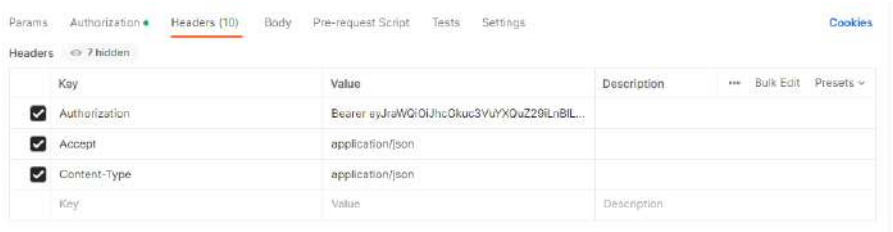
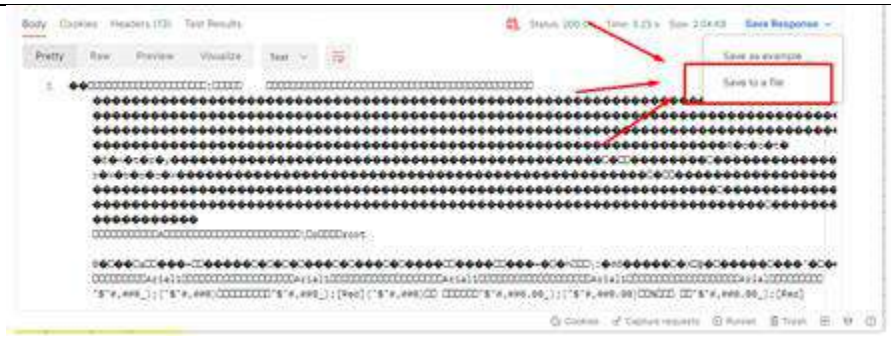
| | |
|---|---|
| | • 1061 - El campo "codTipoArchivo" es nulo o vacío<br>• 1028 – El campo “codOrigen” no enviado o es vacío<br>• 1029 – Código tipo de Origen de Envio no permitido o no valido<br>• 1030 - Solo se permite dato numérico de 1 dígito para el codOrigenEnvio<br>• 1138 - El campo "codProceso" es nulo o vacío<br>• 1139 – Código de Proceso no permitido o no valido |

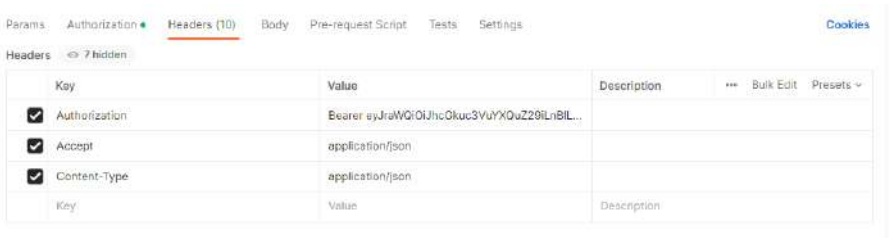
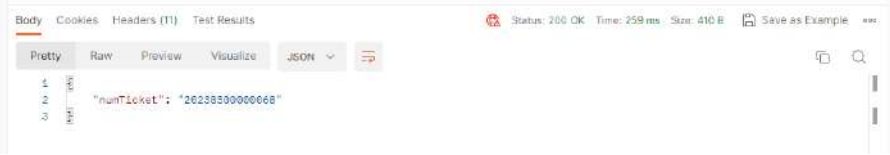
## 5.52 Servicio Web Api descargar reporte inconsistencias por periodo

| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api descargar inconsistencias por periodo |
| **Descripción** | Permite descargar las inconsistencias por periodo. |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/gestionconsultas/web/registrolibro/archivoreporte?nomArchivo={nomArchivo}&codTipoArchivoReporte={codTipoArchivoReporte}&codOrigen={codOrigen} |
| **Parámetros[URL]** | **Param-formato-tipo** / **Descripción**<br>nomArchivo: Nombre del archivo utilizado para la descarga o nombre de archivo generado (Obligatorio)<br>codTipoArchivoReporte-numérico-int: Extension del archivo a descargar (Ver Anexo III: Extension del archivo a descargar) (Obligatorio)<br>codOrigen-alfanumérico-string: Código de origen de envío: 2 Servicio web (Obligatorio) |
| **Parámetros[body]** | No aplica. |
| **Parámetros[header]** | **Descripción:**<br>Content-type: tipo de contenido a enviar<br>**Valores:**<br>Content-type: application/x-www-form-urlencoded<br>**Parámetros** / **Valor**<br>Content-Type: application/json<br>Accept: application/json<br>Authorization: Bearer token obtenido de la autenticación<br>Método: GET |
| **Parámetros[salida]** | **Parámetros de Salida** / **Descripcion**<br>buffer-binary-binary: buffer: Arreglo de bits |
| **Evidencias** | **URL**<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/gestionconsultas/web/registrolibro/archivoreporte?nomArchivo=LE20100176450202212000804000 2PCW2.zip&codTipoArchivoReporte=00&codOrigen=1<br>**Headers**<br>  <br>**Body**<br>(No aplica)<br>**Result OK** |

| | |
|---|---|
| |  <br>**Result Fail**<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." } |
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 1059 - Código tipo de Archivo no permitido o no valido<br>• 1060 - Solo se permite dato numérico de 1 dígito para el codTipoArchivo<br>• 1061 - El campo "codTipoArchivo" es nulo o vacío<br>• 1028 - El campo “codOrigen” no enviado o es vacío<br>• 1029 - Código tipo de Origen de Envio no permitido o no valido<br>• 1030 - Solo se permite dato numérico de 1 dígito para el codOrigenEnvio<br>• 2278 - El campo 'codTipoArchivoReporte' no enviado o es vacío |

## 5.53 Servicio Web Api descargar reporte CAR por periodo y fase

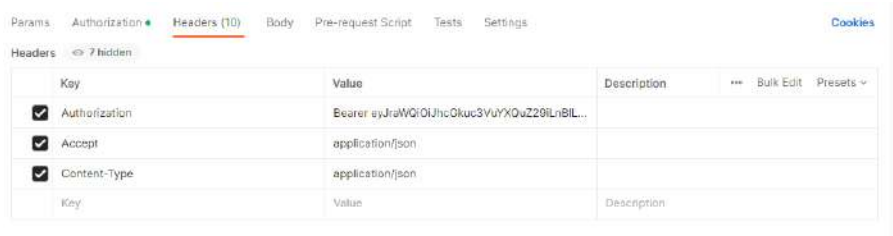
| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api descargar reporte de CAR |
| **Descripción** | Permite descargar la lista de CAR dependiendo de la fase en que se encuentre. |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/gestionlibro/web/comprobanteslibros/{perTributario}/reportecar?codOrigenEnvio={codOrigenEnvio}&codLibro={codLibro}&codFase={codFase} |
| **Parámetros[URL]** | **Param-formato-tipo** / **Descripción**<br>perTributario-alfanumérico-String: Periodo tributario (Obligatorio)<br>codLibro-alfanumérico-String: Código de libro: 080000 RCE (Obligatorio)<br>codFase-alfanumérico-String: Código de fase (Obligatorio)<br>codOrigen-alfanumérico-string: Código de origen de envío: 2 Servicio web (Obligatorio) |
| **Parámetros[body]** | No aplica. |
| **Parámetros[header]** | **Descripción:**<br>Content-type: tipo de contenido a enviar<br>**Valores:**<br>Content-type: application/x-www-form-urlencoded<br>**Parámetros** / **Valor**<br>Content-Type: application/json<br>Accept: application/json<br>Authorization: Bearer token obtenido de la autenticación<br>Método: GET |
| **Parámetros[salida]** | **Parámetros de Salida** / **Descripcion** / **Formato** / **Tipo dato**<br>numTicket: Número de ticket de envío [AAAA99999999]<br>AAAA: Año<br>99: Tipo de correlativo<br>99999999: Número correlativo de envío / alfanumerico / String |
| **Evidencias** | **URL** |

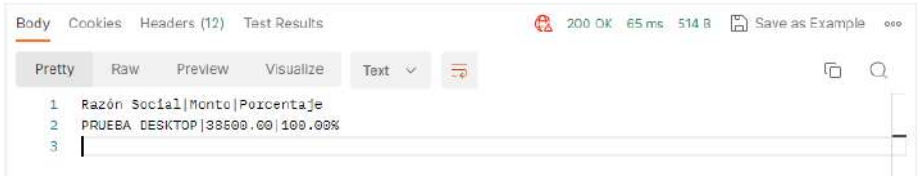
| | |
|---|---|
| | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/gestionlibro/web/comprobanteslibros/202302/reportecar?codOrigenEnvio=2&codLibro=080000&codFase=1<br>**Headers**<br>  <br>**Body**<br>(No aplica)<br>**Result OK**<br>  <br>**Result Fail**<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." } |
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1005 – El campo “perTributario” no enviado o es vacío<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 2002 - Solo se permite valor número para el campo "codFase"<br>• 2003 - El valor enviado para el campo "codFase" no es el correcto.<br>• 1140 - El campo “codLibro” no enviado o es vacío<br>• 1028 - El campo “codOrigen” no enviado o es vacío<br>• 1029 - Código tipo de Origen de Envio no permitido o no valido<br>• 1030 - Solo se permite dato numérico de 1 dígito para el codOrigenEnvio |

## 5.54 Servicio Web Api descargar reporte estadístico compras por proveedor por periodo

| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api descargar reporte estadístico compras por proveedor por periodo |
| **Descripción** | Permite exportar resumen estadístico |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc={numRuc}&numRuc={perTributario}&fechaini={fechaini}&?fechafin={fechafin}&numRucproveedor={numRucproveedor}&odTipoCDP={codTipoCDP}&codTipoArchivo={codTipoArchivo}&codTipoReporte={codTipoReporte}&codLibro={codLibro} |
| **Parámetros[URL]** | |

| Param-formato-tipo | Descripción |
|---|---|
| numRuc-alfanumérico-String | Número de RUC del generador (Opcional) |
| perTributario-numérico-int | Periodo tributario (Obligatorio) |
| fechaIni-dd/mm/yyyy-date | Fecha emision desde del comprobante de pago (Opcional) |
| fechaFin-dd/mm/yyyy-date | Fecha emision hasta del comprobante de pago (Opcional) |
| numRucProveedor-alfanumerico-alfanumerico | Numero del RUC o Documento de identidad del proveedor (Opcional) |
| codTipoCDP-alfanumerico-alfanumerico | Tipo de comprobante (Opcional) |
| codTipoArchivo-numérico-Integer | Extension del archivo a descargar (Ver Anexo |

