# Solución de Analítica y Gobierno de Datos para Optimizar la Selección de Personal en Periferia IT

**Versión:** 1.0
**Desarrollador:** Freller Jose Blanchar Alvarez
**Fecha:** 24/09/2024

## Índice

# 1 Introducción

Este manual de usuario tiene como objetivo guiar a los usuarios en el uso de una solución de analítica y gobierno de datos, diseñada para optimizar los procesos de selección de personal en Periferia IT. La solución utiliza datos anonimizados extraídos desde Azure Blob Storage y procesados en una base de datos PostgreSQL. Los resultados se visualizan a través de Power BI, proporcionando una herramienta eficiente para mejorar la toma de decisiones en el equipo de reclutamiento.

## Objetivos del Sistema

- Procesar datos de selección de personal almacenados en formato Parquet desde un Blob Storage de Azure.
- Almacenar y analizar los datos utilizando PostgreSQL.
- Visualizar y reportar los resultados en Power BI para facilitar el análisis y seguimiento de los procesos de selección.

# 2 Requisitos previos

Software y herramientas necesarias:

- Python 3.x con las siguientes bibliotecas instaladas:

  Pandas
  psycopg2
  sqlalchemy
  azure-storage-blob
  tkinter
  Pillow

- PostgreSQL: Un servidor PostgreSQL para almacenar los datos.
- Power BI Desktop: Herramienta para la visualización y análisis de datos.
- Acceso a Azure Blob Storage: Las credenciales de acceso (URL de cuenta, nombre del contenedor y token SAS).

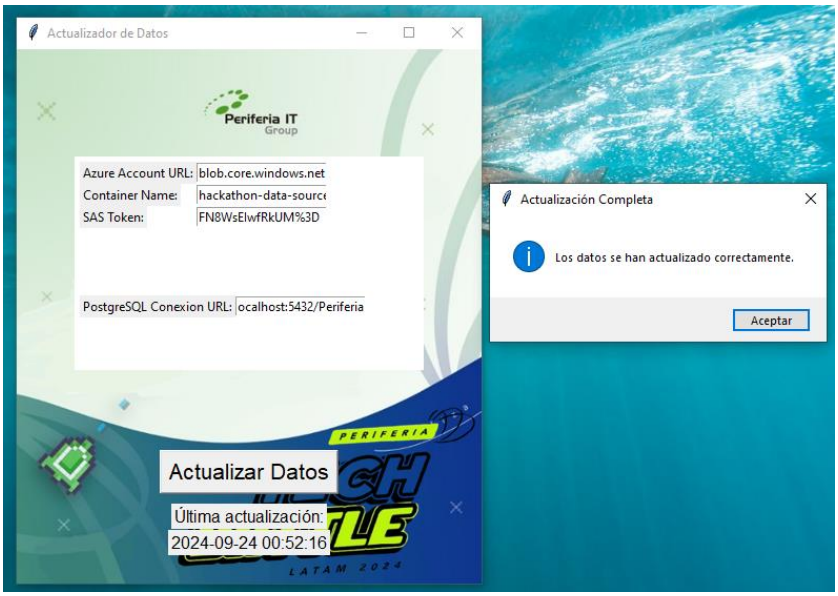
# 3 Configuración

- Azure Blob Storage
  Se obtiene el **URL de la cuenta**, el **nombre del contenedor** y el **SAS Token** desde Azure para acceder a los datos almacenados en formato Parquet.
  -account_url = https://hackathonperiferiait.blob.core.windows.net
  -container_name = hackathon-data-source
  -sas_token = sp=rl&st=2024-09-17T22:37:26Z&se=2024-10-04T06:37:26Z&spr=https&sv=2022-11-02&sr=c&sig=CYqWUat230DmfqyGFxs32vGff5LkkPFN8WsEIwfRkUM%3D
- PostgreSQL
  Se prepara la URL de conexión PostgreSQL en el siguiente formato:
  -postgresql://postgres:292754bB@localhost:5432/Periferia
  Esta URL será ingresada en la aplicación para conectarse al servidor de base de datos.

