



1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Università degli Studi di Urbino Carlo Bo

Applicazione Web

Reti di Calcolatori

Michelangelo Ungolo
Matricola: 313188

Anno Accademico: 2023/2024

Docente del Corso: Antonio Della Selva

Indice

1	Titolo	2
2	Introduzione	2
2.1	Configurazione ed esecuzione	2
3	Discussione critica della tematica	3
3.1	Passi e casi d'uso	3
3.2	Progettazione	3
3.3	Autenticazione - primo fattore	3
3.4	Autenticazione - secondo fattore	3
3.5	Download	4
3.6	Implementazione primo fattore	4
3.7	Implementazione secondo fattore	5
3.8	Implementazione download	7
3.9	Test e Validazione	8
4	Conclusione	10

1 Titolo

Sviluppo di un'applicazione web basata su un'architettura client-server che consente ai client di accedere, previa registrazione e autenticazione a due fattori, ad un cloud in sola lettura per il download dei file presenti.

2 Introduzione

Il progetto sviluppato è un'applicazione web realizzata con Django un framework del linguaggio Python. Inoltre sono state utilizzate librerie per l'implementazione del secondo fattore di autenticazione, in particolare per la generazione di codici OTP e per l'invio di questi ultimi tramite email. Come presente in specifica, il progetto permette l'accesso in lettura ad un file system virtuale garantendo agli utenti in sessione di effettuare il download dei file disponibili. L'obiettivo principale del progetto è quello di sviluppare un corretto sistema di autenticazione a due fattori mettendo in pratica le conoscenze teoriche apprese durante il corso di Reti di Calcolatori. Come obiettivo secondario, si punta a realizzare un progetto scalabile, ipotizzando un futuro accesso in scrittura con conseguente spazio cloud personalizzato per i singoli utenti, costruendo un'architettura solida ideale per l'implementazione di nuove funzionalità.

2.1 Configurazione ed esecuzione

Per eseguire sulla propria macchina locale il seguente progetto, dopo il download e il salvataggio del repository contenente il progetto, sono necessari diversi comandi da linea di comando come elencato di seguito:

1. `cd path\`
2. `python -m venv myenv`
3. `source myenv\bin\activate` [da Unix/MacOS]
4. `.\myenv\Scripts\activate` [da Windows]
5. `pip install -r requirements.txt`
6. `cd EliteCloud\`
7. `python manage.py migrate`
8. `python manage.py runserver`

All'avvio del server locale, Django fornirà l'URL necessario per l'accesso all'applicativo nel web, indicando generalmente l'indirizzo *localhost* : 8000. Da qui, il debug Django mostrerà la presenza di due applicazioni: 'admin' per la gestione del database e elite-cloud che è l'applicazione principale. Per accedere quindi al core del progetto visitare l'indirizzo *localhost* : 8000/*elite - cloud*.

3 Discussione critica della tematica

Ricercando l'indirizzo `localhost:8000/elite-cloud` si può notare una risposta 30X, indicando un reindirizzamento della pagina da parte del server. Infatti la logica dell'applicazione mostrerà fin da subito la pagina di accesso all'utente poichè tutte le operazioni saranno possibili esclusivamente dopo l'autenticazione.

3.1 Passi e casi d'uso

All'avvio l'utente potrà autenticarsi utilizzando le proprie credenziali fornite a tempo di registrazione, oppure essere reindirizzato ad un processo di registrazione di un nuovo account. Verranno richieste le informazioni di base come username, password e email, per completare l'iter di accesso al proprio account. Dopo l'autenticazione l'utente sarà reindirizzato al cloud server dove potrà visualizzare e scaricare i file disponibili.

3.2 Progettazione

Durante la fase di progettazione è stato fondamentale raccogliere tutte le informazioni necessarie per il raggiungimento dell'obiettivo. In questa sezione quindi si definiscono le scelte e le strategie adottate per finalizzare gli obiettivi, astruendo dalle tecnologie utilizzate e che saranno analizzate successivamente.

3.3 Autenticazione - primo fattore

Il primo sistema di sicurezza è il criptaggio delle informazioni sensibili, come la password di un utente in quanto chiave fondamentale per l'autenticazione. La strategia implementativa prevede un processo di hashing, che trasforma una sequenza di caratteri alfanumerici (la password in chiaro) in un valore numerico intero (l'hash). Questo sistema nasconde la reale password originale, garantendone la sicurezza: infatti se un malintenzionato ottenesse l'hash, non potrebbe falsificare l'accesso senza conoscere la chiave di criptazione. Sono diversi gli algoritmi idonei; il target di sufficienza ricade in garanzie di resistenza verso gli attacchi di forza bruta.

