

Lab 10: Deployment

Obiettivo del laboratorio

L'obiettivo di questo laboratorio è pubblicare online il social network sviluppato durante il corso, utilizzando la piattaforma [PythonAnywhere](https://www.pythonanywhere.com). In questo modo, l'applicazione sarà accessibile via web da qualsiasi dispositivo connesso a Internet.

Descrizione dell'attività

Di seguito sono riportate le istruzioni dettagliate per completare il deploy:

1. Crea un account gratuito sulla piattaforma:

<https://www.pythonanywhere.com/registration/register/beginner/>

2. Comprimi il progetto del **laboratorio 9** (puoi anche usare la [soluzione su GitHub](#)) in un file `.zip` e caricalo tramite la pagina dei file accessibile dalla tua Dashboard personale. Clicca su **“Files”** (**Figura 1**) e troverai un pulsante **“Upload a file”** che ti consentirà di caricare il file `.zip` (**Figura 2**).

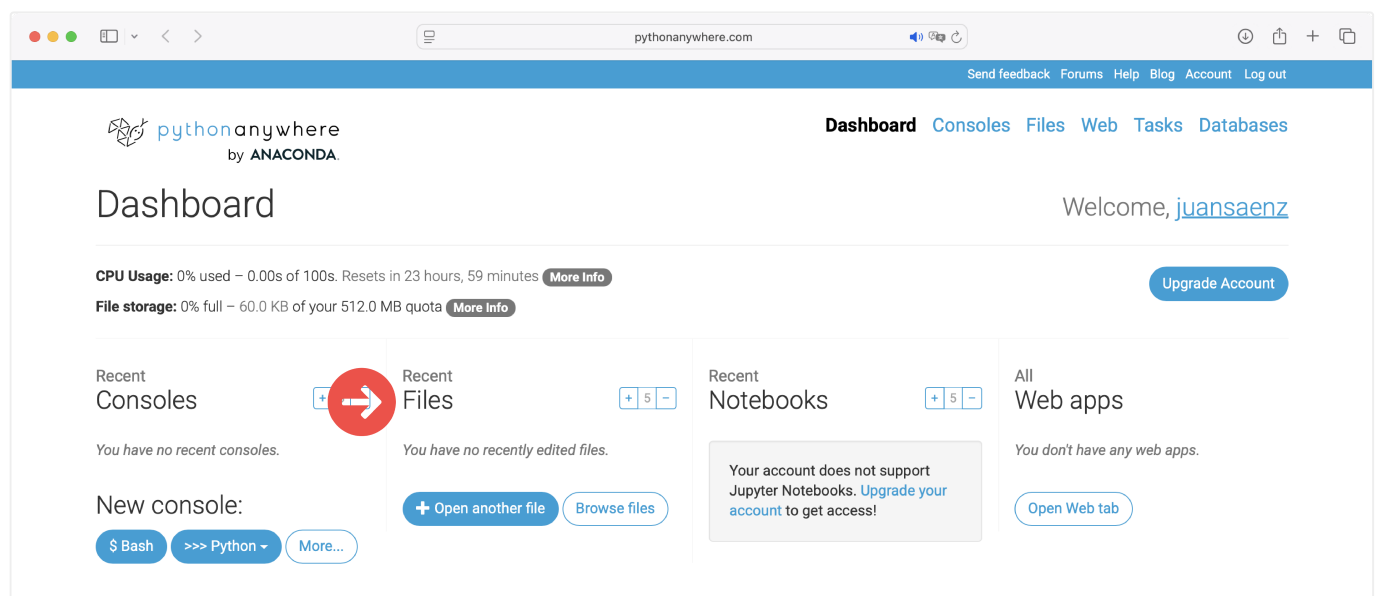


Figura 1: Dashboard PythonAnywhere.

