



Djagram

Filippo Ciarlo

Matr. 116685

Unimore FIM, Informatica

Progetto di Tecnologie Web

Indice

- | | |
|--------------------------------------------|------------|
| 1. Traccia del Progetto | pag. 3-5 |
| 2. Diagramma delle Classi del DB | pag. 6 |
| 3. Diagramma Use Case UML | pag. 7 |
| 4. Organizzazione Logica dell'Applicazione | pag. 8-10 |
| 5. Tecnologie Usate | pag. 11 |
| 6. Tests | pag. 12-15 |
| 7. Problemi Riscontrati | pag. 16 |
| 8. Risultati | pag. 17-21 |

Traccia del Progetto

Il progetto sviluppato consiste in un applicazione web ispirata al popolare social network di photo-sharing, *Instagram*. All'interno del sito web esistono diverse tipologie di utenti che possono interagire con esso, questi sono:

- **Utenti Anonimi:**

Con Utenti Anonimi si intendono quegli utenti che non sono ancora registrati al sito o sono registrati ma devono effettuare il "login" all'interno del sito web.

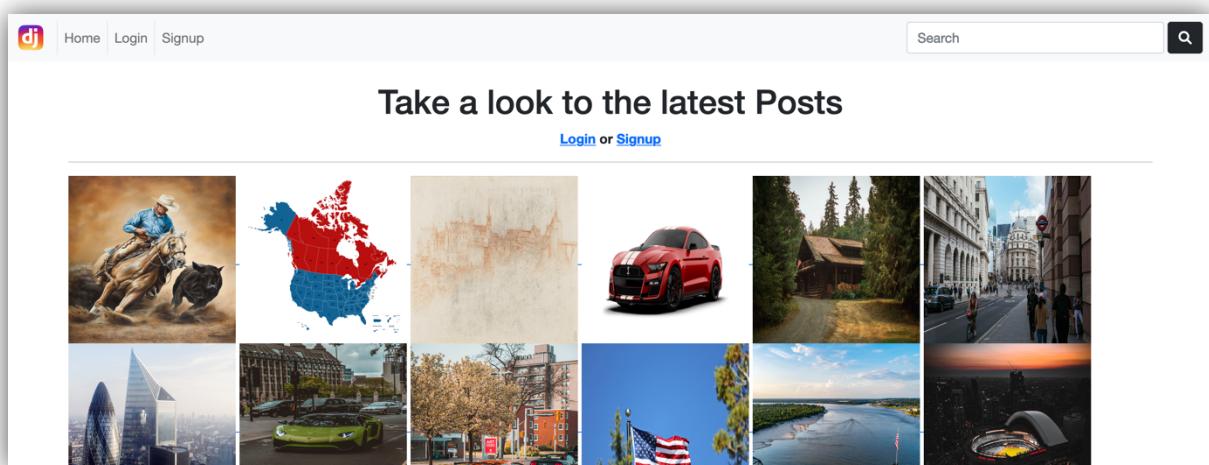
- **Utenti Registrati:**

Con Utenti Registrati si intendono tutti quegli utenti che sono registrati al sito e che sono già "loggati" all'interno del sito web.

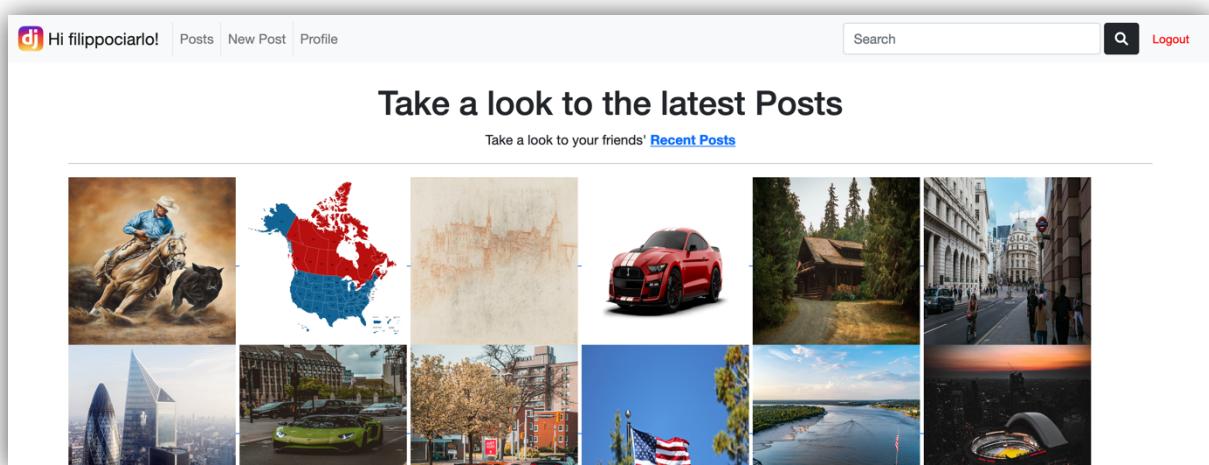
- **Utente Amministratore:**

Con Utente Amministratore (o *superuser*) si intende l'utente che ha accesso alla pagina di amministrazione del sito e che tramite questa svolge anche ruolo di moderatore di contenuti / utenti all'interno del sito web.

Home del sito web:



Vista della Home del sito da parte di un Utente Anonimo.



Vista della Home del sito da parte di un Utente Registrato (utente *@filippociarlo*).

Traccia del Progetto

Tutti gli utenti possono navigare all'interno del sito tramite la barra di navigazione questa mostrerà al suo interno, in base alla tipologia di utente, le funzioni a questo disponibili.

Di seguito vengono riportate le azioni che è possibile svolgere dai due principali gruppi di utenti che interagiscono con il sito web, gli **Utenti Anonimi** e gli **Utenti Registrati**.

