

MANUAL DE INSTALACION /CONFIGURACION

Bulk V2

Claro - Campañas efectivas

**Control de Versiones**

| **Fecha** | **Versión** | **Tipo de Cambio** | **Descripción** | **Autor** |
| --- | --- | --- | --- | --- |
| 31-08-2021 | 0.4 | Creación del documento | Primera Versión | SQDM |

1. **Índice**

[**RECURSOS DE HARDWARE**](#_heading=h.1fob9te) **3**

[**Servidores**](#_heading=h.2et92p0) **4**

[**Estaciones cliente**](#_heading=h.tyjcwt) **5**

[**Conectividad**](#_heading=h.3dy6vkm) **5**

[**RECURSOS DE SOFTWARE**](#_heading=h.1t3h5sf) **5**

[**ARQUITECTURA PROYECTO**](#_heading=h.nmj9w8p1xei0) **6**

[**Carpetas**](#_heading=h.395old8hayvs) **7**

[**paquetes y clases**](#_heading=h.u0dum2avwiel) **9**

[**graells.bulkapiv2**](#_heading=h.579su2nu91lx) **9**

[**Bulkapiv2Application.java**](#_heading=h.eiht25oslkvs) **9**

[**Conexion.java**](#_heading=h.rqs2xudyb3d0) **15**

[**utils**](#_heading=h.bv6ah1d07n12) **16**

[**ManageFile.java**](#_heading=h.u4v0g5og8mvm) **16**

[**INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA**](#_heading=h.a0y5xoiqbb4n) **18**

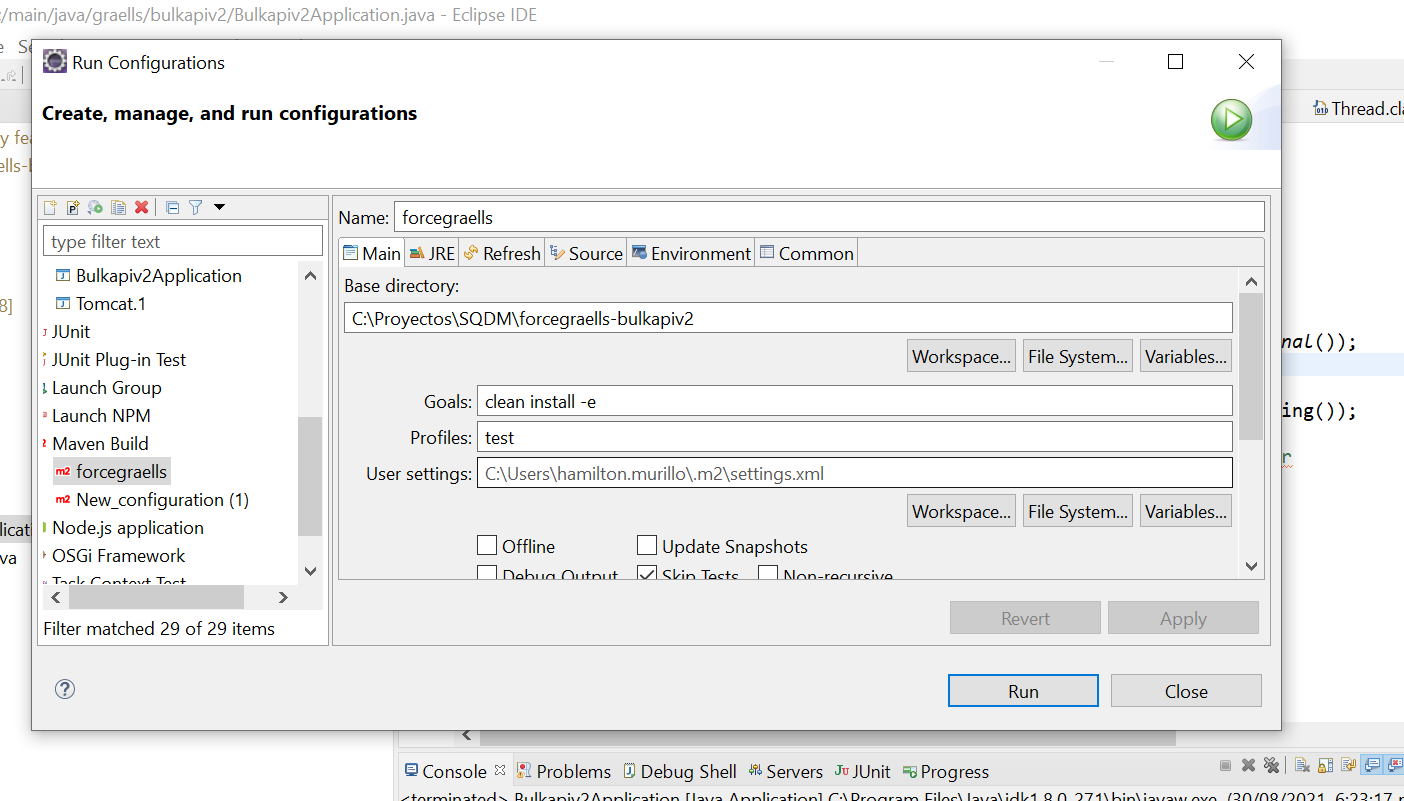
[**ERRORES CONOCIDOS**](#_heading=h.frxvvrpm18l2) **18**

