UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

APOYO EN LOS PROCESOS DE CUENTAS MAESTRAS

INFORME DE PASANTÍA PARA OPTAR POR EL TÍTULO DE MATEMÁTICO
PROYECTO CURRICULAR DE MATEMÁTICAS

Laura Katherine Pira Rincón

Jefe Inmediato: Fernando Olivera Villanueva

Profesor Evaluador: Carlos Orlando Ochoa

BOGOTÁ DC
SEPTIEMBRE DE 2021

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente de jurado

_____

Firma del jurado

_____

Firma del jurado

Agradecimientos

Quiero agradecer a las personas que contribuyeron a mi crecimiento personal, académico e intelectual durante mi paso por la carrera de Matemáticas y mis prácticas laborales en el Ministerio de Hacienda y Crédito Público, en especial a Juan Camilo Murillo por creer en mis capacidades y darme la confianza para enfrentarme a nuevos retos en el ámbito laboral y profesional.

Tabla de Contenido

Lista de figuras

Lista de tablas

1. Introducción

La presente pasantía fue realizada en el Equipo de Cuentas Maestras del grupo 028 de la Subdirección Financiera de Apoyo fiscal y Saneamiento Básico de la Dirección General de Apoyo Fiscal del Ministerio de Hacienda y Crédito Público del Gobierno de Colombia.

El mencionado equipo cumple como función el seguimiento a la información reportada por parte de los establecimientos bancarios de las Cuentas Maestras y Cuentas Maestras Pagadoras por medio de las cuales se administra recurso público girado por parte del Tesoro Nacional del Ministerio de Hacienda y Crédito Público, y que a su vez están a nombre de las Entidades Territoriales y sus Entidades Descentralizadas, Instituciones Educativas, Resguardos Indígenas, entre otros, con el fin de subsanar las diferentes necesidades y requerimientos de las distintas entidades.

Las Cuentas Maestras y Cuentas Maestras Pagadoras están reglamentadas por medio de una serie de leyes y resoluciones las cuales tienen por objetivo regular la apertura y operación de estos productos bancarios, con el fin de garantizar la trazabilidad y transparencia en el uso y ejecución de los recursos.

El enfoque para el periodo 2021-I se inclinó hacia el reproceso de la información que recibe el Ministerio de las Cuentas Maestras Pagadoras, las cuales cumplen el papel de ser un producto complementario a las Cuentas Maestras para realizar transacciones electrónicas por PSE y que por lo tanto solo recibirán recursos desde su respectiva Cuenta Maestra.

El objetivo de esta pasantía radicó en el entendimiento de los procesos y el tema de Cuentas Maestras junto con su normatividad para con ello generar una relación matemática con la información reportada por los bancos, para así añadir rigurosidad y agilidad en el análisis a la calidad de la información.

Según lo anterior, la información reportada de las Cuentas Maestras Pagadoras se divide en tres(3) tipos de registros que deben llenar cada uno de los bancos mensualmente, en donde el Registro Tipo 1 es utilizado para identificar la fuente de información, el Registro Tipo 2 Presenta la información de las Cuentas Maestras Pagadoras tal como sus nombres, tipos de cuenta, número de cuenta, información de la entidad territorial o institución titular de la cuenta, entre otros y el Registro Tipo 3 incluye la información del total de movimientos realizados con cada una de estas cuentas.

De esta forma, el cuerpo de este informe se divide en cuatro capítulos los cuales son: Python para la validación del error estándar, evaluación y caracterización de riesgos, modelo en Python para la validación de información reportada y blog de Cuentas Maestras.

Para el primer capítulo se pretendió realizar un informe sobre el uso de un lenguaje de programación para validar el error máximo tolerado, verificado a partir de una muestra calculada y consistente sobre la base total de datos. Por otro lado, para el capítulo dos, se describe el trabajo hecho en conjunto con el equipo del Ministerio en donde se procedió a realizar un sistema

de ponderización de los errores e inconsistencias encontradas en los datos para cada banco y así clasificarlos por nivel de riesgo con el fin de establecer un panorama en las situaciones de cada uno de los establecimientos bancarios. En cuanto al capítulo 3, se presenta un conjunto de problemas planteados frente a los cuales se realiza el proceso de generar un código de validación para analizar y revisar la incidencia o no en estas inconsistencias para cada uno de los bancos; Finalmente, para el último capítulo, pero no menos importante, se describe la necesidad de integrar los conocimientos aprendidos durante la carrera de matemáticas con las necesidades del Ministerio por medio de un blog de Cuentas Maestras implementado a través del Share Point, con el cual se pretende trasmitir de la mejor forma conceptos matemáticos y que son bases importantes en el tratamiento de datos y análisis de datos lo cual podría resultar de utilidad para los diferentes equipos del Ministerio.

2. Objetivos

2.1. Objetivo General

Integración al Equipo de Cuentas Maestras del Ministerio de Hacienda y Crédito Público, entendiendo y acoplándose a su dinámica para desde las matemáticas y la programación aportar conocimientos y herramientas para agilizar los procesos de análisis a la calidad de la información reportada por los establecimientos bancarios.

2.2. Objetivos específicos

- **Objetivo 1:** Elaborar un código por medio de un lenguaje de programación con el fin de generar operaciones de validación para detectar errores e inconsistencias en la información que viaja al Ministerio optimizando el tiempo de trabajo del equipo enfocado en la revisión y detección de estos problemas

- **Objetivo 2:** Elaborar una propuesta de un Blog de Cuentas Maestras por medio del cual se pretenda generar un lazo entre los conocimientos adquiridos en la carrera de Matemáticas con los temas de interés del Ministerio y el Equipo de Cuentas Maestras en el uso de bases de datos.

3. Python para la validación del error estándar

3.1. Contexto

En la medida en que se generó la resolución que reglamentó estas cuentas en el año 2018, las Entidades Territoriales y Establecimientos Bancarios atravesaron una etapa de adaptación y procesamiento del ejercicio de apertura, uso y reporte de las Cuentas Maestras Pagadoras incidiendo en diversas inconsistencias evidenciadas por los Ministerios, razón por la cual la revisión a la calidad de la información comienza con las bases de datos registradas desde el año 2019.

De tal parte, para realizar un análisis y revisión efectiva a la información reportada por los establecimientos bancarios, el Ministerio estableció una serie de errores implementados a través de Power BI por medio del cual se pretende evaluar la incidencia o no en alguno de ellos por parte de los bancos y así mismo determinar el número de errores y las Cuentas Maestras Pagadoras en las que se presentan; estas inconsistencias van desde errores en el código DIVIPOLA de los municipios o departamentos, nombre del titular de la Cuenta Maestra Pagadora, hasta errores en el reporte incompleto o incorrecto en los movimientos de estas cuentas.

