



Proyecto 1

Servicio web SOAP

Integrantes

Gonzalo Salinas

Ignacio Valdes

Camila Carrasco

Asignatura

Computación paralela y distribuida

Docente

Sebastian Salazar

Entrega

Sábado, 01 de Agosto de 2020

Introducción

Hoy en día existe una tendencia muy marcada en las empresas por el desarrollo de aplicaciones que trabajen sobre internet, principalmente por la necesidad de comunicación y transmisión de datos. Frente a esta necesidad se crea una nueva tecnología llamada web service, la cual permite a distintas aplicaciones, de distinto origen, comunicarse entre sí. SOAP(Simple Objects Access Protocol) nace como estrategia de desarrollo para resolver los problemas de inoperabilidad entre diversas tecnologías, este protocolo permitirá la comunicación por medio de intercambio de datos XML.

El presente trabajo tiene como propósito comprender el funcionamiento del protocolo SOAP y su identificación como protocolo para promover la interoperabilidad entre aplicaciones web. Este conocimiento nos permitirá proponer un servicio web SOAP que resuelva una necesidad específica de la Universidad Tecnológica Metropolitana.

Definición Servicio Web

Un Servicio Web es un servicio de negocio programado en algún lenguaje, que envuelve cierta funcionalidad y la expone en una red de manera estandarizada, para que cualquiera conectado a esta misma red, que además soporte dichos estándares, pueda acceder a ella. Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos. El cliente solicita información, enviando a veces datos al servidor para que pueda procesar su solicitud. El servidor genera una respuesta que envía de vuelta al cliente, adjuntando otra serie de datos que forman parte de esa respuesta. Por tanto, podemos entender un servicio web como un tráfico de mensajes entre dos máquinas.

Servicio Web Soap

SOAP se define como un protocolo estándar de comunicación (conjunto de reglas), un intercambio de mensajes basado en la especificación de XML. SOAP utiliza diferentes protocolos de transporte, tales como HTTP y SMTP . El protocolo HTTP estándar hace que sea más fácil para el modelo de SOAP para túnel a través de cortafuegos y proxies sin ninguna modificación en el protocolo SOAP

Los mensajes SOAP se estructuran con etiquetas XML junto a la intervención de un lenguaje adicional para su definición, *WSDL (Web Services Description Language)* que se usa para generar una interfaz del Servicio Web, misma que será su punto de entrada. A éste se le conoce como Archivo wsdl o Contrato y una vez que el servicio está publicado, éste archivo estará accesible en una red por medio de un url que tiene terminación *?wsdl*.

Soap especifica el formato de los mensajes de la siguiente manera:

Envelope (envoltura): Es el elemento raíz del mensaje para describir su contenido y la forma de procesarlo.

Header (encabezado): Es la información de identificación del contenido. Un grupo de reglas de codificación para expresar las instancias de tipos de datos definidos por la aplicación.

Body (cuerpo): Es el contenido del mensaje. Una convención para representar las llamadas y las respuestas a procedimientos remotos.

Tecnología utilizada

Para el desarrollo de este proyecto es importante conocer los distintos lenguajes de programación web, cada uno tiene su peculiaridad, por lo cual es importante tener claros los criterios al momento de elegir. Es necesario hacer la distinción entre los lenguajes de programación del lado del cliente y los del lado del servidor, para efectos de la asignatura nos enfocaremos en el servidor.

El primer criterio es seleccionar un lenguaje de alto nivel que nos permita resolver problemas de forma sencilla. El lenguaje Python resulta ser la respuesta a esta elección de creación de código backend, ya que posee un formato simple y su sintaxis está orientada a escribir menos líneas de código pero más claras que en otros lenguajes, lo que facilita enormemente su mantenimiento, por otra parte se considera por muchos el lenguaje más limpio a la hora de programar.

Una consideración adicional a la tecnología de este lenguaje fue el conocimiento previo de este lenguaje, ya que por su sencillez se ha vuelto un lenguaje de referencia en educación de diversas instituciones, incluida nuestra universidad. Además Python es un lenguaje de código abierto y libre lo cual nos dará la posibilidad de compartir, modificar y estudiar el código fuente para adecuarlo a nuestro proyecto.