3.4 Autenticazione - secondo fattore

Dopo aver progettato un sistema sicuro per la conservazione e la gestione della password, è opportuno definire le scelte relative al secondo fattore di autenticazione, quindi alla generazione del codice OTP e l'invio di quest'ultimo ad un recapito dell'utente. Il sistema di doppia autenticazione si basa su un principio molto semplice; nonostante la conservazione accurata della password, un attaccante potrebbe ottenere la chiave di accesso in svariati modi, quindi è necessario disporre di un secondo ostacolo che verifica l'intenzionalità e l'identità dell'utente. L'idea implementativa verte su un metodo poco complesso, che usufruisce dell'indirizzo mail per la ricezione di un token temporaneo, garantendo la volontà della reale persona all'autenticazione in piattaforma.

La generazione di codici OTP deve essere sicura, motivo per cui occorre definire strategie ben chiare seguendo degli standard. Da questo derivano le seguenti scelte:

- Standardizzazione, quindi si procede all'utilizzo di un algoritmo standard usato da molte implementazioni di 2FA, ovvero TOTP o Time Based One Time Password. Esso si basa su un timer di sincronizzazione per generare token randomici a scadenza.
- Temporalità, occorre definire un lasso di tempo breve per la validità del codice OTP per ridurre il rischio di accessi non autorizzati. Una scelta ottimale sarebbe la definizione di un arco temporale che non superi il minuto; una validità di 60 secondi è ritenuta sufficiente per la presa visione del codice e il suo inserimento.

Generato il token temporale, esso deve essere consegnato all'utente. Per ragioni pratiche, si opta per l'utilizzo dell'indirizzo di posta elettronica come sistema di comunicazione.

Tuttavia, è importante valutare attentamente la sicurezza di questo processo. La consegna del token tramite email non rappresenta il mezzo più sicuro tra quelli disponibili. Infatti, questo processo potrebbe essere aggirato da malintenzionati esperti. Dal punto di vista della sicurezza, i sistemi gestiti da app di autenticazione o i protocolli OAuth2, che si basano su un garante di terze parti per autenticare un'identità, sono più efficaci.

Nonostante ciò, considerando la semplicità del mezzo di trasmissione, l'invio tramite email rimane uno dei metodi più pratici. Non richiede servizi aggiuntivi ed è accessibile a un'utenza più ampia.

Per l'invio delle email da parte del server, si utilizza il protocollo SMTP (Simple Mail Transfer Protocol). Essendo uno standard riconosciuto e ampiamente utilizzato, permette di lavorare con diversi sistemi di posta elettronica, garantendo ampia compatibilità e semplicità.

3.5 Download

All'interno della propria sessione, sarà possibile visionare i file presenti messi a disposizione dal server. Tale funzionalità deve essere molto intuitiva e allo stesso tempo sicura, in quanto solo gli utenti autorizzati sono ammessi all'interno dello spazio drive. Ogni file viene individuato da un id, con il quale si può risalire all'univocità del file. Se il file è presente si potrà procedere al download.

3.6 Implementazione primo fattore

La registrazione degli utenti e la gestione delle password sono implementate utilizzando alcune delle funzionalità già esistenti nel framework Django. Le password degli utenti sono criptate utilizzando la funzione `make_password`, che applica l'algoritmo PBKDF2, garantendo così una robusta protezione.

```

1      # codice di views.py per la creazione di un account con
      password criptata
2      user = User.objects.create(
3          username = request.POST.get("username"),
4          name = request.POST.get("name"),
5          surname = request.POST.get("surname"),
6          password = make_password(request.POST.get("
          password"))),
7          email = request.POST.get("email"),
8      )
9      # save new User into DB

```

```

10         user.save()
11         messages.success(request, "Utente creato. Procedi
           ora con il login")
12
13         return redirect("login")

```

Nella fase di autenticazione occorre valutare se la password corrispondente allo username indicato è uguale a quella memorizzata in database. La verifica della password avviene tramite una funzione Django chiamata `check_password()` che permette di confrontare la password immessa dall'utente, con la password criptata originale.

```

1  # codice views.py per la gestione del login, in
   particolare viene
2  # definita la logica del primo fattore
3  if request.method == 'POST':
4      user = User.objects.filter(username=request.POST.get(
           "username")).first()
5
6      if user and check_password(request.POST.get("password"),
           user.password):
7          request.session["username"] = user.username
8          request.session["email"] = user.email
9          send_otp(request)
10         return redirect("otp")

