	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	1 de 25

I. Objetivo del informe.

El presente documento describe los detalles de la implementación de la aplicación web sobre los reportes bancaria en la aria de contabilidad en la empresa Ensa.

II. Proceso y resultado del cambio en la infraestructura.

A. Tecnología utilizada

La aplicación web se ha desarrollado utilizando la tecnología Python versión 3.11.3 usando el marco Flask es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Flask se basa en la biblioteca WSGI Werkzeug, que proporciona herramientas para manejar solicitudes y respuestas HTTP, así como otras utilidades útiles para el desarrollo web.

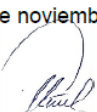
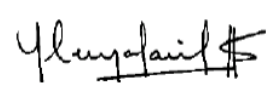
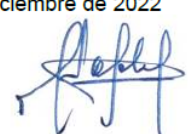
También hemos utilizado Pandas es una biblioteca de código abierto de Python que proporciona estructuras de datos y herramientas de análisis de datos para la modificaciones en los datos del Excel.


En otras tecnologías hemos utilizado Openpyxl es una biblioteca de código abierto de Python que permite leer, escribir y manipular archivos de Excel en formato.xlsx. Y la biblioteca Openpyxl, Workbook es una clase que representa un libro de Excel, es decir, un archivo .xlsx. Un libro de Excel puede contener una o más hojas de cálculo, cada una con sus datos,

B. Ubicación del código fuente

El código fuente se encuentra ubicado en Github y se ha creado una organización exclusiva para dicho fin:

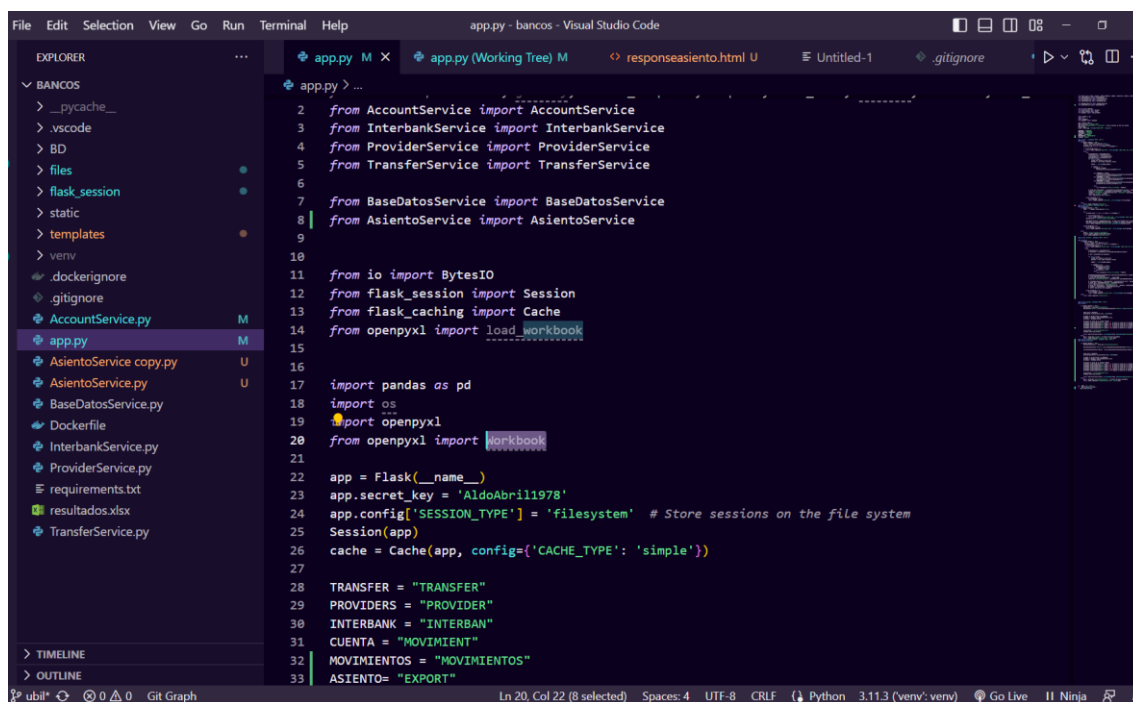
<https://github.com/Electronorte-SA/NetFile.git>

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	2 de 25

C. Edición del código fuente

- ❖ Tener Visual Studio Code en la maquina local
<https://code.visualstudio.com>
- ❖ Clonar el código fuente a la maquina local, utilizando la opción de Visual Studio: Archivo -> Clonar repositorio.
- ❖ Para editar el código se recomienda utilizar Visual Studio Code



```

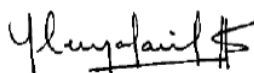
1  from AccountService import AccountService
2  from InterbankService import InterbankService
3  from ProviderService import ProviderService
4  from TransferService import TransferService
5
6
7  from BaseDatosService import BaseDatosService
8  from AsientoService import AsientoService
9
10
11  from io import BytesIO
12  from flask_session import Session
13  from flask_caching import Cache
14  from openpyxl import load_workbook
15
16
17  import pandas as pd
18  import os
19  import openpyxl
20  from openpyxl import Workbook
21
22  app = Flask(__name__)
23  app.secret_key = 'AldoAbril1978'
24  app.config['SESSION_TYPE'] = 'filesystem' # Store sessions on the file system
25  Session(app)
26  cache = Cache(app, config={'CACHE_TYPE': 'simple'})
27
28  TRANSFER = "TRANSFER"
29  PROVIDERS = "PROVIDER"
30  INTERBANK = "INTERBAN"
31  CUENTA = "MOVIMIENT"
32  MOVIMIENTOS = "MOVIMIENTOS"
33  ASIENTO = "EXPORT"

```

Elaborado por:
Felipe Hiromoto Hiromoto
Jefe Corporativo TIC
29 de noviembre de 2022




Revisado por:
Simeón Peña Pajuelo
Coordinador Corp. SIG
30 de noviembre de 2022



Aprobado por:
Lizardo Ojeda López
Gerente General (e)
01 de diciembre de 2022

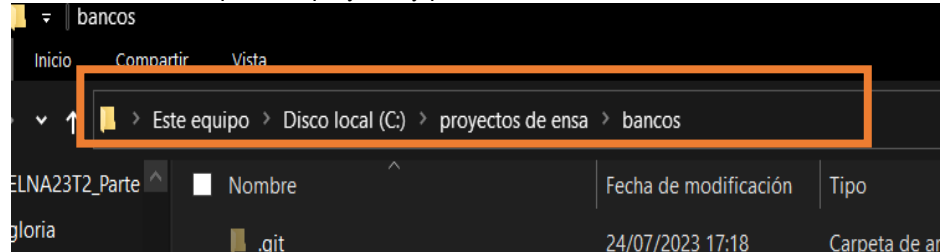


	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	3 de 25

D. Pasos para ejecutar en un entorno local

1. Activa `actíivate` en el entorno de `windns`

- ✓ Se debe ir en la carpeta del proyecto y poner en la barra de ruta CMD



- ✓ Abrimos en cmd y activamos el archivo `actíivate`, para ir a la carpeta `venv/scripts` y damos el comando `pip activate`

```
C:\proyectos de ensa\bancos\venv>cd Scripts
C:\proyectos de ensa\bancos\venv\Scripts>pip activate
```

- ✓ Instalamos las librería: `pandas`, `flask`, `openpyxl` y `Workbook`

```
C:\proyectos de ensa\bancos\venv\Scripts> pip install pandas
```

`pandas`

```
C:\proyectos de ensa\bancos\venv\Scripts> pip install Flask
```

`Flask`

`openpyxl`

```
C:\proyectos de ensa\bancos\venv\Scripts> pip install openpyxl
```

`Workbook`

```
C:\proyectos de ensa\bancos\venv\Scripts>pip install Workbook
```

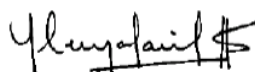
2. Abrimos el visual studios code en la ruta general del proyecto

```
C:\proyectos de ensa\bancos>code .
```

Elaborado por:
Felipe Hiromoto Hiromoto
Jefe Corporativo TIC
29 de noviembre de 2022




Revisado por:
Simeón Peña Pajuelo
Coordinador Corp. SIG
30 de noviembre de 2022

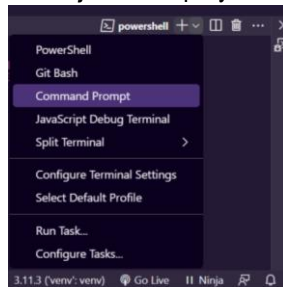


Aprobado por:
Lizardo Ojeda López
Gerente General (e)
01 de diciembre de 2022



	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	4 de 25

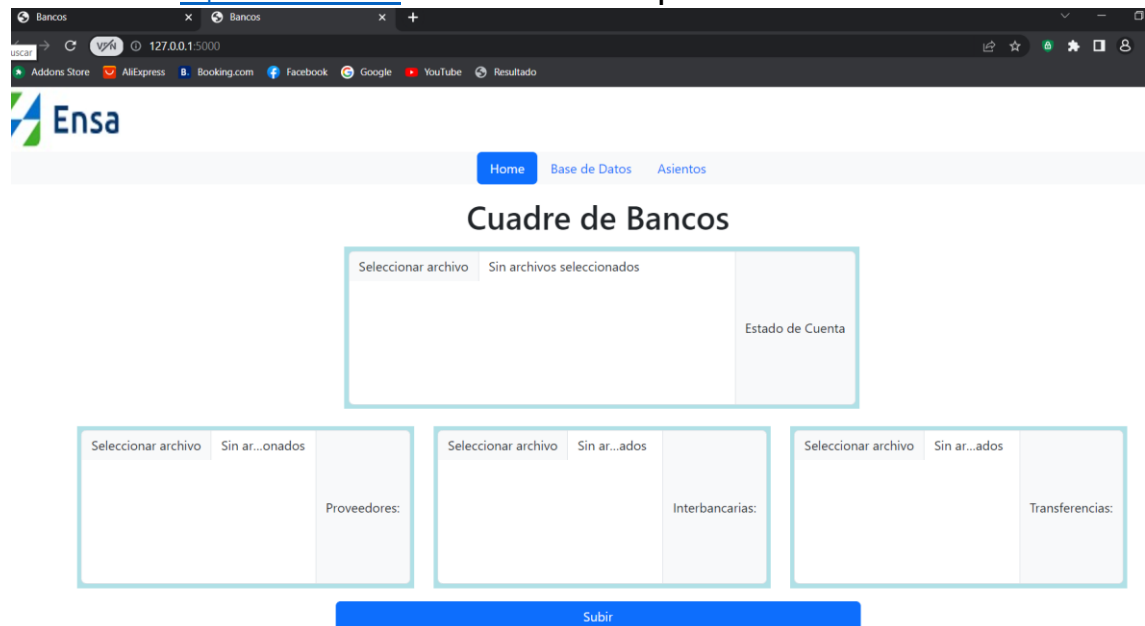
3. Para ejecutar el proyecto abrimos la consola