Gli Utenti Anonimi possono:

- Visualizzare la Home del sito dove vi compaiono i Post caricati dagli utenti registrati al sito web.
- Registrarsi al sito nella sezione "Signup" o effettuare il login nella sezione "Login". In entrambe le sezioni, infondo alla pagina, è presente un collegamento veloce che consente di passare velocemente da una sezione ad un'altra.
- Nella sezione "Login" è presente un collegamento alla pagina di reimpostazione della password nel caso l'utente l'abbia dimenticata, in cui verrà chiesta l'email con cui ci si è registrati al sito (nel caso sia stata cambiata, l'ultima email registrata sul sito).
- Tramite il campo di ricerca presente all'interno della barra di navigazione è possibile cercare i Post degli utenti, su cui è possibili cliccare per avere una vista in dettaglio del Post, da dove è possibile vedere i commenti degli utenti registrati sotto il Post e il numero di Like che il Post ha ricevuto.

Il campo di ricerca è il modo con cui gli utenti anonimi possono interagire di più con il sito ma è anche quello che ne mette in evidenza i loro limiti, dato che nella vista in dettaglio anche se si ha accesso ai collegamenti alla pagina dell'utente creatore del Post e ai tag presenti nella descrizione del Post, questi se cliccati portano alla pagina di "Login", dato che sono funzionalità riservate agli utenti registrati.

Gli Utenti Registrati possono:

(1/2)

- Visualizzare la Home del sito dove vi compaiono i Post caricati dagli utenti registrati al sito web.
- Tramite il collegamento "New Post" presente all'interno della barra di navigazione del sito, pubblicare nuovi Post.
- Interagire con i Post presenti all'interno del sito, lasciando un commentando o un Like, nel caso l'utente sia anche creatore del Post, sotto di questo verranno mostrati due buttons che ne consentiranno uno la modifica del Post in tutte le sue componenti (Immagine, Descrizione, Tags) e l'altro la sua cancellazione.
- Seguire gli altri utenti registrati così da rimanere sempre aggiornati in caso questi pubblichino dei nuovi Post.
- Visualizzare la propria bacheca dove vengono elencati in ordine cronologico in base alla data di creazione, dal più recente (il Post in cima) al meno recente, tutti i propri Post, nel caso si seguano altri utenti i propri Post e quelli degli utenti seguiti verranno combinati e mostrati in ordine cronologico.
- Visualizzare il proprio profilo (e quello degli altri utenti), nello specifico ogni utente tramite il collegamento "Profile" presente nella barra di navigazione del sito ha un accesso rapido al proprio profilo, che rappresenta la propria pagina personale in cui viene mostrato l'username dell'utente, l'email, la Bio e vengono raggruppati tutti i Post caricati sul sito web dall'utente.

Traccia del Progetto

Gli Utenti Registrati possono:

(2/2)

- Tramite il button "Edit Profile" presente nella propria pagina del profilo, modificare i propri dati personali, username ed email oltre a poter inserire una nuova Bio (andando a modificare quella di default) e una nuova immagine del profilo (sostituendo così quella di default). Inoltre, da questa sezione l'utente ha la possibilità di cancellare il proprio account.
- Tramite il campo di ricerca presente all'interno della barra di navigazione è possibile effettuare delle ricerche, più avanzate rispetto a quelle disponibili agli utenti anonimi, in cui data una chiave di ricerca, il sito web effettua una ricerca tra *Users*, *Tags* e *Post* mostrando i risultati ottenuti per ognuna di queste categorie, questo tipo di ricerca consente agli utenti di poter cercare gli altri utenti registrati al sito web, tag a cui si è interessati o semplicemente navigare tra i Post presenti all'interno del sito web.

Post:

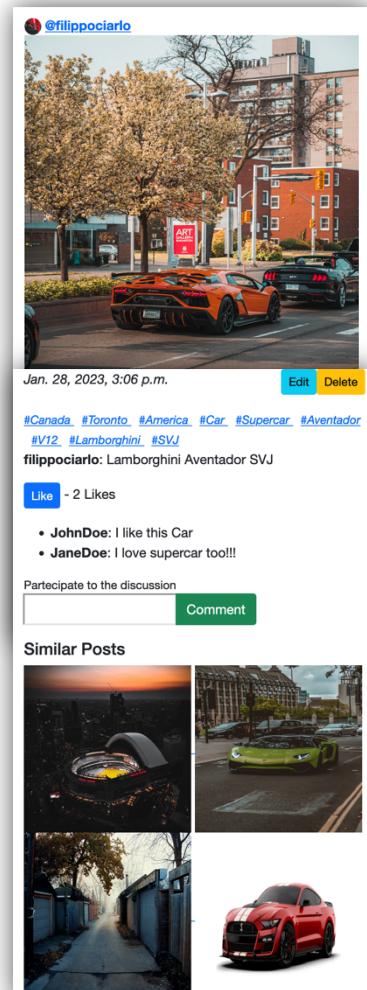
I Post costituiscono il centro dell'esperienza utente all'interno del sito, essi sono composti da:

- Un'immagine (obbligatoria)
- Una descrizione (opzionale)
- Uno o più tag (obbligatori)

I tag costituiscono dei link ipertestuali, dove ognuno di essi rappresenta una parola chiave utilizzata per descrivere il Post, ogni tag collega ad una pagina dove vengono mostrati tutti i Post aventi il tag in questione.