# 4 Uso del Sistema

## Ingreso de Credenciales.

Al ejecutar el programa, se abrirá la interfaz gráfica de usuario (GUI). En esta ventana, ingrese las credenciales solicitadas:
**Azure Account URL**: La URL de la cuenta de almacenamiento en Azure.
**Container Name**: El nombre del contenedor donde están los archivos Parquet.
**SAS Token**: El token SAS que otorga acceso a la cuenta de almacenamiento.
**PostgreSQL Conexion URL**: La URL de conexión de PostgreSQL.




**Imagen**

### Actualización de Datos

Haga clic en el botón **"Actualizar Datos"**. Esto iniciará el proceso de descarga de los archivos desde Azure y la actualización de los datos en la base de datos PostgreSQL.
El sistema mostrará un mensaje cuando los datos hayan sido actualizados correctamente.
La hora de la última actualización se mostrará en la parte inferior de la ventana.

Manejo de Archivos
**Archivos Parquet**: Los archivos descargados se almacenarán en la ruta especificada C:\Users\Gilma\Documents\OperacionM30\Downloads. Si los archivos ya existen en esta carpeta, no se descargarán nuevamente para evitar duplicados.

## 5 Visualización en Power BI

### Importación de Datos

Una vez que los datos se hayan actualizado en la base de datos PostgreSQL, se pueden visualizar en **Power BI**.

### Creación de Informes

Se utilizó los datos obtenidos para realizar los dashboard en donde finalmente se implementaron 3 tableros el primero llamado pagina1, pagina2, y pagina3.
estas paginas fueron asignadas con los botones para su acceso directo.

pagina1  pagina2  pagina3 

**Tablero 1**. Su contenido es la visualización de las vacantes con sus respectivos detalles como los requerimientos fundamentales de las vacantes, la cantidad de puestos disponibles por vacantes y su relación con la tabla Job_Applications la que nos muestra los candidatos que aplican a cada vacante también tenemos una tabla en donde nos muestra todos los candidatos que aplican a cada vacante teniendo así una visualización clara y especifica de los candidatos que aplican a la vacante seleccionada por ultimo tenemos el apartado de resumen de hoja de vida para

tener previamente la opción de observar el perfil de esos candidatos que fueron seleccionados de una forma sencilla y rápida

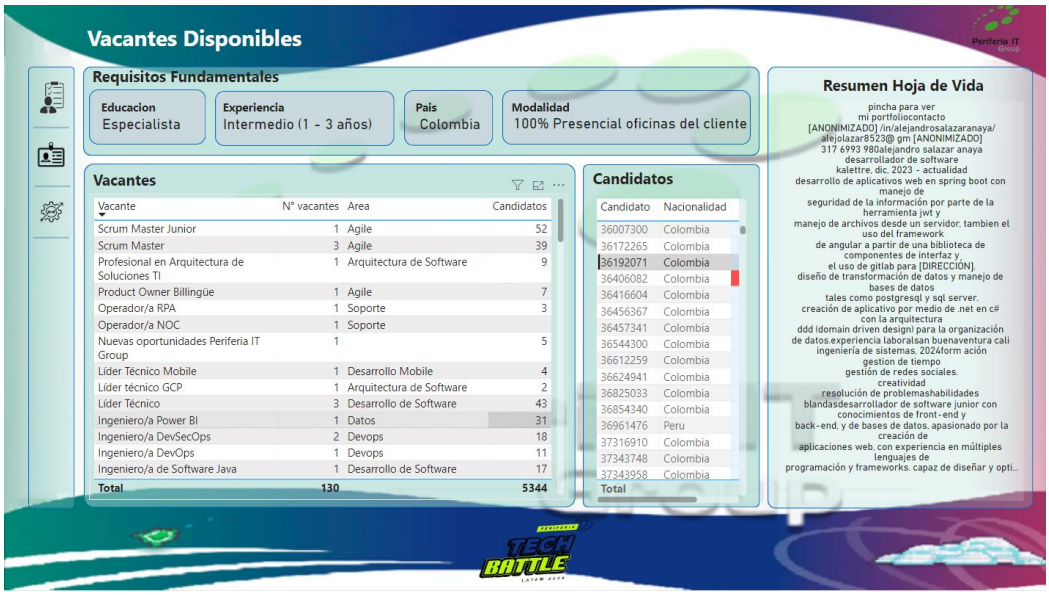



**Imagen 1.**

Se alcanza apreciar en la tabla candidatos un rectángulo de color rojo que indica automáticamente el descarte de ese candidato esto se logro diseñar agregando una columna al tabla llamada ResumeEnBlanco en donde me muestra los candidatos que no tienen resumen de hoja de vida

