

MEMELANDIA

Traccia

Il progetto che porto per l'esame del corso "Tecnologie Web" di Informatica rappresenta un social network per gestire, caricare e condividere meme, oltre ad avere una sua piattaforma di messaggistica.

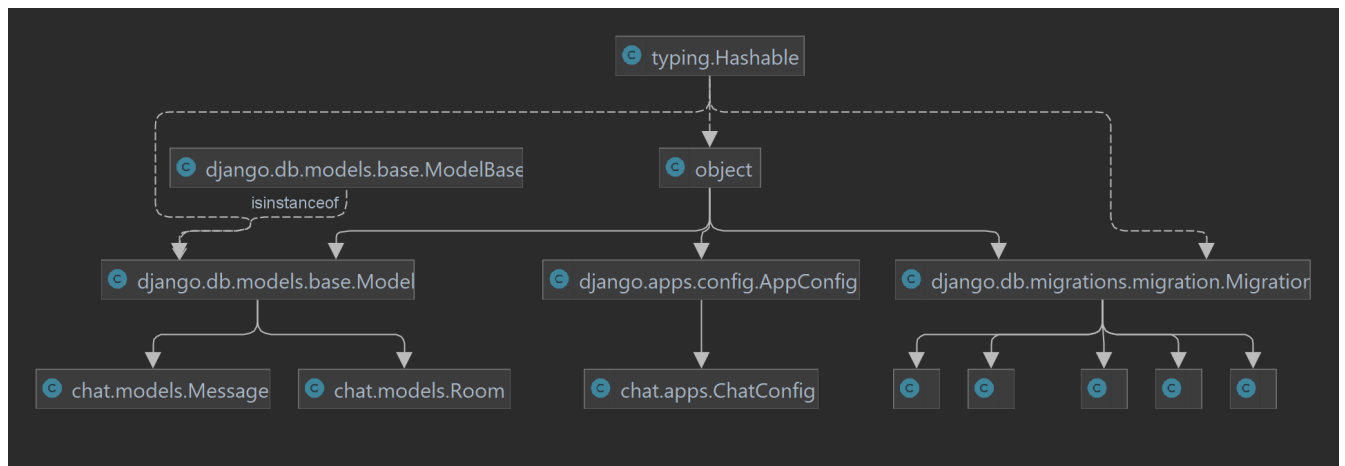
La traccia inviata e pre-approvata dal Professore Nicola Capodieci e dalla Professoressa Claudia Canali è la seguente:

```
Social Network per i meme "MEMELANDIA"
Un sito di social network tra utenti per la condivisione di meme.
Le varie funzioni:
- Iscrizione al sito e gestione di un profilo personale, inclusivo
di foto, una bio e indicazione di una lista dei generi di meme
preferiti
- L'utente iscritto potrà scegliere se rendere il suo
profilo pubblico o privato, nel caso fosse privato gli altri potranno
vedere i dettagli solo se amici di questo
- Creazione di relazioni simmetriche: per essere amici, un utente
A deve fare esplicita richiesta a B di essere accettato tra i suoi
contatti e B deve rispondere positivamente
- Ricerca di utente in base a diverse caratteristiche
- Possibilità di condividere i propri meme ed inserire dei tag per
delineare uno o più generi
- Possibilità di inoltrare meme altrui indicando però l'autore
originale
- Possibilità di invio/scambio di messaggi privati tra utenti
connessi tra di loro
- Per ogni meme è possibile mettere Like, Dislike e commentare
- Un utente non loggato potrà vedere i meme sui profili pubblici,
per ogni altro tipo di attività sarà necessaria l'autenticazione
- La home page conterrà vari meme tra i più votati dei vari amici
dell'utente loggato; nel caso l'utente fosse anonimo vedrà vari meme
tra i più votati di vari profili pubblici sul sito.
Il tutto in ogni caso accompagnato da una classifica dei temi più
usati per i meme
```

Descrizione del progetto dell'applicazione

Sotto a quanto scritto è presente il diagramma UML dei casi d'uso:

chat:



Tecnologie usate

Per la realizzazione del progetto, ho cercato di affidarmi ai migliori programmi/framework possibili in base alle mie esigenze.