Python es ampliamente utilizado, por lo que tiene una gran base de programadores que participan en la actualización de páginas con todos los módulos recomendados, mantenidos activamente para SOAP. Este lenguaje cuenta con un kit de herramientas python RPC que entre otros protocolos, admite SOAP. Dentro de estas herramientas se encuentra Spyne actualmente admite el estándar de descripción de interfaz WSDL 1.1, junto con SOAP 1.1.

Para la creación de un servicio web SOAP en lenguaje Python, se cuenta con las siguientes librerías open source:

- Openpyxl: Biblioteca de Python que permite leer o escribir archivos Excel 2010 xlsx / xlsxm / xlsx / xltm.
- Pandas: Paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para hacer que trabajar con datos estructurados.
- Spyne: Biblioteca rpc agnóstica de transporte y arquitectura que se enfoca en exponer los servicios públicos con una API bien definida.

Requerimientos de utilización para la solución

Para la utilización del SOAP es necesaria la instalación de python 3 y pip en su última versión, luego, se deben instalar las librerías anteriormente descritas, para esto debe ingresar el siguiente código en consola:

Para windows:

```
pip install spyne  
pip install pandas  
pip install openpyxl
```

Para linux:

```
apt install python3-pip
pip3 install spyne
pip3 install pandas
pip3 install openpyxl
sudo apt-get install python-lxml
pip3 install --upgrade lxml
sudo pip3 install suds-jurko
```

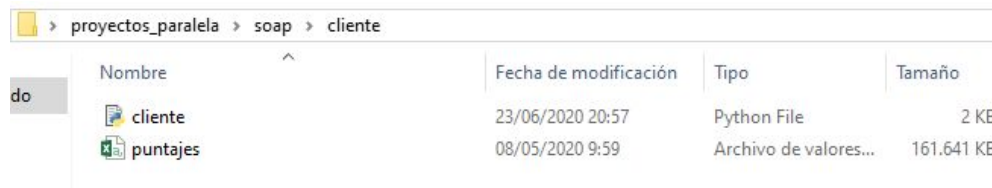
Manual de uso

Para la utilización del programa debe seguir los siguientes pasos:

- 1) Inicialmente, se debe abrir y ejecutar el archivo servidor.py, puede ser mediante VScode o directamente desde el idle de python, lo importante es ver el siguiente mensaje del servidor en consola:

```
INFO:root:listening to http://127.0.0.1:8000
INFO:root:wsdl is at: http://localhost:8000/?wsdl
```

- 2) Con el servidor en ejecución, se debe ingresar a la carpeta “cliente” y colocar el archivo “puntajes.csv” sin codificar dentro de esta, junto al archivo cliente.py, de la siguiente manera:



proyectos_paralela > soap > cliente				
	Nombre	Fecha de modificación	Tipo	Tamaño
do	cliente	23/06/2020 20:57	Python File	2 KB
	puntajes	08/05/2020 9:59	Archivo de valores...	161.641 KB

Archivo puntajes utilizado:

https://drive.google.com/file/d/1GWzW_1Nv_DpqSibhAVStlllSBo9jyJYn/view?usp=sharing

