API Google Cloud, библиотека Aiogoogle | Я.Шпора

Как создать проект для работы с API платформы Google Cloud

Перейдите на консоль разработчика (дашборд) → Нажмите кнопку **Create Project** → Задайте имя проекту → Нажмите кнопку **Create**

Как подключить Google Drive API и Google Sheets API к созданному проекту

1. На плитке **APIs** нажмите **Go to APIs overview**.
2. Нажмите **Enabled APIs and services** или выберите в меню слева пункт **Library**.
3. В открывшемся окне выберите по очереди **Google Drive API** и **Google Sheets API**.
4. Создайте сервисный аккаунт:
    a. Перейдите в раздел **Credentials**.
    b. Нажмите **Create credentials** и выберите пункт **Service account**.
    c. Заполните поля **Service account name**, **Service account ID**, **Service account description**.
    d. Выберите роль для сервисного аккаунта.
    e. Назначьте права администратора вашему пользовательскому аккаунту.

Как получить JSON-файл с ключом доступа к сервисному аккаунту

Перейдите на экран **Credentials/<название вашего сервисного аккаунта>** → Нажмите **Keys** → **Add Key** → **Create New Key** → Выберите формат JSON → Нажмите **Create**

Работа с Google Таблицами

Как создать Google Таблицу через Python-код

1. Установите библиотеку *google-api-python-client*:

```
pip install google-api-python-client==2.45.0
```

2. Задайте константы, необходимые для доступа к API. В целях безопасности используйте .env-файлы.

   Пример кода:

```
import os

from apiclient import discovery
from dotenv import load_dotenv
from google.oauth2.service_account import Credentials

load_dotenv()

SCOPES = [
    os.getenv('SPEEEDSHEETS_URL'),  # Доступ к Google Таблицам.
    os.genenv('DRIVE_URL'),  # Доступ к Google Drive.
```

```
]

CREDENTIALS_FILE = (
     # Имя JSON-файла с ключом доступа к сервисному аккаунту.
    os.genenv('JSON_FILENAME')
)
```

Пример содержимого .env-файла:

```
# URL для авторизации в Google Таблицах.
SPEEEDSHEETS_URL="https://www.googleapis.com/auth/spreadsheets"
# URL для авторизации в Google Drive.
DRIVE_URL="https://www.googleapis.com/auth/drive"
# Имя файла с ключом доступа к сервисному аккаунту.
JSON_FILENAME="имя_файла.json"
```

3. Напишите функцию авторизации в Google API:

```
...


def auth() -> tuple:
    # Создать экземпляр класса Credentials.
    credentials = Credentials.from_service_account_file(
        filename=CREDENTIALS_FILE, scopes=SCOPES
```

```
    )
    # Создать экземпляр класса Resource.
    service = discovery.build('sheets', 'v4', credentials=credentials)
    return service, credentials
```

4. Напишите функцию, которая отвечает за создание таблицы, например:

```
...

def create_spreadsheet(service) -> str:
    # Тело spreadsheet.
    spreadsheet_body = {
        # Свойства документа.
        'properties': {'title': 'Бюджет путешествий', 'locale': 'ru_RU'},
        # Свойства листов документа.
        'sheets': [
            {
                'properties': {
                    'sheetType': "GRID",
                    'sheetId': 0,
                    'title': 'Отпуск 2077',
                    'gridProperties': {
                        'rowCount': 100,  # Количество строк в таблице.
                        'columnCount': 100,  # Количество столбцов в таблице.
                    },
```

```
                }
            }
        ],
    }
    # Получить ID таблицы.
    request = service.spreadsheets().create(body=spreadsheet_body)
    response = request.execute()
    spreadsheet_id = response['spreadsheetId']
    print('https://docs.google.com/spreadsheets/d/' + spreadsheet_id)
    return spreadsheet_id


# Пример использования функции.
service, credentials = auth()
create_spreadsheet(service)
```

5. Сделайте таблицу доступной для вашего аккаунта:

```
...


def set_user_permissions(spreadsheet_id, credentials) -> None:
    permissions_body = {
        'type': 'user',  # Тип учётных данных.
        'role': 'writer',  # Права доступа для учётной записи.
```

```
        'emailAddress': 'practicum@gmail.ru', # Ваш личный гугл-аккаунт.
    }

    # Создать экземпляр класса Resource для Google Drive API.
    drive_service = discovery.build('drive', 'v3', credentials=credentials)

    # Сформировать и сразу выполнить запрос на выдачу прав вашему аккаунту.
    drive_service.permissions().create(
        fileId=spreadsheet_id, body=permissions_body, fields='id'
    ).execute()
```

Как работать с данными таблиц

Данные в таблицу не добавляются, а обновляются в ней, потому что пустые значения в ячейках — тоже значения.

