**Python API Functions**

# Cómo crear una API en Python

Como científico o ingenieros de datos, muchas veces tenemos que compartir nuestro trabajo para que cualquier otra persona de la empresa pueda utilizar los procesos o modelos que hemos creado. Claramente, compartir un script no es una opción, ya que todo el mundo necesitaría tener esos mismos programas que tú. Aquí es cuando entran en juego las APIs y hoy.

## Conceptos básicos sobre APIs

Una API (**Application Programming Interface**) permite que dos sistemas informáticos interactúen entre sí. Por ejemplo, si creamos una automatización que genere un informe y lo mande por email, el envío de ese email no se hace de forma manual, lo hará el propio script. Para ello, Python (o el lenguaje que usemos), deberá pedirle a Gmail que mande ese correo, con ese informe adjunto a ciertas personas. La forma de hacerlo es mediante una API, en este caso la API de Gmail.

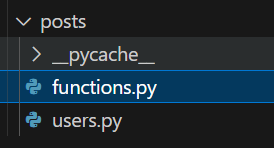
Bien, ahora que sabes lo que es una API, veamos cuáles son las partes principales de una API:

* **Protocolo de transferencia HTTP**: es la forma principal de comunicar información en la web. Existen diferentes métodos, cada uno de ellos utilizados para diferentes cuestiones:
  + **GET**: este método permite obtener información de la base de datos o de un proceso.
  + **POST**: permite mandar información, ya sea para añadir información a una base de datos o para pasar el input de un modelo de machine **learning**, por ejemplo.
  + **PUT**: permite actualizar información. Generalmente se usa para gestionar información en base de datos.
  + **DELETE**: este método se utiliza para eliminar información de la base de datos.
* **Url**: es la dirección en la que podremos encontrar a nuestra API. Básicamente esta URL constará de tres partes:
  + **Protocolo**: como toda dirección, puede ser [http://](NULL) o [https://](NULL)
  + **Dominio**: el host en el que está alojado, que va desde el protocolo hasta el final del .es o .com, o la terminación que sea.
  + **Endpoint**: al igual que una web tiene varias páginas (/blog), (/legal), una misma API puede incluir varios puntos y que cada uno haga cosas diferentes. Al crear nuestra API en Python nosotros indicaremos los **endpoints**, por lo que debemos asegurarnos de que cada **enpoint** sea representativo de lo que haga la API que está por detrás.

Importar Librerías

* pip install fastapi => Permite crear las APIs
* pip install uvicorn => Levantar la APP en un IP y puerto definido

Creamos dos carpetas de trabajo



* functions.py:
  + fastapi para manejo de API (restfull).
  + uvicorn para desplegar nuestro servidor de API (<http://localhost:8585>)
  + Importar el route users (Servicios API de usuarios)

import uvicorn

from fastapi import FastAPI

from users import router

app = FastAPI()

@app.get(“/helloWorld”)

def helloWorld():

  return {“Hola desde Python!”}

app.include\_router(router)

if \_\_name\_\_ == “\_\_main\_\_”:

    uvicorn.run(app, port=8585)

* users.py
  + Apis de Usuarios

from fastapi import APIRouter

# Crear un router con APIRouter

router = APIRouter()

# Definir la ruta con el decorador @router.get

@router.get("/user")

def create\_user():

    return [{"username": "Foo"}, {"username": "Bar"}]