# **RECURSOS DE HARDWARE**

## Servidores

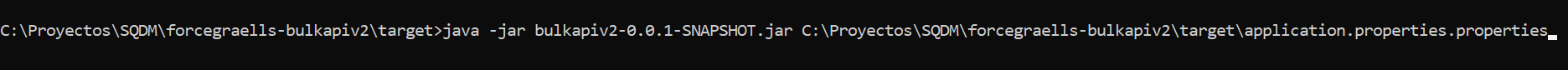
El proyecto se compila en una versión de maven 4.0 y java 8, el tipo de archivo generado después de la compilación es un jar

El proyecto se puede compilar desde eclipse, utilizando la siguiente configuración, el directorio base se acomoda a la ubicación donde está ubicado el proyecto, Es importante marcar la casilla para saltar los test



El .jar quedará en el directorio target del proyecto.

Para la ejecución del proyecto se debe usar la siguiente línea



La ruta del archivo .properties varía según su ubicación.

Para la ejecución del .jar se deben tener las siguientes recomendaciones en el archivo properties:

* Verificar que la ruta de la carpeta de entrada como salida correspondan a ubicaciones reales
* Validar que los parámetros para la creación del job concuerde con la plantilla que se va a subir

## Estaciones cliente

No aplica

## Conectividad

No aplica

# **RECURSOS DE SOFTWARE**

Recursos requeridos

Para realizar el desarrollo se utilizó el IDE Eclipse(Versión: 2020-12 (4.18.0)) con los siguientes complementos.

| Nombre | Versión |
| --- | --- |
| Maven | 4 |
| Java | 1.8 |

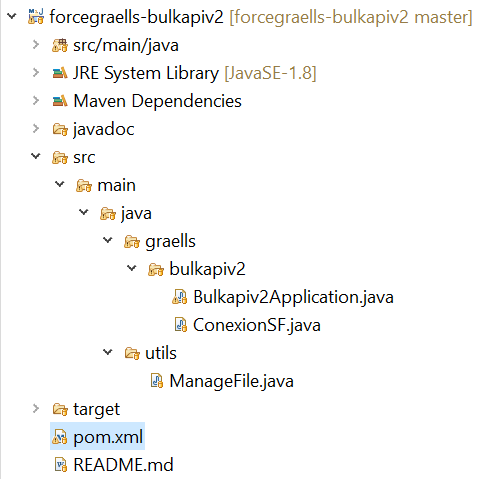
Dependencias

| Nombre | Versión |
| --- | --- |
| net.minidev | 2.3 |
| org.apache.commons | 1.3.2 |
| org.glassfish.jersey.core | 2.26 |
| org.json | 20160810 |
| org.apache.httpcomponents | 4.5.2 |
| org.springframework.boot |  |

# ARQUITECTURA PROYECTO

## Carpetas

En el ide Eclipse versión (2020-12 (4.18.0)) en la parte de la izquierda desplegamos las carpetas del proyecto y observamos su estructura en forma de árbol de directorios