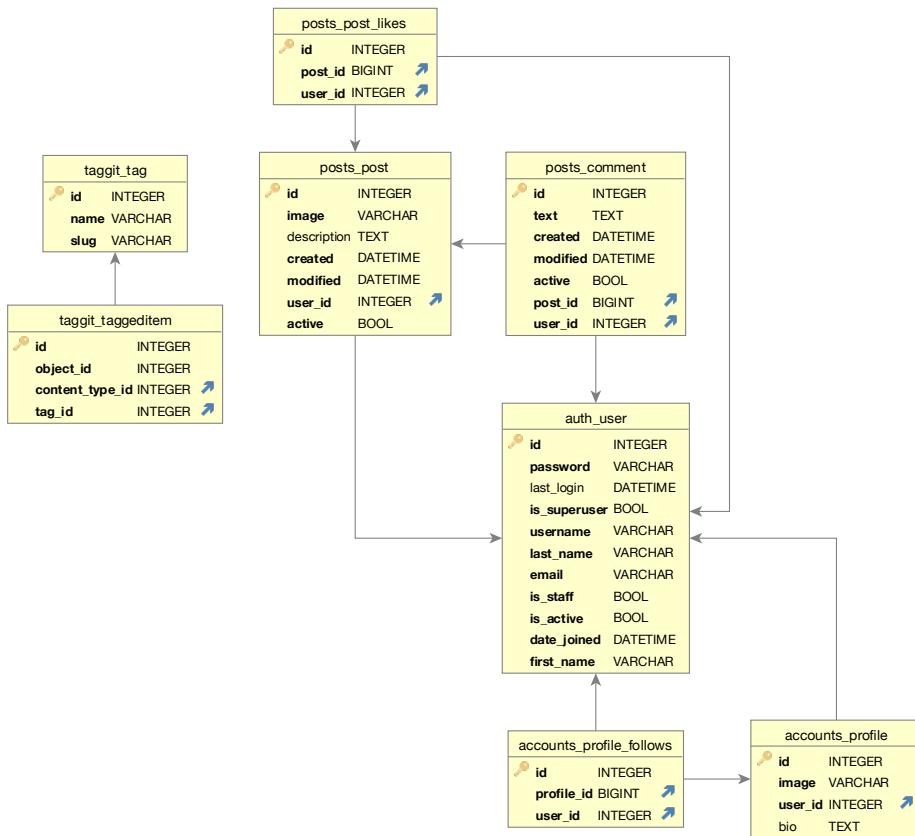
Nella vista in dettaglio dei Post si ha:

- La sezione commenti, dove gli utenti registrati possono creare una discussione sotto un Post.
- Il button Like, che consente di mettere Like ad un Post (o toglierlo nel caso il like sia già stato messo e si ha intenzione di toglierlo).
- All'autore del Post verranno mostrati due ulteriori buttons rispettivamente per la modifica e la cancellazione del Post
- Sotto la sezione commenti è presente la sezione "**Similar Posts**" dove vengono mostrati 4 Post "potenzialmente" simili o "potenzialmente" li correlati al Post che si sta visualizzando.



Vista di un Post in dettaglio.

Diagramma del DB



auth_user (User)

La tabella User contiene le informazioni di base degli utenti registrati al sito, il modello usato è quello predefinito di Django contenuto in `django.contrib.auth.models`.

accounts_profile (Profile)

Alla creazione di un nuovo User a questo viene associato un Profile, questa relazione viene gestita tramite metodi *signal dispatcher*. La tabella Profile amplia la tabella User consentendo di associare a quest'ultimo un'immagine del profilo ed una Bio.

Inoltre, Profile permette di implementare la relazione `accounts_profile_follows` (n,m) che definisce la funzionalità "Follow" in cui gli User possono essere seguiti da altri User.

posts_post (Post)

Un User può pubblicare (0,n) Post, un Post ha (1,1) User (utente creatore). Ogni Post ha un campo per una sola immagine (obbligatoria), una descrizione (opzionale) una data di creazione e una di modifica, le quali vengono generate automaticamente alla creazione/modifica del Post.

Inoltre, tramite l'applicazione `django-taggit` che consente l'implementazione dei Tag all'interno di Django, ogni Post ha è associato in modo non relazionale alla tabella taggit_tag.

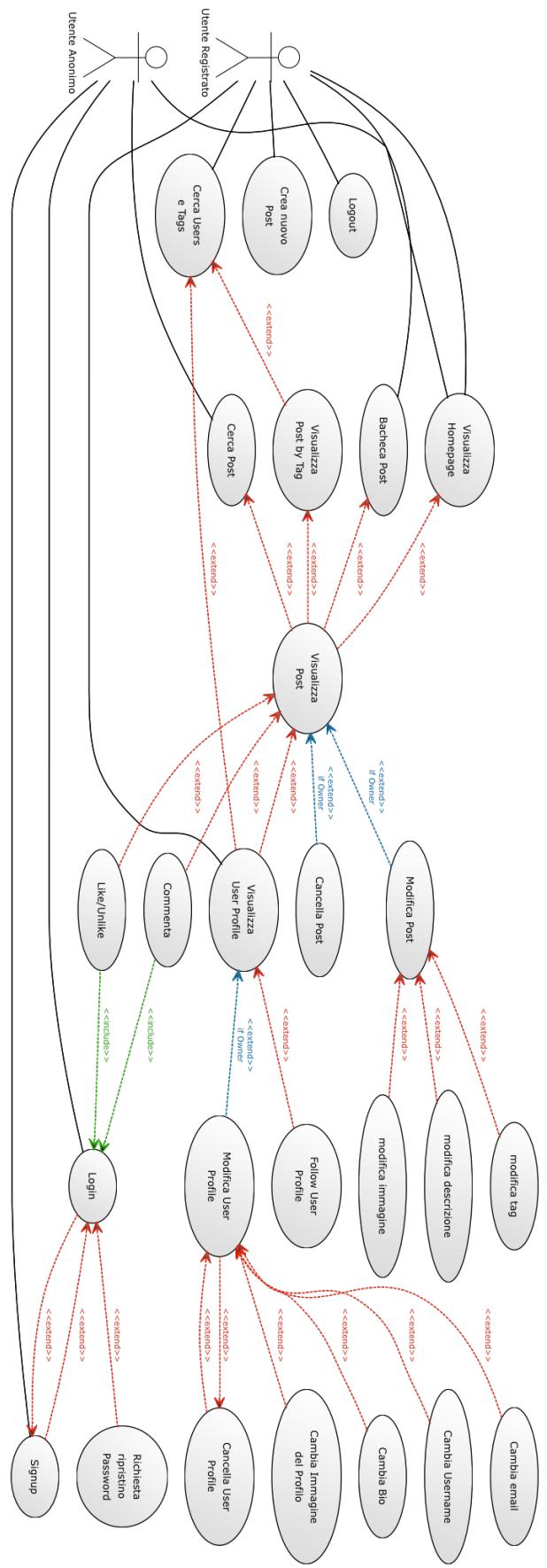
posts_post_like (Likes)