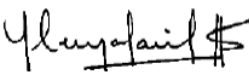




4. Por ultimo ejecutamos el comando python app.py

```
(venv) C:\proyectos de ensa\bancos>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 114-558-188
```

5. Abrimos la ruta <http://127.0.0.1:5000> con las teclas **ctrl + clip derecho**

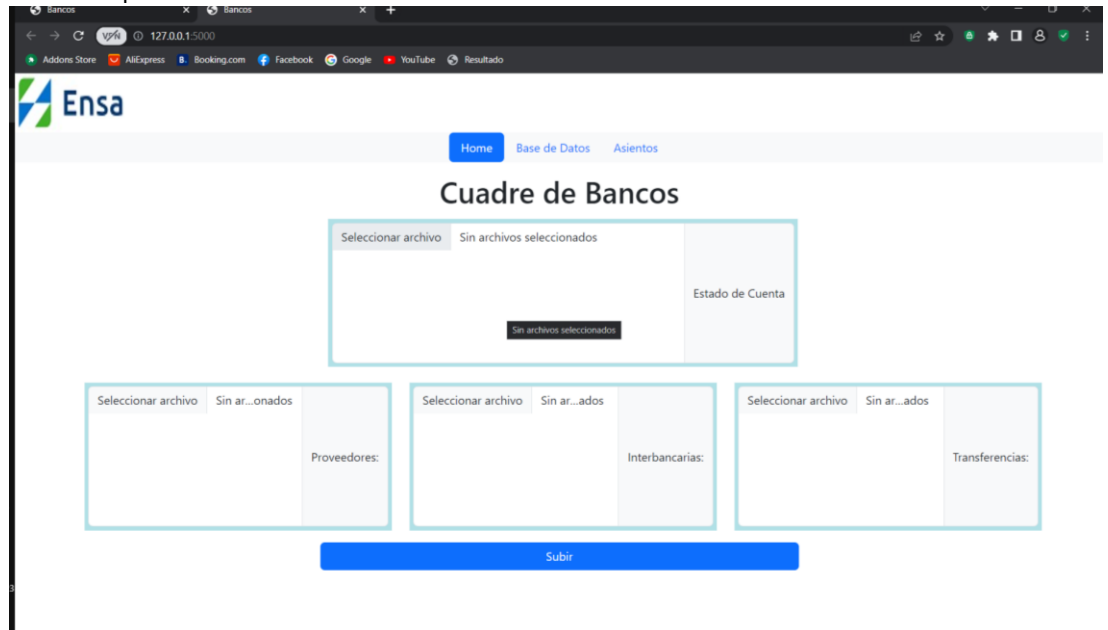


Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

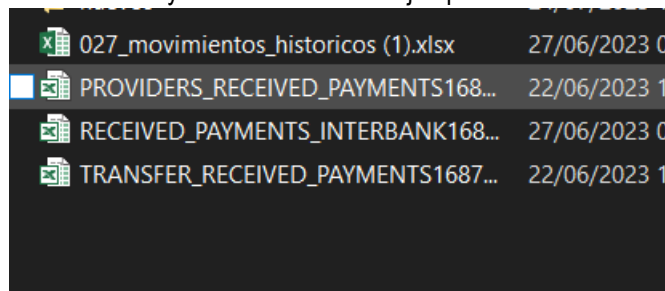
	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	5 de 25


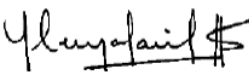

E. Manual de funcionamiento de la aplicación web


✓ Menú de opciones



Aquí podemos seleccionar cada archivo en Excel en cada respectivas casillas los archivos debe tener por mínimo los siguientes nombres: cuenta, proveedores, interbancarias y transferencias. Por ejemplo:

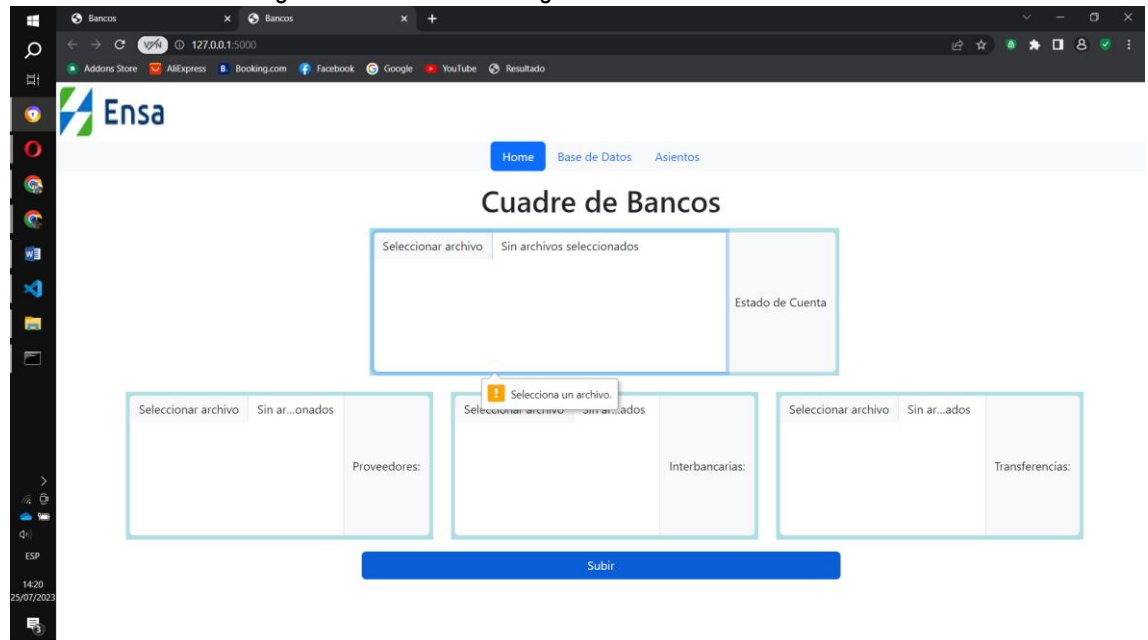


Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

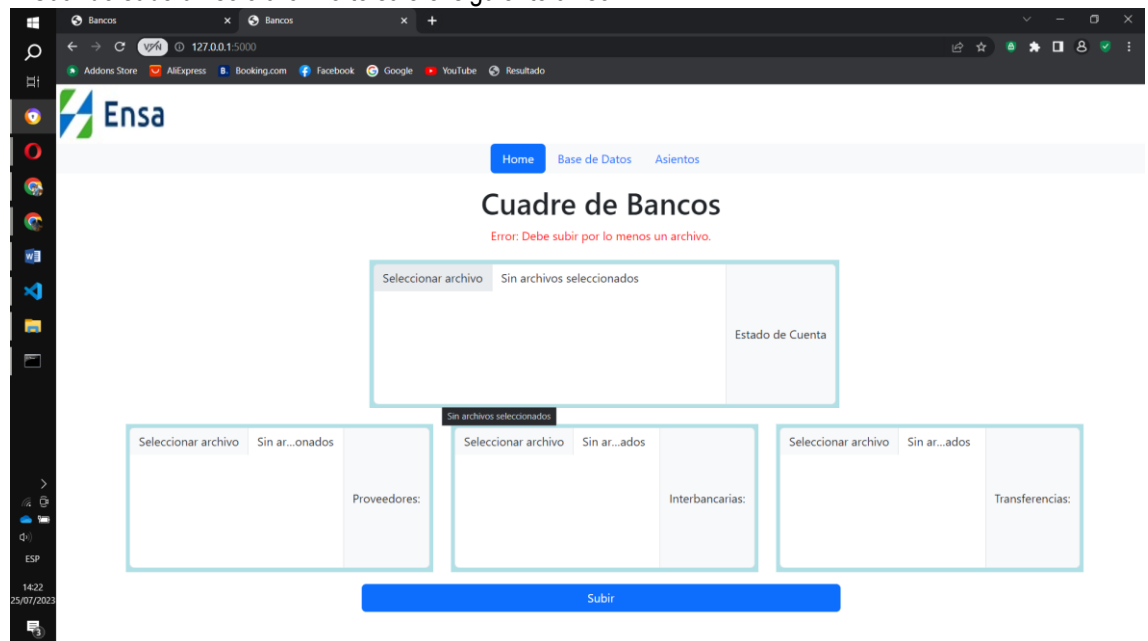
	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	6 de 25

✓ Retenciones de la aplicación web


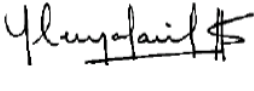

Cuando no colocas ninguno archivo te sale el siguiente aviso :




Cuando sube un solo archivo te sale el siguiente aviso:

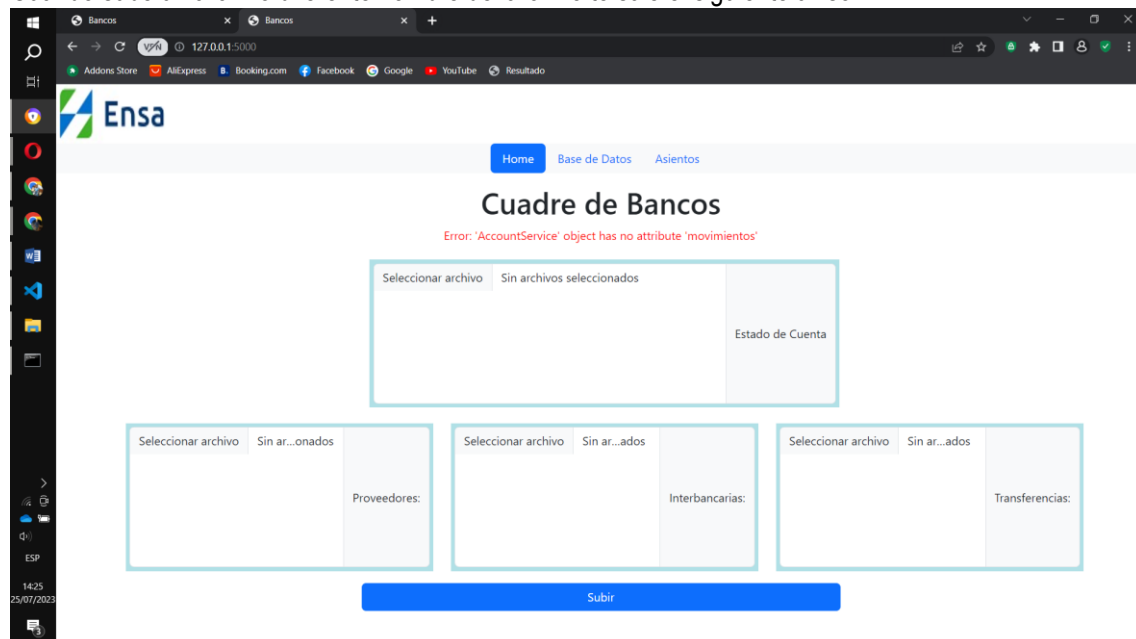


Debes subir mas de dos archivos Excel

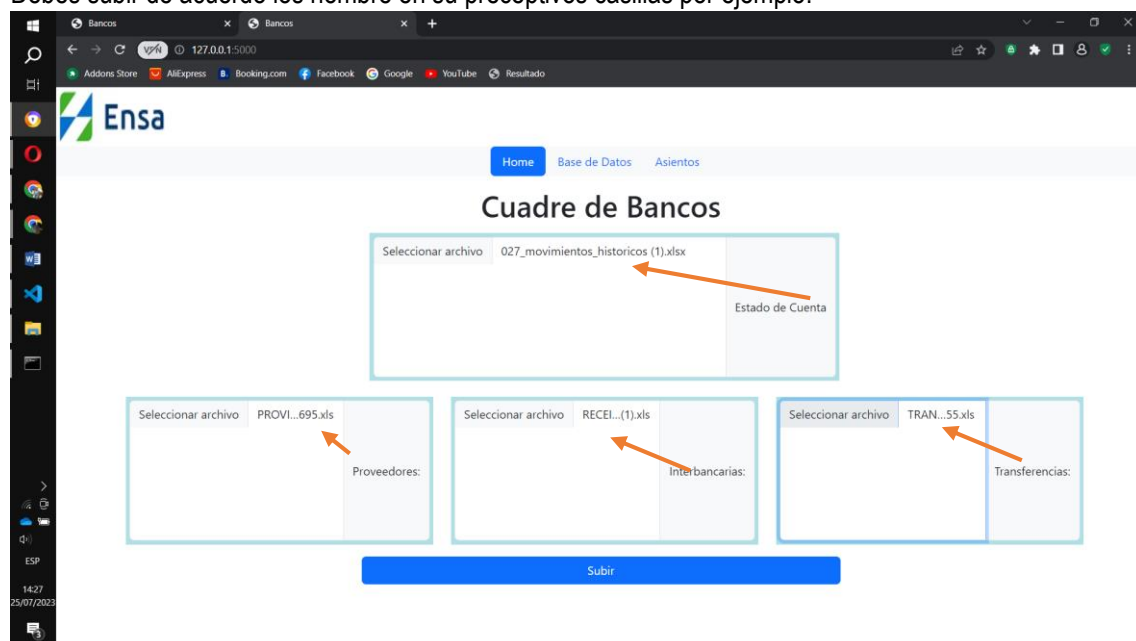
Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--


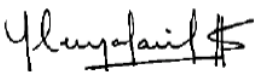

	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	7 de 25


Cuando sube un archivo diferente nombre del archivo te sale el siguiente aviso:



Debes subir de acuerdo los nombre en su preceptivos casillas por ejemplo:



Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	8 de 25

Además el Excel debe estar en la cabecera en la misma posición los 4 excel
Movimiento

	A	B	C	D	E	F	G	H
1	Cuenta	305-0037523-0-27 - EMP. REG. DE SERV. PUB. DE ELECT. DEL NORTE						
2	Moneda	Soles						
3	Tipo de Cuenta	Corriente						
4								
5	Fecha	Fecha valuta	Descripción operación	Monto	Saldo	Sucursal - agencia	Operación - Número	Operación - Hora
27	21/06/2023		VARIOS STATKRAFT PERU	63,894.38	5,000,231.23	111-008	08994377	15:41:34
28	21/06/2023		DE WESTERN UNION PERU	23,355.80	4,936,336.85	111-008	08973726	14:53:59

Proveedores

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Empresa	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.											
2													
3	Ordenante	Ordenante	Ordenante	Documen	N° de doc	Fecha de	Cuenta, c	Cuenta, c	Cuenta, c	Monto ab	Monto ab	Estado	Observac
4	WESTERN	RUC	20106903230			21/06/2023	C	S/	305-00375		2334	Procesada	Ninguna
5	STATKRA	RUC	20502597	Factura de	00000061	21/06/2023	C	S/	305-00375	S/	106	Procesada	Ninguna
6	STATKRA	RUC	20502597	Factura de	00000061	21/06/2023	C	S/	305-00375	S/	126	Procesada	Ninguna

Interbancarias

	A	B	C	D	E	F	G
1	Empresa	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.					
2							
3	Tipo de O	Fecha de	Banco de	Ordenante	N° Operación	Monto ab	Monto ab
328	Pago a Pr	21/06/2023	NACION	MUNI DIST CHC	001800008159500	S/	245.60
329	Pago a Pr	21/06/2023	NACION	MUNI PROV CH	001800008159506	S/	29.90

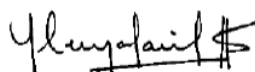
Transferencias

	A	B	C	D	E	F	G	H	I	J
1	Empresa	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.								
2										
3	Ordenante	Fecha de	Cuenta -	Cuenta -	Cuenta -	Monto de	Monto de	T/C	Monto ab	Monto ab
4	EMPRESA	21/06/2023	C	S/	305-00375	S/	#####	0.00	S/	#####
5	CAJA MU	21/06/2023	C	S/	305-00375	S/	2,287.80	0.00	S/	2,287.80
6	VIETTEL	20/06/2023	C	S/	305-00375	S/	21,361.70	0.00	S/	21,361.70
7	GTD PER	20/06/2023	C	S/	305-00375	S/	10,953.52	0.00	S/	10,953.52

Elaborado por:
Felipe Hiromoto Hiromoto
Jefe Corporativo TIC
29 de noviembre de 2022



Revisado por:
Simeón Peña Pajuelo
Coordinador Corp. SIG
30 de noviembre de 2022



Aprobado por:
Lizardo Ojeda López
Gerente General (e)
01 de diciembre de 2022



	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	9 de 25

- ✓ Guardar bases de datos



Home Base de Datos

Subir los archivos de la base de datos

Seleccionar archivo	Sin...os	CODIGO RECAUDO:
Seleccionar archivo	Sin arc...ionados	PREPAGOS:
Seleccionar archivo	Sin ar...nados	TRABAJOES:

Subir

En esta opción es para guardar y actualizar cada Excel con los datos para comprar con el Excel que ha subido en la primera opción de estas casillas:

Home Base de Datos Asientos

Cuadre de Bancos

Seleccionar archivo	Sin archivos seleccionados	Estado de Cuenta
---------------------	----------------------------	------------------


Seleccionar archivo	Sin ar...onados	Proveedores:
Seleccionar archivo	Sin ar...ados	Interbancarias:
Seleccionar archivo	Sin ar...ados	Transferencias:

Subir

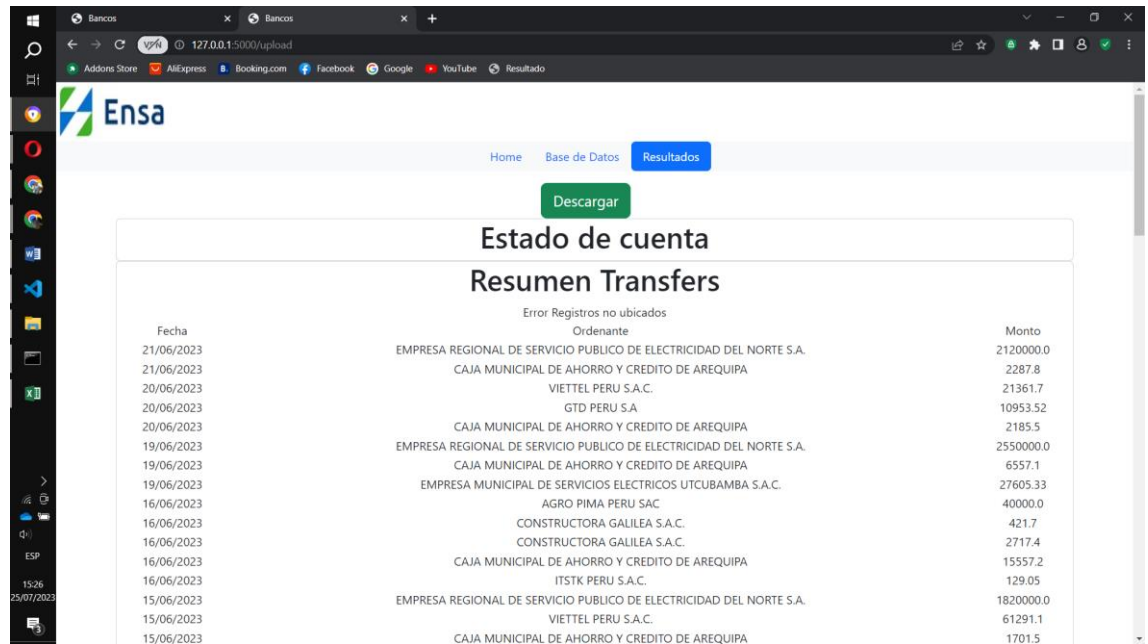
Elaborado por:
Felipe Hiromoto Hiromoto
Jefe Corporativo TIC
29 de noviembre de 2022