За обновление данных отвечает метод Google Sheets API `spreadsheets().values().update`.

Пример функции обновления данных в таблице:

```
def spreadsheet_update_values(
        service,
        spreadsheetId,
        table_values  # Список со значениями для обновления таблицы.
```

```
) -> None:
    # Тело запроса.
    request_body = {
        'majorDimension': 'ROWS',
        'values': table_values
    }
    # Запрос к Google Sheets API.
    request = service.spreadsheets().values().update(
        spreadsheetId=spreadsheetId,
        range='Отпуск 2077!A1:F20',
        valueInputOption='USER_ENTERED',
        body=request_body
    )
    # Выполнить запрос.
    request.execute()
```

Тело запроса — словарь с набором ключей, который подходит для всех методов, отвечающих за управление содержимым таблиц:

- `'majorDimension'` — ключ, значения которого определяют, как будут заполнены данные: `'ROWS'` — по строкам, `'COLUMNS'` — по столбцам.
- `'values'` — ключ со значениями ячеек в виде списка списков.
- `'range'` — название таблицы и диапазон ячеек, в которые нужно внести изменения. Этот ключ необязательный, в примере он опущен **в теле запроса,** но задан **в параметрах метода** (`range='Отпуск 2077!A1:F20'`).

- `valueInputOption` — параметр, который отвечает за то, как будут интерпретироваться входные данные. У этого параметра может быть одно из трёх значений:
  - `'INPUT_VALUE_OPTION_UNSPECIFIED'` — значение по умолчанию, документация требует его переопределения на любой из параметров ниже. Иначе программа выдаст ошибку.
  - `'RAW'` — входные данные будут использоваться так, как и написаны в коде.
  - `'USER_ENTERED'` — данные для таблицы будут проанализированы в коде и выведутся в ячейки таблицы в отформатированном виде. Будут соблюдены все правила форматирования, как если бы пользователь вводил эти данные через интерфейс Google таблиц в браузере.

Новые значения для таблицы можно задать подобным образом:

```
# Данные для заполнения: выводятся в таблице сверху вниз, слева направо.
table_values = [
    ['Бюджет путешествий'],
    ['Весь бюджет', '5000'],
    ['Все расходы', '=SUM(E7:E30)'],
    ['Остаток', '=B2-B3'],
    ['Расходы'],
    ['Описание', 'Тип', 'Кол-во', 'Цена', 'Стоимость'],
    ['Перелет', 'Транспорт', '2', '400', '=C7*D7']
]
```

Альтернативы методу `update`

- Метод `batchUpdate` — работает так же, как метод `update`, но может за один запрос записать несколько пачек данных в разные диапазоны.
- Метод `append` — записывает данные в ближайшую пустую ячейку ниже.

Работа с Google Drive API

Пример функции авторизации в Google Drive API:

```
def auth() -> tuple:
    credentials = Credentials.from_service_account_info(
        info=info, scopes=SCOPES)
    service = discovery.build('drive', 'v3', credentials=credentials)
    return service, credentials
```

Как получить список всех файлов на диске

Пример функции получения списка всех файлов на диске:

```
def get_list_obj(service):
    response = service.files().list()
    return response.execute()
```

Результат работы функции:

```
{
    "files": [
        {
            "id": "1ROfBpouFFpzILWXekFo9M8emMnCfP8OJ8-g-cO-Ireg",
            "kind": "drive#file",
            "mimeType": "application/vnd.google-apps.spreadsheet",
            "name": "Первый тестовый документ",
        },
        ...,
    ],
    "incompleteSearch": False,
    "kind": "drive#fileList",
    "nextPageToken": "2",
}}
```

- `'files'` — список файлов на диске в виде списка словарей, может быть пустым.
- `'incompleteSearch'` — указывает, каким был поиск: полным или неполным. Полный поиск (`False`) — поиску по всему диску, неполный (`True`) — только в какой-то конкретной папке.
- `'kind'` — фиксированное значение для метода `list()` (`'drive#fileList'`). Показывает тип объекта по отношению к гугл-диску: файл или папка.
- `'nextPageToken'` — токен на каждую следующую страницу с файлами на гугл-диске. При помощи этого ключа можно делать запросы, пока не вернётся значение `None` .

Информация о каждом файле приходит в виде словаря со следующими ключами:

- `'id'` — ID файла.
- `'kind'` — тип возвращаемого ресурса Google Drive: файл или папка.
- `'mimeType'` — тип документа, например, гугл-документы, гугл-таблицы, карты, аудиофайлы.
- `'name'` — имя объекта гугл-диска.

Если нужно получить список файлов определённого типа, используется метод `service.files.list` с параметром `q`:

```
response = service.files().list(
    # mimeType — тип искомого файла.
        q='mimeType="application/vnd.google-apps.spreadsheet"'
)
```

Как удалить файлы с гугл-диска

Удалить файлы можно с помощью метода `service.files().delete(<file_id>)`.