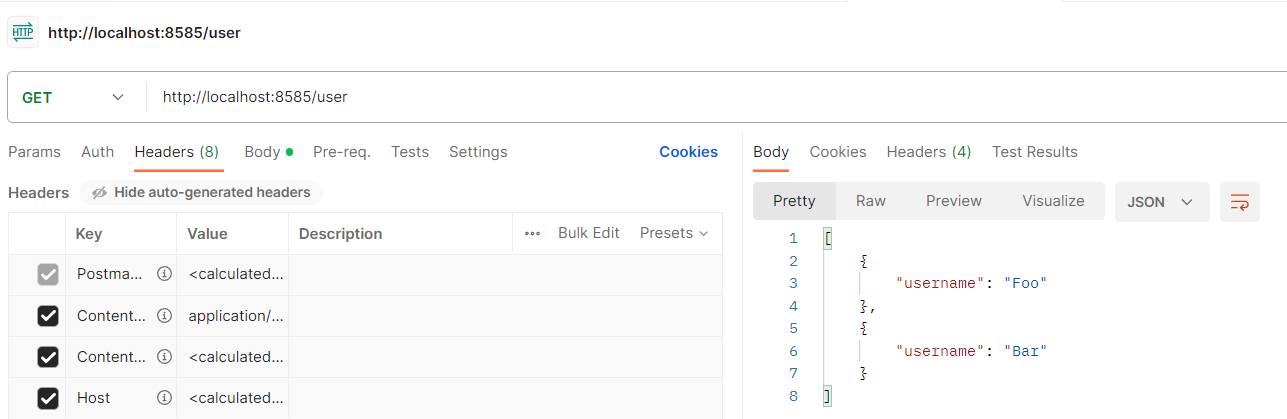
# Ruta para agregar un elemento usando POST

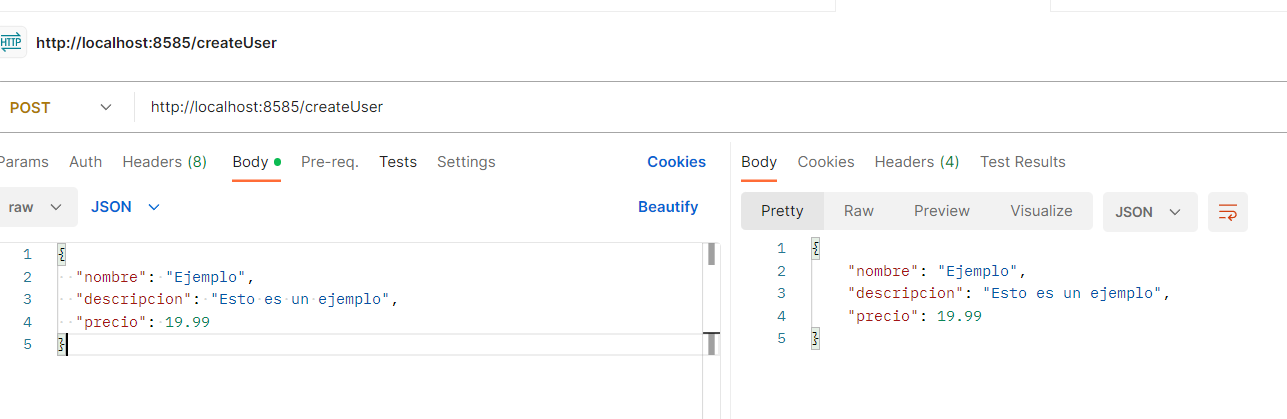
@router.post("/createUser")

def create\_user(body: dict):

    return body

* Levanto el servidor



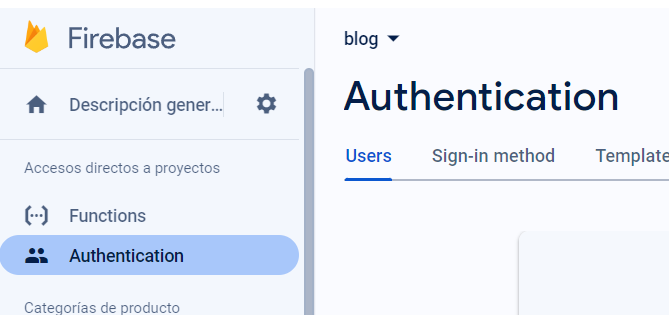


* En firebase creo un proyecto

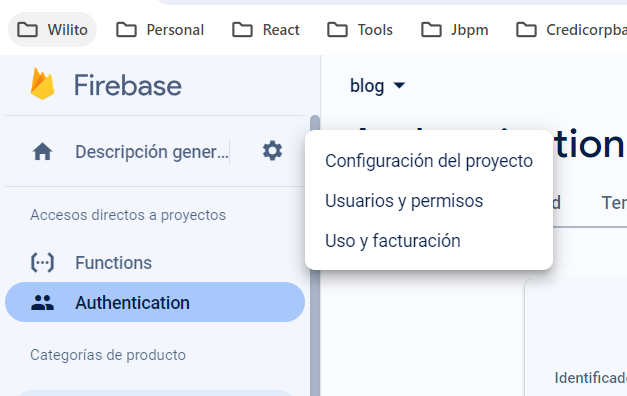


* Agrego los servicios de Functions (se requiere cuenta paga de firebase) y Authentication