Posteriormente, en el inicio del proceso de revisión de información, dentro de los errores en el código DIVIPOLA se evidenció una alta incidencia por parte de los bancos por lo cual se hizo necesario establecer un error máximo tolerado, es decir, un error en el que se establece un límite y sobrepasarlo indicaría un riesgo en la transparencia de los datos remitidos por las entidades bancarias por medio de una plataforma de integración denominada PISIS, y que a su vez es una plataforma que alimenta las bases de datos del Ministerio.

3.2. Desarrollo del Código

Como resultado de la necesidad anteriormente descrita, se procedió a la búsqueda de herramientas matemáticas que permitieran establecer un error máximo tolerado y a su vez determinar la base de datos sobre la cual se debiera hacer el calculo requerido.

A partir de lo anterior y entendiendo que nuestras variables a analizar son los porcentajes de error de los bancos a través del tiempo, el número de cuenta en la que se presenta esta inconsistencia y el banco en la que se encuentra, se hizo uso de la siguiente ecuación:

$$n = \frac{N * Z^2_{\alpha/2} * p * q}{e * (N-1) + Z^2_{\alpha/2} * p * q}$$

en donde como ya lo conocemos es la formula determinada para el calculo del tamaño de la muestra conociendo previamente el tamaño real de la población total.

Posteriormente se calcula la desviación estándar (σ), y haciendo uso de la muestra de la

población obtenida y finalmente calculamos el error máximo tolerado a partir de la formula

$$E = Z_{\alpha/2} * \frac{\sigma}{\sqrt{n}}$$

Dicho procedimiento fue trasladado al lenguaje de programación Python con el fin de poder procesar la información dispuesta en las bases de datos de manera agil y eficaz.

```
import pandas as pd
import numpy as np
import math
#Cargue del archivo de información correspondiente
data=pd.read_excel("Grupo_Errores_DIVIPOLA.xlsx")
#Toma de las columnas de interés
df=pd.DataFrame(data,columns=['Codigo_Banco_Reporte','Grupo1↩
   '])
df['Grupo1']=[float(i) for i in df['Grupo1']]
pd.set_option('display.max_rows',df.shape[0]+1)
#toma de los valores de la columna como valor float
#Extracción de los valores estadísticos como suma total, ↩
   media y desvición estándar
Total=df["Grupo1"].sum()
#print("Total",Total)
media=df["Grupo1"].mean()
#print("media", media)
sd=df["Grupo1"].std()
#Se debe extraer el total de los datos para incluirlos en la↩
    población total.
poblacion=df['Grupo1'].count()
print("Población Total:", poblacion)
#Para mas información sobre los datos, revisar
#print(df["Valores"].describe())
#Extracción del tamaño de la muestra a partir del tamaño de ↩
   la población dada
n=((1.96)**2*(0.5)*(0.5)*(poblacion))/((0.01)**2*(poblacion↩
   -1)+(1.96)**2*(0.5)*(0.5))
print('tamaño muestra',n)
#a partir de la muestra generada se calcula el error máximo ↩
```

```
   admisible
EE=(1.96)*(sd/math.sqrt(n))
print("Error Máximo Admisible:",EE*100)
```

Ahora bien, teniendo en cuenta que nuestra población total es 354 datos, como resultado del cálculo estadístico se encontró una muestra de 341 datos, sobre la cual, imponiendo un error de precisión sobre el muestreo aleatorio de un 1 %, el error máximo admisible encontrado por el programa es de un $2,67\,\%$. Sin embargo, cabe aclarar que a medida que la cantidad de error sobre el muestreo aleatorio sea aumentado, el tamaño de la muestra será cada vez mas reducido y de igual forma el error máximo admisible irá creciendo.

```
>>>
Población Total: 354
tamaño muestra 341.449834287436
Error Máximo Admisible: 2.670971211614465
<<<
```

4. Evaluación y Categorización de Riesgos

4.1. Evaluación

Como ya mencionamos anteriormente, para la realización de un análisis a la calidad de la información efectivo se establecieron una serie de inconsistencias sobre las cuales los bancos podrían incurrir o no; esta serie de errores se articularon por medio de Power BI como validador y a su vez se dividieron en seis(6) grupos los cuales denominaremos criterios de evaluación

- Código DIVIPOLA
- Nombre del titular de la Cuenta Maestra Pagadora
- Nomenclatura de la Cuenta Maestra Pagadora
- Tipo de Cuenta bancaria de la Cuenta Maestra Pagadora.
- Tipo de Cuenta Maestra Pagadora no contemplado en el anexo técnico
- Número de Convenio

Ahora bien, entendiendo estos criterios como variables cualitativas, se realizó un estudio integral con el equipo de trabajo del Ministerio para determinar el impacto de cada una de ellas sobre la calidad de la información reportada a partir del conocimiento empírico adquirido a través del tiempo con los establecimientos bancarios.

De esta forma, se estableció un porcentaje a cada uno de estos criterios, teniendo en cuenta además el número de items que se evaluan en cada uno de estos.

Tabla 4.1

1. Criterios de Evaluación

Criterios	Porcentaje
Código DIVIPOLA	15 %
Nombre del titular de la Cuenta Maestra Pagadora	50 %
Nomenclatura de la Cuenta Maestra Pagadora	25 %
Tipo de Cuenta bancaria de la Cuenta Maestra Pagadora	2,5 %
Tipo de Cuenta Maestra Pagadora no contemplado en el anexo técnico	5 %
Número de Convenio	2,5 %

Para determinar la valorización de los criterios se asignan los errores calificables dentro de un rango teniendo en cuenta que el porcentaje de error máximo proporcionado por el evaluador es 3.5

Tabla 4.2

2. Categorización de Riesgos

Categorización	Valor Calificable	
Muy Bajo	1	**Riesgo Bajo**
Bajo	2	
Medio	3	**Riego Medio**
Alta	4	**Riesgo Alto**
Muy Alta	5	

4.2. Evaluación por Criterio

Dentro del análisis de los distintos errores producto del parámetro de validación establecido en Power BI, extraemos los resultados por cada criterio para la información reportada por un Establecimiento Bancario cualquiera, de esta forma consideraremos muy bajo o bajo (1 o 2) al error encontrado entre 0 % y 25 %. Para un error entre 26 % y 74 % se indica un valor de Medio (3) y para un valor de alto o muy alto (4 o 5) entendemos al porcentaje de error comprendido entre 75 % y 100 %.

