Tarea de cambiar el estado de un pedido.

CocinaApp

Veronica Cenoz Mostert y Jorge Camacho Ochoa

Indice:

1.- Instalación de Python.....**3**

2.- Creación de Ficheros Python......**5**

3.- Poner una certificación para la Base de Datos......**13**

4.- Poner una certificación para la el envío de emails...**14**

5.- Empaquetado de la tarea.....**20**

6.- Programar las Tareas.....**22**

Notas:

Para conectar con el servidor: Conexion_Servidor.pdf.

Para la transferencia de archivos usar WinSCP (desde Windows) o cp desde Linux o Mac.

Para la conexión a la máquina virtual por terminal usar Putty.

1.- Instalación de Python.

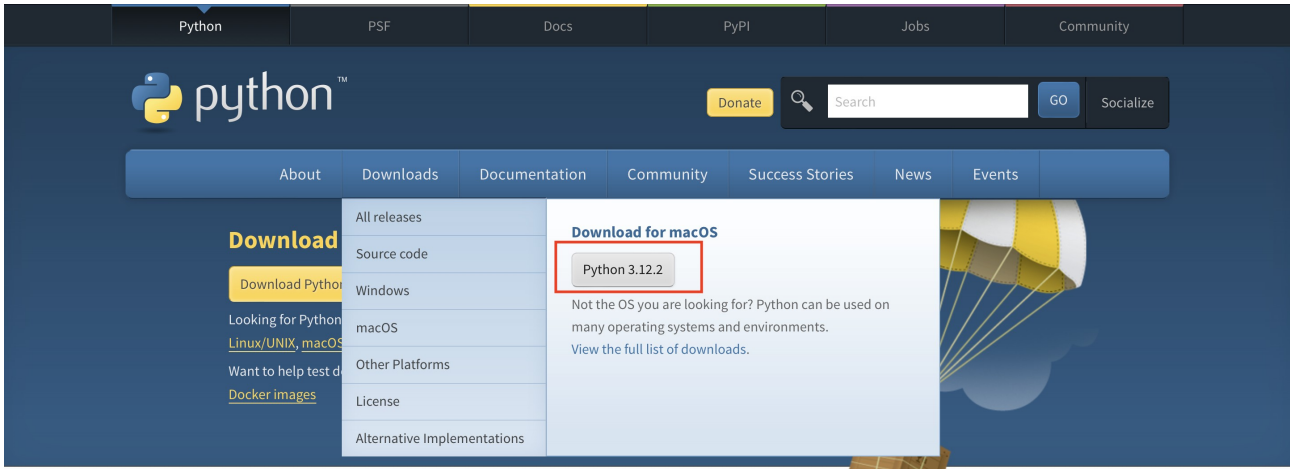
Todos los días a cada hora se ejecutará una tarea que compara que un pedido que haya superado más de una hora, le cambiará a un pedido el campo editable que siempre empezará con true a false y se le enviará un email al usuario de que ese pedido ha sido enviado a preparar y ya no se puede editar y/o eliminar y otro email al usuario de cocina para tener un registro.

Para que esta tarea pueda realizarse, en el servidor, siempre tiene que estar encendido y se ejecutará cada minuto.

Tiene que tener el Python 3.x instalado. Se puede instalar en un servidor Linux.

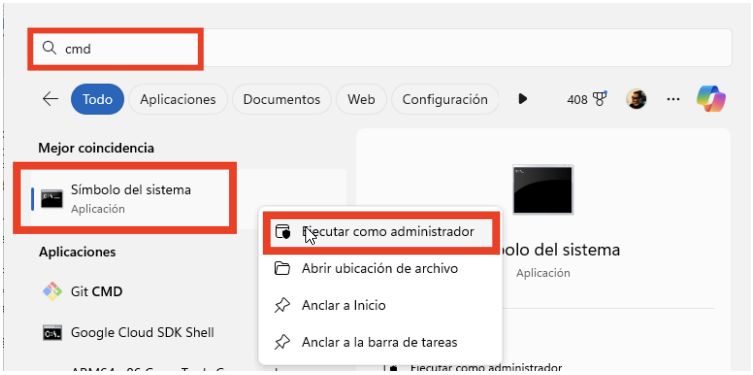
Para Windows:

Abrimos Pyton.org y nos descargamos la última versión para windows.



El paquete de firebase-admin instalado.

Para ello abrimos la consola, escribimos en la barra de búsqueda “cmd” y lo ejecutamos como administrador.



Primero vemos que tiene instalado Python → **py --version**

y después → **py -m pip install firebase-admin**

Para crear el ejecutable → **py -m pip install pyinstaller**

Para Linux (Ubuntu):

Abrimos el terminal y ejecutamos:

sudo apt search python3

para buscar el paquete instalador más reciente de python, en este caso es Python3.11, lo instalamos:

sudo apt install python3.11 -y

después vamos a instalar el paquete instalador de python, pip:

sudo apt install python3-pip

una vez instalados necesitaremos que instalar la librería de firebase para conectar con python:

sudo pip install firebase-admin

Para crear los ejecutables:

sudo pip install pyinstaller

Nota: para Debian 12, su instalación es algo diferente al de Ubuntu, lo dejo explicado en el apartado 5 en el empaquetado de la Tarea. Al estar la tarea alojada en un servidor Debian 12 sin entorno de escritorio (solo CLI) el empaquetado debemos realizarlo obligatoriamente en el mismo servidor u otro servidor Debian 12 similar y luego pasarlo a este.

2.- Creación de Ficheros Python.

Necesitamos 3 ficheros para crear el ejecutable:

- **update_task_min.py** → el archivo principal, conectará con la base de datos, comprobará en los pedidos con el campo editable = true si ha pasado más de una hora, si ha pasado más de una hora, cambiará el campo editable = false en la base de datos de firebase, por lo que el usuario ya no podrá hacer cambios en el pedido y enviará un email al usuario con la información y otro a los de cocina (así hay un registro de los pedidos). Si en algún caso fallase el envío del email al usuario, se enviará un emaila los de cocina reportando el error de que no se ha podido enviar.

- **email_sender.py** → archivo que creará el mensaje a enviar a los usuarios.

- **email_sender_cocinapp** → archivo que creará el mensaje para enviar a los de cocina.

En Windows:

Ahora hay que crear tres Archivos llamados **update_task_min.py** y **email_sender.py y email_sender_cocinapp.py** los creamos con el block de notas, pero lo guardamos con la extensión .py (acordarse de cambiar guardar como .txt, a todos los archivos).