- 3) Finalmente, simplemente se debe ejecutar el archivo “cliente.py” al igual como se hizo con el servidor, de esta forma, usted podrá seguir el proceso de trabajo sobre los datos en el log de consola del servidor. Ejemplo de log exitoso:

```

Validando datos ingresados...
El archivo es correcto.
MIME detectado = text/csv (application/vnd.ms-excel)
Obteniendo ponderaciones por estudiante a cada carrera...
Asignando carreras a los mejores postulantes y a su vez, mejor opcion por postulante...
Carrera: Ingeniería en Construcción...
Carrera: Diseño en Comunicación Visual...
Carrera: Trabajo Social...
Carrera: Ingeniería en Informática...
Carrera: Ingeniería Civil en Mecánica...
Carrera: Dibujante Projectista...
Carrera: Ingeniería en Industria Alimentaria...
Carrera: Ingeniería en Biotecnología...
Carrera: Bibliotecología y Documentación...
Carrera: Ingeniería Civil en Computación, mención Informática...
Carrera: Ingeniería Civil en Ciencia de Datos...
Carrera: Ingeniería Civil Industrial...
Carrera: Ing. en Gestión Turística...
Carrera: Ingeniería Civil Electrónica...
Carrera: Ing. Comercial...
Carrera: Diseño Industrial...
Carrera: Arquitectura...
Carrera: Ing. en Comercio Inter...
Carrera: Contador Público y Auditor...
Carrera: Ing. en Adm. Agroindustrial...
Carrera: Administración Pública...
Carrera: Ingeniería Civil en Obras Civiles...
Carrera: Ingeniería Civil en Prevención de Riesgos y Medioambiente...
Carrera: Ingeniería en Geomensura...
Carrera: Ingeniería en Química...
Carrera: Química Industrial...
Carrera: Bachillerato en Ciencias de la Ingeniería...
Carrera: Ingeniería Industrial...
  
```

Ejemplo de respuesta del servidor:

```
127.0.0.1 - - [04/Jul/2020 00:44:34] "POST / HTTP/1.1" 200 84713
```

4) Finalmente el archivo de respuesta al servidor, se genera junto al archivo cliente.py:

proyectos_paralela > soap > cliente				
	Nombre	Fecha de modificación	Tipo	Tamaño
do	cliente	04/07/2020 0:26	Python File	2 KB
	Matriculados Utem	04/07/2020 0:44	Hoja de cálculo d...	62 KB
	puntajes	08/05/2020 9:59	Archivo de valores...	161.641 KB

Lógica interna y explicación del funcionamiento del servidor:

Función codificar:

```

def codificar(textoSinCodificar):
    base64_cadena_bytes = base64.b64encode(textoSinCodificar.encode('ascii'))
    texto_base64 = base64_cadena_bytes.decode('ascii')
    return texto_base64
  
```

La función codificar se utiliza para codificar un string a un string base64, recibe el texto sin codificar y lo codifica a formato ascii, luego, es codificado en base 64 y finalmente se decodifica la cadena de bytes base64 en ascii, de esta forma se obtiene la cadena de representación del texto ingresado en base64.

Función decodificar:

```
def decodificar(textoCodificado):  
    try:  
        cadena_bytes = base64.b64decode(textoCodificado.encode('ascii'))  
        textoDecode = cadena_bytes.decode('ascii')  
        return textoDecode  
    except:  
        print("Error: Falla inesperada en decodificación, revise la codificación base64 del archivo, esta puede haber sido cortada.")
```

Tiene un funcionamiento similar al de la función codificar, solo que se realiza el proceso inverso, en caso de que reciba un texto con un error en la codificación, arroja la excepción.

Función revisarMime:

```
def revisarMime(nombreDelArchivo): # Revisa el mime del nombre de archivo ingresado. Retorna el MIME de csv cuando es correcto. fuente  
    resultado=guess_type(nombreDelArchivo)  
    if(resultado[0]=="application/vnd.ms-excel" and re.search(".csv$", nombreDelArchivo)): #Detecta Mime del archivo  
        print("MIME detectado = text/csv (" +str(resultado[0])+")")  
        return "text/csv"  
    else:  
        print("MIME detectado = " +str(resultado[0]))  
        print("Error: MIME Incorrecto, debe utilizar text/csv. El archivo no corresponde al formato admitido, por favor, reintentar.")  
        return resultado[0]
```

Utilizada para revisar el mime del archivo recibido en la solicitud al servidor, esta función solo se enfoca en la detección de la extensión y su comparación con el resultado de guess_type(), si corresponde a un nombre de archivo csv, retorna el mime esperado, en caso contrario, la excepción.