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>III: Extension del archivo a descargar) (Obligatorio)</td></tr>
<tr><td>codTipoReporte-numérico-Integer</td><td>Código de tipo de reporte: 1 Reporte montos/proveedor (Obligatorio)</td></tr>
<tr><td>codLibro-alfanumérico-String</td><td>Código de libro: 080000 RCE (Obligatorio)</td></tr>
</table>
</td></tr>
<tr><td>Parámetros[body]</td><td>No aplica.</td></tr>
<tr><td>Parámetros[header]</td><td>Descripción:<br>Content-type: tipo de contenido a enviar<br>Valores:<br>Content-type: application/x-www-form-urlencoded
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Accept</td><td>application/json</td></tr>
<tr><td>Authorization</td><td>Bearer token obtenido de la autenticación</td></tr>
</table>
Método: GET</td></tr>
<tr><td>Parámetros[salida]</td><td>
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>HTTP status</td><td>200</td></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Content-Disposition</td><td>El sistema descarga el archivo con el siguiente formato de nombre:<br>1-REPORTE MONTOS/ PROVEEDOR (estadisticaPorProveedor.<extensión>)<br>Razón Social|Monto|Porcentaje<br>Los constructores SAC|127 000|16%<br>El ingeniero perez|86 999|9%<br>El consorcio unido|75 000|7%</td></tr>
</table>
</td></tr>
<tr><td>Evidencias</td><td>URL<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc=20195923753&perTributario=202203&codTipoArchivo=0&codTipoReporte=1&codLibro=080000<br>Headers<br> 

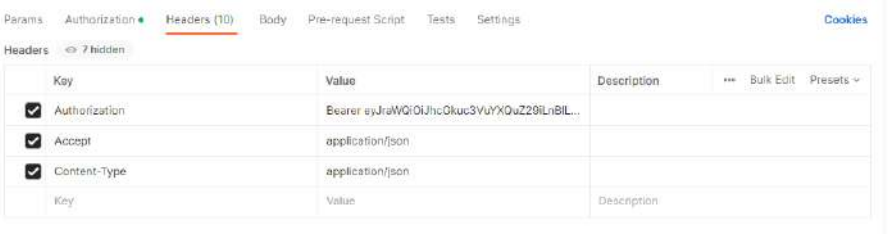
<br>Body<br>(No aplica)<br>Result OK<br> 

<br>Result Fail<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." }</td></tr>
<tr><td>Mensaje Error</td><td>{ "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>Lista de errores 422:
<ul>
<li>1005 – El campo “perTributario” no enviado o es vacío</li>
<li>1006 – Formato de perTributario no cumple con el formato “yyyymm”</li>
<li>1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual</li>
<li>1059 – Código tipo de Archivo no permitido o no valido</li>
<li>1060 - Solo se permite dato numérico de 1 dígito para el codTipoArchivo</li>
<li>1061 - El campo "codTipoArchivo" es nulo o vacío</li>
</ul></td></tr>
</table>

| | |
|---|---|
| | • 1333 - El campo "codTipoReporte" es nulo o vacío<br>• 1334 - El campo "codTipoReporte" solo admite valores: 1, 2, 3 ó 4<br>• 1325 - Fecha de inicio debe estar dentro del Periodo seleccionado<br>• 1115 - Debe cumplir con el siguiente formato “dd/mm/yyyy”.<br>• 1327 - Fecha Fin debe ser mayor o igual a la Fecha Inicio<br>• 1328 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Inicio, Fecha Fin<br>• 1329 - Fecha Fin debe estar dentro del Periodo seleccionado<br>• 1331 - Fecha Fin debe ser mayor o igual al Fecha Inicio<br>• 1332 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Fin, Fecha Inicio<br>• 1011 – El campo “codTipoCDP” no enviado o es vacío |

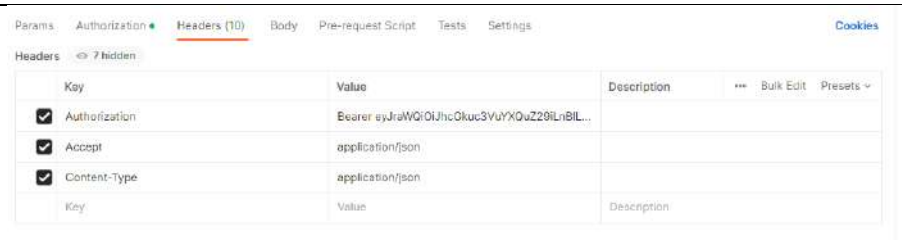
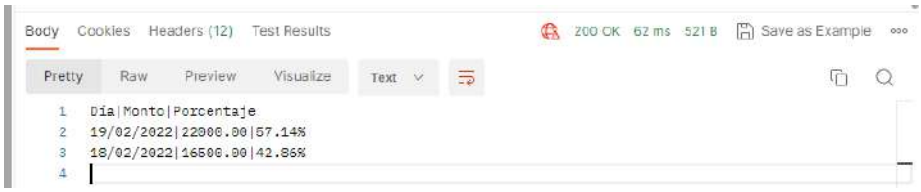
## 5.55 Servicio Web Api descargar reporte estadístico NC-ND por proveedor y periodo

<table>
<tr><td>Nombre Web Services</td><td>Servicio Web Api descargar reporte estadístico NC-ND por proveedor y periodo</td></tr>
<tr><td>Descripción</td><td>Permite exportar resumen estadístico</td></tr>
<tr><td>Url</td><td>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc={numRuc}&perTributario={perTributario}&fechaini={fechaini}&?fechafin={fechafin}&numRucproveedor={numRucproveedor}&odTipoCDP={codTipoCDP}&codTipoArchivo={codTipoArchivo}&codTipoReporte={codTipoReporte}&codLibro={codLibro}</td></tr>
<tr><td>Parámetros[URL]</td><td>
<table>
<tr><th>Param-formato-tipo</th><th>Descripción</th></tr>
<tr><td>numRuc-alfanumérico-String</td><td>Número de RUC del generador (Obligatorio)</td></tr>
<tr><td>perTributario-alfanumérico-String</td><td>Periodo tributario (Obligatorio)</td></tr>
<tr><td>fechaIni-dd/mm/yyyy-date</td><td>Fecha emision desde del comprobante de pago (Opcional)</td></tr>
<tr><td>fechaFin-dd/mm/yyyy-date</td><td>Fecha emision hasta del comprobante de pago (Opcional)</td></tr>
<tr><td>numRucProveedor-alfanumerico-alfanumerico</td><td>Numero del RUC o Documento de identidad del proveedor (Opcional)</td></tr>
<tr><td>codTipoCDP-alfanumerico-alfanumerico</td><td>Tipo de comprobante (Opcional)</td></tr>
<tr><td>codTipoArchivo-numérico-Integer</td><td>Extension del archivo a descargar (Ver Anexo III: Extension del archivo a descargar) (Obligatorio)</td></tr>
<tr><td>codTipoReporte-numérico-Integer</td><td>Código de tipo de reporte: 2 Reporte montos/Notas credito y notas de debito (Obligatorio)</td></tr>
<tr><td>codLibro-alfanumérico-String</td><td>Código de libro: 080000 RCE (Obligatorio)</td></tr>
</table>
</td></tr>
<tr><td>Parámetros[body]</td><td>No aplica.</td></tr>
<tr><td>Parámetros[header]</td><td>Descripción:<br>Content-type: tipo de contenido a enviar<br>Valores:<br>Content-type: application/x-www-form-urlencoded
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Accept</td><td>application/json</td></tr>
<tr><td>Authorization</td><td>Bearer token obtenido de la autenticación</td></tr>
</table>
Método: GET</td></tr>
<tr><td>Parámetros[salida]</td><td>
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>HTTP status</td><td>200</td></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Content-Disposition</td><td>El sistema descarga el archivo con el siguiente formato de nombre:<br>2-REPORTE MONTOS/NOTAS CREDITO Y NOTAS DE DEBITO (estadisticaPorProveedorNotaCreDeb.<extensión> )<br>Razón Social|Monto|Porcentaje</td></tr>
</table>
</td></tr>
</table>

| | | |
|---|---|---|
| | | Los constructores SAC\|12 000\|16%<br>El ingeniero perez\|8 999\|9%<br>El consorcio unido\|7 000\|7% |
| **Evidencias** | **URL**<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc=20195923753&perTributario=202201&codTipoArchivo=0&codTipoReporte=2&codLibro=080000<br>**Headers**<br>  <br>**Body**<br>(No aplica)<br>**Result OK**<br>  <br>**Result Fail**<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." } | |
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1001 – El campo “numRuc” no enviado o es vacío<br>• 1002 – Solo se permite dato numérico de 11 dígitos para el número de RUC.<br>• 1003 - El RUC ingresado no existe o no es válido<br>• 1005 – El campo “perTributario” no enviado o es vacío<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 1059 – Código tipo de Archivo no permitido o no valido<br>• 1060 - Solo se permite dato numérico de 1 dígito para el codTipoArchivo<br>• 1061 - El campo "codTipoArchivo" es nulo o vacío<br>• 1333 - El campo "codTipoReporte" es nulo o vacío<br>• 1334 - El campo "codTipoReporte" solo admite valores: 1, 2, 3 ó 4<br>• 1325 -Fecha de inicio debe estar dentro del Periodo seleccionado<br>• 1326 - Debe cumplir con el siguiente formato “dd/mm/yyyy”.<br>• 1327 - Fecha Fin debe ser mayor o igual a la Fecha Inicio<br>• 1328 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Inicio, Fecha Fin<br>• 1329 - Fecha Fin debe estar dentro del Periodo seleccionado<br>• 1330 - Debe cumplir con el siguiente formato “dd/mm/yyyy”.<br>• 1331 - Fecha Fin debe ser mayor o igual al Fecha Inicio<br>• 1332 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Fin, Fecha Inicio<br>• 1011 – El campo “codTipoCDP” no enviado o es vacío | |

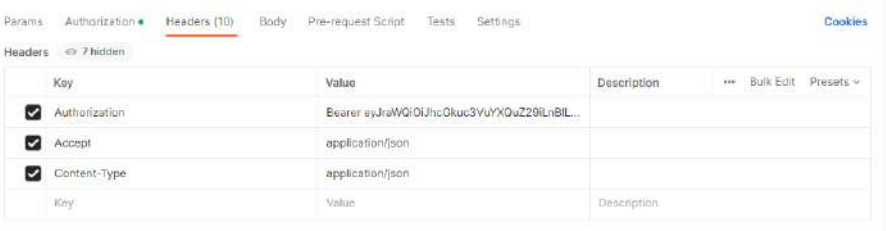
## 5.56 Servicio Web Api descargar reporte estadístico Compras por día y periodo