En Linux:

En el terminal

→ **sudo nano update_task_min.py**

→ **sudo nano email_sender.py**

→ **sudo nano email_sender_cocinapp.py**

update_task :

```
# importa la libreria para la conexión de firebase
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime, timedelta

#importamos la clase email_sender y email_sender_cocinapp
from email_sender import EmailSender
```

```
from email_sender_cocinapp import EmailSenderCocina

#importamos subproceso
import os

# Configura las credenciales de Firebase
cred =
credentials.Certificate(os.path.join(os.path.dirname(os.path.abspa
th(__file__)), "coc[redacted].json"))
firebase_admin.initialize_app(cred, {'databaseURL':
'https://cocin[redacted]
west1.firebasedatabase.app'})

# Obtén una referencia a la base de datos
ref = db.reference('pedidos')

# Obtén la hora actual
hora_actual = datetime.now()

# Obtén todos los pedidos
pedidos = ref.get()

#ponemos el email y las credenciales de cocinApp
email = "[redacted]@gmail.com"
password = "[redacted]"

# Crea una instancia de EmailSender y EmailSenderCocina
email_sender = EmailSender(email, password)
email_sender_cocinapp = EmailSenderCocina(email, password)

# Itera sobre cada pedido
for pedido_id, pedido in pedidos.items():
```

```
    #print(f"Procesando pedido {pedido_id}")

    if pedido['editable']:
        # Convierte la fecha_pedido a formato de datetime
        fecha_pedido = datetime.strptime(pedido['fecha_pedido'],
'%d-%m-%Y %H:%M:%S')

        #print(f"Fecha del pedido: {fecha_pedido}")
        #print(f"Hora actual: {hora_actual}")

        # Comprueba si ha pasado más de una hora desde la
fecha_pedido
        if hora_actual - fecha_pedido > timedelta(hours=1):
            # Actualiza el campo editable a False
            ref.child(pedido_id).update({'editable': False})
            #print(f"Pedido {pedido_id}: editable actualizado a
False.")

            # Enviamos Email
            # Dirección de correo electrónico del destinatario
            destinatario = pedido['usuario']
            # Obtén la información del destinatario desde la base
de datos
            info_destinatario =
db.reference('usuarios').child(destinatario).get()

            try:
                # Intenta enviar un correo electrónico al usuario

email_sender.send_email(info_destinatario["email"],
info_destinatario["nombre"], pedido_id, pedido["detalles"],
pedido["fecha_entrega"],
pedido["fecha_pedido"],pedido["precio_total"],
pedido["comentarios"])
```

```
            except Exception as e:
                # Si hay un error al enviar el correo al usuario,
envía un correo de error a la dirección alternativa

email_sender_cocinapp.send_error_email(info_destinatario["email"],
hora_actual, e)


            # Envía un correo electrónico a la cocina

email_sender_cocinapp.send_email(info_destinatario["email"],
info_destinatario["nombre"], info_destinatario["apellidos"],
info_destinatario["departamento"], info_destinatario["telefono"],
pedido_id, pedido["detalles"], pedido["fecha_entrega"],
pedido["fecha_pedido"],pedido["precio_total"],
pedido["comentarios"])
        else:
            pass
            #print(f"Pedido {pedido_id}: No se actualizó, no ha
pasado suficiente tiempo.")
    else:
        pass
        #print(f"Pedido {pedido_id}: No es editable.")
#input("Presiona Enter para salir...")
```

email_sender.py :

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText


class EmailSender:
    def __init__(self, sender_email, sender_password):
        self.sender_email = sender_email
        self.sender_password = sender_password
```

```
    def send_email(self, recipient_email, recipient_name,
pedido_id, pedido_detalles, pedido_fecha_entrega,
pedido_fecha_pedido, pedido_precio_total, pedido_comentarios):


        detalles = detalles = '\n\t\t'.join([f"Racion:
{detalle['racion']}, cantidad: {detalle['cantidad']}, precio:
{detalle['precio']} €" for detalle in pedido_detalles])
        smtp_server = "smtp.gmail.com"
        smtp_port = 587


        msg = MIMEMultipart()
        msg['From'] = self.sender_email
        msg['To'] = recipient_email
        msg['Subject'] = "Pedido Realizado"


        mensaje = f"""


        Hola {recipient_name},


        Le informamos de que su pedido:


            ID pedido: {pedido_id}


            Detalles:
            \t{detalles}


            Comentarios: {pedido_comentarios}


            Precio total: {pedido_precio_total} €


            Fecha de entrega: {pedido_fecha_entrega}
```

```
        ha sido enviado el {pedido_fecha_pedido}.

        Ya no puede realizar ningun cambio en su pedido.

        ¡Gracias por tu compra!

        Si necesita contactar con nosotros:
            - [redacted] -
            - {self.sender_email} -

        Saludos,



        Equipo de CocinaApp

        [redacted]

                    """
        msg.attach(MIMEText(mensaje, 'plain'))

        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(self.sender_email, self.sender_password)
            texto = msg.as_string()
            server.sendmail(self.sender_email, recipient_email,
texto)
            #print("Correo electrónico enviado correctamente.")
```

email_sender_cocinapp.py :

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

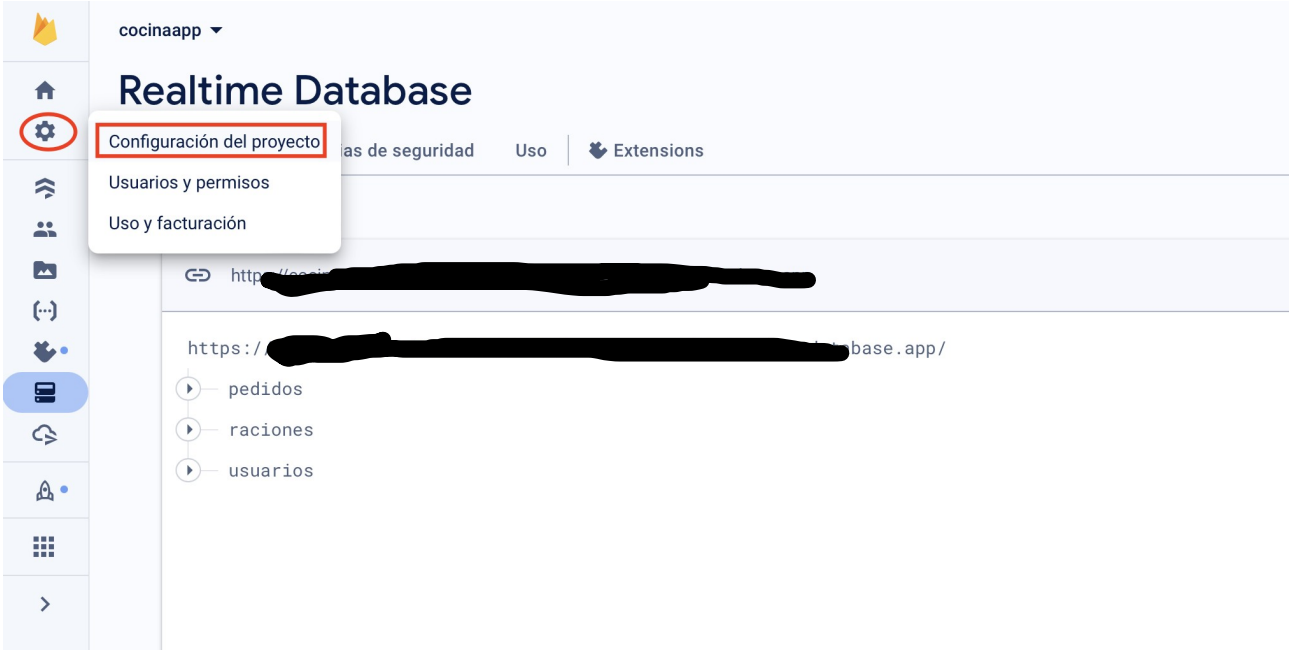
```
class EmailSenderCocina:
    def __init__(self, sender_email, sender_password):
        self.sender_email = sender_email
        self.sender_password = sender_password