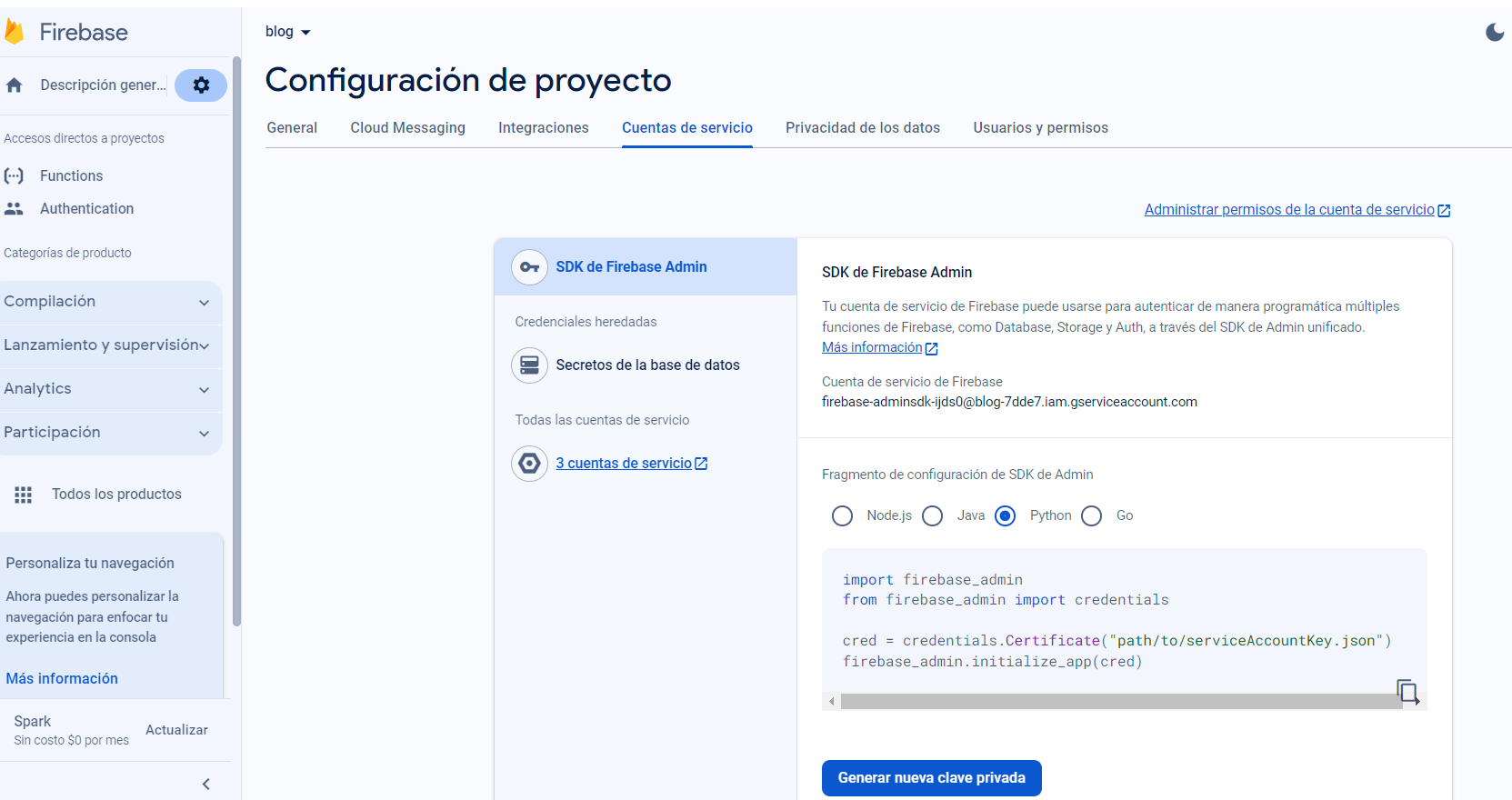
Si quiere puede utilizar Pyhton, Java, Node, etc cualquier backend para crear funciones



En la configuración del proyecto -> Descripción general del proyecto tomo los datos de la cuenta



Tomo los datos de la cuenta para Python, en mi caso.