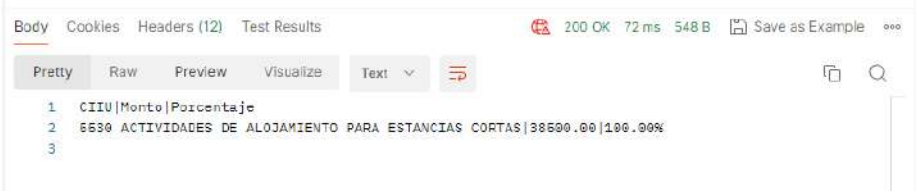
<table>
<tr><td>Nombre Web Services</td><td>Servicio Web Api descargar reporte estadístico Compras por día y periodo</td></tr>
<tr><td>Descripción</td><td>Permite exportar resumen estadístico</td></tr>
<tr><td>Url</td><td>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc={numRuc}&perTributario={perTributario}&fechaIni={fechaIni}&fechaFin={fechaFin}&numRucproveedor={numRucproveedor}&codTipoCDP={codTipoCDP}&codTipoArchivo={codTipoArchivo}&codTipoReporte={codTipoReporte}&codLibro={codLibro}</td></tr>
<tr><td>Parámetros[URL]</td><td>
<table>
<tr><th>Param-formato-tipo</th><th>Descripción</th></tr>
<tr><td>numRuc-alfanumérico-String</td><td>Número de RUC del generador (Obligatorio)</td></tr>
<tr><td>perTributario-numérico-int</td><td>Periodo tributario (Obligatorio)</td></tr>
<tr><td>fechaIni-dd/mm/yyyy-date</td><td>Fecha emision desde del comprobante de pago (Opcional)</td></tr>
<tr><td>fechaFin-dd/mm/yyyy-date</td><td>Fecha emision hasta del comprobante de pago (Opcional)</td></tr>
<tr><td>numRucProveedor-alfanumerico-alfanumerico</td><td>Numero del RUC o Documento de identidad del proveedor / sino se envía listar todos (Opcional)</td></tr>
<tr><td>codTipoCDP-alfanumerico-alfanumerico</td><td>Tipo de comprobante (Opcional)</td></tr>
<tr><td>codTipoArchivo-numérico-Integer</td><td>Extension del archivo a descargar (Ver Anexo III: Extension del archivo a descargar) (Obligatorio)</td></tr>
<tr><td>codTipoReporte-numérico-Integer</td><td>Código de tipo de reporte: 3 Reporte montos/dia (Obligatorio)</td></tr>
<tr><td>codLibro-alfanumérico-String</td><td>Código de libro: 080000 RCE (Obligatorio)</td></tr>
</table>
</td></tr>
<tr><td>Parámetros[body]</td><td>No aplica.</td></tr>
<tr><td>Parámetros[header]</td><td>Descripción:<br>Content-type: tipo de contenido a enviar<br>Valores:<br>Content-type: application/x-www-form-urlencoded
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Accept</td><td>application/json</td></tr>
<tr><td>Authorization</td><td>Bearer token obtenido de la autenticación</td></tr>
</table>
Método: GET</td></tr>
<tr><td>Parámetros[salida]</td><td>
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>HTTP status</td><td>200</td></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Content-Disposition</td><td>El sistema descarga el archivo con el siguiente formato de nombre:<br>3-REPORTE MONTOS/DIA<br>(estadisticaPorDia.<extensión>)<br>Día|Monto|Porcentaje<br>15|12 000|16%<br>20|8 999|9%<br>27|7 000|7%</td></tr>
</table>
</td></tr>
<tr><td>Evidencias</td><td>URL<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc=20195923753&perTributario=202203&codTipoArchivo=0&codTipoReporte=3&codLibro=080000<br>Headers</td></tr>
</table>

| | |
|---|---|
| |  <br>**Body**<br>(No aplica)<br>**Result OK**<br>  <br>**Result Fail**<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." } |
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1001 – El campo “numRuc” no enviado o es vacío<br>• 1002 – Solo se permite dato numérico de 11 dígitos para el número de RUC.<br>• 1003 - El RUC ingresado no existe o no es válido<br>• 1005 – El campo “perTributario” no enviado o es vacío<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 1059 – Código tipo de Archivo no permitido o no valido<br>• 1060 - Solo se permite dato numérico de 1 dígito para el codTipoArchivo<br>• 1061 - El campo "codTipoArchivo" es nulo o vacío<br>• 1333 - El campo "codTipoReporte" es nulo o vacío<br>• 1334 - El campo "codTipoReporte" solo admite valores: 1, 2, 3 ó 4<br>• 1325 -Fecha de inicio debe estar dentro del Periodo seleccionado<br>• 1115 - Debe cumplir con el siguiente formato dd/mm/yyyy.<br>• 1327 - Fecha Fin debe ser mayor o igual a la Fecha Inicio<br>• 1328 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Inicio, Fecha Fin<br>• 1346 - Fecha Fin debe ser menor o igual al Periodo seleccionado.<br>• 1332 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Fin, Fecha Inicio<br>• 1011 – El campo “codTipoCDP” no enviado o es vacío |

## 5.57 Servicio Web Api descargar reporte estadístico Compras por CIIU

| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api descargar reporte estadístico Compras por CIIU |
| **Descripción** | Permite exportar resumen estadístico |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc={numRuc}&perTributario={perTributario}&fechaIni={fechaIni}&fechaFin={fechaFin}&numRucproveedor={numRucproveedor}&odTipoCDP={codTipoCDP}&codTipoArchivo={codTipoArchivo}&codTipoReporte={codTipoReporte}&codLibro={codLibro} |
| **Parámetros[URL]** | **Param-formato-tipo** / **Descripción**<br>numRuc-alfanumérico-String / Número de RUC del generador (Obligatorio)<br>perTributario-numérico-int / Periodo tributario (Obligatorio) |

<table>
<tr><td></td><td>
<table>
<tr><td>fechaIni-dd/mm/yyyy-date</td><td>Fecha emision desde del comprobante de pago (Opcional)</td></tr>
<tr><td>fechaFin-dd/mm/yyyy-date</td><td>Fecha emision hasta del comprobante de pago (Opcional)</td></tr>
<tr><td>numRucProveedor-alfanumerico-alfanumerico</td><td>Numero del RUC o Documento de identidad del proveedor / sino se envía listar todos (Opcional)</td></tr>
<tr><td>codTipoCDP-alfanumerico-alfanumerico</td><td>Tipo de comprobante (Opcional)</td></tr>
<tr><td>codTipoArchivo-numérico-Integer</td><td>Extension del archivo a descargar (Ver Anexo III: Extensión del archivo a descargar). Códigos válidos: 0 y 1 (Obligatorio)</td></tr>
<tr><td>codTipoReporte-numérico-Integer</td><td>Código de tipo de reporte: 4 Reporte montos/CIIU (Obligatorio)</td></tr>
<tr><td>codLibro-alfanumérico-String</td><td>Código de libro: 080000 RCE (Obligatorio)</td></tr>
</table>
</td></tr>
<tr><td>Parámetros[body]</td><td>No aplica.</td></tr>
<tr><td>Parámetros[header]</td><td>Descripción:<br>Content-type: tipo de contenido a envia<br>Valores:<br>Content-type: application/x-www-form-urlencoded
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Accept</td><td>application/json</td></tr>
<tr><td>Authorization</td><td>Bearer token obtenido de la autenticación</td></tr>
</table>
Método: GET</td></tr>
<tr><td>Parámetros[salida]</td><td>
<table>
<tr><th>Parámetros</th><th>Valor</th></tr>
<tr><td>HTTP status</td><td>200</td></tr>
<tr><td>Content-Type</td><td>application/json</td></tr>
<tr><td>Content-Disposition</td><td>El sistema descarga el archivo con el siguiente formato de nombre:<br>4-REPORTE MONTOS/CIIU<br>(estadisticaPorCIIUProveedor.<extensión>)<br>CIIU|Monto|Porcentaje<br>4690 VENTA POR MAYOR|12 000|16%<br>4751 VENTA AL POR MENOR|8 999|9%<br>5510 ACTIVIDADES DE ALOJAMIENTO|7 000|7%</td></tr>
</table>
</td></tr>
<tr><td>Evidencias</td><td>URL<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/estadistica/web/resumenestadistico/exporta?numRuc=20195923753&perTributario=202203&codTipoArchivo=0&codTipoReporte=4&codLibro=080000<br>Headers<br> 

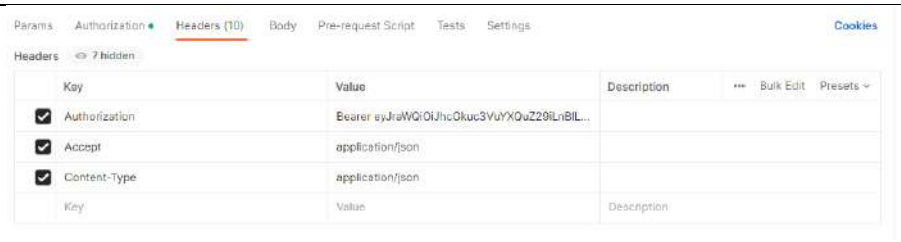
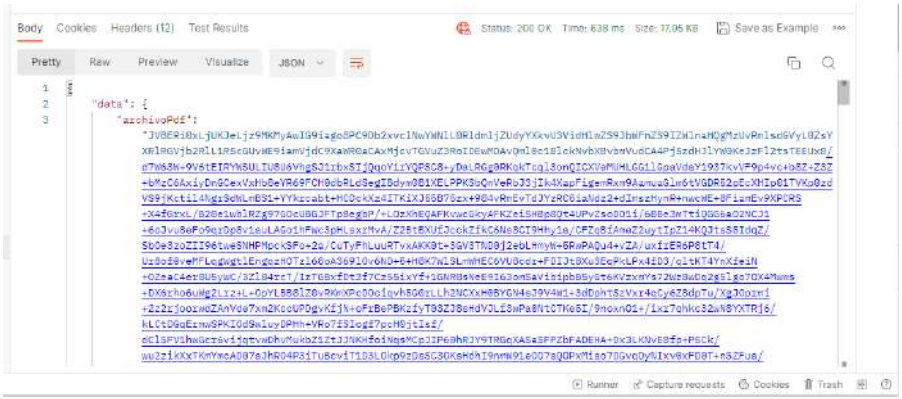
<br>Body<br>(No aplica)<br>Result OK<br> 