```

3.7 Implementazione secondo fattore

Il codice otp è generato tramite un oggetto TOTP, presente nella libreria python pyotp che permette di generare un token con validità temporale. Questa informazione sarà salvata nella sessione associata: in questo modo il token creato potrà essere confrontato successivamente con quello inviato dall'utente per la verifica.

```

1  # service: generate otp code for authenticate 2FA login
2  def send_otp(request):
3      # create code
4      otp_obj = pyotp.TOTP(pyotp.random_base32(), interval =
           60)
5      # start validate
6      otp = otp_obj.now()
7
8      # save informations to session
9      request.session["otp_code"] = otp_obj.secret
10     valid_date = datetime.now() + timedelta(minutes = 1)
11     request.session["otp_date"] = str(valid_date)
12
13     send_email(otp, request.session["email"])

```

Il codice viene successivamente inviato utilizzando una libreria python basata sul protocollo SMTP. Questa libreria consente la connessione ad un server di posta, in questo caso ai server Gmail, in quanto l'indirizzo usato per l'invio del codice rientra in quel dominio. Si opta per una connessione sulla porta 587, comune per le connessioni SMTP con crittografia STARTTLS, necessaria per una comunicazione sicura tra due host. Viene utilizzato il comando 'ehlo' del protocollo SMTP che permette di eseguire il primo handshake della comunicazione. Il comando è un'estensione che permette al client di creare una sessione attiva con il server e abilitare il trasferimento dei dati per la posta elettronica. Di fondamentale importanza è il successivo comando starttls, che permette di inizializzare qualsiasi canale una connessione TLS (transport layer security), ovvero criptata; essa protegge la comunicazione tra client e server garantendo la cifratura del messaggio che sarà trasmesso. Completate tali procedure, sarà possibile accedere al servizio di posta e inviare il messaggio al client dell'applicazione.

```

1  def send_email(otp, to_add):
2      # port 587 is required for smtp
3      server_smtp = smtplib.SMTP("smtp.gmail.com", 587)
4      server_smtp.ehlo()
5
6      # start TLS secure connection
7      server_smtp.starttls()
8      # login to gmail account
9      server_smtp.login(configu.EMAIL_USER, configu.
      EMAIL_PASSWORD)
10
11     server_smtp.sendmail(configu.EMAIL_USER, to_add,
12                           "Subject: Codice OTP elite-cloud\n\n
      "
13                           f"Il tuo codice otp è '{otp}'\n"
14                           "Se non hai richiesto il seguente
      codice OTP da elite-cloud ignora
      questa mail.")

```

L'ultima implementazione che prevede la chiusura del servizio di autenticazione a due fattori è la verifica del codice OTP inviato dall'utente. Questo viene fatto tramite il riutilizzo della libreria 'pyotp' che controlla se il token da validare rientra nel tempo di validità stabilito in precedenza ed è la combinazione corretta. Se tutto ciò è conforme alle regole definite, l'utente è correttamente autenticato e potrà essere rediretto al file system remoto vero e proprio.

```

1  if request.method == 'POST':
2      otp_insert = request.POST["code"]
3
4      otp_valid_code = request.session["otp_code"]
5      otp_date = request.session["otp_date"]
6
7      if otp_valid_code and otp_date is not None:
8          otp_date = datetime.fromisoformat(otp_date)
9          if otp_date > datetime.now():

```

```

10         otp_obj = pyotp.TOTP(otp_valid_code, interval
11                               =60)
12         if otp_obj.verify(otp_insert):
13             del request.session["otp_code"], request.
                session["otp_date"]
                return redirect("cloud")