Ogni Like è associato ad un Post ed un User, ogni Post può avere (0,1) Likes da parte di un User specifico, ogni User può avere (0,n) Likes.

posts_comment (Comment)

Ogni Comment è associato ad un Post ed un User, un User può avere più Comment sotto un solo Post e un Comment ha un solo User.

Diagramma UML



Organizzazione Logica dell'Applicazione

Il programma è suddiviso in due applicazioni principali "accounts" e "posts".

- **accounts:**

In **accounts** vengono gestiti tutti gli aspetti che riguardano la gestione degli Utenti delle informazioni associate al loro Profilo all'interno del sito web.

Signup

La registrazione dell'utente Il cui Form è stato ampliato per permettere l'inserimento, oltre ai classici email e password, dati come Nome, Cognome e Username parte di questi ultimi vengono gestiti dal *signal dispatcher* che alla registrazione di un utente ne crea automaticamente anche il profilo. Viene inoltre controllato che l'utente non si registri con un'email già in uso da un altro utente e questo vale anche per l'username. L'utente è informato sullo stato di successo/insuccesso della registrazione tramite dei messaggi mostrati sulla scherma in cui si trova.

Login & Logout

L'autenticazione utente è gestita per la maggior parte da Django, viene mostrato un messaggio di errore nel caso vengano usate delle credenziali errate o inesistenti se l'autenticazione va a buon fine si è rindirizzati alla bacheca dei Post. Mentre per il Logout viene chiesta conferma dell'azione prima di proseguire, se l'operazione va a buon fine si è reindirizzati alla Homepage del sito e viene mostrato un messaggio che comunica che il logout è avvenuto con successo.

Follow System

Un utente registrato può seguire un altro utente, nel caso in cui si segua già un utente è possibile smettere di seguire l'utente in questione, in sintesi le opzioni due opzioni sono Follow o Unfollow.

Modifica & Cancellazione del Profilo

È possibile modificare i propri dati personali come username, email, Bio (andando a sostituire quella di default), immagine del profilo (andando a sostituire quella di default) quando si sostituisce la vecchia immagine del profilo (a meno che questa non sia l'immagine di default) questa viene cancellata, così da ottimizzare la gestione dello spazio in memoria. Inoltre, è possibile cancellare l'account utente, ciò comporterà l'eliminazione dei Post e del Profilo dal sito oltre alla cancellazione di tutte le immagini appartenenti all'utente dalla memoria (immagine del profilo e immagini dei Post) così da ottimizzare la gestione dello spazio in memoria. In entrambi i casi sia in caso di modifica delle impostazioni che in caso di cancellazione del profilo l'utente viene informato in caso di successo dell'azione tramite dei messaggi mostrati sulla scherma in cui si trova o viene reindirizzato in caso di cancellazione del profilo.

Reimpostazione della Password

Un utente può richiedere il reset della sua password cliccando su "Forgot Password?" al momento del Login. Il link per il recupero è creato in automatico da Django e per l'invio dell'email si utilizza il server SMTP di Gmail. Per il cambio della password erano inizialmente presenti i template di default di Django ma sono stati modificati successivamente per avere una grafica simile a quella del resto del sito.

Organizzazione Logica dell'Applicazione

- **posts**

In **posts** vengono gestiti tutti gli aspetti che riguardano la gestione dei Post all'interno del sito.

Homepage & Profilo

Nella homepage vengono mostrati tutti i Post postati all'interno del sito dal più recente al meno recente. All'interno del profilo di un utente vengono mostrati tutti i Post postati all'interno del sito web dall'utente in questione.

Elenco dei Post

All'interno del sito web esistono due tipi di elencazione dei Post:

- I Post mostrati all'interno della bacheca, in cui vengono mostrati i Post dell'utente e degli utenti seguiti.
- I Post mostrati cliccando su un tag, in cui vengono mostrati tutti i Post presenti nel sito aventi lo stesso Tag.

Dettaglio di un Post

Per ogni Post in dettaglio vengono mostrati la sua data di creazione, l'immagine, la lista di tag, la descrizione, il numero di Like, i commenti associati al Post e un Form per inserire un nuovo commento. Su ogni commento viene eseguito controllo di validità, se il commento inserito è un commento vuoto ne viene impedita la pubblicazione e l'utente è avvisato con un messaggio di errore. Viene inoltre mostrata una sezione "**Similar Posts**" dove vengono consigliati all'utente Post simili a quello che si sta visualizzando.

- **Recommendation System**

Per ogni Post vengono raggruppati tutti i Post del sito che contengono uno o più Tag in comune con il Post in questione, una volta raggruppati questi vengono ordinati in modo per numero di tag in comune, infine vengono selezionati i primi 4 Post, questi rappresentano i Post con il maggior numero di Tag in comune con il Post in questione.

Il recommendation system implementato fa fede sul fatto che i Tag inseriti dagli utenti corrispondano ad una descrizione veritiera del Post.