Revisado por:
Simeón Peña Pajuelo
Coordinador Corp. SIG
30 de noviembre de 2022

Aprobado por:
Lizardo Ojeda López
Gerente General (e)
01 de diciembre de 2022

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	10 de 25

✓ Resultados



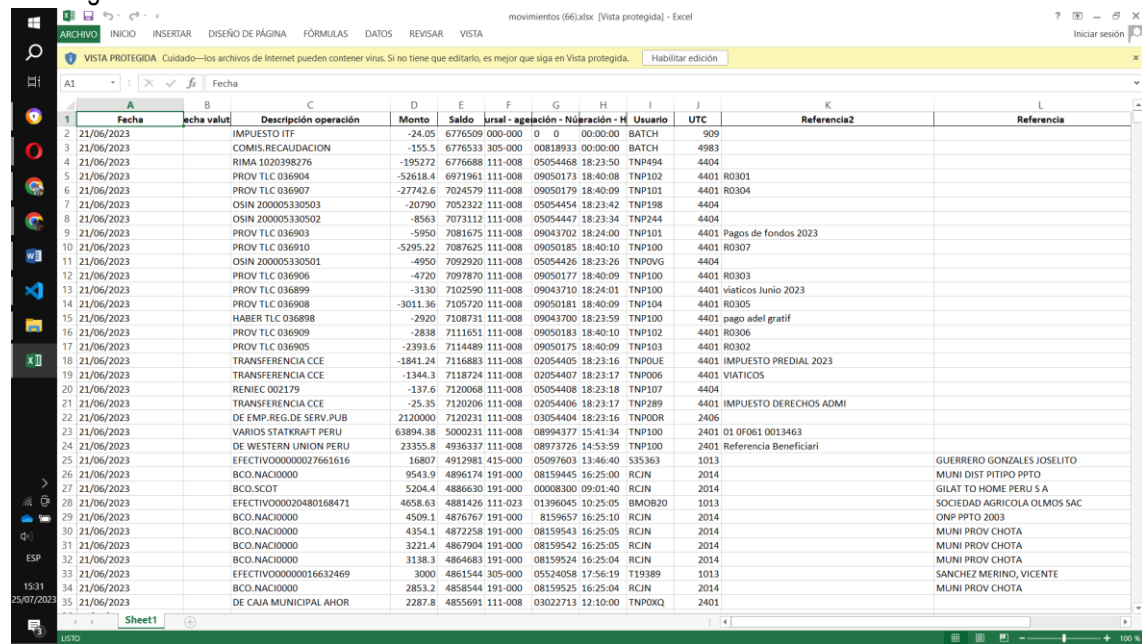
Estado de cuenta
Resumen Transfers

Error Registros no ubicados

Fecha	Ordenante	Monto
21/06/2023	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.	2120000.0
21/06/2023	CAJA MUNICIPAL DE AHORRO Y CREDITO DE AREQUIPA	2287.8
20/06/2023	VIETTEL PERU S.A.C.	21361.7
20/06/2023	GTD PERU S.A.	10953.52
20/06/2023	CAJA MUNICIPAL DE AHORRO Y CREDITO DE AREQUIPA	2185.5
19/06/2023	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.	2550000.0
19/06/2023	CAJA MUNICIPAL DE AHORRO Y CREDITO DE AREQUIPA	6557.1
19/06/2023	EMPRESA MUNICIPAL DE SERVICIOS ELECTRICOS UTCUBAMBA S.A.C.	27605.33
16/06/2023	AGRO PIMA PERU SAC	40000.0
16/06/2023	CONSTRUCTORA GALILEA S.A.C.	421.7
16/06/2023	CONSTRUCTORA GALILEA S.A.C.	2717.4
16/06/2023	CAJA MUNICIPAL DE AHORRO Y CREDITO DE AREQUIPA	15557.2
16/06/2023	ITSTK PERU S.A.C.	129.05
15/06/2023	EMPRESA REGIONAL DE SERVICIO PUBLICO DE ELECTRICIDAD DEL NORTE S.A.	1820000.0
15/06/2023	VIETTEL PERU S.A.C.	61291.1
15/06/2023	CAJA MUNICIPAL DE AHORRO Y CREDITO DE AREQUIPA	1701.5


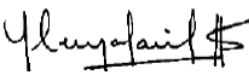

Aquí nos dar el resultados de todas comparación de los Excel que hemos subido y compráramos


✓ Descargar



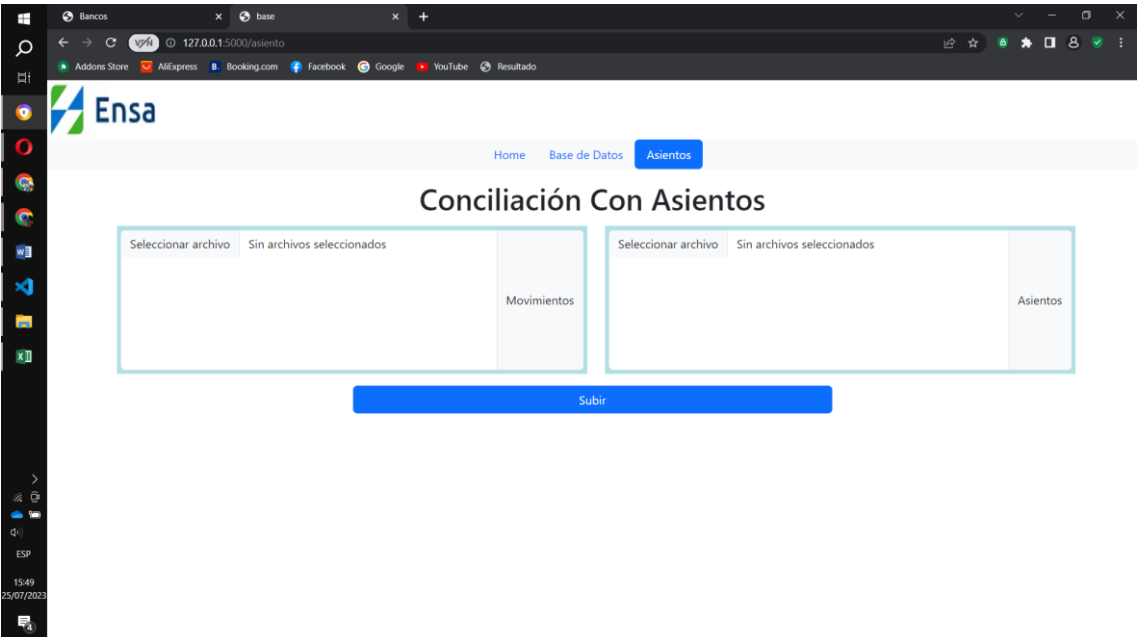
A1	Fecha	Descripción operación	Monto	Saldo	Referencia
1	21/06/2023	IMPUESTO ITF	-24.05	6776509	000-000
2	21/06/2023	COMIS.RECAUDACION	-155.5	6776533	305-000
3	21/06/2023	RIMA 1020398276	-195272	6776688	111-008
4	21/06/2023	PROV TLC 036904	-52618.4	6971961	111-008
5	21/06/2023	PROV TLC 036907	-27742.6	7024579	111-008
6	21/06/2023	OSIN 200005330503	-20790	7052322	111-008
7	21/06/2023	OSIN 200005330502	-8563	7073112	111-008
8	21/06/2023	PROV TLC 036903	-5950	7081675	111-008
9	21/06/2023	PROV TLC 036910	-5295.22	7087625	111-008
10	21/06/2023	OSIN 200005330501	-4950	7092920	111-008
11	21/06/2023	PROV TLC 036906	-4720	7097870	111-008
12	21/06/2023	PROV TLC 036899	-3130	7102590	111-008
13	21/06/2023	PROV TLC 036908	-3011.36	7105720	111-008
14	21/06/2023	HABER TLC 036898	-2920	7108731	111-008
15	21/06/2023	PROV TLC 036909	-2838	7111651	111-008
16	21/06/2023	PROV TLC 036905	-2393.6	7114489	111-008
17	21/06/2023	TRANSFERENCIA CCE	-1841.24	7118883	111-008
18	21/06/2023	TRANSFERENCIA CCE	-1344.3	7118724	111-008
19	21/06/2023	RENIEC 0002179	-137.6	7120068	111-008
20	21/06/2023	TRANSFERENCIA CCE	-25.35	7120206	111-008
21	21/06/2023	DE EMP.REG.DE SERV.PUB	2120000	7120231	111-008
22	21/06/2023	VARIOS STATKRAFT PERU	63894.38	5000231	111-008
23	21/06/2023	DE WESTERN UNION PERU	23355.8	4936337	111-008
24	21/06/2023	EFFECTIVO00000027661616	16807	4912981	415-000
25	21/06/2023	BCO.NACIO000	9543.9	4896174	191-000
26	21/06/2023	BCO.SCOT	5204.4	4886630	191-000
27	21/06/2023	EFFECTIVO0000002480168471	4658.63	4881426	111-023
28	21/06/2023	BCO.NACIO000	4509.1	4876767	191-000
29	21/06/2023	BCO.NACIO000	4354.1	4872258	191-000
30	21/06/2023	BCO.NACIO000	3221.4	4867904	191-000
31	21/06/2023	BCO.NACIO000	3138.3	4864683	191-000
32	21/06/2023	EFFECTIVO00000016632469	3000	4861544	305-000
33	21/06/2023	BCO.NACIO000	2853.2	4858544	191-000
34	21/06/2023	DE CAJA MUNICIPAL AHOR	2287.8	4855691	111-008

Aquí vemos los resultados de que comparado y extraídos los datos de los demás Excel