    def send_email(self, recipient_email, recipient_name,
recipent_apellidos, recipient_departamento, recipient_telefono,
pedido_id, pedido_detalles, pedido_fecha_entrega,
pedido_fecha_pedido, pedido_precio_total, pedido_comentarios):

        detalles = detalles = '\n\t\t'.join([f"Racion:
{detalle['racion']}, cantidad: {detalle['cantidad']}, precio:
{detalle['precio']} €" for detalle in pedido_detalles])
        smtp_server = "smtp.gmail.com"
        smtp_port = 587

        msg = MIMEMultipart()
        msg['From'] = self.sender_email
        msg['To'] = self.sender_email
        msg['Subject'] = f"""Nuevo pedido {pedido_id}"""

        mensaje = f"""

        Nuevo pedido de {recipient_name} {recipent_apellidos} a
fecha del {pedido_fecha_pedido},

            ID pedido: {pedido_id}

            Fecha entrega: {pedido_fecha_entrega}

            Detalles:
            \t{detalles}

            Comentarios: {pedido_comentarios}

            Precio total: {pedido_precio_total} €

        ------------------------------------------------------------

        Datos Contacto :

            Nombre: {recipient_name}

            Apellidos: {recipent_apellidos}

            Email: {recipient_email}

            Departamento: {recipient_departamento}

            Teléfono: {recipient_telefono}
```

```
                """
        msg.attach(MIMEText(mensaje, 'plain'))

        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(self.sender_email, self.sender_password)
            texto = msg.as_string()
            server.sendmail(self.sender_email, self.sender_email,
texto)
            #print("Correo electrónico enviado correctamente.")


    def send_error_email(self, recipient_email, fecha_envio,
exception):
        smtp_server = "smtp.gmail.com"
        smtp_port = 587

        msg = MIMEMultipart()
        msg['From'] = self.sender_email
        msg['To'] = self.sender_email
        msg['Subject'] = "Error al enviar correo electrónico"

        mensaje = f"""

        {fecha_envio}

        Se produjo un error al enviar un correo electrónico a:

        {recipient_email}

        Detalles del error:
        {exception}
        """

        msg.attach(MIMEText(mensaje, 'plain'))