4.3. Evaluación General

Frente a lo anterior, procedemos a calcular el resultado final considerando el valor de la variable o criterio según su impacto en el riesgo a la calidad de información dispuesto en la Tabla 1 junto con los resultados particulares sobre los criterios de evaluación encontrados en la información reportada de un banco cualquiera sobre lo cual, teniendo en cuenta el riesgo dispuesto en la Tabla 2 podemos realizar una categorización del riesgo para cada banco. De esta forma haremos uso de la siguiente fórmula para establecer el resultado requerido.

$$Error1*(15\,\%)+Error2*(50\,\%)+Error3*(25\,\%)+Error4*(2{,}5\,\%)+Error5*(5\,\%)+Error6*(2{,}5\,\%) \tag{4.1}$$

Ahora bien, para poner en práctica lo anterior se dispuso de los resultados obtenidos a partir de los datos reportados en un periodo de tiempo determinado para la totalidad de los bancos que tienen abiertas Cuentas Maestras Pagadoras, con lo cual se construyó la siguiente tabla:

Tabla 4.3

3. Errores por Banco

Criterios	Porcentaje de Error por Banco					
	Banco 1	**Banco 2**	**Banco 3**	**Banco 4**	**Banco 5**	**Banco 6**
1	1 %	3 %	4 %	2 %	3 %	6 %
2	99 %	23 %	89 %	98 %	80 %	60 %
3	14 %	2 %	7 %	8 %	72 %	71 %
4	0 %	0 %	0 %	0 %	34 %	0 %
5	0 %	0 %	0 %	0 %	0 %	0 %
6	0 %	0 %	26 %	0 %	0 %	0 %

Tabla 4.4

4. Errores por Banco 2

Criterios	Porcentaje de Error por Banco		
	Banco 7	**Banco 8**	**Banco 9**
1	9 %	12 %	80 %
2	26 %	70 %	89 %
3	1 %	2 %	7 %
4	0 %	0 %	0 %
5	0 %	0 %	0 %
6	0 %	0 %	26 %

Realizando la categorización de los criterios correspondientes a los errores en los que incide cada banco obtenemos una calificación del 1 al 5 para cada uno de estos porcentajes y posteriormente haciendo uso de la ecuación (1) encontramos que:

Tabla 4.5

3. Categorización de Riegos en Bancos

Bancos	Categorización del Riesgo
Banco1	Medio
Banco2	Bajo
Banco3	Medio
Banco4	Medio
Banco5	Alto
Banco6	Medio
Banco7	Bajo
Banco8	Medio
Banco9	Bajo

5. Python para la validación de Información Reportada

Debido a las extensas bases de datos que se reciben mensualmente en el Ministerio, se hace necesario plantear métodos de validación de la información para verificar que esta sea coherente y esté completa según lo indica el Anexo Técnico dispuesto en la resolución que reglamenta las Cuentas Maestras Pagadoras, lo anterior con el fin de optimizar el tiempo de trabajo y de igual forma plantear las bases para el desarrollo progresivo de modelos de validación de datos que den un alcance mayor a análisis requerido. Para este fin, se hizo uso de lenguaje de programación: Python, por medio del cual se desarrollaron dos operadores de validación articulados en una ventana gráfica de escritorio

5.1. Método de Validación 1: Función Ejecutar

Este método consiste en el cruce de información dispuesta en el Registro Tipo 1 (Fuente de información, periodo reportado, conteo del total de la información que viaja) y el Registro Tipo 3 (Movimientos de las Cuentas) con el propósito de corroborar que las fechas registradas en los movimientos de las cuentas en el Tipo 3 estén contenidas en el periodo de reporte indicado en el Tipo 1. Frente a lo anterior se configuraron alertas de riesgo en caso de que se detecten inconsistencias y así mismo permitiendo la visualización de las cuentas que presentan esta situación a través de un archivo generado en formato Excel.

5.2. Método de Validación 2: Función Verificar Saldos

Este método consiste en la verificación de la operación aritmética

$$(Saldo\ Inicial + Ingresos) - Egresos = Saldo\ Final \tag{5.1}$$

dentro de los movimientos de las cuentas en el periodo de tiempo reportado, configurando alertas de riesgo en caso de que la ecuación (4.1) no se satisfaga y así mismo permitiendo la visualización de las cuentas que presentan esta situación a través de un archivo generado en formato Excel.