Elaborado por: Felipe Hironoto Hironoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

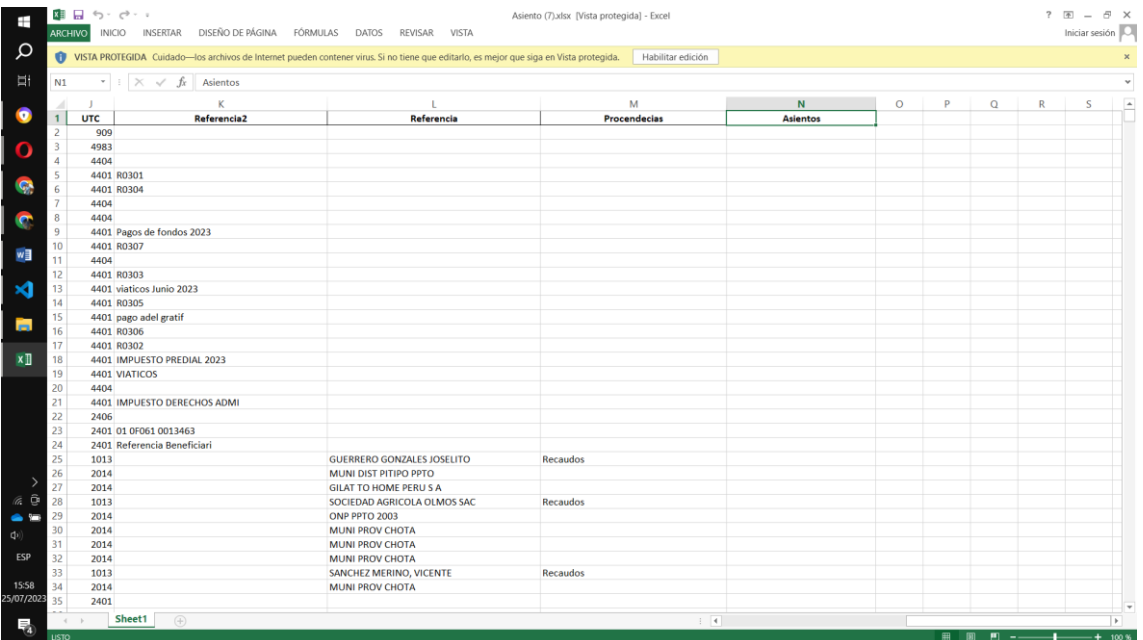
	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	11 de 25

✓ Funcionamiento de asiento




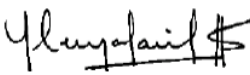

En esta funcionamiento es para comparar los código de N° documento si es igual con el numero Operación – Número para extraer los números de Asignación.


✓ Resultados



UTC	Referencia2	Referencia	Procedencias	Asientos
909				
4983				
4404				
4401 R0301				
4401 R0304				
4404				
4404				
4401 Pagos de fondos 2023				
4401 R0307				
4404				
4401 R0303				
4401 viáticos Junio 2023				
4401 R0305				
4401 pago adel gratif				
4401 R0306				
4401 R0302				
4401 IMPUESTO PREDIAL 2023				
4401 VIATICOS				
4404				
4401 IMPUESTO DERECHOS ADMI				
2406				
2401 01 OF061 0013463				
2401 Referencia Beneficiari				
1013		GUERRERO GONZALES JOSELITO	Recaudos	
2014		MUNI DIST PITIPO PPTO		
2014		GILAT TO HOME PERU S A		
1013		SOCIEDAD AGRICOLA OLMOS SAC	Recaudos	
2014		ONP PPTO 2003		
2014		MUNI PROV CHOTA		
2014		MUNI PROV CHOTA		
1013		SANCHEZ MERINO, VICENTE	Recaudos	
2014		MUNI PROV CHOTA		
2401				

En la columna de asientos es los números extraídos de dos Excel de comparación

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	12 de 25

F. Funciones lógica de cada solución de movimientos


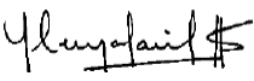
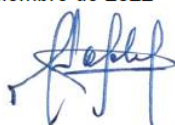
1. **AccountService.py.** En la función de movimientos es el Excel que extraer los datos de cada Excel. En los códigos del archivo AccountService.py hacemos el siguiente algoritmo en una función para importar en el archivo app.py

```

24 AccountService.py > AccountService > process_movimientos
25
26 def process_movimientos(self, movimientos):
27     self.movimientos= pd.read_excel(movimientos, header=4)
28     movimientos_efectivo = self.movimientos.loc[self.movimientos["Descripción operación"].str.contains('EFECTIVO', na=False)]
29     for index, row in movimientos_efectivo.iterrows():
30         cod_recaudo = re.findall(r'\d+', row["Descripción operación"])
31         recaudos = "BD/CODIGO RECAUDO.xls"
32         prepagos = "BD/PREPAGOS.xls"
33         trabajos = "BD/TRABAJORES.xls"
34         # Leer el archivo Excel
35         df_recaudos = pd.read_excel(recaudos, header=None)
36         df_prepagos = pd.read_excel(prepagos, header=None)
37         df_trabajos = pd.read_excel(trabajos, header=None)
38
39         col_codigos_recaudo = df_recaudos[0]
40         col_codigos_prepagos = df_prepagos[0]
41         col_codigos_trabajos = df_trabajos[0]
42         col_descripcion_recaudo = df_recaudos[1]
43         col_descripcion_prepagos = df_prepagos[1]
44         col_descripcion_trabajos = df_trabajos[1]
45         if cod_recaudo:
46             cod_recaudo_entero = [int(digito) for digito in cod_recaudo]
47
48             if cod_recaudo_entero[0] in col_codigos_recaudo.values:
49                 indice = np.where(col_codigos_recaudo == cod_recaudo_entero[0])[0][0]
50                 descripcion = col_descripcion_recaudo[indice]
51                 self.movimientos.at[index, "Referencia"] = descripcion
52                 recaudos = "Recaudos"
53                 self.movimientos.at[index, "Procedencias"] = recaudos
54                 print('df_recaudos')
55             if cod_recaudo_entero[0] in col_codigos_prepagos.values:
56                 indice = np.where(col_codigos_prepagos == cod_recaudo_entero[0])[0][0]
57                 descripcion = col_descripcion_prepagos[indice]
58                 self.movimientos.at[index, "Referencia"] = descripcion
59                 prepagos = "Prepagos"
60                 self.movimientos.at[index, "Procedencias"] = prepagos
61             if cod_recaudo_entero[0] in col_codigos_trabajos.values:
62                 indice = np.where(col_codigos_trabajos == cod_recaudo_entero[0])[0][0]
63                 descripcion = col_descripcion_trabajos[indice]
64                 self.movimientos.at[index, "Referencia"] = descripcion
65                 trabajos = "Trabajadores"
66                 self.movimientos.at[index, "Procedencias"] = trabajos
67
68         if (len(self.movimientos.columns)<11):
69             self.error.message = "Archivo movimientos: Columnas no encontradas, elimine cabeceras innecesarias de movimientos"
70             return
71         if "Fecha" not in self.movimientos.columns:
72             self.error.message = "Columnas no encontradas, elimine cabeceras innecesarias"
73             return

```

- ✓ Lee el archivo de movimientos utilizando pd.read_excel, asumiendo que el archivo tiene una cabecera en la fila 4.
- ✓ Filtra los movimientos que contienen la palabra "EFECTIVO" en la columna "Descripción operación" y los almacena en el DataFrame movimientos_efectivo.
- ✓ Itera sobre los registros de movimientos_efectivo y extrae el código de recaudo presente en la "Descripción operación". Luego, busca el código en los archivos de datos Recaudos, Prepagos y Trabajadores.
- ✓ Si el código de recaudo está presente en uno de los archivos de datos, actualiza la columna "Referencia" del DataFrame self.movimientos con la descripción correspondiente. También actualiza la columna "Procedencias" con el nombre del archivo de datos donde se encontró el código.

Elaborado por: Felipe Hironoto Hironoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	13 de 25

- ✓ Después de procesar todos los movimientos efectivo, verifica si el DataFrame self.movimientos tiene todas las columnas requeridas y la columna "Fecha". Si no se cumplen estas condiciones, se establece un mensaje de error en el atributo self.error.message.
- ✓ En resumen, esta función busca información adicional en los archivos Recaudos, Prepagos y Trabajadores para actualizar las columnas "Referencia" y "Procedencias" en el DataFrame self.movimientos, basándose en el código de recaudo presente en la columna "Descripción operación" de los movimientos efectivo.




2. **ProviderService.py.** La función process_providers(), estar diseñada para procesar un archivo de proveedores, realizar ciertas operaciones en los datos y compararlos con datos existentes en el DataFrame self.movimientos.

```

24
25 def process_providers(self, providersfile):
26     try:
27         proveedores = pd.read_excel(providersfile, header=2)
28         if (len(proveedores.columns) < 13):
29             self.error.message = "Archivo Providers: Columnas no encontradas, elimine cabeceras innecesarias proveccios "
30             return
31         if "Ordenante - Nombre o Razón Social" not in proveedores.columns:
32             self.error.message = "Archivo Estado de Cuenta: Columnas no encontradas, eliminepppppppp cabeceras innecesarias"
33             return
34         proveedores["Monto abonado"] = proveedores["Monto abonado"].astype(str).str.replace(",", "")
35         proveedores["Monto abonado"] = pd.to_numeric(proveedores["Monto abonado"], errors='coerce')
36         proveedores["Ordenante - Nombre o Razón Social"] = proveedores["Ordenante - Nombre o Razón Social"].str.strip()
37         new_proveedores = proveedores[["Monto abonado", "Ordenante - Nombre o Razón Social", "Fecha de pago"]].copy()
38         group_proveedores = new_proveedores.groupby(["Ordenante - Nombre o Razón Social", "Fecha de pago"]).sum().round(2)
39         for index, row in group_proveedores.iterrows():
40             fecha = datetime.strptime(index[1], "%d/%m/%Y").date()
41             monto_abonado = float(row["Monto abonado"])
42
43             reg = self.movimientos.loc[(self.movimientos["Monto"] == monto_abonado) & (self.movimientos["Fecha"] == fecha)]
44
45             if len(reg) > 1:
46                 self.error.message = "Mas de una coincidencia"
47                 self.error.additem(("ordenante": index[0], "monto": monto_abonado, "fecha": fecha))
48             elif len(reg) == 1:
49                 self.movimientos.loc[(self.movimientos["Monto"] == monto_abonado) & (self.movimientos["Fecha"] == fecha), "Referencia"] = index[0]
50             else:
51                 self.error.message = "Registros no ubicados"
52                 self.error.additem(("ordenante": index[0], "monto": monto_abonado, "fecha": fecha))
53
54     except Exception as ex:
55         self.error.message = str(ex)
56


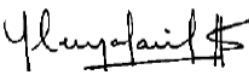