Creazione di un Post

Quando un utente registrato si reca sulla pagina di creazione di un nuovo Post, viene mostrato un Form i cui campi Immagine e Tag sono obbligatori, questo poiché le immagini sono il contenuto principale del sito mentre l'inserimento di almeno un Tag è essenziale per catalogare l'immagine ai fini del Recommendation System. Se il form è riempito correttamente si è reindirizzati alla pagina del nuovo Post creato.

Modifica e Cancellazione di un Post

L'utente creatore di un Post ha accesso, tramite la sua vista in dettaglio, alla modifica e cancellazione del Post stesso. Nella pagina di modifica del Post viene mostrata un'anteprima del Post che si sta andando a modificare e i suoi campi (immagine, descrizione, tags) già precompilati con i dati attualmente in essere. Quando viene sostituita l'immagine del Post la vecchia immagine viene cancellata dalla memoria locale così da ottimizzare la gestione dello spazio in memoria.

Organizzazione Logica dell'Applicazione

- `posts`

Ricerca

Per l'implementazione della ricerca dei Post è stato implementato un filtro sui campi "`username`", "`description`" e "`tags`" del Post così da non limitare la ricerca ad un solo campo, la ricerca è stata poi ulteriormente ampliata per far sì che oltre ai Post siano restituiti anche i Profili utente e i Tag che corrispondono integralmente o in parte alla chiave di ricerca inserita.

Tecnologie Usate

Django:

django==4.1.0

È stato usato il web framework Django il quale fornisce un certo numero di soluzioni integrate che facilitano lo sviluppo rapido di applicazioni per la gestione di contenuti Web.

Django Environ:

django-environ==0.9.0

È stato utilizzato il pacchetto Django Environ per nascondere tramite variabili di ambiente i valori più delicati all'interno del codice che definisce le impostazioni del sito, come la sua Private Key e le credenziali per il server SMTP di Gmail.

Pillow:

pillow==9.4.0

È stato utilizzata la libreria Pillow per la gestione delle immagini che sono state usate sia per l'implementazione dei Post che per l'implementazione dell'immagine del profilo utente.

Taggit:

django-taggit==3.1.0

È stato utilizzato Django Taggit per l'implementazione dei Tag all'interno del sito web, questi sono stati usati per fornire una sorta di link-filtro alle immagini, in cui cliccando su un tag si è reindirizzati ad una pagina con tutti i Post aventi lo stesso Tag, questo meccanismo è stato inoltre usato per l'implementazione del Recommendation System, in cui dati *n*-tag presenti in un Post vengono mostrati sotto di questo 4 Post simili-per-tag, ovvero 4 Post in cui 1 o più tag sono uguali a quelli del Post in questione.

Django Crispy Forms:

django-crispy-forms==1.14.0

È stato utilizzato Django Crispy Form per controllare il rendering dei Form e implementare il concetto di programmazione DRY.

Bootstrap 5:

crispy-bootstrap5==0.6

È stato utilizzato Bootstrap 5 per migliorare l'aspetto grafico del sito in particolare modo è stato fondamentale per l'implementazione della Navbar e tutti gli elementi al suo interno.

Coverage:

coverage==7.0.5

È stato utilizzato Coverage per tracciare il codice eseguito dai test nonché per la creazione di report riguardanti il codice eseguito.

Tests

Sono stati scritti test sulla maggior parte del progetto ma i più rilevanti sono quelli relativi alle viste. All'interno della cartella delle due applicazioni principali del progetto (**accounts** e **posts**) è stata creata una cartella denominata **tests** con all'interno i test fatti all'interno dell'applicazione. È stato inoltre creato uno script BASH (**project_tests_launcher.sh**), presente nella directory principale del progetto, il quale lancia i test presenti all'interno del progetto e tramite l'ausilio di **Coverage** mostra all'interno della SHELL un report riguardante la copertura del codice eseguito dai test e ne crea anche report HTML consultabile, che consente di vedere praticamente quali righe di codice sono state eseguite e quali meno, infine lancia lo script Python (**cleaning_procedure.py**), presente all'interno della directory principale del progetto, per l'eliminazione dei file multimediali creati durante la fase di test.

Per ulteriori approfondimenti vedere la sezione in merito a **Problemi Riscontrati**.

Vengono di seguito riportati alcuni test dell'applicazione **accounts**.

```
# Follow
"""
    Verifica che un utente registrato possa
    seguire (Follow) un'altro utente.
"""

def test_follow_profile_LoggedIn_view_Success_Follow_POST(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.post(self.accounts_follow_url)
    self.assertRedirects(
        response,
        f"/posts/profile/{self.user2.username}",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )
    self.assertEqual(self.user.profile.follows.count(), 1)

"""

    Verifica che un utente registrato possa smettere
    di seguire (Unfollow) un'altro utente.
"""

def test_follow_profile_LoggedIn_view_Success_Unfollow_POST(self):
    self.client.login(username='MarioRossi', password='testpass123')
    self.client.post(self.accounts_follow_url)
    response = self.client.post(self.accounts_follow_url)
    self.assertRedirects(
        response,
        f"/posts/profile/{self.user2.username}",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

    self.assertEqual(self.user.profile.follows.count(), 0)
```