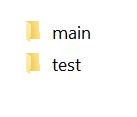


El proyecto contiene la siguiente estructura de carpeta

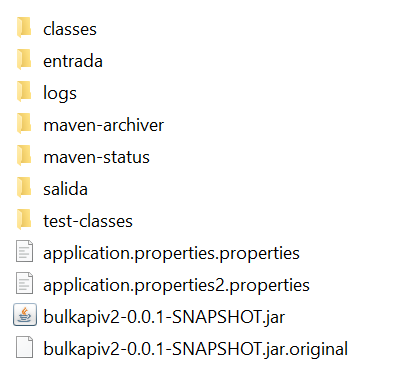
**forcegraells-bulkapiv2**: contiene las carpetas src y target



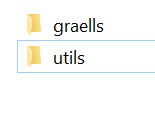
**forcegraells-bulkapiv2/src**: contiene las carpeta main



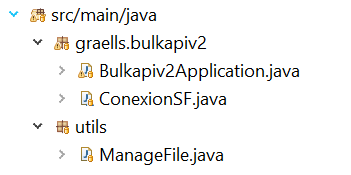
**forcegraells-bulkapiv2/target:** En esta carpeta se encuentran los .class, se guarda el jar y se recomienda ubicar el archivo application.properties.properties



**forcegraells-bulkapiv2/src/main/java:** contiene las carpetas graells y utils, en estas se encuentra toda la lógica de la aplicación

****

## paquetes y clases



### graells.bulkapiv2

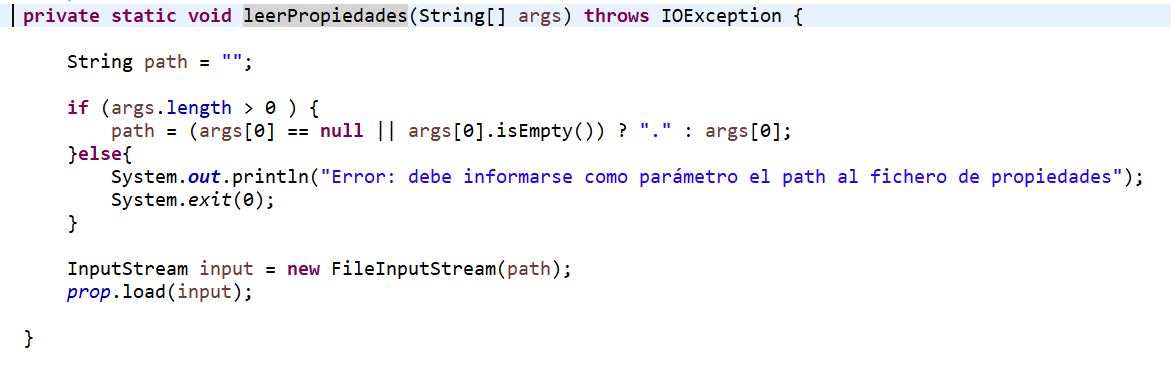
Este paquete contiene las clases Bulkapiv2Application.java y Conexion.java.

#### **Bulkapiv2Application.java**

Esta es la clase principal donde se encuentran los métodos de consumo del salesforce APis, contiene los siguientes métodos:

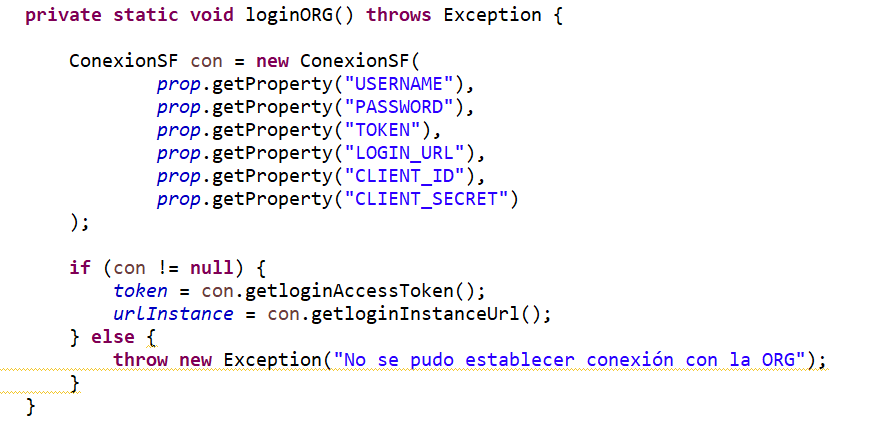
**leerPropiedades()**

Recibe como parámetrola ruta donde se encuentra el archivo application.properties.properties extrae las propiedades, el cual cual contiene los datos de conexión, parámetros para la creación del job y las rutas de los ficheros de entrada,salida y logs



**loginORG()**

Realiza una conexión con la ORG indicada en los miembros de la clase, establece los valores de token y urlInstance si se consigue un login correcto, los parámetros se encuentran en en el archivo application.properties.properties



**crearJobSimple()**

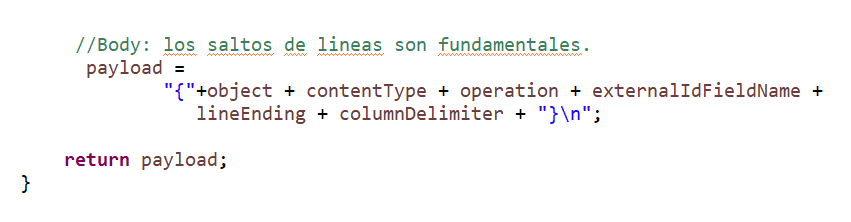
hace una petición post a la URL /services/data/vXX.X/jobs/ingest para crear un nuevo job, con las cabeceras que indica el documento, se debe ser muy cuidadoso con los saltos de línea en el payload, retorna el id del job creado el cual se utiliza para realizar el resto de operaciones.



**getPayload()**

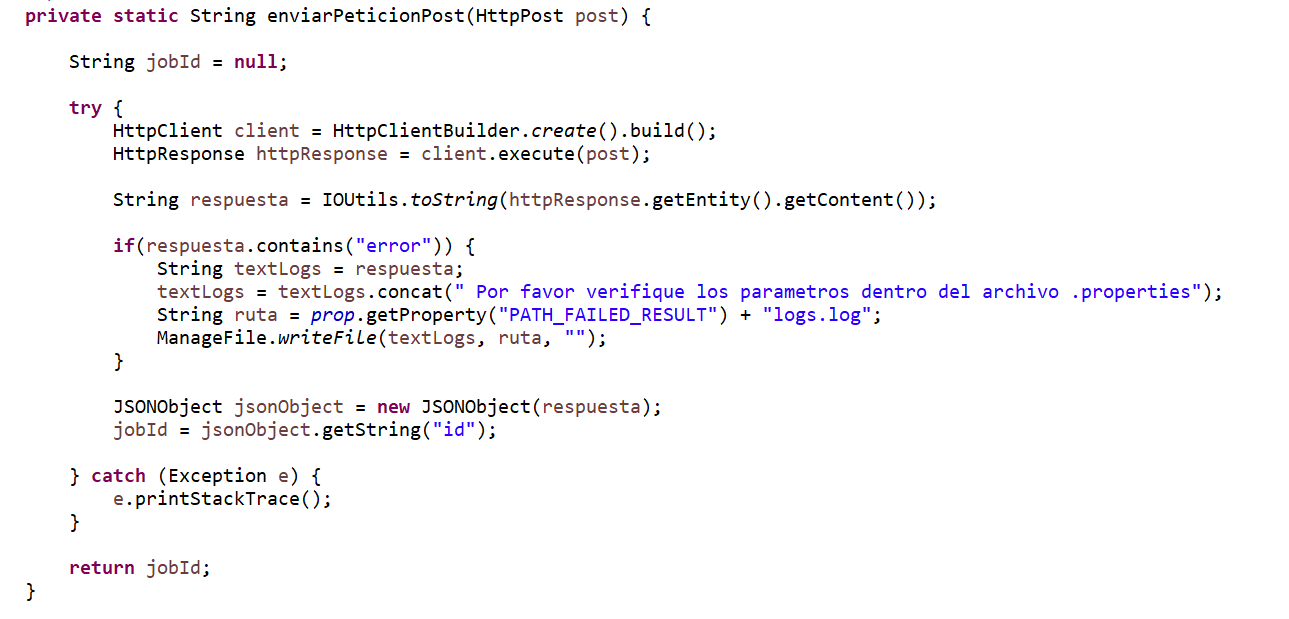
Verifica si los datos obtenidos del archivo properties estan vacios para construir el payload, si estan vacios no se tiene en cuenta para construir el payload, retorna el payload solo con los datos validos