```

- ✓ Lee el archivo de proveedores utilizando pd.read_excel y realiza algunas validaciones para asegurarse de que las columnas requeridas existan en el archivo.
- ✓ Realiza algunos ajustes en los datos del archivo de proveedores, como convertir el monto abonado a tipo numérico y eliminar las comas que puedan haber en los valores.
- ✓ Crea un nuevo DataFrame new_proveedores que contiene solo las columnas "Monto abonado", "Ordenante - Nombre o Razón Social" y "Fecha de pago" del archivo de proveedores.
- ✓ Agrupa los registros del DataFrame new_proveedores por "Ordenante - Nombre o Razón Social" y "Fecha de pago", sumando los montos abonados para obtener un resumen de pagos por proveedor y fecha.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	14 de 25

- ✓ Itera sobre los registros del DataFrame group_proveedores, donde cada registro representa un proveedor y una fecha de pago, y busca coincidencias en el DataFrame self.movimientos para el monto abonado y la fecha de pago correspondientes.
- ✓ Si encuentra más de una coincidencia en self.movimientos, agrega un mensaje de error a self.error y también agrega la información del proveedor y el monto abonado a la lista de errores.
- ✓ Si encuentra exactamente una coincidencia en self.movimientos, actualiza la columna "Referencia" con el valor del proveedor correspondiente.
- ✓ Si no encuentra ninguna coincidencia en self.movimientos, agrega un mensaje de error a self.error y también agrega la información del proveedor y el monto abonado a la lista de errores.
- ✓ En resumen, esta función busca coincidencias en el DataFrame self.movimientos basadas en el monto abonado y la fecha de pago de los proveedores. Si encuentra una coincidencia única, actualiza la columna "Referencia" en el DataFrame self.movimientos, y si encuentra más de una coincidencia o ninguna coincidencia, registra un mensaje de error en el atributo self.error.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	15 de 25


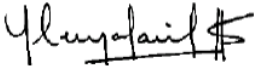

3. **InterbankService.py.** La función `process_interbanks` procesa un archivo de transacciones interbancarias y realiza ciertas operaciones para actualizar la columna "Referencia" en el DataFrame `self.movimientos`. A continuación, se explica el flujo del algoritmo:

```

24
25 def process_interbanks( self,interbankFile):
26     try:
27         Interbancarias = pd.read_excel(interbankFile, header=2)
28         if (len(interbancarias.columns)<7):
29             self.error.message = "Archivo Interbanks: columnas no encontradas, elimine cabeceras innecesarias"
30             return
31
32         if "Tipo de Operación" not in interbancarias.columns:
33             self.error.message = "Archivo Interbanks: columnas no encontradas, elimine cabeceras innecesarias"
34             return
35         interbancarias["Monto abonado"] = interbancarias["Monto abonado"].astype(str).str.replace(",","")
36         interbancarias["Monto abonado"] = pd.to_numeric(interbancarias["Monto abonado"],errors='coerce')
37         interbancarias = interbancarias.loc[interbancarias["Monto abonado - Moneda"]=="S/ "].copy()
38
39         for index, row in interbancarias.iterrows():
40             num_operacion = str(row["Nº Operación"])
41             reg = self.movimientos.loc[(self.movimientos["Monto"].apply(lambda x: round(x, 2))==round(row["Monto abonado"],2)) &
42                                     (self.movimientos["Operación - Número"].astype(str).str[-4:]==num_operacion[-4:]).copy()
43
44             if len(reg)>1:
45                 self.error.message = "Mas de una coincidencia"
46                 self.error.addItem(("ordenante": row["Ordenante"], "monto": row["Monto abonado"], "operacion":num_operacion))
47             elif(len(reg)==1):
48                 self.movimientos.loc[(self.movimientos["Monto"].apply(lambda x: round(x, 2))==round(row["Monto abonado"],2)) &
49                                     (self.movimientos["Operación - Número"].astype(str).str[-4:]==num_operacion[-4:]), "Referencia"] = row["Ordenante"]
50             else:
51                 self.error.addItem(("ordenante": row["Ordenante"], "monto": row["Monto abonado"], "operacion":num_operacion))
52
53         except Exception as ex:
54             self.error.message =str(ex)
55
56
57

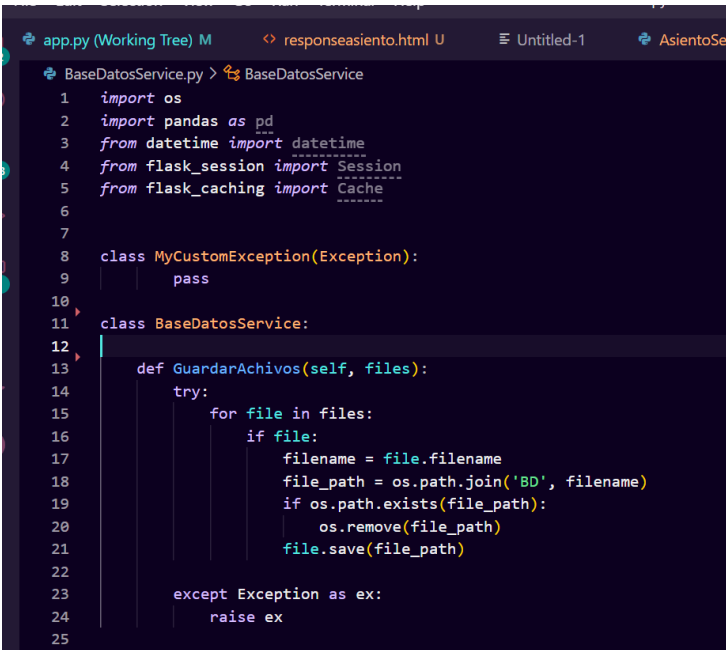
```

- ✓ Lee el archivo de transacciones interbancarias utilizando `pd.read_excel` y realiza algunas validaciones para asegurarse de que las columnas requeridas existan en el archivo.
- ✓ Realiza algunos ajustes en los datos del archivo de transacciones interbancarias, como convertir el monto abonado a tipo numérico y eliminar las comas que puedan haber en los valores.
- ✓ Filtra los registros del DataFrame `interbancarias` para mantener solo aquellos con moneda "S/" en la columna "Monto abonado - Moneda".
- ✓ Itera sobre los registros del DataFrame `interbancarias`, donde cada registro representa una transacción interbancaria, y busca coincidencias en el DataFrame `self.movimientos` para el monto abonado y los últimos 4 dígitos del número de operación.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	16 de 25

- ✓ Si encuentra más de una coincidencia en self.movimientos, agrega un mensaje de error a self.error y también agrega la información de la transacción a la lista de errores.
 - ✓ Si encuentra exactamente una coincidencia en self.movimientos, actualiza la columna "Referencia" con el valor del ordenante correspondiente.
 - ✓ Si no encuentra ninguna coincidencia en self.movimientos, agrega la información de la transacción a la lista de errores sin actualizar la columna "Referencia".
 - ✓ En resumen, esta función busca coincidencias en el DataFrame self.movimientos basadas en el monto abonado y los últimos 4 dígitos del número de operación de las transacciones interbancarias. Si encuentra una coincidencia única, actualiza la columna "Referencia" en el DataFrame self.movimientos, y si encuentra más de una coincidencia o ninguna coincidencia, registra un mensaje de error en el atributo self.error.
4. **BaseDatosService.py** La función GuardarArchivos se encarga de guardar los archivos subidos en una carpeta específica llamada "BD" dentro del directorio actual. Si un archivo con el mismo nombre ya existe en la carpeta, se eliminará antes de guardar el nuevo archivo.






```

1  import os
2  import pandas as pd
3  from datetime import datetime
4  from flask_session import Session
5  from flask_caching import Cache
6
7
8  class MyCustomException(Exception):
9      pass
10
11  class BaseDatosService:
12
13      def GuardarArchivos(self, files):
14          try:
15              for file in files:
16                  if file:
17                      filename = file.filename
18                      file_path = os.path.join('BD', filename)
19                      if os.path.exists(file_path):
20                          os.remove(file_path)
21                      file.save(file_path)
22          except Exception as ex:
23              raise ex
24
25

```

- ✓ La función toma como argumento files, que es una lista de archivos que se subieron mediante una solicitud HTTP.
- ✓ Luego, recorre cada archivo en la lista files.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	17 de 25




- ✓ Comprueba si el archivo existe en la carpeta "BD" usando `os.path.exists(file_path)`. Si existe, elimina el archivo antiguo usando `os.remove(file_path)`.
 - ✓ Luego, guarda el archivo en la ubicación `file_path` usando `file.save(file_path)`.
 - ✓ Si se produce una excepción durante el proceso, como un error al guardar el archivo, la función la captura y la vuelve a lanzar para que se pueda manejar adecuadamente en un nivel superior.
5. **app.py.** Este fragmento de código es un ejemplo de un archivo de Python que utiliza el framework Flask y varias bibliotecas adicionales. Veamos qué hace cada parte del código:

```

app.py M • ProviderService.py M InterbankService.py M AccountService.py M app.py (Working Tree) M •
app.py > ...
1 from flask import Flask, jsonify, render_template, request, send_file, session, redirect, url_for
2 from AccountService import AccountService
3 from InterbankService import InterbankService
4 from ProviderService import ProviderService
5 from TransferService import TransferService
6 from BaseDatosService import BaseDatosService
7 from AsientoService import AsientoService
8 from io import BytesIO
9 from flask_session import Session
10 from flask_caching import Cache
11 from openpyxl import load_workbook
12 import pandas as pd
13 import os
14 import openpyxl
15 from openpyxl import Workbook
16 app = Flask(__name__)
17 app.secret_key = 'AldoAbril1978'
18 app.config['SESSION_TYPE'] = 'filesystem' # Store sessions on the file system
19 Session(app)
20 cache = Cache(app, config={'CACHE_TYPE': 'simple'})
21
22 TRANSFER = "TRANSFER"
23 PROVIDERS = "PROVIDER"
24 INTERBANK = "INTERBAN"
25 CUENTA = "MOVIMIENT"
26 MOVIMIENTOS = "MOVIMIENTOS"
27 ASIENTO= "EXPORT"
28