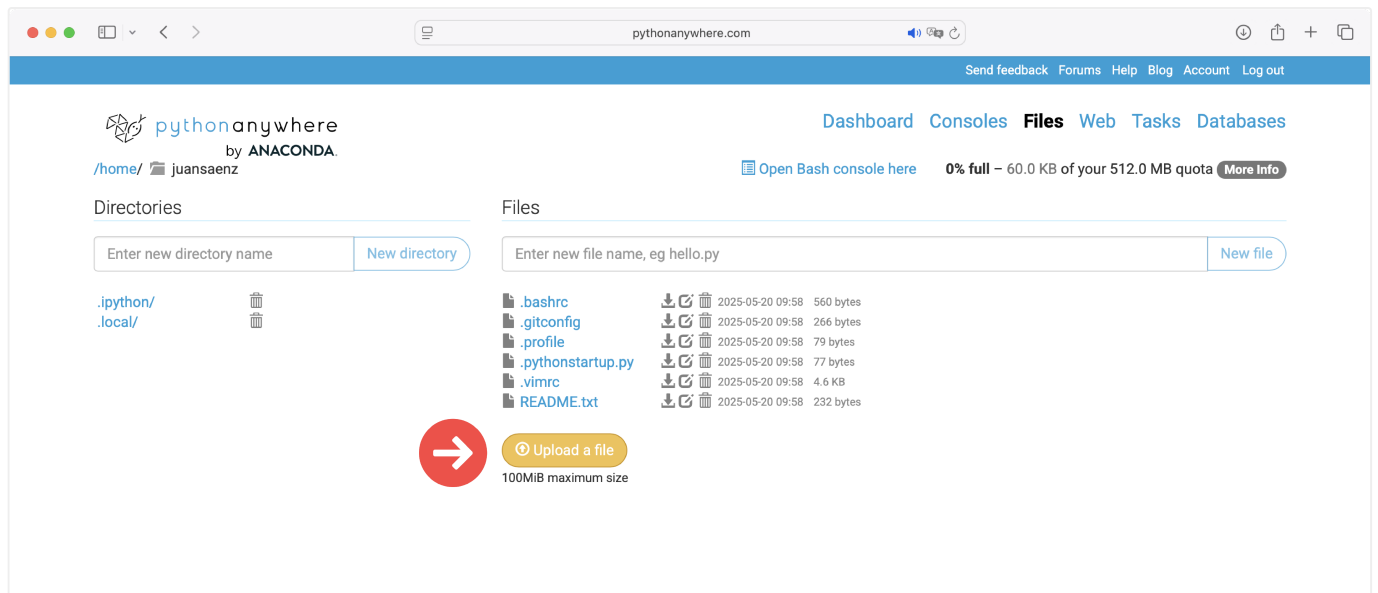


Figura 2: Schermo caricamento file.

3. Apri una nuova console **Bash** dalla Dashboard personale (**Figura 3**), ed esegui il comando `unzip [nome del file]` per scompattare il progetto appena caricato.

