

**MARINA  
MILITARE**



**UNIVERSITÀ  
DEGLI STUDI DI BARI  
ALDO MORO**

**MARINA MILITARE**

**Progetto di Reti di Calcolatori e Comunicazione Digitale**



**SaaS**

**Aggregatore Versioning  
Cloud SaaS**

**Università degli Studi di Bari Aldo Moro 2024/2025**

**STUDENTI :**

**COLAPIETRO GIANVITO PIO**

**STRADA DAVID BRYAN**

**VIENNA CHRISTIAN**

**CUGUTTU ALICE**

**RINALDI SIMONE**

## Sommario

GitHub .....	2
Introduzione .....	3
Dropbox .....	3
Google Drive .....	4
OneDrive.....	4
Obiettivo.....	5
Analisi Preliminare .....	6
Installazione delle dipendenze.....	8
Che cosa sono le 'API' .....	9
Configurazione delle API.....	9
Configurazione API di Dropbox .....	10
Configurazione API di Google Drive .....	11
Configurazione API di Microsoft OneDrive (Azure) .....	12
Ambiente di Sviluppo: Visual Studio .....	14
Produzione .....	16
Script.....	17
Esecuzione: .....	18
Struttura del Progetto.....	18
Interfaccia.....	21
Homepage.....	21
Configuratore .....	21
Risultati.....	22
Implementazioni .....	22
Google Drive Business .....	22
Box.....	23
Conclusione .....	24
Sitografia .....	26

## GitHub

Link : <https://github.com/ChrisViens/Aggregator-Versioning-Cloud-SaaS>

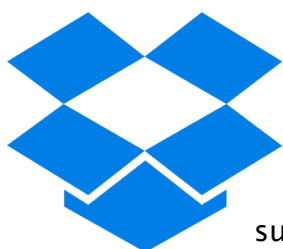
## Introduzione

Nell'era digitale, dove ogni processo aziendale e personale si basa su dati digitali, l'organizzazione e il controllo delle versioni dei file sono diventati fondamentali. Questo progetto introduce un'innovativa applicazione di aggregazione per il versioning dei file, progettata per semplificare e migliorare la gestione dei dati su piattaforme cloud SaaS come Dropbox, Google Drive e OneDrive.

### Caratteristiche principali:

- Controllo delle modifiche ai file.
- Interfaccia user-friendly per una gestione centralizzata.
- Sicurezza integrata per proteggere i dati sensibili degli utenti.

Questo strumento si adatta a una vasta gamma di esigenze, rendendolo utile sia per utenti singoli che per grandi organizzazioni.



### Dropbox

**Dropbox** è un servizio di file hosting gestito dalla società californiana Dropbox Inc., che offre cloud storage, sincronizzazione automatica dei file, cloud personale e software client. Dropbox si basa sul protocollo crittografico Secure Sockets Layer (SSL), i file

immagazzinati e accessibili tramite password vengono cifrati tramite AES con chiave a 256 bit. Il programma per usufruire del servizio, scaricabile gratuitamente, è disponibile per Windows, macOS, Linux, iOS, BlackBerry OS, Android, Windows RT e Windows Phone.

Dropbox utilizza un modello di business freemium, dove viene offerto un account gratuito con una capacità di 2 GB di base, estendibili, in vari modi, fino a 18 GB in totale. Ad esempio, si guadagnano 500 MB per ogni nuova persona invitata che si registri al sito e installi il software sul proprio computer. È possibile aumentare ulteriormente lo spazio gratuito collegando il proprio account ai social network (fino a 640 MB) oppure usando le versioni beta del programma (fino a 5 GB).

I piani tariffari a pagamento permettono di aumentare lo spazio fino a 1TB e di guadagnarne altro invitando nuove persone a utilizzare il servizio.

Il servizio può essere usato anche via web, caricando e visualizzando i file tramite il browser, oppure tramite il driver locale che sincronizza automaticamente una cartella locale del file system con quella condivisa, notificando le sue attività all'utente. L'interfaccia web consente il caricamento di file con dimensione massima pari a 300 MB ciascuno.



## Google Drive

Lanciato da Google nel 2012, **Google Drive** offre agli utenti uno spazio di archiviazione gratuito di 15 GB, condiviso tra Drive, Gmail e Google Foto. È accessibile tramite browser web e applicazioni dedicate per Windows, macOS, Android e iOS. Tra le sue funzionalità principali:

- **Integrazione con Google Workspace:** Drive si integra con applicazioni come Documenti, Fogli e Presentazioni, permettendo la creazione e modifica collaborativa di documenti in tempo reale.
- **Condivisione e collaborazione:** Gli utenti possono condividere file e cartelle con altri, impostando permessi specifici per la visualizzazione o modifica.
- **Sicurezza:** Google Drive utilizza protocolli di sicurezza avanzati per proteggere i dati degli utenti, inclusa la crittografia dei dati in transito e a riposo.
- **Piani di archiviazione:** Oltre ai 15 GB gratuiti, sono disponibili piani a pagamento tramite Google One, che offrono spazi di archiviazione maggiori, come 100 GB o 2 TB.



## OneDrive

**OneDrive** è il servizio di cloud storage di Microsoft, integrato con Windows e parte della suite Microsoft 365. Offre 5 GB di spazio di archiviazione gratuito, con la possibilità di espandere lo spazio tramite piani a pagamento.

Le sue caratteristiche includono:

- **Integrazione con Microsoft 365:** OneDrive si integra con applicazioni come Word, Excel e PowerPoint, facilitando la collaborazione e la modifica dei documenti direttamente dal cloud.
- **Sincronizzazione:** I file e le cartelle si sincronizzano automaticamente tra dispositivi, permettendo l'accesso ai dati da PC, Mac, tablet e smartphone.
- **Condivisione:** Gli utenti possono condividere facilmente file e cartelle con altri, impostando permessi di visualizzazione o modifica.
- **Sicurezza:** OneDrive protegge i file con crittografia e offre funzionalità come il "Personal Vault" per un ulteriore livello di sicurezza.
- **Piani di archiviazione:** Oltre ai 5 GB gratuiti, sono disponibili piani a pagamento che offrono fino a 1 TB di spazio, spesso inclusi negli abbonamenti a Microsoft 365.

