МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, ФІЗИКИ ТА МАТЕМАТИКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Проектування та розроблення додатку автоматичного трансферу даних з хмарного середовища Zoom в корпоративний Google диск

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «магістр»

Виконав: здобувач Манзюк В.В.
Спеціальності: **122 Комп'ютерні науки**
Освітньо-професійної (наукової)
програми: **Комп'ютерні науки**
Керівник: д.пед.н, к.ф.-м.н.,
проф. Співаковський О.В.
Рецензент: Сенчишен Д.О., Middle Backend Developer, IT компанія DataArt

Херсон – Івано-Франківськ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Запит HTTP - це процес передачі запиту від клієнта до сервера за допомогою протоколу HTTP.

Консоль GCP (Google Cloud Platform Console) - це веб-інтерфейс для управління ресурсами GCP. Консоль GCP надає користувачам доступ до всіх послуг GCP, включаючи Compute Engine, App Engine, Cloud Storage, Cloud SQL та інших.

Сервісний обліковий запис (service account) - це обліковий запис, який використовується для автоматизації завдань на платформі Google Cloud Platform. Сервісні облікові записи не пов'язані з жодним конкретним користувачем і можуть використовуватися для доступу до ресурсів GCP від імені програми або послуги.

ACCESS_TOKEN - це токен доступу, який дозволяє програмі отримувати доступ до ресурсів сервера.

API (Application Programming Interface) - це інтерфейс прикладного програмування, який дозволяє двом програмам спілкуватися між собою. API - це набір правил і специфікацій, які визначають, як дві програми можуть обмінюватися даними.

Cloud Storage - це об'єктне сховище даних, яке дозволяє зберігати будь-які дані будь-якого розміру в хмарі. Cloud Storage є частиною платформи Google Cloud Platform і пропонує надійне, масштабоване та високодоступне сховище даних.

CRUD Операції - - це абревіатура від чотирьох основних операцій, які використовуються для управління даними в базі даних: Create, Read, Update, Delete

Google Cloud Platform (GCP) - це набір хмарних обчислювальних служб, що надаються Google. GCP пропонує широкий спектр послуг, включаючи обчислення, дані, машинне навчання, аналітику, продуктивність тощо.

Google Drive - служба зберігання та синхронізації файлів, розроблена Google. Google Drive, запущений 24 квітня 2012 року, дозволяє користувачам зберігати файли в хмарі, синхронізувати файли на різних пристроях і ділитися файлами.

Github - це платформа для хостингу коду, яка використовується для керування версіями та співпраці. Вона дозволяє вам та іншим працювати над проектами з будь-якої точки світу.

GET, POST - це два найпоширеніші методи HTTP-запитів.

OAuth 2.0 - це відкритий стандарт авторизації, який дозволяє програмам отримувати доступ до ресурсів, які контролюються користувачем, без необхідності передавати пароль користувача.

Zoom - це хмарна платформа для відеоконференцій, яка дозволяє користувачам спілкуватися один з одним для відеодзвінків, онлайн-зустрічей і вебінарів. Це одна з найпопулярніших платформ для відеоконференцій у світі з понад 500 мільйонами користувачів.

Zoom API - це REST API, який дозволяє розробникам інтегрувати функції Zoom у власні програми. Zoom API надає різноманітні кінцеві точки для керування користувачами, зустрічами, записами та іншими ресурсами Zoom.

Zoom Marketplace - це платформа, де розробники можуть публікувати та продавати свої інтеграції Zoom. Інтеграція Zoom – це програми, служби та інструменти, які розширюють функціональні можливості Zoom. Їх можна використовувати для покращення відеоконференцій, вебінарів та інших можливостей Zoom.

ВСТУП

Zoom є популярною програмою для відеоконференцій та онлайн-зустрічей. Вона дозволяє користувачам здійснювати аудіо- та відеозв'язок з іншими користувачами в реальному часі, а також спільно працювати над проектами, ділитися екраном та виконувати інші функції.

Zoom став поширеним у період пандемії COVID-19, коли багато людей були змушені переходити на роботу з дому та знаходити нові способи комунікації з колегами та друзями. Zoom надавав простий та зручний інструмент для збереження зв'язку з іншими людьми, що знаходилися в різних частинах світу. Саме с тіх часів сервіс відеоконференцій, став одним з основних інструментів, які використовують громадяни Землі для підтримки зв'язку через Інтернет.

Крім того, Zoom став популярним інструментом для навчання на віддаленій основі, як у школах, так і у вищих навчальних закладах. Існує кілька причин, чому він став таким популярним серед педагогів та студентів:

- **Легкість використання:** Zoom має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє вчителям та студентам легко зв'язуватися один з одним.
- **Якість зв'язку:** Zoom надає високоякісний зв'язок з можливістю передачі відео та аудіо в реальному часі, що дозволяє вчителям та студентам проводити уроки та лекції без перебоїв та зависань.
- **Можливості співпраці:** Zoom має функції спільної роботи над документами та екраном, що дозволяє вчителям та студентам працювати разом над проектами та завданнями.
- **Гнучкість:** Zoom дозволяє проводити заняття в режимі реального часу, або записувати їх для перегляду в зручний для студента час.

Zoom було недостатньо стати впізнаваним брендом. Наступним кроком було стати невід'ємною частиною веб-сайтів і застосунків, запропонувавши API і SDK. Для багатьох розробників це дало можливість створювати продукти з потоковим відео або конференц-зв'язком, а також плагіни для популярного сервісу відеодзвінків, як-от транскрипція та доповнення до інтерактивних ігор Zoom.

Але не дивлячись на безліч переваг платформа Zoom має ряд недоліків, особливо що стосується зберігання записів у Zoom Cloud:

- **Вартість:** Zoom Cloud Recording існує тільки для платних планів [1]. Вартість використання хмарного сховища Zoom суттєво відрізняється від інших Cloud Storage.
- **Обмеження за часом:** Zoom Cloud Recording дозволяє зберігати записи протягом 180 днів. Якщо ви хочете зберігати записи довше, вам потрібно буде завантажити їх на свій комп'ютер або інше місце зберігання.
- **Недостатня організація:** Zoom Cloud Recording не пропонує багато функцій для організації записів. Ви можете сортувати записи за датою, але не можете створювати теги або папки.
- **Недостатня безпека:** Zoom Cloud Recording пропонує деякі функції безпеки, такі як шифрування та двофакторна аутентифікація. Однак ці функції не є обов'язковими, і ви можете бути схильні до кіберзагроз, якщо не включите їх.

Завдяки платформі Zoom можна проводити відеоконференції та записувати їх для подальшого перегляду. Але що робити, якщо потрібно зберегти запис на довготривалий період та ділитися ним з іншими користувачами? Ця робота присвячена вирішенню питання переносу файлів з Zoom Cloud до Google Drive.

РОЗДІЛ 1. ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ. МЕХАНІЗМИ РЕАЛІЗАЦІЇ ДОДАТКУ

1.1 Огляд існуючих підходів до розв’язання поставленої задачі

Перенесення хмарних записів Zoom на Google Диск може стати дуже корисним для багатьох користувачів, оскільки це дозволяє зберігати та ділитися своїми записами в безпечному та надійному місці. Проте, є кілька підходів до цього процесу, кожен з яких має свої переваги та недоліки.

Для того, щоб зберегти запис відеоконференції в Google Диск, звичайному користувачу потрібно виконати наступні кроки:

- Завантажити запис з Zoom на свій комп'ютер.
- Зайти на сторінку Google Диска до свого облікового запису та створити нову папку, яка буде містити записи з Zoom.
- Вибрати файл зі свого комп'ютера та завантажити його до папки, яку створили раніше.

Так - це дуже прості кроки, але якщо файлів досить багато, та вони з’являються кожного дня - просте завдання перетворюється на неприємну рутинну роботу. Тому виникає ідея автоматизувати цей процес.

Наприклад, один з підходів - це використання сторонніх програм або скриптів, які дозволяють автоматизувати цей процес та зберігати записи відразу після завершення зустрічі.

Варто зазначити, що кожен підхід має свої плюси та мінуси. Далі ми розглянемо найпоширійшенні існуючи інструменти або методи для вирішення поставленого завдання.

1.1.1 Google Drive for desktop

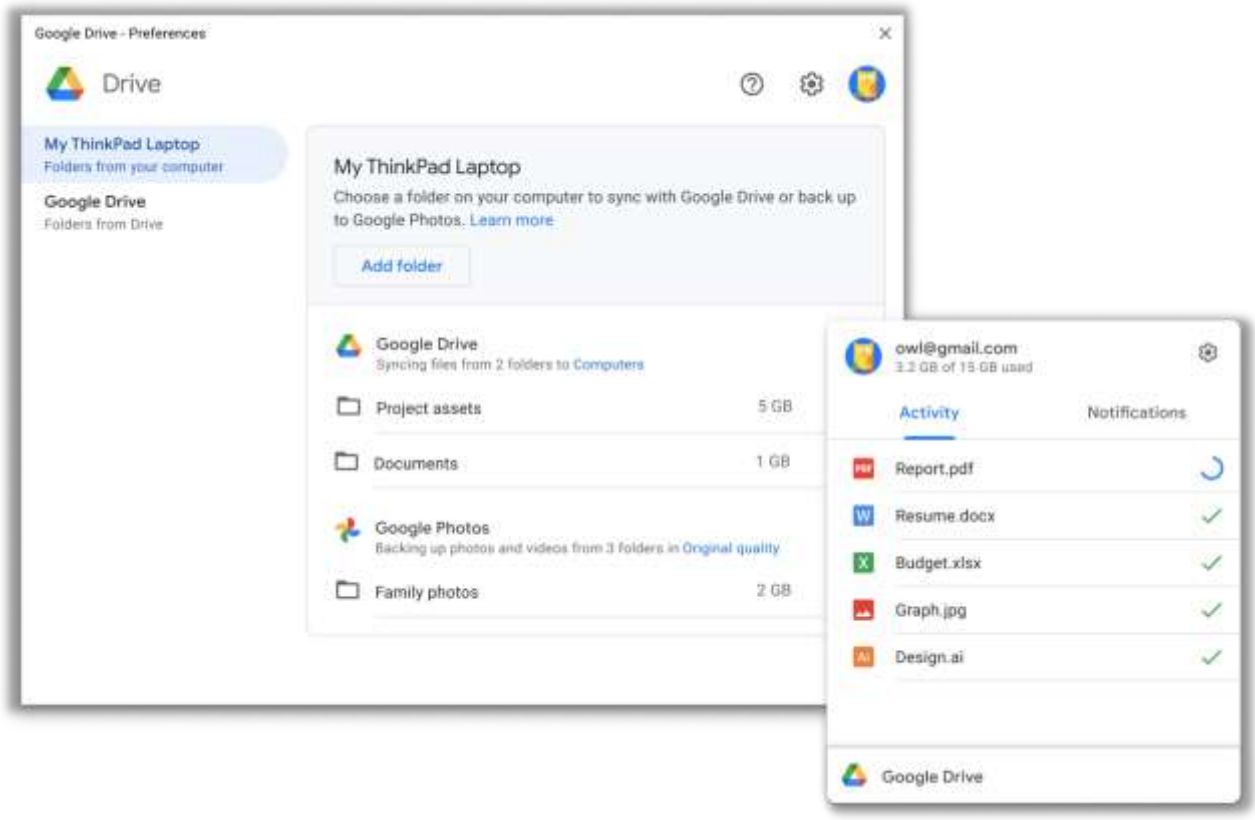


Рисунок 1.1 – Приклад роботи додатку *Google Drive for desktop*.

Google Drive for desktop - це програма для настільних комп'ютерів, яка створює віртуальний диск на вашому комп'ютері, дозволяючи вам отримувати доступ до файлів Google Диску так, ніби вони зберігаються на вашому локальному комп'ютері. Використовуючи цей підхід, ви можете завантажити записи Zoom на свій локальний комп'ютер, а потім перенести їх на віртуальний диск Google Drive File Stream. Цей підхід забезпечує більшу гнучкість з точки зору угод про імена файлів і папок призначення. Однак він вимагає більше ручних зусиль і може займати більше місця в локальному сховищі.

1.1.2 Додаток "Capture - Transfer to Google Drive and Dropbox"

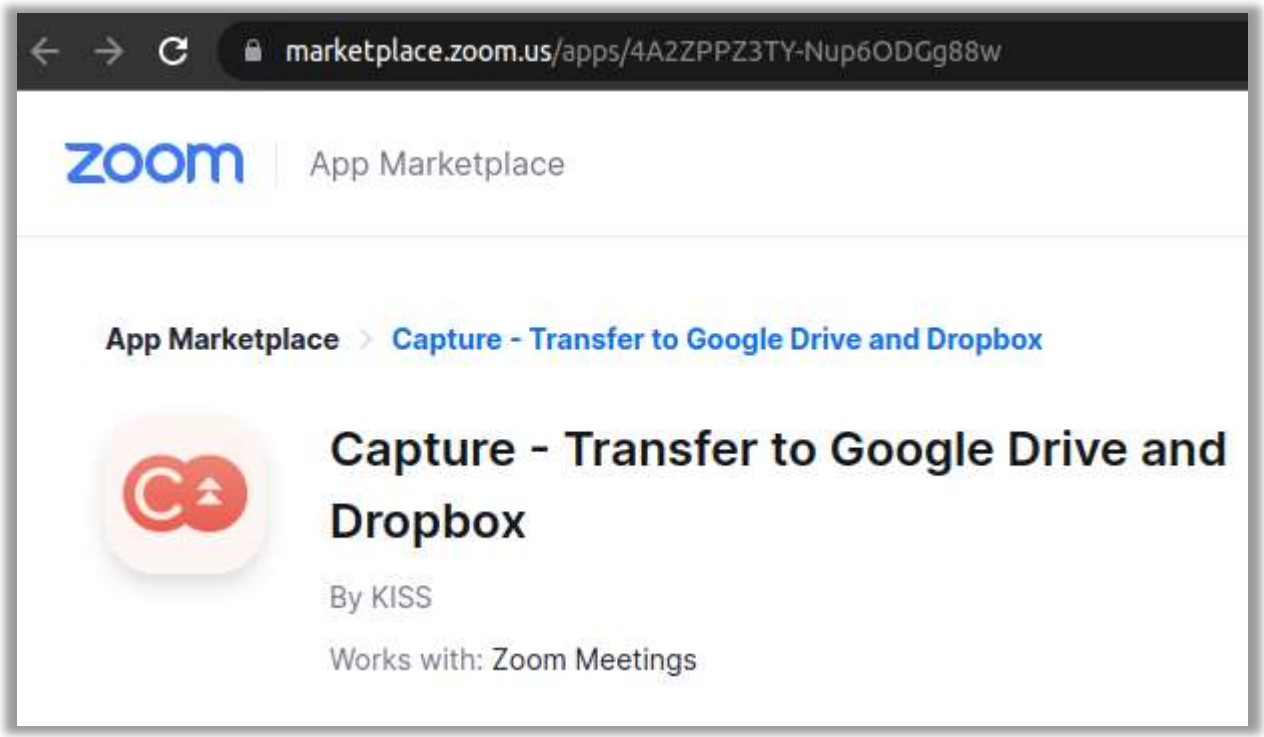


Рисунок 1.2 – Додаток *Capture - Transfer to Google Drive and Dropbox* на Zoom Marketplace.

Сторонній додаток, доступний на Zoom Marketplace [8], який дозволяє користувачам записувати свої зустрічі в Zoom і автоматично передавати записи до своїх облікових записів Google Drive або Dropbox.

За допомогою цієї програми користувачі можуть легко записувати свої зустрічі та вебінари в Zoom у високоякісному відеоформаті, а також автоматично передавати записи до своїх хмарних сховищ для зручного доступу та спільного використання. Додаток надає зручний спосіб створювати резервні копії та ділитися записаними сесіями Zoom з колегами, членами команди або будь-ким, кому потрібен доступ до записів.

Додаток "Capture - Transfer to Google Drive and Dropbox" дозволяє користувачам налаштовувати параметри запису, наприклад, вибирати якість відео, вибирати, яких спікерів захоплювати, і додавати водяні знаки до записів з метою брендингу. Користувачі також можуть налаштувати

автоматичне планування, щоб починати і зупиняти запис у певний час або на основі відвідуваності зустрічі.

Важливо зазначити, що цей додаток вимагає платної підписки для використання. Користувачі повинні спочатку зареєструватися на безкоштовну пробну версію, а потім підписатися на платний план, щоб продовжувати користуватися додатком після закінчення пробного періоду.

1.1.3 Додаток "Google Drive for Zoom by Splain"

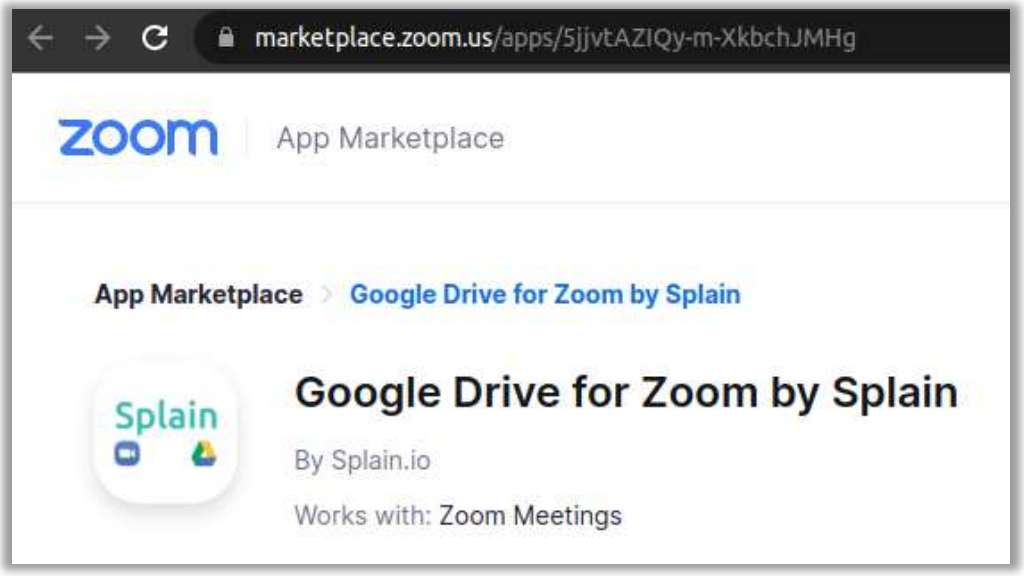


Рисунок 1.3 – Додаток *Google Drive for Zoom by Splain* на Zoom Marketplace.

Дозволяє користувачам отримувати доступ до файлів з Google Диску та ділитися ними безпосередньо під час нарад і вебінарів у Zoom[9].

Після встановлення користувачі можуть легко підключити свій обліковий запис Google Drive до програми, а потім отримати доступ до своїх файлів під час нарад і вебінарів у Zoom. Це усуває необхідність перемикатися між різними програмами і спрощує обмін файлами під час сеансів Zoom.

За допомогою програми Google Диск для Zoom користувачі можуть ділитися файлами безпосередньо з Google Диску без необхідності їх

попереднього завантаження. Додаток також надає такі функції, як співпраця в режимі реального часу, попередній перегляд документів і дозволи на редагування, що полегшує командам спільну роботу над спільними файлами під час нарад у Zoom.

1.1.4 Додаток "Transferring-videos.com"

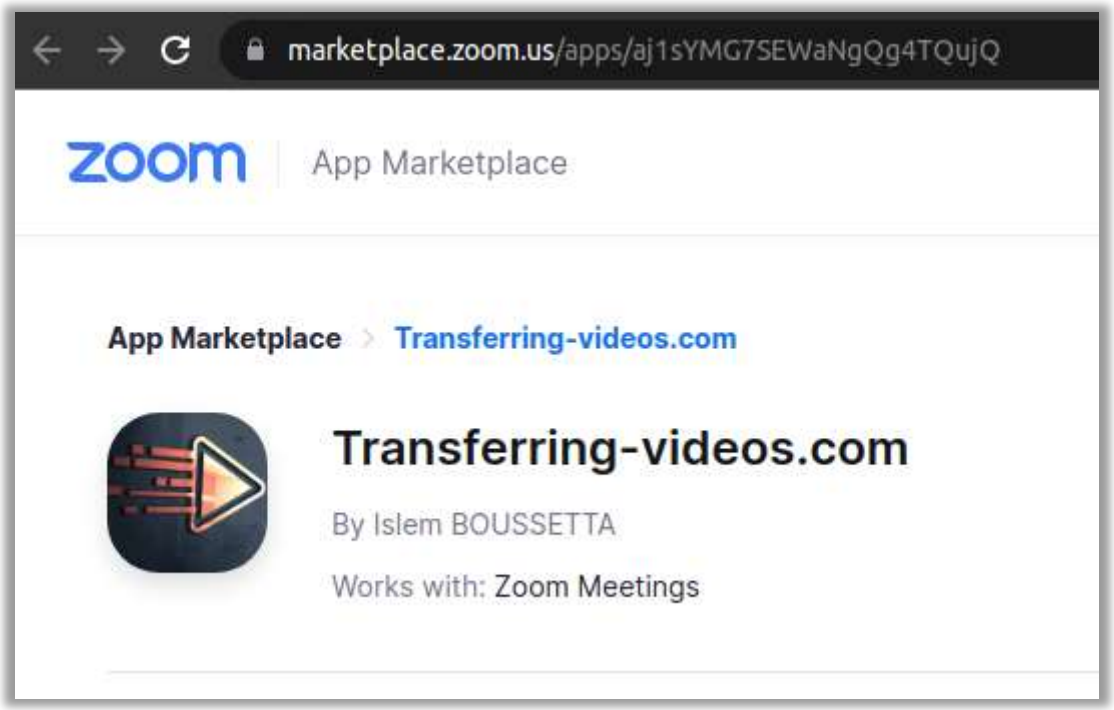


Рисунок 1.4 – Додаток *Transferring-videos.com* на Zoom Marketplace.

Веб-сервіс, який дозволяє користувачам автоматично публікувати свої записи в Zoom на різних платформах соціальних мереж і в хмарних сховищах. Користувачі можуть просто створити обліковий запис на сайті і використовувати сторінку "Групова передача", щоб налаштувати передачу своїх Zoom-записів до потрібного місця призначення[10].

Сервіс підтримує такі популярні соціальні мережі, як YouTube, Facebook (групи і сторінки), LinkedIn, Vimeo, Dailymotion і Twitter, а також хмарні сервіси зберігання даних, такі як Google Drive, Dropbox і OneDrive.

Процес перенесення записів передбачає вибір Zoom як платформи-джерела, вибір бажаної платформи-приймача та виконання кількох простих

кроків для налаштування перенесення. Всього за кілька кліків користувачі можуть автоматично публікувати свої записи в Zoom на своїх улюблених платформах соціальних мереж або зберігати їх на своїх улюблених хмарних сервісах зберігання даних.

1.1.5 Додаток "zBackup.app"

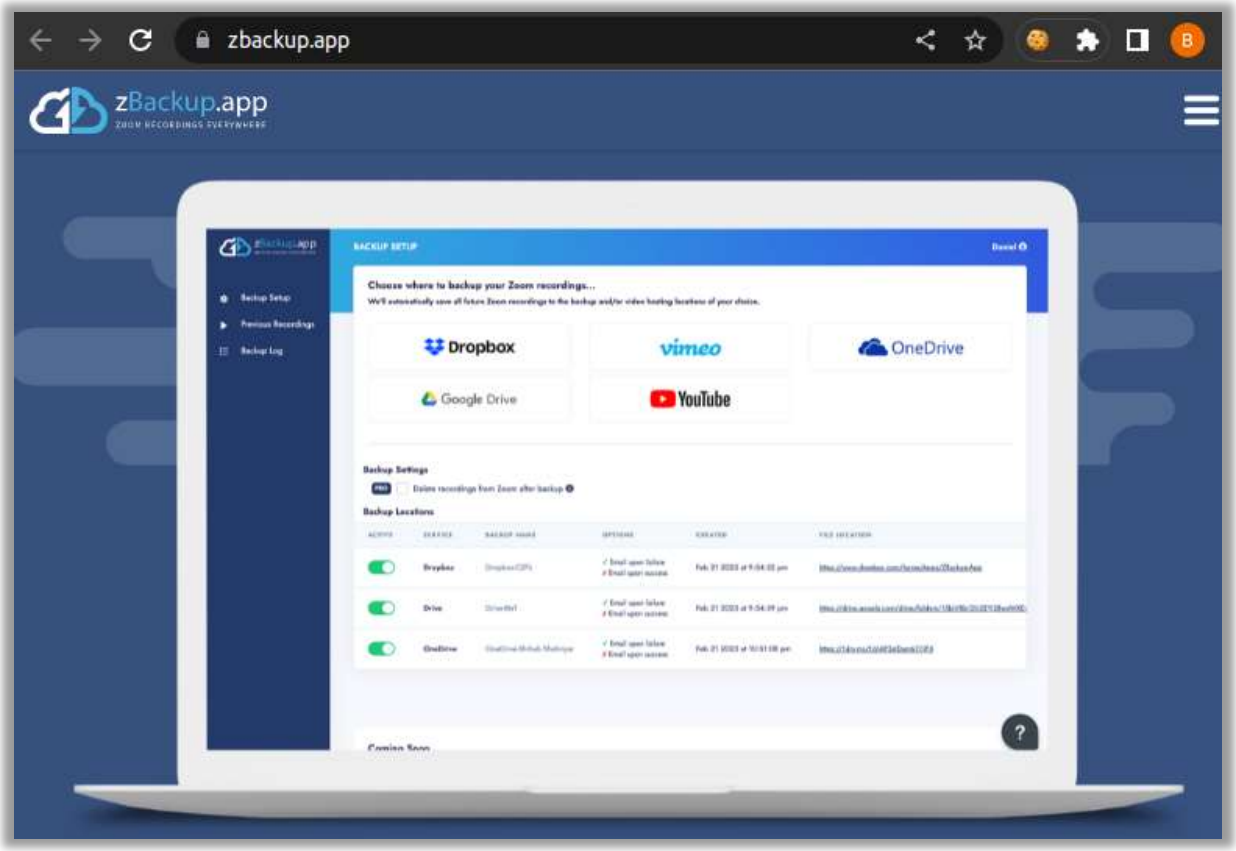


Рисунок 1.5 – Офіційний сайт хмарного сервісу *zBackup.app.*

zBackup.app - це хмарний сервіс резервного копіювання та відновлення, який дозволяє користувачам легко та безпечно створювати резервні копії та відновлювати свої дані.

Сервіс пропонує швидкі, надійні та зашифровані рішення для резервного копіювання для приватних осіб, підприємств та організацій. Користувачі можуть створювати резервні копії своїх файлів, папок і додатків у хмарі та отримувати доступ до своїх даних з будь-якого місця, в будь-який час і з будь-якого пристрою.

zBackup.app пропонує зручний веб-інтерфейс і мобільний додаток, що дозволяє користувачам керувати процесами резервного копіювання та відновлення, планувати автоматичне резервне копіювання та відстежувати хід резервного копіювання. Сервіс також пропонує розширені функції, такі як інкрементне резервне копіювання, керування версіями та архівування для більш надійного резервного копіювання та відновлення.

Згідно з інформацією на сайті, zBackup.app використовує найсучасніші заходи безпеки для забезпечення конфіденційності, цілісності та доступності даних користувачів. Сервіс шифрує всі дані під час передачі та зберігання, а також використовує багатофакторну автентифікацію для запобігання несанкціонованому доступу до облікових записів користувачів.

Загалом, zBackup.app є надійним і безпечним сервісом резервного копіювання та відновлення для приватних осіб і компаній, які шукають хмарні рішення для захисту своїх даних.