In particolare è stato usato:

- Django: ambiente di sviluppo per applicazioni e siti web (uno degli obiettivi del corso è la padronanza di questo framework).
- django-taggit: un'applicazione per la gestione dei tag in maniera semplificata.
- Ajax e JQuery: strumenti per dare della dinamicità al sito, in particolare sono stati usati nella chat, per avere un controllo periodico sui messaggi arrivati e per mostrare quelli appena inviati senza dover ricaricare la pagina.
- pipenv: per la creazione di un ambiente virtuale dove poter installare in modo semplice varie applicazioni senza creare conflitti con altri ambienti.
- Visual Studio Code: per poter scrivere il codice del programma.
- sqlite3: per la gestione semplice e leggera del database.
- Bootstrap: per rendere più gradovi i template.

Organizzazione logica dell'applicazione a livello di codice

“Memelandia” rappresenta un grosso progetto, troppo per essere gestito nella sua interezza, perciò ho suddiviso il lavoro in 3 macroaree (o “applicazioni”) connesse e dipendenti tra loro.

La prima realizzata è stata “users” per poter creare utenti e profili, così da gestire la differenza di diritti tra un AnonymousUser e un utente registrato e loggato sul sito.

Inoltre gestisce un'altra differenza tra profili pubblici e privati: i primi hanno i meme visibili da tutti, i secondi solo da quelli con il quale hanno un legame di amicizia. Il sistema delle amicizie tra utenti è basato su una relazione simmetrica, cioè A per essere amico di B deve chiedergli l'amicizia e quest'ultimo deve confermare.

La seconda applicazione, in ordine cronologico, è “meme” che gestisce tutto quello che riguarda i meme, dal postarli al visualizzarli, commentarli, inoltrarli (nel caso in cui il profilo creatore fosse pubblico), taggarli e mettere like o dislike.

La terza, ma non per importanza, è la chat che permette agli utenti di creare la propria stanza accessibile solo da due utenti e di scambiarsi messaggi.

Una scelta abbastanza peculiare che feci fu la creazione del modello Profile nonostante la presenza di User, perché ho trovato più semplice la gestione di un profilo il più plasmabile e modificabile possibile.

Altre scelte a livello di codice sono anche le varie tecnologie usate presenti nel paragrafo precedente.

Scelte fatte

Ho preferito nella gestione del progetto suddividere il lavoro in tanti modelli e le funzionalità in tante funzioni nei vari file `views.py`, per avere funzioni piccole ma efficaci.

Tutto ciò comunque non deve far dedurre che non ci siano stati dei focus, infatti i modelli più grandi e decisivi sul programma sono Meme e Profile.

Per i tag ho preferito usare `django-taggit`, dato che semplifica di molto la gestione di qualcosa che in alternativa sarebbe più complesso.

Tra una relazione simmetrica e una asimmetrica ho preferito la prima opzione, perché nonostante le varie piattaforme di social network ora in voga funzionino con un approccio tendenzialmente asimmetrico, ho preferito la possibilità per ogni utente di essere consapevole delle identità dei suoi “amici digitali”.

Un'altra differenza rispetto a Instagram e Facebook è la presenza del dislike, più simile a Reddit, per dare una maggiore possibilità a un utente di esprimere il suo disappunto, più veloce e immediato di un commento.

Test

I test realizzati per questo progetto sono due: uno è per la gestione delle amicizie e l'altro per i like e dislike. I due campi che ho scelto di testare risultano, infatti, quelli che computazionalmente hanno più regole da seguire per il corretto comportamento delle funzioni che rappresentano, e quindi si possono definire i più delicati.

Il primo test crea 3 utenti/profili e analizza gli scambi di richiesta tra loro e l'esito, in particolare il primo manda una richiesta di amicizia a sé stesso, fallisce e poi la manda al terzo, controlla che il secondo utente non riesca a intercettare la richiesta e infine il terzo diventa amico del primo dopo aver accettato.