```

3.8 Implementazione download

Il download dei file avviene attraverso un'esplicita richiesta da parte dell'utente, che fornirà al server l'identificativo univoco del file scelto. Tramite l'id del file sarà possibile risalire al corretto documento in database e servirlo per il download. Si predispone il file con valori appropriati in modo tale che possa essere trasmesso in formato binario e per mostrare la finestra di download a seconda del browser utilizzato dall'utente.

```

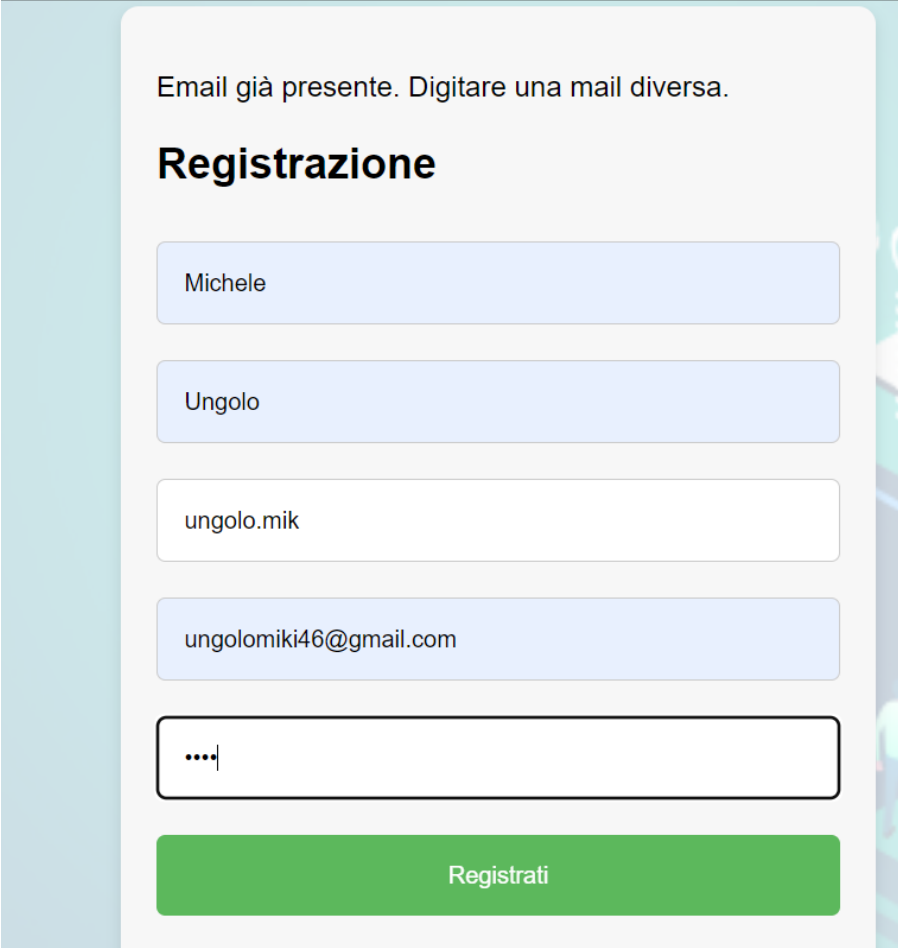
1 def download(request, file_id):
2     # get info of selected path and its path
3     file = get_object_or_404(File, pk = file_id)
4     file_path = file.file.path
5
6     # open file in read mode and assign Content for downlaod
7     response = FileResponse(open(file_path, "rb"))
8     response["Content-Type"] = "application/octet-stream"
9     response["Content-Disposition"] = f'attachment;␣filename
       ="{file.name}"'
10    return response

```


3.9 Test e Validazione

REGISTRAZIONE

Nello specifico si visualizza il controllo che viene fatto alla creazione dei nuovi utenti, testando la presenza di email o suername già presenti nel server.



Email già presente. Digitare una mail diversa.

Registrazione

Michele

Ungolo

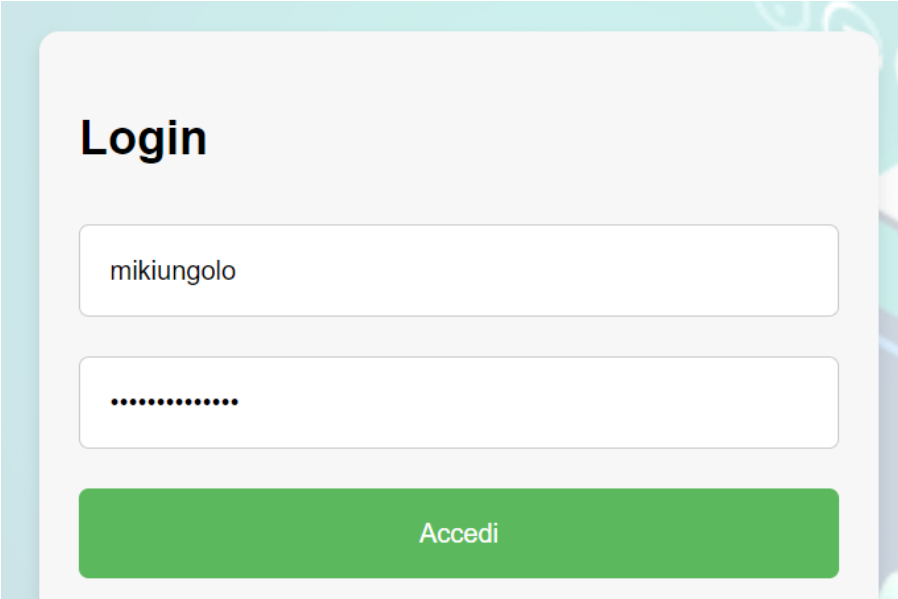
ungolo.mik

ungolomiki46@gmail.com

...