1.1.6 Додаток "Zapier"

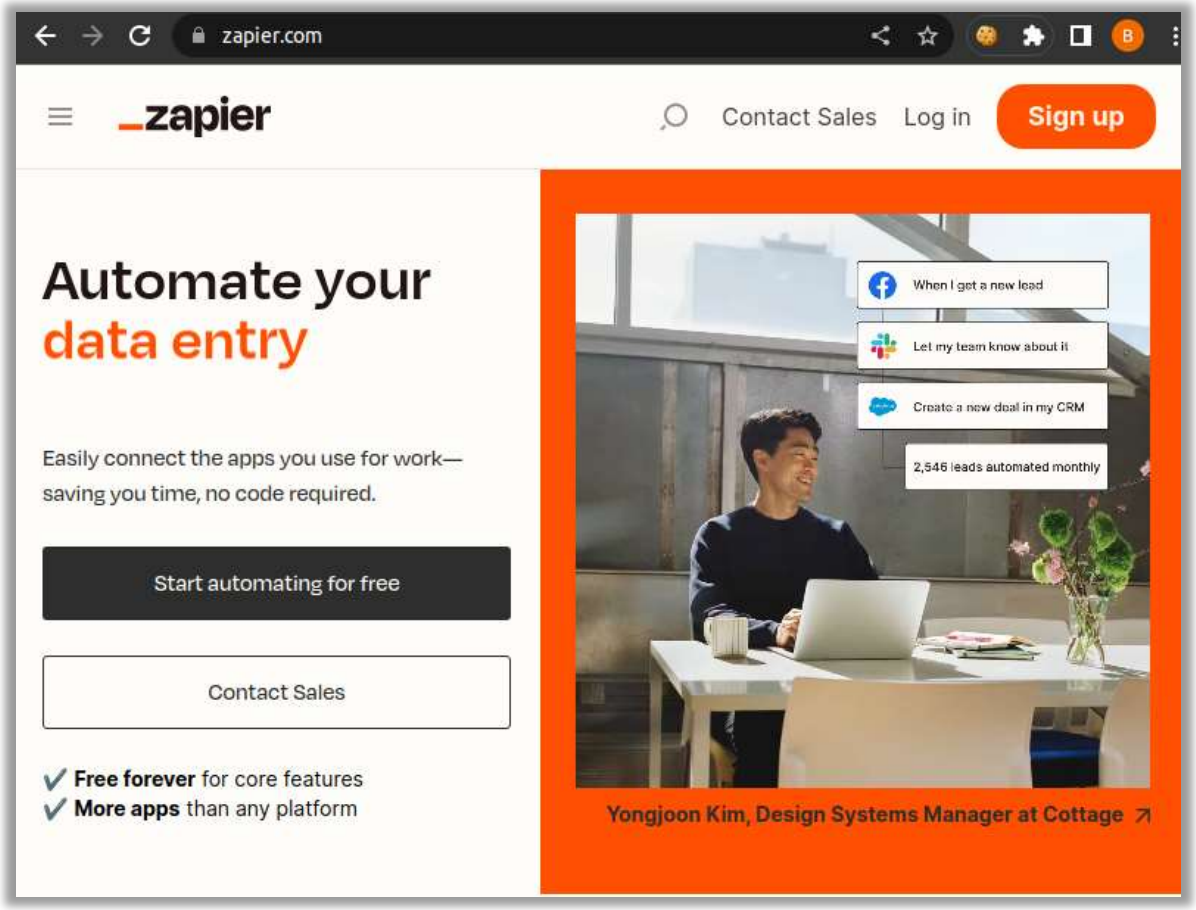


Рисунок 1.6 – Офіційний сайт додатку *Zapier*.

Zapier - це веб-інструмент автоматизації, який дозволяє користувачам підключати різні веб-додатки та автоматизувати завдання між ними без будь-якого кодування або технічних навичок. Сервіс покликаний спростити процес інтеграції веб-додатків і робочих процесів, дозволяючи користувачам налаштовувати автоматизовані дії між ними за допомогою попередньо створених або налаштованих робочих процесів, які називаються "Zaps".

Zaps - це комбінація тригерів і дій, де тригер - це подія, яка відбувається в одному додатку, а дія - це результуюча подія, яка відбувається в іншому додатку. Наприклад, користувач може створити капчу, яка автоматично зберігає вкладення електронної пошти до певної папки на Google Диску, або капчу, яка надсилає сповіщення Slack, коли нові записи додаються до електронної таблиці Google Таблиці.

Zapier підтримує інтеграцію з понад 3 000 веб-додатків, включаючи такі популярні інструменти, як Gmail, Trello, Salesforce, Dropbox та багато інших. Користувачі можуть створювати власні робочі процеси за допомогою інтерфейсу drag-and-drop або використовувати готові шаблони для швидкої автоматизації типових завдань.

Zapier призначений як для приватних осіб, так і для бізнесу, і пропонує низку тарифних планів, що базуються на кількості запитів і дій на місяць, а також преміум-функції, такі як багатокрокові запити, умовна логіка та пріоритетна підтримка.

Однак Zapier - платний сервіс, і для його налаштування можуть знадобитися певні технічні навички [**Ошибка! Источник ссылки не найден.**].

1.1.7 Пошук аналогів на Github

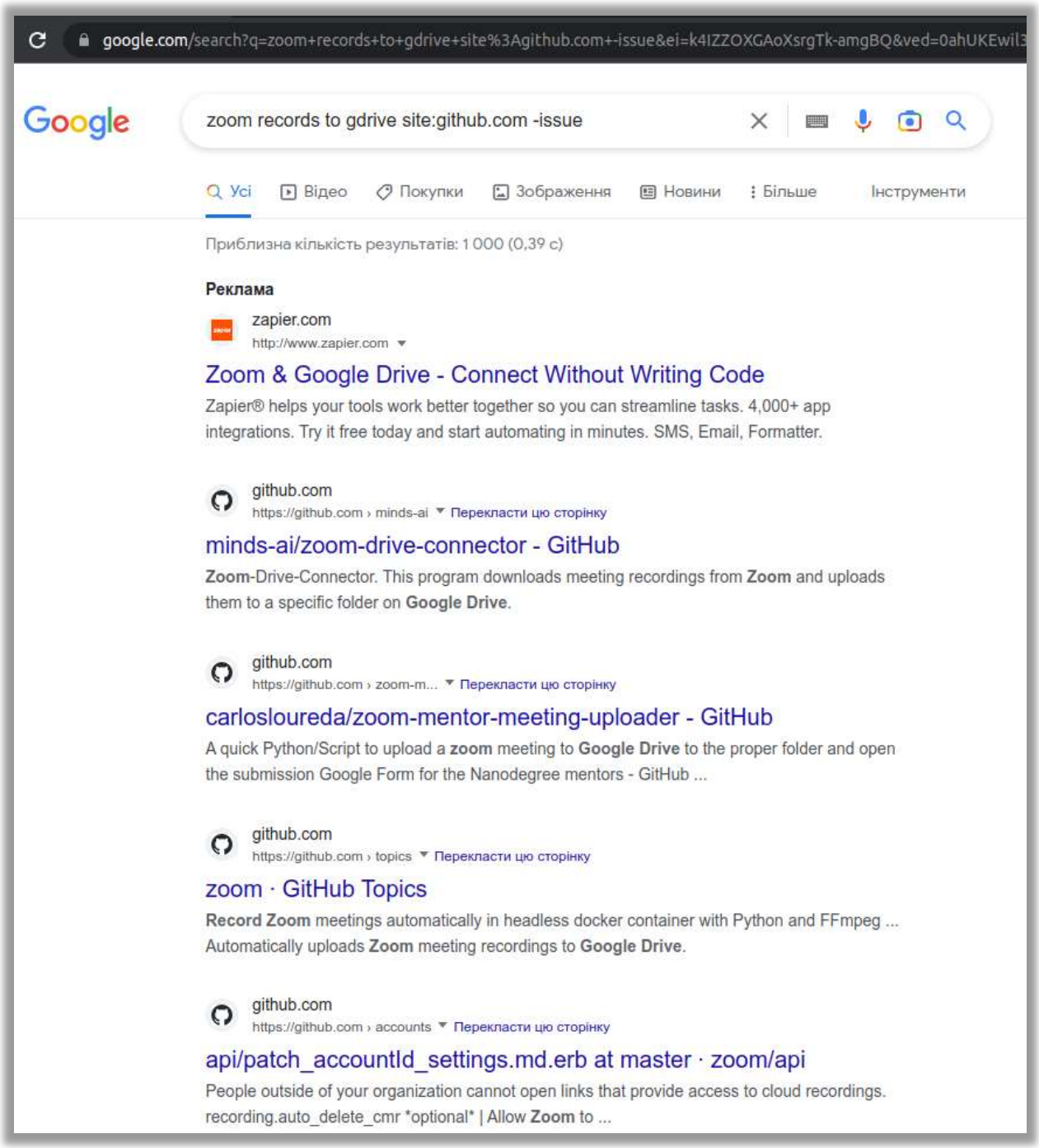


Рисунок 1.7 – Приклад пошуку аналогів додатку на *Github*.

Як можна бачити, пошук за даною тематикою видає відносно невелику кількість результатів - 1000. Але дійсно релевантних де-кілька десятків. Однак більшість з них застаріли. Якщо обмежити пошук останнім роком, кількість всіх результатів буде менш десяти[14], та всі вони не будуть

нерелевантними. Можна зробити висновок, що пік популярності даної теми (перенос zoom records to gdrive) приходиться на 2019-2020 роки - початок пандемії коронавірусної хвороби та поширення додатку Zoom. Звідси, витікають основні недоліки - різні версії мов програмування (наприклад python2, python3) та різні версії Google Drive API (поточна версія 3 [15]) та Zoom API (поточна версія 3 [16]). Тобто готового рішення, як то кажуть, з коробки, не існує.

Нижче наведено різні версії API Google Drive та Zoom API, а також їх дати виходу та характеристики:

Google Drive API

- **Версія 2:** 14 квітня 2010 року
- **Версія 3:** 28 листопада 2019 року

Zoom API

- **Версія 2:** 24 вересня 2014 року
- **Версія 3:** 18 жовтня 2022 року
-

Google Drive API v2 та v3 є стабільними та надійними API, які надають доступ до файлів, папок, користувачів і спільного доступу в Google Drive. Zoom API v2 та v3 також є стабільними та надійними API, які надають доступ до конференцій, записів, користувачів і спільного доступу в Zoom.

Однак Google Drive API v3 пропонує більше функціональних можливостей, ніж Google Drive API v2. Наприклад, Google Drive API v3 підтримує багатокористувацькі документи, редагування в реальному часі та нові API для пошуку і аналізу. Zoom API v3 також пропонує більше функціональних можливостей, ніж Zoom API v2. Наприклад, Zoom API v3 підтримує багатокористувацькі записи, редагування в реальному часі та нові API для пошуку і аналізу.

В цілому, Google Drive API v3 і Zoom API v3 є кращими варіантами, ніж їх попередні версії.

Таблиця 1.1 Порівняння API

Характеристика	Google Drive API v2	Google Drive API v3	Zoom API v2	Zoom API v3
Дата виходу	14 квітня 2010 року	28 листопада 2019 року	24 вересня 2014 року	18 жовтня 2022 року
Стабільність	Стабільна	Стабільна	Стабільна	Стабільна
Функціональність	Доступ до файлів, папок, користувачів і спільного доступу	Доступ до файлів, папок, користувачів і спільного доступу, а також нові функції, такі як підтримка багатокористувацьких документів, редагування в реальному часі та нові API для пошуку і аналізу	Доступ до конференцій, записів, користувачів і спільного доступу	Доступ до конференцій, записів, користувачів і спільного доступу, а також нові функції, такі як підтримка багатокористувацьких записів, редагування в реальному часі та нові API для пошуку і аналізу
Доступність	Java, Python, C++, Node.js, PHP, Ruby, Go, .NET	Java, Python, C++, Node.js, PHP, Ruby, Go, .NET	Java, Python, C++, Node.js, PHP, Ruby, Go, .NET	Java, Python, C++, Node.js, PHP, Ruby, Go, .NET
Інструменти розробки	Google Cloud Platform, Google Cloud Shell, Google Cloud SDK, сторонні інструменти	Google Cloud Platform, Google Cloud Shell, Google Cloud SDK, сторонні інструменти	Zoom SDK, сторонні інструменти	Zoom SDK, сторонні інструменти

1.2 Порівняння та висновки по існуючим додаткам

Таблиця 1.2 Порівняння існуючих аналогів

Назва додатку	Вартість на одного користувача в місяць
Google Drive for desktop	безкоштовно
Capture - Transfer to Google Drive	$10
Google Drive for Zoom	$5.99
Transferring-videos.com	9,99€
zBackup	$7.99
Zapier	$0 - 100 tasks $19.99D - 750 tasks
Github	безкоштовно

Найпривабливішими є безкоштовні варіанти.

Google Drive for desktop - як описано у розділі 1.1.1, цей підхід не влаштовує нас. Навантаження перепадає на користувача, тому тут неминучі помилки.

Вариат з Github теж відпадає, в розділі 1.1.7 обґрунтовується чому.

З економічної точки зору, якщо порівнювати Zoom Cloud та Google Drive, останній значно дешевший. Тому є сенс переносити та зберігати видеозаписи в університетській хмарі Google Drive. Розробка додатка ведеться з точки зору використання в навчальному закладі, де працює багата кількість викладачів. Херсонський державний університет нараховує понад 350

викладачів[13]. Якщо навіть використовувати найдешевший додаток, загальна сума коштів становить понад $2000 щомісяця.

Крім того існує дуже важлива тема безпеки даних. Перш ніж надавати дозволи на доступ стороннім особам або організаціям, важливо ретельно перевірити їхні облікові дані та визначити рівень доступу, який їм потрібен. Також важливо встановити чітку політику та процедури надання та відкликання дозволів на доступ. Крім того, важливо регулярно переглядати дозволи на доступ і проводити аудит безпеки, щоб виявити будь-які потенційні вразливості або ризики для безпеки. Все це потребує додаткових зусиль та ресурсів.

Зрозуміло, що виникає питання розробки власного додатку.

1.3 Уточнена постановка задачі на розробку ПЗ

- Переглянути дані, які потрібно перенести. Визначити типи файлів, їхні розміри та кількість файлів, які потрібно перенести.
- Визначити структуру папок для файлів у Google Диску, який буде створено для зберігання переданих файлів.
- Переконатися, що Google Диск налаштований і пристосований для використання в організації. Перевірити дозволи та права доступу запису файлів у Google Диску.
- Автентифікувати обліковий запис Zoom, який буде використовуватися для передачі. Переконайтеся, що обліковий запис має необхідні дозволи для доступу до потрібних даних.
- Створити токен API від Zoom, який буде використовуватися для доступу до API Zoom.
- Написати скрипт, який отримає доступ до API Zoom за допомогою токена API і завантажить необхідні файли. Скрипт також повинен завантажити файли у відповідну папку на Google Диску.
- Протестувати скрипт, щоб переконатися, що він працює належним чином. Переконатися, що файли завантажуються з Zoom і вивантажуються на Google Диск без помилок і втрати даних.
- Запланувати перенесення, щоб переконатися, що воно відбудеться у зручний для організації та користувачів час. Враховувати будь-які обмеження або політики щодо передачі даних, які можуть діяти в організації.
- Після завершення передачі переконатися, що всі файли були успішно перенесені на Google Диск. Переконатися, що до файлів є доступ і що вони зберігаються у відповідних папках.

1.3.1 Вибору мови програмування

При виборі мови програмування для додатку слід враховувати такі критерії:

- Складність додатку - чим складніший додаток, тим більш потужною і гнучкою повинна бути мова програмування.
- Функціональність додатку - мова програмування повинна підтримувати всі необхідні функції для додатку.
- Ефективність додатку - мова програмування повинна бути ефективною, щоб додаток працював швидко і без збоїв.
- Доступність - мова програмування повинна бути доступною, щоб її було легко вивчити і використовувати.
- Спільнота - мова програмування повинна мати велику і активну спільноту розробників, щоб можна було отримати допомогу і підтримку.

Для додатку перенесення даних з Zoom Cloud до Google Drive можна використовувати такі мови програмування:

Python - це популярна і потужна мова програмування, яка підходить для створення широкого спектру додатків. Python має велику і активну спільноту розробників, а також безліч бібліотек і фреймворків, які можна використовувати для створення додатку.

Java - це ще одна популярна і потужна мова програмування, яка підходить для створення корпоративних додатків. Java має велику і активну спільноту розробників, а також безліч бібліотек і фреймворків, які можна використовувати для створення додатку.

JavaScript - це мова програмування, яка використовується для створення веб-додатків. JavaScript можна використовувати для створення веб-інтерфейсу для додатку перенесення даних.

Для нашого додатку ми виберемо Python. Python підходить для створення цього додатку, оскільки він:

- Потужний і гнучкий - Python може використовуватися для створення широкого спектру додатків, включаючи веб-додатки, мобільні додатки, ігри та інші.
- Ефективний - Python може бути ефективним для створення додатків, які працюють швидко і без збоїв.
- Доступний - Python є безкоштовним і відкритим кодом, що робить його доступним для всіх.
- Спільнота - Python має велику і активну спільноту розробників, яка може надати допомогу і підтримку.

Python є найкращим вибором для розробки додатку, оскільки він відповідає всім наведеним критеріям. Він є потужною та універсальною мовою програмування, яка може використовуватися для створення широкого спектру додатків. Крім того, Python є мовою з гарною перспективою на майбутнє, яка підтримується компанією Google і продовжує розвиватися [45].

1.3.2. Архітектура додатку

Додаток складається з двох основних компонентів, які виконують операції з Zoom API та Google Drive API.

Для авторизації користувача в Zoom клієнтський компонент використовує OAuth 2.0. Для цього потрібно створити додаток типу JWT App Type або Server-to-Server OAuth. Після отримання токена доступу клієнтський компонент може виконувати операції з Zoom API, зокрема отримувати список записів Zoom.

Для завантаження записів Zoom у Google Drive клієнтський компонент використовує Google Drive API. Для цього потрібно створити обліковий запис для доступу до Google Drive API. Після отримання облікового запису клієнтський компонент може виконувати операції з Google Drive API, зокрема створювати файли та папки.Для авторизації користувачів серверний компонент використовує OAuth 2.0. Для цього потрібно створити додаток типу JWT App Type або Server-to-Server OAuth. Після авторизації користувача серверний компонент зберігає токен доступу в локальній базі даних.

Клієнтський компонент взаємодіє з серверним компонентом за допомогою HTTP запитів. Клієнтський компонент відправляє запит на серверний компонент для отримання токена доступу. Після отримання токена доступу клієнтський компонент використовує його для виконання операцій з Zoom API та Google Drive API.

Архітектура додатку має такі переваги:

- **Простота реалізації.** Додаток розроблений на Python, що робить його простим у реалізації.
- **Безпека.** Авторизація користувачів здійснюється за допомогою OAuth 2.0.
- **Ефективність.** Додаток використовує відповідні API для доступу до Zoom API та Google Drive API.

Архітектура додатку має такі недоліки:

- **Централізованість.** Серверний компонент є єдиним компонентом, який відповідає за авторизацію користувачів.
- **Недостатня масштабованість.** Серверний компонент може стати вузьким місцем для додатку, якщо кількість користувачів буде великою.

Архітектура додатку є простою та ефективною. Додаток розроблений на Python, що робить його простим у реалізації та підтримці. Авторизація користувачів здійснюється за допомогою OAuth 2.0, що забезпечує безпеку додатку. Додаток використовує відповідні API для доступу до Zoom API та Google Drive API, що забезпечує ефективність додатку.

РОЗДІЛ 2. АЛГОРИТМИ ВЗАЄМОДІЇ КОМПОНЕНТІВ ДОДАТКУ

Алгоритмом взаємодії компонентів додатку є використання синхронних API Zoom Cloud і Google Drive. У цьому випадку додаток не потребує сервера для зберігання файлів. Додаток просто використовує API Zoom Cloud для завантаження файлів з Zoom Cloud і API Google Drive для завантаження файлів у Google Drive. Алгоритм з синхронними API простий у реалізації та ефективний, оскільки компонент, який викликає API, не блокується, поки API не завершить свою роботу.

Алгоритм взаємодії компонентів можна представити в наступному вигляді:

Рис. 2.1 Алгоритм взаємодії компонентів.

1. Запуск додатку

- Користувач запускає додаток, використовуючи командний рядок, або додаток автоматично виконується у відповідності з певним розкладом.
- Додаток отримує доступ до облікового запису Zoom Cloud, використовуючи API Zoom Cloud.
- Додаток отримує доступ до облікового запису Google Drive, використовуючи API Google Drive.

2. Вибір файлів для перенесення

- Додаток вибирає файли, які потрібно перенести. Це визначається параметрами командного рядка.
- Додаток отримує інформацію про вибрані файли, включаючи їхні назви, розміри та типи, використовуючи API Zoom Cloud.

3. Перенесення файлів

- Додаток завантажує вибрані файли з Zoom Cloud на сервер додатку за допомогою протоколу передачі даних, такого як HTTP або FTP.
- Додаток завантажує файли з сервера додатку в Google Drive.

4. Завершення перенесення

- Додаток інформує про завершення перенесення за допомогою GUI або командного рядка.

2.1 Zoom API

Zoom API надає розробникам можливість легко інтегрувати широкі сервіси відеоконференцій і нарад Zoom у свої додатки, покращуючи функціональність і користувацький досвід. API-інтерфейси Zoom входять до складу платформи для розробників, яка включає не лише API, але й веб-хуки та SDK, пропонуючи комплексний набір інструментів для налаштування та використання екосистеми Zoom[17].

Одним з випадків використання Zoom API є автоматизація робочого процесу. Zoom API надає можливість спростити роботу, автоматизувавши керування подіями в бізнес-процесах у відповідь на певні тригери.

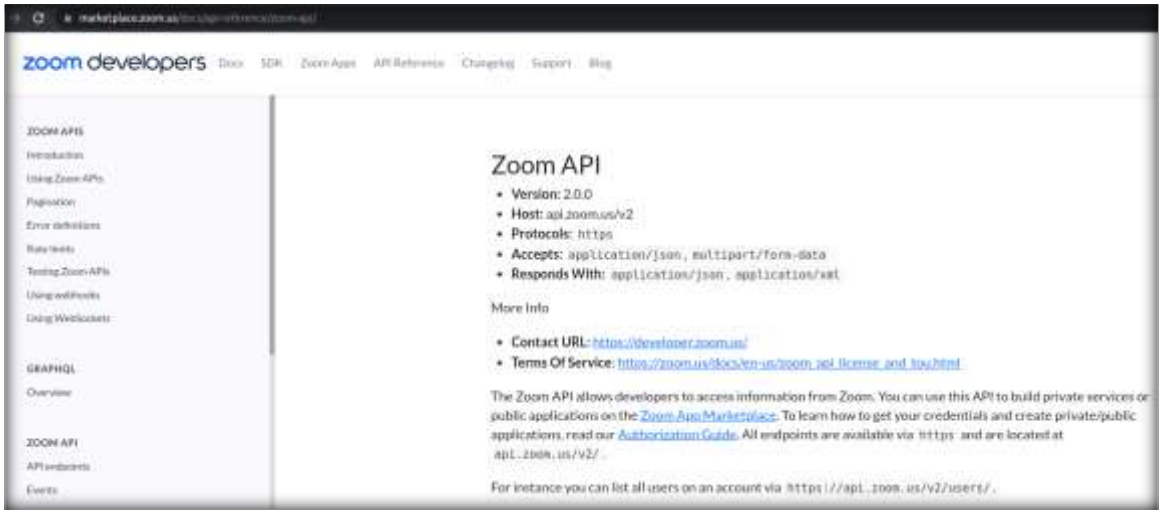


Рис. 2.1 - Zoom API.

Щоб почати використовувати Zoom API, розробникам необхідно створити обліковий запис Zoom, зареєструвати свій додаток в Zoom Marketplace, щоб отримати облікові дані API (JWT або OAuth токен), і ознайомитися з документацією Zoom API. Розуміння принципів RESTful та досвід роботи з форматами даних JSON є дуже важливими, оскільки Zoom API - це RESTful сервіс, який повертає дані у форматі JSON.

Універсальність і простота інтеграції Zoom API роблять його привабливим варіантом для компаній і розробників, які бажають інтегрувати можливості відеоконференцій у свої додатки, розширюючи спектр послуг, які вони можуть запропонувати, і підвищуючи ефективність своєї діяльності.

2.1.1 Додаток типу JWT App Type

Для автентифікації доступу на рівні облікового запису в API Zoom використовувався тип додатків JSON Web Tokens (JWT). Токени JWT пропонують метод встановлення безпечної автентифікації між серверами шляхом передачі компактного JSON-об'єкта з підписаним корисним навантаженням, що містить ключ і секрет API вашого облікового запису [20].

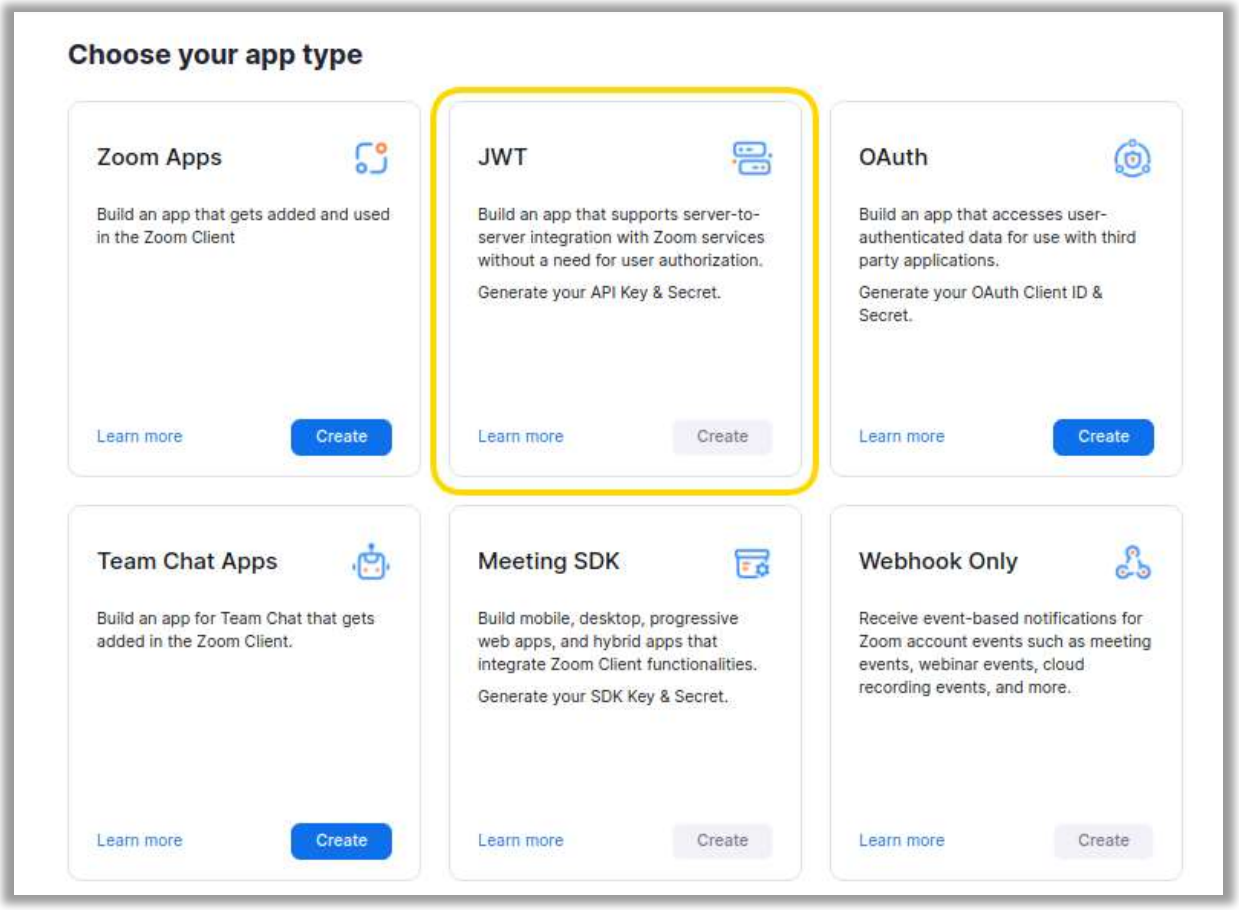


Рис. 2.2 - Створення додатку типу JWT App Type.