L'intenzione del secondo test è controllare le funzionalità del like e del dislike di un meme. Questo si propone di controllare che 2 like non si sovrappongono, che quando è attivo il like non è attivo il dislike e viceversa e che il like o dislike di un utente non influisca su quello di un altro. A livello tecnico, questo test si occupa di controllare le funzioni `checkLike` e `checkDislike` (presenti nel `views.py` dell'app “meme”)

Primo test:

```
class ProfileFriendsTest(TestCase):
    def setUp(self):
        self.acredentials = {
            "username": "test1",
            "password": "test1",
        }
        self.auser = User.objects.create_user(
            **self.acredentials,
        )

        self.bcredentials = {
            "username": "test2",
            "password": "test2",
        }
        self.buser = User.objects.create(
            **self.bcredentials,
        )

        self.ccredentials = {
            "username": "test3",
            "password": "test3",
        }
        self.cuser = User.objects.create(
            **self.ccredentials,
        )
```

```

def test_friends(self):
    self.assertFalse(self.auser.profile in self.buser.profile.friends.all())

    # auser non può accettare la sua stessa richiesta di amicizia indirizzata a cuser
    self.client.login(**self.acredentials)
    res = self.client.get(reverse('users:send_friend_request', args=[self.cuser.profile.pk]))
    self.assertEqual(res.status_code, 200)
    res = self.client.get(reverse('users:accept_friend_request', args=[1]))
    self.assertEqual(res.status_code, 200)
    res = self.client.get(reverse('users:friends_list'))
    self.assertNotContains(res, self.cuser.username)
    self.assertFalse(self.auser.profile in self.auser.profile.friends.all())
    self.assertFalse(self.cuser.profile in self.auser.profile.friends.all())

    # Neanche buser può accettare la richiesta di amicizia indirizzata a cuser
    self.client.force_login(self.buser, backend=None)
    res = self.client.get(reverse('users:accept_friend_request', args=[1]))
    self.assertEqual(res.status_code, 200)
    self.assertFalse(self.auser.profile in self.buser.profile.friends.all())

    # Un AnonymousUser non può accettare la richiesta
    self.client.logout()
    res = self.client.get(reverse('users:accept_friend_request', args=[1]))
    self.assertEqual(res.status_code, 302)

    # cuser accetta e diventa amico di auser
    self.client.force_login(self.cuser, backend=None)
    res = self.client.get(reverse('users:accept_friend_request', args=[1]))
    self.assertEqual(res.status_code, 200)
    self.assertTrue(self.auser.profile in self.cuser.profile.friends.all())

```

Secondo test:

```
class LikeTest(TestCase):
    def setUp(self):
        self.acredentials = {
            "username": "testA",
            "password": "testA",
        }
        self.auser = User.objects.create_user(
            **self.acredentials,
        )

        self.bcredentials = {
            "username": "testB",
            "password": "testB",
        }
        self.buser = User.objects.create(
            **self.bcredentials,
        )

        self.meme = Meme.objects.create(**{"user": self.auser})
        self.meme.save()
```

```

def test_like(self):
    # auser mette like al meme
    checkLike(self.meme,self.auser)
    self.assertEqual(self.meme.likes.count(),1)

    # auser rimette like al meme e si azzera il conteggio
    checkLike(self.meme,self.auser)
    self.assertEqual(self.meme.likes.count(),0)

    # auser rimette like al meme
    checkLike(self.meme,self.auser)
    self.assertEqual(self.meme.likes.count(),1)

    # auser mette dislike al meme: il contatore dei
    # like torna a zero e va a 1 quello dei dislike
    checkDislike(self.meme,self.auser)
    self.assertEqual(self.meme.likes.count(),0)
    self.assertEqual(self.meme.dislikes.count(), 1)

    # auser mette due volte dislike e il contatore ritorna a 1
    checkDislike(self.meme,self.auser)
    checkDislike(self.meme,self.auser)
    self.assertEqual(self.meme.dislikes.count(),1)

    # auser mette like al meme: il contatore dei
    # dislike torna a zero e va a 1 quello dei like
    checkLike(self.meme,self.auser)
    self.assertEqual(self.meme.dislikes.count(), 0)
    self.assertEqual(self.meme.likes.count(),1)

    # buser mette like al meme: il contatore dei like
    # va a 2 mentre quello dei dislike 0
    checkLike(self.meme,self.buser)
    self.assertEqual(self.meme.likes.count(),2)
    self.assertEqual(self.meme.dislikes.count(), 0)

    # buser mette dislike al meme: il contatore dei like
    # torna a 1 mentre quello di dislike diventa 1
    checkDislike(self.meme,self.buser)
    self.assertEqual(self.meme.likes.count(),1)
    self.assertEqual(self.meme.dislikes.count(), 1)

