RELAZIONE PROGETTO
DATA BASE
RICETTARIO

TRUPIA LUDOVICO
X81000042
04/04/2017

OBIETTIVO DEL PROGETTO

L'obiettivo di questo progetto è la realizzazione di un'interfaccia che permetta la gestione di un database contenente ricette per uso culinario.

Questo progetto è indirizzato a persone fuorisede che non mostrano particolari abilità culinarie. Grazie all'interfaccia web sarà possibile registrarsi al sito, visualizzare ricette e persino aggiungerne di nuove.

INTERFACCIA

L'interfaccia è stata realizzata principalmente per essere ***responsive*** alla dimensione del dispositivo di visualizzazione.

All'avvio del sito web l'utente si troverà davanti un'interfaccia semplice e di facile intuizione.

Tramite la ***barra di navigazione*** sarà possibile accedere alle varie pagine presenti nel sistema dove l'utente potrà effettuare l'accesso e registrarsi, qualora visitasse il sito per la prima volta.

Anche senza registrazione è possibile poter ***visualizzare*** una lista contenente le ultime ricette inserite. In fondo a questa sezione sarà presente un tasto per ***l'aggiunta*** di nuove ricette *(soltanto gli utenti registrati potranno usufruire di questa funzione)*.

Sempre nella barra di navigazione è presente un link che permetterà di effettuare una ***ricerca*** delle ricette presenti nel database secondo i seguenti criteri:

- **Ricerca per nome**. L'utente dovrà inserire una stringa all'interno della casella di testo e avrà come risultato tutte le ricette il cui nome presenta all'interno quella stringa inserita.
- **Ricerca per categoria e tipo di piatto**. L'utente selezionerà, tramite un elenco a tendina, una categoria o una sottocategoria *(che indica il tipo di pietanza)* per trovarsi davanti l'elenco delle ricette appartenenti a quella categoria o tipologia. *(es. Categoria = Primo, l'utente visualizzerà tutti i Primi)*.
- **Ricerca per difficoltà**. Sempre come al punto precedente l'utente dovrà scegliere la difficoltà *(valore da 1 a 5)*.
- **Ricerca per tempo di preparazione**. Questo tempo, espresso in minuti, sarà inteso come tempo massimo impiegato per una ricetta, quindi visualizzeremo tutte le ricette con tempo di preparazione inferiore o uguale a quello inserito.

Nel momento in cui si voglia ***cancellare*** una ricetta inserita si dovrà, per questioni di sicurezza, contattare direttamente l'amministratore.

L'amministratore avrà pieno accesso all'interfaccia admin creata direttamente dal framework, in cui potrà visualizzare completamente tutte le ricette inserite e decidere, anche su richiesta degli utenti, di toglierne alcune.

Inoltre tutta questa interfaccia è contenuta all'interno dei nostri ***template*** *(file .html)* che prelevano i dati necessari dall'utente per passarli successivamente alle ***View*** di back-end, tramite il metodo *POST*.

PROGETTAZIONE

Per la realizzazione di questo progetto è stato utilizzato ***SQLite3*** come DataBase relazionale e il framework ***Django***.

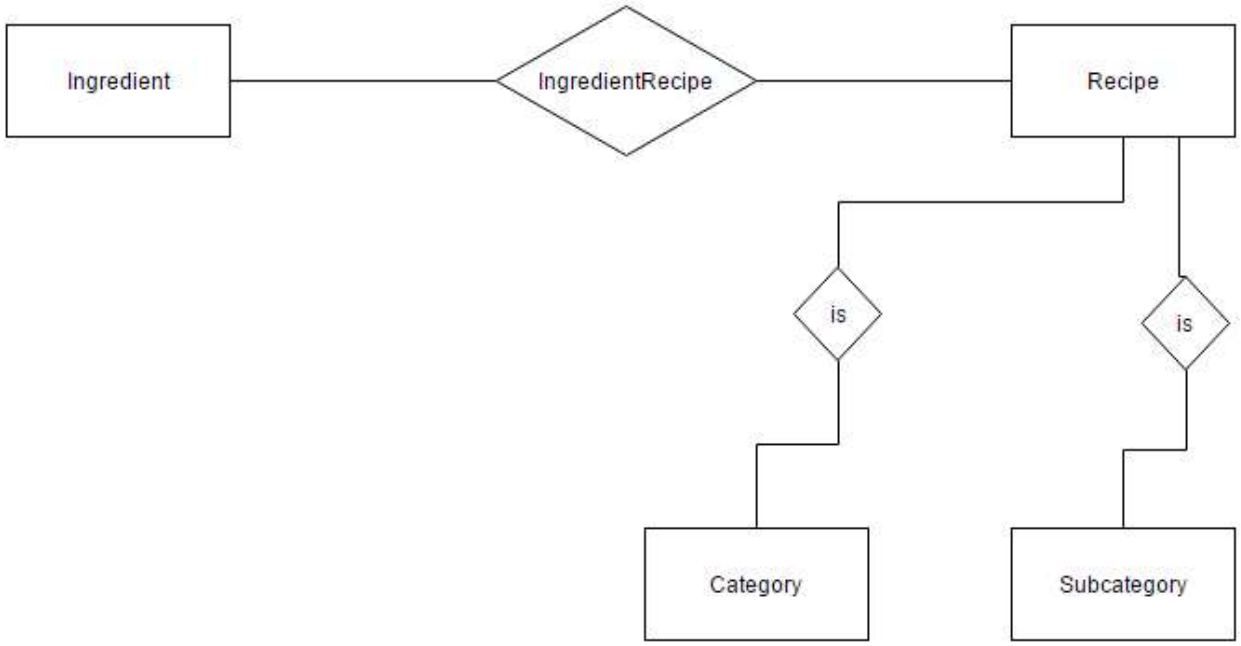
È stato usato SQLite3 per via della sua natura open source e per la sua facile gestione. In pratica è una libreria software che implementa un DBMS SQL di tipo ACID e quindi presenta transazioni atomiche, consistenti, isolate e durabili anche in caso di crash del sistema. È stato usato inoltre grazie alla sua facile integrazione all'interno del framework Django.