Пример функции, которая удаляет файлы по списку:

```
# Переменная spreadsheets содержит список файлов для удаления.
def clear_disk(service, spreadsheets) -> None:
    for spreadsheet in spreadsheets:
        response = service.files().delete(fileId=spreadsheet['id'])
        response.execute()
```

Асинхронная библиотека Aiogoogle

Aiogoogle — библиотека для работы с Google Drive API в асинхронном варианте.

Установить библиотеку:

```
pip install aiogoogle==5.6.0
```

Настроить библиотеку:

```
# Подключить классы асинхронной бибилиотеки.
from aiogoogle import Aiogoogle
from aiogoogle.auth.creds import ServiceAccountCreds


# Список разрешений.
SCOPES = [
    'https://www.googleapis.com/auth/spreadsheets',
    'https://www.googleapis.com/auth/drive'
]
# Словарь с учётными данными сервисного аккаунта из JSON-файла.
INFO = {
    'type': settings.type,
    'project_id': settings.project_id,
    'private_key_id': settings.private_key_id,
    'private_key': settings.private_key,
```

```
    'client_email': settings.client_email,
    'client_id': settings.client_id,
    'auth_uri': settings.auth_uri,
    'token_uri': settings.token_uri,
    'auth_provider_x509_cert_url': settings.auth_provider_x509_cert_url,
    'client_x509_cert_url': settings.client_x509_cert_url
}
# Получить объект учётных данных.
cred = ServiceAccountCreds(scopes=SCOPES, **INFO)

# Создать экземпляр класса Aiogoogle.
async def get_service():
        async with Aiogoogle(service_account_creds=cred) as aiogoogle:
        yield aiogoogle
```

Создать таблицу:

```
async def spreadsheets_create(wrapper_services: Aiogoogle) -> str:
    # Получить текущую дату для заголовка документа.
    now_date_time = datetime.now().strftime(FORMAT)
    # Создать экземпляр класса Resourse.
    service = await wrapper_services.discover('sheets', 'v4')
    # Тело запроса.
    spreadsheet_body = {
        'properties': {
```

```
            'title': f'Отчет на {now_date_time}',
            'locale': 'ru_RU',
        },
        'sheets': [
            {
                'properties': {
                    'sheetType': 'GRID',
                    'sheetId': 0,
                    'title': 'Лист1',
                    'gridProperties': {'rowCount': 100, 'columnCount': 11},
                }
            }
        ],
    }
    # Выполнить запрос.
    response = await wrapper_services.as_service_account(
        service.spreadsheets.create(json=spreadsheet_body)
    )
    spreadsheetid = response['spreadsheetId']
    return spreadsheetid
```

Предоставить права доступа вашему пользователю:

```
async def set_user_permissions(
```

```
    spreadsheetid: str, wrapper_services: Aiogoogle
) -> None:
    # Запросить права доступа для файла таблицы.
    permissions_body = {
        'type': 'user',
        'role': 'writer',
        'emailAddress': settings.email,
    }
    # Получить объект сервиса для доступа к сервисному аккаунту.
    service = await wrapper_services.discover('drive', 'v3')
    # Выполнить запрос.
    await wrapper_services.as_service_account(
        service.permissions.create(
            fileId=spreadsheetid, json=permissions_body, fields='id'
        )
    )
```

Задать права для пользователя:

```
spreadsheetid = await spreadsheets_create(wrapper_services)
await set_user_permissions(spreadsheetid, wrapper_services)
```

Обновить данные:

```python
async def spreadsheets_update_value(
    spreadsheetid: str,
    data: list,  # Список словарей с данными для добавления в таблицу.
    wrapper_services: Aiogoogle,
) -> None:
    now_date_time = datetime.now().strftime(FORMAT)
    service = await wrapper_services.discover('sheets', 'v4')
    # Тело таблицы.
    table_values = [
        ['Отчет от', now_date_time],
        ['Количество регистраций переговорок'],
        ['ID переговорки', 'Кол-во бронирований'],
    ]
    # Здесь в таблицу добавляются строки.
    for res in data:
        new_row = [str(res['meetingroom_id']), str(res['count'])]
        table_values.append(new_row)
    update_body = {'majorDimension': 'ROWS', 'values': table_values}
    response = await wrapper_services.as_service_account(
        service.spreadsheets.values.update(
            spreadsheetId=spreadsheetid,
            range='A1:E30',
            valueInputOption='USER_ENTERED',
            json=update_body,
```

```
        )
    )
```

Полезные ресурсы

Перечень методов, которые отвечают за управление содержимым таблиц

Документация метода batchUpdate

Документация метода append

Перечень всех типов mimeTipe

Документация библиотеки Aiogoogle

Практикум