Tests

Vengono di seguito riportati alcuni test dell'applicazione **accounts**.

```
# Edit Profile
"""
    Verifica che un utente registrato abbia accesso
    alla pagina di modifica delle proprie impostazioni.
"""

def test_edit_profile_LoggedIn_view_Success(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.get(self.accounts_edit_profile_url)
    self.assertEqual(response.status_code, 200)

    # Test HTML code
    self.assertTemplateUsed(response, "accounts/edit_profile.html")
    self.assertContains(response, "MarioRossi")
    self.assertContains(response, "Profile Info")

"""

    Verifica che un utente non-registrato abbia accesso
    alla pagina di modifica delle impostazioni utente
"""

def test_edit_profile_LoggedOut_view_Insuccess(self):
    response = self.client.get(self.accounts_edit_profile_url)
    self.assertRedirects(
        response,
        "/accounts/login/?next=/accounts/profile/edit",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

"""

    Verifica che un utente registrato possa modificare
    le proprie impostazioni.
"""

def test_edit_profile_view_Success_POST(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.post(self.accounts_edit_profile_url,
        {
            "username": "NewMarioRossi",
            "email": "newmariorossi@email.com",
            "image": self.image_2,
            "bio": "This is the New Mario Rossi bio."
        }
    )
    self.assertRedirects(
        response,
        "/accounts/profile/edit",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

    # test website messages
    messages = list(get_messages(response.wsgi_request))
    self.assertEqual(len(messages), 1)
    self.assertEqual(str(messages[0]), "Your profile was updated successfully.")
```

Tests

Vengono di seguito riportati alcuni test dell'applicazione `posts`.

```
# Post Create
"""
    Verifica che un utente registrato abbia accesso
    alla pagina di creazione di un nuovo post.
"""

def test_post_create_view_LoggedIn_Success(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.get(self.post_create_url)
    self.assertEqual(response.status_code, 200)

    # Test HTML code
    self.assertTemplateUsed(response, "posts/post_create.html")
    self.assertContains(response, "Create a new Post")

"""

    Verifica che un utente non-registrato non abbia
    accesso alla pagina di creazione di un nuovo post.
"""

def test_post_create_view_LoggedOut_Insuccess(self):
    response = self.client.get(self.post_create_url)
    self.assertRedirects(
        response,
        "/accounts/login/?next=/posts/new/",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

"""

    Verifica che un utente registrato
    possa creare un nuovo post.
"""

def test_post_create_view_Success_POST(self):
    image = self.image_2
    description = "This is the description for a New Post"
    tag = self.post.tags

    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.post(self.post_create_url,
        {
            "image":image,
            "description":description,
            "tags":tag
        }
    )
    self.assertRedirects(
        response,
        "/posts/2/",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )
    self.assertEqual(Post.objects.all().count(), 2)
```

Tests

Vengono di seguito riportati alcuni test dell'applicazione **posts**.

```
# Search
#####
    Verifica che dato un database adeguatamente popolato per il test
    la ricerca vada a buon fine e ritorni un Username ed un Tag che
    soddisfino la query.
#####

def test_post_search_view_Success(self):
    self.client.login(username='MarioRossi', password='testpass123')
    post_search_url = '{url}?{filter}={value}'.format(
        url=reverse("search_results"),
        filter="q",
        value="Mar"
    )
    response = self.client.get(post_search_url)
    self.assertEqual(response.status_code, 200)

    # Test HTML code
    self.assertTemplateUsed(response, "posts/search_results.html")
    self.assertContains(response, "Search Results")
    self.assertContains(response, "MarioRossi")
    self.assertContains(response, "MarTag")
    self.assertNotContains(response, "No Posts found.")

#####

    Verifica un utente non registrato riceva come
    risultato della ricerca solo post.
#####

def test_post_search_LoggedOut_view_Success(self):
    response = self.client.get(self.post_search_url)
    self.assertEqual(response.status_code, 200)

    # Test HTML code
    self.assertTemplateUsed(response, "posts/search_results.html")
    self.assertNotContains(response, "Users")
    self.assertNotContains(response, "Tags")
    self.assertContains(response, "Posts")

#####

    Verifica un utente registrato riceva come
    risultato della ricerca Utenti, Tag e Post.
#####

def test_post_search_LoggedIn_view_Success(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.get(self.post_search_url)
    self.assertEqual(response.status_code, 200)

    # Test HTML code
    self.assertTemplateUsed(response, "posts/search_results.html")
    self.assertContains(response, "Users")
    self.assertContains(response, "Tags")
    self.assertContains(response, "Posts")
```

Problemi Riscontrati

Nei test non si è riuscito a far eseguire il codice delle funzioni del sito il che hanno come conseguenza della loro esecuzione l'eliminazione delle immagini in locale, in particolare il codice in questione riguarda le viste:

- `accounts.views.edit_profile_view`
- `accounts.views.delete_profile_view`

Questo nonostante il meccanismo di cancellazione dei file vecchi funziona durante il normale utilizzo del sito, per sopperire a questo è stato creato uno script Python (`cleaning_procedure.py`), presente all'interno della directory principale del progetto, per l'eliminazione di questi file immagine creati durante la fase di test e non cancellati al termine della fase di test.

```
"""
    Verifica che un utente registrato possa
    cancellare il proprio account.
"""

def test_delete_profile_view_Success_POST(self):
    username = self.user.username
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.post(self.accounts_delete_profile_url)

    self.assertRedirects(
        response,
        '/',
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

    # test website messages
    messages = list(get_messages(response.wsgi_request))
    self.assertEqual(len(messages), 1)
    self.assertEqual(str(messages[0]), "The user @{ username } is been deleted successfully.")

"""

    Verifica che un utente registrato possa modificare
    le proprie impostazioni.

"""

def test_edit_profile_view_Success_POST(self):
    self.client.login(username='MarioRossi', password='testpass123')
    response = self.client.post(self.accounts_edit_profile_url,
    {
        "username": "NewMarioRossi",
        "email": "newmariorossi@email.com",
        "image": self.image_2,
        "bio": "This is the New Mario Rossi bio.",
    })
    self.assertRedirects(
        response,
        "/accounts/profile/edit",
        status_code=302,
        target_status_code=200,
        fetch_redirect_response=True
    )

    # test website messages
    messages = list(get_messages(response.wsgi_request))
    self.assertEqual(len(messages), 1)
    self.assertEqual(str(messages[0]), "Your profile was updated successfully.")
```

Risultati

Hi filippociarlo! Posts New Post Profile Search Logout

JohnDoe - Jan. 28, 2023, 3:16 p.m.



