|  | Universidad Nacional Autónoma de México  Dirección General de Cómputo y de Tecnologías de Información y Comunicación |  |
| --- | --- | --- |

| Clave:  2022-12/6-1840 | Servicio Social:  Análisis de Datos Educativos |
| --- | --- |

| Supervisores:  Angélica Ma. Ramírez <  Miguel Zuñiga < |  |
| --- | --- |
|  | Alumno:  García García Juan David |

| Documentación del proyecto |  |
| --- | --- |
|  | Proyecto:  Google Analytics V3  (Universal Analytics) |

**Índice**

[**Introducción**](#_54c16b33qhge) **3**

[**Ambientación del equipo de trabajo**](#_160wldcsxept) **3**

[**Introducción (API)**](#_6y8tuoo1log5) **3**

[**Documentación (API)**](#_7gza7lnipqix) **4**

[Librerías](#_jhv0ctomf4oz) 4

[Funciones](#_9mgbqqkcwsiu) 4

[initialize\_analyticsreporting](#_fvpsro82hkcv) 4

[get\_report](#_oas5psvr1zfh) 5

[handle\_report](#_jzp0coaqcvlx) 6

[fechas](#_m4j3n6oqonie) 6

[reportes\_por\_fecha\_pagina](#_fj5vv372iyn1) 7

[metricas\_dimensiones](#_pey8bpt1sceg) 7

[reportes\_usuario](#_3k64g7e9201h) 8

[concatenar](#_ilzh8eaxsgr9) 9

[reportes\_pagina](#_gkblxfsz1f7w) 10

[generardor\_carpetas](#_derr751vw4ww) 11

[inicio del programa](#_lsmxh9j70pfg) 12

[**Anexo (API)**](#_9to26ynky670) **12**

[1. KEY\_FILE\_LOCATION](#_h4nqt9ux68z5) 12

[2. Archivo ID](#_7ghp4b736ybf) 13

[3. Código](#_9x1o0drjyzcq) 13

[**Errores comunes (API)**](#_e4hbtqwmpa9s) **14**

[**Introducción (Valores Únicos)**](#_w81o0sdxumif) **14**

[**Anexo (Valores Únicos)**](#_es3gd57ellfa) **14**

[1. Código](#_ubpimhzc3z8y) 14

# **Introducción**

Este archivo servirá para almacenar toda la documentación relacionada con el proyecto sobre importar los datos de Google Analytics V3, así como las herramientas necesarias para trabajar con los datos para dejarlos dentro de un ambiente productivo.

Con el tiempo, puede que la documentación se vea modificada debido a actualización del código o integración de nuevas funcionalidades o herramientas que faciliten el productivizar los datos arrojados por esta herramienta de Google.

# **Ambientación del equipo de trabajo**

Para que el código tenga un buen funcionamiento dentro del equipo, este requerirá tener instalados algunos programas para la ejecución óptima del equipo, los cuales son:

* Anaconda >= Python 3.9 + 64-bit
* Algunas librerías extra de Python (se dejarán dentro de cada código)
* PostgreSQL >= 15.1

# **Introducción (API)**

Esta parte del documento tiene como objetivo almacenar la documentación del código programado en Python, el cual cumple con distintas funciones que son:

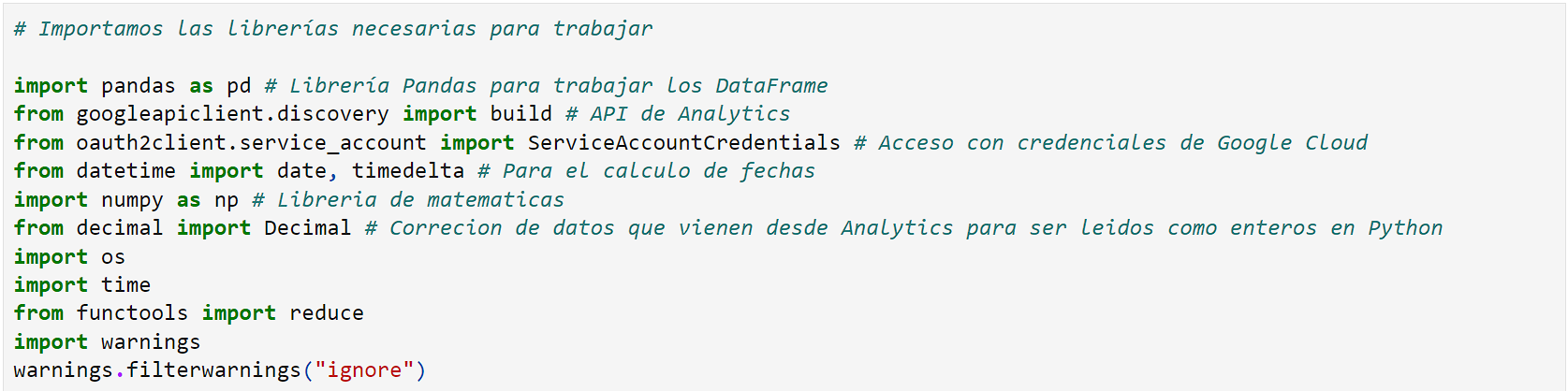
* Hacer las consultas automatizadas (por parámetros y por fecha) a Google Analytics V3 API
* Ordenar la información consultada en DataFrames de Pandas
* Hacer una limpieza rápida de los datos
* Crear las carpetas de almacenamiento
* Guardar la información en archivos CSV (por usuario y por página)

Es importante recordar que este código, sin modificaciones, solo funciona con la versión 3 de Google Analytics, la cual quedará obsoleta a mediados del año siguiente. También, es posible que con la modificación ligera del código se pueda adaptar a la versión 4, pues la gran parte del mismo está enfocada en otras tareas.

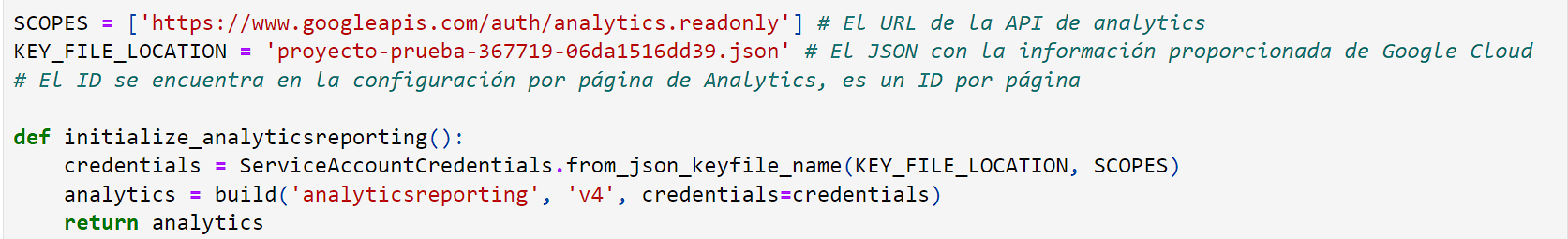
# **Documentación (API)**

Para realizar la documentación, se pondrán trozos de código explicando brevemente qué hace cada línea, dejando de lado particularidades como la sintaxis de las diversas paqueterías usadas.