Función revisarContenidoBase64:

```
def revisarContenidoBase64(TextoCodificado):  
    contador = 0  
    try:  
        revision = csv.Sniffer().sniff(decodificar(TextoCodificado),";")  
        mensaje = decodificar(TextoCodificado).split("\n")  
        for linea in mensaje:  
            for caracter in linea:  
                contador = contador + 1  
                if (contador>32):  
                    print("Error: Excedido tamaño de filas, existe al menos una fila mal ingresada en el a")  
                    return 0  
                if ((caracter!=";" and caracter.isnumeric()) or caracter!=";"):  
                    pass  
                else:  
                    print("Error: Letras detectadas en contenido, el archivo debe contener solo numeros y")  
                    return 0  
            if (contador!=0 and contador<32):  
                print("Error: Insuficiente tamaño de filas, existe al menos una fila mal ingresada en el a")  
                return 0  
            else:  
                contador = 0  
        print("El archivo es correcto.")  
        return 1  
    except csv.Error:  
        print("Error: Separadores y delimitadores no corresponden. El archivo no corresponde al formato ad")  
        return 0
```


Función utilizada para la revisión total del archivo ingresado en la solicitud al servidor. Esta función en primera instancia, utiliza el método de “csv.Sniffer().sniff()” junto a “decodificar” para revisar separadores y otros aspectos característicos de un archivo csv, en caso de que ocurra un error, retorna la excepción al final de la función. Luego de decodificar con éxito los datos, define un contador, el cual se utiliza para detectar el largo de las filas del archivo csv (o supuesto csv), si el largo es superior a 32, retornará un error de exceso de tamaño de la fila, si es menor a 32, retornará un error de tamaño insuficiente de filas. ¿Porque se usa el número 32? Porque el largo de las filas de ruts mas los 6 puntajes junto a sus separadores (;) suman exactamente 32 caracteres. Al mismo tiempo que se revisa el largo de las filas del archivo, también se revisa cada uno de los caracteres, si se encuentra cualquiera que no sea número o un separador (;), retornara el error correspondiente. Finalmente, si el archivo pasa todos los controles, se aprueba como correcto.

Función generarXlsx:

```
def generarXlsx(rutss,puntajess,nombreXlsx):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    writer = pandas.ExcelWriter(dir_path+"/"+nombreXlsx) # pylint: disable=abstract-class-instantiated
    for i in range(0,28):
        ordenarEsto = []
        for j in range(len(puntajess[i])):
            ordenarEsto.append([rutss[i][j],puntajess[i][j]])
        ordenarEsto.sort(key=lambda x: x[1], reverse=1)
        r=[]
        p=[]
        for k in range(len(ordenarEsto)):
            r.append(ordenarEsto[k][0])
            p.append(ordenarEsto[k][1])

        df = pandas.DataFrame({
            "Nº": [int(j)+1 for j in range(len(puntajess[i]))],
            "RUT Matriculado": r,
            "Puntaje": p})
        df = df[["Nº", "RUT Matriculado", "Puntaje"]]
        if (i==0):
            df.to_excel(writer, "Administración Pública", index=False)
        if (i==1):
            df.to_excel(writer, "Bibliotecología y Documentación", index=False)
        if (i==2):
            df.to_excel(writer, "Contador Público y Auditor", index=False)
        if (i==3):
            df.to_excel(writer, "Ing. Comercial", index=False)
        if (i==4):
            df.to_excel(writer, "Ing. en Adm. Agroindustrial", index=False)
        if (i==5):
            etc.... etc....
    writer.save()
```

Esta función genera el Xlsx requerido por el cliente, recibe los datos ya tratados por la función principal y simplemente, utilizando la librería pandas, crea el xml hoja a hoja con los datos ordenados de los puntajes de cada carrera de mayor a menor. Es importante destacar que este archivo es creado en la ubicación del archivo servidor.py, y luego se utiliza para enviar su información como respuesta al cliente solicitante, después de eso, se elimina.

