**Relazione tecnica Web Application Redazione Culinaria (Sweet News)**

**Studente:** Gherardi Gianmattia  
**Matricola:** STA07049/L8  
**Progetto:** Redazione Culinaria (Sweet News)  
**Repository:** <https://github.com/ggherardi/sweetnews>  
**Indirizzo:** http://2.236.247.217:82/

1. **Tecnologie utilizzate**

Per la realizzazione del progetto sono state utilizzate le seguenti tecnologie:

* PHP 7
* DB MySQL (InnoDB)
* JavaScript ES 6
* jQuery 3.2.1
* BootStrap 4
* CSS3

Il gestionale è stato realizzato secondo il paradigma SPA (Single Page Application): non appoggiandosi ad alcun framework JavaScript esterno, è stato realizzato un meccanismo di gestione delle view ad hoc, utilizzando JavaScript e come uniche librerie esterne, jQuery (DataTable e UI) e BootStrap 4. L’accesso al DB è garantito tramite dei servizi realizzati in PHP 7, sotto forma di classi, che comunicano con altrettante classi JavaScript che fanno da proxy ed effettuano le chiamate REST.  
Il server utilizzato è un Raspberry 3 modello B+, sul quale risiedono il DB e il file system dal quale attinge il web server.

1. **Comunicazione con il database**

Alcune operazioni svolte dagli utenti implicheranno la persistenza di dati sul database MySQL. Questi dati verranno comunicati tramite delle chiamate REST a dei servizi PHP, che a loro volta andranno ad interrogare il database per restituire infine al client una response in formato JSON.

Da notare che non tutte le richieste API potranno essere effettuate se l’utente non ha effettuato l’accesso al sito: alcuni endpoint necessitano, infatti, dei Token di sicurezza staccati durante l’autenticazione. Per quanto concerne la preparazione delle query, sono stati utilizzati, per una maggior sicurezza atta a prevenire con maggiore facilità attacchi di tipo SQL Injection, dei Prepared Statement anziché delle query formattate con delle sprintf.

1. **Autenticazione e sicurezza**

All’accesso degli utenti, se l’autenticazione andrà a buon fine verrà creato un cookie http-only per mantenere la sessione di quell’utente attiva: questo cookie ha durata di 12 ore e conterrà i dati del contesto attuale più un token, criptati. La creazione del Token avviene utilizzando due diverse chiavi di cifratura e la stringa JSON contenente i dati. In questa maniera viene generato un Token che sarà indissolubilmente legato a queste informazioni, così da impedire tentativi di manomissione dei cookie e quindi delle chiamate ai servizi: infatti, alcuni endpoint (in particolare quelli che andranno a scrivere sul DB) necessitano della validazione del Token che verrà passato assieme al cookie criptato durante la chiamata; se la decriptazione non andasse a buon fine, la chiamata al servizio termina e viene restituito al client un messaggio di errore di autenticazione. Inoltre, per garantire una maggiore sicurezza, determinati endpoint verificheranno che la Delega dell’utente che ha effettuato la chiamata al sia sufficiente per poter consumare il servizio stesso. In caso contrario, verrà restituito un altro messaggio di accesso negato.

1. **Navigazione del sito**

Essendo una SPA (Single Page Application), esisterà una singola pagina (index.html). Tramite il menù di navigazione un Controller si occuperà di richiamare la view corrispondente alla voce selezionata, e caricare HTML e JavaScript relativi, sostituendo l’HTML del container principale. E’ stata inoltre prevista una logica di multi view: alcune pagine potranno switchare tra una view e l’altra senza bisogno ogni volta di ricostruire l’HTML della view stessa.

1. **Sistema di logging**

E’ stato creato un sistema di logging che permette di tracciare le operazioni eseguite dai servizi PHP. Verrà creato ogni giorno (eventualmente da suddividere ulteriormente) un file di log che scriverà le operazioni eseguite ed eventuali eccezioni. Ogni singola chiamata al servizio verrà dotata di un ID di correlazione (CorrelationID) che permetterà di tracciare in sequenza tutte le operazioni effettuate dal servizio stesso sul log, fino al termine dell’esecuzione.