## *Librerías*



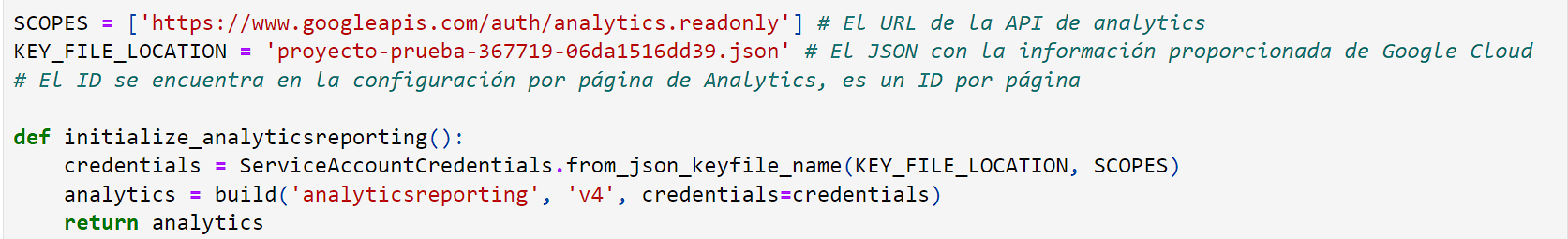
En esta sección importamos todas las librerías que usamos durante el proceso. Una vez con ello, se meterán tres cosas fundamentales para correr el código, para la obtención del ID así como la KEY\_FILE\_LOCATION, se detallará en el anexo de este documento.



## *Funciones*

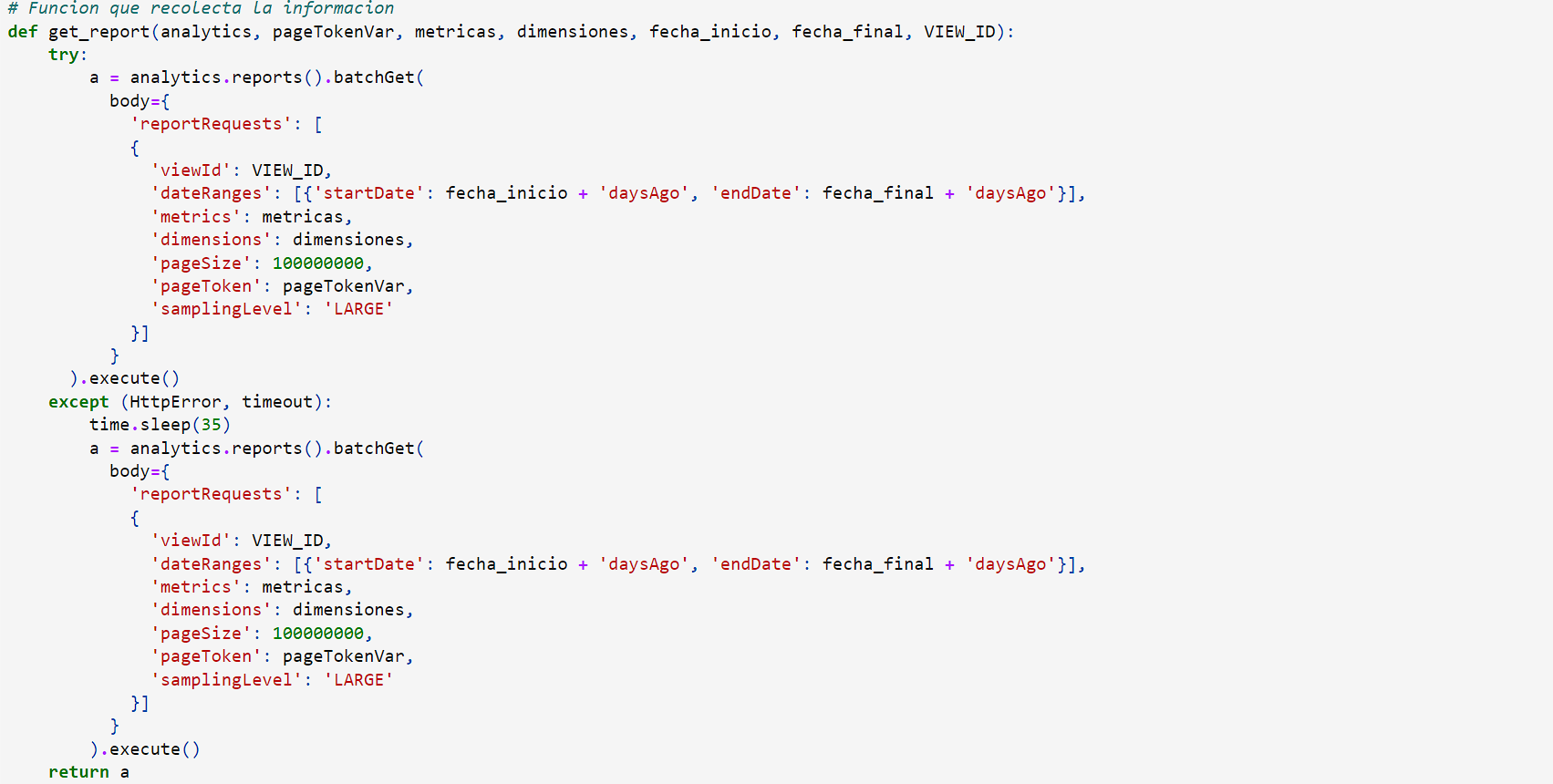
Una vez que las librerías fueron importadas, toca cargar todas las funciones necesarias para correr el programa. La mayoría de las funciones son usadas dentro de otras funciones, por lo que no tendrán interacción con el usuario, mientras que otras estarán de forma explícita durante el inicio de ejecución del código.

### *initialize\_analyticsreporting*



Sirve para establecer la conexión con la API de Google Analytics V3, básicamente le pasamos las credenciales para crear la conexión y la devolvemos para usarla al momento de generar las consultas.