Clase servicios:

```
class servicios(ServiceBase):

    @rpc(Unicode, Unicode, Unicode, _returns = Iterable(Unicode))
    def programaPrincipal(ctx,mime,nombreDelArchivo,datosBase64):#Funcion de consumo
        logger = logging.getLogger('spyne.protocol')
        logger.disabled = True
```

La clase servicios contiene la función de consumo, y como se puede ver, recibe los datos entregados por el cliente los cuales corresponden a mime, nombreDelArchivo y datosBase64 en su declaración.

Lo primero que se hace en la función programaPrincipal, es la declaración de las variables que contendrán la información de los matriculados, en otras palabras, los resultados finales del algoritmo:

```
matriculadosPorCarreraRuts=[[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]] # Co
matriculadosPorCarreraPuntajes=[[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]] #
lista = []
```

Los datos se separan en puntajes y ruts en diferentes listas, las cuales conservan el orden de los índices de cada dato en conjunto, de esta forma y a modo de ejemplo, el rut de la posición [0][12] de la lista de rutas, tendrá el puntaje de la posición [0][12] de la lista de puntajes. En este ejemplo, los datos del matriculado corresponden a un ingreso a la carrera en la posición [0], la cual no es más que una opción de otras 27, así se deduce, que la lista tiene 28 sub-listas cada una de ellas para cada carrera.

El dato lista = [], se utiliza más adelante para guardar la información de todos los postulantes a todas las carreras.

Luego, dentro de la misma función, se realiza la comprobación de todos los datos ingresados o recibidos desde el cliente a través de la siguiente operación:

```
if (not (revisarContenidoBase64(datosBase64)) or revisarMime(nombreDelArchivo)!="text/csv" or mime!="text/csv"):
    if (mime!="text/csv"):
        print("Recepcion de MIME incorrecto, usar text/csv.")
    return "Finalizando ejecucion..."
```

En ella se utilizan los métodos “revisarContenidoBase64” y “revisarMime”.

Luego de los pasos anteriores, comienza la operación sobre los datos, los cuales comienzan con el siguiente código:


```

arregloDePuntajesPsu = decodificar(datosBase64).split("\n") # Separa el string en los saltos de linea
contadorPostulante = 0
print("Obteniendo ponderaciones por estudiante a cada carrera...")
for datosPostulante in arregloDePuntajesPsu:

    arregloDePuntajesPsu[contadorPostulante]=datosPostulante.split(";")

    if (len(arregloDePuntajesPsu[contadorPostulante])==7):
        arregloDePuntajesPsu[contadorPostulante][0]=str(arregloDePuntajesPsu[contadorPostulante][0])
        arregloDePuntajesPsu[contadorPostulante][1]=float(arregloDePuntajesPsu[contadorPostulante][1])
        arregloDePuntajesPsu[contadorPostulante][2]=float(arregloDePuntajesPsu[contadorPostulante][2])
        arregloDePuntajesPsu[contadorPostulante][3]=float(arregloDePuntajesPsu[contadorPostulante][3])
        arregloDePuntajesPsu[contadorPostulante][4]=float(arregloDePuntajesPsu[contadorPostulante][4])
        arregloDePuntajesPsu[contadorPostulante][5]=float(arregloDePuntajesPsu[contadorPostulante][5])
        arregloDePuntajesPsu[contadorPostulante][6]=float(arregloDePuntajesPsu[contadorPostulante][6])
        np = arregloDePuntajesPsu[contadorPostulante]

```

El primer paso es la decodificación de todos los datos sobre los postulantes, luego, se obtienen todos los datos por postulante y se separan del string original, obteniendo 7 datos, el rut y sus 6 puntajes asociados y se transforman a tipo float.