        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(self.sender_email, self.sender_password)
            texto = msg.as_string()
            server.sendmail(self.sender_email, self.sender_email,
texto)
```

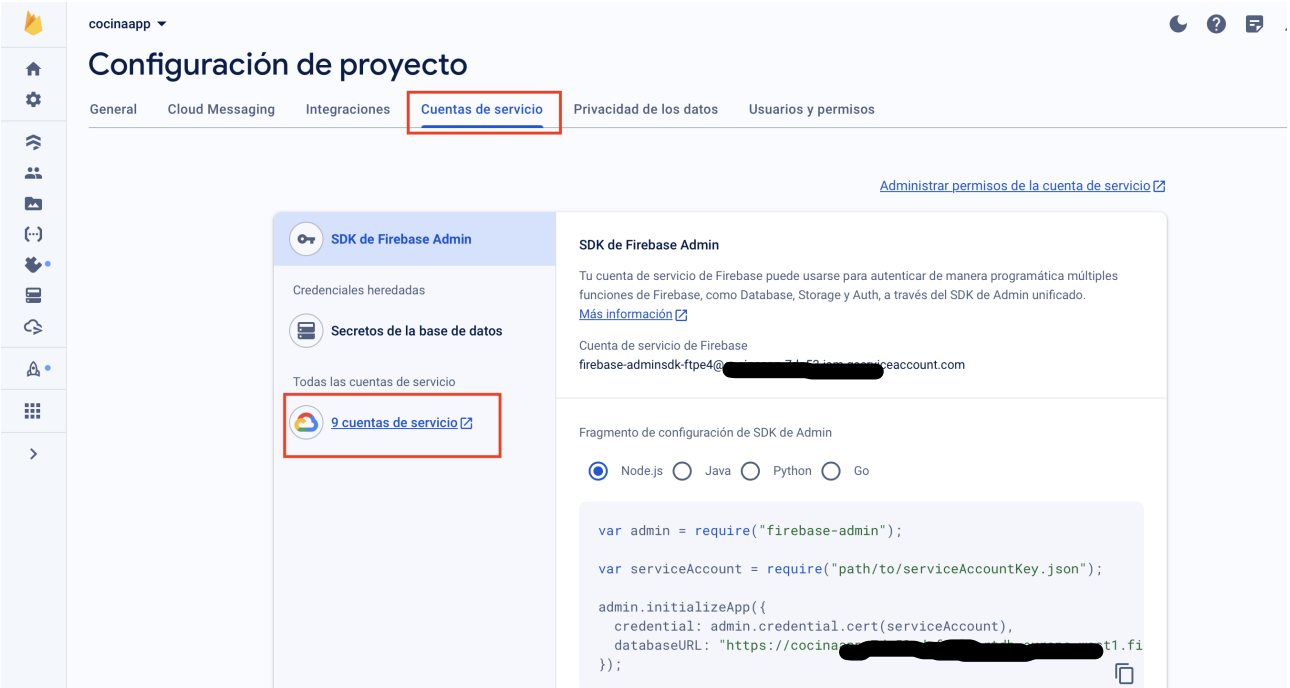
Nota: En Manual Desarrollador en Tarea.zip están estos tres archivos.

3.- Poner una certificación para la Base de Datos.

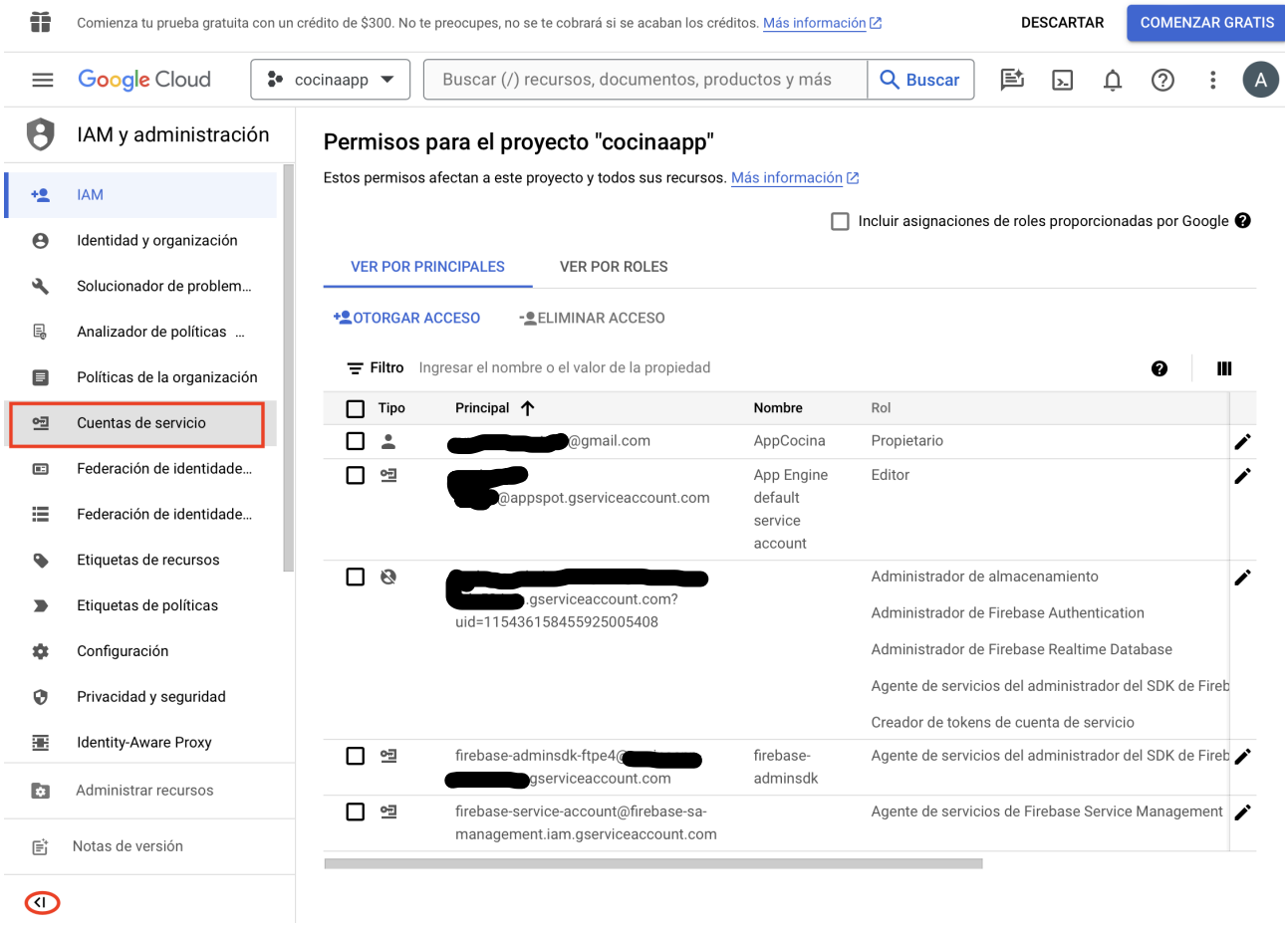
Para poner una certificación para conectar a la Base de Datos Firebase Realtime Database, lo que debemos hacer es acceder a configuración del proyecto:



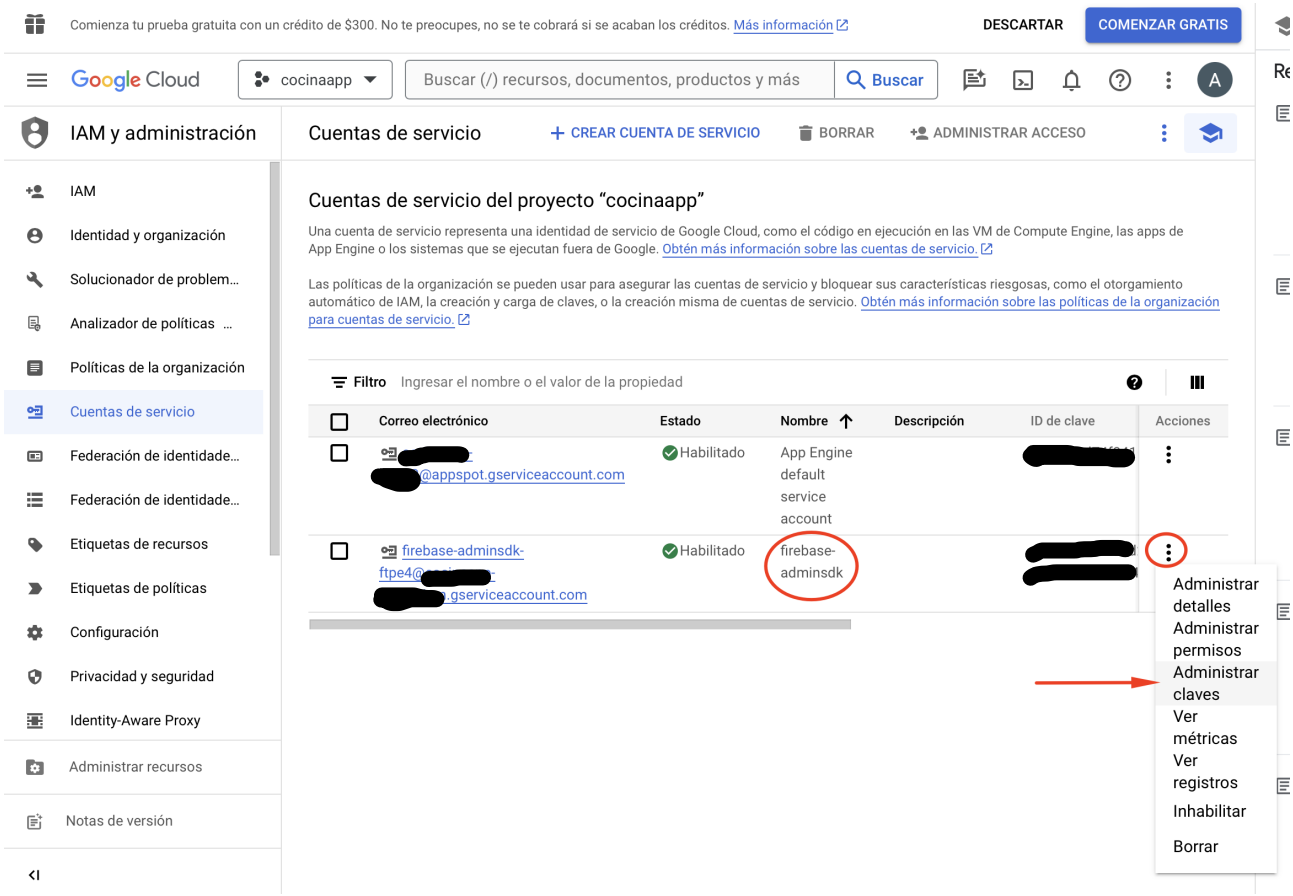
Después vamos a **Cuentas de servicio** y le damos al panel izquierdo en **X cuentas de servicio.**



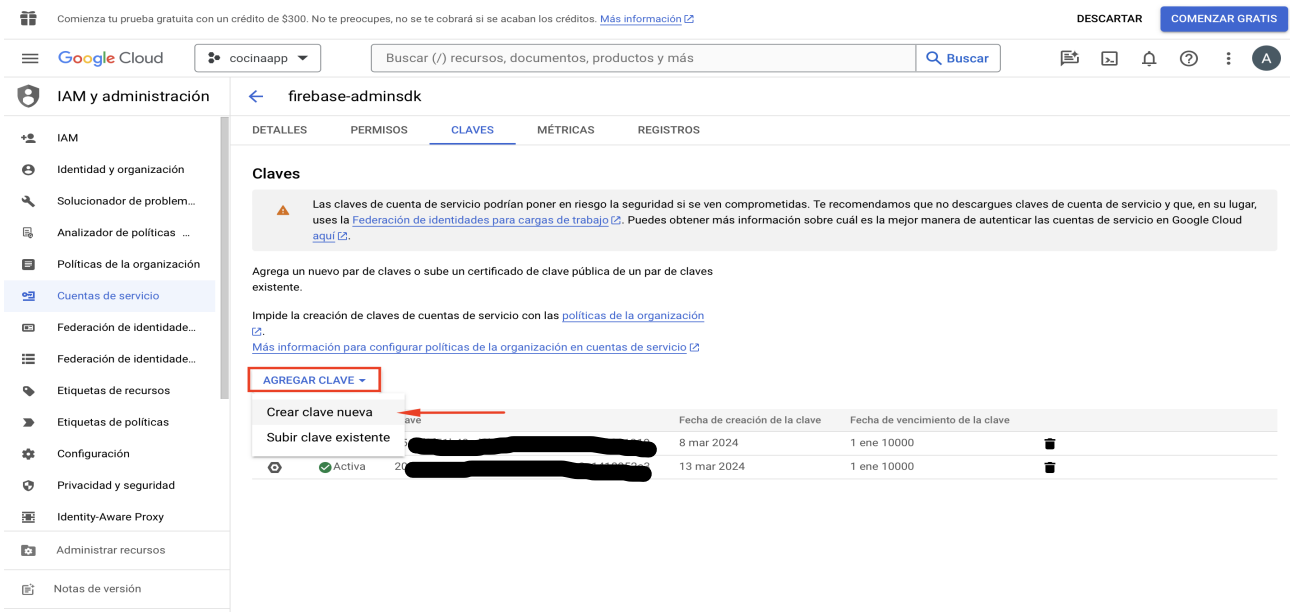
Volvemos a seleccionar cuentas de servicio y si no lo vemos le damos a la flecha para que se extienda el panel.



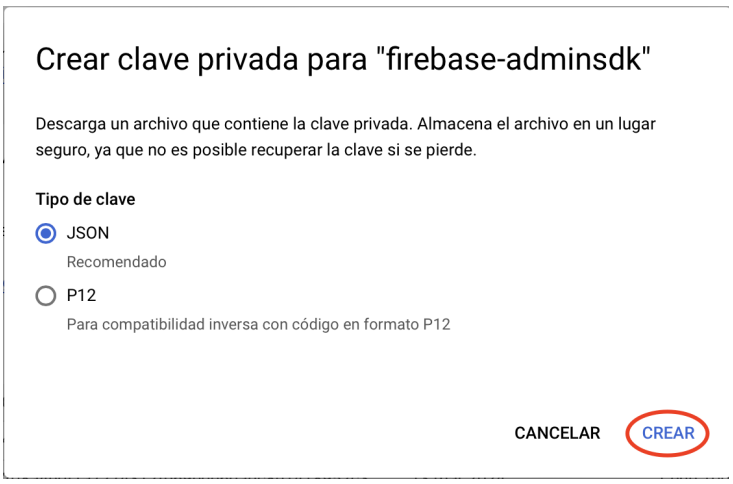
Donde pone **firebase adminsdk** seleccionamos en el menú de los tres puntos y seleccionamos **Administrar claves.**



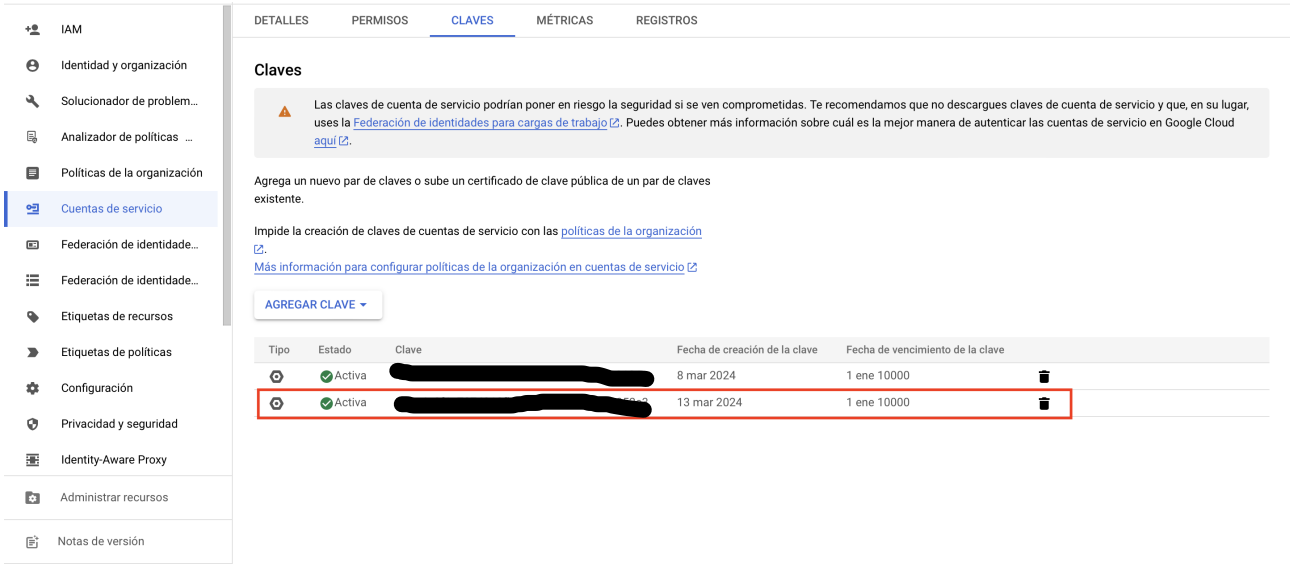
Y le damos a **Agregar Clave → Crear clave nueva.**