### *get\_report*

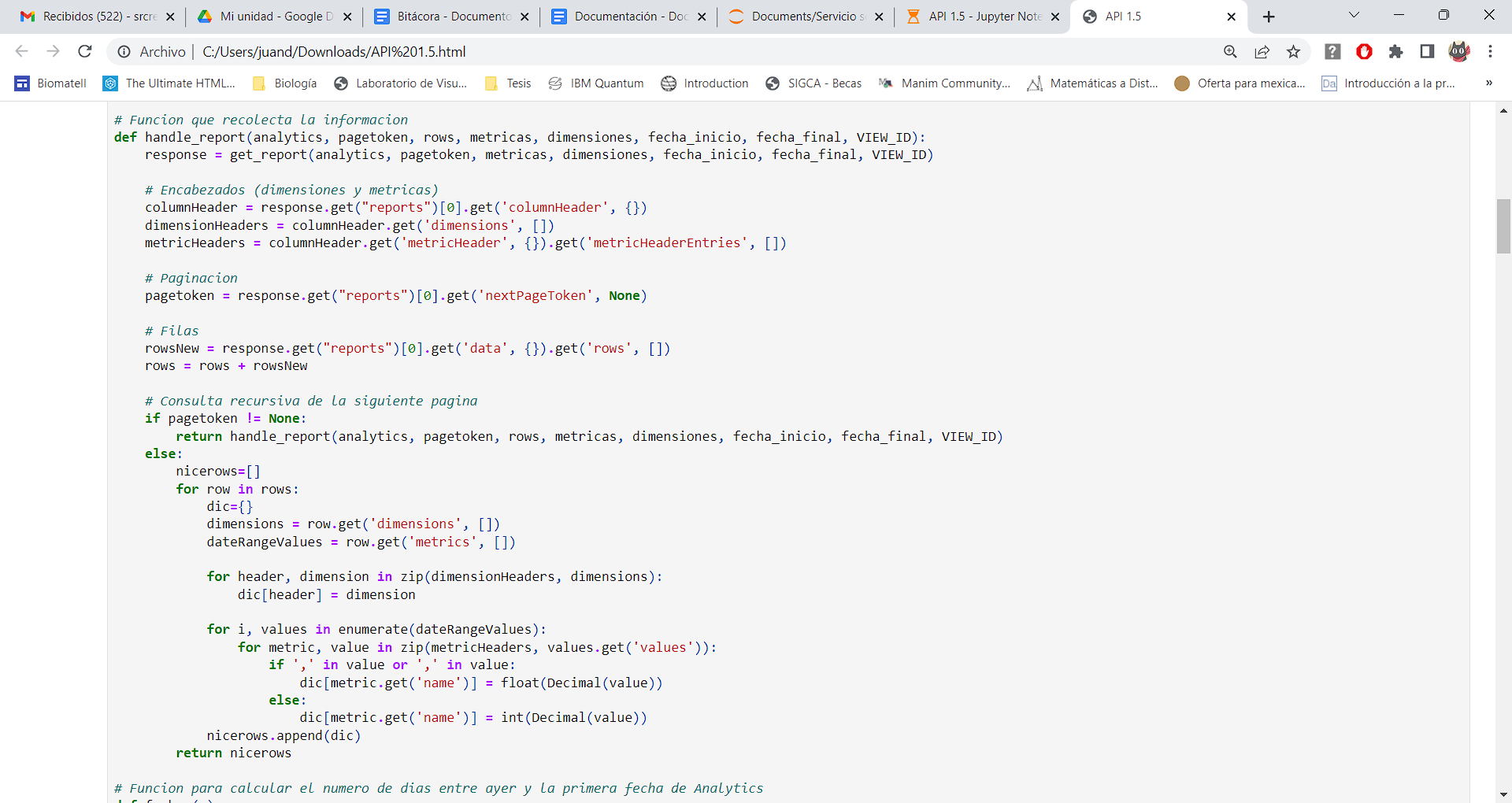


Esta función recibe como entrada los datos necesarios para establecer la conexión, así como los datos consultados (métricas, dimensiones, fechas e ID de la vista de cada Analytics).

Lo que hace es intentar establecer la conexión con el servidor mediante la API, para retraer la información con los datos consultados. En caso de que existan errores de conexión de parte del programa, tratará de manejarlos con la espera de 35 segundos (pues cada consulta fallida obliga a la API a no recibir más consultas en un periodo de 30 segundos) y después lo vuelve a intentar.

Como resultado, regresa la variable “a” que tiene el reporte en crudo.

### *handle\_report*

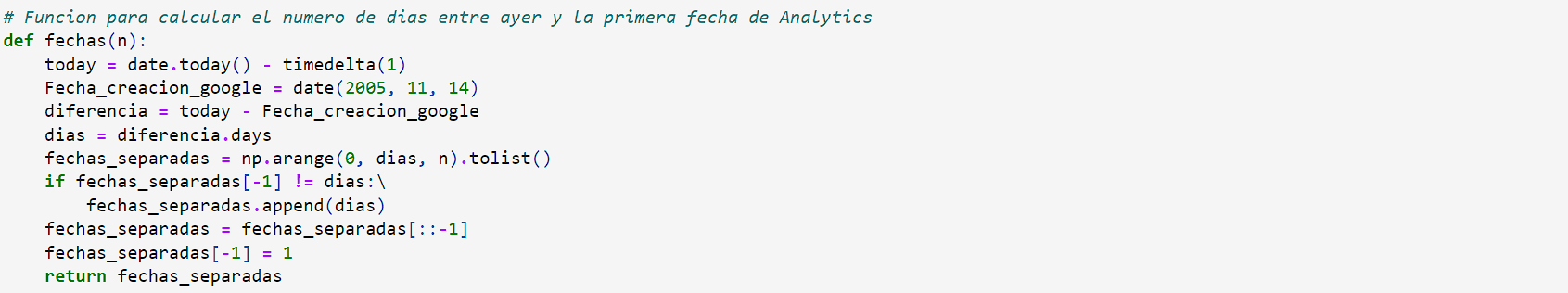


Es una función que sirve para recolectar la información generada por las consultas de Analytics. Primero usa la función *get\_report* para obtener el reporte, una vez que la información esté en el equipo local, lo que hace es trabajar la información cruda para devolvernos diccionarios de cada métrica y dimensión consultada, esto para poder tener la información acomodada de una manera más fácil.