1. **Configurazioni server**

Per l’ambiente di sviluppo è stato utilizzato IIS, mentre per quello di produzione, è stato utilizzato Apache. E’ stato necessario aggiungere mappatura del tipo MIME 7zip nel web.config.

1. **Analisi funzionale**

Al fine di realizzare un applicativo conforme alla esigenze del cliente, è stato necessario rivedere con il cliente stesso alcune delle richieste avanzate e dei dettagli relativi al funzionamento aziendale, così da poter risolvere ambiguità che avrebbero impedito una corretta redazione dello schema concettuale.

* “*Nella redazione lavorano da 1 a 5 redattori e 1 caporedattore – La redazione sta valutando la possibilità di mettere su un sito internet […] per favorire anche i visitatori […]..*”

**Domanda**: Nel caso un redattore avesse anche un’utenza come visitatore, potremmo prevedere la possibilità avere un’unica utenza centrale e tramite un sistema di deleghe consentire all’utente, al login, di scegliere con quale tipo di utenza loggarsi. Potrebbe essere una soluzione?

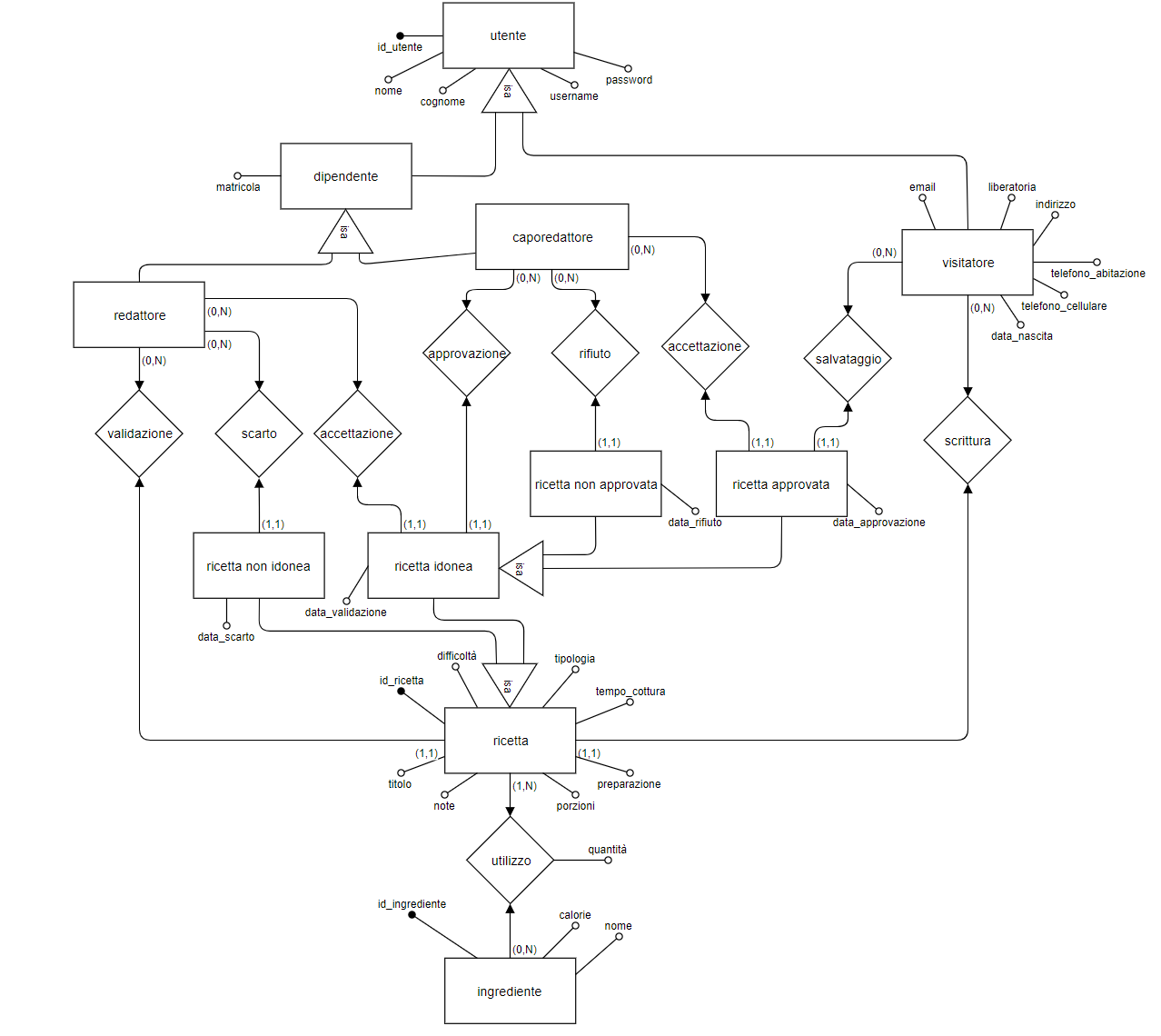
**Risposta**: Sì, purché il caporedattore abbia la possibilità di aggiungere o rimuovere deleghe ai visitatori, ma non di creare un’utenza da visitatore a redattori già esistenti.