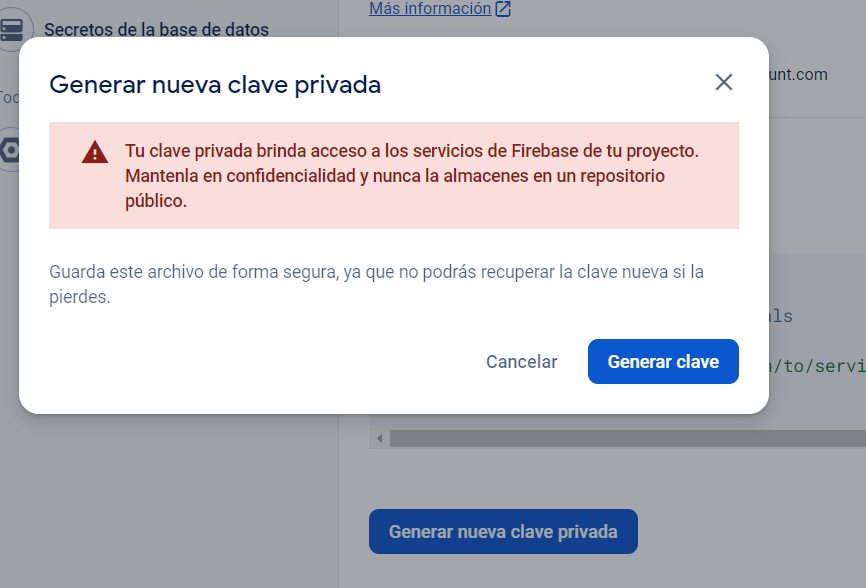
import firebase\_admin

from firebase\_admin import credentials

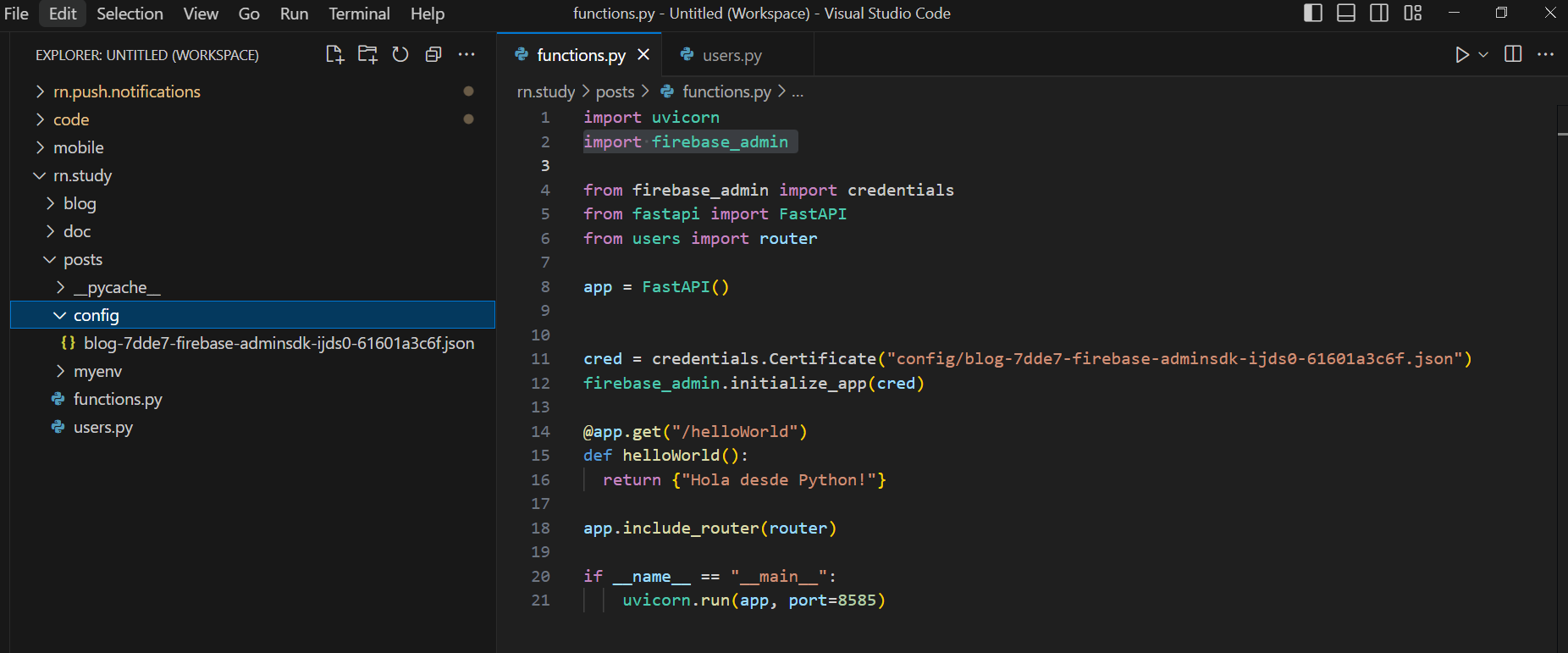
cred = credentials.Certificate("path/to/serviceAccountKey.json")