Seguido a lo anterior, se realiza el cálculo de todas las ponderaciones de dicho postulante para cada una de las carreras:

```

for datosPostulante in arregloDePuntajesPsu:

    arregloDePuntajesPsu[contadorPostulante]=datosPostulante.split(";")

    if (len(arregloDePuntajesPsu[contadorPostulante])==7):
        arregloDePuntajesPsu[contadorPostulante][0]=str(arregloDePuntajesPsu[contadorPostulante][0])
        arregloDePuntajesPsu[contadorPostulante][1]=float(arregloDePuntajesPsu[contadorPostulante][1])
        arregloDePuntajesPsu[contadorPostulante][2]=float(arregloDePuntajesPsu[contadorPostulante][2])
        arregloDePuntajesPsu[contadorPostulante][3]=float(arregloDePuntajesPsu[contadorPostulante][3])
        arregloDePuntajesPsu[contadorPostulante][4]=float(arregloDePuntajesPsu[contadorPostulante][4])
        arregloDePuntajesPsu[contadorPostulante][5]=float(arregloDePuntajesPsu[contadorPostulante][5])
        arregloDePuntajesPsu[contadorPostulante][6]=float(arregloDePuntajesPsu[contadorPostulante][6])
        np = arregloDePuntajesPsu[contadorPostulante]

        ponderacionesDeIAlumno=[]
        if (np[5]>=np[6] and (np[3]+np[4])/2>450): # Verifica que el puntaje de Ciencias sea mayor o igual al de Matemáticas
            ponderacionesDeIAlumno.append(np[0]) # Agrega el rut
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.2+np[3]*0.25+np[4]*0.3+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.2+np[3]*0.1+np[4]*0.4+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.2+np[3]*0.15+np[4]*0.3+np[5]*0.15,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.2+np[3]*0.3+np[4]*0.3+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.25+np[3]*0.2+np[4]*0.2+np[5]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.2+np[3]*0.35+np[4]*0.15+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.35+np[3]*0.2+np[4]*0.2+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.25+np[3]*0.3+np[4]*0.2+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.25+np[3]*0.3+np[4]*0.15+np[5]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.4+np[3]*0.1+np[4]*0.3+np[5]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.3+np[3]*0.1+np[4]*0.2+np[5]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.25+np[3]*0.35+np[4]*0.2+np[5]*0.1,2))
        elif((np[3]+np[4])/2>450): # Si lo anterior no es ejecutado, revisa que la razon de ello sea que
            ponderacionesDeIAlumno.append(np[0])
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.2+np[3]*0.25+np[4]*0.3+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.2+np[3]*0.1+np[4]*0.4+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.2+np[3]*0.15+np[4]*0.3+np[6]*0.15,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.2+np[3]*0.3+np[4]*0.3+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.25+np[3]*0.2+np[4]*0.2+np[6]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.35+np[3]*0.2+np[4]*0.15+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.35+np[3]*0.2+np[4]*0.2+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.15+np[2]*0.25+np[3]*0.3+np[4]*0.2+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.25+np[3]*0.3+np[4]*0.15+np[6]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.4+np[3]*0.1+np[4]*0.3+np[6]*0.1,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.2+np[2]*0.3+np[3]*0.1+np[4]*0.2+np[6]*0.2,2))
            ponderacionesDeIAlumno.append(round(np[1]*0.1+np[2]*0.25+np[3]*0.35+np[4]*0.2+np[6]*0.1,2))
        lista.append(ponderacionesDeIAlumno) # Se guarda el rut del estudiante junto a todas sus ponderaciones
    contadorPostulante = contadorPostulante + 1

```

Existen dos ramificaciones de cálculo las cuales se deben a la elección del puntaje de mejor conveniencia para el postulante entre historia y ciencias, además, se descarta el postulante si su puntaje promedio de matemáticas y lenguaje es inferior a 450 pts. Finalmente, se agrega el postulante y sus 12 ponderaciones a la lista de resultados, llamada “lista”.

Pero, ¿Por qué 12 ponderaciones si son 28 carreras?, esto se debe a que existen carreras con igual exigencia en ponderación, por lo que al agrupar cada una de ellas en grupos de semejantes, se obtienen 12 cálculos base, de esta forma se evita realizar otros 16 cálculos.

