Informe sobre el Proceso de Envío y Recepción de Correos de Rendimiento

1. Introducción

Este informe documenta el proceso de envío y recepción de correos electrónicos con un informe de rendimiento para los empleados de la organización. Los correos se envían automáticamente cada día a las 9 AM, utilizando un script Python programado mediante una tarea crontab en un script Bash. Los informes se personalizan para cada empleado y se envían adjuntos en formato PDF.

2. Automatización del Proceso de Envío

2.1 Script Bash

El proceso de automatización se inicia con un script Bash denominado `automate_birthday_emails.bash`, cuyo propósito es programar la ejecución diaria del script Python encargado del envío de correos. El script Bash establece una tarea en crontab que asegura que el script Python se ejecute todos los días a las 9 AM.

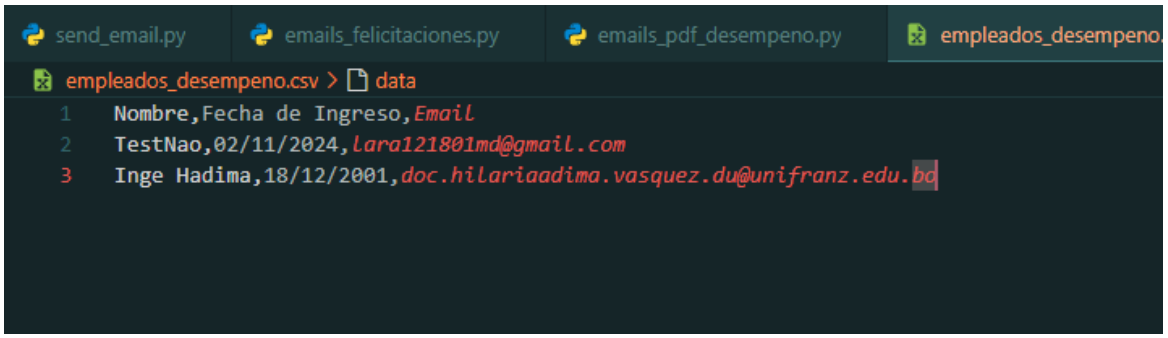
- **Crontab**: Utiliza `crontab` para programar la tarea diaria.
- **Ejemplo de comando crontab**:

```
(crontab -l ; echo "0 9 * * * /usr/bin/python3 /ruta/a/emails_pdf_desempeno.py") | crontab -
```

2.2 Script Python

El script Python (`emails_pdf_desempeno.py`) es el encargado de leer la lista de empleados desde un archivo CSV (`empleados_desempeno.csv`) y enviar correos electrónicos personalizados, adjuntando un informe de rendimiento en PDF. Las principales tareas del script son:

- **Leer la lista de empleados**: Se lee el archivo CSV que contiene los nombres, fechas de ingreso y correos electrónicos de los empleados.
- **Generar el informe PDF**: Para cada empleado, se genera un informe PDF que detalla el rendimiento del último periodo, el cual se genera de manera aleatoria a partir de cinco posibles evaluaciones ("Excelente", "Muy bueno", "Bueno", "Regular", "Necesita mejorar").
- **Enviar el correo**: Se configura el servidor SMTP para Gmail, se personaliza el cuerpo del correo con el nombre del empleado y se adjunta el PDF generado antes de enviar el mensaje.



3. Flujo del Proceso de Envío

3.1 Configuración y Ejecución del Script Bash

- **Ejecución Inicial**: El administrador ejecuta el script Bash para establecer la tarea programada.
- **Programación Crontab**: Se agrega una entrada a crontab que ejecuta el script Python cada día a las 9 AM.
- **Confirmación**: El script Bash muestra un mensaje de confirmación indicando que la tarea ha sido programada con éxito.