## Obiettivo

L'obiettivo principale del progetto è creare un sistema scalabile e innovativo che semplifichi la gestione centralizzata delle versioni dei file su diverse piattaforme cloud come Dropbox, Google Drive e OneDrive. Questo strumento punta a risolvere le difficoltà legate alla frammentazione dei dati, offrendo agli utenti un'unica interfaccia per controllare e proteggere i propri file.

Il progetto è stato sviluppato con i seguenti intenti principali:

### 1. Accesso Centralizzato:

- Consentire agli utenti di accedere rapidamente a tutte le versioni dei file su più piattaforme cloud, eliminando la necessità di accedere separatamente a ciascun servizio.

### 2. Miglioramento della Sicurezza:

- Utilizzare protocolli di crittografia avanzati e autenticazione OAuth2 per garantire la protezione dei dati durante il trasferimento e la conservazione.

### 3. Ottimizzazione della Collaborazione:

- Offrire funzionalità che facilitino il lavoro di team distribuiti, permettendo di verificare il versionamento dei file e controllare l'ultimo accesso.

### 4. Interfaccia Intuitiva:

- Creare un sistema user-friendly, facilmente accessibile anche da utenti non esperti, con funzionalità chiare e un design semplice.

Questo progetto rappresenta una soluzione efficace per utenti singoli, aziende e team che necessitano di una gestione avanzata dei propri file su piattaforme cloud. La capacità di monitorare le versioni precedenti, combinata con un'interfaccia intuitiva e funzionalità di sicurezza avanzate, garantisce un notevole miglioramento nell'efficienza operativa e nella protezione dei dati.

L'aggregatore di versioning per piattaforme cloud è stato concepito per essere ampliabile, adattabile e capace di rispondere alle esigenze di un mondo digitale in continua evoluzione. Questo progetto segna un passo avanti nella semplificazione della gestione dei dati, migliorando la produttività e la sicurezza degli utenti.



## Analisi Preliminare

L'analisi preliminare del progetto si concentra sull'identificazione delle piattaforme cloud supportate, delle tecnologie da utilizzare e delle esigenze degli utenti. Di seguito, sono riportati i punti principali che hanno guidato lo sviluppo del progetto.

### 1. Identificazione delle Piattaforme Cloud:

- **Dropbox:** Utilizza l'API ufficiale per fornire funzionalità di versioning avanzate, inclusa la possibilità di accedere a tutte le revisioni di un file.
- **Google Drive:** Offre le **API Google Drive Activity** che permette una gestione dettagliata dei file, consentendo di monitorare e recuperare le versioni.
- **OneDrive:** Grazie a Microsoft Graph API, supporta il monitoraggio e la gestione avanzata dei file con un'attenzione particolare alla sicurezza e alle autorizzazioni.

## 2. Scelta delle Tecnologie:

- **Python:** Per il **backend**, grazie alla sua versatilità e all'ampio supporto per l'integrazione con API esterne.
- **Flask:** Utilizzato per sviluppare un **backend** leggero e scalabile per la gestione delle richieste RESTful.
- **JavaScript** – Per la gestione delle interazioni dinamiche nel frontend, come l'aggiornamento del configuratore e l'invio di richieste API.
- **HTML/CSS/Bootstrap:** Garantiscono un'interfaccia (**frontend**) reattiva e semplice da usare, migliorando l'interazione utente.

## 3. Sicurezza:

- L'autenticazione **OAuth2** è stata implementata per assicurare che solo utenti autorizzati possano accedere ai dati.

## 4. Esigenze degli Utenti:

- **Facilità d'Uso:** L'interfaccia deve essere chiara e intuitiva, consentendo anche agli utenti meno esperti di gestire facilmente i file.
- **Efficienza:** Gli utenti necessitano di tempi di risposta rapidi per il recupero delle versioni dei file.
- **Flessibilità:** Il sistema deve essere in grado di adattarsi a diverse esigenze, come la gestione di più account o l'integrazione con ulteriori piattaforme in futuro.

## 5. Obiettivi Tecnici:

- Creare un'architettura modulare che permetta di estendere facilmente le funzionalità e di integrare nuove piattaforme cloud.
- Ottimizzare le richieste API per minimizzare i tempi di attesa e migliorare l'efficienza del sistema.
- Assicurare la scalabilità per supportare un numero crescente di utenti e dati senza compromettere le prestazioni.

## Installazione delle dipendenze

Per configurare e avviare il progetto, seguire questi passaggi:

### 1. Creazione di un Ambiente Virtuale

- Linux/macOS:

```
python3 -m venv env
```

```
source env/bin/activate
```

- Windows:

```
python -m venv env
```

```
.\env\Scripts\activate
```

### 2. Installazione delle Librerie Necessarie

Utilizzare il comando seguente per installare le librerie richieste:

```
pip install flask dropbox google-api-python-client msal
```

```
pip install flask-socketio
```

```
pip install requests
```

```
pip install dropbox
```

```
pip install google-api-python-client google-auth google-auth-oauthlib
```

```
pip install azure-identity msal msgraph-core
```

### 3. Configurazione delle API

- Dropbox:** Ottieni un Access Token dalla console sviluppatori di Dropbox.
- Google Drive:** Genera una chiave API tramite Google Cloud Console e scarica il file **secret.json**.
- OneDrive:** Registra un'applicazione su Azure Portal per ottenere **client\_id**, **client\_secret** e **tenant\_id**.

### 4. Avvio del Server

Dopo aver configurato le credenziali, avvia il server utilizzando:

```
python app.py
```

### 5. Test delle Funzionalità

Una volta avviato il server, testare gli endpoint API tramite **Postman** o **curl** per verificare l'integrazione con le piattaforme cloud.



