МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №2

з дисципліни «Системи з розподіленими даними»

**Виконав:**

Студент 5-го курсу

ТЕФ, гр. ТР-02мп

Круглий Д.В.

**Перевірив:**

Варава І.А.

**Київ-2021**

**Тема:** розробка клієнтського додатка Google Drive для Windows.

**Мета:** навчитись розробляти додатки для управління даними на гугл диску і керувати за допомогою цього додатку файловою системою.

**Представлення коду програми**

import io

import pickle

import os.path

from google.auth.transport.requests import Request

from googleapiclient.http import MediaIoBaseDownload

from google\_auth\_oauthlib.flow import InstalledAppFlow

from googleapiclient.discovery import build, MediaFileUpload

SCOPES = ['https://www.googleapis.com/auth/drive']

class API(object):

   def \_\_init\_\_(self):

       creds = None

       if os.path.exists('token.pickle'):

           with open('token.pickle', 'rb') as token:

               creds = pickle.load(token)

       if not creds or not creds.valid:

           if creds and creds.expired and creds.refresh\_token:

               creds.refresh(Request())

           else:

               flow = InstalledAppFlow.from\_client\_secrets\_file('credentials.json', SCOPES)

               creds = flow.run\_local\_server(port=0)

           with open('token.pickle', 'wb') as token:

               pickle.dump(creds, token)

       self.\_service = build('drive', 'v3', credentials=creds)

       self.upd()

       self.act\_folder\_id   = '1dIYVcBUBy2AMOqDjVmr214uuTeW2Q7or'

       self.act\_folder\_name = "Root"

       self.root\_path       = 'Root'

   def upd(self):

       results = self.service.files().list(

           pageSize = 1000,

           fields   = "nextPageToken, files(id, ownedByMe, name, parents, mimeType, webContentLink)",

       ).execute()

       self.\_items = results.get('files', [])

   @property

   def service(self):

       return self.\_service

   @property

   def items(self):

       return self.\_items

   def cd(self, argue):

       print(f'Change directory: {argue}')

       if argue == "/.":

           for elem in self.items:

               if elem["id"] == self.act\_folder\_id and elem['mimeType'] == 'application/vnd.google-apps.folder':

                   self.act\_folder\_id = elem["parents"][0]

                   self.act\_folder\_name = elem["name"]

                   self.upd()

                   self.ls()

       else:

           for elem in self.items:

               if elem["name"] == argue and elem['mimeType'] == 'application/vnd.google-apps.folder':

                   self.act\_folder\_id = elem["id"]

                   self.act\_folder\_name = argue

                   self.upd()

                   self.ls()

   def ls(self):

       print(f"Show files: {self.act\_folder\_name}")

       for elem in self.items:

           if 'parents' in elem:

               if elem["parents"][0] == self.act\_folder\_id:

                   print(elem['name'], elem['mimeType'])

   def mkdir(self, arg):

       print(f"Making directory: {arg}")

       file\_metadata = {'parents': [str(self.act\_folder\_id)], 'name': arg, 'mimeType': 'application/vnd.google-apps.folder'}

       file = self.service.files().create(body=file\_metadata).execute()

       self.upd()

       self.ls()

   def rename(self, arg):

       print(f"Renaming directory: {self.act\_folder\_name} {arg}")

       self.service.files().update(fileId=self.act\_folder\_id, body={'name': arg}, fields='name').execute()

       self.upd()

       self.ls()

   def rmdir(self):

       print(f"Deleting: {self.act\_folder\_name}")

       file = self.service.files().get(fileId=self.act\_folder\_id, fields="parents, name").execute()

       self.service.files().delete(fileId=self.act\_folder\_id, fields="parents").execute()

       self.act\_folder\_id = file["parents"][0]

       self.act\_folder\_name = self.service.files().get(fileId=self.act\_folder\_id, fields="parents, name").execute()["name"]

       self.upd()

       self.ls()

   def upload(self, arg, name):

       print(f'Uploading file: {arg}')

       file\_metadata = {'name': name}

       media = MediaFileUpload(arg, mimetype='image/jpeg')

       self.service.files().create(body=file\_metadata, media\_body=media, fields='id').execute()

   def download(self, arg):

       print(f'Downloading file: {arg}')

       for elem in self.items:

           if elem["name"] == arg and elem['parents'][0] == self.act\_folder\_id:

               request = self.service.files().get\_media(fileId=elem["id"])

               fh = io.FileIO(arg, 'wb')

               downloader = MediaIoBaseDownload(fh, request)

               done = False

               while done is False:

                   status, done = downloader.next\_chunk()

                   print("Download %d%%." % int(status.progress() \* 100))

               self.upd()

def parse\_command():

   print('>>', end=' ')

   gline = input().split(' ')