<br>Result Fail<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." }</td></tr>
</table>

| | |
|---|---|
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1001 – El campo “numRuc” no enviado o es vacío<br>• 1002 – Solo se permite dato numérico de 11 dígitos para el número de RUC.<br>• 1003 - El RUC ingresado no existe o no es válido<br>• 1005 – El campo “perTributario” no enviado o es vacío<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 1059 – Código tipo de Archivo no permitido o no valido<br>• 1060 - Solo se permite dato numérico de 1 dígito para el codTipoArchivo<br>• 1061 - El campo "codTipoArchivo" es nulo o vacío<br>• 1333 - El campo "codTipoReporte" es nulo o vacío<br>• 1334 - El campo "codTipoReporte" solo admite valores: 1, 2, 3 ó 4<br>• 1325 -Fecha de inicio debe estar dentro del Periodo seleccionado<br>• 1115 - Debe cumplir con el siguiente formato “dd/mm/yyyy”.<br>• 1327 - Fecha Fin debe ser mayor o igual a la Fecha Inicio<br>• 1328 - Si se realiza búsqueda por Fecha de emisión de cp, se debe ingresar los campos: Fecha Inicio, Fecha Fin<br>• 1329 - Fecha Fin debe estar dentro del Periodo seleccionado<br>• 1332 - Si se realiza busqueda por Fecha de emision de cp, se debe ingresar los campos: Fecha Fin, Fecha Inicio<br>• 1011 – El campo “codTipoCDP” no enviado o es vacío |

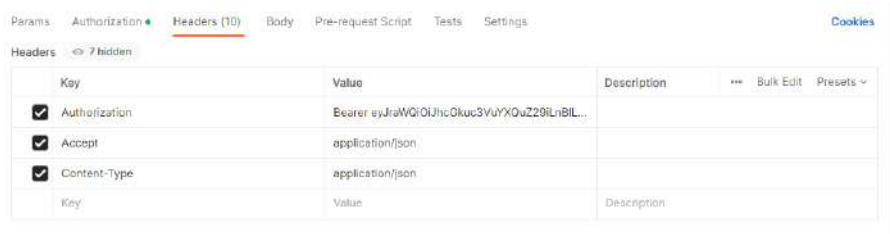
## 5.58 Servicio Web Api descargar reporte de cumplimiento

| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api descargar reporte de cumplimiento |
| **Descripción** | Permite descargar el reporte de cumplimiento. |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/cumplimiento/web/omisos/{perTributario}/{codLibro}/consultaReporteCumplimiento/exportardocumento |
| **Parámetros[URL]** | **Param-formato-tipo** / **Descripción**<br>perTributario-alfanumérico-String / Periodo tributario (Obligatorio)<br>codLibro-alfanumérico-String / Código de libro: 080000 RCE (Obligatorio) |
| **Parámetros[body]** | No aplica |
| **Parámetros[header]** | **Descripción:**<br>Content-type: tipo de contenido a enviar<br>**Valores:**<br>Content-type: application/x-www-form-urlencoded<br>**Parámetros** / **Valor**<br>Content-Type / application/json<br>Accept / application/json<br>Authorization / Bearer token obtenido de la autenticación<br>Método: GET |
| **Parámetros[salida]** | **Parámetros de Salida** / **Descripción**<br>archivoPdf-Base64-String / Base64 representando el archivo<br>nombreArchivoPdf-afanumérico-String / Nombre de archive de descarga |
| **Evidencias** | **URL**<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rvierce/cumplimiento/web/omisos/202305/080000/consultaReporteCumplimiento/exportardocumento<br>**Headers** |

| | |
|---|---|
| |  <br>**Body**<br>(No aplica)<br>**Result OK**<br>  <br>**Result Fail**<br>{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." } |
| **Mensaje Error** | { "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>**Lista de errores 422:**<br>• 1005 – El campo “perTributario” no enviado o es vacío<br>• 1006 – Formato de perTributario no cumple con el formato “yyyymm”<br>• 1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual<br>• 1140 – El campo “codLibro” no enviado o es vacío |

## 5.59 Servicio Web Api consultar ajustes posteriores RCE

| | |
|---|---|
| **Nombre Web Services** | Servicio Web Api consultar ajustes posteriores RCE |
| **Descripción** | Permite consultar el código de ajuste posterior y comprobantes del ajustes posterior RCE |
| **Url** | https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rce/propuesta/web?periodoSeleccionado={periodoSeleccionado}&tipoInfo={tipoInfo}<br>https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rce/ajustesposteriores/web/comprobantesajuspost/{periodoSeleccionado}/listarcap?page={page}&perPage={perPage} |
| **Parámetros[URL]** | **Param-formato-tipo** / **Descripción**<br>periodoSeleccionado -alfanumérico-String / Periodo tributario (Obligatorio)<br>page-numérico-entero / Número de pagina<br>perPage-numérico-entero / Número de registros a obtener |
| **Parámetros[body]** | No aplica. |
| **Parámetros[header]** | **Descripción:**<br>Content-type: tipo de contenido a enviar<br>**Valores:**<br>**Parámetros** / **valor**<br>Content-Type / application/json<br>Accept / application/json |

<table>
<tr><td></td><td>
<table>
<tr><td>Authorization</td><td>Bearer token obtenido de la autenticación</td></tr>
</table>
Método: PUT
</td></tr>
<tr><td>Parámetros[salida]</td><td>
<table>
<tr><th>Parámetros de Salida</th><th>Descripción</th></tr>
<tr><td>registros-array-array</td><td>Array de la Propuesta FV621 - inicio</td></tr>
<tr><td>registros.factprorrata-numérico-decimal128</td><td>Coeficiente de Prorrata</td></tr>
<tr><td>registros.valorRCF-numérico-decimal128</td><td>reintegro del crédito fiscal</td></tr>
<tr><td>registros.valorCFE-numérico-decimal128</td><td>crédito fiscal especial</td></tr>
<tr><td>registros-array-array</td><td>Array de la Propuesta FV621 - fin</td></tr>
</table>
</td></tr>
<tr><td>Evidencias</td><td>
URL<br>
https://api-sire.sunat.gob.pe/v1/contribuyente/migeigv/libros/rce/propuesta/web?periodoSeleccionado=202301&tipoInfo=FV0621<br>
Headers<br>


<br>
Body<br>
(No aplica)<br>
Result OK<br>


<br>
Result Fail<br>
{ "cod":"500", "msg":"Internal Server Error - Se presento una condicion inesperada que impidio completar el Request", "exc":"java.lang.NullPointerException at ..." }
</td></tr>
<tr><td>Mensaje Error</td><td>
{ "cod":"422", "msg":"Unprocessable Entity - Se presentaron errores de validacion que impidieron completar el Request", "errors":[ { "cod":"1001", "msg":"El campo “numRuc” no enviado o es vacío" }] }<br>
Lista de errores 422:
<ul>
<li>1005 - El campo “perTributario” no enviado o es vacío</li>
<li>1006 – Formato de perTributario no cumple con el formato “yyyymm”</li>
<li>1007 – El perTributario de búsqueda no debe ser mayor a la fecha actual</li>
</ul>
</td></tr>
</table>

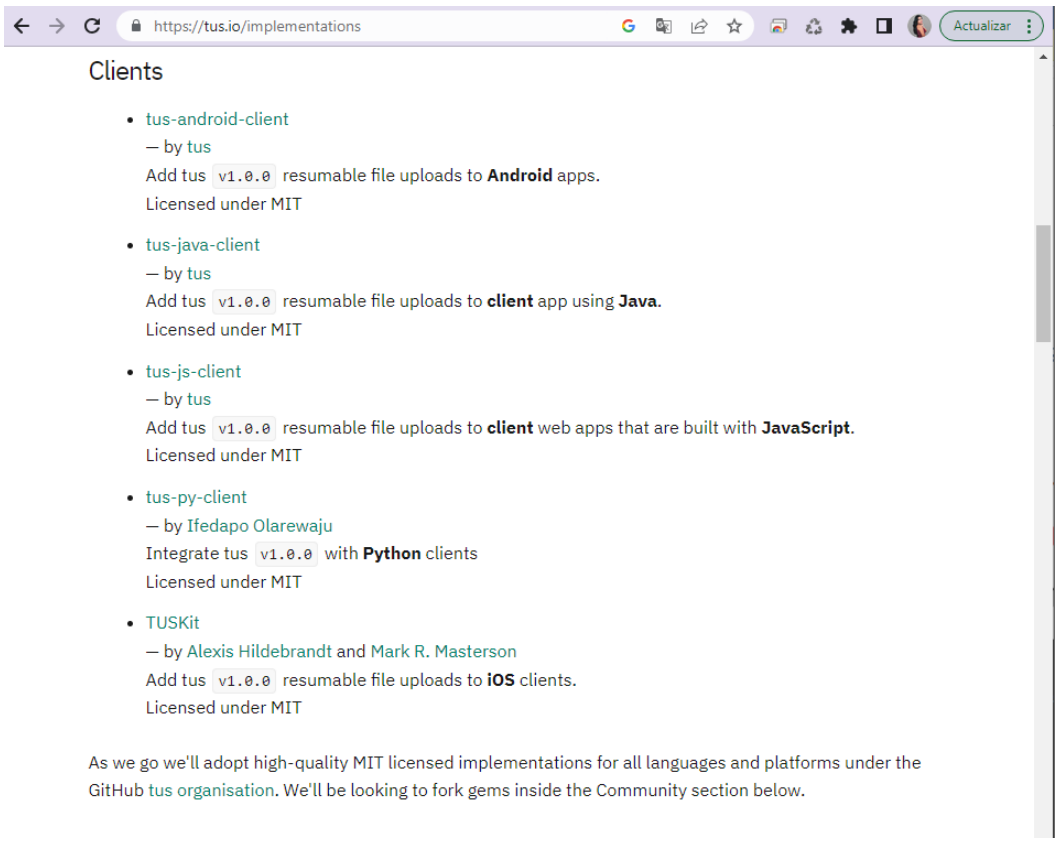
## 6. Documentación TUS.IO

Tus.io es un protocolo abierto para carga reanudable basado en HTTP el cual te permite poder implementarlo con cualquier lenguaje de programación, ya sea angular, java, Python, Android, etc. Pero, lo único que necesita es de la configuración de un servicio como servidor y otro como cliente.

Este manual api lo que proporciona es el endpoint servidor en donde los usuarios pueden generar sus peticiones para la importación/carga de archivos reanudables de manera rápida y segura.