Однак були виявлені великі недоліки в безпеці використання додатків такого типу. Через природу інтеграцій JWT, будь-хто може створити інтеграцію, не маючи облікового запису Zoom. Zoom не має можливості відстежувати ці інтеграції або їхніх розробників.

Тому компанією Zoom було вирешено відмовитися від додатків такого типу. Нижче наведено скріншот з електроного листа, якій було отримано адміністратором додатку типу JWT App Type.

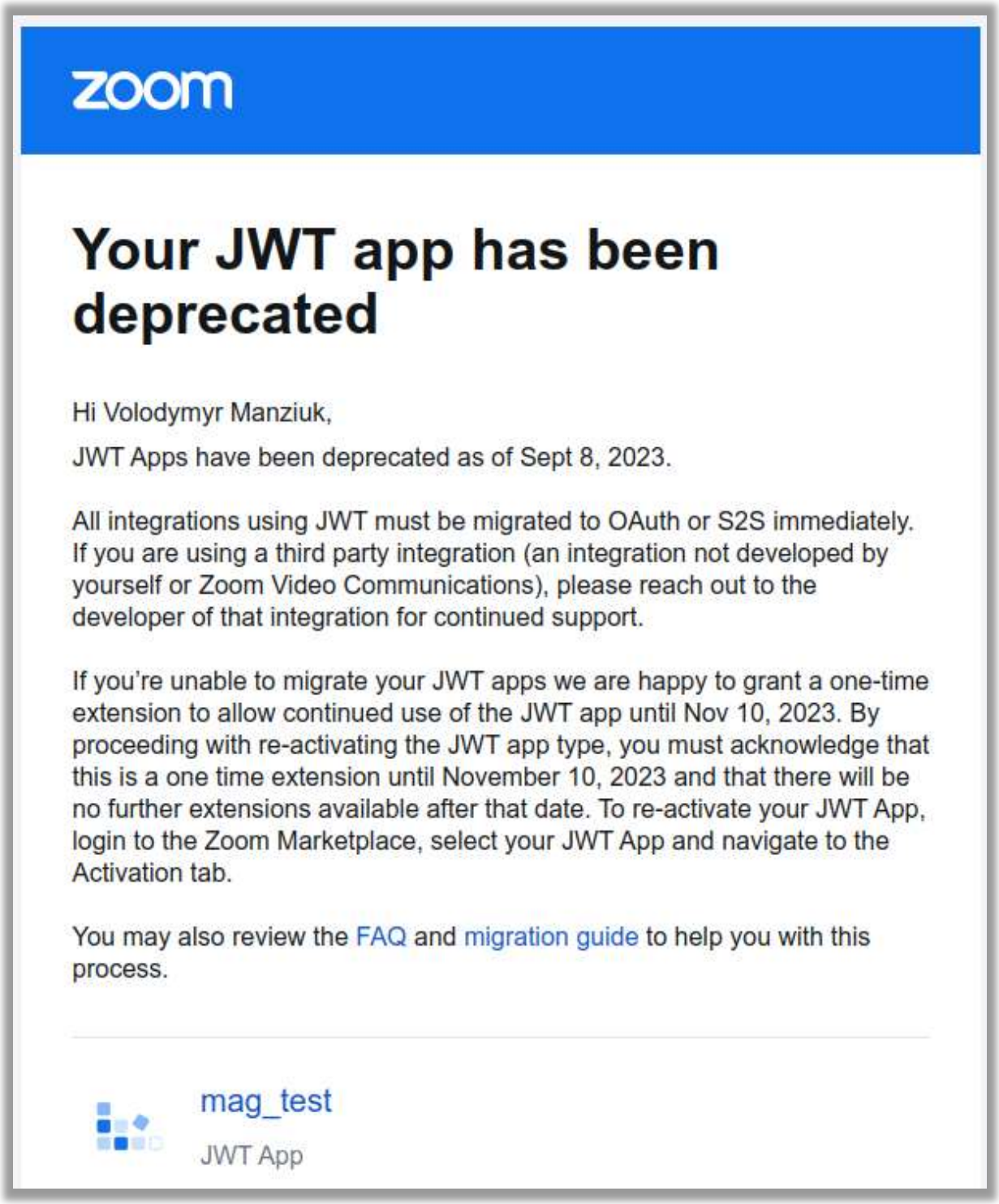


Рис. 2.3 - Попередження про припинення підтримки додатку JWT App Type (скріншот з електроного листа).

Zoom попереджає, що підтримка додатку JWT App Type буде повнійстью припинена 31 січня 2024 року. Zoom рекомендує розробникам, які використовують додатки JWT App Type, перейти на інший тип додатка,

наприклад, OAuth App Type. OAuth App Type є більш безпечним і надійним типом додатка, який використовує протокол OAuth 2.0 для аутентифікації та авторизації користувачів.

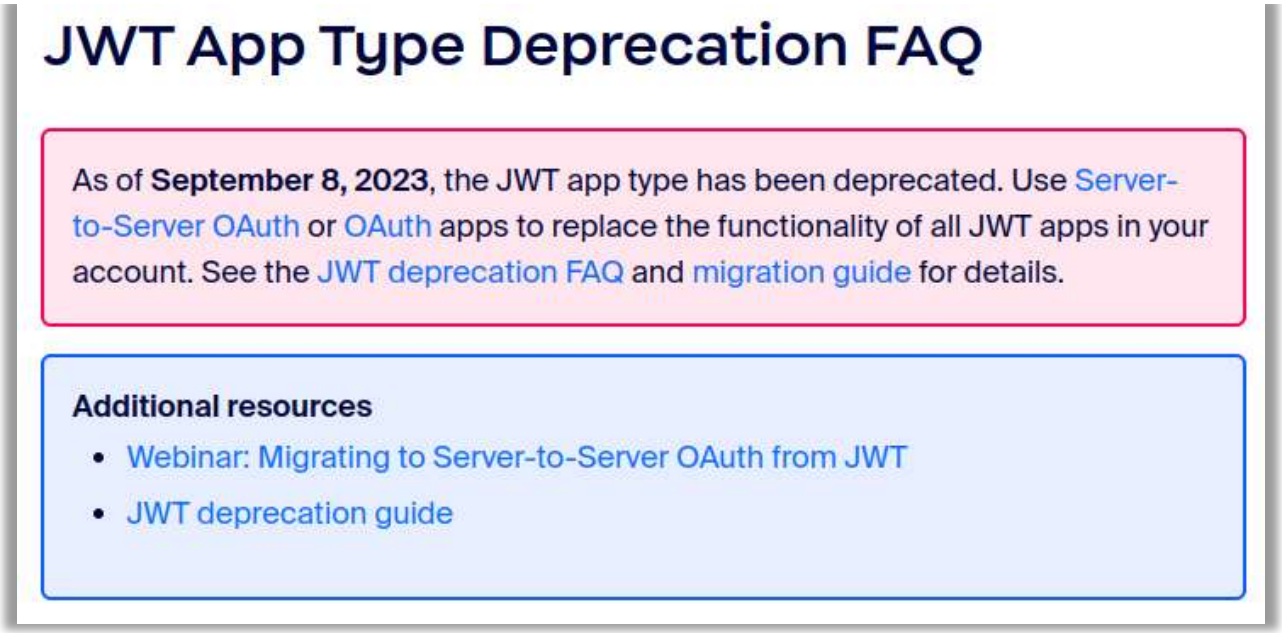


Рис. 2.4 - Попередження про припинення підтримки додатку типу JWT App Type на офіційному сайті [21].

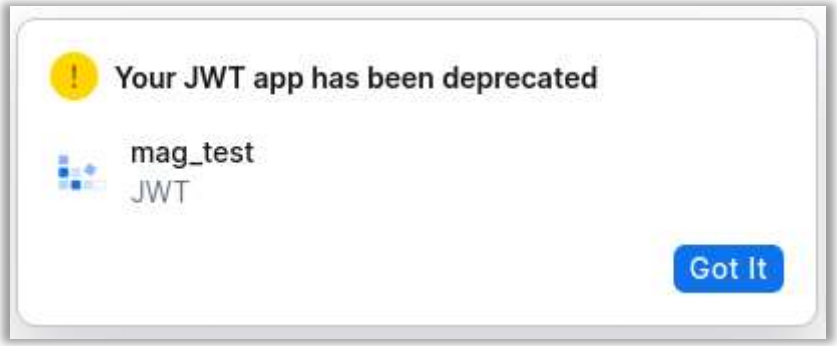


Рис. 2.5 - Попередження про припинення підтримки додатку типу JWT App Type на сторінке розробника Zoom App Marketplace.

2.1.2 Додаток типу Server-to-Server OAuth

Server-to-Server OAuth додатки є відносно новим типом автентифікації, введеним Zoom для забезпечення більш безпечного та ефективного способу управління доступом до API на рівні сервера. Ця модель автентифікації відрізняється від традиційного OAuth 2.0, зокрема своєю здатністю дозволяти серверам взаємодіяти з Zoom API без необхідності вручну обмінюватися токенами доступу та залучення кінцевого користувача для авторизації.

Додаток типу Server-to-Server OAuth має ряд переваг перед іншими типами додатків Zoom, включаючи:

- Більшу безпеку: цей тип додатку не вимагає від користувачів Zoom вводити свої облікові дані в сторонній додаток.
- Більшу надійність: цей тип додатку використовує протокол OAuth 2.0, який є стандартом у галузі.
- Більшу гнучкість: цей тип додатку дозволяє розробникам отримувати доступ до більш широкого спектру ресурсів Zoom.

Щоб створити Server-to-Server OAuth, розробники повинні створити обліковий запис розробника Zoom і зареєструвати свій додаток на веб-сайті Zoom Developer Platform. Після реєстрації додатка розробники отримають секрети додатка, які вони можуть використовувати для аутентифікації свого додатка в Zoom API [22].

Server-to-Server OAuth app - це потужний інструмент, який можна використовувати для інтеграції широкого спектру функцій Zoom у власні програми. Цей тип додатка є більш безпечним і надійним, ніж інші типи додатків Zoom, і він пропонує розробникам більше можливостей.

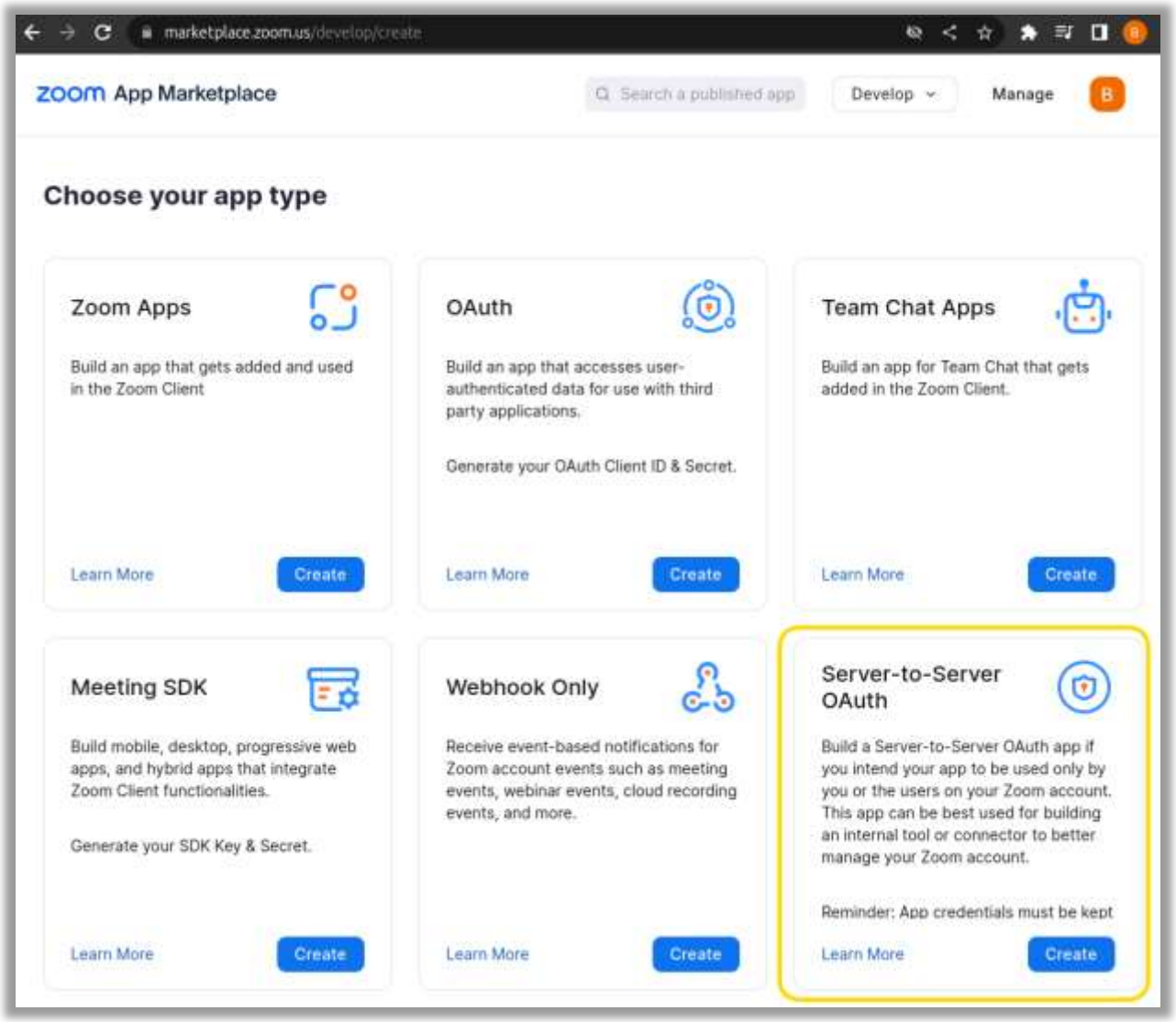


Рис. 2.6 - Створення додатку типу Server-to-Server OAuth.

2.1.3 Фреймворк авторизації OAuth 2.0

OAuth 2.0 - це широко розповсюджений фреймворк авторизації, який дозволяє програмам отримувати обмежений доступ до облікових записів користувачів на HTTP-сервісах, таких як Facebook, GitHub, Google та сервіси Microsoft. Він покликаний забезпечити більш безпечний доступ до ресурсів, ніж попередні методи, особливо коли йдеться про розкриття облікових даних, таких як імена користувачів і паролі [18].

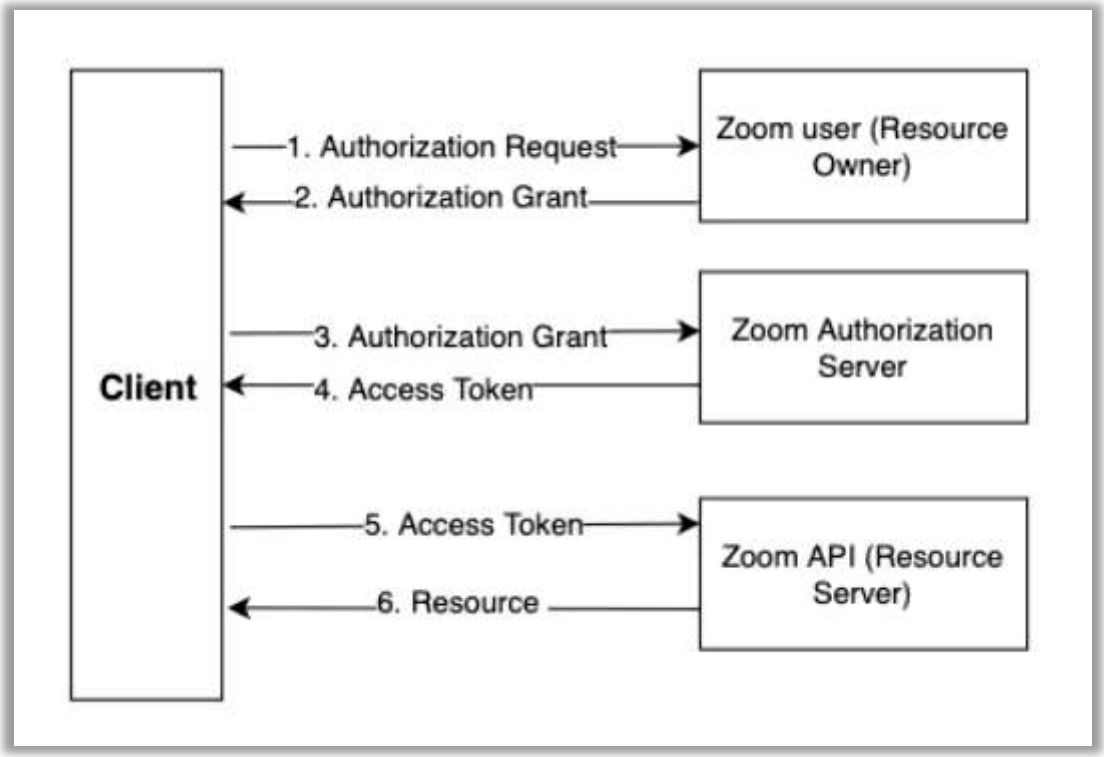


Рис. 2.7 - Взаємодія між клієнтом (додатком), користувачем Zoom, сервером авторизації Zoom та API Zoom (Автентифікація додатку типу OAuth).

OAuth 2.0 - це безпечний і надійний спосіб доступу сторонніх додатків до облікових записів користувачів на HTTP-сервісах. Він надає ряд переваг у порівнянні з іншими протоколами авторизації, зокрема:

- *Покращений користувацький досвід:* OAuth 2.0 дозволяє користувачам авторизувати сторонні додатки для доступу до свого облікового запису без необхідності повідомляти свій пароль.

- *Підвищена безпека:* OAuth 2.0 знижує ризик фішингових атак та інших порушень безпеки.
- *Підвищена гнучкість:* OAuth 2.0 дозволяє стороннім додаткам отримувати доступ до різноманітних даних користувача, включаючи інформацію про профіль, фотографії та контакти.

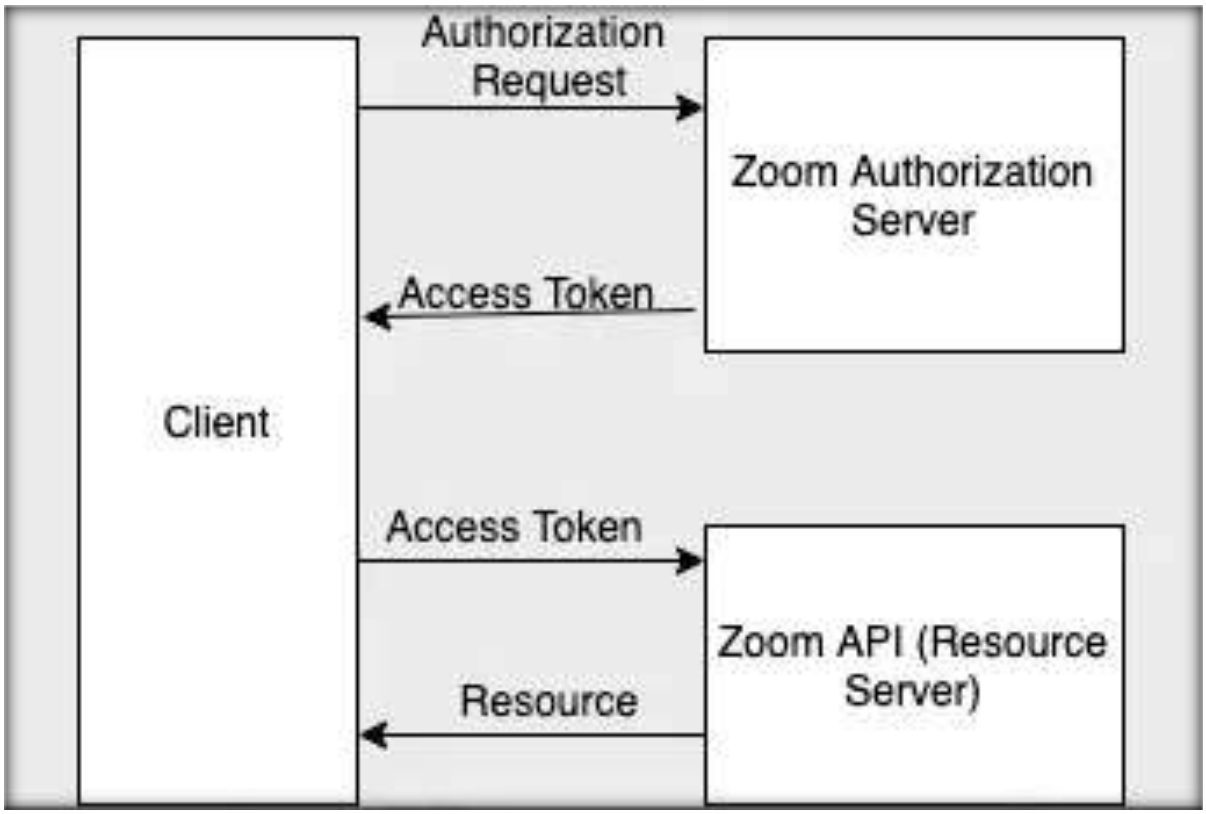


Рис. 2.8 - Автентифікація додатку Server-to-Server.

OAuth 2.0 - це потужний інструмент, який можна використовувати для безпечної та надійної авторизації сторонніх додатків для доступу до облікових записів користувачів на HTTP-сервісах. Це широко використовуваний протокол, який підтримується широким спектром додатків і сервісів.

2.1.4 Отримання токену доступу

Отримання токена доступу додатку тип Server-to-Server OAuth - це процес, який дозволяє додатку отримати дозвіл на доступ до даних Zoom. Токен доступу є тимчасовим і дає додатку обмежений доступ до вашого облікового запису.

Щоб отримати токен доступу, по-перше потрібно створити додаток Zoom Server-to-Server OAuth [23]. Після створення додатку буде надано секретний ключ. Цей ключ необхідно використовувати для отримання токена доступу.

Для отримання токену доступу, необхідно виконати наступний POST HTTP запит [24], для прикладу використовується інструмент командной строки *curl* [37]:

```
curl -X POST https://zoom.us/oauth/token -d 'grant_type=account_credentials' \
-d 'account_id={accountID}' -H 'Host: zoom.us' \
-H 'Authorization: Basic Base64Encoded(clientId:clientSecret)'
```

параметри:

- account_id: ідентифікатор облікового запису Zoom.
- client_id: ідентифікатор додатка Zoom.
- client_secret: секретний ключ додатка Zoom.

Успішною відповідь JSON буде включати токен доступу, який є типом Bearer токен, термін дії якого закінчується через годину, з областю застосування (scopes), яку було обрано при налаштувані додатку:

{

“access_token“: [String],

“token_type”: “bearer”,

“expires_in”: long,

“scope” : [String]

}

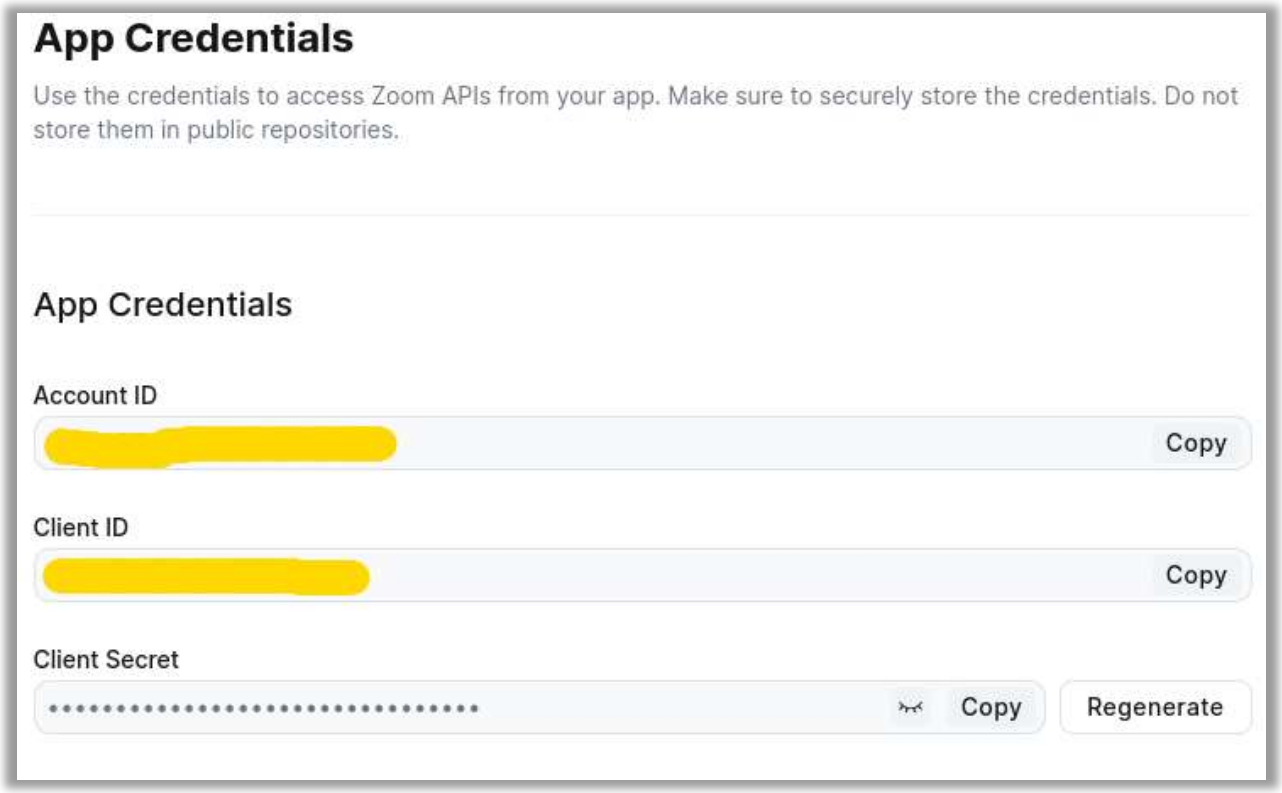


Рис. 2.9 - Облікові дані додатку Server-to-Server в панелі адміністратора Zoom App Marketplace.

2.1.5 Операції з хмарними записами в Zoom API

Zoom API надає доступ до хмарних записів Zoom, які є записами відеоконференцій, збережених на серверах Zoom. Хмарні записи Zoom можна використовувати для перегляду, завантаження, редагування та видалення [25].

Zoom API надає наступні операції з хмарними записами:

- **Створення хмарного запису:** Ця операція створює новий хмарний запис на основі відеоконференції.
- **Отримання хмарного запису:** Ця операція отримує інформацію про хмарний запис, наприклад, його назву, дату та час створення, тривалість та розмір.
- **Перегляд хмарного запису:** Ця операція дозволяє переглядати хмарний запис.
- **Завантаження хмарного запису:** Ця операція завантажує хмарний запис на локальний комп'ютер чи сервер.
- **Редагування хмарного запису:** Ця операція дозволяє редагувати хмарний запис, наприклад, додавати або видаляти фрагменти.
- **Видалення хмарного запису:** Ця операція видаляє хмарний запис з серверів Zoom.

Приклади операцій з хмарними записами в Zoom API

- **Отримати список усіх користувачів [27]**

```
curl -X GET \
-H "Content-Type: application/json" \
-H "Authorization: Bearer APP_ACCESS_TOKEN" \
https://api.zoom.us/v2/users/?page_size=30&next_page_token=
```

параметри:

- APP_ACCESS_TOKEN: Bearer токен для доступу до API.
- page_size: кількість записів, повернутих протягом одного виклику API.
- next_page_token: використовується для розбиття на сторінки великих наборів результатів.

- **Список усіх записів у хмарі для користувача [27]**

```
curl -X GET \
-H "Authorization: Bearer APP_ACCESS_TOKEN" \
https://api.zoom.us/v2/users/{userId}/recordings?from={}&to={}& \
page_size=300&next_page_token=
```

параметри:

- APP_ACCESS_TOKEN: Bearer токен для доступу до API.
- userId: ідентифікатор користувача або адреса електронної пошти.
- from: дата початку для діапазону дат, коли ви хочете отримати записи. Максимальний діапазон може становити місяць.
- to: дата закінчення.
- page_size: кількість записів, повернутих протягом одного виклику API.
- next_page_token: використовується для розбиття на сторінки великих наборів результатів.