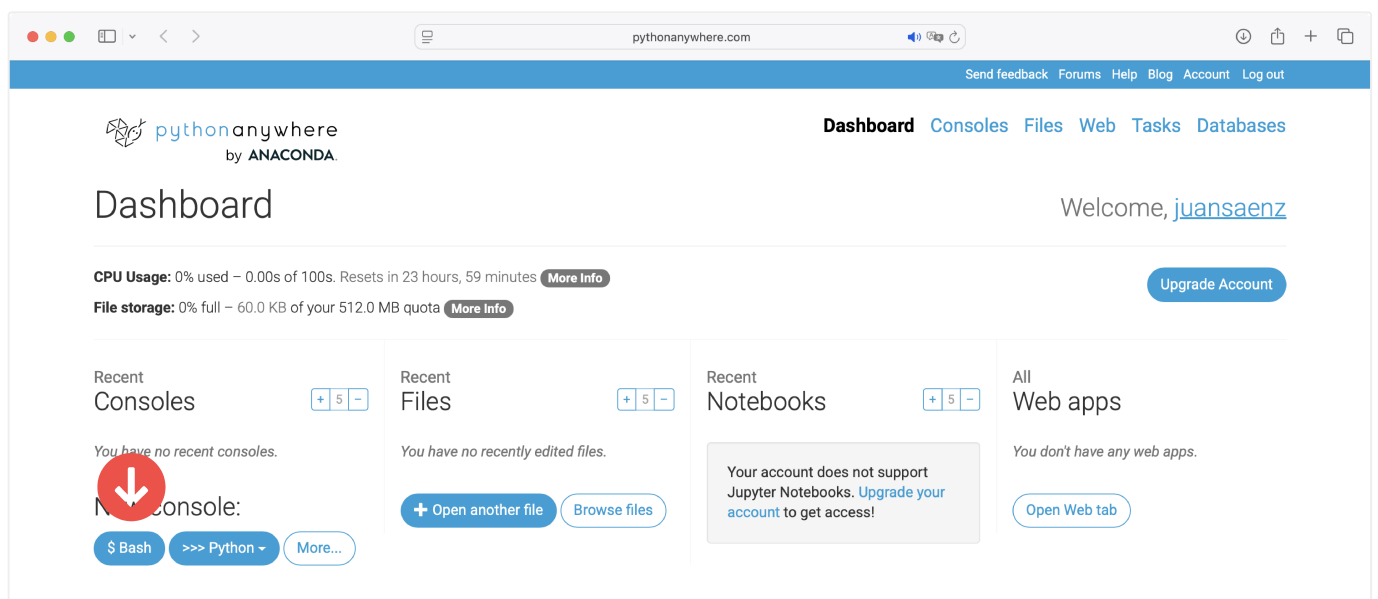


Figura 3: Creazione di una nuova console Bash.

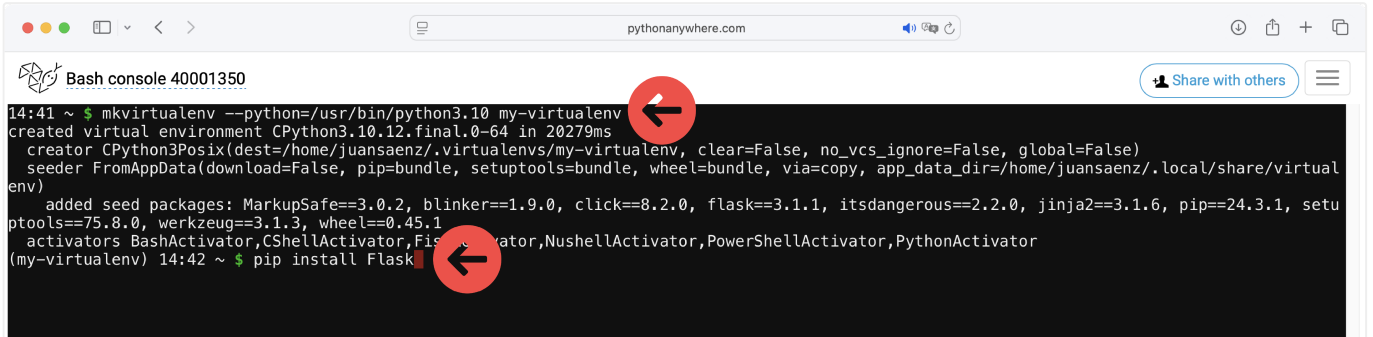
4. Dalla stessa console, creare un nuovo ambiente virtuale eseguendo i seguenti comandi:

```
mkvirtualenv --python=/usr/bin/python3.10 my-virtualenv
pip install flask
```

Il prompt visualizzato sulla console dovrebbe a questo punto passare da `$` a `(my-virtualenv) $`: è così che si capisce che il `virtualenv` è attivo.

NOTA: Ogni volta che si vuole lavorare sul progetto nella console, bisogna assicurarsi che il virtualenv sia attivo. È possibile riattivarlo in un secondo momento con `workon my-virtualenv`

5. Tramite la console, installa tutte le dipendenze utilizzate dal tuo social network attraverso il comando pip. Tipicamente, `Flask`, `flask-login`, `werkzeug` e `Pillow`.



The screenshot shows a terminal window titled "Bash console 40001350" on the pythonanywhere.com website. The terminal output shows the command `mkvirtualenv --python=/usr/bin/python3.10 my-virtualenv` being executed, which creates a new virtual environment. The output lists the added seed packages: MarkupSafe, blinker, click, flask, itsdangerous, Jinja2, pip, setuptools, Werkzeug, and wheel. The terminal then shows the command `pip install Flask` being executed. Two red arrows point to the `my-virtualenv` and `Flask` in the terminal output.

Figura 4: Creazione virtual environment dalla console e installazione delle dipendenze.

6. Dalla dashboard personale, apri la sezione dedicata alle “**Web apps**” (raggiungibile direttamente da questo link: https://www.pythonanywhere.com/web_app_setup).
7. Clicca su “**Add a new web app**” e segui i passaggi, avendo cura di selezionare “**Manual configuration**” quando è richiesto di selezionare il framework Python desiderato.

NOTA: Assicurati di scegliere la stessa versione di Python utilizzata nell’ambiente virtuale creato al punto 4 (cioè la 3.10).