* “*Ogni settimana i redattori verificano le ricette inviate dai lettori e scartano quelle ritenute non idonee, le idonee vengono sottoposte, alla fine del mese, al caporedattore per l’approvazione definitiva che comporterà la pubblicazione sulla rivista.*”.

**Domanda**: dobbiamo fare in modo che le ricette inviate per validazione vengano mostrate solamente dopo una settimana per i redattori, e quelle inviate in approvazione dopo un mese per i caporedattori?

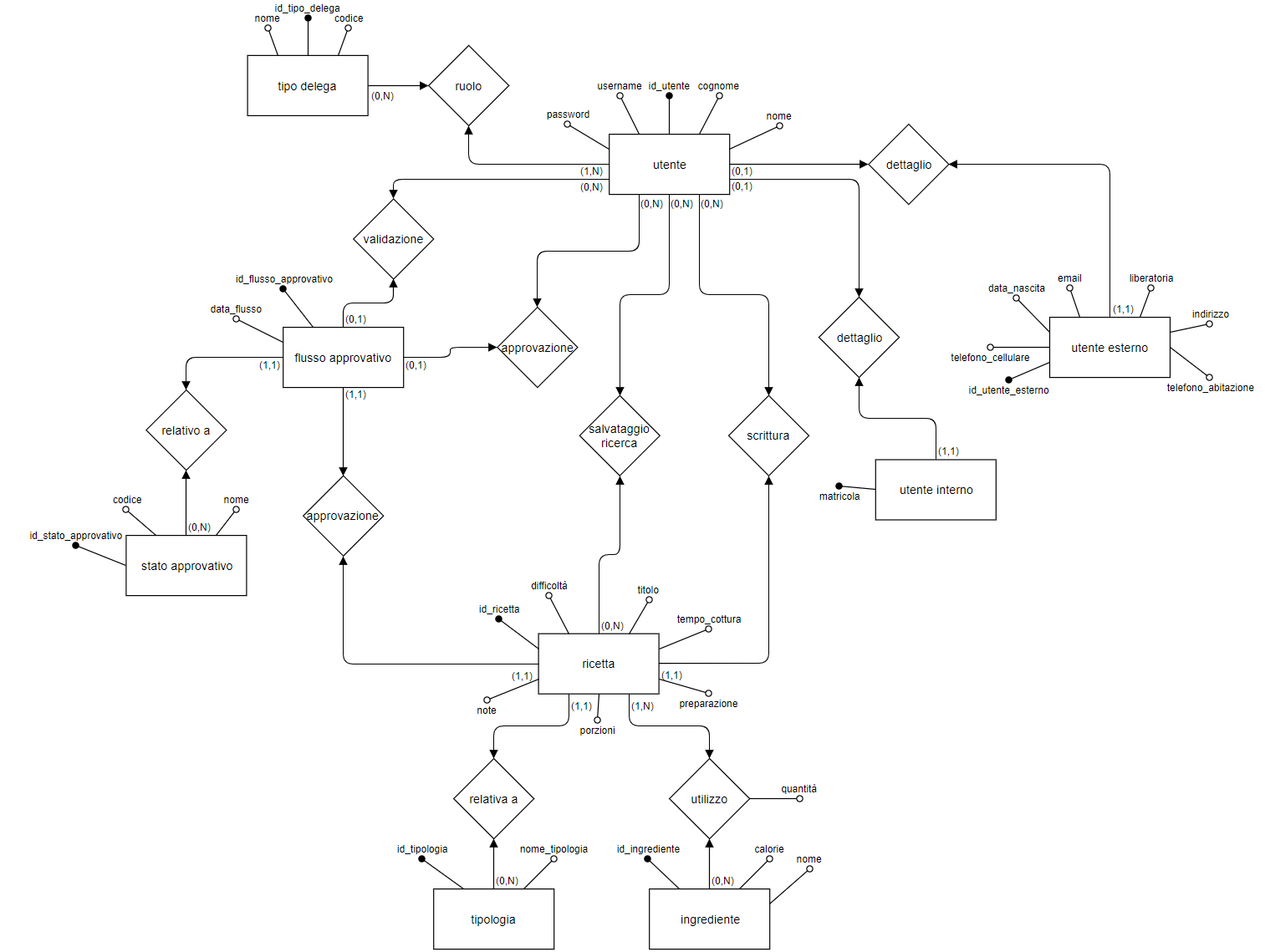
**Risposta**: per quanto riguarda i redattori, non è necessario applicare restrizione sulle ricette da prendere in validazione, mentre per i caporedattori dovranno essere mostrate unicamente quelle idonee già da un mese.