Por lo tanto, el usuario que quiera cargar archivos a los servicios api mencionados en el apartado “5. Documentación Servicios Web API”, deberá configurar su cliente de acuerdo a su necesidad, por lo que podrá ubicar más detalle de ello en

la documentación que es propia del protocolo tus.io en la página web https://tus.io/implementations




Servicios Web Api que funcionan como servidor para la importación de archivos:

5.3 Servicio Web Api importar reemplazo de la propuesta
5.4 Servicio Web Api importar nuevos comprobantes propuesta
5.5 Servicio Web Api importar nuevos comprobantes preliminar
5.6 Servicio Web Api importar ajustes posteriores
5.7 Servicio Web Api importar ajustes posteriores de periodos anteriores

# 7. Anexos

## 7.1 Anexo I: Indicador de carga masiva

| Código | Descripción |
|---|---|
| 1 | Importar CP - Propuesta |
| 2 | Aceptar propuesta |
| 3 | Reemplazo de la Propuesta |
| 4 | Importa CP - Preliminar |
| 5 | Generar libro RVIE |
| 6 | Cargar Ajuste posteriores al periodo actual |
| 6 | Cargar Ajuste posteriores de periodos del sire |
| 7 | Cargar Ajuste posteriores anteriores a la vigencia |
| 8 | Generar registro Ajustes Posterior RVIE |
| 9 | Generar registro Ajustes Posterior Anterior RVIE |
| 10 | Generar archivo exportar propuesta |
| 11 | Generar archivo exportar no incluidos |
| 12 | Generar archivo exportar preliminar |
| 13 | Generar archivo exportar inconsistencias |

| | |
|---|---|
| 14 | Generar archivo exportar propuesta ajustes posteriores |
| 15 | Generar archivo exportar CAR |
| 16 | Generar reporte de observaciones de comparación |
| 17 | Generar archivo exportar Libro Venta |
| 18 | Generar reporte de ajustes posteriores individual |
| 19 | Generar reporte de ajustes posteriores consolidado |
| 20 | Generar reporte de ajustes posteriores de periodos anteriores individual |
| 21 | Generar reporte de ajustes posteriores de periodos anteriores consolidado |
| 22 | Generar reporte consolidado del libro y ajustes |
| 23 | Generar reporte Libro RVIE |
| 24 | Generar Archivo personalizado Libros RVIE |
| 25 | Generar Archivo personalizado Propuesta RVIE |
| 26 | Generar Archivo personalizado Ajustes Posteriores RVIE |
| 27 | Carga archivo de comparación - validación |
| 28 | Generar archivo exportar preliminar registrado |
| 29 | Generar archivo exportar preliminar ajustes posteriores registrado |
| 30 | Generar reporte de inconsistencias generación del RVIE |
| 31 | Generar reporte de inconsistencias ajustes posteriores del RVIE |
| 32 | Generar libro RVIE - Archivo exportar Libro Venta |
| 33 | Generar libro RVIE - Archivo reporte inconsistencias |
| 34 | Generar libro RVIE - Achivo Reporte Exportadores |
| 35 | Generar libro RVIE - Archivo Propuesta Casillas |
| 36 | Generar Ajustes Posteriroes RVIE - Archivo exportar Ajuste |
| 37 | Generar Ajustes Posteriroes RVIE - Archivo reporte inconsistencias |
| 38 | Generar Ajustes Posteriores de periodos anteriores RVIE - Archivo exportar Ajuste |
| 39 | Aceptar propuesta sin Movimiento |
| 40 | Carga Tipo de Cambio |
| 41 | Generar reportes Estadisticos |
| 42 | Generar Reporte de comparación |
| 43 | Descargar Registros Electronicos RVIE |
| 44 | Descargar Registros Electronicos RCE |
| 45 | Descargar Constancia de Recepción RVIE |
| 46 | Descargar Constancia de Recepción RCE |
| 47 | Descargar Reporte de Inconcistencias RVIE |
| 48 | Descargar Reporte de Inconcistencias RCE |
| 49 | Descargar Reporte de Ajustes Posteriores RVIE |
| 50 | Descargar Reporte de Ajustes Posteriores RCE |
| 51 | Descargar Reporte de Casillas Vista Comparada |
| 52 | Descargar Reporte de Inconsistencias de Casillas |
| 53 | Generar Reporte de comparación RCE |
| 54 | Carga Complementar |
| 55 | Carga Incluir Excluir |
| 56 | Carga No Domiciliados |
| 57 | Carga Comparacion RCE |
| 58 | Carga Comparacion RVIE |
| 59 | importar CP en Ajustes Posteriores RCE |
| 60 | importar CP no domiciliados en Ajustes Posteriores |
| 61 | Reemplazo de la Propuesta |
| 62 | Generacion de Inconsistencia por Casilla |
| 63 | Generación de ventas por Casilla (100. 101) |
| 64 | Generacion Inconsistencias en Registros para Casillas |
| 65 | Validar Propuesta |
| 66 | Validar Preliminar |
| 67 | Validar No Domiciliados |
| 68 | Reporte de Ajustes posteriores de periodos anteriores del RCE |
| 69 | Descarga Consolidada de registros del RCE |
| 70 | Descarga RCE |
| 71 | Reporte de ajustes posteriores del RVIE |
| 72 | Reporte de Ajustes posteriores de periodos anteriores del RVIE |

| | |
|---|---|
| 73 | Descarga Consolidada de registros del RVIE |
| 74 | Descarga RVIE |
| 75 | Generación de ventas por Casilla (100. 101) |
| 76 | Generacion Inconsistencias en Registros para Casillas |
| 77 | Validar Propuesta |
| 78 | Validar Preliminar |
| 79 | Validar No Domiciliados |
| 80 | Generación de archivo personalizado Propuesta RCE |
| 81 | Generación de archivo personalizado Preliminar RCE |
| 82 | Generación de archivo personalizado Preliminar Registrado RCE |
| 83 | Generación de archivo personalizado Registro Compras |
| 84 | Generación de archivo personalizado Ajuste Posterior RCE |
| 85 | Generacion de archivo del libro de Ajustes Posteriores RVIE |
| 86 | Generacion de archivo de inconsistencias de libro de Ajustes Posteriores RVIE |
| 87 | Importar CP en Ajustes Posteriores RVIE |
| 88 | Importar CP en Ajustes Posteriores de periodos anteriores RVIE general |
| 89 | Importar CP en Ajustes Posteriores de periodos anteriores RVIE simplificado |
| 90 | Generación de documentos para Intranet |
| 91 | Exportar detalle propuesta casilla - Registro |
| 92 | Exportar inconsistencias en registro |
| 93 | Importar CP en Ajustes Posteriores RCE de Periodos Anteriores Simplificado |
| 94 | Importar CP en Ajustes Posteriores RCE de Periodos Anteriores General |
| 95 | Importar CP no domiciliados en Ajustes Posteriores RCE de Periodos Anteriores |
| 96 | Generar archivo exportar preliminar - RCE No Domiciliados |
| 97 | Exportar comprobantes excluidos |

## 7.2 Anexo II: Tipo de ajuste posterior

| Código | Descripción |
|---|---|
| 1 | Ajuste Posterior |
| 2 | Ajuste Posterior con No Domiciliados |
| 3 | Ajuste Posteriores de periodos anteriores general |
| 4 | Ajuste Posteriores de periodos anteriores simplificado |
| 5 | Ajuste Posteriores de periodos anteriores con No Domiciliados |

## 7.3 Anexo III: Extension del archivo a descargar

| Código | Descripción |
|---|---|
| 0 | txt |
| 1 | csv |
| 2 | excel |

## 7.4 Anexo IV: Ejemplo cliente TUS JAVA

Para poder hacer uso de las API-REST de carga de archivos **SIRE** es necesario tener configurado en su proyecto la librería "**tus-java-client"** en la versión "**0-5-0**", la cual se puede encontrar en el repositorio Maven https://repo1.maven.org/maven2/io/tus/java/client/tus-java-client/0.5.0/tus-java-client-0.5.0.jar
Esta librería tiene algunas deficiencias para la gestión de los mensajes de retorno y error.
Por lo cual se tienen que hacer algunas adecuaciones para que se muestren de forma correcta los errores proporcionados por el servicio SIRE.

1. Crear el paquete: **io.tus.java.client** y dentro crear las clases:
   - TusResponseBody.java: transforma los errores a texto que se puede imprimir en la bitácora.
   - Http401And403CodeException.java: Muestra los errores de autenticación o de autorización
   - Http422CodeException.java: Muestra los errores de validación de negocio
   - HttpErrorCodeException.java: Clase de ayuda para la gestión de errores

- TusClientCustom.java: Cliente tus personalizado que permite ver la gestión de errores.
- TusUploaderCustom.java: Uploader tus personalizado que permite ver la gestión de errores.

2. Crear la clase: Demo.java (puede usar el paquete que desee): se usará para realizar la operación de carga de archivos del SIRE.

## Clase:: TusResponseBody.java

```
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.Serializable;
import java.net.HttpURLConnection;

public class TusResponseBody implements Serializable {

  private int responseCode;
  private String responseMessage;
  private String responseBody;

  public TusResponseBody(HttpURLConnection connection)
    throws IOException {
    if (connection != null) {
      responseCode = connection.getResponseCode();
      responseBody = readResponseBody(connection);
      responseMessage = connection.getResponseMessage();
    }
  }

  public int getResponseCode() {
```

```
    return responseCode;
  }

  public String getResponseBody() {
    return responseBody;
  }

  public String getResponseMessage() {
    return responseMessage;
  }

  private String readResponseBody (
    HttpURLConnection connection
  ) throws IOException {

    ByteArrayOutputStream result = new ByteArrayOutputStream();
    byte[] buffer = new byte[1024];
    int length;
    InputStream inputStream;
    if (connection.getErrorStream() != null) {
      inputStream = connection.getErrorStream();
    } else {
      try {
        inputStream = connection.getInputStream();
      } catch (IOException e) {
        return e.getMessage();
      }
    }

    if (inputStream == null) {
      return "";
    }
    try {
      while ((length = inputStream.read(buffer)) != -1) {
        result.write(buffer, 0, length);
      }
      connection.disconnect();
    } catch (IOException e) {
      return e.getMessage();
    }
    return result.toString("UTF-8");


  }

  @Override
  public String toString() {
    return responseBody.isEmpty()? responseMessage: responseBody;
  }
```

```
}
```

Clase:: Http401And403CodeException.java

```java
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.net.HttpURLConnection;

public class Http401And403CodeException extends HttpErrorCodeException {

  public Http401And403CodeException(
    TusResponseBody response,
    HttpURLConnection connection
  ) {
    super(response, connection);
  }
}
```