**enviarPeticionPost()**

Envía una petición Post preconfigurada y retorna el id del job



**enviarDatos()**

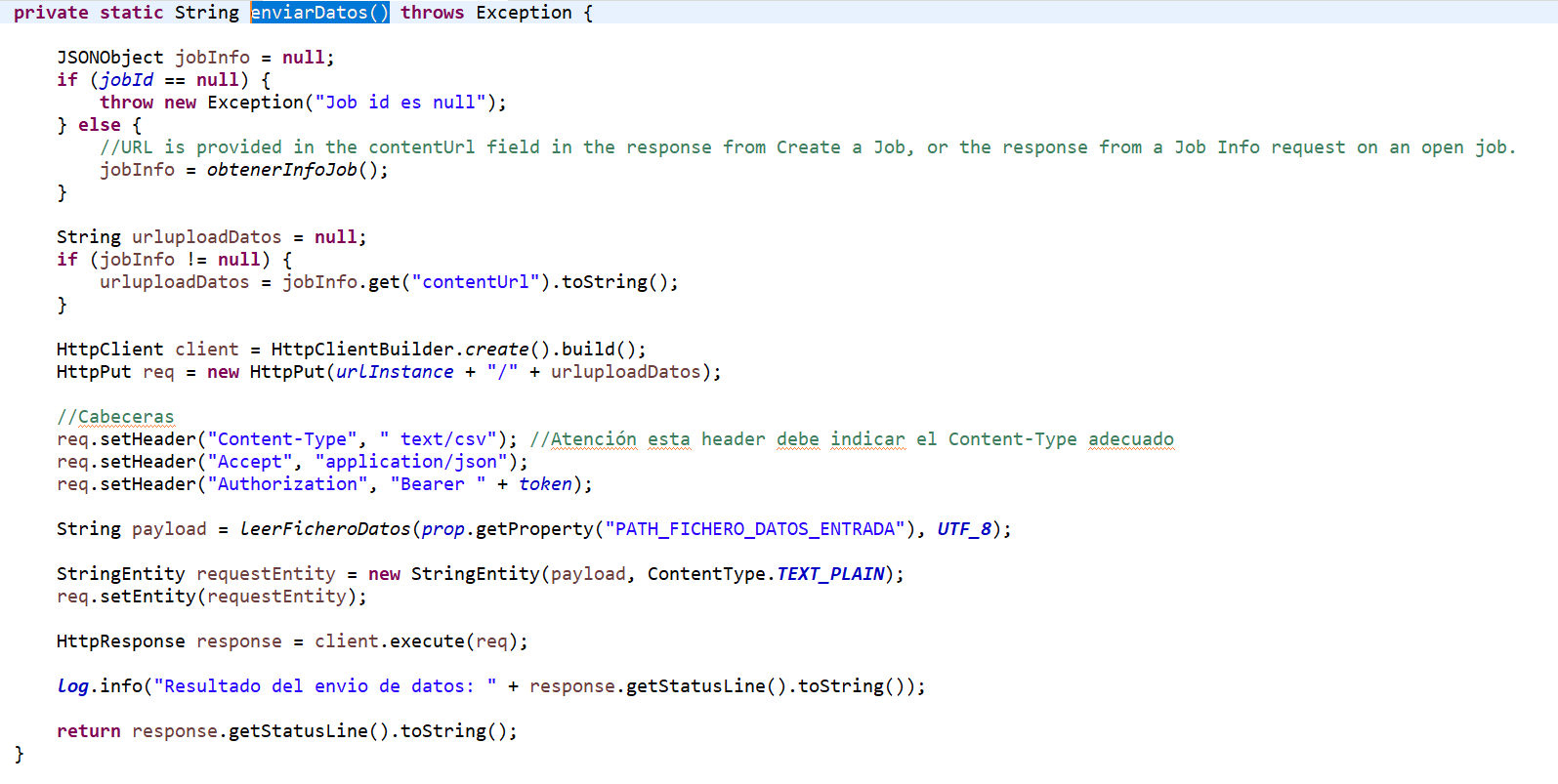
Se encarga de hacer la carga de los datos del CSV