En este proceso, se filtra el diccionario para obtener el verdadero tipo de datos (lo que evita, entre otras cosas, que los números sean número y no cadenas de carácteres). En caso de que no existan elementos, solo pasa la información vacía.

Por último, regresa el diccionario filtrado.

### *fechas*

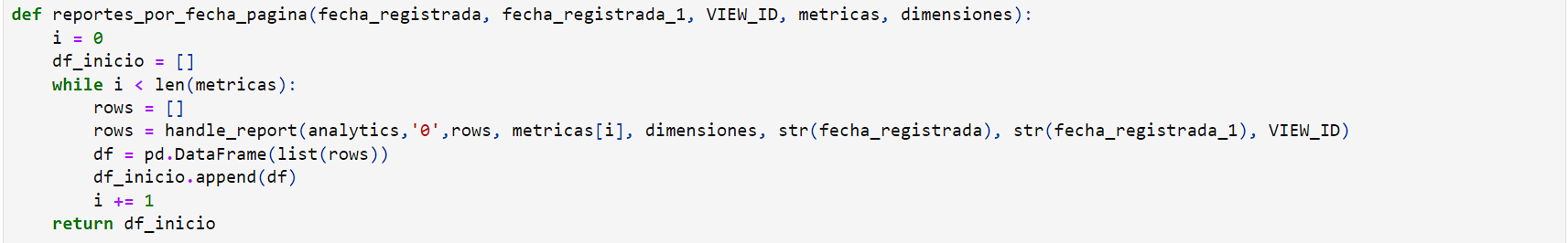


Esta función recibe como entrada un parámetro “n” que no es más que la diferencia entre periodo de fechas, es decir, si se quieren generar periodos de 20 días, solo habrá que pasarle dicho número.

Lo que hace la función es obtener el día actual para calcular el de ayer (ya que el día más cercano al presente que permite usar la API de Google es ayer), luego obtener la fecha de creación de Analytics V3 (esto porque cada proyecto pudo ser ingresado en distintas fechas, por lo que usaremos como día base el primer día que Analytics V3 vió la luz) y después divide la cantidad de días entre el periodo de días que consultamos, generando así una lista de las fechas separadas.

La función devuelve la lista de días separados por intervalos de longitud “n”.