Registrati

LOGIN



Login

mikiungolo

.....

Accedi

VERIFICA DEL SECONDO FATTORE

Dopo il classico login sarà possibile reuperare il codice OTP all'indirizzo mail indicato durante la registrazione. Inserendo il token correttamente sarà possibile accedere crrettamente al servizio cloud.

A screenshot of a two-factor verification interface. It features a light blue background with a faint illustration of a person standing next to a large screen. In the center, there is a white rounded rectangle containing the title "Verifica a due fattori" in bold black text. Below the title, a message reads: "Inserisci il codice OTP ricevuto all'indirizzo email di registrazione." Underneath this message is a white input field with the placeholder text "OPT Code". At the bottom of the white rectangle is a green button with the text "Invia" in white.

Codice OTP elite-cloud

Posta in arrivo x



provamichelangelo2@gmail.com

a ▼

Il tuo codice otp e' 852256

Se non hai richiesto il seguente codice OTP da elite-cloud ignora questa mail.

CLOUD

Loggato come mikiungolo [Logout](#)

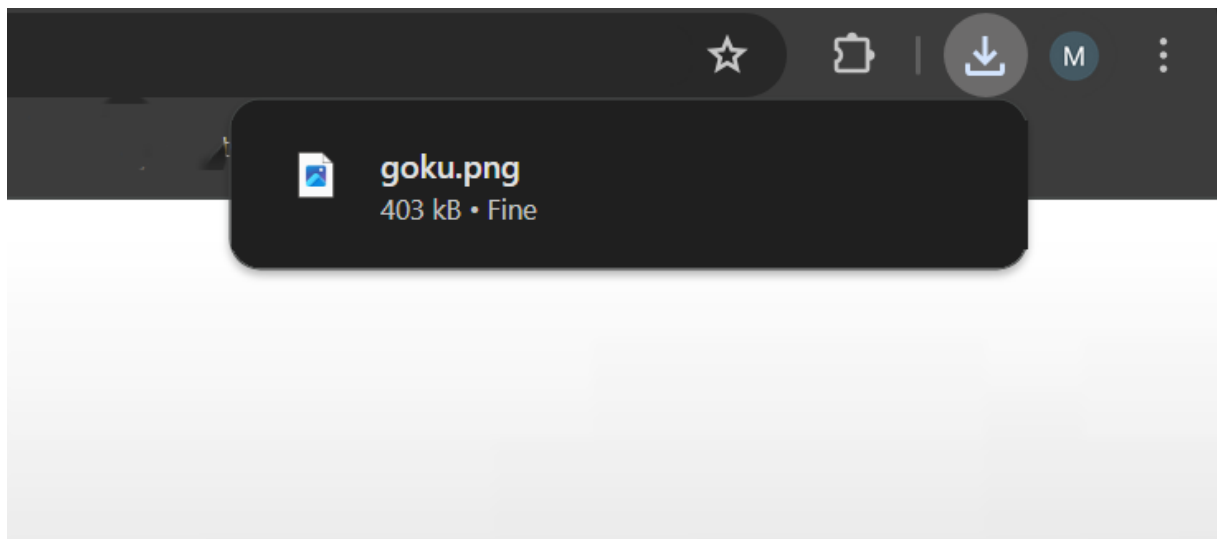


goku.png



harry.png

DOWNLOAD



4 Conclusione

Il progetto ha raggiunto con successo gli obiettivi iniziali, creando un'applicazione web sicura con autenticazione a due fattori. Attraverso le conoscenze sviluppate sui diversi protocolli e sulla sicurezza nelle reti, è stato possibile comprendere le fonti Django e di libreria e applicare le conoscenze al mio caso di studio. Nonostante un linguaggio e di un framework nuovi, il dominio applicativo ha alleggerito il carico di lavoro permettendo di realizzare le idee rapidamente. L'applicazione può essere estesa facilmente in futuro, trasformandola in un potenziale cloud di sistema, permettendo, a livello sperimentale, la gestione di un file system personale per ogni utente, incluso il supporto per l'upload.