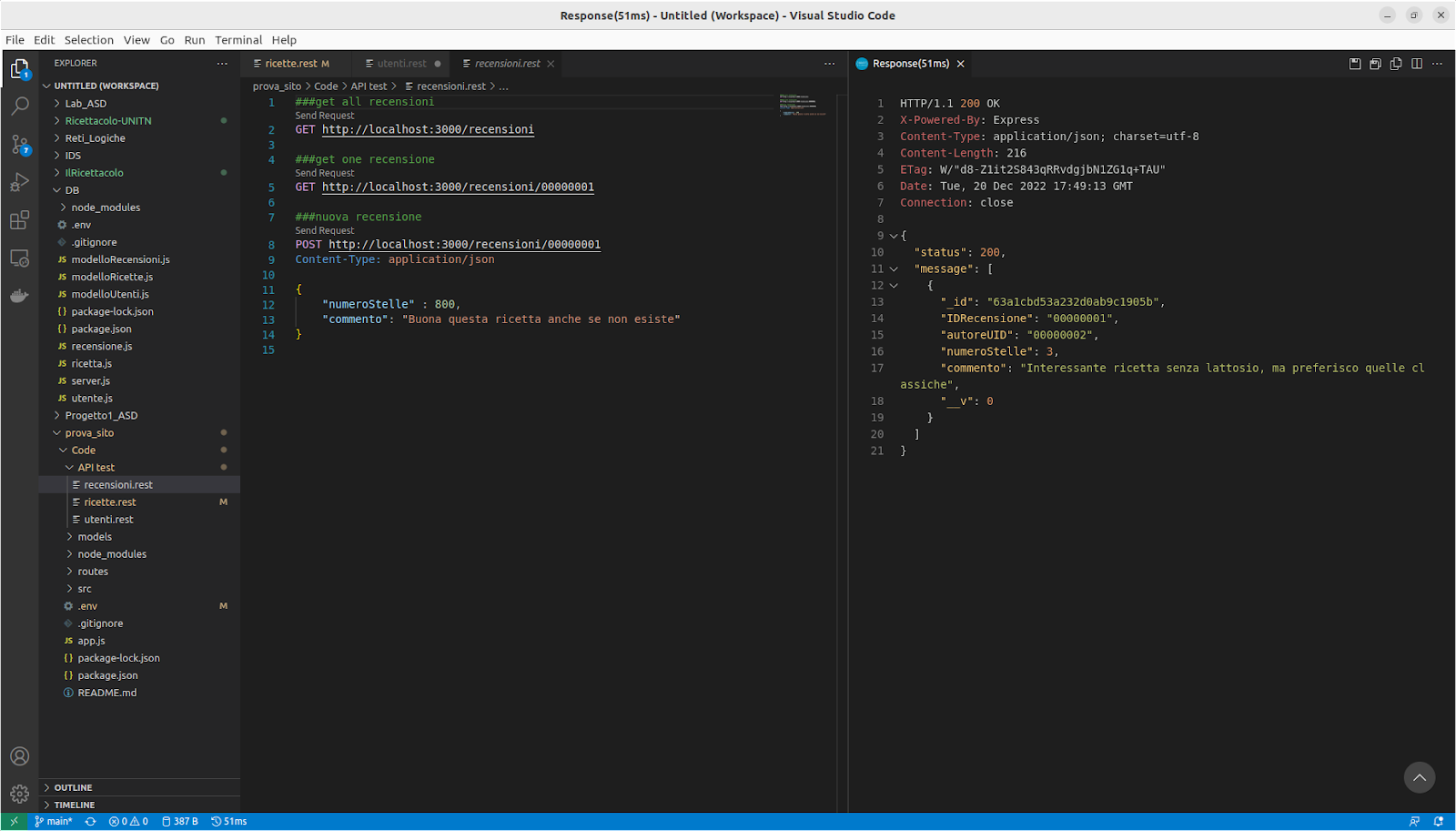


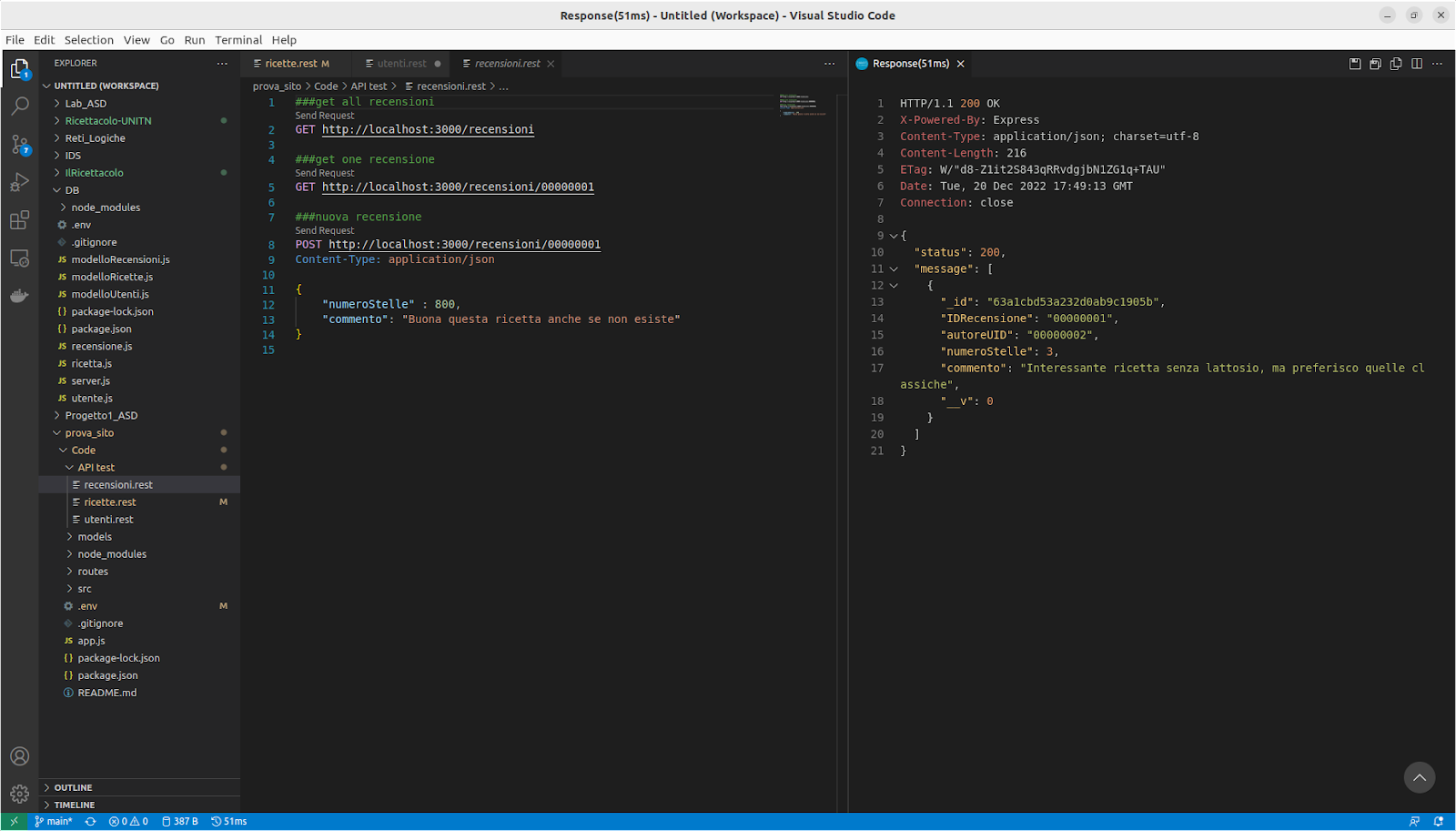
Anche il testing tramite REST Client ritorna lo stesso utente

GET recensioni/IDRecensione



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





Anche il testing tramite REST Client ritorna la stessa recensione.