#Horse #Cowboying #Cowboy #Art #Rodeo
@JohnDoe: Rodeo!

JaneDoe - Jan. 28, 2023, 3:15 p.m.



Bacheca dei Post.

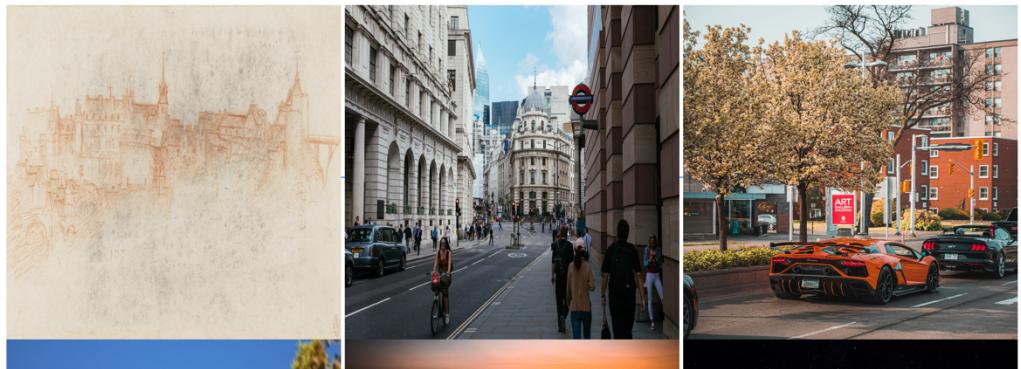
Hi filippociarlo! Posts New Post Profile Search Logout



@filippociarlo
filippo@email.com

Hi i'm Filippo and these are some picture that i like to share with you.

Edit Profile



Profilo utente.

Risultati

The screenshot shows a user profile page for 'filippociarlo'. At the top, there's a navigation bar with links for 'Posts', 'New Post', and 'Profile'. On the right side of the header are a search bar, a magnifying glass icon, and a 'Logout' link. Below the header is a circular profile picture of a person sitting at a desk. The username '@filippociarlo' is displayed below the picture, along with the email 'filippo@email.com'. A section titled 'Profile Info' follows, containing fields for 'Username*' (set to 'filippociarlo'), 'Email*' (set to 'filippo@email.com'), and 'Image*'. The 'Image*' field shows a preview of a profile picture and a file selection button. Below these fields is a 'Bio' section with the text: 'Hi i'm Filippo and these are some picture that i like to share with you.' At the bottom of the page are two buttons: 'Go Back' and 'Update', and a 'Danger Zone' section with a 'Delete Account' button.

Pagina di modifica delle impostazioni del Profilo utente.

The screenshot shows a 'Delete Account' confirmation page. At the top, there's a navigation bar with links for 'Posts', 'New Post', and 'Profile'. On the right side of the header are a search bar, a magnifying glass icon, and a 'Logout' link. The main content area is titled 'Delete Account' and contains the question 'Are you sure you want to Delete this Account?'. Below this question is the user's handle '@filippociarlo'. At the bottom of the page are two buttons: 'Delete Account' (in red) and 'Go Back'.

Pagina di cancellazione del Profilo utente.

Risultati

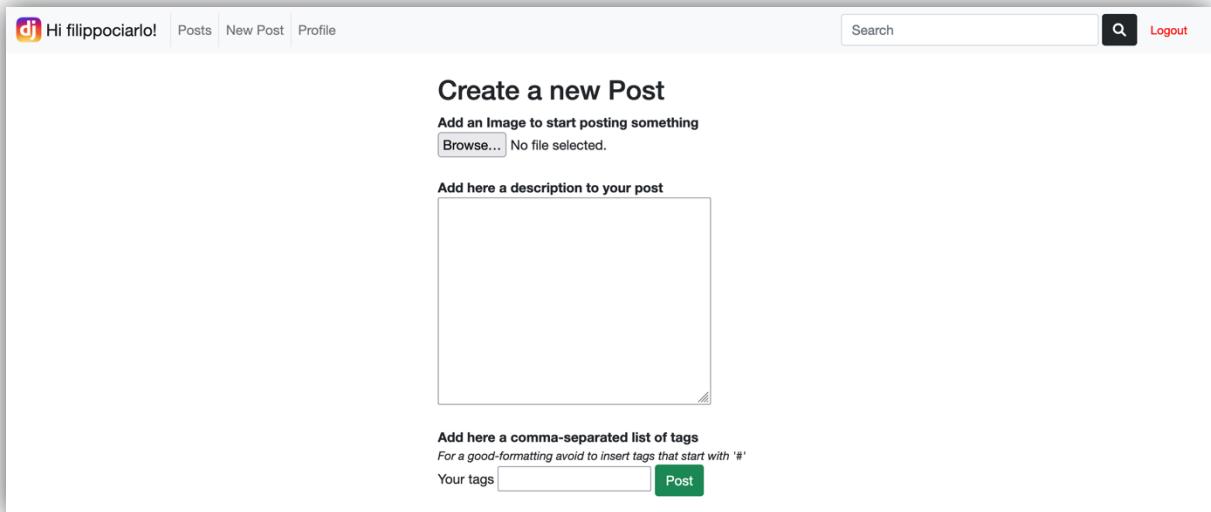
Hi filippociaro! Posts New Post Profile Search Logout

Create a new Post

Add an Image to start posting something
 No file selected.

Add here a description to your post

Add here a comma-separated list of tags
For a good-formatting avoid to insert tags that start with '#'
Your tags Post

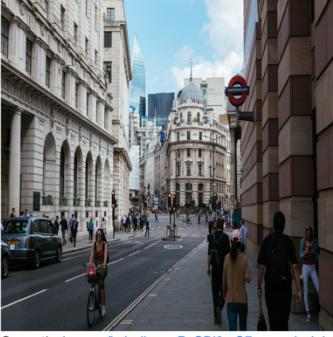


Pagina di creazione di un nuovo Post.