- **Завантажити окремий запис користувача**

Щоб завантажити файл запису використовується властивість *download_url*, указана у відповіді на запит усіх записів користувача. Посилання на завантаження має вигляд:

https://{{base-domain}}/rec/archive/download/xyz

параметри:

base-domain: для нашого додатку це буде **ksu-ks-ua.zoom.us**

- **Видалити окремий запис користувача [28]**

```
curl -X DELETE \
-H "Authorization: Bearer APP_ACCESS_TOKEN" \
https://api.zoom.us/v2/meetings/{meetingId}/recordings/{recordingId}?action=trash
```

параметри:

- meetingId: ідентифікатор .
- recordingId: ідентифікатор записів, API.
- action: дії видалення запису, trash - перемістити запис у кошик.

2.2 Google Drive API

Google API - це універсальний API, який можна використовувати для взаємодії з різноманітними продуктами та сервісами Google, наприклад з Google Maps, YouTube, Google Drive, Gmail, Google Search, Google Translate, і багато інших. Ці API дають розробникам доступ до функціональностей цих сервісів, дозволяючи використовувати їх у власних додатках.

Google Drive API - це спеціалізований API, який можна використовувати для взаємодії з Google Drive, хмарним сервісом зберігання даних [**Ошибка! Источник ссылки не найден.**]. Google Drive API дозволяє розробникам створювати додатки, які взаємодіють з Google Drive, включаючи читання, запис, та управління файлами та папками.

Основні можливості Google Drive API:

- CRUD операції: створення (Create), читання (Read), оновлення (Update) та видалення (Delete) файлів та папок.
- Управління дозволами: налаштування доступу та дозволів до файлів та папок.
- Завантаження файлів: передача файлів між локальною системою та Google Drive.
- Пошук файлів: пошук файлів за різними параметрами, такими як ім'я, тип, власник і так далі.
- Діліться файлами та папками з іншими користувачами.
- Трансляція відео- та аудіофайлів зі сховища Google Drive.
- Перетворення файлів з одного формату в інший.

Щоб використовувати Google Drive API, потрібно створити проект Google Drive і отримати ідентифікатор клієнта та секрет клієнта. Отримавши необхідні облікові дані, можна використовувати їх для авторизованих запитів до Google Drive API.

2.2.1 Використання OAuth 2.0 для доступу до Google API

OAuth 2.0 - це відкритий стандарт авторизації, який дозволяє користувачам відкривати доступ до своїх приватних даних (файли, фотографії, відео, списки контактів), що зберігаються на одному сайті, іншому сайту, без необхідності вводу імені користувача та паролю.

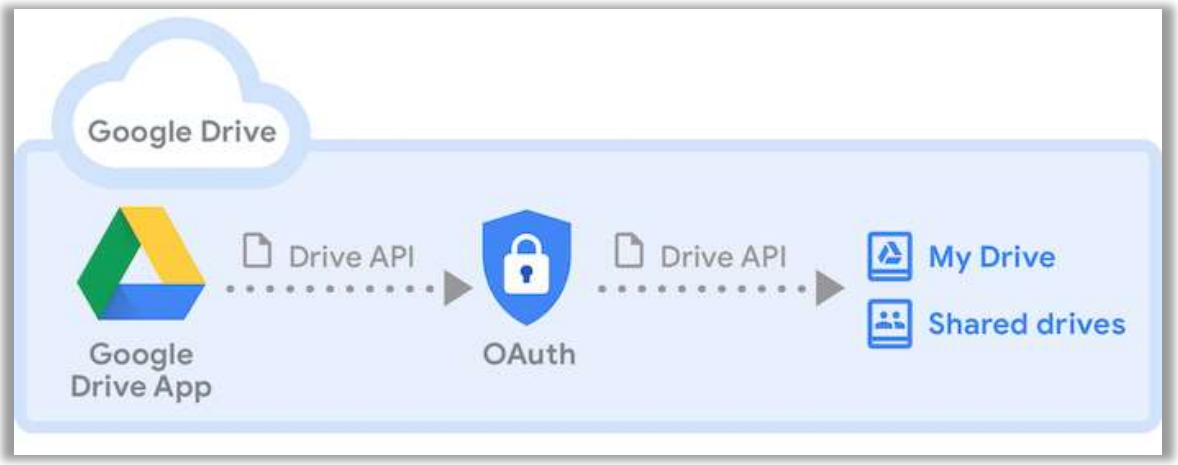


Рис. 2.10 - OAuth 2.0 для доступу до Google API [**Ошибка! Источник ссылки не найден.**].

Щоб використовувати Google API, потрібно отримати токен доступу OAuth 2.0. Токен доступу - це тимчасовий ключ, який дає додатку обмежений доступ до даних Google [6].

У контексті OAuth 2.0, токен доступу дозволяє клієнтській програмі отримати доступ до певного ресурсу для виконання певних дій від імені користувача. Це те, що відомо як сценарій делегованої авторизації: користувач делегує клієнтській програмі доступ до ресурсу від свого імені. Це означає, наприклад, що можна дозволити додатку отримати доступ до Google Drive API від імені користувача. Але також молжливо обмежити поведінку додатку. Це обмеження є дуже важливим у сценарії делегованої авторизації та досягається за допомогою область дії (scopes). Області дії – це

механізм, який дозволяє користувачеві авторизувати програму третьої сторони для виконання лише певних операцій.

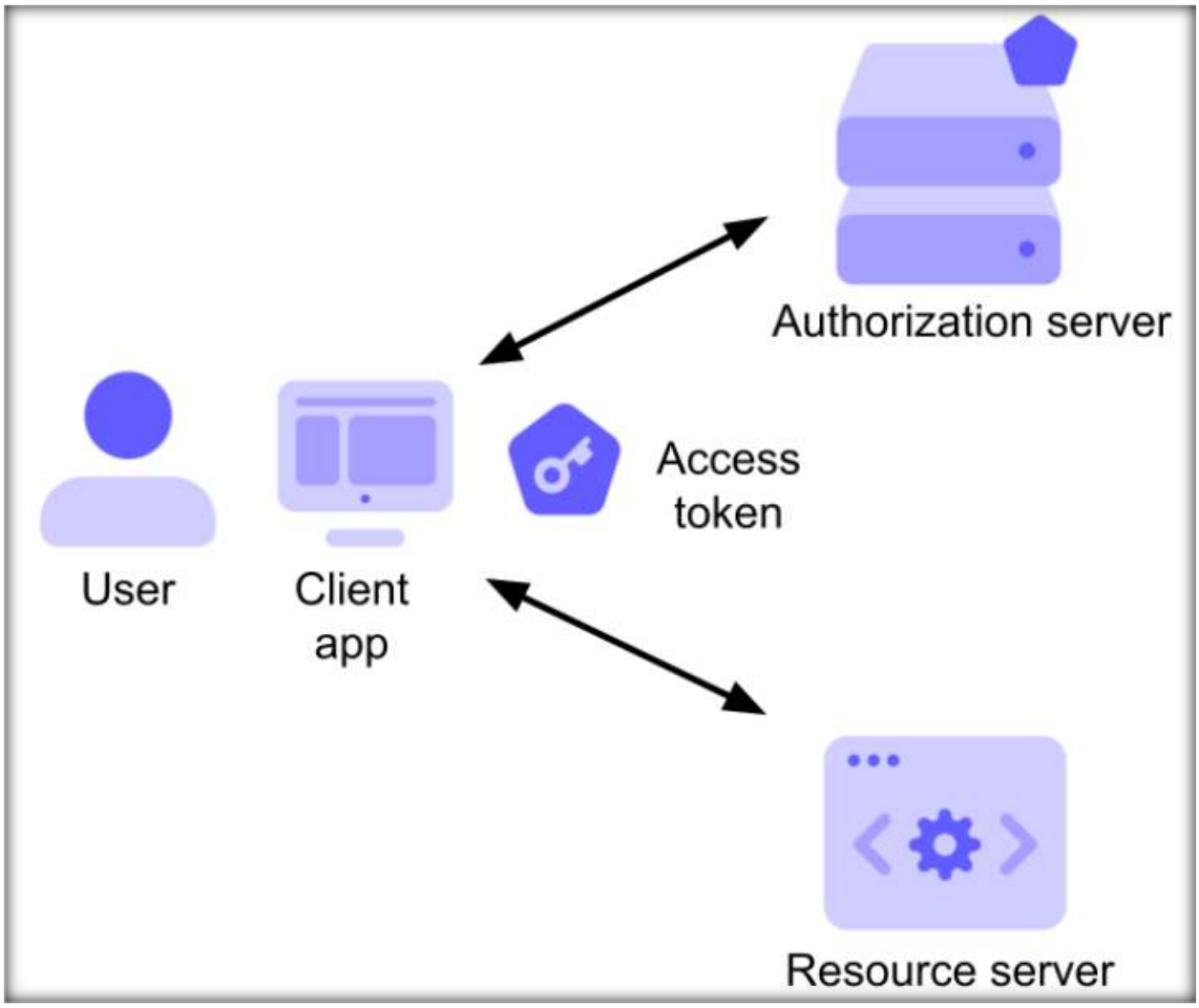


Рис. 2.11 - Отримання доступу додатком до ресурсів користувача [30].

Загальний огляд процесу використання OAuth 2.0 для доступу до Google API:

1. Створення проекту в Google Cloud Platform (консоль GCP):

- В Google Cloud Console (https://console.cloud.google.com/) створюємо новий проект або вибираємо існуючий.
- Включаємо API, який нам потрібен (наприклад, Google Drive API) в менеджері бібліотеки API.

2. Створення облікових даних:

- У консолі GCP у розділі "Облікові дані" створюємо облікові дані та вибираємо "ID клієнта OAuth".
- Вибираємо тип додатка (наприклад, веб-додаток, Android, iOS).

3. Отримання та Збереження ID клієнта та секрету клієнта:

- Після створення ID клієнта та секрету клієнта, зберегаємо дані. Ці облікові дані будуть використовуватися в додатку для аутентифікації через Google.

4. Імплементація OAuth 2.0 у додатку:

- Використовуємо обрану бібліотеку OAuth 2.0.
- Перенаправляємо користувача до Google для аутентифікації та авторизації. Користувач вибирає свій обліковий запис Google та надає дозволи, якщо він їх ще не надавал.
- Обробляємо код перенаправлення, який Google надсилає додатку після авторизації користувача.
- Використовуємо цей код для отримання токенів доступу та оновлення від сервера Google.

5. Використання токенів доступу:

- Використовуємо отриманий токен доступу для виконання запитів до Google API від імені користувача. Токени доступу зазвичай мають обмежений термін дії, тому може знадобитися використати токен оновлення для отримання нового токену доступу, коли старий закінчується.

6. Обробка токена оновлення:

- Для додатків, які потребують тривалого доступу до ресурсів користувача, використовуємо токен оновлення для отримання нових токенів доступу, коли поточний токен спливає.

Використання OAuth 2.0 дозволяє вашому додатку безпечно та ефективно взаємодіяти з Google API, забезпечуючи доступ до ресурсів Google від імені користувачів.

2.2.2 Обліковий запис для доступу до Google Drive API

Система Google OAuth 2.0 підтримує міжсерверну взаємодію, наприклад між додатком і службою Google. Для цього сценарію нам потрібен сервісний обліковий запис (service account), який належить нашому додатку, а не окремому кінцевому користувачеві. Наш додаток буде викликати Google Drive API від імені сервісного облікового запису, тому користувачі не беруть безпосередньої участі. Цей сценарій іноді називають «двостороннім OAuth» або «2LO» [5].

Кроки для створення Service Account для доступу до Google Drive API:

1. На Google Cloud Console вибіраємо проект або створюємо новий.

2. В панелі "Бібліотека" у розділі "API & Services" підключаємо "Google Drive API".

3. Створюємо облікові дані та вибераємо "Service Account". Після створення, на сторінці service account, натисніть на створений аккаунт для налаштування деталей та управління ключами.

4. У розділ "Ключі" створюємо ключ у форматі JSON та завантажуємо його. Зберегаємо файл ключа у безпечному місці.

5. Додавання Service Account до Google Drive

Для того щоб Service Account міг отримувати доступ до певних файлів або папок у Google Drive, додаємо електронну адресу Service Account (знайдете у деталях Service Account в Google Cloud Console) як користувача з відповідними правами доступу (читання, редагування тощо) до потрібних файлів або папок.

У коді нашого додатку будемо використовувати завантажений файл ключа для аутентифікації через бібліотеку клієнта Google API.

При аутентифікації вказуємо область дії (scopes) відповідно до потреб додатку в доступі до Google Drive API.

Використовуючи Service Account, ми маємо можливість взаємодіяти з Google Drive API на сервері без необхідності ручної аутентифікації користувача, що дуже зручно для автоматизованих фонових процесів та сервер-сервер взаємодій.

2.2.3 Керування файлами та папками у Google Drive API

Google Drive API дозволяє керувати файлами та папками у Google Drive за допомогою HTTP запитів. API використовується для створення, читання, оновлення та видалення файлів і папок у обліковому записі Google Drive. Для керування файлами та папками у Google Drive API через Service Account або через аутентифікацію OAuth 2.0, ми можемо виконувати наступні дії [34]:

Для всіх прикладів використовуються:

- APP_ACCESS_TOKEN: Bearer токен для доступу до API.
- APP_API_KEY: унікальний ключ, який дає додатку доступ до API.

- **Завантаження файлів.** Використовується метод files.create() для завантаження нових файлів на Google Drive.

```
curl --request POST \
'https://www.googleapis.com/drive/v3/files?key=[APP_API_KEY]' \
--header 'Authorization: Bearer [APP_ACCESS_TOKEN]' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '{}' \
--compressed
```

параметри:

- data '{}': тіло POST запросу наступного вигляду

```
{
"mimeType": "",
"name": "",
"parents": [""]
}
```

- **Читання файлів.** Використовується files.get() для отримання інформації про файл. Щоб прочитати вміст файлу, використовуйте files.export() для Google Docs файлів або files.get() з параметром alt=media для інших типів файлів.

```
curl \
'https://www.googleapis.com/drive/v3/files/[FILEID]?key=[YOUR_API_KEY]' \
--header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]' \
--header 'Accept: application/json' \
--compressed
```

параметри:

FILEID: ідентифікатор файлу.

- **Видалення файлів.** Використовується files.delete() для видалення файлів.

```
curl --request DELETE \
'https://www.googleapis.com/drive/v3/files/[FILEID]?key=[YOUR_API_KEY]' \
--header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]' \
--header 'Accept: application/json' \
--compressed
```

параметри:

FILEID: ідентифікатор файлу.

- **Створення папок.** Папки у Google Drive є звичайними файлами з MIME типом *application/vnd.google-apps.folder*, тому будемо створювати їх за допомогою files.create(), вказавши відповідний MIME тип.

```
curl --request POST \
'https://www.googleapis.com/drive/v3/files?key=[APP_API_KEY]' \
```

```
--header 'Authorization: Bearer [APP_ACCESS_TOKEN]' \
--header 'Accept: application/json' \
--header 'mimeType: application/vnd.google-apps.folder' \
--header 'Content-Type: application/json' \
--data '{}' \
--compressed
```

параметри:

- data '{}': тіло POST запросу наступного вигляду

```
{
"mimeType": "",
"name": "",
"parents": [""]
}
```

- **Переміщення файлів та папок.** Використовується метод files.update(), змінюючи параметри addParents та/або removeParents для переміщення файлів чи папок в іншу папку.

```
curl --request PATCH \
'https://www.googleapis.com/drive/v3/files/[FILEID]?addParents={}&removeParents={}&
key=[YOUR_API_KEY]' \
--header 'Authorization: Bearer [YOUR_ACCESS_TOKEN]' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '{}' \
--compressed
```

параметри:

- FILEID: ідентифікатор файлу
- addParents: список ідентифікаторів, розділених комами, які потрібно додати.
- removeParents: Відокремлений комами список ідентифікаторів, які потрібно видалити.
- data '{}': тіло POST запросу наступного вигляду

```
{
"mimeType": "",
"name": "",
"parents": [""]
}
```

Усі ці методи дозволяють фективно інтегрувати та керувати файлами та папками в Google Drive.

Зазвичай, для роботи з Google Drive API потрібно встановити офіційну клієнтську бібліотеку Google для обраної мови програмування, наприклад, для Python це буде google-api-python-client [35].

РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ

3.1 Створення Zoom додатку Server-to-Server OAuth

Було отримано обліковий запис Zoom від ХДУ - volodymyr.manziuk@university.kherson.ua.

Заходимо до https://marketplace.zoom.us/ та створюємо новий додаток.

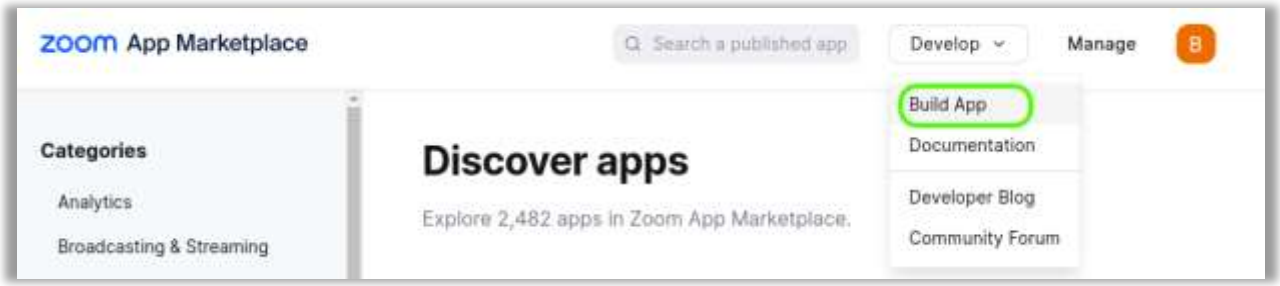


Рис. 3.1 - Створення нового додатку Zoom.

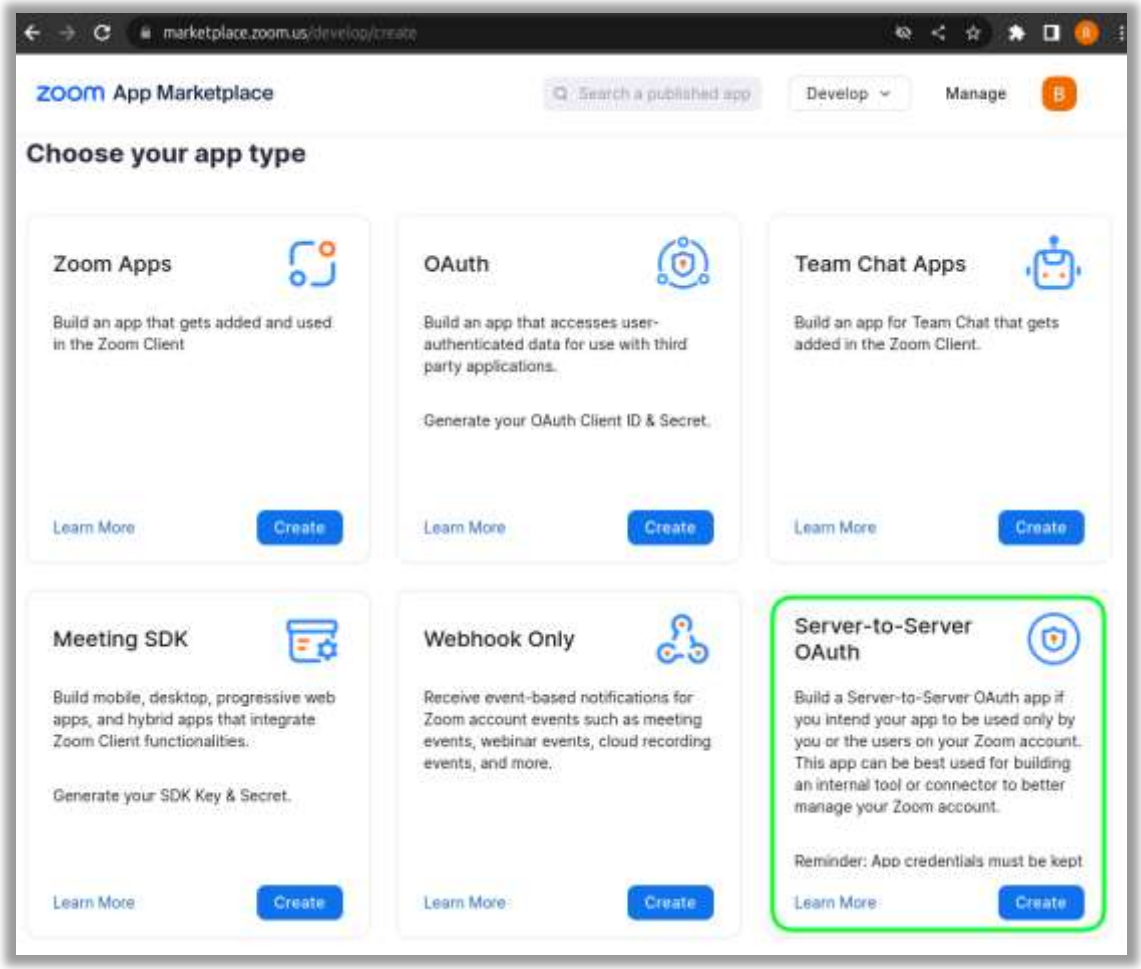


Рис. 3.2 - Створення додатку Server-to-Server OAuth.

Рис. 3.3 - Нове ім’я додатку.

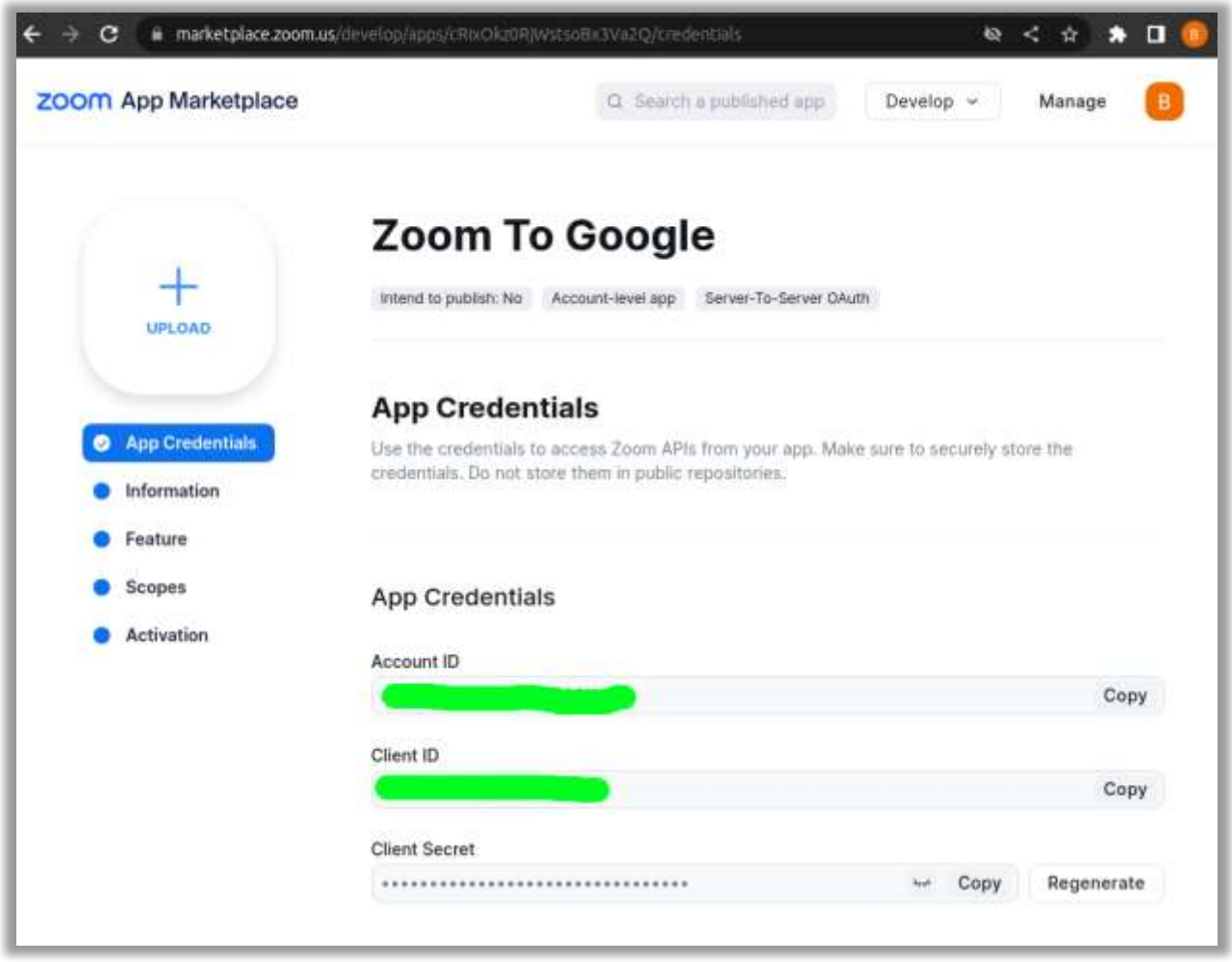


Рис. 3.4 - Отриманні облікові дані додатку.

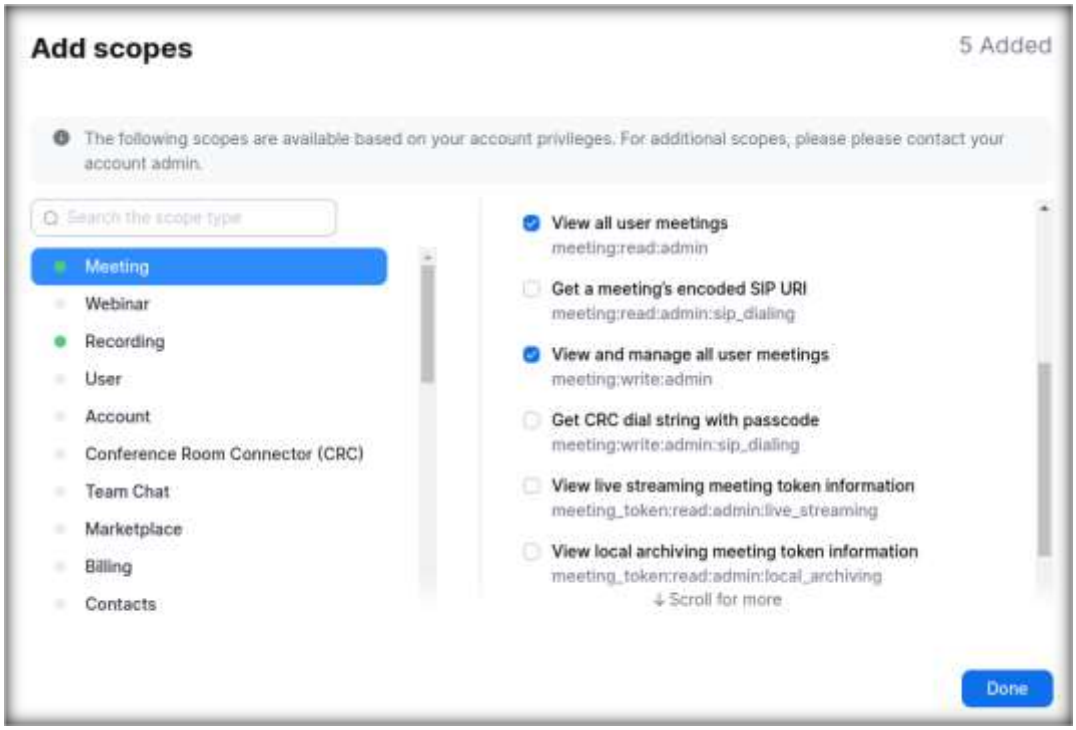


Рис. 3.5 - Області застосування (scopes) додатка.