ResumeEnBlanco = IF(ISBLANK('public Job_Applications'[anonymized_resume]), 1, 0)

con este resultado podemos crear una condición en donde si el valor es 1 rellenar de rojo la columna.

**Tablero 2**. En este tablero tenemos la información detallada del candidato mientras que en el tablero1 tenemos la relación vacante – candidato en este tenemos a detalle su información registrada tanto como fechas, clasificación, motivo de rechazo, si es empleado interno, si es referido, si fue rechazado por la empresa, si sigue conectado a la red, etc.

modificaciones que se realizaron en este tablero agregado de medidas y agregados de columnas para el cambio de formato de fecha ya que este dato de fecha venia en formato ISO y era necesario transformar estos datos a una fecha elegible para visualizar correctamente las fechas de ingreso y rechazo, teniendo este conocimiento se aplicó la siguiente fórmula para las columnas nueva

= Text.Middle('public Job_Applications'[job-app-created-at], 1, 10)

= Text.Middle('public Job_Applications'[job-app-rejected-at], 1, 10)


**Imagen 2.**

La tabla información fue diseñada en base a los datos relevantes que acompañaban los datos principales de esta manera para poder utilizarlos de modo que me dieran un resultado (si o no) fue necesario crear medidas para cada dato booleano utilizado.
se aplico la siguiente medida para cada dato booleano.

```
conectado_red =
IF (
    SELECTEDVALUE('public Job_Applications'[candidate-connected]) = TRUE,
    "Si",
    "No"
)

Cand_Ref =
IF (
    SELECTEDVALUE('public Job_Applications'[candidate-referred]) = TRUE,
    "Si",
    "No"
)
```