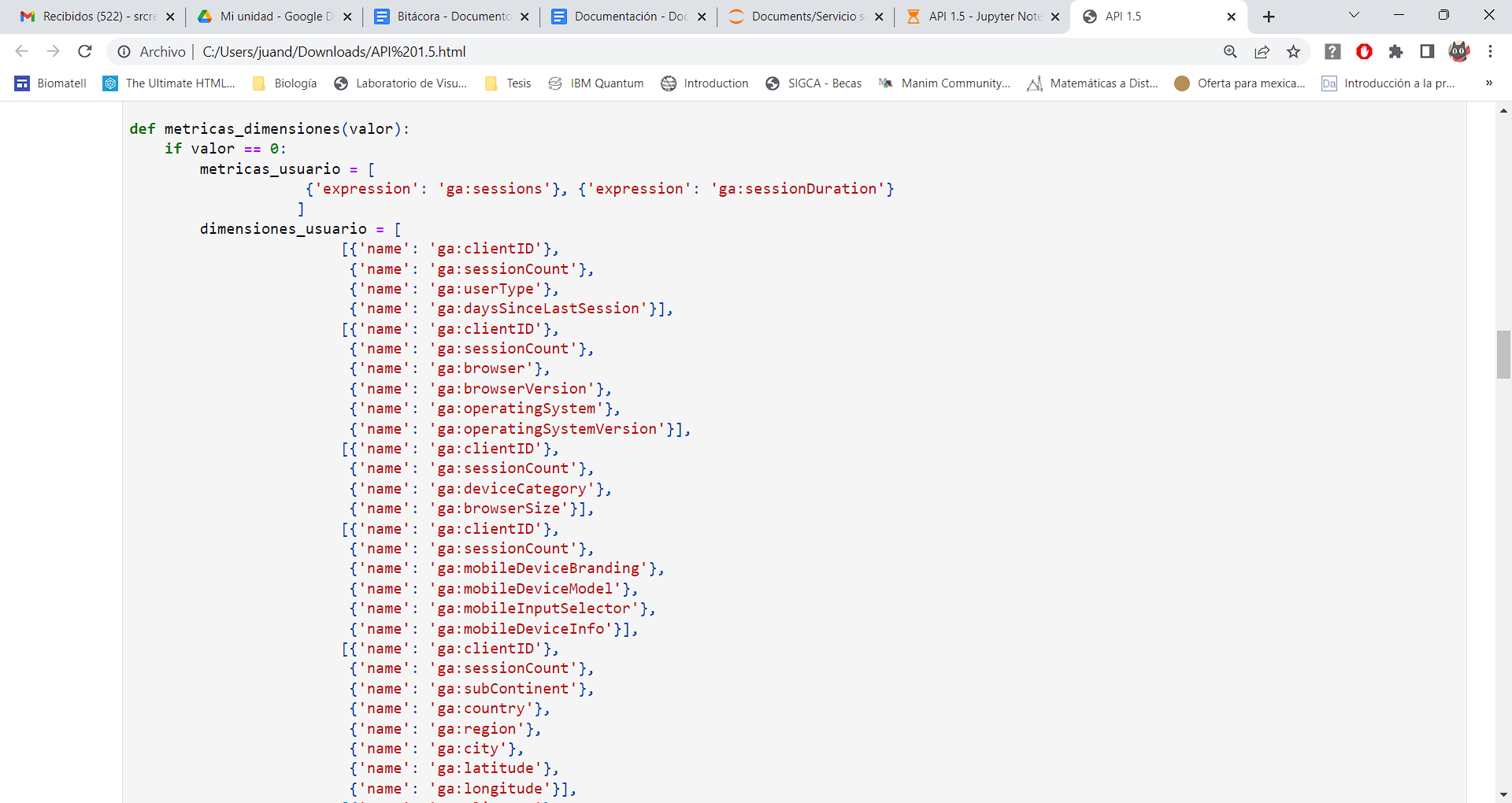
### *reportes\_por\_fecha\_pagina*

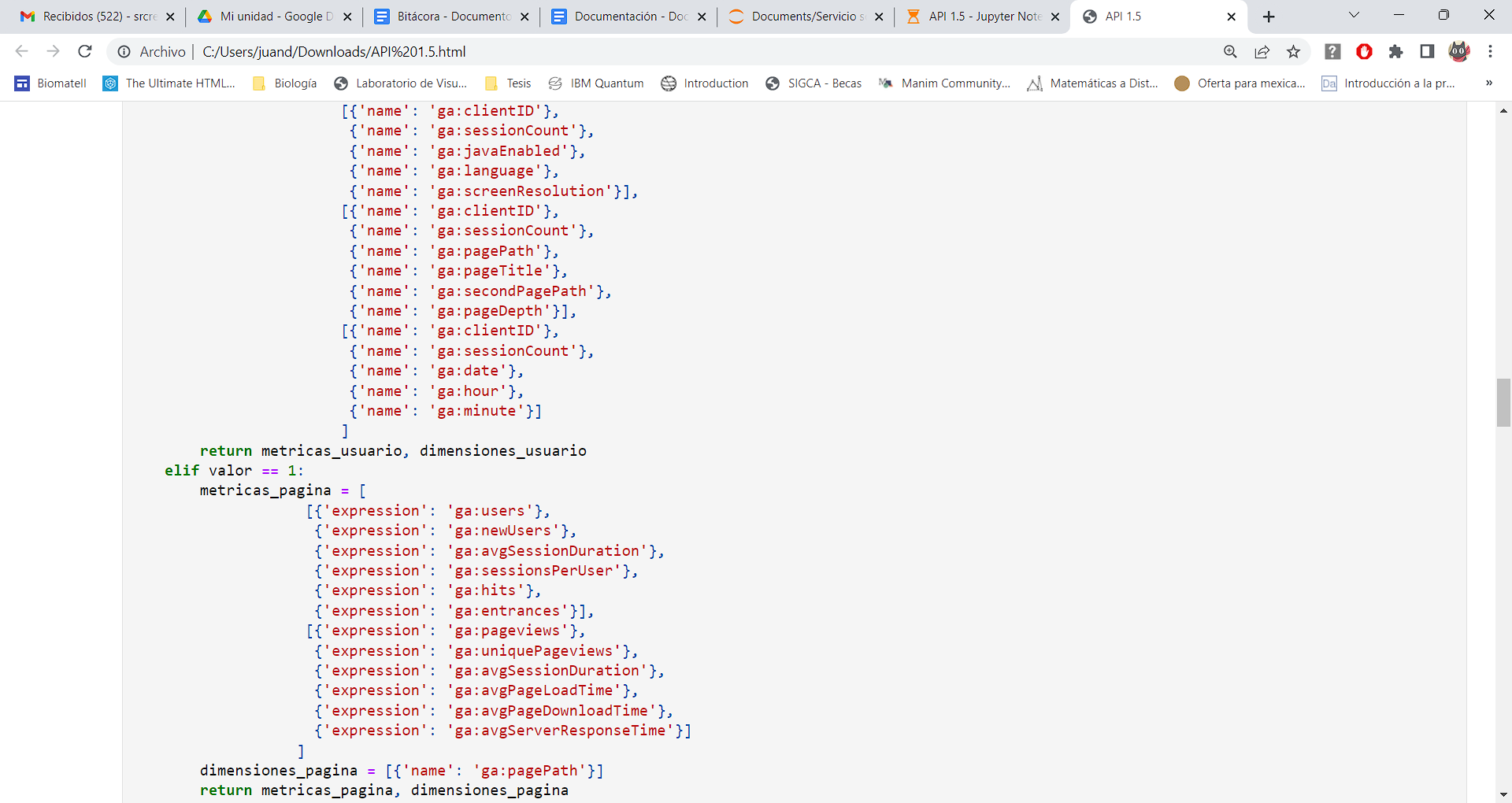


Esta función obtiene dos fechas distintas, así como las métricas, dimensiones e ID de la vista de Analytics, para generar las consultas necesarias dada las fechas otorgadas.

Devuelve la información consultada como un DataFrame de pandas.

### *metricas\_dimensiones*





Esta función recibe un valor binario y sirve para decidir entre qué métricas y dimensiones deben de ser otorgadas, teniendo 1 para los reportes por página y 0 para los reportes de usuario.

Esta función puede ser modificada quitando o añadiendo valores para tener reportes personalizados, siempre y cuando se respete el límite por dimensiones y métricas que Analytics establece, en caso contrario, siempre se pueden añadir más listas con dicha información.

Devuelve las listas de listas para ser usadas en las consultas.