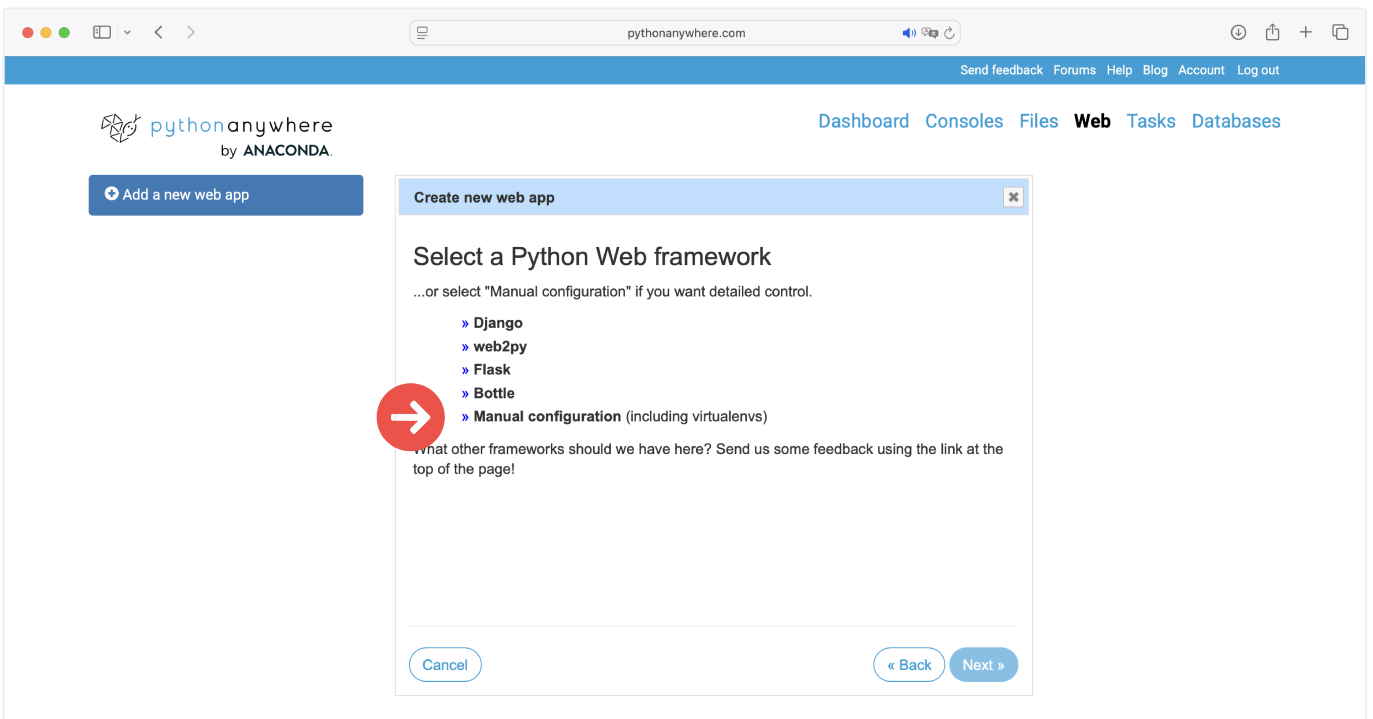


Figura 5: Menu per la creazione di una nuova Web app.

8. Nella pagina che si apre, vai alla sezione **“Code”** e inserisci il path per il codice sorgente caricato sulla piattaforma (esempio: `/home/[username]/mysite`, dove `mysite` è il nome della cartella del progetto che è stata compressa. Se invece non è presente nessuna cartella “contenitore” che racchiude i file di progetto, il path sarà semplicemente `/home/[username]/`).
9. Nella sezione **“Virtualenv”**, invece, inserisci il nome del virtual environment creato al punto 4 (`my-virtualenv`, se hai seguito fedelmente le istruzioni precedenti).