Y nos descargara un archivo .json con las credenciales.

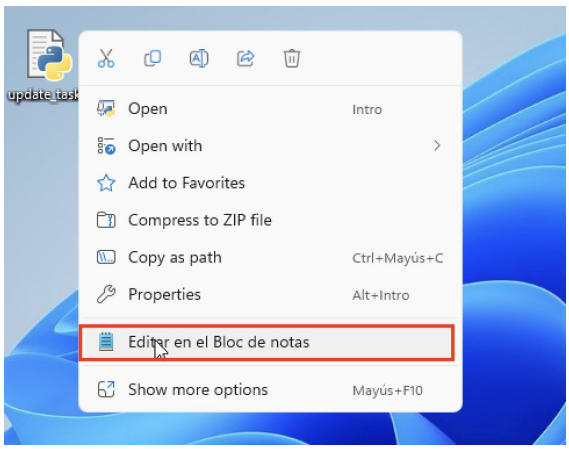


y no s habrá creado la clave en el proyecto de la base de datos.



El archivo .json que nos ha descargado lo ponemos en una localización o carpeta, en este caso se ha colocado en el escritorio.

Ahora vamos al archivo update_task_min.py y lo editamos con el block de notas.



Y ponemos la ruta donde esta el archivo .json y la url de la base de datos.

update_task_min.py - D:\Share\comprobar_editable\update_task_min.py (3.12.2)

File Edit Format Run Options Window Help