   command, argue, thrird = gline[0], None, None

   if len(gline) > 1:

       argue = gline[1]

   if len(gline) > 2:

       thrird = gline[2]

   return command, argue, thrird

if \_\_name\_\_ == '\_\_main\_\_':

   api = API()

   api.mkdir('some\_folder')

   input('')

   api.upload('image.jpeg', 'new\_image.jpeg')

   input('')

   api.download('other\_image.jpeg')

   input('')

   api.cd('some\_folder')

   input('')

   api.mkdir('help')

   input('')

   api.cd('help')

   input('')

   api.rmdir()

   input('')

   api.rename('other\_folder')

   while True:

       command, argue, third = parse\_command()

       if command == 'ls':

           api.ls()

       elif command == 'cd':

           api.cd(argue)

       elif command == 'mkdir':

           api.mkdir(argue)

       elif command == 'rmdir':

           api.rmdir()

       elif command == 'rename':

           api.rename(argue)

       elif command == 'upload':

           api.upload(argue, third)

       elif command == 'download':

           api.download(argue)

       else:

           print("Unknown command")

**Приклад роботи програми**

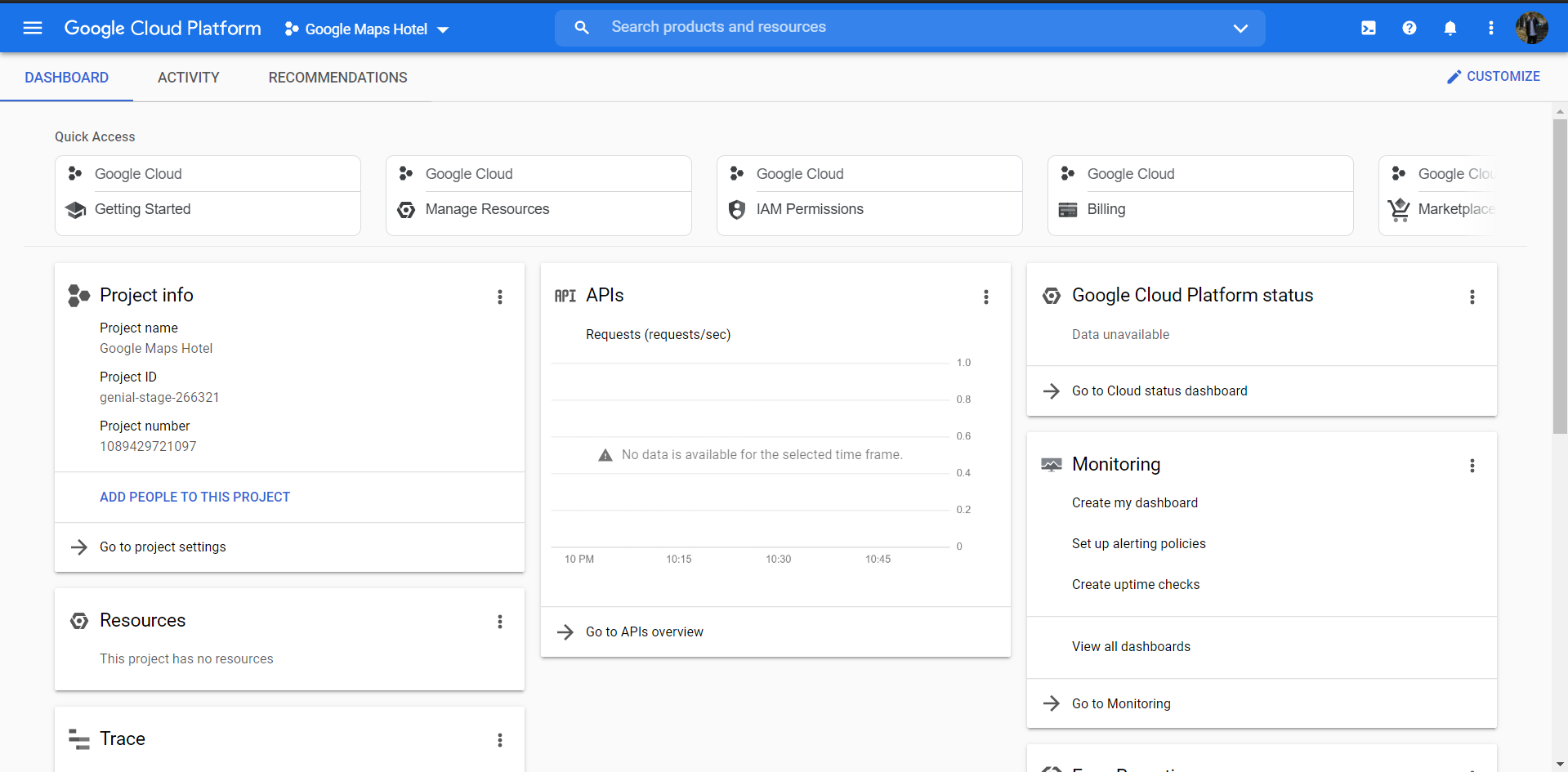


Рис. 1 – Створення Google Cloud Platform application