```

- ✓ Importaciones: Aquí se importan varios módulos y clases necesarios para el funcionamiento del programa, incluidos los módulos de Flask, servicios personalizados y otras bibliotecas como openpyxl y pandas.
- ✓ Creación de la aplicación Flask: Se crea una instancia de la clase Flask con el nombre de la aplicación como argumento.
- ✓ Configuración de la aplicación: Se configuran algunas opciones de la aplicación Flask, como la clave secreta (`secret_key`), el tipo de almacenamiento de sesiones (`SESSION_TYPE`), el tipo de caché (`CACHE_TYPE`) y otros parámetros.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO		Código	FC22-3
	INFORME TÉCNICO		Versión	01/01-12-22
			Página	18 de 25

- ✓ Definición de rutas y funciones de vista: Aquí se definen las rutas de la aplicación y las funciones que se ejecutarán cuando se acceda a cada ruta.
- ✓ Constantes: Se definen algunas constantes con nombres como TRANSFER, PROVIDERS, INTERBANK, etc. Estos nombres parecen ser utilizados como identificadores para diferentes tipos de archivos o servicios.


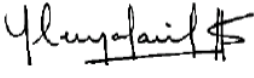

La función home() Esta función llamada home() es una vista de Flask que maneja las solicitudes HTTP a la página principal del sitio web. Veamos qué hace cada parte del código:


```

28
29 @app.route('/', methods=['POST', 'GET'])
30 def home():
31     if request.method == 'POST':
32         files = request.files.getlist('file')
33         filtered_files = [x for x in files if x.filename!=""]
34         if len(filtered_files) <= 1:
35             return render_template('home.html', error_message= 'Debe subir por lo menos un archivo.')
36         else:
37             try:
38                 accountService = AccountService()
39                 transferService = TransferService()
40                 interbankService = InterbankService()
41                 providerService = ProviderService()
42                 for file in files:
43                     nombre = file.filename.upper()
44                     if (nombre != ""):
45                         if CUENTA in nombre:
46                             accountService.process_movimientos(file)
47                         elif TRANSFER in nombre:
48                             transferService.setMovimientos(accountService.movimientos)
49                             transferService.process_transfers(file)
50                         elif INTERBANK in nombre:
51                             interbankService.setMovimientos(accountService.movimientos)
52                             interbankService.process_interbanks(file)
53                         elif PROVIDERS in nombre:
54                             providerService.setMovimientos(accountService.movimientos)
55                             providerService.process_providers(file)
56                         else:
57                             raise Exception("Archivo no ubicado: "+nombre)
58                 cache.set('movimientos', accountService.movimientos, timeout=600)
59                 resumen = {'movimientos': accountService.error, "providers": providerService.error, "transfers": transferService.error, "interbanks": interbankService.error}
60                 cache.set('resumen', resumen)
61                 return redirect(url_for('upload'))
62             except Exception as e:
63                 error_message = str(e)
64                 return render_template('home.html', error_message= error_message)
65     else:
66         return render_template('home.html')


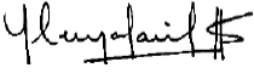

```

- ✓ Verificación del método de la solicitud: Primero, la función verifica si la solicitud es una solicitud POST o GET utilizando request.method. Si es una solicitud POST, significa que el usuario ha enviado archivos, y si es GET, significa que el usuario ha accedido a la página por primera vez.
- ✓ Procesamiento de la solicitud POST: Si la solicitud es POST, el código procede a procesar los archivos recibidos. Aquí hay una secuencia de pasos para procesar diferentes tipos de archivos:
 - Se obtiene una lista de archivos adjuntos con request.files.getlist('file').
 - Se filtran los archivos para eliminar los que no tienen un nombre de archivo válido (nombre vacío) utilizando una comprensión de lista: filtered_files = [x for x in files if x.filename!=""].

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	19 de 25

- Si no hay archivos válidos en la lista, se muestra un mensaje de error en la página de inicio.
 - Si hay al menos un archivo válido, se inician varias clases de servicios (AccountService, TransferService, InterbankService y ProviderService) para procesar los archivos de acuerdo con su tipo.
 - Cada archivo se procesa en función de su tipo y se invoca el método correspondiente en la clase de servicio adecuada (por ejemplo, accountService.process_movements(file)).
 - Se almacenan los resultados y errores de cada servicio en una variable de resumen llamada resumen.
 - Se almacenan los datos de movimiento en caché para su uso posterior en la aplicación.
 - Finalmente, se redirige al usuario a otra vista llamada upload.
- ✓ Procesamiento de la solicitud GET: Si la solicitud es GET, significa que el usuario ha accedido a la página principal por primera vez. En este caso, simplemente se muestra la página de inicio home.html.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	20 de 25


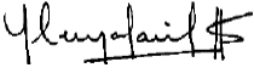

La función `upload()` es una vista de Flask que maneja las solicitudes HTTP a la ruta `'/upload'`. Veamos qué hace cada parte del código:

```

app.py M • ProviderService.py M InterbankService.py M AccountService.py M app.py (Working Tree) M • responseas
app.py > ...
139
140
141
142 @app.route('/upload', methods=['POST', 'GET'])
143 def upload():
144
145     if request.method == 'POST':
146         movimientos = cache.get("movimientos")
147         movimientos["Fecha"] = pd.to_datetime(movimientos["Fecha"], format="%d/%m/%Y").dt.strftime("%d/%m/%Y")
148
149
150         excel_file = BytesIO()
151         movimientos.to_excel(excel_file, index=False)
152
153         # Cargar el archivo Excel con Openpyxl
154         workbook = openpyxl.load_workbook(excel_file)
155         worksheet = workbook.active
156
157         # Ajustar el ancho de la columna "Fecha"
158         worksheet.column_dimensions["A"].width = 20 # Ajusta el ancho de la columna A
159         worksheet.column_dimensions["C"].width = 30 # Ajusta el ancho de la columna A
160         worksheet.column_dimensions["K"].width = 40 # Ajusta el ancho de la columna A
161         worksheet.column_dimensions["L"].width = 40 # Ajusta el ancho de la columna A
162         worksheet.column_dimensions["M"].width = 35 # Ajusta el ancho de la columna A
163
164         ruta_archivo = 'files/movimientos.xlsx'
165         workbook.save(ruta_archivo)
166
167         return send_file(ruta_archivo, as_attachment=True, download_name="movimientos.xlsx")
168     else:
169         data = cache.get('resumen') # Access the query parameter
170         return render_template("response.html", data=data)
171

```

- ✓ Verificación del método de la solicitud: La función verifica si la solicitud es una solicitud POST o GET utilizando `request.method`. Si es una solicitud POST, significa que el usuario ha enviado el formulario con los datos procesados y ahora quiere descargar el archivo Excel resultante. Si es una solicitud GET, significa que el usuario ha accedido a la página de respuesta por primera vez.
- ✓ Procesamiento de la solicitud POST: Si la solicitud es POST, el código procede a realizar algunas modificaciones en el DataFrame 'movimientos' y luego guarda ese DataFrame como un archivo Excel en una ruta específica. Luego, el archivo Excel se envía al cliente como una descarga utilizando la función `send_file`.
 - Primero, se obtiene el DataFrame 'movimientos' desde la caché utilizando `cache.get("movimientos")`.
 - Luego, se convierte la columna 'Fecha' del DataFrame en formato de fecha utilizando `pd.to_datetime` y se formatea como `"%d/%m/%Y"` con `dt.strftime`.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	21 de 25

- A continuación, el DataFrame 'movimientos' se guarda en un archivo Excel utilizando BytesIO y la función to_excel.
 - Después, se carga el archivo Excel utilizando la librería openpyxl.
 - Se ajusta el ancho de algunas columnas del archivo Excel utilizando la propiedad column_dimensions.
 - Finalmente, el archivo Excel se guarda en una ruta específica y se envía como una descarga utilizando send_file.
- ✓ Procesamiento de la solicitud GET: Si la solicitud es GET, significa que el usuario ha accedido a la página de respuesta por primera vez. En este caso, se obtiene la variable 'data' desde la caché utilizando cache.get('resumen'), que contiene un resumen de los resultados y errores de procesamiento de archivos. Luego, se muestra la plantilla 'response.html' con los datos del resumen.