### *reportes\_usuario*

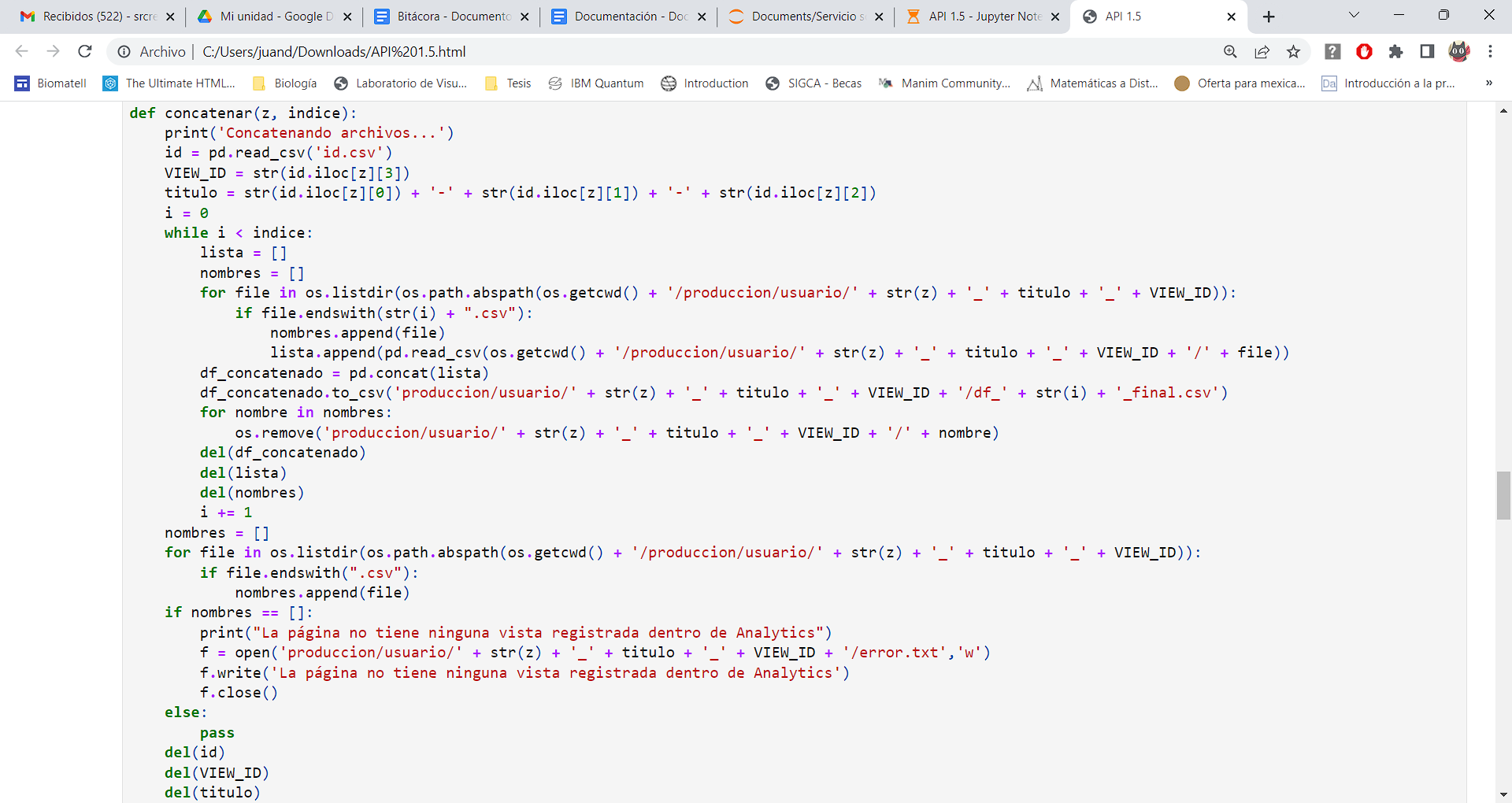


Recibe como parámetro un número z, que indica el ID de la página que debe de procesarse. Dicho ID se encontrará en el documento ID.csv, el cual se dejará en el anexo y es fundamental para el buen funcionamiento del código.

Lo que hace es generar los títulos de cada reporte para después hacer uso de la función *reportes\_por\_fecha\_1* y así guardar cada csv de cada reporte generado por cada intervalo de fecha. En caso de que no se generen vistas, ignorará ese periodo de fechas.

Regresa el índice de fechas, que no es más que el recuento de los intervalos de fecha que tienen al menos una fila por cada consulta generada.

### *concatenar*

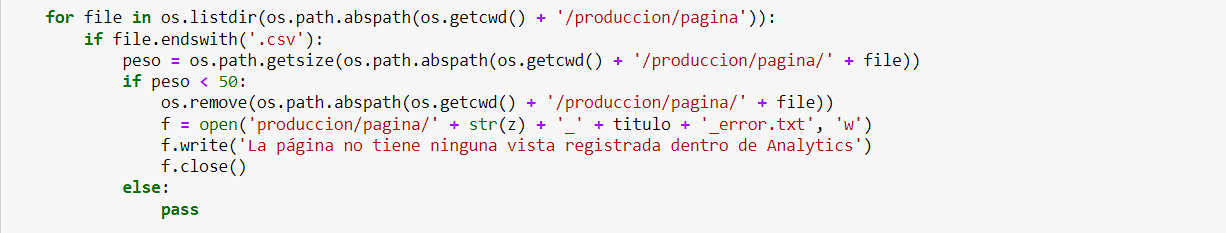


Esta función sirve para concatenar los archivos de la función anterior. Recibe como parámetros el índice así como el ID del archivo en curso.

Lo que hace es que, toma todos los archivos CSV que contienen las mismas columnas pero que fueron generados en diferentes fechas y con ello los une en un solo archivo CSV, borrando los pequeños y dejando como resultado los CSV concatenados. En caso de que no existan archivos (que significa que la vista no tuvo registros en Analytics) guardará un archivo TXT que muestra la razón por la cual no hay registros.

Esta función no retorna nada.

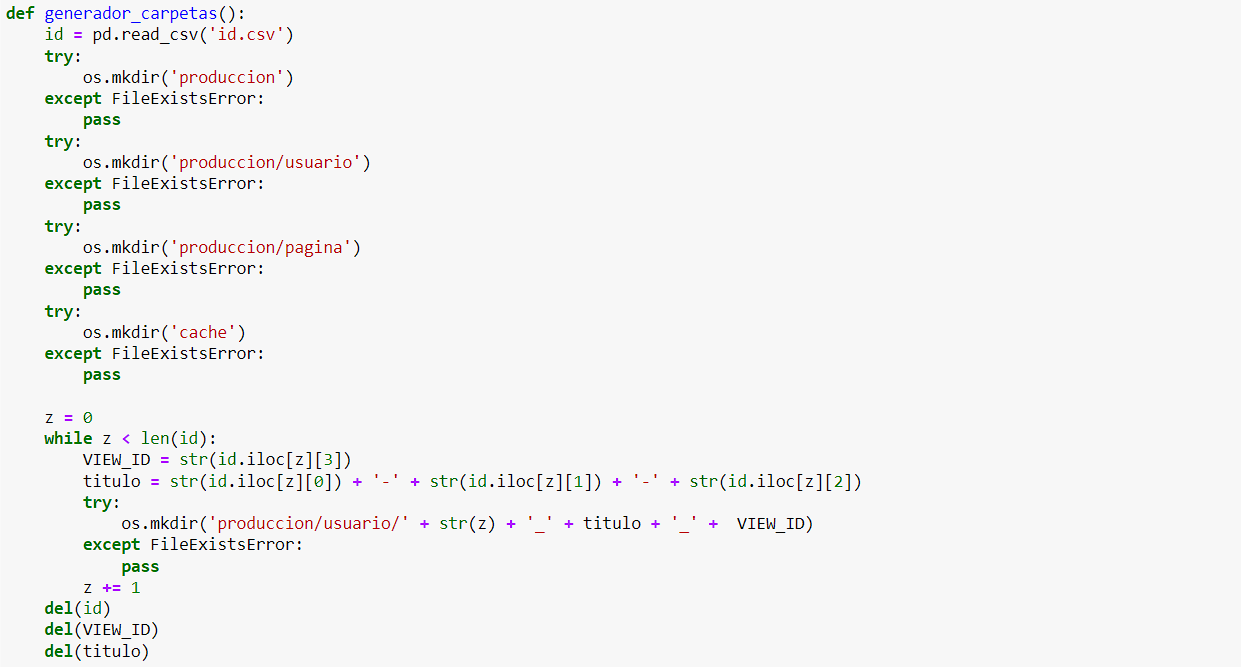
### *reportes\_pagina*



Esta función recibe como parámetro el ID de la vista en curso.