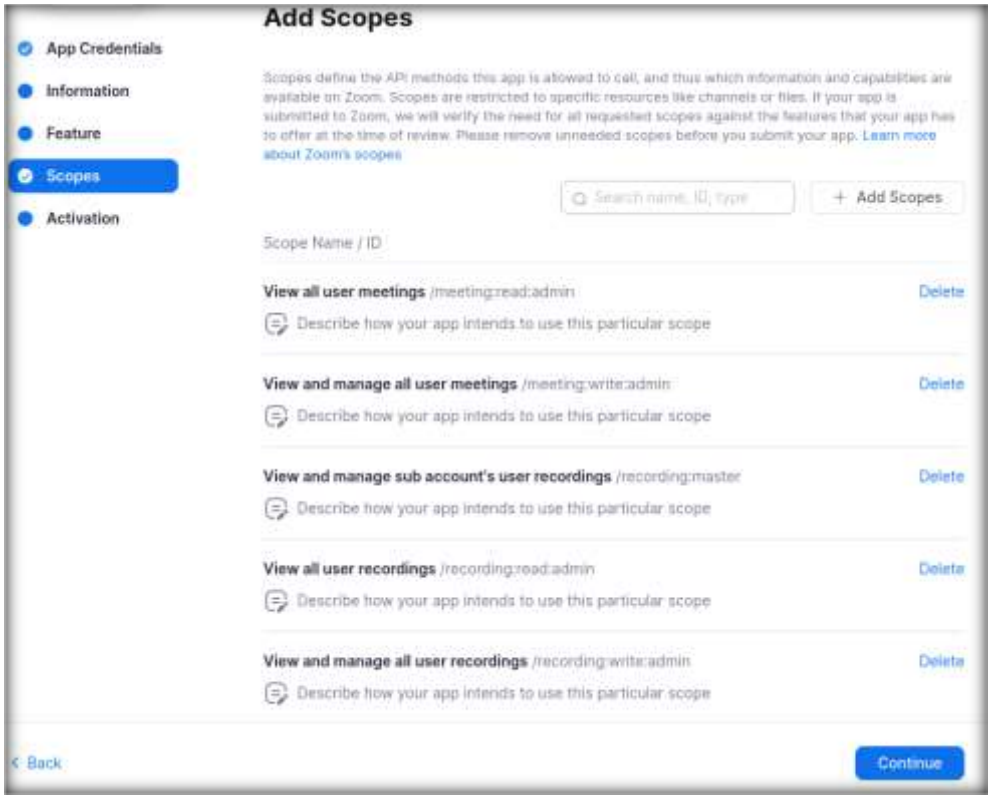


Рис. 3.6 - Області застосування (scopes) додатка.

Були обрані наступні області застосування (scopes):

- View all user meetings
- View and manage all user meetings
- View and manage sub account's user recordings
- View all user recordings
- View and manage all user recordings

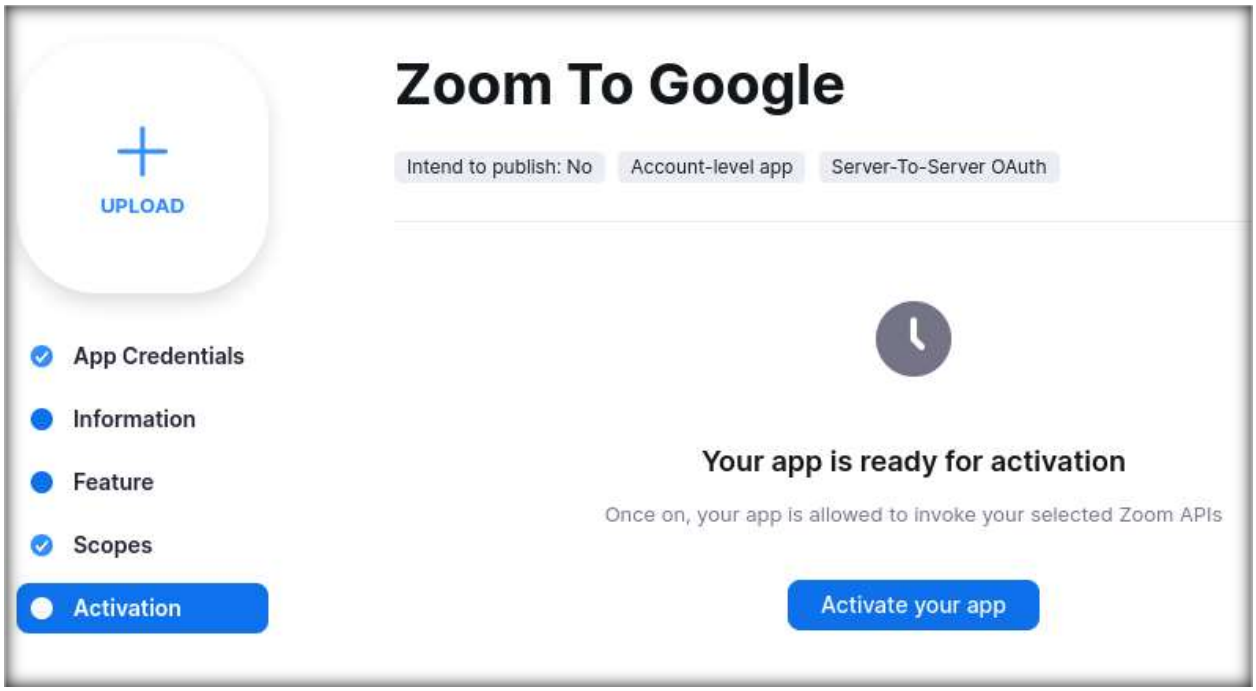


Рис. 3.7 - Додаток Zoom готовий до активації.

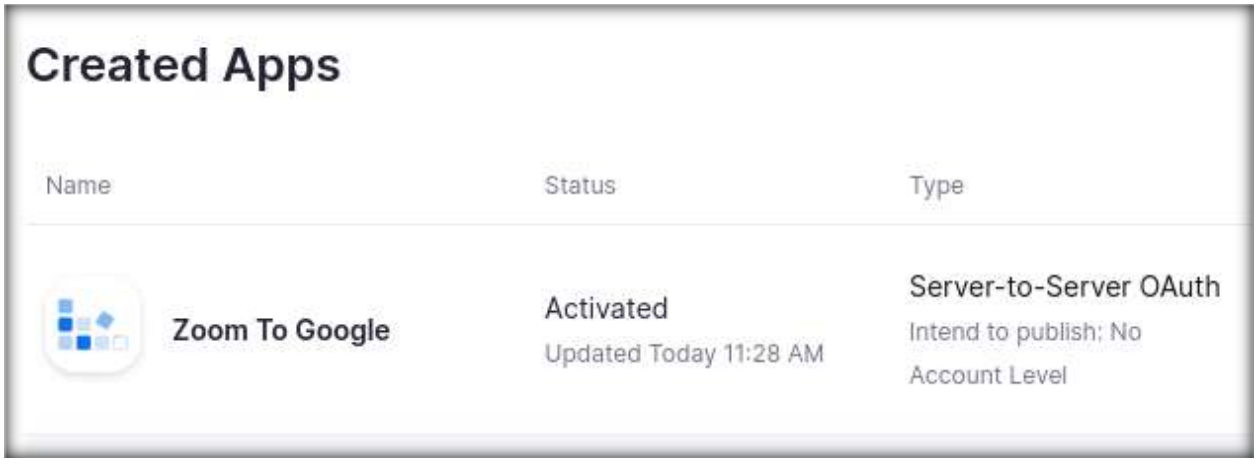


Рис. 3.8 - Створення додатку Zoom завершено.

3.2 Створення Google проекту типу Server-to-Server OAuth

На веб-сайті Google Cloud Platform створюємо новий проект.

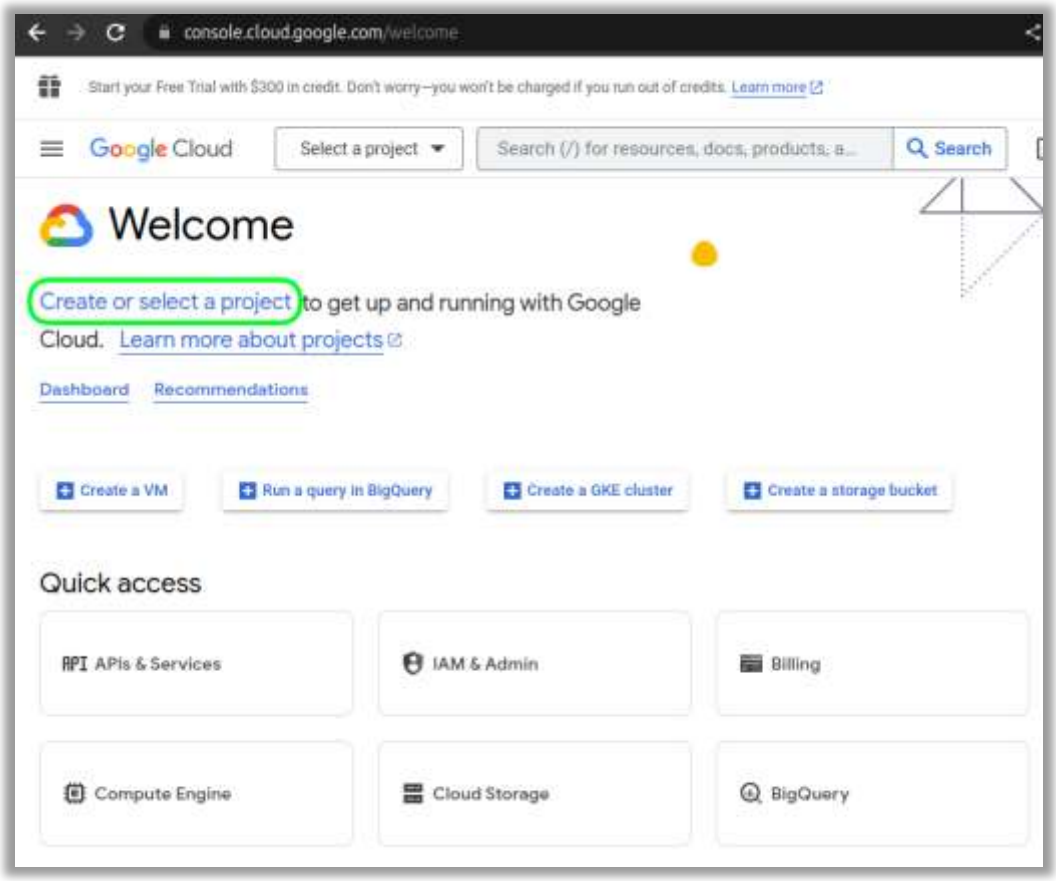


Рис. 3.9 - Початок створення нового проекту в Google Cloud Platform.

Рис. 3.10 - Створення нового проекту в Google Cloud Platform.

Вибіраємо нове ім’я проекту.

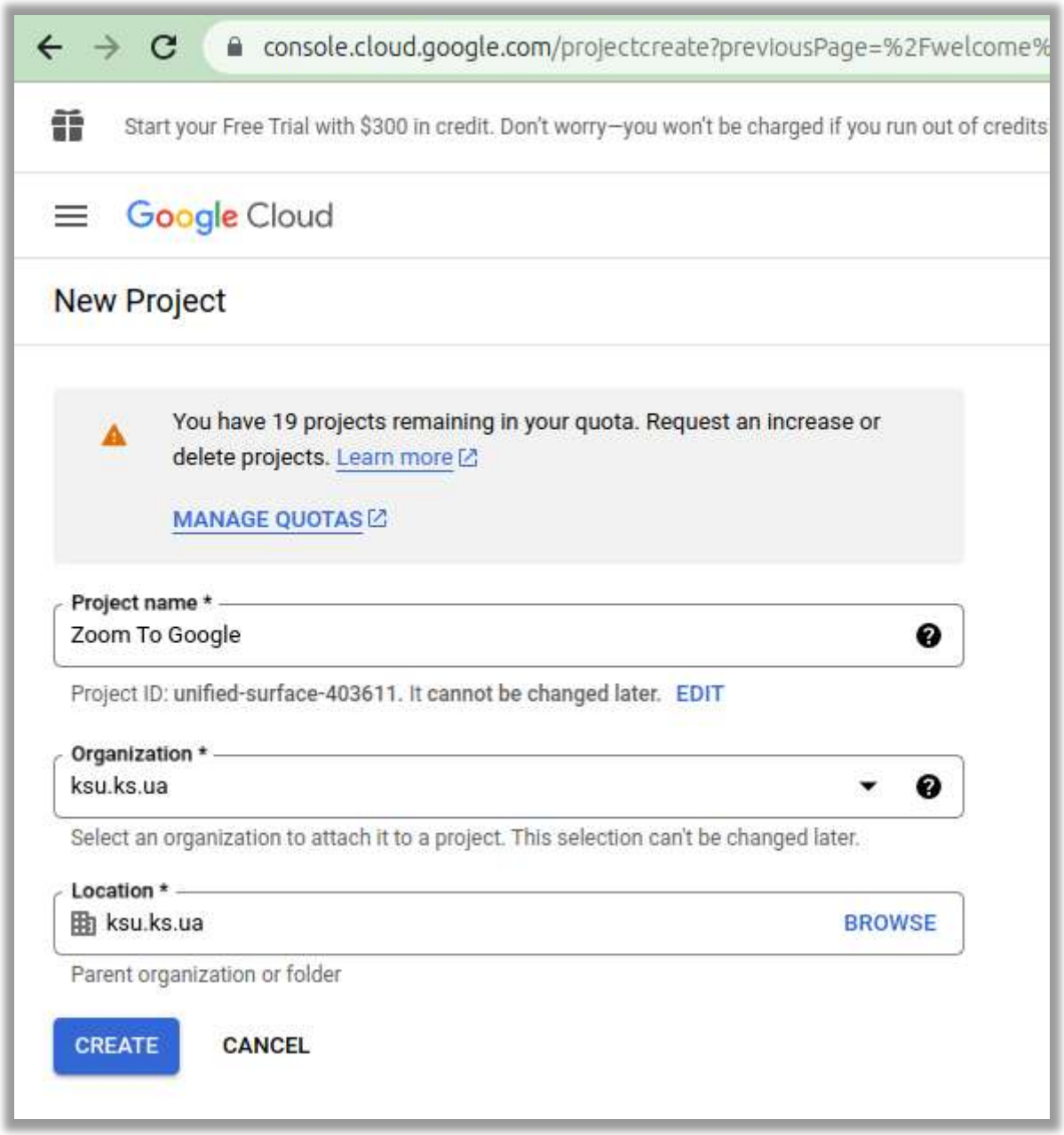


Рис. 3.11 - Задаємо ім’я додатку.

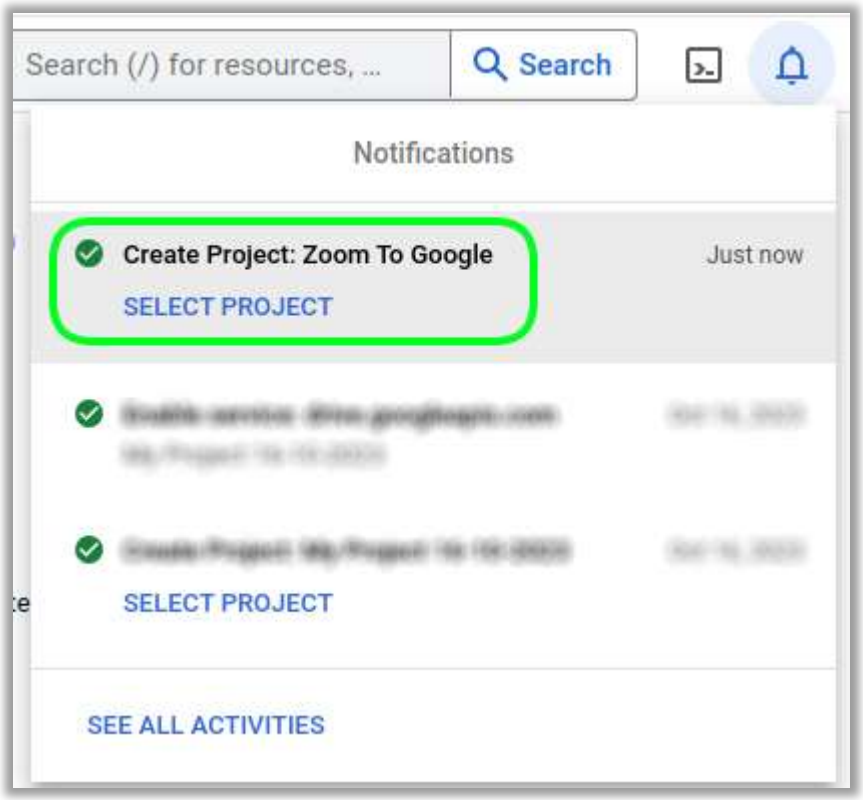


Рис. 3.12 - Отримане повідомлення про створення нового проекту.

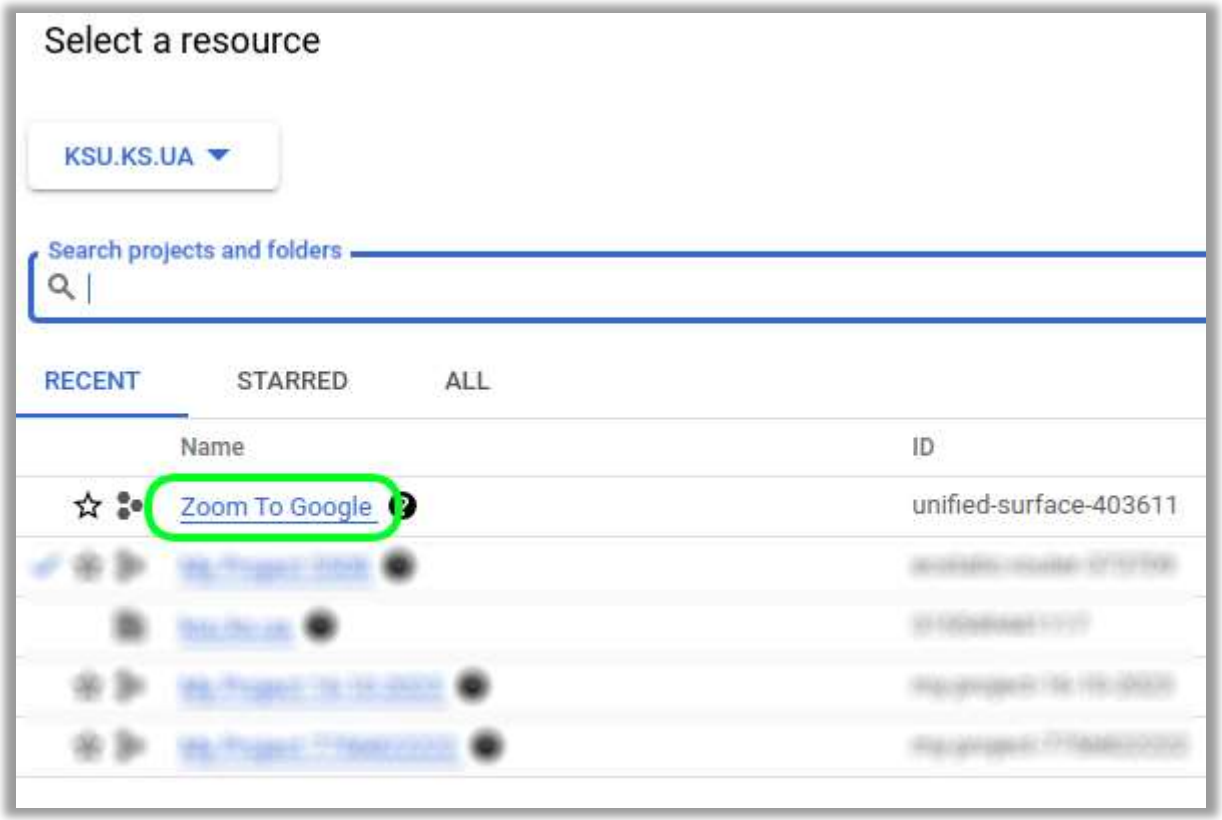


Рис. 3.13 - Обираємо новий проект для налаштування.

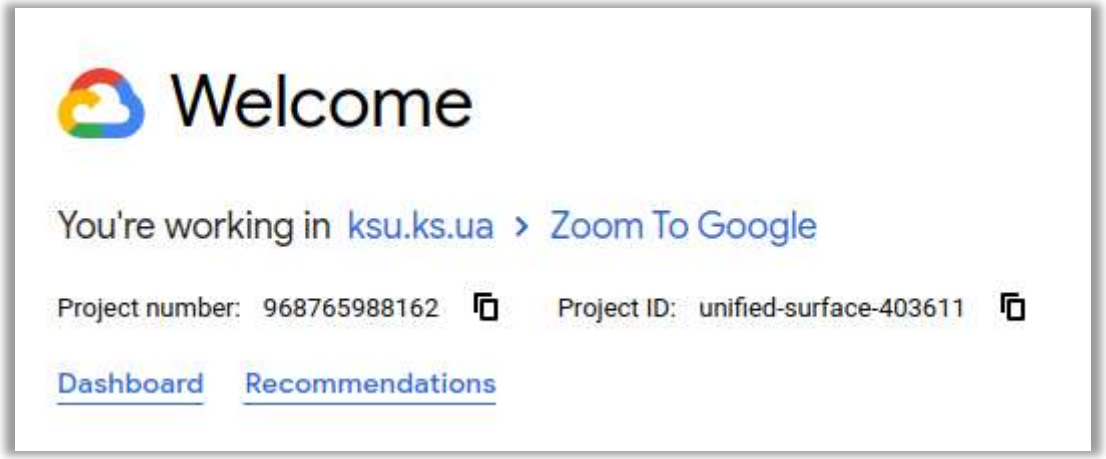


Рис. 3.14 - Проект обрано.

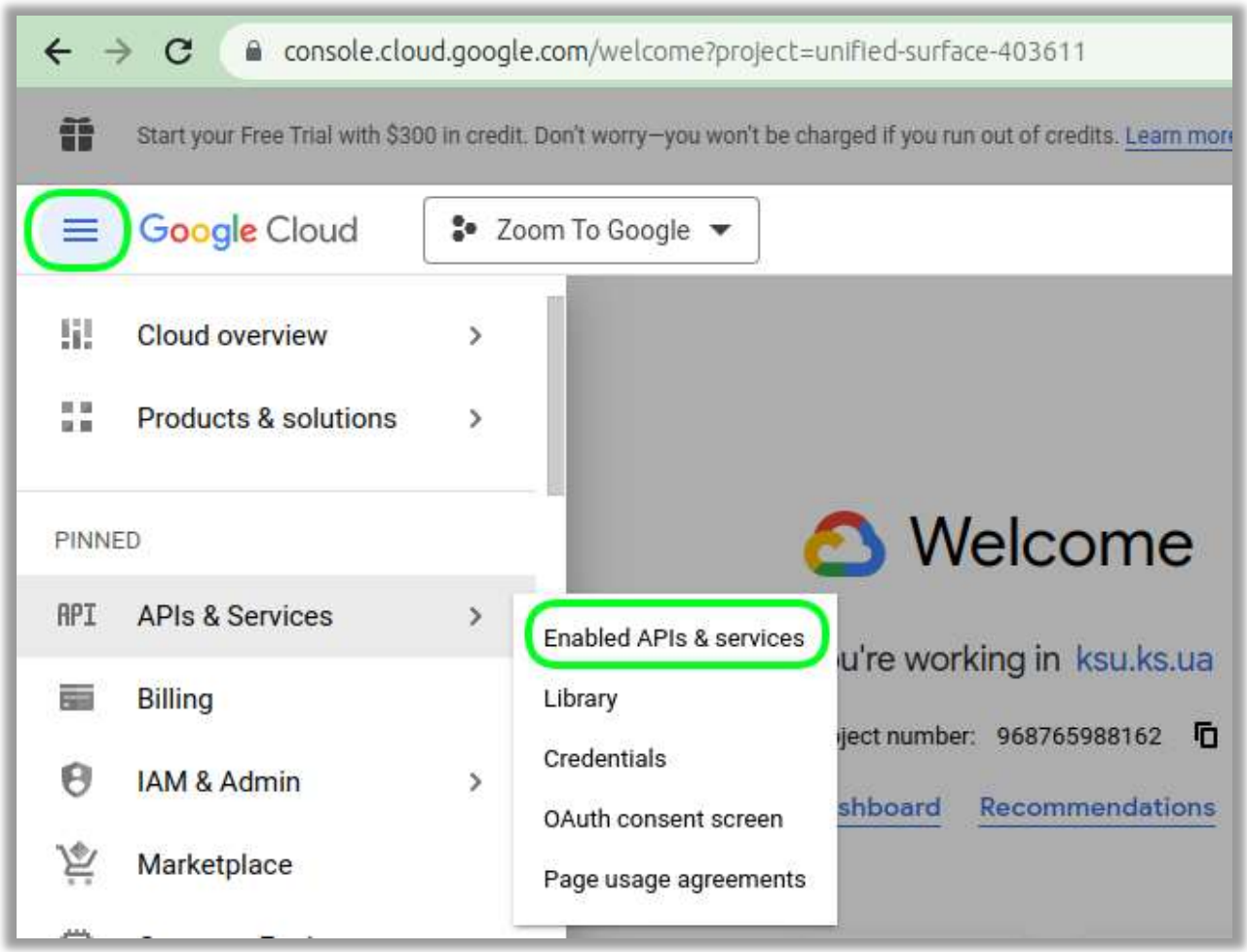


Рис. 3.15 - Заходимо в підменю APIs & Services.

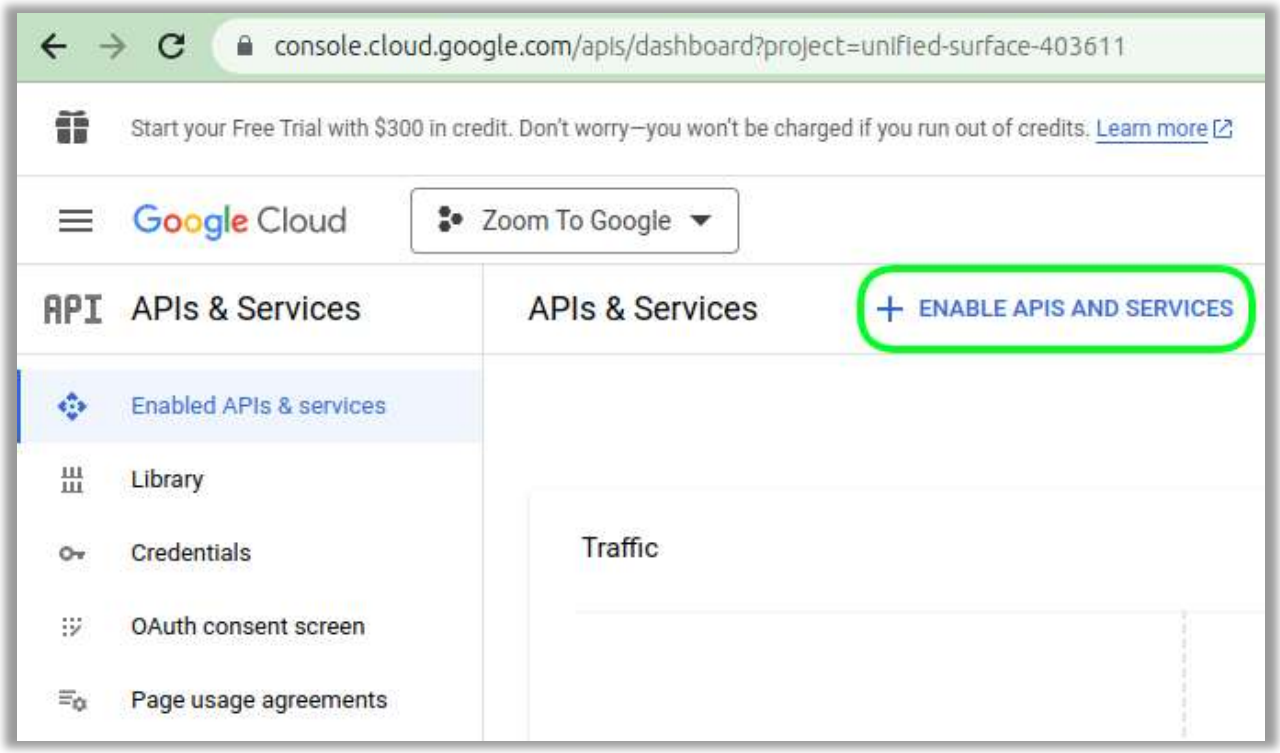


Рис. 3.16 - Додаємо необхідне API.