## Che cosa sono le 'API

Le **API (Application Programming Interfaces)** sono un insieme di regole e strumenti che permettono a diversi software di comunicare tra loro. Funzionano come un ponte, consentendo a un'applicazione di accedere a funzionalità o dati di un'altra senza dover conoscere i dettagli del funzionamento interno. Ad esempio, le API permettono a un'app di mostrare una mappa utilizzando il servizio di Google Maps. Possono essere utilizzate per integrare servizi, automatizzare processi o estendere le capacità di un software. Sono fondamentali nello sviluppo moderno per costruire applicazioni modulari e interoperabili.

## Configurazione delle API

Per utilizzare l'applicazione e accedere ai dati su Dropbox, Google Drive e OneDrive, è necessario configurare le rispettive API.

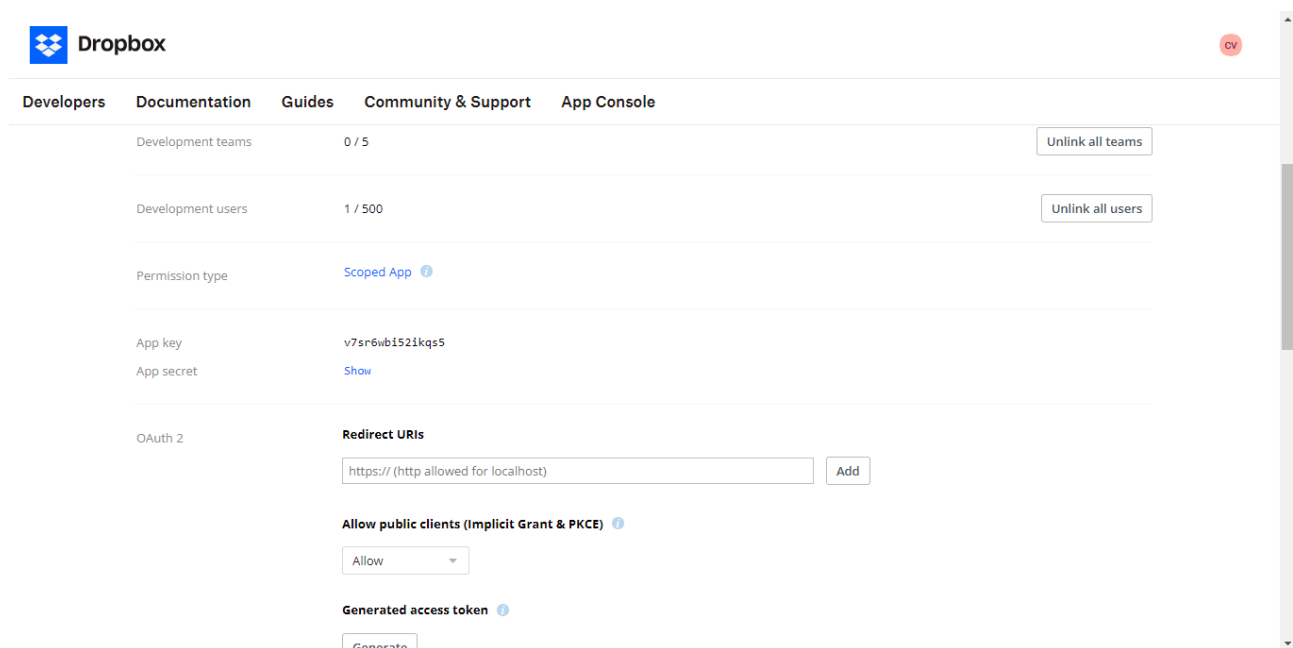
La configurazione delle API di OneDrive, Google Drive Activity e Dropbox richiede particolare attenzione alla sicurezza e alla gestione delle chiavi di autenticazione. Per OneDrive, è necessario registrare un'applicazione nel portale Azure, generando un Client ID e un Client Secret, con i permessi API specifici (Scopes) come Files.Read, Files.ReadWrite e User.Read. Questi dati devono essere protetti, utilizzando file di configurazione sicuri o sistemi di gestione segreti.

Google Drive Activity richiede la creazione di credenziali OAuth 2.0 nella Google Cloud Console, generando un file secret.json che include diversi parametri tra cui il Client ID e il Client Secret. È fondamentale conservare questo file in un ambiente protetto e limitare gli ambiti (scopes) di accesso solo a quelli strettamente necessari, come drive.activity.readonly.

Per Dropbox, si deve creare un'app nella console sviluppatori per ottenere l'Access Token oltre all'App\_Key e all'App\_Secret. Questo token fornisce accesso diretto all'account e deve essere trattato come una chiave privata, evitando di esporlo nel codice o nei repository. In tutte le configurazioni, è consigliato utilizzare protocolli di crittografia TLS per proteggere i dati in transito e monitorare regolarmente l'uso delle chiavi per rilevare attività sospette.

## Configurazione API di Dropbox

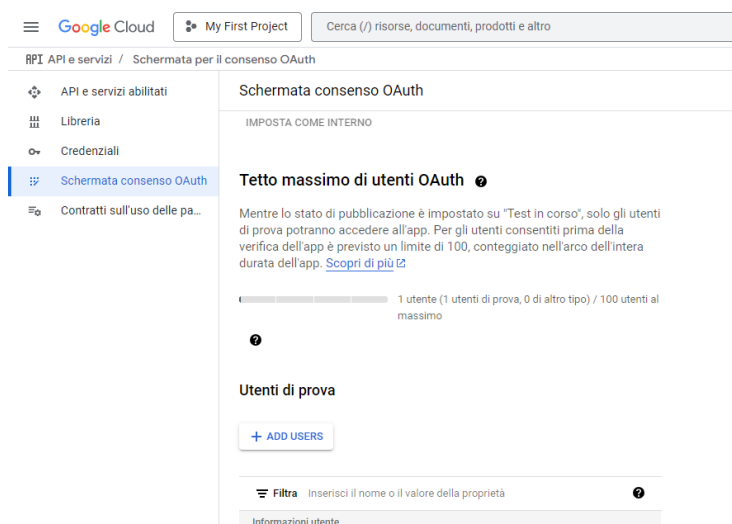
1. Accedi alla [console sviluppatori di Dropbox](#).
2. Crea una nuova applicazione:
  - Seleziona il pulsante "Create App".
  - Scegli "Choose an API" e seleziona "Full Dropbox" come tipo di accesso.
  - Inserisci un nome univoco per l'applicazione.
3. Configura i permessi:
  - Dopo aver creato l'app, vai alla sezione "Permissions".
  - Abilitare *"enable additional teams"* e *"enable additional users"*.
  -
4. Genera un Access Token:
  - Vai alla sezione "Settings".
  - Clicca su "Generate" accanto alla voce "OAuth 2.0 Access Token".
  - Memorizza l'App Key tramite il Configuratore.
  - Memorizza l'App Secret tramite il Configuratore.
5. Salva il token per poi inserirlo manualmente nel configuratore.