Clase:: Http422CodeException.java

```java
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.io.IOException;
import java.net.HttpURLConnection;

public class Http422CodeException extends HttpErrorCodeException {
  public Http422CodeException(
    TusResponseBody response,
    HttpURLConnection connection
  ) throws IOException {
    super(response, connection);
  }
}
```

Clase:: HttpErrorCodeException.java

```java
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.net.HttpURLConnection;

public class HttpErrorCodeException extends ProtocolException {

  private static final long serialVersionUID = 0l;

  private final TusResponseBody response;

  public HttpErrorCodeException(
    TusResponseBody response,
    HttpURLConnection connection
  ) {
    super(response.getResponseMessage(), connection);
    this.response = response;
  }

  public TusResponseBody getResponseBody() {
    return response;
  }


}
```

Clase:: TusClientCustom.java

```java
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.net.Proxy;
import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Map;

/**
 * This class is used for creating or resuming uploads.
 */
public class TusClientCustom {
    /**
     * Version of the tus protocol used by the client. The remote server
needs to support this
     * version, too.
     */
    public static final String TUS_VERSION = "1.0.0";

    private URL uploadCreationURL;
    private Proxy proxy;
    private boolean resumingEnabled;
    private boolean removeFingerprintOnSuccessEnabled;
    private TusURLStore urlStore;
    private Map<String, String> headers;
    private int connectTimeout = 5000;
```

```
    /**
     * Create a new tus client.
     */
    public TusClientCustom() {

    }

    /**
     * Set the URL used for creating new uploads. This is required if you
want to initiate new
     * uploads using {@link #createUpload} or {@link #resumeOrCreateUpload}
but is not used if you
     * only resume existing uploads.
     *
     * @param uploadCreationURL Absolute upload creation URL
     */
    public void setUploadCreationURL(URL uploadCreationURL) {
        this.uploadCreationURL = uploadCreationURL;
    }

    /**
     * Get the current upload creation URL.
     *
     * @return Current upload creation URL
     */
    public URL getUploadCreationURL() {
        return uploadCreationURL;
    }

    /**
     * Set the proxy that will be used for all requests.
     *
     * @param proxy Proxy to use
     */
    public void setProxy(Proxy proxy) {
        this.proxy = proxy;
    }

    /**
     * Get the current proxy used for all requests.
     *
     * @return Current proxy
     */
    public Proxy getProxy() {
        return proxy;
    }

    /**
```

```
     * Enable resuming already started uploads. This step is required if you
want to use
     * {@link #resumeUpload(TusUpload)}.
     *
     * @param urlStore Storage used to save and retrieve upload URLs by its
fingerprint.
     */
    public void enableResuming(@NotNull TusURLStore urlStore) {
        resumingEnabled = true;
        this.urlStore = urlStore;
    }

    /**
     * Disable resuming started uploads.
     *
     * @see #enableResuming(TusURLStore)
     */
    public void disableResuming() {
        resumingEnabled = false;
        this.urlStore = null;
    }

    /**
     * Get the current status if resuming.
     *
     * @see #enableResuming(TusURLStore)
     * @see #disableResuming()
     *
     * @return True if resuming has been enabled using {@link
#enableResuming(TusURLStore)}
     */
    public boolean resumingEnabled() {
        return resumingEnabled;
    }

    /**
     * Enable removing fingerprints after a successful upload.
     *
     * @see #disableRemoveFingerprintOnSuccess()
     */
    public void enableRemoveFingerprintOnSuccess() {
        removeFingerprintOnSuccessEnabled = true;
    }

    /**
     * Disable removing fingerprints after a successful upload.
     *
     * @see #enableRemoveFingerprintOnSuccess()
     */
```

```
    public void disableRemoveFingerprintOnSuccess() {
        removeFingerprintOnSuccessEnabled = false;
    }

    /**
     * Get the current status if removing fingerprints after a successful
upload.
     *
     * @see #enableRemoveFingerprintOnSuccess()
     * @see #disableRemoveFingerprintOnSuccess()
     *
     * @return True if resuming has been enabled using {@link
#enableResuming(TusURLStore)}
     */
    public boolean removeFingerprintOnSuccessEnabled() {
        return removeFingerprintOnSuccessEnabled;
    }


    /**
     * Set headers which will be added to every HTTP requestes made by this
TusClient instance.
     * These may to overwrite tus-specific headers, which can be identified
by their Tus-*
     * prefix, and can cause unexpected behavior.
     *
     * @see #getHeaders()
     * @see #prepareConnection(HttpURLConnection)
     *
     * @param headers The map of HTTP headers
     */
    public void setHeaders(@Nullable Map<String, String> headers) {
        this.headers = headers;
    }

    /**
     * Get the HTTP headers which should be contained in every request and
were configured using
     * {@link #setHeaders(Map)}.
     *
     * @see #setHeaders(Map)
     * @see #prepareConnection(HttpURLConnection)
     *
     * @return The map of configured HTTP headers
     */
    @Nullable
    public Map<String, String> getHeaders() {
        return headers;
    }
```

```
    /**
     * Sets the timeout for a Connection.
     * @param timeout in milliseconds
     */
    public void setConnectTimeout(int timeout) {
        connectTimeout = timeout;
    }

    /**
     * Returns the Connection Timeout.
     * @return Timeout in milliseconds.
     */
    public int getConnectTimeout() {
        return connectTimeout;
    }

    /**
     * Create a new upload using the Creation extension. Before calling this
function, an "upload
     * creation URL" must be defined using {@link #setUploadCreationURL(URL)}
or else this
     * function will fail.
     * In order to create the upload a POST request will be issued. The
file's chunks must be
     * uploaded manually using the returned {@link TusUploader} object.
     *
     * @param upload The file for which a new upload will be created
     * @return Use {@link TusUploader} to upload the file's chunks.
     * @throws ProtocolException Thrown if the remote server sent an
unexpected response, e.g.
     * wrong status codes or missing/invalid headers.
     * @throws IOException Thrown if an exception occurs while issuing the
HTTP request.
     */
    public TusUploaderCustom createUpload(@NotNull TusUpload upload) throws
ProtocolException, IOException {
        HttpURLConnection connection = openConnection(uploadCreationURL);
        connection.setRequestMethod("POST");
        prepareConnection(connection);

        String encodedMetadata = upload.getEncodedMetadata();
        if (encodedMetadata.length() > 0) {
            connection.setRequestProperty("Upload-Metadata",
encodedMetadata);
        }

        connection.addRequestProperty("Upload-Length",
Long.toString(upload.getSize()));
```

```
        connection.connect();

        int responseCode = connection.getResponseCode();
        if (responseCode == 401 || responseCode == 403) {
            throw new Http401And403CodeException(new
TusResponseBody(connection), connection);
        }

        if(responseCode == 422) {
            throw new Http422CodeException(new TusResponseBody(connection),
connection);
        }

        if(!(responseCode >= 200 && responseCode < 300)) {
            throw new ProtocolException("unexpected status code (" +
responseCode + ") while creating upload", connection);
        }

        String urlStr = connection.getHeaderField("Location");
        if (urlStr == null || urlStr.length() == 0) {
            throw new ProtocolException("missing upload URL in response for
creating upload", connection);
        }

        // The upload URL must be relative to the URL of the request by which
is was returned,
        // not the upload creation URL. In most cases, there is no difference
between those two
        // but there may be cases in which the POST request is redirected.
        URL uploadURL = new URL(connection.getURL(), urlStr);

        if (resumingEnabled) {
            urlStore.set(upload.getFingerprint(), uploadURL);
        }

        return createUploader(upload, uploadURL, 0L);
    }

    @NotNull
    private HttpURLConnection openConnection(@NotNull URL uploadURL) throws
IOException {
        if (proxy != null) {
            return (HttpURLConnection) uploadURL.openConnection(proxy);
        }
        return (HttpURLConnection) uploadURL.openConnection();
    }

    @NotNull
```

```
    private TusUploaderCustom createUploader(@NotNull TusUpload upload,
@NotNull URL uploadURL, long offset)
        throws IOException {
        TusUploaderCustom uploader = new TusUploaderCustom(this, upload,
uploadURL, upload.getTusInputStream(), offset);
        uploader.setProxy(proxy);
        return uploader;
    }

    /**
     * Try to resume an already started upload. Before call this function,
resuming must be
     * enabled using {@link #enableResuming(TusURLStore)}. This method will
look up the URL for this
     * upload in the {@link TusURLStore} using the upload's fingerprint (see
     * {@link TusUpload#getFingerprint()}). After a successful lookup a HEAD
request will be issued
     * to find the current offset without uploading the file, yet.
     *
     * @param upload The file for which an upload will be resumed
     * @return Use {@link TusUploader} to upload the remaining file's chunks.
     * @throws FingerprintNotFoundException Thrown if no matching fingerprint
has been found in
     * {@link TusURLStore}. Use {@link #createUpload(TusUpload)} to create a
new upload.
     * @throws ResumingNotEnabledException Throw if resuming has not been
enabled using {@link
     * #enableResuming(TusURLStore)}.
     * @throws ProtocolException Thrown if the remote server sent an
unexpected response, e.g.
     * wrong status codes or missing/invalid headers.
     * @throws IOException Thrown if an exception occurs while issuing the
HTTP request.
     */
    public TusUploaderCustom resumeUpload(@NotNull TusUpload upload) throws
            FingerprintNotFoundException, ResumingNotEnabledException,
ProtocolException, IOException {
        if (!resumingEnabled) {
            throw new ResumingNotEnabledException();
        }

        URL uploadURL = urlStore.get(upload.getFingerprint());
        if (uploadURL == null) {
            throw new FingerprintNotFoundException(upload.getFingerprint());
        }

        return beginOrResumeUploadFromURL(upload, uploadURL);
    }
```

```
    /**
     * Begin an upload or alternatively resume it if the upload has already
been started before. In contrast to
     * {@link #createUpload(TusUpload)} and {@link
#resumeOrCreateUpload(TusUpload)} this method will not create a new
     * upload. The user must obtain the upload location URL on their own as
this method will not send the POST request
     * which is normally used to create a new upload.
     * Therefore, this method is only useful if you are uploading to a
service which takes care of creating the tus
     * upload for yourself. One example of such a service is the Vimeo API.
     * When called a HEAD request will be issued to find the current offset
without uploading the file, yet.
     * The uploading can be started by using the returned {@link TusUploader}
object.
     *
     * @param upload The file for which an upload will be resumed
     * @param uploadURL The upload location URL at which has already been
created and this file should be uploaded to.
     * @return Use {@link TusUploader} to upload the remaining file's chunks.
     * @throws ProtocolException Thrown if the remote server sent an
unexpected response, e.g.
     * wrong status codes or missing/invalid headers.
     * @throws IOException Thrown if an exception occurs while issuing the
HTTP request.
     */
    public TusUploaderCustom beginOrResumeUploadFromURL(@NotNull TusUpload
upload, @NotNull URL uploadURL) throws
            ProtocolException, IOException {
        HttpURLConnection connection = openConnection(uploadURL);
        connection.setRequestMethod("HEAD");
        prepareConnection(connection);

        connection.connect();

        int responseCode = connection.getResponseCode();

        if (responseCode == 401 || responseCode == 403) {
            throw new Http401And403CodeException(new
TusResponseBody(connection), connection);
        }

        if(responseCode == 422) {
            throw new Http422CodeException(new TusResponseBody(connection),
connection);
        }

        if(!(responseCode >= 200 && responseCode < 300)) {
```

```
            throw new HttpErrorCodeException(new TusResponseBody(connection),
connection);
        }

        String offsetStr = connection.getHeaderField("Upload-Offset");
        if (offsetStr == null || offsetStr.length() == 0) {
            throw new ProtocolException("missing upload offset in response
for resuming upload", connection);
        }
        long offset = Long.parseLong(offsetStr);

        return createUploader(upload, uploadURL, offset);
    }

    /**
     * Try to resume an upload using {@link #resumeUpload(TusUpload)}. If the
method call throws
     * an {@link ResumingNotEnabledException} or {@link
FingerprintNotFoundException}, a new upload
     * will be created using {@link #createUpload(TusUpload)}.
     *
     * @param upload The file for which an upload will be resumed
     * @throws ProtocolException Thrown if the remote server sent an
unexpected response, e.g.
     * wrong status codes or missing/invalid headers.
     * @throws IOException Thrown if an exception occurs while issuing the
HTTP request.
     * @return {@link TusUploader} instance.
     */
    public TusUploaderCustom resumeOrCreateUpload(@NotNull TusUpload upload)
throws ProtocolException, IOException {
        try {
            return resumeUpload(upload);
        } catch (FingerprintNotFoundException e) {
            return createUpload(upload);
        } catch (ResumingNotEnabledException e) {
            return createUpload(upload);
        } catch (ProtocolException e) {
            // If the attempt to resume returned a 404 Not Found, we
immediately try to create a new
            // one since TusExectuor would not retry this operation.
            HttpURLConnection connection = e.getCausingConnection();
            if (connection != null && connection.getResponseCode() == 404) {
                return createUpload(upload);
            }

            throw e;
        }
    }
```

```
    /**
     * Set headers used for every HTTP request. Currently, this will add the
Tus-Resumable header
     * and any custom header which can be configured using {@link
#setHeaders(Map)},
     *
     * @param connection The connection whose headers will be modified.
     */
    public void prepareConnection(@NotNull HttpURLConnection connection) {
        // Only follow redirects, if the POST methods is preserved. If
http.strictPostRedirect is
        // disabled, a POST request will be transformed into a GET request
which is not wanted by us.

        // CHECKSTYLE:OFF
        // LineLength - Necessary because of length of the link
        // See:https://github.com/openjdk/jdk/blob/jdk7-
b43/jdk/src/share/classes/sun/net/www/protocol/http/HttpURLConnection.java#L2
020-L2035
        // CHECKSTYLE:ON
        connection.setInstanceFollowRedirects(Boolean.getBoolean("http.strict
PostRedirect"));

        connection.setConnectTimeout(connectTimeout);
        connection.addRequestProperty("Tus-Resumable", TUS_VERSION);

        if (headers != null) {
            for (Map.Entry<String, String> entry : headers.entrySet()) {
                connection.addRequestProperty(entry.getKey(),
entry.getValue());
            }
        }
    }

    /**
     * Actions to be performed after a successful upload completion.
     * Manages URL removal from the URL store if remove fingerprint on
success is enabled
     *
     * @param upload that has been finished
     */
    protected void uploadFinished(@NotNull TusUpload upload) {
        if (resumingEnabled && removeFingerprintOnSuccessEnabled) {
            urlStore.remove(upload.getFingerprint());
        }
    }
}
```