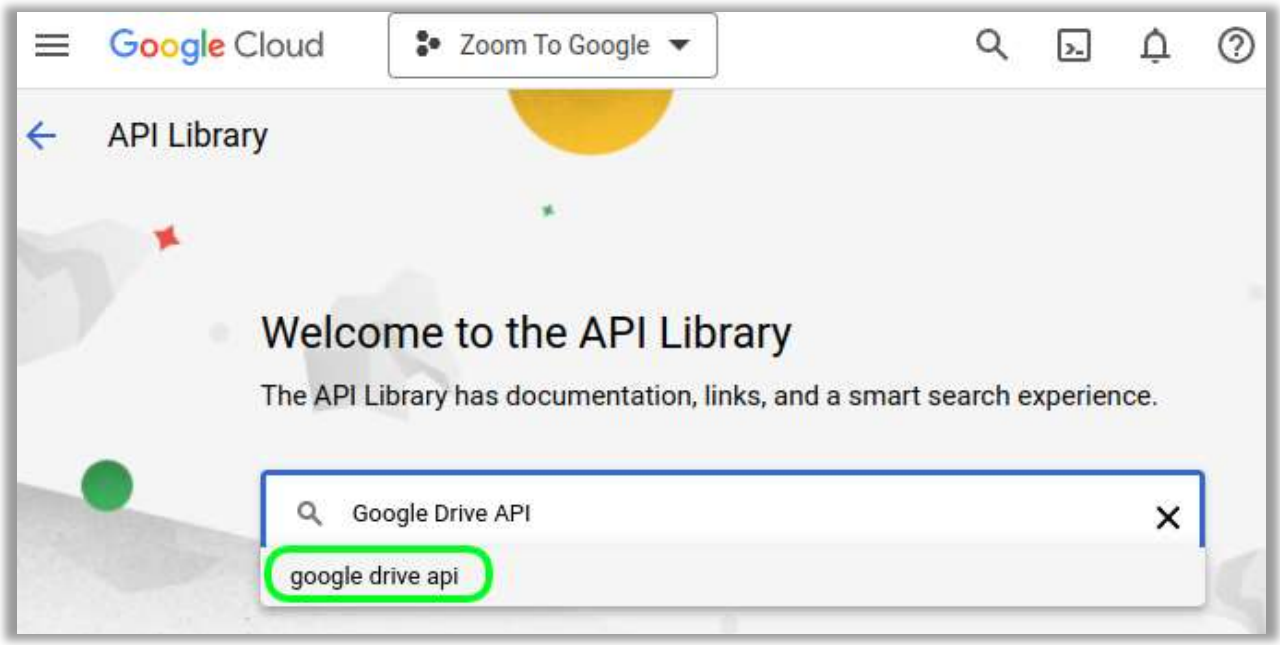


Рис. 3.17 - Шукаємо Google Drive API.

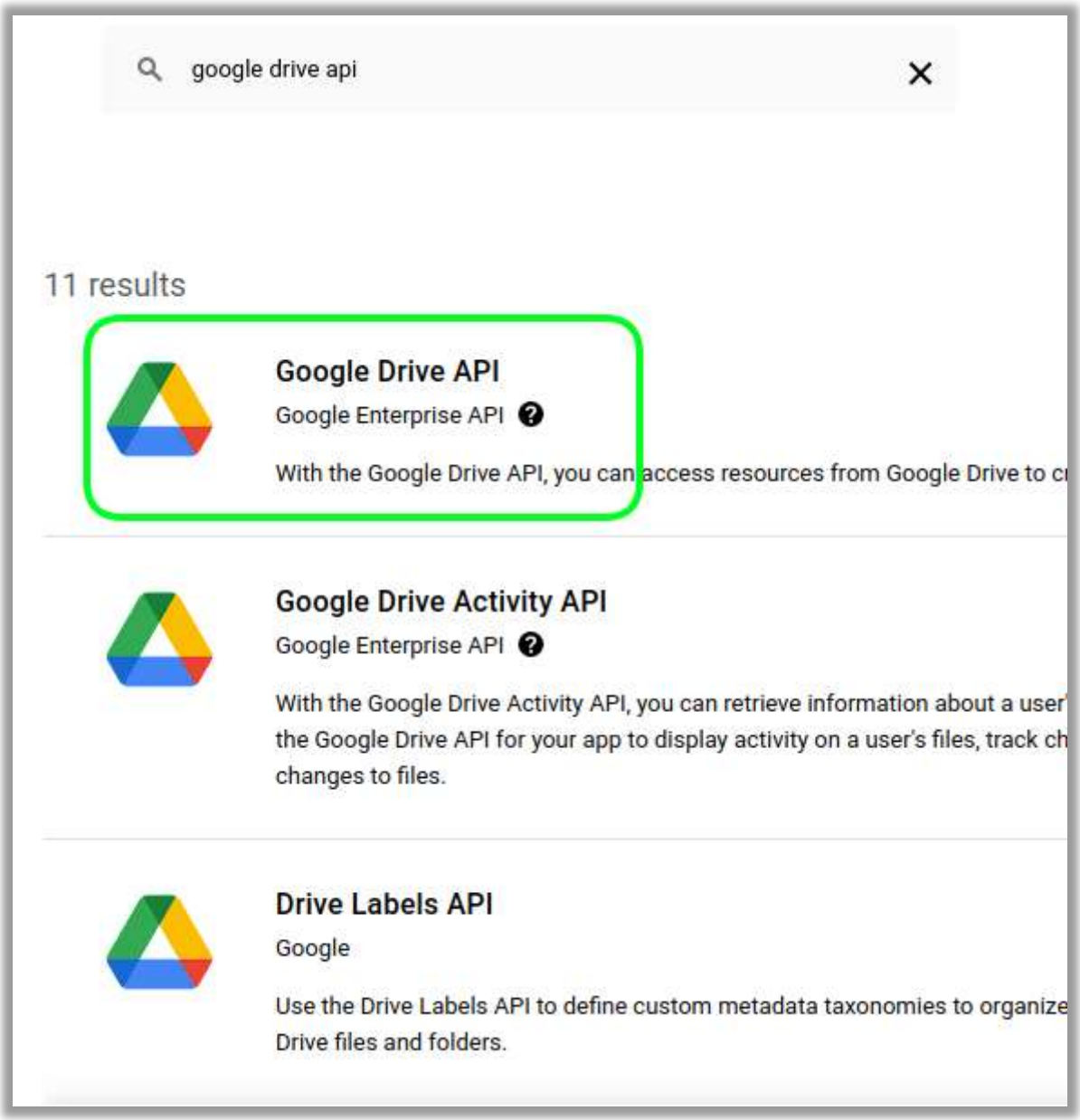


Рис. 3.18 - Вибираемо Google Drive API.

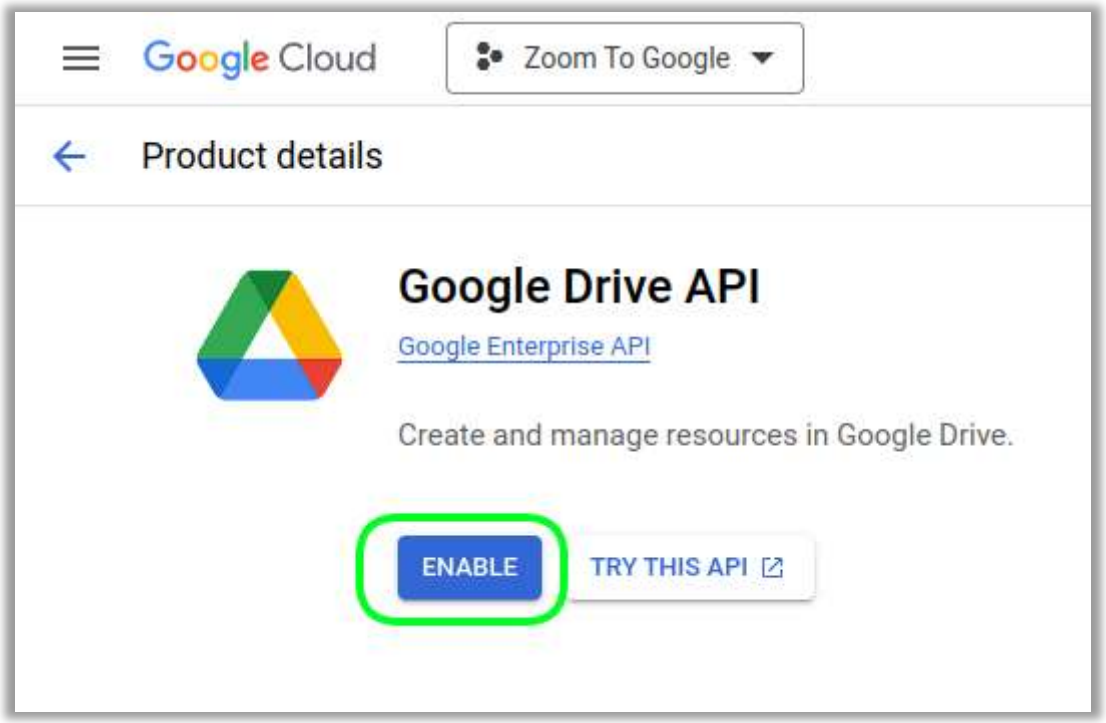


Рис. 3.19 - Підключаємо та активуємо Google Drive API.

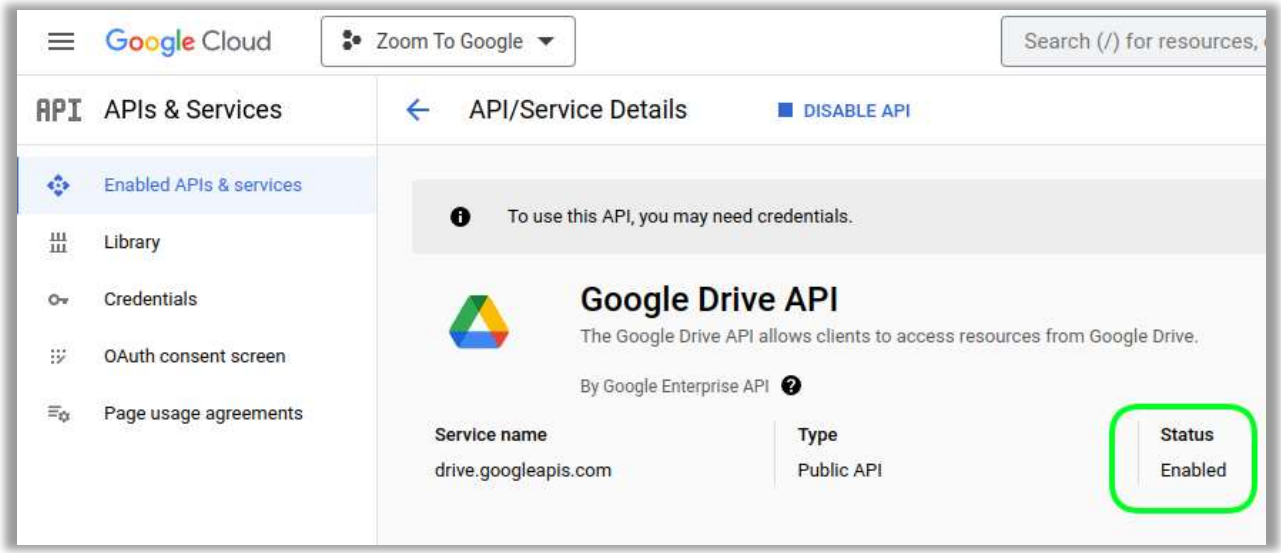


Рис. 3.20 - Google Drive API підключено.

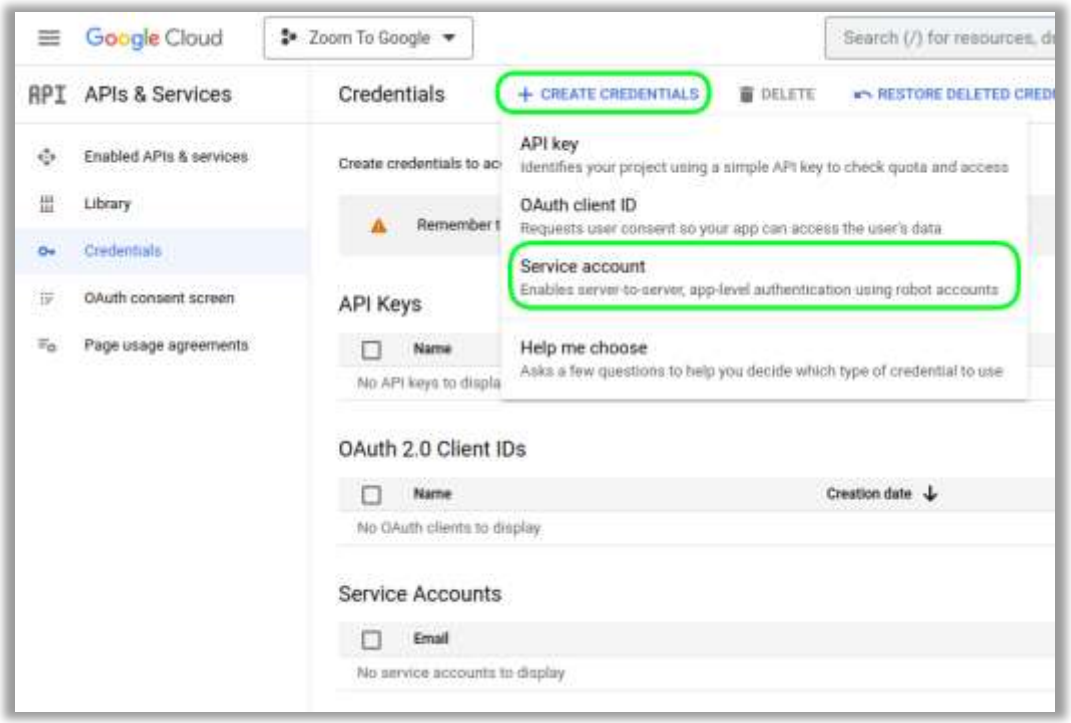


Рис. 3.21 - Початок створення обликових даних.

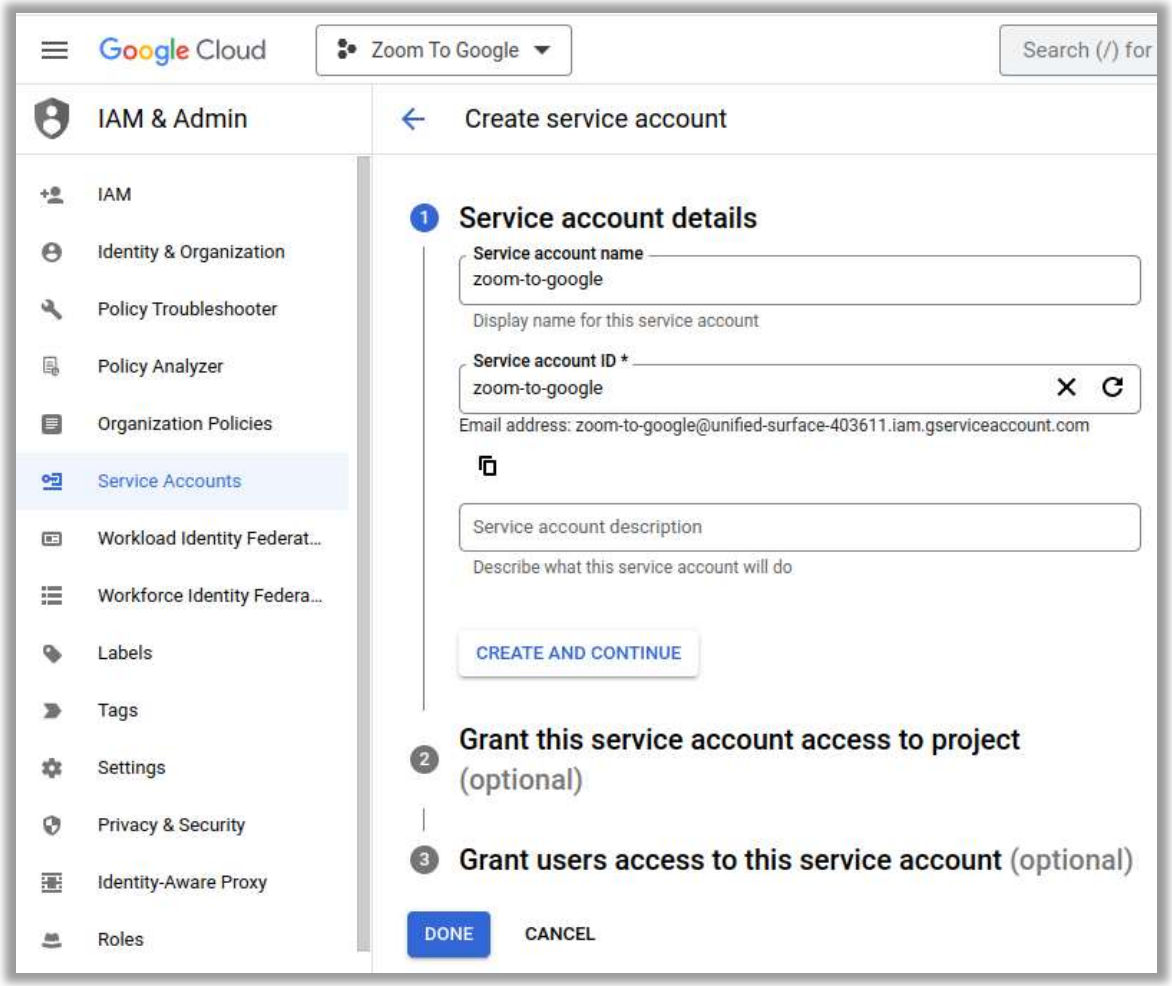


Рис. 3.22 - Додаємо необхідні дані.

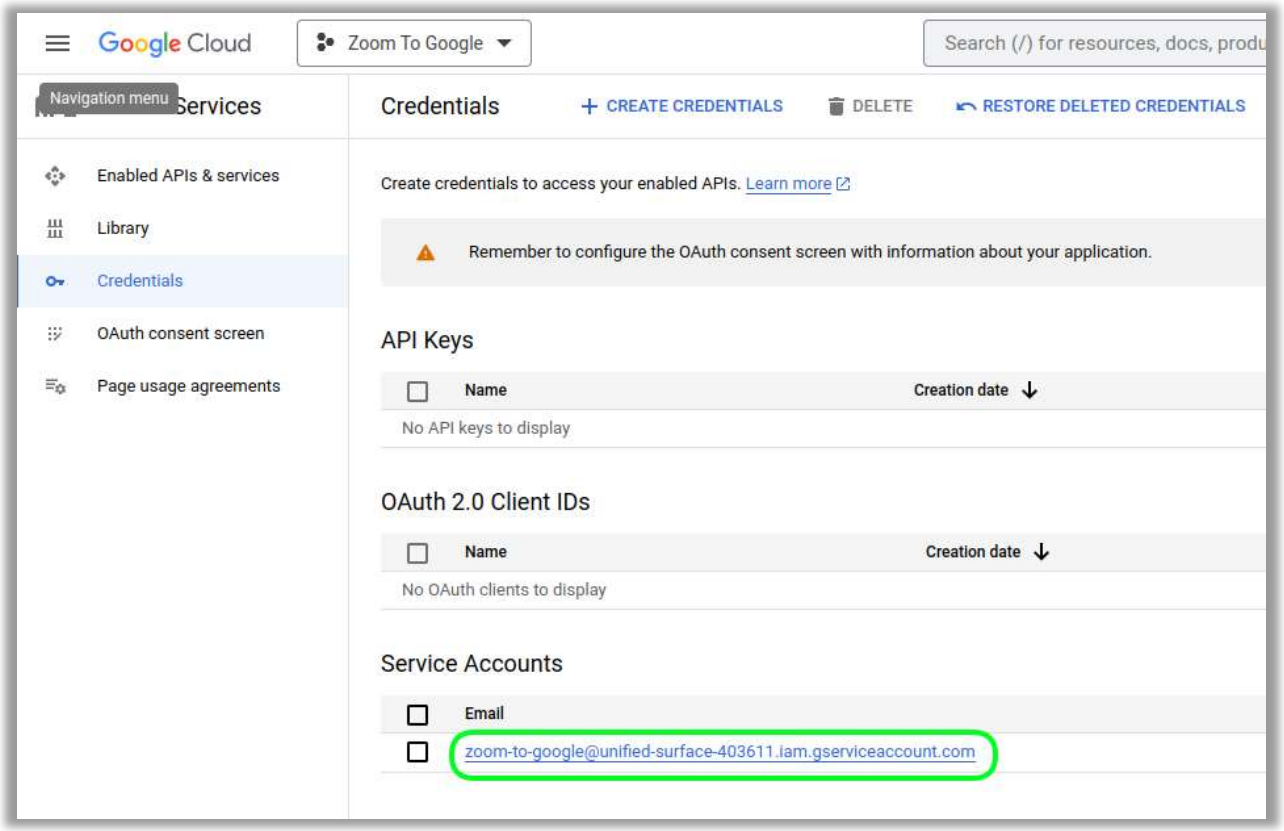


Рис. 3.23 - Отримано електронну пошту для додатка.

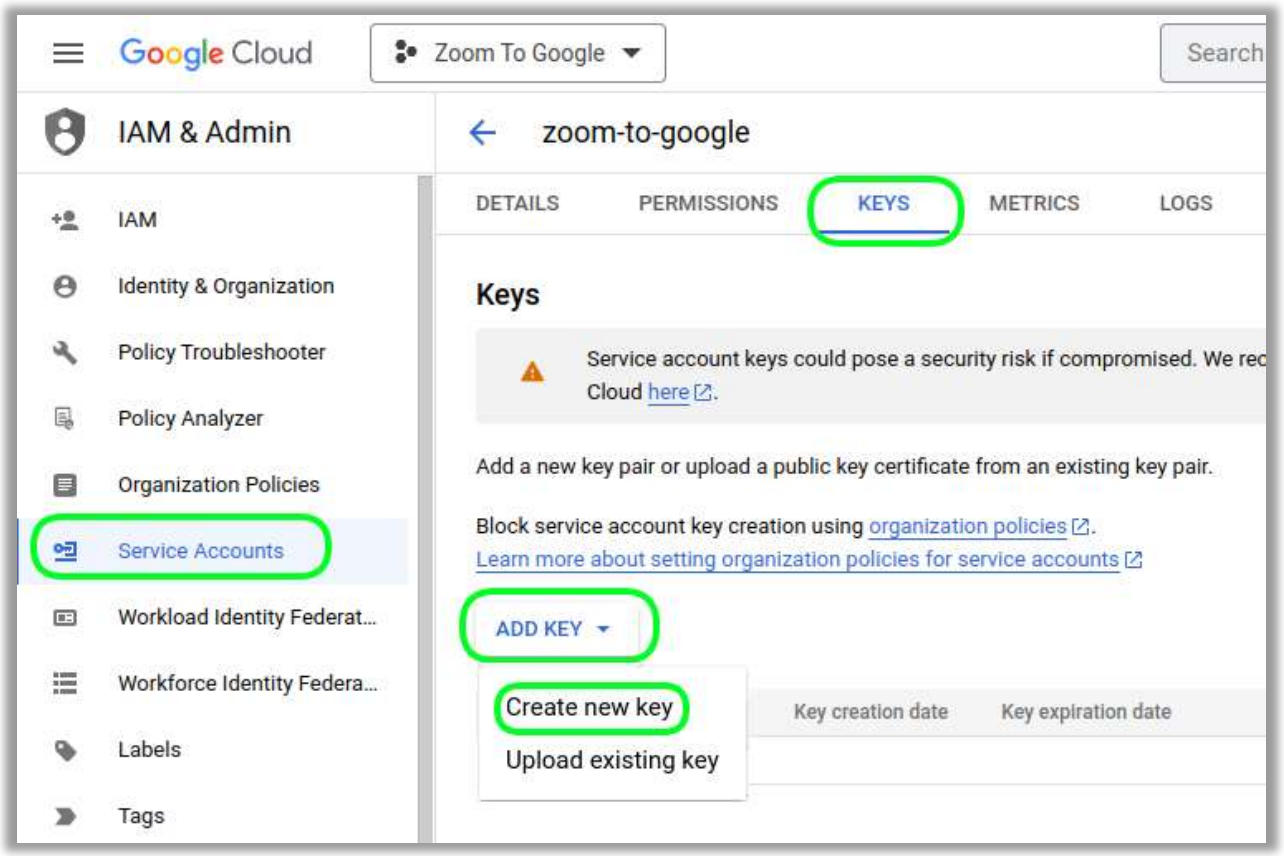


Рис. 3.24 - Додаємо нового ключа до сервісного облікового запису.

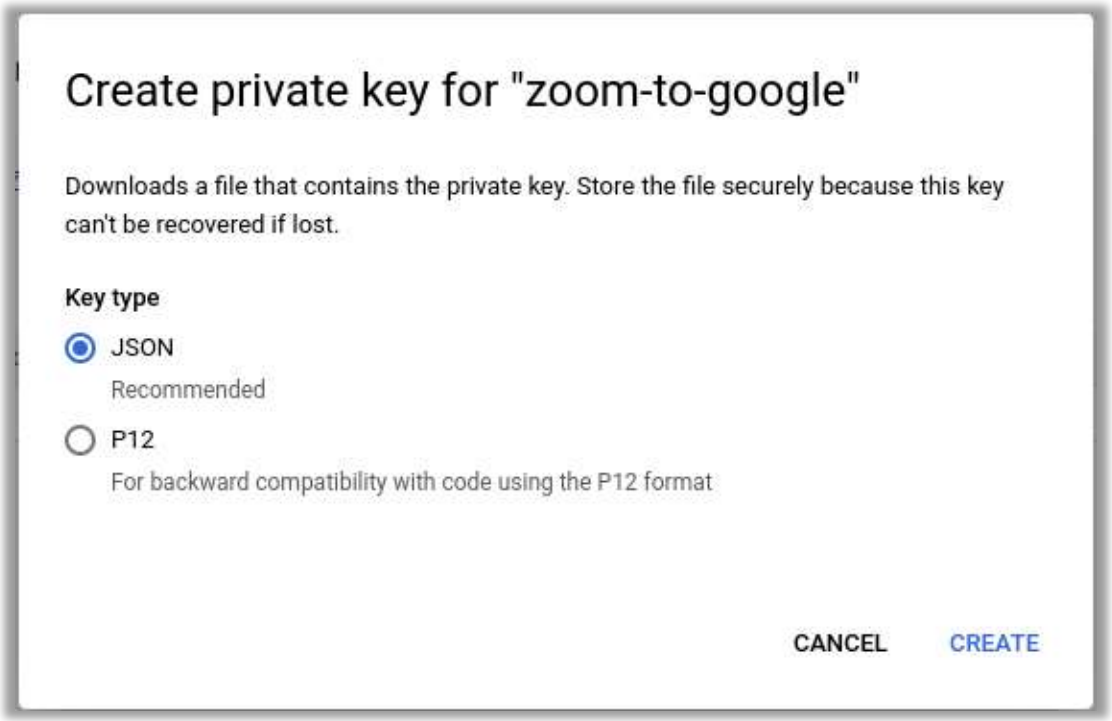


Рис. 3.25 - Вибираємо тип JSON.

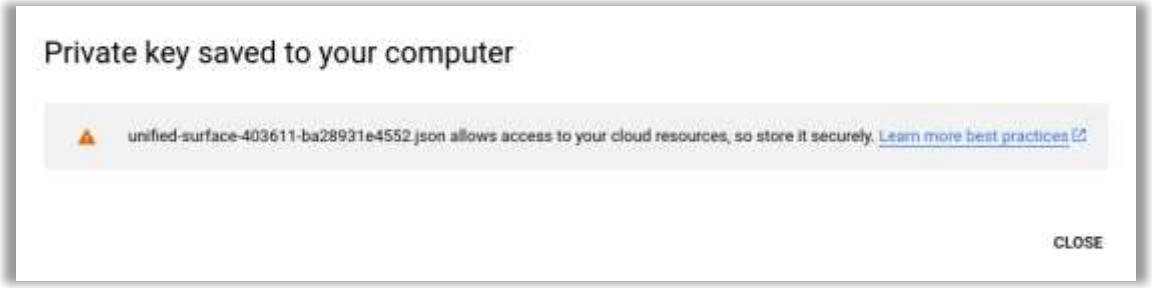


Рис. 3.26 - Нова пара ключей створенна.