The screenshot shows the Dropbox App Console interface. At the top, there's a navigation bar with the Dropbox logo and the text "Dropbox". Below this, there's a horizontal menu with links: "Developers", "Documentation", "Guides", "Community & Support", and "App Console". The main content area is divided into several sections. The first section is "Development teams" with a count of "0 / 5" and a button "Unlink all teams". The second section is "Development users" with a count of "1 / 500" and a button "Unlink all users". The third section is "Permission type" with a link "Scoped App" and an information icon. The fourth section is "App key" with the value "v7sr6wb1521kqs5" and a "Show" link. The fifth section is "App secret" with a "Show" link. The sixth section is "OAuth 2" with a "Redirect URIs" section containing a text input field with the value "https:// (http allowed for localhost)" and an "Add" button. Below this is a section "Allow public clients (Implicit Grant & PKCE)" with a dropdown menu set to "Allow". At the bottom, there's a section "Generated access token" with a "Generate" button.

## Configurazione API di Google Drive

1. Accedi alla [Google Cloud Console](#).
2. Crea un nuovo progetto:
  - Vai su "Select Project" e clicca su "New Project".
  - Assegna un nome al progetto e clicca su "Create".
3. Abilita l'API di Google Drive:
  - Vai su "API & Services > Library".
  - Cerca "Google Drive Activity API" e clicca su "Enable".
4. Crea le credenziali:
  - Vai su "API & Services > Credentials".
  - Clicca su "Create Credentials" e seleziona "OAuth Client ID".
  - Configura la schermata di consenso OAuth seguendo le istruzioni.
  - Aggiungi gli ambiti :  
`/auth/drive.file – /auth/drive.readonly – /auth/drive.metadata.readonly`
  - Seleziona "Desktop app" come tipo di applicazione.
  - Da “Credenziali” andare a “URI di reindirizzamento autorizzati”.
  - Inserire <http://localhost:8080/> e cliccare su “AGGIUNGI URI”.
  - Scarica il file secret.json.
  - Aggiungere un utente di prova da “Schermata di consenso OAuth”, poi “Utenti di prova”->”Add Users”.



5. Salva il file secret.json in una directory sicura e caricalo successivamente tramite il configuratore.

## Configurazione API di Microsoft OneDrive (Azure)

1. Accedi al [Microsoft Azure Portal](#).
2. Registra una nuova applicazione:
  - Vai su "Azure Active Directory > App Registrations".
  - Clicca su "New Registration".
  - Assegna un nome all'app e seleziona "Accounts in any organizational directory".
  - Clicca su "Register".
3. Configura i permessi API:
  - Vai alla sezione "API Permissions".
  - Clicca su "Add a permission" e seleziona "Microsoft Graph".
  - Inserire i seguenti Scopes sia per "Autorizzazioni Delegate" che per "Autorizzazioni Applicazione": Files.Read – Files.ReadWrite – User.Read – User.ReadWrite – openid

#### 4. Autenticazione :

- Andare su “Gestione” -> “Autenticazione” -> “Consenti i flussi client pubblici” e selezionare “SI”

#### 5. Consenso :

- Andare su “API Permissions” e cliccare su “Concedere consenso Amministratore per Default Directory”

#### 6. Recupera i dettagli dell'app:

- Vai su "Overview" e copia Application (client) ID e Directory (tenant) ID.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with the Microsoft Azure logo, a search bar, and user information. The main content area is titled 'OneDrive API App'. On the left, there's a sidebar with navigation options like 'Panoramica', 'Avvio rapido', 'Assistente all'integrazione', 'Diagnostica e risolvi i problemi', 'Gestione', and 'Supporto e risoluzione dei problemi'. The main content area is divided into sections: 'Informazioni di base' (Basic Information) and 'Attività iniziali' (Getting started). The 'Informazioni di base' section contains a table with the following data:

Nome visualizzato	Credenziali client
OneDrive API App	<a href="#">Aggiungi un certificato o un segreto</a>
ID applicazione (client) : c4d8190a-f92d-4b14-92cf-d9537791f1cd	URI di reindirizzamento : <a href="#">0 per Web, 0 per applicazione a pagina singola...</a>
ID oggetto : 5a058104-5253-4b19-8337-60ad5af34e3e	URI dell'ID applicazione : <a href="#">Aggiungi un URI ID applicazione</a>
ID della directory (tenant) : 511cdae7-25f8-41c1-847d-c396be998ca6	Applicazione gestita nell... : <a href="#">OneDrive API App</a>
Tipi di account supportati : <a href="#">Tutti gli utenti di account Microsoft</a>	

Below the table, there's a section titled 'Creazione dell'applicazione con Microsoft Identity Platform'. It contains a paragraph explaining that Microsoft Identity Platform is a service for authentication and authorization, and it's possible to create modern authentication solutions based on standards, access and protect APIs, and add access for users and clients. A link 'Altre informazioni' is provided. At the bottom, there are several icons representing different services and tools.

# Ambiente di Sviluppo: Visual Studio

## Prerequisiti

- Python
- Pip

Visual Studio Code (VS Code) è un editor di codice sorgente potente, flessibile e gratuito, ideale per lo sviluppo di progetti come il nostro aggregatore di versioning di piattaforme cloud SaaS. Grazie alle sue estensioni e funzionalità integrate, VS Code offre un ambiente completo per sviluppare, testare e gestire il progetto in modo efficiente.