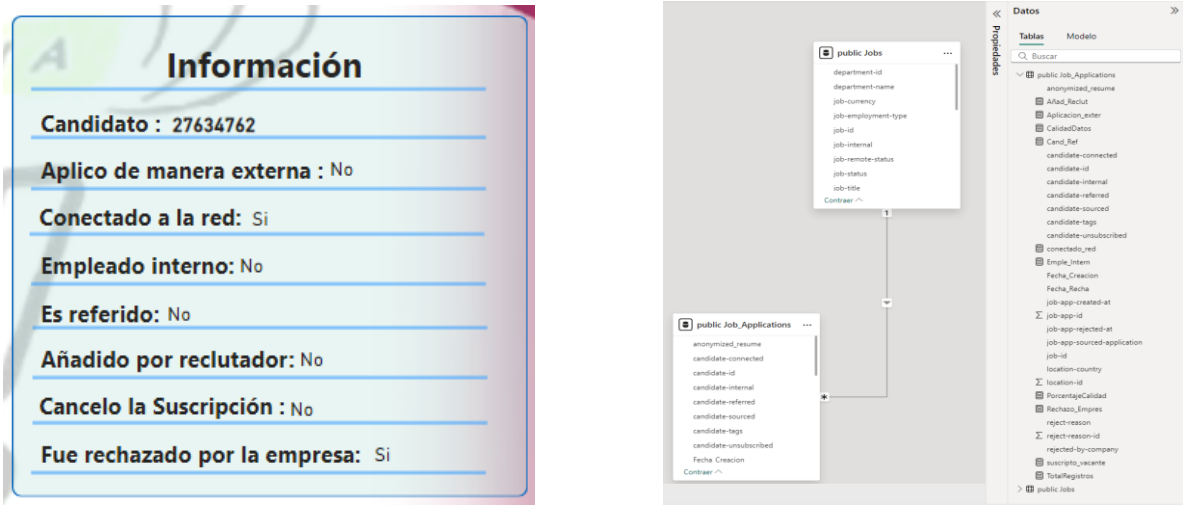



**Imagen 3.**

Tablero3. Indicador de calidad de datos en este tablero se evalúa la calidad de los datos proporcionado de manera que muestre un indicativo de la calidad de los datos extraídos esto se logra analizando los datos de la tabla Job_Applications en donde destacan datos faltantes o defectuosos esto quiere decir que en la columna anonymized_resume hay campos vacíos y esta columna es resumen hoja de vida lo cual este candidato estaría descartado automáticamente sin poder hacerle evaluarlo.

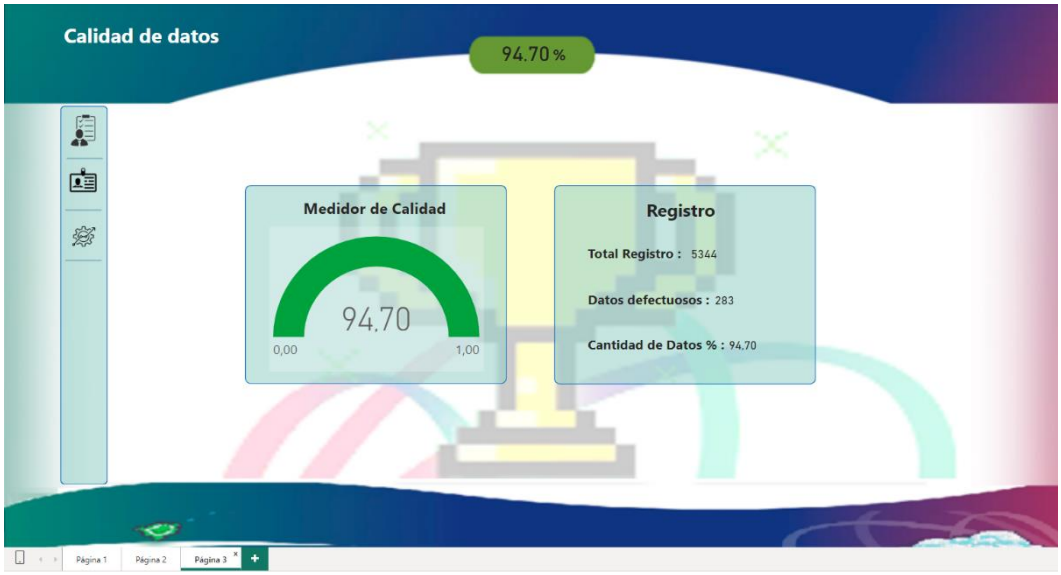



**Imagen 4.**

Teniendo en cuenta estos parámetros la calidad de los datos supera el 90% de su calidad teniendo en cuenta que los datos defectuosos son solo el 5.3% de los 5.344 datos analizados.

estos resultados fueron posibles por medio del cálculo de medidas.

**porcentajeCalidad**

PorcentajeCalidad = DIVIDE('public Job_Applications'[TotalRegistros] - 'public

Job_Applications'[CalidadDatos], 'public Job_Applications'[TotalRegistros], 0) * 100

**TotalRegistros**

TotalRegistros = COUNTROWS('public Job_Applications')

**CalidadDatos**

CalidadDatos = COUNTROWS(FILTER('public Job_Applications', ISBLANK('public Job_Applications'[anonymized_resume])))


**Imagen 5.**

Por último tenemos el componente que contienen todas las tablas. Son botones que llevan a cada tabla a la cual corresponde su funsionalidad

**Imagen 6.**

## 6 Conclusión

Esta solución proporciona una herramienta integral de analítica y gobierno de datos para optimizar el proceso de selección de personal en **Periferia IT**. A través de la integración de Azure Blob Storage, PostgreSQL y Power BI, el equipo de reclutamiento puede tomar decisiones más informadas y eficientes basadas en datos anonimizados y reportes visuales.