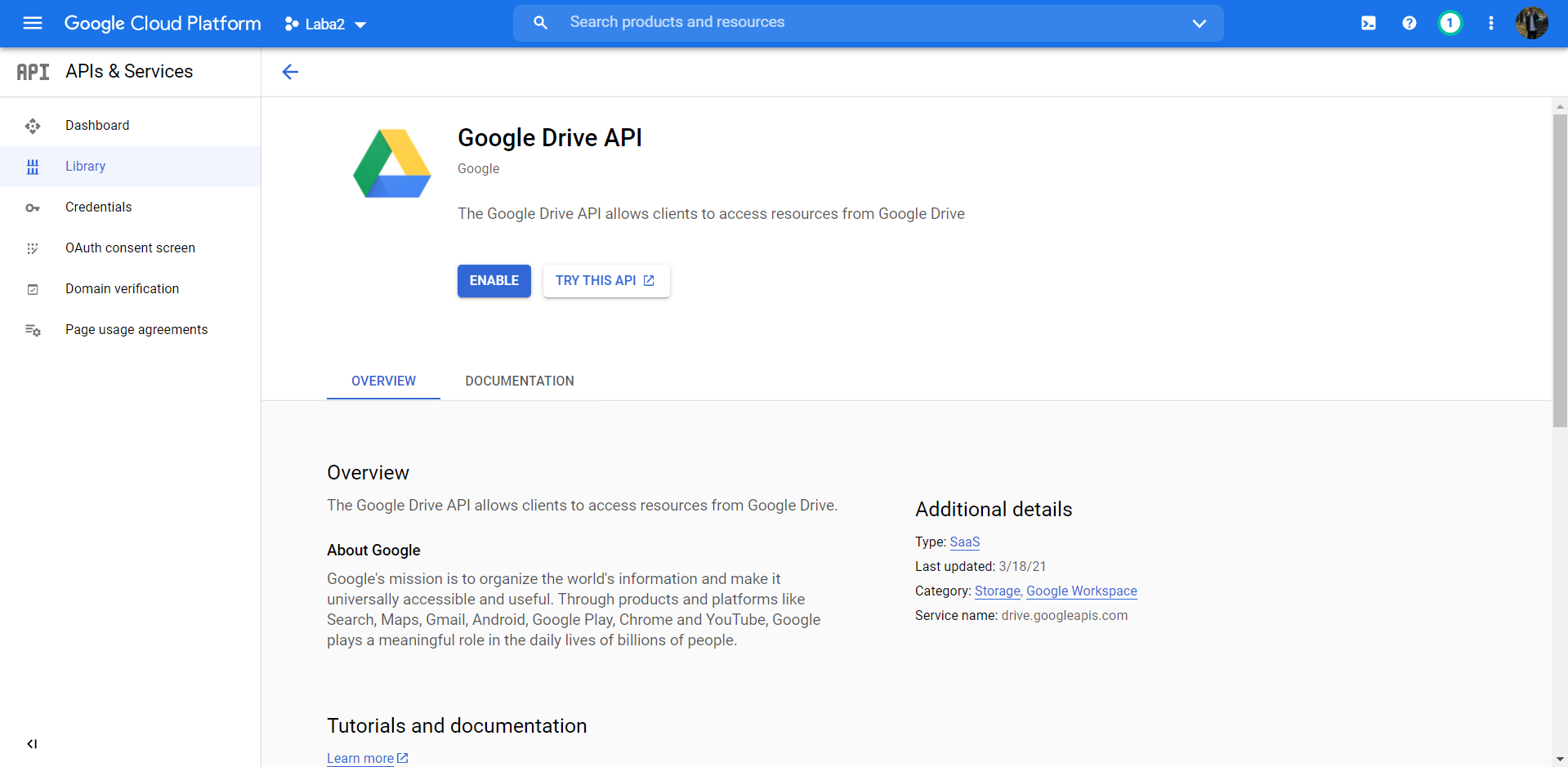


Рис. 2 – Додавання Google Drive API

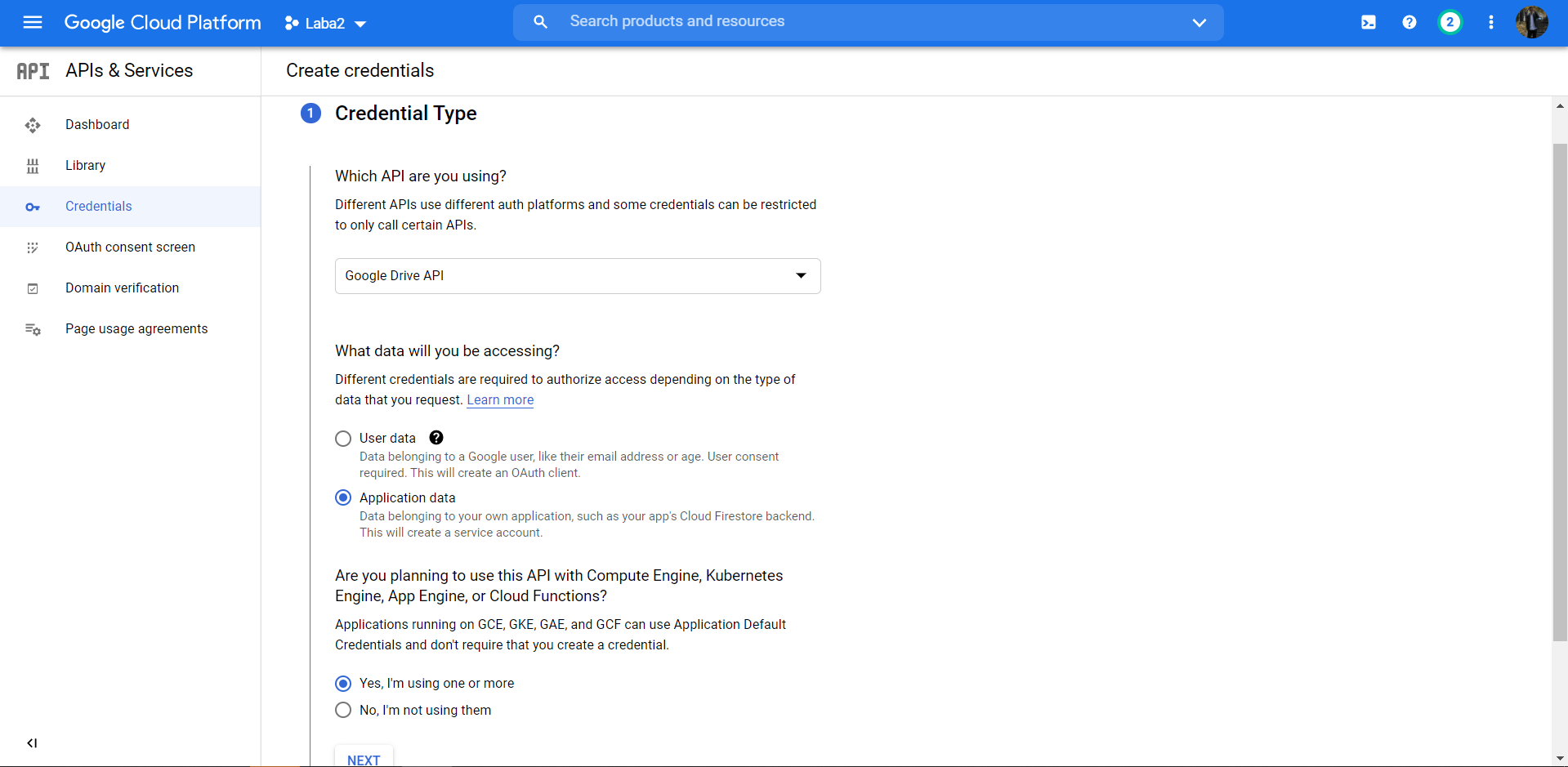


Рис. 3 – Налаштування *credentials.json* для доступу до створеного додатку

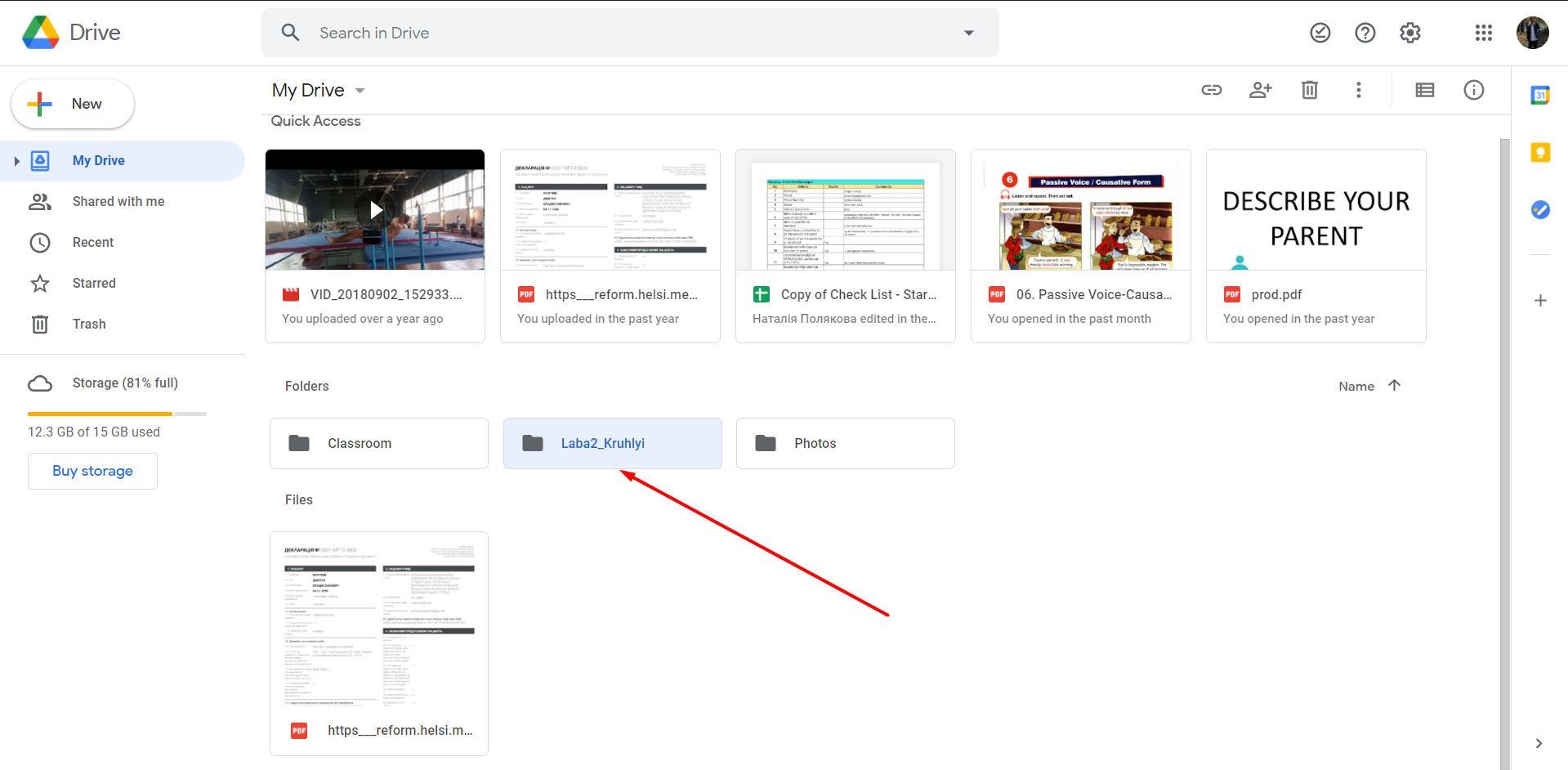


Рис. 4 – Створення нової папки Laba2\_Kruhlyi

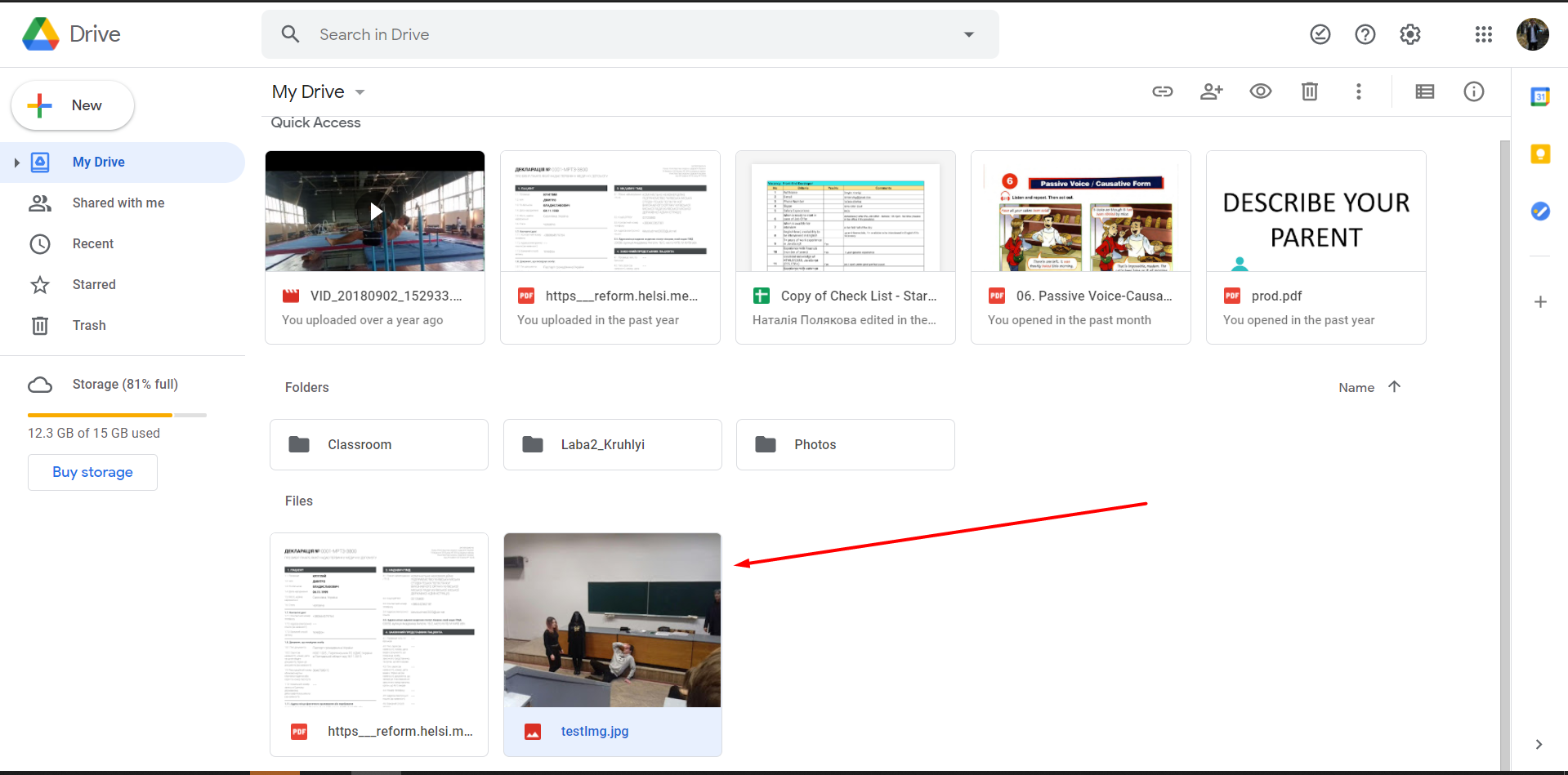