## Installazione di Visual Studio Code

1. **Scaricare VS Code:** Visita il sito ufficiale [Visual Studio Code](#) e scarica l'ultima versione disponibile per il tuo sistema operativo (Windows, macOS, Linux).
2. **Installazione:** Segui le istruzioni fornite dall'installer per completare l'installazione.
3. **Configurazione Iniziale:**
  - Avvia Visual Studio Code.
  - Installa le estensioni suggerite per il progetto.

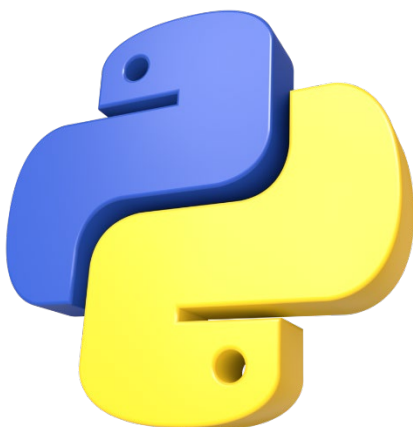
## Configurazione dell'Ambiente in VS Code

1. **Aprire la Cartella del Progetto:**
  - In VS Code, seleziona File > Open Folder e scegli la directory principale del progetto.
2. **Estensioni Consigliate:**
  - **Python:** Per il supporto di debugging e suggerimenti sul codice Python.
  - **Flask :** Per una scrittura più veloce e precisa del codice Flask.
  - **HTML :** Per suggerimenti e correzione automatica del codice.
3. **Impostare l'Ambiente Virtuale:**
  - Dopo aver creato l'ambiente virtuale (`python -m venv env`), configura VS Code per utilizzarlo:
    1. Premi `Ctrl+Shift+P` o `Cmd+Shift+P` su macOS.
    2. Cerca Python: Select Interpreter.
    3. Seleziona l'interprete corrispondente al tuo ambiente virtuale.

## Vantaggi dell'Uso di VS Code

- **Produttività Aumentata:** Grazie a strumenti come IntelliSense, linting e snippet.
- **Debugging Potente:** Debug visivo e console integrata per un controllo dettagliato.
- **Integrazione Completa:** Supporto per Git, Docker e altri strumenti di sviluppo.
- **Leggero e Veloce:** Ottimizzato per essere reattivo anche su macchine con risorse limitate.

Visual Studio Code è uno strumento essenziale per lo sviluppo di questo progetto, fornendo un ambiente completo e personalizzabile che semplifica ogni fase del processo di sviluppo. La sua flessibilità lo rende ideale per gestire sia il backend che il frontend del sistema, migliorando l'efficienza e la qualità del codice.



Flask

# Produzione

Nelle seguenti immagini è possibile visualizzare parte del codice HTML, CSS e JavaScript.

## Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document Version Aggregator</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha/dist/css/bootstrap.min.css" rel="stylesheet">
  <!-- Bootstrap JavaScript -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha/dist/js/bootstrap.bundle.min.js"></script>
  <script src="https://cdn.socket.io/4.4.1/socket.io.min.js"></script>
  <style>
    .action-btn {
      position: relative;
      transition: transform 0.2s ease, opacity 0.2s ease, background-color 0.3s ease;
    }

    .action-btn:active {
      transform: scale(0.95); /* Effetto di compressione */
      background-color: rgba(0, 0, 0, 0.1); /* Leggera ombreggiatura */
    }

    .action-btn:disabled {
      opacity: 0.6;
      cursor: not-allowed;
    }

    .icon-container {
      display: flex;
      justify-content: center;
      align-items: center;
      margin-bottom: 20px; /* Spaziatura tra l'icona e il titolo */
    }
  </style>
</head>
```

```
const socket = io(); // Inizializza la connessione WebSocket

socket.on('output', (data) => {
  try {
    const parsedData = data.data;

    if (!parsedData || typeof parsedData !== 'object') {
      throw new Error("Formato dei dati non valido");
    }

    // Gestione dei vari stadi
    if (parsedData.stage === 1 || parsedData.stage === 2) {
      showPopup(parsedData.message || "Messaggio non disponibile");
    } else if (parsedData.stage === 3) {
      showLoading(false);
      const tableBody = document.getElementById('outputTable');
      tableBody.innerHTML = ''; // Svuota la tabella

      if (!parsedData.files || parsedData.files.length === 0) {
        const noDataRow = `<tr><td colspan="5" class="text-center">Nessun file trovato</td></tr>`;
        tableBody.innerHTML = noDataRow;
        return;
      }

      parsedData.files.forEach(file => {
        file.versions.forEach(version => {
          const row = `<tr>
            <td>${file.path || file.name || "N/A"}</td>
            <td>${version.author || file.author || "N/A"}</td>
            <td>${version.lastModified || "N/A"}</td>
            <td>${version.size || "N/A"} bytes</td>
          </tr>`;
          tableBody.innerHTML += row;
        });
      });

      if (activeButtonId) {
        toggleButtonState(activeButtonId, false);
        activeButtonId = null; // Resetta la variabile
      }
    } else if (parsedData.stage === "error") {
      showPopup("Errore: ${parsedData.message}");
    } else {
      console.warn("Stadio sconosciuto:", parsedData.stage);
    }
  } catch (error) {
    console.error("Errore durante l'elaborazione dei dati:", error);
    alert("Errore nella risposta ricevuta dal server.");
    if (activeButtonId) {
      toggleButtonState(activeButtonId, false);
      activeButtonId = null; // Resetta la variabile
    }
  }
});

let currentConfig = {};
```

```
async function fetchGDriveData(buttonId) {
  try {
    toggleButtonState(buttonId, true);
    showLoading(true);
    const response = await fetch('/extract/gdrive', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(currentConfig)
    });
    const data = response.json();

    if (data.error) {
      showPopup(data.error);
    } else {
      const tableBody = document.getElementById('outputTable');
      tableBody.innerHTML = ''; // Svuota la tabella