5.3. Código Python Comentado

```
#Importamos los paquetes requeridos
from os import name, startfile
import pandas as pd
```

```
import numpy as np
import datetime as dt
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox as MessageBox
from pandas.core.reshape.merge import merge
from PIL import Image, ImageTk

# GUI for dataframe tkinter

#Usamos clases para tener todas las variables globales
class Ventanita( Tk ):
    def __init__(self):
        super().__init__()

        self.title("Validación Reportes")#Nombre ventana
        self.geometry("400x400")#Tamaño ventana
        self.image = Image.open("C:\\Users\\jppri\\Downloads↩
           \\Fondo.gif")
        self.img_copy= self.image.copy()
        self.img1 = ImageTk.PhotoImage(self.image)
        self.img = Label(self, image=self.img1)
        self.img.pack(fill=BOTH, expand=YES)
        self.img.bind('<Configure>', self._resize_image)

        self.tlabel1 = StringVar()
        self.label1 = Label(self, textvariable=self.tlabel1)↩
           # label
        self.label1.place(x=110 , y= 40, width=95, height↩
           =25)#Ubicacion y tamaño de label

        self.tlabel2 = StringVar()
        self.label2 = Label(self, textvariable=self.tlabel2)↩
           #label
        self.label2.place(x=110, y= 70, width=95, height=25)↩
           #Ubicacion y tamaño
```

```
        self.tlabel3 = StringVar()
        self.label3 = Label(self, textvariable=self.tlabel3)↩
           #label
        self.label3.place(x=90 , y= 230, width=200, height↩
           =30)#ubicacion y tamaño

        self.tlabel4 = StringVar()
        self.label4 = Label(self, textvariable=self.tlabel4)↩
           #label
        self.label4.place(x=240 , y= 300, width=40, height↩
           =40)

        self.tlabel5 = StringVar()
        self.label5 = Label(self, textvariable=self.tlabel5)↩
           #label
        self.label5.place(x=110 , y= 100, width=210, height↩
           =25)#ubicacion y tamaño

        self.etiq1 = Label(self, text="Fecha Inicial:", ↩
           relief="flat") #Boton con etiqueta
        self.etiq1.place(x=20, y= 40, width=80, height=25)
        self.etiq2 = Label(self, text="Fecha Final:")
        self.etiq2.place(x=20 , y= 70, width=80, height=25)
        self.etiq3=Label(self, text="Banco:", relief="flat")
        self.etiq3.place(x=20 , y= 100, width=80, height=25)

#Boton cargar archivo, con respectivo comando, tamaño y ↩
   ubicacion en ventana
        self.boton1 = Button(self, command=self.↩
           cargarArchivo, text="Cargar Archivo", relief="↩
           raised", borderwidth=5, bg="#F0C130")
        self.boton1.place(x=230 , y= 40, width=120, height↩
           =30)
#Boton ejecutar, con respectivo comando, tamaño y ubicacion ↩
   en ventana
```

```
        self.boton2 = Button(self, command=self.Ejecutar, ↩
           text="Ejecutar", relief="raised", borderwidth=5)
        self.boton2.place(x=50 , y= 170, width=160, height↩
           =50)
#Boton errores, con respectivo comando, tamaño y ubicacion ↩
   en ventana
        self.boton3 = Button(self, command=self.↩
           Cargar_Errores, text="Cargar Errores", relief="↩
           raised", borderwidth=5)
        self.boton3.place(x=220 , y= 170, width=160, height↩
           =50)
#Boton para imprimir error en los saldos
        self.boton4=Button(self,command=self.↩
           Verificar_Saldos, text="Validar Saldos", relief="↩
           raised", borderwidth=5)
        self.boton4.place(x=50 , y= 300, width=160, height↩
           =50)
#Boton descarga
        self.photo = PhotoImage(file = r"C:\Users\jppri\↩
           Downloads\down-arrow.png")
        self.boton5=Button(self, command=self.descargar, ↩
           image=self.photo, border=0,relief="flat")
        self.boton5.place(x=310 , y= 230, width=40, height↩
           =40)
        self.boton6=Button(self,command=self.Descargar_Saldo↩
           , image=self.photo, border=0, relief="flat")
        self.boton6.place(x=310, y= 300, width=40, height↩
           =40)

    #Función para ajuste de la imagen de fondo al tamaño de ↩
       la ventana
    def _resize_image(self, event):
            new_width = event.width
            new_height = event.height
            self.image = self.img_copy.resize((new_width, ↩
               new_height))
```

```
            self.img1 = ImageTk.PhotoImage(self.image)
            self.img.configure(image =  self.img1)

# Funcion para el primer boton
    def cargarArchivo(self):
    #hacemos uso de filedialog.askopenfile para indicar que ↩
       tipos de archivos podermos abrir para luego cargarlo ↩
       a nuestro codigo
        archivo = filedialog.askopenfilename(title="Cargar",↩
            initialdir="C:/", filetypes=(("Archivos de texto↩
           ",
        "*.txt"),("Archivos csv", "*.csv"),("Todos los ↩
           Archivos", "*.*")))
    #Para nuestro archivo de texto como este no indica un ↩
       nombre para cada columna, se los asignaremos por ↩
       defecto para poder acceder a cada una de ellas
        data = pd.read_csv(archivo, sep= ';', decimal=',', ↩
           names = ["Columna1", "Columna2","Columna3","↩
           Columna4","Columna5","Columna6","Columna7",
        "Columna8","Columna9","Columna10", "Columna11","↩
           Columna12","Columna13", "Columna14","Columna15"])
        self.df = pd.DataFrame(data)#convertimos nuestro ↩
           archivo plano en un DataFrame
    #Generamos un nuevo DataFrame esta vez filtrado por ↩
       registro tipo 1 del anexo de Cuentas Maestras ↩
       Pagadoras, y renombrasos sus columnas según el ↩
       mencionado anexo
        self.Registro_Tipo1 = self.df[self.df["Columna1↩
           "]==1]
        self.Registro_Tipo1 = self.Registro_Tipo1.rename(↩
           columns={"Columna1":"CPT1_TipoRegistro", "↩
           Columna2":"CPT1_TipoIdBanco",
        "Columna3":"CPT1_No.IdBanco", "Columna4":"↩
           Dia_InicioPeriodoReporte", "Columna5":"↩
           Dia_FinPeriodoReporte", "Columna6":"↩
           Total_Registros"})
```

```
#Generamos un nuevo DataFrame esta vez filtrado por ↩
   registro tipo 2 del anexo de Cuentas Maestras ↩
   Pagadoras, y renombramos sus columnas según el ↩
   mencionado anexo
    self.Registro_Tipo2 = self.df[self.df["Columna1↩
       "]==2]
    self.Registro_Tipo2 = self.Registro_Tipo2.rename(↩
       columns={"Columna1":"CPT2_TipoRegistro", "↩
       Columna2":"CPT2_ConsecutivoRegistro",
    "Columna3":"Codigo_Departamento", "Columna4":"↩
       Codigo_Municipio", "Columna5":"↩
       CPT2_NombreEntidadTerritorial", "Columna6":"↩
       CPT2_TipoIdEntidadTerritorial",
    "Columna7":"CPT2_NumeroIdEntidadTerritorial", "↩
       Columna8":"CPT2_DigitoVerificacionNIT_ET","↩
       Columna9":"CPT2_TipoCuentaMaestraPagadora", "↩
       Columna10":"CPT2_NombreCuentaMaestraPagadora",
    "Columna11":"CPT2_IndicadorRegistro", "Columna12":"↩
       CPT2_NumeroConvenio", "Columna13":"↩
       CPT2_FechaConvenio", "Columna14":"↩
       CPT2_NumeroCuentaMaestraPagadora",
    "Columna15":"CPT2_TipoCuentaBancaria"})

#Generamos un nuevo DataFrame esta vez filtrado por ↩
   registro tipo 2 del anexo de Cuentas Maestras ↩
   Pagadoras, y renombramos sus columnas según el ↩
   mencionado anexo
    self.Registro_Tipo3 = self.df[self.df["Columna1↩
       "]==3]
    self.Registro_Tipo3 = self.Registro_Tipo3.rename(↩
       columns={"Columna1":"CPT3_TipoRegistro", "↩
       Columna2":"CPT3_ConsecutivoRegistro",
    "Columna3":"Codigo_Departamento", "Columna4":"↩
       Codigo_Municipio", "Columna5":"↩
       CPT3_TipoCuentaMaestraPagadora", "Columna6":"↩
```

```
        CPT3_NumeroCuentaMaestraPagadora",
     "Columna7":"CPT3_FechaMovimiento", "Columna8":"↩
        CPT3_TipoRegistroMovimiento","Columna9":"↩
        CPT3_DescripcionMovimiento", "Columna10":"↩
        CPT3_ValorRegistroDetalle",
     "Columna11":"CPT3_TipoIdBeneficiario", "Columna12":"↩
        CPT3_NumeroIdBeneficiario", "Columna13":"↩
        CPT3_NombreBeneficiario", "Columna14":"↩
        CPT3_CodigoServicio",
     "Columna15":"CPT3_NumeroCUS"})

# Extraemos la fecha inicio y fecha fin del periodo ↩
   indicado en regitro tipo 1 con iloc
    self.fecha_inicio = self.Registro_Tipo1.iloc[0]["↩
       Dia_InicioPeriodoReporte"]
    self.fecha_final  = self.Registro_Tipo1.iloc[0]["↩
       Dia_FinPeriodoReporte"]
#Convertimos las fechas inicio y fin del tipo 1 en ↩
   formato de fecha datetime para que python las ↩
   reconozca
    self.start = dt.datetime.strptime( self.fecha_inicio↩
       ,"%Y-%m-%d")
    self.start = self.start.date()# con .date() ↩