```

Risultati




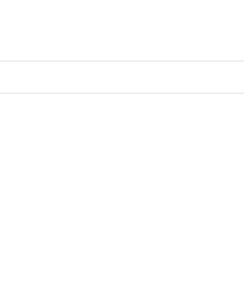
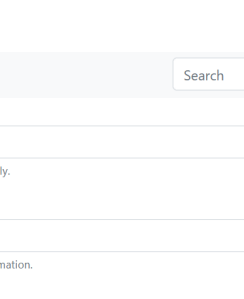


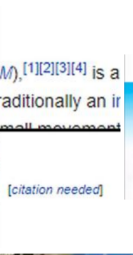


Homepage:

[Benvenuto](#) [Home](#) [Registrati](#) [Login](#)


Search


Search

Memelandia




Benvenuto Malferrari


Home

Logout
Chat
Meme
Gestione amici

Utente1


wikipedia



M), [1][2][3][4] is a
traditionally an ir
moll movement



[citation needed]

Like 1

Dislike 0

Inoltra


Comments:

Malferrari - Sept. 1, 2022, 10:27 a.m.
Molto divertente hahaha

Add comment

Chat:

Benvenuto Utente1

Home

Logout
Chat
Meme
Gestione amici

ChatDiProva - DjChat

Malferrari
Buongiorno Utente1
2022-09-01T12:13:39.101Z

Utente1
Buongiorno anche a Lei
2022-09-01T12:14:14.294Z

Send

Eventuali problemi riscontrati

Django risulta essere una novità per la mia esperienza attuale.

Per esempio la creazione dei modelli, la loro gestione e lo scambio dei valori nei template ai primi tentativi risulta molto complesso. Oltre a ciò bisogna tenere conto della difficoltà nel lavorare con numerosi file tutti collegati tra loro.

Per quanto efficiente Django risulta non semplice da debuggare quando ci sono problemi nella scrittura del codice, oltre a portare la quasi assente tipizzazione, peculiarità tipica di Python. Nei template invece spesso non viene segnalata la presenza di un errore, ma viene ignorata la riga che crea problemi.

Un altro inconveniente durante la realizzazione dell'applicazione è la consapevolezza di non riuscire a completare uno degli obiettivi prefissati a giugno, infatti nella vecchia lista dei requisiti vi era la presenza di creare e gestire community

di meme. Purtroppo i tempi e la complicatezza di tale funzione mi hanno convinto a chiedere un cambio di requisiti.

In particolare la traccia iniziale era questa:

Social Network per i meme "MEMELANDIA"

Un sito di social network tra utenti per la condivisione di meme.

Le varie funzioni:

- Iscrizione al sito e gestione di un profilo personale, inclusivo di foto, una bio e indicazione di una lista dei generi di meme preferiti
- L'utente iscritto potrà scegliere se rendere il suo profilo pubblico o privato, nel caso sia privato gli altri potranno vedere i dettagli solo se amici di questo
- Creazione di relazioni simmetriche: per essere amici, un utente A deve fare esplicita richiesta a B di essere accettato tra i suoi contatti e B deve rispondere positivamente
- Ricerca di utente in base a diverse caratteristiche
- Possibilità di condividere i propri meme ed inserire dei tag per delineare uno o più generi
- Possibilità di inoltrare meme altrui indicando però l'autore originale
- Possibilità di creare proprie community su un tema dove tutti possono pubblicare meme in base a quel tema, queste possono essere private o pubbliche. In caso di private per vedere i meme sarà necessario fare parte della community
- Per ogni meme è possibile mettere Like, Dislike e commentare
- L'utente che crea la comunità ha il diritto di moderare i contenuti e le attività della stessa
- Un utente non loggato potrà vedere i meme sui profili pubblici e sulle community pubbliche, per ogni altro tipo di attività sarà necessaria l'autenticazione
- La home page conterrà vari meme tra i più votati delle varie community frequentate dall'utente loggato; nel caso l'utente sia anonimo vedrà vari meme tra i più votati delle varie community pubbliche presenti sul sito. Il tutto in ogni caso accompagnato da una classifica dei temi più usati per i meme

In base ai facoltativi elencati per l'esempio visto a lezione:

- Recommendation system sui meme suggeriti in base ai tag preferiti e a ciò che seguono i propri amici sulla piattaforma
- Possibilità di invio/scambio di messaggi privati tra utenti connessi tra di loro