Clase: TusUploaderCustom.java

```java
/**
 *
 * Copyright 2011-2024 the original author or authors.
 *  *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *  *
 *      https://www.apache.org/licenses/LICENSE-2.0
 *  *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */
package io.tus.java.client;

import java.io.IOException;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.Proxy;
import java.net.URL;
import java.net.URLConnection;
import java.util.Optional;

/**
 * This class is used for doing the actual upload of the files. Instances are
returned by
 * {@link TusClientCustom#createUpload(TusUpload)}, {@link
TusClientCustom#createUpload(TusUpload)} and
 * {@link TusClientCustom#resumeOrCreateUpload(TusUpload)}.
 * <br>
 * After obtaining an instance you can upload a file by following these
steps:
 * <ol>
 *  <li>Upload a chunk using {@link #uploadChunk()}</li>
 *  <li>Optionally get the new offset ({@link #getOffset()} to calculate the
progress</li>
 *  <li>Repeat step 1 until the {@link #uploadChunk()} returns -1</li>
 *  <li>Close HTTP connection and InputStream using {@link #finish()} to free
resources</li>
 * </ol>
 */
public class TusUploaderCustom {
```

```
    private URL uploadURL;
    private Proxy proxy;
    private TusInputStream input;
    private long offset;
    private TusClientCustom client;
    private TusUpload upload;
    private byte[] buffer;
    private int requestPayloadSize = 10 * 1024 * 1024;
    private int bytesRemainingForRequest;

    private HttpURLConnection connection;
    private OutputStream output;

    /**
     * Begin a new upload request by opening a PATCH request to specified
upload URL. After this
     * method returns a connection will be ready and you can upload chunks of
the file.
     *
     * @param client Used for preparing a request ({@link
TusClient#prepareConnection(HttpURLConnection)}
     * @param upload {@link TusUpload} to be uploaded.
     * @param uploadURL URL to send the request to
     * @param input Stream to read (and seek) from and upload to the remote
server
     * @param offset Offset to read from
     * @throws IOException Thrown if an exception occurs while issuing the
HTTP request.
     */
    public TusUploaderCustom(TusClientCustom client, TusUpload upload, URL
uploadURL, TusInputStream input, long offset)
        throws IOException {
        this.uploadURL = uploadURL;
        this.input = input;
        this.offset = offset;
        this.client = client;
        this.upload = upload;

        input.seekTo(offset);

        setChunkSize(2 * 1024 * 1024);
    }

    private void openConnection() throws IOException, ProtocolException {
        // Only open a connection, if we have none open.
        if (connection != null) {
            return;
        }
```

```
        bytesRemainingForRequest = requestPayloadSize;
        input.mark(requestPayloadSize);

        if (proxy != null) {
            connection = (HttpURLConnection) uploadURL.openConnection(proxy);
        } else {
            connection = (HttpURLConnection) uploadURL.openConnection();
        }
        client.prepareConnection(connection);
        connection.setRequestProperty("Upload-Offset",
Long.toString(offset));
        connection.setRequestProperty("Content-Type",
"application/offset+octet-stream");
        connection.setRequestProperty("Expect", "100-continue");

        try {
            connection.setRequestMethod("PATCH");
            // Check whether we are running on a buggy JRE
        } catch (java.net.ProtocolException pe) {
            connection.setRequestMethod("POST");
            connection.setRequestProperty("X-HTTP-Method-Override", "PATCH");
        }

        connection.setDoOutput(true);
        connection.setChunkedStreamingMode(0);
        try {
            output = connection.getOutputStream();
        } catch (java.net.ProtocolException pe) {
            // If we already have a response code available, our expectation
using the "Expect: 100-
            // continue" header failed and we should handle this response.
            if (connection.getResponseCode() != -1) {
                finish();
            }

            throw pe;
        }
    }

    /**
     * Sets the used chunk size. This number is used by {@link
#uploadChunk()} to indicate how
     * much data is uploaded in a single take. When choosing a value for this
parameter you need to
     * consider that uploadChunk() will only return once the specified number
of bytes has been
     * sent. For slow internet connections this may take a long time. In
addition, a buffer with
     * the chunk size is allocated and kept in memory.
```

```
     *
     * @param size The new chunk size
     */
    public void setChunkSize(int size) {
        buffer = new byte[size];
    }

    /**
     * Returns the current chunk size set using {@link #setChunkSize(int)}.
     *
     * @return Current chunk size
     */
    public int getChunkSize() {
        return buffer.length;
    }

    /**
     * Set the maximum payload size for a single request counted in bytes.
This is useful for splitting
     * bigger uploads into multiple requests. For example, if you have a
resource of 2MB and
     * the payload size set to 1MB, the upload will be transferred by two
requests of 1MB each.
     *
     * The default value for this setting is 10 * 1024 * 1024 bytes (10 MiB).
     *
     * Be aware that setting a low maximum payload size (in the low megabytes
or even less range) will result in
     * decreased performance since more requests need to be used for an
upload. Each request will come with its overhead
     * in terms of longer upload times.
     *
     * Be aware that setting a high maximum payload size may result in a high
memory usage since
     * tus-java-client usually allocates a buffer with the maximum payload
size (this buffer is used
     * to allow retransmission of lost data if necessary). If the client is
running on a memory-
     * constrained device (e.g. mobile app) and the maximum payload size is
too high, it might
     * result in an {@link OutOfMemoryError}.
     *
     * This method must not be called when the uploader has currently an open
connection to the
     * remote server. In general, try to set the payload size before invoking
{@link #uploadChunk()}
     * the first time.
     *
     * @see #getRequestPayloadSize()
```

```
     *
     * @param size Number of bytes for a single payload
     * @throws IllegalStateException Thrown if the uploader currently has a
connection open
     */
    public void setRequestPayloadSize(int size) throws IllegalStateException
{
        if (connection != null) {
            throw new IllegalStateException("payload size for a single
request must not be "
                    + "modified as long as a request is in progress");
        }

        requestPayloadSize = size;
    }

    /**
     * Get the current maximum payload size for a single request.
     *
     * @see #setChunkSize(int)
     *
     * @return Number of bytes for a single payload
     */
    public int getRequestPayloadSize() {
        return requestPayloadSize;
    }

    /**
     * Upload a part of the file by reading a chunk from the InputStream and
writing
     * it to the HTTP request's body. If the number of available bytes is
lower than the chunk's
     * size, all available bytes will be uploaded and nothing more.
     * No new connection will be established when calling this method,
instead the connection opened
     * in the previous calls will be used.
     * The size of the read chunk can be obtained using {@link
#getChunkSize()} and changed
     * using {@link #setChunkSize(int)}.
     * In order to obtain the new offset, use {@link #getOffset()} after this
method returns.
     *
     * @return Number of bytes read and written.
     * @throws IOException  Thrown if an exception occurs while reading from
the source or writing
     *                      to the HTTP request.
     */
    public int uploadChunk() throws IOException, ProtocolException {
        openConnection();
```

```
        int bytesToRead = Math.min(getChunkSize(), bytesRemainingForRequest);

        int bytesRead = input.read(buffer, bytesToRead);
        if (bytesRead == -1) {
            // No bytes were read since the input stream is empty
            return -1;
        }

        // Do not write the entire buffer to the stream since the array will
        // be filled up with 0x00s if the number of read bytes is lower then
        // the chunk's size.
        output.write(buffer, 0, bytesRead);
        output.flush();

        offset += bytesRead;
        bytesRemainingForRequest -= bytesRead;

        if (bytesRemainingForRequest <= 0) {
            finishConnection();
        }

        return bytesRead;
    }

    /**
     * Upload a part of the file by read a chunks specified size from the
InputStream and writing
     * it to the HTTP request's body. If the number of available bytes is
lower than the chunk's
     * size, all available bytes will be uploaded and nothing more.
     * No new connection will be established when calling this method,
instead the connection opened
     * in the previous calls will be used.
     * In order to obtain the new offset, use {@link #getOffset()} after this
method returns.
     *
     * This method ignored the payload size per request, which may be set
using
     * {@link #setRequestPayloadSize(int)}. Please, use {@link
#uploadChunk()} instead.
     *
     * @deprecated This method is inefficient and has been replaced by {@link
#setChunkSize(int)}
     *             and {@link #uploadChunk()} and should not be used anymore.
The reason is, that
     *             this method allocates a new buffer with the supplied chunk
size for each time
```

```
     *             it's called without reusing it. This results in a high
number of memory
     *             allocations and should be avoided. The new methods do not
have this issue.
     *
     * @param chunkSize Maximum number of bytes which will be uploaded. When
choosing a value
     *                  for this parameter you need to consider that the
method call will only
     *                  return once the specified number of bytes have been
sent. For slow
     *                  internet connections this may take a long time.
     * @return Number of bytes read and written.
     * @throws IOException  Thrown if an exception occurs while reading from
the source or writing
     *                      to the HTTP request.
     */
    @Deprecated public int uploadChunk(int chunkSize) throws IOException,
ProtocolException {
        openConnection();

        byte[] buf = new byte[chunkSize];
        int bytesRead = input.read(buf, chunkSize);
        if (bytesRead == -1) {
            // No bytes were read since the input stream is empty
            return -1;
        }

        // Do not write the entire buffer to the stream since the array will
        // be filled up with 0x00s if the number of read bytes is lower then
        // the chunk's size.
        output.write(buf, 0, bytesRead);
        output.flush();

        offset += bytesRead;

        return bytesRead;
    }

    /**
     * Get the current offset for the upload. This is the number of all bytes
uploaded in total and
     * in all requests (not only this one). You can use it in conjunction
with
     * {@link TusUpload#getSize()} to calculate the progress.
     *
     * @return The upload's current offset.
     */
    public long getOffset() {
```

```
        return offset;
    }

    /**
     * This methods returns the destination {@link URL} of the upload.
     * @return The {@link URL} of the upload.
     */
    public URL getUploadURL() {
        return uploadURL;
    }

    /**
     * Set the proxy that will be used when uploading.
     *
     * @param proxy Proxy to use
     */
    public void setProxy(Proxy proxy) {
        this.proxy = proxy;
    }

    /**
     * This methods returns the proxy used when uploading.
     *
     * @return The {@link Proxy} used for the upload or null when not set.
     */
    public Proxy getProxy() {
        return proxy;
    }

    /**
     * Finish the request by closing the HTTP connection and the InputStream.
     * You can call this method even before the entire file has been
uploaded. Use this behavior to
     * enable pausing uploads.
     * This method is equivalent to calling {@code finish(false)}.
     *
     * @throws ProtocolException Thrown if the server sends an unexpected
status
     * code
     * @throws IOException  Thrown if an exception occurs while cleaning up.
     */
    public Optional<TusResponseBody> finish() throws ProtocolException,
IOException {
        return finish(true);
    }

    /**
     * Finish the request by closing the HTTP connection. You can choose
whether to close the InputStream or not.
```

```
     * You can call this method even before the entire file has been
uploaded. Use this behavior to
     * enable pausing uploads.
     * Be aware that it doesn't automatically release local resources if
{@code closeStream == false} and you do
     * not close the InputStream on your own. To be safe use {@link
TusUploader#finish()}.
     * @param closeInputStream Determines whether the InputStream is closed
with the HTTP connection. Not closing the
     *                         Input Stream may be useful for future upload a
future continuation of the upload.
     * @throws ProtocolException Thrown if the server sends an unexpected
status code
     * @throws IOException  Thrown if an exception occurs while cleaning up.
     */
    public Optional<TusResponseBody> finish(boolean closeInputStream) throws
ProtocolException, IOException {
      Optional<TusResponseBody> response = finishConnection();
        if (upload.getSize() == offset) {
            client.uploadFinished(upload);
        }

        // Close the TusInputStream after checking the response and closing
the connection to ensure
        // that we will not need to read from it again in the future.
        if (closeInputStream) {
            input.close();
        }
        return response;
    }

    private Optional<TusResponseBody> finishConnection() throws
ProtocolException, IOException {
        if (output != null) {
            output.close();
        }

        if (connection == null) {
            return Optional.empty();
        }

        int responseCode = connection.getResponseCode();
        TusResponseBody response = new TusResponseBody(connection);
        connection.disconnect();

        if (responseCode == 401 || responseCode == 403) {
            throw new Http401And403CodeException(response, connection);
        }
```

```
        if(responseCode == 422) {
             throw new Http422CodeException(response, connection);
        }

        if (!(responseCode >= 200 && responseCode < 300)) {
            throw new ProtocolException("unexpected status code (" +
responseCode + ") while uploading chunk",
                    connection);
        }

        // TODO detect changes and seek accordingly
        long serverOffset = getHeaderFieldLong(connection, "Upload-Offset");
        if (serverOffset == -1) {
            throw new ProtocolException("response to PATCH request contains
no or invalid Upload-Offset header",
                    connection);
        }
        if (offset != serverOffset) {
            throw new ProtocolException(
                   String.format("response contains different Upload-Offset
value (%d) than expected (%d)",
                       serverOffset,
                        offset),
                   connection);
        }

        connection = null;

        return Optional.of(response);
    }

    private long getHeaderFieldLong(URLConnection connection, String field) {
        String value = connection.getHeaderField(field);
        if (value == null) {
            return -1;
        }

        try {
            return Long.parseLong(value);
        } catch (NumberFormatException e) {
            return -1;
        }
    }
}
```