       eliminamos el formato de Horas Minutos y segundos↩
        que no necesitamos
    self.end = dt.datetime.strptime( self.fecha_final,"%↩
       Y-%m-%d")
    self.end = self.end.date()
    self.tlabel1.set(self.start)
    self.tlabel2.set(self.end)
    self.tabla=pd.DataFrame()
    idBancos↩
       =[860002964,860007738,890903937,890903938,860050750,↩

    860003020,890300279,860007335,860034313,860034594,
    800037800,860035827, 890200756]
```

```
        bancos=['BANCO DE BOGOTÁ','BANCO POPULAR S.A.','ITAÚ↩
            CORPBANCA COLOMBIA S.A.','BANCOLOMBIA','BANCO ↩
           GNB SUDAMERIS','BBVA COLOMBIA S.A.','BANCO DE ↩
           OCCIDENTE S.A.','BANCO CAJA SOCIAL',
        'BANCO DAVIVIENDA S.A.','SCOTIABANK COLPATRIA S.A↩
           .','BANCO AGRARIO DE COLOMBIA S.A.','BANCO AV ↩
           VILLAS','BANCO PICHINCHA S.A.']
        self.tabla["CPT1_No.IdBanco"]=idBancos
        self.tabla["RAZÓN SOCIAL DEL BANCO"]=bancos
        self.cruce=pd.merge(left=self.Registro_Tipo1, right=↩
           self.tabla, how='left',left_on="CPT1_No.IdBanco",↩
            right_on= "CPT1_No.IdBanco")
        self.cruce.shape
        self.tlabel5.set(self.cruce["RAZÓN SOCIAL DEL BANCO↩
           "][0])


#Función para verificar el periodo de las fechas reportadas

    def Ejecutar(self):

    #Extraemos la columna con las fechas de los movimientos ↩
       del registro tipo 3 y las convertimos en formato ↩
       datetime para que python los reconozca como fechas
        Movimientos= self.Registro_Tipo3
        Movimientos["CPT3_FechaMovimiento"] = pd.to_datetime↩
           (Movimientos["CPT3_FechaMovimiento"])
    #Generamos un dataframe vacio que luego iremos llenando
        Periodo_Reportado = pd.DataFrame()
    # generamos el rango de fechas con pandas desde nuestra ↩
       variable start hasta end ya definidas como datetime, ↩
       a una frecuencia de dias "D".
        rango = pd.date_range(self.start, self.end,freq="D")
    # generamos un rango de enteros determinado por el ↩
       cardinal del rango de fechas desde start hasta end, ↩
       esto nos permitirá asignar un número validador a ↩
```

```
   estas fechas
    x = range(len(rango))
#Llenamos nuestro dataframe vacio con las columnas "↩
   rango" y "x"
    Periodo_Reportado["CPT3_FechaMovimiento"] = rango
    Periodo_Reportado["x"] = x
#Ahora creamos un nuevo datafrane con las columnas que ↩
   necesitamos visualizar del registro tipo 3
    Movimientos=pd.DataFrame(Movimientos,columns=["↩
       CPT3_NumeroCuentaMaestraPagadora","↩
       CPT3_FechaMovimiento","↩
       CPT3_TipoCuentaMaestraPagadora", "↩
       CPT3_TipoRegistroMovimiento"])
#Hacemos uso de merged_left (unión a izquierda) para añ↩
   adir al dataframe del Registro tipo 3 la información ↩
   de la tabla "Periodo_Reportado" uniendolos
# por medio de la llave CPTE_FechaMovimiento
    merge_left = pd.merge(left=Movimientos, right=↩
       Periodo_Reportado, how='left',left_on="↩
       CPT3_FechaMovimiento", right_on= "↩
       CPT3_FechaMovimiento")
    merge_left.shape
#Ahora extraemos las fechas indicadas en la columna ↩
   CPT3_FechaMovimiento del dataframe Registro Tipo 3 ↩
   que al unirlas con las fechas de la tabla del "↩
   Periodo_Reportado"
#no tienen un numero validador(este numero existe si la ↩
   fecha corresponde a una fecha que este dentro del  ↩
   periodo de reporte) y llenamos un nuevo dataframe
    self.error=merge_left[pd.isnull(merge_left.x)]
#Realizamos el condicional que nos indica que si la ↩
   tabla "error" que se llena con las fechas que estan ↩
   fuera del periodo de reporte es vacia, genere un ↩
   mensaje de
# de información que nos confirma que no hay problemas ↩
   en las fechas de la información reportada. En caso ↩
```

```
       contrario generará un mensaje de alerta
        if self.error.empty: MessageBox.showinfo(message="Se↩
            reportan movimientos dentro del mes indicado.", ↩
           title="Validación Reportes")
        else:MessageBox.showwarning("Alerta!", "Reporte de ↩
           información en meses diferentes.")
    #Función para imprimir en ventana el número de errores ↩
       en la fecha y en caso contrario un mensaje de no ↩
       hallazgo de errores
    def Cargar_Errores(self):
        if self.error.empty:
            MessageBox.showinfo(message="No se encuentran ↩
               Errores.", title="Validación Reportes")
            self.tlabel3.set("Número de errores: 0")
        elif self.error is not np.empty:
            error1 = self.error.groupby(["↩
               CPT3_NumeroCuentaMaestraPagadora"]).size().↩
               reset_index(name="numero")
            print(error1)
            error2 = self.error.groupby(["↩
               CPT3_FechaMovimiento"]).size().reset_index(↩
               name="fechas")
            self.tlabel3.set("Número de errores: "+str(↩
               error2["fechas"][0]))
            print("contador=", error2)

# Función para descargar el archivo de las fechas con el ↩
   nombre del banco
    def descargar(self):
        self.error.insert(0,"Banco",self.cruce["RAZÓN SOCIAL↩
            DEL BANCO"][0])
        self.error.to_excel(r"ERROR_FECHAS_"+str(self.cruce↩
           ["RAZÓN SOCIAL DEL BANCO"][0])+".xlsx", index=↩
           False)

#Función para validar la operacion de saldo inicial+ ↩
```

```
ingresos -egresos=saldo final
 def Verificar_Saldos(self):
     self.Registro_Tipo3["↩
        CPT3_NumeroCuentaMaestraPagadora"]=self.↩
        Registro_Tipo3["CPT3_NumeroCuentaMaestraPagadora↩
        "].astype(float)
     self.Registro_Tipo3["CPT3_ValorRegistroDetalle"]=↩
        self.Registro_Tipo3["CPT3_ValorRegistroDetalle"].↩
        apply(lambda x: x.replace(',','.'))
     self.Registro_Tipo3["CPT3_ValorRegistroDetalle"]=↩
        self.Registro_Tipo3["CPT3_ValorRegistroDetalle"].↩
        astype(float)

     self.Registro_Tipo3.to_excel(r"data_saldos.xlsx", ↩
        index=False)
     self.df=pd.read_excel('data_saldos.xlsx')

     self.Saldos = {}
     for i in range(len(self.df)):
         if self.df["CPT3_NumeroCuentaMaestraPagadora"][i↩
            ] not in self.Saldos: self.Saldos[self.df["↩
            CPT3_NumeroCuentaMaestraPagadora"][i]] = [↩
            self.df["CPT3_NumeroCuentaMaestraPagadora"][i↩
            ], 0, 0, 0, 0, 0]
         tipo = self.df["CPT3_TipoRegistroMovimiento"][i]
         if tipo == 800 or tipo == 360:
             self.Saldos[self.df["↩
                CPT3_NumeroCuentaMaestraPagadora"][i]][1]↩
                 += self.df["CPT3_ValorRegistroDetalle"][↩
                i]
             self.Saldos[self.df["↩
                CPT3_NumeroCuentaMaestraPagadora"][i]][4]↩
                 -= self.df["CPT3_ValorRegistroDetalle"][↩
                i]
         elif tipo == 100 or tipo == 110 or tipo == 120 ↩
            or tipo == 500:
```

```
                self.Saldos[self.df["↩
                   CPT3_NumeroCuentaMaestraPagadora"][i]][2]↩
                    += self.df["CPT3_ValorRegistroDetalle"][↩
                   i]
                self.Saldos[self.df["↩
                   CPT3_NumeroCuentaMaestraPagadora"][i]][4]↩
                    += self.df["CPT3_ValorRegistroDetalle"][↩
                   i]
            elif tipo == 600: self.Saldos[self.df["↩
               CPT3_NumeroCuentaMaestraPagadora"][i]][3] += ↩
               self.df["CPT3_ValorRegistroDetalle"][i]
        for i in self.Saldos:
            if abs(self.Saldos[i][4] - self.Saldos[i][3]) <=↩
                0.000001: self.Saldos[i][5]= True
            else: self.Saldos[i][5]= False
        self.valide = pd.DataFrame.from_dict(self.Saldos)
        self.valide.index = ["Cuenta", "Egreso", "Ingreso", ↩
           "Final", "Ingreso - Egreso", "Se Corresponden"]
        self.valide = self.valide.T
        self.valide.reset_index(drop=True, inplace=True)
        Error_Saldo=self.valide[self.valide["Se Corresponden↩
           "]==False]
        if Error_Saldo.empty:
            MessageBox.showinfo(message="Los saldos se ↩
               corresponden", title="Validación Reportes")
        else: MessageBox.showwarning("Alerta!", "Operación ↩
           de Saldos no es Coincidente!.")
        print(Error_Saldo)
        if Error_Saldo is not np.empty:
            self.Saldos_Tabla=pd.DataFrame(Error_Saldo)
        self.tlabel4.set(Error_Saldo["Se Corresponden"].↩
           count())
#Función para descargar el archivo con la información de las↩
    cuentas que no satisfacen la operación mencionada
    def Descargar_Saldo(self):
        self.Saldos_Tabla.insert(0,"Banco",self.cruce["RAZÓN↩
```