Luego en base a las ponderaciones de todos los postulantes, se realiza la asignación de los mejores estudiantes a las carreras, pero, si y sólo si, el puntaje por el cual se asigna un estudiante a una carrera corresponde al mejor puntaje de dicho estudiante. A continuación se muestra el código que realiza esta acción para la carrera de diseño en comunicación visual:

```
print("Carrera: Diseño en Comunicación Visual...")
lista.sort(key=lambda x: x[10], reverse=1)
b=0
a=0
c=0
while(b<100):
    if ((lista[a][10]>=max(lista[a][1:12]) and lista[a][0]!=0) or (c and lista[a][0]!=0)):
        matriculadosPorCarreraRuts[15].append(lista[a][0])
        matriculadosPorCarreraPuntajes[15].append(lista[a][10])
        lista[a][0] = 0
        b=b+1
    if(a==len(lista)-1 and c==1):
        break
    if (a==len(lista)-1 and b<100):
        c=1
        a=0
    else:
        a=a+1
```

El primer paso, es el ordenamiento de la lista de resultados en base al dato de la posición de la sub-lista que se le asigna a la carrera de diseño en comunicación, el cual corresponde al dato lista[x][10], recordar los 12 grupos de ponderaciones (el dato lista[x][0] corresponde al rut):

```
# IMPORTANTE: Se separaron las ponderaciones exigidas de las carreras en 12 grupos, ya que habian carreras con la misma exigencia, a conti
# grupo1 Contendrá los mejores puntajes ordenados de las carreras: 21089
# grupo2 Contendrá los mejores puntajes ordenados de las carreras: 21002
# grupo3 Contendrá los mejores puntajes ordenados de las carreras: 21012
# grupo4 Contendrá los mejores puntajes ordenados de las carreras: 21048, 21015, 21081 y 21082
# grupo5 Contendrá los mejores puntajes ordenados de las carreras: 21047
# grupo6 Contendrá los mejores puntajes ordenados de las carreras: 21074 y 21032
# grupo7 Contendrá los mejores puntajes ordenados de las carreras: 21087
# grupo8 Contendrá los mejores puntajes ordenados de las carreras: 21073 y 21039
# grupo9 Contendrá los mejores puntajes ordenados de las carreras: 21080 y 21083
# grupo10 Contendrá los mejores puntajes ordenados de las carreras: 21024 y 21023
# grupo11 Contendrá los mejores puntajes ordenados de las carreras: 21043
# grupo12 Contendrá los mejores puntajes ordenados de las carreras: 21046, 21071, 21041, 21076, 21049, 21075, 21096, 21031, 21030 y 21045
```

Entonces de esta forma, se ordenan en la parte superior o inicial de la lista los mejores candidatos, los cuales son revisados si efectivamente el puntaje de la carrera que se analiza (en este caso diseño en comunicación visual) es el más alto que tienen. Es aquí donde entra esta línea de código:

```
if ((lista[a][10]>=max(lista[a][1:12]) and lista[a][0]!=0) or (c and lista[a][0]!=0)):
```

Es un If maravilloso que hace cumplir 2 condiciones inicialmente antes de agregar un estudiante a una carrera, las cuales son:

1era: (lista[a][6]>=max(lista[a][1:12])); asegura que el puntaje ponderado de la carrera que se revisa sea el mejor ponderado del alumno, esto evita el re-posicionamiento de los matriculados.

2da: lista[a][0]!=0; que el alumno no esté matriculado en una carrera (si está matriculado, Rut=0)

Luego, si al recorrer TODA la lista de postulantes, no se logran completar las vacantes de la carrera, se entra en una fase de llenado de cupos en base a los mejores postulantes no anteriormente inscritos. En un principio puede parecer incorrecto realizar esta acción, pero se debe destacar que las carreras están en orden de prioridad en el código secuencial, por lo que las primeras carreras serán las primeras en llenarse, con los mejores y su mejor opción, y luego si no basta, con los mejores puntajes de postulantes restante, este criterio fue tomado en base a la información de puntajes de corte del año anterior de cada carrera y cupos de cada una de estas. Lo anteriormente explicado corresponde a la última parte del código IF: c and lista[a][0]!=0).