      if (!data.files || data.files.length === 0) {
        const noDataRow = `<tr><td colspan="5" class="text-center">Nessun file trovato</td></tr>`;
        tableBody.innerHTML = noDataRow;
      } else {
        data.files.forEach(file => {
          file.versions.forEach(version => {
            const row = `<tr>
              <td>${file.path || file.name || "N/A"}</td>
              <td>${version.author || file.author || "N/A"}</td>
              <td>${version.lastModified || "N/A"}</td>
              <td>${version.size || "N/A"} bytes</td>
            </tr>`;
            tableBody.innerHTML += row;
          });
        });

        if (activeButtonId) {
          toggleButtonState(activeButtonId, false);
          activeButtonId = null; // Resetta la variabile
        }
      }
    }
  } catch (error) {
    console.error("Errore durante l'elaborazione dei dati:", error);
    alert("Errore nella risposta ricevuta dal server.");
    if (activeButtonId) {
      toggleButtonState(activeButtonId, false);
      activeButtonId = null; // Resetta la variabile
    }
  }
});

let currentConfig = {};
```



# Script

## App.py

```
app.py > ...
1
2 import os
3 import json
4 import csv
5 import glob
6 from flask import Flask, request, jsonify, render_template
7 import subprocess
8 import sys
9 from flask_socketio import SocketIO
10
11 app = Flask(__name__)
12 socketio = SocketIO(app)
13
14 @app.route('/')
15 def index():
16     return render_template('index.html')
17
18 CONFIG_FILE = "config.json"
19
20 @socketio.on('connect')
21 def handle_connect():
22     print("Client connesso")
23
24 def load_configurations():
25     if os.path.exists(CONFIG_FILE):
26         try:
27             with open(CONFIG_FILE, "r") as file:
28                 return json.load(file)
29         except json.JSONDecodeError:
30             # Ripristina il file se è malformato
31             default_config = {"gdrive": {}, "dropbox": {}, "onedrive": {}}
32             save_configurations(default_config)
33             return default_config
34
35     # Ripristina il file se non esiste
36     default_config = {"gdrive": {}, "dropbox": {}, "onedrive": {}}
37     save_configurations(default_config)
38     return default_config
39
40 def save_configurations(configurations):
41     with open(CONFIG_FILE, "w") as file:
42         json.dump(configurations, file, indent=4)
43
44 configurations = load_configurations()
45
46 def run_extractor(script_path, args=None):
47     try:
48         python_path = sys.executable
49         print(f"DEBUG: Python Path: {python_path}")
50         print(f"DEBUG: Script Path: {script_path}")
51
52         command = [python_path, script_path]
53         if args:
54             command.extend(args)
55         print(f"DEBUG: Command: {' '.join(command)}")
56
57         result = subprocess.run(command, capture_output=True, text=True)
58         print(f"DEBUG: Subprocess stdout: {result.stdout}")
59         print(f"DEBUG: Subprocess stderr: {result.stderr}")
60
61         if result.returncode != 0:
62             return {"success": False, "error": str(e)}
63
64     except Exception as e:
65         return {"success": False, "error": str(e)}
66
67 @app.route('/upload/secret', methods=['POST'])
68 def upload_secret():
69     try:
70         # Recupera il file caricato
71         secret_file = request.files["secret"]
72         secret_data = json.load(secret_file)
73
74         # Aggiorna config.json con i dati di Google Drive
75         with open('config.json', 'r+') as config_file:
76             config = json.load(config_file)
77             config['gdrive'] = secret_data
78             config_file.seek(0)
79             json.dump(config, config_file, indent=4)
80             config_file.truncate()
81
82         return jsonify({"success": True, "message": "Configurazione aggiornata con successo"}), 200
83     except Exception as e:
84         return jsonify({"success": False, "error": str(e)}), 500
85
86 @app.route('/configure/extractor', methods=['POST'])
87 def configure_extractor(extractor):
88     if extractor not in configurations:
89         return jsonify({"message": "Estrattore non valido"}), 400
90
91     data = request.json
92     configurations[extractor] = data
93     save_configurations(configurations)
94     return jsonify({"message": f"Configurazione salvata per {extractor}"}), 200
95
96 @app.route('/get-configurations', methods=['GET'])
97 def get_configurations():
98     try:
99         with open('config.json', 'r') as file:
100             configurations = json.load(file)
101         return jsonify(configurations)
102     except Exception as e:
103         return jsonify({"error": str(e)})
```

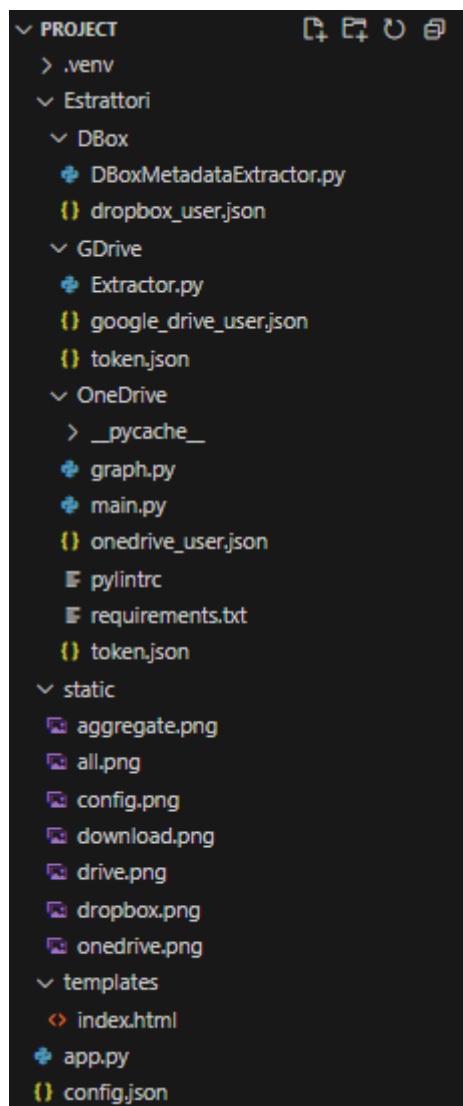
## Esecuzione:

Per eseguire il tool bisogna, tramite PowerShell, spostarsi nella cartella in cui è contenuto il tool ed eseguirlo con il comando: **“python app.py”**