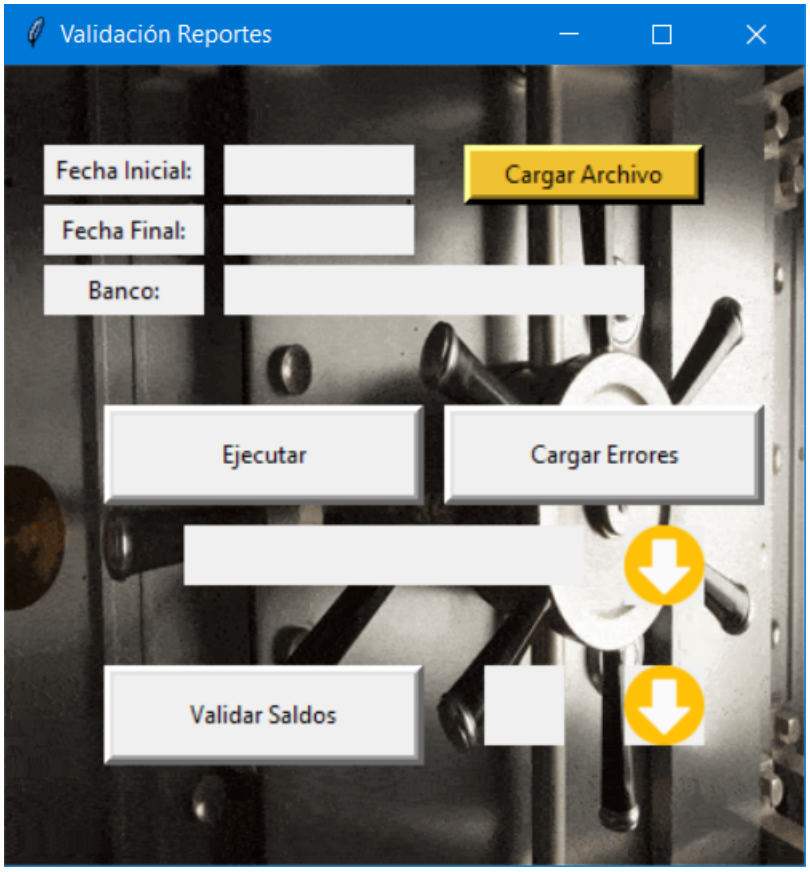
```
            SOCIAL DEL BANCO"][0])
        self.Saldos_Tabla.to_excel(r"ERROR_SALDOS_"+self.↩
           cruce["RAZÓN SOCIAL DEL BANCO"][0]+".xlsx", index↩
           =False)

Ventana = Ventanita()
Ventana.mainloop()
```