Рис. 5 – Завантаження тестового зображення з назвою *testImg.jpg*

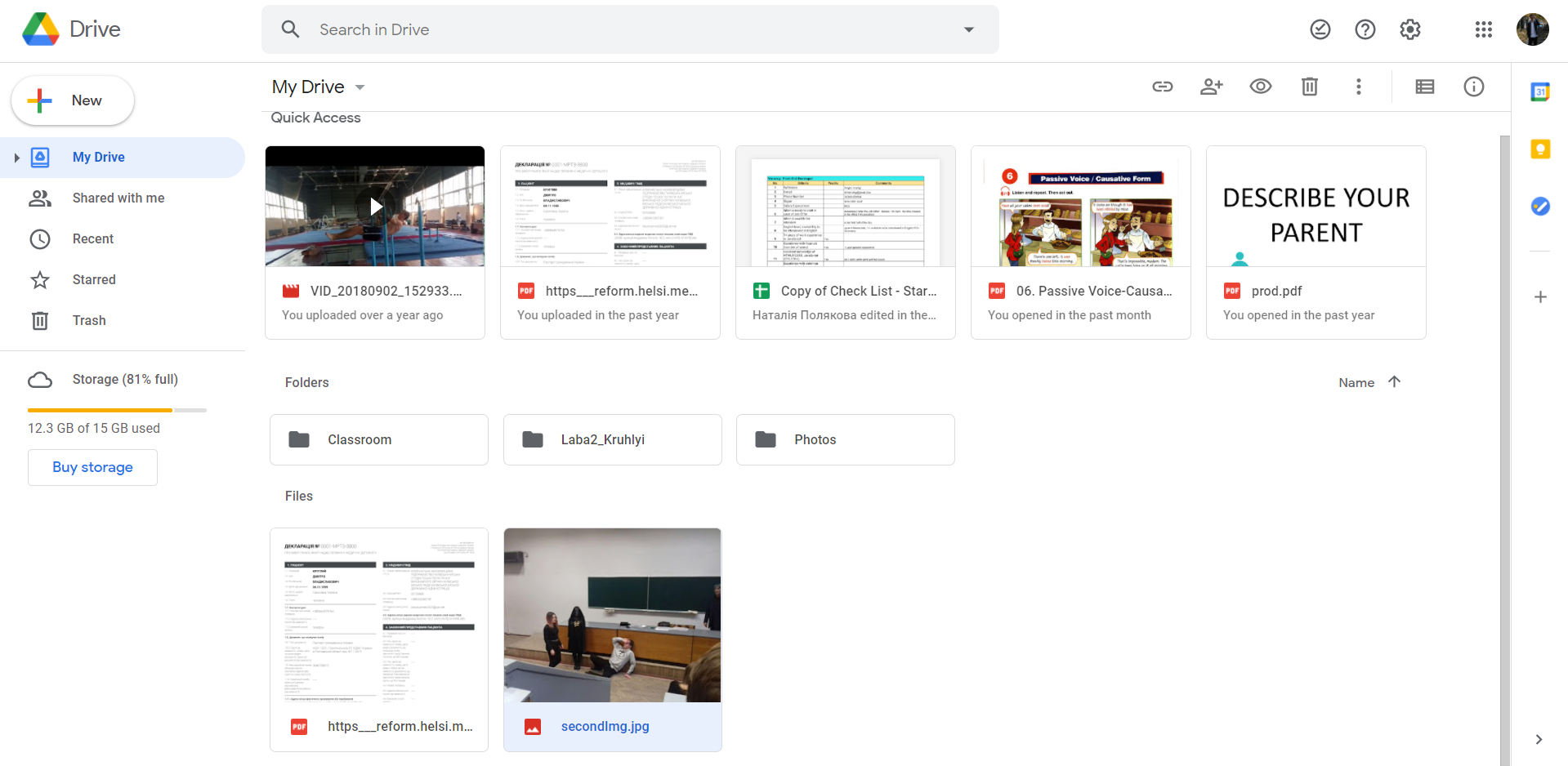


Рис. 6 – Перейменування зображення з назвою *testImg.jpg* на *secondImg.jpg*

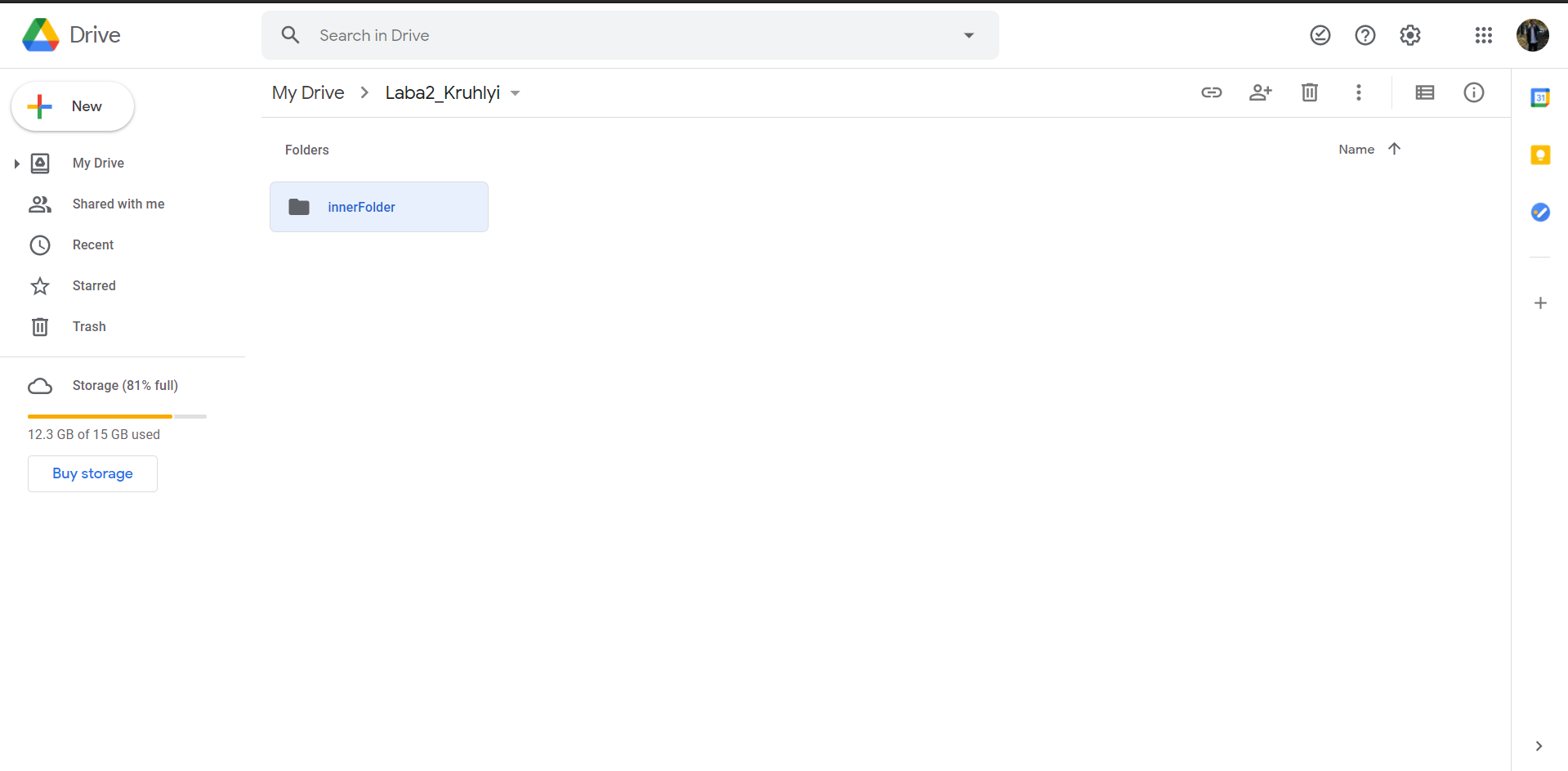


Рис. 7 – Перехід в створену директорію та створення допоміжної папки *innerFolder*