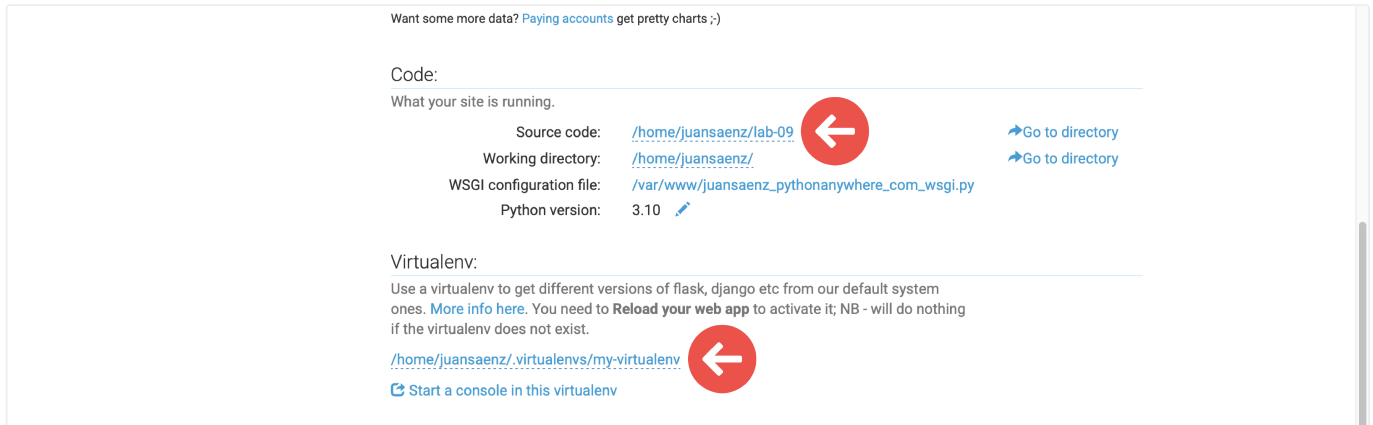


Figura 6: Impostazione del codice e del virtual environment.

10. Dalla stessa sezione **“Code”**, apri il file di configurazione WSGI cliccando sull’apposito link. Nel file WSGI che si apre, vai alla sezione Flask e decommentala, rendendola simile a questa:

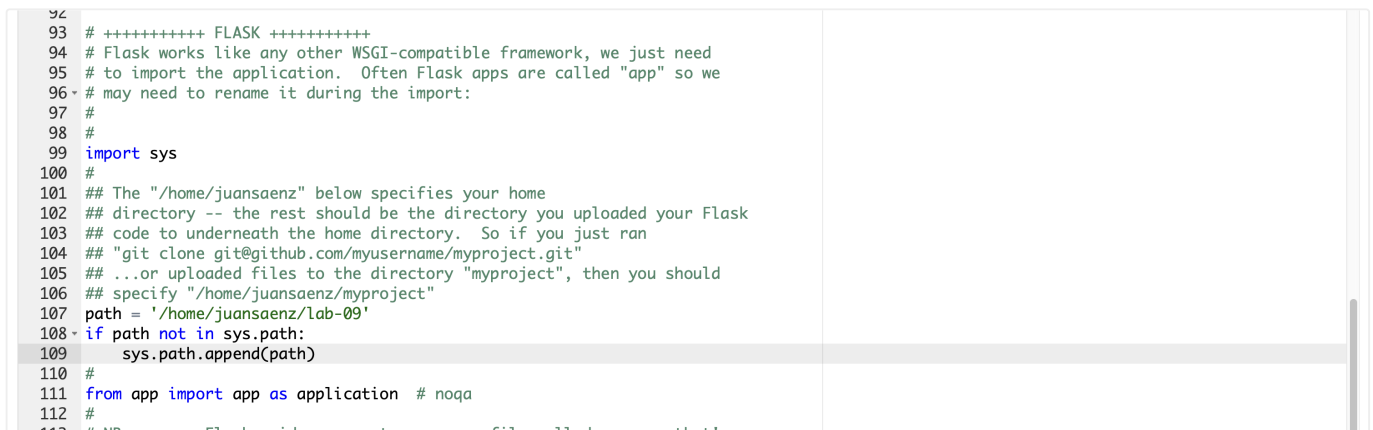


Figura 7: File di configurazione WSGI.

11. Modifica la riga 111 del file WSGI (**Figura 7**). Nota che il primo termine `“app”` corrisponde al nome del file principale del tuo progetto (ad esempio, `app.py`), mentre il secondo termine `“app”` si riferisce al nome della variabile dell'applicazione (come in `app = Flask(__name__)`).
12. Poiché il progetto ora viene eseguito da un percorso diverso, devi aggiornare i percorsi della cartella static e del database all'interno del tuo file `app.py` e dei file DAO in modo che siano accessibili correttamente. Per farlo, torna alla pagina dedicata alle **“Web apps”** e nella sezione Code apri la source code cliccando su **“Go to directory”**.