\* Se realiza a posteriori de su creación a la creación del job.

\* Por tanto, esta función realiz un PUT sobre la url que se obtiene del Job, para cargar en el payload

\* el contenido del CSV y enviarlo.

\* Recalcar la importancia del contenido de las cabeceras siguiendo el documento oficial



**abortarCerrarJob()**

Recibe como parámetro la operación que se va a realizar

Llamada PATCH sobre la url /services/data/vXX.X/jobs/ingest/jobID para indicar a Salesforce que ejecute un cambio de estado sobre el job, para cerrarlo y que se proceda a su ejecución o para cancelarlo



**poolingJobInfoHastaEstadoFinal()**

Realiza pooling de la info de un Job, cada n segundos, hasta que el Job finaliza en estado "JobComplete" ó "Failed"



**obtenerInfoJob()**

retorna un JSONObject

Get a la url /services/data/vXX.X/jobs/ingest/jobID para obtener la info de un Job



**failedResults()**

Llamada PATCH sobre la url /services/data/vXX.X/jobs/ingest/jobID/failedResults para consultar los resultados fallidos para luego guardar el resultado en un archivo



#### **Conexion.java**

En esta clase se hace la petición para obtener el token de acceso



### utils

Este paquete contiene la clase ManageFile.java

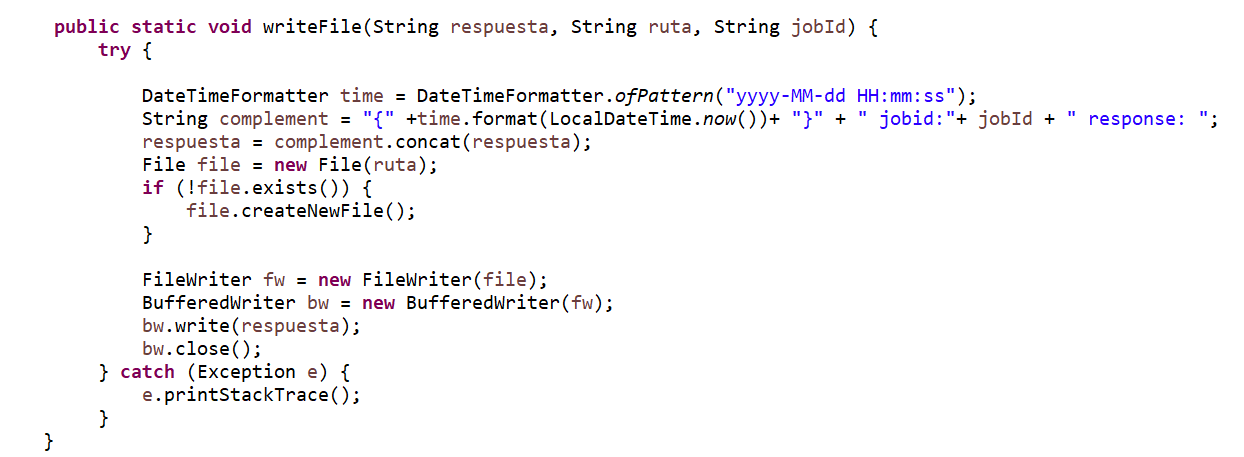
### ManageFile.java

Esta clase contiene los siguientes métodos.