Como resultado del anterior código se obtuvo la siguiente aplicación de escritorio, la cual permite el cargue directo del archivo que se requiera validar, a través del botón Cargar Archivo y el acceso a la fecha final e inicial indicada en el Registro Tipo 1, así como la búsqueda del NIT que permite conocer el nombre del banco del cual se remitió el archivo plano:

Figura 5.1

Ventana de Validación de Reportes



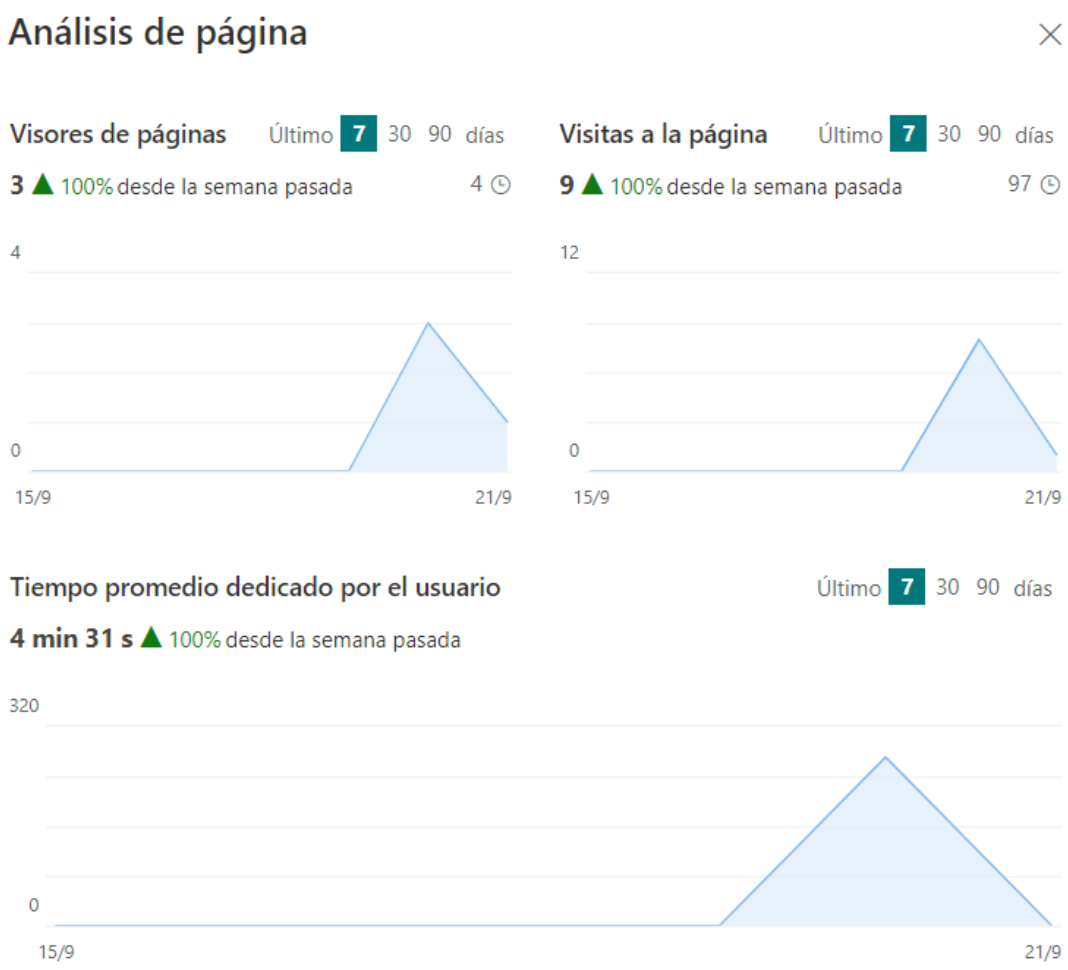
6. Blog Cuentas Maestras

El blog de Cuentas Maestras del Ministerio de Hacienda y Crédito Público nació como una propuesta para establecer un vinculo entre los conocimientos sobre el manejo de bases de datos necesarios dentro de los distintos equipos de trabajo del Ministerio y los conocimientos adquiridos durante la carrera de Matemáticas, todo esto con el fin de maximizar las herramientas a emplear para el mejor uso y manejo de la información y potenciar los análisis que sobre estos se hagan.

En este sentido, el blog implementado a través de la plataforma Share Point del Ministerio consiste en la trasmisión de conceptos básicos sobre temas como el análisis multivariado de datos, mapas de calor y teoría de grafos, asociando cada uno de estos con el análisis de datos sirviéndose de la información proporcionada por el equipo de Cuentas Maestras.

Figura 6.1

Visitas al Blog Cuentas Maestras



7. Resultados y Conclusiones

En primer lugar, frente al trabajo realizado durante el tiempo de pasantía se concluye que fue posible integrarse y acoplarse a las dinámicas del equipo de Cuentas Maestras y a su vez entender de la mejor manera los temas tratados, por lo cual se logró desde las matemáticas y la programación contribuir a la mejora de los procesos de análisis y revisión sobre la calidad de la información. De esta manera, me permito exponer los resultados y comentarios sobre los trabajos realizados:

7.0.1. Sobre el Capitulo 2

Respecto al trabajo realizado en el hallazgo del error estándar se concluyó que para una población total bien conocida y a su vez no muy grande, es conveniente establecer un error del muestreo aleatorio mínimo para contar con la mejor precisión, si es que no se quiere tomar el total de los datos. De tal parte, se concluye que el error máximo admisible calculado se ajusta a los estándares de calidad requeridos por el Ministerio y por lo tanto, se recomienda establecer el mismo estudio para el total de los errores evaluados a través de los cinco (5) criterios restantes generados por el equipo de trabajo, para conocer los errores máximos admisibles para cada uno de ellos.