```
PS C:\Users\Christian\Desktop\Progetto_RETI\Project> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 816-780-245
127.0.0.1 - - [27/Jan/2025 18:00:59] "GET /socket.io/?EIO=4&transport=polling&t=PIejP3g HTTP/1.1" 200 -
Client connesso
127.0.0.1 - - [27/Jan/2025 18:00:59] "POST /socket.io/?EIO=4&transport=polling&t=PIejPI-&sid=4Mcp5cQ5asqmD2ZIAAAA HTTP/1.1" 200 -
127.0.0.1 - - [27/Jan/2025 18:00:59] "GET /socket.io/?EIO=4&transport=polling&t=PIejPJ4&sid=4Mcp5cQ5asqmD2ZIAAAA HTTP/1.1" 200 -
```

## Struttura del Progetto

Nella barra laterale sinistra di VSCode è possibile visualizzare i sorgenti che compongono il progetto.



## Cartella Principale

- **.venv**: Una directory contenente l'ambiente virtuale Python. Viene utilizzata per isolare le dipendenze del progetto.

## Cartella "Estrattori"

### 1. DBox:

- **dropbox\_user.json**: Un file JSON che contiene i dati dell'utente di Dropbox.
- **DBoxMetadataExtractor.py**: Uno script Python per estrarre i dati e le versioni dei file da Dropbox.

### 2. GDrive:

- **google\_drive\_user.json**: Un file JSON simile al precedente, ma relativo a Google Drive. Contiene i dati dell'utente recuperati tramite l'API di Google Drive.
- **Extractor.py**: Uno script Python per gestire l'interazione con l'API di Google Drive. Si occupa di effettuare autenticazione ed il recupero dei file e degli inerenti metadati.

### 3. OneDrive:

- **graph.py**: Gestisce le interazioni con Microsoft Graph API per accedere ai dati di OneDrive.
- **main.py**: Contiene il flusso principale per eseguire operazioni legate a OneDrive, come il recupero dei file o delle loro versioni.
- **requirements.txt**: Elenca le librerie Python necessarie per eseguire il progetto (come Flask, requests, Google API Client, etc.).
- **onedrive\_user.json**: Un file JSON simile al precedente per OneDrive. Contiene i dati dell'utente recuperati tramite l'API di Graph.
- **.pylintrc**: Un file di configurazione per pylint, usato per definire regole di linting e stile del codice.

## Cartella “static”

Contiene risorse statiche utilizzate nell'applicazione web, come immagini.

- **aggregate.png**: Un'icona relativa all'aggregazione dei dati.
- **config.png**: Rappresenta un'immagine per configurare le impostazioni.
- **drive.png**: Un'icona per Google Drive.
- **dropbox.png**: Un'icona per Dropbox.
- **onedrive.png**: Un'icona per OneDrive.
- **all.png**: Un'icona per estrarre da tutti i Cloud.
- **download.png**: Un'icona per scaricare il report.

## Cartella “templates”

- **index.html**: Il file HTML principale per l'interfaccia utente dell'applicazione web che viene renderizzata da Flask. Include elementi come pulsanti, tabelle per mostrare i dati, ecc.

## File principali

- **app.py**:
  - Contiene il server Flask che gestisce le richieste per estrarre dati da Dropbox, Google Drive e OneDrive.
  - Implementa endpoint per configurare le API e avviare il processo di estrazione.
- **config.json**:
  - File di configurazione che include le chiavi API e altre informazioni necessarie per interagire con Dropbox, Google Drive e OneDrive.
- **TO\_DO.txt**:
  - File di testo che contiene note o attività da completare per il progetto.

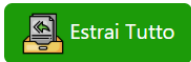
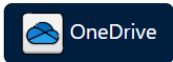
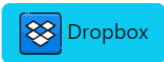
Interfaccia  
Homepage



File Versioning Aggregator

Gestisci il versioning dei tuoi file su diverse piattaforme cloud.

Configura Estrattori



Risultati

Percorso	Nome File	Tipo File	Autore	Versione	Ultima Modifica	Autore Ultima Modifica	Dimensione	Service
----------	-----------	-----------	--------	----------	-----------------	------------------------	------------	---------

Scarica Report

Configuratore



Configura Estrattori

Tipo di Estrattore

Dropbox

App Key

App Secret

Access Token

Salva Configurazione

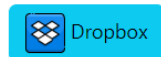
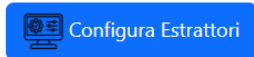
Nome File	Autore	Dimensione
-----------	--------	------------

## Risultati



### File Versioning Aggregator

Gestisci il versioning dei tuoi file su diverse piattaforme cloud.



#### Risultati per Dropbox

Utenti Connessi :

Percorso	Nome File	Tipo File	Autore	Versione	Ultima Modifica	Autore Ultima Modifica	Dimensione	Service
/TID_VIENNA.pdf		PDF		1.0	2025-02-05 19:20:22Z		48599 bytes	Dropbox
/TID_VIENNA.pdf		PDF		2.0	2025-01-24 09:53:39Z		49612 bytes	Dropbox
/aac.pdf		PDF		1.0	2025-01-24 09:53:39Z		49612 bytes	Dropbox



## Implementazioni

### Google Drive Business

- **Autenticazione:** avviene tramite un Service Account utilizzando il file JSON generato nella Google Cloud Console. La libreria **google.oauth2.service\_account** consente di autenticare il client senza interazione utente. Il Service Account deve avere i permessi necessari per accedere ai file e ai Drive Condivisi (Shared Drives).
- **Recupero File e Metadati:** il recupero si basa sull'API **drive.files().list()**, con **q='trashed=false'** per escludere i file eliminati. Per accedere ai Drive Condivisi è necessario abilitare **supportsAllDrives=True** e **includeItemsFromAllDrives=True**. I metadati vengono estratti con **fields='files(id, name, mimeType, owners(emailAddress), parents)'**.
- **Gestione delle Versioni:** viene implementata attraverso **drive.revisions().list(fileId=...)**, che consente di ottenere lo storico delle modifiche. Per ogni revisione vengono recuperati **modifiedTime** e **lastModifyingUser(displayName)** per identificare l'ultima modifica del file.
- **Navigazione delle Cartelle:** gestita con un sistema ricorsivo che esplora le sottocartelle partendo dalla root. La gerarchia viene mantenuta utilizzando il campo **parents** nei metadati per risalire la struttura delle cartelle e costruire il percorso completo.