firebase\_admin.initialize\_app(cred)

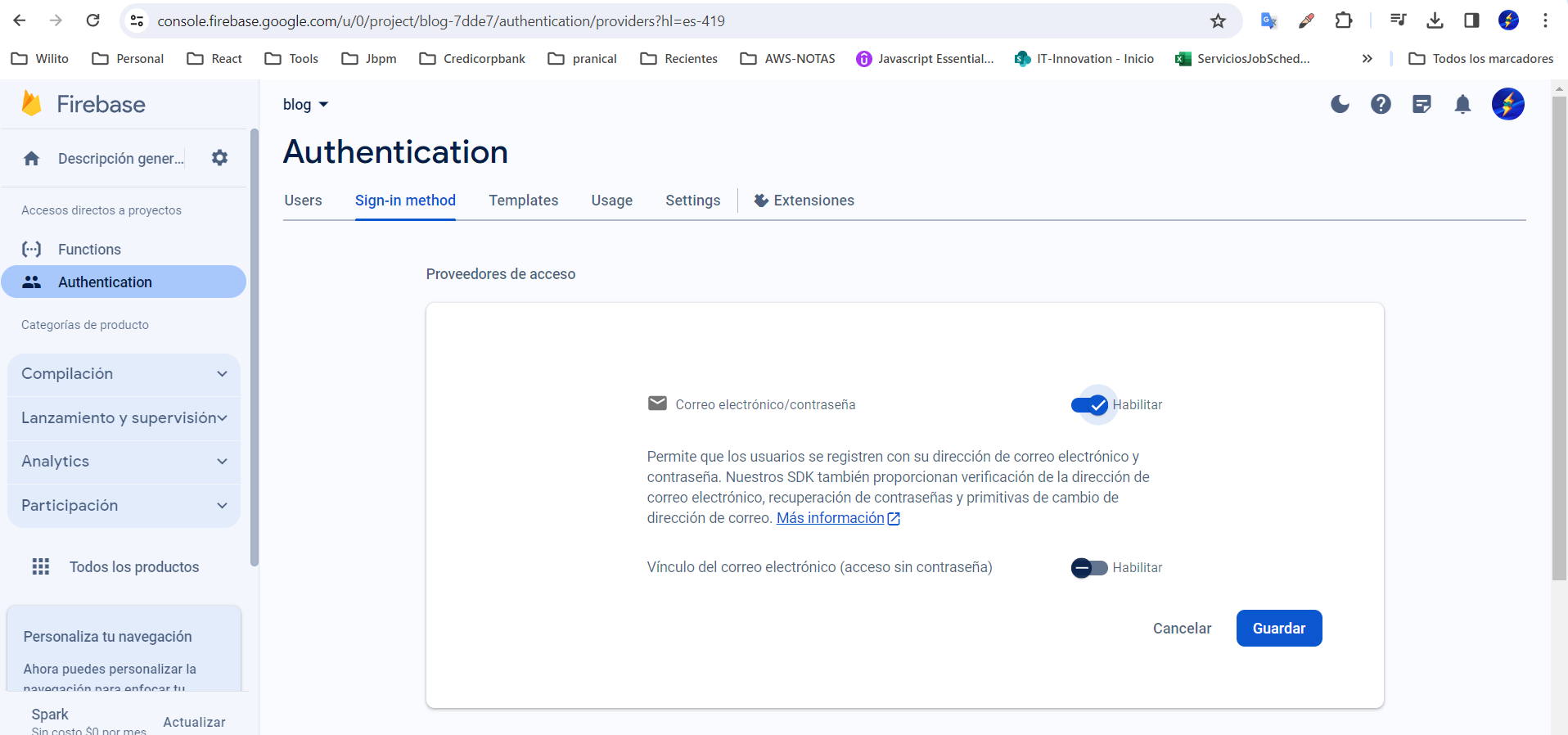
* Importamos la libreria: $> pip install firebase-admin
* Si da error de permisos utilizar: $> pip install --user firebase\_admin
* Crear una llave privada que nos proporciona firebase para tener acceso al mismo de manera segura al servicio de certificación