13. Modifica il percorso all'interno dei file aggiungendo “mysite/” all'inizio, dove “mysite” rappresenta sempre la cartella principale del progetto. Ad esempio, con la soluzione del laboratorio 9, nelle funzioni responsabili della gestione delle immagini, è stato inserito il prefisso “lab-9/” come mostrato nella **Figura 8** (riga 92). Allo stesso modo, nei file DAO, è stato necessario adattare il percorso al database, poiché il file SQLite era situato nella cartella “db”, come mostrato in **Figura 9** (riga 6).

```
82     size = POST_IMG_WIDTH, new_height
83     img.thumbnail(size, Image.Resampling.LANCZOS)
84
85     # Extracting file extension from the image filename
86     ext = post_image.filename.split(".")[-1]
87     # Getting the current timestamp in seconds
88     secondi = int(datetime.now().timestamp())
89
90     # Saving the image with a unique filename in the 'static' directory
91     img.save(
92         "lab-09/static/@" + current_user.nickname.lower() + "-" + str(secondi) + "." + ext
93     )
94
95     # Updating the 'immagine_post' field in the post dictionary with the image filename
96     post["immagine_post"] = (
97         "@" + current_user.nickname.lower() + "-" + str(secondi) + "." + ext
98     )
99
100     post["id_utente"] = int(current_user.id)
101     success = posts_dao.add_post(post)
102
```

Figura 8: Esempio di modifica del percorso della cartella static (app.py).

```
1 import sqlite3
2
3 # Operazioni sui Post
4
5 def get_posts():
6     conn = sqlite3.connect('lab-09/db/social_network.db')
7     conn.row_factory = sqlite3.Row
8     cursor = conn.cursor()
9
10     sql = 'SELECT posts.id, posts.data_pubblicazione, posts.testo, posts.immagine_post, utenti.nickname, utenti.immagine_profilo FROM post'
11     cursor.execute(sql)
12     posts = cursor.fetchall()
13
14     cursor.close()
15     conn.close()
16
17     return posts
18
```

Figura 9: Esempio di modifica del percorso della cartella db (posts_dao.py).

14. Se tutti i passaggi sono stati eseguiti correttamente, il social network è ora accessibile all'indirizzo `http://[username].pythonanywhere.com/`, ad esempio <http://juansaenz.pythonanywhere.com>.

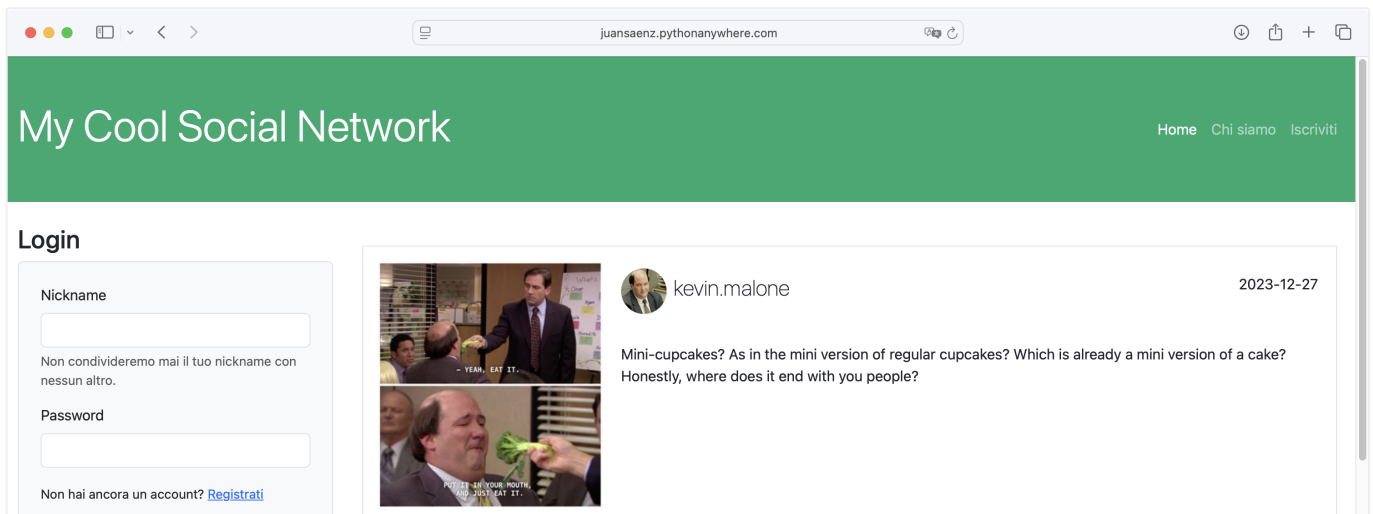


Figura 10: Deploy riuscito del social network su PythonAnywhere.