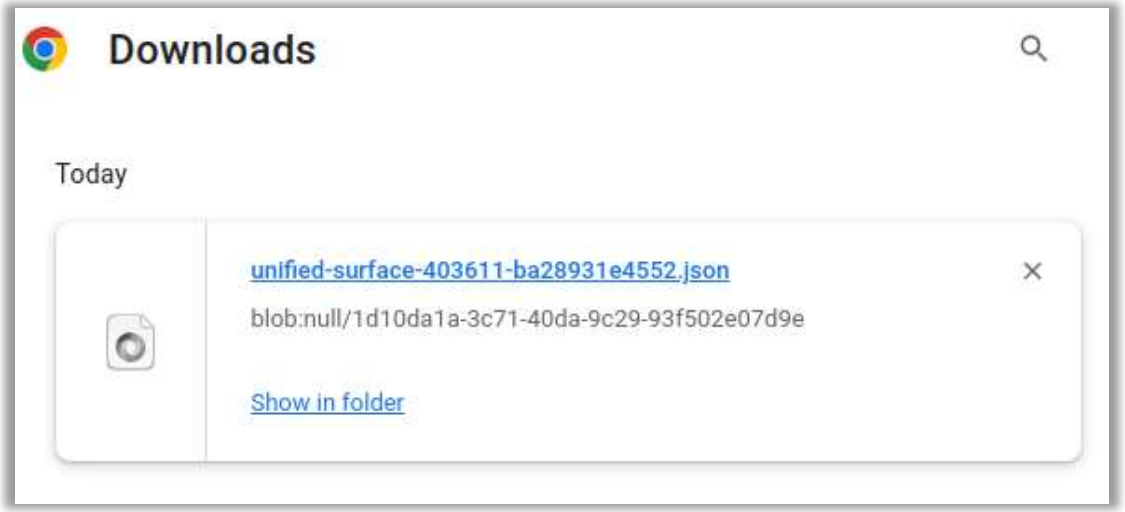


Рис. 3.27 - Завантажуємо нові ключи.

Рис. 3.28 - Вигляд файла з секретними ключами.

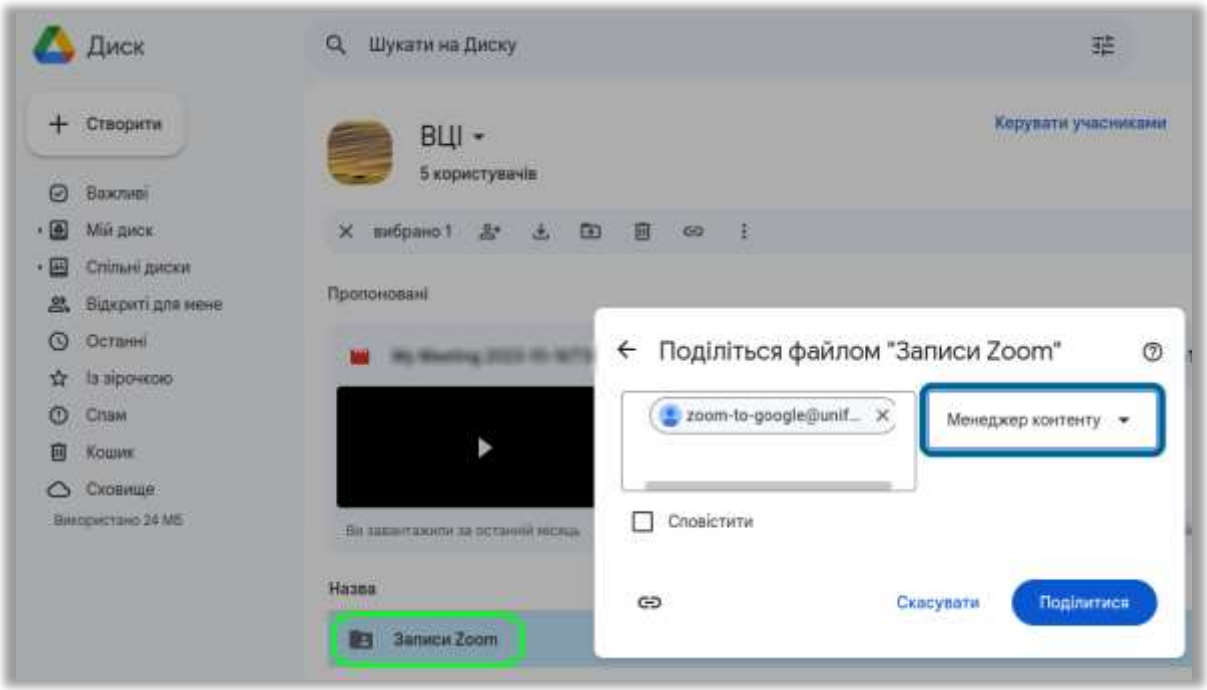


Рис. 3.29 - Додаємо необхідних прав сервісному обліковому запису.

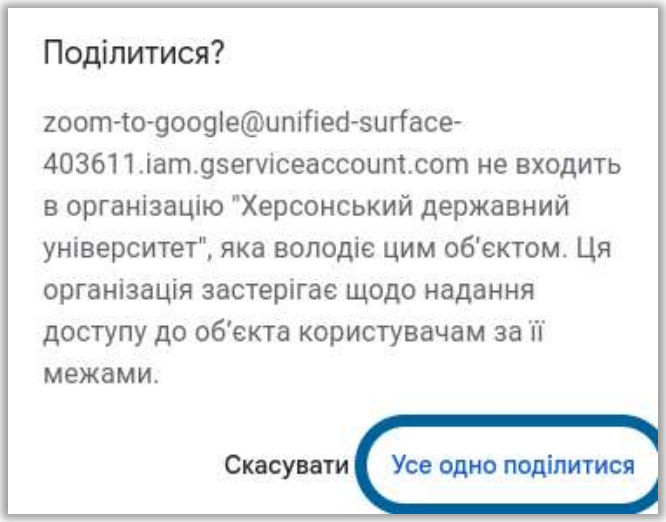


Рис. 3.30 - Підтвердження наданя прав.

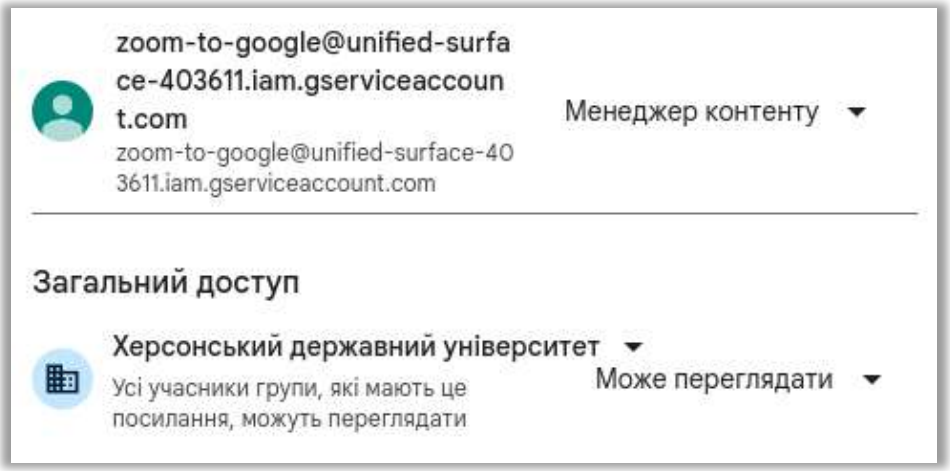


Рис. 3.31 - Сервісний обліковий запис має доступ з рівнем “Менеджер контенту”.

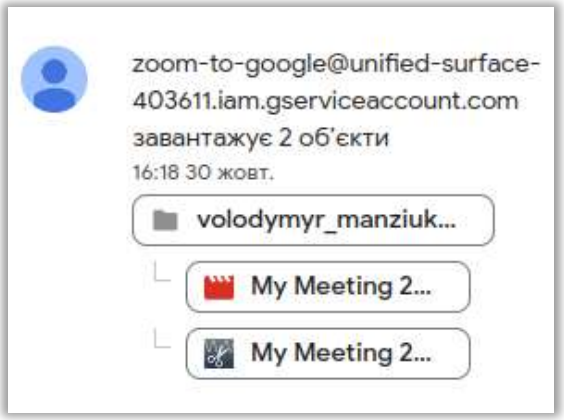


Рис. 3.32 - Зафіксовани дії сервісного облікового запису. Трансфер файлів успішно завершився.

3.3 Функції програми для роботи з Zoom API

3.3.1 Отримання токену доступу Zoom

В однієй функції впроваджено отримання нового токену, або його оновлення, якщо термін дії закінчився (становить 1 годину).

```
def update_token():
    global zoom_token
    global token_ttl
    if not zoom_token:
        logger.debug('Requesting token ...')
        client_id = config.client_id
        client_secret = config.client_secret
        account_id = config.account_id
        auth_base_str = client_id + ':' + client_secret
        url = "https://zoom.us/oauth/token?grant_type=account_credentials&account_id=" + account_id
        payload = {}
        headers = {
            'Authorization': 'Basic ' + base64.b64encode(auth_base_str.encode('utf8')).decode('utf8')
        }
        json_response = requests.request("POST", url, headers=headers, data=payload).json()
        logger.debug('Token retrieved...')
        logger.debug(json_response['access_token'])
        zoom_token = json_response['access_token']
    else:
        if datetime.datetime.now().timestamp() - token_ttl > 3500:
            token_ttl = datetime.datetime.now().timestamp()
            zoom_token = ''
            update_token()
        else:
            return zoom_token
    return zoom_token
```

3.3.2 Отримання списку всіх користувачей

Отримуємо список користувачей по 30 на кожну сторінку. Кожного разу перевіряємо параметр *next_page_token*, котрий вказує, що існує наступна сторінка з з іменами користувачей. Якщо параметр *next_page_token* відсутній, виконання виклику закінчується. Токен доступу та необхідні параметри HTTP виклику передаються через функцію *update_headers()*.

```
headers1 = update_headers()
while True:
    url1 = 'https://api.zoom.us/v2/users/?page_size=30'
    if next_p_token != '':
        url1 += '?&next_page_token=' + next_p_token
    response1 = requests.get(url1, headers=headers1)
    data1 = response1.json()
    for user in data1['users']:
        logger.info(user['email'])
        users_count += 1
        filelist = filelist + getAccountCloudRecordings(st, en, user['email'], query_params.action,
                                        headers1)
    next_p_token = data1["next_page_token"]
    if next_p_token == '':
        break
```

3.3.3 Отримання хмарних записів окремого користувача

```
def getAccountCloudRecordings(start_d, end_d, user, action, headers2):
    url = 'https://api.zoom.us/v2/users/{}/recordings?from={}&to={}&page_size=30'.
format(user, start_d, end_d)
    try:
        response = requests.get(url, headers=headers2)
        data = response.json()
        records = []  # list
        for meetings in data['meetings']:
            if ('recording_files' in meetings):
                for recs in meetings['recording_files']:
                    recoding_name = meetings['topic'] + ' ' + recs['recording_start'] + '.' + recs['file_type']
                    records.append(recoding_name)
            else:
                logger.debug("No recording_files for " + meetings['topic'] + ". Meetings uuid " + meetings['uuid'])
    except HttpError as error:
        logger.error(F'An error occurred: {error}')
    return records
```

3.3.4 Завантаження хмарних записів окремого користувача

```
downloadCloudRecording(recs['download_url'], recoding_name)
def downloadCloudRecording(download_link, file_name):
    headers = {
        "Authorization": "Bearer " + update_token()
    }
    logger.info(file_name + ' downloading...')
    response = requests.get(download_link, headers=headers)
    # Check if the response was successful (status code 200)
    if response.status_code == 200:
        # Save the recording to a file
        with open(PATH_TO_DOWNLOAD_DIR + file_name, "wb") as f:
            f.write(response.content)
        logger.info('{} downloaded successfully.'.format(file_name))
    else:
        logger.error('Failed to download {}. Error code:'.format(file_name), response.status_code)
        sys.exit(2)
```

3.3.4 Видаленя хмарних записів окремого користувача

```
def delAccountCloudRecording(meetingId, recordingId):
    logger.info("Moving to trash...")
    logger.debug("recordingId {}    meetingId {} ".format(recordingId, meetingId))
    headers = {
        'Authorization': 'Bearer {}'.format(update_token())
    }
    # If a UUID starts with "/" or contains "//" (example: "/ajXp112QmuoKj4854875=="),
    # you must double encode the UUID before making an API request.
    meetingId = urllib.parse.quote(meetingId, safe='')
    recordingId = urllib.parse.quote(recordingId, safe='')
    meetingId = urllib.parse.quote(meetingId, safe='')
    recordingId = urllib.parse.quote(recordingId, safe='')
    logger.debug("recordingId {}    meetingId {} ".format(recordingId, meetingId))
    url = 'https://api.zoom.us/v2/meetings/{}/recordings/{}?action=trash'.format(meetingId, recordingId)
    logger.debug(url)
    response = requests.delete(url, headers=headers)
    logger.debug(response)
    if response.status_code in (200, 204):
        # Recording deleted  Meeting recording file deleted
        logger.info('Moved to trash successfully.' + '\n')
    else:
        logger.error('Failed to Delete. Error code:', response.status_code)
    return
```

3.3 Функції програми для роботи з Google Drive API

Для кожної функції буде розглянуто два варіанти отримання результатів.

- використовуючи ідентифікатор клієнта.

Створюється ідентифікатор клієнта OAuth 2.0. Для цього необхідно пройти автентифікацію як кінцевий користувач і отримати доступ до даних користувача [40]. Користувач має доступ до корпоративного диску з необхідними правами.

```
from pydrive2.auth import GoogleAuth
from pydrive2.drive import GoogleDrive
gauth = GoogleAuth()
gauth.LocalWebserverAuth()
drive = GoogleDrive(gauth)
```

- використовуючи облікові дані сервісного облікового запису (service account).

Сервісний обліковий запис має необхідні права. Доступ до корпоративного диску було надано зазделегить адміністратором домену.

```
scope = ['https://www.googleapis.com/auth/drive']
service_account_json_key = '/home/user/z2g/unified-surface-403611-ba28931e4552.json'
credentials = service_account.Credentials.from_service_account_file(
    filename=service_account_json_key,
    scopes=scope)
service = build('drive', 'v3', credentials=credentials)
```

3.3.1 Перевірка існування або створення папки користувача на диску

- використовуючи ідентифікатор клієнта.

```
def gdrive_check_folder_OAuth(gdrive, user_folder_name):
    # check or create
    files = []
    page_token = None
    query_string = "q: '12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and " \
                   "mimeType = 'application/vnd.google-apps.folder'"
    response = gdrive.ListFile({'q': "'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and trashed=false and
mimeType = "
                                     "'application/vnd.google-apps.folder'"}).GetList()
    logger.info("Reading folders list in shared Drive ...")
    #logger.debug(response)
    for rec in response:
        # logger.debug('title: %s' % (rec['title']))
        if rec['title'] == user_folder_name:
            logger.info(user_folder_name + " folder exist. This is good!")
            return rec['id']
    logger.info(user_folder_name + " folder does not exist --> create it")
    newFolder = gdrive.CreateFile({'title': user_folder_name, "parents": [{"kind": "drive#fileLink", "id": \
      "12waDsJhsnsiPjQplgKaEbe9RBEiycj90"}], "mimeType": "application/vnd.google-apps.folder"})
    newFolder.Upload()
    logger.info('user_folder_created')
    return newFolder.get('id')
```

- використовуючи облікові дані сервісного облікового запису (service account).

```
def gdrive_check_folder_api_v3(gdrive_service, user_folder_name):
    page_token = None
    i = 1
    j = 1
    while True:
        response = gdrive_service.files().list(
            pageSize=1000,
            supportsAllDrives=True,
            includeItemsFromAllDrives=True,
            fields='nextPageToken, files(id,parents,name,mimeType)',
            q="mimeType='application/vnd.google-apps.folder' and trashed=false and
'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents",
            pageToken=page_token
        ).execute()
        print("Reading folders list in shared Drive ...")
        for rec in response.get('files', []):
            if rec['name'] == user_folder_name:
                print(user_folder_name + " folder exist. This is good!")
                return rec['id']
            j += 1
        page_token = response.get('nextPageToken', None)
        i += 1
        if page_token is None:
            break
    print(user_folder_name + " folder does not exist --> create it")
    folder_metadata = {
        'name': user_folder_name,
        'mimeType': 'application/vnd.google-apps.folder',
        'parents': ["12waDsJhsnsiPjQplgKaEbe9RBEiycj90"]
    }
```

```
    newFolder = gdrive_service.files().create(body=folder_metadata, fields='id', supportsAllDrives=True
                                ).execute()
    print(F'Folder ID: "{newFolder.get("id")}".')
    logger.info('Folder ID: ' + newFolder.get("id"))
    return newFolder.get('id')
```

➢

3.3.2 Вивантаження записів окремого користувача на диск

- використовуючи ідентифікатор клієнта.

```
def upload_to_gdrive_OAuth(user):
    gauth = GoogleAuth()
    gauth.LocalWebserverAuth()
    drive = GoogleDrive(gauth)  # Create GoogleDrive instance with authenticated GoogleAuth instance
    # oauth2client
    upl_folder_id = gdrive_check_folder_OAuth(drive, user)  # return google folder ID
    for file_to_upl in os.listdir(PATH_TO_DOWNLOAD_DIR):
        f = os.path.join(PATH_TO_DOWNLOAD_DIR, file_to_upl)
        if os.path.isfile(f):
            try:
                logger.info(str(file_to_upl) + '  uploading...')
                upl_file = drive.CreateFile({'title': file_to_upl, 'parents': [{'id': upl_folder_id}]})
                upl_file.SetContentFile(f)
                upl_file.Upload()
                logger.info(str(file_to_upl) + '  uploaded successfully.')
            except HttpError as error:
                logger.error(F'An error occurred: {error}')
            else:
                os.remove(f)
```

- використовуючи облікові дані сервісного облікового запису (service account).

```
def upload_to_gdrive_api_v3(user):
    # connection to the Drive API service_account_json_key **********************************
    scope = ['https://www.googleapis.com/auth/drive']
    service_account_json_key = '/home/user/OneDrive-KHDU/z2g/my-project-16-10-2023-60fe78a7b6ef.json'
    credentials = service_account.Credentials.from_service_account_file(
        filename=service_account_json_key,
        scopes=scope)
    service = build('drive', 'v3', credentials=credentials)
    # connection to the Drive API service_account_json_key **********************************
    upl_folder_id = gdrive_check_folder_api_v3(service, user)  # return google folder ID
    for file_to_upl in os.listdir(PATH_TO_DOWNLOAD_DIR):
        f = os.path.join(PATH_TO_DOWNLOAD_DIR, file_to_upl)
        if os.path.isfile(f):
            try:
                logger.info(str(file_to_upl) + '  uploading...')
                file_metadata = {
                    'name': file_to_upl,
                    'parents': [upl_folder_id]
                }
                media = MediaFileUpload(f, resumable=True)
                upl_file = service.files().create(body=file_metadata, media_body=media,supportsAllDrives=True,
                                          fields='id').execute()
                print(F'uploaded successfully File ID: "{upl_file.get("id")}".')
                logger.info('File ID: ' + upl_file.get("id"))
                logger.info(str(file_to_upl) + '  uploaded successfully.')
            except HttpError as error:
                logger.error(F'An error occurred: {error}')
            else:
                os.remove(f)
```

3.4 Логування подій

Функція logger.debug() у Python використовується для реєстрації повідомлень про налагодження. Повідомлення про налагодження зазвичай використовуються для надання детальної інформації про внутрішню роботу програми.

Щоб використовувати функцію logger.debug(), спочатку потрібно створити об’єкт logger. Ми робимо це за допомогою модуля журналювання. Після того, як створено об’єкт реєстратора, використовується функція logger.debug() для реєстрації повідомлень про налагодження [41].

Повідомлення про налагодження буде зареєстровано на консолі або у файлі, залежно від того, як налаштовано реєстратор.

```
def main():
    params = parse_args()
    logger.debug(params)
    logger.setLevel(params.log_level)
    logger.info("User --> " + params.user)
    logger.info("Action --> " + params.action)
    logger.info("The start date --> " + params.start)
    logger.info("The end date --> " + params.end)
    logger.info("Skip users --> " + str(params.skip_emails))
    logger.info("Log level --> " + params.log_level)
    logger.info("********************************")
    if params.action == 'download':
        confirm("All the parameters are correct?")
    files = setZoomApiRequest(params)
```

3.5 Опис інтерфейсу

Інтерфейс командної строки (CLI) - це тип інтерфейсу користувача, який використовує текст для взаємодії з користувачем. CLI не має графічного інтерфейсу, а використовує лише текстовий режим.

Рис. 3.33 - Інтерфейс додатку.

argparse — це модуль Python, який дозволяє легко писати зручні інтерфейси командного рядка. Він забезпечує простий і послідовний спосіб визначення та аналізу параметрів командного рядка.

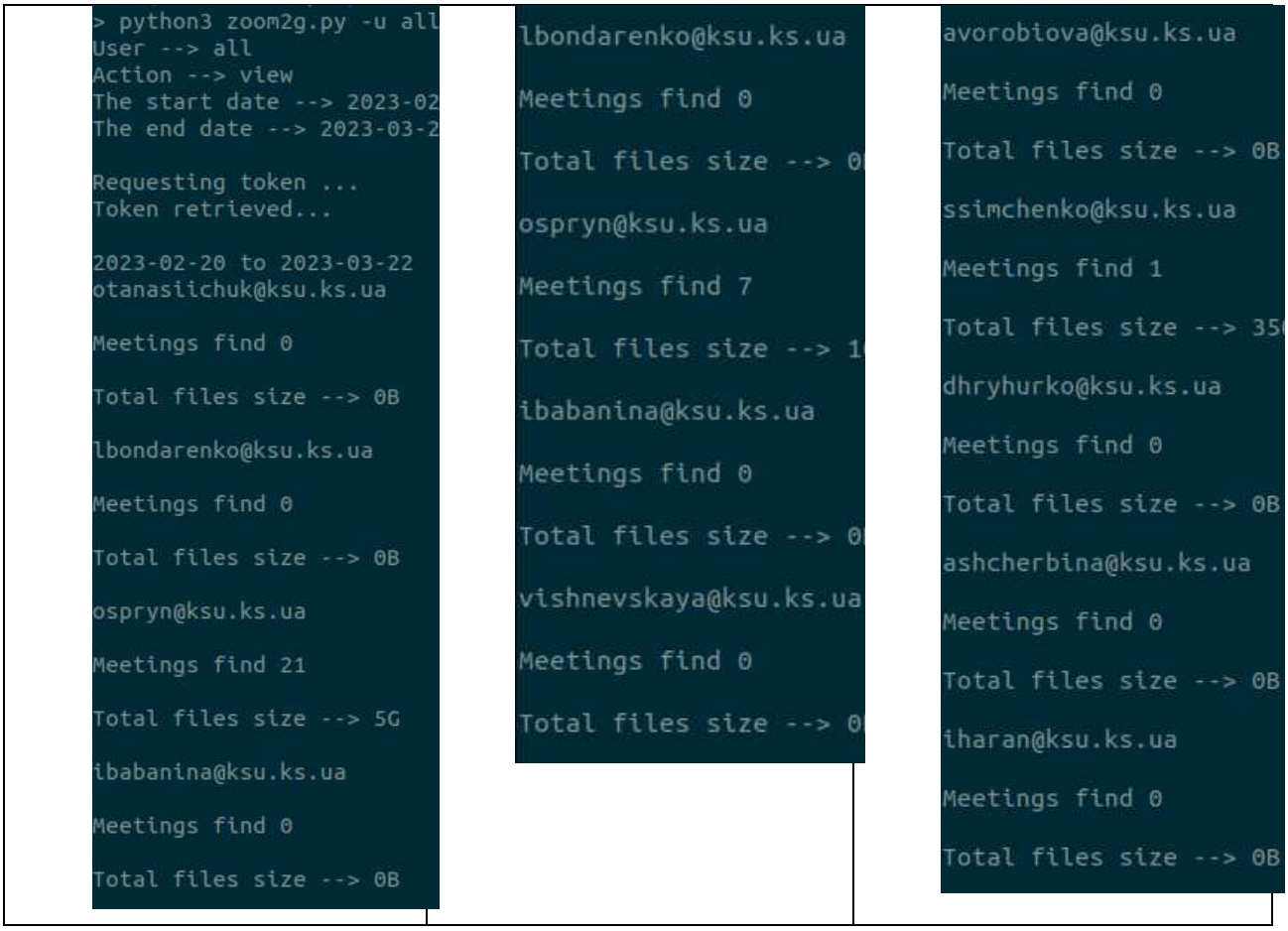
Щоб використовувати argparse, потрібно створити об’єкт ArgumentParser. Цей об’єкт буде містити всю інформацію, необхідну для аналізу командного рядка на типи даних Python.

Після створення об’єкту ArgumentParser, можна почати додавати до нього аргументи. Це можна зробити за допомогою методу add_argument(). Метод add_argument() приймає ряд параметрів, включаючи назву аргументу, тип аргументу та те, чи є аргумент обов’язковим.[42]

3.6 Результати роботи додатку.

Опція **--help**

Опція **-u all** виводить статистику по всім користувачам Zoom, що належать до university.kherson.ua.



Запуск без параметрів - приймаються параметри за замовчуванням.

Користувач - volodymyr.manziuk@university.kherson.ua, режим перегляду записів за останні 30 днів.

```
> python3 zoom2g.py
User --> volodymyr.manziuk@university.kherson.ua
Action --> view
The start date --> 2023-02-20
The end date --> 2023-03-22

Requesting token ...
Token retrieved...

2023-02-20 to 2023-03-22

Meetings find 3

Total files size --> 122M

2023-03-20 to 2023-03-22

Meetings find 0

Total files size --> 0B

Володимир Володимирович Манзюк's Zoom Meeting 2023-03-03T14:14:13Z.MP4
Володимир Володимирович Манзюк's Zoom Meeting 2023-03-03T14:14:13Z.M4A
Володимир Володимирович Манзюк's Zoom Meeting 2023-03-02T14:13:04Z.M4A
Володимир Володимирович Манзюк's Zoom Meeting 2023-03-02T14:13:04Z.MP4
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.M4A
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.MP4
```

Запуск скрипта з параметрами.

python3 zoom2g.py -u volodymyr.manziuk@university.kherson.ua -a download -s 2023-02-28 -e 2023-03-01

Користувач - volodymyr.manziuk@university.kherson.ua,з 28 лютого 2023р по 1 березня 2023р, завантажити.

```
> python3 zoom2g.py -u volodymyr.manziuk@university.kherson.ua -a download -s 2023-02-28 -e 2023-03-01
User --> volodymyr.manziuk@university.kherson.ua
Action --> download
The start date --> 2023-02-28
The end date --> 2023-03-01
All the parameters are correct? Are you sure? [y/n]: y

Requesting token ...
Token retrieved...

2023-02-28 to 2023-03-01

Meetings find 1

Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.M4A downloading...
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.M4A downloaded successfully.
Moving to trash...
Delete successfully.
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.MP4 downloading...
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.MP4 downloaded successfully.
Moving to trash...
Delete successfully.
Total files size --> 1M

Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.M4A
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.MP4
title: volodymyr_manziuk_university_kherson_ua
user_folder_exist
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.MP4  uploading...
Володимир Володимирович Манзюк's Personal Meeting Room 2023-02-28T14:50:49Z.M4A  uploading...
End of script
```

ВИСНОВКИ

Записи Zoom Cloud є цінним активом, але їх збереження в хмарі Zoom коштує набагато більш ніж зберегання на корпоративному Google диску. Додаток трансферу записів з хмарного середовища Zoom в корпоративний Google диск вирішує цю проблему, автоматично переносячи всі записи з Zoom Cloud у Google Drive. Програма проста у використанні та може бути налаштована для передачі даних за розкладом.