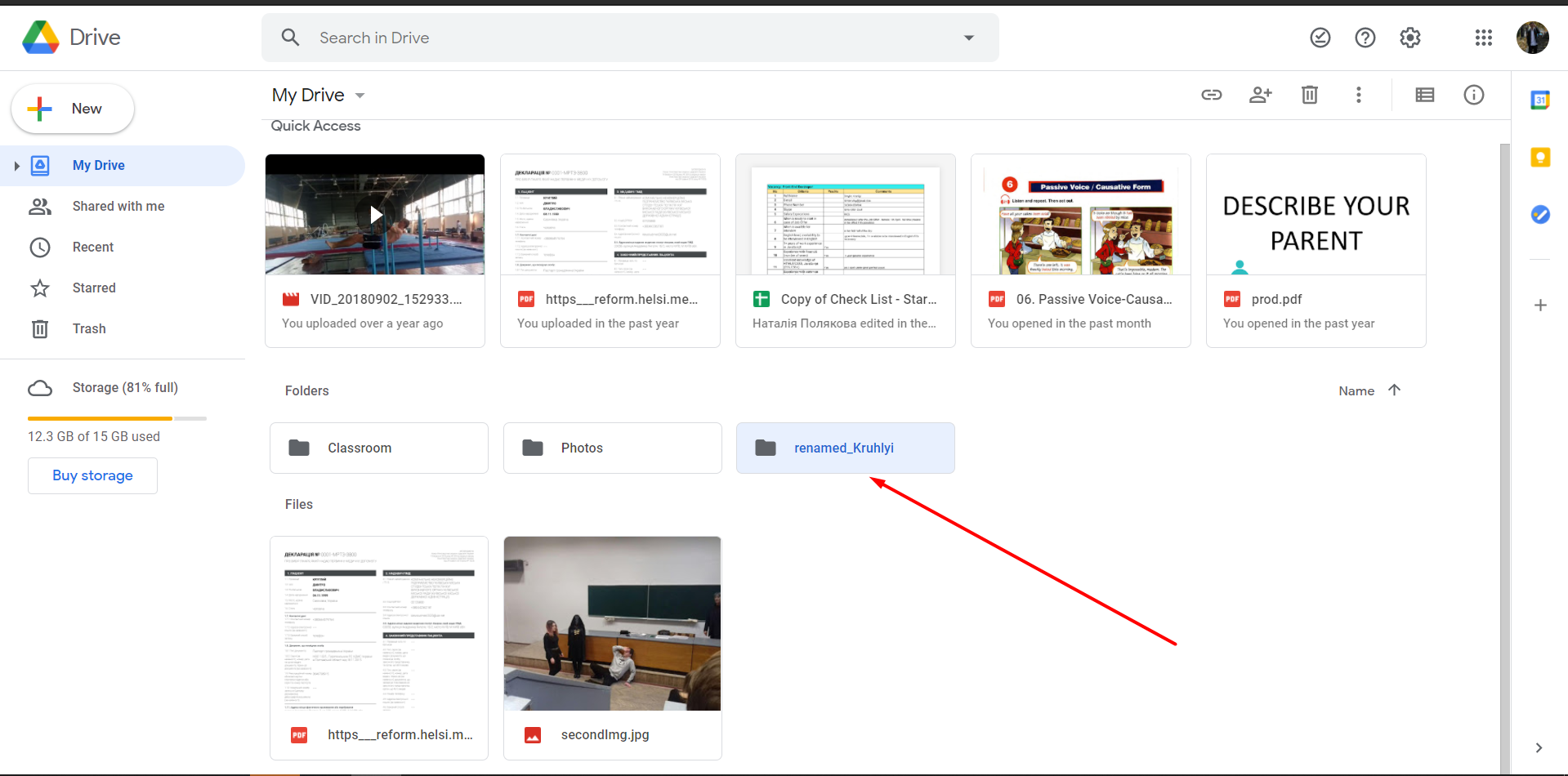


Рис. 8 – Видалення допоміжної папки та перейменування поточної папки на *renamed\_Kruhlyi*

**Висновок:** у ході виконання лабораторної роботи, я навчився розробляти додатки для управління даними за допомогою Google Drive API.