Note

1. Se al momento dell'accesso al sito ci fossero errori, dovresti consultare il registro degli errori (**Error log**) accessibile dalla pagina dedicata all'applicazione, nella sezione **Log files**, come indicato nella **Figura 11**.

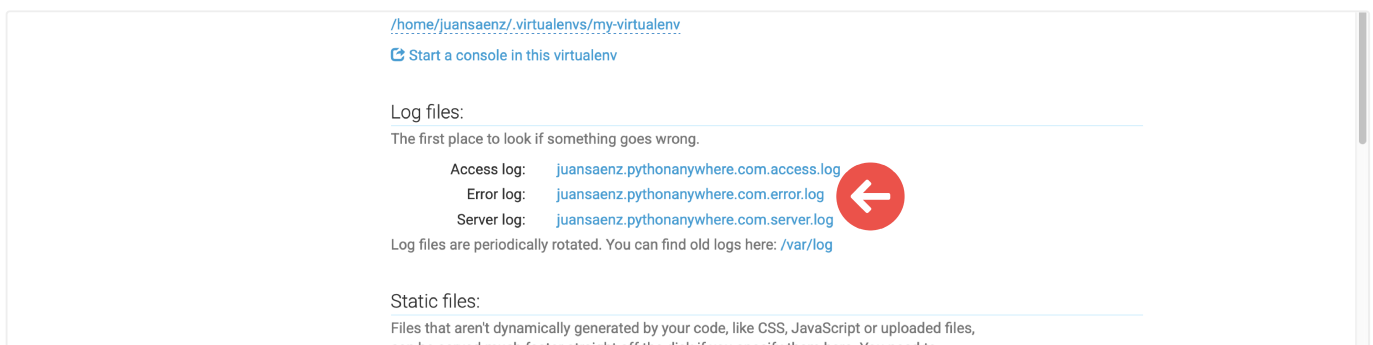


Figura 11: Registro degli errori.

2. Allo stesso modo, se apporti modifiche al codice, devi salvarlo e poi fare clic sul pulsante per ricaricare l'applicazione (**Figura 12**).

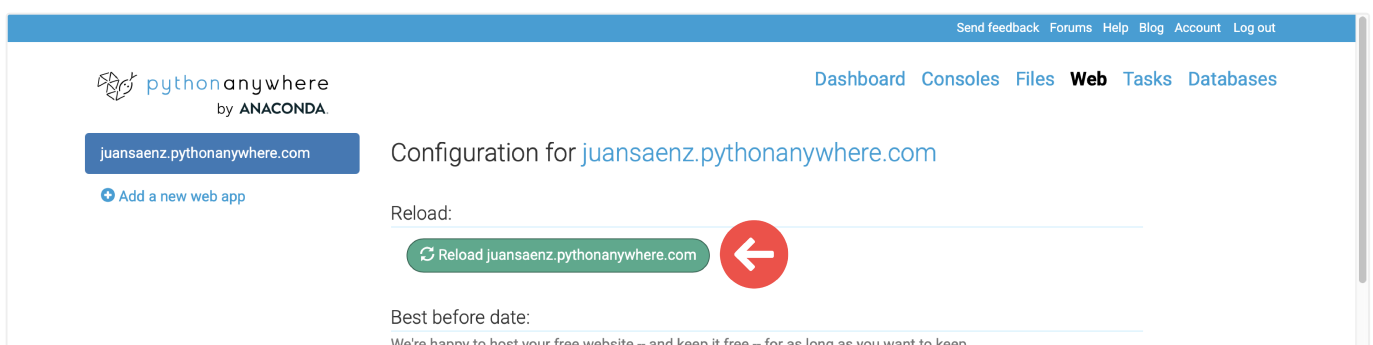


Figura 12: Ricarica dell'applicazione.