Django è un web framework che facilita lo sviluppo di applicazioni web ed è scritto in Python. Questo framework segue il pattern architetturale MVC *(Model-View-Controller)*. In questo pattern abbiamo tre componenti:

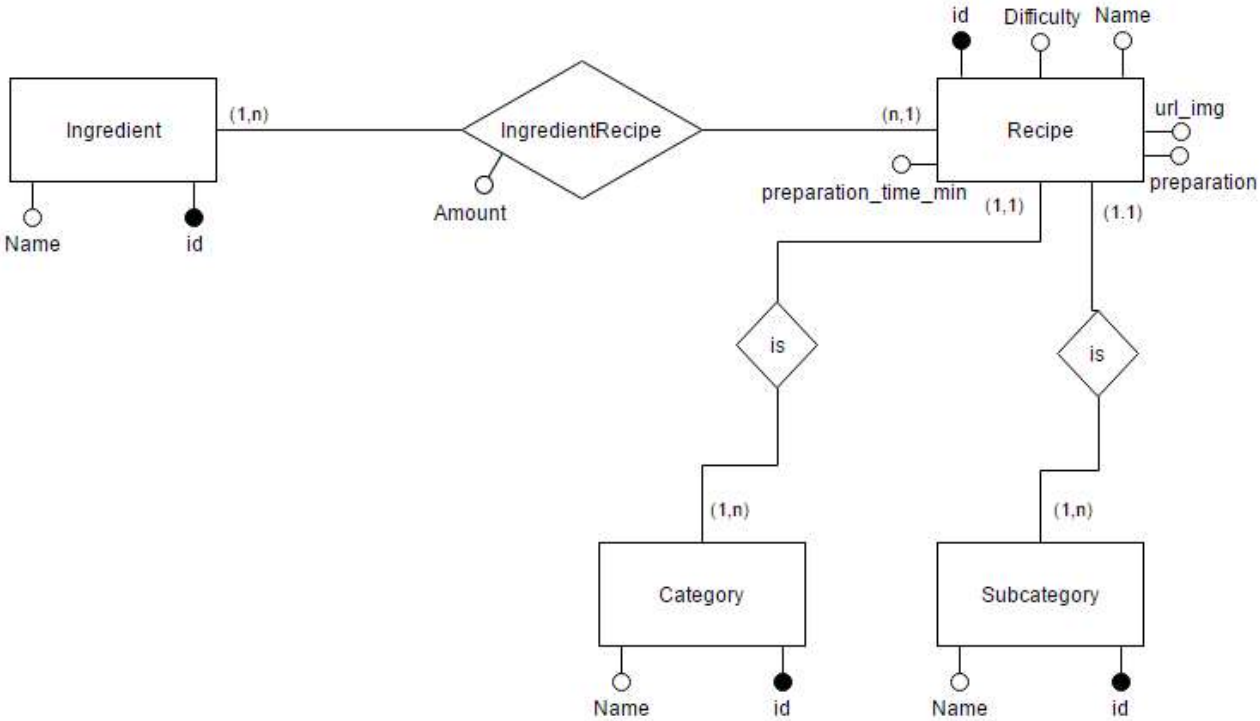
1. il ***modello (Model)***, contiene i metodi di accesso ai dati;
2. le ***viste (View)***, si occupano di gestire l'interazione tra l'utente e tutta la struttura sottostante, permettendone la visualizzazione;
3. il ***controller (Controller)***, riceve i comandi dell'utente attraverso le View, che vengono elaborate portando ad un risultato capace di modificare lo stato delle View stesse, oppure viene destinato ad essere usato nel Model.

Come strategia di sviluppo viene usata la ***strategia mista*** che combina i vantaggi della strategia top-down, per quanto riguarda lo ***schema scheletro*** creato, con quelli della strategia bottom-up, per quanto riguarda la suddivisione dei requisiti in componenti separate. Lo schema scheletrico contiene, a livello astratto, i concetti principali dell'applicazione e fornisce una visione unitaria dell'intero progetto.