3.2 Generación y Envio del Correo con el Script Python

- **Lectura del Archivo CSV**: El script Python abre el archivo `empleados_desempeno.csv` y lee la información de los empleados.
- **Generación del PDF**: Para cada empleado, se genera un PDF con un informe de rendimiento personalizado. Este PDF se guarda con un nombre que incluye el nombre del empleado, por ejemplo: `informe_rendimiento_TestNao.pdf`.
- **Envío del Correo**: Utilizando la librería `smtplib`, el script establece una conexión con el servidor SMTP, prepara el mensaje con el cuerpo HTML, y adjunta el informe en PDF. Luego, el correo se envía al destinatario.
- **Manejo de Errores**: Si ocurre un error durante el envío (por ejemplo, un error de autenticación), el script lo detecta y muestra un mensaje de error en la consola.

```
import csv
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
import os
from dotenv import load_dotenv
from datetime import datetime
from fpdf import FPDF
import random

# Cargar variables de entorno desde el archivo .env
load_dotenv()

# Configuración del servidor SMTP
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
USERNAME = os.getenv('EMAIL_USERNAME')
PASSWORD = os.getenv('EMAIL_PASSWORD')  # Usar variable
de entorno para la contraseña


def crear_pdf(nombre_empleado, rendimiento):
```

```
    """
    Crea un PDF personalizado con un informe de rendimie
nto aleatorio.

    :param nombre_empleado: Nombre del empleado para per
sonalizar el PDF.
    :param rendimiento: Descripción del rendimiento del
empleado.
    :return: Ruta del archivo PDF creado.
    """
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=16)
    pdf.cell(200, 10, txt="Informe de Rendimiento de:
{}".format(nombre_empleado), ln=True, align='C')
    pdf.ln(10)
    pdf.set_font("Arial", size=12)
    pdf.multi_cell(0, 10, txt="Este informe contiene un
resumen del desempeño del empleado durante el último per
íodo. El rendimiento ha sido evaluado como: {}".format(r
endimiento))
    nombre_pdf = "informe_rendimiento_{}.pdf".format(nom
bre_empleado.replace(" ", "_"))
    pdf.output(nombre_pdf)
    return nombre_pdf


def enviar_saludo(destinatario, nombre_empleado, rendimi
ento):
    """
    Envía un saludo al destinatario especificado, adjunt
ando un PDF personalizado.

    :param destinatario: Correo electrónico del destinat
ario.
    :param nombre_empleado: Nombre del empleado para per
sonalizar el mensaje.
    :param rendimiento: Descripción del rendimiento del
```

```
empleado para personalizar el PDF.
    """
    # Configurar el correo electrónico
    mensaje = MIMEMultipart()
    mensaje['From'] = USERNAME
    mensaje['To'] = destinatario
    mensaje['Subject'] = 'Saludos desde Soluciones innov
adoras en el lenguaje de programación local'

    # Leer el cuerpo del correo desde el archivo HTML
    with open('felicitaciones.html', 'r', encoding='utf-
8') as file:
        cuerpo = file.read().replace("{nombre_emplead
o}", nombre_empleado)

    mensaje.attach(MIMEText(cuerpo, 'html'))

    # Crear el PDF personalizado y adjuntarlo al correo
    pdf_path = crear_pdf(nombre_empleado, rendimiento)
    with open(pdf_path, "rb") as pdf_file:
        mensaje.attach(MIMEApplication(pdf_file.read(),
_subtype="pdf", Name=pdf_path))

    # Enviar el correo
    try:
        servidor = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        servidor.starttls()
        servidor.login(USERNAME, PASSWORD)
        servidor.sendmail(USERNAME, destinatario, mensaj
e.as_string())
        servidor.quit()
        print(f'Correo enviado a {destinatario}')
    except smtplib.SMTPAuthenticationError:
        print(f'Error de autenticación al enviar correo
a {destinatario}: Verifique las credenciales.')
    except Exception as e:
        print(f'Error al enviar correo a {destinatario}:
{e}')
```

```
def leer_lista_desempeno(archivo_csv):
    """
    Lee la lista de empleados desde un archivo CSV y env
ía informes de desempeño aleatorios.

    :param archivo_csv: Ruta del archivo CSV con la list
a de empleados.
    """
    evaluaciones = ["Excelente", "Muy bueno", "Bueno",
"Regular", "Necesita mejorar"]
    with open(archivo_csv, mode='r') as file:
        reader = csv.reader(file)
        next(reader)  # Saltar la cabecera
        for row in reader:
            if row:
                nombre_empleado = row[0]
                destinatario = row[2]  # Utilizar el ema
il especificado en la tercera columna
                rendimiento = random.choice(evaluacione
s)
                enviar_saludo(destinatario, nombre_emple
ado, rendimiento)


# Ruta del archivo CSV con la lista de empleados
archivo_csv = 'empleados_desempeno.csv'

# Ejecutar la función para leer la lista y enviar los sa
ludos
leer_lista_desempeno(archivo_csv)
```

4. Ejemplo de Archivo CSV

El archivo CSV (`empleados_desempeno.csv`) contiene la lista de empleados que recibirán el informe de rendimiento. La estructura del archivo es la siguiente:

```
Nombre,Fecha de Ingreso,Email
TestNao,02/11/2024,lara121801md@gmail.com
Inge Hadima,18/12/2001,doc.hilariaadima.vasquez.du@unifranz.edu.bo
```

- **Nombre**: Nombre del empleado.
- **Fecha de Ingreso**: Fecha de ingreso del empleado a la empresa.
- **Email**: Correo electrónico del empleado.

5. Recepción del Correo

5.1 Contenido del Correo Electrónico

Cada correo electrónico enviado contiene:

- **Asunto**: "Saludos desde Soluciones innovadoras en el lenguaje de programación local".
- **Cuerpo del Correo**: Un mensaje personalizado dirigido al empleado, extraído de un archivo HTML (`felicitaciones.html`), que incluye el nombre del empleado.
- **Adjunto**: Un archivo PDF (`informe_rendimiento_<nombre_empleado>.pdf`) que contiene un resumen del desempeño del empleado.

5.2 Recepción por parte del Empleado

El empleado recibe el correo electrónico con el informe de rendimiento adjunto. El correo está personalizado con su nombre y el contenido refleja un reconocimiento de su trabajo durante el último periodo.

6. Conclusiones

El proceso descrito asegura que los empleados reciban sus informes de rendimiento de manera regular y automática. Esta automatización permite reducir la carga administrativa y asegura que todos los empleados reciban el reconocimiento adecuado por su trabajo. Además, el uso de crontab y scripts Python asegura una solución eficiente y escalable.

7. Posibles Mejoras

- **Integración con Bases de Datos**: Sustituir el archivo CSV por una base de datos para mejorar la escalabilidad y la seguridad.
- **Notificaciones de Entrega**: Implementar notificaciones para confirmar la entrega exitosa de los correos.
- **Interfaz Gráfica**: Crear una interfaz para que los administradores puedan gestionar la lista de empleados y visualizar el estado de los envíos.