**writeFile()**

recibe como parámetro la respuesta de la petición a los elementos fallidos, la ruta donde va a quedar guardado el archivo(esta se encuentra en el archivo .properties y el id del job para nombrar el archivo)

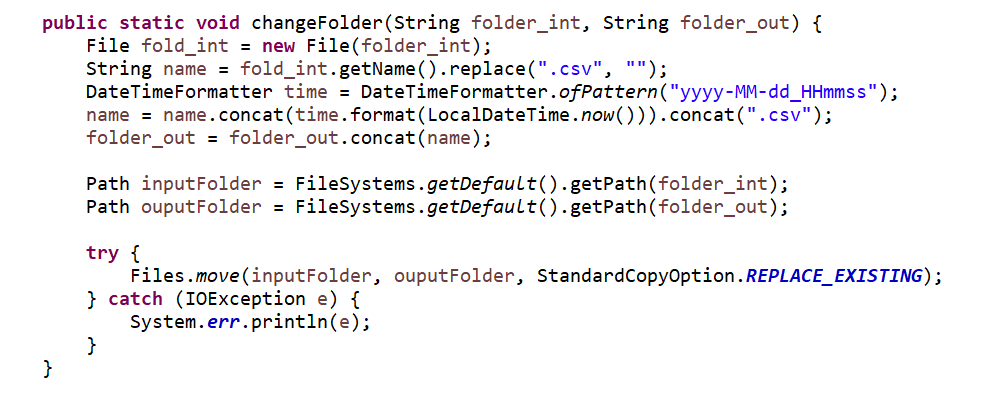
Crea archivo con los resultados fallidos



**changeFolder()**

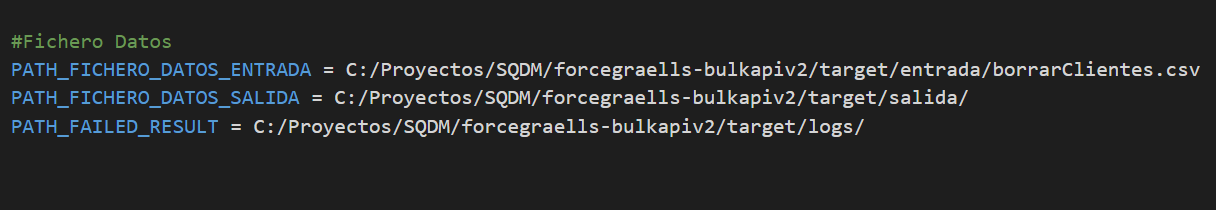
Recibe como parámetros la carpeta de entrada y la carpeta de salida ambas rutas se encuentran en el archivo .properties

mueve el archivo de la carpeta de entrada a la carpeta de salida cuando se ha procesado



# INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA

Se deben tener en cuenta las siguientes recomendaciones en el archivo application.properties.properties



Los valores registrados en las propiedades son usados como ejemplo para este manual, se deben reemplazar por las rutas donde se encuentren los archivos y donde se desean guardar los resultados.

# ERRORES CONOCIDOS

**No encuentra el Archivo**

Posibles soluciones

* verificar que la la ruta de la carpeta donde se guardó el archivo si está bien referenciada.
* verificar que el nombre del archivo si es correcto



**La ruta del archivo de entrada es correcta pero no lo reconoce**

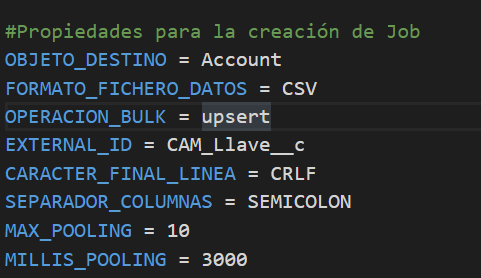
posible solución

* verificar que la ruta del archivo no contenga caracteres como ñ o tilde

**Fallo en la creación del JOB**

posible solución

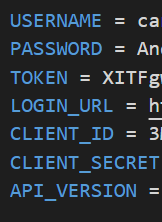
* Verificar que los parámetros en el archivo .properties si sean los correctos según el tipo de operación que se desea hacer.



**Fallo en la obtención del token de acceso**

posible solución

* Verificar que los parámetros en el archivo application.properties.properties sean los correctos.



**No se encuentran las rutas donde se deben guardar los archivos**

posibles soluciones

* Se debe verificar las rutas en el archivo application.properties.properties