7.0.2. Sobre el Capitulo 3

Frente a la categorización y evaluación de los riesgos para cada banco, se entiende que el proceso de ponderación y clasificación de los resultados de los errores en términos cuantitativos y cualitativos permite visualizar el estado real de los bancos frente a la calidad de la información reportada. De este modo, este trabajo permite enfatizar los esfuerzos de capacitación, acompañamiento y mejora a los establecimientos bancarios que se encuentran en riesgo alto, incluso medio y así mismo acercar a la perfección, que es lo que se busca, a aquellos bancos es riesgo bajo.

7.0.3. Sobre el Capitulo 4

En cuanto a la validación de la información reportada, se concluye que poner en marcha el desarrollo de estructuras lógicas y procesos matemáticos a través del lenguaje de programación Python (o cualquier lenguaje de programación con las herramientas necesarias) permite una notable reducción en los tiempos de revisión de información y de igual forma permite llevar a cabo un análisis mas detallado y preciso, así como contemplar y ejecutar un mayor número de procesos de validación que posibilita una mayor rigurosidad y un mejor control sobre los recursos públicos administrados a través de las Cuentas Maestras y sus Cuentas Maestras Pagadoras.

7.0.4. Sobre el Capitulo 5

Finalmente, respecto al Blog de Cuentas Maestras, se reconoce la necesidad de trasmisión de conocimientos que sirven de herramientas para enfrentarse al mejor tratamiento de las inmensas bases de datos que recibe el Ministerio, por lo cual este blog se establece como un medio amable e inclusivo para no solo abarcar la atención de los diferentes equipos del Ministerio, sino que reta a los matemáticos para buscar la mejor manera de trasmitir de la mejor manera los conocimientos adquiridos y a su vez relacionarlos con los temas de interés de esta Entidad Pública que incide enormemente en los sectores económicos, gubernamentales y políticos; y gestiona los recursos públicos de la Nación.

A. Anexo: Titulo del anexo A

FECHA	10/09/2021

No. de Seguimiento	2

NOMBRE COMPLETO PRACTICANTE		Laura Katherine Pira Rincón
C.C. [X] **C.E .** [ ]	**No**	1.022.437.096
INSTITUCIÓN DE EDUCACIÓN SUPERIOR		Universidad Distrital Francisco José de Caldas
PROGRAMA ACADÉMICO		Matemáticas
NOMBRE DEL TUTOR- MHCP		Fernando Olivera Villanueva
NOMBRE DEL MONITOR - UNIVERSIDAD		Carlos Orlando Ochoa Castillo
FECHA INICIO DE LA PRÁCTICA		12/02/2021
FECHA TERMINACIÓN PREVISTA		**10/09/2021**
DEPENDENCIA A LA QUE PERTENECE		Dirección General de Apoyo Fiscal

ACTIVIDADES DEL PLAN DE PRÁCTICA	% DE CUMPLIMIENTO	PRODUCTO ENTREGADO	OBSERVACIONES
Elaborar un código por medio de un lenguaje de programación con el fin de generar operaciones de validación para detectar errores e inconsistencias en la información que viaja al Ministerio optimizando el tiempo de trabajo del equipo enfocado en la revisión y detección de estos problemas	**100%**	Se entregaron una aplicación en lenguaje Python que permite validar el estado de los reportes de Cuentas Maestras en lo que respecta a fechas de periodo de reporte, validación aritmética y control de pago de rendimientos. Se documenta al APP y se deja como producto en repositorio de Cuentas Maestras para uso cotidiano.	Se establece la aplicación como vehículo fundamental para la validación mensual de los reportes entregados por los Bancos. El desarrollo en fase y con versionamiento permitió fomentar buenos hábitos del proceso de desarrollo.
Elaborar una propuesta de un Blog de Cuentas Maestras por medio del cual se pretenda generar un lazo entre los conocimientos adquiridos en la carrera de Matemáticas con los temas de interés del Ministerio y el Equipo de Cuentas Maestras en el uso de bases de datos.	**100%**	Se entregó y presentó a manera de producción un Blog con tres entradas de temas de analítica y matemática aplicada, para fomentar el conocimiento en los miembros de los diferentes grupos de la Dirección General de Apoyo Fiscal	La versión de consulta de las entradas del blog se encuentra habilitada en SharePoint sin embargo pasará a producción junto con el Boletín de Noticias.

ACTIVIDADES DEL PLAN DE PRÁCTICA	% DE CUMPLIMIENTO	PRODUCTO ENTREGADO	OBSERVACIONES
Elaboración de presentaciones para capacitaciones a entidades territoriales y/o establecimientos bancarios	**100%**		
Consolidación de base de Datos de embargos aplicados a Cuentas Maestras.	**100%**	Se realizó el cargue de dos (2) oficios remitidos por el Banco Agrario de Colombia.	
Acompañar el proceso de implementación de la FASE II del modelo analítico de Cuentas Maestras.	**100%**		La pasante ha acompañado los diferentes espacios con la Dirección de Tecnología y el proveedor IDATA, adicionalmente, se acordó un espacio diario de seguimiento al modelo analítico

CONCEPTO DEL CUMPLIMIENTO A LA FECHA DEL SEGUIMIENTO

BUENO	REGULAR	DEFICIENTE
X		

OBSERVACIONES ADICIONALES

Se resalta el trabajo realizado por la pasante, quien con dedicación y empeño realizó las tareas asignadas entendiendo desde su marco de conocimiento las dificultades de análisis de información del grupo, siempre se mostro participativa y dispuesta a aprender, así como a compartir su conocimiento con el resto del equipo.

Se destaca de su labor en la pasantía la capacidad de articular el mundo de las matemáticas con el tema financiero y el normativo, dejando un plan de trabajo para las siguientes pasantías.

APRUEBAN

NOMBRE DEL TUTOR

NOMBRE DEL MONITOR

Referencias

[1] Documentación de Python - 3.9.7. (s.f.). Descargado de https://docs.python.org/es/3/ (Library Catalog: Documentación Oficial de Python)

[2] Resolución 2248 de 2018. (s.f.). Descargado de https://www.mineducacion.gov.co/1759/articles-360925_Documentos_02.pdf (Library Catalog: Biblioteca Digital)

[3] Resolución 4835 de 2015. (s.f.). Descargado de https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/INEC/IGUB/resolucion-4835-de-2015-min-hacienda.pdf (Library Catalog: Biblioteca Digital)

[4] Resolución No. 0660 de 2018. (s.f.). Descargado 2018-03-09, de https://normas-apa.org/estructura/introduccion/ (Library Catalog: Biblioteca Digital)

[5] William Mendenhall, Robert J. Beaver, Barbara M. Beaver. (2006). Introducción a la probabilidad y estadística. , *13*, 300-324.