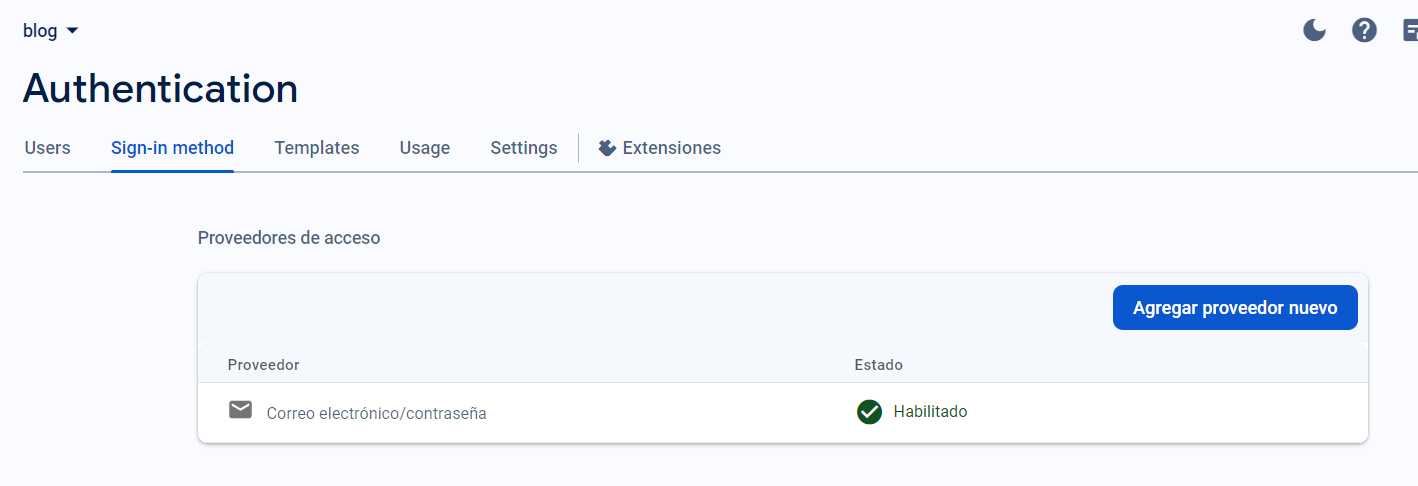


* El archivo generado se agrega a mi carpeta config.
* Luego agrego el código que me proporciona firebase a mi .py



* En ***Autentication*** ***Firebase*** debemos habilitar el uso de correo electrónico y contraseña. Esto nos permite a utilizar Firebase para autenticar a usuarios por correo / contraseña.





* Administrar usuarios en Firebase

<https://firebase.google.com/docs/auth/web/manage-users?hl=es>



* Crear un usuario