Розробка архітектури додатку для автоматичного трансферу даних з хмарного середовища Zoom в корпоративний Google Диск включає інтеграцію двох основних компонентів: Google Drive API та Zoom API.

Для авторизації користувача в Zoom використовується відкритий стандарт авторизації OAuth 2.0. Для цього потрібно створити додаток типу Server-to-Server OAuth. Після отримання токена доступу клієнтський компонент може виконувати операції з Zoom API.

Для завантаження записів Zoom у Google Drive клієнтський компонент використовує Google Drive API. Для цього потрібно створити обліковий запис для доступу до Google Drive API. Після отримання облікового запису клієнтський компонент може виконувати операції з Google Drive API, зокрема створювати файли та папки.

Загалом, проект "Автоматичного трансферу даних з хмарного середовища Zoom в корпоративний Google Диск" є успішним і може бути використаний для автоматизації процесу зберігання записів Zoom в корпоративному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zoom Plans & Pricing.
   URL: https://zoom.us/pricing
2. Introduction to Google Drive API.
   URL: https://developers.google.com/drive/api/guides/about-sdk
   URL: https://developers.google.com/static/drive/images/drive-intro.png
3. Google Drive API scopes.
   URL: https://developers.google.com/drive/api/guides/api-specific-auth
4. Google Drive API v3 Reference.
   URL: https://developers.google.com/drive/api/v3/reference
5. Creating a service account.
   URL:https://developers.google.com/identity/protocols/oauth2/service-account#python
6. Using OAuth 2.0 to Access Google APIs
   URL: https://developers.google.com/identity/protocols/oauth2
7. Zapier tasks
   URL: https://help.zapier.com/hc/en-us/articles/8496196837261
   URL: https://help.zapier.com/hc/en-us/articles/8496181725453-Learn-key-concepts-in-Zapier#task-0-7
8. Capture - Transfer to Google Drive and Dropbox By KISS.
   URL: https://marketplace.zoom.us/apps/4A2ZPPZ3TY-Nup6ODGg88w
9. Google Drive for Zoom by Splain.
   URL: https://marketplace.zoom.us/apps/5jjvtAZIQy-m-XkbchJMHg
10. Transferring-videos.com By Islem BOUSSETTA.
    URL: https://transferring-videos.com/en/pricing
    URL: https://marketplace.zoom.us/apps/aj1sYMG7SEWaNgQg4TQujQ
11. zBackup.app.
    URL: https://zbackup.app/
12. Zapier zoom to google.

URL: https://zapier.com/blog/upload-zoom-recordings-google-drive/

13. понад 350 викладачів.
URL: https://www.kspu.edu/PublisherReader.aspx?newsId=15762&lang=uk
14. Застаріли результати google за мінувший рік - меньш десяти
URL: https://www.google.com/search?q=zoom+records+to+gdrive+site:github.com+-issue&sxsrf=AJOqlzWW-zdvsKzOr7CeibIXFE333JjJJw:1679476806410&source=lnt&tbs=qdr:y&sa=X&ved=2ahUKEwi1ltHTmu_9AhVsi8MKHapyBzwQpwV6BAgBEB4&biw=1920&bih=922&dpr=1
15. Google Drive API Reference.
URL: https://developers.google.com/drive/api/v3/reference
16. Zoom API Cloud Recording operations.
URL: https://marketplace.zoom.us/docs/api-reference/zoom-api/ma/#tag/Cloud-Recording
17. Introduction to Zoom APIs
URL: https://developers.zoom.us/docs/api/
18. Zoom API Using OAuth 2.0
URL: https://developers.zoom.us/docs/api/rest/using-zoom-apis/#using-oauth-20
19. Zoom API Server-to-Server authentication
URL: https://developers.zoom.us/docs/api/rest/using-zoom-apis/#server-to-server-authentication
20. Zoom API Using JWT apps.
URL: https://developers.zoom.us/docs/api/rest/using-zoom-apis/#using-jwt-apps
21. JWT App Type.
URL: https://developers.zoom.us/docs/internal-apps/jwt-faq/
22. Server-to-Server OAuth.
URL: https://developers.zoom.us/docs/internal-apps/s2s-oauth/
23. Create a Server-to-Server OAuth app.

URL: https://developers.zoom.us/docs/internal-apps/create/

24. Request access token.
URL: https://developers.zoom.us/docs/integrations/oauth/#step-2-request-access-token
25. Операції з хмарними записами Zoom.
URL: https://developers.zoom.us/docs/api/rest/reference/zoom-api/ma/#tag/Cloud-Recording
26. Операції з хмарними записами Zoom. List users.
URL: https://developers.zoom.us/docs/api/rest/reference/zoom-api/methods/#operation/users
27. Операції з хмарними записами Zoom. List all recordings.
URL: https://developers.zoom.us/docs/api/rest/reference/zoom-api/methods/#operation/recordingsList
28. Операції з хмарними записами Zoom. Delete a meeting recording file.
URL: https://developers.zoom.us/docs/api/rest/reference/zoom-api/methods/#operation/recordingDeleteOne
29. OAuth2 and OpenID Connect: The Professional Guide. Vittorio Bertocci
30. Scenario where the access token fits.
URL: https://auth0.com/blog/id-token-access-token-what-is-the-difference/ https://images.ctfassets.net/23aumh6u8s0i/6bFrgMoBLfHk65ZWvenpuY/e6ee7bc645dec122d6206ba7440d41d7/access-token-scenario.png
31. How to Upload Files to Google Drive with a Service Account.
URL: https://www.labnol.org/google-api-service-account-220404
32. Using Google Drive API with Python and a service account.
URL: https://medium.com/@matheodaly.md/using-google-drive-api-with-python-and-a-service-account-d6ae1f6456c2
33. Creating and managing projects.
URL: https://cloud.google.com/resource-manager/docs/creating-managing-projects
34. Manage files and folders in the Google Drive API

URL: https://developers.google.com/drive/api/guides/about-files

35. Google API Client

URL: https://pypi.org/project/google-api-python-client/

36. Storing the secrets (passwords) in a separate file.

URL: https://stackoverflow.com/questions/25501403/storing-the-secrets-passwords-in-a-separate-file

37. Command line tool and library for transferring data with URLs.

URL: https://curl.se/

38. Authorize credentials for a desktop application Google Drive API

URL: https://developers.google.com/drive/api/quickstart/python#authorize_credentials_for_a_desktop_application

39. Using OAuth 2.0 for Server to Server Applications

URL: https://developers.google.com/identity/protocols/oauth2/service-account

40. Google Drive API made easy. Maintained fork of PyDrive.

URL: https://pypi.org/project/PyDrive2/

41. Logging facility for Python.

URL: https://docs.python.org/3/library/logging.html

42. argparse — Parser for command-line options, arguments and sub-commands.

URL: https://docs.python.org/3/library/argparse.html

43. The Ultimate Guide to the Zoom API.

URL: https://spectralops.io/blog/guide-zoom-api/

44. Zoom API Postman Collection.

URL: https://www.postman.com/solar-satellite-690492/workspace/zoom/request/7450186-b6917806-03c8-47e7-b95f-c83589e3f688

45. Python programming language: Google funds projects aimed at supply-chain security.

URL: https://www.zdnet.com/article/python-programming-language-google-funds-projects-aimed-at-supply-chain-security/

Додаток А. Текст програми

```
import config
import datetime
# import plogger.info
import re
import os
import requests
import urllib.parse
# import json
import base64
import google.auth
from google.oauth2 import service_account
from googleapiclient.http import MediaFileUpload
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from pydrive2.auth import GoogleAuth
from pydrive2.drive import GoogleDrive
import sys
import argparse
from dateutil.rrule import rrule, MONTHLY
from hurry.filesize import size
import logging
# from logging import Formatter, Logger, StreamHandler
from logging import Formatter, StreamHandler
from logging.handlers import RotatingFileHandler

PATH_TO_DOWNLOAD_DIR = '/home/user/1/'
# Create logger z2g
logger = logging.getLogger("z2g")
# DEBUG (low) # INFO # WARNING # ERROR # CRITICAL (high)
logger.setLevel(logging.DEBUG)
# create rotating file handler
now = datetime.datetime.now().strftime("%Y-%m-%d.%H.%M.%S")
# check for log folder!!!
file_handler = RotatingFileHandler("logs/" + now + "-z2g.log")
file_handler.setLevel(logging.DEBUG)
# create console handler
console_handler = StreamHandler()
console_handler.setLevel(logging.INFO)
# create a formatter and add it to the handlers
formatter = Formatter("%(asctime)s %(levelname)s - %(message)s")
file_handler.setFormatter(formatter)
console_handler.setFormatter(formatter)
# add the handlers to the logger
logger.addHandler(file_handler)
logger.addHandler(console_handler)
total_files_size_all_users = 0
zoom_token = ''
token_ttl = datetime.datetime.now().timestamp()


# ********************************************************************************
# *    GET ZOOM TOKEN
# ********************************************************************************
def update_token():
    global zoom_token
    global token_ttl
    if not zoom_token:
        logger.debug('Requesting token ...')
        client_id = config.client_id
        client_secret = config.client_secret
```

```
        account_id = config.account_id
        auth_base_str = client_id + ':' + client_secret
        # logger.debug (base64.b64encode(auth_base_str.encode('utf8')).decode('utf8'))
        url = "https://zoom.us/oauth/token?grant_type=account_credentials&account_id=" + account_id
        payload = {}
        headers = {
            'Authorization': 'Basic ' + base64.b64encode(auth_base_str.encode('utf8')).decode('utf8')
        }
        json_response = requests.request("POST", url, headers=headers, data=payload).json()
        # logger.debug('To' + str(json_response['access_token']))
        # Bearer_token --> json.dumps(json_response, indent=1)
        logger.debug('Token retrieved...')
        logger.debug(json_response['access_token'])
        zoom_token = json_response['access_token']
    else:
        if datetime.datetime.now().timestamp() - token_ttl > 3500:
            token_ttl = datetime.datetime.now().timestamp()
            zoom_token = ''
            update_token()
        else:
            return zoom_token
    #    print(zoom_token)
    return zoom_token


def update_headers():
    new_headers = {
        'Authorization':
            'Bearer {}'.format(update_token()),
        'content-type':
            'application/json',
    }
    logger.debug('New headers...')
    logger.debug(new_headers)
    return new_headers


# ********************************************************************************
# *    API REQUEST List recordings of an account
# *     https://marketplace.zoom.us/docs/api-reference/zoom-api/ma/#operation/getAccountCloudRecording
# *     https://api.zoom.us/v2/users/{}/recordings?from={}&to={}&page_size=300&next_page_token=bbbbbb
# *     https://api.zoom.us/v2/users/
# *     logger.info(user_id.strip('\n'))
# ********************************************************************************

def setZoomApiRequest(query_params):
    global total_files_size_all_users
    start_day = datetime.datetime.strptime(query_params.start, "%Y-%m-%d")
    end_day = datetime.datetime.strptime(query_params.end, "%Y-%m-%d")
    # diff = end_day - start_day
    #    logger.debug("time diff --> " + str(diff.days))
    # logger.debug(access_token)
    filelist = []
    next_p_token = ''
    for dt in rrule(MONTHLY, dtstart=start_day, until=end_day):
        st = dt.strftime("%Y-%m-%d")
        en = (dt + datetime.timedelta(days=30)).strftime("%Y-%m-%d")
        if (dt + datetime.timedelta(days=30)) > end_day:
            en = end_day.strftime("%Y-%m-%d")
        logger.info('***** Time period ***** from ' + st + ' to ' + en)
        total_files_size_all_users = 0
        headers1 = update_headers()
        if query_params.user == 'all':
            users_count = 0
            while True:
                url1 = 'https://api.zoom.us/v2/users/?page_size=30'
```

```
            if next_p_token != '':
                url1 += '?&next_page_token=' + next_p_token
            # the expiration time of the next_page_token is 15 minutes !!!
            response1 = requests.get(url1, headers=headers1)
            data1 = response1.json()
            logger.debug(data1)
            logger.debug("All users list - next_p_token ==> " + (
                data1["next_page_token"] if (data1["next_page_token"]) else "NONE"))
            logger.debug("Total users in response ==> " + str(data1["total_records"]))
            for user in data1['users']:
                logger.info(user['email'])
                users_count += 1
                if user['email'] in query_params.skip_emails:
                    logger.debug("Skipping email ==> " + user['email'])
                    break
                user_files_size = 0
                filelist = filelist + getAccountCloudRecordings(st, en, user['email'], query_params.action,
                                                headers1)
            # next_p_token = data1["next_page_token"]
            # update next_page_token
            next_p_token = data1["next_page_token"]
            if next_p_token == '':
                break
        else:
            if query_params.user in query_params.skip_emails:
                logger.debug("Skipping email ==> " + query_params.user)
                logger.debug("One email and one skip ==> exit")
                sys.exit(2)
            filelist = filelist + getAccountCloudRecordings(st, en, query_params.user, query_params.action, headers1)
            users_count = 1
        logger.warning("******* Summary from " + st + ' to ' + en)
        logger.warning("Total users count --> " + str(users_count))
        logger.warning("All Users Total files size --> " + size(total_files_size_all_users))
        logger.warning("**********" + "\n")
    return filelist


def getAccountCloudRecordings(start_d, end_d, user, action, headers2):
    # https://stackoverflow.com/questions/34503540/why-does-python-give-oserror-errno-36-file-name-too-long-for-filename-short
    global total_files_size_all_users
    url = 'https://api.zoom.us/v2/users/{}/recordings?from={}&to={}&page_size=30'.format(user, start_d, end_d)
    # logger.info(url)
    try:
        response = requests.get(url, headers=headers2)
        data = response.json()
        logger.debug(data)
        # logger.info(data)
        logger.info("Meetings find " + str(data["total_records"]))
        logger.debug("Records for one user - next_p_token ==> "
                     + ((data["next_page_token"]) if data["next_page_token"] else "NONE"))
        files_size = 0
        records = []  # list or json?
        for meetings in data['meetings']:
            if ('recording_files' in meetings):
                for recs in meetings['recording_files']:
                    logger.debug("id --> " + recs['id'] + "   meeting_id --> " + recs['meeting_id'] +
                                 "  file_type --> " + recs['file_type'] + "  file_size --> " + str(recs['file_size']))
                    '''one_record = [recs['id'], recs['meeting_id'], recs['file_type'], str(recs['file_size']),
recs['download_url']]'''
                    files_size += recs['file_size']
                    # the ecryptfs layer imposes a limit of 143 characters to the filename length
                    TMP_NAME = meetings['topic']
                    if len(TMP_NAME) > 110:
                        recoding_name = TMP_NAME[0:110] + ' ' + recs['recording_start'] + '.' + recs['file_type']
                        logger.info("Found record too long name --> " + (meetings['topic']))
```

```
                logger.info("Cutting to --> " + (TMP_NAME[0:110]))
            else:
                recoding_name = meetings['topic'] + ' ' + recs['recording_start'] + '.' + recs['file_type']
                logger.info("Found record --> " + (
                    recoding_name + ' ' + size(recs['file_size']) if recoding_name else "NONE"))
            if action == 'download':
                downloadCloudRecording(recs['download_url'], recoding_name)
                #upload_to_gdrive_OAuth(re.sub(r'[^A-Za-z]+', '_', user))
                upload_to_gdrive_api_v3(re.sub(r'[^A-Za-z]+', '_', user))
                #delAccountCloudRecording(recs['meeting_id'], recs['id'])
            records.append(recoding_name)
        else:
            logger.debug("No recording_files for " + meetings['topic'] + ". Meetings uuid " + meetings['uuid'])
    except HttpError as error:
        logger.error(F'An error occurred: {error}')
    logger.info("files size --> " + size(files_size) +
                (" " + user + " from " + start_d + ' to ' + end_d + " NOT NULL!" if (files_size > 0) else "") + " \n")
    total_files_size_all_users += files_size
    return records
    # logger.debug('data ==> ', json.dumps(data, indent=1))


# ***************************************************************************
# *    DOWNLOAD AND DELETE RECORDING FROM ZOOM
# *     https://developers.zoom.us/docs/guides/guides/managing-recordings/
# ***************************************************************************
# *     Get meeting recordings*
# *     https://marketplace.zoom.us/docs/api-reference/zoom-api/methods/#operation/recordingGet
# *
# *     API REQUEST Delete a meeting recording file
# *     https://marketplace.zoom.us/docs/api-reference/zoom-api/ma/#operation/recordingDeleteOne
# *     DELETE /accounts/{accountId}/meetings/{meetingId}/recordings/{recordingId}
# *     https://api.zoom.us/v2/accounts/{accountId}/meetings/{meetingId}/recordings/{recordingId}
# *     Default: trash
# *     response HTTP/1.1 204 No Content --> Meeting recording file deleted.
# ***************************************************************************

def downloadCloudRecording(download_link, file_name):
    headers = {
        "Authorization": "Bearer " + update_token()
    }

    # Make a GET request to the API endpoint to download the recording
    logger.info(file_name + ' downloading...')
    response = requests.get(download_link, headers=headers)
    # file_name need to sanitize
    # Check if the response was successful (status code 200)
    if response.status_code == 200:
        # Save the recording to a file
        with open(PATH_TO_DOWNLOAD_DIR + file_name, "wb") as f:
            f.write(response.content)
        logger.info('{} downloaded successfully.'.format(file_name))
    else:
        logger.error('Failed to download {}. Error code:'.format(file_name), response.status_code)
        sys.exit(2)


def delAccountCloudRecording(meetingId, recordingId):
    logger.info("Moving to trash...")
    logger.debug("recordingId {}    meetingId {} ".format(recordingId, meetingId))
    headers = {
        'Authorization': 'Bearer {}'.format(update_token())
    }
    # If a UUID starts with "/" or contains "//" (example: "/ajXp112QmuoKj4854875=="),
    # you must double encode the UUID before making an API request.
    meetingId = urllib.parse.quote(meetingId, safe='')
```

```
    recordingId = urllib.parse.quote(recordingId, safe='')
    meetingId = urllib.parse.quote(meetingId, safe='')
    recordingId = urllib.parse.quote(recordingId, safe='')
    logger.debug("recordingId {}    meetingId {} ".format(recordingId, meetingId))
    url = 'https://api.zoom.us/v2/meetings/{}/recordings/{}?action=trash'.format(meetingId, recordingId)
    logger.debug(url)
    response = requests.delete(url, headers=headers)
    logger.debug(response)
    if response.status_code in (200, 204):
        # Recording deleted  Meeting recording file deleted
        logger.info('Moved to trash successfully.' + '\n')
    else:
        logger.error('Failed to Delete. Error code:', response.status_code)
    return


# ***************************************************************************
# *    UPLOAD TO GDRIVE  Drive API v2 OAuth
# ***************************************************************************
def gdrive_check_folder_OAuth(gdrive, user_folder_name):
    # check or create
    files = []
    page_token = None
    query_string = "q: '12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and " \
                   "mimeType = 'application/vnd.google-apps.folder'"
    response = gdrive.ListFile({'q': "'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and trashed=false and
mimeType = "
                                     "'application/vnd.google-apps.folder'"}).GetList()
    logger.info("Reading folders list in shared Drive ...")
    #logger.debug(response)
    for rec in response:
        # logger.debug('title: %s' % (rec['title']))
        if rec['title'] == user_folder_name:
            logger.info(user_folder_name + " folder exist. This is good!")
            return rec['id']
    logger.info(user_folder_name + " folder does not exist --> create it")
    ##########################################
    '''file_list = drive.ListFile(
        {
            "q": '"{}" in parents and title="{}" and trashed=false'.format(
                parentId, myFileName
            )
        }
    ).GetList()'''
    ########################################################
    newFolder = gdrive.CreateFile({'title': user_folder_name, "parents": [{"kind": "drive#fileLink", "id": \
        "12waDsJhsnsiPjQplgKaEbe9RBEiycj90"}], "mimeType": "application/vnd.google-apps.folder"})
    newFolder.Upload()
    logger.info('user_folder_created')
    return newFolder.get('id')


def upload_to_gdrive_OAuth(user):
    '''creds, _ = google.auth.default()
    try:
        # create drive api client
        service = build('drive', 'v3', credentials=creds)
        gdrive_check_folder(service, user)

    except HttpError as error:
        logger.info(F'An error occurred: {error}')
        files = None'''

#oauth2client Guide to Drive API v2 _OAuth
    gauth = GoogleAuth()
    gauth.LocalWebserverAuth()
```

```
    drive = GoogleDrive(gauth)  # Create GoogleDrive instance with authenticated GoogleAuth instance
    # oauth2client
    upl_folder_id = gdrive_check_folder_OAuth(drive, user)  # return google folder ID
    for file_to_upl in os.listdir(PATH_TO_DOWNLOAD_DIR):
        f = os.path.join(PATH_TO_DOWNLOAD_DIR, file_to_upl)
        if os.path.isfile(f):
            try:
                logger.info(str(file_to_upl) + '  uploading...')
                upl_file = drive.CreateFile({'title': file_to_upl, 'parents': [{'id': upl_folder_id}]})
                upl_file.SetContentFile(f)
                upl_file.Upload()
                logger.info(str(file_to_upl) + '  uploaded successfully.')
            except HttpError as error:
                logger.error(F'An error occurred: {error}')
            else:
                os.remove(f)

    '''file_list = drive.ListFile({'q': "'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and
trashed=false"}).GetList()
    logger.info('Upload to GDRIVE --> ' + user)
    for file1 in file_list:
        logger.info('title: %s, id: %s, type: %s' % (file1['title'], file1['id'], file1['mimeType']))'''

    # shred disk
    '''folder_id = '12waDsJhsnsiPjQplgKaEbe9RBEiycj90'

    # download path
    file1 = drive.CreateFile({"mimeType": "text/csv", "parents": [{"kind": "drive#fileLink", "id": folder_id}]})
    file1.SetContentFile("test222.txt")
    file1.Upload() # Upload the file.
    logger.info('Created file %s with mimeType %s' % (file1['title'], file1['mimeType']))
    file_list = drive.ListFile({'q': "'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents and trashed=false"}).GetList()
    pp = plogger.info.Prettylogger.infoer(indent=4)
    plogger.info.pp(json.dumps(file_list, indent=1))'''

# ********************************************************************************
# *    UPLOAD TO GDRIVE  Drive API v2 OAuth
# ********************************************************************************

def upload_to_gdrive_api_v3(user):
    # connection to the Drive API service_account_json_key **********************************
    scope = ['https://www.googleapis.com/auth/drive']
    service_account_json_key = '/home/user/OneDrive-KHDU/z2g/my-project-16-10-2023-60fe78a7b6ef.json'
    credentials = service_account.Credentials.from_service_account_file(
        filename=service_account_json_key,
        scopes=scope)
    service = build('drive', 'v3', credentials=credentials)
    # connection to the Drive API service_account_json_key **********************************
    upl_folder_id = gdrive_check_folder_api_v3(service, user)  # return google folder ID
    for file_to_upl in os.listdir(PATH_TO_DOWNLOAD_DIR):
        f = os.path.join(PATH_TO_DOWNLOAD_DIR, file_to_upl)
        if os.path.isfile(f):
            try:
                logger.info(str(file_to_upl) + '  uploading...')
                file_metadata = {
                    'name': file_to_upl,
                    'parents': [upl_folder_id]
                }
                media = MediaFileUpload(f, resumable=True)
                upl_file = service.files().create(body=file_metadata, media_body=media,supportsAllDrives=True,
                                          fields='id').execute()
                print(F'uploaded successfully File ID: "{upl_file.get("id")}".')
                logger.info('File ID: ' + upl_file.get("id"))
                logger.info(str(file_to_upl) + '  uploaded successfully.')
            except HttpError as error:
                logger.error(F'An error occurred: {error}')
```

```
            else:
                os.remove(f)

def gdrive_check_folder_api_v3(gdrive_service, user_folder_name):
    page_token = None
    i = 1
    j = 1
    # Call the Drive v3 API
    while True:
        response = gdrive_service.files().list(
            pageSize=1000,
            supportsAllDrives=True,
            includeItemsFromAllDrives=True,
            fields='nextPageToken, files(id,parents,name,mimeType)',
            q="mimeType='application/vnd.google-apps.folder' and trashed=false and
'12waDsJhsnsiPjQplgKaEbe9RBEiycj90' in parents",
            pageToken=page_token
        ).execute()
        # get the results
        print("Reading folders list in shared Drive ...")
        #print("*** page = ", i)
        #print(response)
        for rec in response.get('files', []):
            #print(rec)
            #print("*** row num = ", j)
            if rec['name'] == user_folder_name:
                print(user_folder_name + " folder exist. This is good!")
                return rec['id']
            j += 1
        page_token = response.get('nextPageToken', None)
        i += 1
        if page_token is None:
            break
    print(user_folder_name + " folder does not exist --> create it")
    folder_metadata = {
        'name': user_folder_name,
        'mimeType': 'application/vnd.google-apps.folder',
        'parents': ["12waDsJhsnsiPjQplgKaEbe9RBEiycj90"]
    }
    newFolder = gdrive_service.files().create(body=folder_metadata, fields='id', supportsAllDrives=True
                                              ).execute()
    print(F'Folder ID: "{newFolder.get("id")}".')
    logger.info('Folder ID: ' + newFolder.get("id"))
    return newFolder.get('id')


# ********************************************************************************
# *    PARSE INPUT ARGUMENTS
# ********************************************************************************
def parse_args():
    today = datetime.date.today().strftime("%Y-%m-%d")
    first_day_of_month = (datetime.datetime.today().replace(day=1)).strftime("%Y-%m-%d")
    # one_month_ago = (date.today() - datetime.timedelta(days=30)).strftime("%Y-%m-%d")
    # command line args
    input_args = argparse.ArgumentParser(description='ZOOM cloud recordings processing.',
                                         epilog='KHDU 2023. Volodymyr Manziuk',
                                         formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    input_args.add_argument('-u', '--user', default='volodymyr.manziuk@university.kherson.ua',
                            help='username (email) - download this Zoom user\'s recordings')
    # https://stackoverflow.com/questions/15753701/how-can-i-pass-a-list-as-a-command-line-argument-with-argparse
    # list-as-a-command-line-argument-with-argparse
    input_args.add_argument('-k', '--skip_emails', nargs='*', default=[],
                            help='username (email) - skip this Zoom user\'s recordings')
    input_args.add_argument('-a', '--action', choices=['view', 'download'], default='view',
                            help='view/download')
    input_args.add_argument('-s', '--start',
```

```
                    default=first_day_of_month, help='The start date to download recordings, yyyy-mm-dd')
    input_args.add_argument('-e', '--end',
                    default=today, help='The end date to which to download recordings, yyyy-mm-dd')
    input_args.add_argument('-v', '--log_level', choices=['NOTSET', 'DEBUG', 'INFO', 'WARNING', 'ERROR',
'CRITICAL'],
                    default='DEBUG',
                    help='Logging Levels')
    return input_args.parse_args()


# ********************************************************************************
# *    CONFIRM MESSAGE
# ********************************************************************************
def confirm(message):
    answer = None
    while answer != "y":
        answer = input(message + " Are you sure? [y/n]: ").lower()
        if answer == "n":
            sys.exit(2)


# ********************************************************************************
# *    MAIN
# ********************************************************************************

def main():
    params = parse_args()
    logger.debug(params)
    logger.setLevel(params.log_level)
    logger.info("User --> " + params.user)
    logger.info("Action --> " + params.action)
    logger.info("The start date --> " + params.start)
    logger.info("The end date --> " + params.end)
    logger.info("Skip users --> " + str(params.skip_emails))
    logger.info("Log level --> " + params.log_level)
    logger.info("********************************")
    if params.action == 'download':
        confirm("All the parameters are correct?")
    files = setZoomApiRequest(params)
    '''for n in files:
        logger.info(n)'''


if __name__ == '__main__':
    main()
```