1. **Schema concettuale E-R**

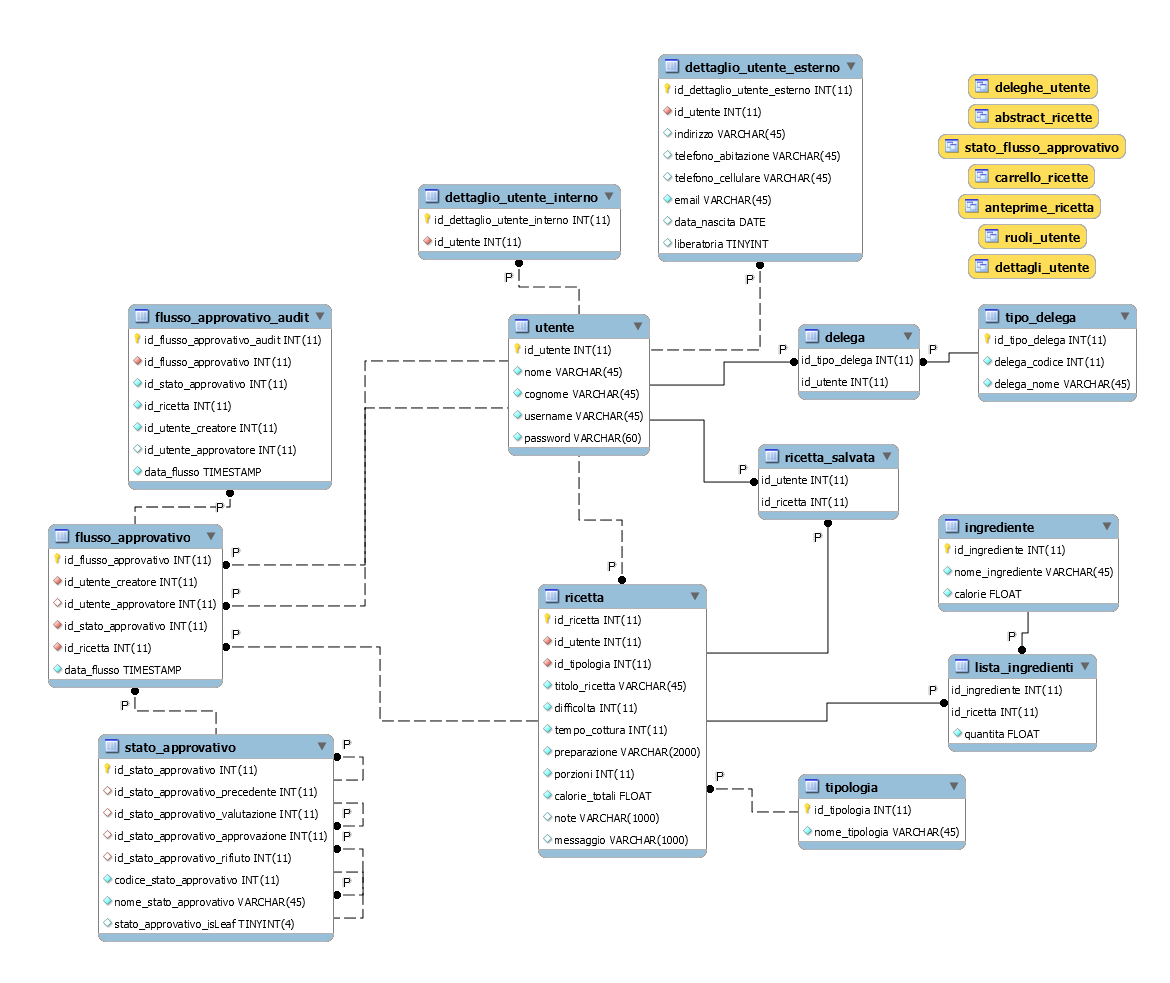


1. **Ristrutturazione schema E-R**

Al fine di poter tradurre lo schema concettuale in uno schema logico efficiente, è stato necessario effettuare una ristrutturazione dello schema E-R:

* Rimozione delle gerarchia ISA relative alle utenze, tramite un accorpamento nell’unica entità “Utente”.
* Per garantire la corretta suddivisione dei ruoli, viene creata una nuova entità “Tipologia Delega”; questa entità avrà una relazione N a N con l’entità Utente, poiché come specificato precedentemente, una singola utenza potrà avere diversi ruoli. Inoltre, questo garantirà un buon livello di scalabilità laddove venga riscontrata la necessità di aggiungere una nuova tipologia di ruoli.
* Vengono introdotte due nuove entità: “Utente Interno” e “Utente Esterno”, che avranno istanze unicamente se l’utenza avrà un ruolo di tipo interno o da visitatore.
* Rimozione delle gerarchie ISA relative alle ricette: viene eseguito un accorpamento mantenendo come unica entità “Ricetta”.
* Al fine di mantenere il corretto stato di flusso, vengono introdotte le nuova entità “Flusso Approvativo” e “Stato Approvativo”. “Flusso Approvativo” rappresenterà in ogni singolo istante lo stato del flusso di approvazione. In questa maniera ci sarà sempre e soltanto un’istanza di “Ricetta” con diversi livello di approvazione. Lo stato del flusso rappresenterà se la ricetta è idonea, non idonea, in validazione, ecc.
* Trasformazione dell’attributo “tipologia” dell’entità “Ricetta” in una nuova entità “Tipologia”. In modo da garantire un bacino chiuso di tipologie dalle quali attingere durante la creazione di una nuova ricetta.Si ottiene, dunque, il seguente schema E-R ristrutturato:

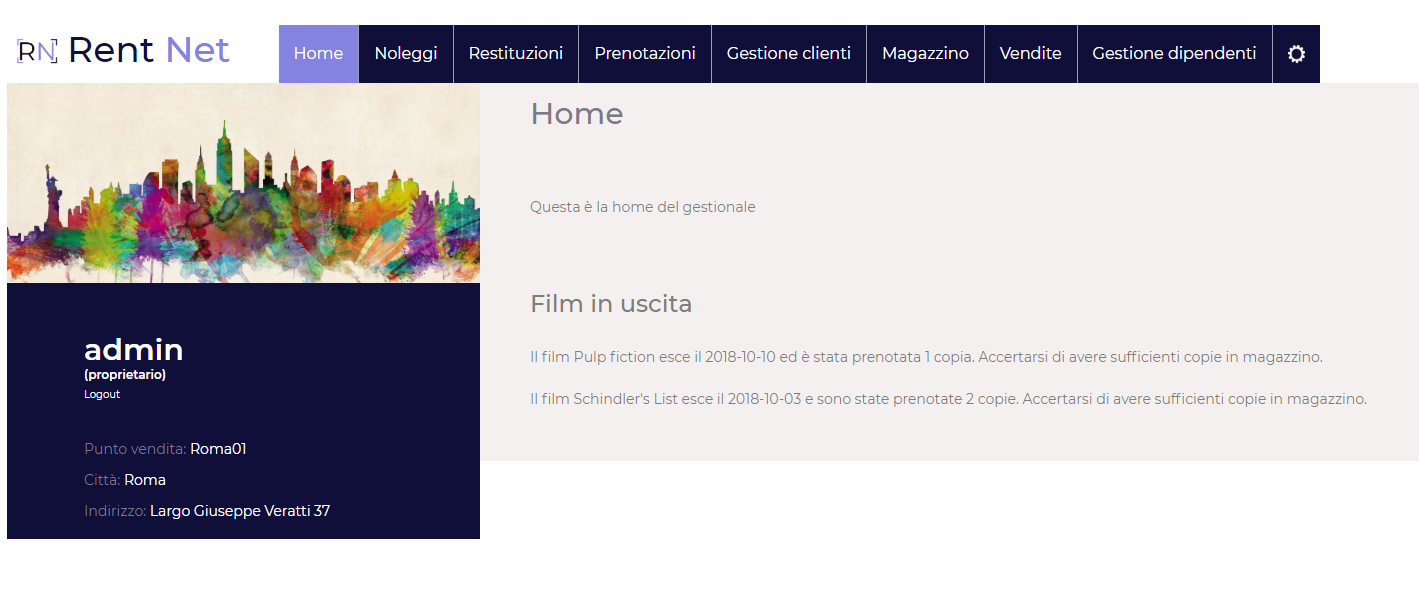
1. **Schema logico**

Dopo aver redatto lo schema concettuale e averlo ristrutturato, si concretizza lo schema logico, che andrà a formare le tabelle vere e proprie utilizzate dal database. Si noti che viene introdotta una nuova tabella, chiamata “flusos\_approvativo\_audit”. Questa tabella servirà a mantenere uno storico di tutti i cambiamenti di stato relativi ai flussi di approvazione, così da poter ricostruire la storia approvativa di ogni singola ricetta. Qui di seguito lo schema:

Vengono inoltre generati i seguenti trigger:

* Tabella “ricetta”: alla creazione di una nuova ricetta, viene generata automaticamente una tupla nella tabella “flusso\_approvativo” con uno stato pari a “bozza”.
* Tabella “flusso\_approvativo”: alla creazione e all’update di un flusso approvativo, viene generata una nuova tupla all’interno di “flusso\_approvativo\_audit”, al fine di storicizzare il cambiamento di stato.

Viene generata inoltre una stored procedure “deleteRecipe” per la cancellazione di una ricetta da parte di un utente. La stored procedure potrà essere richiamata unicamente dall’utente che ha creato la ricetta e quando la ricetta è ancora in fase di bozza.

1. **Dettaglio funzionalità**
2. **Homepage**