SCHEMA SCHELETRO DELL'APPLICAZIONE



SCHEMA COMPLETO DELL'APPLICAZIONE



BACK-END

Nella parte di ***back-end*** si utilizzano tutti i dati prelevati dalla parte di ***front-end***, ovvero dai vari template presenti.

Tramite Django facciamo uso delle *View*, in genere una per pagina, in cui elaboriamo i dati prelevati dai vari *Form*. Un esempio di *View* utilizzata è quella per il *login* di un utente:

ESEMPIO DI VIEW PER IL LOGIN

```
def login_view(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            u = form.cleaned_data['username']
            p = form.cleaned_data['password']
            user = authenticate(username = u, password = p)
            if user is not None:
                if user.is_active:
                    login(request, user)
                    return HttpResponseRedirect('/profile/'+u)
                else:
                    print("The account has been disabled!")
            else:
                print("The username and password were incorrect.")
                return render(request, 'login.html', {'flag': True, 'form': form})
    else:
        form = LoginForm()
        return render(request, 'login.html',
                                {'form': form})
```

In questa View prendiamo i dati dai campi “username” e “password” e verifichiamo, tramite la funzione authenticate() se l’utente può effettuare il login o meno. Nel caso in cui verranno inseriti dati errati si avrà un messaggio di errore. Se i dati saranno corretti si verrà reindirizzati al profilo dell’utente collegato.

Tra le altre funzioni presenti nel sistema abbiamo:

- *Login() / Logout()*, rispettivamente permettono l’accesso al sistema come utente;
- *Register()*, permetterà ad un nuovo utente di registrarsi tramite il Form corrispondente;
- *Search()*, ha la funzione di poter effettuare una ricerca all’interno del database, come spiegato nelle pagini precedenti.
- *addNewRecipe()*, consentirà, ad un utente correttamente registrato, di poter aggiungere una nuova ricetta al database. Si potranno inserire i vari ingredienti della ricetta, il titolo, la difficoltà, la categoria e corrispettiva sottocategoria (*tipologia di piatto)*, tempo di preparazione espresso in minuti, le istruzioni per la preparazione del piatto e un’eventuale immagine da mostrare nel *cookbook (qualora non si inserisca un’immagine personalizzata il sistema provvederà ad inserirne una di default)*.

- *showRecipe()*, permette la visualizzazione dell'intera ricetta presa in considerazione. Possiamo vederlo come il *profilo* della stessa.

Un esempio di prelievo di un record dal *DB* è il seguente (preso dalla funzione di *registrazione* di un nuovo utente):

```
try:
    username1 = User.objects.get(username=username)
    flagUs = True
except User.DoesNotExist:
    flagUs = False

try:
    email1 = User.objects.get(email=email)
    flagEmail = True
except User.DoesNotExist:
    flagEmail = False

if flagEmail == False and flagUs == False:
    user = User.objects.create_user(username=username,
                password=form.cleaned_data['password1'],email=email)
    return HttpResponseRedirect('/login')
```

Nella variabile *username1* e *email1* avremo il risultato della query che segue

User.objects.get(username = username)

User.objects.get(email = email)

La prima riga non fa altro che prelevare dalla tabella *User*, presente nel Database, il cui *username* corrisponde alla variabile *username*. Stessa identica cosa per la seconda riga. Entrambe, quindi, generano delle *SELECT* al Database ritornando il record voluto. Se questo record esiste allora non possiamo portare a termine la registrazione dell'utente con quei valori di email e username *(sarà generato un errore mostrato nell'interfaccia web)*.

La riga *User.objects.create_user(….)* aggiunge *(comando INSERT INTO Table di SQL)* un nuovo record alla tabella *User* con gli attributi *username, email* e *password* corretti.

Alla fine verremo reindirizzati alla pagina di *login*.

DETTAGLI ENTITÀ

Nome	Relazione	Descrizione	Attributi
Ricetta	Ingrediente, Categoria, Sottocategoria	Rappresenta le ricette aggiunte dagli utenti.	**Id_Ricetta**, Nome, Categoria, Sottocategoria, Difficoltà, Tempo di preparazione, Preparazione, Url_immagine
Ingrediente	Ricetta	Rappresenta i nomi degli ingredienti che saranno utilizzati all'interno delle varie ricette.	**Id_Ingrediente**, Nome
Categoria	Ricetta	Rappresenta tutte le categorie di piatti presenti nel sistema.	**Id_Categoria**, Nome
Sottocategoria	Ricetta	Rappresenta tutte le tipologie di piatti presenti nel sistema.	**Id_Sottocategoria**, Nome
IngredienteRicetta	Ingrediente, Ricetta	Rappresenta la relazione tra l'entità Ingrediente e l'entità Ricetta. Ovviamente più ricette possono avere più ingredienti e vice versa.	**Id_Ingrediente, Id_Ricetta**, Quantità_Ingrediente