Clase:: Demo.java

```
package pe.gob.sunat.tecnologia.client.tus.main;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import io.tus.java.client.HttpErrorCodeException;
import io.tus.java.client.ProtocolException;
import io.tus.java.client.TusClientCustom;
import io.tus.java.client.TusExecutor;
import io.tus.java.client.TusResponseBody;
import io.tus.java.client.TusURLMemoryStore;
import io.tus.java.client.TusUpload;
import io.tus.java.client.TusUploaderCustom;


public class Demo {

  private static final String TOKEN = "";

  private static final String HOST_PUBLICA = "https://api-
sire.sunat.gob.pe/v1/contribuyente/migeigv/";
  private static final String END_POINT_PROPUESTA =
"libros/rvierce/receptorpropuesta/web/propuesta/upload";

  private static final Logger log = LoggerFactory.getLogger(Demo.class);


  public static void main(String[] args) {
    try {
      // Create a new TusClient instance
      final TusClientCustom client = new TusClientCustom();


      client.setUploadCreationURL(new URL(HOST_PUBLICA +
END_POINT_PROPUESTA));
      client.enableResuming(new TusURLMemoryStore());
      Map<String, String> headers = new HashMap<>();
      headers.put("authorization", "Bearer " + TOKEN);
      client.setHeaders(headers);
      File archivo = new File(
```

```
        "D:\\migeigv\\archivos\\ajuste_posterior\\LE20100017491202311001404000
03111201.zip"
      );
      final TusUpload upload = new TusUpload(archivo);
      String[] extension = archivo.getName().split("\\.");
      Map<String, String> metaData = new HashMap<>();
      metaData.put("filename", archivo.getName());
      metaData.put("filetype", extension[1]);
      metaData.put("numRuc", "20108745216");
      metaData.put("perTributario", "202304");
      metaData.put("codOrigenEnvio", "3");
      metaData.put("codLibro", "140000");
      metaData.put("codProceso", "1");
      metaData.put("codTipoCorrelativo", "1");
      metaData.put("nomArchivoImportacion", archivo.getName());
      upload.setMetadata(metaData);
      log.info("Starting upload...");
      TusExecutor executor = new TusExecutor() {
        @Override
        protected void makeAttempt() throws ProtocolException, IOException {
          TusUploaderCustom uploader = client.resumeOrCreateUpload(upload);
          uploader.setChunkSize(1024);
          do {
            long totalBytes = upload.getSize();
            long bytesUploaded = uploader.getOffset();
            double progress = (double) bytesUploaded / totalBytes * 100;
            String porcentaje = String.format("%06.2f%%", progress);
            log.info("Upload at {}.", porcentaje);
          } while (uploader.uploadChunk() > -1);


          Optional<TusResponseBody> response = uploader.finish();

          response.ifPresent(resp -> {
            log.info(
              "************** ==============> code: {} message: {}
<============================== ****************",
              resp.getResponseCode(),
              resp.getResponseMessage()
            );
            log.info(
              "************** ==============> Result {} {}
<============================== ****************",
              "20108745216",
              resp.getResponseBody()
            );
          });
        }
      };
```

```
      executor.makeAttempts();
    } catch (HttpErrorCodeException e) {
        log.info(
              "************** ==============> code: {} message: {}
<============================== ****************",
              e.getResponseBody().getResponseCode(),
              e.getResponseBody().getResponseMessage()
            );
      log.error("Error en la peticion: {}", e.getResponseBody(), e);
    } catch (ProtocolException e) {
      log.error("/////// //// ==> Error inesperado en la petición: ", e);
    } catch (IOException e) {
      e.printStackTrace();
    } catch (Exception ex) {

        log.error("ex.getCause(): {}", ex.getMessage(), ex);

    }
  }
}
```