```
# importa la libreria para la conexión de firebase
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime, timedelta

#importamos la clase email_sender y email_sender_cocinapp
from email_sender import EmailSender
from email_sender_cocinapp import EmailSenderCocina

#importamos subproceso
import os

# Configura las credenciales de Firebase
cred = credentials.Certificate(os.path.join(os.path.dirname(os.path.abspath(__file__)), "coci[redacted].json"))
firebase_admin.initialize_app(cred, {'databaseURL': 'https:/[redacted]asedatabase.app'})

# Obtén una referencia a la base de datos
ref = db.reference('pedidos')

# Obtén la hora actual
hora_actual = datetime.now()

# Obtén todos los pedidos
pedidos = ref.get()

#ponemos el email y las credenciales de cocinApp
email = "[redacted].com"
password = "[redacted]"
```

aqui ponemos la credencial que nos hemos descargado en el .json debería de estar en la misma carpeta o directorio que el update_task.js

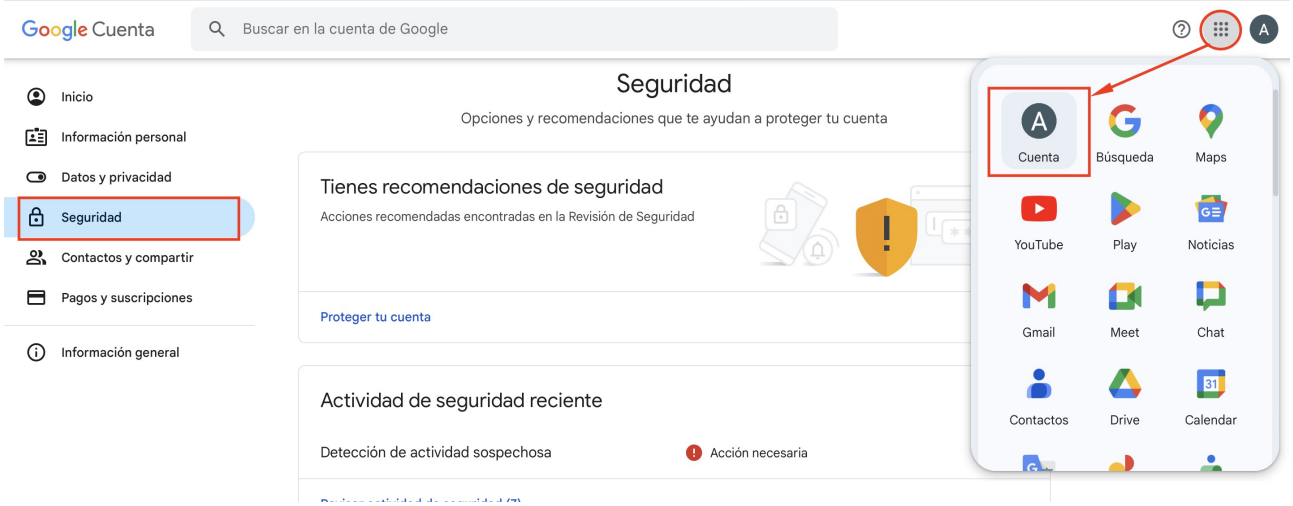
url de la base de datos

credenciales para el email.

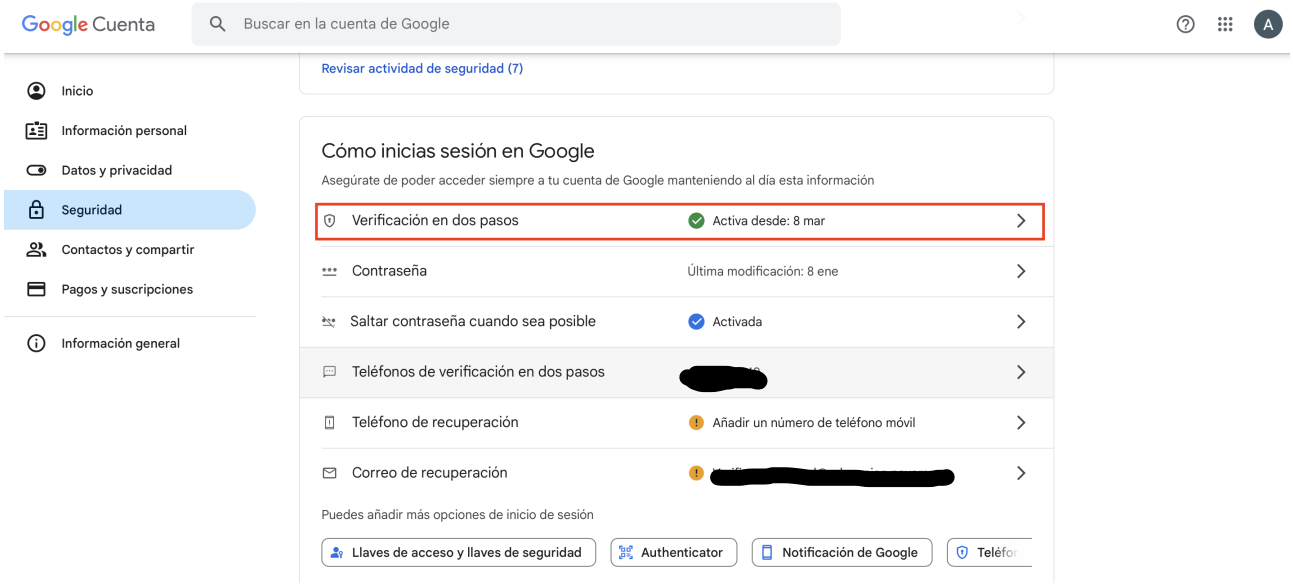
En Linux editamos con **nano,** aunque podemos editar con gedit que se puede instalar con → **sudo apt install gedit**

4.- Poner una certificación para la el envío de emails.

Abrimos la cuenta desde la que queremos enviar emails seleccionamos **cuenta** y nos ponemos en el menú de **seguridad**



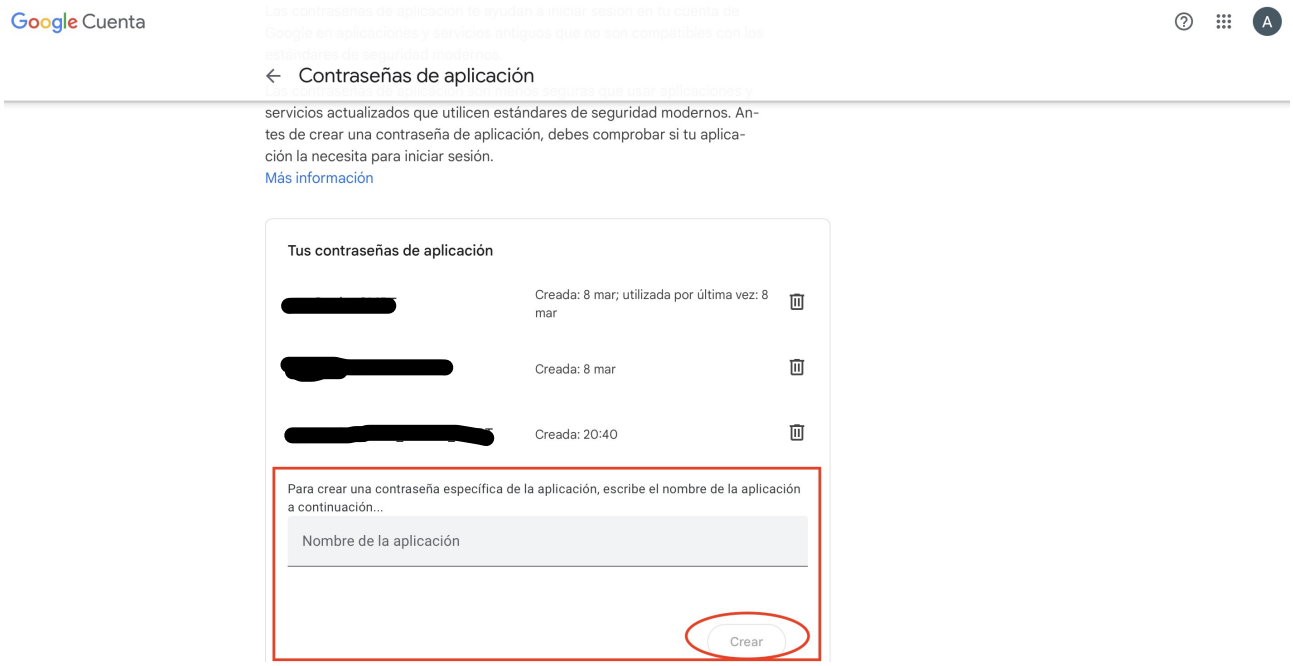
Seleccionamos **Verificación en dos pasos.**



Hacemos scroll hacia abajo hasta que encontramos **Contraseñas de aplicación.**

Nota: Si no se muestra Contraseñas de aplicación, usar este Link para que se muestre: https://myaccount.google.com/apppasswords?pli=1&rapt=AEjHL4PVBeRvsoo4g3FPzMcG7S70E7TNZ8V9I1m44HphLc1_cxMb-HY2HCDQyE04YkeWNGmAEvBXJg9Cd8-wZjYCQdjviHKGA-h_tSJHuzp1MmZwhnzQFSg

Antes de crear la contraseña de aplicación asegurarse de que lo estas realizando desde la cuenta que se quiere usar en cocina (mirar en la esquina superior derecha).



Se pone un nombre cualquiera y se le da ha crear te generará un código de 16 caracteres que habrá que ponerlo en password y en el email, el email de cocina, en **update_task_min.py**

```
hora_actual = datetime.now()