Trabaja de forma parecida a los reportes de usuario, solo que en vez de utilizar múltiples dimensiones, se utilizan múltiples métricas para retraer toda la información relacionada con las páginas web.

### *generardor\_carpetas*



Esta función sirve para generar las carpetas necesarias para separar la información, siguiendo el siguiente esquema:

Carpeta madre (depende de la ubicación del archivo .ipynb)

|--- Producción

| |--- Usuario

| | |--- ID\_nombreVistaPagina\_IDAnalytics (carpeta única por cada entrada)

| |

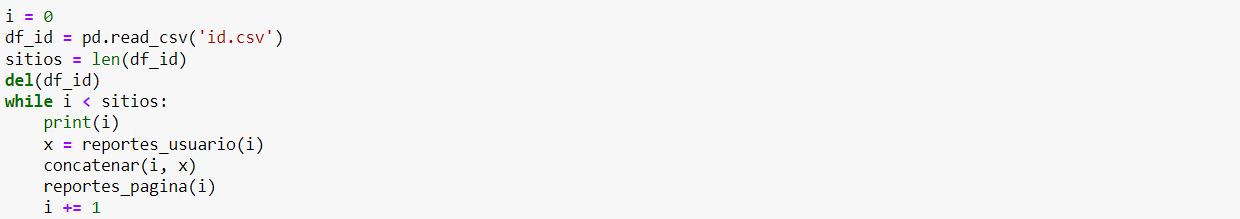
| |--- Página

|

|--- Cache

Los archivos productivos se encontrarán dentro de las carpetas integradas en usuario, o bien, por archivos csv dentro de la carpeta Página.

### *inicio del programa*



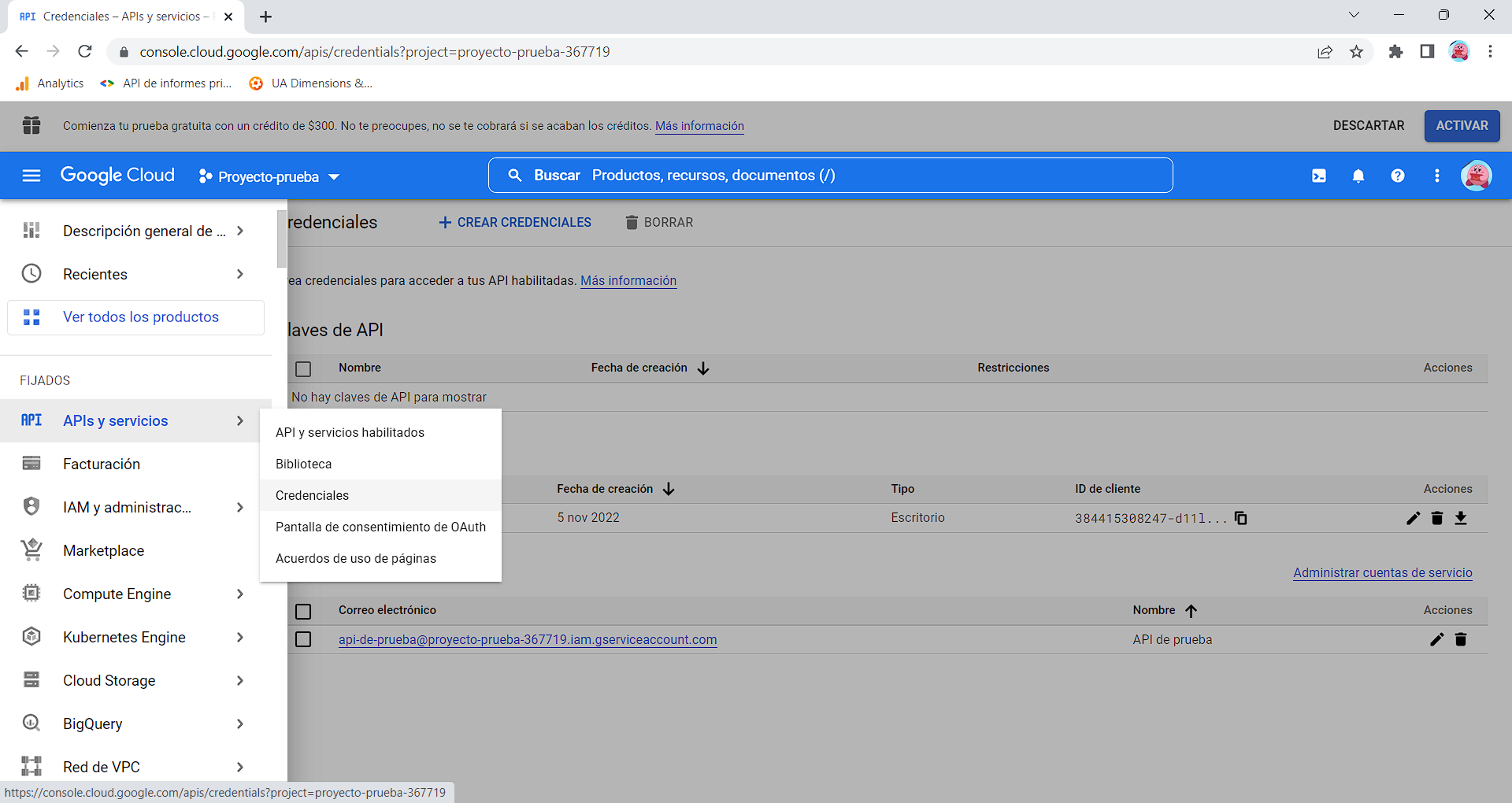
No es como tal una función, pero esta línea de código se encarga de ejecutar tres funciones (una para generar reportes por usuario, otra para concatenar estos mismos y la última para generar los reportes por página). En principio, de no presentarse ningún error, el código corre para generar dichos reportes por cada registro dentro del documento ID.csv