El proceso anterior se repite 28 veces, una vez por carrera, y finalmente se obtiene toda la información necesaria para la creación del archivo xlsx de respuesta, el cual se genera en el siguiente bloque de código:

```
##### RESPUESTA DEL SERVIDOR AL CLIENTE

yield "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
yield "Matriculados Utem.xlsx"
generarXlsx(matriculadosPorCarreraRuts,matriculadosPorCarreraPuntajes,"Matriculados UTEM.xlsx")
dir_path = os.path.dirname(os.path.realpath(__file__))
contenidoArchivoCreado=open(dir_path+"/"+ "Matriculados UTEM.xlsx", 'rb').read()
# print(contenidoArchivoCreado)
contenidoArchivoCreado64=base64.b64encode(contenidoArchivoCreado).decode('UTF-8')
os.remove(dir_path+"/"+ "Matriculados UTEM.xlsx")
yield contenidoArchivoCreado64
```

Al mismo tiempo, se realiza la respuesta al cliente del Mime del archivo, el nombre del archivo y su contenido codificado en base64.

Finalmente, se presenta la declaración de la aplicación y el servidor :


```
application = Application([servicios], 'spyne.servicio.soap',
                           in_protocol=Soap11(validator='lxml'),
                           out_protocol=Soap11())

wsgi_application = WsgiApplication(application)

if __name__ == '__main__':
    import logging
    from wsgiref.simple_server import make_server

    logging.basicConfig(level=logging.DEBUG)
    logging.getLogger('spyne.protocol.xml').setLevel(logging.DEBUG)
    #El servidor esta escuchando por el puerto 8000
    #El documento wsd1 estara alojado en http://localhost:8000/?wsdl
    logging.info("listening to http://127.0.0.1:8000")
    logging.info("wsdl is at: http://localhost:8000/?wsdl")
    wsgi_app = WsgiApplication(application, chunked = True, max_content_length = 2097152*1000, block_length = 1024*1024*5000)
    server = make_server('127.0.0.1', 8000, wsgi_app)
    server.serve_forever()
```

Cliente:

```
from suds.client import Client
import base64, os, io

wsdl = Client('http://localhost:8000/?wsdl', timeout = 1200)

def codificar(nombreArchivo):
    #Detecta la posición real del archivo soap.py, para evitar errores de búsqueda de archivos en el mismo directorio.
    dir_path = os.path.dirname(os.path.realpath(__file__))
    # Lectura de archivo de puntajes sin codificar
    archivoPuntajes=open(dir_path+"/"+nombreArchivo, "r")
    texto=archivoPuntajes.read()
    archivoPuntajes.close()
    # Codificación
    base64_cadena_bytes = base64.b64encode(texto.encode('ascii'))
    base64_message = base64_cadena_bytes.decode('ascii')
    return base64_message

#####SOLICITUD Y RESPUESTA DE SERVIDOR

respuesta = wsdl.service.programaPrincipal("text/csv","puntajes.csv",codificar("puntajes.csv"))

if (respuesta[0][0]=="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    arregloDatos = base64.b64decode(respuesta[0][2])
    toread = io.BytesIO()
    toread.write(arregloDatos) # pass your `decrypted` string as the argument here
    toread.seek(0) # reset the pointer
    with open(dir_path+"/"+respuesta[0][1], "wb") as f:
        f.write(toread.read())
    f.close()
else:
    print("Error: Falla en MIME... ")
```

El cliente cuenta con un solo método, llamado “codificar”, y se utiliza para entregar los datos del archivo “puntajes.csv” en base64 en la solicitud al servidor. El cliente envía una solicitud con los datos del mismo, nombre y contenido del archivo al servidor, y la respuesta de este se guarda en la variable del mismo nombre. Finalmente, si la respuesta contiene el mime esperado, se realiza la reconstrucción del xlsx en base a todos los datos entregados por el servidor, trabajando la información del contenido de dicho archivo en BytesIO.

Información adicional

El proyecto fue probado en un equipo con las siguientes características:

Procesador: Intel i3-6100

Ram: 8 GB DDR 4 2400 MHZ

Y su tiempo de ejecución es de 8 min aproximadamente.