# Obtén todos los pedidos
pedidos = ref.get()

#ponemos el email y las credenciales de cocinApp
email = "[redacted]@gmail.com"
password = "[redacted]"
```

credenciales para el email.

5.- Empaquetado de la tarea.

Nota: Para conectar con el servidor: Conexion_Servidor.pdf.

Ubuntu

Una vez que todos los archivos están en la misma carpeta, abrimos el terminal en esa misma carpeta y ejecutamos

→ **sudo pyinstaller --onefile --hidden-import firebase-admin --add-data "nuestra-credencial-descargada.json:." update_task_min.py**
ejem:
→ **sudo pyinstaller --onefile --hidden-import firebase-admin --add-data "coci[redacted].json:." update_task_min.py**

esto creará un ejecutable en la carpeta dist, que no necesitará de archivos adicionales, ni de instalaciones para ejecutarse.

En Windows se creará un exe.
En Linux se crea un ejecutable que hay cambiar lo permisos.

→ **sudo chmod -x /ruta/donde/esta/el/archivo/update_task_min**
→ **sudo chmod 775 /ruta/donde/esta/el/archivo/update_task_min**

y para probar si funciona:

sudo **/ruta/donde/esta/el/archivo/update_task_min**

DEBIAN 12

Nota: El servidor donde esta alojada la aplicación web CocinApp y la tarea cambiar el estado de un pedido (update_task_min), no tiene instalado Python3, todo esto se realizó en otra máquina virtual Debian 12.

El ejecutable en Ubuntu no se ejecuta de forma correcta en Debian (nos da un error de 1. se espera)…) , por lo que hay que hacer algunos cambios.
La instalación en Debian es un poco diferente, no permite la instalación de paquetes por **pip** por lo que debemos hacer un entrono virtual de python para ello, primero nos ponemos en superusuario.

→ $ **su**
instalacion de python:
→ # **sudo apt install python3**
instalamos el entorno virtual de python.
→ # **sudo apt install python3-venv**
Crear un entorno virtual

→ # **python3 -m venv myenv**
Activar el entorno virtual
→ # **source myenv/bin/activate**
Instalar 'firebase-admin' dentro del entorno virtual
→ (myenv) # **pip install firebase-admin**
Instalar 'pyinstaller' dentro del entorno virtual
→ **(myenv) # pip install pyinstaller**
Creamos el ejecutable, estando en la misma carpeta que están los 4 archivos (update_task, email_sender.py, email_sender_cocinapp.py y coci[redacted].json) vamos a crearlo.
→ (myenv) # **pyinstaller --onefile --hidden-import firebase-admin --add-data "cocina[redacted].json:." update_task_min.py**
Una vez que se ha creado el instalable, vamos a la carpeta **dist.**
→ (myenv) # **cd dist**
Ahora le damos los permisos de ejecución y cambiamos los permisos de la carpeta.
→ (myenv) # **chmod -x /home/cocinapp/Tarea/dist/update_task_min**
→ (myenv) # **chmod 755 /home/cocinapp/Tarea/dist/update_task_min**
Salimos del entorno virtual.
→ (myenv) # **deactivate**
Ahora podemos ejecutar el archivo.
→ # **/home/cocinapp/Tarea/dist/update_task_min**
y vemos que se arregla el error y se ejecuta la tarea.

6.- Programar las Tareas

luego vamos a usar cron que es el programador por defecto de Linux.

→ **sudo crontab -e**
seleccionamos 1 para nano y añadimos al final del todo

*** * * * * /ruta/donde/esta/el/archivo/update_task_min**

esto creará una tarea que se ejecutará cada minuto. Si queremos ver si se ha guardado bien ponemos

→ **sudo crontab -l**

Si queremos deshabilitarlo → **sudo crontab -e**

y le ponemos una '#' delante para que se quede como un comentario.

 # * * * * * /ruta/donde/esta/el/archivo/update_task_min

En Windows crear una tarea que ejecute el .exe creado cada minuto.

Instalación de la aplicación servNotificaciones.jar para el envío de avisos al movil en forma de pop up.

Descomprimimos ejecutable7.z y los colocamos dentro del directorio **/home/usuario/** quedando de esta forma:

/home/usuario/ejecutable/key.json
/home/usuario/ejecutable/servNotificaciones.jar

Miramos a ver si estamos en superusuario.
→ **$ su**

Instalamos el entorno de rutina de Java para que se puedan ejecutar los JAR.
→ **# sudo apt install default-jre**

Ahora nos ponemos en el mismo directorio que donde esta la tarea
→ **# cd /home/usuario/ejecutable**

Ahora desde el directorio que estamos vamos a ejecutar el archivo.
→ **# java -jar servNotificaciones.jar**

Ahora ponemos todas las tareas en el programador de tareas.
→ **#crontab -e**

y añadimos esta otra línea donde @reboot, solo se iniciará una vez que se inicie el el servidor.

@reboot cd /home/usuario/ejecutable && java -jar servNotificaciones.jar

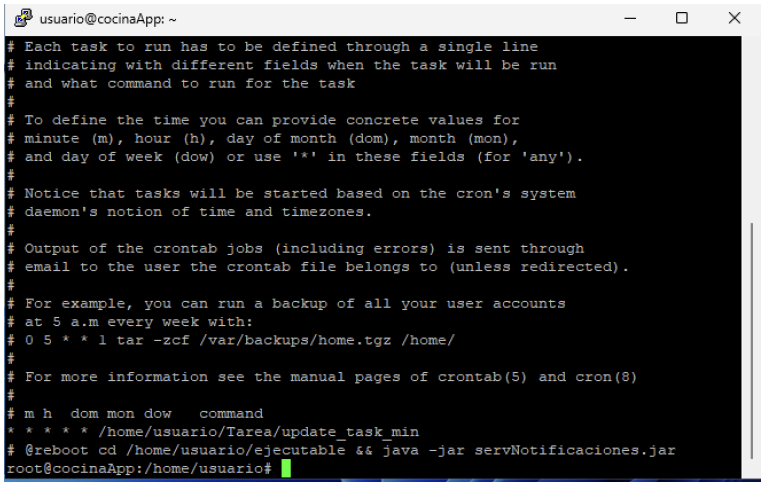
cerramos y guardamos y si lo queremos ver

→ **#crontab -l**

Nota: se deja comentado en crontab servNotificaciones.jar para que no se ejecute debido a que aparece problemas con la ejecución simultanea junto con la tarea update_task_min.

 # @reboot cd /home/usuario/ejecutable && java -jar servNotificaciones.jar

Visualización de como se queda el crontab:



Visualización de como se queda las carpetas de las Tareas.