Hi filippociaro! Posts New Post Profile Search Logout

Edit Post

click on the image to get back

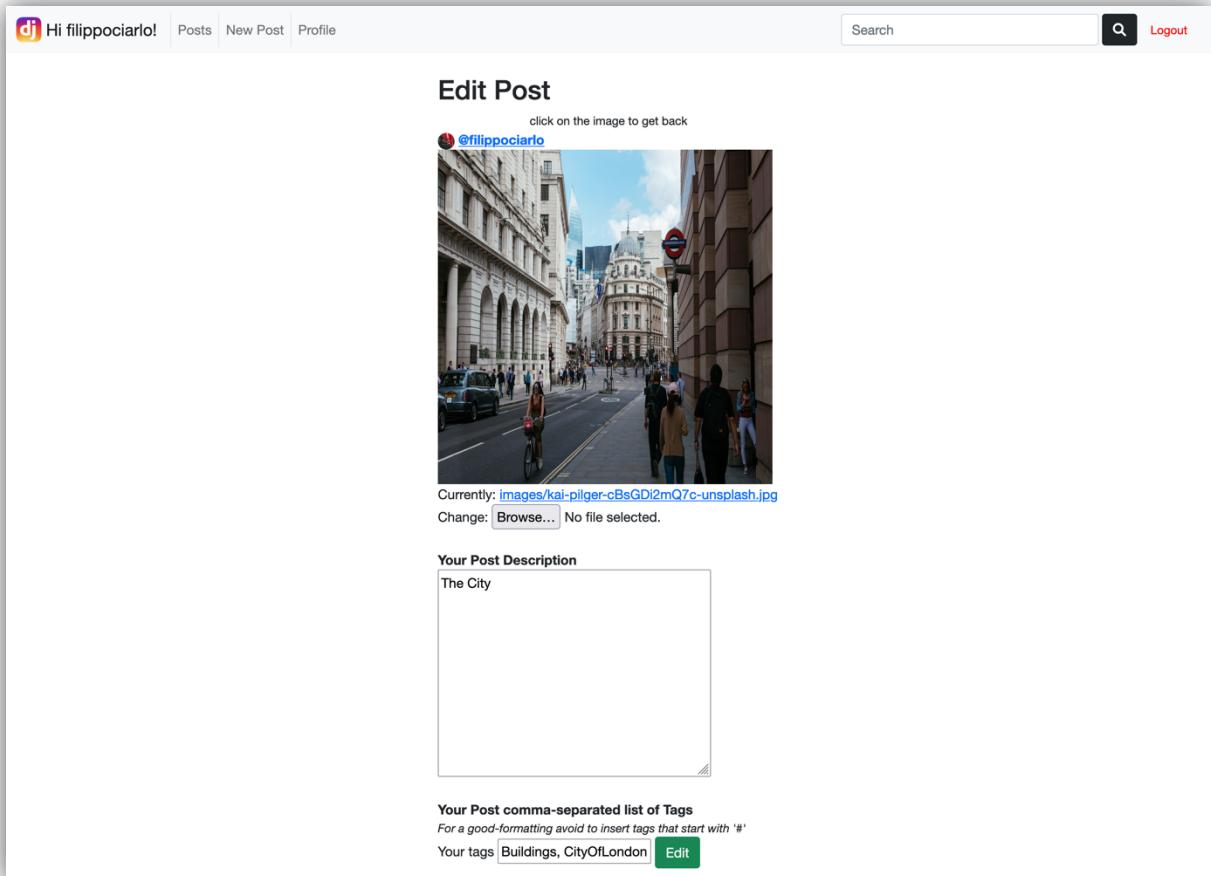

@filippociaro

Currently: [images/kai-pilger-cBsGDi2mQ7c-unplash.jpg](#)
Change: No file selected.

Your Post Description

The City

Your Post comma-separated list of Tags
For a good-formatting avoid to insert tags that start with '#'
Your tags Buildings, CityOfLondon Edit



Pagina di creazione di modifica di un Post.

Risultati



Post n.4

Similar Posts



Post suggeriti per il Post n.4

dj Hi filippociarlo! Posts New Post Profile Search Logout

#Horse

🕒 @JohnDoe - Jan. 28, 2023, 3:16 p.m.

#Horse #Cowboying #Cowboy #Art #Rodeo
@JohnDoe: Rodeo!

🕒 @JaneDoe - Jan. 28, 2023, 2:59 p.m.

#USA #Horse #Rodeo #Cowboying #Cowboy #Arena
#Horses #Flag #Fun
@JaneDoe: Rodeo season is started!!!

Elenco dei Post con il Tag "Horse".

Risultati

The screenshot shows a search results page for the query "j". The interface includes a navigation bar with "Home", "Login", and "Signup" buttons, and a search bar with the letter "j" and a magnifying glass icon. The main content area is titled "Search Results" and "Posts". It displays a grid of nine images: a cowboy on a horse, a map of North America with red and blue states, a red sports car, a forest scene, a modern building, and a city street.

Risultato della ricerca della parola chiave "j" da parte di un Utente Anonimo.

The screenshot shows a search results page for the query "j" by a registered user (@filippociarlo). The interface includes a navigation bar with the user's name, "Posts", "New Post", and "Profile" buttons, and a search bar with the letter "j" and a magnifying glass icon. The main content area is titled "Search Results" and "Users", showing two users: @JohnDoe and @JaneDoe. It also includes a "Tags" section with hashtags: #JacksonHole, #BlueJays, and #SVJ. The "Posts" section is identical to the first screenshot, displaying the same grid of nine images.

Risultato della ricerca della parola chiave "j" da parte di un Utente Registrato (utente @filippociarlo).