- **Reportistica:** i dati vengono salvati in un file Excel strutturato con **pandas**, includendo informazioni su percorso, nome file, ID, proprietario, ultima modifica, autore dell'ultima modifica e dimensione del file. I dati di tutti i Drive Condivisi possono essere aggregati in un unico report centralizzato.
- **Gestione degli Errori:** gli errori di autenticazione vengono gestiti con il refresh automatico del token OAuth2. Il **logging** è implementato per monitorare l'estrazione e l'elaborazione di grandi volumi di dati, identificando eventuali errori API o problemi di autorizzazione.

## Box

- **Autenticazione:** avviene tramite OAuth2 utilizzando la libreria **boxsdk**, con configurazione di **client\_id**, **client\_secret**, **access\_token** e **refresh\_token**. Il sistema di refresh automatico del token è implementato per garantire un accesso continuo ai dati.
- **Recupero File e Metadati:** il recupero è gestito tramite **client.folder(folder\_id).get\_items()**, che elenca i file e le cartelle presenti in una directory specifica. I metadati vengono estratti con **fields=['id', 'name', 'modified\_by', 'modified\_at', 'size']**. Il percorso completo di ogni file viene costruito navigando la **path\_collection** della cartella.
- **Gestione delle Versioni:** sfrutta **file.get\_previous\_versions()**, che permette di ottenere lo storico del file. Per ogni versione vengono registrati **modified\_by['name']** e **modified\_at**, consentendo di monitorare le modifiche e gli utenti che hanno effettuato gli aggiornamenti.
- **Navigazione delle Cartelle:** implementata ricorsivamente, esplorando le sottocartelle a partire dalla root. La gerarchia delle directory viene recuperata utilizzando **path\_collection['entries']**, che permette di costruire il percorso assoluto di ogni file.
- **Reportistica:** i dati vengono salvati in un file Excel tramite **pandas**, con un formato standardizzato che include informazioni sul percorso, nome file, ID, proprietario, ultima modifica e dimensione del file. È possibile aggregare i dati provenienti da più cartelle aziendali per generare report centralizzati.
- **Gestione della Sicurezza:** garantita attraverso la gestione automatica del token OAuth2, che previene l'uso di credenziali scadute. Il sistema di **logging** traccia gli errori API, i problemi di autorizzazione e il superamento delle **rate limit** imposte da Box, migliorando l'affidabilità del processo di estrazione.

## Conclusione

Il progetto "Aggregatore di Versioning per Piattaforme Cloud SaaS" rappresenta una soluzione innovativa e scalabile per la gestione delle versioni dei file su diverse piattaforme cloud, come Dropbox, Google Drive e OneDrive. Durante il processo di sviluppo, ci siamo concentrati su tre aspetti fondamentali: efficienza, sicurezza e semplicità d'uso.

### Punti di Forza del Progetto

- **Centralizzazione della Gestione dei File:**
  - L'applicazione consente di accedere a file e versioni precedenti da un'unica interfaccia, riducendo la necessità di passare da una piattaforma all'altra.
- **Sicurezza Avanzata:**
  - Grazie all'uso di protocolli di crittografia TLS e all'autenticazione OAuth2, i dati degli utenti sono protetti in tutte le fasi del processo.
- **Efficienza Operativa:**
  - L'architettura modulare garantisce tempi di risposta rapidi e la possibilità di scalare il sistema per gestire un numero crescente di utenti e dati.
- **Interfaccia Intuitiva:**
  - Il design dell'applicazione è stato pensato per utenti di ogni livello, garantendo un'esperienza utente fluida e accessibile.
- **Integrazione Completa:**
  - L'applicazione utilizza le API ufficiali di Dropbox, Google Drive e OneDrive, assicurando una gestione ottimale delle funzionalità offerte da ciascuna piattaforma.

### Impatto e Benefici

Questo progetto offre numerosi vantaggi per aziende, team di lavoro e utenti singoli:

- **Risparmio di Tempo:** Gli utenti possono accedere rapidamente alle versioni dei file con pochi clic.
- **Collaborazione Migliorata:** La gestione delle versioni riduce i conflitti e migliora la trasparenza nel lavoro di gruppo.
- **Versatilità:** L'applicazione è adattabile a molteplici scenari, dall'uso personale alla gestione di grandi volumi di dati aziendali.



## **Prospettive Future**

L'applicazione è stata progettata per essere espandibile e adattabile. Tra le possibili evoluzioni future:

- **Integrazione con Altre Piattaforme:**
  - Estendere il supporto a servizi cloud come Amazon S3, Box e altri provider emergenti.
- **Automazione Avanzata:**
  - Implementare workflow automatici per il backup e la sincronizzazione dei file.
- **Funzionalità Analitiche:**
  - Aggiungere reportistica dettagliata sulle modifiche ai file e sull'utilizzo delle risorse cloud.

L'Aggregatore di Versioning per Piattaforme Cloud SaaS con ulteriori sviluppi previsti, l'applicazione potrebbe diventare uno strumento indispensabile in ambienti personali e aziendali.

## Sitografia

### Dropbox API

- Console sviluppatori di Dropbox: <https://www.dropbox.com/developers>
- Documentazione Dropbox API:  
<https://www.dropbox.com/developers/documentation>

### Google Drive API

- Google Cloud Console: <https://console.cloud.google.com/>
- Documentazione ufficiale della Google Drive API:  
<https://developers.google.com/drive>
- OAuth 2.0 per Google API:  
<https://developers.google.com/identity/protocols/oauth2>

### Microsoft OneDrive API (Microsoft Graph)

- Portale Microsoft Azure: <https://portal.azure.com/>
- Documentazione Microsoft Graph API: <https://learn.microsoft.com/en-us/graph/>
- Guida introduttiva di Microsoft Graph API: <https://developer.microsoft.com/en-us/graph/quick-start>