# **Anexo (API)**

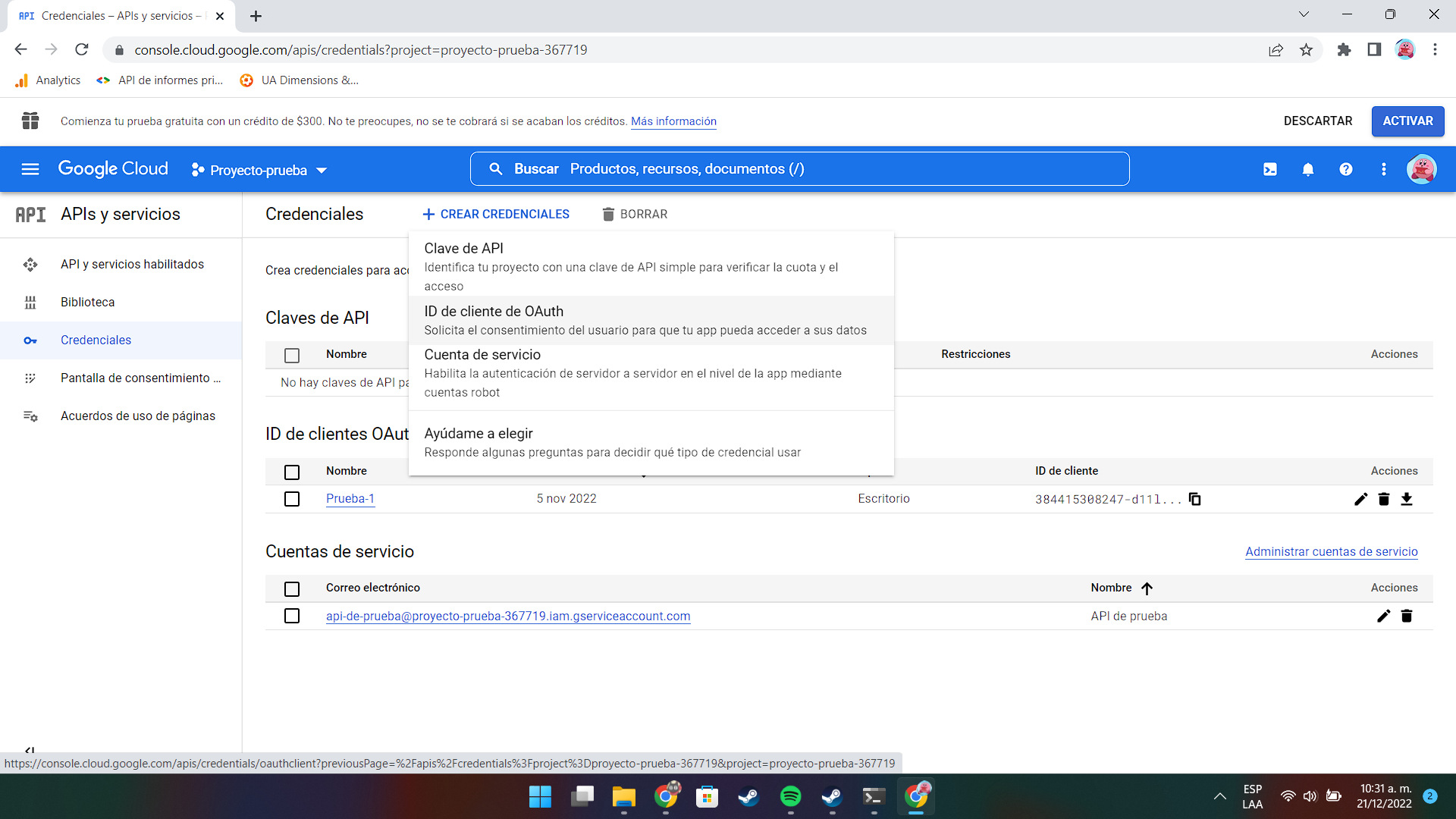
### *KEY\_FILE\_LOCATION*

Para crear la llave necesaria para acceder a las cuentas de Google Analytics es necesario crear una cuenta dentro de un entorno de Google Cloud; este paso no requiere pagar ningún tipo de servicio. Las instrucciones son:

* Entrar a <https://console.cloud.google.com/>
* De ser necesario, registrarse y crear un proyecto nuevo (el nombre no importa)
* Ingresar a APIs y Servicios > Credenciales



* Creamos una credencial (ID de cliente de OAuth)



* El nombre puede ser el que gustes, con las opciones predeterminadas. Se descarga un archivo JSON en el cual viene una cuenta (dicha cuenta tendrá que agregarse al Analytics con permisos de visualización) y se deberá de guardar dentro de la misma carpeta de ejecución del código.

### *Archivo ID*

El enlace de descarga de dicho archivo es:

<https://drive.google.com/file/d/1a_BuPc0LJqkeRNqE2M9JFECq6sJ7fe60/view?usp=share_link>

Se pueden agregar más filas al archivo en caso de que se requieran más vistas del Analytics.

### *Código*

El código se puede encontrar en:

<https://drive.google.com/file/d/1T9IuH5zAdOwt_cnRgSBgTFbZZoxzmjXz/view?usp=share_link>

Se podrá descargar para ejecutar en un entorno privado, o bien, se puede configurar el Google Colab para tener la ejecución en la nube de Google. (Poco recomendado).

# **Errores comunes (API)**

En principio, el código captura los errores comunes del programa mediante las rutinas *try-catch*, pero esto no implica que esté exento de errores que quedan fuera del alcance del mismo, teniendo el más común:

Exception: *time out*

Este error surge como consecuencia directa de un error propio de la API de Analytics, suele ocurrir cuando el servidor no recibe respuesta (a nivel Google para la API) y por tanto la API no nos otorga respuesta alguna (de API para Python). El error se soluciona con reescribir el valor i, asignándole el ID del documento en el que el programa dejó de trabajar, esto para continuar con el trabajo.

# **Introducción (Valores Únicos)**

Este nuevo programa tiene el objetivo de separar los valores únicos de columnas que tienen valores repetidos, para así retomarlos y evitar la repetición de información. El programa no usa ninguna función, pues todas las acciones se repiten con el uso de ciclos *while* (ya que el propósito es obtener los valores únicos por todos los documentos, no únicos valores por cada documento).

# **Anexo (Valores Únicos)**

### *Código*

El código se puede descargar del siguiente enlace:

<https://drive.google.com/file/d/1ZCadjDeYPG3aW0C9ZGOHbjFrALmoaaQn/view?usp=share_link>

El resultado son dos archivos csv llamados “unique\_pagina” y “unique\_usuario” que se guardan dentro de la carpeta de producción. Estos archivos contienen los valores únicos de las siguientes columnas:



Y sirven para cualquier documento, pues todos los valores únicos se guardan dentro del mismo. Estos archivos servirán para la creación de la base de datos dentro de PostgreSQL, indexando dichos valores y reduciendo la recurrencia de carácteres dentro de las tablas, obteniendo así una mayor velocidad de procesamiento, así como un ahorro de espacio de almacenamiento.