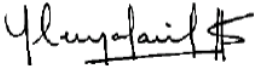

Función de asiento() Esta función llamada asiento() es una vista de Flask que maneja las solicitudes HTTP a la ruta '/asiento'. Veamos qué hace cada parte del código:

```

app.py M x ProviderService.py M InterbankService.py M AccountService.py M app.py (Working Tree) M responseasiento
app.py > asiento
91
92 @app.route('/asiento', methods=['POST','GET'])
93
94 def asiento():
95     if request.method == 'POST':
96         files = request.files.getlist('file')
97         filtered_files = [x for x in files if x.filename!=""]
98         if len(filtered_files) <= 1:
99             return render_template('asiento.html', error_message= 'Debe subir por lo menos un archivo.')
100         else:
101             try:
102                 asientoService = AsientoService()
103                 for file in files:
104                     nombre = file.filename.upper()
105
106                     if (nombre != ""):
107                         if MOVIMIENTOS in nombre:
108                             movimientosfile= file
109                         elif ASIENTO in nombre:
110                             asientosfile = file
111                         else:
112                             raise Exception("Archivo no ubicado: "+nombre)
113                 movimientosAsientos=asientoService.conciliar(movimientosfile, asientosfile)
114                 print(movimientosAsientos)
115                 cache.set('movimientosAsientos', asientoService.resultadosMovimiento, timeout=600)
116                 return redirect(url_for('respuestaasiento'))
117             except Exception as e:
118                 error_message = str(e)
119                 return render_template('asiento.html', error_message= error_message)
120         else:
121             return render_template('asiento.html')
122
123


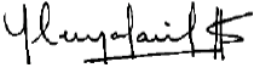

```

- ✓ Verificación del método de la solicitud: La función verifica si la solicitud es una solicitud POST o GET utilizando request.method. Si es una solicitud POST, significa que el usuario ha enviado el formulario con los archivos que desea conciliar, mientras que si es una solicitud GET, significa que el usuario ha accedido a la página de conciliación por primera vez.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	22 de 25

- ✓ Procesamiento de la solicitud POST: Si la solicitud es POST, el código procede a procesar los archivos subidos por el usuario y realizar la conciliación utilizando la clase AsientoService. Luego, los resultados de la conciliación se guardan en la caché y se redirige al usuario a la página de respuesta.
 - Primero, se obtienen los archivos subidos por el usuario utilizando request.files.getlist('file').
 - Luego, se filtran los archivos para eliminar aquellos que no tienen un nombre de archivo válido (en este caso, se consideran válidos aquellos que no tienen un nombre vacío).
 - A continuación, se verifica si se han subido al menos dos archivos válidos. Si no es así, se muestra un mensaje de error en la plantilla 'asiento.html'.
 - Si se han subido al menos dos archivos válidos, se crea una instancia de la clase AsientoService.
 - A continuación, se recorren los archivos subidos y se identifica el tipo de archivo (MOVIMIENTOS o ASIENTO) según su nombre. Los archivos se asignan a las variables movimientosfile y asientosfile.
 - Después de identificar los archivos correspondientes, se llama al método conciliar() de la clase AsientoService, pasando los archivos movimientosfile y asientosfile como argumentos. Este método procesa los archivos y realiza la conciliación, generando los resultados de la conciliación en la variable movimientosAsientos.
 - Luego, los resultados de la conciliación se guardan en la caché utilizando cache.set.
 - Finalmente, se redirige al usuario a la página de respuesta utilizando redirect(url_for('respuestaasiento')).
- ✓ Procesamiento de la solicitud GET: Si la solicitud es GET, significa que el usuario ha accedido a la página de conciliación por primera vez. En este caso, simplemente se muestra la plantilla 'asiento.html', donde el usuario puede subir los archivos para la conciliación.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	23 de 25


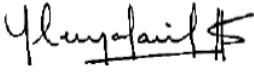

La función `respuestaasiento()`. Esta función llamada `respuestaasiento()` es una vista de Flask que maneja las solicitudes HTTP a la ruta `/respuestaasiento`. Veamos qué hace cada parte del código:

```

158
159 app.route('/respuestaasiento', methods=['POST','GET'])
160 def respuestaasiento():
161
162     if request.method == 'POST':
163         movimientosAsientos = cache.get("movimientosAsientos")
164
165         movimientosAsientos["Fecha"] = pd.to_datetime(movimientosAsientos["Fecha"], format="%d/%m/%Y").dt.strftime("%d/%m/%Y")
166         excel_file = BytesIO()
167         movimientosAsientos.to_excel(excel_file, index=False)
168         workbook = openpyxl.load_workbook(excel_file)
169         worksheet = workbook.active
170         worksheet.column_dimensions["A"].width = 20 # Ajusta el ancho de la columna A
171         worksheet.column_dimensions["C"].width = 30 # Ajusta el ancho de la columna A
172         worksheet.column_dimensions["K"].width = 40 # Ajusta el ancho de la columna A
173         worksheet.column_dimensions["L"].width = 40 # Ajusta el ancho de la columna A
174         worksheet.column_dimensions["M"].width = 35 # Ajusta el ancho de la columna A
175         worksheet.column_dimensions["N"].width = 28 # Ajusta el ancho de la columna A
176
177         ruta_archivo = 'files/movimientos.xlsx'
178         workbook.save(ruta_archivo)
179
180         return send_file(ruta_archivo, as_attachment=True, download_name="Asiento.xlsx")
181     else:
182         data = cache.get("movimientosAsientos") # Access the query parameter
183         return render_template("responseasiento.html", data= data)
184
185
186
187 if __name__ == '__main__':
188     # app.run(host='0.0.0.0')
189     app.run(debug=True)
190
191

```

- ✓ Verificación del método de la solicitud: La función verifica si la solicitud es una solicitud POST o GET utilizando `request.method`. Si es una solicitud POST, significa que el usuario ha hecho clic en el botón de descarga para obtener el archivo de resultados de la conciliación, mientras que si es una solicitud GET, significa que el usuario ha accedido a la página de respuesta por primera vez.
- ✓ Procesamiento de la solicitud POST: Si la solicitud es POST, el código procede a preparar y enviar el archivo de resultados de la conciliación para su descarga.
 - Primero, se obtienen los resultados de la conciliación almacenados en la caché utilizando `cache.get("movimientosAsientos")`.
 - Luego, se formatea la columna "Fecha" de los resultados de la conciliación a un formato adecuado utilizando `pd.to_datetime` y `dt.strftime`, para asegurarse de que la fecha se muestre correctamente en el archivo Excel.

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--


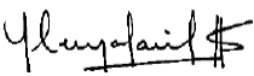

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	24 de 25


- A continuación, se crea un archivo Excel en memoria (utilizando BytesIO) y se escribe el DataFrame de resultados de la conciliación en el archivo Excel utilizando to_excel().
 - Luego, se carga el archivo Excel utilizando openpyxl.load_workbook() y se ajustan los anchos de las columnas del archivo Excel para que los datos se muestren correctamente.
 - Después de ajustar los anchos de las columnas, se guarda el archivo Excel en el directorio 'files/' con el nombre 'movimientos.xlsx'.
 - Finalmente, se envía el archivo de resultados de la conciliación para su descarga al usuario utilizando send_file, con el nombre de descarga "Asiento.xlsx".
- ✓ Procesamiento de la solicitud GET: Si la solicitud es GET, significa que el usuario ha accedido a la página de respuesta por primera vez. En este caso, se obtienen los resultados de la conciliación almacenados en la caché y se muestran en la plantilla 'responseasiento.html', donde el usuario puede ver los resultados.

G. Arquitectura de comunicaciones

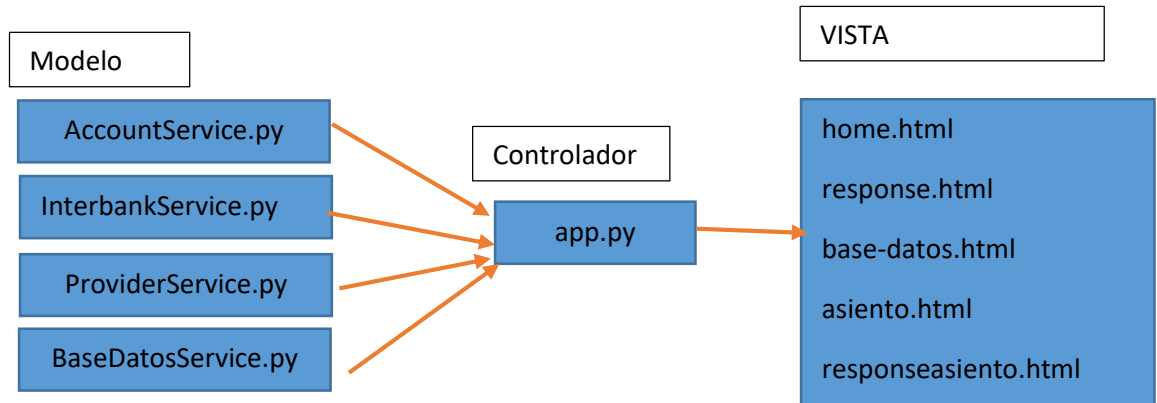
El patrón de arquitectura En Flask, la arquitectura se basa en el patrón Modelo-Vista-Controlador (MVC), que es un patrón de diseño de software que separa la lógica de negocio, la presentación y la interacción del usuario en tres componentes principales:

1. Modelo (MODELO): Es responsable de representar los datos y la lógica de negocio de la aplicación. Aquí se definen las clases y métodos que interactúan con la base de datos y manejan la lógica para procesar y manipular los datos.
2. Vista (HTML): Es responsable de la presentación de los datos al usuario. Aquí se definen las plantillas HTML que muestran la información al usuario y proporcionan una interfaz para que el usuario interactúe con la aplicación.
3. Controlador (APP.PY): Es responsable de manejar las solicitudes del usuario y coordinar la interacción entre el modelo y la vista. Aquí se definen las rutas y funciones de Flask que procesan las solicitudes HTTP del usuario, interactúan con el modelo para obtener o modificar los datos y renderizan las plantillas HTML para mostrar la respuesta al usuario

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--

	FORMATO	Código	FC22-3
	INFORME TÉCNICO	Versión	01/01-12-22
		Página	25 de 25

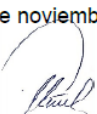
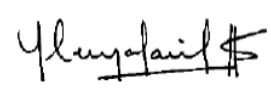
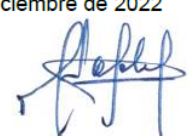
Las comunicaciones al servidor se realizan con los métodos GET y POST



III. Conclusiones

- La aplicación web es más rápido en las consultas y las modificaciones.
- La aplicación web es depende la cantidad de datos que se encuentra en el Excel para la demora para el demora.

IV. Firma del encargado responsable del cambio

Elaborado por: Felipe Hiromoto Hiromoto Jefe Corporativo TIC 29 de noviembre de 2022 	Revisado por: Simeón Peña Pajuelo Coordinador Corp. SIG 30 de noviembre de 2022 	Aprobado por: Lizardo Ojeda López Gerente General (e) 01 de diciembre de 2022 
---	--	--